**Пользовательские функции**

**len\_ampl(x) - """вычисление числа ненулевых элементов временного ряда"""**

def len\_ampl(x):

"""вычисление числа ненулевых элементов временного ряда"""

y = 0

for i in range(len(x)):

if x[i] != 0:

y +=1

return y

**index(x) - """вычисление индексов ненулевых элементов временного ряда"""**

def index(x):

"""вычисление индексов ненулевых элементов временного ряда"""

y = []

for i in range(len(x)):

if x[i] != 0:

y.append(i)

return y

**sequence\_max(x) - """вычисление максимумов временного ряда x"""**

def sequence\_max(x):

"""вычисление максимумов временного ряда"""

y = []

for i in range(1,len(x)-1):

if x[i-1] <= x[i] and x[i+1] <= x[i]:

y.append(x[i])

else: y.append(0)

return y

**sequence\_max0(x) - """вычисление индикаторов 1 максимумов временного ряда"""**

def sequence\_max0(x):

"""вычисление индикаторов 1 максимумов временного ряда"""

y = []

for i in range(1,len(x)-1):

if x[i-1] <= x[i] and x[i+1] <= x[i]:

y.append(1)

else: y.append(0)

return y

**sequence\_max1(x,v) - """вычисление максимумов временного ряда с ограничением амплитуды"""**

def sequence\_max1(x,v):

"""вычисление максимумов временного ряда с ограничением амплитуды"""

y = []

for i in range(1,len(x)-1):

if x[i-1] <= x[i] and x[i+1] <= x[i] and x[i] > v:

y.append(x[i])

else: y.append(0)

return y

**sequence\_max\_ampl(x,v) - """вычисление индикаторов 1 максимумов временного ряда с ограничением амплитуды"""**

def sequence\_max\_ampl(x,v):

"""вычисление индикаторов 1 максимумов временного ряда с ограничением амплитуды"""

y = []

for i in range(1, len(x)-1):

if x[i-1] <= x[i] and x[i+1] <= x[i] and x[i] > v:

y.append(1)

else: y.append(0)

return y

**sequence\_distance(x, y) - """вычисление расстояний от максимумов образца x до ближайшего максимума эталона y"""**

def sequence\_distance(x, y):

"""вычисление расстояний от максимумов образца до ближайшего максимума эталона"""

u = []

for i in range(len(x)):

if x[i] != 0:

for j in range(len(y)):

if ( i - j >= 0 and y[i - j ] != 0) and ( i + j < len(y) and y[i + j ] != 0):

if j == 0:

u.append(j)

else:

u.append(j)

u.append(-j)

break

elif ( i - j >= 0 and y[i - j ] != 0) :

u.append(j)

break

elif ( i + j < len(y) and y[i + j ] != 0):

u.append(-j)

break

return u

**norm\_serie(x) - """Нормализация временного ряда"""**

def norm\_serie(x):

"""Нормализация временного ряда"""

y = []

for i in range (len(x)):

y.append((x[i] - min(x))/(max(x) - min(x)))

return y

**delta\_serie(x) - """Приращения временного ряда"""**

def delta\_serie(x):

"""Приращения временного ряда"""

y = []

for i in range (1, len(x)):

y.append((x[i] - x[i - 1]))

return y

**func(x, y) - """ Суммирование почленное элементов двух разных списков """**

def func(x, y):

""" Суммирование почленное элементов двух разных списков """

return x + y

**sum\_list(x) - """Почленное суммирование списков списка"""**

def sum\_list(x):

"""Почленное суммирование списков списка"""

y = [0]\*len(x[0])

for i in range(len(x)):

y = list( map(func, y, x[i]) )

return y

**sum\_max\_lists(x) - """Почленное суммирование индикаторов 1 максимумов списков списка"""**

def sum\_max\_lists(x):

"""Почленное суммирование индикаторов 1 максимумов списков списка"""

y = sequence\_max0(x[0])

for i in range(1,len(x)):

y = list( map(func, y, sequence\_max0(x[i])))

return y

**stat\_analys(z) - """Статистический анализ ряда распределений z"""**

def stat\_analys(z):

"""Статистический анализ ряда распределений"""

return [np.mean(z), np.std(z), st.t.interval(0.95, len(z)-1, loc=np.mean(z), scale=st.sem(z))]

