Lab3

Bet Bardají, Jens Lagemann

2023-01-12

Exercise 1

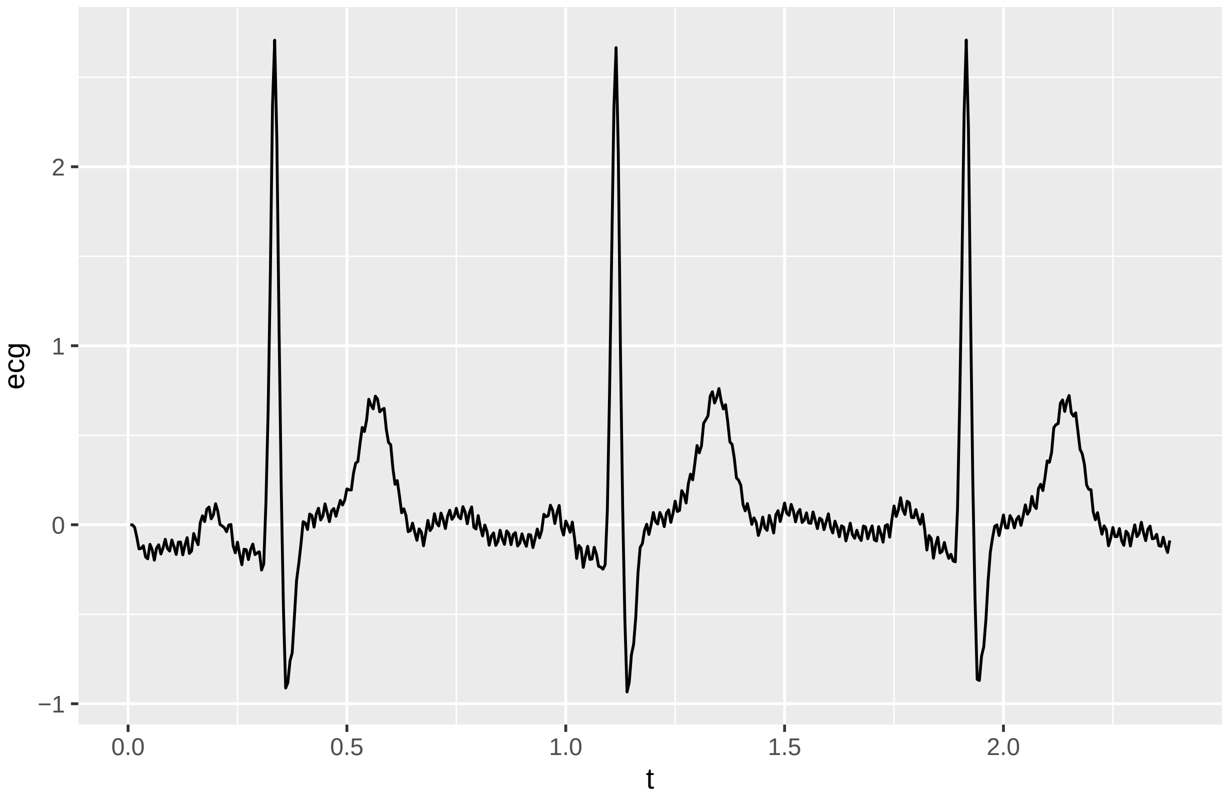
Loading Data

```
ecg_data <- read_table("./ecg2x60.dat", col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
fs <- 200 # sample frequency
t <- (1:length(ecg_data))/ 200 # time steps in seconds
ecg2 <- data.frame(t, t(ecg_data))
colnames(ecg2) <- c("t", "ecg")
ggplot(ecg2, aes(x = t, y = ecg)) + geom_line() + ggtitle("Original ECG data")
```

Original ECG data



Designing the notch filter

We will use the transfer function in the complex-domain of the filter, given by H(z) Using these equations we find the position of all zeros and poles required

$$H(z) = G(z - e^{jw_0})(z - e^{-jw_0})(z - R^{jw_0})(z^r e^{-jw_0}) = G\frac{1 - 2\cos(w_0)z^{-1} + z^{-2}}{1 - 2r\cos(w_0)z^{-1} + r^2 z^{-2}}$$

$=> H(z) = \frac{(z-\exp(jw_0))\cdot(z-\exp(-jw_0))}{z^2}$ The denominator must have the order greater or equal to the numerator, otherwise the system cannot be physically realized. $H(z) = \frac{(z-\exp(j1.885))\cdot(z-\exp(-j1.885))}{z^2} = \frac{(z^2+0.6181z+1)}{z^2} = \frac{1+0.6181z^{-1}}{z^{-2}}$ And we know that $w_0 = \pm\frac{f}{f_s}\cdot(2\pi)$ $K = \frac{(1-2\cos(\theta+r))}{(2-2\cos\theta)}$

$R_1 = 1$ -> because it has to be on the unit circle.

$F_s = 200$ Sampling frequency

$F_c = 60$ Frequency to be eliminated

$\theta = \frac{F_c}{F_s}\cdot 2\pi$

Calculation of the parameters

$W_0 - > \theta = 2\pi\cdot\frac{60}{200} = \pm 1.855\frac{rad}{\text{samples}}0 \pm 108$ř

Zero locations:

