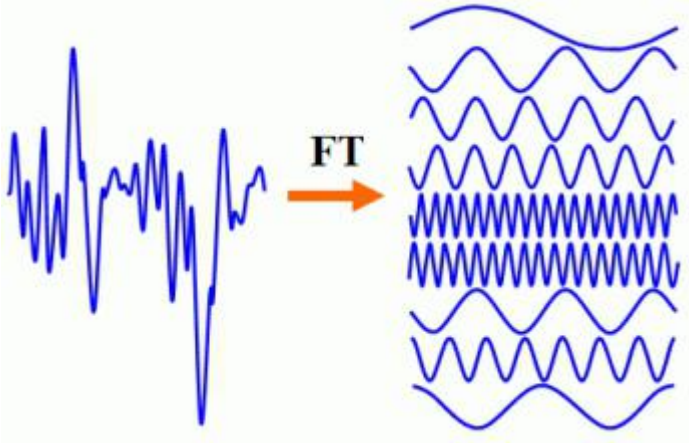
Ресурсы

https://habr.com/ru/post/196374/
https://medium.com/orikami-blog/exploring-heart-rate-variability-using-python-483a7037c64d
https://nicolasfauchereau.github.io/climatecode/posts/wavelet-analysis-in-python/
http://math.phys.msu.ru/data/189/MM_lec8.pdf
https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B9%D0%B2%D0%BB%D0%B5%D1%82
https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B9%D0%B2%D0%BB%D0%B5%D1%82%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D
https://habr.com/ru/post/253447/
https://www.micard.ru/novosti/vsr-v-programme-rabochee-mesto-vracha-dlya-windows
https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D0%BA%D1%82%D1%80%D0%BE

Теория

Преобразование Фурье



Для анализа периодических сигналов в инженерной практике широко используют мощный математический аппарат, именуемый в общем «**Фурье-анализ**».

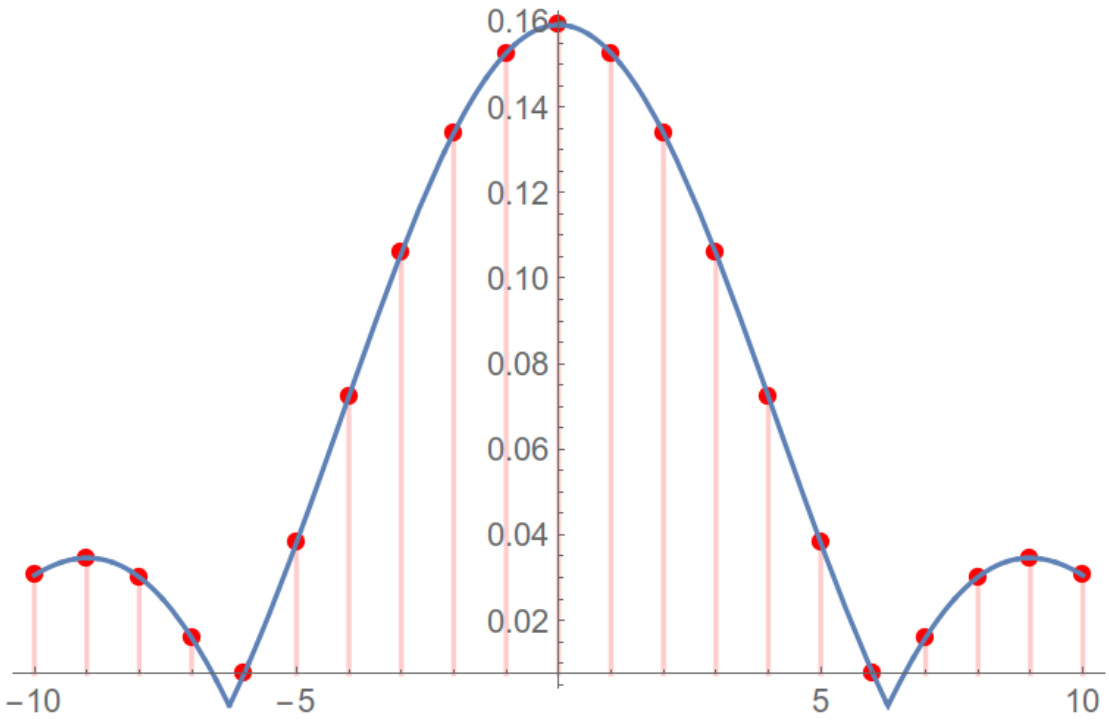
Не так давно, в 19 веке, французский математик Жан Батист Жозеф Фурье показал, что любую функцию, удовлетворяющую некоторым условиям (непрерывность во времени, периодичность, удовлетворение условиям Дирихле) можно разложить в ряд, который в дальнейшем получил его имя — **ряд Фурье**.

