09-Batch_processing_meditation

June 5, 2021

1 Batch processing for meditation recordings

1.1 Facultad de Ciencias, UNAM

1.1.1 Ana Daniela del Río Pulido and Erin C. McKiernan

Functions used in previous notebooks will be used.

If you are in this notebook, I guess you have passed through the previous techniques. In this notebook, we will see the effects of medidation techniques with different techniques.

2 Setting up the notebook

We begin by setting up the Jupyter notebook and importing the Python modules needed for plotting figures, create animations, etc. We include commands to view plots in the Jupyter notebook, and to create figures with good resolution and large labels. These commands can be customized to produce figures with other specifications.

[1]:
```
# Imports python libraries
import numpy as np
import random as rd
import wave
import sys
import os
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import matplotlib as mpl
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
sys.path.insert(1, r'./../functions') # add to pythonpath

# commands to create high-resolution figures with large labels
%config InlineBackend.figure_formats = {'png', 'retina'}
plt.rcParams['figure.dpi'] = 50
plt.rcParams['axes.labelsize'] = 16 # fontsize for figure labels
plt.rcParams['axes.titlesize'] = 18 # fontsize for figure titles
plt.rcParams['font.size'] = 14 # fontsize for figure numbers
plt.rcParams['lines.linewidth'] = 1.4 # line width for plotting
```

2.1 Extracting data

ECG recordings were obtained using the Backyard Brains Heart and Brain Spiker Box. The recordings are saved as audio files in .wav format. The first thing we have to do is open the .wav files and extract the data. We can extract the number of recording channels, sampling rate, etc.

[2]:
```
#Function that extracts the number of recording channels, sampling rate, time␣
↪and signal
#variable is the path and filename of the .wav file
def ecg(variable):
    record = wave.open(variable, 'r') # load the data

    # Get the number of channels, sample rate, etc.
    numChannels = record.getnchannels() #number of channels
    numFrames = record.getnframes() #number of frames
    sampleRate = record.getframerate() #sampling rate
    sampleWidth = record.getsampwidth()

    # Get wave data
    dstr = record.readframes(numFrames * numChannels)
    waveData = np.frombuffer(dstr, np.int16)

    # Get time window
    timeECG = np.linspace(0, len(waveData)/sampleRate, num=len(waveData))

    return timeECG, waveData
```

3 R peaks

Function for detecting R peaks. We will be able to calculate the heart frequency and R-R intervals.

The following function creates an array of values which surpass a certain threshold. Afterwards, it determines the maximum value of this array and adds it in the R-vector. And this is repeated until the end of the time series.

[3]:
```
def detecta_maximos_locales(timeECG, waveData, threshold_ratio=0.7):
    # If not all the R peaks are detected, lower the threshold_ratio
    # If components that are not R peaks (like T waves) are detected, higher␣
↪the threshold_ratio

    if len(timeECG) != len(waveData): #Raises an error if the two arrays have␣
↪different lengths
        raise Exception("The two arrays have different lengths.")

    interval = max(waveData) - min(waveData)
    threshold = threshold_ratio*interval + min(waveData)
    maxima = []
    maxima_indices = []
```

```
    mxs_indices = []
    banner = False

    for i in range(0, len(waveData)):

        if waveData[i] >= threshold:#If a threshold value is surpassed,
            # the indices and values are saved
            banner = True
            maxima_indices.append(i)
            maxima.append(waveData[i])

        elif banner == True and waveData[i] < threshold: #If the threshold␣
↪value is crossed
            # the index of the maximum value in the original array is saved
            index_local_max = maxima.index(max(maxima))
            mxs_indices.append(maxima_indices[index_local_max])
            maxima = []
            maxima_indices = []
            banner = False

    return mxs_indices
```

[4]:
```
# If the input of this function is time, the intervals will be given in those␣
↪same units
# Obtaining the indexes at which the R peaks occur.
def R_intervals(time_indices):

    length = len(time_indices)
    intervals = np.zeros(length-1)

    for i in range(0, length-1):
        intervals[i] = time_indices[i+1]-time_indices[i]

    return intervals
```

4 Comencing the analysis

For analyzing several recordings at the same time, we must obtain the names of every file. One option is to extract the file names one by one, but another option is just to give a folder's name and extract the recordings from there.

[5]:
```
InputPath = "ECG_samples/meditation_data/" #Folder where the original files are
recordings_path = []
corresponding_folder = []
```

```
#In this case, files must be inside folders inside the folder where this␣
 ↪notebook is
TheList = os.listdir(InputPath)

for Folder in TheList:

    for TheFile in os.listdir(InputPath+Folder):
        corresponding_folder.append(Folder)
        TheFileName, TheFileExtension = os.path.splitext(TheFile) # breaks file␣
 ↪name into pieces based on periods

        InputFilePath = InputPath + Folder +"/"+ TheFileName + TheFileExtension␣
 ↪# Full path to file

        if (TheFileExtension==".wav"): # Only interested in .wav files
            recordings_path.append(InputFilePath)
```

Now we introduce an object. Notice that in this notebook we are dealing with t This is because we are dealing with several subjects and each of them has the same properties: an electrocardiogram recording at rest, an electrocardiogram after performing exercise. And these two recordings have other properties related with them like R-R intervals and time when these R peaks happen. Instead of dealing with several arrays, an object is a data structure that could help you organize the data better.

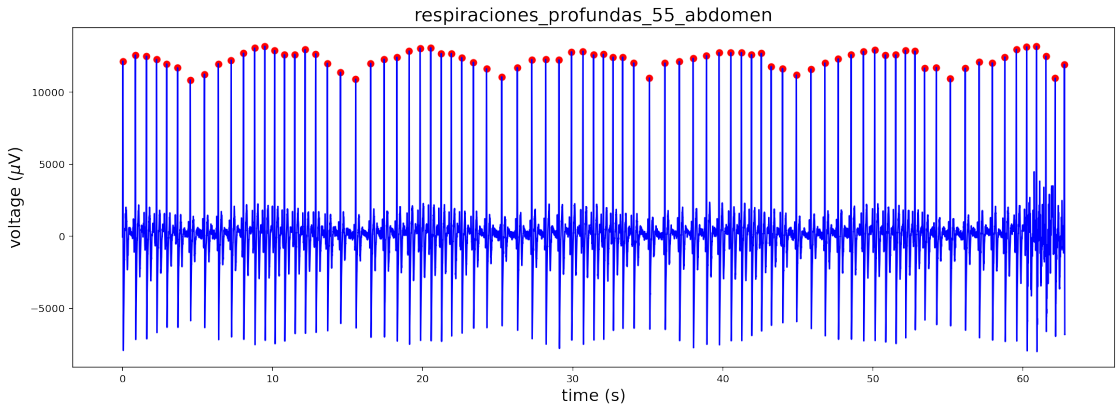
[6]:
```
# Object
class Sujeto:
  def __init__(self, timeECG, waveData):
    #Rest
    self.timeECG = timeECG
    self.waveData = waveData

    self.mxs_indices = detecta_maximos_locales(timeECG, waveData)
    self.RR = R_intervals(timeECG[self.mxs_indices])
    self.timeRpeaks = timeECG[self.mxs_indices]
```

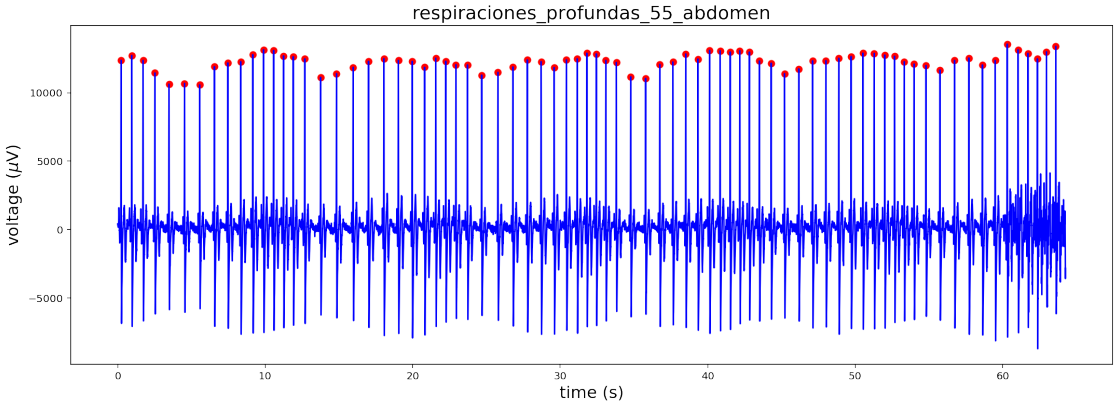
[7]:
```
recordings = []

for i in range(0, len(recordings_path)):
    timeECG, waveData = ecg(recordings_path[i])
    recordings.append(Sujeto(timeECG, waveData))
    print("Finished recording ", i+1)
```

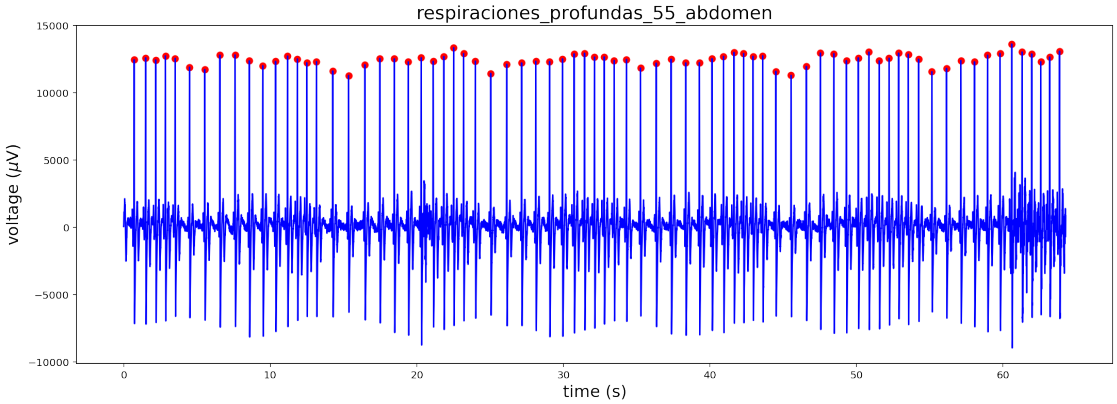
```
Finished recording  1
Finished recording  2
Finished recording  3
Finished recording  4
Finished recording  5
Finished recording  6
```

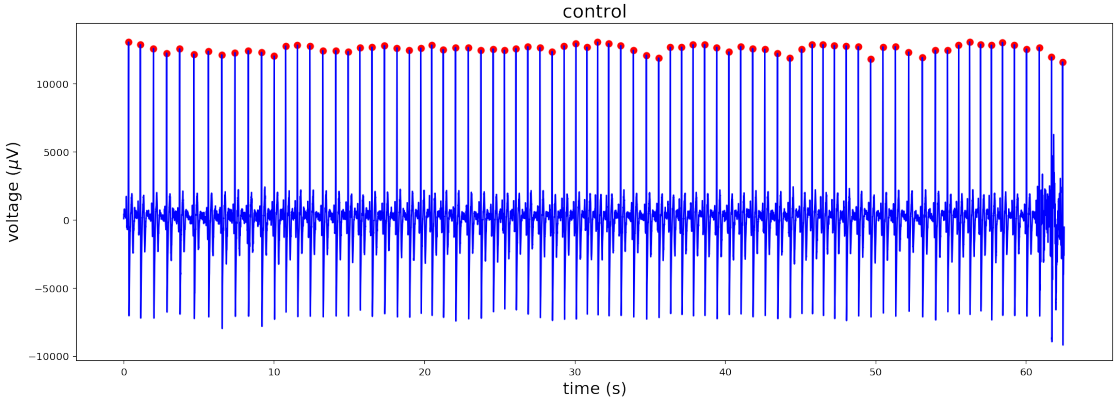
```
Finished recording  7
Finished recording  8
Finished recording  9
Finished recording  10
Finished recording  11
Finished recording  12
Finished recording  13
Finished recording  14
Finished recording  15
Finished recording  16
Finished recording  17
Finished recording  18
```

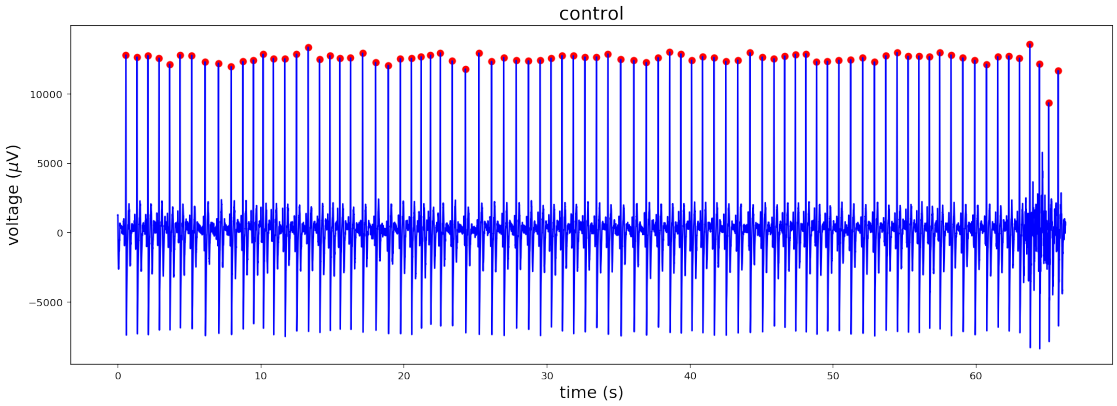
[8]:
```
# Plotting ECG signal
for i in range(0, len(recordings)):
    fig = plt.figure(figsize=(18,6))
    plt.xlabel(r'time (s)')
    plt.ylabel(r'voltage ($\mu$V)')
    plt.plot(recordings[i].timeECG, recordings[i].waveData, 'b')
    plt.scatter(recordings[i].timeECG[recordings[i].mxs_indices],
                recordings[i].waveData[recordings[i].mxs_indices],
                color='r')
    plt.title(corresponding_folder[i])

    fig.savefig('meditation_imgs/ecg_'+str(corresponding_folder[i])+'.
↪jpg')#Saves images in folder

    plt.show()
```

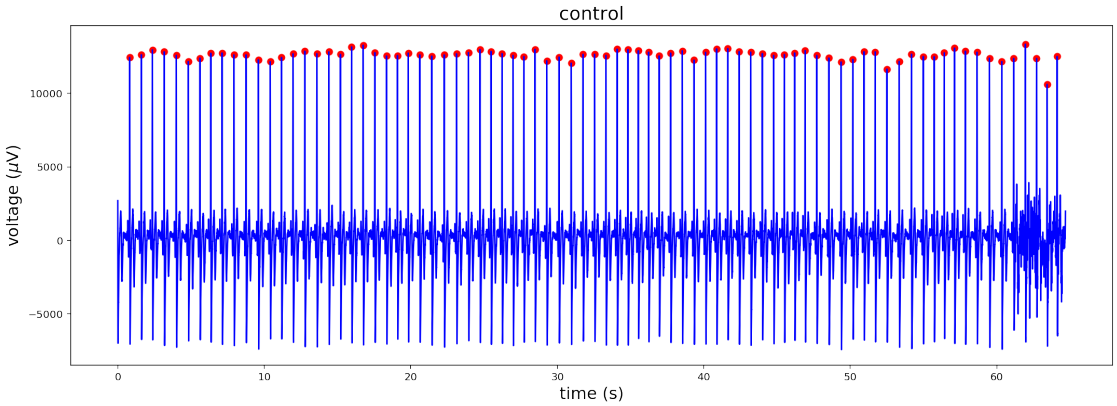


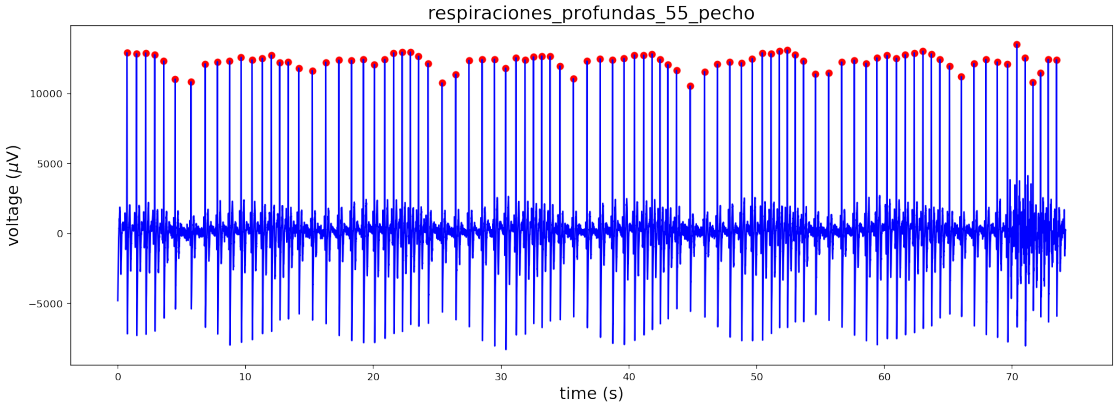

respiraciones_profundas_55_abdomen
voltage (μV)
time (s)

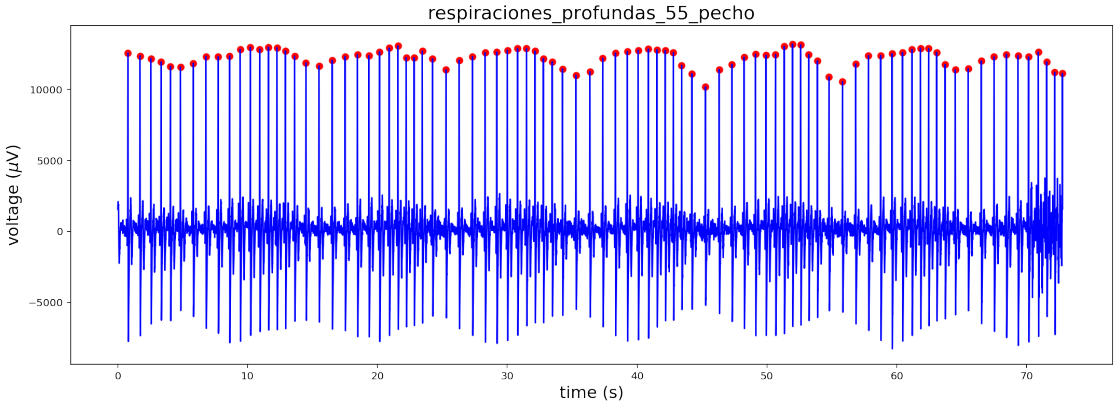

respiraciones_profundas_55_abdomen
voltage (μV)
time (s)

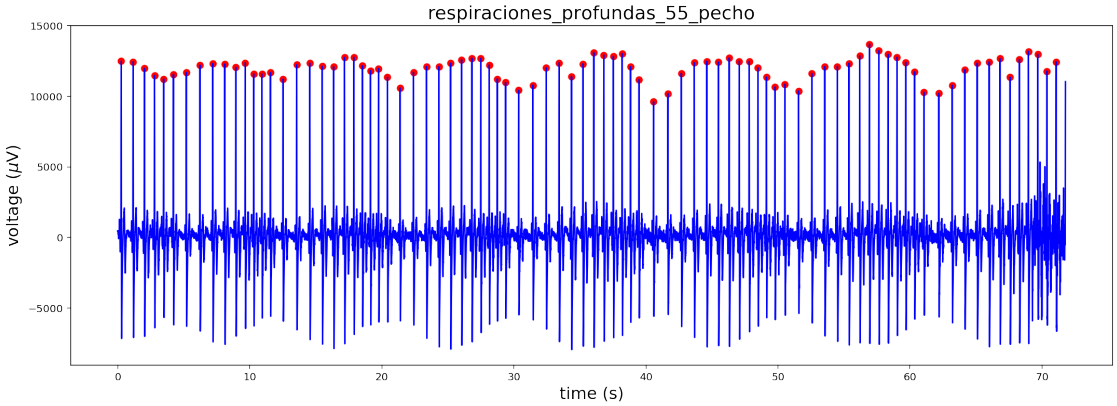

control
voltage (μV)
time (s)

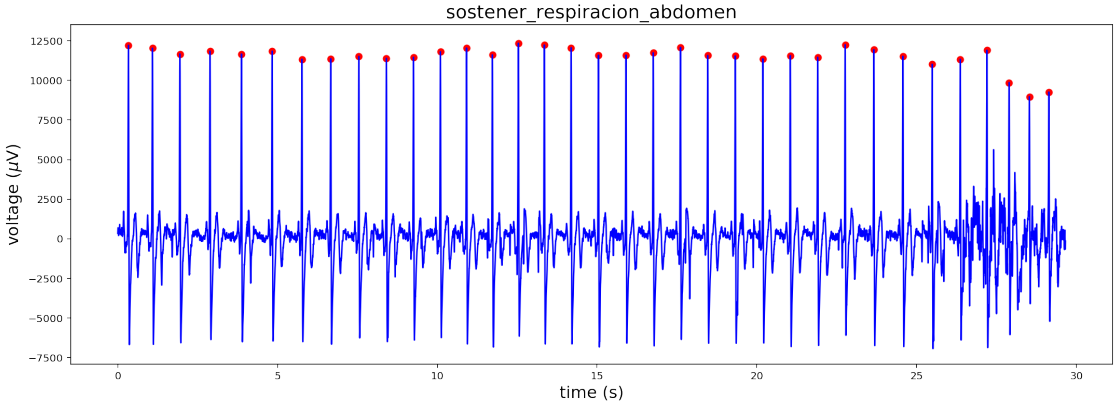

control
voltage (μV)
10000
5000
0
−5000
0
10
20
30
40
50
60
time (s)

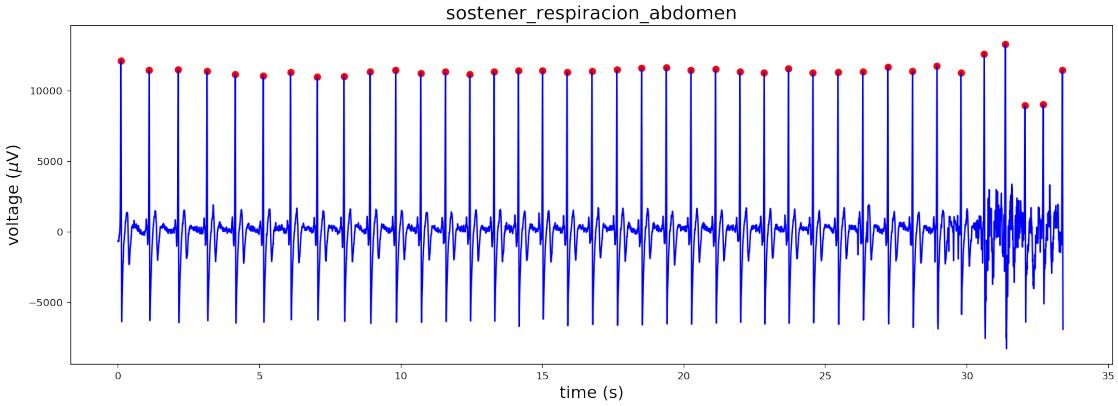

control
voltage (μV)
10000
5000
0
−5000
0
10
20
30
40
50
60
time (s)

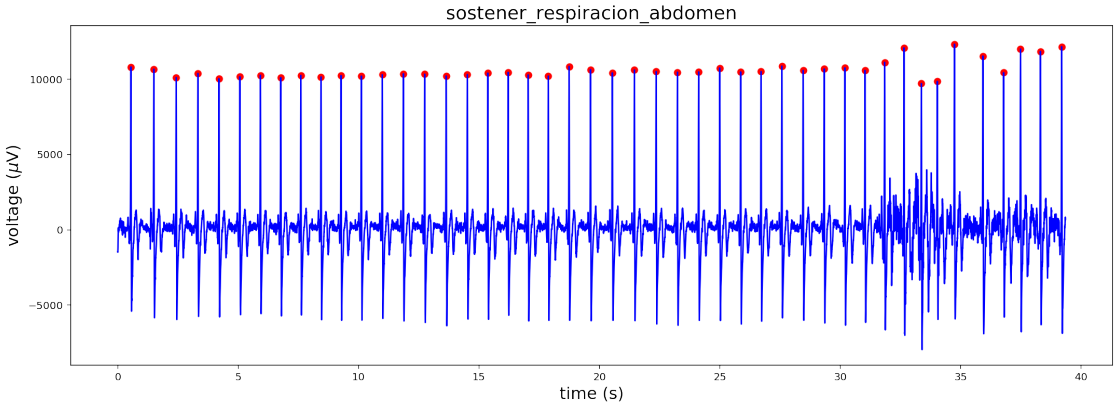

respiraciones_profundas_55_pecho
voltage (μV)
10000
5000
0
−5000
0
10
20
30
40
50
60
70
time (s)

