МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт № 3 «Системы управления, информатика и электроэнергетика»

Кафедра 311 «Прикладные программные средства и математические методы»

КУРСОВАЯ РАБОТА

Дисциплина: «Алгоритмы и обработка данных»

Выполнила:

Студентка гр. М3О-216Б-22

Хутиева Эрика Арсеновна

Научный руководитель:

Проф. Агамиров Л.В.

Оценка КР(КП): ______________

Москва 2023г.

Оглавление

1. Метод наименьших квадратов 3
2. Определение квантиля и дисперсии 5
3. Планирование дисперсии 6
4. Метод максимального правдоподобия 8
5. Распределение Вейбула-Гнеденко 10
6. Критерий Граббса. 11
7. Критерий Фишера (F-критерий) – об однородности двух дисперсий 13
8. Критерий Стьюдента (t-критерий) 15
9. Критерий Бартлета 17
10. Однофакторный дисперсионный анализ 18
11. Критерий Шапиро-Уилка 20
12. Критерий Смирнова 22
13. Критерий Андерсона-Дарлинга 23
14. Критерий $\chi\mathbf{2}$ 24
15. Критерий знаков для медианы 25
16. Двухвыборочный критерий Уилкоксона 26
17. Критерий Колмогорова-Смирнова 27
18. Критерий Краскела-Уоллиса 28

Все программы вы можете найти по ссылке

https://github.com/erikahutieva/Python/tree/main/matstat

Метод наименьших квадратов

Пусть имеется линейная модель [2,9]:

$$y = X \cdot b + \varepsilon, \qquad (2.59)$$

где y - вектор-столбец наблюдений размерности n, X - матрица размерности $n \times k_1$ известных коэффициентов ($n > k_1$), b - вектор столбец параметров размерности k_1 и ε - вектор-столбец случайных «ошибок» размерности n с нулевым математическим ожиданием и матрицей рассеяния размерности $n \times n$:

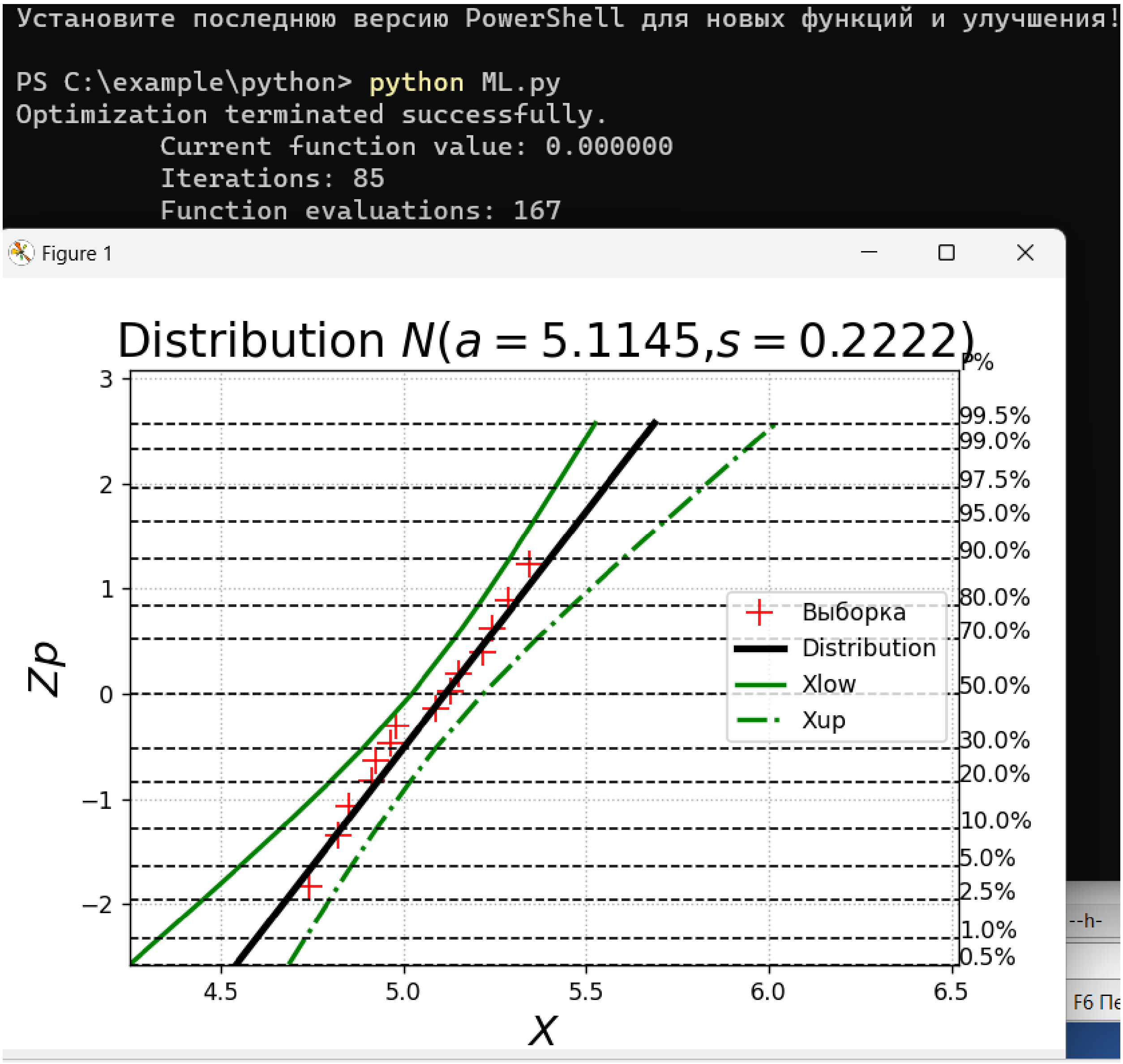
$$D(\varepsilon) = \sigma^2 \cdot V. \qquad (2.60)$$

