

Universidade Federal de Uberlândia

Listado Módulo 7

Levy Gabriel da Silva Galvão

Uberlândia
2021

Módulo 7

Exercício 1

Gere um sinal amostrado a partir de uma distribuição normal, com amplitude variando entre 0 e 1, de duração de 10 s, e amostrado a 5 kHz. Ilustre o processo de janelamento sem sobreposição aplicado a este sinal. A duração de cada janela deverá ser de 500 ms. O sinal gerado, e o instante de início e fim de cada janela deverão ser plotados utilizando-se a biblioteca gráfica dyraphs. Dica: a função dyEvent pode ser utilizada para gerar linhas verticais em instantes desejados. Veja o trecho de código abaixo que ilustra esta situação. O resultado do gráfico gerado deve ser apresentado como parte do relatório.

```
library(htmltools)
library(htmlwidgets)
library(dygraphs)
norm_dist <- function(length_, min_, max_){return(qnorm(runif(length_,min=pnorm(min_),max=pnorm(max_))))}
t_end <- 10 # s
fs <- 5000 # Hz
dt <- 1/fs # s
t <- seq(from=0, to=t_end-dt, by=dt)
n <- length(t)
process <- norm_dist(n, 0, 1)
print(max(process)) # max amplitude of random signal
```

```
## [1] 0.9999523
```

```
print(min(process)) # max amplitude of random signal
```

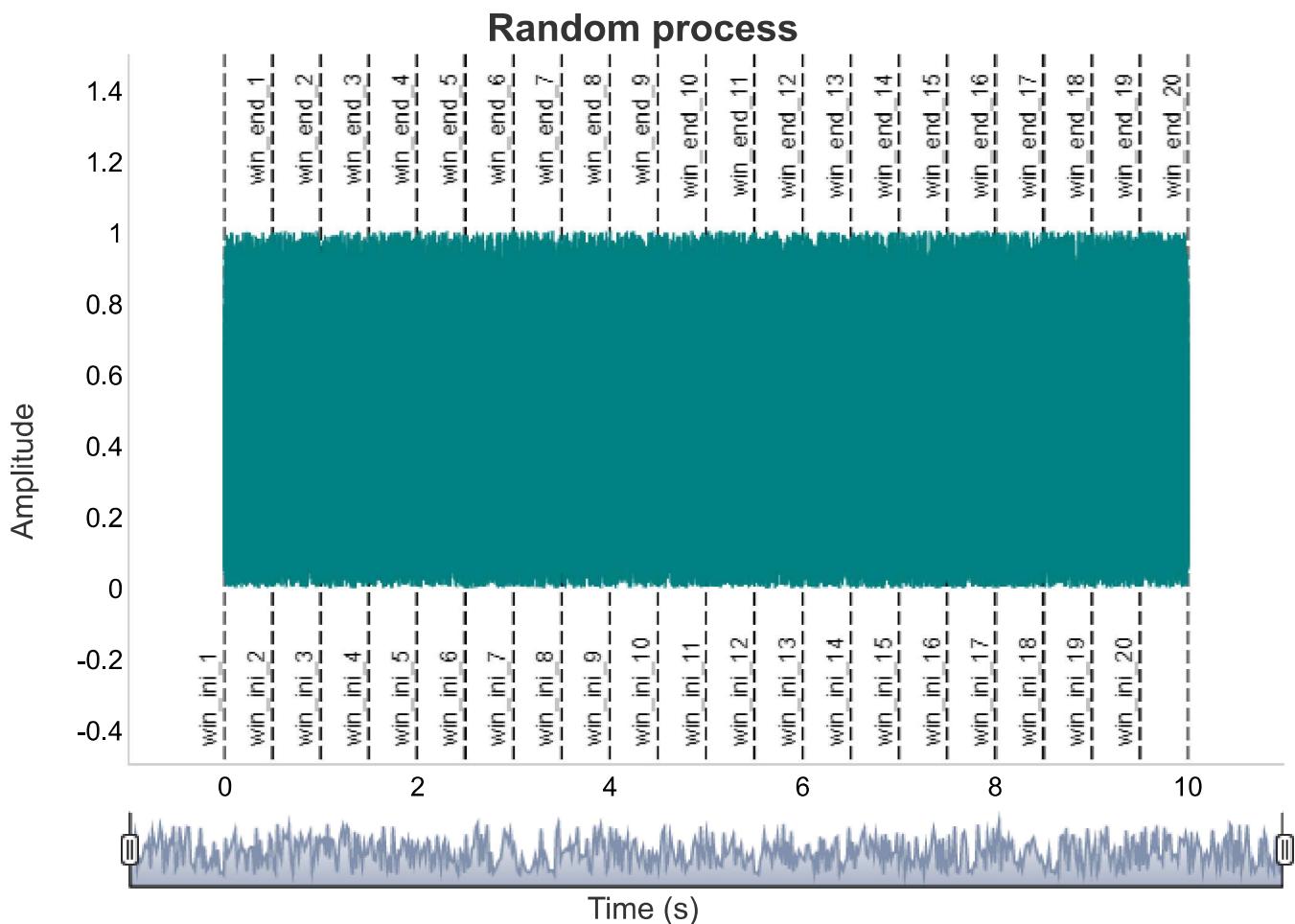
```
## [1] 9.739052e-06
```

```
win.dur <- 500/1000 # window dur. (s)
win.size <- win.dur * fs # window dur. (samples)
win.number <- round(n/win.size) # number of windows

win.idx_ini <- seq(from=1,to=(win.number)*win.size, by=win.size)
win.idx_end <- win.idx_ini+(win.size-1)
win.idx_end[win.idx_end>n] = n

# visualization of random process with normal dist.
data.frame(time=t, process=process) %>%
  dygraph(main='Random process') %>%
  dyEvent(x=(win.idx_end-1)*dt, label=sprintf('win_end_%d',1:win.number), labelLoc='top') %>%
  dyEvent(x=(win.idx_ini-1)*dt, label=sprintf('win_ini_%d',1:win.number), labelLoc='bottom') %>%
  dyAxis('y', label='Amplitude', valueRange=c(-0.5,1.5)) %>%
```

```
dyAxis('x', label='Time (s)') %>%
dyOptions(axisLineWidth=0.1, fillGraph=FALSE, drawGrid=FALSE) %>%
dyRangeSelector(dateWindow = c(-1,11))
```