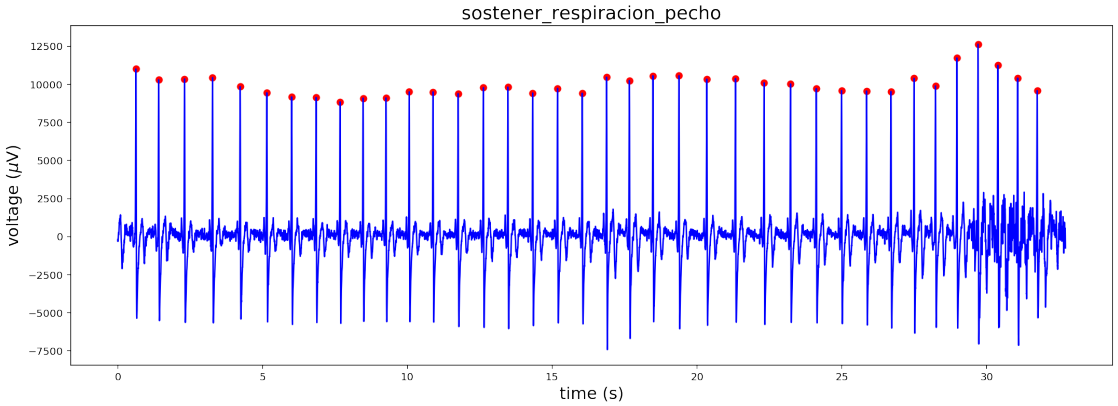

respiraciones_profundas_55_pecho
voltage (μV)
time (s)

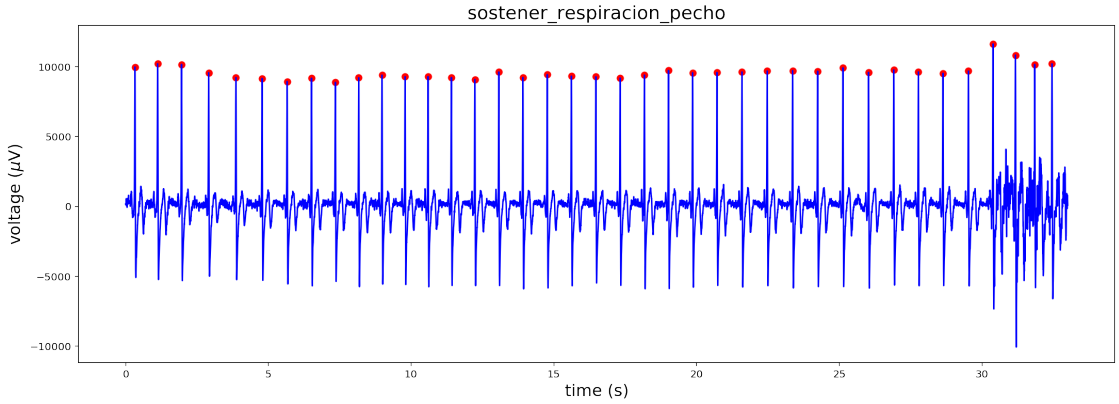

respiraciones_profundas_55_pecho
voltage (μV)
time (s)

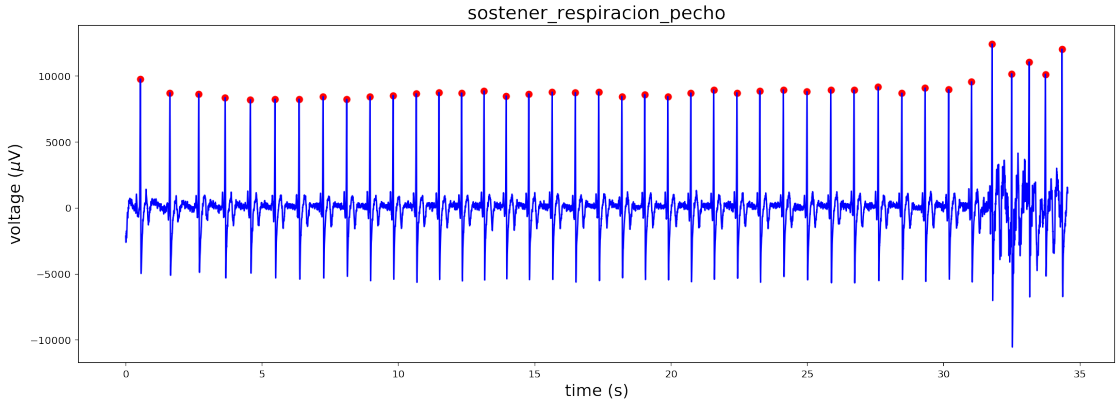

sostener_respiracion_abdomen
voltage (μV)
time (s)

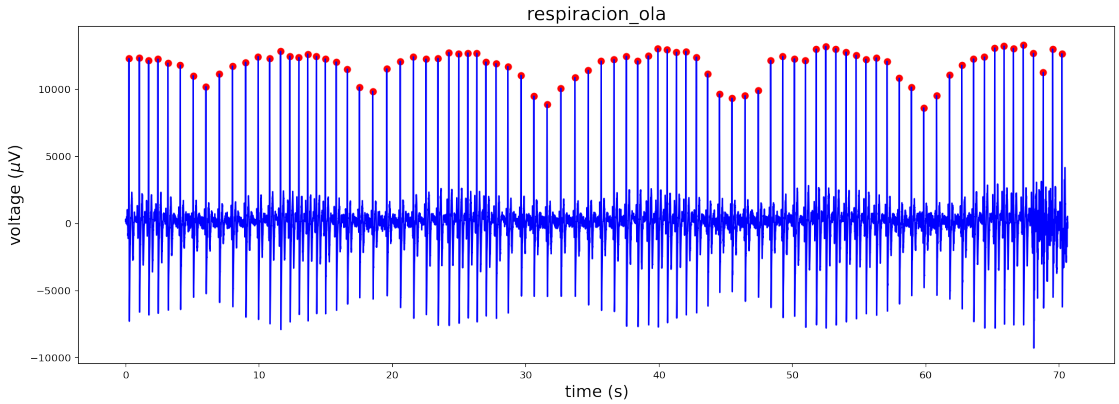

sostener_respiracion_abdomen
voltage (μV)
time (s)

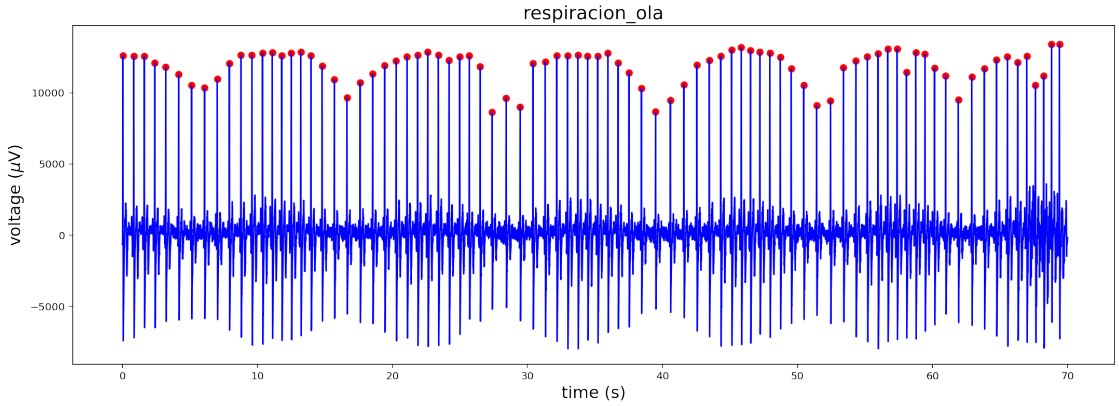

sostener_respiracion_abdomen
voltage (μV)
time (s)

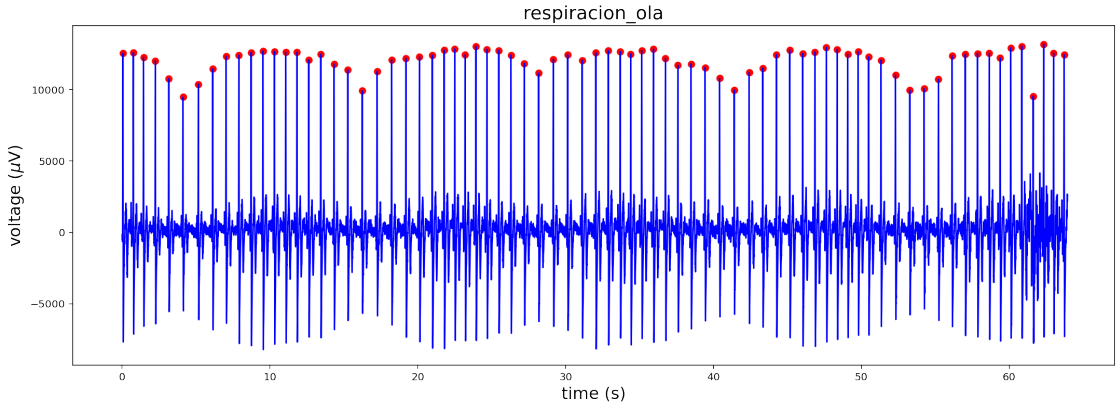

sostener_respiracion_pecho
voltage (μV)
time (s)


sostener_respiracion_pecho
voltage (μV)
time (s)


sostener_respiracion_pecho
voltage (μV)
time (s)


respiracion_ola
voltage (μV)
time (s)

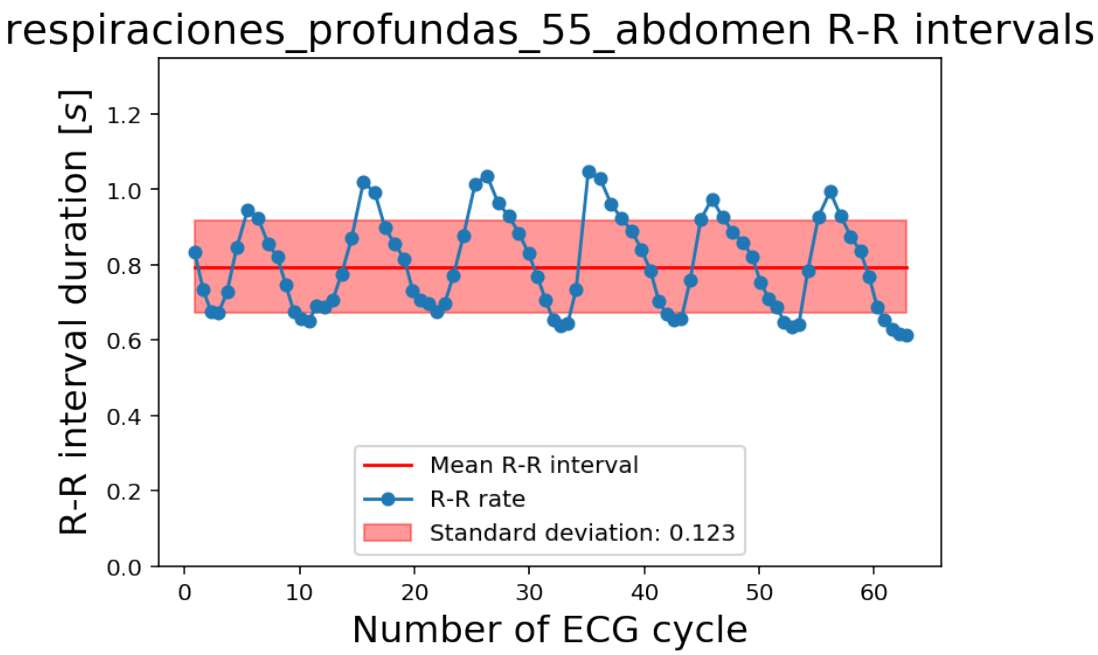


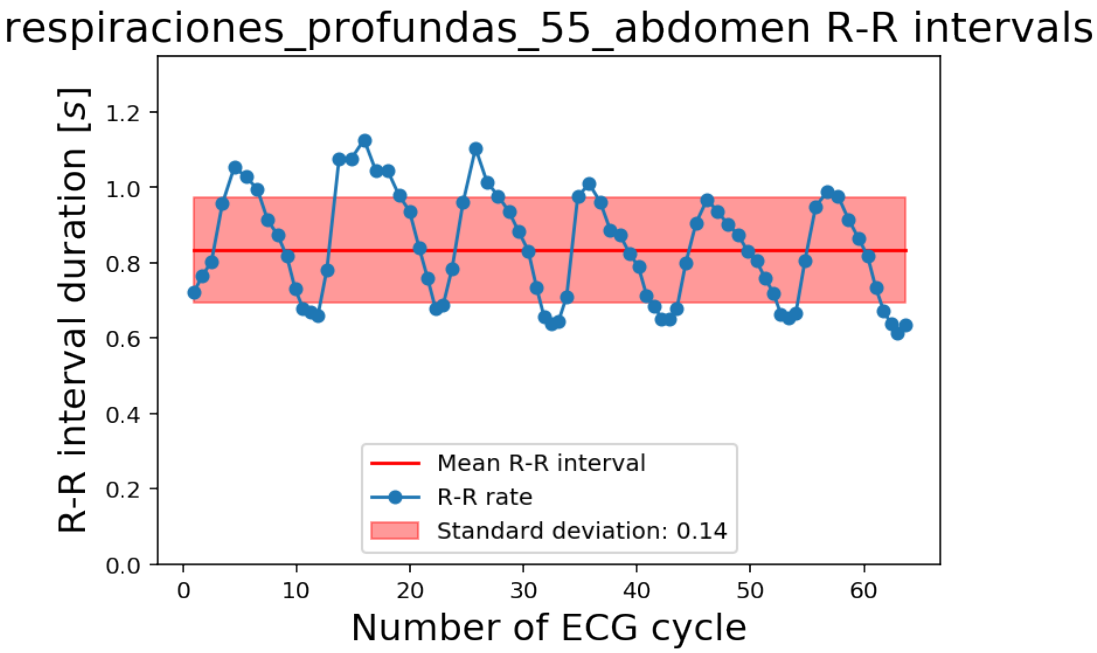


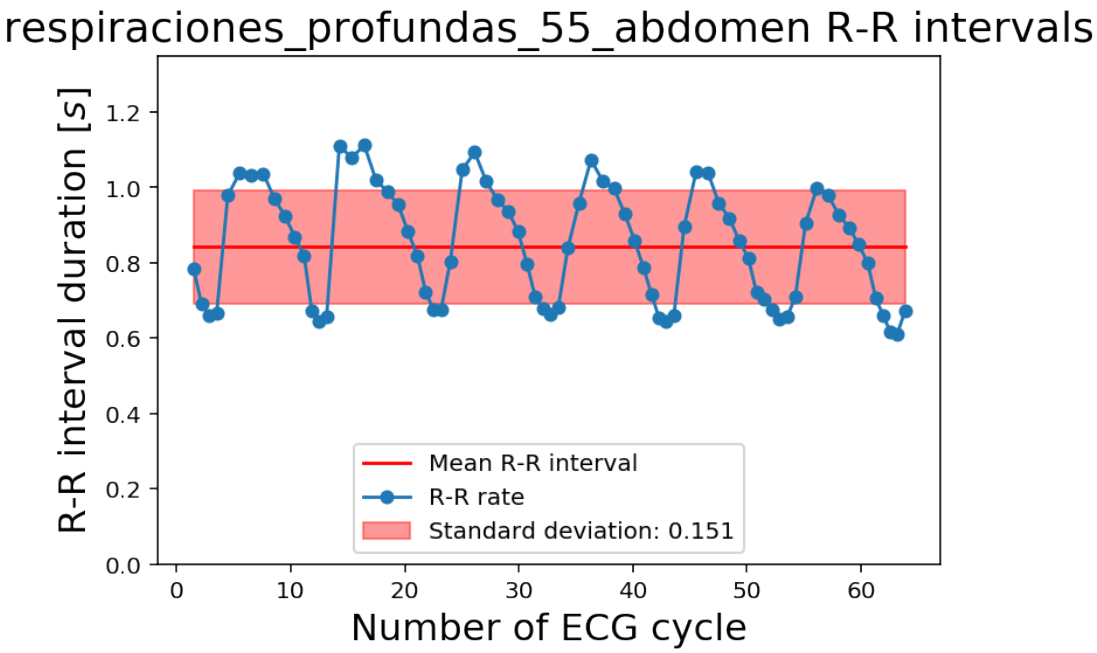
Is there a bad peak detection in the previous recordings? If there is, exclude that recording, if not, let's continue.

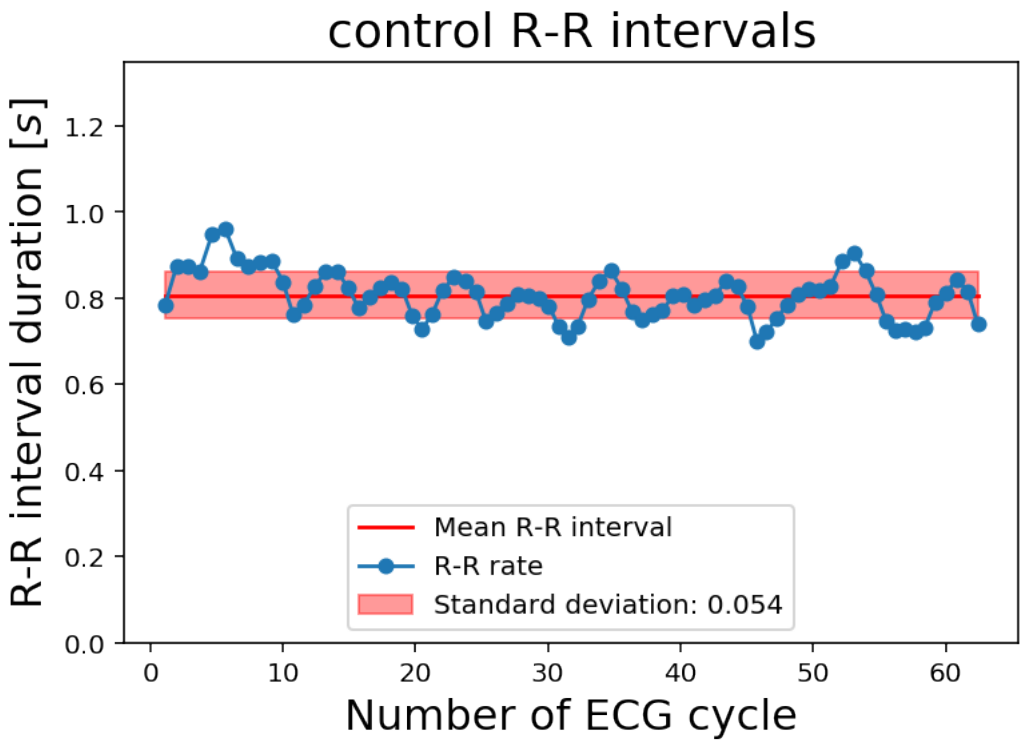
[9]:
```
for i in range(len(recordings)):
    mean_rr = np.mean(recordings[i].RR)
    stdev = np.std(recordings[i].RR)

    mean_vec = [mean_rr for i in range(0, len(recordings[i].timeRpeaks)-1)]
    std_vec_plus = [mean_rr + stdev for i in range(0, len(recordings[i].
 ↪timeRpeaks)-1)]
    std_vec_minus = [mean_rr - stdev for i in range(0, len(recordings[i].
 ↪timeRpeaks)-1)]

#     fig = plt.figure()
    plt.fill_between(recordings[i].timeRpeaks[1:], std_vec_minus,
                     std_vec_plus, facecolor="red",
                     label = "Standard deviation: "+str(round(stdev,3)),␣
 ↪color='red', alpha=0.4)
```

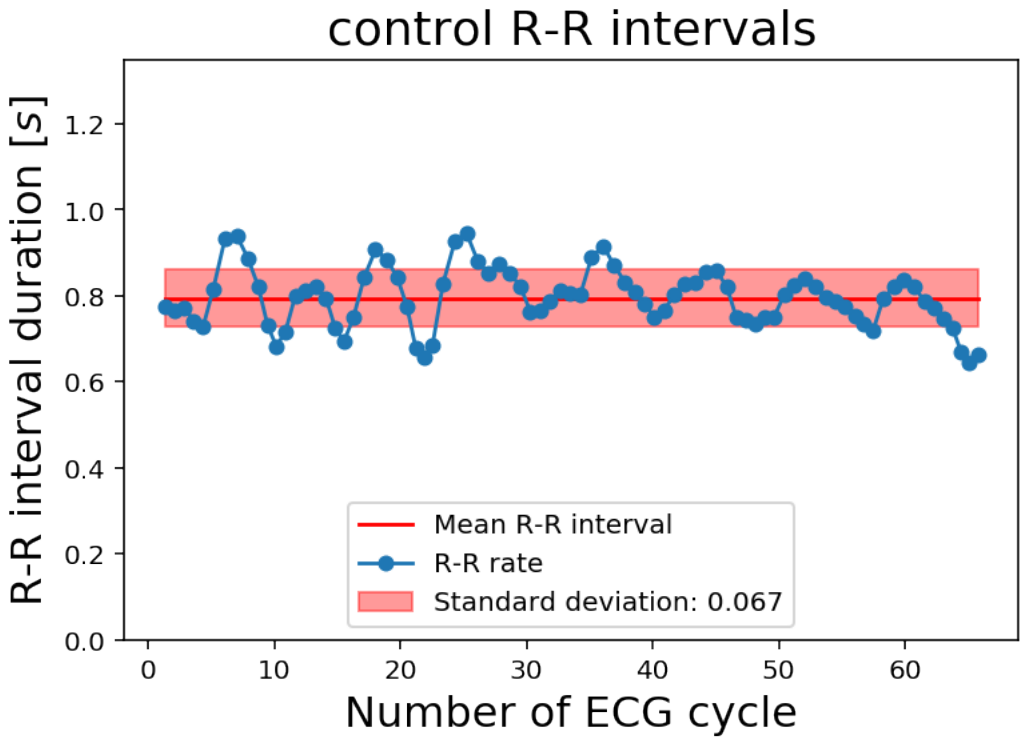
```
    plt.plot(recordings[i].timeRpeaks[1:], mean_vec,
             c="r", label = "Mean R-R interval")
    plt.plot(recordings[i].timeRpeaks[1:], recordings[i].RR,
             markersize=5, marker = "o", label="R-R rate")

    plt.ylim((0, 1.35))
    plt.title(corresponding_folder[i] + " R-R intervals")
    plt.xlabel(r'Number of ECG cycle')
    plt.ylabel(r'R-R interval duration [$s$]')
    plt.legend(loc='lower center')

    plt.savefig('meditation_imgs/desvest_'+str(corresponding_folder[i])+'.
↪jpg')#Saves images in folder

    plt.show()
```

