

ECG_Longitudinales

German Gallego Garcia

Ve a la página: <https://www.physionet.org/content/ephnogram/1.0.0/> y descarga un ECG. En mi caso es el Individuo / Paciente número 12.

1. Investiga qué función de R puedes usar para importar el ECG
2. Busca alguna librería que calcule los picos R
3. Calcula los intervalos RR
4. Usa algún método estadístico para detectar valores anómalos en la serie

1) Importar el ECG

El primero paso es la lectura de los datos. Tenemos 2 archivos **ECGPGC0012.hea** y **ECGPGC0012.dat**. En el archivo con la extensión **hea** tenemos información de como está organizado el archivo con la extensión **dat**

Archivo HEA

```
rm(list=ls())
pacman::p_load(tidyverse, pracma, ggplot2)
hea_file <- "Data/ECGPGC0012.hea"
hea_content <- readLines(hea_file)
hea_content
```



```
## [1] "ECGPGC0012 2 8000 14400000"
## [2] "ECGPGC0012.dat 16 49488.8024(-16698)/mV 0 0 -20613 -20293 0 ECG"
## [3] "ECGPGC0012.dat 16 33090.146(-16)/mV 0 0 646 9447 0 PCG"
## [4] "# The simultaneous electrocardiogram and phonocardiogram database"
```

El archivo .hea contiene metadatos del registro, como:

- Nombre del registro: ECGPGC0012
- Número de señales: 2 (ECG y PCG)
- Frecuencia de muestreo: 8000 Hz
- Duración total: 14400000 muestras
- Detalles de cada canal:

- ECGPCG0012.dat 16 49488.8024(-16698)/mV 0 0 -20613 -20293 0 ECG (Canal 1 - ECG)
- ECGPCG0012.dat 16 33090.146(-16)/mV 0 0 646 9447 0 PCG (Canal 2 - PCG)

Los datos que nos interesean son los siguientes

- Número de señales: 2 (ECG y PCG)
- Frecuencia de muestreo: 8000 Hz
- Duración total: 14400000 muestras
- Factor de conversión (Ganancias): 49488.8024(-16698)/mV

```
n_sample<-14400000 # Muestra
freq<-8000 # Frecuencia (cada momento que se toma una medida)
factor_conver <- 49488.8024 # las ganancias
num_signals <- 2 #el número de señales en el archivo (ECG y PCG)
```

Archivo DAT

Ahora gracias a la información del archivo **hea** podremos hacer la lectura del archivo **dat** mediante la función **readBin** (dado que el archivo .dat es de origen binario en este caso).

```
dat_file <- "Data/ECGPCG0012.dat"
raw_data <- readBin(dat_file,
                      what = "integer",
                      size = 2,
                      n=n_sample*num_signals,
                      signed = TRUE,
                      endian = "little")
```

En el archivo vendrán los datos tanto de ECG como de PCG pero solo nos interesan los datos de ECG (son los primeros)

```
data_matrix <- matrix(raw_data, ncol = num_signals, byrow = TRUE)
ECG<-data_matrix[,1]
```

Ahora aplicaremos la escala de conversión entre las unidades de la señal digitalizada y las unidades físicas reales, en este caso en milivoltios (mV). Este dato nos aparece en el archivo **hea**

