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

Heitor Pereira Nunes Fernandes Cunha

Processamento de Sinais Biomédicos: Módulo 2

Uberlândia, MG
2025

Questão 1

Gere sinais sintéticos que simulem a forma de onda do ciclo respiratório, de acordo a faixa etária:

do nascimento a 6 semanas: 30 a 40 respirações por minuto; 6 meses: 25 a 40 respirações por minuto; 3 anos: 20 a 30 respirações por minuto; 6 anos: 18 a 25 respirações por minuto; 10 anos: 17 a 23 respirações por minuto; Adultos: 12 a 18 respirações por minuto; Idosos (>= 65 anos): 12 a 28 respirações por minuto; Idosos (>= 80 anos): 10 a 30 respirações por minuto.

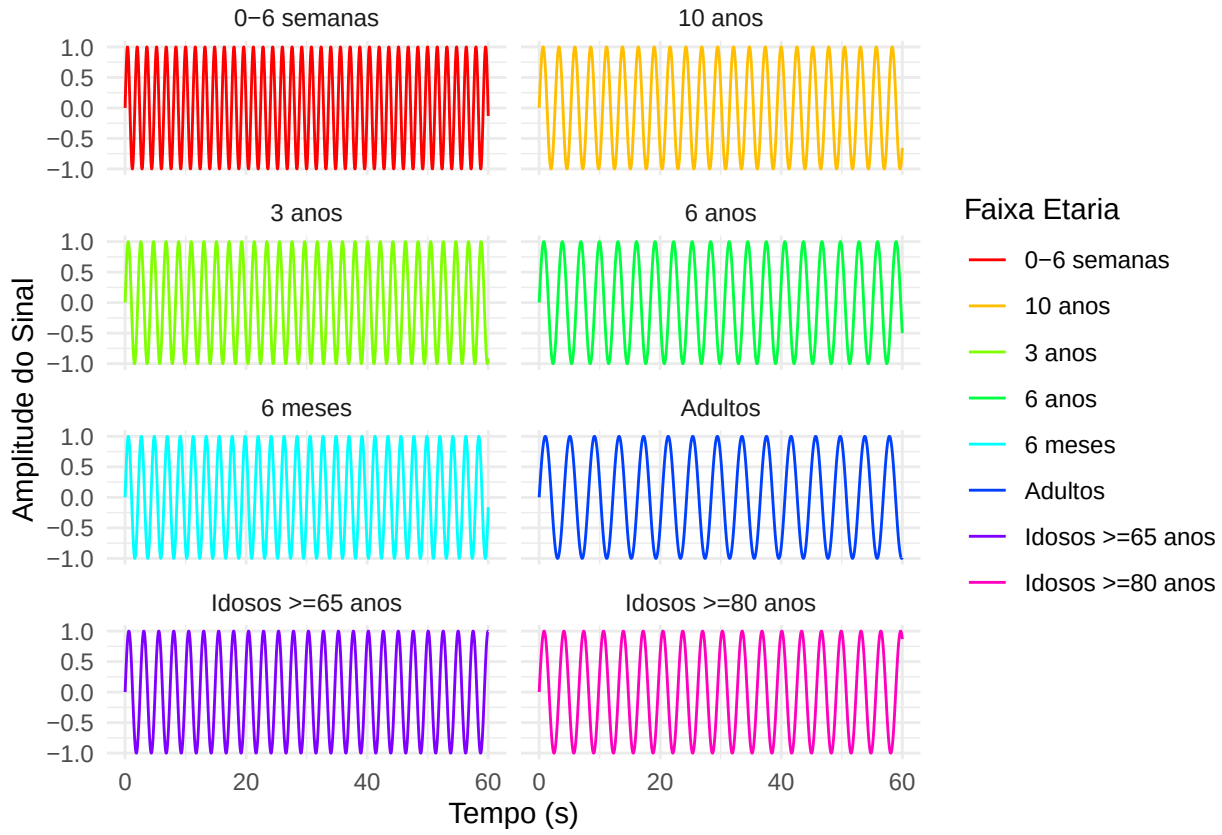
Para gerar sinais sintéticos que simulem a forma de onda do ciclo respiratório de acordo com a faixa etária fornecida, podemos usar uma abordagem baseada em funções de onda senoidal ou cossenoidal, representando a inspiração e a expiração. A frequência respiratória de cada faixa etária será usada para determinar a taxa de oscilação das ondas.

```
# Definição dos parâmetros
library(ggplot2)
fs <- 100  # Frequência de amostragem em Hz
t <- seq(0, 60, by = 1/fs)  # Sinal de 1 minuto

# Faixas de frequência respiratória por idade (em ciclos por minuto)
faixas_respiratorias <- list(
  "0-6 semanas" = c(30, 40),
  "6 meses" = c(25, 40),
  "3 anos" = c(20, 30),
  "6 anos" = c(18, 25),
  "10 anos" = c(17, 23),
  "Adultos" = c(12, 18),
  "Idosos >=65 anos" = c(12, 28),
  "Idosos >=80 anos" = c(10, 30)
)

# Inicializando o dataframe
 df_sinais <- data.frame(Tempo = numeric(), Sinal = numeric(), Idade = character())

# Gerando os sinais sintéticos
for (idade in names(faixas_respiratorias)) {
  freq_min <- faixas_respiratorias[[idade]][1]
  freq_max <- faixas_respiratorias[[idade]][2]
  freq_respiratoria <- runif(1, freq_min, freq_max) / 60  # Convertendo para Hz
  sinal <- sin(2 * pi * freq_respiratoria * t)  # Onda senoidal simulando respiração
  df_sinais <- rbind(df_sinais, data.frame(Tempo = t, Sinal = sinal, Idade = rep(idade, length(t))))
}
```

Utilize a função ggplot para plotar cada um dos sinais gerados no item a. A imagem resultante deverá conter um painel dividido em duas colunas e quatro linhas. Dica: utilize a função facet_wrap da biblioteca ggplot2. Adicione legendas para os eixos x e y de cada gráfico.

```
library(ggplot2)
library(htmltools)

# Plotando os sinais
graficos <- ggplot(df_sinais, aes(x = Tempo, y = Sinal, color = Idade)) +
```

```
  geom_line() +
  facet_wrap(~ Idade, ncol = 2) +
  scale_color_manual(values = rainbow(length(faixas_respiratorias))) +
  labs(x = "Tempo (s)", y = "Amplitude do Sinal", color = "Faixa Etaria") +
  theme_minimal()

print(graficos)
```