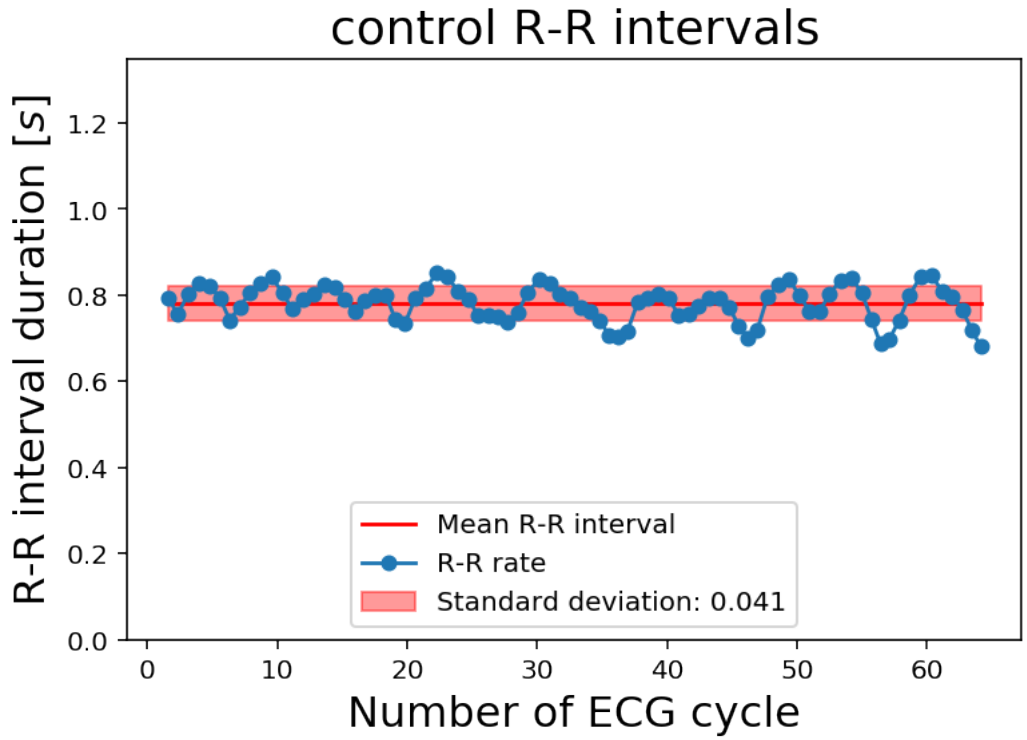


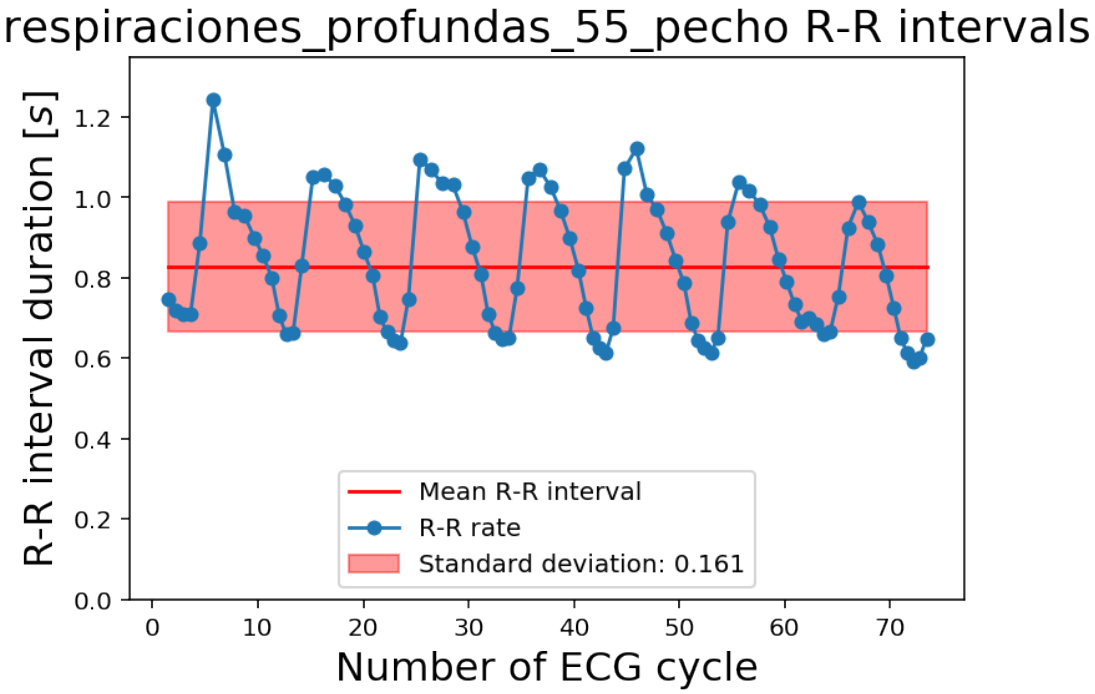

respiraciones_profundas_55_abdomen R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.14

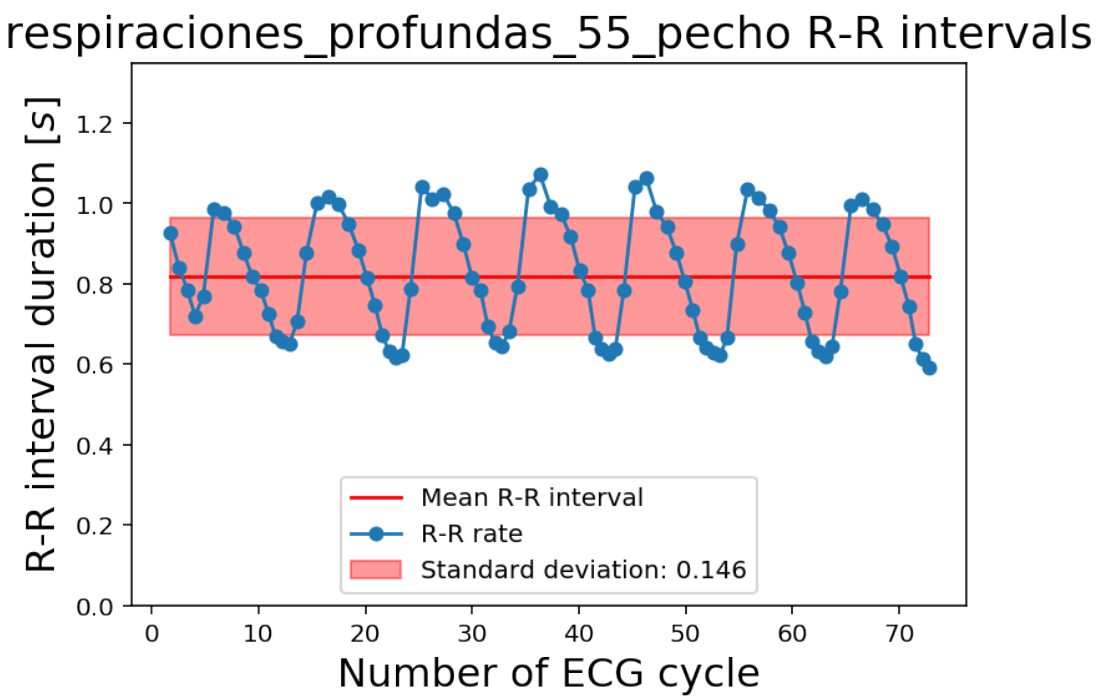

respiraciones_profundas_55_abdomen R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.151

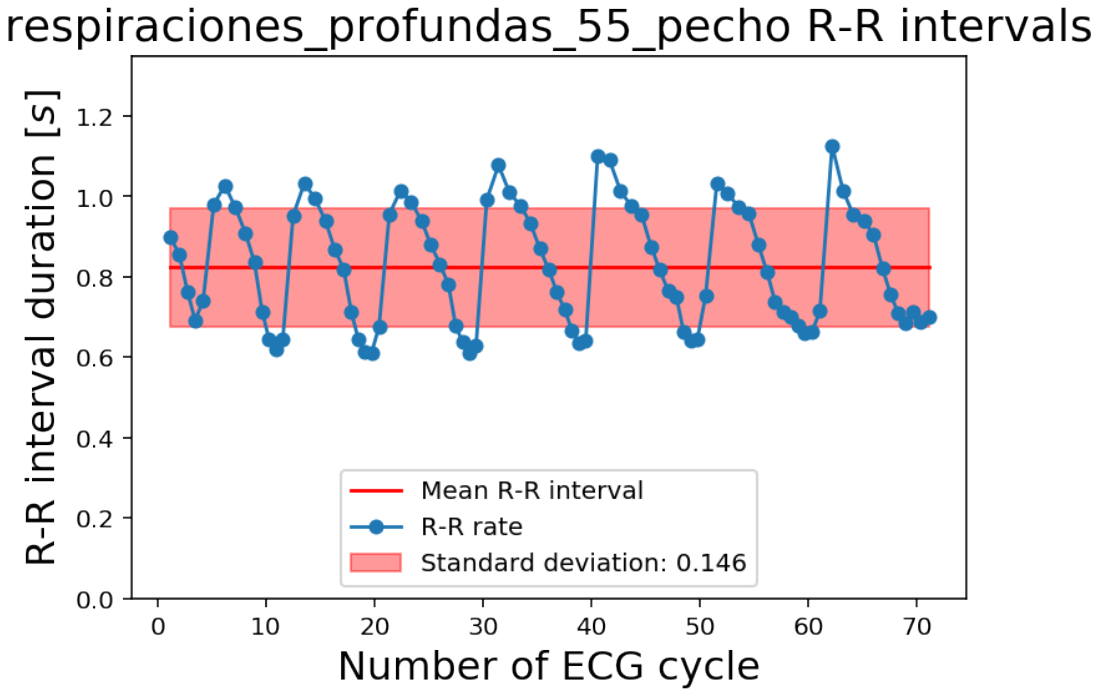

control R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.054

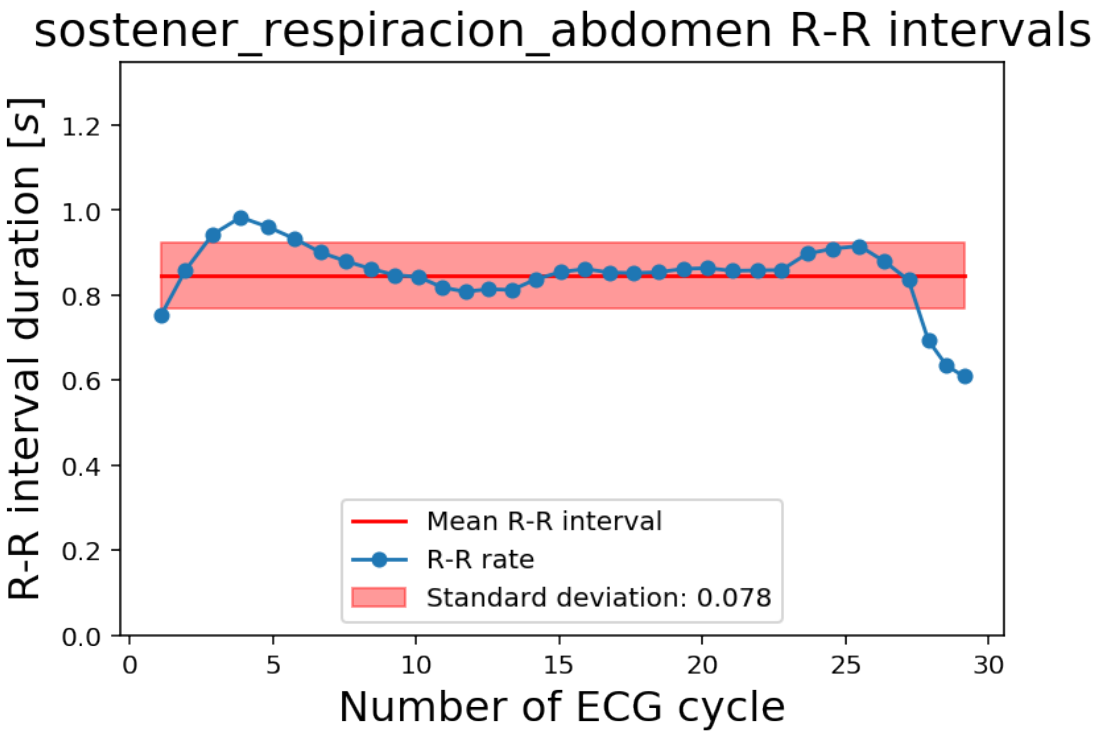

control R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.067

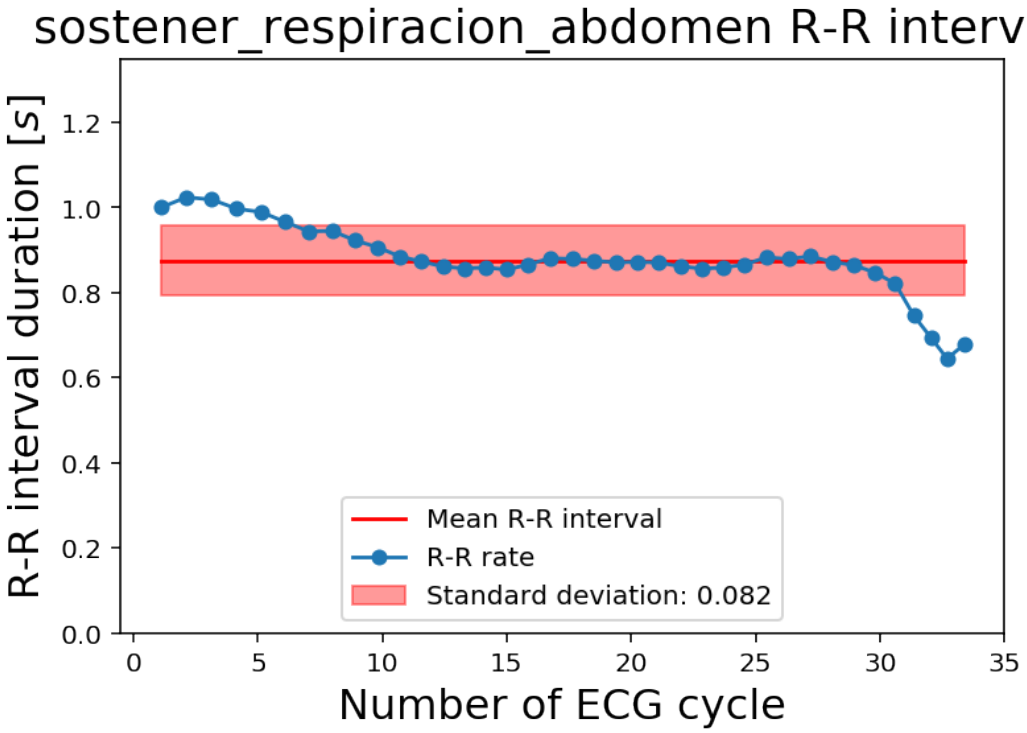

control R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.041

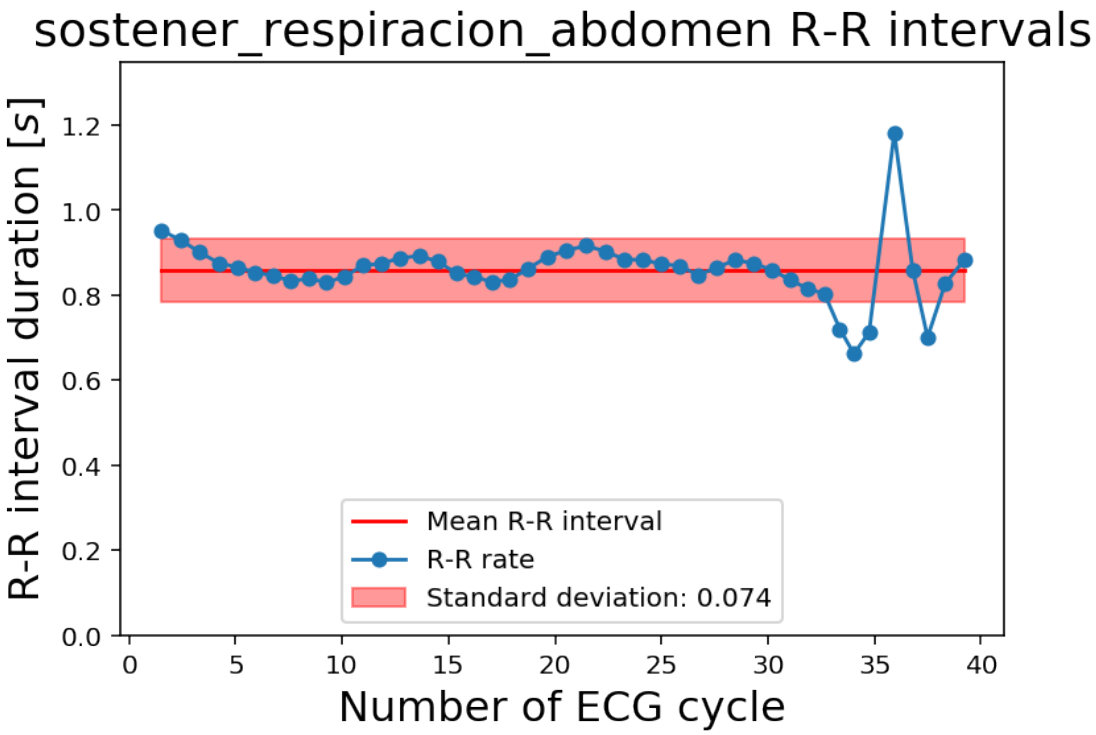

respiraciones_profundas_55_pecho R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.161

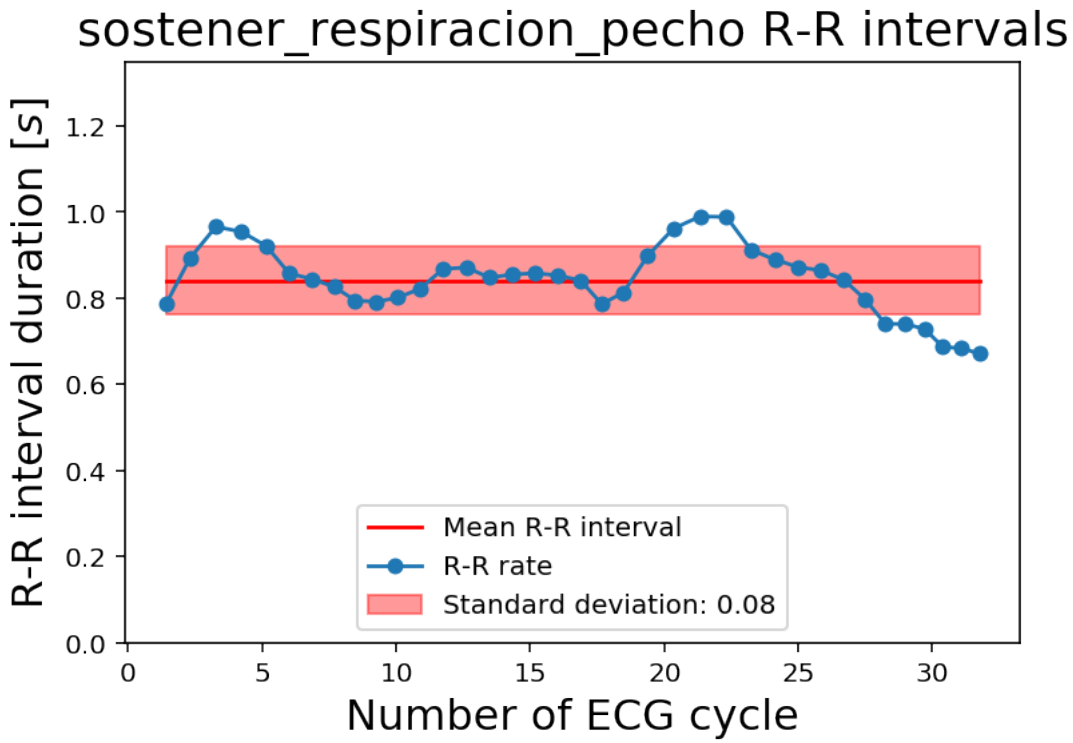

respiraciones_profundas_55_pecho R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.146

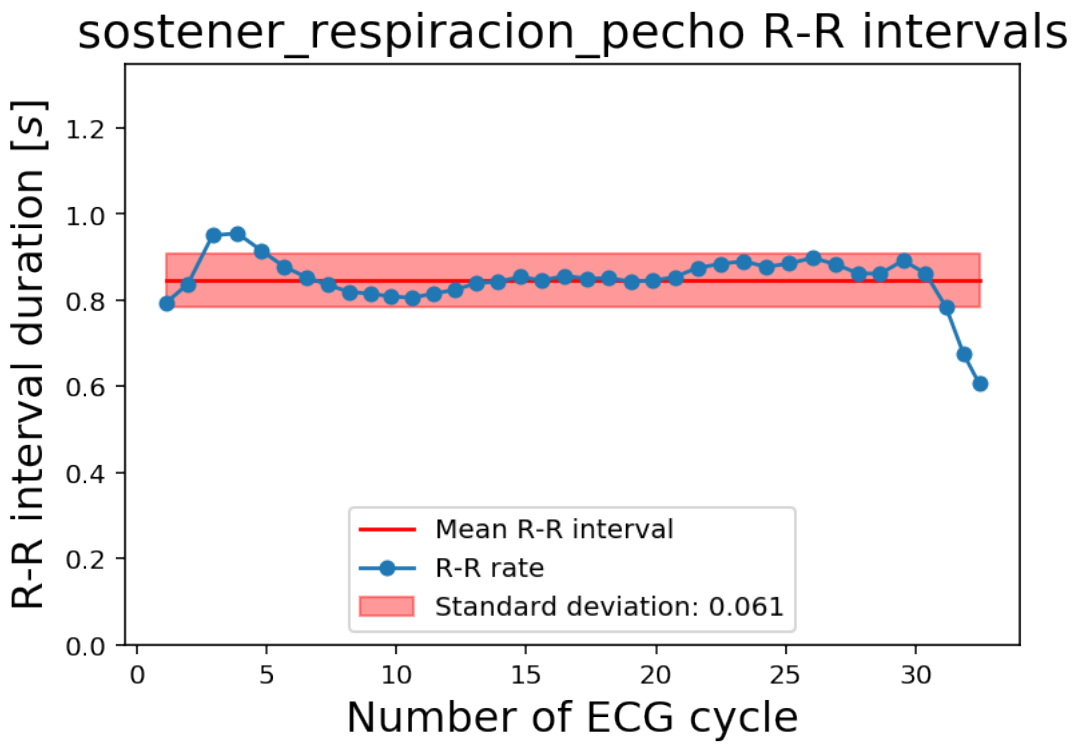

respiraciones_profundas_55_pecho R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.146

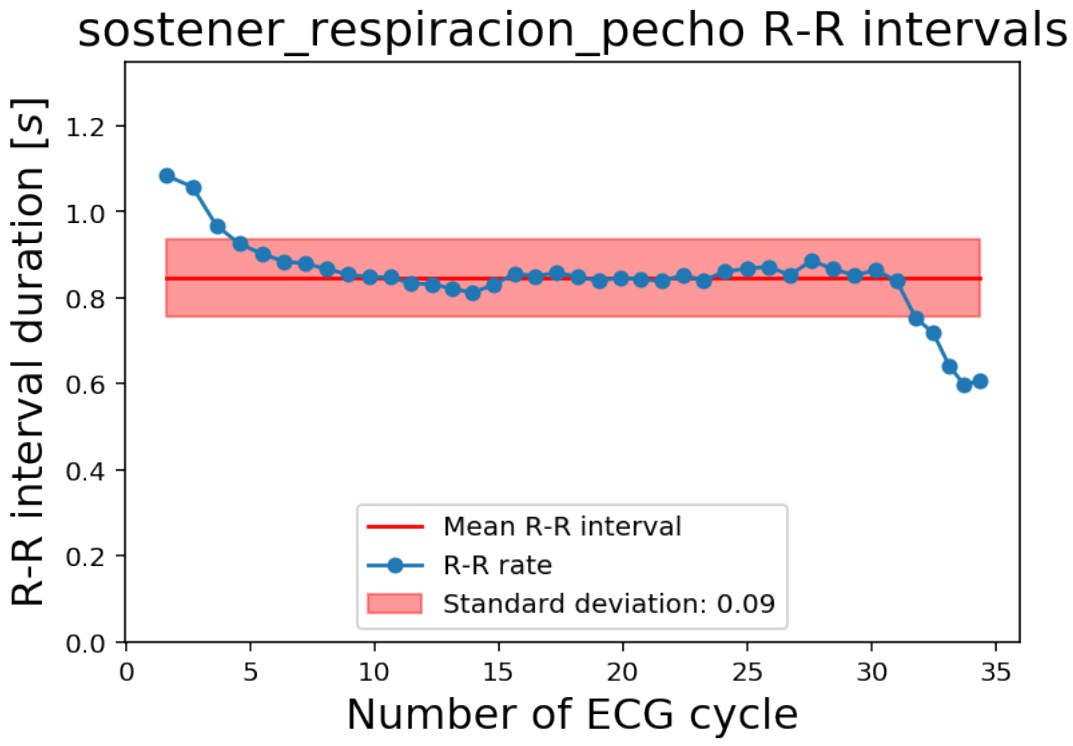

sostener_respiracion_abdomen R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.078

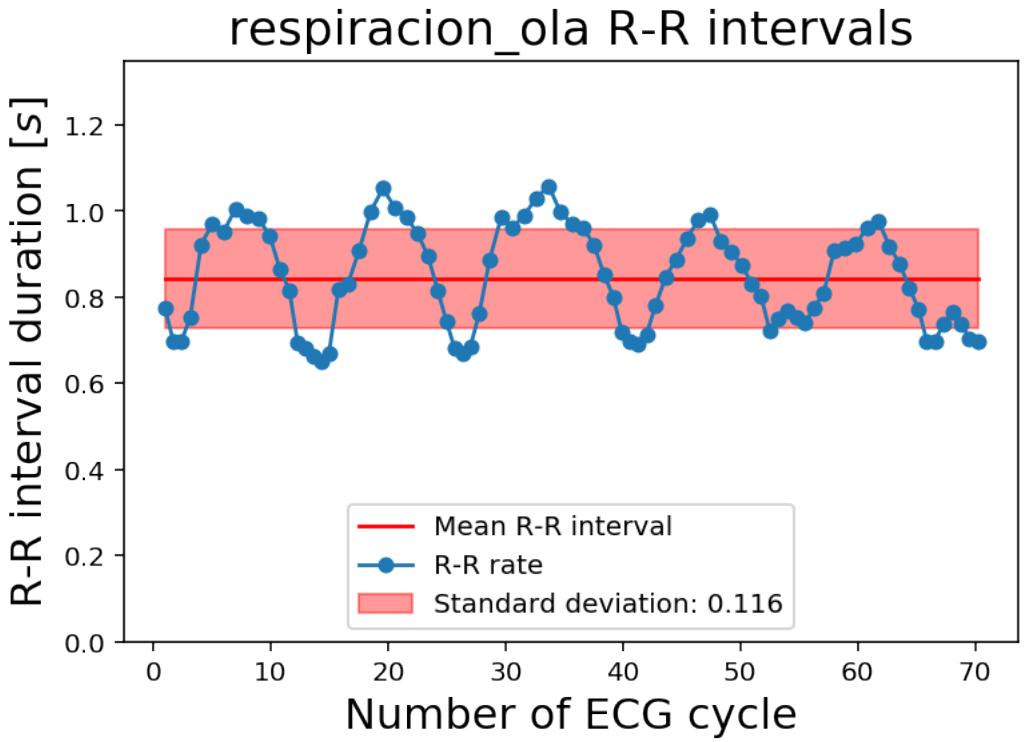

sostener_respiracion_abdomen R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.082