**visual\_analysis(x) - """Визуальный анализ ряда распределений""" (гистограмма, ядерная оценка плотности и кривая Гаусса)**

def visual\_analysis(x):

"""Визуальный анализ ряда распределений"""

fig, ax = plt.subplots()

\_range = np.linspace(-3,4, 100)

plt.style.use('seaborn-white')

ax.hist(x, bins=7, range = (-3,4), normed=True, alpha=0.5,

histtype='stepfilled', color='steelblue', edgecolor='none')

ax.plot(\_range, norm.pdf(\_range, np.mean(x), np.std(x)), '-.k')

ax.plot(\_range, st.gaussian\_kde(x)(\_range), color = 'blue')

ax.set(xlim=(-3, 4), ylim=(0, 0.5),

xlabel='x', ylabel='',

title='синий график - ядерная оценка плотности распределения, \n черный штрихпунктирный график - кривая Гаусса')

return plt.show()

**visual\_analys2(x,y) - """Визуальный анализ двух рядов распределений""" (два рисунка с гистограммой, ядерной оценкой плотности и кривой Гаусса)**

def visual\_analys2(x,y):

"""Визуальный анализ двух рядов распределений"""

fig, ax = plt.subplots(2)

\_range = np.linspace(-3, 4, 100)

plt.style.use('seaborn-white')

ax[0].hist(x, bins=11, range = (-3,4), normed=True, alpha=0.5,

histtype='stepfilled', color='steelblue',

edgecolor='none')

ax[0].plot(\_range, norm.pdf(\_range, np.mean(x), np.std(x)))

ax[0].plot(\_range, st.gaussian\_kde(x)(\_range))

ax[1].hist(y, bins=11, range = (-3,4), normed=True, alpha=0.5,

histtype='stepfilled', color='steelblue',

edgecolor='none')

ax[1].plot(\_range, norm.pdf(\_range, np.mean(y), np.std(y)))

ax[1].plot(\_range, st.gaussian\_kde(y)(\_range))

return plt.show()

**test\_normal(x) - """Тестирование распределения на нормальность""" (100 прогонов теста К-С и без QQ-теста)**

def test\_normal(x):

"""Тестирование распределения на нормальность"""

print("Тест нормальности Шапиро-Вилка")

stat, p = shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))

alpha = 0.05

if p > alpha:

print('Образец выглядит гауссовским (не может отклонить гипотезу H0)')

else:

print('Образец не выглядит гауссовским (отклонить гипотезу H0)')

print("Тест нормальности Д'Агостино-Пирсона")

if len(x) > 19:

x1 = x

else:

x1 = []

for i in range(50):

x1.append(random.choice(x))

stat, p = normaltest(x1)

print('Statistics=%.3f, p=%.3f' % (stat, p))

if p > alpha:

print('Образец выглядит гауссовским (не может отклонить гипотезу H0)')

else:

print('Образец не выглядит гауссовским (отклонить H0)')

print("Тест нормальности Андерсона-Дарлинга")

result = anderson(x)

print('Statistic: %.3f' % result.statistic)

p = 0

for i in range(len(result.critical\_values)):

sl, cv = result.significance\_level[i], result.critical\_values[i]

if result.statistic < result.critical\_values[i]:

print('%.3f: %.3f, Образец выглядит гауссовским (не может отклонить гипотезу H0)' % (sl, cv))

else:

print('%.3f: %.3f, Образец не выглядит гауссовским (отклонить H0)' % (sl, cv))

print("Тест нормальности Колмогорова-Смирнова")

num\_tests = 10\*\*2

num\_rejects = 0

for i in range(num\_tests):

normed\_data = (x - mean(x)) / std(x)

D, pval = stats.kstest(normed\_data, 'norm')

if pval < alpha:

num\_rejects += 1

ratio = float(num\_rejects) / num\_tests

print("Результаты теста Колмогорова-Смирнова: ", "из 100 прогонов доля", '{}/{} = {:.2f} отклоняет гипотезу H0 на уровне отклонения {}'.format(num\_rejects, num\_tests, ratio, alpha))

**test\_normal\_q(x) - """Тестирование распределения на нормальность""" (1000 прогонов теста К-С и с QQ-тестом)**

def test\_normal\_q(x):