Utilize a bliblioteca dygraph para gerar um gráfico para cada um dos sinais simulados no item a. Adicione legendas para os eixos x e y de cada gráfico. O resultado final deverá ser apresentado em um único painel html contendo os gráficos gerados. Dica: veja o código exemplo que faz algo semelhando ao que está sendo solicitado na questão.

```
library(dygraphs)
library(RColorBrewer)
library(htmltools)
library(magrittr)

# Criando uma lista para armazenar os gráficos
graficos <- list()

# Gerando os sinais sintéticos e os gráficos dygraph
for (idade in names(faixas_respiratorias)) {
  freq_min <- faixas_respiratorias[[idade]][1]
  freq_max <- faixas_respiratorias[[idade]][2]
```

```
  freq_respiratoria <- runif(1, freq_min, freq_max) / 60
  sinal <- sin(2 * pi * freq_respiratoria * t)
  df_sinal <- data.frame(Tempo = t, Sinal = sinal)

  graficos[[idade]] <- tagList(
    tags$h3(idade),
    dygraph(df_sinal) %>%
      dyAxis("x", label = "Tempo (s)") %>%
      dyAxis("y", label = "Amplitude do Sinal") %>%
      dyOptions(colors = RColorBrewer::brewer.pal(8, "Dark2")) %>%
      dyLegend("always")
  )
}

# Criando painel HTML
html_page <- tagList(
  tags$html(
    tags$head(tags$title("Sinais Respiratórios")),
    tags$body(
      lapply(graficos, function(g) tagList(g, tags$br()))
    )
  )
)

# Salvando o painel HTML
save_html(html_page, "sinais_respiratorios.html")
```

Escolha um sinal simulado no item a, e plote-o utilizando a função dyStemSeries. Por que há um intervalo temporal entre cada amostra? Dica: sua resposta deve ser completa e deve abordar conceitos como amostragem e digitalização de sinais.

```
library(xts)
```

```
## Carregando pacotes exigidos: zoo
```

```
##
## Anexando pacote: 'zoo'
```

```
## Os seguintes objetos são mascarados por 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(magrittr)
library(dygraphs)

# Criando um dataframe apenas para 3 anos
df_3_anos <- data.frame(Tempo = t, Sinal = sinal, Idade = "3 anos")

# Criando um objeto xts para uso com dygraphs
data_xts <- xts(df_3_anos$Sinal, order.by = as.POSIXct(df_3_anos$Tempo, origin = "1970-01-01"))
```

```r
# Plotando o sinal com dygraph utilizando dyStemSeries
dygraph(data_xts, main = "Respiração - Criança de 3 anos") %>%
  dyStemSeries("V1", label = "Respiração") %>%
  dyAxis("x", label = "Tempo (s)") %>%
  dyAxis("y", label = "Amplitude do Sinal") %>%
  dyOptions(drawGrid = TRUE)
```

O intervalo temporal entre as amostras ocorre porque o dygraph trabalha com objetos xts, que exigem que o eixo do tempo seja definido com valores do tipo POSIXct (data e hora). No seu código, os tempos são convertidos usando as.POSIXct(df_3_anos$Tempo, origin = "1970-01-01"), o que significa que o tempo está sendo tratado como uma série temporal. Portanto, mesmo que os valores originais de Tempo sejam espaçados uniformemente (0, 0.01, 0.02, ..., 60 segundos), o dygraph pode exibir a série de forma que os pontos pareçam estar distribuídos com espaços irregulares, dependendo da escala automática aplicada pelo gráfico.

Questão 2

O armazenamento de dados2 biomédicos pode acontecer em diverso formatos, incluindo o formato de programas comerciais como o Matlab. Neste caso, é relevante conseguir abrir arquivos gerados em outros formatos no ambiente do R. Nesta questão você deverá abrir e plotar dados2 biomédicos de uma arquivo no formato .mat (Matlab). Os seguintes passos devem ser executados:

```
library(R.matlab)
```

```
## R.matlab v3.7.0 (2022-08-25 21:52:34 UTC) successfully loaded. See ?R.matlab for help.
```

```
##
## Anexando pacote: 'R.matlab'
```

```
## Os seguintes objetos são mascarados por 'package:base':
##
##     getOption, isOpen
```

```
dados2 <- readMat("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 2/A07.mat")
```

Gerar um gráfico de cada um das três variáveis disponíveis no arquivo em função de tempo. A imagem final deverá conter um painel de uma coluna e três linhas. Dica: (i) utilize a função facet_wrap. (ii) a resolução temporal, ou seja, o intervalo entre amostras está disponível na variável isi cuja a unidade é "ms". (iii) deve-se gerar legendas para os eixos dos gráficos. (iv) filtre os dados2 em um intervalo de tempo para que seja possível visualizar as informações. Você pode considerar o tempo entre 0 e 10 segundos, por exemplo.

```
# Carregar os pacotes
library(ggplot2)
library(dplyr)
```

```
##
## ########################## Warning from 'xts' package ##########################
## #                                                                              #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                             #
## #                                                                              #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                 #
## #                                                                              #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                              #
## ################################################################################
```

```
##
## Anexando pacote: 'dplyr'
```

```
## Os seguintes objetos são mascarados por 'package:xts':
##
##     first, last

## Os seguintes objetos são mascarados por 'package:stats':
##
##     filter, lag

## Os seguintes objetos são mascarados por 'package:base':
##
##     intersect, setdiff, setequal, union
```

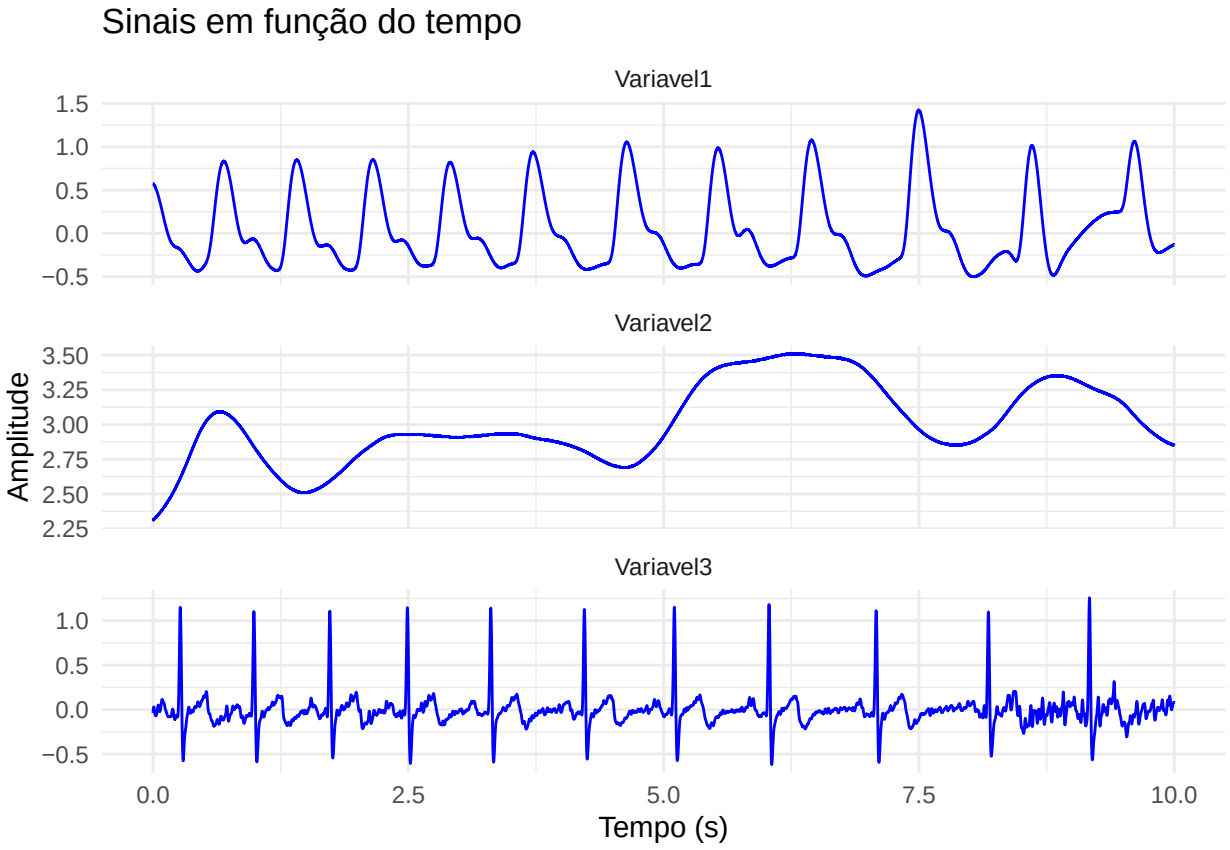
```r
library(tidyr)
```

```
##
## Anexando pacote: 'tidyr'

## O seguinte objeto é mascarado por 'package:magrittr':
##
##     extract
```

```r
library(readxl)

# Verificar estrutura dos dados antes de converter
str(dados2$data)
```

```
##  num [1:3759612, 1:3] 0.575 0.575 0.573 0.571 0.57 ...
```

```r
# Converter os dados para dataframe corretamente
data <- as.data.frame(dados2$data)
colnames(data) <- c("Variavel1", "Variavel2", "Variavel3")

# Garantir que isi seja um valor único e válido
isi <- as.numeric(dados2$isi[1]) / 1000  # Convertendo de ms para segundos

# Criar vetor de tempo corretamente
tempo <- seq(0, (nrow(data) - 1) * isi, length.out = nrow(data))

# Adicionar o tempo ao dataframe
data$Tempo <- tempo

# Filtrar o intervalo de tempo entre 0 e 10 segundos
data_filtrada <- data %>% filter(Tempo >= 0 & Tempo <= 10)

# Converter para formato longo para ggplot
data_long <- pivot_longer(data_filtrada, cols = c("Variavel1", "Variavel2", "Variavel3"),
                          names_to = "Variável", values_to = "Valor")

# Criar o gráfico com facet_wrap
ggplot(data_long, aes(x = Tempo, y = Valor)) +
  geom_line(color = "blue") +
  facet_wrap(~Variável, ncol = 1, scales = "free_y") +  # Uma coluna e três linhas
  labs(x = "Tempo (s)", y = "Amplitude", title = "Sinais em função do tempo") +
  theme_minimal()
```