Exercício 2

Considerando o problema da questão 1, altere a duração da janela para 1.000 ms, e apresente a solução para o mesmo problema adotando-se a sobreposição de janelas de 50%. O resultado do gráfico gerado deve ser apresentado como parte do relatório.

```
win.dur <- 1000/1000 # window dur. (s)
win.size <- win.dur * fs # window dur. (samples)
win.overlap_per <- 0.5 # window ovelarp (percentage)
win.overlap <- win.overlap_per*win.size # window overlap (samples)
win.number <- round(n/win.size/win.overlap_per) # number of windows

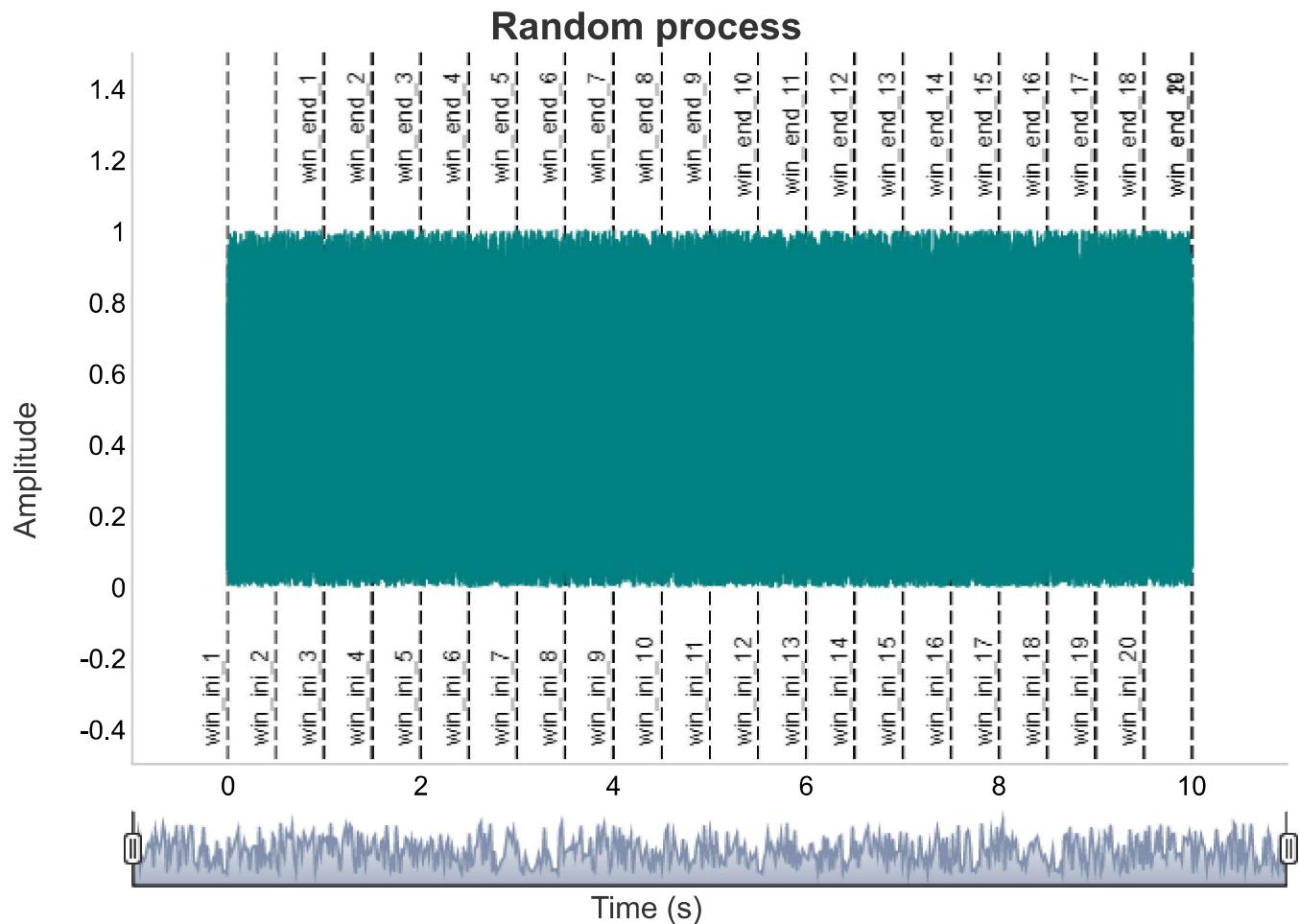
win.idx_ini <- seq(from=1,to=n, by=win.size-win.overlap)
win.idx_end <- win.idx_ini+(win.size-1)
win.idx_end[win.idx_end>n] = n

# visualization of random process with normal dist.
data.frame(time=t, process=process) %>%
  dygraph(main='Random process') %>%
  dyEvent(x=(win.idx_end-1)*dt, label=sprintf('win_end_%d',1:win.number), labelLoc='top') %>%
```

```

dyEvent(x=(win.idx_ini-1)*dt, label=sprintf('win_ini_%d',1:win.number), labelLoc='bottom')
%>%
dyAxis('y', label='Amplitude', valueRange=c(-0.5,1.5)) %>%
dyAxis('x', label='Time (s)') %>%
dyOptions(axisLineWidth=0.1, fillGraph=FALSE, drawGrid=FALSE) %>%
dyRangeSelector(dateWindow = c(-1,11))

```



Exercício 3

Na seção Estimativa de parâmetros ao longo do tempo foi dado um exemplo de como diferentes características captam informações distintas dos sinais. Explique que tipo de informação é mensurada pelas estatísticas: rms, média, desvio padrão, variância.

Os valores RMS e a média dizem respeito às alterações na amplitude do sinal, esta em relação a um deslocamento partindo do ponto central e aquele em relação a uma representação contínua da excursão alternada do sinal.

Já a variância e desvio padrão informam a variabilidade dos sinais devido às diversas transições e que, em alguns casos esta pode informar a excursão da amplitude do sinal e aquela pode informar sobre a potência do sinal.

Exercício 4

Gere um sinal sintético que represente a atividade eletromiográfica. Este sinal deve ser amostrado a 1 kHz e a sua duração deve ser de 1 segundo. Adicione ao sinal gerado uma componente linear e outra não linear. Desenvolva uma função que remova essas componentes do sinal corrompido. Como parte da solução, você deverá apresentar o gráfico do sinal simulado (sem qualquer tendência), o gráfico do sinal simulado com a presença da tendência linear, o gráfico do sinal simulado com a presença das tendências linear e não linear, e finalmente o gráfico do sinal após a remoção das tendências.

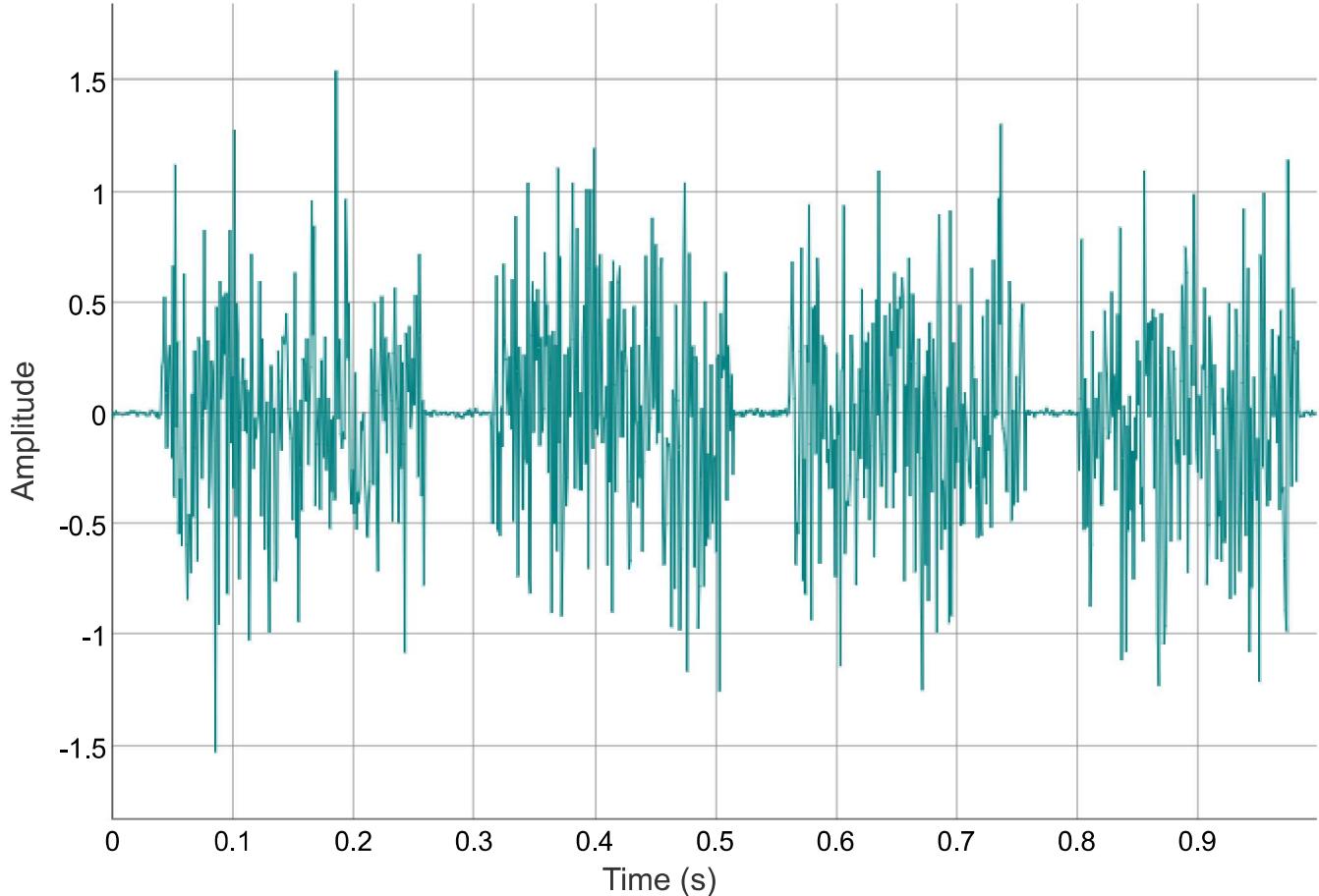
O sinal EMG foi sintetizado com a ajuda da biblioteca `biosignalEMG` :

```
library(biosignalEMG)

## Warning: package 'biosignalEMG' was built under R version 4.1.1

fs <- 1000 # Hz
dt <- 1/fs # Hz
t_end <- 1 # s
t <- seq(from=0, to=t_end-dt, by=dt)
n = length(t)
emg <- syntheticemg(n.length.out=n, samplingrate=fs,
                      on.sd=0.5, on.duration.mean=200, on.duration.sd=10,
                      off.sd=0.01, off.duration.mean=50, off.duration.sd=10,
                      on.mode.pos=0.10, shape.factor=0.10)$values
data.frame(x=t,y=emg) %>%
  dygraph(main='Uncorrupted EMG signal') %>%
  dyAxis('y', label='Amplitude') %>%
  dyAxis('x', label='Time (s)')
```