В инженерной практике разложение периодических функций в ряд Фурье широко используется, например, в задачах теории цепей: несинусоидальное входное воздействие раскладывают на сумму синусоидальных и рассчитывают необходимые параметры цепей, например, по методу наложения.

Существует несколько возможных вариантов записи коэффициентов ряда Фурье, нам же лишь необходимо знать суть. Разложение в ряд Фурье позволяет разложить непрерывную функцию в сумму других непрерывных функций. И в общем случае, ряд будет иметь бесконечное количество членов.

Дальнейшим усовершенствованием подхода Фурье является интегральное преобразование его же имени -- **преобразование Фурье**. **Преобразование Фурье позволяет разложить исходный сигнал на гармонические составляющие**. В отличие от ряда Фурье, преобразование Фурье раскладывает функцию не по дискретным частотам (набор частот ряда Фурье, по которым происходит разложение, вообще говоря, дискретный), а по непрерывным. Давайте взглянем на то, как соотносятся коэффициенты ряда Фурье и результат преобразования Фурье, именуемый, собственно, **спектром**. Небольшое отступление: спектр преобразования Фурье — в общем случае, функция комплексная, описывающая **комплексные амплитуды** соответствующих гармоник. Т.е., значения спектра — это комплексные числа, чьи модули являются амплитудами соответствующих частот, а аргументы — соответствующими начальными фазами. На практике, рассматривают отдельно **амплитудный спектр** и **фазовый спектр**.

Соответствие ряда Фурье и преобразования Фурье на примере амплитудного спектра.



Однако, преобразование Фурье сопоставляет непрерывной во времени, бесконечной функции другую, непрерывную по частоте, бесконечную функцию — спектр. Как быть, если у нас нет бесконечной во времени функции, а есть лишь какая-то записанная её дискретная во времени часть? Ответ на этот вопрос даёт дальнейшей развитие преобразования Фурье — дискретное преобразование Фурье (ДПФ).

Дискретное преобразование Фурье призвано решить проблему необходимости непрерывности и бесконечности во времени сигнала. По сути, мы полагаем, что

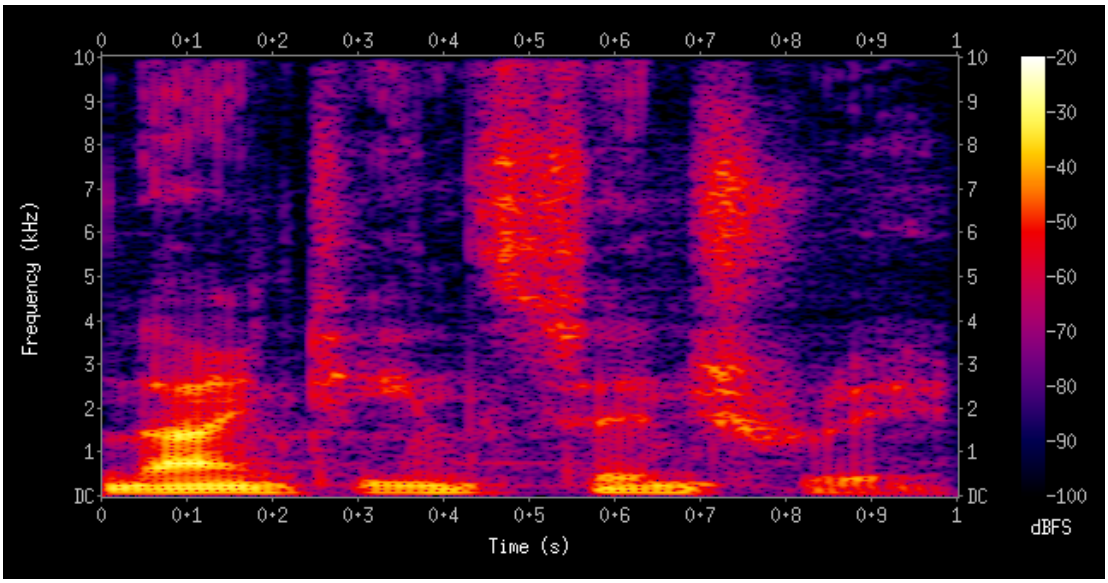
вырезали какую-то часть бесконечного сигнала, а всю остальную временную область считаем этот сигнал нулевым.

