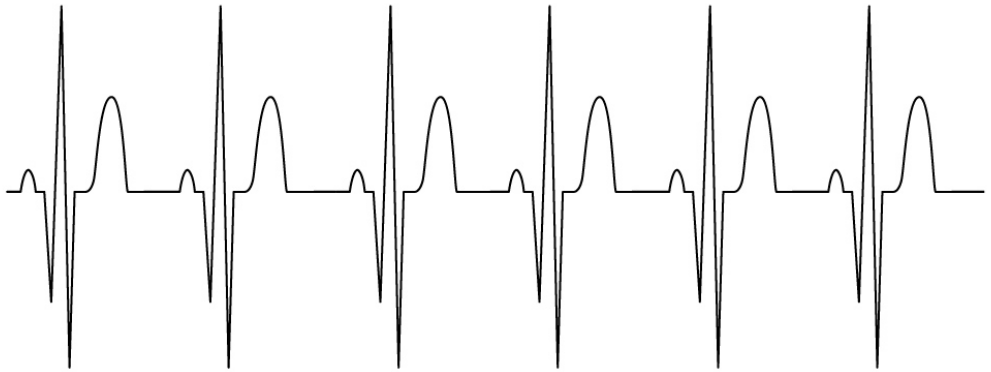
# ICT for HEART MONITORING

Federico Mason
federico.mason@unipd.it

# PART 2
# HEART RATE VARIABILITY

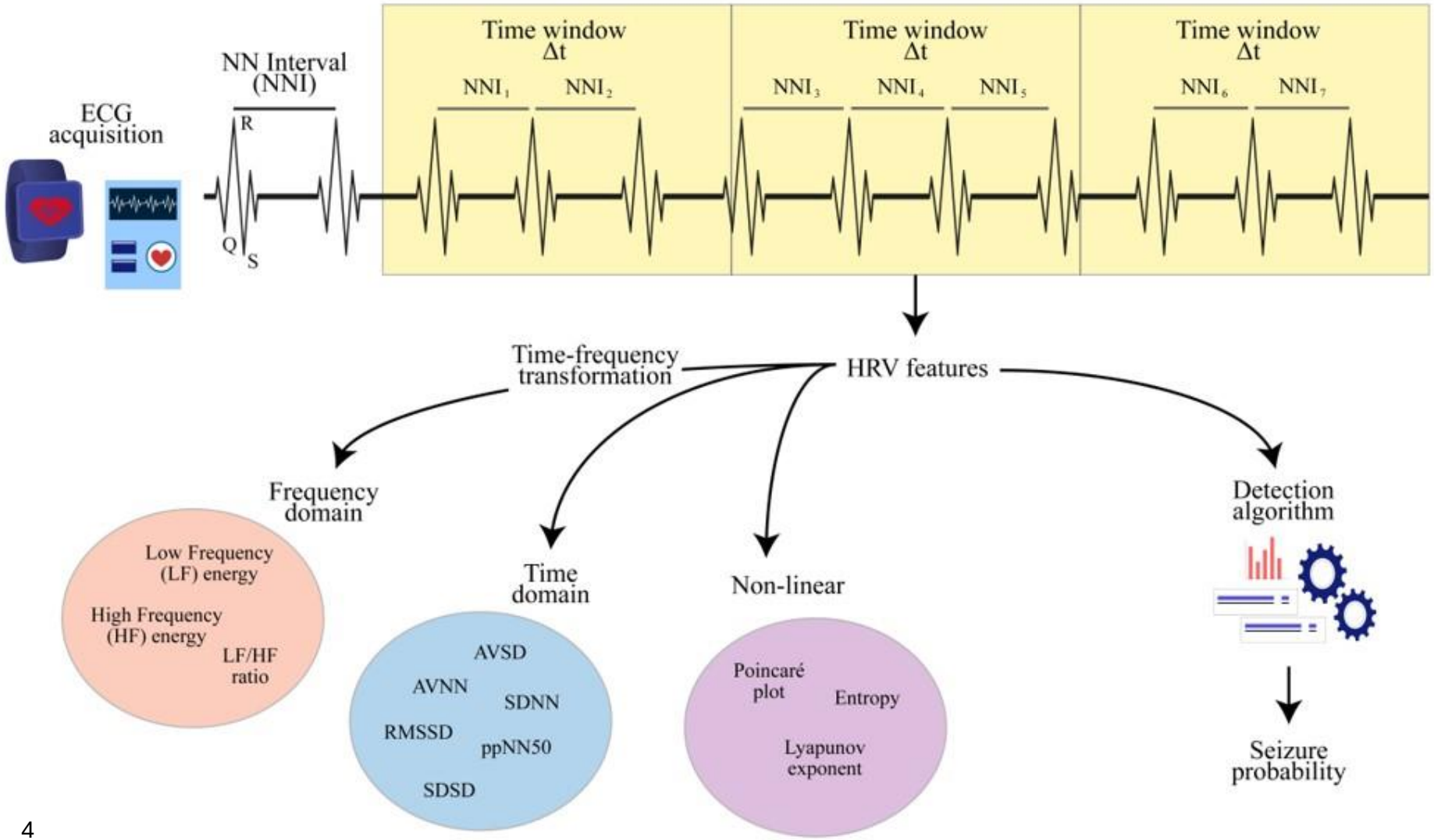
# SINGLE LEAD ECG RECORDING

The recording of a single-lead ECG can be achieved via **minimal invasive** devices, such as wearables or subcutaneous patches

- A single-lead ECG is sufficient to analyze the heartbeats and estimate both the heart rate and its variability

# SINGLE LEAD ECG RECORDING



# HEART RATE VARIABILITY ESTIMATION

To estimate the HRV, it is first necessary to compute the **Normal-to-Normal Intervals** (NNIs), i.e., the intervals between consecutive R peaks

To make the HRV estimation time-dependent, a common choice is to select a **time window** $\Delta t$, so that any feature $HRV(t)$ is computed according to the NNIs measured between time $t$ and time $t + \Delta t$

The selection of $\Delta t$ trades off between **estimation accuracy** and **time sensitivity** → A value of at least $\Delta t = 4\,min$ is usually recommended!

# TIME DOMAIN (I)

The **average of the NNI series** (AVNN) represents the reciprocal of the heart rate, while the **standard deviation of the NNI series** (SDNN) represents a naïve HRV estimation

$$AVNN(t) = \frac{\sum_{t}^{t+\Delta t} NNI_i}{N(t)}$$

$$SDNN(t) = \sqrt{\frac{\sum_{t}^{t+\Delta t} (NNI_i - AVNN(t))^2}{N(t)}}$$