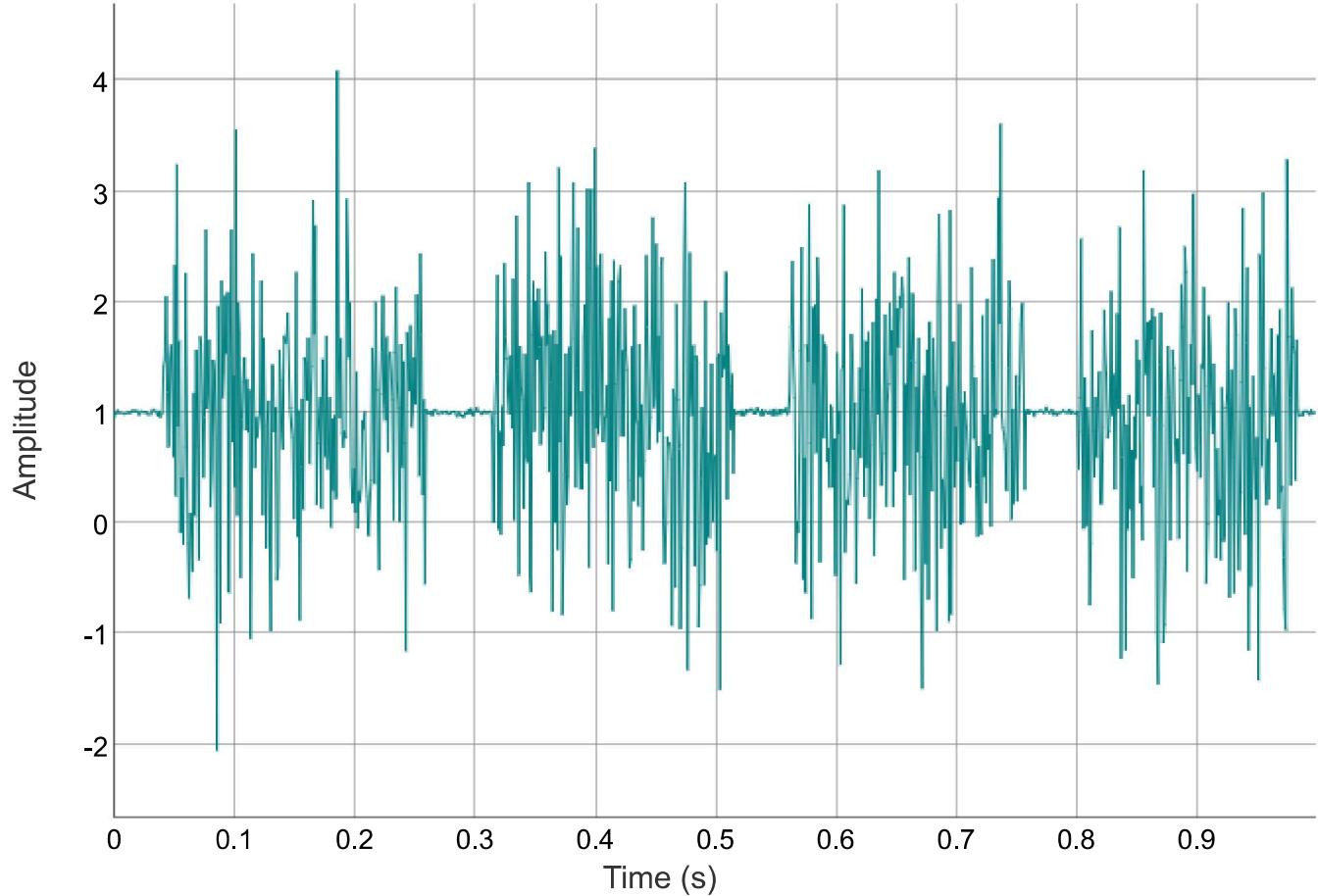
Uncorrupted EMG signal



Corrupção do sinal EMG com componentes lineares e não lineares:

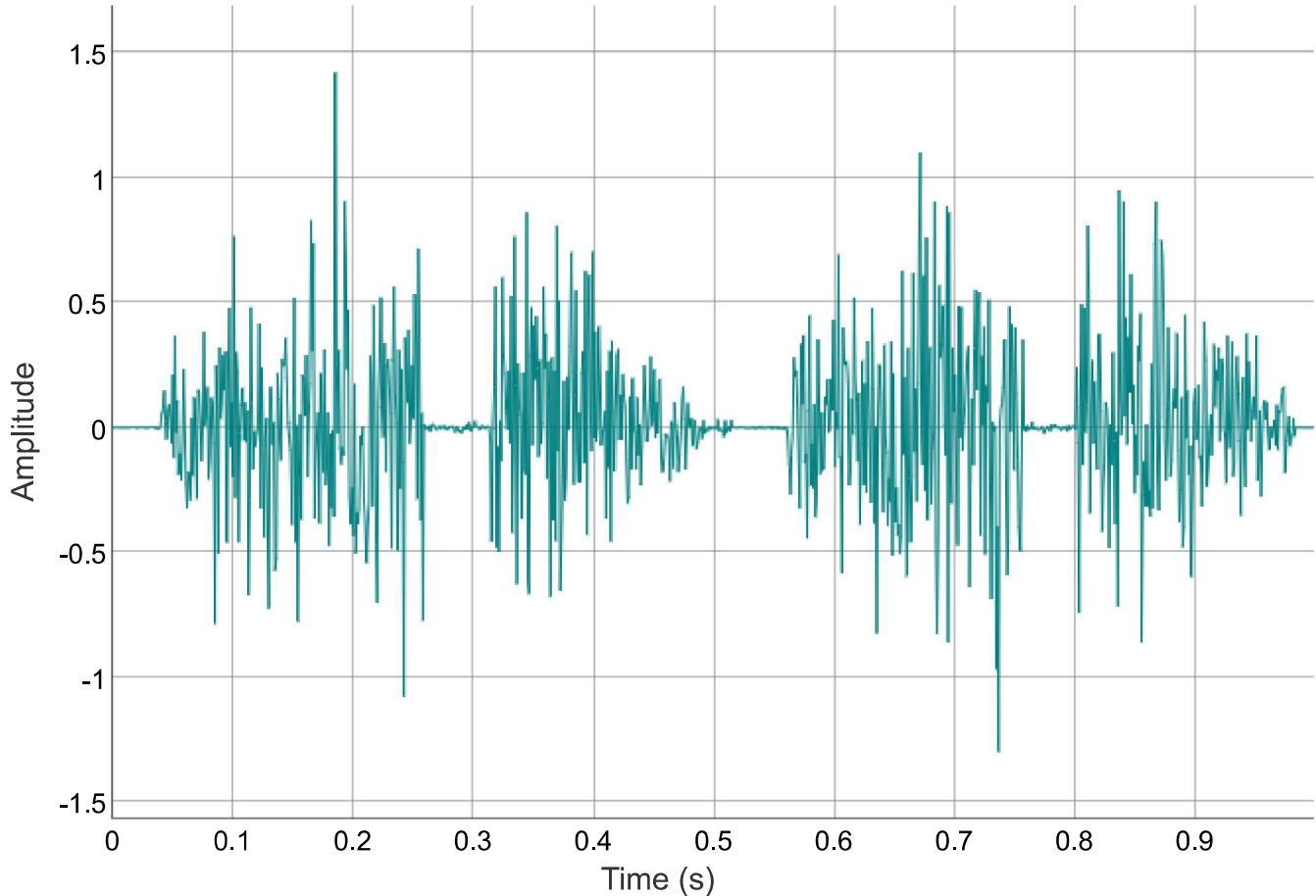
```
# Linear coefficients por EMG corruption
a <- 1
b <- 2
emg_linear <- a+b*emg # Linear corruption in the form a+bx
data.frame(x=t,y=emg_linear) %>%
  dygraph(main='Corrupted EMG signal with linear tendency') %>%
  dyAxis('y', label='Amplitude') %>%
  dyAxis('x', label='Time (s)')
```

