```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

1. Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе впервые приехал таксист, который уже раньше приезжал к туристу. Для упрощения предположим, что все n таксистов Самарканда всегда на работе и приезжают равновероятно.

а) [5] Постройте график функции правдоподобия как функции от общего количества такси

n. Найдите оценку числа n методом максимального правдоподобия.

б) [5] Постройте график математического ожидания номера заказа, на котором происходит первый повторный приезд, как функции от общего количества такси n. Найдите оценку числа n методом моментов.

в) [15] Предположим, что настоящее n равно 100. Проведя 10000 симуляций вызовов такси до первого повторного, рассчитайте 10000 оценок методом моментов и 10000 оценок меотодом максимального правдоподобия. Постройте гистограммы для оценок двух методов. Оцените смещение, дисперсию и среднеквадратичную ошибку двух методов.

Вероятность встретить того же человека на k-ый шаг: P(X=k) = (n-1)/n * (n-2)/n ... (n-k+1)/n * k/n Так как в нашей задаче только одно наблюдение, то эта вероятность и будет функцией правдоподобия

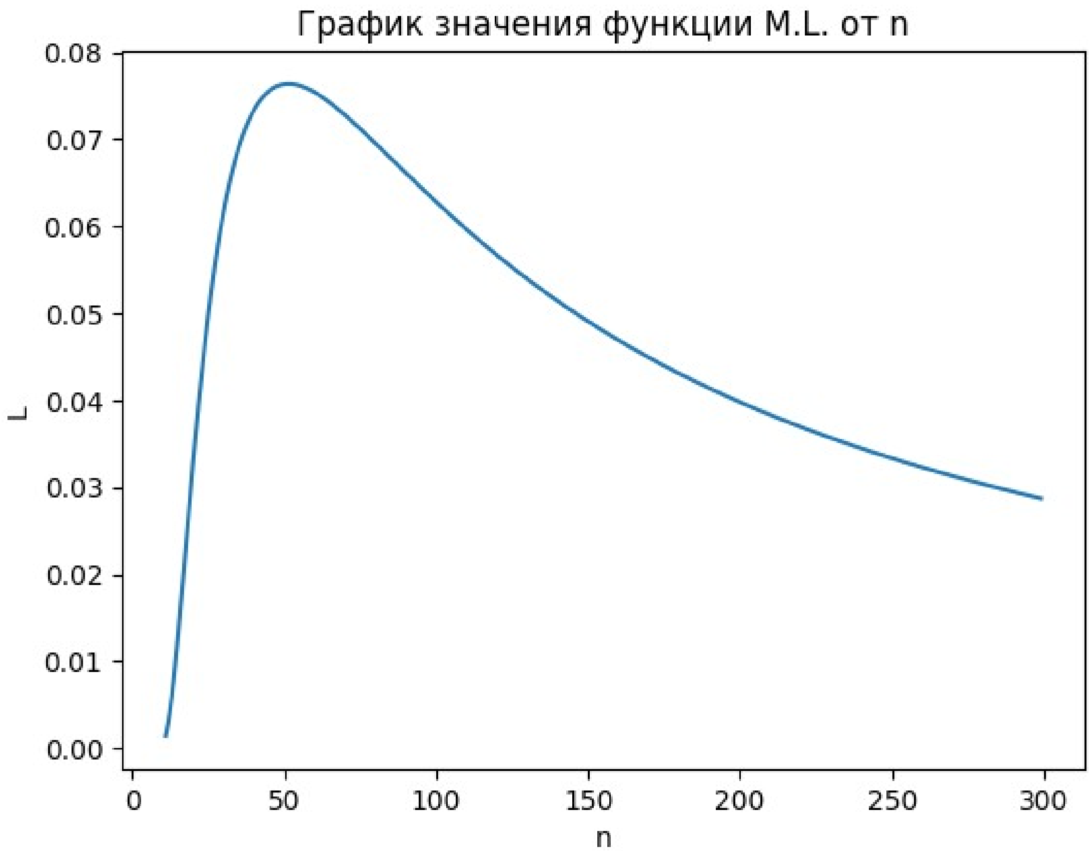
##Пункт а решение

```python
def new_prod(a, b):
  pr = 1
  for i in range(a, b):
    pr *= i
  return pr

def L(n, k = 10):
  return new_prod(n-k+1, n)/(n**(k))*k

L(150)
np.prod(np.arange(141, 150))
```

```
-8599630564824979712
```

```python
vec = np.arange(11, 300)
l_c = np.vectorize(L)
y = l_c(vec)
plt.plot(np.arange(11, 300), y)
plt.title('График значения функции M.L. от n')
plt.xlabel('n')
plt.ylabel('L')
```

```
Text(0, 0.5, 'L')
```



Оценка максимального правдоподобия:

