

Домашнее задание по математической статистике

Выполнила Зарянкина Варвара БЭК 211

In [1]:

```
# импорт необходимых библиотек
import pandas as pd
import numpy as np
import scipy.stats as sts
import math
import itertools

import matplotlib.pyplot as plt

# Отключение некоторых лишних предупреждений
import warnings
warnings.filterwarnings("ignore")
```

Задача №1:

Пункт первый

Для начала давайте определимся с вероятностями первого повтора таксиста для нашего туриста. Обозначим за X - с. в. , которая будет показывать номер вызова, на котором это повторение происходит впервые. Нетрудно заметить, что данная случайная величина будет принимать целые значения от 2 и так далее. На самом деле, с увеличением номера вызова у нас уже возникает большее количество таксистов, которых мы можем вновь встретить. Так при втором вызове вероятность встретить того же таксиста одна n -ная, а при третьем уже она видоизменяется и банк таксистов увеличивается (получаем произведение вероятностей встретить первого на встретить второго и нового таксиста). Давайте запишем это формально:

$$P(X = 1) = 0$$

$$P(X = 2) = \frac{1}{n}$$

$$P(X = 3) = \left(1 - \frac{1}{n}\right) \cdot \frac{2}{n}$$

$$P(X = 4) = \left(1 - \frac{1}{n}\right) \cdot \left(1 - \frac{2}{n}\right) \cdot \frac{3}{n}$$

и так далее... Для нашей задачи первое повторение случилось на 10 вызове, т.е.:

$$P(X = 10) = \left(1 - \frac{1}{n}\right) \cdot \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{3}{n}\right) \cdot \dots \cdot \frac{9}{n}$$

Запишем теперь кодом данную задачу и посмотрим на функцию правдоподобия:

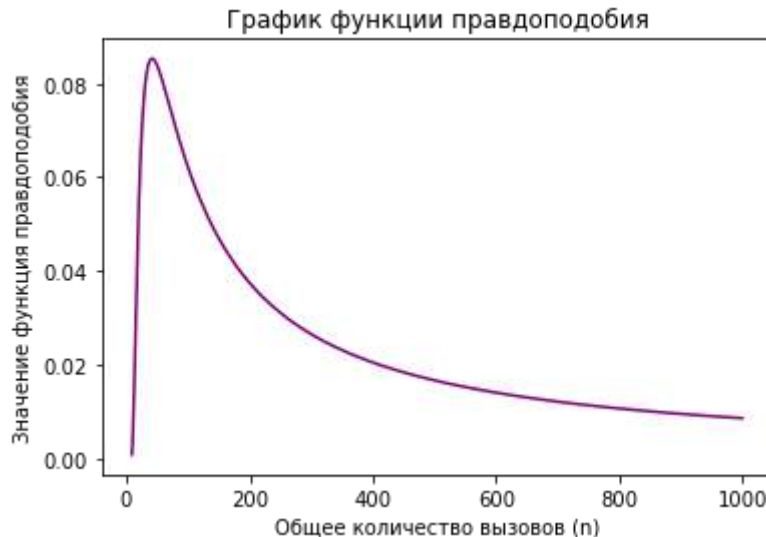
In [2]:

```
def likelihood_days(x):
    l = 1
    for i in range(1,9):
        l *= (x-i)/x
    return l*(9)/x

x_values = np.arange(9, 1000)
```

```
likelihood_values = []
for i in x_values:
    likelihood_values.append(likelihood_days(i))

plt.plot(x_values, likelihood_values, color = 'purple')
plt.xlabel('Общее количество вызовов (n)')
plt.ylabel('Значение функция правдоподобия')
plt.title('График функции правдоподобия')
plt.show()
```



Теперь давайте найдем саму оценку числа n , для нашей задачи это просто максимум нашей функции, который определен, исходя из графика:

```
In [3]: maximum = np.max(likelihood_values)
maximum_i = np.where(likelihood_values == maximum)[0]
max_likelihood_x_values = x_values[maximum_i]
res = max_likelihood_x_values
print(f'Оценка мп для количества вызов равна:{res}')
```

Оценка мп для количества вызов равна:[42]

Пункт второй

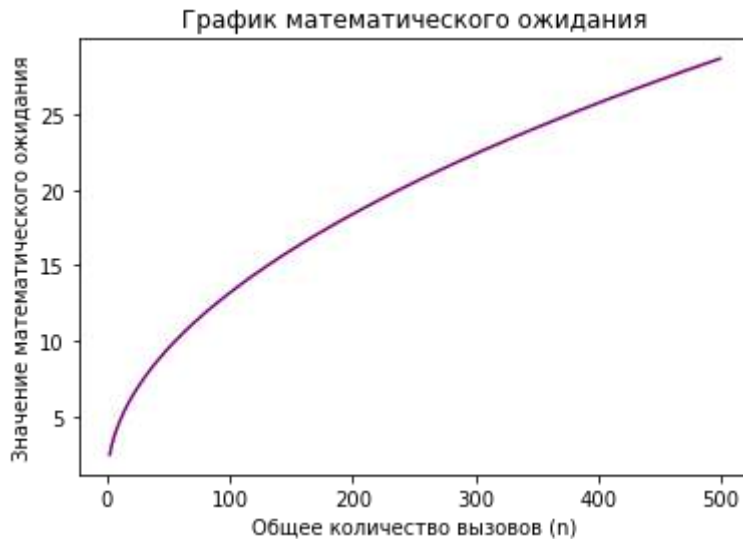
Теперь давайте построим график математического ожидания номера заказа, на котором происходит первое повторение. Но для начала запишем соответствующую функцию:

```
In [4]: def prob(k, n): # запишем функцию для подсчета вероятности
    a = 1
    for i in range(2, k+1):
        a = a*(n-i+2)/n
    return a*(k-1)/n

def E_(n): # запишем функцию для подсчета мо
    res = 0
    for j in range(2, n+2):
        res += j*prob(j, n)
    return(res)
```

```
In [5]: gen = np.arange(2, 500)
E = np.array([E_(n) for n in gen])
plt.plot(gen, E, color = 'purple')
plt.xlabel('Общее количество вызовов (n)')
```

```
plt.ylabel('Значение математического ожидания')
plt.title('График математического ожидания');
```



Пункт третий

Теперь давайте предположим, что истинное n равняется 100 и проведем 10000 симуляций вызовов такси до первого повторного, рассчитаем 10000 оценок методом моментов и 10000 оценок методом максимального правдоподобия.

```
In [6]: np.random.seed(8)
n = 100
t = np.arange(1, 101)
res = []
for i in range(1, 10001):
    odin = np.random.choice(t)
    ar = []
    while np.isin(odin, ar) == False:
        ar.append(odin)
        odin = np.random.choice(t)

    k = len(ar)+1
    res.append(k)
```