Corrupted EMG signal with linear tendency



```
# non-linear EMG corruption
alpha <- 1
corr <- sin(2*pi*alpha*t)
emg_nonlinear <- emg*corr
data.frame(x=t,y=emg_nonlinear) %>%
  dygraph(main='Corrupted EMG signal with nonlinear tendency') %>%
  dyAxis('y', label='Amplitude') %>%
  dyAxis('x', label='Time (s)')
```

Corrupted EMG signal with nonlinear tendency



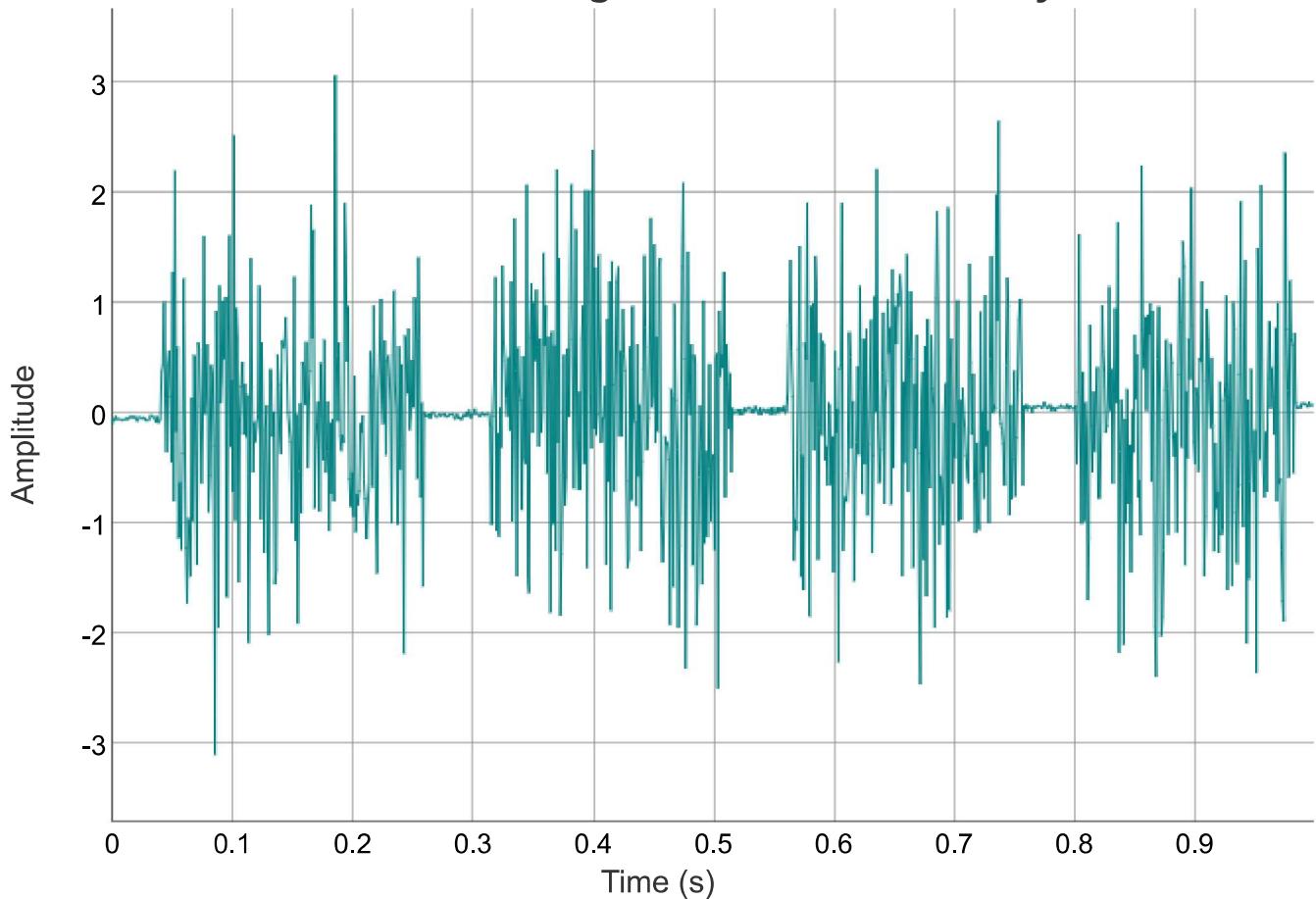
Removendo tendência linear por meio de um modelo linear aos dados e, em seguida removendo-o:

```
data <- data.frame(time=t,emg_linear)
linear_tendency <- lm(emg_linear~t, data=data)
print(linear_tendency)
```

```
##
## Call:
## lm(formula = emg_linear ~ t, data = data)
##
## Coefficients:
## (Intercept)          t
##       1.0474      -0.1279
```

```
linear_prediction <- predict(linear_tendency, data)
emg_linear_corrected <- data$emg_linear - linear_prediction
data.frame(x=t,y=emg_linear_corrected) %>%
  dygraph(main='Corrected EMG signal with linear tendency') %>%
  dyAxis('y', label='Amplitude') %>%
  dyAxis('x', label='Time (s)')
```

Corrected EMG signal with linear tendency



Apesar dos esforços o modelo não detectou o fator de escalonamento.

