

# ДЗ матстат

```
In [151]: import numpy as np
from scipy.stats import expon
import math
from scipy.stats import bootstrap
from scipy.stats import norm
from scipy.stats import t
import pandas as pd
from scipy import stats
import random
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
from statistics import mean
from statistics import variance
from tqdm import tqdm
```

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.

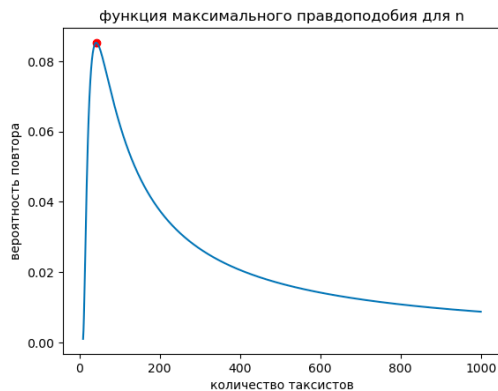
Функция правдоподобия в данной задаче:

$$L = P(X = 10) = \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \dots \cdot \frac{n-8}{n} \cdot \frac{9}{n} = \frac{9(n-1) \dots (n-8)}{n^9}$$

```
In [2]: def likelihood_func():
    res = []
    n = 1000
    for i in range(9, n+1):
        p = 9*(i-1)*(i-2)*(i-3)*(i-4)*(i-5)*(i-6)*(i-7)*(i-8)/(i**9)
        res.append(p)
    return res
```

```
In [3]: p_ml = max(likelihood_func())
n_ml = likelihood_func().index(p_ml) + 9
print('оценка числа n методом максимального правдоподобия:', n_ml)
print('вероятность для оценки:', p_ml)
plt.plot([j for j in range(9, 1001)], likelihood_func())
plt.scatter(n_ml, p_ml, color='red')
plt.title('функция максимального правдоподобия для n')
plt.xlabel('количество таксистов')
plt.ylabel('вероятность повтора');
```

оценка числа  $n$  методом максимального правдоподобия: 42  
вероятность для оценки: 0.08525937285627631



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.

```
In [4]: def taxi_prob(n, x):
    prob = 1
    for i in range(0, x-1):
        prob *= (n-i)
    prob = prob * (x-1) / (n**x)
    return prob #вероятность при n таксистах получить первый повтор на x день
```

```
In [5]: def exp_taxi(n):
    exp_taxi = 0
    for j in range(2, 100):
        exp_taxi += (j * taxi_prob(n, j))
    return exp_taxi # функция математического ожидания
```

```
In [6]: expectations = []
for n in range(1, 100):
    expectations.append(exp_taxi(n))
```

```

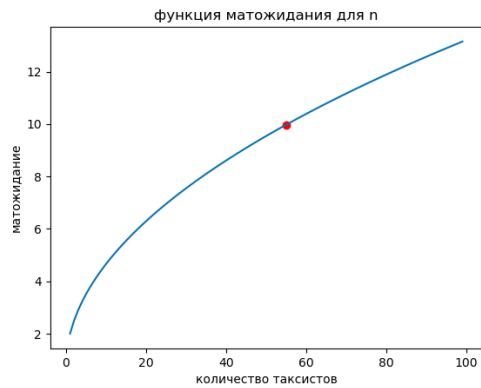
n_mm = [n for n in range(1,100)][expectations.index(min(expectations, key=lambda x: abs(x - 10)))]
exp_mm = min(expectations, key=lambda x: abs(x - 10))

print('оценка числа n методом моментов:', n_mm)

plt.plot([i for i in range(1,100)], expectations)
plt.scatter(n_mm, exp_mm, color='red')
plt.title('функция матожидания для n')
plt.xlabel("количество таксистов")
plt.ylabel("матожидание");

```

оценка числа n методом моментов: 55



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

```
In [39]: n_sim = 10**4
```