Это означает, что случайные ошибки наблюдений не коррелированы, но имеют различные дисперсии. Метод наименьших квадратов состоит в минимизации скалярной суммы квадратов:

$$S = (y - X \cdot b)^T \cdot V^{-1} \cdot (y - X \cdot b) \qquad (2.61)$$

```
def MLS_Normal():#метод меньших квадратов
    finp=open("Inp\MLS_Normal.inp")
    finp.readline()
    n=int(finp.readline())
    ss=finp.readline()
    beta=float(finp.readline())
    ss=finp.readline()
    y=tuple(map(float,finp.readline().split(" ")))
    ss=finp.readline()
    kp=int(finp.readline())
    ss=finp.readline()
    p=tuple(map(float,finp.readline().split(" ")))
    finp.close()

    fout=open('Out\MLS_Normal.out','w')
    n=len(y)
    print("MO and Std by observed values",file=fout)

    b,db=mlsordern(n,y)
    a=b[0]
    s=b[1]

    print("a=",a,file=fout)
    print("s=",s,file=fout)
    print("Sample size n=",n,file=fout)
    prints("Sample:",y,fout)

    w=cum1(n)
    print("Observed values sample size n=",len(w),file=fout)
    prints("Empirical probability:",w,fout)
    w=stats.norm.ppf(w)
    zp=stats.norm.ppf(p)
    delta=zp*np.sqrt(n)
    print("Tolerance probability=",beta,file=fout)
    f=n-1

    db=db*n
    print("D{b}=",db,file=fout)
    t1=db[0][0];t2=db[1][1];t12=db[0][1]

    print("Tolerance probability=",beta,file=fout)
    tlow,tup=nctlimit_app(n,beta,zp,t1,t2,t12)

    #tlow,tup=nctlimit_exact(f,beta,delta)

    prints("Probability range:",p,fout)
    prints("Normal quantiles:",w,fout)
    prints("Upper non central t quantile",tup,fout)
    prints("Low non central t quantile",tlow,fout)
    xp=a+s*zp
    xpup=a+s*tup/np.sqrt(n)
    xplow=a+s*tlow/np.sqrt(n)
    prints("Upper tolerance limit",xpup,fout)
    prints("Quantile estimations",xp,fout)
    prints("Low tolerance limit",xplow,fout)
    fout.close()

    show_distr("Normal",True,True,y,w,xp,zp,xplow,zp,xpup,zp,grid_size=n,distr_name=r'$N({a=}$'+str(round(a,4))+",${s=}$"+str(round(s,4))+")")
```

Функция не принимает ничего на вход, но читает данные или файла INP, а выводит верхние и нижние пределы, а также оценки квантиля графическое представление выборочных данных и полученного нормального распределения.



Определение квантиля и дисперсии

Квантиль распределения определяется соотношением:

$$F(x_p) = P. \tag{2.3}$$

или в соответствии с (2.1)

$$P\{X \le x_p\} = P. \tag{2.4}$$

Квантиль x_p уровня P представляет собой значение случайной величины, вероятность не превышения которого равна P. Следовательно, доля значений случайной величины в генеральной совокупности, не превышающих x_p, равна P. Квантиль $x_{0,5}$ уровня P=0,5 называется медианой распределения.

Планирование дисперсии

Матрица рассеяния оценок b [9] определяется из следующего уравнения:

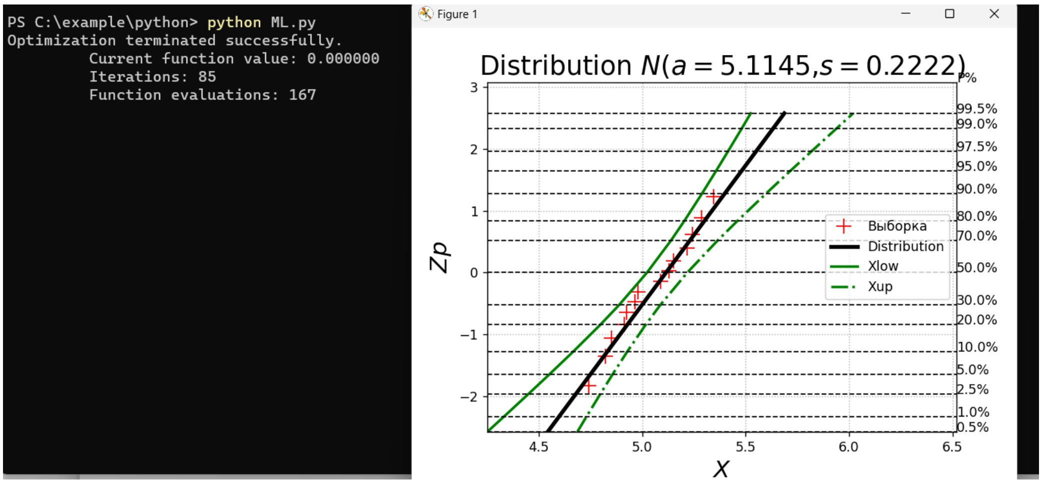
$$D(\bar{b}) = (\nu) = \frac{\sigma^2}{n}(\nu^*);\ (\nu^*) = n \cdot (X^T \cdot V^{-1} \cdot X)^{-1} \quad , \tag{2.64}$$

где несмещенная оценка для остаточной дисперсии σ^2 определяется формулой:

$$\bar{\sigma}^2 = s^2 = \frac{1}{n-k_1} \cdot (y - X \cdot \bar{b})^T \cdot V^{-1} \cdot (y - X \cdot \bar{b}). \tag{2.65}$$

```
def Plan_Disp():
    txt="Inp/Plan_Disp.inp"
    finp=open(txt)
    st=finp.readline()
    beta=list(map(float,finp.readline().split()))
    st=finp.readline()
    kstart=int(finp.readline())
    st=finp.readline()
    kfinish=int(finp.readline())
    finp.close()

    kb=len(beta)
    txt="Out/Plan_Disp.out"
    fout=open(txt,'w')
    print("Sample size to evaluate the variance with beta=",file=fout)
    print(beta,file=fout)
    for i in range(kb):
        print("Beta=",beta[i],file=fout)
        bb1=1.-beta[i]
        bb2=1.+beta[i]
        for j in range(kstart,kfinish+1):
            b1=stats.chi2.ppf(1-0.5*bb1,j)
            b2=stats.chi2.ppf(1-0.5*bb2,j)
            delta =np.sqrt(b1/b2)-1.
            print("f=",j,": delta=",delta,file=fout)
    fout.close()
```

Уравнения (2.63), (2.64) позволяют оценивать параметры расположения (сдвига) и масштаба, исходя из порядковых статистик, то есть выборочных наблюдений, упорядоченных по величине.