"""Тестирование распределения на нормальность"""

print("Тест нормальности Шапиро-Вилка")

stat, p = shapiro(x)

print('Statistics=%.3f, p=%.3f' % (stat, p))

alpha = 0.05

if p > alpha:

print('Образец выглядит гауссовским (не может отклонить гипотезу H0)')

else:

print('Образец не выглядит гауссовским (отклонить гипотезу H0)')

print("Тест нормальности Д'Агостино-Пирсона")

if len(x) > 19:

x1 = x

else:

x1 = []

for i in range(50):

x1.append(random.choice(x))

stat, p = normaltest(x1)

print('Statistics=%.3f, p=%.3f' % (stat, p))

if p > alpha:

print('Образец выглядит гауссовским (не может отклонить гипотезу H0)')

else:

print('Образец не выглядит гауссовским (отклонить H0)')

print("Тест нормальности Андерсона-Дарлинга")

result = anderson(x)

print('Statistic: %.3f' % result.statistic)

p = 0

for i in range(len(result.critical\_values)):

sl, cv = result.significance\_level[i], result.critical\_values[i]

if result.statistic < result.critical\_values[i]:

print('%.3f: %.3f, Образец выглядит гауссовским (не может отклонить гипотезу H0)' % (sl, cv))

else:

print('%.3f: %.3f, Образец не выглядит гауссовским (отклонить H0)' % (sl, cv))

print("Тест нормальности Колмогорова-Смирнова")

num\_tests = 10\*\*3

num\_rejects = 0

for i in range(num\_tests):

normed\_data = (x - mean(x)) / std(x)

D, pval = stats.kstest(normed\_data, 'norm')

if pval < alpha:

num\_rejects += 1

ratio = float(num\_rejects) / num\_tests

print("Результаты теста Колмогорова-Смирнова: ", "из 1000 прогонов доля", '{}/{} = {:.2f} отклоняет гипотезу H0 на уровне отклонения {}'.format(num\_rejects, num\_tests, ratio, alpha))

\_range = np.linspace(0.9 \* np.min(x), 1.1 \* np.max(x), 100)

stats.probplot(x, dist="norm", plot=plt)

return plt.show()

**graph\_kde(xr1, xr2, xr3, xr4) - """Построение 4-х ядерных оценок плотности и кривой Гаусса"""**

def graph\_kde(xr1, xr2, xr3, xr4):

"""Построение 4-х ядерных оценок плотности и кривой Гаусса"""

\_range = np.linspace(0.9 \* np.min(xr1), 1.1 \* np.max(xr1), 100)

plt.plot(\_range, st.gaussian\_kde(xr1)(\_range), color = 'blue')

plt.plot(\_range, st.gaussian\_kde(xr2)(\_range), color = 'red')

plt.plot(\_range, st.gaussian\_kde(xr3)(\_range), color = 'green')

plt.plot(\_range, st.gaussian\_kde(xr4)(\_range), color = 'yellow')

plt.plot(\_range, norm.pdf(\_range, 0, 1), '-.k')

plt.style.use('seaborn-white')

ax.set(xlim=(-4, 4), ylim=(0, 0.5),

xlabel='x', ylabel='',

title='синий график - без нагрузки, красный график - с физ.нагрузкой, \n зеленый график - после отдыха, желтый график - с эмоц.нагрузкой, \n черный штрихпунктирный график - стандартная кривая Гаусса')

plt.show()

**graph\_kde3(xr1, xr2, xr3) - """Построение 3-х ядерных оценок плотности и кривой Гаусса"""**

def graph\_kde3(xr1, xr2, xr3):

"""Построение 3-х ядерных оценок плотности и кривой Гаусса"""

\_range = np.linspace(0.9 \* np.min(xr1), 1.1 \* np.max(xr1), 100)

plt.plot(\_range, st.gaussian\_kde(xr1)(\_range), color = 'blue')

plt.plot(\_range, st.gaussian\_kde(xr2)(\_range), color = 'red')

plt.plot(\_range, st.gaussian\_kde(xr3)(\_range), color = 'green')

plt.plot(\_range, norm.pdf(\_range, 0, 1), '-.k')

plt.style.use('seaborn-white')

ax.set(xlim=(-4, 4), ylim=(0, 0.5),

xlabel='x', ylabel='',

title='синий график - с физ.нагрузкой, красный график - после отдыха , \n зеленый график - с эмоц.нагрузкой, \n черный штрихпунктирный график - стандартная кривая Гаусса')

plt.show()