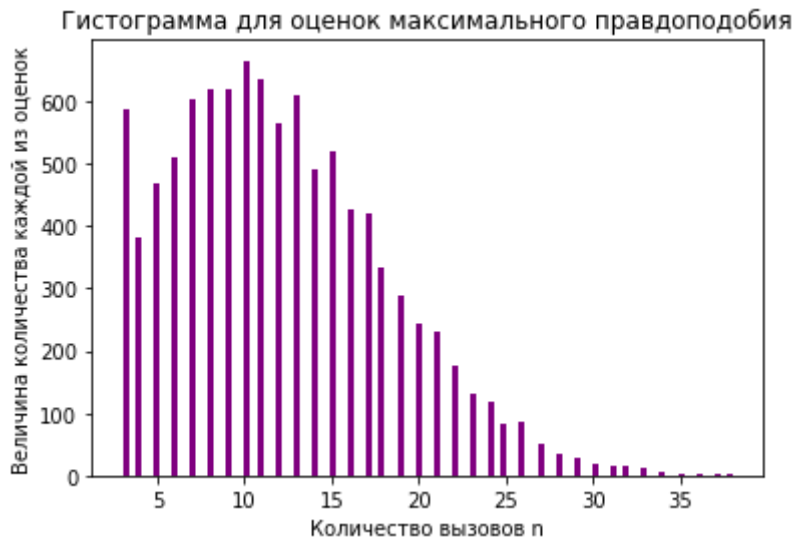
Метод максимального правдоподобия

```
In [7]: def ML(k): # запишем функцию для метода максимального правдоподобия, обеспечивающая сдвиг
    n = k-1
    ml_1 = prob(n, k)
    ml_2 = prob(n, k)
    while ml_1 <= ml_2:
        ml_1 = prob(n, k)
        n += 1
        ml_2 = prob(n, k)
    return n-1
```

```
In [8]: func = np.vectorize(ML) # применим функцию к сгенерированным значениям
otzen = func(res)
```

```
In [9]: plt.hist(otzen, bins = 100,color='purple')
plt.title('Гистограмма для оценок максимального правдоподобия')
```

```
plt.xlabel('Количество вызовов n')
plt.ylabel('Величина количества каждой из оценок');
```



```
In [10]: ml_bias = abs(100 - np.mean(otzen))
ml_var = np.var(otzen)
ml_mse = np.std(otzen)
```

```
In [11]: print('смещение:', ml_bias)
print('дисперсия:', ml_var)
print('среднеквадратичная ошибка:', ml_mse)
```

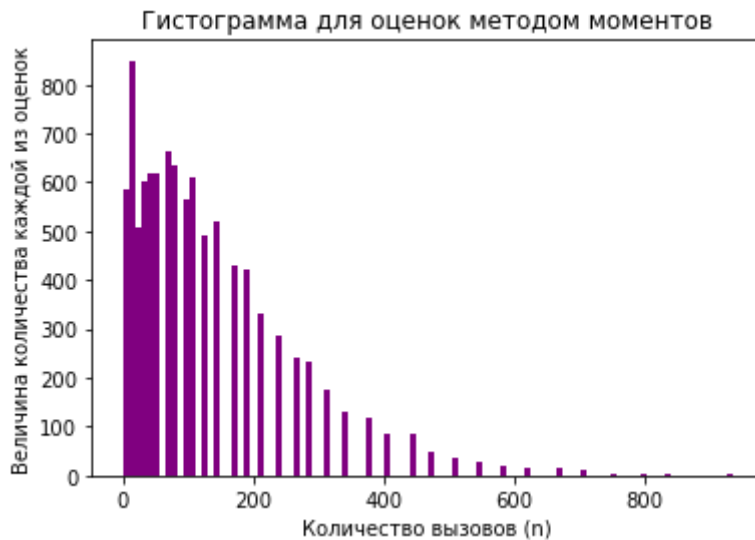
```
смещение: 87.8746
дисперсия: 37.19527484
среднеквадратичная ошибка: 6.098792900238538
```

Метод моментов

Теперь проделаем ту же процедуру, но для метода моментов, где мы будем использовать первый начальный момент - мат ожидание

```
In [12]: res = np.array(res) # для этого преобразуем функцию мат ожидания на наши с.в.
n = np.arange(1, 1001)
func = np.vectorize(E_)
mo = func(n)
mo_new = mo[:, np.newaxis]
otz = res[np.newaxis, :]
otv = np.absolute(mo_new - otz)
mm = np.argmin(otv, axis = 0)
```

```
In [13]: plt.hist(mm, bins = 100, color='purple')
plt.title('Гистограмма для оценок методом моментов')
plt.xlabel('Количество вызовов (n)')
plt.ylabel('Величина количества каждой из оценок');
```



```
In [14]: mm_bias = abs(100 - np.mean(mm))
mm_var = np.var(mm)
mm_mse = np.std(mm)
```

```
In [15]: print('смещение:', mm_bias)
print('дисперсия:', mm_var)
print('среднеквадратичная ошибка:', mm_mse)
```

```
смещение: 21.274199999999993
дисперсия: 13464.25901436
среднеквадратичная ошибка: 116.03559373899027
```

Таким образом, если сравнивать две оценки между собой, то оценка методом максимального правдоподобия оказывается наиболее точной, ведь она в среднем меньше отклоняется от истинного значения, но имеет большее смещение.

Задача № 2

Пункт первый

Теперь наш турист сфокусируется на именах таксистов, которые по его замечаниям из 10 заказов насчитывали 6 значений.

```
In [16]: def function_names(a, b, x=10): # запишем нашу функцию правдоподобия кодом
ver = [(b - i) / b for i in range(1, a)]
p = math.prod(ver)

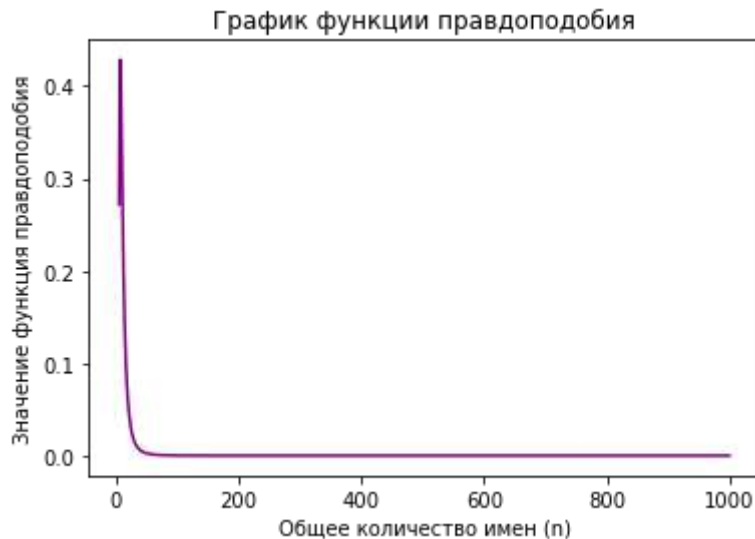
comb = itertools.combinations_with_replacement(range(1, a + 1), x - a)
s = sum(math.prod(j) for j in comb)

p *= s / (b ** (x - a))
return p
```

```
In [17]: n = np.arange(6, 10**3)
arr = []
for i in n:
    arr.append(function_names(6, i))

plt.plot(n, arr, color = 'purple')
plt.title('График функции правдоподобия')
```