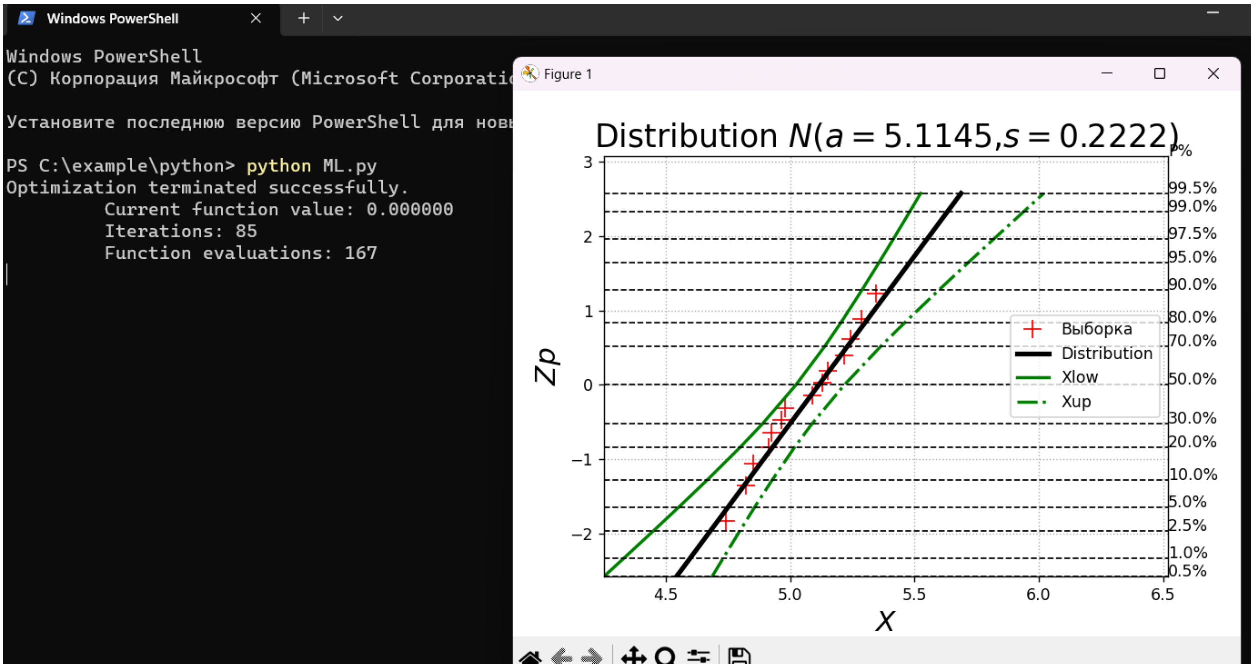

PS C:\example\python> python ML.py
Optimization terminated successfully.
Current function value: 0.000000
Iterations: 85
Function evaluations: 167
Figure 1
Distribution N(a = 5.1145,s = 0.2222)
Zp
X
P%
99.5%
99.0%
97.5%
95.0%
90.0%
80.0%
70.0%
50.0%
30.0%
20.0%
10.0%
5.0%
2.5%
1.0%
0.5%
Выборка
Distribution
Xlow
Xup
4.5
5.0
5.5
6.0
6.5
3
2
1
0
−1
−2

Метод максимального правдоподобия

В соответствии с методом максимального правдоподобия (ММП) [1-4] оценки параметров непрерывной, не менее двух раз дифференцируемой, функции распределения случайной величины, в общем случае прогрессивно цензурированной выборки определяются решением системы уравнений максимального правдоподобия. Оценки максимального правдоподобия (ММП-оценки) определяются в точках экстремума функции максимального правдоподобия:

$$L = \prod_{i=1}^{k} f_x(x_i) \cdot \prod_{j=1}^{m} \left[1 - F_x(x_{бj})\right]^{r_j}, \tag{2.39}$$

```
def MLE_Minimize():
    finp=open("Inp\MLE_Normal.inp")
    finp.readline()
    n=int(finp.readline())
    ss=finp.readline()
    beta=float(finp.readline())
    ss=finp.readline()
    y=tuple(map(float,finp.readline().split(" ")))
    ss=finp.readline()
    r=tuple(map(int,finp.readline().split(" ")))
    ss=finp.readline()
    kp=int(finp.readline())
    ss=finp.readline()
    p=tuple(map(float,finp.readline().split(" ")))
    finp.close()

    fout=open('Out\MLE_Normal.out','w')
    n=len(y)
    k=sum(r)
    m=n-k

    print("MO and Std by observed values",file=fout)

    yy=tuple(map(float,(y[i] for i in range(n)  if(r[i]==0))))
    xx=tuple(map(float,(y[i] for i in range(n)  if(r[i]==1))))

    cp=np.average(yy)
    cko=np.std(yy)
    print("a0=",cp,file=fout)
    print("s0=",cko,file=fout)

    bnds = ((0, None), (0, None))
    res = minimize(NormalMinFunction,(cp,cko),args=(xx,m,cp,cko), method='Nelder-Mead',bounds=bnds, tol=1e-12,options={'disp': True})

    mo=res.x[0]
    s=res.x[1]

    print("MO and Std by MLE",file=fout)
    print("cp=",mo,"cko=",s,file=fout)
    print("FunMin="+str(NormalMinFunction(res.x,xx,m,cp,cko)),file=fout)

    print("Sample size n=",n,file=fout)
    prints("Sample:",y,fout)

    ycum,w=cum(n,y,r)
    v=CovMatrixMleN(n,y,r,mo,s)

    print("Observed values sample size m=",len(w),file=fout)
    prints("Observed values:",ycum,fout)

    print("Censorized sample size k=",k,file=fout)
    prints("Censorized values:",xx,fout)
    prints("Kaplan-Meier probability:",w,fout)

    w=stats.norm.ppf(w)
    zp=stats.norm.ppf(p)
    print("Tolerance probability=",beta,file=fout)
    t1=v[0][0];t2=v[1][1];t12=v[0][1]
    tlow,tup=nctlimit(n,beta,zp,t1,t2,t12)
    prints("Probability range:",p,fout)
    prints("Normal quantiles:",w,fout)
    prints("Upper non central t quantile",tup,fout)
    prints("Low non central t quantile",tlow,fout)
    xp=mo+s*zp
    xpup=mo+s*tup/np.sqrt(n)
    xplow=mo+s*tlow/np.sqrt(n)
    prints("Upper tolerance limit",xpup,fout)
    prints("Quantile estimations",xp,fout)
    prints("Low tolerance limit",xplow,fout)
    fout.close()

    show_distr("Normal",True,True,ycum,w,xp,zp,xplow,zp,xpup,zp,grid_size=n, distr_name=r'$N({a=}$'+str(round(mo,4))+",${s=}$"+str(round(s,4))+")")
```