```
np.argmax(y)+11
```

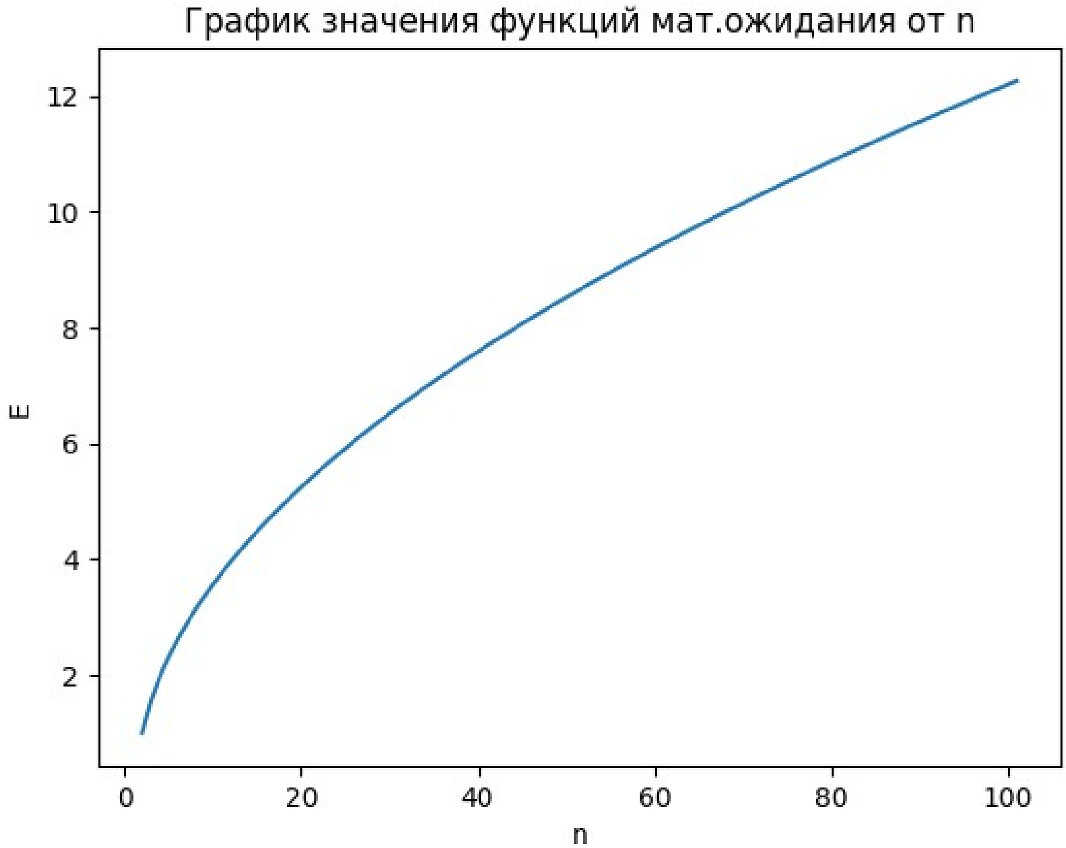
```
51
```

##Пункта б решение

Расчет мат ожидания

```
def E(n):
    vec_k = np.arange(2, n+1)
    P_v = np.vectorize(L)# так как L и P в этой заче одно и то же
    Vec_p = P_v(n, vec_k)
    return vec_k@Vec_p

e_c = np.vectorize(E)
n_all = np.arange(2, 102)
otv = e_c(n_all)
plt.title('График значения функций мат.ожидания от n')
plt.xlabel('n')
plt.ylabel('E')
plt.plot(n_all, otv)
```

```
[<matplotlib.lines.Line2D at 0x7f98759f53c0>]
```




```
for i in range(len(otv)):
  if abs(otv[i] - 10) < 0.001:
    print(i)
```

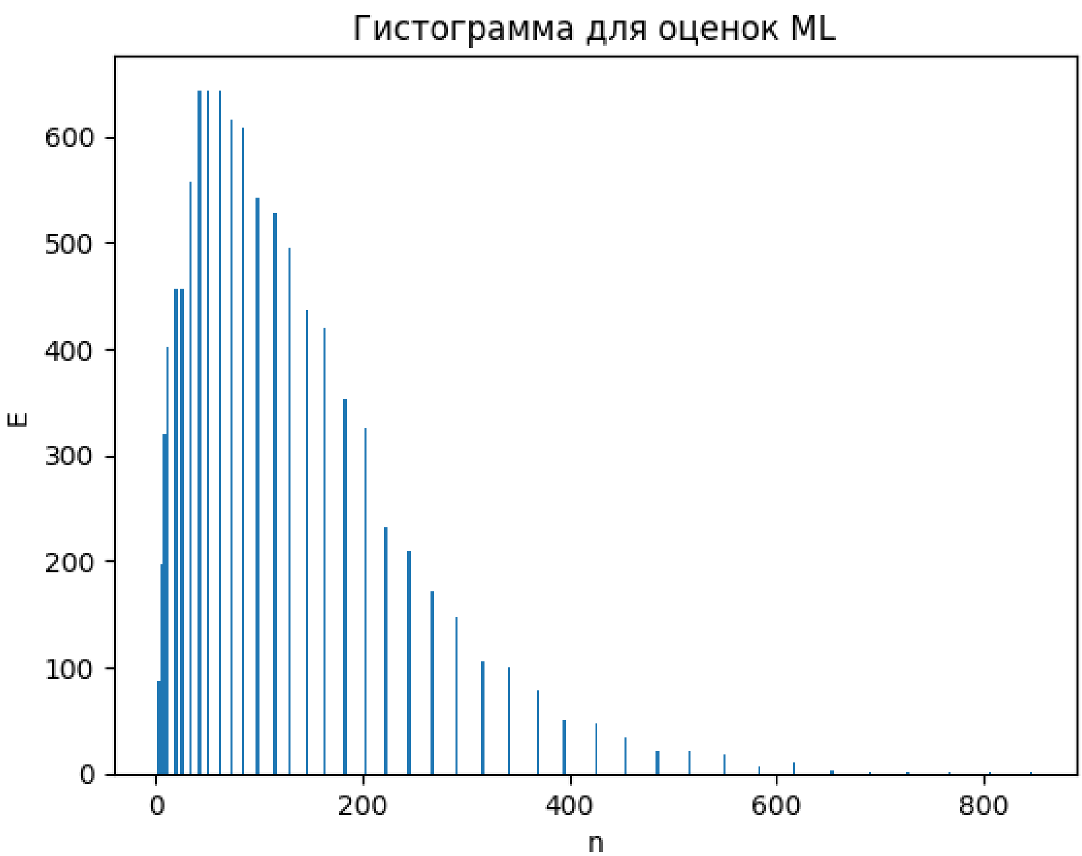
```
66
```

##Пункт в решение

