# Tutorial NeuroKit 2

Neurokit 2 es un paquete de herramientas de Python para el procesamiento de señales neurofisiológicas que brinda un acceso sencillo a rutinas de programación avanzadas para el procesamiento de estas.

Inicialmente, este paquete debe ser descargado empleando la siguiente línea de código en Python: *pip install neurokit2*

Posteriormente debe ser importado: *import neurokit2 as nk*

Luego de esto, es necesario cargar los datos con los que se va a trabajar.

- **Evaluar la calidad de la señal**

La función ecg_quality() permite evaluar la calidad de la señal de ECG y definir si esta es útil para un análisis posterior o si tiene presencia de artefactos. Esta función evalúa características específicas de la señal y devuelve un puntaje, este puntaje va 0 a 1, donde cero significa una señal completamente ruidosa y 1 una señal de alta calidad.

```
[24] #Evaluación de calidad
     a=nk.ecg_quality(ecg_signal, rpeaks = None , sampling_rate = 250 , method = 'averageQRS' , approach = None)
     print(a)
```

```
[0.10876084 0.10876084 0.10876084 ... 0.         0.         0.        ]
```

Donde:

- **ecg_signal**: La señal ECG que se desea evaluar.
- **rpeaks**: Los picos R de la señal ECG, que son opcionales y pueden ser detectados automáticamente si no se pasan explícitamente.
- **sampling_rate**: La tasa de muestreo de la señal ECG.
- **method**: El método para evaluar la calidad del ECG, que por defecto es 'averageQRS'.
- **approach**: Un enfoque adicional para la evaluación, que por defecto no se usa.

- **Extraer y visualizar los picos P, Q, R, S y T de un ECG**

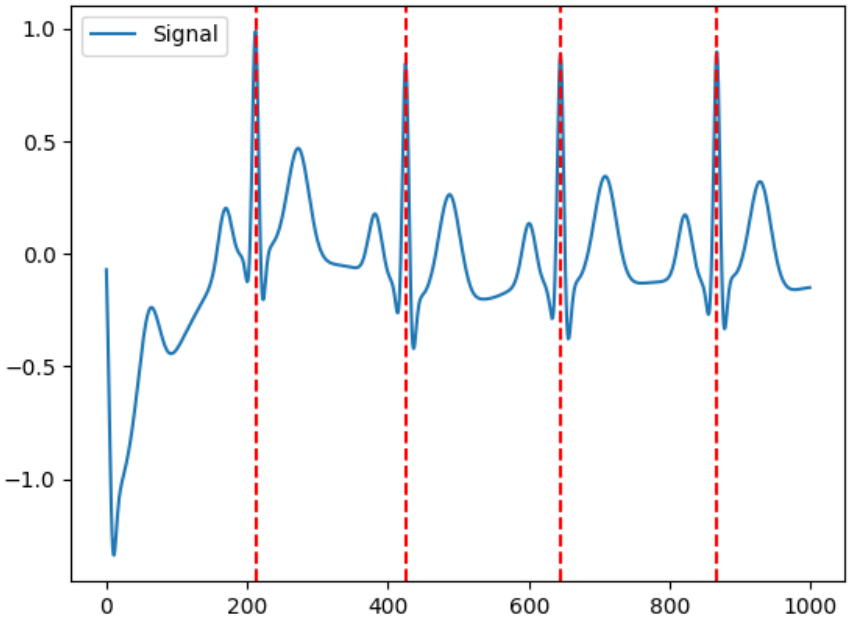
En caso de que las señales ECG estén sin procesar, se puede hacer uso del comando *ecg_process()* pasando como argumentos el conjunto de datos y la frecuencia de muestreo. Esta función retorna las señales (tanto sin procesar como las limpias) y datos adicionales como la ubicación de los picos R.

```
# Generación de una señal ECG
ecg_signal = nk.ecg_simulate(duration=30, sampling_rate=250)
```

```
# Procesamiento de la señal ECG
signals, info = nk.ecg_process(ecg_signal, sampling_rate=250)
```

```
# Extraer señal ECG limpia y los picos R
rpeaks = info["ECG_R_Peaks"]
cleaned_ecg = signals["ECG_Clean"]
```