Utilize a função ggsave da biblioteca ggplot2 para salvar o gráfico resultante. Considere o trecho de sinal entre 0 e 10 segundos. Você deve salvar as imagens nos formatos pdf e png, considerando as dimensões de 100 mm de largura por 200 mm de altura, e a resolução (dpi) de 300. As imagens geradas em cada um dos arquivos deve ser apresentada no relatório.

```
# Salvar o gráfico nos formatos PDF e PNG
ggsave("sinais_respiracao.pdf", width = 100, height = 200, units = "mm", dpi = 300)
ggsave("sinais_respiracao.png", width = 100, height = 200, units = "mm", dpi = 300)
```

Questão 3

A base de dados disponível na plataforma Moodle (ver figura abaixo) foi coletada utilizando-se o dispositivo TREMSEN (Precise Tremor Sensing), que é um dispositivo para a coleta de dados inerciais (i.e., movimentos).

A coleta de dados foi executada considerando os seguinte protocolo:

a) Um acelerômetro triaxial foi posicionado no dorso da mão, sobre o osso Capitato.

b) O eixo X do acelerômetro foi alinhado paralelamente à terceira falange distal.

c) Os dados foram coletados enquanto o participante realizava os seguintes movimentos: – 5 flexões do punho; – 5 extensões do punho; – 5 adução̧es do punho; – 5 abduções do punho.

Os dados coletados foram salvos no formato EDF e TXT, e nomeados de acordo com o tipo de movimento realizado (Adução, Flexão e Rotação).

```
library(readr)
library(writexl)

arquivos_txt <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.txt",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.txt",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.txt")
arquivos_xls <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.xlsx",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.xlsx",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.xlsx")

# Função para converter TXT para XLSX
converter_para_xlsx <- function(arquivo_txt, arquivo_xlsx) {
  dados <- read_delim(arquivo_txt, delim = "\t", skip = 2, col_names = TRUE)
  write_xlsx(dados, arquivo_xlsx)

  cat(paste("Arquivo salvo:", arquivo_xlsx, "\n"))
}

# Converter cada arquivo
mapply(converter_para_xlsx, arquivos_txt, arquivos_xls)
```

```
## New names:
## Rows: 490 Columns: 42
## -- Column specification
## -------------------------------------------------------- Delimiter: "\t" dbl
## (41): 0.0000000000, -0.7705806058, 2.2964827955, 3.5629816129, 0.4654001... lgl
## (1): ...42
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `0.0076295109` -> `0.0076295109...8`
## * `0.0076295109` -> `0.0076295109...9`
## * `0.0076295109` -> `0.0076295109...10`
## * `0.0076295109` -> `0.0076295109...11`
## * `0.0076295109` -> `0.0076295109...12`
```

```
## * `0.0076295109` -> `0.0076295109...13`
## * `0.0000610361` -> `0.0000610361...20`
## * `0.0000610361` -> `0.0000610361...21`
## * `0.0000610361` -> `0.0000610361...22`
## * `0.0000610361` -> `0.0000610361...23`
## * `0.0000610361` -> `0.0000610361...24`
## * `0.0000610361` -> `0.0000610361...25`
## * `0.0000305180` -> `0.0000305180...32`
## * `0.0000305180` -> `0.0000305180...33`
## * `0.0000305180` -> `0.0000305180...34`
## * `0.0000305180` -> `0.0000305180...35`
## * `0.0000305180` -> `0.0000305180...36`
## * `0.0000305180` -> `0.0000305180...37`
## * `` -> `...42`

## Arquivo salvo: C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.xlsx

## New names:
## Rows: 536 Columns: 42
## -- Column specification
## -------------------------------------------------------- Delimiter: "\t" dbl
## (41): 0.0000000000, -1.0605020218, -6.5232318608, 2.3880369268, 0.526436... lgl
## (1): ...42
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `0.0076295109` -> `0.0076295109...8`
## * `0.0076295109` -> `0.0076295109...9`
## * `0.0076295109` -> `0.0076295109...10`
## * `0.0076295109` -> `0.0076295109...11`
## * `0.0076295109` -> `0.0076295109...12`
## * `0.0076295109` -> `0.0076295109...13`
## * `0.0000610361` -> `0.0000610361...20`
## * `0.0000610361` -> `0.0000610361...21`
## * `0.0000610361` -> `0.0000610361...22`
## * `0.0000610361` -> `0.0000610361...23`
## * `0.0000610361` -> `0.0000610361...24`
## * `0.0000610361` -> `0.0000610361...25`
## * `0.0000305180` -> `0.0000305180...32`
## * `0.0000305180` -> `0.0000305180...33`
## * `0.0000305180` -> `0.0000305180...34`
## * `0.0000305180` -> `0.0000305180...35`
## * `0.0000305180` -> `0.0000305180...36`
## * `0.0000305180` -> `0.0000305180...37`
## * `` -> `...42`

## Arquivo salvo: C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.xlsx

## New names:
## Rows: 517 Columns: 42
## -- Column specification
## -------------------------------------------------------- Delimiter: "\t" dbl
## (41): 0.0000000000, -437.0565346761, 64.1412985428, 128.9158464942, 0.95... lgl
## (1): ...42
```

```
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘0.0076295109‘ -> ‘0.0076295109...8‘
## * ‘0.0076295109‘ -> ‘0.0076295109...9‘
## * ‘0.0076295109‘ -> ‘0.0076295109...10‘
## * ‘0.0076295109‘ -> ‘0.0076295109...11‘
## * ‘0.0076295109‘ -> ‘0.0076295109...12‘
## * ‘0.0076295109‘ -> ‘0.0076295109...13‘
## * ‘0.0000610361‘ -> ‘0.0000610361...20‘
## * ‘0.0000610361‘ -> ‘0.0000610361...21‘
## * ‘0.0000610361‘ -> ‘0.0000610361...22‘
## * ‘0.0000610361‘ -> ‘0.0000610361...23‘
## * ‘0.0000610361‘ -> ‘0.0000610361...24‘
## * ‘0.0000610361‘ -> ‘0.0000610361...25‘
## * ‘0.0000305180‘ -> ‘0.0000305180...32‘
## * ‘0.0000305180‘ -> ‘0.0000305180...33‘
## * ‘0.0000305180‘ -> ‘0.0000305180...34‘
## * ‘0.0000305180‘ -> ‘0.0000305180...35‘
## * ‘0.0000305180‘ -> ‘0.0000305180...36‘
## * ‘0.0000305180‘ -> ‘0.0000305180...37‘
## * ‘‘ -> ‘...42‘
```

```
## Arquivo salvo: C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.xlsx
```

```
## $‘C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.txt‘
## NULL
##
## $‘C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.txt‘
## NULL
##
## $‘C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.txt‘
## NULL
```