```
np.random.seed(3)
n = 100
taxis = np.arange(1, n+1)
otv1 = []
for i in range(1, 10001):
    one = np.random.choice(taxis)
    biv = []
    while np.isin(one, biv) == False:
        biv.append(one)
        one = np.random.choice(taxis)

    k = len(biv)+1
    otv1.append(k)
```

```python
def L_max(k):
    n = k-1
    L_f1 = L(n, k)
    L_f2 = L(n, k)
    while L_f1 <= L_f2:
        L_f1 = L(n, k)
        n += 1
        L_f2 = L(n, k)
    return n-1

L_v = np.vectorize(L_max)
L_h = L_v(otv1)
plt.hist(L_h, bins = 300);
plt.title('Гистограмма для оценок ML')
plt.xlabel('n')
plt.ylabel('E')
```

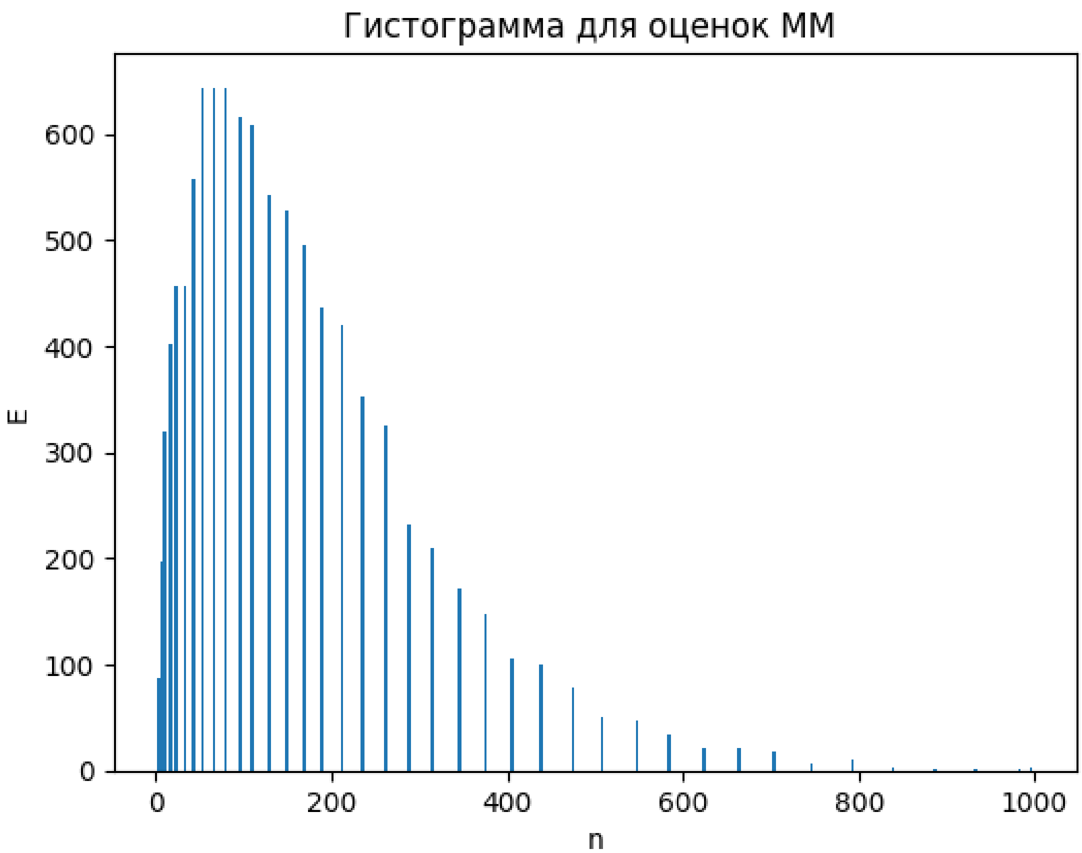
```
Text(0, 0.5, 'E')
```




```python
otv2 = np.array(otv1)
n1 = np.arange(2, 1000)
e_c1 = np.vectorize(E)
Er = e_c1(n1)
Er = Er[:, np.newaxis]
```

```
Dr = otv2[np.newaxis, :]
Er = np.absolute(Er - Dr)
all_Mu = np.argmin(Er, axis = 0)
plt.hist(all_Mu, bins = 300);
plt.title('Гистограмма для оценок ММ ')
plt.xlabel('n')
plt.ylabel('E')
```

```
Text(0, 0.5, 'E')
```




```
s_L = abs(100 - np.mean(L_h))
s_M = abs(100 - np.mean(all_Mu))
var_L = np.var(L_h)
var_M = np.var(all_Mu)
sig_L = np.std(L_h)
sig_M = np.std(all_Mu)
print(s_L, var_L, sig_L)
print(s_M, var_M, sig_M)
```

