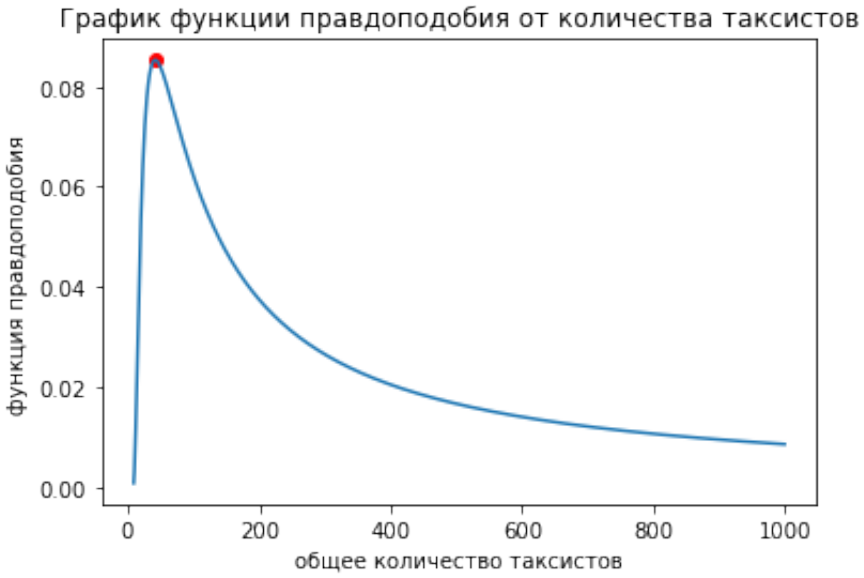
In [99]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.stats as stats
import math
```

1 задача (такси в Самарканде)

In [64]:
```
#пункт а
def likelihood(x): #функция правдоподобия
    p = 1
    for i in range(1, 9):
        p *= (x - i)/x
    return 9*p/x

n_list = np.arange(9, 1000)
probabilities = likelihood(n_list) #список вероятностей при различн
#print(probabilities)
```

In [38]:
```
plt.plot(n_list, probabilities)
p_max = max(probabilities) #находим максимальную вероятность из пол
index_max = np.argmax(probabilities) #находим количество таксистов,
n_max = n_list[index_max]
plt.scatter(n_max, p_max, c = 'red')
plt.xlabel('общее количество таксистов')
plt.ylabel('функция правдоподобия')
plt.title('График функции правдоподобия от количества таксистов');
```



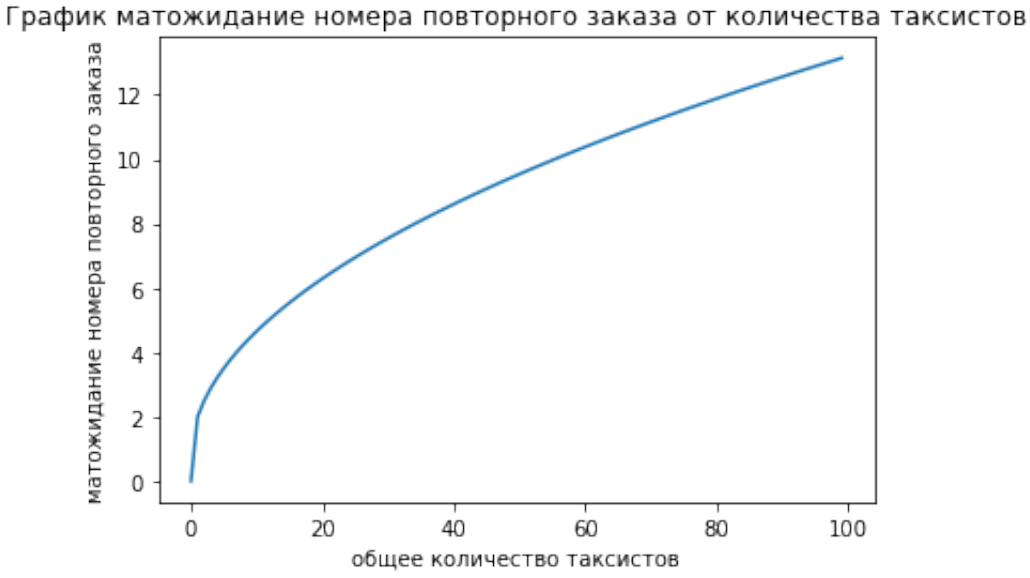
In [39]:
```
n_max #оценка числа таксистов методом максимального правдоподобия (
```

Out[39]: 42

In [324]:
```
#пункт б
def expectations_n(n):
    expect = 0
    for i in range(2, n + 2):
        prod = 1
        for j in range(i - 1):
            prod *= (n - j)/n
        expect += i * (i-1) * prod / n
    return expect
#expectation_n(20)
```

In [325]:
```
n_list = np.arange(100)
expectations = []
for n in n_list:
    expectations.append(expectations_n(n))
expectations = np.array(expectations)
#expectations
```

In [326]:
```
plt.plot(n_list, expectations)
plt.xlabel('общее количество таксистов')
plt.ylabel('матожидание номера повторного заказа')
plt.title('График матожидание номера повторного заказа от количеств
```



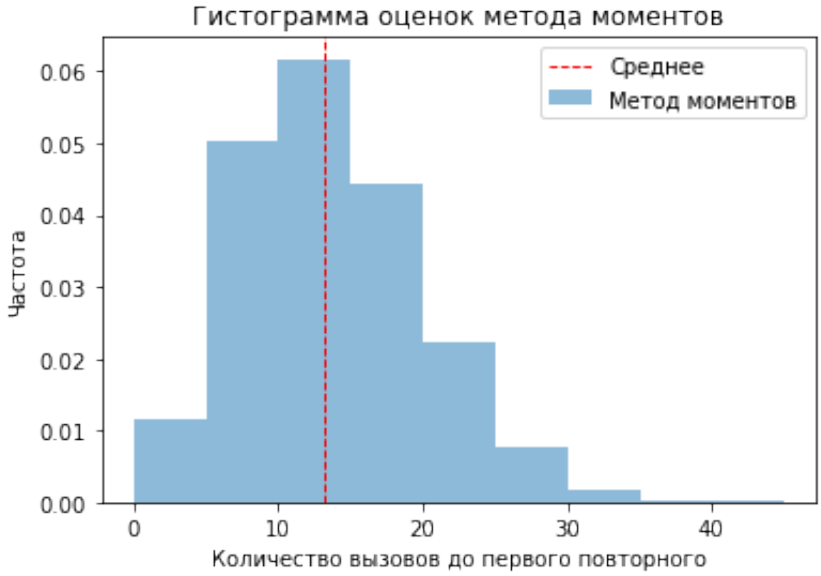
In [328]:
```
 = n_list[abs(expectations - 10) == min(abs(expectations - 10))][0]
#оценка числа таксистов методом моментов (нашли самое ближайщее ма
```

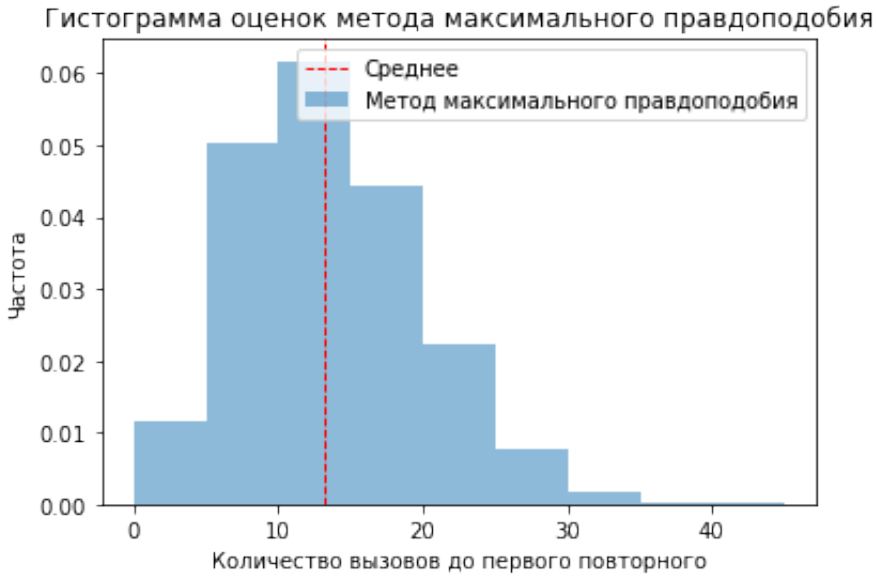
Out[328]: 55

In [330]:
```
#пункт в