Salvar o arquivo xls gerado no formato csv.

```
# Carregar os pacotes
library(readxl)

# Definir os caminhos dos arquivos XLSX e CSV
arquivos_xlsx <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.xlsx",
                   "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.xlsx",
                   "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.xlsx")

arquivos_csv <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.csv",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.csv",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.csv")

# Função para converter XLSX para CSV
converter_para_csv <- function(arquivo_xlsx, arquivo_csv) {
  # Ler o arquivo XLSX
  dados <- read_excel(arquivo_xlsx)

  # Salvar como CSV
```

```r
  write.csv(dados, arquivo_csv, row.names = FALSE)

  cat(paste("Arquivo salvo:", arquivo_csv, "\n"))
}

# Converter cada arquivo
mapply(converter_para_csv, arquivos_xlsx, arquivos_csv)
```

```
## Arquivo salvo: C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.csv
## Arquivo salvo: C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.csv
## Arquivo salvo: C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.csv
```

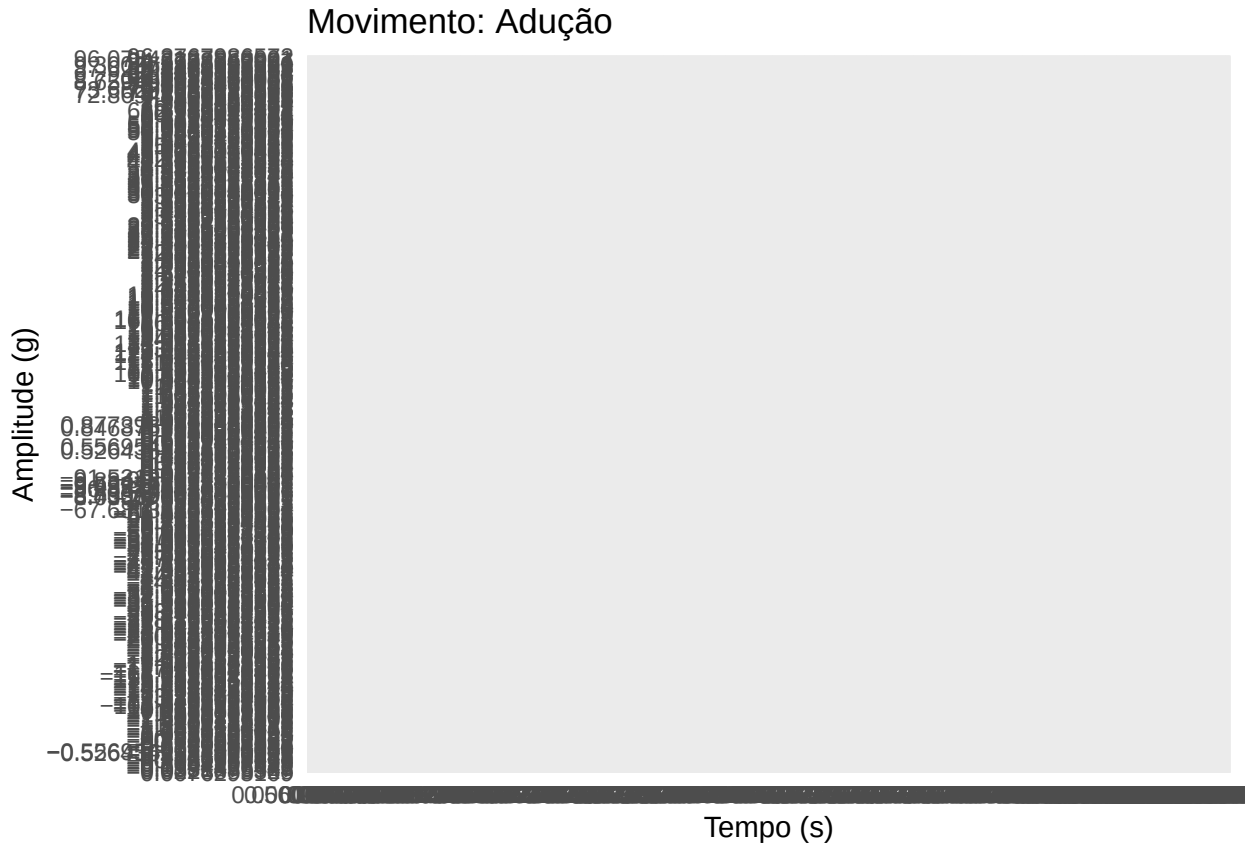
```
## $`C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.xlsx`
## NULL
##
## $`C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.xlsx`
## NULL
##
## $`C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.xlsx`
## NULL
```

Abrir o arquivo coletado em xls no R e gerar um gráfico utilizando o ggplot. Incluir legendas dos eixos x - tempo (s) e y - amplitude (g).

```r
# Carregar os pacotes
library(readxl)
library(ggplot2)
library(dplyr)

# Definir os caminhos dos arquivos XLSX
arquivos_xlsx <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.xlsx",
                   "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.xlsx",
                   "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.xlsx")

nomes_movimentos <- c("Adução", "Rotação", "Flexão")

# Função para carregar e organizar os dados
carregar_dados <- function(arquivo, movimento) {
  dados <- read_excel(arquivo, col_names = FALSE)  # Ler sem nomes de colunas
  colnames(dados)[1:2] <- c("Tempo", "Amplitude")  # Nomear as duas primeiras colunas relevantes
  dados <- dados %>% select(Tempo, Amplitude)  # Selecionar apenas Tempo e Amplitude
  dados$Movimento <- movimento
  return(dados)
}

# Criar gráficos separados para cada movimento
for (i in 1:3) {
  dados <- carregar_dados(arquivos_xlsx[i], nomes_movimentos[i])

  p <- ggplot(dados, aes(x = Tempo, y = Amplitude)) +
    geom_line(color = "blue") +
    labs(x = "Tempo (s)", y = "Amplitude (g)", title = paste("Movimento:", nomes_movimentos[i])) +
```