```
8.165899999999993 9706.89297719 98.52356559316152
39.0761 16009.858708790001 126.53007037376531
```

2) Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе он обнаружил, что у таксистов было 6 разных имён. Для упрощения предположим, что все n имён среди таксистов встречаются равновероятно и независимо от поездки к поездке.

а) [5] Постройте график функции правдоподобия как функции от общего количества имён n. Найдите оценку числа n методом максимального правдоподобия.

б) [5] Постройте график математического ожидания числа разных имён у 10 таксистов, как функции от общего количества имён n. Найдите оценку числа n методом моментов.

в) [15] Предположим, что настоящее n равно 20. Проведя 10000 симуляций десяти вызовов такси, рассчитайте 10000 оценок методом моментов и 10000 оценок методом максимально- го правдоподобия. Постройте гистограммы для оценок двух методов. Оцените смещение, дисперсию и среднеквадратичную ошибку двух методов.

Update 2023-06-07: если по выборке в симуляциях оценка метода моментов или метода максимального правдоподобия стремится к бесконечности, то можно ограничить её свер- ху большим числом, например, 100.

Решение пункта а

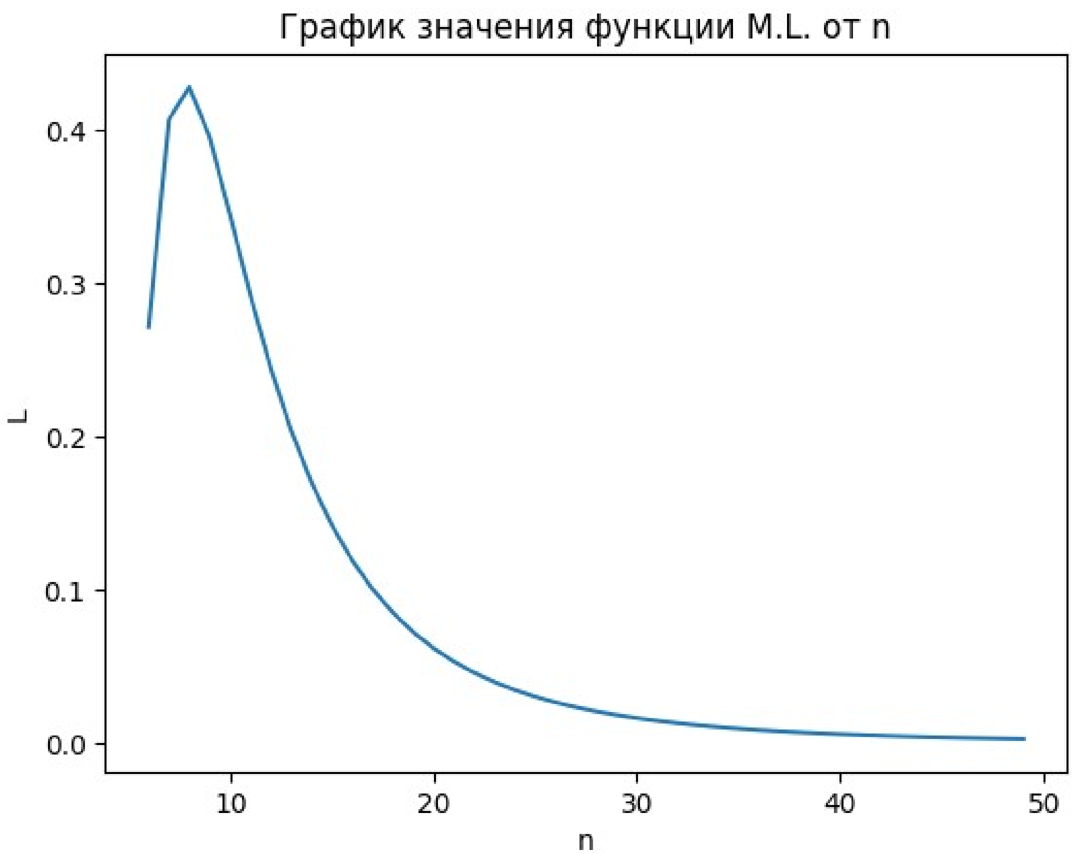
так снова нужна функция правдоподобия и она тут будет тоже совпадать с вероятностью

$P(X=k)=(n-1)/n*(n-2)/n*\ldots*(n-k+1)/n*const$

$const$ - сумма произведений всех комбинаций по N-k чисел от 1 до к

```
from itertools import combinations_with_replacement as cwm
def const_counter(k, n, N):
    comb = cwm(np.arange(1, k+1), N - k)
    cnt = 0
    for i in comb:
        mult = 1
        for j in range(N - k):
            mult *= i[j]
        cnt += mult
    return cnt

def L_2(n, k = 6, N = 10):
    p=1
    for i in range(1, k):
        p*= ((n-i)/n)
    p *= (const_counter(k, n, N)/(n**(N - k)))
    return p

n_vec = np.arange(6, 50)
L_c = np.vectorize(L_2)
```

```python
L3 = L_c(n_vec)
plt.plot(n_vec, L3)
plt.title('График значения функции M.L. от n')
plt.xlabel('n')
plt.ylabel('L')
plt.show()
```