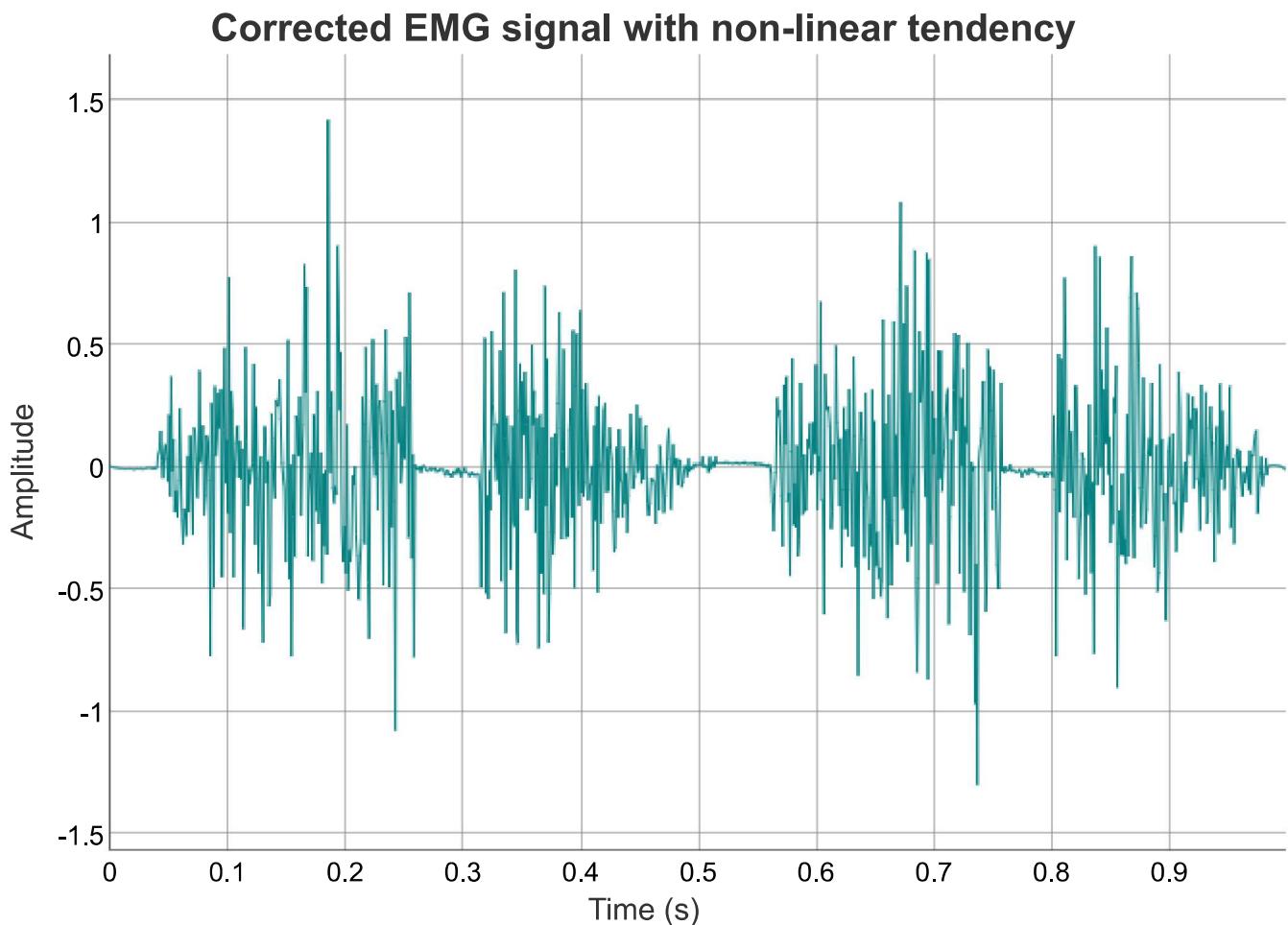
Removendo tendência não-linear por meio do ajuste por um polinômio de ordem 15 aos dados e, em seguida removendo-o:

```
data <- data.frame(time=t,emg_nonlinear)
nonlinear_tendency <- lm(emg_nonlinear~poly(t,15), data=data)
print(nonlinear_tendency)
```

```
##
## Call:
## lm(formula = emg_nonlinear ~ poly(t, 15), data = data)
##
## Coefficients:
## (Intercept)  poly(t, 15)1   poly(t, 15)2   poly(t, 15)3   poly(t, 15)4
##          0.014530     0.228332    -0.112450     0.144611    -0.032134
##  poly(t, 15)5   poly(t, 15)6   poly(t, 15)7   poly(t, 15)8   poly(t, 15)9
##         -0.417715     0.005417     0.147391    -0.159430    -0.089133
## poly(t, 15)10  poly(t, 15)11  poly(t, 15)12  poly(t, 15)13  poly(t, 15)14
##          0.208862     0.071563    -0.278319    -0.020758     0.187623
## poly(t, 15)15
##          0.091550
```

```
nonlinear_fit <- fitted(nonlinear_tendency)
emg_nonlinear_corrected <- data$emg_nonlinear - nonlinear_fit
data.frame(x=t,y=emg_nonlinear_corrected) %>%
  dygraph(main='Corrected EMG signal with non-linear tendency') %>%
```

```
dyAxis('y', label='Amplitude') %>%
dyAxis('x', label='Time (s)')
```



Apesar dos esforços a remoção da tendência não linear também não foi tão efetiva.

Exercício 5

Faça um programa que ilustre o conceito de média coerente. Este programa deverá considerar a média de 10 séries temporais, de 20 amostras cada uma. Gere gráficos que ilustrem os resultados obtidos.