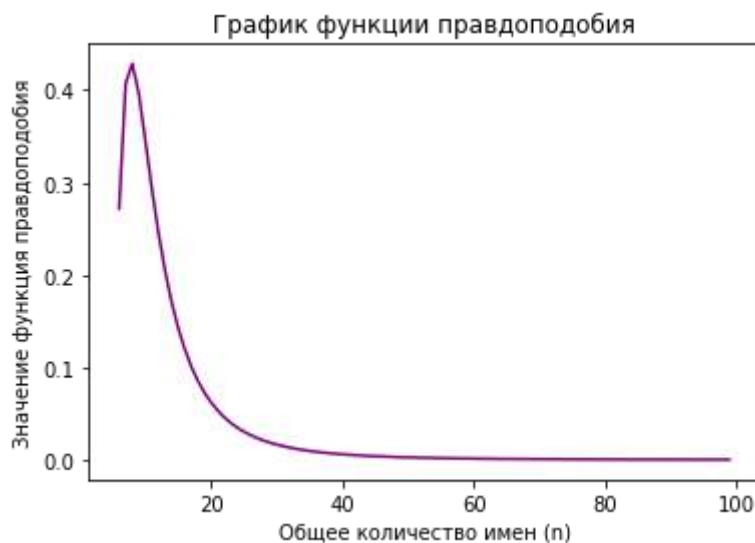
```
plt.xlabel('Общее количество имен (n)')
plt.ylabel('Значение функция правдоподобия');
```



In [18]:

```
n = np.arange(6,100) # приблизим график
arr = []
for i in n:
    arr.append(function_names(6, i))

plt.plot(n, arr, color = 'purple')
plt.title('График функции правдоподобия')
plt.xlabel('Общее количество имен (n)')
plt.ylabel('Значение функция правдоподобия');
```



Теперь давайте найдем саму оценку числа n , для нашей задачи это просто максимум нашей функции, который определен, исходя из графика:

In [19]:

```
maximum = np.max(arr)
maximum_i = np.where(arr == maximum)[0]
max_likelihood_x_values = n[maximum_i]
res = max_likelihood_x_values
print(f'Оценка мп для количества имен равна:{res}')
```

Оценка мп для количества имен равна:[8]

Пункт второй

Во втором пункте построим график математического ожидания числа разных имен у 10 таксистов. А так же найти оценку методом моментов.

Для начала определим функцию, которая будет считать необходимое нам математическое ожидание относительно заданного числа имен и количества вызовов (в нашей задаче = 10)

In [20]:

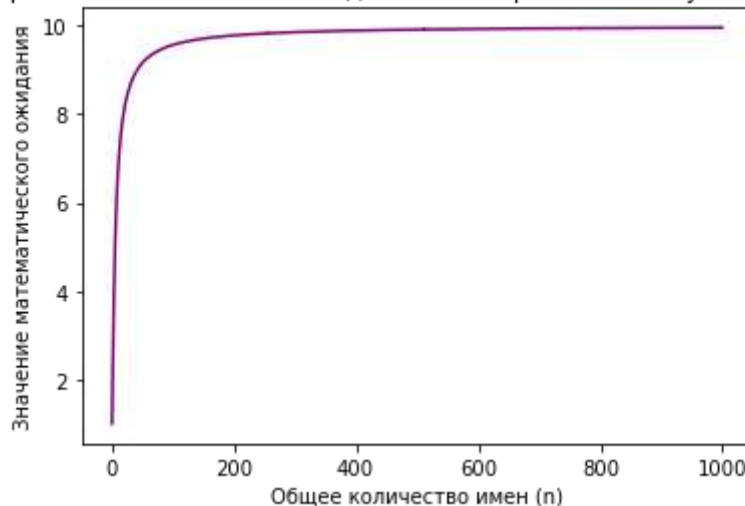
```
def E(b, m):
    mo = sum(a * function_names(a, b, m) for a in range(1, 11))
    return mo
```

In [21]:

```
n = np.arange(1, 10**3)
array = []
for i in n:
    array.append(E(i, 10))

plt.plot(n, array, color = 'purple')
plt.title('График математического ожидания числа разных имен у 10 таксистов')
plt.xlabel('Общее количество имен (n)')
plt.ylabel('Значение математического ожидания');
```

График математического ожидания числа разных имен у 10 таксистов



Таким образом, значения математического ожидания ожидаемо увеличивается от количества имен таксистов. Теперь найдем оценку данного параметра с помощью метода моментов.

In [22]:

```
array = np.array(array)
minimum = min(abs(array - 6))
res = n[abs(array - 6) == minimum][0]
print(f'Оценка мм для количества имен равна: {res}')
```

Оценка мм для количества имен равна: 8

Пункт третий

Теперь мы знаем, что истинное количество имен равно 20. Проведем 10^4 симуляций, рассчитаем оценки методом мп и мм, а также потом посчитаем их основные характеристики:

In [23]:

```
# аналогично предыдущей задаче генерируем наши эксперименты
np.random.seed(8)
```

```

n = 20
t = np.arange(1, 21)
res = []
for i in range(1, 10001):
    odin = np.random.choice(t)
    ar = []
    M = 10
    for i in range(M):
        ar.append(odin)
        odin = np.random.choice(t)
    a = len(set(ar))
    res.append(a)

```

Метод максимального правдоподобия

```

In [24]: def ml_func(k):
        n = 1
        ml_1 = function_names(k, n, 10)
        ml_2 = function_names(k, n, 10)
        while (ml_1 <= ml_2) and (n < 100): # берем ограничение из условия задачи
            ml_1 = function_names(k, n, 10)
            n += 1
            ml_2 = function_names(k, n, 10)
        return n-1