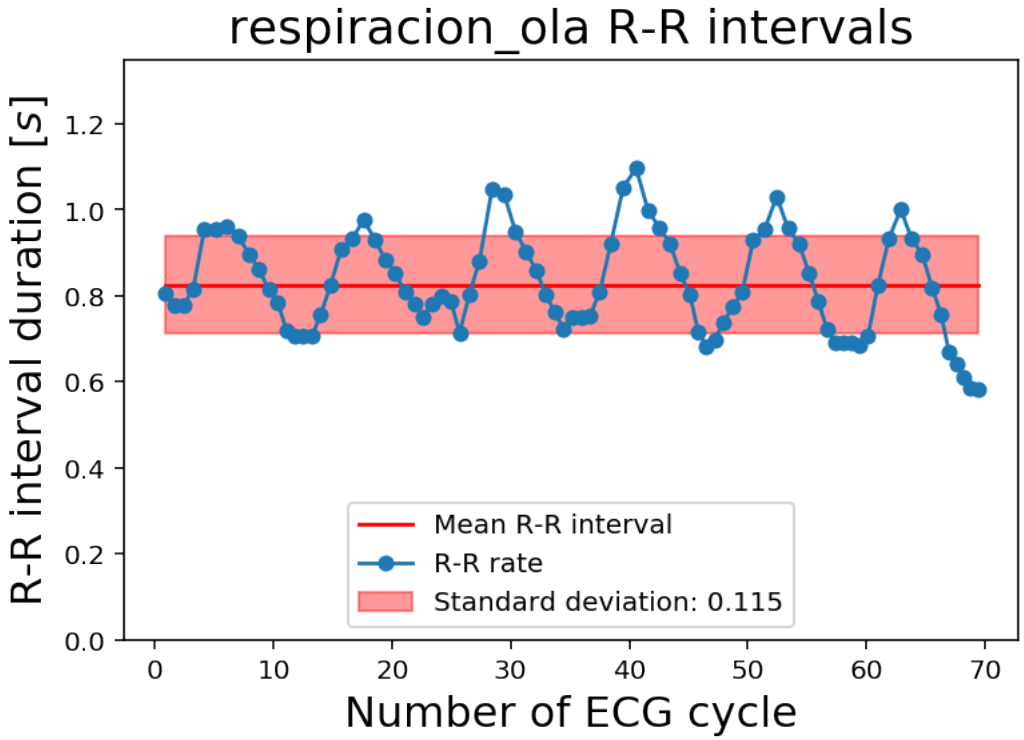

sostener_respiracion_abdomen R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.074

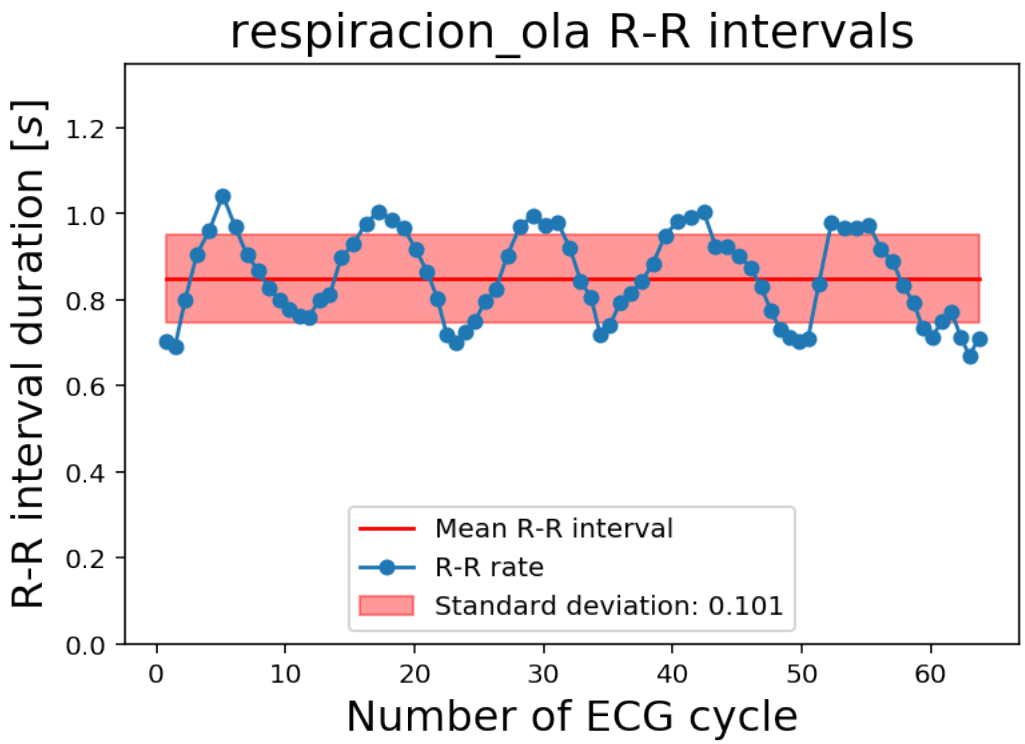

sostener_respiracion_pecho R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.08


sostener_respiracion_pecho R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.061


sostener_respiracion_pecho R-R intervals
R-R interval duration [s]
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.09


respiracion_ola R-R intervals
R-R interval duration [s]
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.116


respiracion_ola R-R intervals
R-R interval duration [s]
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
Number of ECG cycle
Mean R-R interval
R-R rate
Standard deviation: 0.115



Notice how the R-R rate is modulated by the breathing frequency. Besides, notice the standard deviation. When comparing the control to meditation techniques without controlling breathing frequency, the standard deviation is greater. It seems there is no substantial change when comparing control with meditation techniques of holding the breath.

[ ]:

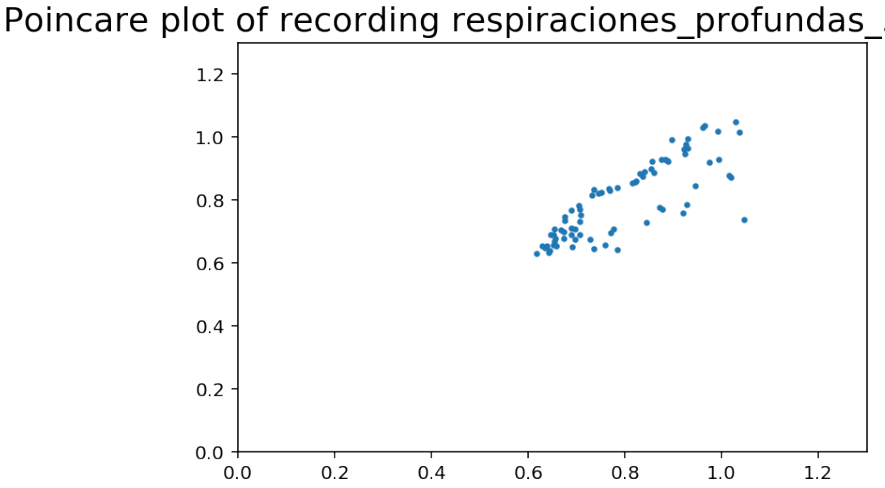
[ ]:

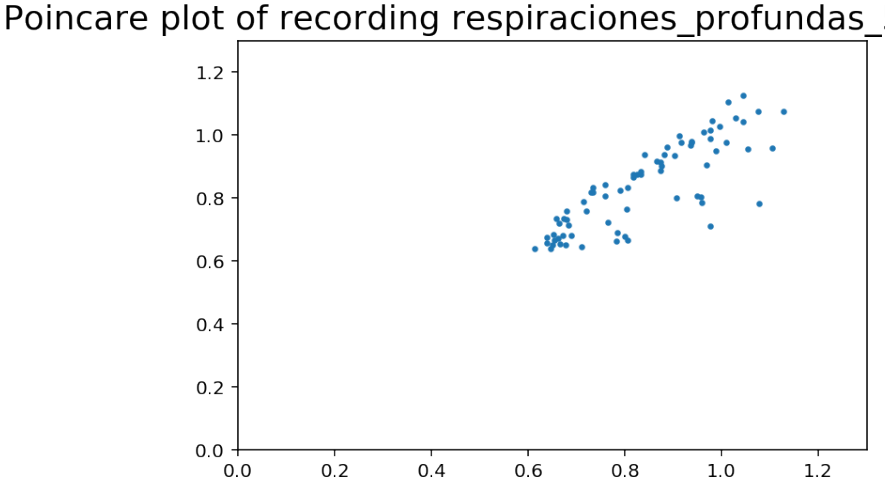
5 Poincaré plots

We already have the `mxs_indices` which correspond to the R-peaks, and `RR` corresponds to the R intervals. With this information, we can make a plot where we compare each one of these peaks to the next one. As this technique uses the R-R interval time series, we will only use recordings of subjects 1, 2 and 3.

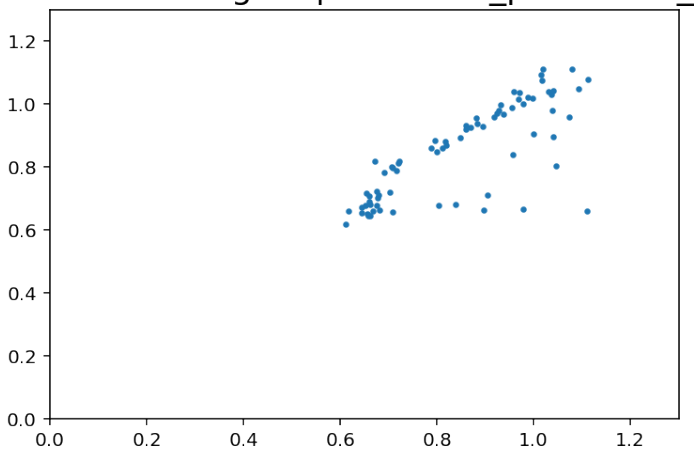
[10]:
```
for i in range(len(recordings)):
    plt.scatter(recordings[i].RR[1:-1], recordings[i].RR[:-2], s = 5)
    plt.title("Poincare plot of recording " + corresponding_folder[i])
    plt.xlim(0,1.3)
```

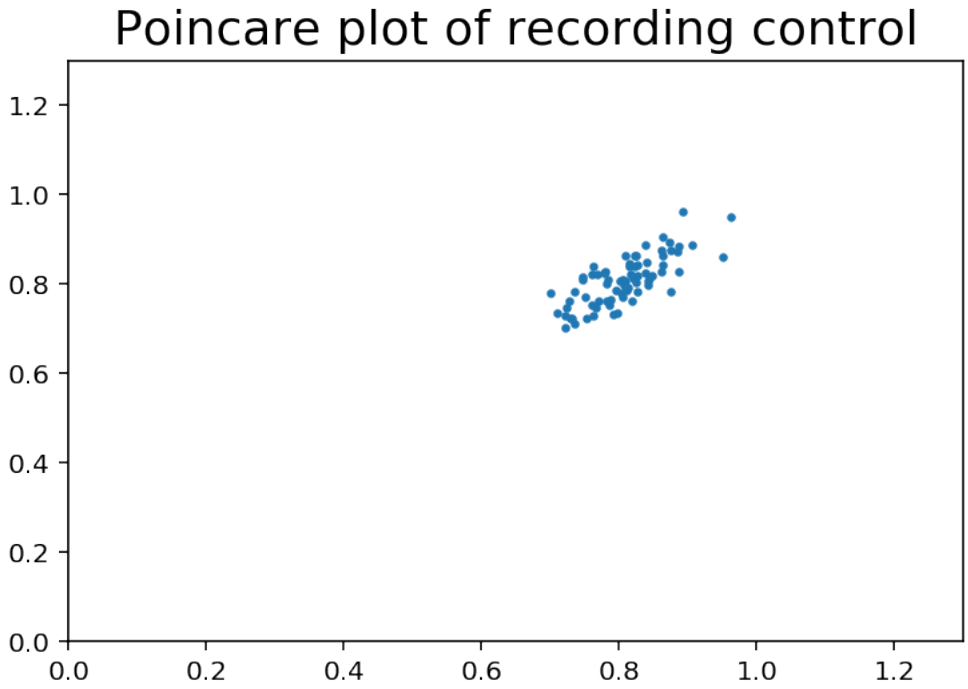
```
    plt.ylim(0,1.3)
    plt.show()
```

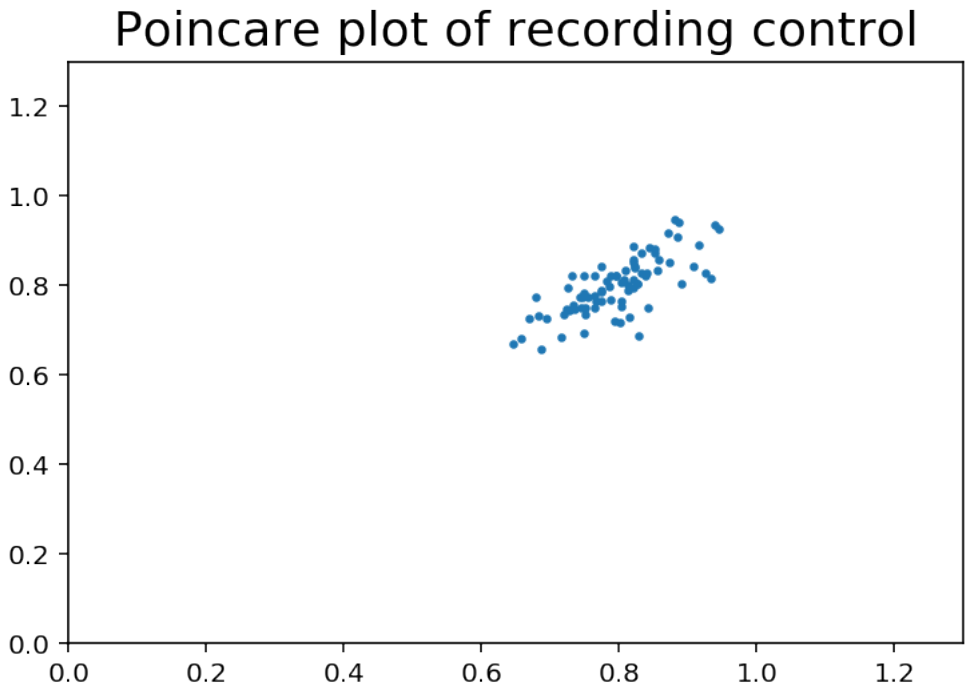


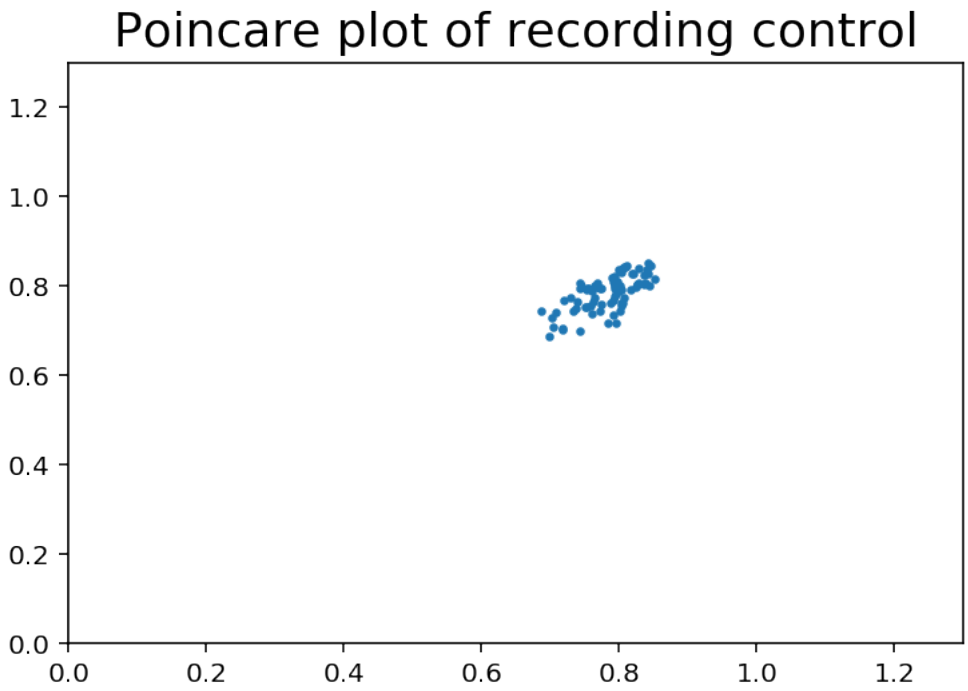


Poincare plot of recording respiraciones_profundas_55_abdomen

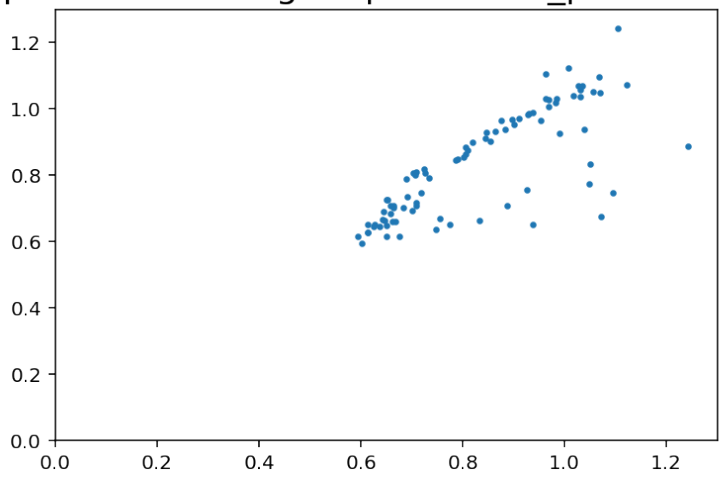




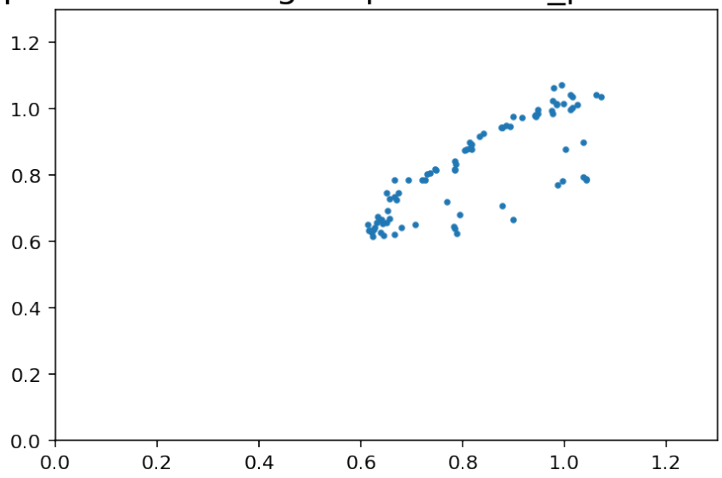

Poincare plot of recording control
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Poincare plot of recording control
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2

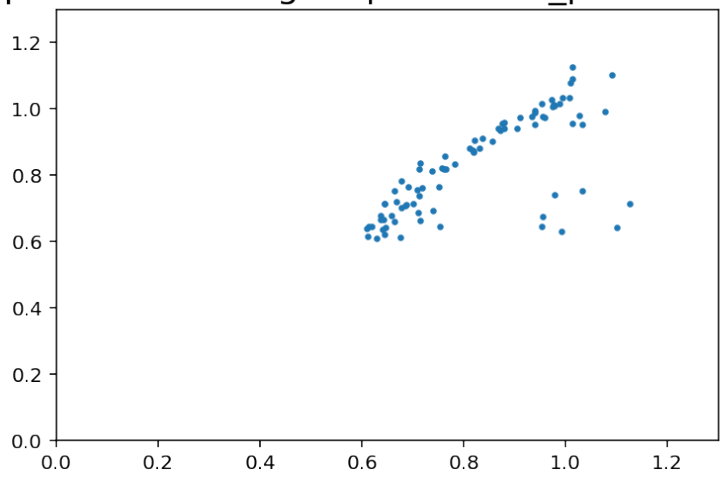
Poincare plot of recording respiraciones_profundas_55_pecho



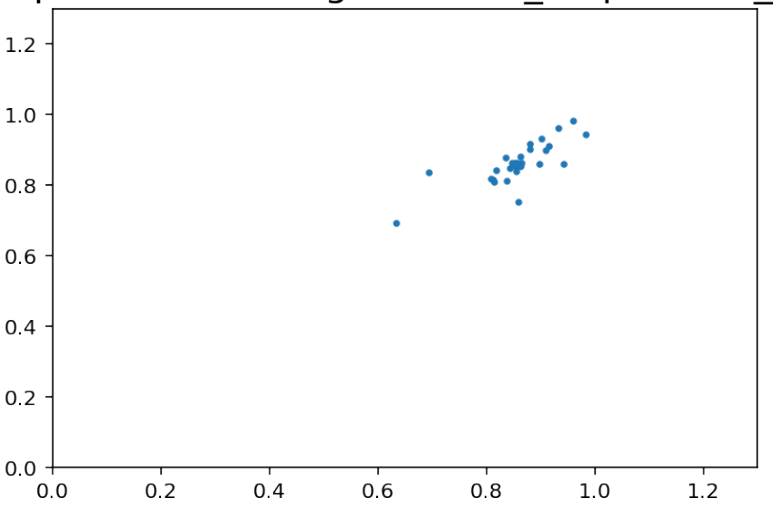
Poincare plot of recording respiraciones_profundas_55_pecho



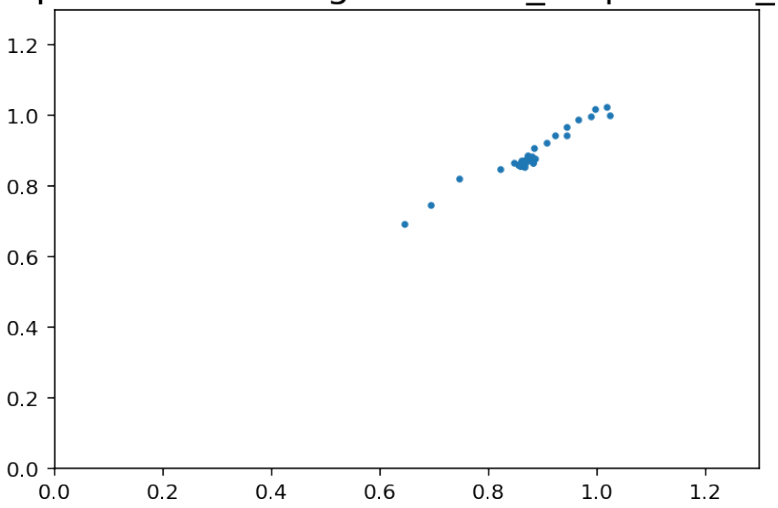
Poincare plot of recording respiraciones_profundas_55_pecho



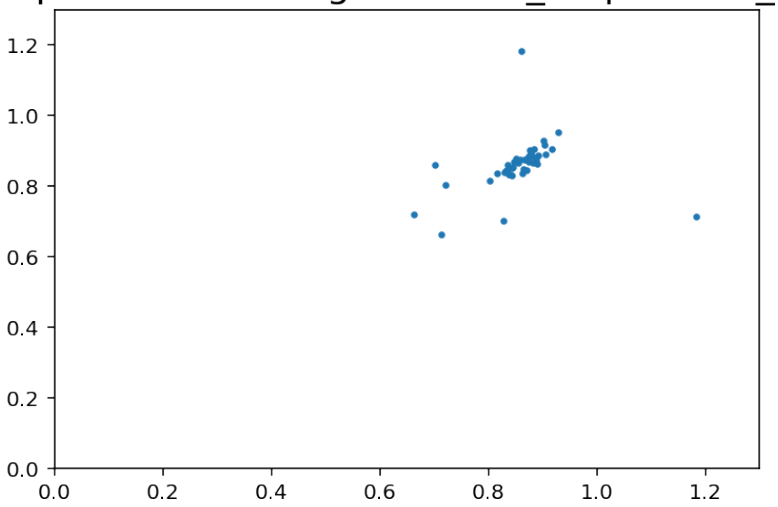
Poincare plot of recording sostener_respiracion_abdomen