Windows PowerShell
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporati
Установите последнюю версию PowerShell для новы
PS C:\example\python> python ML.py
Optimization terminated successfully.
Current function value: 0.000000
Iterations: 85
Function evaluations: 167
Figure 1
Distribution N(a = 5.1145,s = 0.2222)
P%
Zp
X
Выборка
Distribution
Xlow
Xup
99.5%
99.0%
97.5%
95.0%
90.0%
80.0%
70.0%
50.0%
30.0%
20.0%
10.0%
5.0%
2.5%
1.0%
0.5%
3
2
1
0
−1
−2
4.5
5.0
5.5
6.0
6.5

Распределение Вейбула-Гнеденко

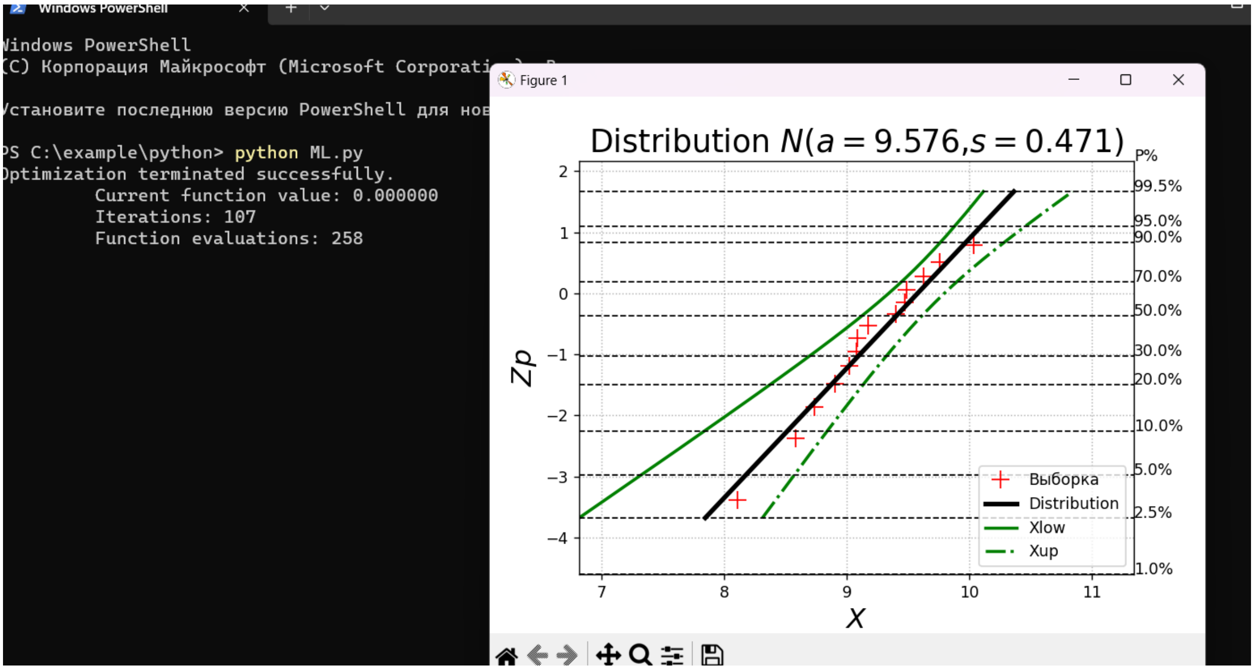
ММП-оценки параметров b, x_0 распределения Вейбулла-Гнеденко (2.21), (2.22) в соответствии с уравнениями (2.40) рассчитывают как корни системы уравнений:

$$\left.\frac{d\ln L}{db}\right|_{b=\bar{b}} = \left[\frac{k}{\bar{b}} + \sum_{i=1}^{k}\ln(x_i - \bar{x}_0)\right]\cdot\left[\sum_{i=1}^{k}(x_i - \bar{x}_0)^{\bar{b}} + \sum_{j=1}^{m} r_j\cdot(x_{бj} - \bar{x}_0)^{\bar{b}}\right] - \\ - k\cdot\left[\sum_{i=1}^{k}(x_i - \bar{x}_0)^{\bar{b}}\cdot\ln(x_i - \bar{x}_0) + \sum_{j=1}^{m} r_j\cdot(x_{бj} - \bar{x}_0)^{\bar{b}}\cdot\ln(x_{бj} - \bar{x}_0)\right] = 0, \quad (2.56)$$

$$\left.\frac{d\ln L}{dx_0}\right|_{x_0=\bar{x}_0} = \sum_{i=1}^{k}(x_i - \bar{x}_0)^{\bar{b}-1} + \sum_{j=1}^{m} r_j\cdot(x_{бj} - \bar{x}_0)^{\bar{b}-1} - \\ - \frac{1-\frac{1}{\bar{b}}}{k}\cdot\sum_{i=1}^{k}(x_i - \bar{x}_0)^{-1}\cdot\left[\sum_{i=1}^{k}(x_i - \bar{x}_0)^{\bar{b}} + \sum_{j=1}^{m} r_j\cdot(x_{бj} - \bar{x}_0)^{\bar{b}}\right] = 0, \quad (2.57)$$

после чего оценку параметра c определяют из уравнения:

$$\bar{c}^{\bar{b}} = \frac{1}{k}\cdot\left[\sum_{i=1}^{k}(x_i - \bar{x}_0)^{\bar{b}} + \sum_{j=1}^{m} r_j\cdot(x_{бj} - \bar{x}_0)^{\bar{b}}\right]. \quad (2.58)$$



Критерий Граббса

Критерий Граббса применяют в тех случаях, когда имеются статистические данные по рассматриваемой выборке. Для этого рассчитывают статистики:

$$u_1 = \frac{\bar{x} - x_1}{s}, \; u_n = \frac{x_n - \bar{x}}{s}$$

где x, s - выборочное среднее и среднее квадратичное отклонение, n x , x 1 - крайние члены вариационного ряда.

Рассчитанное значение u сопоставляют с критическим uα для заданного уровня значимости α и объема выборки n. Критические значения определяются из уравнения:

$$u_\alpha = (n-1) \cdot \sqrt{\frac{t^2_{\alpha/n,n-2}}{n \cdot \left(n - 2 + t^2_{\alpha/n,n-2}\right)}}$$

где $t^2_{\alpha/n,n-2}$ - квантиль распределения Стьюдента уровня α/n с числом степеней свободы $f = n-2$. Для двустороннего критерия α/n заменяют на $\alpha/2n$.

