In [540…
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats as sts
from sklearn.metrics import mean_squared_error, mean_absolute_error
import itertools
import random
```

Задание 3

а)

In [509…
```python
#Создадим выборку!
np.random.seed(1)
values = np.random.exponential(scale=1.0, size=(10000,20))

#сделаем асимптотический нормальный интервал для каждой симуляции
def norm_interv(v):
    interv = sts.norm.interval(confidence=0.95, loc=np.mean(v), scale=(np.var(v)/len(v))**0.5)
    return interv
norm_int_intervals = np.apply_along_axis(norm_interv, axis=1, arr=values)

#проверим покрытие матожидания интервалом
def in_int(i):
    return i[0] <= 1 <= i[1]

norm_int_results = np.apply_along_axis(in_int, axis=1, arr=norm_int_intervals)
norm_int_prob = np.mean(norm_int_results)

#наивно пробутстрапим
def naive_boot_interval(a):
    samples = np.mean(np.random.choice(a, size=(10000,20), replace=True),axis=1)
    return [np.quantile(samples, 0.025), np.quantile(samples, 0.975)]

naive_boot_intervals = np.apply_along_axis(naive_boot_interval, axis=1, arr=values)
naive_boot_results = np.apply_along_axis(in_int, axis=1, arr=naive_boot_intervals)
naive_boot_prob = np.mean(naive_boot_results)

#сделаем те же операции но для t-статистики
def tstat_interval(s):
    mean = np.mean(s)
    samples = np.random.choice(s, size=(10000,20), replace=True)
    samples_means = np.mean(samples,axis=1)
    se_init = np.std(s, ddof=1)/(len(s))**0.5
    se = np.std(samples, axis=1, ddof=1)/(len(s))**0.5
    quantiles = [np.quantile((samples_means-mean)/se, 0.025), np.quantile((samples_means-mean)/se, 0.975)]
    interval = [mean-quantiles[1]*se_init,mean-quantiles[0]*se_init]
    return interval
tstat_boot_intervals = np.apply_along_axis(tstat_interval, axis=1, arr=values)
tstat_boot_results = np.apply_along_axis(in_int, axis=1, arr=tstat_boot_intervals)
tstat_boot_prob = np.mean(tstat_boot_results)
print('Вероятность накрытия нормальным интервалом =', norm_int_prob)
print('Вероятность накрытия наивным бутстреп интервалом =', naive_boot_prob)
print('Вероятность накрытия t-статическим бутстреп интервалом =', tstat_boot_prob)
```

```
Вероятность накрытия нормальным интервалом = 0.8985
Вероятность накрытия наивным бутстреп интервалом = 0.9063
Вероятность накрытия t-статическим бутстреп интервалом = 0.9485
```

б)

In [510…
```python
#Сделаем то же для Стьюдента
np.random.seed(2)
values_student = np.random.standard_t(df=3, size=(10000,20))
def in_int(i):
    return i[0] <= 0 <= i[1]

norm_int_intervals = np.apply_along_axis(norm_interv, axis=1, arr=values_student)
norm_int_results = np.apply_along_axis(in_int, axis=1, arr=norm_int_intervals)
norm_int_prob = np.mean(norm_int_results)

naive_boot_intervals = np.apply_along_axis(naive_boot_interval, axis=1, arr=values_student)
naive_boot_results = np.apply_along_axis(in_int, axis=1, arr=naive_boot_intervals)
naive_boot_prob = np.mean(naive_boot_results)

tstat_boot_intervals = np.apply_along_axis(tstat_interval, axis=1, arr=values_student)
tstat_boot_results = np.apply_along_axis(in_int, axis=1, arr=tstat_boot_intervals)
tstat_boot_prob = np.mean(tstat_boot_results)

print('Вероятность накрытия нормальным интервалом =', norm_int_prob)
print('Вероятность накрытия наивным бутстреп интервалом =', naive_boot_prob)
print('Вероятность накрытия t-статическим бутстреп интервалом =', tstat_boot_prob)
```

```
Вероятность накрытия нормальным интервалом = 0.9335
Вероятность накрытия наивным бутстреп интервалом = 0.9177
Вероятность накрытия t-статическим бутстреп интервалом = 0.9216
```

в) Для первом случае t-статический бутстреп интервал оказался точнее, а во втором нормальный.

Задание 4

In [95]:
```python
df = pd.read_csv('/Users/cernenkostanislav/Desktop/Учеба другое/22-23_hse_probability - Exam.csv')
```

In [138…
```python
#Сделаем функцию для проверки первой буквы
def start_with_glasn(s):
    glasn = ['а', 'и', 'е', 'ё', 'о', 'у', 'ы', 'э', 'ю', 'я']
    return s.lower()[0] in glasn
start_with_glasn_vec = np.vectorize(start_with_glasn)

#Достанем из таблицы фамилии и оценки
surnames = df['Last name'][df['Last name'].isnull() == False]
marks = df['Unnamed: 72'][df['Last name'].isnull() == False]
glasn_marks = marks[start_with_glasn_vec(surnames)]
soglasn_marks =  marks[start_with_glasn_vec(surnames)==False]

# Сделаем функцию, которая будет говорить, отвергается гипотеза или нет
def otverg(p_val, alpha):
    if p_val >= alpha:
        return f'H0 не отвергается на уровне значимости {alpha}'
    else:
        return f'H0 отвергается на уровне значимости {alpha}'
```

а) Тест Уэлча

In [573…
```python
#Проверим с помощью F-теста, равны ли дисперсии, чтобы использовать этот результат в t-тесте
p = sts.f_oneway(glasn_marks, soglasn_marks)[1]
print('P-value =',p)
print(otverg(p_val=p, alpha=0.05))
```

```
P-value = 0.3799864037939755
H0 не отвергается на уровне значимости 0.05
```

In [574…
```python
#Дисперсии равны, используем t-тест
results_ttest = sts.ttest_ind(glasn_marks, soglasn_marks, equal_var=True)
print('P-value =',results_ttest[1])
print(otverg(p_val=results_ttest[1], alpha=0.05))
```

```
P-value = 0.3799864037939753
H0 не отвергается на уровне значимости 0.05
```

б) Наивный бутсрап

In [475…
```python
def naive_boot_interval(a, b):
    mean_a = np.mean(a)
    mean_b = np.mean(b)
    dif_mean = mean_a - mean_b
    samples_a = np.mean(np.random.choice(a, size=(10000,len(a)), replace=True),axis=1)
    samples_b = np.mean(np.random.choice(b, size=(10000,len(a)), replace=True),axis=1)
    dif_samples = samples_a - samples_b
    return [dif_mean - np.quantile(dif_samples-dif_mean, 0.975), dif_mean - np.quantile(dif_samples-dif_mean, 0.02
```

In [479…
```python
#Проверим, входит ли 0 в наш интервал
if in_int(naive_boot_interval(glasn_marks, soglasn_marks)):
    print('H0 не отвергается на уровне значимости 0.05')
else:
    print('H0 отвергается на уровне значимости 0.05')
```