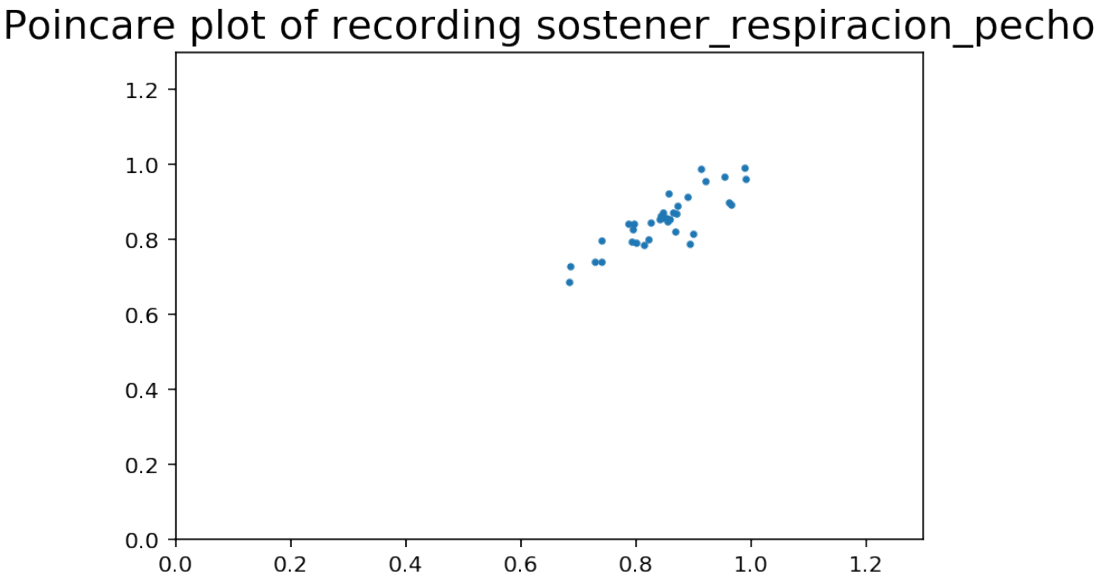


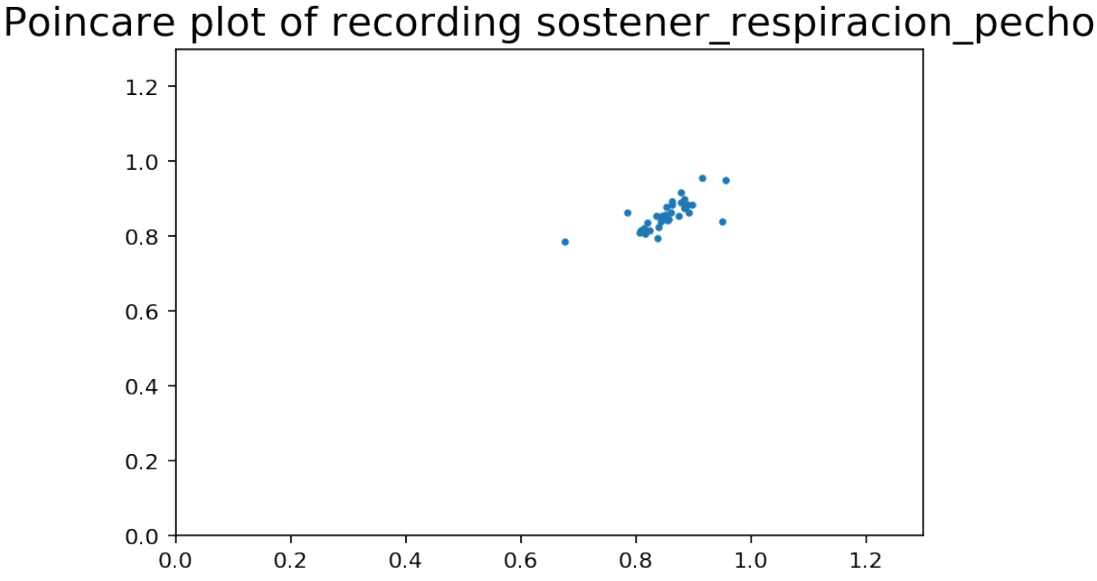
Poincare plot of recording sostener_respiracion_abdomen

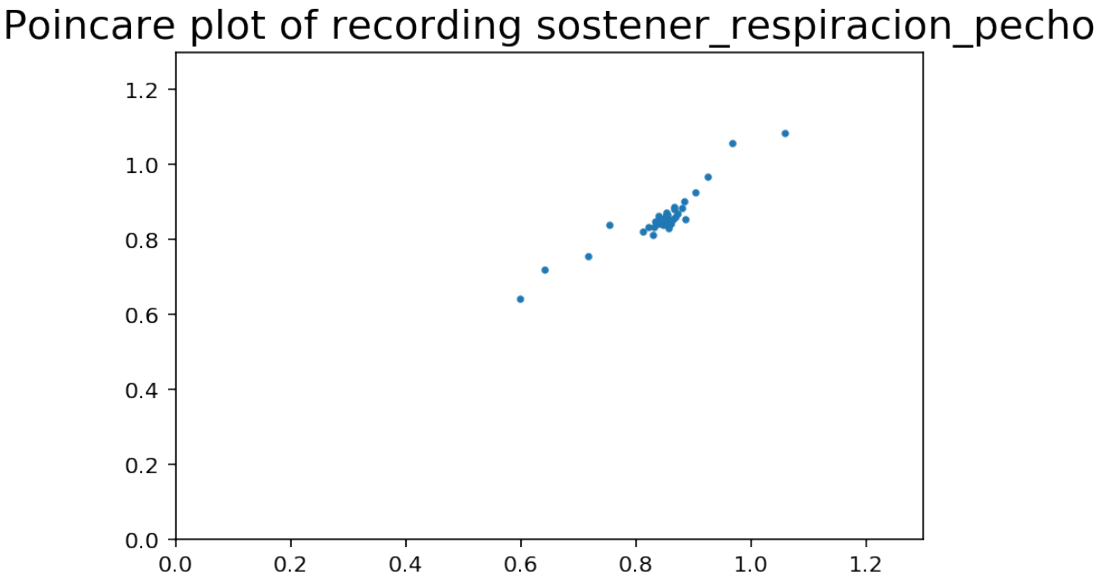


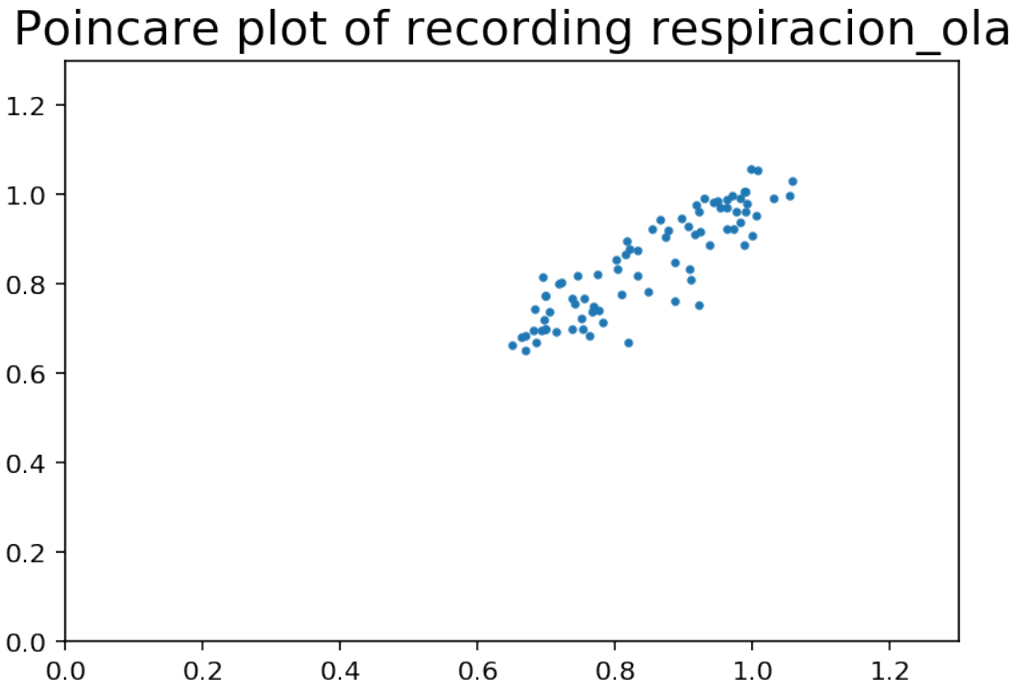
Poincare plot of recording sostener_respiracion_abdomen

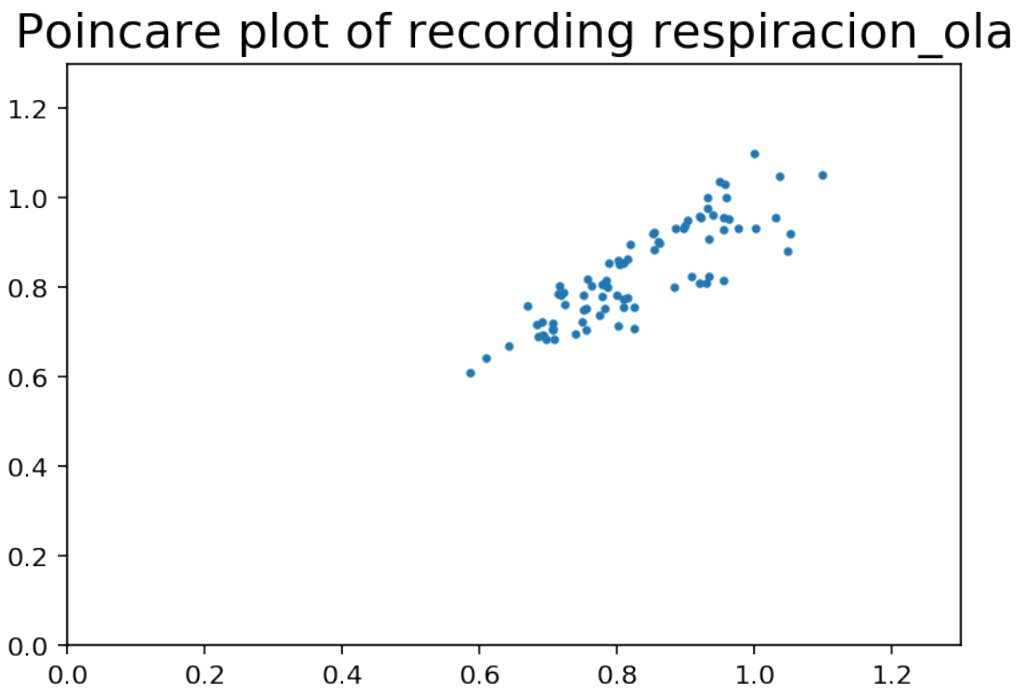


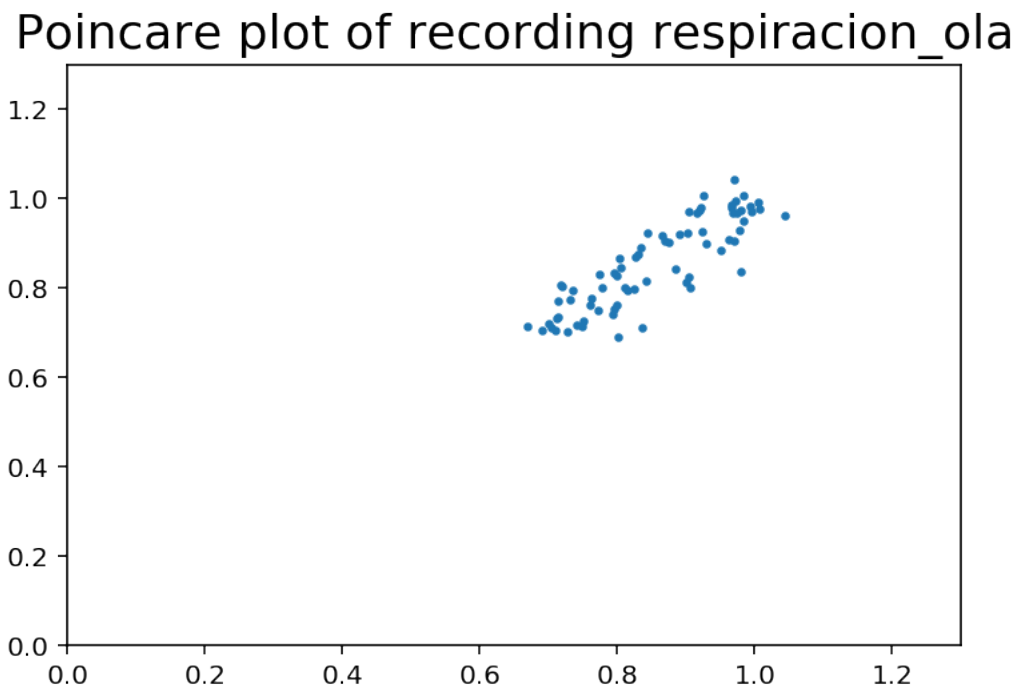

Poincare plot of recording sostener_respiracion_pecho
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Poincare plot of recording sostener_respiracion_pecho
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Poincare plot of recording sostener_respiracion_pecho
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2

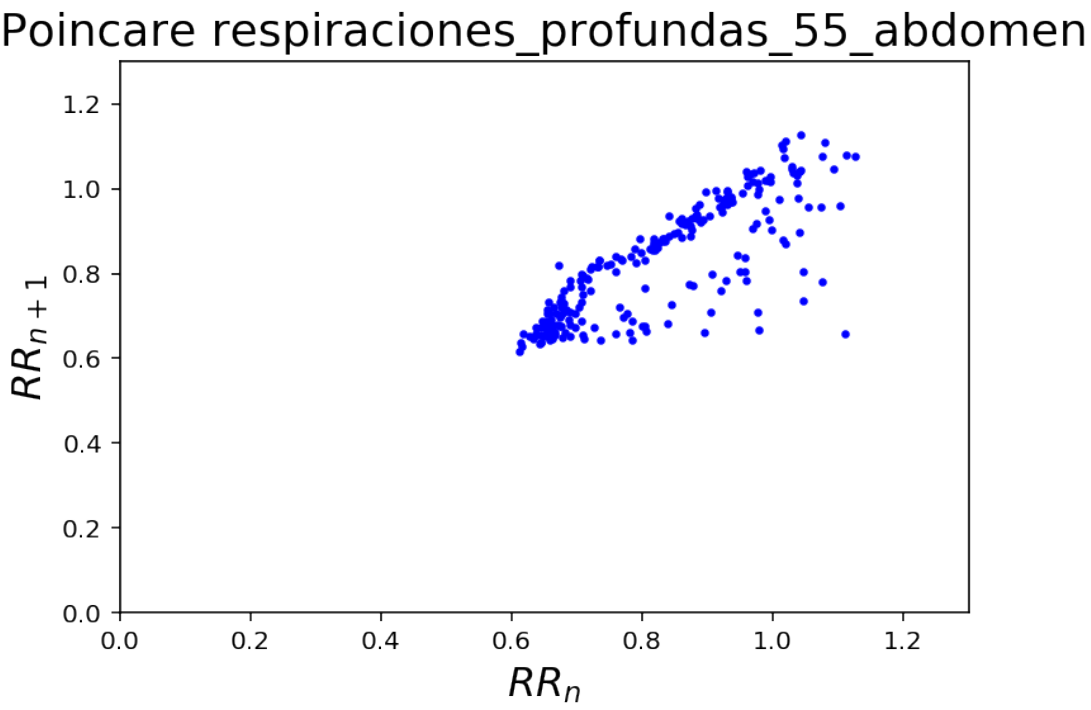

Poincare plot of recording respiracion_ola
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2

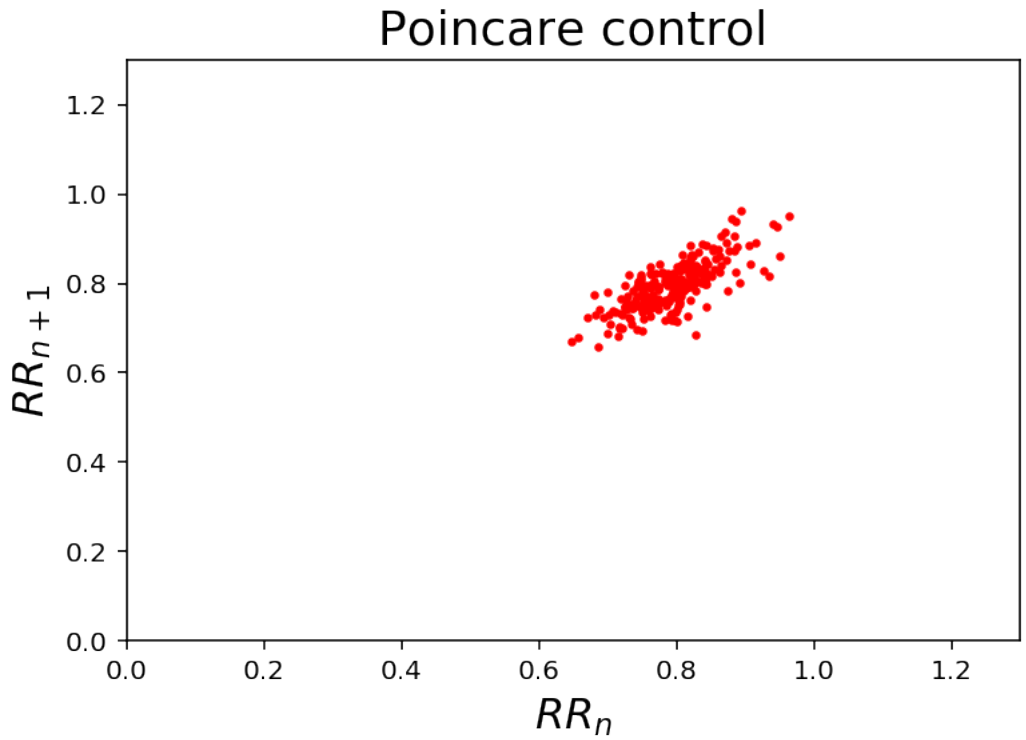


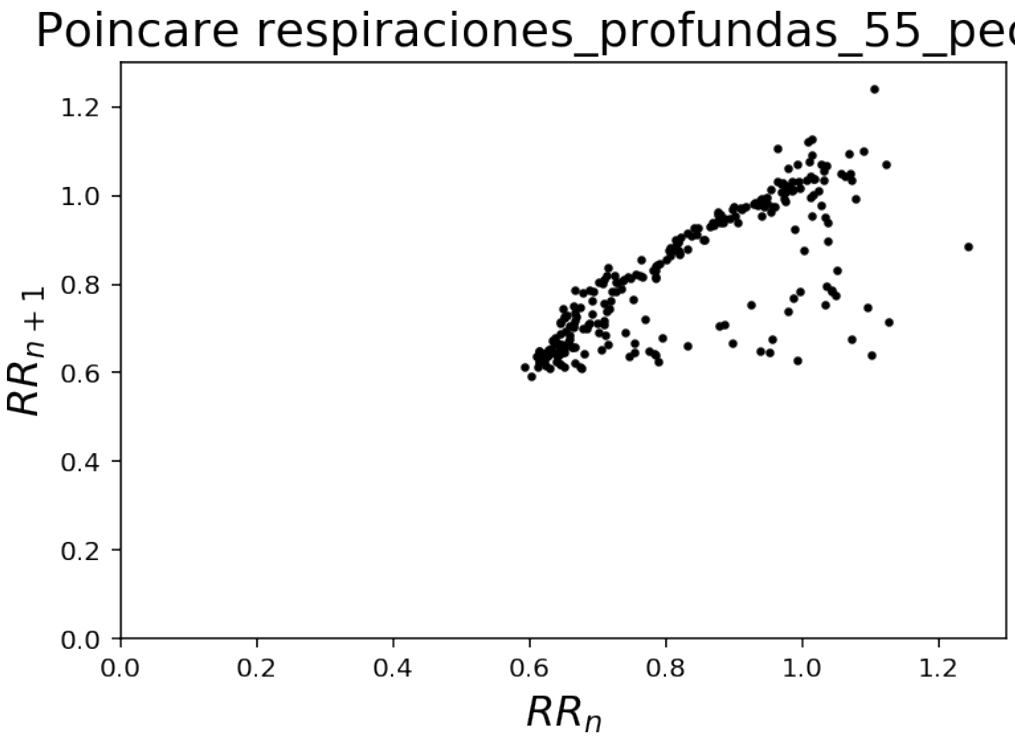


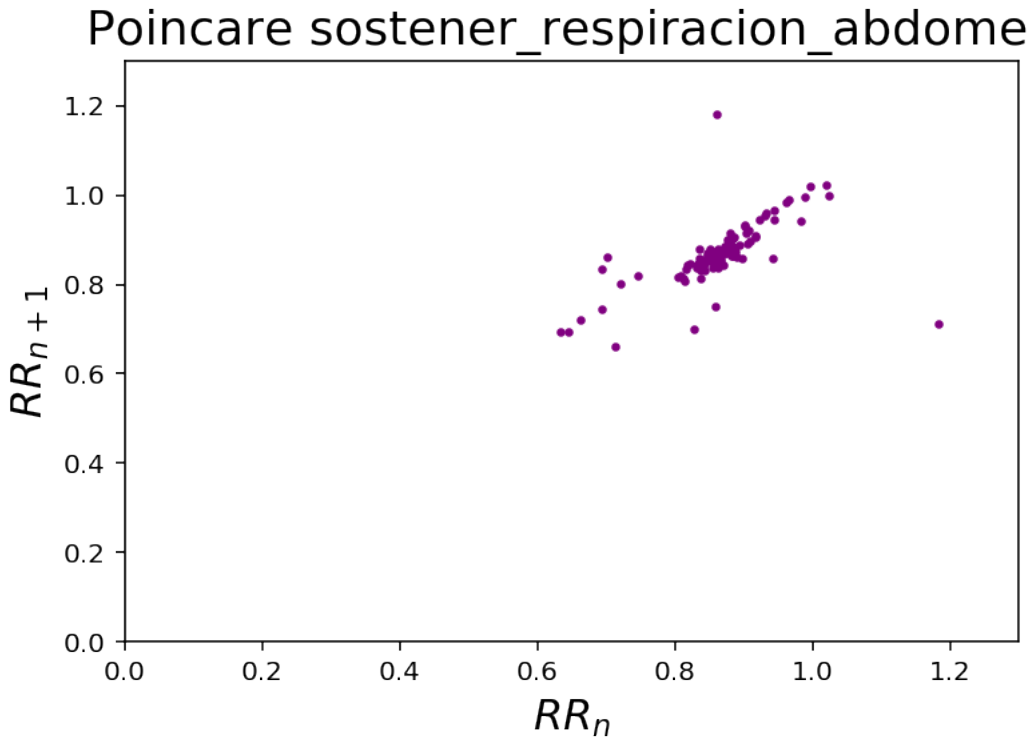
What if we plot the same kind of recording with the same color and compare between different types of recordings?