```

```

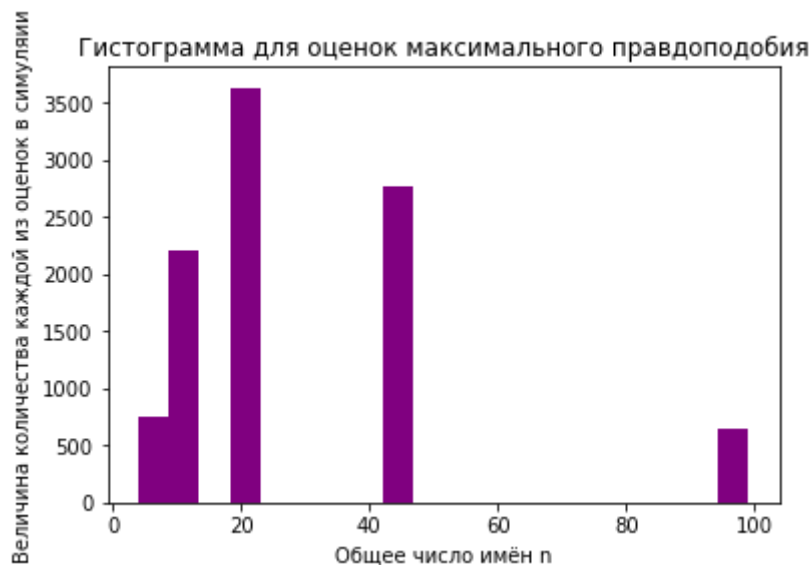
In [25]: func = np.vectorize(ml_func)
        otzen = func(res)

```

```

In [26]: plt.hist(otzen, bins = 20, color='purple')
        plt.title('Гистограмма для оценок максимального правдоподобия')
        plt.xlabel('Общее число имён n')
        plt.ylabel('Величина количества каждой из оценок в симуляции')
        plt.show()

```



```

In [27]: ml_bias = abs(100 - np.mean(otzen))
        ml_var = np.var(otzen)
        ml_mse = np.std(otzen)

```


In [28]:

```
print('смещение:', ml_bias)
print('дисперсия:', ml_var)
print('среднеквадратичная ошибка:', ml_mse)
```

```
смещение: 71.928300000000001
дисперсия: 493.45775911000015
среднеквадратичная ошибка: 22.213909136169622
```

Метод моментов

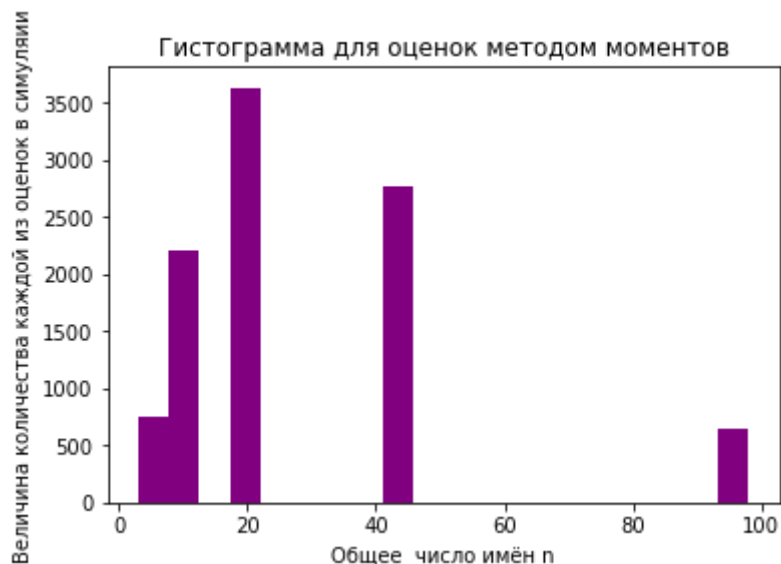
Для каждой из сгенерированных выше случайных величин найдем оценки методом моментов, такие, чтобы значение функции от количества имен имело минимальное расстояние до самого значения случайной величины.

In [29]:

```
res = np.array(res) # для этого преобразуем функцию мат ожидания на наши с.в.
n = np.arange(1, 100)
func = np.vectorize(E)
mo = func(n, 10)
mo_new = mo[:, np.newaxis]
otz = res[np.newaxis, :]
otv = np.absolute(mo_new - otz)
mm = np.argmin(otv, axis = 0)
```

In [30]:

```
plt.hist(mm, bins = 20, color='purple')
plt.title('Гистограмма для оценок методом моментов',)
plt.xlabel('Общее число имён n')
plt.ylabel('Величина количества каждой из оценок в симуляции')
plt.show()
```



In [31]:

```
mm_bias = abs(100 - np.mean(mm))
mm_var = np.var(mm)
mm_mse = np.std(mm)
```

In [32]:

```
print('смещение:', mm_bias)
print('дисперсия:', mm_var)
print('среднеквадратичная ошибка:', mm_mse)
```

```
смещение: 72.5585
дисперсия: 486.79657775
```

среднеквадратичная ошибка: 22.063467038296587

Итак, если проводить сравнение между полученными оценками, то метод моментов показывает лучшее значение среднеквадратичной ошибки, показывая, что в отличие от метода максимального правдоподобия оценки, полученные при симуляциях, реже отклоняются от истинного значения.

Задача №3:

Пункт первый

Сгенерируем выборку из экспоненциального распределения

```
In [33]: n = 20
```

```
In [34]: np.random.seed(7)
vyb = np.random.exponential(1, size=(10**4, 20))
```

Построим классический доверительный интервал:

```
In [35]: m = np.mean(vyb, axis=1) # считаем среднее
std = np.std(vyb, ddof = 1, axis = 1) / np.sqrt(n) # считаем стандартную ошибку
```

```
In [36]: l = m - 1.96 * std
r = m + 1.96 * std
print(l, r)
```

```
[0.52921633 0.60521522 0.54198197 ... 0.41874255 0.47760867 0.38741254] [1.412
85337 1.26351705 1.01447306 ... 1.5596751 1.12149683 1.25514637]
```

```
In [37]: res = np.logical_and(l <= 1, r >= 1)
print(f'Вероятность накрытия для классического ДИ: {np.mean(res)}')
```

Вероятность накрытия для классического ДИ: 0.908

Теперь сделаем то же самое для наивного бутстрэпа:

```
In [38]: np.random.seed(8)
arr = [] # создаем массив для бут условия
for i in vyb:
    boot = np.random.choice(np.arange(n), size=(10**4, n), replace = True)
    boot_means = np.mean(i[boot], axis=1)

    q_l = np.quantile(boot_means, 0.025)
    q_r = np.quantile(boot_means, 0.975)

    res = np.logical_and(q_l <= 1, q_r >= 1) # прописываем логическое условие
    arr.append(res)
print(f'Вероятность накрытия для наивного бутстрэпа: {np.mean(arr)}')
```