Математически это означает, что, имея исследуемую бесконечную во времени функцию f(t), мы умножаем ее на некоторую оконную функцию w(t), которая обращается в ноль везде, кроме интересующего нас интервала времени.

Если «выходом» классического преобразования Фурье является спектр – функция, то «выходом» дискретного преобразования Фурье является дискретный спектр. И на вход тоже подаются отсчёты дискретного сигнала.

Спектрограмма

Спектрограмма мужского голоса



Спектрограмма — изображение, показывающее зависимость спектральной плотности мощности сигнала от времени. Спектрограммы применяются для идентификации речи, анализа звуков животных, в различных областях музыки, радио- и гидролокации, обработке речи, сейсмологии и в других областях.

Наиболее распространенным представлением спектрограммы является двумерная диаграмма: на горизонтальной оси представлено время, по вертикальной оси — частота; третье измерение с указанием амплитуды на определенной частоте в конкретный момент времени представлено интенсивностью или цветом каждой точки изображения.

Есть много вариантов представления: иногда вертикальная и горизонтальная оси включены так, что время бежит вверх и вниз, иногда амплитуда представлена вершинами в трёхмерном пространстве, а не цветом или интенсивностью. Частота и амплитуда осей может быть линейными или логарифмическими, в зависимости от того, с какой целью используется график. Аудио обычно может быть представлено с логарифмической осью амплитуды (зачастую, в децибелах или дБ), и частота будет

линейной, чтобы подчеркнуть гармонические отношения, или логарифмической, чтобы подчеркнуть музыкальные, тональные отношения.

Спектрограмма обычно создаётся одним из двух способов: аппроксимируется, как набор фильтров, полученных из серии полосовых фильтров (это был единственный способ до появления современных методов цифровой обработки сигналов), или **рассчитывается по сигналу времени, используя оконное преобразование Фурье**. Эти два способа фактически образуют разные квадратичные частотно-временные распределения, но эквивалентны при некоторых условиях.