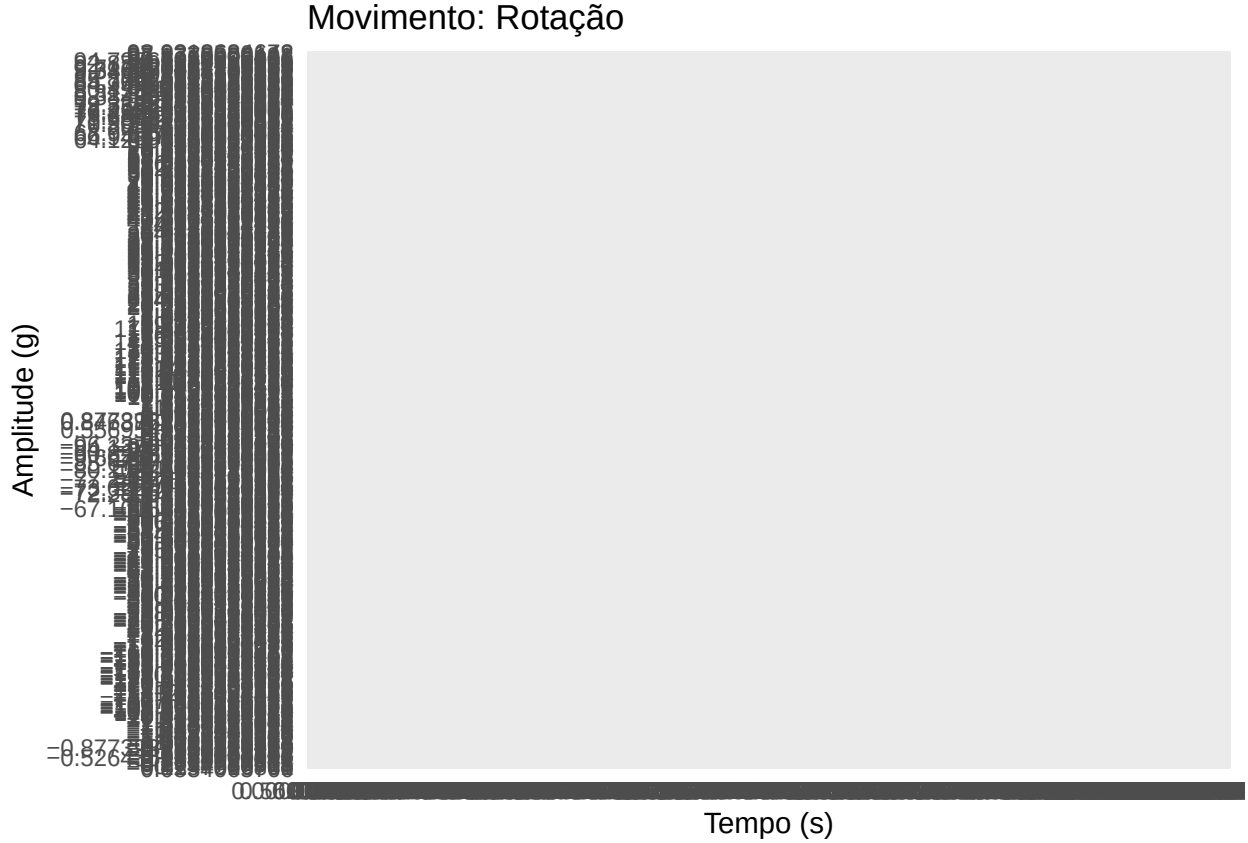
```
    theme_minimal()

  print(p)
}
```

```
## New names:
## `geom_line()`: Each group consists of only one observation. i Do you need to
## adjust the group aesthetic?
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
## * `` -> `...15`
## * `` -> `...16`
## * `` -> `...17`
## * `` -> `...18`
## * `` -> `...19`
## * `` -> `...20`
## * `` -> `...21`
## * `` -> `...22`
## * `` -> `...23`
## * `` -> `...24`
## * `` -> `...25`
## * `` -> `...26`
## * `` -> `...27`
## * `` -> `...28`
## * `` -> `...29`
## * `` -> `...30`
## * `` -> `...31`
## * `` -> `...32`
## * `` -> `...33`
## * `` -> `...34`
## * `` -> `...35`
## * `` -> `...36`
## * `` -> `...37`
## * `` -> `...38`
## * `` -> `...39`
## * `` -> `...40`
## * `` -> `...41`
## * `` -> `...42`
```