Вероятность накрытия для наивного бутстрэпа: 0.9066

Далее делаем для бутстрэпа t-статистики

```
In [39]: np.random.seed(8)
arr = [] # создаем массив для бут условия
for i in vyb:
```

```

boot = np.random.choice(np.arange(n), size=(10**4, n), replace = True)

boot_means = np.mean(i[boot], axis=1) # считаем средние и std
se = (np.std(i[boot], axis=1, ddof = 1))/np.sqrt(n)
mean_new = np.mean(i)

q_l = np.quantile((boot_means - mean_new)/se, 0.025)
q_r = np.quantile((boot_means - mean_new)/se, 0.975)

res = np.logical_and(mean_new - q_r*(np.std(i, ddof = 1))/np.sqrt(n) <= 1
arr.append(res) # прописываем логическое условие
print(f'Вероятность накрытия для бутстрапа t-статистики: {np.mean(arr)}') # расчитать

```

Вероятность накрытия для бутстрапа t-статистики: 0.9475

Пункт второй:

Теперь сделаем вновь данную процедуру, но для выборки из распределения студента

```

In [40]: np.random.seed(7)
vyb_2 = np.random.standard_t(3, size=(10**4,20))

```

Для классического Д.И.:

```

In [41]: m = np.mean(vyb_2, axis=1) # считаем среднее
std = np.std(vyb_2, ddof = 1, axis = 1)/ np.sqrt(n) # считаем стандартную ошибку

```

```

In [42]: l = m - 1.96 * std
r = m + 1.96 * std
print(l,r)

[-0.18696865 -1.25168332 -1.03269744 ... -1.09419249 -0.61222192
-2.26289657] [1.14235291 1.12117275 0.34358406 ... 0.73682321 0.38748509 0.69
824797]

```

```

In [43]: res = np.logical_and(l <= 0, r >= 0)
print(f'Вероятность накрытия для классического ДИ: {np.mean(res)}')

```

Вероятность накрытия для классического ДИ: 0.9473

Для наивного бутстрапа:

```

In [44]: np.random.seed(8)
arr = []
for i in vyb_2:
    boot = np.random.choice(np.arange(n), size=(10**4, n), replace = True)
    boot_means = np.mean(i[boot], axis=1)

    q_l = np.quantile(boot_means, 0.025)
    q_r = np.quantile(boot_means, 0.975)

    res = np.logical_and(q_l <= 0, q_r >= 0) # прописываем логическое условие
    arr.append(res)
print(f'Вероятность накрытия для наивного бутстрапа: {np.mean(arr)}')

```

Вероятность накрытия для наивного бутстрапа: 0.9243

Для бутстрап t - статистики:

In [45]:

```

np.random.seed(8)
arr = []
for i in vyb_2:

    boot = np.random.choice(np.arange(n), size=(10**4, n), replace = True)

    boot_means = np.mean(i[boot], axis=1)
    se = (np.std(i[boot], axis=1, ddof = 1))/np.sqrt(n)
    mean_new = np.mean(i)

    q_l = np.quantile((boot_means - mean_new)/se, 0.025)
    q_r = np.quantile((boot_means - mean_new)/se, 0.975)

    res = np.logical_and(mean_new - q_r*(np.std(i, ddof = 1))/np.sqrt(n) <= 0
    arr.append(res) # прописываем логическое условие
print(f'Вероятность накрытия для бутстрэпа t-статистики: {np.mean(arr)}')
```

Вероятность накрытия для бутстрэпа t-статистики: 0.9318

Пункт третий

Вывод: Чем выше вероятность накрытия, тем лучше метод. Идеальный доверительный интервал будет иметь вероятность накрытия, близкую к заданному уровню доверия (в нашем случае 95%). В первом эксперименте лучшим методом оказался бутстрэп t - статистики, показавший вероятность накрытия близкую к 0.95, а во втором случае классический доверительный интервал (с вероятностью накрытия в 0.9473). Если смотреть по распределениям, то в целом распределение Стьюдента с 3-мя степенями свободы обеспечивает нам более высокие вероятности накрытия во всех трех методах.

Задача №4:

Перед началом выполнения задания загрузим необходимые данные. В 4-6 задачах нам нужны будут результаты экзамена по теории вероятности за 2022-2023 года. Переделаем все в табличку, содержащую фамилии студентов и их баллы за экзамен.

In [46]:

```
data = pd.read_excel('22-23_hse_probability.xlsx', sheet_name = 'Лист4')
```

In [47]:

```
print(data.shape) # всего у нас 332 наблюдения
data.head()
```

(332, 2)

Out[47]:

	Фамилия	Результат
0	Репенкова	16
1	Ролдугина	0
2	Сафина	19
3	Сидоров	26
4	Солоухин	21

Будем проверять гипотезу о том, что ожидаемые результаты экзамена по теории вероятностей тех, у кого фамилия начинается с гласной буквы и с согласной буквы, равны.

Уровень значимости: 5%

$$H_0 : \mu_{glas} = \mu_{soglas}$$

$$H_1 : \mu_{glas} \neq \mu_{soglas}$$

Для начала разобьем наших студентов на две большие группы: Фамилии на согласные и фамилии на гласные

```
In [48]: data['Гласная'] = 0
vowels = ["А", "У", "О", "Е", "И", "Ю", "Ё", "Я", "Э"]
data['Гласная'][data['Фамилия'].str[0].str.upper().isin(vowels)] = 1
```

```
In [49]: data # посмотрим на нашу таблицу
```

```
Out[49]:
```

	Фамилия	Результат	Гласная
0	Репенкова	16	0
1	Ролдугина	0	0
2	Сафина	19	0
3	Сидоров	26	0
4	Солоухин	21	0
...
327	Сенников	19	0
328	Ся	0	0
329	Сятова	0	0
330	Темиркулов	0	0
331	Эшмеев	16	1

332 rows × 3 columns

Пункт первый

Воспользуемся тестом Уэлча, который используется, когда дисперсии оказываются неравными.

```
In [50]: vowel_scores = data['Результат'][data['Гласная'] == 1]
consonant_scores = data['Результат'][data['Гласная'] != 1]

t_statistic, p_value = sts.ttest_ind(vowel_scores, consonant_scores, equal_va

print("t-статистика:", t_statistic)
print("p-value:", p_value)
```

t-статистика: -0.8519661870595602

p-value: 0.3974027153843839

Таким образом, p-value оказывается больше нашего уровня значимости, что свидетельствует о том, что нет оснований отвергать нулевую гипотезу о равенстве ожидаемых результатов экзамена.

Пункт второй

Теперь воспользуемся наивным бутстрэпом для проверки нашей гипотезы

```
In [51]: mean_r = np.mean(consonant_scores) - np.mean(vowel_scores) # рассчитываем наблюд
ny = vowel_scores.shape[0]
nx = consonant_scores.shape[0]
se_r = np.sqrt((consonant_scores.std()2)/nx + (vowel_scores.std()2/ny) #
```

```
In [52]: print(nx, ny) # посмотрим на размеры наших групп
```

283 49

```
In [53]: np.random.seed(7)

i = np.random.choice(np.arange(nx), size=(104, nx)) # сгенерируем для согласных
j = np.random.choice(np.arange(ny), size=(104, ny)) # сгенерируем для гласных

diff = np.mean(np.array(consonant_scores)[i], axis=1) - np.mean(np.array(vowe
p_value = 2*np.min([np.mean((mean_r < diff)), np.mean(mean_r >= diff)])
```

```
In [54]: print(f'p-value: {p_value}')
```

p-value: 0.9762

Таким образом, p-value оказывается много больше нашего уровня значимости, что свидетельствует о том, что нет оснований отвергать нулевую гипотезу о равенстве ожидаемых результатах экзамена.