n_sample = 100
n_obs = 10000
counts = []
```

```
for i in range(n_obs):
    drivers = set()
    count = 1

    while True:
        driver = random.randint(1, n_sample)

        if driver in drivers:  #проверка был ли водитель уже вызван
            counts.append(count)
            break

        drivers.add(driver)
        count += 1

# метод моментов
mean_moments = np.mean(counts)
variance_moments = np.var(counts)

# метод максимального правдоподобия
lambda_mle = 1 / mean_moments
variance_mle = mean_moments**2 / len(counts)

# гистограмма оценок метода моментов
plt.hist(counts, bins=range(0, max(counts) + 1, 5), alpha=0.5, dens
plt.axvline(mean_moments, color='r', linestyle='dashed', linewidth=
plt.legend()
plt.xlabel('Количество вызовов до первого повторного')
plt.ylabel('Частота')
plt.title('Гистограмма оценок метода моментов')
plt.show()

# гистограмма оценок метода максимального правдоподобия
plt.hist(counts, bins=range(0, max(counts) + 1, 5), alpha=0.5, dens
plt.axvline(1 / lambda_mle, color='r', linestyle='dashed', linewidt
plt.legend()
plt.xlabel('Количество вызовов до первого повторного')
plt.ylabel('Частота')
plt.title('Гистограмма оценок метода максимального правдоподобия')
plt.show()