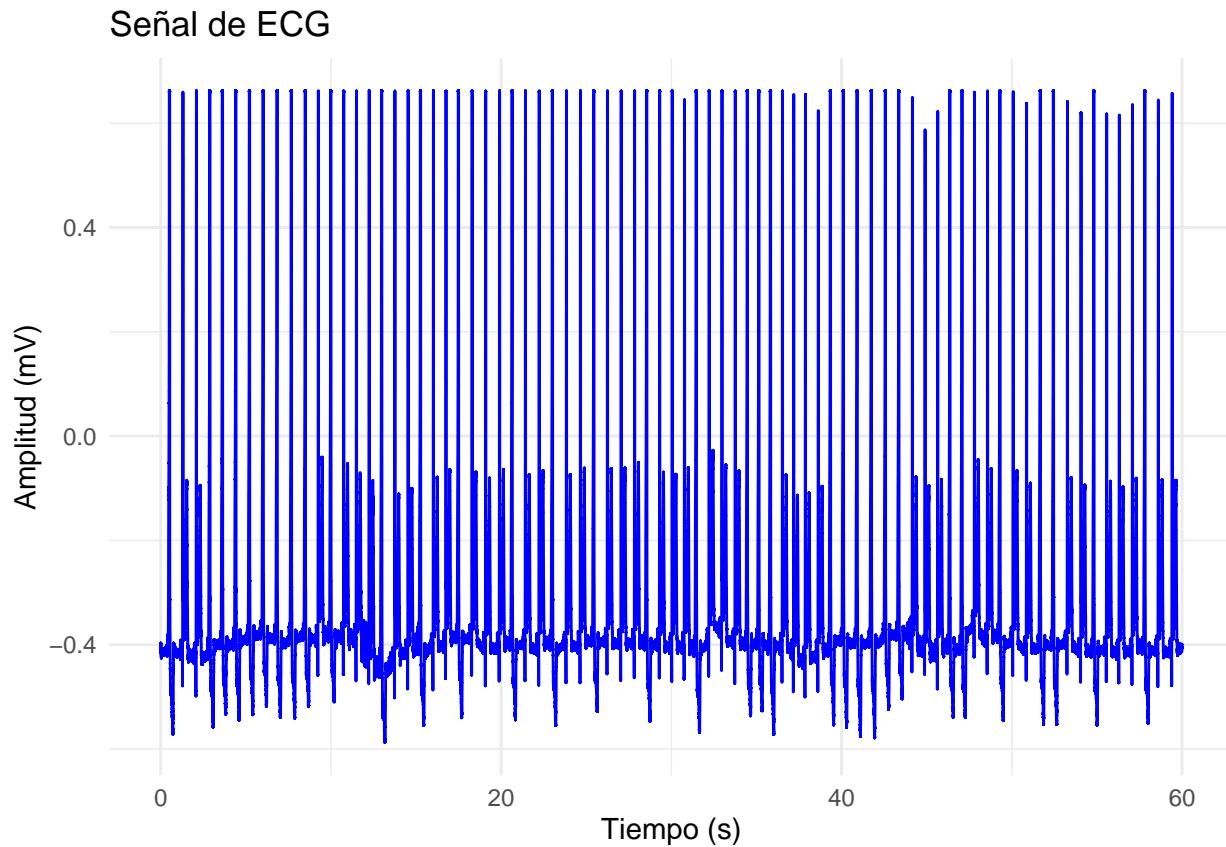
```
ECG_mV <- ECG/factor_conver
```

Graficar el ECG

Dado que son muchas observaciones, mostraremos hasta un minuto (correspondientes a los primeros).

```
ECG_data <- data.frame(Time = seq(0, length(ECG_mV) - 1) / freq, # Tiempo en segundos
                        ECG = ECG_mV)
muestra<-length(ECG_data[,1])*0.1
ECG_data_sampled<-ECG_data[ECG_data$Time <= 60, ]
```

```
ggplot(ECG_data_sampled, aes(x = Time, y = ECG)) +
  geom_line(color = "blue") +
  labs(title = "Señal de ECG", x = "Tiempo (s)", y = "Amplitud (mV)") +
  theme_minimal()
```



2) Busca alguna librería que calcule los picos R

Se ha encontrado la librería **pracma** para detectar los picos R del ElectroCardioGramma mediante la función **findpeaks**.

De todos los argumentos que tiene la función **findpeaks** utilizaremos los argumentos *mindpeakheight* y *minpeakdistance*. En *mindpeakheight* utilizaremos un valor relacionado con el pico más alto y en *minpeakdistance* una distancia mínima entre los picos equivalente a $0.6 * \text{freq}$ (frecuencia de muestreo)

```
# Definir un umbral basado en la señal ECG
umbral <- max(ECG_mV) * 0.5 # La mitad del pico más alto

# Detectar picos R en la señal ECG
picos <- findpeaks(ECG_mV,
                     minpeakheight = umbral,
                     minpeakdistance = freq * 0.6)
```

Ahora visualizaremos los picos conjuntamente los datos. Observaremos igual que antes, los picos que pueden haber pasado en 1 minuto (debido a la cantidad de datos que hay).