Нулевую гипотезу принимают, если u ≤ uα и отвергают в противном случае.

```python
#Критерий Грабса аномальный результат,наблюденияб резко выделяющиеся
def grubs_test(sample1):
    print('Критерий Грабса')
    n = len(sample1)  # Объем выборки
    mean = np.mean(sample1)  # Среднее значение
    std = np.std(sample1)  # Стандартное отклонение

    # Вычисляем критическое значение
    critical = t.ppf(1 - 0.05 / (2 * n), n - 2)

    # Вычисляем критерий Грабса
    max_otkl = np.max(np.abs(sample1 - mean))  # Максимальное отклонение
    g = max_otkl / std  # Считаем значение критерия Граббса
    print("Критерий Грабса:", g)
    print("Критическое значение:", max_otkl)

    # Проверяем гипотезу: если значение критерия Граббса меньше критического значения, то гипотезу принимают, иначе - не принимают.
    if g < max_otkl:
        print("Гипотезу принимают")
    else:
        print("Гипотезу не принимают")

    alpha = 0.05  # Уровень значимости
    dvustor_g = grubbs.test(sample1, 0.05)  # Двусторонний критерий Граббса
    print('Двусторонний критерий Граббса',dvustor_g)
    mini = grubbs.min_test(sample1, alpha)  # Односторонний критерий Граббса для минимального значения
    maxi = grubbs.max_test(sample1, alpha)  # Односторонний критерий Граббса для максимального значения
    print('Односторонний критерий Граббса для минимального и максимального значения', mini,maxi)
    # Определяем индексы и значения аномальных наблюдений
    index = grubbs.two_sided_test_indices(sample1, alpha)
    anom = grubbs.two_sided_test_outliers(sample1, alpha)
    print("Индексы отклоняющихся чисел: ", index)
```

Функция принимает на вход выборку, а выводит статистику критерий Граббса, критические значения, сообщение о том, принимают ли гипотезу, двусторонний критерий Граббса, односторонний критерий Граббса для максимального и минимального значения, индексы отклоняющихся чисел.

```
PS C:\example\python> python Shapirpy.py
Критерий Грабса
Критерий Грабса: 2.1170014234622236
Критическое значение: 0.638418032820474
Гипотезу не принимают
Двусторонний критерий Граббса [3.74809518 4.76911046 4.45256609 4.20623284 4.02841205 3.68617029
 4.05605011 4.24351705 3.990043   4.12672721]
Односторонний критерий Граббса для минимального и максимального значения [3.74809518 4.76911046 4.45256609 4.20623284 4.
02841205 3.68617029
 4.05605011 4.24351705 3.990043   4.12672721] [3.74809518 4.76911046 4.45256609 4.20623284 4.02841205 3.68617029
 4.05605011 4.24351705 3.990043   4.12672721]
Индексы отклоняющихся чисел:  []
```

Критерий Фишера (F-критерий) – об однородности двух дисперсий

Дисперсии двух совокупностей объемами n1 и n2 , подчиняющихся нормальному (логарифмически нормальному) закону распределения, сравнивают с помощью двустороннего критерия F. Для этого рассчитывают дисперсионное отношение F по формуле:

$$F = \frac{s_1^2}{s_2^2} \text{ - при } s_1^2 > s_2^2$$

или

$$F = \frac{s_2^2}{s_1^2} \text{ - при } s_2^2 > s_1^2,$$

где s_1^2, s_2^2 - выборочные дисперсии.

Дисперсионное отношение F сопоставляют с критическим значением Fα для заданного уровня значимости α и чисел степеней свободы f1 = n1 –1, f2 = n2 –1 , где f1- число степеней свободы для большей дисперсии. В случае соблюдения условия F ≤ Fα , принимают нулевую гипотезу о равенстве генеральных дисперсий. В противном случае нулевая гипотеза отвергается.

```
def fishsher(sample1,sample2):
    print('Критерий Фишера (F критерий)')
    n1=len(sample1)
    n2=len(sample2)
    # Вычисляем дисперсию и выборочные средние
    var1 = np.var(sample1, ddof=1)
    var2 = np.var(sample2, ddof=1)
    mean1 = np.mean(sample1)
    mean2 = np.mean(sample2)
    # Определяем какая выборка имеет большую дисперсию
    if var1>var2:
       sx=var1
       sy=var2
       nx=n1
       ny=n2
    else:
       sx=var2
       sy=var1
       nx=n2
       ny=n1

   # Вычисляем дисперсионное отношение
    f_stat = sx/ sy
    print("F-статистика:", f_stat)
    # Определяем критическое значение дисперсионного отношения
    alpha = 0.05 #-точность
    crit_value = stats.f.ppf(1-alpha, nx-1, ny-1)
    #Сравниваем с критическим значением дисперсионного отношения
    if f_stat <= crit_value:
        print("Принимают")
    else:
        print("Отвергают")
```

Функция принимает 2 выборки, а выводит F-статистику (дисперсионное отношение), критическое значение и сообщение о том, принимают ли нулевую гипотезу.

```
PS C:\example\python> python Shapirpy.py
Критерий Фишера (F критерий)
F-статистика: 1.410019141229893
Принимают
```

Критерий Стьюдента (t-критерий)

Критерий Стьюдента применяют для сравнения средних значений двух нормально распределенных совокупностей при неизвестных, но равных дисперсиях (σ1)^2 = (σ2)^2 . Нулевая гипотеза заключается в предположении о равенстве средних H0 : a = a . Для проверки этой гипотезы рассчитывают статистику t:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s \cdot \sqrt{\dfrac{1}{n_1} + \dfrac{1}{n_2}}},$$