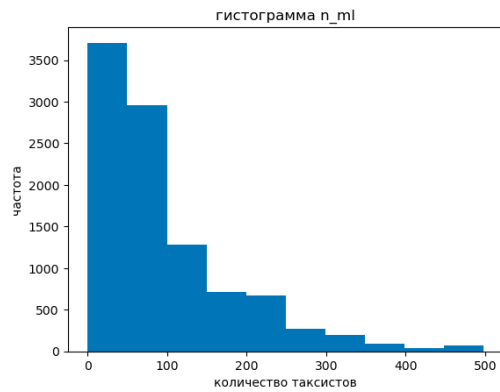
```
In [52]: taxis = np.arange(1,102) #т.к. настоящее n=100, то на 101 такси мы точно получим повторение
```

```
In [40]: def l_func(x, n):#значение ф-ции правдоподобия, что на x заказ придет повторное такси, если всего таксистов n
l_fun = 1
for i in range(x-1):
    l_fun *= ((n - i) / n)
return l_fun*(x-1)/n
```

```
In [61]: def func(x):
res = []
for n in range(1, 500):
    res.append(l_func(x, n))
return res
```

```
In [63]: np.random.seed(16)
ml = []
for simulation in range(n_sim):
    previous_orders = []
    for i in taxis:
        taxi = random.choice(taxis)
        if taxi in previous_orders:
            ml.append(func(i).index(max(func(i))))
            break
        else:
            previous_orders.append(taxi)

plt.hist(ml)
plt.title('гистограмма n_ml')
plt.xlabel("количество таксистов")
plt.ylabel("частота");
```



```
In [68]: print('Оценка смещения ml-оценки:', round(mean(ml) - 100, 3))
```

Оценка смещения ml-оценки: -5.126

```
In [71]: print('Оценка дисперсии ml-оценки:', round(variance(ml), 3))
```

Оценка дисперсии ml-оценки: 8138.732

```
In [78]: s = 0
for i in ml:
    s += ((i - 100) ** 2) / 10000
print('Среднеквадратичная ошибка ml-оценки:', round(s, 3))
```

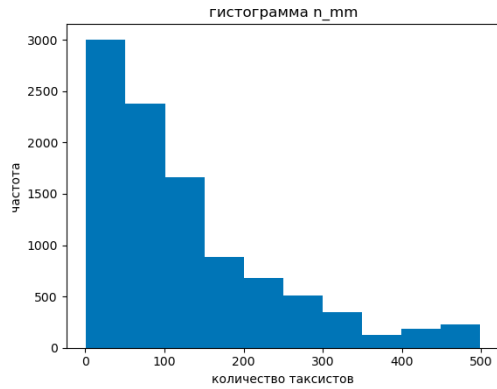
Среднеквадратичная ошибка ml-оценки: 8164.19

```
In [79]: def taxi_prob(n, x):
    prob = 1
    for i in range(0, x-1):
        prob *= (n-i)
    prob = prob * (x-1) / (n**x)
    return prob #вероятность при n таксистах получить первый повтор на x день
```

```
In [91]: np.random.seed(16)
mm = []
num_repeat = []
for simulation in range(n_sim):
    previous_orders = []
    for i in taxis:
        taxi = random.choice(taxis)
        if taxi in previous_orders:
            num_repeat.append(i)
            break
        else:
            previous_orders.append(taxi)
ns = [n for n in range(1, 500)]
expectations = []
for n in range(1, 500):
    s = 0
    for x in range(2, 500):
        s += x * taxi_prob(n, x)
    expectations.append(s)

mm = []
for taxi in num_repeat:
    mm.append(ns[expectations.index(min(expectations, key=lambda x: abs(x - taxi)))]))

plt.hist(mm)
plt.title('гистограмма n_mm')
plt.xlabel("количество таксистов")
plt.ylabel("частота");
```



```
In [92]: print('Оценка смещения  $\mu$ -оценки:', round(mean(mm) - 100, 3))
print('Оценка дисперсии  $\mu$ -оценки:', round(variance(mm), 3))
s = 0
for i in mm:
    s += ((i - 100) ** 2) / 10000
print('Среднеквадратичная ошибка  $\mu$ -оценки:', round(s, 3))
```

Оценка смещения  $\mu$ -оценки: 27.206  
 Оценка дисперсии  $\mu$ -оценки: 12862.779  
 Среднеквадратичная ошибка  $\mu$ -оценки: 113.61.67