$Z_1 = -0.3090 + j\cdot 0.9519$

$Z_2 = -0.3090 - j\cdot 0.9519$

$K = \frac{1-2r_1\cos\theta+r_1^2}{2-2\cos\theta}$

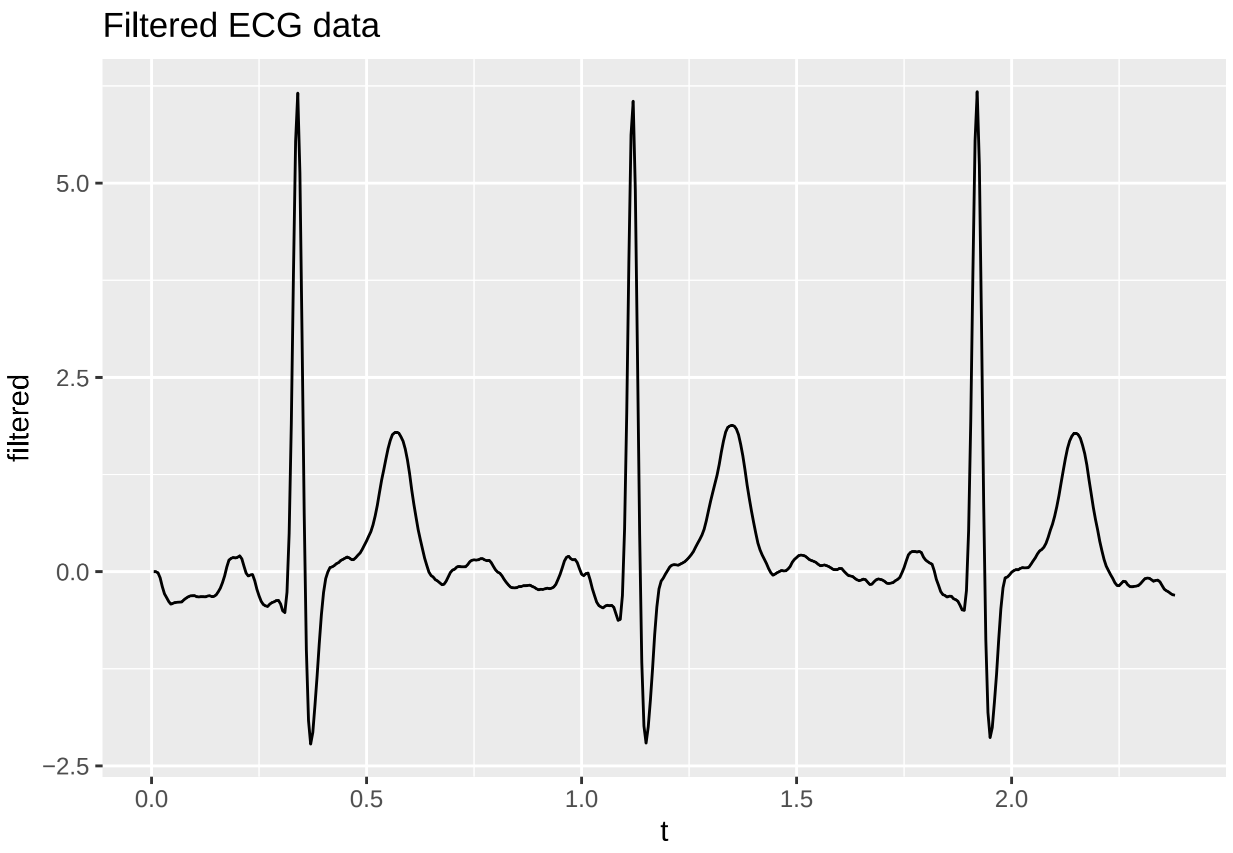
From the $H(z)$ numerator we can extract the zeros: $K \cdot z^2 - 2z\cos\theta + r_1^2$

And from the denominator the poles: $z^2 - (2r_1 z\cos\theta + r_1^2$

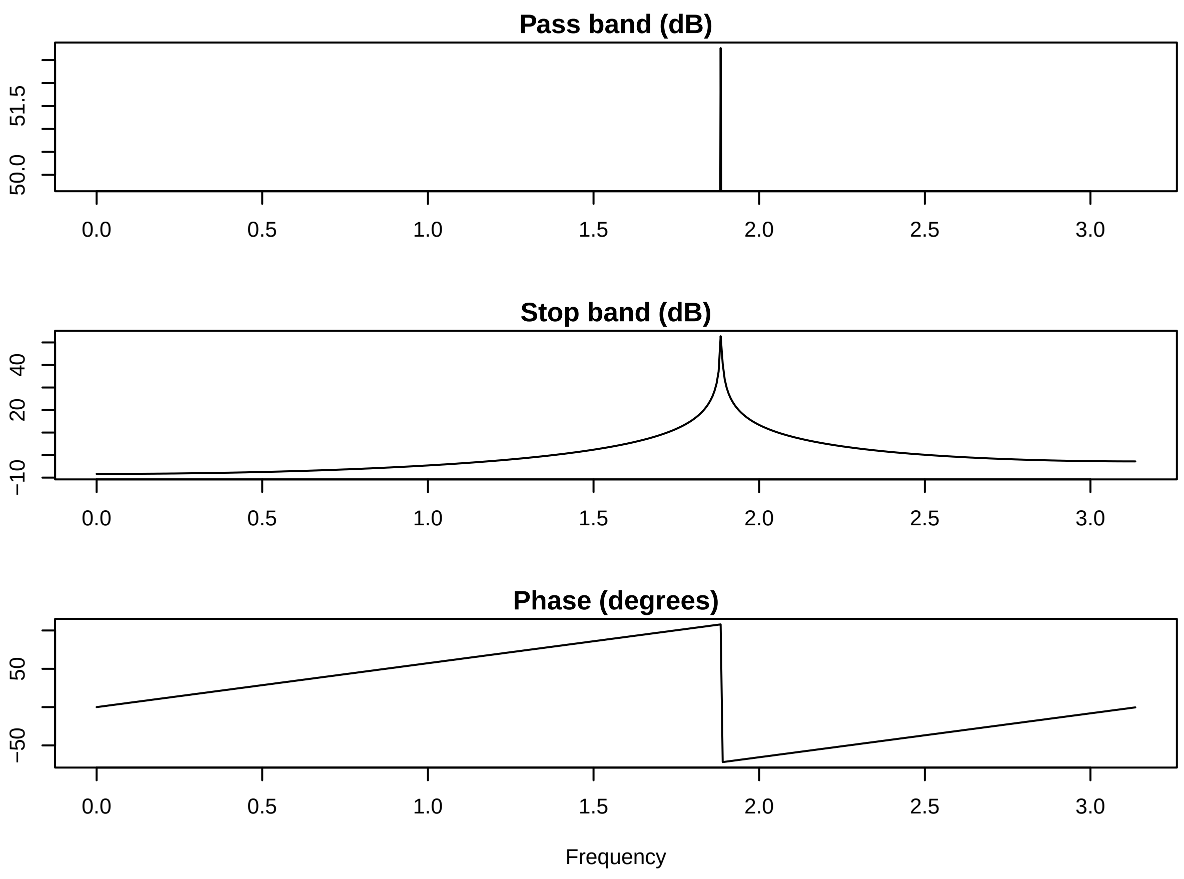
Replacing z_1 and z_2 in $H(z) = 1 + 0.618z^{-1} + \frac{z^{-2}}{2.618}$ we get

```
a <- c(1, 0.618, 1) # Denominator coefficients
b <- c(1,0,0) # Numerator coefficients
ecg2$filtered <- filter(a, b, ecg_data)
ggplot(ecg2, aes(x = t, y = filtered)) + geom_line() + ggtitle("Filtered ECG data")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