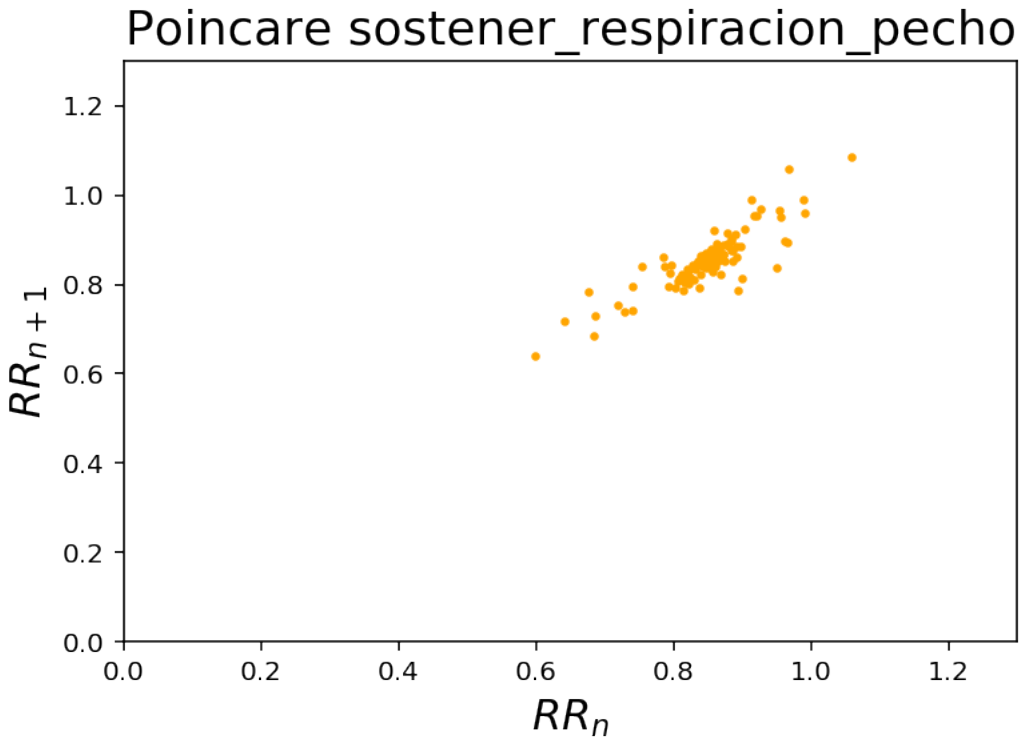
[11]:
```
colors = ['blue', 'red', 'black', 'purple', "orange", "green"]
color_index = 0

for i in range(len(recordings)):

    if i!= 0 and corresponding_folder[i] != corresponding_folder[i-1]:
        color_index = color_index+1
        plt.title("Poincare "+str(corresponding_folder[i-1]))
        plt.xlim(0,1.3)
        plt.ylim(0,1.3)
        plt.xlabel(r"$RR_n$")
        plt.ylabel(r"$RR_{n+1}$")
        plt.savefig('meditation_imgs/poincare_'+str(corresponding_folder[i])+'.
 ↪jpg')#Saves images in folder

        plt.show()

    plt.scatter(recordings[i].RR[1:-1], recordings[i].RR[:-2],
                s = 5, c = colors[color_index])

plt.title("Poincare "+str(corresponding_folder[i-1]))
plt.xlim(0,1.3)
plt.ylim(0,1.3)
plt.xlabel(r"$RR_n$")
plt.ylabel(r"$RR_{n+1}$")
plt.savefig('meditation_imgs/poincare_'+str(corresponding_folder[i])+'.
 ↪jpg')#Saves images in folder
plt.show()
```

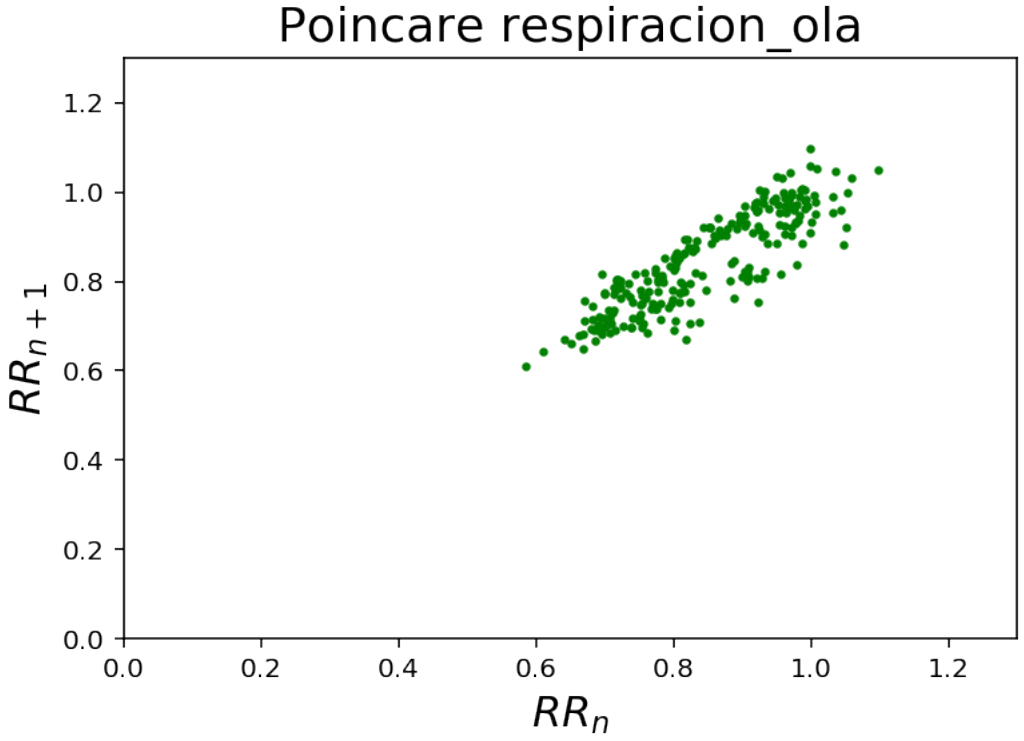

Poincare respiraciones_profundas_55_abdomen
RR_{n+1}
RR_n
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Poincare control
RR_{n+1}
RR_n
0.0
0.2
0.4
0.6
0.8
1.0
1.2

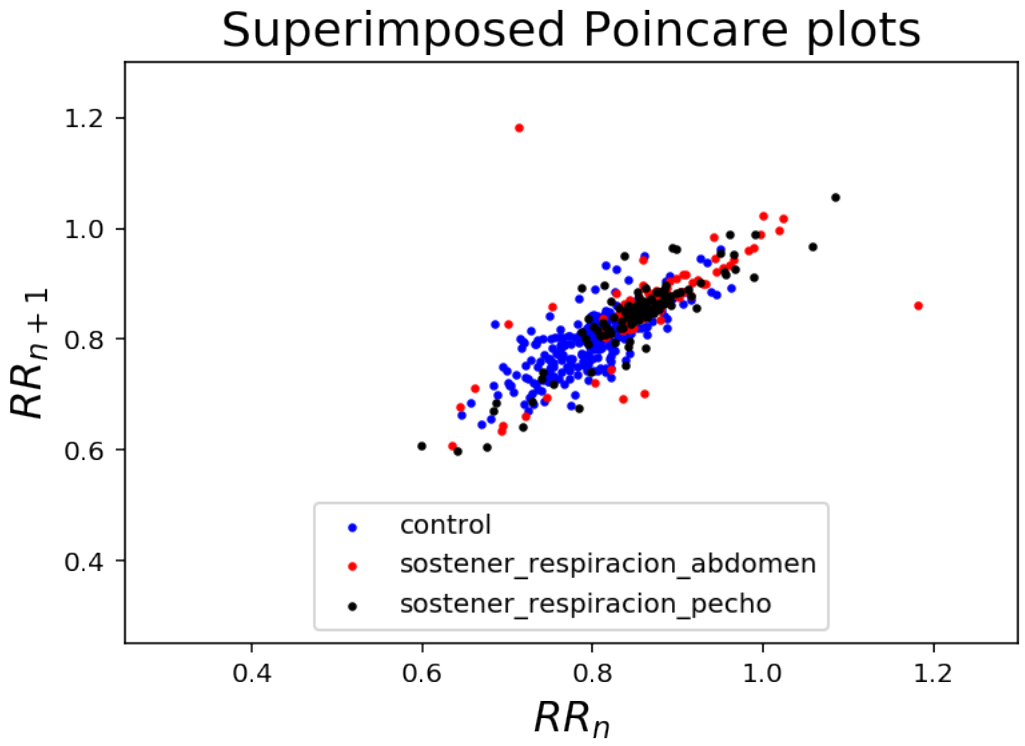

Poincare respiraciones_profundas_55_pecho
RR_{n+1}
RR_n
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Poincare sostener_respiracion_abdomen
RR_{n+1}
RR_n
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Poincare sostener_respiracion_pecho
RR_{n+1}
RR_n
0.0
0.2
0.4
0.6
0.8
1.0
1.2

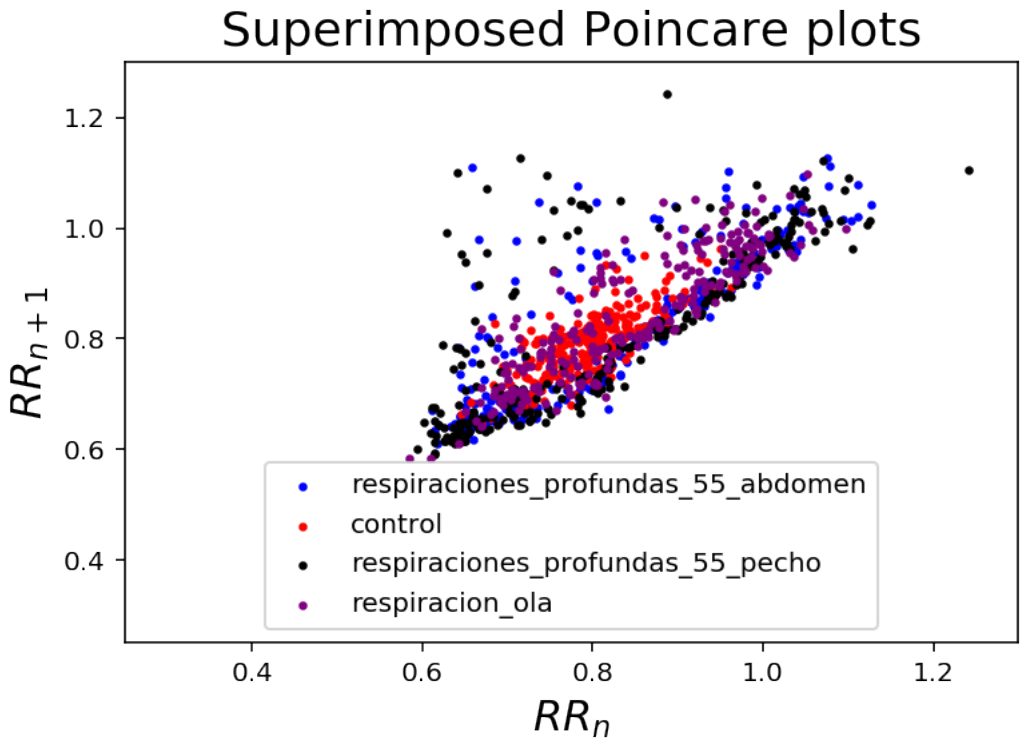

Poincare respiracion_ola
RR_{n+1}
RR_n
0.0
0.2
0.4
0.6
0.8
1.0
1.2

[12]:
```
# Comparison between control, holding breath in chest and holding breath in␣
 ↪abdomen
colors = ['blue', 'red', 'black', 'purple', "orange", "green"]
color_index = 0
banner = False
time_delay = 1

for i in range(len(recordings)):
    if corresponding_folder[i]=="control" or "sostener" in␣
 ↪corresponding_folder[i]:
        n = np.size(recordings[i].RR) #size of the voltage vector

        if banner == False: #First event
            plt.scatter(recordings[i].RR[0: n-time_delay], recordings[i].
 ↪RR[time_delay: n],
                        s = 5, c = colors[color_index], label =␣
 ↪corresponding_folder[i]) #plots label
        else: #Other events, does not plot label
            plt.scatter(recordings[i].RR[0: n-time_delay], recordings[i].
 ↪RR[time_delay: n],
                s = 5, c = colors[color_index])

        banner = True

    if len(recordings) != i:
        if banner == True and corresponding_folder[i] !=␣
 ↪corresponding_folder[i+1]:
            color_index = color_index+1
            banner = False

plt.title("Superimposed Poincare plots")
plt.xlim(0.25,1.3)
plt.ylim(0.25,1.3)
plt.xlabel(r"$RR_n$")
plt.ylabel(r"$RR_{n+1}$")
plt.legend(loc='lower center')
plt.savefig('meditation_imgs/poincare_superimposed_sostener.jpg')#Saves images␣
 ↪in folder
plt.show()
```



[13]:
```
# Comparar: control, respiraciones profundas 55 pecho, respiraciones profundas␣
↪55 abdomen, respiraciones en ola.

colors = ['blue', 'red', 'black', 'purple', "orange", "green"]
color_index = 0
banner = False
time_delay = 1

for i in range(len(recordings)):
    if corresponding_folder[i]=="control" or "55" in corresponding_folder[i] or␣
↪"ola" in corresponding_folder[i]:
        n = np.size(recordings[i].RR) #size of the voltage vector

        if banner == False: #First event
            plt.scatter(recordings[i].RR[0: n-time_delay], recordings[i].
↪RR[time_delay: n],
                        s = 5, c = colors[color_index], label =␣
↪corresponding_folder[i]) #plots label
        else: #Other events, does not plot label
            plt.scatter(recordings[i].RR[0: n-time_delay], recordings[i].
↪RR[time_delay: n],
                s = 5, c = colors[color_index])
```

```
        banner = True

    if len(recordings)-1 != i:
        if banner == True and corresponding_folder[i] !=␣
 ↪corresponding_folder[i+1]:
            color_index = color_index+1
            banner = False

plt.title("Superimposed Poincare plots")
plt.xlim(0.25,1.3)
plt.ylim(0.25,1.3)
plt.xlabel(r"$RR_n$")
plt.ylabel(r"$RR_{n+1}$")
plt.legend(loc='lower center')
plt.savefig('meditation_imgs/poincare_superimposed_respirar.jpg')#Saves images␣
 ↪in folder
plt.show()
```



Notice how Techniques respiraciones profundas 55 pecho, respiraciones profundas 55 abdomen, respiraciones en ola make the R R intervals change. Very interesting what happens to the heartbeat while meditating!

Interpretación: Al someter al cuerpo bajo un esfuerzo físico como aguantar la resipración, los

intervalos R-R Se vuelven más regulares y el corazón comienza a asemejarse a un reloj. Tanto sus intervalos entre latidos se vuelven más regulares y aumenta la frecuencia de los picos R.

[ ]:

[ ]:

[ ]:

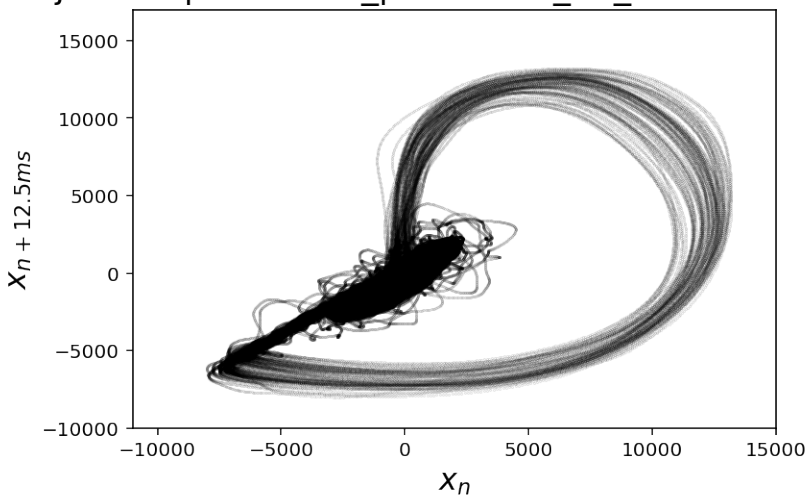
6 Phase space reconstruction

In the previos technique we only compared the R intervals with the next interval. What would happen if we compare not the R interval, but the whole recording with a certain time delay we choose. Instead of comparing one data point with its next in time, we choose an arbitrary time delay? In other words, we are going to generalize what we applied in the previous technique with the whole ECG data series and with an arbitrary time delay.

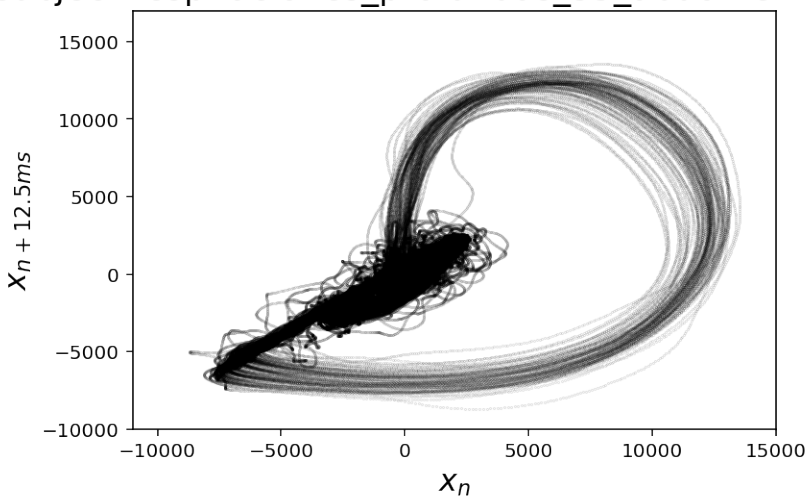
[14]:
```
#Generating a function that will reconstruct the phase space for a certain time␣
 ↪delay
# data_series is the voltage of our signal
# period is the time delay
# identifier is a string that will help us identify that particular graph

def graph_phase_space(waveData, period = 210, identifier = "xx"):
    time = period*0.1 #time is in miliseconds
    n = np.size(waveData) #size of the voltage vector

    plt.figure(2)
    plt. plot(waveData[0: n-period], waveData[period: n],
              marker = "o", markersize = 0.05, linewidth = 0.005, color =␣
 ↪"black")
    plt.title(identifier+r": $t+"+str(time)+"ms$")
    plt.xlabel(r"$x_n$")
    y = r'$x_{n+'+str(time)+"ms}$"
    plt.ylabel(y)
    plt.xlim(-11000, 15000)
    plt.ylim(-10000, 17000)
    plt.show()
    return None
```

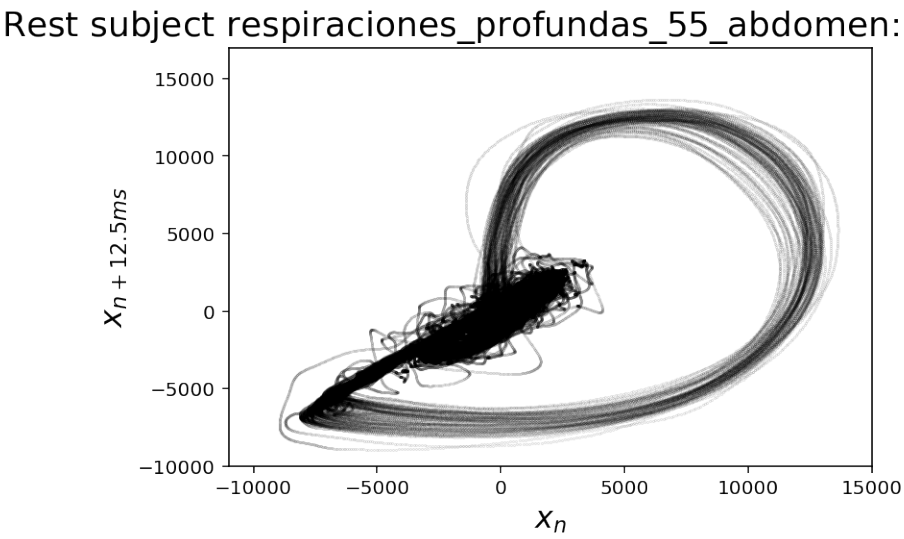
[15]:
```
time_delay = 125 #12.5 ms

for i in range(len(recordings)):
    identifier = "Rest subject " + corresponding_folder[i]
    graph_phase_space(recordings[i].waveData, time_delay, identifier)
```

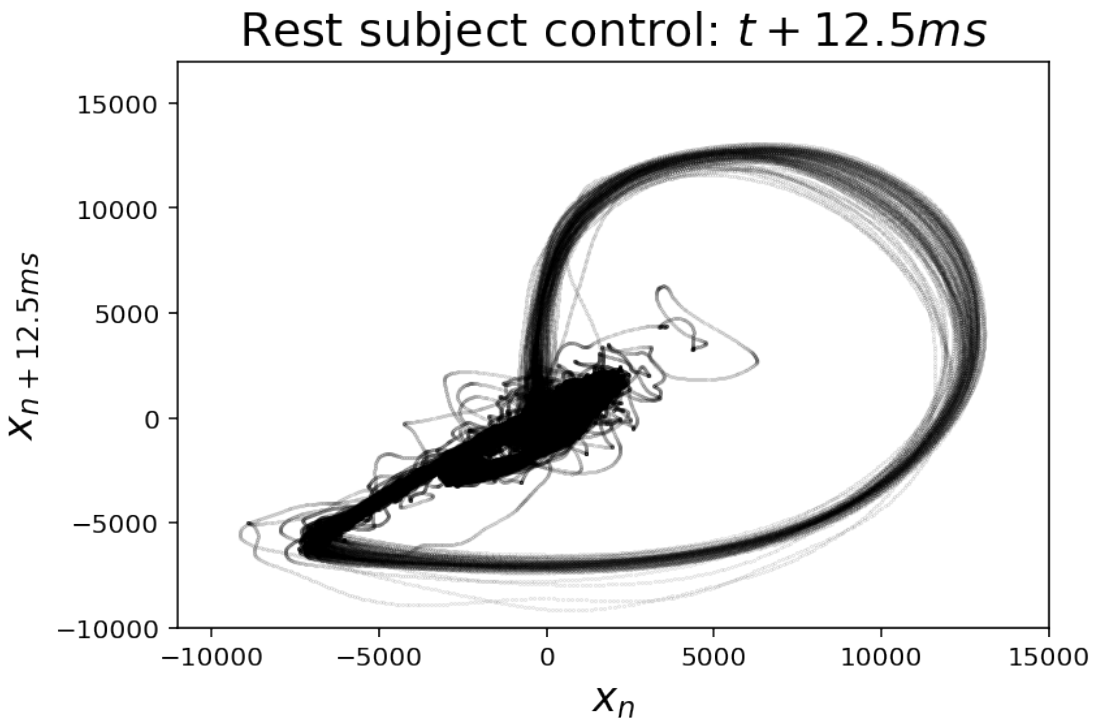
Rest subject respiraciones_profundas_55_abdomen: $t + 12.5ms$

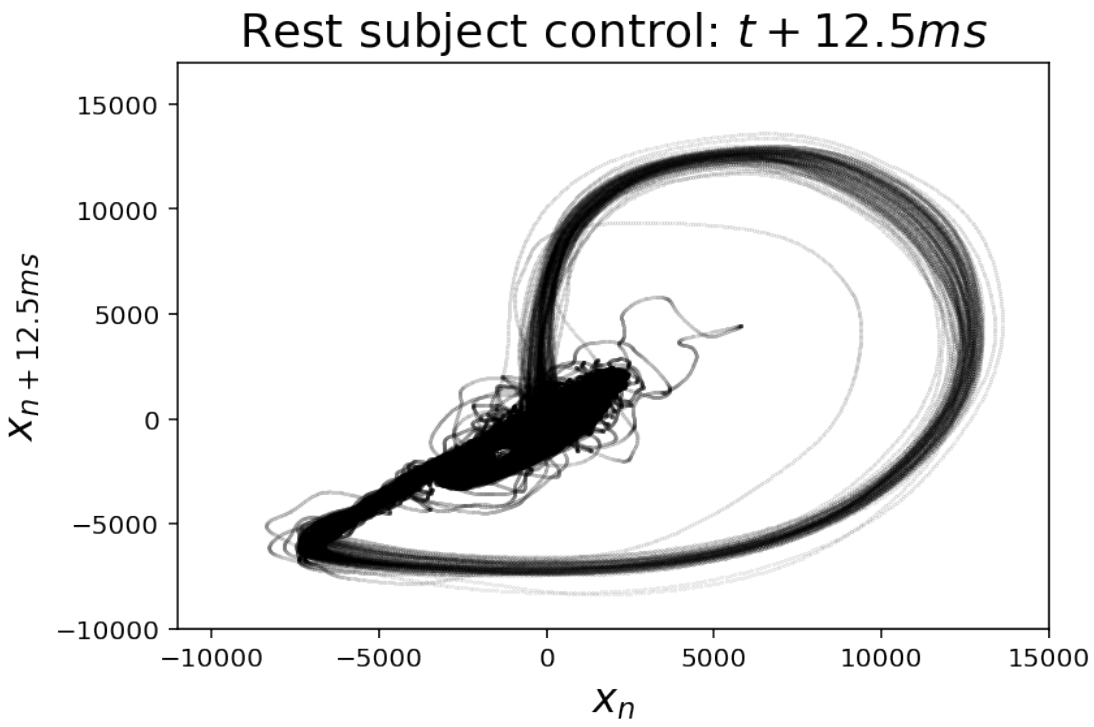


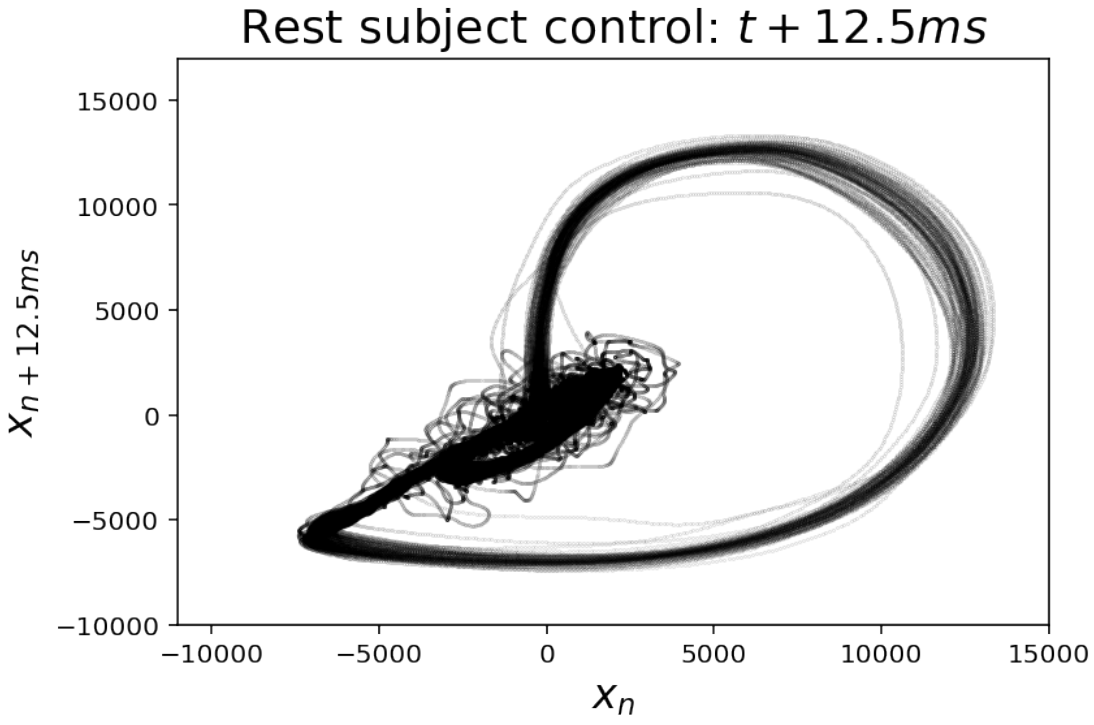
Rest subject respiraciones_profundas_55_abdomen: $t + 12.5ms$