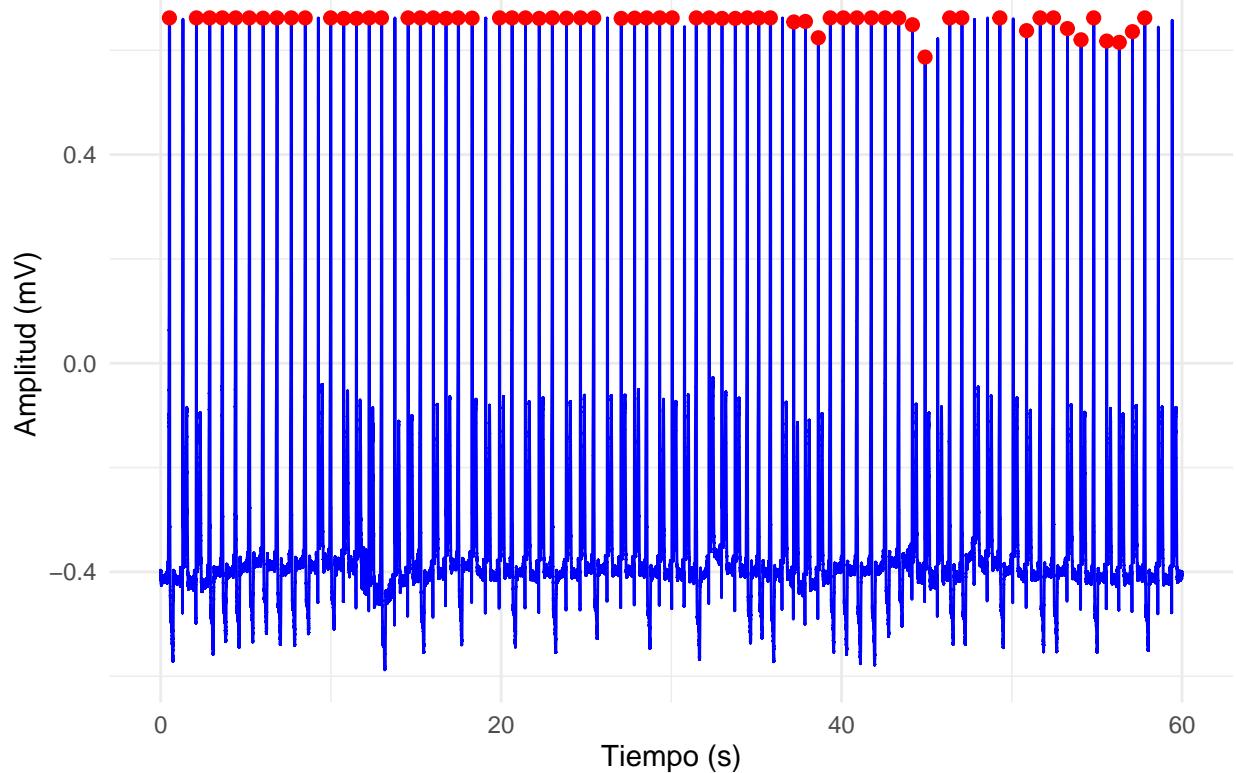
```

# Extraer tiempos de los picos
tiempo_picos <- ECG_data$Time[picos[, 2]]
amplitud_picos <- picos[, 1]
# Ordenarlos
orden_picos <- order(tiempo_picos)
tiempo_picos_ordenados <- tiempo_picos[orden_picos]
amplitud_picos_ordenados <- amplitud_picos[orden_picos]
# Crear un data frame con los picos detectados
picos_df <- data.frame(Time = tiempo_picos_ordenados,
                        Amplitud = amplitud_picos_ordenados)
picos_df_muestra<-picos_df%>%
  filter(Time<=60)

# Graficar la señal ECG con los picos R resaltados
ggplot(ECG_data_sampled, aes(x = Time, y = ECG)) +
  geom_line(color = "blue") + # Señal ECG
  geom_point(data = picos_df_muestra, aes(x = Time, y = Amplitud), color = "red", size = 2) + # Picos R
  labs(title = "Señal ECG con detección de picos R", x = "Tiempo (s)", y = "Amplitud (mV)") +
  theme_minimal()