где

$$s^2 = \frac{f_1 \cdot s_1^2 + f_2 \cdot s_2^2}{f_1 + f_2}.$$

Полученное значение t критерия сравнивают с критическим для уровня значимости α и числа степеней свободы f = f1 + f2 . Если t ≤ t(α/2) , то нулевую гипотезу о равенстве средних принимают. В противном случае a1 ≠ a2 .

```
def st(sample1,sample2 ):
    print('Критерий Стьюдента')
    n1=len(sample1)
    n2=len(sample2)
    # Вычисляем дисперсию и выборочные средние
    var1 = np.var(sample1, ddof=1)
    var2 = np.var(sample2, ddof=1)
    mean1 = np.mean(sample1)
    mean2 = np.mean(sample2)
    # Определяем какая выборка имеет большую дисперсию
    if var1>var2:
       sx=var1
       sy=var2
       nx=n1
       ny=n2
    else:
       sx=var2
       sy=var1
       nx=n2
       ny=n1
    # Находим статистику t
    f1=n1-1 #Число степеней свободы
    f2=n2-1
    if flag==0 :
        f=f1+f2
        skv= (f1 * var1 + (f2) * var2) / (f)
        t =abs( (mean1 - mean2) / math.sqrt(skv* (1/n1 + 1/n2)))#находим статистику

    else:
        t = abs((mean1 - mean2) / math.sqrt((var1/n1 + var2/n2)))
        c= (var1/n1)/(var1/n1+var2/n2)
        f=1/(c**2/f1+(1-c**2)/f2)
    # Находим критическое значение для уровня значимости альфа
    t_alpha=stats.t.ppf(1-alpha/2,f)
    print("Статистика",t)
    print("Критическое для уровня значимости альфа", t_alpha)
    # Сравниваем со статистикой t
    if t<t_alpha:
        print("Принимается")
    else:
        print("Отвергается")
```

Функция принимает на вход две выборки, а выводит статистику критерия t, его критическое значение и сообщение о том, принимают ли гипотезу о равенстве средних.

```
PS C:\example\python> python Shapirpy.py
Критерий Стьюдента
Статистика 0.2544332372045981
Критическое для уровня значимости альфа 2.2621571627409915
Принимается
```

Критерий Бартлета

Однородность (равенство) дисперсий ряда выборок из нормально распределенных совокупностей оценивают с помощью критерия Бартлета. Для этого рассчитывают статистику критерия по формуле:

$$\chi^2 = \frac{1}{c} \cdot \left[\ln(s^2) \cdot \left(\sum_{i=1}^{m} n_i - m \right) - \sum_{i=1}^{m} (n_i - 1) \cdot \ln s_i^2 \right],$$

где m - количество выборок, s_i^2 - выборочная дисперсия,

$$c = 1 + \frac{1}{3 \cdot (m-1)} \cdot \left[\sum_{i=1}^{m} \frac{1}{n_i - 1} - \frac{1}{\sum_{i=1}^{m} n_i - m} \right],$$

$$s^2 = \frac{\sum_{i=1}^{m} (n_i - 1) \cdot s_i^2}{\sum_{i=1}^{m} n_i - m}.$$

Если $\chi^2 \le \chi^2_{\alpha, f=m-1}$, то нулевая гипотеза об однородности ряда дисперсий подтверждается. В противном случае принимается альтернативная гипотеза о неравенстве дисперсий.

```
def bartlett_test(sample1,sample2):
    # Вычисляем критерий Бартлетта
    bartle, p_value = stats.bartlett(sample1,sample2)
    print("Статистика теста:", bartle)
    print("p-значение:", p_value)
    if  p_value<0.05:
         print("Принимается")
     else:
         print("Отвергается")
```

Функция принимает 2 выборки, а возвращает статистику критерия Бартлета, критическое значение, сообщение о том, принимается ли нулевая гипотеза.

```
PS C:\example\python> python Shapirpy.py
Статистика теста: 2.778447813093432
p-значение: 0.09554072173703637
Отвергается
```

Однофакторный дисперсионный анализ

Равенство (однородность) ряда средних значений оценивают с помощью однофакторного дисперсионного анализа. В основе его лежит предположение о нормальности закона распределения случайной величины в каждой выборке и однородности ряда дисперсий. Проверка нулевой гипотезы о равенстве всех средних производят с помощью F- критерия дисперсионного отношения:

$$F = \frac{s_1^2}{s_2^2}\ , \tag{3.13}$$

где s_1^2 - дисперсия между m выборками объемом n_i, характеризующая рассеяние по факторам,
s_2^2 - внутренняя дисперсия, характеризующая внутреннее рассеяние, связанное со случайными колебаниями внутри каждой выборки.

$$s_1^2 = \frac{\sum_{i=1}^{m} n_i \cdot (\bar{x}_i - \bar{a})^2}{f_1}, \quad f_1 = m - 1\ , \tag{3.14}$$

$$s_2^2 = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(x_{i,j} - \bar{x}_i\right)^2}{f_2} = \frac{\sum_{i=1}^{m}(n_i - 1)\cdot s_i^2}{f_2}, \quad f_2 = \sum_{i=1}^{m} n_i - m\ , \tag{3.15}$$

$$\bar{a} = \frac{\sum_{i=1}^{m} n_i \cdot \bar{x}_i}{\sum_{i=1}^{m} n_i}\ . \tag{3.16}$$

Если дисперсионное отношение (3.13) окажется меньше критического значения Fα критерия Фишера, найденного для уровня значимости α и чисел степеней свободы f2 , f1 , то нулевая гипотеза о равенстве средних a1 = a2 = ... = am = a подтверждается

```
#Однофакторный дисперсионный анализ
def fishsher2(sample1,sample2,sample3):
    print('Критерий Фишера для нескольких выборок')

    # Выполнение однофакторного дисперсионного анализа распространен на несколько групп
    F, p_value = f_oneway(sample1, sample2, sample3)

    # Объединение выборок в одну
    vib = np.concatenate([sample1, sample2, sample3])

    # Вычисление выборочного общего среднего
    sr = np.mean(vib)

    # Вывод результатов
    print("Выборочное общее среднее, a: ", sr)
    print("F: ", F)
    print("p_value: ", p_value)
```

Функция принимает 3 выборки, а выводит выборочное общее среднее, F – статистику и p_value.

```
Критерий Фишера для нескольких выборок
Выборочное общее среднее, a:  3.9721457972219434
F:  4.343251798367742
p_value:  0.023152975840800526
```

Критерий Шапиро-Уилка

Критерий Шапиро-Уилка (W-критерий) предназначен для проверки гипотезы о нормальном (логарифмически нормальном) распределении. Результаты испытаний располагают в вариационный ряд и вычисляют статистику критерия:

$$W = \frac{b^2}{s^2},$$

где s^2 вычисляется по формуле:

$$s^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2,$$

а величина оценки b определяется по уравнению [15,16]:

$$b = \sum_{i=1}^{n} a_i \cdot x_i, \quad a_i = \frac{\sum_{j=1}^{n} \alpha_j \cdot v_{i,j}}{\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{n} \alpha_j \cdot v_{i,j}\right)^2}},$$