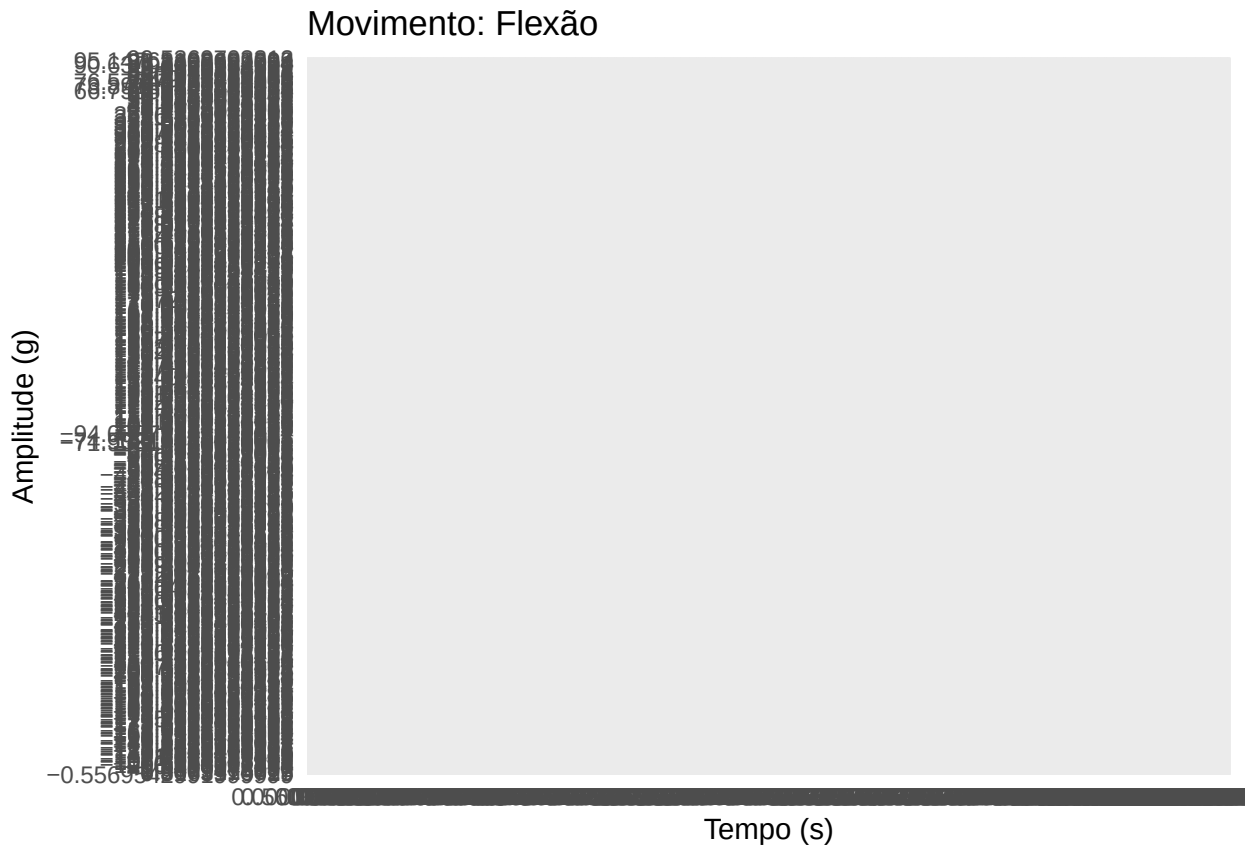

```
## New names:
## `geom_line()`: Each group consists of only one observation. i Do you need to
## adjust the group aesthetic?
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
## * `` -> `...15`
## * `` -> `...16`
## * `` -> `...17`
## * `` -> `...18`
## * `` -> `...19`
## * `` -> `...20`
## * `` -> `...21`
## * `` -> `...22`
## * `` -> `...23`
```

```
## * `` -> `...24`
## * `` -> `...25`
## * `` -> `...26`
## * `` -> `...27`
## * `` -> `...28`
## * `` -> `...29`
## * `` -> `...30`
## * `` -> `...31`
## * `` -> `...32`
## * `` -> `...33`
## * `` -> `...34`
## * `` -> `...35`
## * `` -> `...36`
## * `` -> `...37`
## * `` -> `...38`
## * `` -> `...39`
## * `` -> `...40`
## * `` -> `...41`
## * `` -> `...42`
```




```
## New names:
## `geom_line()`: Each group consists of only one observation. i Do you need to
## adjust the group aesthetic?
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
```

```
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
## * `` -> `...14`
## * `` -> `...15`
## * `` -> `...16`
## * `` -> `...17`
## * `` -> `...18`
## * `` -> `...19`
## * `` -> `...20`
## * `` -> `...21`
## * `` -> `...22`
## * `` -> `...23`
## * `` -> `...24`
## * `` -> `...25`
## * `` -> `...26`
## * `` -> `...27`
## * `` -> `...28`
## * `` -> `...29`
## * `` -> `...30`
## * `` -> `...31`
## * `` -> `...32`
## * `` -> `...33`
## * `` -> `...34`
## * `` -> `...35`
## * `` -> `...36`
## * `` -> `...37`
## * `` -> `...38`
## * `` -> `...39`
## * `` -> `...40`
## * `` -> `...41`
## * `` -> `...42`
```



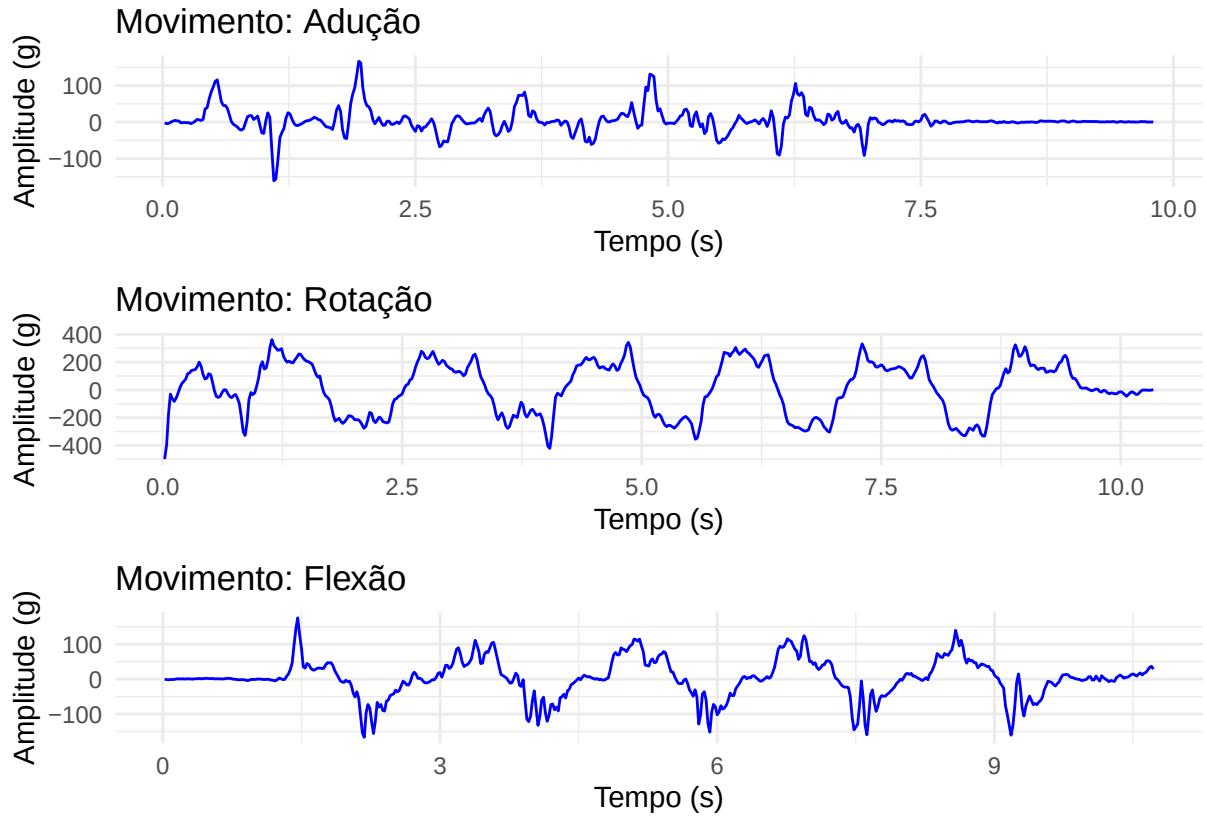
Abrir o arquivo coletado em csv no R e gerar um gráfico utilizando o ggplot. Incluir as legendas dos eixos x - tempo (s) e y - amplitude (g).

```
# Carregar os pacotes
library(ggplot2)
library(readr)
library(patchwork)

# Definir os caminhos dos arquivos CSV
arquivos_csv <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.csv",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.csv",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.csv")

nomes_movimentos <- c("Adução", "Rotação", "Flexão")

# Função para carregar os dados e gerar o gráfico
gerar_grafico <- function(arquivo_csv, movimento) {
  # Ler o arquivo CSV
  dados <- read_csv(arquivo_csv)

  # Ajustar os nomes das colunas, se necessário
  colnames(dados)[1:2] <- c("Tempo", "Amplitude")  # Ajuste conforme necessário

  # Criar o gráfico
  p <- ggplot(dados, aes(x = Tempo, y = Amplitude)) +
```

```
    geom_line(color = "blue") +
    labs(x = "Tempo (s)", y = "Amplitude (g)", title = paste("Movimento:", movimento)) +
    theme_minimal()

  return(p)
}

# Gerar os três gráficos
graficos <- lapply(1:3, function(i) gerar_grafico(arquivos_csv[i], nomes_movimentos[i]))
```

```
## Rows: 490 Columns: 42
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (41): 0.0000000000, -0.7705806058, 2.2964827955, 3.5629816129, 0.4654001...
## lgl  (1): ...42
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 517 Columns: 42
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (41): 0.0000000000, -437.0565346761, 64.1412985428, 128.9158464942, 0.95...
## lgl  (1): ...42
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 536 Columns: 42
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (41): 0.0000000000, -1.0605020218, -6.5232318608, 2.3880369268, 0.526436...
## lgl  (1): ...42
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Combinar os gráficos um abaixo do outro
final_plot <- graficos[[1]] / graficos[[2]] / graficos[[3]] + plot_layout(nrow = 3)

# Exibir os gráficos
print(final_plot)
```