Диаграмма рассеяния

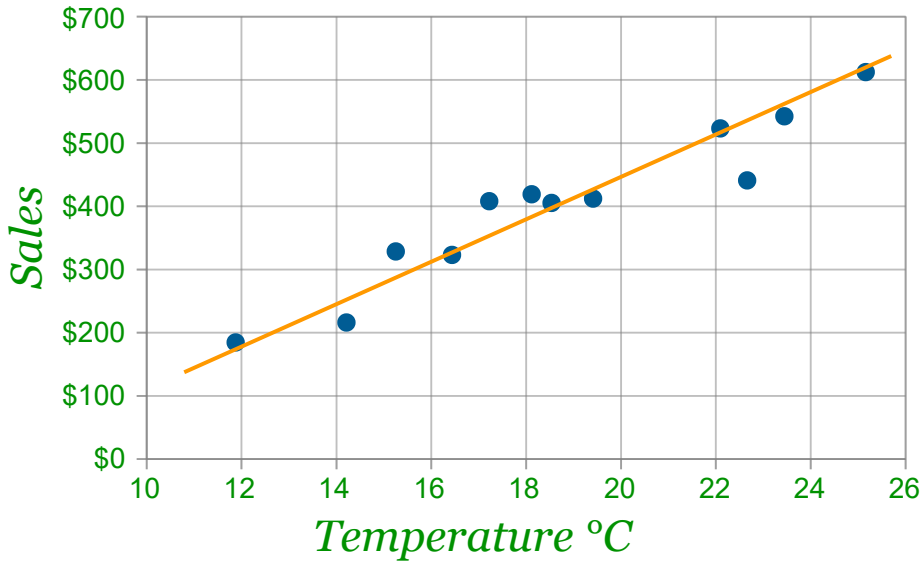
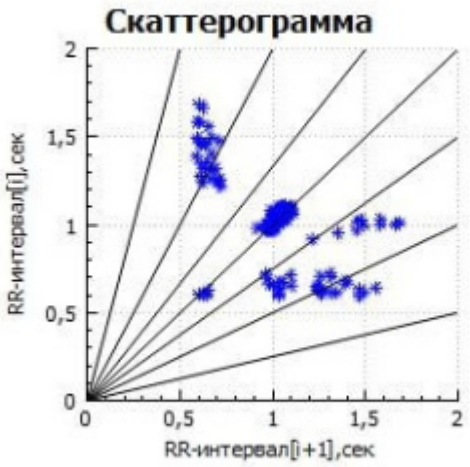


Диаграмма рассеяния (также точечная диаграмма, англ. scatter plot) — математическая диаграмма, изображающая значения двух переменных в виде точек на декартовой плоскости.

На диаграмме рассеяния каждому наблюдению соответствует точка, координаты которой равны значениям двух каких-то параметров этого наблюдения. Если предполагается, что один из параметров зависит от другого, то обычно значения независимого параметра откладывается по горизонтальной оси, а значения зависимого — по вертикальной. Диаграммы рассеяния используются для демонстрации наличия или отсутствия корреляции между двумя переменными.

При анализе деятельности сердца **скаттерограмма дает наглядное представление об общем характере и закономерностях сердцечного ритма.**

Скаттерограмма рассчитывается с учетом выбранного типа коррекции RR-интервалов и только по выделенной в данный момент области на ритмограмме.



Обработка данных

Импортируем необходимые модули:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy.signal import find_peaks
%matplotlib inline
```

In [ ]:
```
data_normal1 = pd.read_csv("normal1.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['A1(X)',        'A1(Y)',        'A2(X)',        'A2(Y)',
```

Каждый столбец X убираем, так как каждая строка соответсвует отдельному моменту времени, числовые значения которых мы не будем использовать

In [ ]:
```
data_normal1 = data_normal1.drop(labels=["A1(X)", "A2(X)", "A3(X)", "B3(X)", "B2
```

In [ ]:
```
data_normal2 = pd.read_csv("normal2.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['D1(X)',        'D1(Y)',        'D3(X)',        'D3(Y)',
```

In [ ]:
```
data_normal2 = data_normal2.drop(labels=["D1(X)", "D3(X)", "D4(X)", "C3(X)", "C2
```

In [ ]:
```
data_normal3 = pd.read_csv("normal3.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['E2(X)',        'E2(Y)',        'E3(X)',        'E3(Y)',
```

In [ ]:
```
data_normal3 = data_normal3.drop(labels=["E2(X)", "E3(X)", "E4(X)", "F1(X)", "F3
```

In [ ]:
```
data_normal4 = pd.read_csv("normal4.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['F4(X)',        'F4(Y)',        'F5(X)',        'F5(Y)',
```

In [ ]:
```
data_normal4 = data_normal4.drop(labels=["F4(X)", "F5(X)", "G3(X)", "G2(X)"], ax
```

In [ ]:
```
data_normal5 = pd.read_csv("normal5.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['H2(X)',        'H2(Y)',        'H3(X)',        'H3(Y)',
```

In [ ]:
```
data_normal5 = data_normal5.drop(labels=["H2(X)", "H3(X)", "H4(X)", "J3(X)"], ax
```

Соединяем все фрагменты и получаем целый массив данных для нормы.

In [ ]:
```
data_normal = pd.concat([data_normal1, data_normal2, data_normal3, data_normal4,
```

Переименуем все столбцы:

In [ ]:
```
new_columns = []
for i in range(len(data_normal.columns)):
    new_columns.append(data_normal.columns[i][:2])
data_normal.columns = new_columns
```

Теперь считываем данные при аритмии:

In [ ]:
```
data_arrhythmia1 = pd.read_csv("arrhythmia1.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['A1(X)',        'A1(Y)',        'A2(X)',        'A2(Y)',
```

In [ ]:
```
data_arrhythmia1 = data_arrhythmia1.drop(labels=["A1(X)", "A2(X)", "A3(X)", "B3(
```

In [ ]:
```
data_arrhythmia2 = pd.read_csv("arrhythmia2.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['D1(X)',        'D1(Y)',        'D3(X)',        'D3(Y)',
```

In [ ]:
```
data_arrhythmia2 = data_arrhythmia2.drop(labels=["D1(X)", "D3(X)", "D4(X)", "C3(
```

In [ ]:
```
data_arrhythmia3 = pd.read_csv("arrhythmia3.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['E2(X)',        'E2(Y)',        'E3(X)',        'E3(Y)',
```

In [ ]:
```
data_arrhythmia3 = data_arrhythmia3.drop(labels=["E2(X)", "E3(X)", "E4(X)", "F1(
```

In [ ]:
```
data_arrhythmia4 = pd.read_csv("arrhythmia4.dat",
                  sep="\t+",
```