# TIME DOMAIN (II)

More advanced metrics can be derived from the **series SD of subsequent NNI differences** $SD_i = |NNI_{i+1} - NNI_i|$

$$AVSD(t) = \frac{\sum_t^{t+\Delta t} SD_i}{N(t) - 1}$$

$$SDSD(t) = \sqrt{\frac{\sum_t^{t+\Delta t} (SD_i - AVSD(t))^2}{N(t) - 1}}$$

$$RMSD(t) = \sqrt{\frac{\sum_t^{t+\Delta t} (SD_i)^2}{N(t) - 1}}$$

# NNI DISTRIBUTION (I)

Another HRV metric is the **probability** (ppNN$\tau$) **that two consecutive NNIs differ more than $\tau$ milliseconds**, where $\tau$ is usually set to $50\ ms$ or $20\ ms$

$$ppNN\tau(t) = \frac{\sum_{t}^{t+\Delta t} \chi(SD_i > \tau)}{N(t) - 1}$$

$$\chi(SD_i > \tau) = \begin{cases} 1 & SD_i > \tau \\ 0 & otherwise \end{cases}$$

Both RMSD and ppNN$\tau$ are considered markers of the **vagal** (i.e., parasympathetic) **tone** of the nervous system

# NNI DISTRIBUTION (II)

The **triangular index (TI)** is defined as the ratio between the total number of NNIs and the number of NNIs in the modal bin

$$TI(t) = \frac{N(t)}{\sum_{t}^{t+\Delta t} \chi(NNI_i = Mode[NNI(t)])}$$

The **triangular interpolation (TINN)** of NNIs is defined as the baseline width of the NNI distribution

$$TINN(t) \cong max[NNI(t)] - \min[NNI(t)]$$

In the case of TI and TINN, we should consider $\Delta t \geq 20\, min$

# FREQUENCY DOMAIN (I)

We can implement a **time-frequency transformation** (e.g., the Fourier transform) and analyze the NNI spectrum