```
freqz(b, a)
```

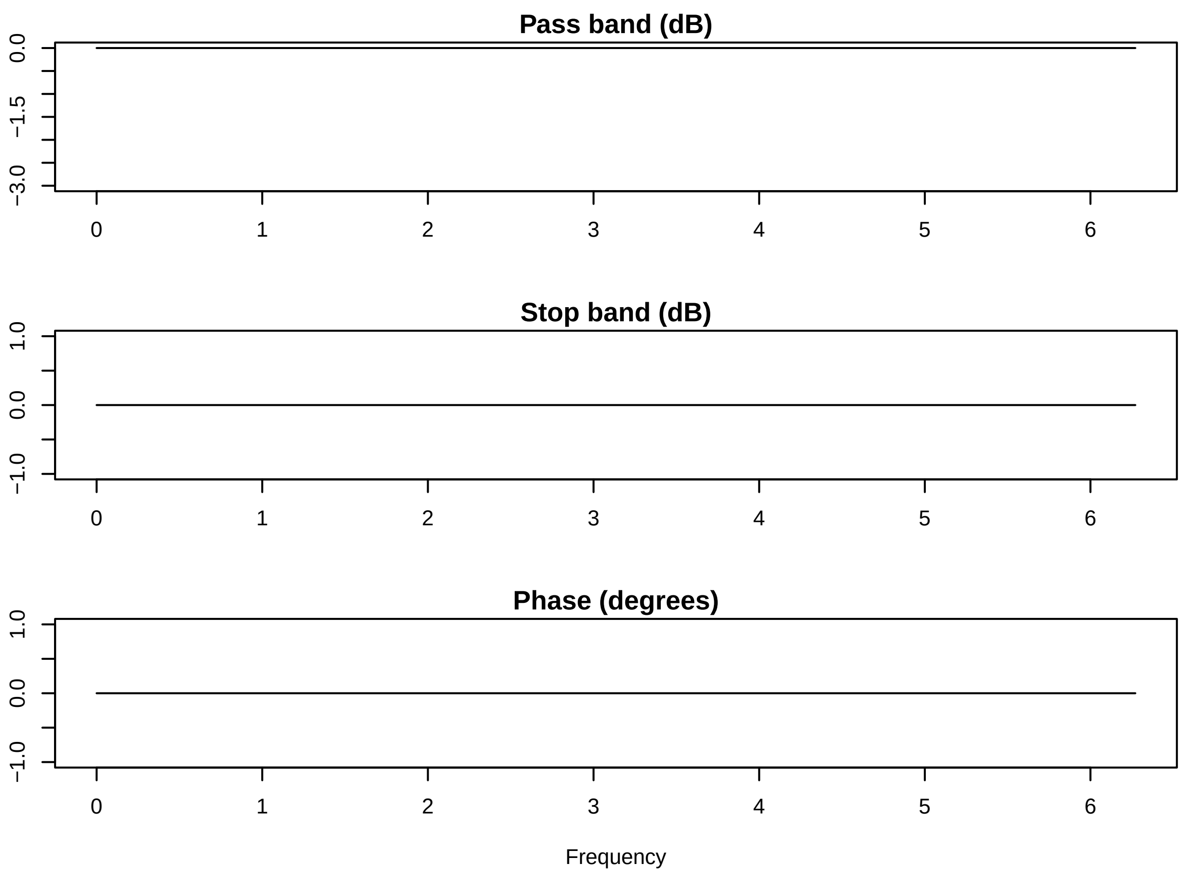


Graphing the magnitude and phase spectrum pf filter for pole radius 0.88 and 0.99

```
w0 <- 1.885
# Now add the poles
r <- c(0.8, 0.9, 0.99)
a <- c()
for (k in 1:length(r)){
  j <- complex(imaginary = 1)
  x <- c(r[k] * exp(j * w0), r[k] * exp(-1*j*w0))
  a[k] <- poly(x)
}
```

```
## Warning in a[k] <- poly(x): number of items to replace is not a multiple of
## replacement length

