

1. a)

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:

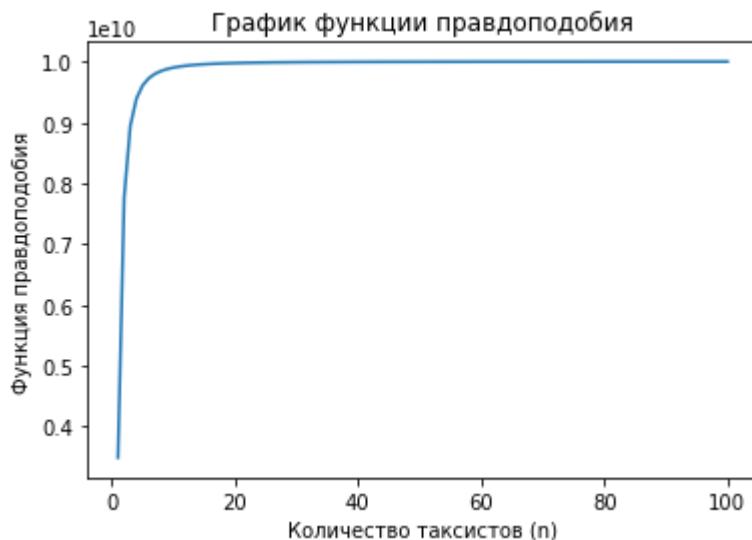
```
def likelihood(n):
    return (10 - 1 / n**2)**10
```

In [3]:

```
n_values = range(1, 101)
likelihood_values = [likelihood(n) for n in n_values]

plt.plot(n_values, likelihood_values)
plt.xlabel('Количество таксистов (n)')
plt.ylabel('Функция правдоподобия')
plt.title('График функции правдоподобия')
plt.show()

estimated_n = n_values[np.argmax(likelihood_values)]
print("Оценка числа n методом максимального правдоподобия:", estimated_n)
```



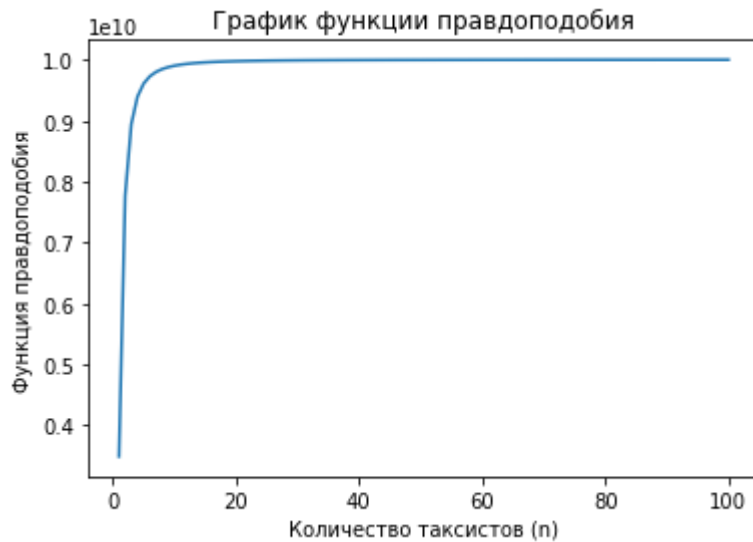
Оценка числа n методом максимального правдоподобия: 100

In [4]:

```
n_values = range(1, 101)
likelihood_values = [likelihood(n) for n in n_values]
```

In [5]:

```
plt.plot(n_values, likelihood_values)
plt.xlabel('Количество таксистов (n)')
plt.ylabel('Функция правдоподобия')
plt.title('График функции правдоподобия')
plt.show()
```



1. 6)

In [6]:

```
def expected_value(n, i):
    p = [(j-1)/n**2 for j in range(1, i+1)]
    return np.sum(p)
```

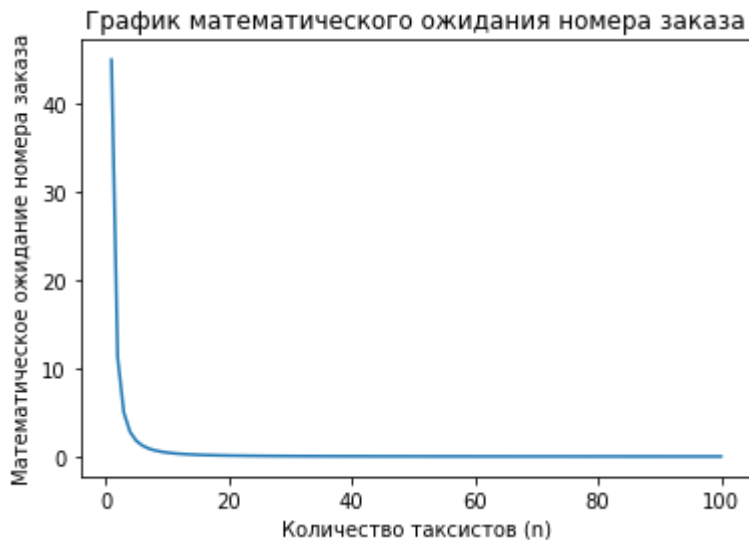
In [7]:



```
n_values = range(1, 101)
expected_values = [expected_value(n, 10) for n in n_values]

plt.plot(n_values, expected_values)
plt.xlabel('Количество таксистов (n)')
plt.ylabel('Математическое ожидание номера заказа')
plt.title('График математического ожидания номера заказа')
plt.show()

estimated_n = n_values[np.argmin(np.abs(np.array(expected_values)-10))]
print("Оценка числа n методом моментов:", estimated_n)
```