```
H0 не отвергается на уровне значимости 0.05
```

в) бутстреп t-статистики

In [515…
```python
def tstat_boot_interval(a, b):
    mean_a = np.mean(a)
    mean_b = np.mean(b)
    samples_a = np.random.choice(a, size=(10000,20), replace=True)
    samples_b = np.random.choice(a, size=(10000,20), replace=True)
    samples_a_means = np.mean(samples_a,axis=1)
    samples_b_means = np.mean(samples_b,axis=1)
    diff_mean = mean_a - mean_b
    diff_samples_means = samples_a_means - samples_b_means
    se_init = np.std(a, ddof=1)/(len(a))**0.5 + np.std(b, ddof=1)/(len(b))**0.5
    se = np.std(samples_a, axis=1, ddof=1)/(len(s))**0.5 + np.std(samples_b, axis=1, ddof=1)/(len(s))**0.5
    quantiles = [np.quantile((diff_samples_means-diff_mean)/se, 0.025), np.quantile((diff_samples_means-diff_mean)
    interval = [diff_mean-quantiles[1]*se_init,diff_mean-quantiles[0]*se_init]
    return interval
```

In [516…
```python
#Проверим, входит ли 0 в наш интервал
if in_int(tstat_boot_interval(glasn_marks, soglasn_marks)):
    print('H0 не отвергается на уровне значимости 0.05')
else:
    print('H0 отвергается на уровне значимости 0.05')
```

```
H0 отвергается на уровне значимости 0.05
```

г) Перестановочный тест

In [529…
```python
all_marks = list(glasn_marks) + list(soglasn_marks)
```

In [566…
```python
glasn_indexes = list(range(len(glasn_marks)))
soglasn_indexes = list(range(len(glasn_marks),len(all_marks)))
all_indexes = list(range(len(all_marks)))

mean_glasn = np.mean(glasn_marks)
mean_soglasn = np.mean(soglasn_marks)
mean_diff = mean_glasn - mean_soglasn

boot_means_diff = []
for i in range(10001):
    new_positions = random.sample(all_indexes, k = len(all_indexes))
    d = dict(zip(new_positions, all_marks))
    boot_mean_glasn = np.mean([d.get(g) for g in glasn_indexes])
    boot_mean_soglasn = np.mean([d.get(s) for s in soglasn_indexes])
    boot_mean_diff = boot_mean_glasn - boot_mean_soglasn
    boot_means_diff.append(boot_mean_diff)
quantiles = [np.quantile(boot_means_diff,0.025), np.quantile(boot_means_diff,0.975)]

if quantiles[0] <= 0 <= quantiles[1]:
    print('H0 не отвергается на уровне значимости 0.05')
else:
    print('H0 отвергается на уровне значимости 0.05')
```

```
H0 не отвергается на уровне значимости 0.05
```

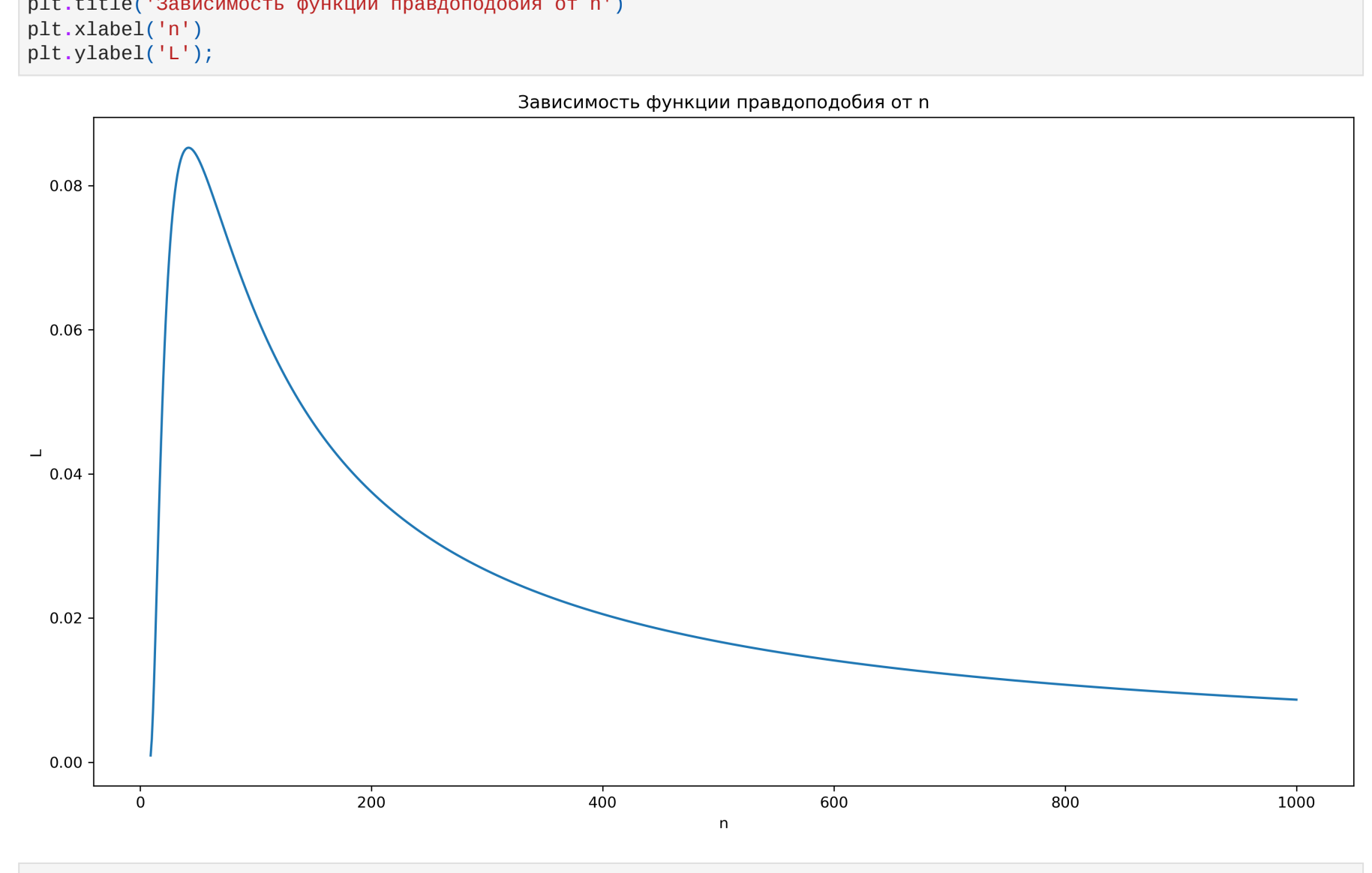
Задание 1

а)

Вероятность для первого дня встретить уникальное имя = 1. Для второго вероятность уже будет $p = \frac{n-1}{n}$, для третьего $p = \frac{n-2}{n}$ и тд до 10 дня, в 10 день $p = \frac{9}{n}$

In [261…
```python
#Запишем функцию максимального правдоподобия
Ls=[]

def ML(n,x):
    L = 1
    for i in range(2,x):
        L = L*((n-(i-1))/n)
    L = L*(x-1)/n
    return L