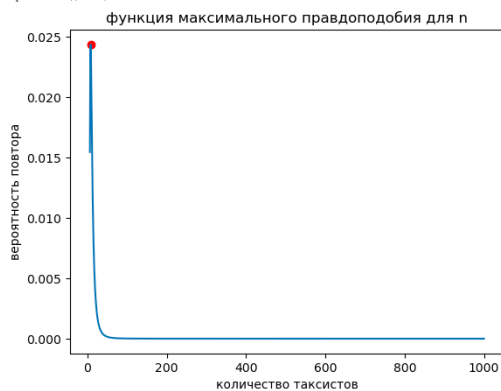
2. Однажды в Самарканде турист заказывал Яндекс-такси. На десятом заказе он обнаружил, что у таксистов было 6 разных имён. Для упрощения предположим, что все  $n$  имён среди таксистов встречаются равновероятно и независимо от поездки к поездке.

Update 2023-06-07: если по выборке в симуляциях оценка метода моментов или метода максимального правдоподобия стремится к бесконечности и, строго говоря, не существует, то можно ограничить её сверху большим числом, например, 100.

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

```
In [103]: def likelihood_func():
    res = []
    n = 1000
    for i in range(6, n+1):
        p = (1*(i-1)*(i-2)*(i-3)*(i-4)*(i-5)/(i**5))*(6**4)/(i**4)
        res.append(p)
    return res
p_ml = max(likelihood_func())
n_ml = likelihood_func().index(p_ml) + 6
print('оценка числа n методом максимального правдоподобия:', n_ml)
print('вероятность для оценки:', p_ml)
plt.plot([j for j in range(6, 1001)], likelihood_func())
plt.scatter(n_ml, p_ml, color='red')
plt.title('функция максимального правдоподобия для n')
plt.xlabel('количество таксистов')
plt.ylabel('вероятность повтора');
```

оценка числа  $n$  методом максимального правдоподобия: 8  
 вероятность для оценки: 0.02433300018310547



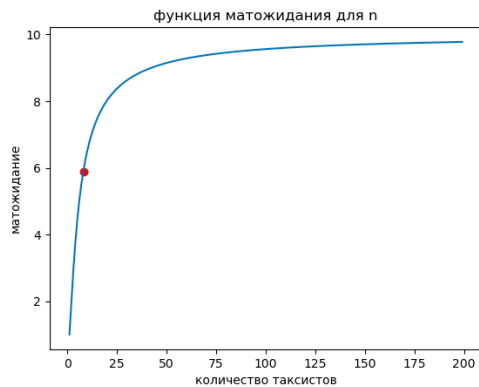
б) [5] Постройте график математического ожидания числа разных имён у 10 таксистов, как функции от общего количества имён n. Найдите оценку числа n методом моментов.

```
In [106]: def expect(n):
            return (1-(1-1/n)**10)*n
            expectations = []
            for i in range(1, 200):
                expectations.append(expect(i))
            n_mm = [n for n in range(1,200)][expectations.index(min(expectations, key=lambda x: abs(x - 6)))]
            exp_mm = min(expectations, key=lambda x: abs(x - 6))

            print('оценка числа n методом моментов:', n_mm)

            plt.plot([i for i in range(1,200)], expectations)
            plt.scatter(n_mm, exp_mm, color='red')
            plt.title('функция матожидания для n')
            plt.xlabel('количество таксистов')
            plt.ylabel('матожидание');
```

оценка числа n методом моментов: 8



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

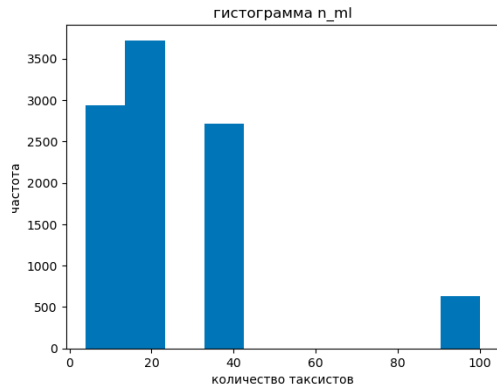
```
In [139]: names = np.arange(1, 21)
            np.random.seed(16)
            samples = np.random.choice(names, size = (10000, 10))

            def l_f(x, n):
                pr = 1 / n ** (10 - x)
                for i in range(1, x + 1):
                    pr *= (n - i + 1) / n
                return pr

            ml = []

            for s in samples:
                obs = np.unique(s).shape[0]
                if obs == 10:
                    ml.append(100)
                    continue
                m = -1
                a = -1
                for n in range(obs, 1000):
                    L = l_f(obs, n)
                    if L > m:
                        m = L
                        a = n
                ml.append(a)
```

```
In [140]: plt.hist(ml)
            plt.title('гистограмма n_ml')
            plt.xlabel('количество таксистов')
            plt.ylabel('частота');
```