```python
np.argmax(L3)+6
```

8

8 это и есть ML оценка

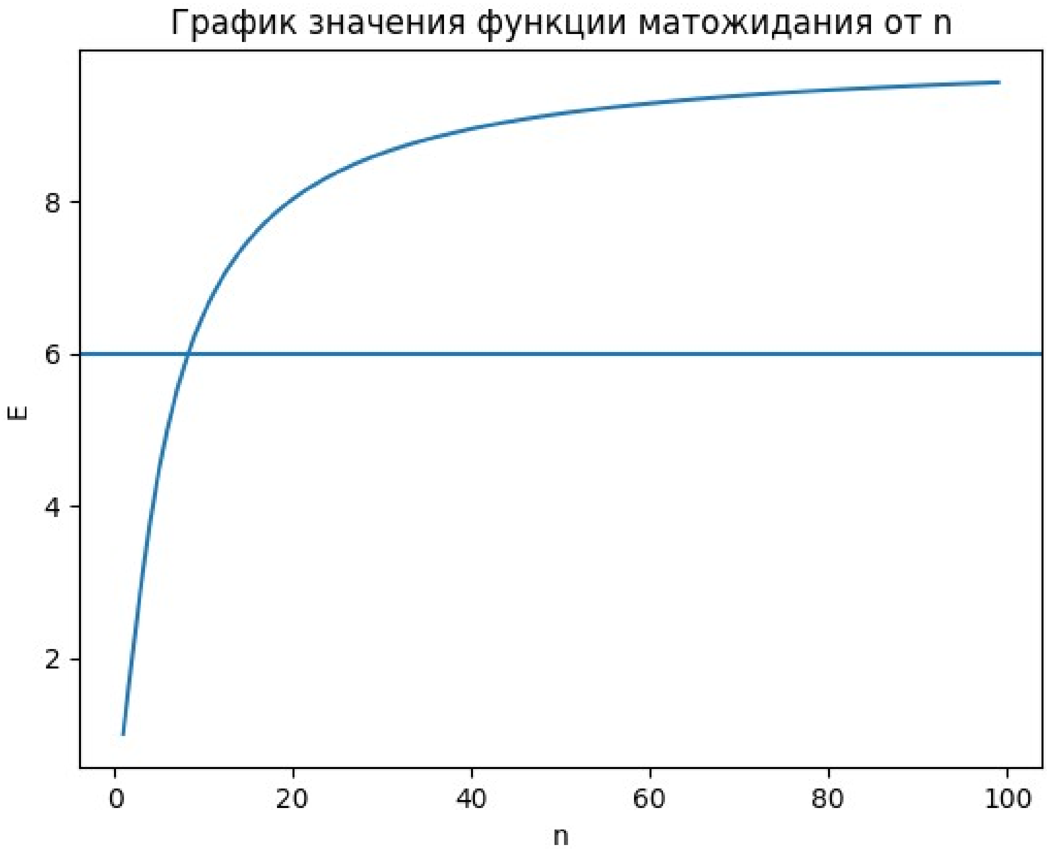
Решение пункта б

```python
def E_2(n, N):
    vec_k = np.arange(1, N+1)
    P_v = np.vectorize(L_2)
    Vec_p = P_v(n, vec_k, N)
    return vec_k@Vec_p
```

Рисуем теперь картинку

```python
n_all = np.arange(1, 100)
all_mu1 = []
```

```
for n in n_all:
    all_mu1.append(E_2(n, 10))
plt.plot(n_all, all_mu1);

plt.title('График значения функции матожидания от n')
plt.xlabel('n')
plt.ylabel('E')
plt.axhline(y= 6);
plt.show()
```




```
n_all[abs(np.array(all_mu1) - 6) == min(abs(np.array(all_mu1) - 6))]
```

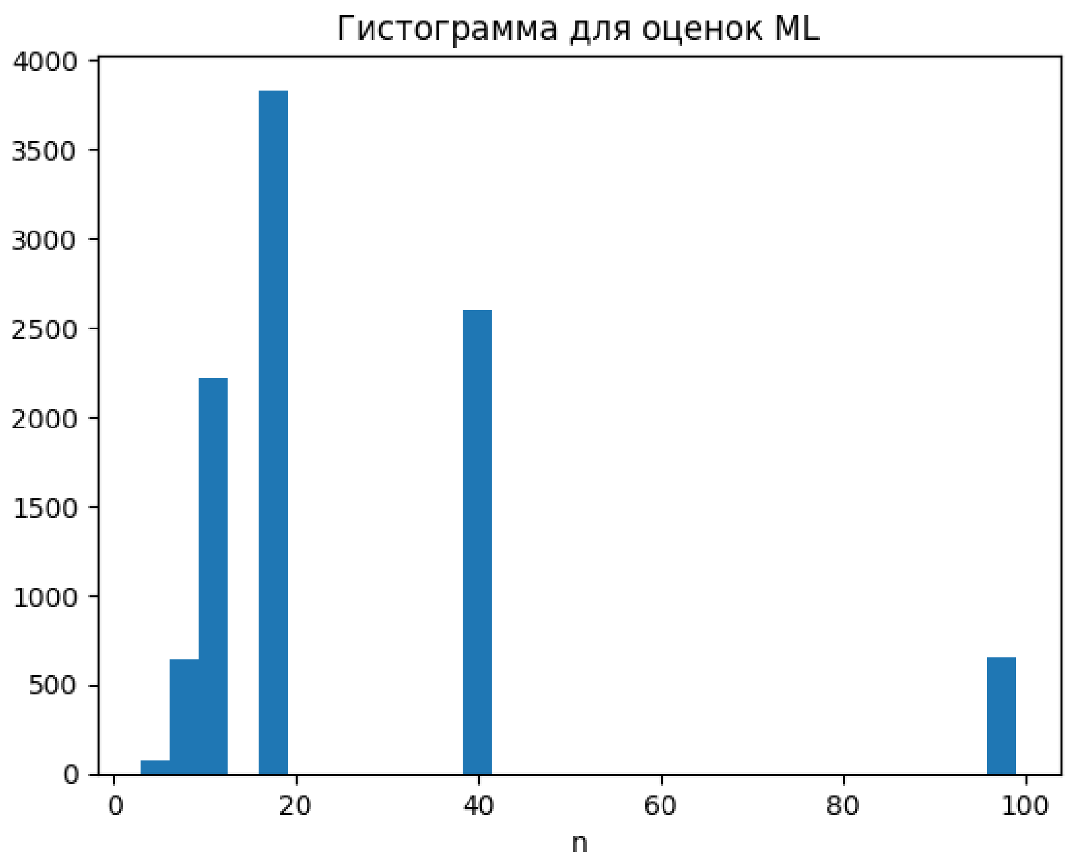
```
array([8])
```

оценка методом моментов тоже 8

Решение пункта в

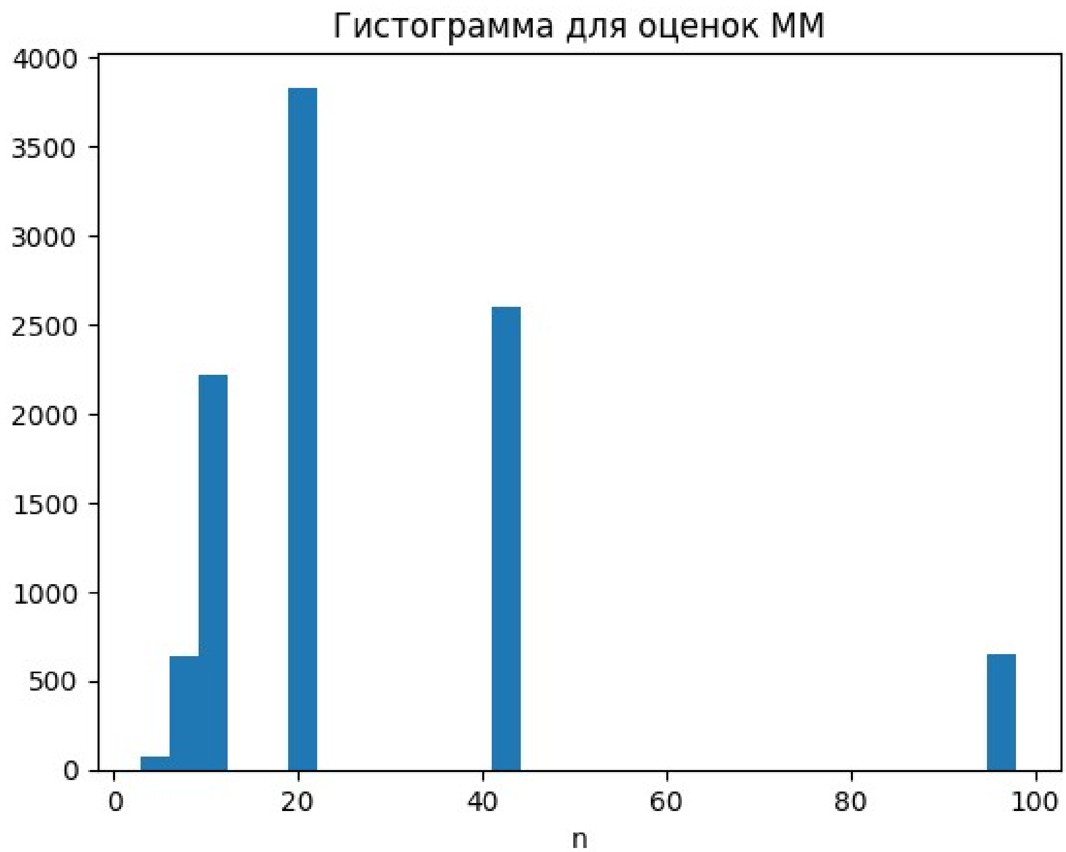
```
np.random.seed(42)
names = np.arange(1, 21)
ans = []
for i in range(1, 10001):
    one = np.random.choice(names, 10)
```