```
                  skiprows=1,
                  engine='python',
                  names=['F4(X)',        'F4(Y)',        'F5(X)',        'F5(Y)',
```

In [ ]:
```
data_arrhythmia4 = data_arrhythmia4.drop(labels=["F4(X)", "F5(X)", "G3(X)", "G2(
```

In [ ]:
```
data_arrhythmia5 = pd.read_csv("arrhythmia5.dat",
                  sep="\t+",
                  skiprows=1,
                  engine='python',
                  names=['H2(X)',        'H2(Y)',        'H3(X)',        'H3(Y)',
```

In [ ]:
```
data_arrhythmia5 = data_arrhythmia5.drop(labels=["H2(X)", "H3(X)", "H4(X)", "J3(
```

Соединяем все фрагменты и получаем целый массив данных для аритмии.

In [ ]:
```
data_arrhythmia = pd.concat([data_arrhythmia1, data_arrhythmia2, data_arrhythmia
```

In [ ]:
```
new_columns = []
for i in range(len(data_arrhythmia.columns)):
    new_columns.append(data_arrhythmia.columns[i][:2])
data_arrhythmia.columns = new_columns
```

Поиск длин интервалов RR и значений напряжений пиков R

Находим зубцы R для всех электродов в норме, выставляем минимальную дистанцию между пиками = 300:

In [ ]:
```
peaks_normal = []

for col in data_normal.columns:
    l = len(data_normal[col])
    # l = 2000
    threshold = 5000

    x, _ = find_peaks(data_normal.head(l)[col], height=threshold, distance=300)
    peaks_normal.append(x)
```

Находим RR-интервалы в норме

In [ ]:
```
intervals_normal = []

for col in range(0, len(data_normal.columns)):
    t = []
    for x in range(0, len(peaks_normal[col]) - 1):
        t.append(peaks_normal[col][x + 1] - peaks_normal[col][x])
    intervals_normal.append(t)
```

Находим пики R для всех электродов при аритмии:

In [ ]:
```
peaks_arrhythmia = []
```