#Запишем возможные значения для n
ns=range(9,1001)

for n in range(9,1001):
    Ls.append(ML(n,10))
```

In [262…
```python
#Построим график функции правдоподобия от n
plt.figure(figsize=(15, 8), dpi = 300)
plt.plot(ns,Ls)
plt.title('Зависимость функции правдоподобия от n')
plt.xlabel('n')
plt.ylabel('L');
```

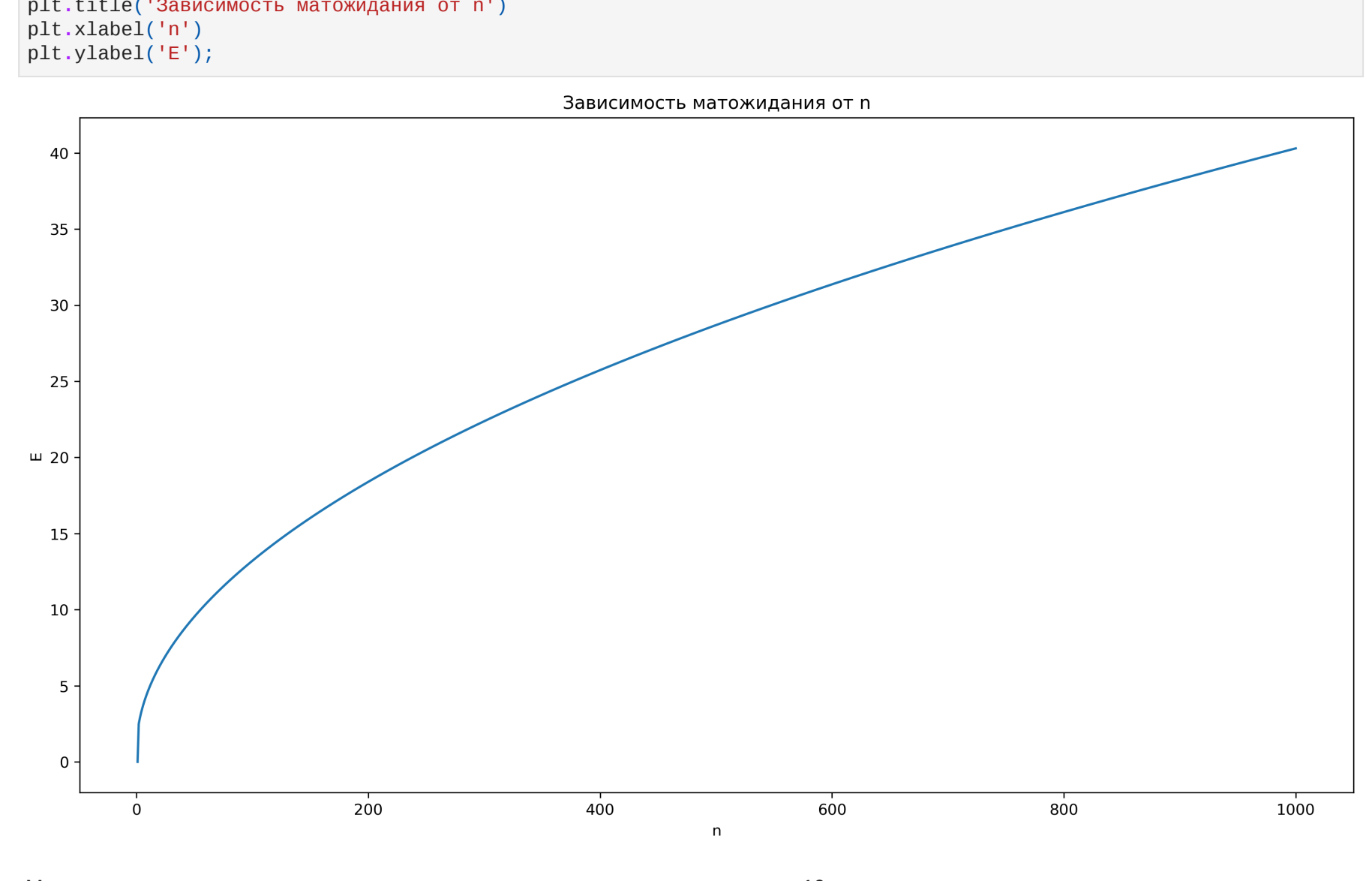


In [263…
```python
#Найдем максимум
best_n = ns[np.argmax(Ls)]
print('Оценка n методом максимального правдоподобия =', best_n)
```

```
Оценка n методом максимального правдоподобия = 42
```

б)

In [456…
```python
#Функция, чтобы находить вероятность в определенный день
def prob(x,n):
    if x == 1:
        return 0
    prob = 1
    for i in range(x-1):
        prob = prob*(n-i)/n
    prob = prob * (x-1)/n
    return prob

#Функция для поиска матожидания
def E(n):
    E_n = 0
    if n <= 1:
        return E_n
    for x in range(n+2):
        E_n += prob(x,n)*x
    return E_n

#Найдем матожидания от разных n
Es = []
for n in range(1,1001):
    Es.append(E(n))