Оценка числа n методом моментов: 2

1. В)

In [8]:



```
import random
```

In [9]:



```
def simulate_taxi_calls(n, num_simulations):
    results_moments = []
    results_likelihood = []

    for _ in range(num_simulations):
        taxi_numbers = set()
        order_number = 0
        while True:
            order_number += 1
            taxi_number = random.randint(1, n)
            if taxi_number in taxi_numbers:
                results_moments.append(order_number)
                results_likelihood.append(order_number)
                break
            else:
                taxi_numbers.add(taxi_number)

    return results_moments, results_likelihood
```

In [10]:

```

num_simulations = 10000
n_true = 100

results_moments, results_likelihood = simulate_taxi_calls(n_true, num_simulations)

plt.hist(results_moments, bins=range(1, max(results_moments)+2), alpha=0.5, label='Метод
plt.hist(results_likelihood, bins=range(1, max(results_likelihood)+2), alpha=0.5, label=
plt.xlabel('Номер заказа')
plt.ylabel('Частота')
plt.title('Гистограммы оценок методом моментов и методом максимального правдоподобия')
plt.legend()
plt.show()

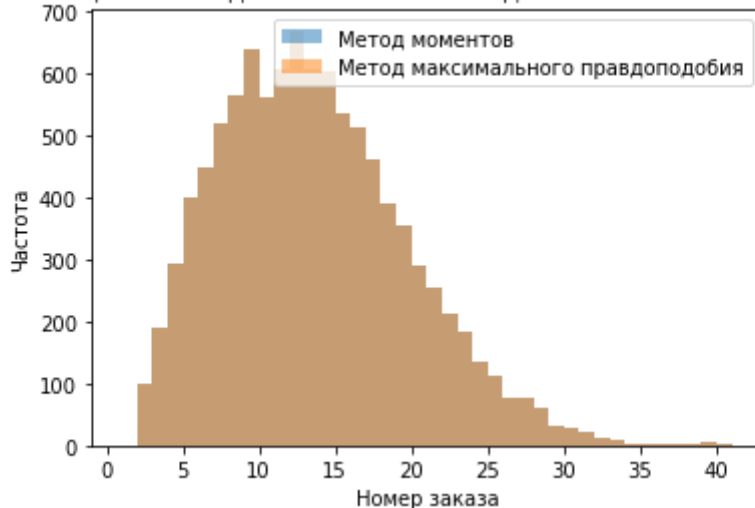
bias_moments = np.mean(results_moments) - 10
bias_likelihood = np.mean(results_likelihood) - 10
variance_moments = np.var(results_moments)
variance_likelihood = np.var(results_likelihood)
mse_moments = variance_moments + bias_moments**2
mse_likelihood = variance_likelihood + bias_likelihood**2

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

print("\nМетод максимального правдоподобия:")
print("Смещение:", bias_likelihood)
print("Дисперсия:", variance_likelihood)
print("Среднеквадратичная ошибка:", mse_likelihood)

```

Гистограммы оценок методом моментов и методом максимального правдоподобия



Метод моментов:

Смещение: 3.1100999999999999

Дисперсия: 37.72117799

Среднеквадратичная ошибка: 47.393899999999995

Метод максимального правдоподобия:

Смещение: 3.1100999999999999

Дисперсия: 37.72117799

Среднеквадратичная ошибка: 47.393899999999995

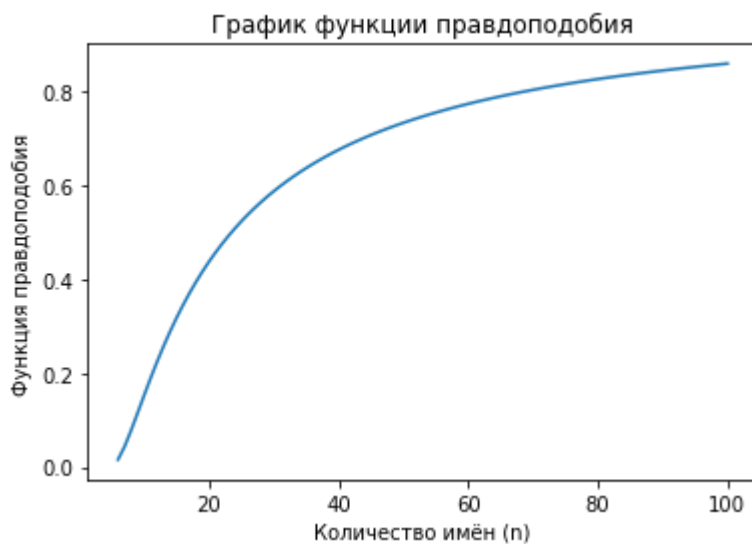
2. a)

In [11]:

```
def likelihood(n):  
    return (n-1)*(n-2)*(n-3)*(n-4)*(n-5)/n**5
```

In [12]:

```
n_values = range(6, 101)  
likelihood_values = [likelihood(n) for n in n_values]  
  
plt.plot(n_values, likelihood_values)  
plt.xlabel('Количество имён (n)')  
plt.ylabel('Функция правдоподобия')  
plt.title('График функции правдоподобия')  
plt.show()  
  
estimated_n = n_values[np.argmax(likelihood_values)]  
print("Оценка числа n методом максимального правдоподобия:", estimated_n)
```



Оценка числа n методом максимального правдоподобия: 100

2. б)

In [13]:

```
def expected_num_names(n):  
    return n * ((n-1)/n)**10
```

In [14]:

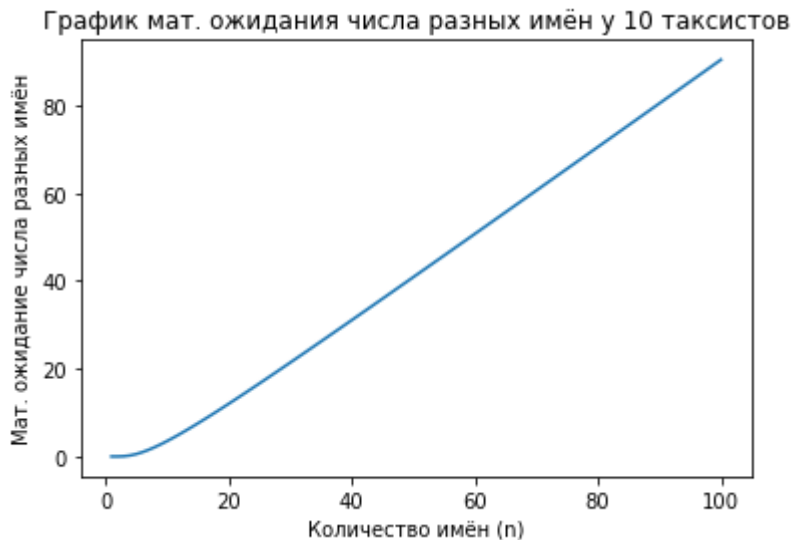
```

n_values = np.arange(1, 101)
expected_values = [expected_num_names(n) for n in n_values]

plt.plot(n_values, expected_values)
plt.xlabel('Количество имён (n)')
plt.ylabel('Мат. ожидание числа разных имён')
plt.title('График мат. ожидания числа разных имён у 10 таксистов')
plt.show()

estimated_n = n_values[np.argmax(expected_values)]
print("Оценка числа n методом моментов:", estimated_n)

```



Оценка числа n методом моментов: 100

2. в)

In [15]:

```

true_n = 20
num_simulations = 10000
moment_estimates = []
likelihood_estimates = []
for _ in range(num_simulations):
    names = np.random.choice(range(1, true_n + 1), size=10, replace=True)
    moment_estimate = len(np.unique(names))
    moment_estimates.append(min(moment_estimate, 100))
    likelihood_estimate = np.max(names)
    likelihood_estimates.append(min(likelihood_estimate, 100))