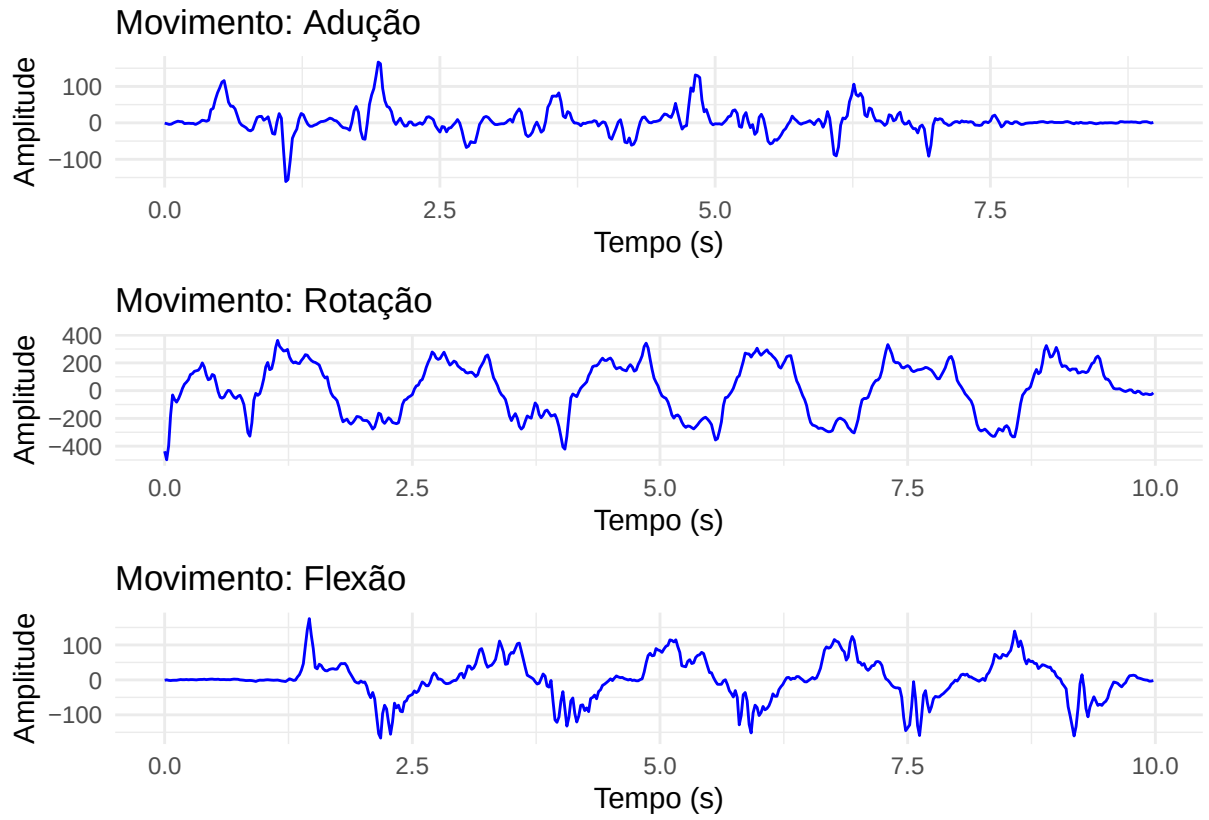
Abrir o arquivo coletado em edf no R e gerar um gráfico utilizando o ggplot. Incluir as legendas dos eixos x - tempo (s) e y - amplitude (g).

```
# Carregar os pacotes
library(edfReader)
library(ggplot2)
library(patchwork)

# Definir os caminhos dos arquivos EDF
arquivos_edf <- c("C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Aduccao.edf",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Rotacao.edf",
                  "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 3/Flexao.edf")

nomes_movimentos <- c("Adução", "Rotação", "Flexão")

# Função para carregar os dados e gerar o gráfico
gerar_grafico <- function(arquivo_edf, movimento) {
  # Ler o arquivo EDF
  edf <- readEdfHeader(arquivo_edf)  # Lê o cabeçalho do arquivo
  sinais <- readEdfSignals(edf)  # Lê os sinais do arquivo

  # Exibir os nomes dos sinais disponíveis
  print(names(sinais))

  # Selecionar o primeiro sinal disponível para plotagem
```

```r
  tempo <- seq(0, length(sinais[[1]]$signal) - 1) / sinais[[1]]$sRate  # Criar vetor de tempo
  amplitude <- sinais[[1]]$signal  # Sinal correspondente

  # Criar dataframe
  dados <- data.frame(Tempo = tempo, Amplitude = amplitude)

  # Criar o gráfico
  p <- ggplot(dados, aes(x = Tempo, y = Amplitude)) +
    geom_line(color = "blue") +
    labs(x = "Tempo (s)", y = "Amplitude", title = paste("Movimento:", movimento)) +
    theme_minimal()

  return(p)
}

# Gerar os três gráficos
graficos <- lapply(1:3, function(i) gerar_grafico(arquivos_edf[i], nomes_movimentos[i]))
```

```
##  [1] "Ch: 0-:G1.X"     "Ch: 1-:G1.Y"     "Ch: 2-:G1.Z"     "Ch: 3-:G2.X"
##  [5] "Ch: 4-:G2.Y"     "Ch: 5-:G2.Z"     "Ch: 6-:G2.X"     "Ch: 7-:G2.Y"
##  [9] "Ch: 8-:G2.Z"     "Ch: 9-:G2.X"     "Ch: 10-:G2.Y"    "Ch: 11-:G2.Z"
## [13] "Ch: 12-:A1.X"    "Ch: 13-:A1.Y"    "Ch: 14-:A1.Z"    "Ch: 15-:A2.X"
## [17] "Ch: 16-:A2.Y"    "Ch: 17-:A2.Z"    "Ch: 18-:A1.X"    "Ch: 19-:A1.Y"
## [21] "Ch: 20-:A1.Z"    "Ch: 21-:A2.X"    "Ch: 22-:A2.Y"    "Ch: 23-:A2.Z"
## [25] "Ch: 24-:M1.X"    "Ch: 25-:M1.Y"    "Ch: 26-:M1.Z"    "Ch: 27-:M2.X"
## [29] "Ch: 28-:M2.Y"    "Ch: 29-:M2.Z"    "Ch: 30-:M1.X"    "Ch: 31-:M1.Y"
## [33] "Ch: 32-:M1.Z"    "Ch: 33-:M2.X"    "Ch: 34-:M2.Y"    "Ch: 35-:M2.Z"
## [37] "Ch: 36-:EMG1"    "Ch: 37-:EMG2"    "Ch: 38-:Pulse A" "Ch: 39-:Pulse B"
## [41] "EDF Annotations"
##  [1] "Ch: 0-:G1.X"     "Ch: 1-:G1.Y"     "Ch: 2-:G1.Z"     "Ch: 3-:G2.X"
##  [5] "Ch: 4-:G2.Y"     "Ch: 5-:G2.Z"     "Ch: 6-:G2.X"     "Ch: 7-:G2.Y"
##  [9] "Ch: 8-:G2.Z"     "Ch: 9-:G2.X"     "Ch: 10-:G2.Y"    "Ch: 11-:G2.Z"
## [13] "Ch: 12-:A1.X"    "Ch: 13-:A1.Y"    "Ch: 14-:A1.Z"    "Ch: 15-:A2.X"
## [17] "Ch: 16-:A2.Y"    "Ch: 17-:A2.Z"    "Ch: 18-:A1.X"    "Ch: 19-:A1.Y"
## [21] "Ch: 20-:A1.Z"    "Ch: 21-:A2.X"    "Ch: 22-:A2.Y"    "Ch: 23-:A2.Z"
## [25] "Ch: 24-:M1.X"    "Ch: 25-:M1.Y"    "Ch: 26-:M1.Z"    "Ch: 27-:M2.X"
## [29] "Ch: 28-:M2.Y"    "Ch: 29-:M2.Z"    "Ch: 30-:M1.X"    "Ch: 31-:M1.Y"
## [33] "Ch: 32-:M1.Z"    "Ch: 33-:M2.X"    "Ch: 34-:M2.Y"    "Ch: 35-:M2.Z"
## [37] "Ch: 36-:EMG1"    "Ch: 37-:EMG2"    "Ch: 38-:Pulse A" "Ch: 39-:Pulse B"
## [41] "EDF Annotations"
##  [1] "Ch: 0-:G1.X"     "Ch: 1-:G1.Y"     "Ch: 2-:G1.Z"     "Ch: 3-:G2.X"
##  [5] "Ch: 4-:G2.Y"     "Ch: 5-:G2.Z"     "Ch: 6-:G2.X"     "Ch: 7-:G2.Y"
##  [9] "Ch: 8-:G2.Z"     "Ch: 9-:G2.X"     "Ch: 10-:G2.Y"    "Ch: 11-:G2.Z"
## [13] "Ch: 12-:A1.X"    "Ch: 13-:A1.Y"    "Ch: 14-:A1.Z"    "Ch: 15-:A2.X"
## [17] "Ch: 16-:A2.Y"    "Ch: 17-:A2.Z"    "Ch: 18-:A1.X"    "Ch: 19-:A1.Y"
## [21] "Ch: 20-:A1.Z"    "Ch: 21-:A2.X"    "Ch: 22-:A2.Y"    "Ch: 23-:A2.Z"
## [25] "Ch: 24-:M1.X"    "Ch: 25-:M1.Y"    "Ch: 26-:M1.Z"    "Ch: 27-:M2.X"
## [29] "Ch: 28-:M2.Y"    "Ch: 29-:M2.Z"    "Ch: 30-:M1.X"    "Ch: 31-:M1.Y"
## [33] "Ch: 32-:M1.Z"    "Ch: 33-:M2.X"    "Ch: 34-:M2.Y"    "Ch: 35-:M2.Z"
## [37] "Ch: 36-:EMG1"    "Ch: 37-:EMG2"    "Ch: 38-:Pulse A" "Ch: 39-:Pulse B"
## [41] "EDF Annotations"
```

```r
# Combinar os gráficos um abaixo do outro
final_plot <- graficos[[1]] / graficos[[2]] / graficos[[3]] + plot_layout(nrow = 3)

# Exibir os gráficos
print(final_plot)
```