#Построим график матожидания от n
ns=range(1,1001)
plt.figure(figsize=(15, 8), dpi = 300)
plt.plot(ns,Es)
plt.title('Зависимость матожидания от n')
plt.xlabel('n')
plt.ylabel('E');
```



Методом моментов мы предполагаем, что наше реальное матожидание равно 10

In [274…
```python
#Найдем n, при котором матожидание максимально близко к 10.
ns=range(1,1001)
print('Оценка n методом моментов =', ns[np.argmin(abs(np.array(Es)-10))])
```

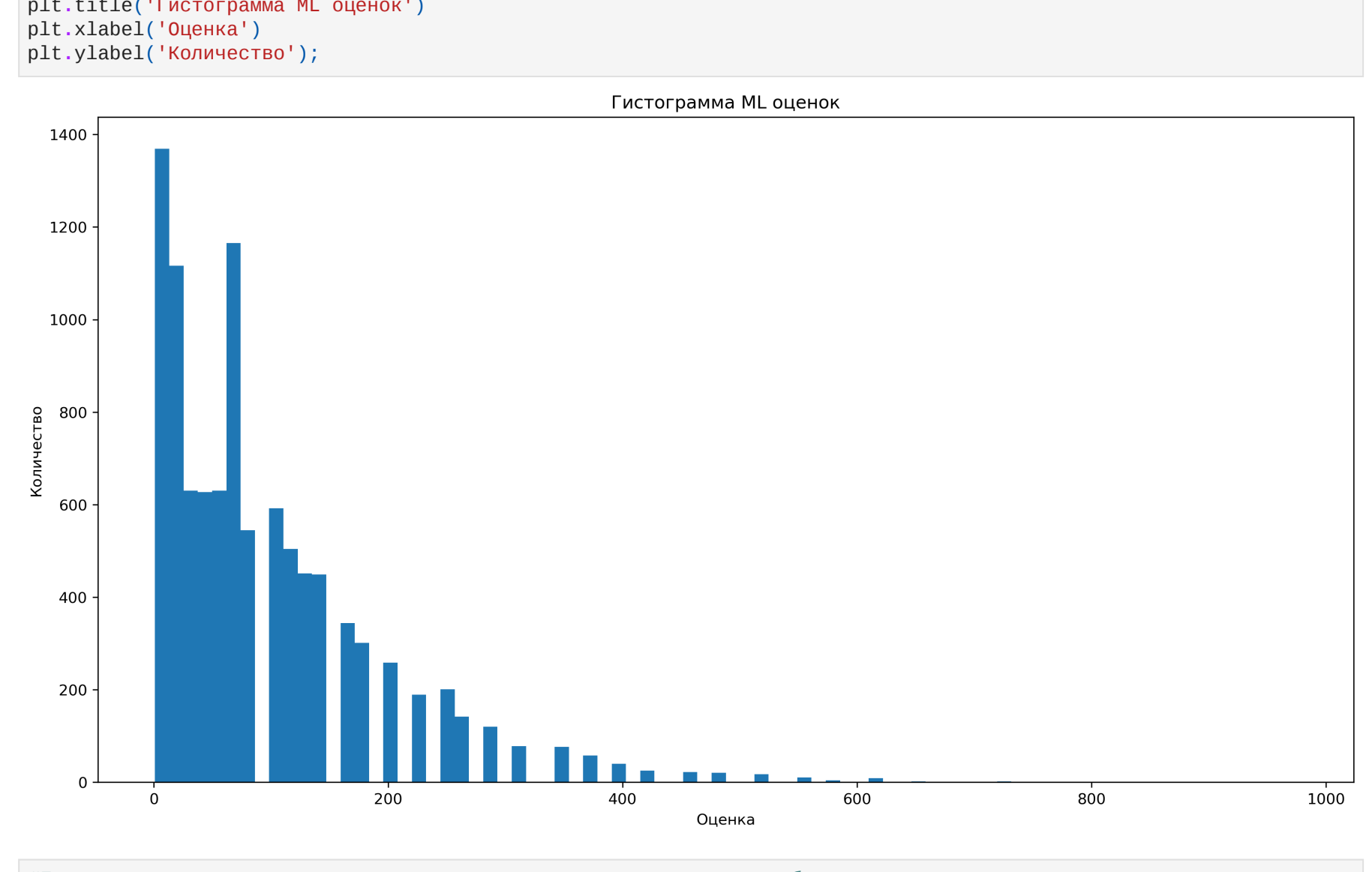
```
Оценка n методом моментов = 55
```

в)

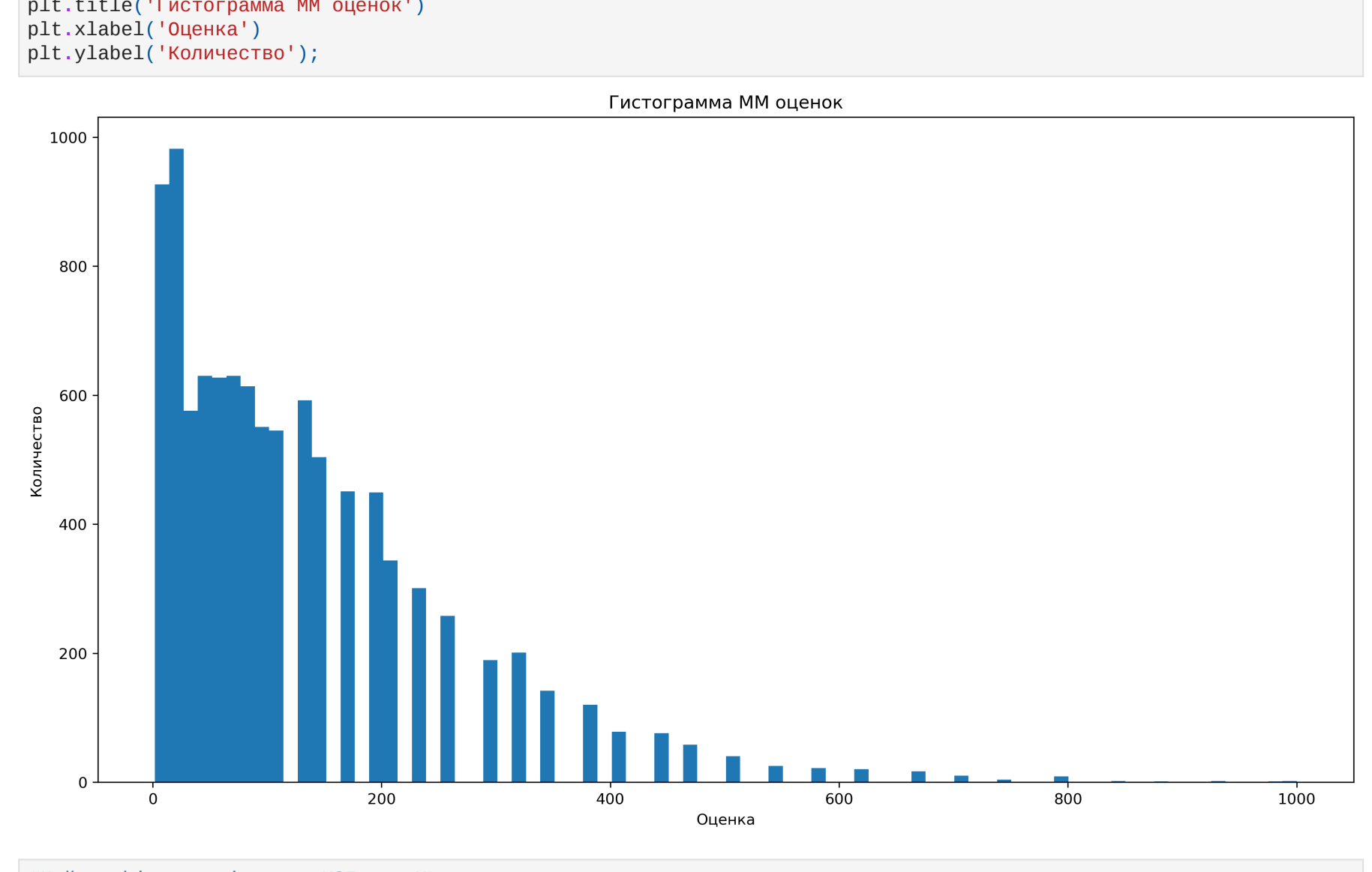
In [457…
```python
#сгенерируем номера дней, когда приехал тот же таксист
days = []
for i in range(10000):
    l = []
    day = 1
    taxi = np.random.choice(range(1,101))
    while taxi not in l:
        l.append(taxi)
        day +=1
        taxi = np.random.choice(range(1,101))
    days.append(day)
```

In [458…
```python
#Векторизуем функцию максимального правдоподобия
ML_vec = np.vectorize(ML, excluded = set([0]))