Если W больше критического значения $W\alpha$ для объема выборки n, то нулевая гипотеза принимается. В противном случае принимается альтернативная гипотеза.

```
def shapiro_test(sample1):
    print('Критерий Шапиро-Уилка')
    # Применение критерия Шапиро-Уилка
    stat, p = shapiro(sample1)

    # Вывод результатов
    print('Статистика критерия Шапиро-Уилка :', stat)
    print('p-значение критерия Шапиро-Уилка :', p)
    alpha = 0.05
    if p > alpha:
        print('Принимаем гипотезу: данные распределены нормально')
    else:
        print('Отвергаем гипотезу: данные не распределены нормально')
```

Функция принимает выборку, а выводит статистику критерия Шапиро-Уилка, p-значение и сообщение о том, нормальная ли выборка.

```
Критерий Шапиро-Уилка
Статистика критерия Шапиро-Уилка : 0.9836090207099915
p-значение критерия Шапиро-Уилка : 0.9815335273742676
Принимаем гипотезу: данные распределены нормально
PS C:\example\python>
```

Критерий Смирнова

Критерий Смирнова рекомендуется использовать для проверки гипотезы о соответствии эмпирического распределения нормальному закону. Для этого вычисляют статистику:

$$\omega^2 = \frac{1}{12 \cdot n} + \sum_{i=1}^{n} \left[F(x_i) - W(x_i) \right]^2, \qquad (3.20)$$

где $W(x_i)$ - накопленная частость (см. раздел 2.8.) и составляют неравенство:

$$\omega^2 \cdot (1 + 0{,}5 \cdot n) \le \Omega_\alpha . \qquad (3.21)$$

Если неравенство (3.21) выполняется, то нулевая гипотеза принимается, в противном случае нулевая гипотеза отвергается.

```
def smirnov_test(sample1):
    print('Критерий Колмогорова-Смирнова')
    statistic, p_value = kstest(sample1, 'norm')
    print('statistic:', statistic)
    print('p-value:', p_value)
    alpha = 0.05
    if p > alpha:
        print('Принимаем гипотезу: данные распределены нормально')
    else:
        print('Отвергаем гипотезу: данные не распределены нормально')
```

Функция принимает выборку, а выводит статистику критерия Смирнова, p-значение и сообщение о том, нормальная ли выборка

```
Критерий Колмогорова-Смирнова
statistic: 0.9993975473982702
p-value: 1.2596759510078808e-32
Принимаем гипотезу: данные распределены нормально
```

Критерий Андерсона-Дарлинга

Критерий Андерсона-Дарлинга используют для проверки нормальности в тех случаях, когда больший интерес представляет соответствие эмпирической функции распределения теоретической в области крайних значений случайной величины. С этой целью вычисляют статистику:

$$A^2 = -n - 2 \cdot \sum_{i=1}^{n} \left\{ \frac{2 \cdot i - 1}{2 \cdot n} \ln F(x_i) + \left(1 - \frac{2 \cdot i - 1}{2 \cdot n}\right) \ln\left[1 - F(x_i)\right] \right\},$$

и составляют неравенство:

$$\left(A^2 - \frac{0{,}7}{n}\right) \cdot \left(1 + \frac{3{,}6}{n} - \frac{8{,}0}{n^2}\right) \le A_\alpha .$$

Если неравенство выполняется, то нулевая гипотеза принимается, в противном случае нулевая гипотеза отвергается.

```
def anderson_test(sample1):
    print('Критерий Андерсона-Дарлинга')
    result = anderson(sample1, 'norm')
    print('Статистика критерия Андерсона-Дарлинга :', result.statistic)
    print('Критические значения:', result.critical_values)
    print('Уровни значимости:', result.significance_level)
    if result.statistic<=result.critical_values:
        print('Принимаем гипотезу: данные распределены нормально')
    else:
        print('Отвергаем гипотезу: данные не распределены нормально')
```

Функция принимает выборку, а выводит статистику критерия Андерсона-Дарлинга, p-значение, уровни значимости и сообщение о том, нормальная ли выборка

```
PS C:\example\python> python Shapirpy.py
Критерий Андерсона-Дарлинга
Статистика критерия Андерсона-Дарлинга : 0.1346515425389896
Критические значения: [0.501 0.57  0.684 0.798 0.95 ]
Уровни значимости: [15.  10.   5.   2.5  1. ]
Принимаем гипотезу: данные распределены нормально
```

Критерий χ^2

Критерий согласия применяется для проверки гипотезы о соответствии эмпирического распределения произвольному теоретическому распределению, параметры которого оцениваются по выборке. С этой целью рассчитывают статистику:

$$\chi^2 = \sum_{i=1}^{e} \frac{(n_i - n \cdot P_i)^2}{n \cdot P_i}.$$

Если значение статистики меньше критического, то нулевая гипотеза о соответствии опытных данных выбранному гипотетическому распределению принимается, в противном случае нулевая гипотеза отвергается.

```
def hi2(sample1,sample2):
    print('Критерий хи-квадрат')
    sample3=list(sample1)+list(sample2)
    chi2, p,f,b = chi2_contingency(sample3)
    print("Статистика хи-квадрат:", chi2)
    print("p-значение критерия хи-квадрат :", p)
    alpha = 0.05
    if p < alpha:
        print('Принимаем гипотезу: данные распределены нормально')
    else:
        print('Отвергаем гипотезу: данные не распределены нормально')
```

Функция принимает 2 выборки, а выводит статистику критерия согласия, p-значение и сообщение о том, нормальная ли выборка

```
PS C:\example\python> python Shapirpy.py
Критерий хи-квадрат
Статистика хи-квадрат: 0.0
p-значение критерия хи-квадрат : 1.0
Отвергаем гипотезу: данные не распределены нормально
```

Критерий знаков для медианы

Пусть в n пар испытаний получены k положительных разностей, т - отрицательных и l нулевых; n1=n- l. Нулевую гипотезу о равенстве медиан двух совокупностей не отвергают, если число k попадает в область допустимых значений с уровнем значимости α. Границы допустимых значений рассчитывают по формулам :

$$\alpha = 0{,}5^{n_1} \cdot \sum_{i=0}^{k_{cl}} \frac{n_1!}{i! \cdot (n_1 - i)!}, k_{cu} = n_1 - k_{cl}.$$