Questão 4

Arquivos do tipo EDF são bastante utilizados na indústria pois eles possuem uma padronização que permite o fácil compartilhamento de dados. Nesta questão você deverá compreender a estrutura básica de um arquivo EDF (https://www.edfplus.info/specs/edf.html) e responder os seguintes itens:

Importe o arquivo binário V16C1RCC92.edf. Dica: utilize a biblioteca readr e a função read_file_raw.

```
# Instalar o pacote caso não esteja instalado
if (!require(readr)) install.packages("readr", repos = "http://cran.us.r-project.org")

# Carregar o pacote
library(readr)

# Definir o caminho do arquivo
arquivo_edf <- "C:/Users/heito/Desktop/Módulo 2 - PSB/Questão 4/V16C1RCC92.edf"

# Ler o arquivo binário
dados_4 <- read_file_raw(arquivo_edf)
```

Considerando as informações abaixo que podem ser extraídas do arquivo V16C1RCC92.edf, informe o contéudo de cada um dos itens da tabela. Dica: após fazer a leitura do arquivo binário use a função rawToChar.

```
# Carregar o pacote
library(tibble)

# Criar a tabela com os dados extraídos
tabela_cabecalho <- tibble::tibble(
  "Tamanho" = c("8 ascii", "80 ascii", "80 ascii", "8 ascii", "8 ascii",
                "8 ascii", "44 ascii", "8 ascii", "8 ascii", "4 ascii"),

  "Explicação" = c(
    "Version of this data format (0)",
    "Local patient identification",
    "Local recording identification",
    "Start date of recording (dd.mm.yy)",
    "Start time of recording (hh.mm.ss)",
    "Number of bytes in header record",
    "Reserved",
    "Number of data records (-1 if unknown)",
    "Duration of a data record, in seconds",
    "Number of signals (ns) in data record"
  ),

  "Conteúdo Extraído" = c(
    "0",
    "X X X X",
    "Startdate 24-JAN-2018 X X TREMESEN_Firmware_version:_V2.4-2017",
    "24.01.18",
    "14.13.47",
```

```
    "10752",
    "EDF+C",
    "21",
    "1",
    "41"
  )
)

# Exibir a tabela
print(tabela_cabecalho)
```

```
## # A tibble: 10 x 3
##    Tamanho  Explicação                              ‘Conteúdo Extraído‘
##    <chr>    <chr>                                   <chr>
##  1 8 ascii  Version of this data format (0)         0
##  2 80 ascii Local patient identification            X X X X
##  3 80 ascii Local recording identification          Startdate 24-JAN-2018 X X TR~
##  4 8 ascii  Start date of recording (dd.mm.yy)      24.01.18
##  5 8 ascii  Start time of recording (hh.mm.ss)      14.13.47
##  6 8 ascii  Number of bytes in header record        10752
##  7 44 ascii Reserved                                EDF+C
##  8 8 ascii  Number of data records (-1 if unknown) 21
##  9 8 ascii  Duration of a data record, in seconds  1
## 10 4 ascii  Number of signals (ns) in data record  41
```