```
    ans.append(len(np.unique(one)))


def L_max2(k):
  n = np.arange(1, 101)
  a = L_c(n,k,  10)
  return np.argmax(a)

L_v = np.vectorize(L_max2)
all_L = L_v(ans)

plt.hist(all_L, bins = 30);
plt.title('Гистограмма для оценок ML')
plt.xlabel('n')
plt.show()
```




```
ans3 = np.array(ans)
n_mu = np.arange(1, 100)
E_c2 = np.vectorize(E_2)
E_r2 = E_c2(n_mu, 10)
E_r2 = E_r2[:, np.newaxis]
Dr2 = ans3[np.newaxis, :]
E_r2 = np.absolute(E_r2 - Dr2)
all_mu2 = np.argmin(E_r2, axis = 0)
```

```
plt.hist(all_mu2, bins = 30);
plt.title('Гистограмма для оценок ММ')
plt.xlabel('n')
plt.show()
```




```
s_L2 = abs(20 - np.mean(all_L))
s_M2 = abs(20 - np.mean(all_mu2))
var_L2 = np.var(all_L)
var_M2 = np.var(all_mu2)
sig_L2 = np.std(all_L)
sig_M2 = np.std(all_mu2)
print(s_L2, var_L2, sig_L2)
print(s_M2, var_M2, sig_M2)
```

```
6.885400000000001 505.28846684 22.47862244088814
7.2094999999999985 489.1584097500001 22.116925865725555
```

3.Иноагент Иннокентий по 20 наблюдениям строит 95%-й доверительный интервал для математического ожидания несколькими способами: классический асимптотический нормальный интервал, с помощью наивного бутстрэпа, с помощью бутстрэпа t-статистики.

а) [15]Для каждого способа с помощью 10000 симуляций оцените вероятность того,что номинально 95%-й доверительный интервал фактически накрывает математическое ожидание, если наблюдения распределены экспоненциально с интенсивностью 1.

б) [5]Пересчитайте вероятности накрытия, если наблюдения имеют распределение Стьюдента с тремя степенями свободы.

в) [5] Какой способ оказался лучше?

Решение пункт а

```
np.random.seed(322)
all = np.random.exponential(1, size=(10**4, 20))
```

```
from statsmodels.stats import weightstats as stests
```

```
import scipy.stats as sps
def z_test(obs, h0_mu, alpha, n):
    dist = sps.norm(loc=0, scale=1)
    mu = np.mean(obs)
    std = np.std(obs)/np.sqrt(n)
    q_r = dist.ppf(1 - alpha/2)
    q_l = dist.ppf(alpha/2)
    left_bord = mu + q_l*std
    right_bord = mu + q_r*std
    ci = np.logical_and(left_bord <= h0_mu, right_bord >= h0_mu)
    return ci
```