# Оценка смещения, дисперсии и среднеквадратичной ошибки
bias_moments = mean_moments - n_sample
bias_mle = 1 / lambda_mle - n_sample

mse_moments = variance_moments + bias_moments**2
mse_mle = variance_mle + bias_mle**2

print("Метод моментов:")
print("Смещение:", bias_moments)
print("Дисперсия:", variance_moments)
print("Среднеквадратичная ошибка:", mse_moments)

print("\nМетод максимального правдоподобия:")
print("Смещение:", bias_mle)
```

```
print("Дисперсия:", variance_mle)
print("Среднеквадратичная ошибка:", mse_mle)
```

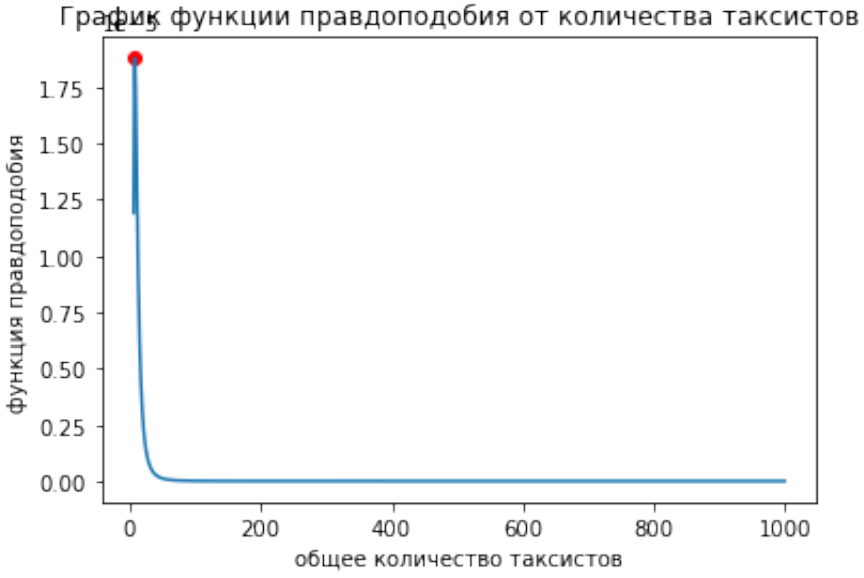





```
Метод моментов:
Смещение: -86.8438
Дисперсия: 38.49060156
Среднеквадратичная ошибка: 7580.336200000001

Метод максимального правдоподобия:
Смещение: -86.8438
Дисперсия: 0.017308559844000003
Среднеквадратичная ошибка: 7541.862906999845
```

2 задача (такси в Самарканде x2)

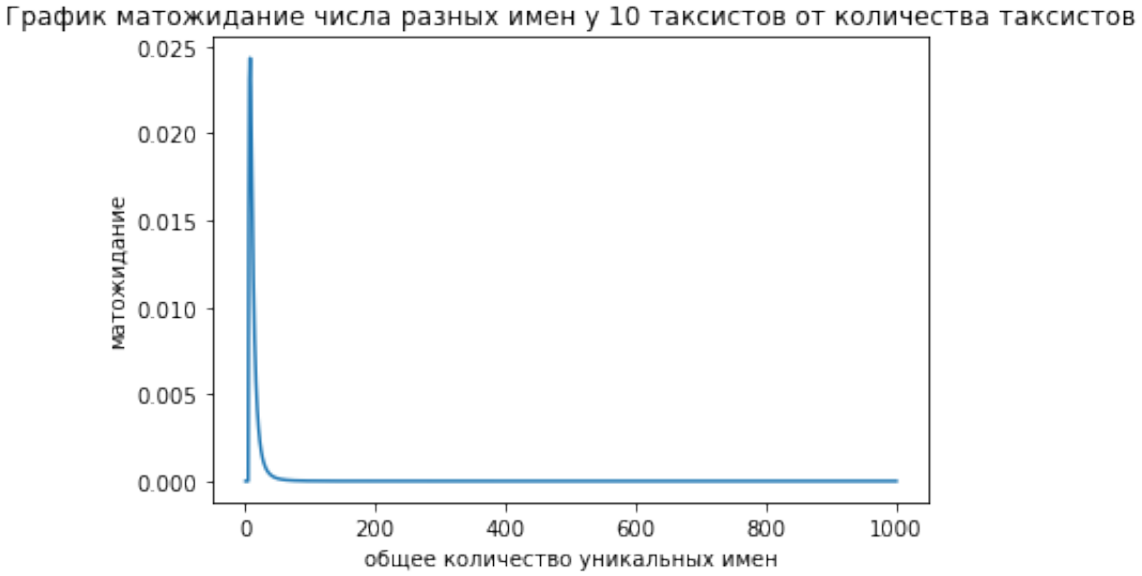
In [313]:
```
#пункт а
import itertools

def likelihood(n):
    L = 1
    for i in range(6):
        L *= (n-i) / n
    return (L / (n**4))

n_first = 6
n_taxi = 10
n_list = np.arange(6, 1000)
probabilities = list()
for n in n_list:
    probabilities.append(likelihood(n))

plt.plot(n_list, probabilities)
p_max = max(probabilities) #находим максимальную вероятность из полу
index_max = np.argmax(probabilities) #находим количество таксистов,
n_max = n_list[index_max]
plt.scatter(n_max, p_max, c = 'red')
plt.xlabel('общее количество таксистов')
plt.ylabel('функция правдоподобия')
plt.title('График функции правдоподобия от количества таксистов');
```



In [314]:
```
n_max #ml оценка числа таксистов
```

Out[314]: 8

In [318]:
```
#пункт б - матождиание имен таксистов
def expectation_names(n):
    e = 1
    for i in range(6):
        e *= (n-i)/n
    e = e*6**4/(n**4)
    return e

n_list = np.arange(1, 1001)
expectations = list()
for n in n_list:
    expectations.append(expectation_names(n))
expectations = np.array(expectations)