## Warning in a[k] <- poly(x): number of items to replace is not a multiple of
## replacement length

## Warning in a[k] <- poly(x): number of items to replace is not a multiple of
## replacement length
```

```
freqz(b, a[2])
```

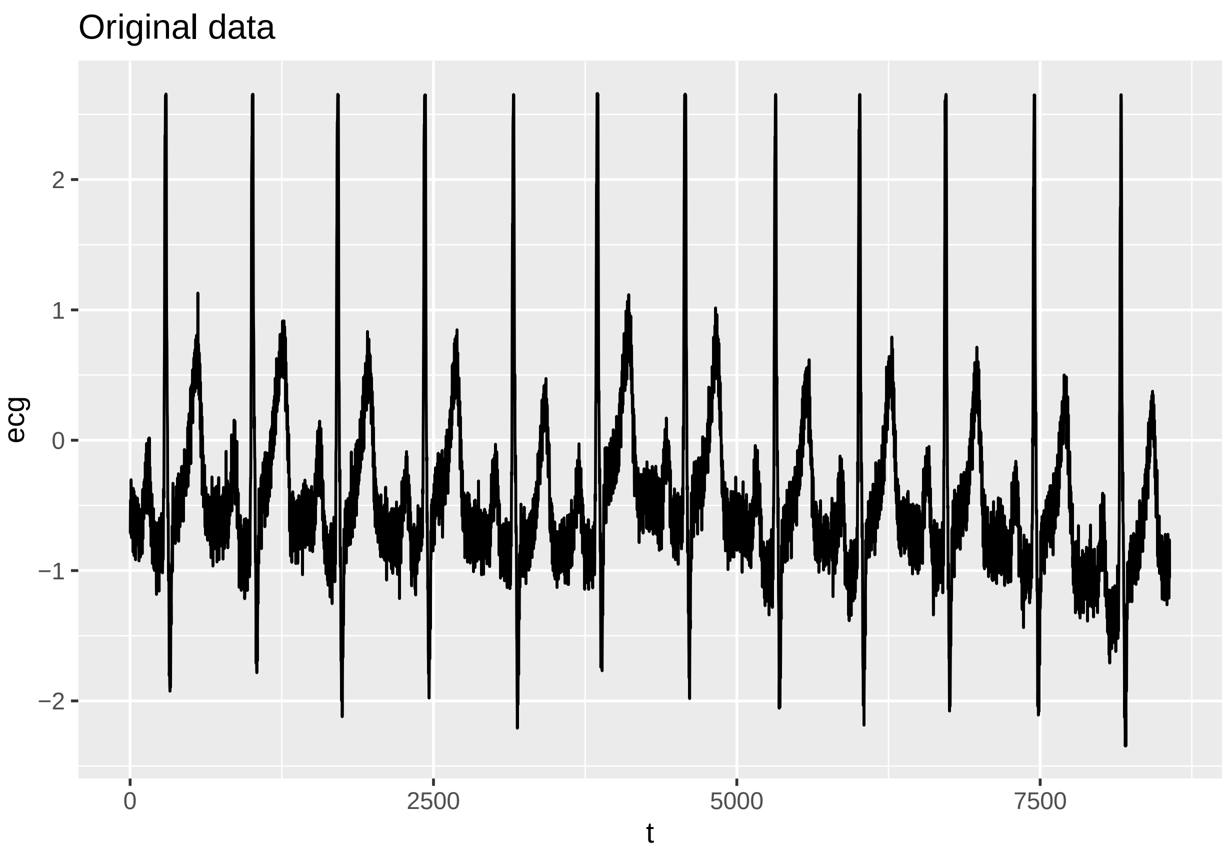


Exercise 2

```
hf_ecg <- read_table("./ecg_hfn.dat", col_names = FALSE)
```

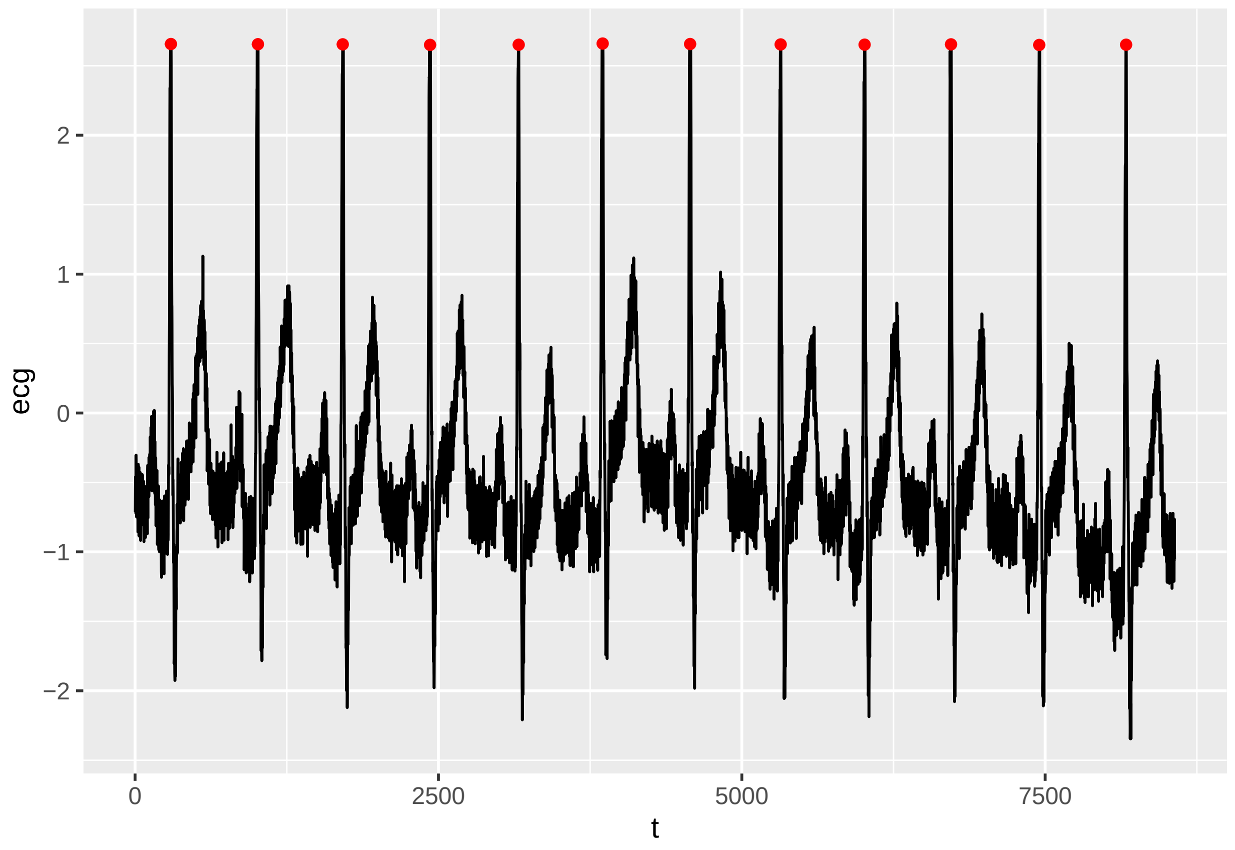
```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double()
## )
```

```
t <- (1:length(hf_ecg$X1))
hf_data <- data.frame(t, hf_ecg)
colnames(hf_data) <- c("t", "ecg")
ggplot(hf_data, aes(x=t, y=ecg)) + geom_line() +ggtitle("Original data")
```



In the data it is visible that the beats are peaks that exceed the value of 2. Therefore we can isolate every peak with that minimum value, by testing it was found that the peaks are usually only a few samples long, so as to not double identify the same beat twice, a minimum distance between peaks of 30 samples is set. Now we can determine the average distance between the beats and split the time series accordingly. Though noticeably, we have a shortened frame just before the first peak.

```
hf_num <- as.numeric(as.matrix(hf_ecg))
# Find the peaks
peaks <- findpeaks(hf_num, minpeakheight = 2, minpeakdistance = 30)
peaks <- data.frame(peaks)
colnames(peaks) <- c("ecg", "t", "start", "end")
# Mark the identified peaks in a plot
ggplot(hf_data, aes(x=t, y=ecg)) +
  geom_line() +
  geom_point(data = peaks, colour="red")
```



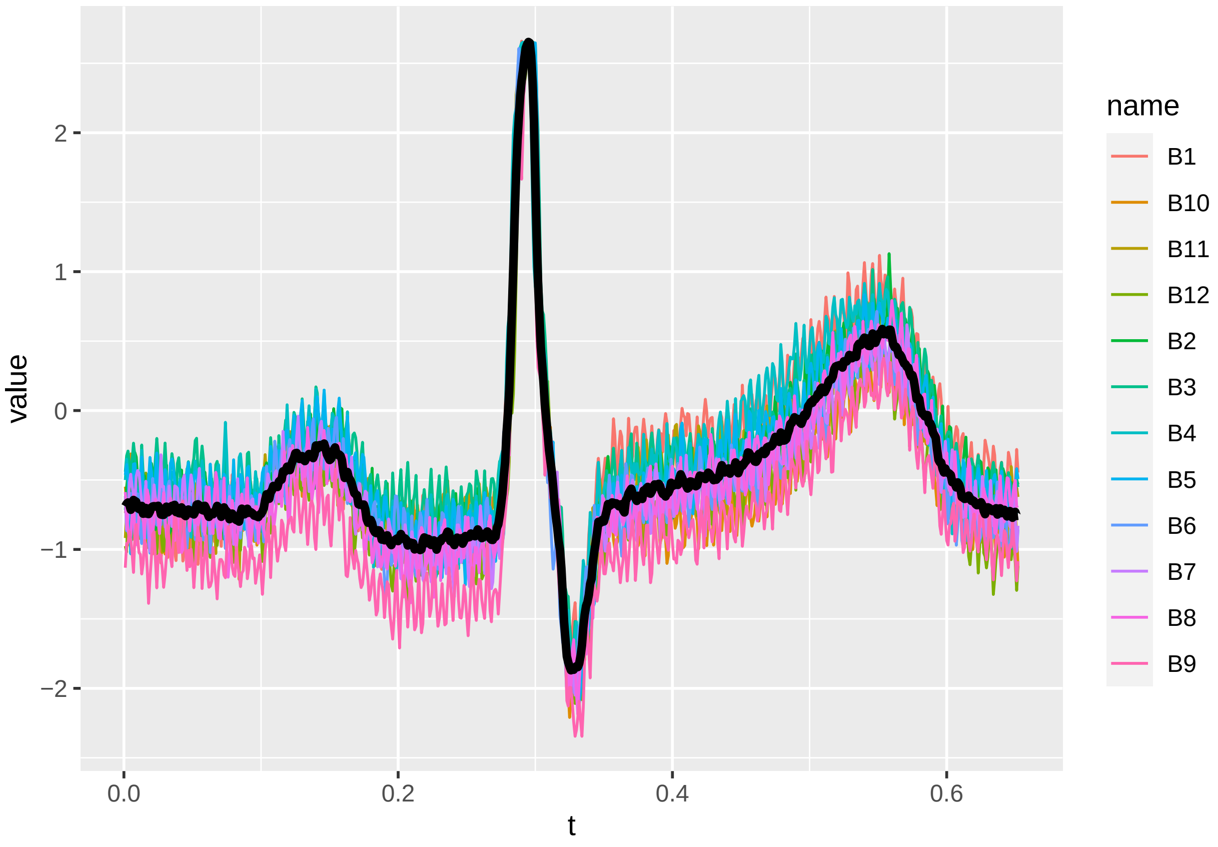
```
# position of peaks sorted by t
sorted <- sort(peaks$t)
# average distance between peaks
print(mean(sorted[2:12] - sorted[1:11]))
```