Для проверки нулевой гипотезы H_0: P=0,5 при альтернативной гипотезе H_A: P<0,5 должно выполняться неравенство $k \ge k_{cl}$. При альтернативной гипотезе H_A:- P>0,5 должно выполняться неравенство $k \le k_{cu} = n_1 - k_{cl}$. При двусторонней альтернативной гипотезе выполняется неравенство H_A: $P \ne 0{,}5$; $k_{cl} \le k \le k_{cu}$ с уровнем значимости 2α.

```
def bini(sample1, sample2):
    count=0
    print('Критерий знаков для медианы')
    for i in range(len(sample1)-1):
          if sample1[i]>sample2[i]:
               count+=1
    b=binomtest(count,len(sample1),alternative='two-sided')

    print('p-value:', b.pvalue)
```

Функция принимает 2 выборки, а выводит p-значение.

```
PS C:\example\python> python Sh
Критерий знаков для медианы
p-value: 0.34375
```

Двухвыборочный критерий Уилкоксона

Критерий знаковых рангов Уилкоксона учитывает расстояние наблюдений относительно нуля посредством рангов.

$$x_1, x_2, \ldots, x_n;$$
$$y_1, y_2, \ldots, y_n;$$
$$z_1 = x_1 - y_1, z_2 = x_2 - y_2, \ldots, z_n = x_n - y_n.$$

Абсолютные значения разностей z располагают в порядке возрастания (ранжируют) и подсчитывают сумму рангов T (порядковых номеров) положительных значений z в этом ряду.

Для приближенного расчета при больших n вычисляют статистики T_1, T_1^* по формулам [22-24]:

$$T_1 = \frac{T - \frac{n_1 \cdot (n_1+1)}{4}}{\sqrt{\frac{n_1 \cdot (n_1+1) \cdot (2 \cdot n_1 + 1)}{24}}}; \quad T_1^* = \frac{T_1}{2} \cdot \left(1 + \sqrt{\frac{n_1 - 1}{n_1 - T_1^2}}\right). \qquad (3.28)$$

Нулевую гипотезу принимают, если $T^*_{\frac{\alpha}{2}} < T_1^* < T^*_{1-\frac{\alpha}{2}}$, где

$$T^*_{1-\frac{\alpha}{2}} = 0{,}5 \cdot \left[t_{1-\frac{\alpha}{2}} + z_{1-\frac{\alpha}{2}}\right] = -T^*_{\frac{\alpha}{2}}; \qquad (3.29)$$

$t_{1-\frac{\alpha}{2}}$ - квантиль уровня $1-\frac{\alpha}{2}$ распределения Стьюдента с числом степеней свободы $f = n_1 - 1$;

$z_{1-\frac{\alpha}{2}}$ - квантиль уровня $1-\frac{\alpha}{2}$ нормированного нормального распределения.

```
def wilcoxon_test(sample1, sample2):
    print('Критерий знаковых рангов Уилкоксона')
    sample1 = np.array(sample1)
    sample2 = np.array(sample2)
    st, p = ranksums(sample1, sample2) #сколько раз одна выборка превышает другую
    print('Статистика критерия Уилкоксона:', st)
    print("p_value: ", p_value)
    alpha = 0.05
     if p < alpha:
        print('Принимаем')
    else:
        print('Отвергаем')
```

Функция принимает 2 выборки, а выводит статистику критерия Уилкоксона , p-значение и сообщение о том, принимаем ли гипотезу

```
Критерий знаковых рангов Уилкоксона
Статистика критерия Уилкоксона: 0.680336051416609
p_value:  0.49629170223109287
Отвергаем
```

Критерий Колмогорова-Смирнова

Критерий предназначен для проверки гипотезы о принадлежности двух независимых выборок одной и той же генеральной совокупности. При произвольном распределении в качестве статистики критерия Колмогорова-Смирнова служит наибольшая разность между накопленными частостями, которые рассчитывают для каждого значения случайной величины X обеих выборочных совокупностей объемом n1 и n2 :

$$k = \max\left|W_1(x) - W_2(x)\right| \ .$$

Рассчитанное значение k сравнивают с критическим $k\alpha$. Если $k \leq k\alpha$, гипотеза о принадлежности двух независимых выборок одной генеральной совокупности подтверждается, в противном случае нулевая гипотеза отвергается.

```
def kolmogorov_smirnov_test(sample1, sample2):
    print('Критерий Колмогорова-Смирнова')
    p_value, statistic = ks_2samp(sample1, sample2)
    print('statistic',statistic)
    print('p_value', p_value)
```

Функция принимает 2 выборки, а выводит статистику критерия Колмогорова-Смирнова, p-значение.

```
Критерий Колмогорова-Смирнова
statistic 0.7869297884777761
p_value 0.3
```

Критерий Краскела-Уоллиса

Критерий Краскела-Уоллиса обобщает задачу о двух выборках на случай k выборок. Нулевая гипотеза утверждает, что k выборок из произвольных совокупностей можно рассматривать как одну (объединенную) выборку из общей совокупности, то есть утверждается равенство параметров сдвига , когда не задано значение общего параметра масштаба $H_0: \theta_1 = \theta_2 = ... = \theta_k$ против альтернативы $H_A: \theta_1, ..., \theta_k$ не все равны. Для проверки нулевой гипотезы строят общий вариационный ряд наблюдений и рассчитывают статистику:

$$H = \frac{12}{N \cdot (N+1)} \cdot \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3 \cdot N \cdot (N+1), \qquad (3.36)$$

где R_i - сумма рангов iой выборки в общем вариационном ряду. Далее рассчитывают величину H_1:

$$H_1 = \frac{H}{2} \cdot \left(1 + \frac{N-k}{N-1-H}\right), \qquad (3.37)$$

которую сравнивают с критическим значение H_α:

$$H_\alpha = 0{,}5 \cdot \left[(k-1) \cdot F_{1-\alpha} + \chi^2_{1-\alpha}\right]; \qquad (3.38)$$

Нулевую гипотезу принимают, если $H1 \le H\alpha$ с уровнем значимости α . В противном случае принимают альтернативную гипотезу.

```
def kraskel_yollis(sample1,sample2):
   print('Критерий Краскела-Уоллиса')
   stat, p_val = stats.kruskal(sample1,sample2)
   print("Статистика Краскела-Уоллиса:", stat)
   print("p-значение:", p_val)
```

Функция принимает 2 выборки, а выводит статистику критерия Колмогорова-Смирнова, p-значение.

```
Критерий Краскела-Уоллиса
Статистика Краскела-Уоллиса: 0.4628571428571391
p-значение: 0.49629170223109464
```