plt.plot(n_list, expectations)
plt.xlabel('общее количество уникальных имен')
plt.ylabel('матожидание')
plt.title('График матожидание числа разных имен у 10 таксистов от к
```



In [322]:
```
n_min = n_list[abs(expectations - 6) == min(abs(expectations - 6))]
n_min
```

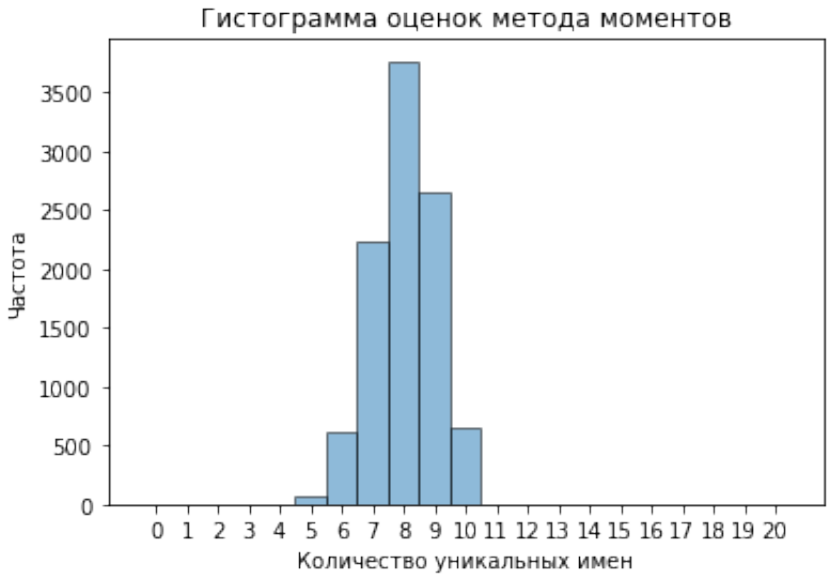
Out[322]: 8

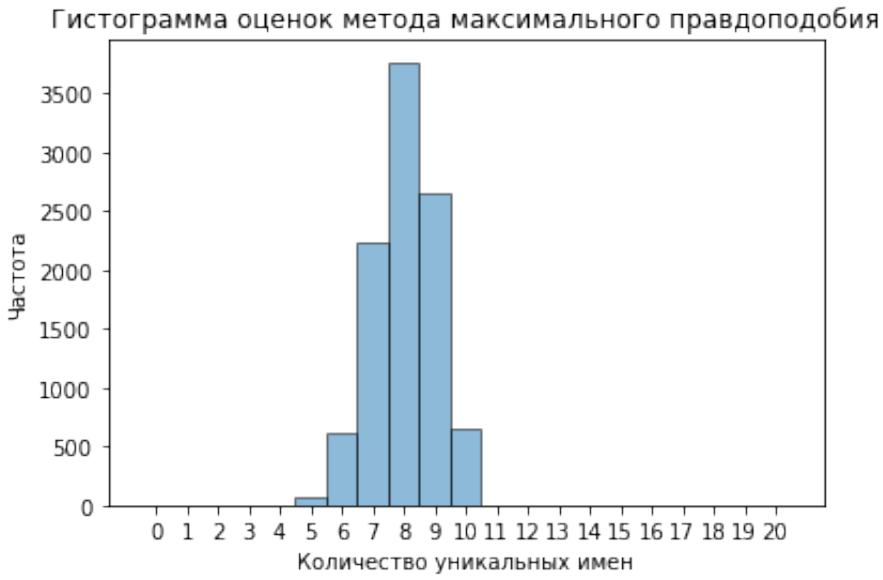
In [333]:
```
#пункт в

def simulate_taxi_calls(n_simulations, n_names, n_calls):
    unique_names_counts = []

    for _ in range(n_simulations):
        names = np.random.choice(range(1, n_names+1), size=n_calls,
        unique_names = np.unique(names)
        unique_names_counts.append(len(unique_names))

    return unique_names_counts
```

```
def method_of_moments_estimation(unique_names_counts):
    return np.mean(unique_names_counts)

def maximum_likelihood_estimation(unique_names_counts):
    def likelihood(n):
        p = 1 / n
        q = (n - 1) / n
        return np.prod([(p**k) * (q**(n_calls - k)) for k in unique

    n_values = range(1, 101)  # Ограничение сверху до 100
    likelihood_values = [likelihood(n) for n in n_values]
    return n_values[np.argmax(likelihood_values)]

# Параметры симуляции
n_simulations = 10000
n_names = 20
n_calls = 10

# Выполнение симуляций
unique_names_counts = simulate_taxi_calls(n_simulations, n_names, n

# Оценка параметров методом моментов и методом максимального правдо
mm_estimate = method_of_moments_estimation(unique_names_counts)
mle_estimate = maximum_likelihood_estimation(unique_names_counts)

# Расчет смещения, дисперсии и среднеквадратичной ошибки
bias_mm = mm_estimate - n_names
bias_mle = mle_estimate - n_names
variance_mm = np.var(unique_names_counts)
variance_mle = np.var(unique_names_counts)
mse_mm = variance_mm + bias_mm**2
mse_mle = variance_mle + bias_mle**2

# Вывод результатов
print("Метод моментов:")
print("Оценка:", mm_estimate)
print("Смещение:", bias_mm)
print("Дисперсия:", variance_mm)
print("Среднеквадратичная ошибка:", mse_mm)

print("\nМетод максимального правдоподобия:")
print("Оценка:", mle_estimate)
print("Смещение:", bias_mle)
print("Дисперсия:", variance_mle)
print("Среднеквадратичная ошибка:", mse_mle)