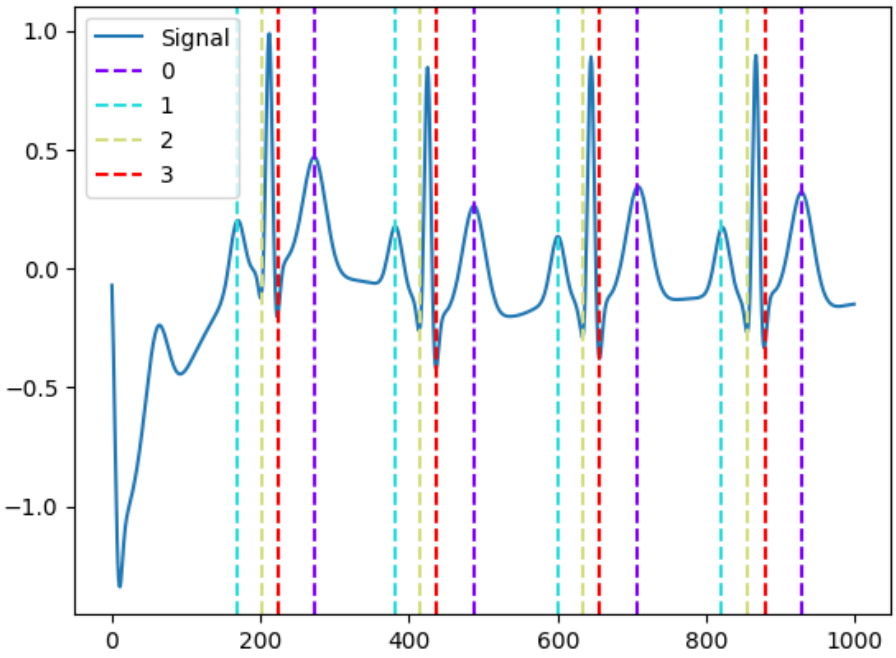
```
# Ver los picos R en la señal ECG
plot = nk.events_plot(rpeaks, cleaned_ecg)
```




Para extraer la ubicación de los picos P, Q, S, T se puede hacer de la misma manera como se muestra en la siguiente imagen

```
# Extraer los picos T, P, Q y S
p_peaks = info["ECG_P_Peaks"]
t_peaks = info["ECG_T_Peaks"]
q_peaks = info["ECG_Q_Peaks"]
s_peaks = info["ECG_S_Peaks"]
```

```
# Ver los picos T, P, Q y S
nk.events_plot([t_peaks[0:4], p_peaks[0:4], q_peaks[0:4], s_peaks[0:4]], cleaned_ecg[0:1000])
```




Otro comando que puede ser empleado para determinar la ubicación de los picos es *ecg_findpeaks()* a la cual se le pasa la señal de interés y retorna un diccionario con múltiples datos sobre todos los picos de la señal.

```
picos = nk.signal_findpeaks(ecg_signal)
```

```
picos.keys()
```

```
dict_keys(['Peaks', 'Distance', 'Height', 'Width', 'Onsets', 'Offsets'])
```

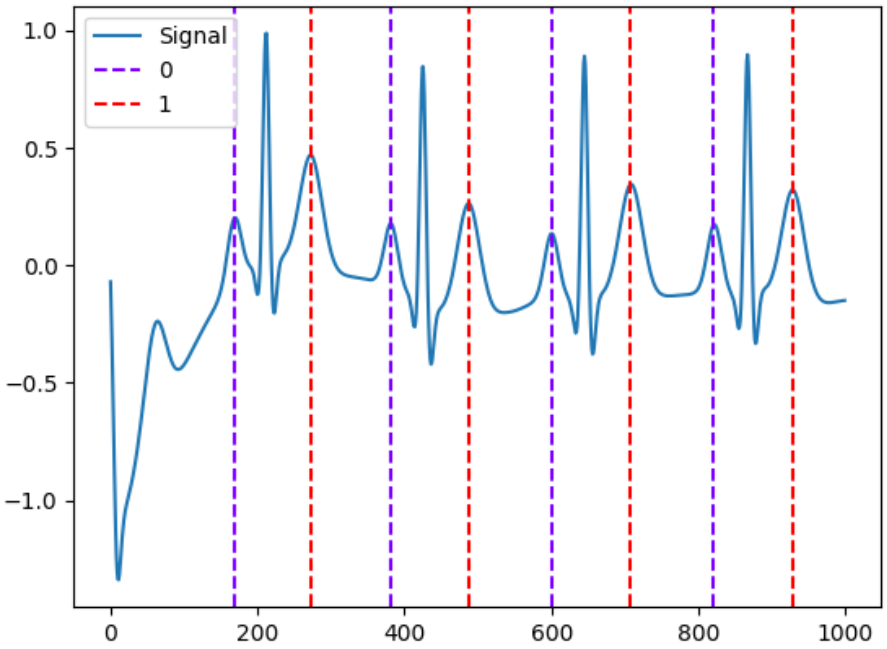
- **Delinear el complejo QRS**

Para delimitar el complejo QRS, es decir, las diferentes ondas de los ciclos cardíacos se implementa la función *ecg_delineate()* a la cual se le pasan como argumentos la señal de interés, los picos R de esta señal y la frecuencia de muestreo.

```
# Se delinea el complejo
signals, waves = nk.ecg_delineate(cleaned_ecg, rpeaks, sampling_rate=250)
```

```
# Se grafican los picos P y los picos T
nk.events_plot([waves["ECG_P_Peaks"][0:4], waves["ECG_T_Peaks"][0:4]], cleaned_ecg[0:1000])
```

*Waves* también almacena información del inicio y fin de cada una de las ondas de la señal.