```

Señal ECG con detección de picos R



3) Calcula los intervalos RR

Ahora calcularemos los intervalos entre cada pico R

```
# Calcular los intervalos RR (diferencias entre los tiempos de los picos R)
intervalos_RR <- diff(tiempo_picos_ordenados)
```

Calcularemos algunos estadísticos de estos intervalos

```
# Estadísticos
rr_mean <- mean(intervalos_RR)
rr_sd <- sd(intervalos_RR)
rr_min <- min(intervalos_RR)
rr_max <- max(intervalos_RR)

cat("Intervalo RR promedio:", rr_mean, "s\n")
```

Intervalo RR promedio: 0.9569542 s

```
cat("Desviación estándar:", rr_sd, "s\n")
```

Desviación estándar: 0.4258507 s

```
cat("Intervalo RR mínimo:", rr_min, "s\n")
```

Intervalo RR mínimo: 0.63475 s

```
cat("Intervalo RR máximo:", rr_max, "s\n")
```

Intervalo RR máximo: 3.494125 s

A continuación calcularemos un IC para la media de los intervalos de confianza:

```
n <- length(intervalos_RR)
z <- qnorm(0.975)
IC_inf <- rr_mean - z * (rr_sd / sqrt(n))
IC_sup <- rr_mean + z * (rr_sd / sqrt(n))

paste0("Intervalo de confianza del 95% para la media del intervalo RR: ", round(rr_mean,4), " (", round(IC_inf,4), " , ", round(IC_sup,4), ")")
```

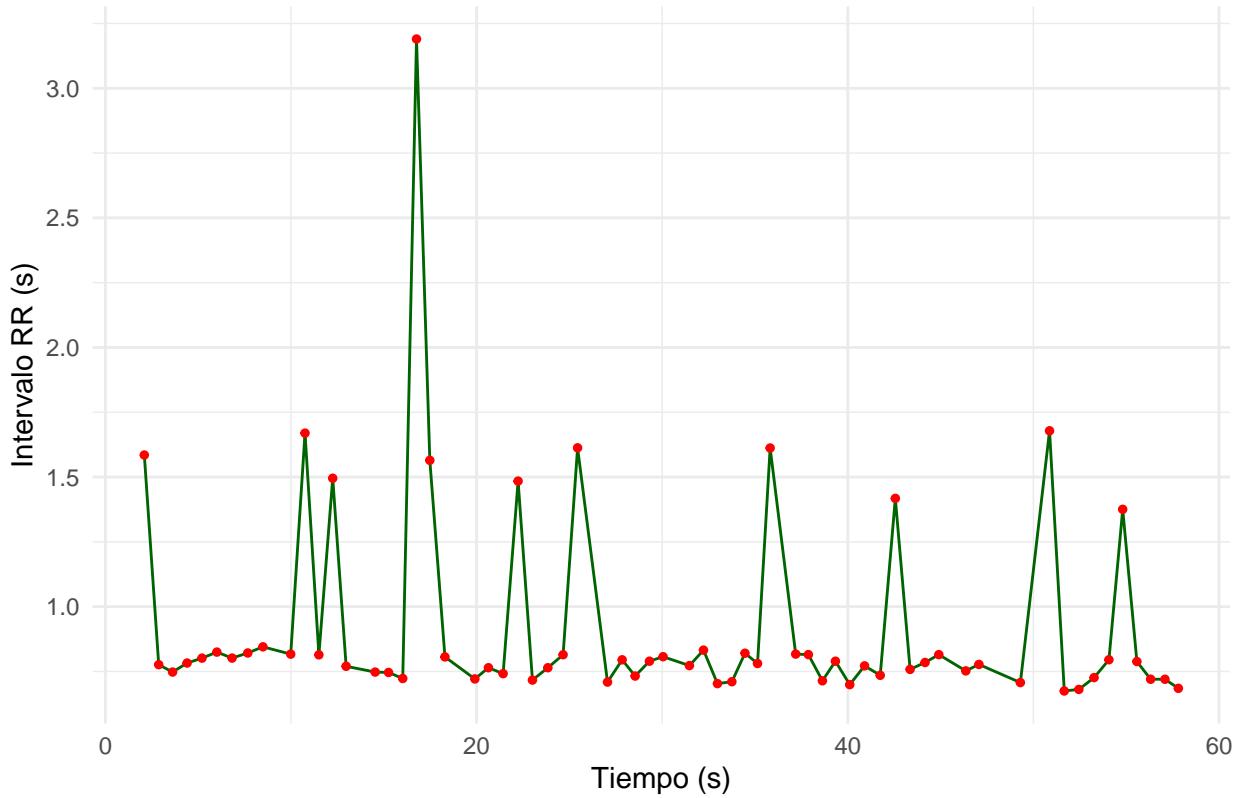
[1] "Intervalo de confianza del 95% para la media del intervalo RR: 0.957 (0.9377-0.9762)"