#Запишем функцию для поиска оценки методом максимального правдоподобия
def find_ML(x):
    ns=range(x-1, 1001)
    Ls = []
    for n in range(x-1,1001):
        Ls.append(ML(n,x))
    best_n = ns[np.argmax(Ls)]
    return best_n

#Векторизуем
find_ML_vec = np.vectorize(find_ML)

#Найдем оценки методом максимального правдоподобия для каждой симуляции
ML_estimates = find_ML_vec(days)

Es = []
for n in range(1,1001):
    Es.append(E(n))

#Запишем функцию для поиска оценки методом моментов
def find_MM(x):
    ns=range(1, 1001)
    return ns[np.argmin(abs(np.array(Es)-x))]

#Векторизуем ее
find_MM_vec = np.vectorize(find_MM)

#Найдем оценки методом моментов правдоподобия для каждой симуляции
MM_estimates = find_MM_vec(days)
```

In [459…
```python
#Построим гистограмму для оценок методом максимального правдоподобия
plt.figure(figsize=(15, 8), dpi = 300)
plt.hist(ML_estimates, bins=80)
plt.title('Гистограмма ML оценок')
plt.xlabel('Оценка')
plt.ylabel('Количество');
```



In [460…
```python
#Построим гистограмму для оценок методом максимального правдоподобия
plt.figure(figsize=(15, 8), dpi = 300)
plt.hist(MM_estimates, bins=80)
plt.title('Гистограмма MM оценок')
plt.xlabel('Оценка')
plt.ylabel('Количество');
```



In [461…
```python
#Найдем bias, variance и MSE для ML оценок
bias = mean_absolute_error([100]*10000, ML_estimates)
variance = np.var(ML_estimates)
mse = mean_squared_error([100]*10000, ML_estimates)
print('Для ML оценок:')
print('Среднее =', np.mean(ML_estimates))
print('Смещение =', bias)
print('Дисперсия =', variance)
print('Среднеквадратичная ошибка =', mse)
print('')

#Найдем bias, variance и MSE для MM оценок
bias = mean_absolute_error([100]*10000, MM_estimates)
variance = np.var(MM_estimates)
mse = mean_squared_error([100]*10000, MM_estimates)
print('Для MM оценок:')
print('Среднее =', np.mean(MM_estimates))
print('Смещение =', bias)
print('Дисперсия =', variance)
print('Среднеквадратичная ошибка =', mse)
```

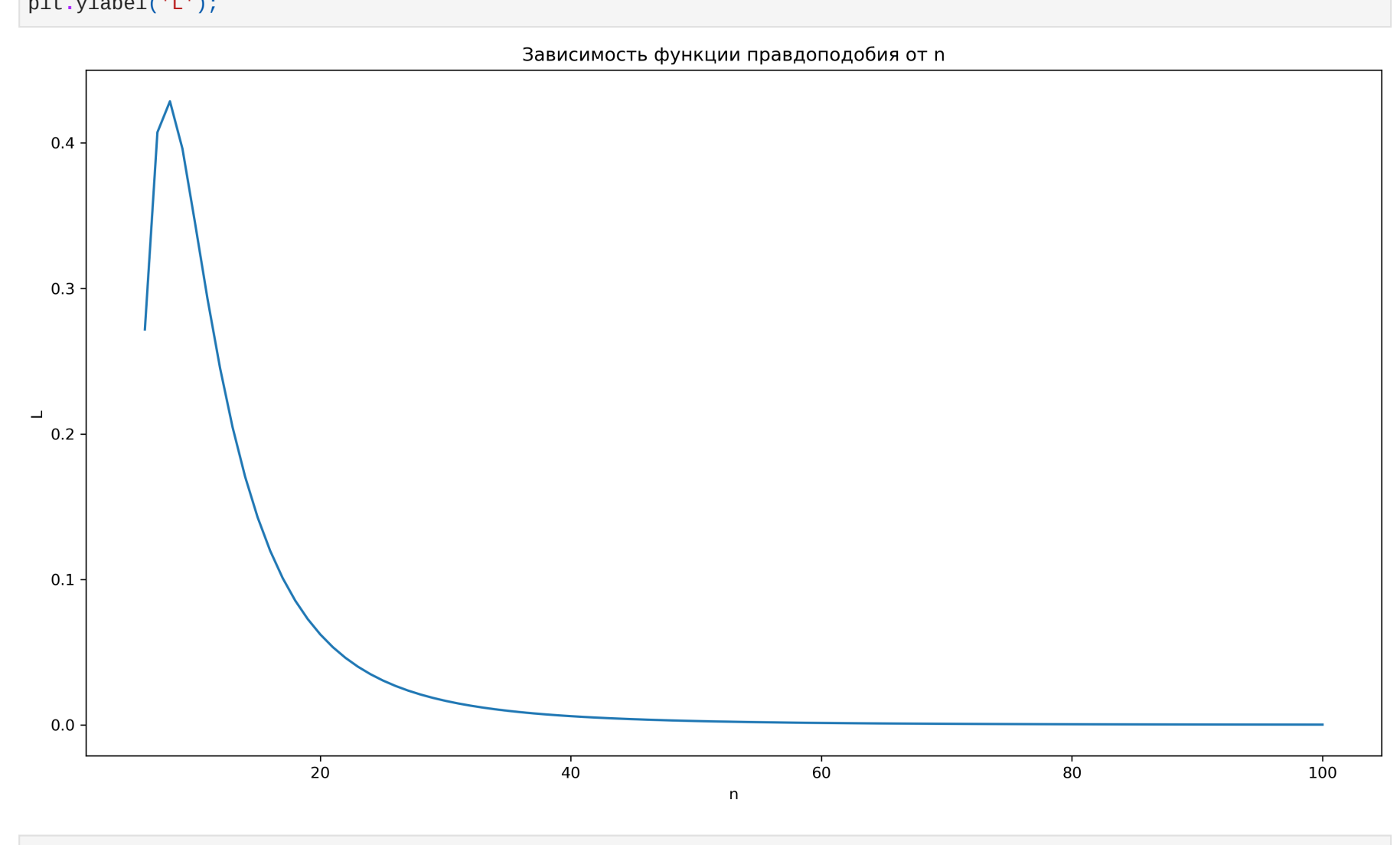
```
Для ML оценок:
Среднее = 95.7254
Смещение = 69.3406
Дисперсия = 8546.852994840001
Среднеквадратичная ошибка = 8565.1252