**def graph\_kde\_all(x, y, u, v, w) - """Построение 4-х ядерных оценок плотности и кривой Гаусса для всех пациентов и эталона w"""**

def graph\_kde\_all(x, y, u, v, w):

"""Построение 4-х ядерных оценок плотности и кривой Гаусса для всех пациентов и эталона w"""

for j in range(len(x)): graph\_kde(sequence\_distance(sequence\_max(x[j]), sequence\_max(w)), sequence\_distance(sequence\_max(y[j]), sequence\_max(w)), sequence\_distance(sequence\_max(u[j]), sequence\_max(w)), sequence\_distance(sequence\_max(v[j]), sequence\_max(w)))

**Терминология**

**Эталон –** ведущий временной ряд:

* либо один из параметров космической погоды(всего 11 эталонов как 11 списков по 60 элементов),
* либо один из временных рядов значений коэффициента симметрии зубца Т электрокардиограммы (всего 18х4=72 эталонов как 72 списка по 60 элементов)

**Образец** *–* ведомый временной ряд:

* либо один из 4 списков (из 60 элементов каждый) временных рядов значений коэффициента симметрии зубца Т электрокардиограммы для 4-х факторов «без нагрузки», «с физической нагрузкой», «после отдыха», «с эмоциональной нагрузкой» для каждого пациента (всего 18х4=72 фактор-образцов для 18 образцов),
* либо один из параметров космической погоды(всего 11 образцов как 11 списков по 60 элементов)

**Фактор-образец** *–* список (из 60 элементов) временного ряда значений коэффициента симметрии зубца Т электрокардиограммы для одного из 4-х факторов «без нагрузки», «с физической нагрузкой», «после отдыха», «с эмоциональной нагрузкой» для каждого пациента (всего 4x18=72 фактор-образцов)

**Групповой фактор-образец -** список, которые получается для каждого из 4-х факторов суммированием по дням соответствующих значений фактор-образцов членов группы (всего 4 списка по 60 элементов).

**Групповой образец** - 4 списка групповых фактор-образцов, которые получается суммированием по дням соответствующих значений фактор-образцов членов группы.

**Групповой эталон** **всех максимумов значений группы эталонов** - список всех максимумов группы эталонов (из 58 элементов).

**Пользовательские действия и алгоритм обработки данных**

***Загрузка данных и их подготовка для анализа***

* *Ввод таблицы «эталонов» из -файла* (либо 11 столбцов по **60 элементов** параметров космической погоды, либо списки (из **60 элементов** каждый) временных рядов значений коэффициента симметрии зубца Т электрокардиограммы для 4-х факторов «без нагрузки», «с физической нагрузкой», «после отдыха», «с эмоциональной нагрузкой» для каждого пациента - всего 18х4=72 фактор-образцов для 18 образцов),

Сформировать *групповой эталон*, который получается суммированием по дням соответствующих значений эталонов группы – *в программе используется функция* sum\_list(x)

(при этом в случае, когда эталоны – параметры погоды, предварительно осуществляется нормировка эталонов с помощью функции norm\_serie(x))

Сформировать по столбцам файла список списков эталонов standart, чтобы использовать мои программы

*В окне «Эталоны» отображать названия эталонов* и должна быть возможность их индивидуального и группового выбора при анализе образцов.

1. *Ввод таблицы «образцов» из -файла* (либо 11 столбцов по **60 элементов** параметров космической погоды, либо списки (из **60 элементов** каждый) временных рядов значений коэффициента симметрии зубца Т электрокардиограммы для 4-х факторов «без нагрузки», «с физической нагрузкой», «после отдыха», «с эмоциональной нагрузкой» для каждого пациента - всего 18х4=72 фактор-образцов для 18 образцов).

Сформировать *групповой образец*, который получается суммированием по дням соответствующих значений образцов членов группы – *в программе используется функция* sum\_list(x)

В частности, для 4-х факторов «без нагрузки», «с физической нагрузкой», «после отдыха», «с эмоциональной нагрузкой» *в программе строятся 4 списка*

sum\_list(sample), sum\_list(sample\_n), sum\_list(sample\_o), sum\_list(sample\_e)

*В окне «Образцы» отображать названия образцов* и *одного группового образца*, а при анализе образцов должна быть возможность, как их индивидуального выбора (целиком или отдельных фактор-образцов), так и группового анализа всех образцов (целиком или отдельных фактор-образцов) с каждым из выбранных эталонов или с выбранной группой эталонов.