A média coerente consiste na média das mesmas amostras entre várias cópias de um mesmo sinal na tentativa de superar o ruído e melhorar a relação sinal ruído. O conceito é ilustrado abaixo, tomando 10 séries temporais diferentes (porém com mesma distribuição) e tomando a média coerente considerando 20 amostras de cada uma das séries.

```
# generate 10 time-series
eq = function(t,u){u*t*(2-u*t)*exp(-u*t)}

t <- 1:1000
df <- data.frame(matrix(nrow=length(t),ncol=10))
df$t <- t
df$X1 <- eq(df$t, 0.001)
df$X2 <- eq(df$t, 0.002)
df$X3 <- eq(df$t, 0.003)
df$X4 <- eq(df$t, 0.004)
```

```

df$X5 <- eq(df$t, 0.005)
df$X6 <- eq(df$t, 0.006)
df$X7 <- eq(df$t, 0.007)
df$X8 <- eq(df$t, 0.008)
df$X9 <- eq(df$t, 0.009)
df$X10 <- eq(df$t, 0.01)

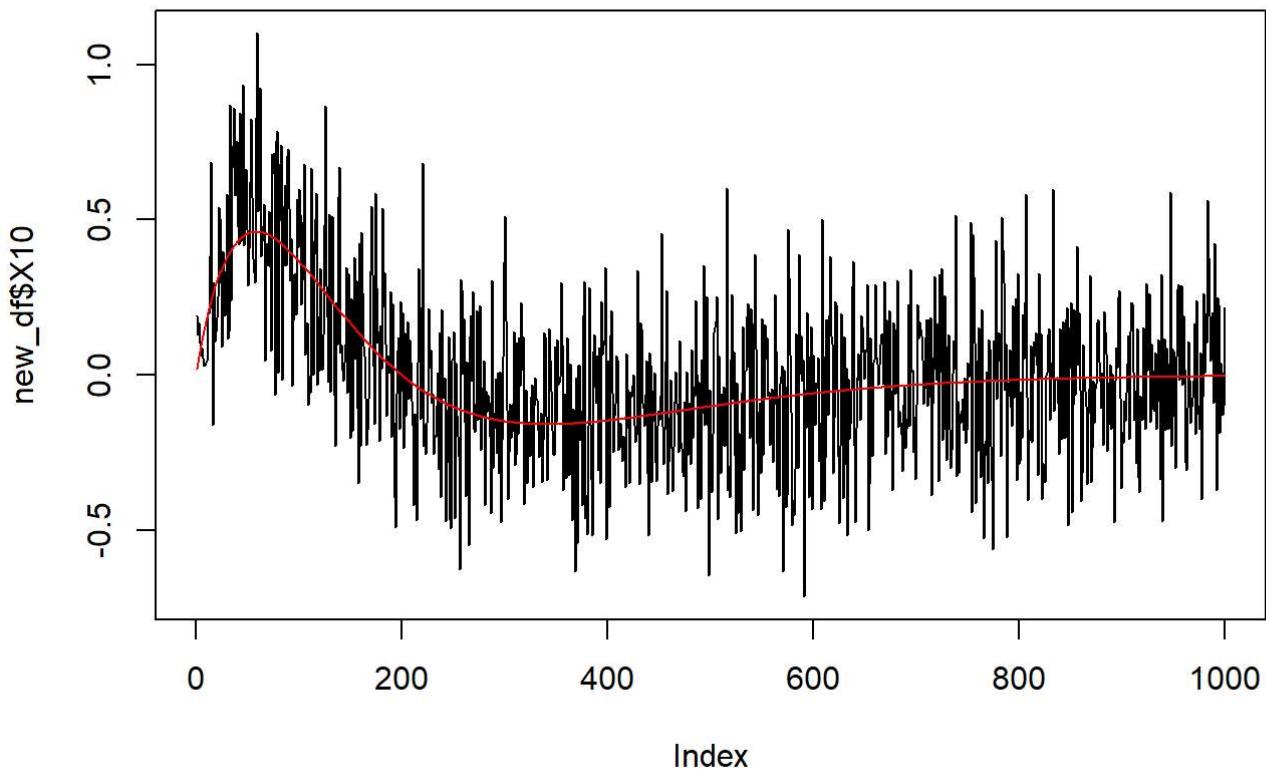
cohe_mean<-function(x,n){

  A <- matrix(NA,nrow=n, ncol = length(x))
  for (i in 1:n){
    A[i,] <- x+ rnorm(length(x))
  }
  cleaned <- colMeans(A)
  return(cleaned)
}

new_df <- data.frame(matrix(nrow=length(t),ncol=10))
names <- colnames(df)[1:length(df)-1]
for(i in length(names)){new_df[,i] <- cohe_mean(df[,i],20)}

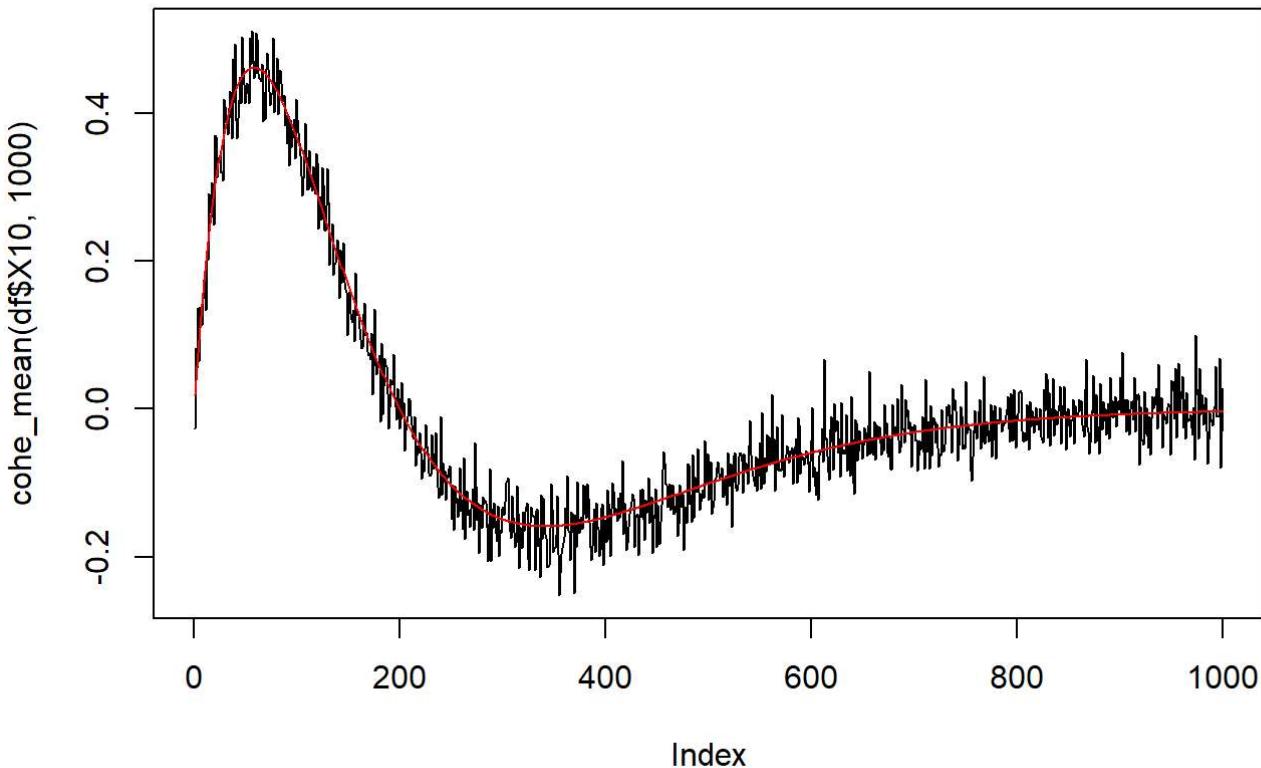
# result for one time-series
plot(new_df$X10, type='l')
lines(df$X10, col='red', type = 'l')

```



Assim esse exemplo ilustra um plot para o caso da última série temporal que, apesar do ruído, a média coerente fez com que o sinal fosse levemente reconhecido em relação ao sinal original. Para melhorar a SNR resultante a quantidade de amostras poderia ser aumentada. Como abaixo:

```
# result for one time-series
plot(cohe_mean(df$X10,1000), type='l')
lines(df$X10, col='red', type = 'l')
```



Exercício 6

Encontre a convolução entre as duas sequências $x[n]=\{ \blacktriangleright 2, 2, 4 \}$ e $h[n]=\{\blacktriangleright 1, 1, 1, 1, 1\}$. O símbolo \blacktriangleright indica o zero. Apresente todos os passos da solução, tal como ilustrado neste módulo.

Inicialmente foi implementada uma função que realiza a convolução com todos os passos descritos ao longo do código:

```
conv_ <- function(x,h)
{
  # step 1 - list k
  hLen <- length(h)-1
  xLen <- length(x)-1
  k <- seq(-hLen+1, hLen+xLen-1)
  cat('k:',k, '\n')

  # step 2 - revert h and aligned both vectors' zeroes
  h <- h[seq(hLen+1, 1, -1)]
  cat('reverted h',h, '\n')
  zeroX <- which(is.na(x)) # find x zero
  zeroH <- which(is.na(h))-1 # find h zero
```

```

x <- x[!is.na(x)] # remove x na
h <- h[!is.na(h)] # remove h na

# step 3 - sliding cross multiplication
xx <- rep(0, xLen+2*hLen-1)
hh <- xx
y <- rep(0, 1+length(hh)-hLen)
xx[(zeroH-zeroX+1):(zeroH+xLen-zeroX)] <- x
hh[1:zeroH] <- h
cat('sliding', '\n')
for(i in c(1:(1+length(hh)-hLen)))
{
  y[i] <- sum(xx*hh)
  cat('x:', xx, ', h:', hh, '\n')
  hh[(i+1):(i+hLen)] <- hh[(i):(i+hLen-1)]
  hh[i] <- 0
}
cat('y:', y, '\n')
return(y)
}

```

O ponto que indica zero foi simbolizado por um array cuja posição possui um valor NA, indicando que o valor imediatamente à direita é o zero.

```

x <- c(NA,2,2,4)
h <- c(NA,1,1,1,1,1)
y <- conv_(x,h)

```

```

## k: -4 -3 -2 -1 0 1 2 3 4 5 6 7
## reverted h 1 1 1 1 1 NA
## sliding
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 1 1 1 1 1 0 0 0 0 0 0 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 1 1 1 1 1 0 0 0 0 0 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 1 1 1 1 1 0 0 0 0 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 0 1 1 1 1 1 0 0 0 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 0 0 1 1 1 1 1 0 0 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 0 0 0 1 1 1 1 1 0 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 0 0 0 0 1 1 1 1 1 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 0 0 0 0 0 1 1 1 1 1 0
## x: 0 0 0 0 2 2 4 0 0 0 0 0 , h: 0 0 0 0 0 0 0 0 1 1 1 1 1 1
## y: 2 4 8 8 8 6 4 0

```

Exercício 7

Encontre a convolução entre as duas sequências $x[n]=\{2, \blacktriangleright 1, -2, 3, -4\}$ e $h[n]=\{\blacktriangleright 3, 2, 2, 1, 4\}$. O símbolo \blacktriangleright indica o zero. Apresente todos os passos da solução, tal como ilustrado neste módulo.