```
ans4 = np.apply_along_axis(z_test, 1, all, 1, 0.05, 20)
np.mean(ans4)
```

```
0.8982
```

2-ой метод

```
def n_boot(obs, h0_mu, alpha, n):
    pr = alpha*100
    indices = np.random.choice(np.arange(n), size=(10**4, n))
    means = np.mean(obs[indices], axis=1)
    q_l = np.percentile(means, pr/2)
    q_r = np.percentile(means, 100- pr/2)
    ci = np.mean(np.logical_and(q_l <= h0_mu, q_r >= h0_mu))
    return ci
```

```
ans5 = np.apply_along_axis(n_boot, 1, all, 1, 0.05, 20)
naive_boot_exp_res = np.mean(ans5)
naive_boot_exp_res
```

```
0.9014
```

3-ий метод

```
from scipy.stats import t
def t_stat_boot(obs, h0_mu, alpha, n):
```

```
    pr = alpha*100
    indices = np.random.choice(np.arange(n), size=(10**4, n))
    means = np.mean(obs[indices], axis=1)
    mean_obs = np.mean(obs)
    se = (np.std(obs[indices], axis=1, ddof = 1))/np.sqrt(n)
    t_stat = (means - mean_obs)/se
    q_l = np.percentile(t_stat, pr/2)
    q_r = np.percentile(t_stat, 100- pr/2)
    l_bord = mean_obs - q_r*(np.std(obs, ddof = 1))/np.sqrt(n)
    r_bord = mean_obs - q_l*(np.std(obs, ddof = 1))/np.sqrt(n)
    ci = np.mean(np.logical_and(l_bord <= h0_mu, r_bord >= h0_mu))
    return ci
```

```
ans6 = np.apply_along_axis(t_stat_boot, 1, all, 1, 0.05, 20)
r3 = np.mean(ans6)
r3
```

```
0.9445
```

Решение пункт б

```
np.random.seed(322)
all1 = np.random.standard_t(3, size=(10**4,20))
```

```
a = np.apply_along_axis(z_test, 1, all1, 0, 0.05, 20)
np.mean(a)
```

```
0.9344
```

```
an1 = np.apply_along_axis(n_boot, 1, all1, 0, 0.05, 20)
np.mean(an1)
```

```
0.9187
```

```
an2 = np.apply_along_axis(t_stat_boot, 1, all, 1, 0.05, 20)
np.mean(an2)
```

```
0.9454
```

И в первом и втором случае луше всего проявмл себя бутстрэп т статистики

4) Проверьте гипотезу о том, что ожидаемые результаты экзамена по теории вероятностей тех, у кого фамилия начинается с гласной буквы и с согласной буквы, равны. В качестве альтернатив- ной гипотезы возьмите гипотезу о неравенстве.

а) [5] Используйте тест Уэлча.

б) [5] Используйте наивный бутстрэп.

в) [5] Используйте бутстрэп t-статистики.

г) [5] Используйте перестановочный тест.

```
df = pd.read_csv('exam.csv')
```

```
df
```

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

[332 rows x 2 columns]
```

```
def fl(surname):
    glasn= {'А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы', 'Э', 'Ю', 'Я'}
    if surname[0] in glasn:
        return True
    return False
```

```
a = np.vectorize(fl)(df['Surname'])
s = df[~a]
g = df[a]
```

```
from scipy.stats import ttest_ind
welch = ttest_ind(s["Score"], g["Score"], equal_var=False,
alternative='two-sided')
welch
```

```
Ttest_indResult(statistic=0.8519661870595602,
pvalue=0.3974027153843839)
```

Нет причин отвергать нулевую гипотезу

Пункт б

```
np.random.seed(110)
s_boot = np.random.choice(s['Score'], size=(10000,
s['Score'].shape[0]))
g_boot = np.random.choice(g['Score'], size=(10000,
g['Score'].shape[0]))
```

```
b_s= np.array([np.mean(s_s) - np.mean(g_s) for s_s, g_s in zip(s_boot,
g_boot)])
```

```
np.quantile(b_s, q=0.025), np.quantile(b_s, q=0.975)
```

```
(-1.3219315641450935, 3.557375423667702)
```

0 попадает в интервал нет причин отвергать нулевую гипотезу

Пункт в

```
np.random.seed(444)
s_boot1 = np.random.choice(s['Score'], size=(10000,
s['Score'].shape[0]))
g_boot1 = np.random.choice(g['Score'], size=(10000,
g['Score'].shape[0]))
b_s1= np.array([np.mean(s_s) - np.mean(g_s) - (np.mean(s_s) -
np.mean(g_s))/np.sqrt(np.var(s_s)/s['Score'].shape[0]+np.var(g_s)/
g['Score'].shape[0]) for s_s, g_s in zip(s_boot1, g_boot1)])
```

```
np.quantile(b_s1, q=0.025), np.quantile(b_s1, q=0.975)
```

```
(-0.19248677455706376, 0.8601854353005459)
```

```
q1 = s['Score'].mean()-g['Score'].mean()-np.quantile(b_s1,
q=0.975)*np.sqrt(np.var(s['Score'])/s['Score'].shape[0]+np.var(g['Scor
e'])/g['Score'].shape[0])
q2 = s['Score'].mean()-g['Score'].mean()-np.quantile(b_s1,
q=0.025)*np.sqrt(np.var(s['Score'])/s['Score'].shape[0]+np.var(g['Scor
e'])/g['Score'].shape[0])
q1, q2
```

```
(-0.0005049154960243829, 1.3196386644191713)
```

Хоть интервал и сжался, но всё ещё нет оснований отвергать H0

Пункт г

```
s_sample = s['Score'].values
g_sample = g['Score'].values
n_s, n_g = len(s_sample), len(g_sample)
```

```
from mlxtend.evaluate import permutation_test
p_value = permutation_test(s_sample, g_sample, method='approximate',
num_rounds=10000,seed=144)
print(p_value)
```

```
0.3818
```

H0 не отвергается

5. Составьте таблицу сопряжённости, поделив студентов писавших экзамен на четыре группы по двум признакам: набрал ли больше медианы или нет, на согласную или гласную букву начинается фамилия.