```
## [1] 715.6364
```

```
# length of beat, considering the shortened first beat
b <- data.frame(c(1:652))
# extract the sample window around each peak
for (i in peaks$t) {
  print(i)
  start <- i - 294
  end <- i + 357
  beat_seq <- hf_num[start:end]
  b <- cbind(b, beat_seq)
}
```

```
## [1] 3853
## [1] 295
## [1] 4574
## [1] 1012
## [1] 1711
## [1] 6724
## [1] 5320
## [1] 6012
## [1] 8167
## [1] 3161
## [1] 2431
## [1] 7451
```

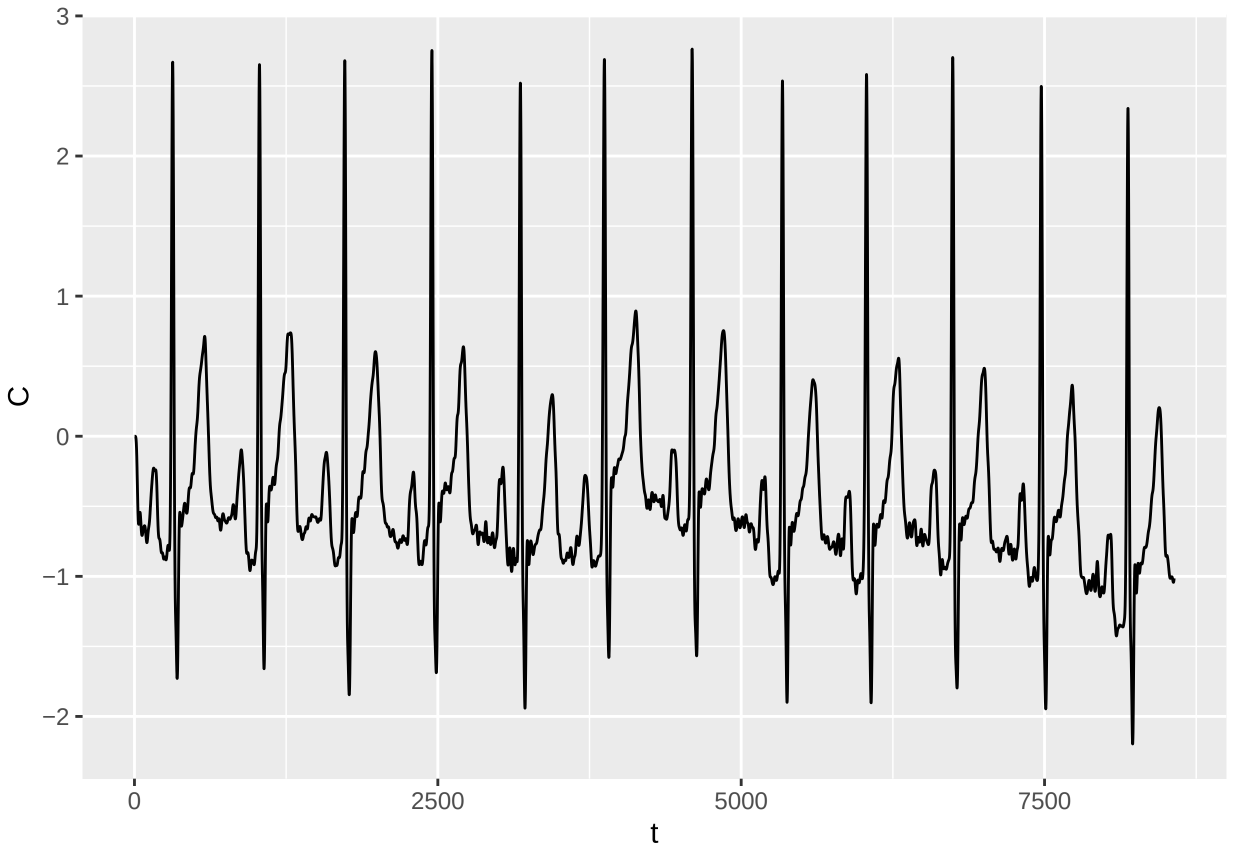
```
colnames(b) <- c("dummy", "B1", "B2", "B3", "B4", "B5", "B6", "B7", "B8", "B9", "B10", "B11", "B12")
b <- subset(b, select = -c(dummy))
# Calculate the synchronized average
b$average <- rowMeans(b)
b$t <- (1:652)/1000
data_long <- pivot_longer(b, c("B1", "B2", "B3", "B4", "B5", "B6", "B7", "B8", "B9", "B10", "B11", "B12
ggplot(data_long, aes(x=t, y=value, color=name)) + geom_line() + geom_line(aes(y=average), color="black
```



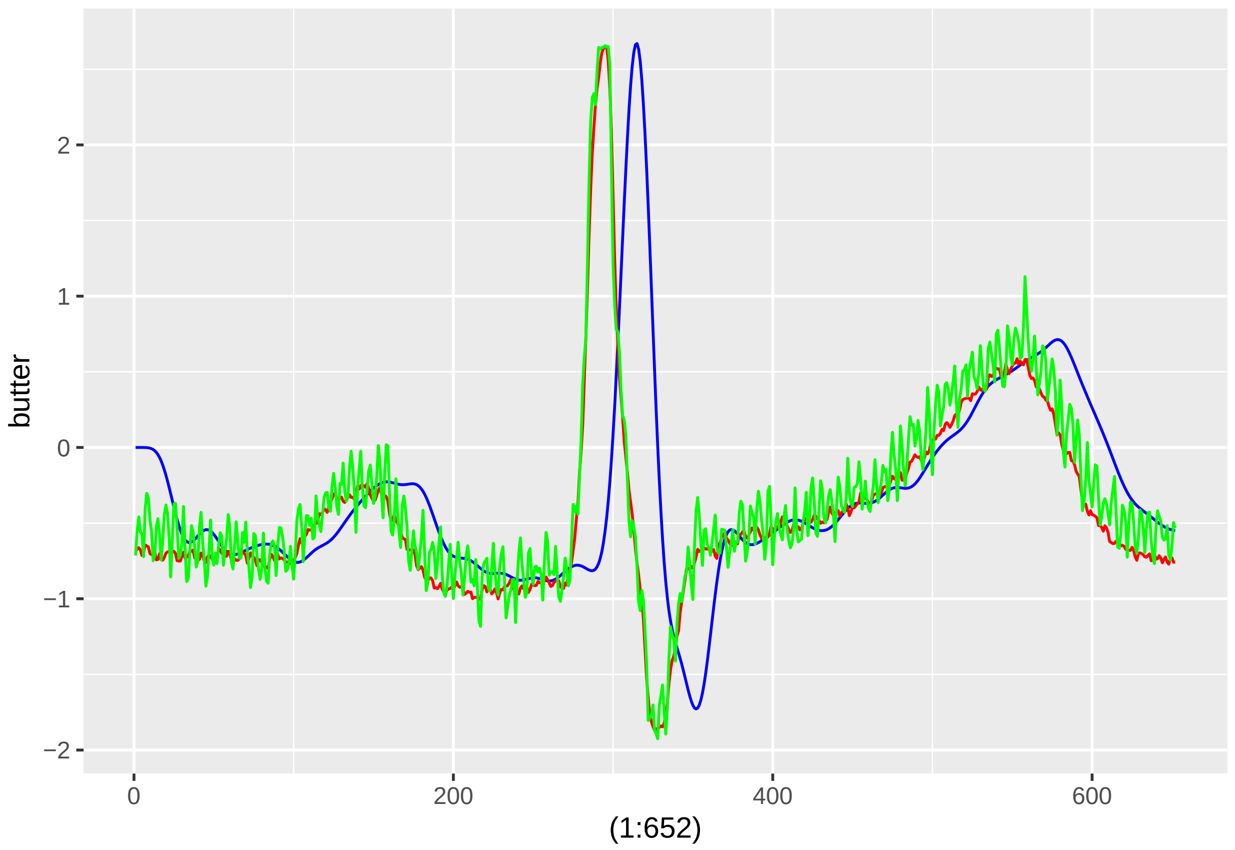
## Exercise 3 ### Loading data