O mesmo se aplica no exemplo a seguir:

```

x <- c(2,NA,1,-2,3,-4)
h <- c(NA,3,2,2,1,4)
y <- conv_(x,h)

```

```

## k: -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
## reverted h 4 1 2 2 3 NA
## sliding
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 4 1 2 2 3 0 0 0 0 0 0 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 4 1 2 2 3 0 0 0 0 0 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 4 1 2 2 3 0 0 0 0 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 0 0 4 1 2 2 3 0 0 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 0 0 0 4 1 2 2 3 0 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 0 0 0 0 4 1 2 2 3 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 0 0 0 0 0 4 1 2 2 3 0 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 0 0 0 0 0 0 4 1 2 2 3 0 0
## x: 0 0 0 2 1 -2 3 -4 0 0 0 0 0 0 , h: 0 0 0 0 0 0 0 0 0 4 1 2 2 3 0
## y: 7 0 9 -1 0 -13 8 -16 0 0

```

Exercício 8

Utilizando o R faça um programa que realize de forma automática a detecção de bursts de atividade eletromiográfica. O sinal eletromiográfico deverá ser gerado de acordo com os seguintes critérios:

- frequência de amostragem = 1000 Hz
- duração da simulação = 10 segundos
- número total de bursts = 5
- duração de cada burst = 1 segundo
- trechos de ruído amostrados a partir de uma distribuição normal

Dica: utilize o valor RMS estimado a partir de janelas de 100 ms com sobreposição de 30%; defina um limiar e quando o valor RMS ultrapassá-lo, significa que o início de um burst está acontecendo; o término do burst pode ser detectado sempre que o valor RMS retornar ao nível inferior ao do limiar.

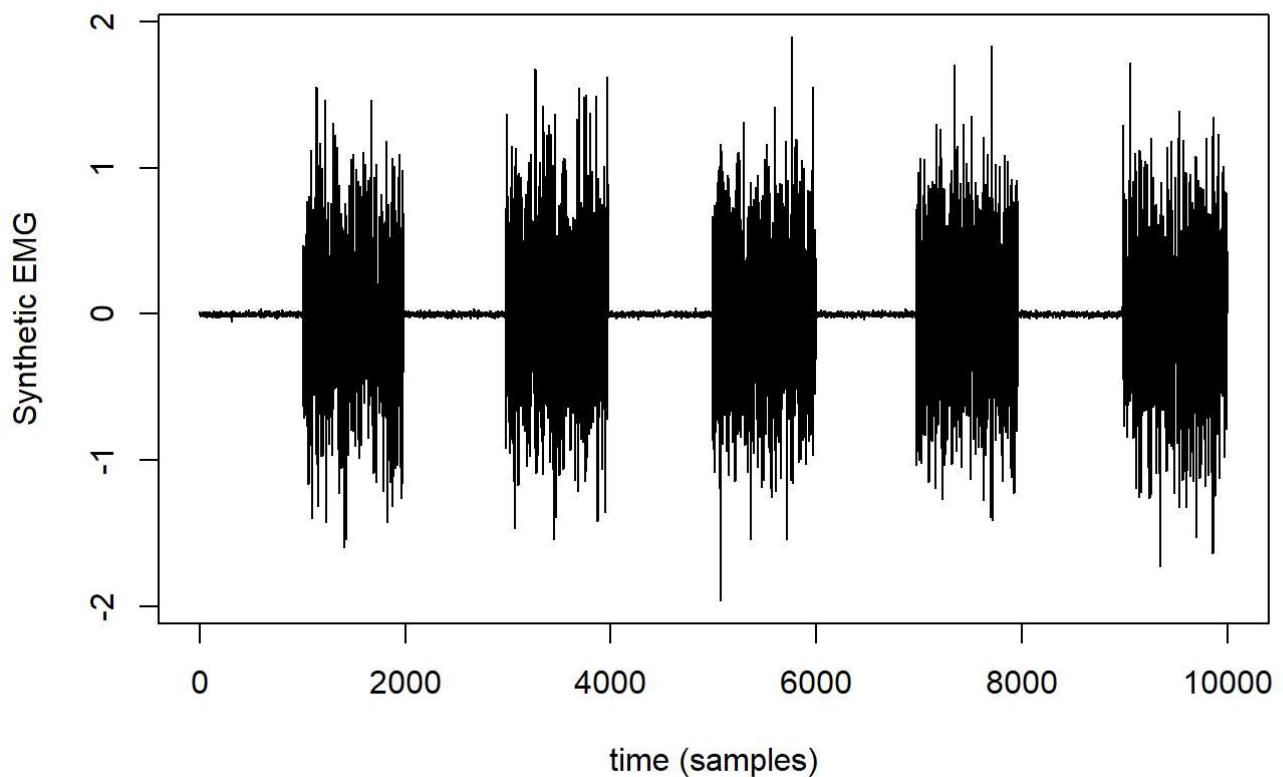
Geração do sinal EMG:

```

library(biosignalEMG)
fs <- 1000 # Hz
dt <- 1/fs # s
t_end <- 10 # s
t <- seq(from=0, to=t_end-dt, by=dt)
n = length(t)
n_bursts <- 5
dur_bursts <- 1000 # ms

emg <- syntheticemg(n.length.out=n, samplingrate=fs,
                      on.sd=0.5, on.duration.mean=dur_bursts, on.duration.sd=10,
                      off.sd=0.01, off.duration.mean=(n-n_bursts*dur_bursts)/n_bursts, off.duration.sd=10,
                      on.mode.pos=0.010, shape.factor=0.010)
plot(emg)

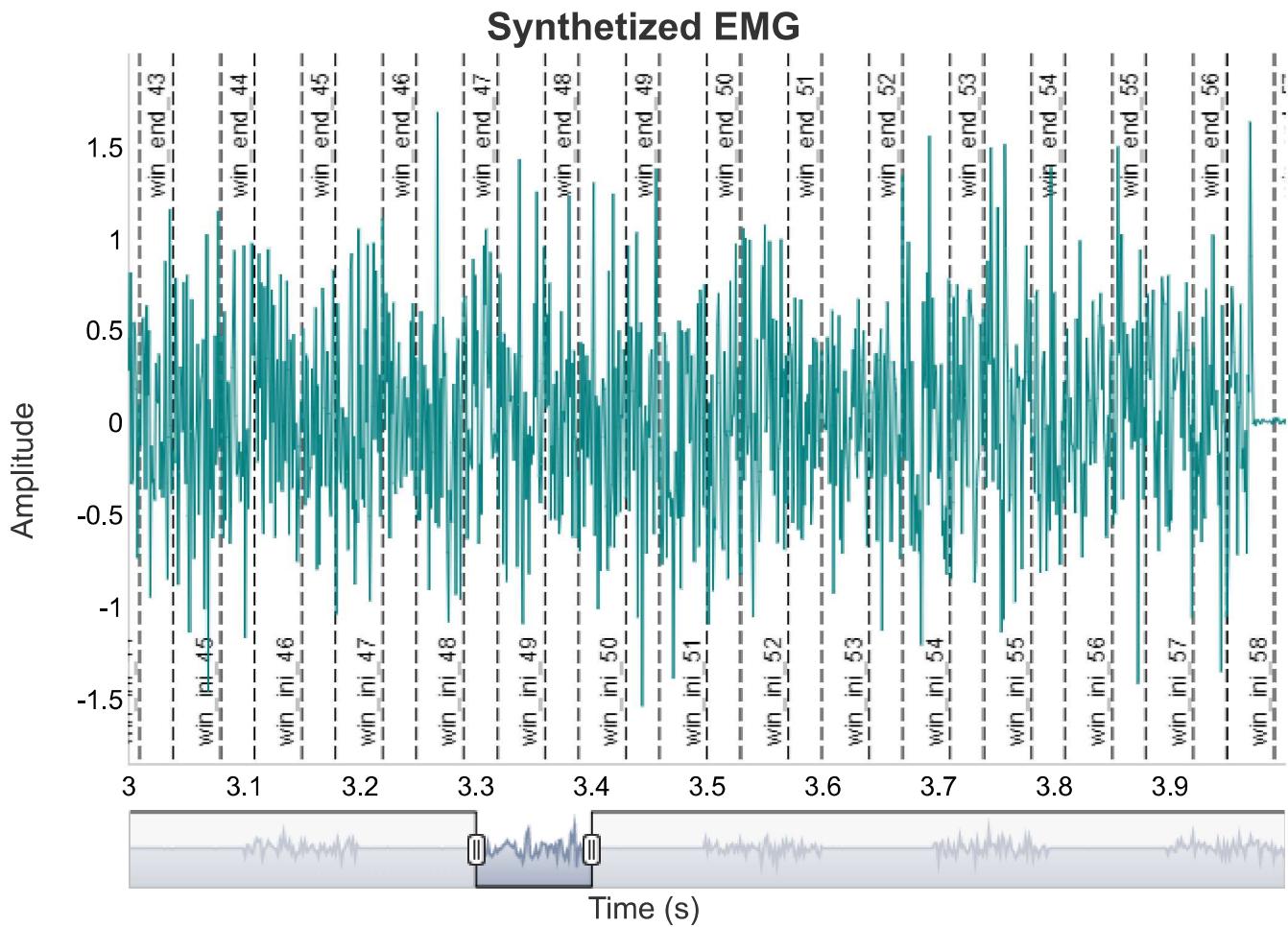
```



```
emg <- emg$values
```

```
n <- length(emg)
win.dur <- 100/1000 # window dur. (s)
win.size <- win.dur * fs # window dur. (samples)
win.overlap_per <- 0.3 # window ovelarp (percentage)
win.overlap <- win.overlap_per*win.size # window overlap (samples)
win.number <- round(n/win.size/win.overlap_per) # number of windows
win.idx_ini <- seq(from=1,to=n, by=win.size-win.overlap)
win.idx_end <- win.idx_ini+(win.size-1)
win.idx_end[win.idx_end>n] = n

# windowing
data.frame(t=t, y=emg) %>%
  dygraph(main='Synthesized EMG') %>%
  dyEvent(x=(win.idx_end-1)*dt, label=sprintf('win_end_%d', 1:length(win.idx_end)), labelLoc='top') %>%
  dyEvent(x=(win.idx_ini-1)*dt, label=sprintf('win_ini_%d', 1:length(win.idx_ini)), labelLoc='bottom') %>%
  dyAxis('y', label='Amplitude') %>%
  dyAxis('x', label='Time (s)') %>%
  dyOptions(axisLineWidth=0.1, fillGraph=FALSE, drawGrid=FALSE) %>%
  dyRangeSelector(dateWindow = c(3,4))
```