Для MM оценок:
Среднее = 124.8462
Смещение = 84.2032
Дисперсия = 14090.255145560002
Среднеквадратичная ошибка = 14707.5888
```

Задача 2

а)

Спустя бесчисленное количество попыток удалось вывести функцию максимального правдоподобия с помощью построения деревьев

In [399…
```python
#Запишем функцию максимального правдоподобия, где u - количество уникальных имен, d - число дней
#нам надо будет считать произведения сочетаний, сделаем для этого функцию
def proizved(tuple):
    proizv = 1
    for i in tuple:
        proizv *= i
    return proizv

def L(n,u,d):
    L = 1
    for i in range(1, u):
        L = L * (n-i)/n
    combinations = itertools.combinations_with_replacement(range(1,u+1), r=d-u)
    summa = np.sum([proizved(t) for t in list(combinations)])
    L = L * summa/(n**(d-u))
    return L

#Найдем функцию правдоподобия для разных n
Ls = []
ns = range(6,101)
for n in ns:
    Ls.append(L(n,6,10))

#Построим график
plt.figure(figsize=(15, 8), dpi = 300)
plt.plot(ns, Ls)
plt.title('Зависимость функции правдоподобия от n')
plt.xlabel('n')
plt.ylabel('L');
```



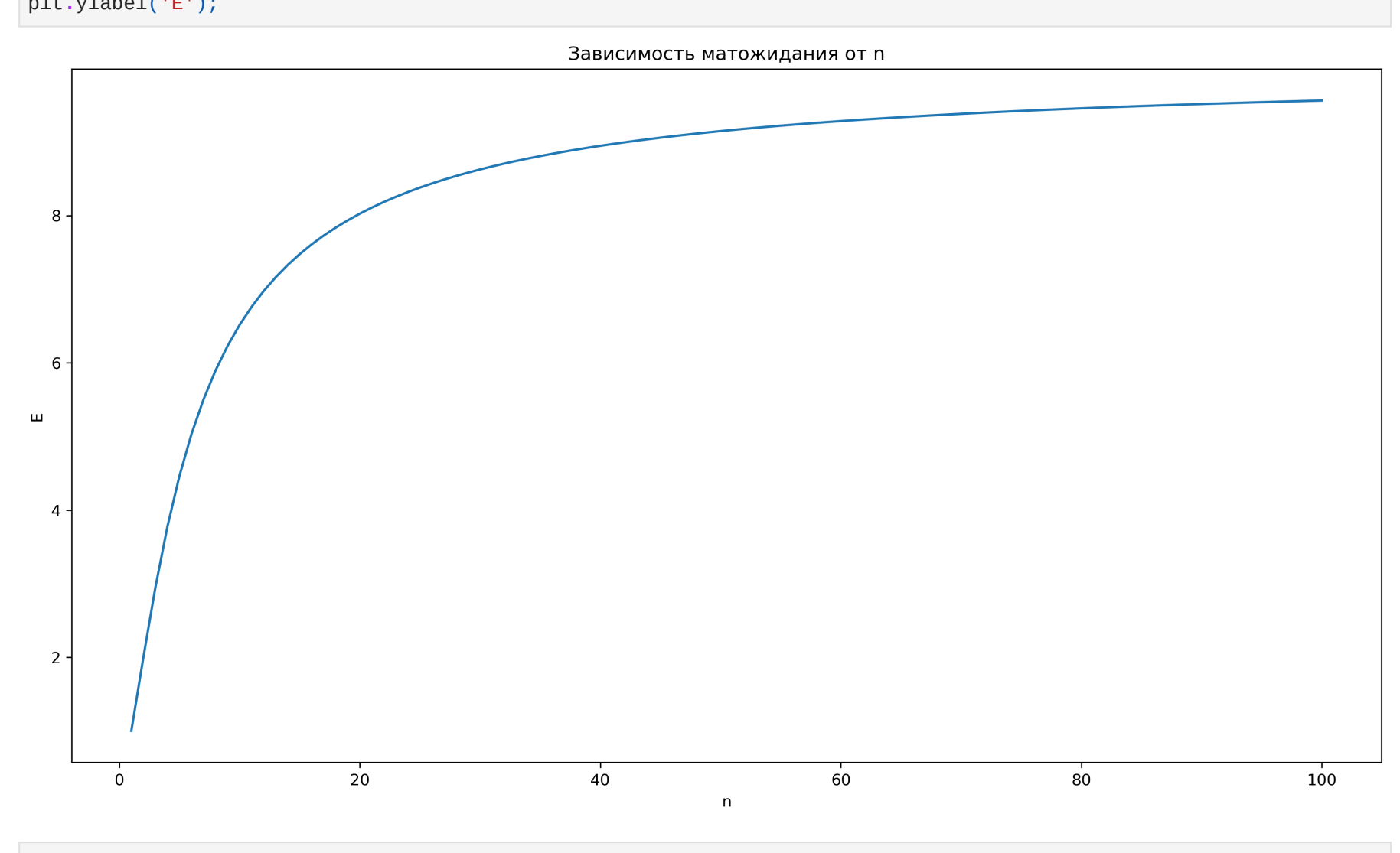
In [418…
```python
#Запишем функцию для поиска оценки методом максимального правдоподобия
def find_ML(u,d):
    ns=range(u, 101)
    Ls = []
    for n in ns:
        Ls.append(L(n,u,d))
    best_n = ns[np.argmax(Ls)]
    return best_n

#Найдем ML оценку
print('Оценка n методом максимального правдоподобия =', find_ML(6,10))
```

```
Оценка n методом максимального правдоподобия = 8
```

б)

In [401…
```python
#Запишем функцию для матожидания
def E(n,d):
    E_n = 0
    for u in range(d+1):
        E_n += L(n,u,d)*u
    return E_n
```

In [402…
```python
Es= []
ns=range(1,101)
for n in ns:
    Es.append(E(n,10))