В частности, для 4-х факторов «без нагрузки», «с физической нагрузкой», «после отдыха», «с эмоциональной нагрузкой» сформировать по столбцам эталонов 4 списка из 18 списков образцов

sample - список из 18 списков образцов «без нагрузки»,

sample\_n - список из 18 списков образцов «с физ.нагрузкой»,

sample\_o - список из 18 списков образцов «после отдыха»,

sample\_e - список из 18 списков образцов «с эмоц.нагрузкой».

Это позволит использовать мои программы

***Предполагаемые виды анализа данных***

**Для выбранного одного фактор-образца (индивидуального или группового) и одного эталона:**

1. вычисление распределения расстояний от максимумов фактор-образца до максимумов эталона и построение гистограммы, ядерной оценки плотности и кривой Гаусса – *в программе функция*

visual\_analysis(x)

1. тестирование нормальности этого распределения расстояний от максимумов фактор-образца до максимумов эталона – *в программе функция*

test\_normal(x)

1. проведение статистического анализа этого распределения расстояний от максимумов фактор-образца до максимумов эталона – *в программе функция*

stat\_analysis(x,y).

***Отчет: Построение распределений.py*** *(для фактор-образца «без нагрузки»)*

**Для выбранного одного образца (индивидуального или группового) и одного эталона:**

1. вычисление 4-х распределений расстояний от максимумов каждого из факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») до максимумов эталона и построение:
2. рисунка с ядерными оценками плотности 4-х распределений и кривой Гаусса – *в программе функция*

graph\_kde(xr1, xr2, xr3, xr4)

1. для каждого из 4-х распределений (на отдельных рисунках) гистограммы, ядерной оценки плотности и кривой Гаусса – *в программе функция*

visual\_analysis(x)

1. тестирование нормальности 4-х распределений расстояний от максимумов факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») до максимумов эталона – *в программе функция*

test\_normal(x)

1. проведение статистического анализа 4-х распределений расстояний от максимумов факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») до максимумов эталона – *в программе функция*

stat\_analysis(x,y).

***Отчет-совместные фактор-образцы-сокращен.py*** *(для фактор-образца «без нагрузки»)*

1. в резерве: вычисление 3-х распределений расстояний от максимумов 3-х фактор-образцов («с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») до максимумов исходного стандарта «без нагрузки», для которых необходимо:
2. построение для этих распределений 3-х ядерных оценок плотности и кривой Гаусса – *в программе функция*

graph\_kde3(xr1, xr2, xr3)

1. тестирование нормальности – *в программе функция*

test\_normal(x)

1. проведение статистического анализа – *в программе функция*

stat\_analysis(x,y).

***Отчет-сравнение фактор-образцов-сокращен.py***

**Для выбранной группы образцов и одного эталона:**

1. вычисление 4-х распределений по 18 средних значений членов группы от максимумов каждого из факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») до максимумов эталона – *в программе вычисляются списки:*

max\_sample\_list, max\_sample\_list\_n, max\_sample\_list\_o, max\_sample\_list\_e

1. построение рисунка с ядерными оценками плотности этих 4-х распределений и кривой Гаусса – *в программе функция*

graph\_kde(xr1, xr2, xr3, xr4)

1. построение для каждого из этих 4-х распределений (на отдельных рисунках) гистограммы, ядерной оценки плотности и кривой Гаусса visual\_analysis(x)
2. тестирование нормальности этих 4-х распределений – *в программе пункты 2)-4) выполнены в одном цикле только для одного группового фактор-образца «без нагрузки» и для распределения средних значений группы для фактор-образца «без нагрузки»":*

***Отчет-групповой анализ средних значений фактор-образцов-сокращен.py***

**Для выбранного одного фактор-образца (индивидуального или группового) и группы эталонов:**

1. вычисление распределений расстояний от максимумов фактор-образца до максимумов всех эталонов группы – *в программе функция используются функции*

sequence\_max(x), sequence\_distance(x, y) и sequence\_distance1(x, y)

1. тестирование нормальности – *в программе функция*

test\_normal(x)

1. проведение статистического анализа этих распределений расстояний от максимумов фактор-образца до максимумов всех эталонов группы – *в программе функция*

stat\_analysis(x,y).