# Построение гистограммы для оценок метода моментов
plt.hist(unique_names_counts, bins=range(n_names+2), align='left',
plt.xlabel("Количество уникальных имен")
plt.ylabel("Частота")
plt.title("Гистограмма оценок метода моментов")
plt.xticks(range(n_names+1))
plt.show()
```

```
# Построение гистограммы для оценок метода максимального правдоподо
plt.hist(unique_names_counts, bins=range(n_names+2), align='left',
plt.xlabel("Количество уникальных имен")
plt.ylabel("Частота")
plt.title("Гистограмма оценок метода максимального правдоподобия")
plt.xticks(range(n_names+1))
plt.show()
```

```
Метод моментов:
Оценка: 8.0248
Смещение: -11.9752
Дисперсия: 1.06538496
Среднеквадратичная ошибка: 144.47079999999997

Метод максимального правдоподобия:
Оценка: 1
Смещение: -19
Дисперсия: 1.06538496
Среднеквадратичная ошибка: 362.06538496
```





3 задача (*данное сообщение создано иностранным средством массовой информации, выполняющим функцию иностранного агента)

In [214]:
```
mu = 1
std = 1
```

In [222]:
```
#a
#асимпт норм интервал
np.random.seed(42)
n_obs = 10000
n_sample = 20
sample_exp = np.random.exponential(scale = mu, size = (n_obs, n_sam
exp_means = sample_exp.mean(axis = 1)
exp_vars = np.sqrt(sample_exp.var(ddof = 1, axis = 1)/n_sample)
```

In [225]:
```
z = norm.ppf(0.975)

exp = np.array([exp_means - z * exp_vars, exp_means + z * exp_vars]
cnt = 0
for l, r in exp:
    if mu > l and mu < r:
        cnt += 1
print(cnt/n_obs) #с такой вероятностью ди накрывает настоящее матож
```

```
0.9036
```

In [269]:
```
#наивный бутстрэп
np.random.seed(42)
cnt_boot = 0
for n in range(n_obs):
    sample_exp_i = np.random.exponential(scale = mu, size = n_sampl
    bootstrap_sample = np.random.choice(sample_exp_i, size = (n_obs
    mean_boot = bootstrap_sample.mean(axis = 1)

    left_boot = np.percentile(mean_boot, 2.5)
    right_boot = np.percentile(mean_boot, 97.5)

    ci_boot = [left_boot, right_boot]
    if ci_boot[0] < 1 < ci_boot[1]:
        cnt_boot += 1

print(cnt_boot/n_obs) #с такой вероятностью ди накрывает настоящее
```

```
0.9062
```

In [267]:
```
#бутстрэп t-статистики
np.random.seed(42)
cnt_boot_t = 0
for n in range(n_obs):
    sample_exp_i = np.random.exponential(scale = mu, size = n_sample
    bootstrap_sample = np.random.choice(sample_exp_i, size = (n_obs,
    R_sample = (bootstrap_sample.mean(axis = 1) - sample_exp_i.mean

    left_boot = np.percentile(R_sample, 2.5)
    right_boot = np.percentile(R_sample, 97.5)

    ci_boot_t = [sample_exp_i.mean() - right_boot * np.sqrt(sample_
    if ci_boot_t[0] < 1 < ci_boot_t[1]:
        cnt_boot_t += 1
print(cnt_boot_t/n_obs) #с такой вероятностью ди накрывает настояще
```

```
0.9468
```

In [271]:
```
#пункт б - распределение стьюдента
#асимпт норм интервал
np.random.seed(42)
df = 3

sample_t = np.random.standard_t(df = df, size = (n_obs, n_sample))
t_means = sample_t.mean(axis = 1)
t_vars = np.sqrt(sample_t.var(ddof = 1, axis = 1)/n_sample)
z = norm.ppf(0.975)

t = np.array([t_means - z * t_vars, t_means + z * t_vars]).T
cnt_t = 0
for l, r in t:
    if mu > l and mu < r:
        cnt_t += 1
print(cnt_t/n_obs) #с такой вероятностью ди накрывает настоящее мат
```

```
0.1921
```

In [273]:
```
#наивный бустрэп
np.random.seed(42)
cnt_t_boot = 0
for n in range(n_obs):
    sample_t_i = np.random.standard_t(df=df, size = n_sample)
    bootstrap_sample = np.random.choice(sample_t_i, size = (n_obs,
    mean_boot_t = bootstrap_sample.mean(axis = 1)

    left_boot = np.percentile(mean_boot_t, 2.5)
    right_boot = np.percentile(mean_boot_t, 97.5)

    ci_boot = [left_boot, right_boot]
    if ci_boot[0] < 1 < ci_boot[1]:
        cnt_t_boot += 1

print(cnt_t_boot/n_obs)
```

```
0.1811
```

In [274]:
```
#бутстрэп t-статистики
np.random.seed(42)
cnt_boot_t = 0
for n in range(n_obs):
    sample_t_i = np.random.standard_t(df=df, size = n_sample)
    bootstrap_sample = np.random.choice(sample_t_i, size = (n_obs,
    R_sample = (bootstrap_sample.mean(axis = 1) - sample_t_i.mean()

    left_boot = np.percentile(R_sample, 2.5)
    right_boot = np.percentile(R_sample, 97.5)

    ci_boot_t = [sample_t_i.mean() - right_boot * np.sqrt(sample_t_
    if ci_boot_t[0] < 1 < ci_boot_t[1]:
        cnt_boot_t += 1
print(cnt_boot_t/n_obs)
```