Ahora vamos a graficar el comportamiento de los intervalos en un 1 minuto

```
RR_data <- data.frame(Time = tiempo_picos[-1], RR_Interval = intervalos_RR)

ggplot(RR_data %>% filter(Time <= 60), aes(x = Time, y = RR_Interval)) +
  geom_line(color = "darkgreen") +
  geom_point(color = "red", size = 1) +
  labs(title = "Intervalos RR a lo largo del tiempo", x = "Tiempo (s)", y = "Intervalo RR (s)") +
  theme_minimal()
```

Intervalos RR a lo largo del tiempo



4) Usa algún método estadístico para detectar valores anómalos en la serie

Hay diversos métodos para detectar Outliers, aplicaremos 3 de ellos.

El método Rango Intercuartílico (IQR)

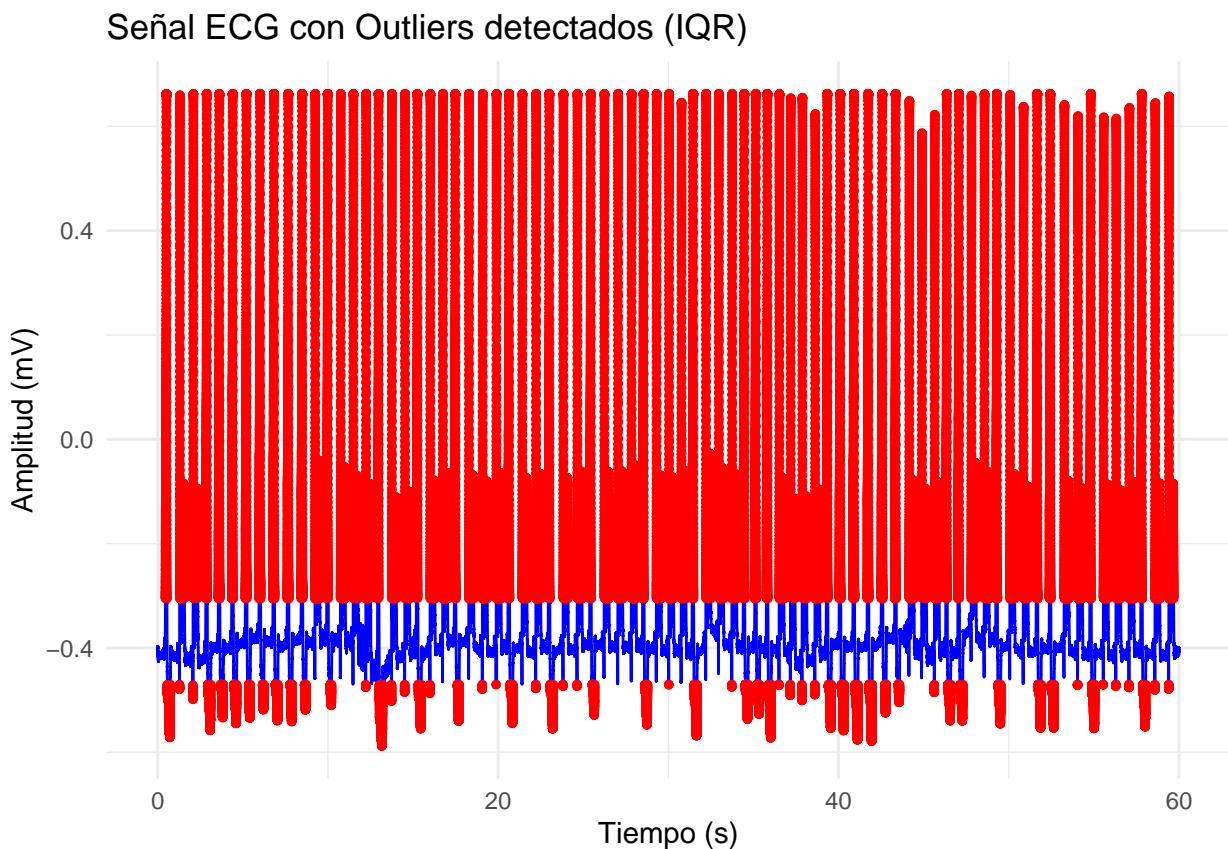
El método del Rango Intercuartílico (IQR) es una técnica estadística que se utiliza para identificar valores atípicos (outliers) en un conjunto de datos. Este método se basa en la diferencia entre los cuartiles de los datos y se utiliza para detectar valores que están muy alejados de la tendencia central

```
# Calcular los cuartiles
Q1 <- quantile(ECG_mV, 0.25)
Q3 <- quantile(ECG_mV, 0.75)
IQR <- Q3 - Q1

# Definir los límites inferior y superior para detectar outliers
limite_inferior <- Q1 - 1.5 * IQR
limite_superior <- Q3 + 1.5 * IQR

# Detectar los valores anómalos
IQR<-ECG_data%>%
  filter(ECG<limite_inferior | ECG>limite_superior)
```

```
# Graficar la señal ECG con los outliers detectados usando IQR
ggplot(ECG_data%>%filter(Time<=60),
       aes(x = Time, y = ECG)) +
  geom_line(color = "blue") + # Señal ECG
  geom_point(data = IQR%>%filter(Time<=60),
             aes(x = Time, y = ECG),
             color = "red",
             size = 1) + # Outliers
  labs(title = "Señal ECG con Outliers detectados (IQR)",
       x = "Tiempo (s)",
       y = "Amplitud (mV)") +
  theme_minimal()
```



El método Z-score

El método Z-score es otra técnica estadística utilizada para identificar valores atípicos (outliers). Este método se basa en la transformación de los datos a una escala estándar y se utiliza especialmente cuando los datos siguen una distribución aproximadamente normal.

```
# Calcular la media y la desviación estándar
media <- mean(ECG_data$ECG)
desviacion_estandar <- sd(ECG_data$ECG)
```