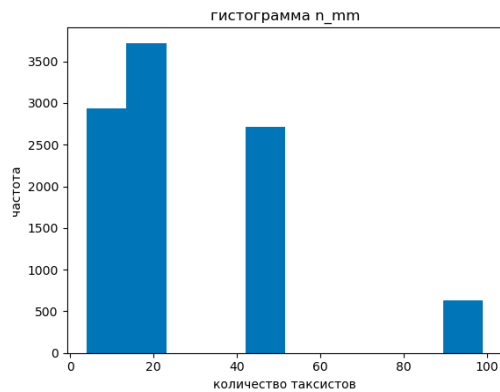


```
In [141]: print('Оценка смещения ml-оценки:', round(mean(ml) - 20, 3))
print('Оценка дисперсии ml-оценки:', round(variance(ml), 3))
s = 0
for i in ml:
    s += ((i - 20) ** 2) / 10000
print('Среднеквадратичная ошибка ml-оценки:', round(s, 3))
```

Оценка смещения ml-оценки: 7.99  
 Оценка дисперсии ml-оценки: 496.445  
 Среднеквадратичная ошибка ml-оценки: 560.234

```
In [142]: np.random.seed(16)
tx=[]
for i in range(10**4):
    ts = np.random.choice(names, 10, True)
    tx.append(len(set(ts)))
mm=[]
expectations=[]
for n in ns:
    expectations.append((1-(1-1/n)**10)*n)
for i in tx:
    n_mm = [n for n in range(1,100)][expectations.index(min(expectations, key=lambda x: abs(x - i)))]
    mm.append(n_mm)

plt.hist(mm)
plt.title('гистограмма n_mm')
plt.xlabel("количество таксистов")
plt.ylabel("частота");
```



```
In [143]: print('Оценка смещения  $\mu$ -оценки', round(mean(mm) - 20, 3))
print('Оценка дисперсии  $\mu$ -оценки:', round(variance(mm), 3))
s = 0
for i in mm:
    s += ((i - 20) ** 2) / 10000
print('Среднеквадратичная ошибка  $\mu$ -оценки:', round(s, 3))
```

Оценка смещения  $\mu$ -оценки: 8.308  
 Оценка дисперсии  $\mu$ -оценки: 480.609  
 Среднеквадратичная ошибка  $\mu$ -оценки: 549.584

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

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

```
In [145]: E = 1
std = 1
z_crit = norm.ppf(0.975)
n_sim = 10**4
```

```
In [147]: np.random.seed(16)
k = 0
for i in range(n_sim):
    rvs = expon.rvs(size = 20, scale = 1)
    mean = rvs.mean()
    if (mean - z_crit*std/math.sqrt(20)) < E and (mean + z_crit*std/math.sqrt(20)) > E:
        k += 1
k/n_sim
```

Out[147]: 0.9519

```
In [149]: np.random.seed(16)
k = 0
for i in range(n_sim):
    rvs = expon.rvs(size = 20, scale = 1)
    values = []
    for i in range(1000):
        values.append(np.random.choice(rvs, 20, True).mean())
    q_l = np.quantile(values, 0.025)
    q_r = np.quantile(values, 0.975)
    if q_l < E and q_r > E:
        k += 1
k/n_sim
```

Out[149]: 0.8987

```
In [152]: np.random.seed(16)
k = 0
for i in tqdm(range(n_sim)):
    mean = expon.rvs(size = 20, scale = 1).mean()
    se = expon.rvs(size = 20, scale = 1).std() / math.sqrt(20)
    values = []
    for i in range(1000):
        new_rvs = np.random.choice(expon.rvs(size = 20, scale = 1), 20, True)
        se_new = new_rvs.std() / math.sqrt(20)
        new_mean = new_rvs.mean()
        R = (new_mean - mean) / se_new
        values.append(R)
    q_l = np.quantile(values, 0.025)
    q_r = np.quantile(values, 0.975)
    if mean - q_l * se > E and mean - q_r * se < E:
        k += 1
k/10000
```