Пункт третий

В третьем пункте проверим гипотезу с помощью t-бутстрэпа

```
In [55]: mean_r = np.mean(consonant_scores) - np.mean(vowel_scores) # рассчитываем наблюд
```

```
In [56]: np.random.seed(8)

i = np.random.choice(np.arange(nx), size=(104, nx)) # сгенерируем для согласных
j = np.random.choice(np.arange(ny), size=(104, ny)) # сгенерируем для гласных

b_means = np.mean(np.array(consonant_scores)[i], axis=1) - np.mean(np.array(v
# найдем стандартную ошибку
se_b = np.sqrt(((np.std(np.array(consonant_scores)[i], axis=1, ddof=1))2 / (
diff = (b_means - mean_r) / se_b

p_value = 2*np.min([np.mean((t_statistic < diff)), np.mean(t_statistic >= dif
```

```
In [57]: print(f'p-value: {p_value}')
```

p-value: 0.4362

Таким образом, вновь p-value оказывается больше нашего уровня значимости, что свидетельствует о том, что нет оснований отвергать нулевую гипотезу о равенстве ожидаемых результатах экзамена.

Пункт четвертый

А теперь воспользуемся перестановочным тестом, идея которого заключается в намеренном перемешивании одного из признаков

```
In [58]: mean_r = np.mean(vowel_scores) - np.mean(consonant_scores) # рассчитываем наблю
```

```
In [59]: arr_1 = np.ones_like(consonant_scores) # создаем массивы для будущих перестановок
arr_2 = np.zeros_like(vowel_scores)
null_ones = np.hstack((arr_1, arr_2))
combined_group = np.hstack((consonant_scores, vowel_scores ))
```

```
In [60]: np.random.seed(8)
num_permutations = 10000
perm_diff_means = np.zeros(num_permutations) # создаем массив для будущих различий

for i in range(num_permutations):
    np.random.shuffle(null_ones)
    perm_group_vowel = combined_group[null_ones == 0]
    perm_group_consonant = combined_group[null_ones == 1]
    perm_diff_mean = np.mean(perm_group_consonant) - np.mean(perm_group_vowel)

    perm_diff_means[i] = perm_diff_mean

p_value = 2*(np.min([np.mean(( mean_r < perm_diff_means )), np.mean( mean_r >
```

```
In [61]: print(f'p-value: {p_value}')
```

p-value: 0.3776

Мы вновь получили высокое значение p-value, что сигнализирует о том, что нет оснований отвергать нулевую гипотезу.

Таким образом, во всех случаях мы не отвергаем H_0

Задача №5:

Поделим студентов на 4 группы:

```
In [62]: med = data['Результат'].median()
```

```
In [63]: data['Больше медианы'] = 0
```

```
In [64]: data['Больше медианы'][data['Результат'] > med] = 1
```

```
In [65]: data.head()
```

```
Out[65]:
```

	Фамилия	Результат	Гласная	Больше медианы
0	Репенкова	16	0	0
1	Ролдугина	0	0	0

	Фамилия	Результат	Гласная	Больше медианы
2	Сафина	19	0	1
3	Сидоров	26	0	1
4	Солоухин	21	0	1

```
In [66]: cross_tab = pd.crosstab(data['Гласная'], data['Больше медианы'])
cross_tab # таблица сопряженности
```

```
Out[66]: Больше медианы    0    1
         Гласная
         0  138  145
         1   28   21
```

Пункт первый

В первом пункте нам необходимо построить 95-% асимптотический доверительный интервал для отношения шансов:

```
In [67]: odd_a = 21/28
odd_b = 145/138
```

```
In [68]: diff = np.log(odd_a) - np.log(odd_b) # как мы делали на семинарах запишем через раз
se = np.sqrt((1/138) + (1/145) + (1/21) + (1/28))

s_obs = (diff - 0)/(se) # считаем наблюдаемую статистику
s_obs
```

```
Out[68]: -1.0799144576000155
```

Запишем теперь границы интервала:

```
In [69]: l = diff - 1.96*se
r = diff + 1.96*se
print(f' границы интервала: {np.exp([l,r])}')
```

```
границы интервала: [0.38709024  1.31623208]
```

```
In [70]: p_value = 2*np.min([sts.norm(loc = 0, scale = 1).cdf(s_obs), 1-sts.norm(loc = 0, scale = 1).cdf(s_obs)])
```

```
In [71]: print(f'p-value: {np.round(p_value, 4)}')
```

```
p-value: 0.2802
```

Итак, p-value оказался больше нашего уровня доверия, и как следствие, у нас нет оснований отвергать нулевую гипотезу о равенстве отношения шансов единице.

Пункт второй

Теперь построим доверительный интервал для вероятностей хорошо написать экзамен.


```
In [72]: p_1 = 21/49 # вероятности хорошо написать
         p_2 = 145/283
```

```
In [73]: diff = np.log(p_1) - np.log(p_2)
         se = np.sqrt(1/145 - 1/283 + 1/21 - 1/49)

         s_obs = (diff - 0)/(se)
         s_obs # считаем наблюдаемую статистику
```

```
Out[73]: -1.021337019974948
```

```
In [74]: l = diff - 1.96*se
         r = diff + 1.96*se
         print(f' границы интервала: {np.exp([l,r])}')
```

```
границы интервала: [0.59374922 1.17836612]
```

```
In [75]: p_value = 2*np.min([sts.norm(loc = 0, scale = 1).cdf(s_obs), 1-sts.norm(loc = 0, scale = 1).cdf(s_obs)])
```

```
In [76]: print(f'p-value: {np.round(p_value, 4)}')
```

```
p-value: 0.3071
```

Таким образом, наше наблюдаемое значение не входит в критическую область, и у нас нет оснований отвергать нулевую гипотезу о равенстве отношения вероятностей единице.