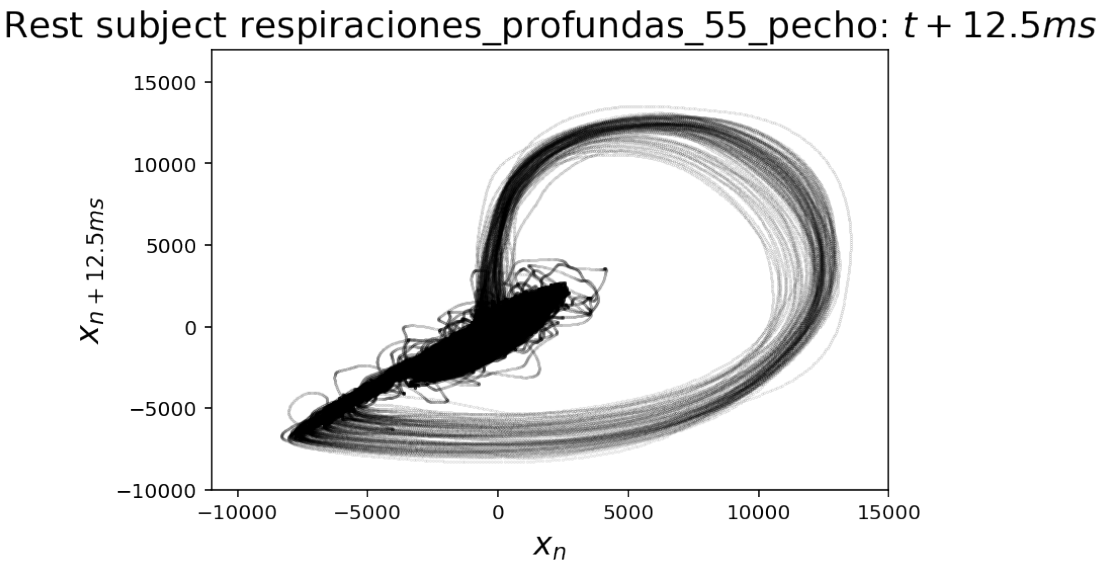


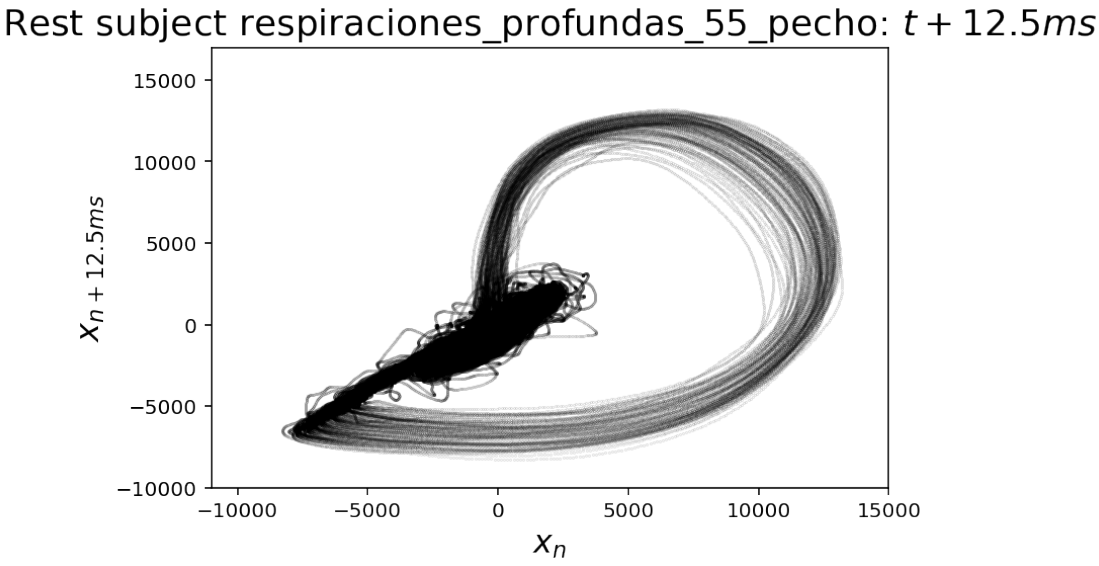

Rest subject respiraciones_profundas_55_abdomen: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$x_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
x_n


Rest subject control: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$X_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
x_n

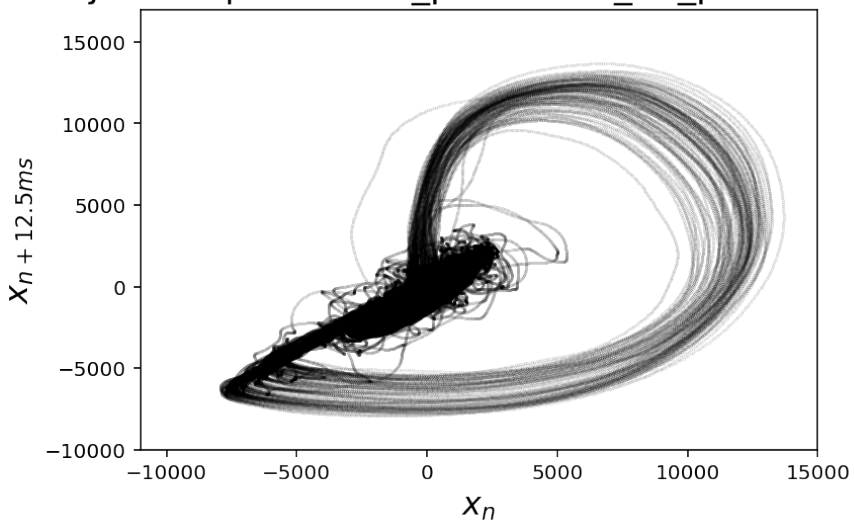

Rest subject control: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$X_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
X_n


Rest subject control: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$X_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
X_n

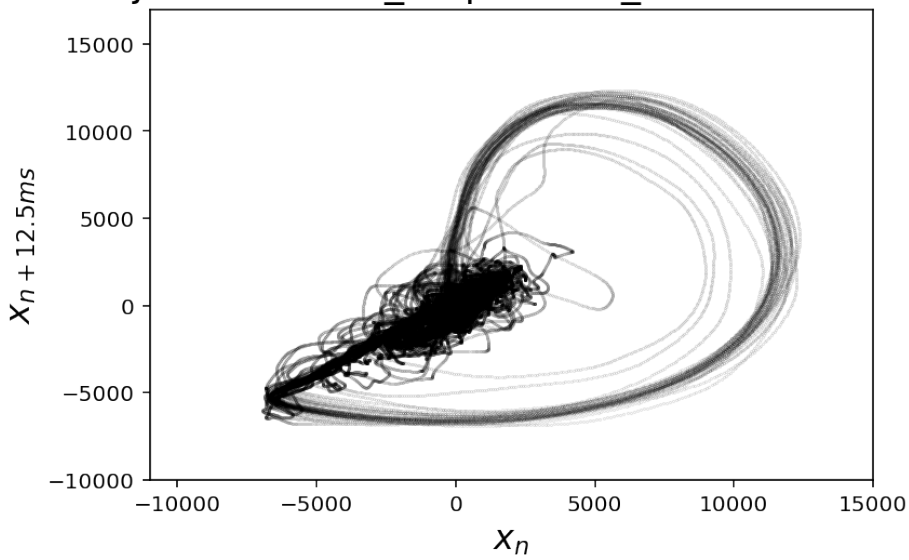

Rest subject respiraciones_profundas_55_pecho: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$X_n + 12.5ms$
X_n


Rest subject respiraciones_profundas_55_pecho: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$X_n + 12.5ms$
X_n

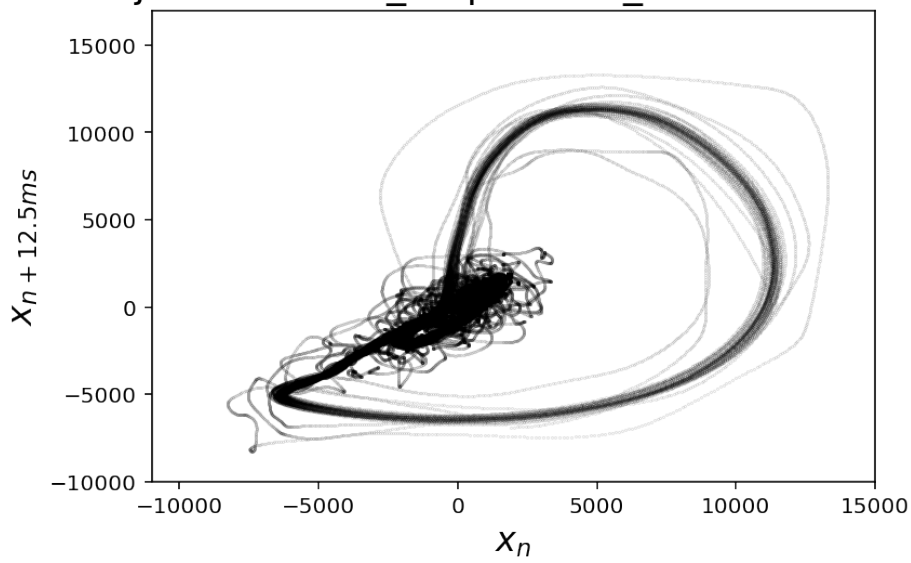
Rest subject respiraciones_profundas_55_pecho: $t + 12.5ms$



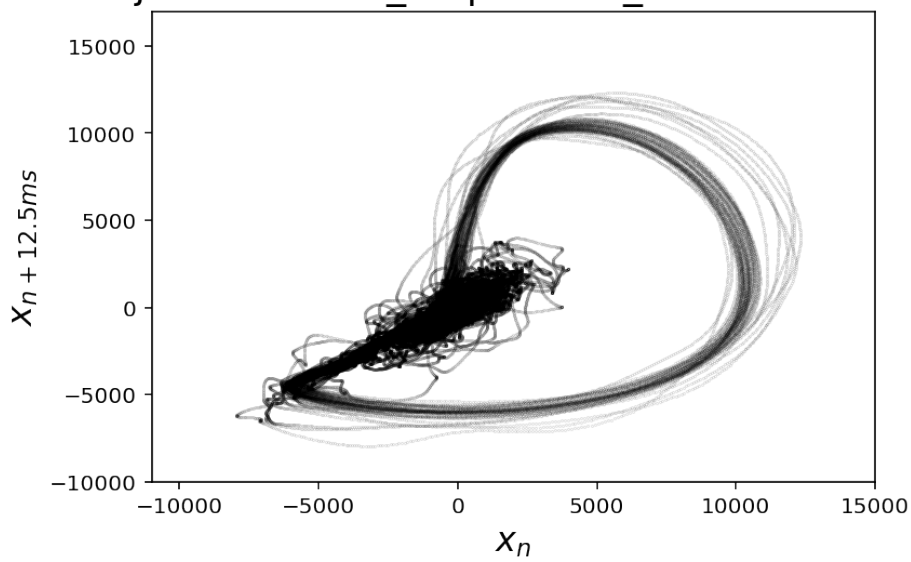
Rest subject sostener_respiracion_abdomen: $t + 12.5ms$

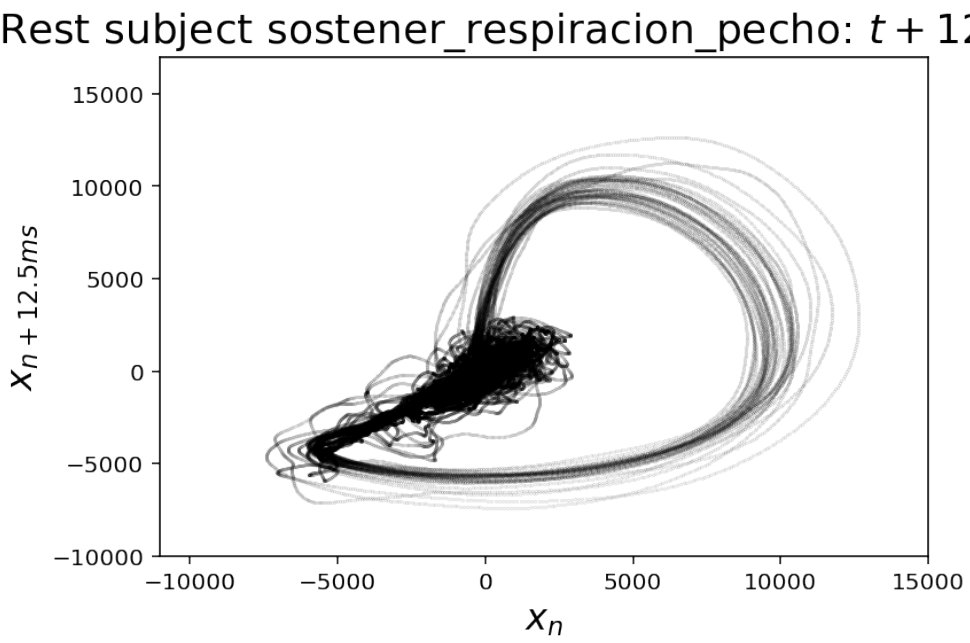


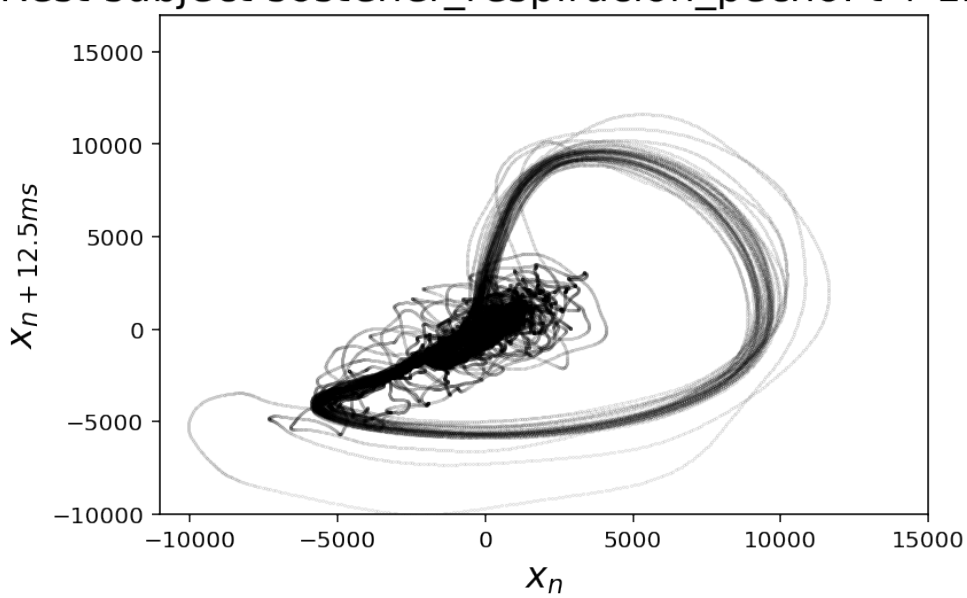
Rest subject sostener_respiracion_abdomen: $t + 12.5ms$

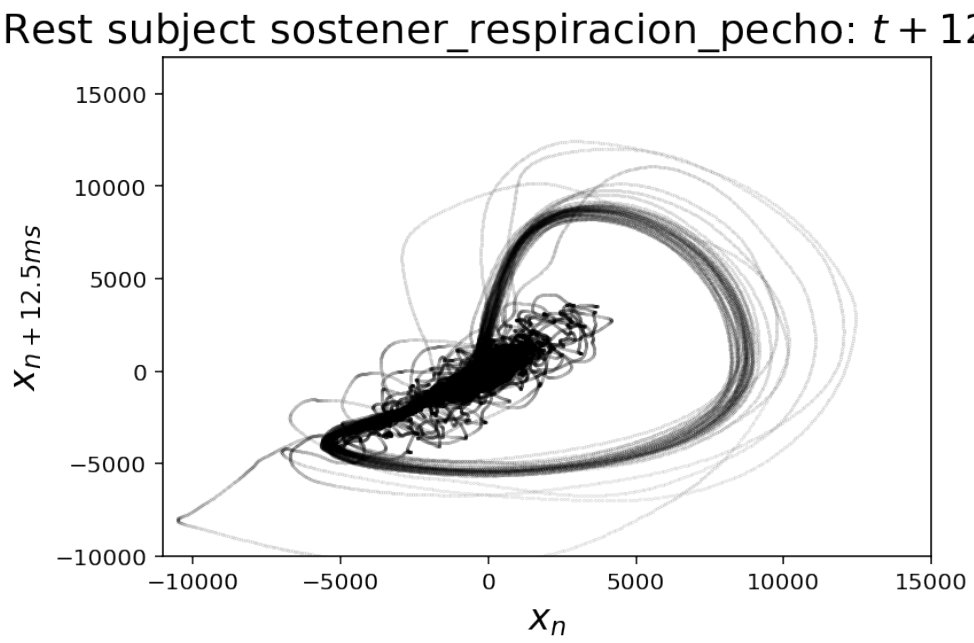


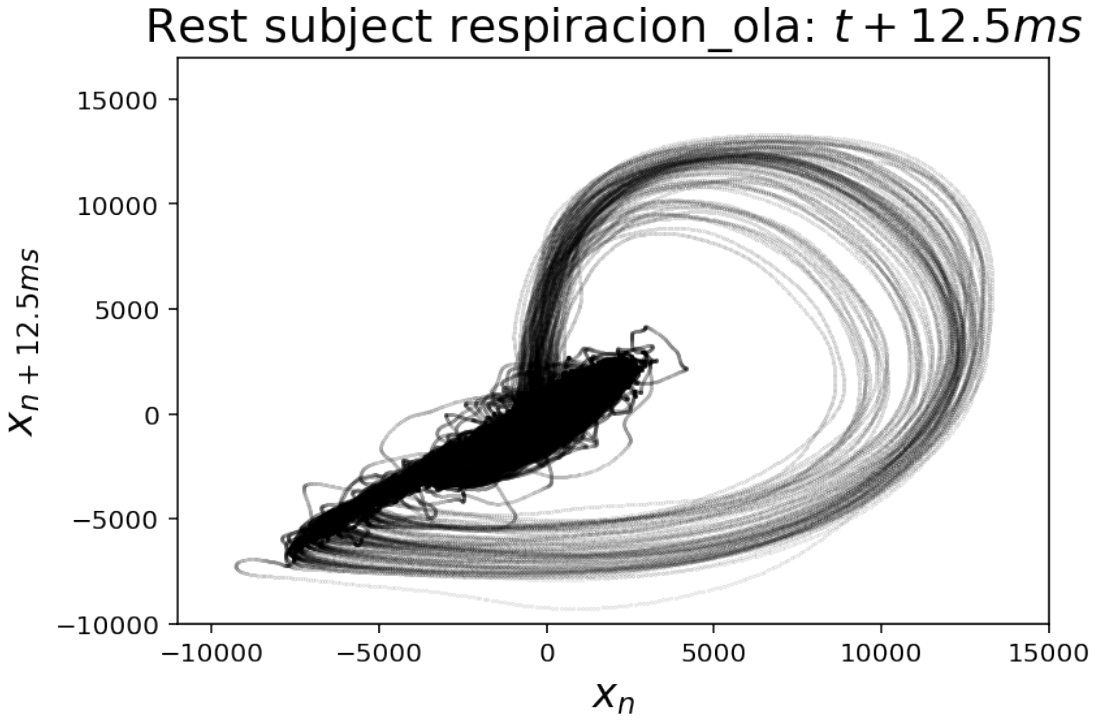
Rest subject sostener_respiracion_abdomen: $t + 12.5ms$

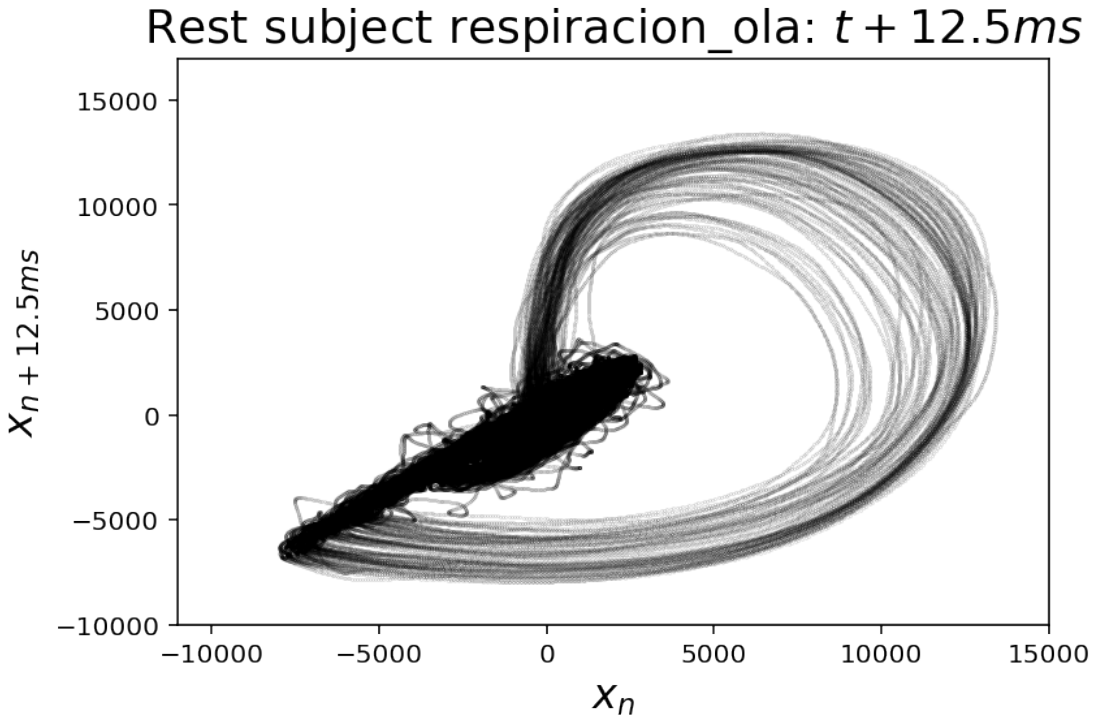


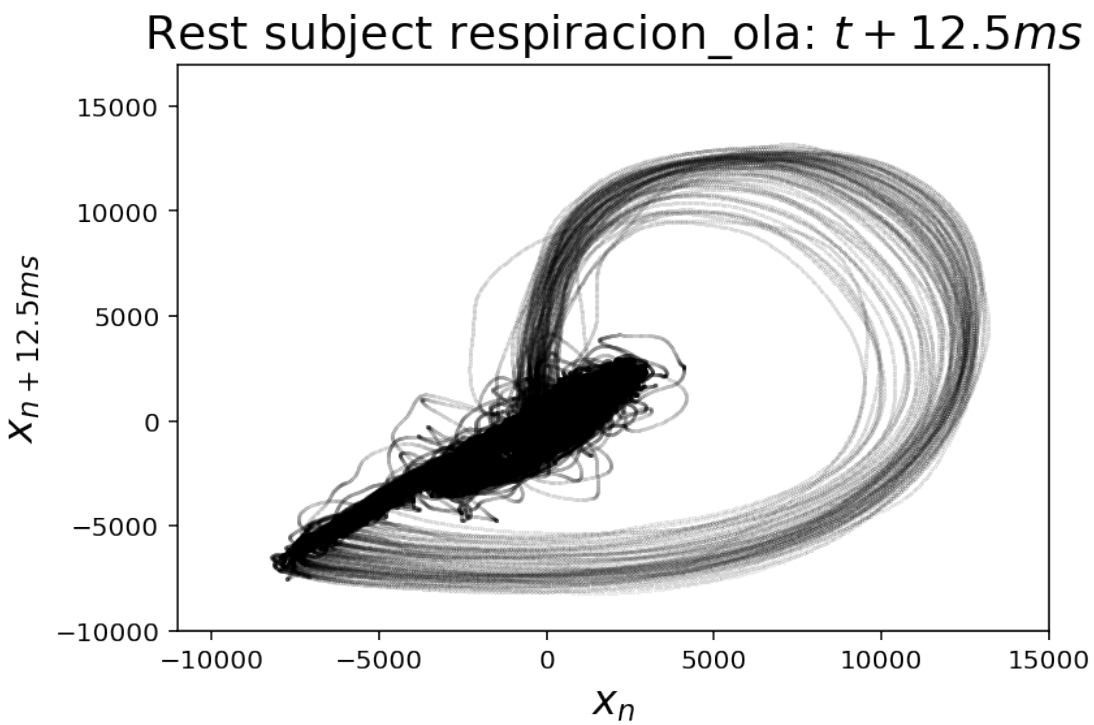

Rest subject sostener_respiracion_pecho: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$x_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
x_n