```
for col in data_arrhythmia.columns:
    l = len(data_arrhythmia[col])
    # l = 2000
    threshold = 7000

    x, _ = find_peaks(data_arrhythmia.head(l)[col], height=threshold, distance=3
    peaks_arrhythmia.append(x)
```

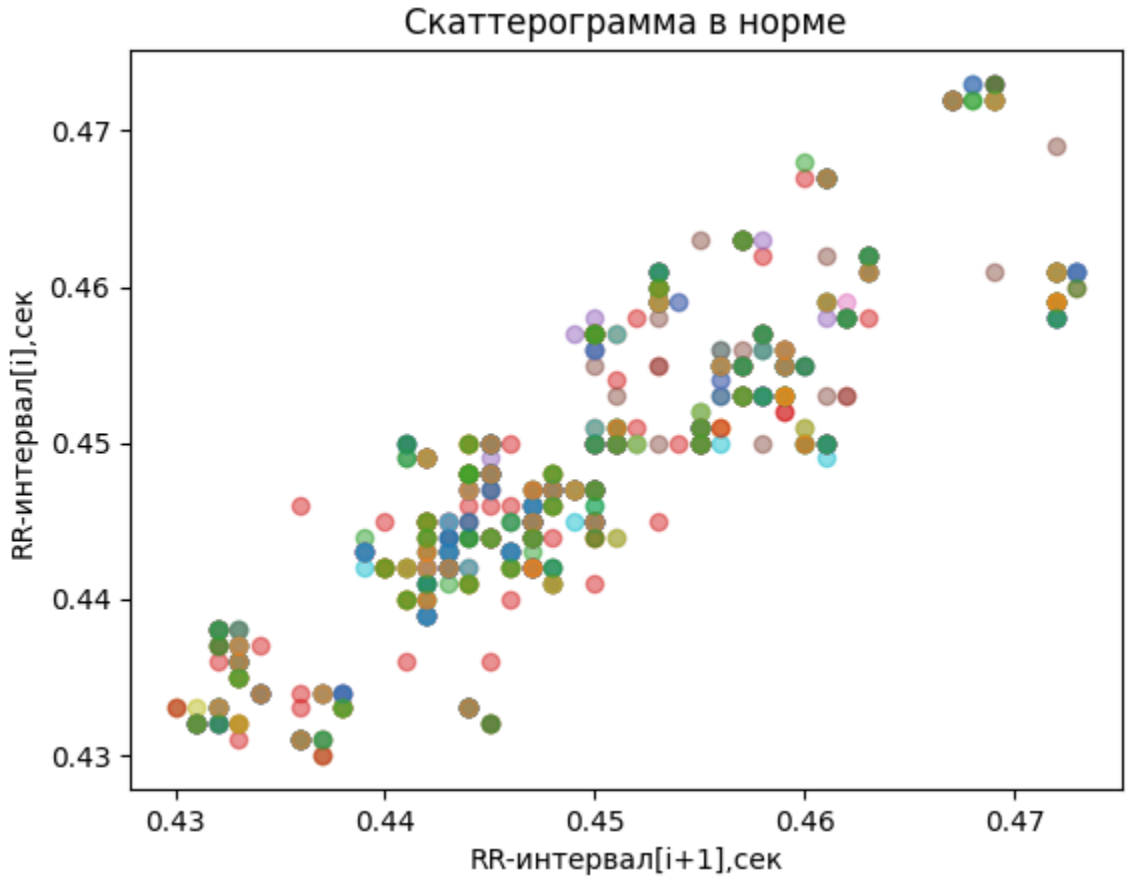
Находим RR-интервалы при аритмии

In [ ]:
```
intervals_arrhythmia = []