а) [5]Постройте 95% асимптотический интервал для отношения шансов хорошо написать экзамен («несогласных» к «согласным»). Проверьте гипотезу о том, что отношение шансов равно 1 и укажите P-значение.

б) [5] Постройте 95% асимптотический интервал для отношения вероятностей хорошо напи- сать экзамен. Проверьте гипотезу о том, что отношение вероятностей равно 1 и укажите P-значение.

в) [5] Постройте 95% интервал для отношения шансов хорошо написать экзамен с помощью наивного бутстрэпа. Проверьте гипотезу о том, что отношение шансов равно 1 и укажите P-значение.

```
import scipy.stats as sts
```

```
z_crit = sts.norm.ppf(0.975)
```

```
med = np.median(df['Score'])
```

```
s_m, s_nm = s.iloc[np.where(s['Score'] >= med)],
s.iloc[np.where(s['Score'] < med)]
g_m, g_nm = g.iloc[np.where(g['Score'] >= med)],
g.iloc[np.where(g['Score'] < med)]
```

```
a = s_m.shape[0]
b = s_nm.shape[0]
c = g_m.shape[0]
d = g_nm.shape[0]
o_s = a/(b+a)
o_g = c/(d+c)
```

```
OR = (o_g / (1 - o_g )) / (o_s / (1 - o_s))
se = np.sqrt(1 / ((1 - o_g) * o_g * d) + 1 / ((1 - o_s) * o_s * b))
OR, se
```

```
(0.7137931034482758, 0.41813464976072384)
```

```
q_l = np.exp(np.log(OR) - z_crit * se)
q_r = np.exp(np.log(OR) + z_crit * se)
q_l, q_r
```

```
(0.31452522985825765, 1.6199037347812448)
```

```
p_value = 2 * sts.norm.cdf(np.log(OR)/se)
```

```
p_value
```

```
0.42004209535763193
```

H0 не отвергается

пункт б

```
dis = sts.norm(loc = 0, scale = 1)
ver_ras = np.log(c/(c+d)) - np.log(a/(a+b))
ver_se = np.sqrt(1/a-1/(a+b)+1/c-1/(c+d))
obs= (ver_ras)/ver_se
dist = sps.norm(loc=0, scale=1)
```

```
ci = np.exp([ver_ras - dist.ppf(0.975)*ver_se, ver_ras +
dist.ppf(0.975)*ver_se])
p_value =  2*min([dis.cdf(obs), 1-dis.cdf(obs)])

print(p_value, ci)
```

```
0.3070947928050546 [0.59375296 1.1783587 ]
```

H0 не отвергается

Пункт в

```
s_boot2 = np.random.choice(s_sample, size=(10**4, s_sample.shape[0]))
g_boot2 = np.random.choice(g_sample, size=(10**4, g_sample.shape[0]))

def one_count1(x, y):
    a = (x > med).sum() / len(x)
    b = (y > med).sum() / len(y)
    return (b / (1 - b)) / (a / (1 - a))

boot_count = np.array([one_count1(s, g) for s, g in zip(s_boot2,
g_boot2)])

np.quantile(boot_count, q=0.025), np.quantile(boot_count, q=0.975)
```

```
(0.3700680272108843, 1.32653743315508)
```

```
OR1 = ((o_g) / (1 - (o_g))) / ((o_s) / (1 - (o_s)))
p_value = 2 * min(np.mean(boot_count <= OR1), np.mean(boot_count >
OR1))
p_value
```

```
0.981
```

Гипотеза не отвергается

6. Иноагент Иннокентий Вероятностно-Статистический считает, что длина фамилии положительно влияет на результат экзамена по теории вероятностей. А именно, он предполагает, что ожидаемый результат за экзамен прямо пропорционален длине фамилии, $E(Y_i) = \beta F_i$, где Y_i — результат за экзамен по 30-балльной шкале, F_i — количество букв в фамилии.

а) [10] Оцените β методом моментов. Рассчитайте выборочную корреляцию.

б) [5] С помощью перестановочного теста найдите P -значение и формально протестируйте гипотезу о том, что корреляция равна нулю.

```
df['length'] = df['Surname'].apply(len)
m1, m2 = df['Score'].mean(), df['length'].mean()
beta = m1/m2
beta
```

```
2.0613026819923372
```

```
np.corrcoef(df['Score'], df['length'])[0,1]
```

```
0.025328052669147665
```

ПУНКТ Б

```
def p_test(df, n, alpha):
    alpha = 5
    corrs = []
    for i in range(n):
        p1 = np.random.choice(df['Score'], size=len(df['Score']),
replace=False)
        corrs.append(np.corrcoef(p1, df['length'])[0][1])
    q_l, q_r = np.percentile(corrs, alpha/2), np.percentile(corrs, 100
- alpha/2)
    p_value = min(1 - (np.array(corrs) < 0).sum()/len(corrs),
(np.array(corrs) < 0).sum()/len(corrs)) * 2
    return p_value
```

```
p_test(df, 10**4, 0.05)
```

```
0.9847999999999999
```

Гипотеза не отвергается

Задача 7

```
https://chat.openai.com/share/3065a1be-49ad-44ee-bcbd-a28358198d5a
```

Я так напугал ghat-gpt, что он перестал пытаться пересечь события, а посчитал по формуле полной вероятности

Задача 8

https://bdemeshev.github.io/sc401/notes_ranepa_2016/condition-on-algebra.html
Помогло разобраться с условными мат ожиданиями, и решило пару задача о которых были вопросы