Rest subject sostener_respiracion_pecho: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$x_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
x_n


Rest subject sostener_respiracion_pecho: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$x_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
x_n


Rest subject respiracion_ola: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$x_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
x_n

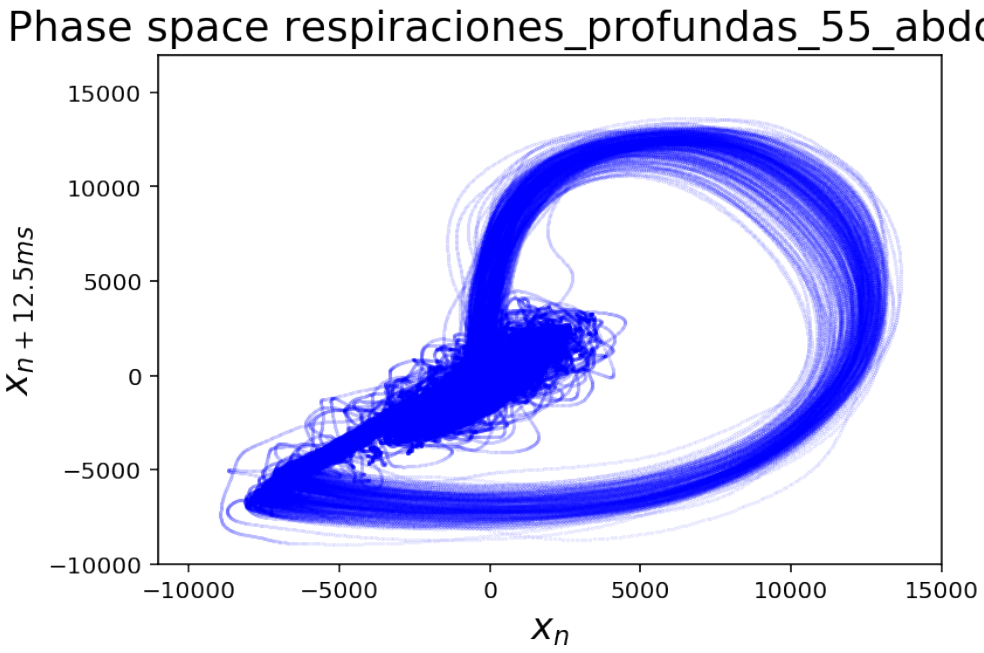

Rest subject respiracion_ola: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$X_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
X_n

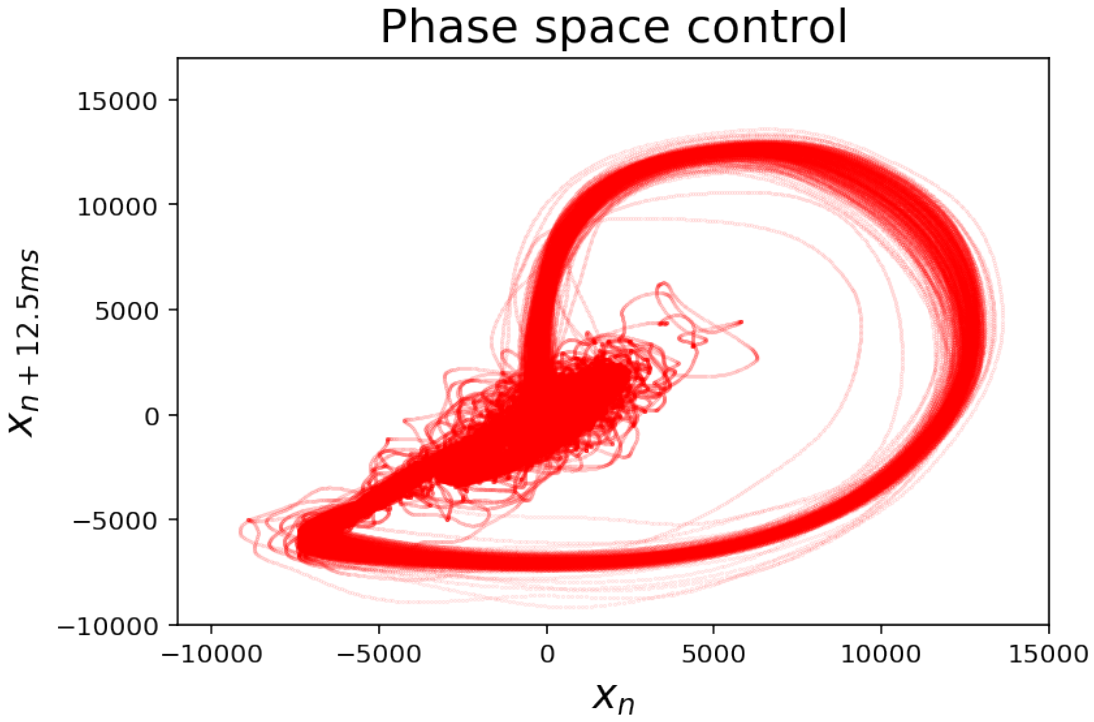

Rest subject respiracion_ola: $t + 12.5ms$
15000
10000
5000
0
−5000
−10000
$X_n + 12.5ms$
−10000
−5000
0
5000
10000
15000
X_n

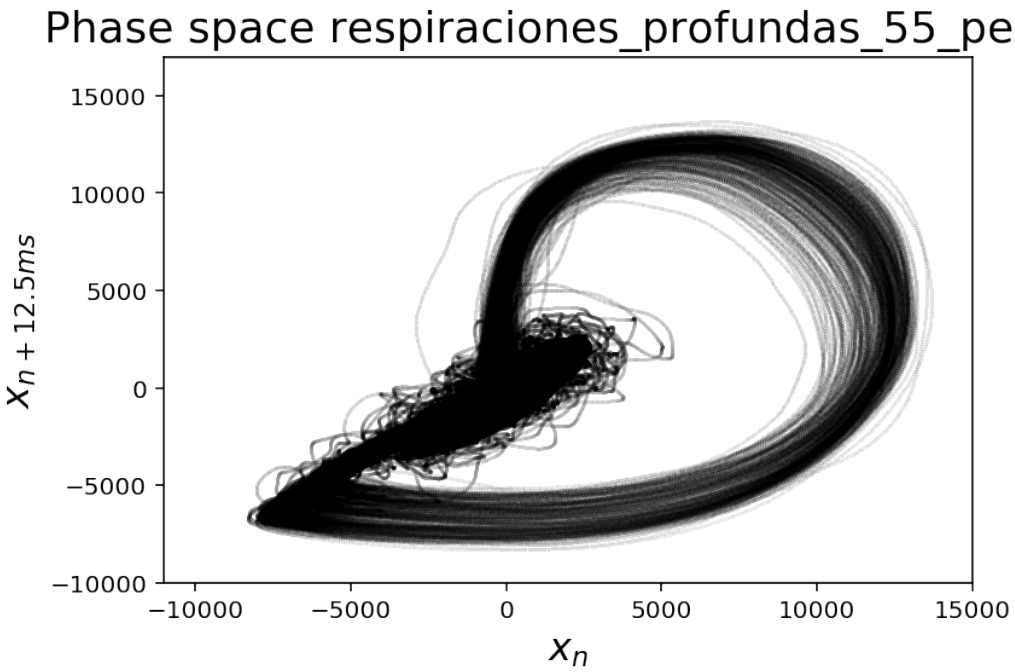
Notice how the trajectory in phase space becomes smaller when the subject is holdings his breath.

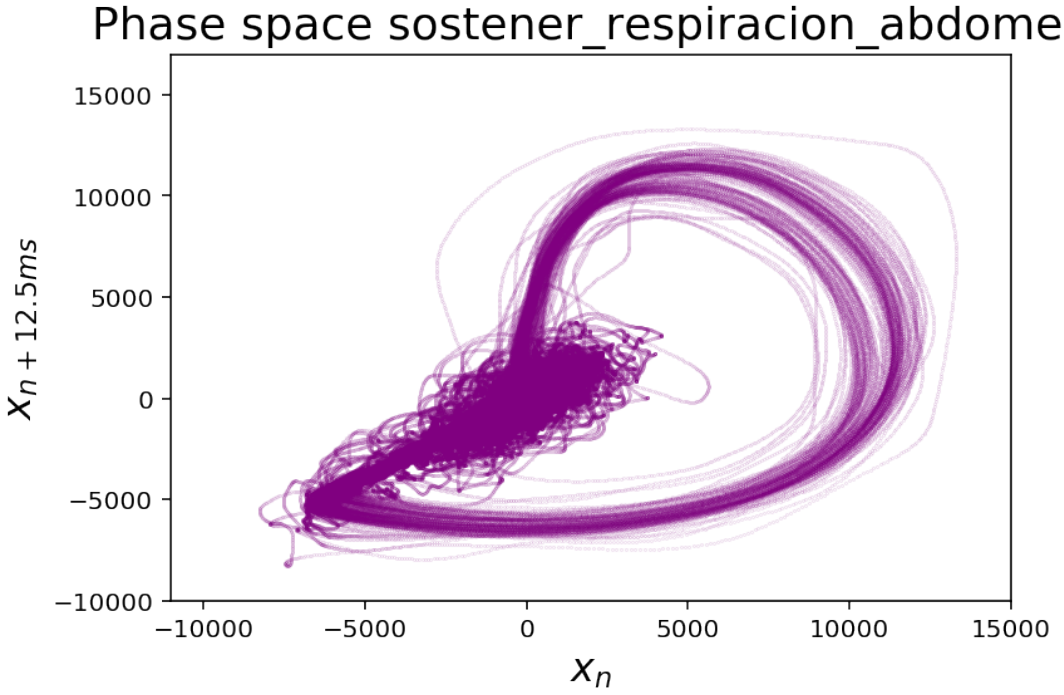
Aqui podrías volver a superponer los registros de sostener la respiración y los otros. Con los mismos colores que usaste en Poincaré a ver si se observa algún patrón.

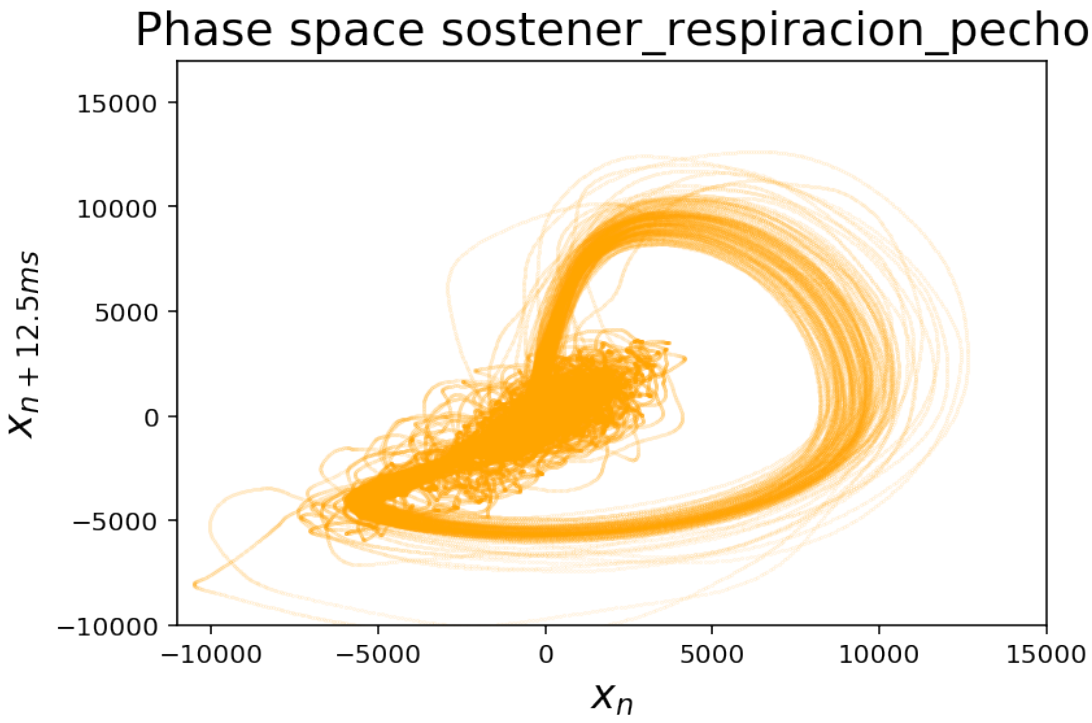
[16]:
```
colors = ['blue', 'red', 'black', 'purple', "orange", "green"]
color_index = 0
time_delay = 125 #12.5 ms

for i in range(len(recordings)):
    n = np.size(recordings[i].waveData) #size of the voltage vector

    if i!= 0 and corresponding_folder[i] != corresponding_folder[i-1]:
        color_index = color_index+1
        plt.title("Phase space " + str(corresponding_folder[i-1]))
        plt.xlim(-11000, 15000)
        plt.ylim(-10000, 17000)
        plt.xlabel(r"$x_n$")
        y = r'$x_{n+'+str(time_delay*0.1)+"ms}$"
        plt.ylabel(y)
        plt.savefig('meditation_imgs/phase_space'+␣
 ↪str(corresponding_folder[i-1])+'.jpg')#Saves images in folder
        plt.show()

    plt. plot(recordings[i].waveData[0: n-time_delay], recordings[i].
 ↪waveData[time_delay: n],
              marker = "o", markersize = 0.05, linewidth = 0.005, c=␣
 ↪colors[color_index])

plt.title(str(corresponding_folder[i-1]) + " Phase space rec")
plt.xlim(-11000, 15000)
plt.ylim(-10000, 17000)
plt.xlabel(r"$x_n$")
y = r'$x_{n+'+str(time_delay*0.1)+"ms}$"
plt.ylabel(y)
plt.savefig('meditation_imgs/phase_space'+ str(corresponding_folder[i-1])+'.
 ↪jpg')#Saves images in folder
plt.show()
```

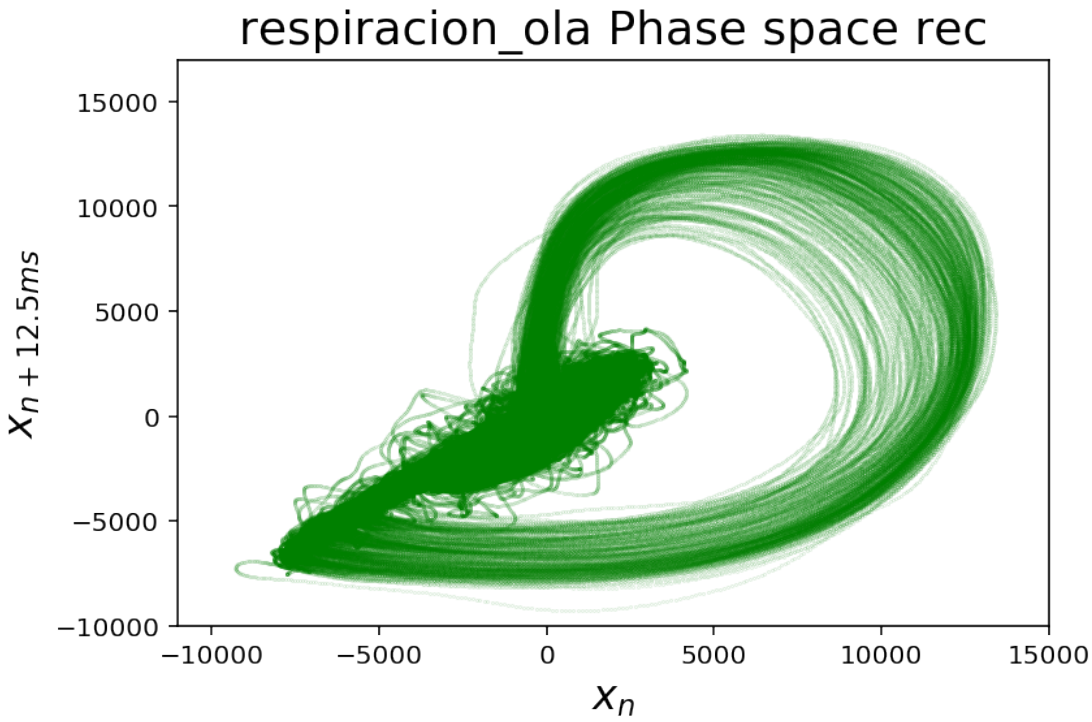

Phase space respiraciones_profundas_55_abdomen
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$x_n + 12.5ms$
x_n


Phase space control
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$x_n + 12.5ms$
x_n

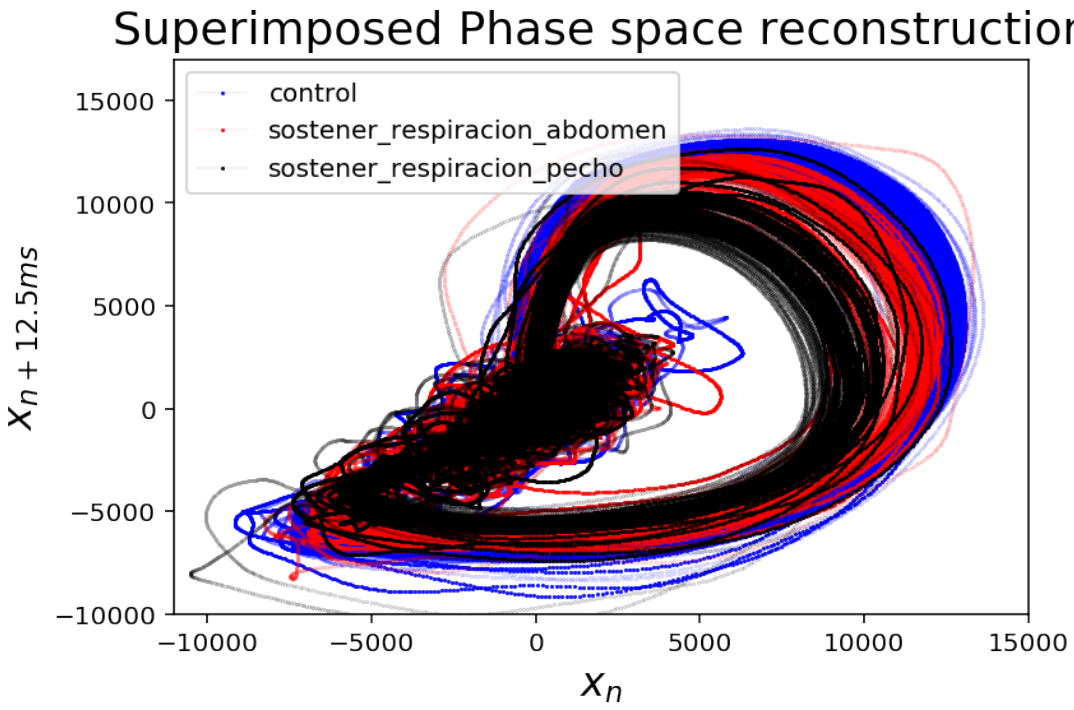

Phase space respiraciones_profundas_55_pecho
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$x_n + 12.5ms$
x_n


Phase space sostener_respiracion_abdomen
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$x_n + 12.5ms$
x_n


Phase space sostener_respiracion_pecho
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$x_n + 12.5ms$
x_n

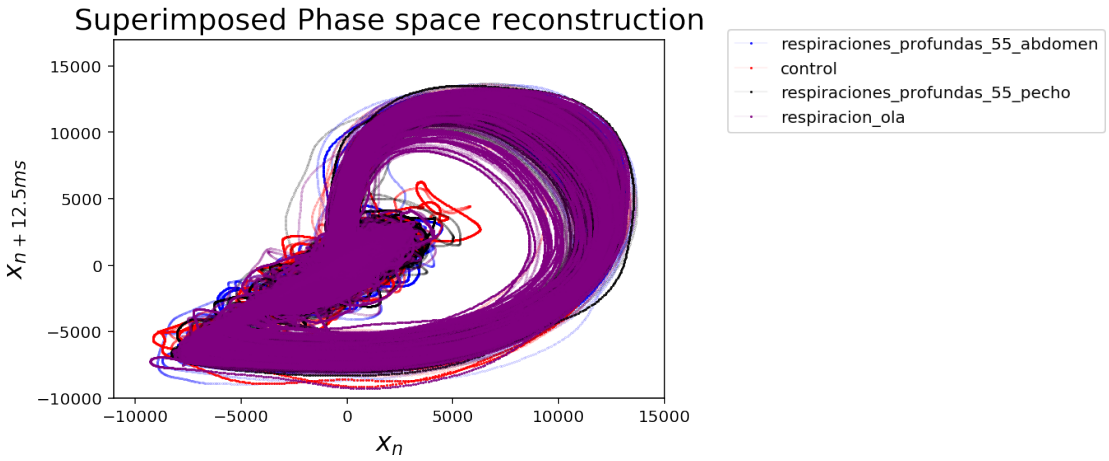

respiracion_ola Phase space rec
15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
$x_n + 12.5ms$
x_n

[17]:
```
# Comparison between control, holding breath in chest and holding breath in␣
↪abdomen
colors = ['blue', 'red', 'black', 'purple', "orange", "green"]
color_index = 0
banner = False
time_delay = 125 #12.5 ms

for i in range(len(recordings)):
    if corresponding_folder[i]=="control" or "sostener" in␣
↪corresponding_folder[i]: #First event
        n = np.size(recordings[i].waveData) #size of the voltage vector
        if banner == False: #First event
            plt. plot(recordings[i].waveData[0: n-time_delay], recordings[i].
↪waveData[time_delay: n],
                      marker = "o", markersize = 0.5, linewidth = 0.05, c=␣
↪colors[color_index],
                      label = corresponding_folder[i])
        else: #Other events, does not plot label
            plt. plot(recordings[i].waveData[0: n-time_delay], recordings[i].
↪waveData[time_delay: n],
                      marker = "o", markersize = 0.1, linewidth = 0.005, c=␣
↪colors[color_index])
        banner = True

    if len(recordings)-1 != i:
        if banner == True and corresponding_folder[i] !=␣
↪corresponding_folder[i+1]:
            color_index = color_index+1
            banner = False

plt.title("Superimposed Phase space reconstruction")
plt.xlabel(r"$x_n$")
y = r'$x_{n+'+str(time_delay*0.1)+"ms}$"
plt.ylabel(y)
plt.xlim(-11000, 15000)
plt.ylim(-10000, 17000)
plt.legend(loc='upper left')
plt.savefig('meditation_imgs/phase_space_superimposed_sostener.jpg')#Saves␣
↪images in folder
plt.show()
```



[18]:
```
# Compare: control, respiraciones profundas 55 pecho, respiraciones profundas␣
↪55 abdomen, respiraciones en ola.

colors = ['blue', 'red', 'black', 'purple', "orange", "green"]
color_index = 0
banner = False
time_delay = 125 #12.5 ms

for i in range(len(recordings)):
    if corresponding_folder[i]=="control" or "55" in corresponding_folder[i] or␣
↪"ola" in corresponding_folder[i]: #First event
        n = np.size(recordings[i].waveData) #size of the voltage vector
        if banner == False: #First event
            plt. plot(recordings[i].waveData[0: n-time_delay], recordings[i].
↪waveData[time_delay: n],
                      marker = "o", markersize = 0.5, linewidth = 0.05, c=␣
↪colors[color_index],
                      label = corresponding_folder[i])
        else: #Other events, does not plot label
            plt. plot(recordings[i].waveData[0: n-time_delay], recordings[i].
↪waveData[time_delay: n],
                      marker = "o", markersize = 0.1, linewidth = 0.005, c=␣
↪colors[color_index])
```

```
        banner = True

    if len(recordings)-1 != i:
        if banner == True and corresponding_folder[i] !=␣
↪corresponding_folder[i+1]:
            color_index = color_index+1
            banner = False

plt.title("Superimposed Phase space reconstruction")
plt.xlabel(r"$x_n$")
y = r'$x_{n+'+str(time_delay*0.1)+"ms}$"
plt.ylabel(y)
plt.xlim(-11000, 15000)
plt.ylim(-10000, 17000)
plt.legend(bbox_to_anchor=(1.1, 1.05))
plt.show()
```



[ ]:

[ ]:

6.0.1 Exercise: Are the results obtained in a Poincaré plot and in a recurrence plot contradictory? While we are observing a loss in the first plot's variability with exercise, we are seeing the opposite in a recurrence plot. Why?

[ ]:

7 Final remarks

As you have seen in this notebook, it is not always valid to use certain techniques nor recordings. There are certain conditions that must be met before we can use these techniques. Besides, recordings must fulfill certain data criteria for being able to perform data analysis with them.