Пункт третий

В последнем пункте задачи нас вновь просят построить 95% интервал для отношения шансов, но теперь с использованием наивного бутстрэпа, а также проверить гипотезу о том, что отношение шансов равно 1.

```
In [77]: n_1 = consonant_scores.shape[0]
         n_2 = vowel_scores.shape[0]
         p_1 = 145/(145+138)
         p_2 = 21/(21+28)
         s_obs = (p_1 / (1 - p_1)) / (p_2 / (1 - p_2)) # найдем наблюдаемое значение
```

```
In [78]: np.random.seed(8) # реализуем наивный бутстрэп
         arr = []
         for _ in range(10**4):

             i = np.random.choice(consonant_scores, size = n_1)
             j = np.random.choice(vowel_scores, size = n_2)

             con_med = np.sum(i > med)
             con_lmed = n_1 - con_med
             vow_med = np.sum(j > med)
             vow_lmed = n_2 - vow_med

             p_1_b = con_med / n_1
             p_2_b = vow_med / n_2

             s_b = (p_1_b / (1 - p_1_b)) / (p_2_b / (1 - p_2_b))
```

```
arr.append(s_b)
```

```
arr = np.array(arr)
p_value = 2*(np.min([np.mean((s_obs < arr)), np.mean( s_obs >= arr)]))
```

In [79]:

```
print(f'p-value: {p_value}')
```

p-value: 0.9954

Таким образом, наше значение p-value оказывается много больше установленного уровня значимости и, как следствие, у нас нет оснований отвергать нулевую гипотезу.

In [80]:

```
# найдем наши границы интервала
q_l = np.quantile(arr, 0.025)
q_r = np.quantile(arr, 0.975)
```

In [81]:

```
print(f' границы интервала: {q_l,q_r}')
```

границы интервала: (0.7703578904591296, 2.7022993141054865)

Задача №6:

Для начала создадим новую переменную, которая будет показывать нам длину фамилии студента.

In [82]:

```
data['Длина Фамилии'] = data['Фамилия'].str.len()
```

In [83]:

```
data.head() # добавляем длину фамилии
```

Out[83]:

	Фамилия	Результат	Гласная	Больше медианы	Длина Фамилии
0	Репенкова	16	0	0	9
1	Ролдугина	0	0	0	9
2	Сафина	19	0	1	6
3	Сидоров	26	0	1	7
4	Солоухин	21	0	1	8

Пункт первый

Для того, чтобы найти оценку коэффициента бэта методом моментов воспользуемся первым начальным моментом, приравняв теоретический и выборочные моменты. Так как $E(Y) = \beta \cdot F$, то взяв мат ожидание от обеих частей мы получим $E(Y) = \beta \cdot E(F)$. Таким образом, наш поиск оценки бэта с помощью метода моментов сходится к $\beta = \frac{E(Y)}{E(F)}$, или же $\beta = \frac{Y_{mean}}{F_{mean}}$ (если мы числитель и знаменатель так же оценим с помощью первого начального момента).

In [84]:

```
beta = np.mean(data['Результат']) / np.mean( data['Длина Фамилии'])
print("Оценка параметра beta:", beta)

# выборочная корреляция
correlation = np.corrcoef(data['Длина Фамилии'],data['Результат'])[0, 1]
print("Выборочная корреляция:", correlation)
```

Оценка параметра β : 2.0613026819923372
Выборочная корреляция: 0.025328052669147665

Пункт второй

Теперь проведем перестановочный тест и протестируем гипотезу о равенстве корреляции нулю

$$H_0 : corr = 0$$

$$H_1 : corr \neq 0$$

In [85]:

```
print("Наблюдаемая корреляция:", correlation)
alpha = 0.05
np.random.seed(8)
arr = [] # создаем массив для будущих корреляций

for _ in range(1000):
    perm_results = np.random.permutation(data['Результат'])
    perm_corr = np.corrcoef(perm_results, data['Длина Фамилии'])[0, 1]
    arr.append(perm_corr)

p_value = 2 * np.min([np.mean(arr > correlation), np.mean(arr <= correlation)])
print("p-value:", p_value)
```

Наблюдаемая корреляция: 0.025328052669147665
p-value: 0.634

Итак, мы получили наше p-value, которое оказалось больше нашего уровня значимости, а следовательно, у нас нет оснований отвергать нулевую гипотезу.

Задача №7:

Для решения седьмой задачи я взяла задачу из прошлогодних вариантов про больших русских котов (кр-4, 2020-2021 год №7). В конце представлен диалог с чатом. Спойлер: обошлось не без помощи.

Задача №8:

Если говорить о сторонних источниках, которые помогли немного лучше углубиться в материал курса по матстату, то я бы отметила семинары Филиппа Ульянкина по дисциплине "Анализ данных на python", который у нас шел в этом году. Несмотря на то, что курс был отчасти построен на лекциях Елены Владимировны, он помог все расставить по полочкам, показывая практическое применение статистическим методам. Так с помощью визуализаций я смогла лучше понять тему доверительных интервалов, а домашка по гипотезам помогла разобраться в предпосылках применения различных статистических тестов (z-test & t-test) и рассмотрению их выполнимости на практике.

3

Рост (в дециметрах) и вес (в килограммах) случайно взятого большого русского кота хорошо описываются независимыми хи-квадрат случайными величинами с 10-ю и k степенями свободы соответственно. Нулевая гипотеза $H_0: k = 1$ отвергается, если отношение суммарного роста к суммарному весу у двух случайно взятых котов превышает 200. Найдите уровень значимости данного теста.



Для нахождения уровня значимости данного теста нам необходимо определить критическую область и вычислить вероятность попадания в неё.



Поскольку рост и вес кота описываются независимыми хи-квадрат распределениями, суммарный рост и суммарный вес также будут иметь хи-квадрат распределения. Для нашего случая общее количество степеней свободы будет равно сумме степеней свободы для роста и веса, то есть $10 + k$.

Пусть X и Y - суммарный рост и суммарный вес двух случайно взятых котов. Тогда статистика теста будет выглядеть следующим образом:

$$Z = X/Y$$

Мы отвергаем нулевую гипотезу, если $Z > 200$.

Для нахождения уровня значимости нам нужно найти вероятность $P(Z > 200)$ при условии, что нулевая гипотеза верна ($k = 1$). Для этого нам нужно построить хи-квадрат распределение с 11 степенями свободы.

↻ Regenerate response

и вычислить вероятность

Send a message.



квадрат распределение с 11 степенями свободы ($10 + 1$) и вычислить вероятность попадания в правый хвост данного распределения.

Окончательный ответ на вопрос будет зависеть от конкретного значения этой вероятности.

3

ты должен складывать не количество степеней свободы у веса и у роста одновременно, а учитывать, что есть два кота независимых, т.е. умножать на 2 каждую из степеней свободы



Извините за путаницу. Действительно, при анализе двух независимых случайно взятых котов, каждая из степеней свободы должна быть умножена на 2. Таким образом, общее количество степеней свободы будет равно $10 * 2 + k * 2 = 20 + 2k$.



Теперь мы можем найти уровень значимости теста, вычислив вероятность $P(Z > 200)$ при условии, что нулевая гипотеза верна ($k = 1$). Для этого мы строим хи-квадрат распределение с 22 степенями свободы ($20 + 2 * 1$) и находим вероятность попадания в правый хвост этого распределения.