```
0.2454
```

In [ ]:
```
ктах бутстрэп t-статистики показал лучший результат, правда при рас
```

4 задача (ждем хороших результатов за экзамен всей маршруткой)

In [167]:
```
data = pd.read_csv('stat_results.csv') #таблица с оценками за экзам
data
```

Out[167]:

	Фамилия	Имя	Оценка за экзамен	Результаты экзамена
0	Репенкова	Полина Александровна	4.0	16.0
1	Ролдугина	Софья Александровна	0.0	0.0
2	Сафина	Алия Линаровна	5.0	19.0
3	Сидоров	Иван Максимович	9.0	26.0
4	Солоухин	Иван Владимирович	6.0	21.0
...	...	...	...	...
327	Сенников	Александр -	5.0	19.0
328	Ся	Юйцянь -	0.0	0.0
329	Сятова	Альфия -	0.0	0.0
330	Темиркулов	Дастан Автандилович	0.0	0.0
331	Эшмеев	Павел Владиславович	4.0	16.0

332 rows × 4 columns

In [168]:
```
def vowel_test(x):
    vowels = set('уеыаоэёяию')
    if x[0].lower() in vowels:
        return 'гласная'
    else:
        return 'согласная'

data['первая буква'] = data['Фамилия'].apply(vowel_test)
data
```

Out[168]:

	Фамилия	Имя	Оценка за экзамен	Результаты экзамена	первая буква
0	Репенкова	Полина Александровна	4.0	16.0	согласная
1	Ролдугина	Софья Александровна	0.0	0.0	согласная
2	Сафина	Алия Линаровна	5.0	19.0	согласная
3	Сидоров	Иван Максимович	9.0	26.0	согласная
4	Солоухин	Иван Владимирович	6.0	21.0	согласная
...	...	...	...	...	...
327	Сенников	Александр -	5.0	19.0	согласная
328	Ся	Юйцянь -	0.0	0.0	согласная
329	Сятова	Альфия -	0.0	0.0	согласная
330	Темиркулов	Дастан Автандилович	0.0	0.0	согласная
331	Эшмеев	Павел Владиславович	4.0	16.0	гласная

332 rows × 5 columns

In [169]:
```
data_vowel = data[data['первая буква'] == 'гласная']['Результаты эк
data_consonant = data[data['первая буква'] == 'согласная']['Результ
#data_vowel, data_consonant
```

In [170]:
```
#пункт а - Тест Уэлча

p_value = stats.ttest_ind(data_vowel, data_consonant, equal_var = F
alpha = 0.05
print(p_value)

if p_value < alpha:
    print("Отвергаем нулевую гипотезу на уровне значимости 0.05: ож
else:
    print("Не отвергаем нулевую гипотезу на уровне значимости 0.05:
```

```
0.3974027153843839
Не отвергаем нулевую гипотезу на уровне значимости 0.05: ожидаемые
результаты равны
```

In [171]:
```
#пункт б - наивный бутстрэп

np.random.seed(42)
bootstrap = list()
for i in range(10000):
    boot_consonant = np.random.choice(data_consonant, size = data_c
    boot_vowel = np.random.choice(data_vowel, size = data_vowel.sha
    boot_stat = boot_consonant.mean() - boot_vowel.mean()
    bootstrap.append(boot_stat)

alpha = 0.05
left, right = np.percentile(bootstrap, [100*alpha/2, 100 - 100*alph
if 0 < left or 0 > right:
    print("Отвергаем нулевую гипотезу на уровне значимости 0.05: ож
else:
    print("Не отвергаем нулевую гипотезу на уровне значимости 0.05:
```

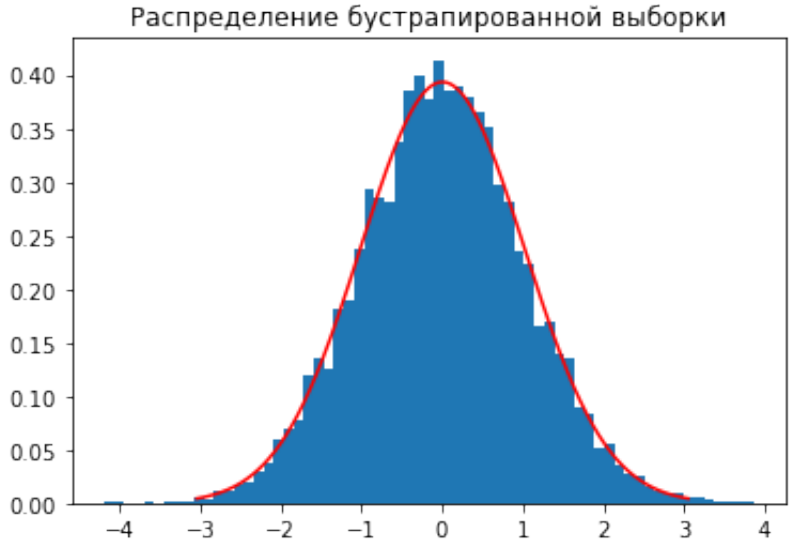
```
Не отвергаем нулевую гипотезу на уровне значимости 0.05: ожидаемые
результаты равны
```

In [172]:
```
lt.hist(bootstrap, bins = 'auto', density=True) #для наглядности пос
u, sigma = np.mean(bootstrap), np.std(bootstrap)
 = np.linspace(mu - 3*sigma, mu + 3*sigma, 10000)