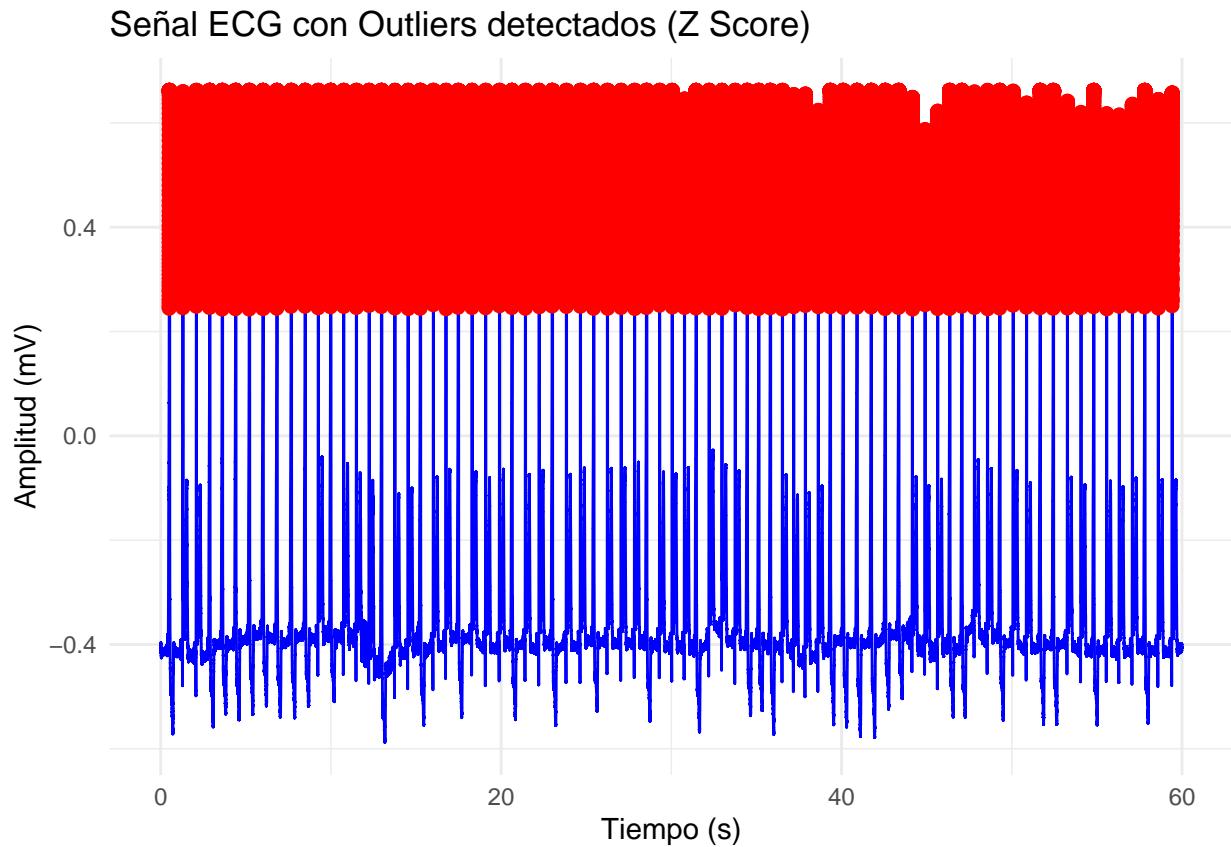
```

# Calcular el Z-score para cada valor
ECG_data$z_scores <- (ECG_data$ECG - media) / desviacion_estandar

Z_outliers<-ECG_data%>%
  filter(abs(z_scores) > 3)

ggplot(ECG_data%>%filter(Time<=60),
       aes(x = Time, y = ECG)) +
  geom_line(color = "blue") + # Señal ECG
  geom_point(data =Z_outliers%>%filter(Time<=60),
             aes(x = Time, y = ECG),
             color = "red",
             size = 2) + # Outliers en rojo
  labs(title = "Señal ECG con Outliers detectados (Z Score)",
       x = "Tiempo (s)",
       y = "Amplitud (mV)") +
  theme_minimal()

```



El método de los percentiles

El método de los percentiles es una técnica utilizada para identificar valores atípicos (outliers) mediante el análisis de la distribución de los datos en términos de percentiles. Este enfoque es especialmente útil cuando los datos no siguen una distribución normal, ya que no depende de la media o la desviación estandar.

```

# Calcular los percentiles 2.5% y 97.5%
percentil_2_5 <- quantile(ECG_data$ECG, 0.025)
percentil_97_5 <- quantile(ECG_data$ECG, 0.975)

# Detectar los valores anómalos
outliers_percentiles <- which(ECG_mV < percentil_2_5 | ECG_mV > percentil_97_5)

Outlier_Percentil<-ECG_data%>%
  filter(ECG<percentil_2_5 | ECG>percentil_97_5)

ggplot(ECG_data%>%filter(Time<=60),
       aes(x = Time, y = ECG)) +
  geom_line(color = "blue") + # Señal ECG
  geom_point(data = Outlier_Percentil%>%filter(Time<=60),
             aes(x = Time, y = ECG),
             color = "red",
             size = 1) + # Outliers en rojo
  labs(title = "Señal ECG con Outliers detectados (Percentil)",
       x = "Tiempo (s)",
       y = "Amplitud (mV)") +
  theme_minimal()

```