***Отчет: Построение распределений.py*** *(для фактор-образца «без нагрузки»)*

**Для выбранного одного образца (индивидуального или группового) и группы эталонов:**

1. для каждого из факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») вычисление распределений расстояний от максимумов факторов-образца до максимумов до максимумов всех эталонов группы – *в программе функция используются функции*

sequence\_max(x), sequence\_distance(x, y) и sequence\_distance1(x, y)

1. проведение статистического анализа этих распределений расстояний от максимумов всех фактор-образцов до максимумов всех эталонов группы – *в программе функция*

stat\_analysis(x,y).

***Отчет: Построение распределений.py*** *(для фактор-образца «без нагрузки»)*

**Для выбранного фактор-образца и группы эталонов:**

1. вычисление 11-и распределений по 18 средних значений членов группы от максимумов фактор-образца («без нагрузки», «с физ.нагрузкой», «после отдыха» или «с эмоц.нагрузкой») до максимумов эталонов – *в программе вычисляются списки:*

max\_sample\_list, max\_sample\_list\_n, max\_sample\_list\_o, max\_sample\_list\_e

1. построение графиков с гистограммами, ядерными оценками плотности этих 11-и распределений и кривой Гаусса – *в программе функция*

graph\_kde(xr1, xr2, xr3, xr4)

1. тестирование нормальности этих 11-и распределений – *в программе функция* test\_normal(x)
2. проведение статистического анализа этих средних значений распределений расстояний от максимумов всех фактор-образцов группы всех образцов до максимумов всех эталонов группы– *в программе функция*

stat\_analysis(x,y).

***Отчет-групповой анализ средних значений.py*** *(для фактор-образца «без нагрузки»)*

**Для выбранной группы образцов и группы эталонов:**

1. для каждого из факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») группы всех образцов вычисление распределений расстояний от максимумов факторов-образца до максимумов всех эталонов группы – *в программе функция используются функции*

sequence\_max(x), sequence\_distance(x, y) и sequence\_distance1(x, y)

1. проведение статистического анализа этих распределений расстояний от максимумов всех фактор-образцов группы всех образцов до максимумов всех эталонов группы– *в программе функция*

stat\_analysis(x,y).

**Для выбранной группы образцов и группы эталонов – групповой анализ средних значений:**

1. для каждого из факторов-образцов («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») группы всех образцов вычисление распределений средних значений расстояний от максимумов факторов-образца до максимумов всех эталонов группы – *в программе функция используются функции*

sequence\_max(x), sequence\_distance(x, y) и sequence\_distance1(x, y)

1. тестирование нормальности этих 11-и распределений – *в программе функция* test\_normal(x)
2. проведение статистического анализа этих средних значений распределений расстояний от максимумов всех фактор-образцов группы всех образцов до максимумов всех эталонов группы– *в программе функция*

stat\_analysis(x,y).

***Отчет-групповой анализ средних значений.py*** *(для фактор-образца «без нагрузки»)*

**Для выбранной группы эталонов и группы образцов – групповой анализ средних значений:**

1. для каждого эталона из группы вычисление распределений расстояний от максимумов эталона до максимумов группового факторов-образца («без нагрузки», «с физ.нагрузкой», «после отдыха», «с эмоц.нагрузкой») – *в программе функция используются функции*

sum\_list(x), sequence\_max0(x), sequence\_distance(x, y), visual\_analysis(x)

1. тестирование нормальности этих 11-и распределений – *в программе функция* test\_normal(x)
2. проведение статистического анализа этих распределений расстояний от максимумов эталона до максимумов группового факторов-образца – *в программе функция*

stat\_analysis(x).

***Отчет - групповой эталон-образец анализ.py*** *(для фактор-образца «без нагрузки»)*

***Отчет - групповой\_эталон-образец\_анализ-амплитуд.py*** *(для фактор-образца «без нагрузки»)*

*Анализ синхронизации максимумов значений эталонов с максимумами каждого фактор-образца без нагрузки*