```
sample_rate <- 1000
ny_f <- sample_rate/2
butter_a <- butter(2, 10/ny_f, type="low", plane = "z")
butter_b <- butter(8, 20/ny_f, type="low", plane = "z")
butter_c <- butter(8, 40/ny_f, type="low", plane = "z")
butter_d <- butter(8, 70/ny_f, type="low", plane = "z")

filtered_data <- data.frame(c(1:length(hf_ecg$X1)))
colnames(filtered_data) <- c("t")
filtered_data$A <- filter(butter_a, hf_ecg$X1)
filtered_data$B <- filter(butter_b, hf_ecg$X1)
filtered_data$C <- filter(butter_c, hf_ecg$X1)
filtered_data$D <- filter(butter_d, hf_ecg$X1)

ggplot(filtered_data, aes(x=t)) +
  geom_line(aes(y=C))
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



```
comp <- data.frame(filtered_data$C[1:652], b$average)
colnames(comp) <- c("butter", "average")
comp$original <- hf_num[1:652]
ggplot(comp, aes(x = (1:652), y=butter)) +
  geom_line(colour="blue") +
  geom_line(aes(y=average), color="red")+
  geom_line(aes(y=original), color="green")
```



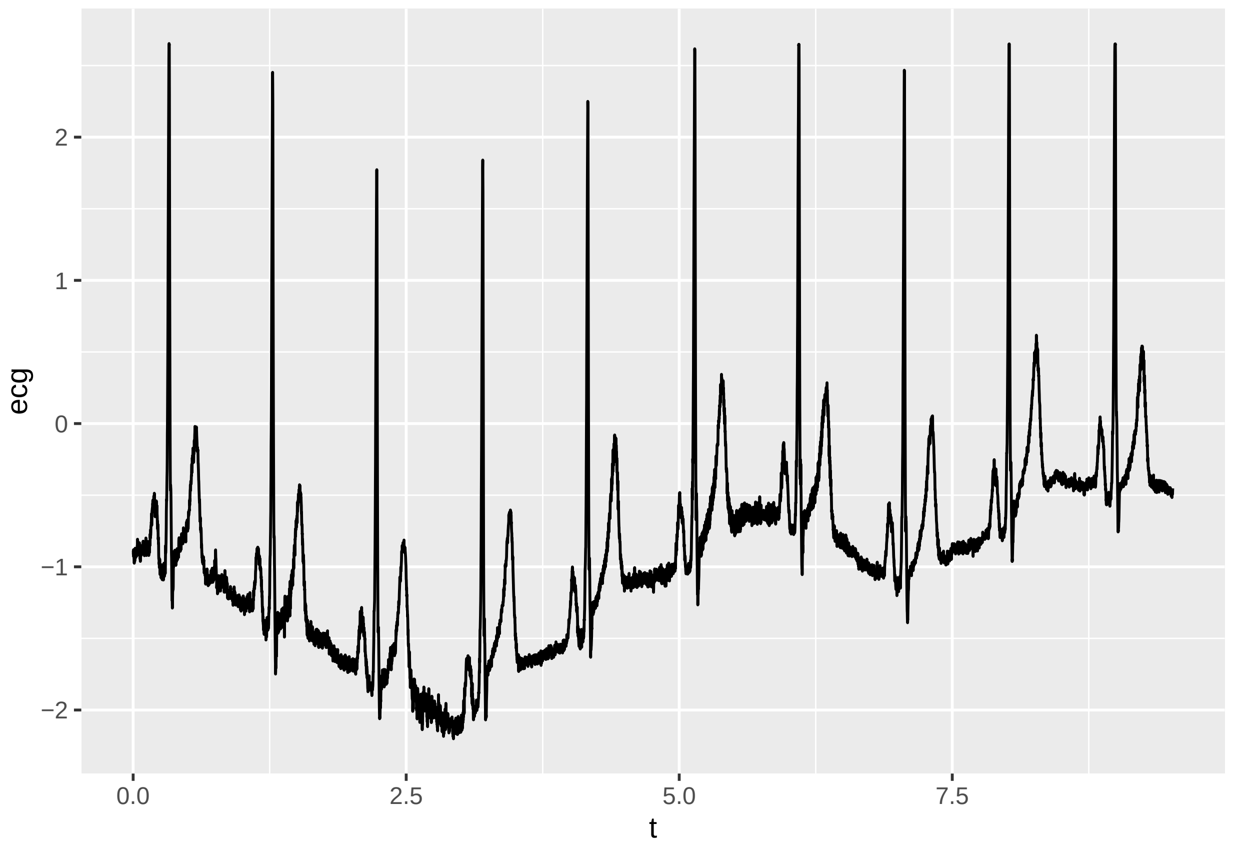
The butterworth filter shifted the peak of the original data to the right. But overall created a smoother line compared to the synchronized average.