lt.plot(x, stats.norm.pdf(x, mu, sigma), color='red', label='Нормаль
lt.title('Распределение бустрапированной выборки');
```

In [173]:
```
#пункт в - бутстрэп t-статистики

np.random.seed(42)
bootstrap_t = list()

vowel_mean = data_vowel.mean()
consonant_mean = data_consonant.mean()
vowel_std = data_vowel.std()
consonant_std = data_consonant.std()

for i in range(10000):
    boot_consonant = np.random.choice(data_consonant, size = data_c
    boot_vowel = np.random.choice(data_vowel, size = data_vowel.sha
    mean_new = boot_consonant.mean() - boot_vowel.mean() - (consona
    std_new = np.sqrt(boot_consonant.std()**2/boot_consonant.shape[
    boot_stat_t = mean_new/std_new
    bootstrap_t.append(boot_stat_t)

alpha = 0.05

left_t,right_t = np.percentile(bootstrap_t, [100*alpha/2, 100 - 100
diff_mean = consonant_mean - vowel_mean
diff_std = np.sqrt(consonant_std**2/data_consonant.shape[0] + vowel

print(left_t, right_t)#доверительный интервал для t-статистики

if 0 > diff_mean - diff_std * left_t or 0 < diff_mean - diff_std *
    print("Отвергаем нулевую гипотезу на уровне значимости 0.05: ож
else:
    print("Не отвергаем нулевую гипотезу на уровне значимости 0.05:
```

```
-2.104776018424727 1.9761593147208938
Не отвергаем нулевую гипотезу на уровне значимости 0.05: ожидаемые
результаты равны
```

In [132]:
```
plt.hist(bootstrap_t, bins = 'auto', density=True) #для наглядности
mu, sigma = np.mean(bootstrap_t), np.std(bootstrap_t)
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 10000)

plt.plot(x, stats.norm.pdf(x, mu, sigma), color='red', label='Норма
plt.title('Распределение бустрапированной выборки');
```



In [174]:
```
#пункт г - перестановочный тест
from itertools import permutations

len_consonant = data_consonant.shape[0]
len_vowel = data_vowel.shape[0]

permute = list()
for n in range(10000):
    perm = np.random.permutation(data['Результаты экзамена'])
    perm_cons = perm[:len_consonant]
    perm_vow = perm[len_consonant:]
    perm_mean = perm_cons.mean() - perm_vow.mean()
    permute.append(perm_mean)

left_p,right_p = np.percentile(permute, [100*alpha/2, 100 - 100*alp
if diff_mean < left_p or diff_mean > right_p:
    print("Отвергаем нулевую гипотезу на уровне значимости 0.05: ож
else:
    print("Не отвергаем нулевую гипотезу на уровне значимости 0.05:
```

```
Не отвергаем нулевую гипотезу на уровне значимости 0.05: ожидаемые
результаты равны
```

5 задача (все еще ждем хороших результатов)

In [177]:
```
def median_test(x):
    median = data['Результаты экзамена'].median()
    if x > median:
        return 'больше медианы'
    else:
        return 'меньше медианы'

data['медиана'] = data['Результаты экзамена'].apply(median_test)
data
```

Out[177]:

	Фамилия	Имя	Оценка за экзамен	Результаты экзамена	первая буква	медиана
0	Репенкова	Полина Александровна	4.0	16.0	согласная	меньше медианы
1	Ролдугина	Софья Александровна	0.0	0.0	согласная	меньше медианы
2	Сафина	Алия Линаровна	5.0	19.0	согласная	больше медианы
3	Сидоров	Иван Максимович	9.0	26.0	согласная	больше медианы
4	Солоухин	Иван Владимирович	6.0	21.0	согласная	больше медианы
...	...	...	...	...	...	...
327	Сенников	Александр -	5.0	19.0	согласная	больше медианы
328	Ся	Юйцянь -	0.0	0.0	согласная	меньше медианы
329	Сятова	Альфия -	0.0	0.0	согласная	меньше медианы
330	Темиркулов	Дастан Автандилович	0.0	0.0	согласная	меньше медианы
331	Эшмеев	Павел Владиславович	4.0	16.0	гласная	меньше медианы

332 rows × 6 columns

In [178]:
```
cross_med_letter = pd.crosstab(index = data['первая буква'], column
cross_med_letter
```

Out[178]:

медиана	больше медианы	меньше медианы
первая буква		
гласная	21	28
согласная	145	138

In [179]:
```
#пункт а - асимпт интервал для отношения шансов
consonant_chances = 145/138
vowel_chances = 21/28
len_cons = 138+145
len_vow = 21+28
```

In [182]:
```
sample_ratio = vowel_chances/consonant_chances
need_ratio = 1
se = math.sqrt(1/145 + 1/138 + 1/28 + 1/21)
left = sample_ratio * math.exp(-1.96 * se)
right = sample_ratio * math.exp(1.96 * se)
left, right #need_ratio = 1 попадает в этот интервал, значит гипоте
```

Out[182]: (0.38709024318230967, 1.3162320763800786)

In [183]:
```
from scipy.stats import norm
p_value = norm.cdf((sample_ratio - need_ratio)/se)*2
p_value # >0.05 значит гипотезу не отвергаем
```

Out[183]: 0.3592960710742057

In [184]:
```
#пункт б - асимпт интервал для отношения вероятностей
consonant_good = 145/len_cons
need_good = 1
vowel_good = 21/len_vow
ratio_good = vowel_good/consonant_good
std_sample = np.sqrt(vowel_good*(1-vowel_good)/(consonant_good**2 *