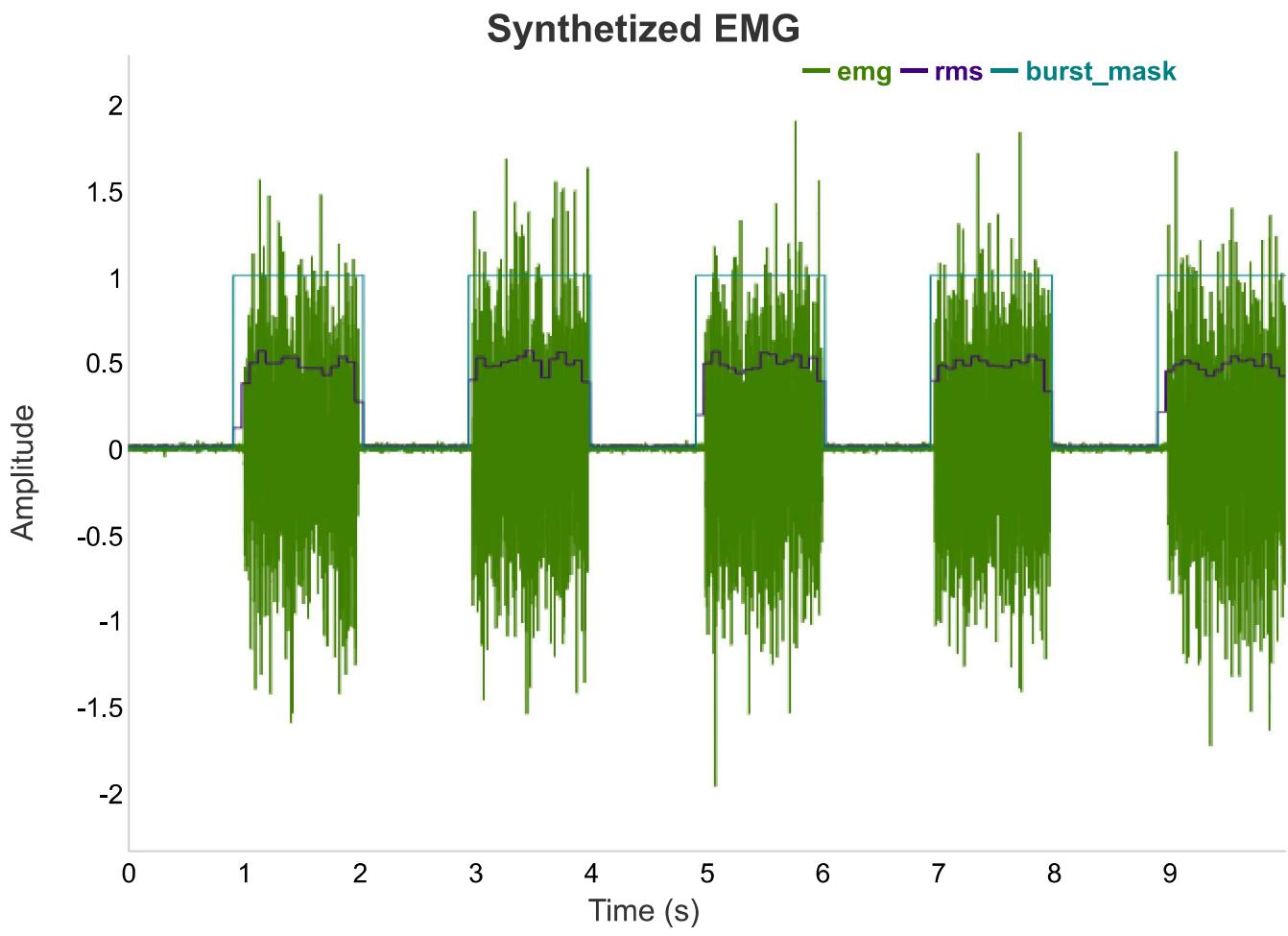
Uma vez o sinal estando devidamente janelado, agora será feita a detecção dos *bursts* (o limiar para o valor RMS foi empiricamente definido como 0.1):

```

rms <- function(x){return(sqrt(mean(x^2)))}
bursts_ <- rep(FALSE, n)
rms_ <- rep(0,n)
threshold <- 0.1
for(i in c(1:length(win.idx_ini)))
{
  win_rms <- rms(emg[win.idx_ini[i]:win.idx_end[i]])
  bursts_[win.idx_ini[i]:win.idx_end[i]] <- win_rms>threshold
  rms_[win.idx_ini[i]:win.idx_end[i]] <- win_rms
}

data.frame(t=t, emg=emg, rms=rms_, burst_mask=bursts_) %>%
  dygraph(main='Synthesized EMG') %>%
  dyAxis('y', label='Amplitude') %>%
  dyAxis('x', label='Time (s') %>%
  dyOptions(axisLineWidth=0.1, fillGraph=FALSE, drawGrid=FALSE)

```



Assim permitindo detectar os *bursts* no sinal EMG.