- **Frecuencia media**

Para calcular la frecuencia media de la señal de ECG, se puede implementar la función *ecg_eventrelated()* que retorna un dataFrame con información relacionada con eventos en épocas que contienen señales de ECG.

Esta función recibe como argumento las épocas de la señal que son determinadas con el comando *epochs_create()* que recibe la señal de interés, los eventos (su lugar de inicio o también puede ser un *None*), la frecuencia de muestreo, el inicio y el final de cada época en relación con el inicio de los eventos. Los eventos a su vez son encontrados con el comando *events_find()*.

```
# Frecuencia media
# Identificacion de eventos
events = nk.events_find(signals, threshold_keep='below', event_conditions=None)
# Identificacion de epocas de acuerdo a los eventos
epochs = nk.epochs_create(signals, events, sampling_rate=250, epochs_start=-0.1, epochs_end=1.9)
# Analisis de las epocas
analyze_epochs = nk.ecg_eventrelated(epochs)
```

Show hidden output

```
analyze_epochs['ECG_Rate_Mean']
```

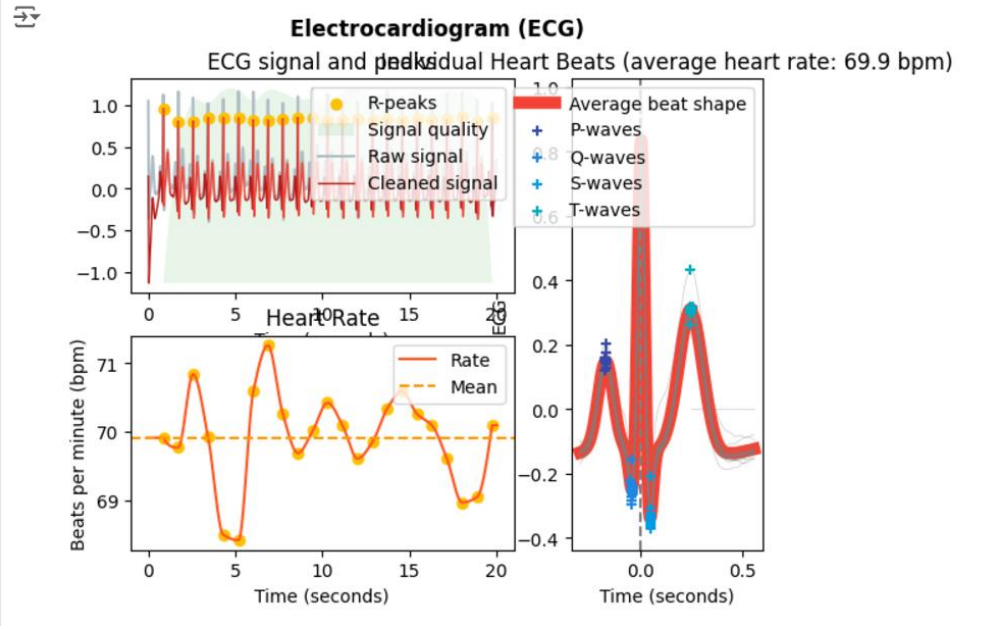
| | ECG_Rate_Mean |
|---|---|
| 1 | 1.066667 |
| 2 | 0.508137 |
| 3 | -0.483508 |

**dtype:** float64

- **Grafica señales de ECG**

La función ecg_plot() utiliza para graficar señales ECG y proporcionar información adicional sobre la señal.

```
ecg, info = nk.ecg_process(signal)  # Se toma la señal y se obtiene el dataframe
info["custom_info"] = "Frecuencia de muestreo: 250 Hz, Duración: 10 segundos"
nk.ecg_plot(ecg, info=info)
plt.show()
```




Donde:

- **ecg_signals**: La señal ECG a graficar, que puede ser un array de NumPy o una lista.
- **info**: Un texto opcional que se puede mostrar en el gráfico para proporcionar información adicional sobre la señal, como la frecuencia de muestreo, la duración, el paciente, etc.