Exercise 4

```
ecg_lf <- read_table("./ecg_lfn.dat", col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double()
## )
```

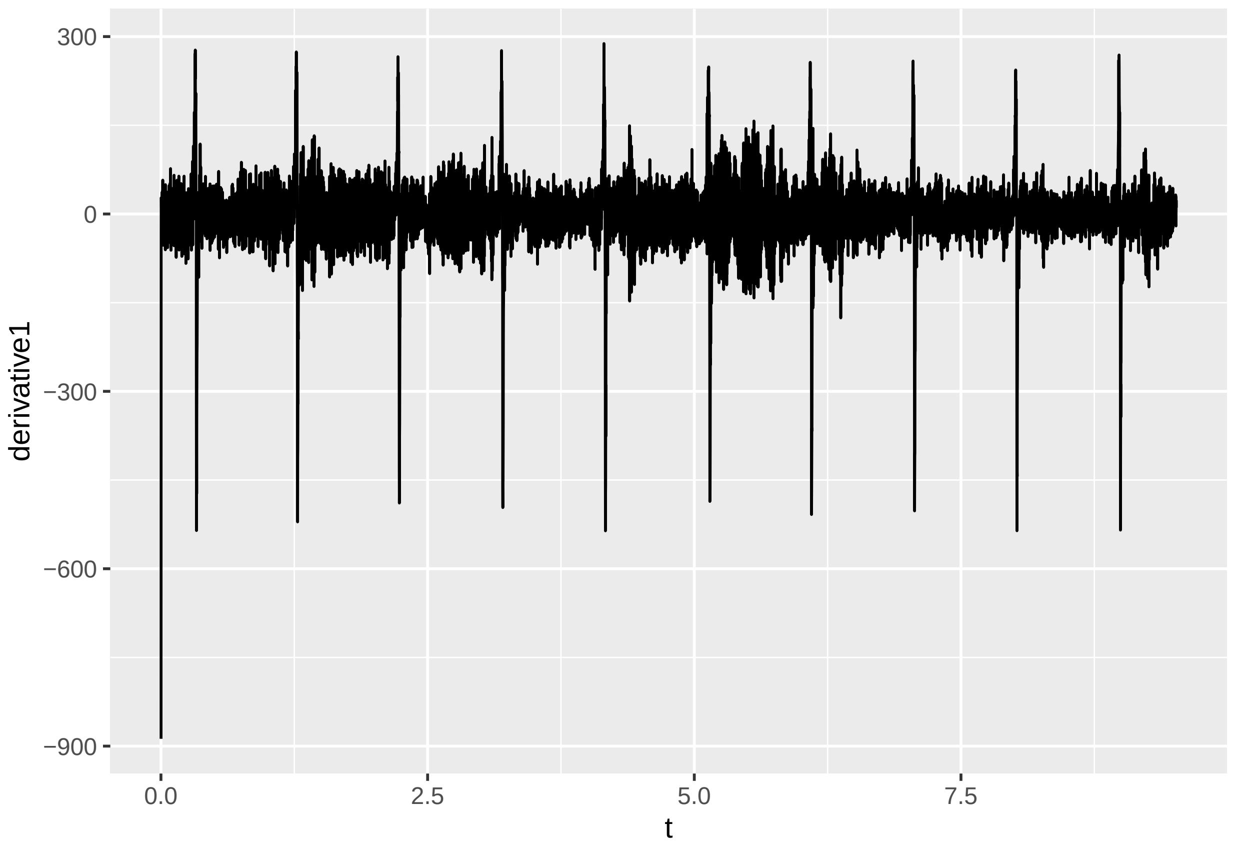
```
fs <- 1000
t <- (1:length(ecg_lf$X1))/fs
lf_data <- data.frame(t, ecg_lf$X1)
colnames(lf_data) <- c("t", "ecg")
ggplot(lf_data, aes(x=t, y=ecg)) + geom_line()
```



A

```
z1 <- c(1, -1)
p1 <- 1
Sf1 <- filter(fs*z1,p1, lf_data$ecg)
lf_data$derivative1 <- Sf1
ggplot(lf_data, aes(x=t, y=derivative1)) + geom_line()
```