Затем, найденная вероятность будет являться уровнем значимости данного теста.

3

необходимо все-так использовать случайных величин с хи=квадрат

↻ Regenerate response

Send a message.



3

необходимо все-так использовать отношение двух случайных величин с хи=квадрат распределением



Понял, извините за путаницу в предыдущих ответах.



Для нахождения уровня значимости данного теста с использованием отношения двух случайных величин с хи-квадрат распределением, нужно сначала построить распределение отношения.

Пусть X и Y - суммарный рост и суммарный вес двух случайно взятых котов, соответственно. Мы хотим проверить нулевую гипотезу $H_0: \kappa = 1$ против альтернативной гипотезы $H_1: \kappa > 1$.

Отношение $Z = X/Y$ имеет F-распределение (F-распределение Снедекора) с $(10, \kappa)$ степенями свободы.

Уровень значимости теста можно определить как вероятность $P(Z > 200)$ при условии H_0 верна ($\kappa = 1$). Для этого мы строим F-распределение с $(10, 1)$ степенями свободы и вычисляем вероятность попадания в правый хвост этого распределения.

Таким образом, уровень значимости данного теста будет равен этой вероятности.

↻ Regenerate response

3

супер, но учти, что котов два и умножь каждую из степеней свободы на два

Send a message.



3

супер, но учти, что котов два и умножь каждую из степеней свободы на два



Извините за путаницу. Действительно, поскольку у нас два случайно выбранных кота, каждая из степеней свободы должна быть умножена на 2.



Таким образом, общее количество степеней свободы будет равно $10 * 2 + k * 2 = 20 + 2k$.

Мы можем найти уровень значимости теста, вычислив вероятность $P(Z > 200)$ при условии, что нулевая гипотеза верна ($k = 1$). Для этого мы строим F-распределение с $(20 + 2 * 1)$ степенями свободы и находим вероятность попадания в правый хвост этого распределения.

Таким образом, найденная вероятность будет являться уровнем значимости данного теста.

3

супер, но учти, что котов два и умножь каждую из степеней свободы на два, поэтому наша случайная величина Z будет иметь распределение Фишера с (20,2) степенями свободы



Понял, извините за неправильн

↻ Regenerate response

ас два кота и каждая из



Send a message.



наша случайная величина Z будет иметь распределение Фишера с $(20, 2)$ степенями свободы



Понял, извините за неправильное понимание. Если у нас два кота и каждая из степеней свободы умножается на два, то для нахождения уровня значимости мы будем использовать распределение Фишера (F-распределение) с $(20, 2)$ степенями свободы.



Мы хотим проверить нулевую гипотезу $H_0: \kappa = 1$ против альтернативной гипотезы $H_1: \kappa > 1$.

Уровень значимости теста будет определен как вероятность $P(Z > 200)$ при условии, что H_0 верна ($\kappa = 1$). Здесь Z - случайная величина, имеющая распределение Фишера с $(20, 2)$ степенями свободы.

Вычисление точной вероятности $P(Z > 200)$ для данного распределения может быть сложной задачей. Чтобы найти уровень значимости, мы можем воспользоваться статистическими таблицами Фишера или использовать программное обеспечение для статистического анализа, которое может вычислить эту вероятность.

Итак, уровень значимости данного теста будет определен как вероятность $P(Z > 200)$ для распределения Фишера с $(20, 2)$ степенями свободы.

↻ Regenerate response

Send a message.



наша случайная величина Z будет иметь распределение Фишера с $(20, 2)$ степенями свободы



Понял, извините за неправильное понимание. Если у нас два кота и каждая из степеней свободы умножается на два, то для нахождения уровня значимости мы будем использовать распределение Фишера (F-распределение) с $(20, 2)$ степенями свободы.



Мы хотим проверить нулевую гипотезу $H_0: \kappa = 1$ против альтернативной гипотезы $H_1: \kappa > 1$.

Уровень значимости теста будет определен как вероятность $P(Z > 200)$ при условии, что H_0 верна ($\kappa = 1$). Здесь Z - случайная величина, имеющая распределение Фишера с $(20, 2)$ степенями свободы.

Вычисление точной вероятности $P(Z > 200)$ для данного распределения может быть сложной задачей. Чтобы найти уровень значимости, мы можем воспользоваться статистическими таблицами Фишера или использовать программное обеспечение для статистического анализа, которое может вычислить эту вероятность.

Итак, уровень значимости данного теста будет определен как вероятность $P(Z > 200)$ для распределения Фишера с $(20, 2)$ степенями свободы.

 Regenerate response

Send a message.



3

но ты должен сравнивать не с 200, а с 20, ведь при переходе к распределению Фишера от хи-квадрата ты делил на числа 20 и 2, поэтому необходимо теперь найти вероятность, что $P(Z > 20)$



Понял, извините за неправильное понимание. Если мы сравниваем отношение с 20, то мы должны найти вероятность $P(Z > 20)$, где Z - случайная величина, имеющая распределение Фишера с (20, 2) степенями свободы.



Для нахождения этой вероятности можно воспользоваться таблицами распределения Фишера или использовать программное обеспечение для статистического анализа.

Итак, уровень значимости данного теста будет определен как вероятность $P(Z > 20)$ для распределения Фишера с (20, 2) степенями свободы.

3

посчитай эту вероятность, как $1 - P(Z < 20)$



Для расчета вероятности $P(Z > 20)$, где Z имеет распределение Фишера с (20, 2) степенями свободы, мы можем использовать свойство симметрии распределения Фишера и вычислить $1 - P(Z < 20)$



↻ Regenerate response

Send a message.



Фишера и вычислить $1 - P(Z < 20)$.

Поэтому мы ищем $P(Z < 20)$ для распределения Фишера с (20, 2) степенями свободы.

К сожалению, точное вычисление этой вероятности требует использования специальных таблиц или программного обеспечения для статистического анализа.

Пожалуйста, обратитесь к соответствующим таблицам или программному обеспечению для нахождения вероятности $P(Z < 20)$ и затем вычислите $1 - P(Z < 20)$, чтобы получить искомую вероятность $P(Z > 20)$.

3

$P(X < 20) \sim 0.95$



Если вероятность $P(X < 20)$ для распределения Фишера с (20, 2) степенями свободы составляет примерно 0.95, то вероятность $P(Z > 20)$ будет примерно $1 - 0.95 = 0.05$.



Таким образом, уровень значимости данного теста будет составлять около 0.05, что означает, что мы можем отвергнуть нулевую гипотезу на уровне значимости 0.05, если отношение суммарного роста к суммарному весу двух случайно взятых котов превышает 200.

↺ Regenerate response

Send a message.