Given the spectrum $NNI(t, f)$ of NNI, we can compute the energy of significant frequency bands:

- the **low-frequency (LF) band** $[0.06, 0.10]\ Hz$ reflects both the sympathetic and vagal tones

- the **high-frequency (HF) band** $[0.15, 0.40]\ Hz$ reflects the vagal tone

# FREQUENCY DOMAIN (II)

$$E_{LF}(t) = \int_{0.06}^{0.10} |NNI(t,f)|^2 \, df$$

$$E_{HF}(t) = \int_{0.15}^{0.40} |NNI(t,f)|^2 \, df$$

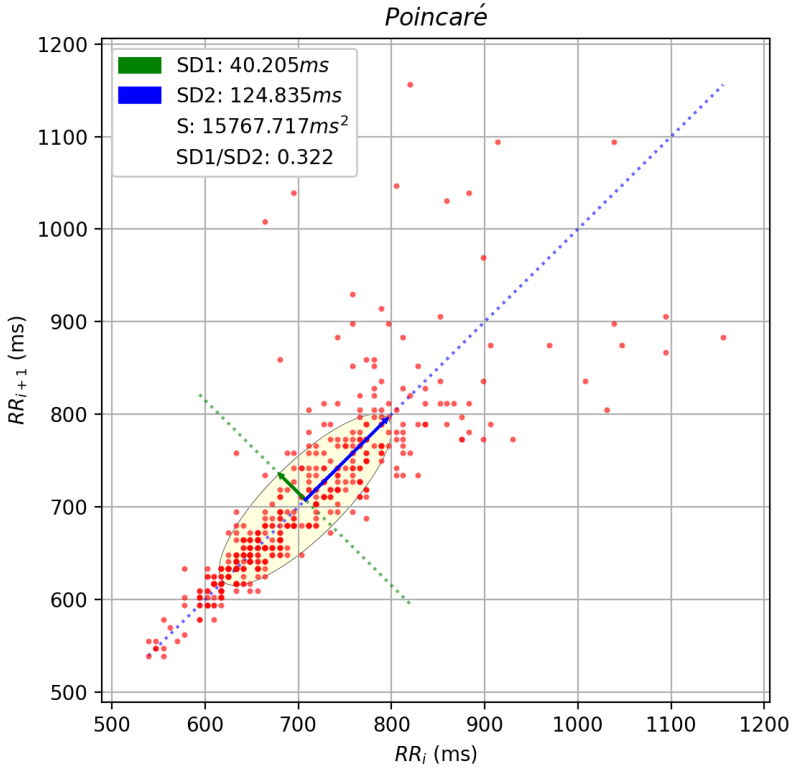
The **energy ratio** between the LF and HF bands is considered a marker of the sympathetic–vagal balance

$$ER(t) = \frac{E_{LF}(t)}{E_{HF}(t)}$$

**Before implementing a time-frequency transformation, we need to interpolate the NNI series to operate with equidistant samples!!!*

# POINCARÉ PLOT

The **Poincaré plot** represents subsequent NNI values in a two-dimensional coordinate system where each point is given by a couple $(NNI_{i+1}, NNI_i)$