#Построим график матожидания от n
plt.figure(figsize=(15, 8), dpi = 300)
plt.plot(ns,Es)
plt.title('Зависимость матожидания от n')
plt.xlabel('n')
plt.ylabel('E');
```



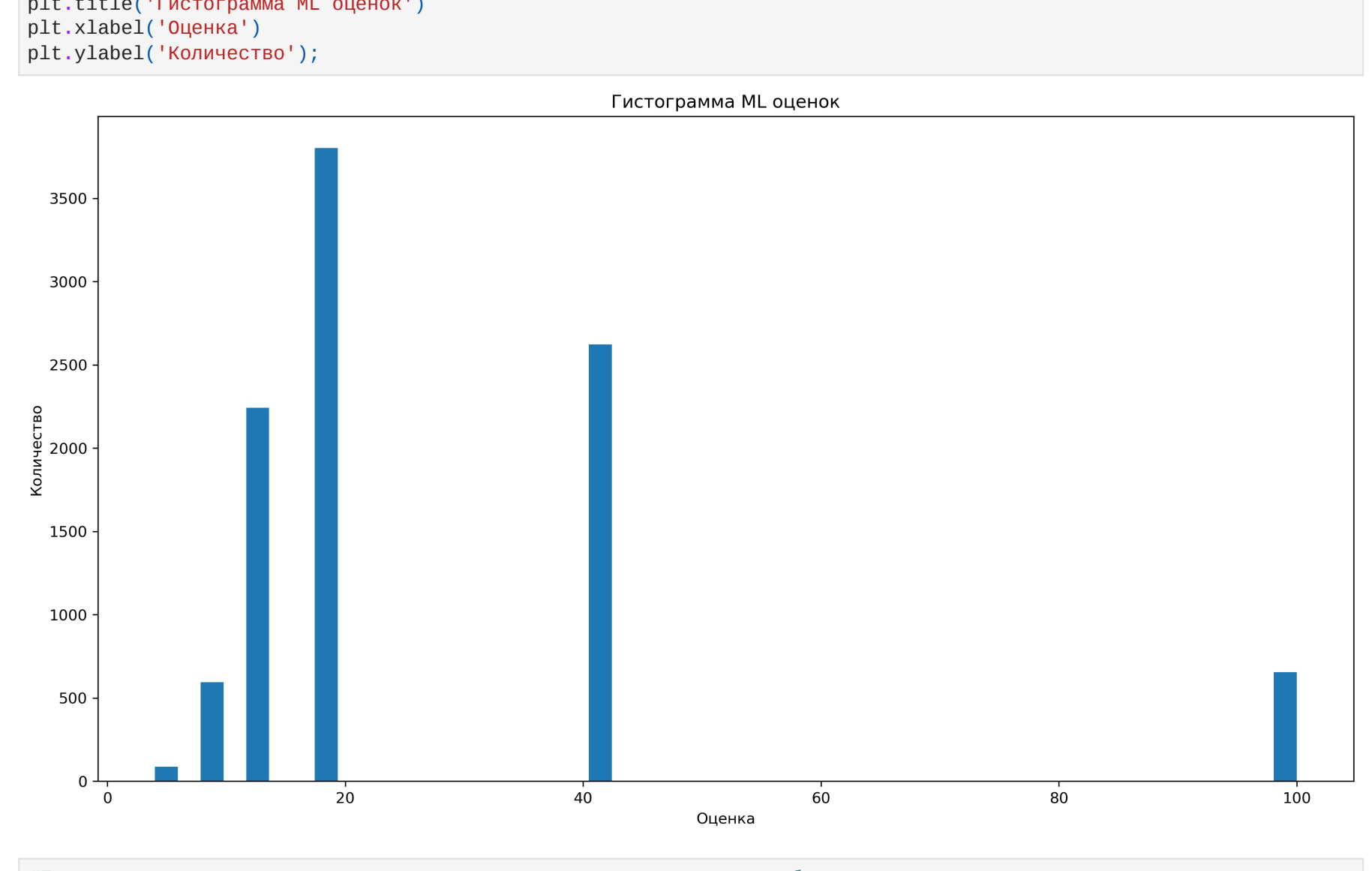
In [448…
```python
#Запишем функцию для поиска оценки методом моментов
def find_MM(x):
    ns=range(1, 101)
    return ns[np.argmin(abs(np.array(Es)-x))]

#Найдем оценку методом моментов
print('Оценка n методом моментов =', find_MM(6))
```

```
Оценка n методом моментов = 8
```

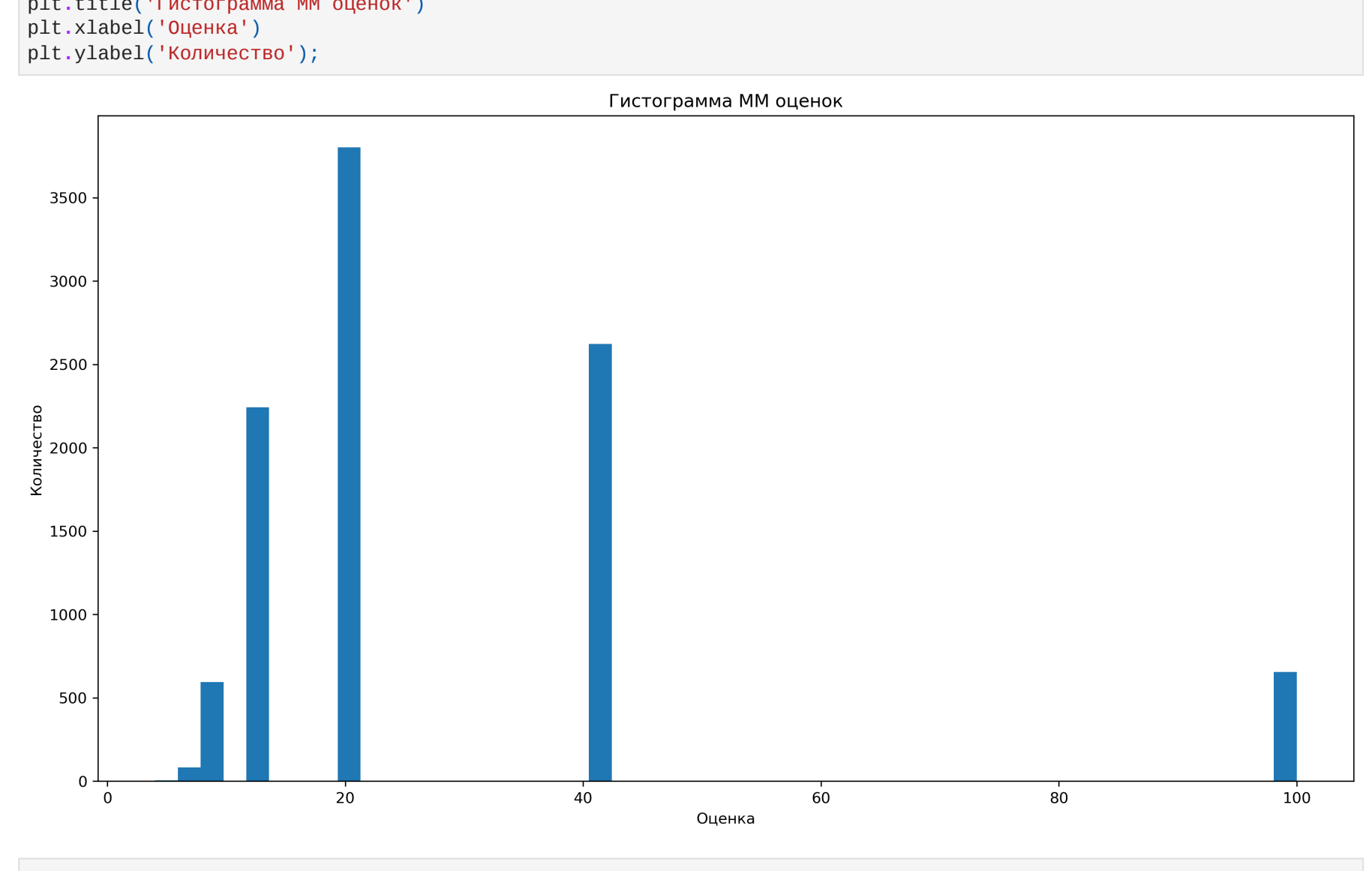
в)

In [449…
```python
#Нагенерируем выборку
samples = []
np.random.seed(3)
taxi = np.random.choice(range(1,21), size=(10000,10))
for i in taxi:
    samples.append(len(set(i)))