for col in range(0, len(data_normal.columns)):
    t = []
    for x in range(0, len(peaks_arrhythmia[col]) - 1):
        t.append(peaks_arrhythmia[col][x + 1] - peaks_arrhythmia[col][x])
    intervals_arrhythmia.append(t)
```

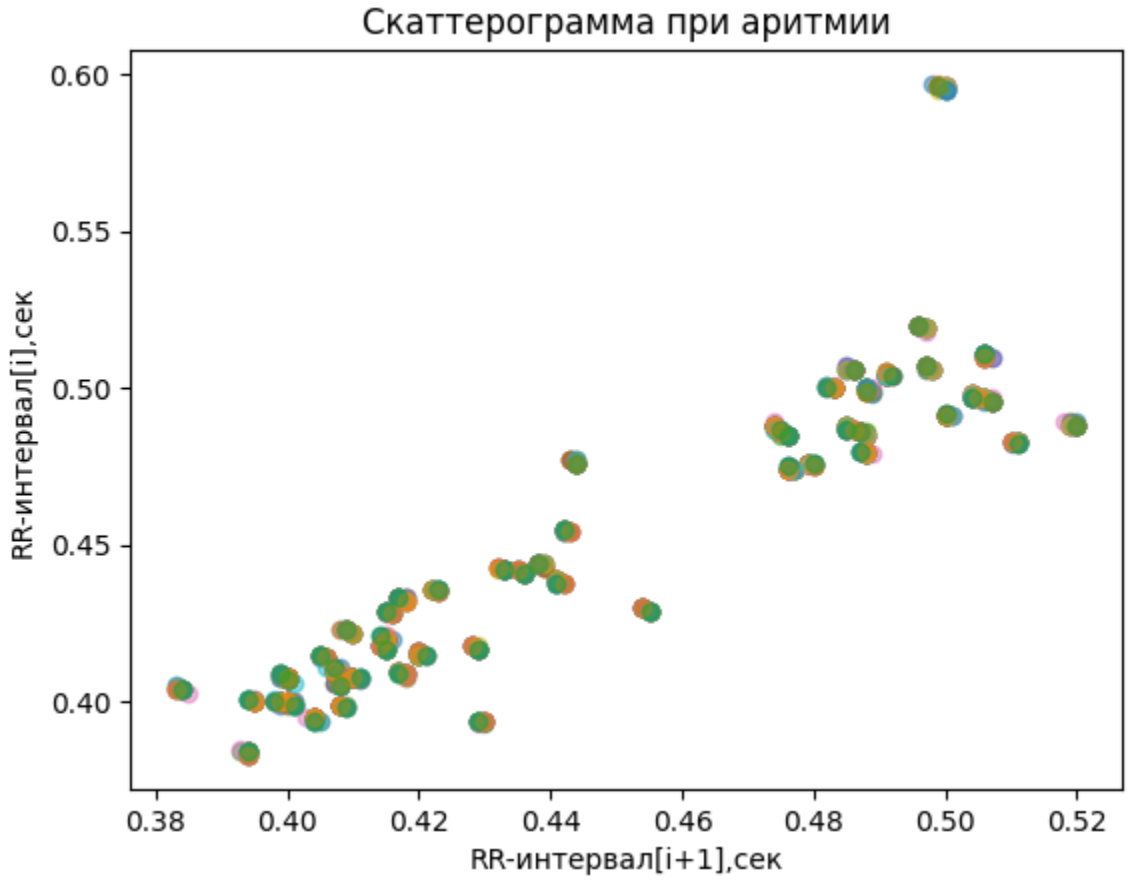
Скаттерограмма (Scatter plot)

Как уже было сказано ранее, скаттерограмма рассчитывается только по выделенной в области. Построим Scatter plot для первых 50 интервалов в норме и при аритмии. Для этого будем отображать данные на графике, проходя по массивам значений со всех электродов и выбирая для оси x с 1 по 51 элементы для k + 1 RR-интервалов, а для оси y выбирая с 0 по 50 элементы.

In [ ]:
```
for int in intervals_normal:
    y = np.array(int[0 : 50]) / 1000.0
    x = np.array(int[0 + 1:51]) / 1000.0
    plt.scatter(x, y, alpha=0.5)

plt.title('Скаттерограмма в норме')
plt.xlabel('RR-интервал[i+1],сек')
plt.ylabel('RR-интервал[i],сек')
plt.show()
```



In [ ]:
```
for int in intervals_arrhythmia:
    y = np.array(int[0:50]) / 1000.0
    x = np.array(int[0 + 1:51]) / 1000.0
    plt.scatter(x, y, alpha=0.5)

plt.title('Скаттерограмма при аритмии')
plt.xlabel('RR-интервал[i+1],сек')
plt.ylabel('RR-интервал[i],сек')
plt.show()
```



Как мы можем видеть из полученных графиков, длины интервалов в норме равномерно увеличиваются за данный промежуток времени с 0,43с до 0,47с.

При аритмии равномерное увеличение сменяется резкими скачками до значений 0,6с.