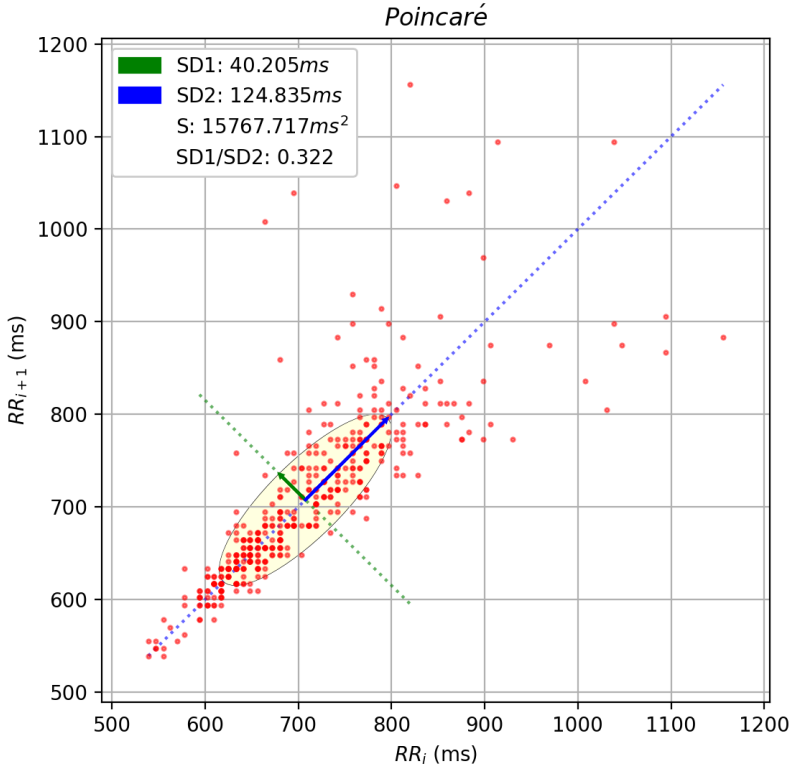

We consider the **standard deviation** perpendicular to $(SD1)$ and along $(SD2)$ the **line-of-identity**

$$SD1^2 = \frac{1}{2} SDSD^2$$

$$SD2^2 = 2SDNN^2 - \frac{1}{2} SDSD^2$$

# POINCARÉ PLOT

The **Poincaré plot** represents subsequent NNI values in a two-dimensional coordinate system where each point is given by a couple $(NNI_{i+1}, NNI_i)$




From SD1 and SD2, we can compute the **Cardiac Sympathetic Index** and the **Cardiac Vagal Index**

$$CSI = {SD1}/{SD2}$$

$$CVI = \log(SD1 \times SD2)$$

# ACTIVITY: SEIZURE DETECTION

HRV has been recognized as a useful biomarker for **detecting** or even **predicting seizures** in people with epilepsy

This makes it possible to trigger alarms to caregivers by using a wearable device

On the other hand, scientific findings show that HRV is effective only in patients with **relevant autonomic changes** during the seizure events

# ACTIVITY: SEIZURE DETECTION

**You are provided with:**

- 5 single-lead ECG recordings of approximately 1.5 hours, during which an epileptic seizure occurs
- a Python framework to extract the NN intervals from the single-lead ECG

**You are asked to:**

- compute at least two HRV metrics among those discussed during the lecture
- analyze the HRV metrics in time and try to determine when the epileptic seizure occurs

# ACTIVITY: SEIZURE DETECTION

**Recommendation:**

- ECG data (like any kind of real-world data) is characterized by noise and artifacts
- we can assume that all the NN intervals larger than $1.5\ s$ are associated with recording errors

$$NNI^* = [NNI_i \in NNI : NNI_i \leq 1.5\ s]$$

- the time domain of time-based HRV features and frequency-based HRV features may be different since the second are obtained after the NNI series interpolation

# PYTHON FRAMEWORK

## Basic usage:

- You can decide which file to process and what part of the code to run by using the command line options
  - to plot the ECG signal associated with *./epoch_0/*, write:

```
:~/Desktop/hrv_lab_full$ python main.py -epoch_folder ./epoch_0/ -plot_ecg
```

  - To compute and plot the NNI intervals associated with *./epoch_0/*, write:

```
:~/Desktop/hrv_lab_full$ python main.py -epoch_folder ./epoch_0/ -compute_nni -plot_nni
```

# PYTHON FRAMEWORK

```
window_times = []
avg_nn_intervals = []
std_nn_intervals = []

last_time = nni_times[-1]

current_time = 0
next_time = window_duration

while next_time < last_time:

    window_times.append(current_time)

    current_time_index = np.argmin(np.abs(current_time-nni_times))
    next_time_index = np.argmin(np.abs(next_time-nni_times))

    window_nn_intervals = nn_intervals[current_time_index:next_time_index]

    avg_nn_intervals.append(np.mean(window_nn_intervals))
    std_nn_intervals.append(np.std(window_nn_intervals))

    current_time += window_shift
    next_time += window_shift
```

# Thank you!

Federico Mason
federico.mason@unipd.it