#Посчитаем матожидания
Es= []
ns=range(1,101)
for n in ns:
    Es.append(E(n,10))

#Векторизуем функции ML и MM
find_ML_vec = np.vectorize(find_ML, excluded = set([1]))
find_MM_vec = np.vectorize(find_MM)

#Посчитаем оценки
ML_estimates = find_ML_vec(samples, 10)
MM_estimates = find_MM_vec(samples)
```

In [450…
```python
#Построим гистограмму для оценок методом максимального правдоподобия
plt.figure(figsize=(15, 8), dpi = 300)
plt.hist(ML_estimates, bins=50)
plt.title('Гистограмма ML оценок')
plt.xlabel('Оценка')
plt.ylabel('Количество');
```



In [451…
```python
#Построим гистограмму для оценок методом максимального правдоподобия
plt.figure(figsize=(15, 8), dpi = 300)
plt.hist(MM_estimates, bins=50)
plt.title('Гистограмма MM оценок')
plt.xlabel('Оценка')
plt.ylabel('Количество');
```



In [453…
```python
#Найдем bias, variance и MSE для ML оценок
bias = mean_absolute_error([20]*10000, ML_estimates)
variance = np.var(ML_estimates)
mse = mean_squared_error([20]*10000, ML_estimates)
print('Для ML оценок:')
print('Среднее =', np.mean(ML_estimates))
print('Смещение =', bias)
print('Дисперсия =', variance)
print('Среднеквадратичная ошибка =', mse)
print('')

#Найдем bias, variance и MSE для MM оценок
bias = mean_absolute_error([20]*10000, MM_estimates)
variance = np.var(MM_estimates)
mse = mean_squared_error([20]*10000, MM_estimates)
print('Для MM оценок:')
print('Среднее =', np.mean(MM_estimates))
print('Смещение =', bias)
print('Дисперсия =', variance)
print('Среднеквадратичная ошибка =', mse)
```

```
Для ML оценок:
Среднее = 27.983
Смещение = 14.0178
Дисперсия = 506.9889109999999
Среднеквадратичная ошибка = 570.7172

Для MM оценок:
Среднее = 28.3714
Смещение = 13.6294
Дисперсия = 500.01606204
Среднеквадратичная ошибка = 570.0964
```

Задание 5

In [587…
```python
more_tha_median = marks > np.median(marks)
df = pd.DataFrame({'surnames':surnames, 'marks': marks, 'start_glasn': start_with_glasn_vec(surnames), 'more_tha_m
```

In [592…
```python
#Сделаем таблицу сопряженности
more_tha_median = marks > np.median(marks)
df = pd.DataFrame({'surnames':surnames, 'marks': marks, 'start_glasn': start_with_glasn_vec(surnames), 'more_tha_m
table = pd.crosstab(df['more_tha_median'], df['start_glasn'])
print(table)
```

```
start_glasn      False  True
more_tha_median
False              138    28
True               145    21
```

In [613…
```python
#Посчитаем odds и OR
odds_hor_glasn = table.iloc[1,1]/table.iloc[0,1]
odds_hor_soglasn = table.iloc[1,0]/table.iloc[0,0]
OR = odds_hor_glasn / odds_hor_soglasn
ln_OR = np.log(OR)
se_lnOR = np.sqrt(1/table.iloc[0,0] + 1/table.iloc[0,1] + 1/table.iloc[1,0] + 1/table.iloc[1,1])

#Запишем интервал
interval_ln_OR = [ln_OR-1.96*se_lnOR,ln_OR+1.96*se_lnOR]
interval_OR = [np.exp(ln_OR - 1.96*se_lnOR), np.exp(ln_OR + 1.96*se_lnOR)]

#Найдем p-value
z_obs = (ln_OR - np.log(1)) / se_lnOR
p_value = 2*(1 - sts.norm.cdf(abs(z_obs)))
print('P-value =', p_value)

if interval_OR[0] <= 1 <= interval_OR[1]:
    print('H0 не отвергается на уровне значимости 0.05')
else:
    print('H0 отвергается на уровне значимости 0.05')
```

```
P-value = 0.280180274566451
H0 не отвергается на уровне значимости 0.05
```

б)

Для удобства будем рассматривать логарифм отношения вероятностей.

$$\ln\frac{p_{glasn}}{p_{soglasn}} = \ln p_{glasn} - \ln p_{soglasn}$$

Используем дельта-метод:

$$\ln\hat{p} \approx \ln p - \frac{1}{p}(\hat{p} - p)$$

$$E(\ln\frac{\hat{p}_{glasn}}{\hat{p}_{soglasn}}) \approx \ln p_{glasn} - \ln p_{soglasn}$$

$$Var(\ln\frac{\hat{p}_{glasn}}{\hat{p}_{soglasn}}) \approx \frac{1 - p_{glasn}}{p_{glasn}n_{glasn}} + \frac{1 - p_{soglasn}}{p_{soglasn}n_{soglasn}}$$

In [631…
```python
#Запишем вероятность
p_soglasn = np.mean(more_tha_median[start_with_glasn_vec(surnames) == False])
p_glasn = np.mean(more_tha_median[start_with_glasn_vec(surnames)])

#Посчитаем необходимые значения для построения асимптотического интервала
ln_p_soglasn = np.log(p_soglasn)
ln_p_glasn = np.log(p_glasn)
n_soglasn = len(soglasn_marks)
n_glasn = len(glasn_marks)
se_ln = np.sqrt((1-p_soglasn)/(n_soglasn*p_soglasn) + ((1-p_glasn)/(n_glasn*p_glasn)))

#Посчитаем интервал
interval_ln = [(ln_p_glasn-ln_p_soglasn)-1.96*se_ln, (ln_p_glasn-ln_p_soglasn)+1.96*se_ln,]
interval = [np.exp(interval_ln[0]), np.exp(interval_ln[1])]

#Посчитаем  p-value
z_obs = ((ln_p_glasn-ln_p_soglasn) - np.log(1)) / se_ln
p_value = 2*(1 - sts.norm.cdf(abs(z_obs)))
print('P-value =', p_value)

#Проверим, входит ли в него 1
if interval[0] <= 1 <= interval[1]:
    print('H0 не отвергается на уровне значимости 0.05')
else:
    print('H0 отвергается на уровне значимости 0.05')
```

```
P-value = 0.3070947928050547
H0 не отвергается на уровне значимости 0.05
```

Задание 7

Вот диалог

Задание 8

Самыми полезными для меня оказались 2 ресурса:\ Помогал для понимания абстрактных понятий - канал 3Blue1Brown \ Помогал для закрепления материалов лекций и семинаров - канал Прикладная статистика