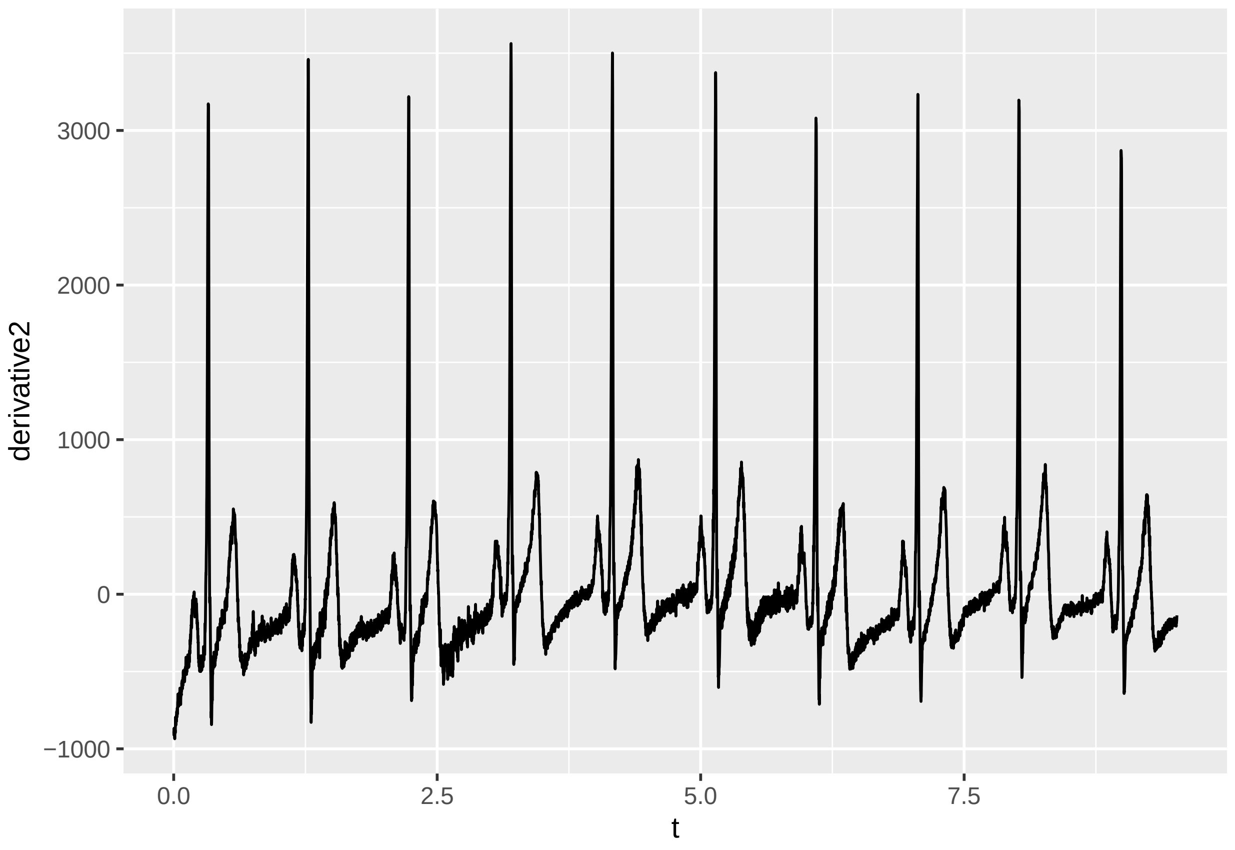
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



B

```
z2 <- c(1, -1)
p2 <- c(1, -0.995)
Sf2 <- filter(fs*z2,p2, lf_data$ecg)
lf_data$derivative2 <- Sf2
ggplot(lf_data, aes(x=t, y=derivative2)) + geom_line()
```

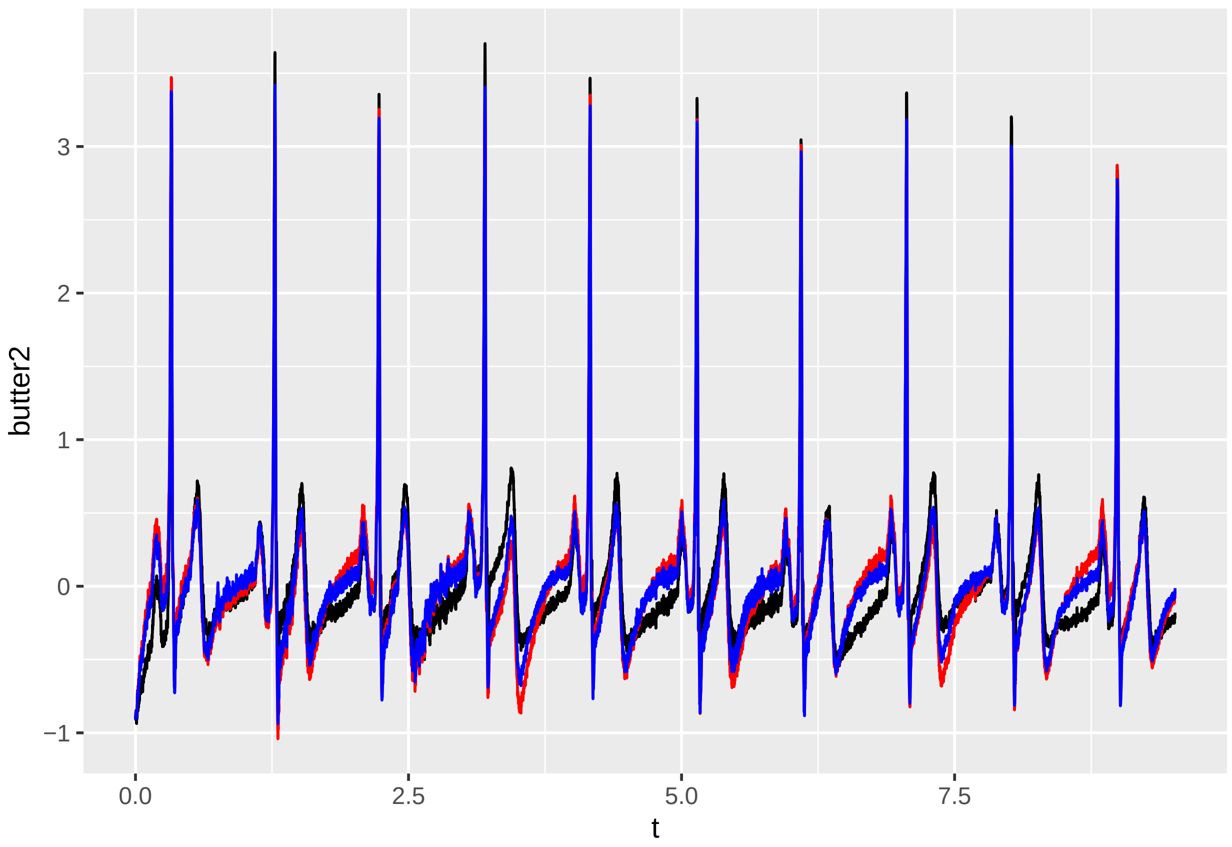
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



Exercise 5

```
fs <- 1000
n <- 2
wnn <- 0.5
wn <- wnn/(fs/2)
bt <- butter(n,wn,"high")
lf_data$butter2 <- filter(bt, lf_data$ecg)

n <- 5
wnn <- 0.5
wn <- wnn/(fs/2)
bt <- butter(n,wn,"high")
lf_data$butter5 <- filter(bt, lf_data$ecg)

n <- 2
wnn <- 1
wn <- wnn/(fs/2)
bt <- butter(n,wn,"high")
lf_data$butter10 <- filter(bt, lf_data$ecg)

ggplot(lf_data, aes(x=t, y=butter2)) + geom_line() +
  geom_line(aes(y=butter5), color="red") +
  geom_line(aes(y=butter10), color="blue")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```


butter2
3
2
1
0
−1
0.0
2.5
5.0
7.5
t