100% |████████████████████| 10000/10000 [30:56<00:00, 5.39it/s]

Out[152]: 0.8326

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

```
In [160]: np.random.seed(16)
k = 0
for i in tqdm(range(n_sim)):
    mean = t.rvs(size = 20, df = 3).mean()
    if (mean - z_crit*3/math.sqrt(20)) < 0 and (mean + z_crit*3/math.sqrt(20)) > 0:
        k += 1
k/n_sim
```

100% |████████████████████| 10000/10000 [00:02<00:00, 3506.01it/s]

Out[160]: 0.995

```
In [162]: np.random.seed(16)
k = 0
for i in tqdm(range(n_sim)):
    values = []
    for i in range(1000):
        values.append(np.random.choice(t.rvs(size = 20, df = 3), 20, True).mean())
    if np.quantile(values, 0.025) < 0 and np.quantile(values, 0.975) > 0:
        k += 1
k/n_sim
```

100% |████████████████████| 10000/10000 [02:59<00:00, 55.77it/s]

Out[162]: 0.9183

```

In [163]: np.random.seed(16)
good = 0
for i in tqdm(range(n_sim)):
    mean = t.rvs(size = 20, df = 3).mean()
    se = t.rvs(size = 20, df = 3).std()/math.sqrt(20)
    values = []
    for i in range(1000):
        new_rvs = np.random.choice(t.rvs(size = 20, df = 3), 20, True)
        se_new = new_rvs.std()/math.sqrt(20)
        new_mean = new_rvs.mean()
        R = (new_mean - mean)/se_new
        values.append(R)
    if mean - np.quantile(values, 0.025)*se>0 and mean - np.quantile(values, 0.975) * se<0:
        k+=1

k/n_sim

100% |████████████████████| 10000/10000 [06:03<00:00, 27.54it/s]
Out[163]: 0.9259

```

в) [5] Какой способ оказался лучше?

Эффективнее всего оказался доверительный интервал

В обоих случаях асимптотический интервал показал наилучший результат. Возможно, дело в том, что надо делать больше бутстрапированных выборок, но, честно говоря, это очень долго!

Задача 4

пункт а

```

In [153]: data = pd.read_csv('оценки.csv')
data

```

```

Out[153]:
   Фамилия  Балл
0  Репенкова    16
1  Ролдугина     0
2   Сафина    19
3  Сидоров    26
4  Солоухин    21
...      ...    ...
327  Сенников    19

```

```

...      ...    ...
327  Сенников    19
328    Ся         0
329  Сятова      0
330  Темиркулов  0
331    Эшмеев    16

```

332 rows x 2 columns

```

In [158]: def vowel(word):
    if word[0].lower() in ["a", "o", "y", "ы", "и", "е", "ё", "ю", "я", "э"]:
        return 1
    else:
        return 0
data['первая гласная'] = data['Фамилия'].apply(vowel)

```

```

In [159]: glasnie = data[data['первая гласная']==1]['Балл'].values
soglasnie = data[data['первая гласная']==0]['Балл'].values
p_value = stats.ttest_ind(glasnie, soglasnie, equal_var = False)[1]
if p_value > 0.05:
    print('Гипотеза не отвергается')
else:
    print('Гипотеза отвергается')

```

Гипотеза не отвергается

## задача 7

я выбрала задачу 3.14 из листочка по терверу

<https://chat.openai.com/share/c677981f-a5c6-440f-8793-288d80155049> - вроде бы это ссылка на сам диалог





У Ивана Грозного  $n$  бояр. Каждый боярин берёт мзду независимо от других с вероятностью  $1/2$ .

а) Какова вероятность того, что все бояре берут мзду, если случайно выбранный боярин берёт мзду?

б) Какова вероятность того, что все бояре берут мзду, если хотя бы один из бояр берёт мзду?



а) Для того чтобы все бояре брали мзду, каждый из них должен взять мзду. Поскольку каждый боярин берет мзду независимо от других с вероятностью  $1/2$ , вероятность того, что определенный боярин берет мзду, составляет  $1/2$ .