left = ratio_good - 1.96*std_sample
right = ratio_good + 1.96*std_sample

left, right #need_good = 1 попадает в этот интервал, значит гипотез
```

Out[184]: (0.5497893669666724, 1.1231170369742145)

In [185]:
```
p_value = norm.cdf((ratio_good - need_good)/std_sample)*2
p_value
```

Out[185]: 0.26347529277764214

In [332]:
```
#пункт в - асимпт интервал для навиного бутстрэпа
n_bootstrap = 1000
odds_ratios = np.zeros(n_bootstrap)
for i in range(n_bootstrap):
    bootstrap_sample = np.random.choice(len(more_median), size=len(
    bootstrap_table = np.zeros((2, 2))
    for j in bootstrap_sample:
        bootstrap_table[more_median[j]][starts_with_consonant[j]] +
    odds_ratio = (bootstrap_table[1][1] * bootstrap_table[0][0]) /
    odds_ratios[i] = odds_ratio

# Расчёт 95% интервала
lower_percentile = np.percentile(odds_ratios, 2.5)
upper_percentile = np.percentile(odds_ratios, 97.5)

# Проверка гипотезы о равенстве отношения шансов 1
observed_odds_ratio = (contingency_table[1][1] * contingency_table[
p_value = np.sum(odds_ratios >= observed_odds_ratio) / n_bootstrap

# Вывод результатов
print("95% интервал для отношения шансов:", (lower_percentile, uppe
print("P-значение:", p_value)
```

6 задача (*данное сообщение создано иностранным средством массовой информации, выполняющим функцию иностранного агента x2)

In [187]:
```
data['длина фамилии'] = data['Фамилия'].apply(lambda x: len(x))
data
```

Out[187]:

	Фамилия	Имя	Оценка за экзамен	Результаты экзамена	первая буква	медиана	длина фамилии
0	Репенкова	Полина Александровна	4.0	16.0	согласная	меньше медианы	9
1	Ролдугина	Софья Александровна	0.0	0.0	согласная	меньше медианы	9
2	Сафина	Алия Линаровна	5.0	19.0	согласная	больше медианы	6
3	Сидоров	Иван Максимович	9.0	26.0	согласная	больше медианы	7
4	Солоухин	Иван Владимирович	6.0	21.0	согласная	больше медианы	8
...	...	...	...	...	...	...	...
327	Сенников	Александр -	5.0	19.0	согласная	больше медианы	8
328	Ся	Юйцянь -	0.0	0.0	согласная	меньше медианы	2
329	Сятова	Альфия -	0.0	0.0	согласная	меньше медианы	6
330	Темиркулов	Дастан Автандилович	0.0	0.0	согласная	меньше медианы	10
331	Эшмеев	Павел Владиславович	4.0	16.0	гласная	меньше медианы	6

332 rows × 7 columns

In [190]:
```
#пункт а - метод моментов и выборочная кореляция
E_Y_mean = data['Результаты экзамена'].mean()
E_F_mean = data['длина фамилии'].mean()
beta = E_Y_mean/E_F_mean
beta
```

Out[190]: 2.0613026819923372

In [207]:
```
corr = data[['Результаты экзамена', 'длина фамилии']].corr()
corr_sample = corr['Результаты экзамена']['длина фамилии']
corr_sample #выборочная кореляция
```

Out[207]: 0.025328052669147543

In [211]:
```
#пункт б - перестановочный тест
np.random.seed(42)
corrs = []
n_obs = 10000
for i in range(n_obs):
    data['Перестановочный результат'] = np.random.permutation(data[
    corr = data[['Перестановочный результат', 'длина фамилии']].cor
    cor = corr['Перестановочный результат']['длина фамилии']
    corrs.append(cor)

plt.hist(np.array(corrs), bins = 'auto',density=True)
mu, sigma = np.mean(np.array(corrs)), np.std(np.array(corrs))
x = np.linspace(mu - 3*sigma, mu + 3*sigma, 10000)

plt.plot(x, stats.norm.pdf(x, mu, sigma), color='red', label='Норма
plt.title('Распределение бустрапированной выборки');
```

In [213]:
```
left_per = np.quantile(np.array(corrs), 0.025)
right_per = np.quantile(np.array(corrs), 0.975)
print(left_per, right_per) #ди
p_value = 2 * min(np.sum(np.array(corrs) > corr_sample)/n_obs, np.s
p_value #гипотеза не отвергается на всех наиболее используемых уров
```

```
-0.10680883506069982 0.11122418770480158
```

Out[213]: 0.658

7 задача (баловство с чатом гпт)

https://chat.openai.com/share/b7261585-84c4-49ac-afb8-c0f136d9d0b3

8 задача (наконец-то разбираемся со статистикой)

https://youtu.be/zeJD6dqJ5lo (https://youtu.be/zeJD6dqJ5lo) - очень полюбилось это видео, помогло разобраться в центральной предельной теореме и наконец узнать откуда в нормальном распределении появилось число π)

А вообще "3 blue 1 brown" мой любимый канал по математике, часто вещи, которые там рассказывают, переворачивают твое сознание и заставляют взглянуть на математику под другим углом