Таким образом, вероятность того, что все  $n$  боярей возьмут мзду, равна произведению вероятностей того, что каждый из них возьмет мзду:

$$P(\text{все бояре берут мзду}) = (1/2) * (1/2) * \dots * (1/2) = (1/2)^n$$

б) Чтобы определить вероятность того, что все бояре возьмут мзду при условии, что хотя бы один из них берет мзду, мы должны использовать условную вероятность.

Вероятность того, что хотя бы один из боярей возьмет мзду, равна дополнению вероятности того, что ни один из них не возьмет мзду. Вероятность того, что определенный боярин не возьмет мзду, составляет  $1/2$ , поэтому вероятность того, что ни один из  $n$  боярей не возьмет мзду, равна  $(1/2)^n$ .

Тогда вероятность того, что хотя бы один из боярей возьмет мзду, равна:

$$P(\text{хотя бы один боярь берет мзду}) = 1 - P(\text{ни один боярь не берет мзду}) = 1 - (1/2)^n$$

Таким образом, вероятность того, что все бояре возьмут мзду при условии, что хотя бы один из них берет мзду, равна:

$$P(\text{все бояре берут мзду} \mid \text{хотя бы один боярь берет мзду}) = P(\text{все бояре берут мзду}) / P(\text{хотя бы один боярь берет мзду}) = ((1/2)^n) / (1 - (1/2)^n)$$



пункт б решен верно, попробуй решить пункт а еще раз



Приношу извинения за предыдущую ошибку в решении пункта (а). Давайте решим его правильно.



Мы знаем, что каждый боярин берет мзду независимо от других с вероятностью  $1/2$ . Вероятность того, что определенный боярин возьмет мзду, составляет  $1/2$ .

Таким образом, вероятность того, что все  $n$  боярей возьмут мзду, равна произведению вероятностей того, что каждый из них возьмет мзду:

$$P(\text{все бояре берут мзду}) = (1/2) * (1/2) * \dots * (1/2) = (1/2)^n$$

Таким образом, вероятность того, что все бояре берут мзду, если случайно выбранный боярин берет мзду, равна  $(1/2)^n$ .

G

все еще неправильно. попробуй решить пункт а через условную вероятность



Приношу извинения за предыдущие ошибки. Решим пункт (а) с использованием условной вероятности.



Пусть А - событие "случайно выбранный боярин берет мзду", и В - событие "все бояре берут мзду".

Мы хотим найти вероятность события В при условии, что событие А произошло, то есть  $P(B|A)$ .

Из определения условной вероятности, мы знаем, что:

$$P(B|A) = P(A \cap B) / P(A),$$

где  $P(A \cap B)$  - вероятность одновременного наступления событий А и В, а  $P(A)$  - вероятность события А.

Вероятность того, что случайно выбранный боярин берет мзду, равна  $1/2$ , то есть  $P(A) = 1/2$ .

Вероятность того, что случайно выбранный боярин берет мзду, равна  $1/2$ , то есть  $P(A) = 1/2$ .

Также, чтобы все бояре брали мзду при условии, что случайно выбранный боярин берет мзду, необходимо, чтобы каждый из них брал мзду. Поэтому  $P(A \cap B)$  равно вероятности того, что все  $n$  боярей берут мзду, что мы обозначили как  $P(\text{все бояре берут мзду})$  из пункта (а).

Таким образом, получаем:

$$P(B|A) = P(A \cap B) / P(A) = P(\text{все бояре берут мзду}) / (1/2) = (1/2)^n / (1/2) = (1/2)^{(n-1)}.$$

Таким образом, вероятность того, что все бояре берут мзду, если случайно выбранный боярин берет мзду, равна  $(1/2)^{(n-1)}$ .



спасибо, все верно



Пожалуйста, рад был помочь! Если у вас возникнут еще вопросы, не стесняйтесь задавать. Я всегда готов помочь.



## задача 8

<https://stepik.org/course/76/promo>

<http://mathprofi.ru/>

два этих источника помогли мне разобраться с более базовыми терминами и понятиями. особенно второй источник помогает, потому что на нем есть конспекты и примеры простых задач по различным темам, после которых проще разобраться более сложными с нашего курса, а еще в него проще подглядеть, если что-то забываешь, чем в свои конспекты или лекции на ютубе))