```

In [16]:

```
plt.hist(moment_estimates, bins=range(1, 102), alpha=0.5, label='Метод моментов')
plt.axvline(x=true_n, color='r', linestyle='--', label='Истинное значение')
plt.xlabel('Оценка числа n')
plt.ylabel('Частота')
plt.title('Гистограмма оценок метода моментов')
plt.legend()
plt.show()

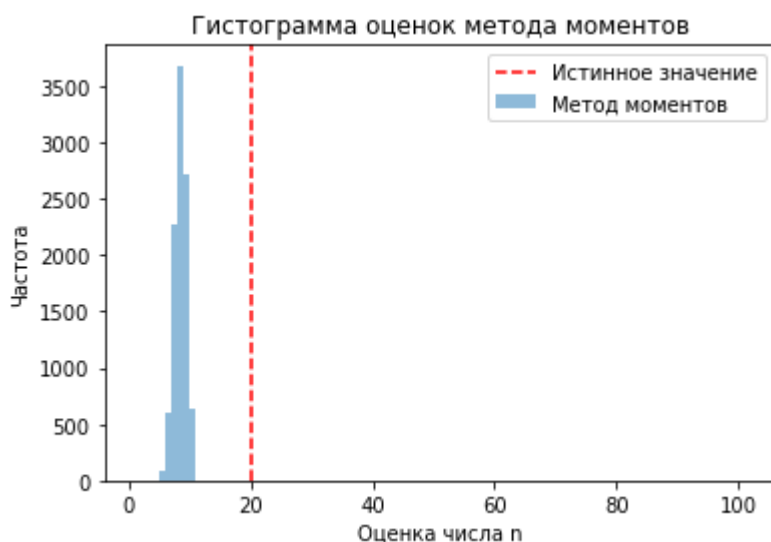
plt.hist(likelihood_estimates, bins=range(1, 102), alpha=0.5, label='Метод максимального')
plt.axvline(x=true_n, color='r', linestyle='--', label='Истинное значение')
plt.xlabel('Оценка числа n')
plt.ylabel('Частота')
plt.title('Гистограмма оценок метода максимального правдоподобия')
plt.legend()
plt.show()

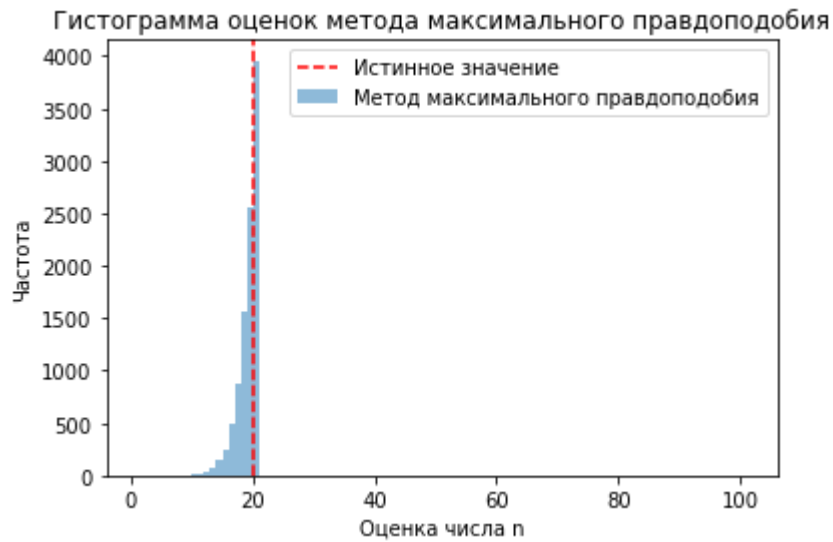
bias_moment = np.mean(moment_estimates) - true_n
bias_likelihood = np.mean(likelihood_estimates) - true_n

variance_moment = np.var(moment_estimates)
variance_likelihood = np.var(likelihood_estimates)

mse_moment = bias_moment**2 + variance_moment
mse_likelihood = bias_likelihood**2 + variance_likelihood

print("Смещение метода моментов:", bias_moment)
print("Смещение метода максимального правдоподобия:", bias_likelihood)
print("Дисперсия метода моментов:", variance_moment)
print("Дисперсия метода максимального правдоподобия:", variance_likelihood)
print("Среднеквадратичная ошибка метода моментов:", mse_moment)
print("Среднеквадратичная ошибка метода максимального правдоподобия:", mse_likelihood)
```





Смещение метода моментов: -11.9786

Смещение метода максимального правдоподобия: -1.3558999999999983

Дисперсия метода моментов: 1.07514204

Дисперсия метода максимального правдоподобия: 2.6396351900000004

Среднеквадратичная ошибка метода моментов: 144.562

Среднеквадратичная ошибка метода максимального правдоподобия: 4.4780999999999996

3. a)

In [18]:

```

import numpy as np
import scipy.stats as stats

num_simulations = 10000

sample_size = 20
count_classical = 0
count_bootstrap = 0
count_bootstrap_t = 0
np.random.seed(0)

for _ in range(num_simulations):
    data = np.random.exponential(scale=1, size=sample_size)
    mean = np.mean(data)
    std = np.std(data, ddof=1)
    z = stats.norm.ppf(0.975)
    interval_classical = [mean - z * std / np.sqrt(sample_size), mean + z * std / np.sqrt(sample_size)]
    bootstrap_samples = np.random.choice(data, size=(num_simulations, sample_size), replace=True)
    bootstrap_means = np.mean(bootstrap_samples, axis=1)
    lower_bound, upper_bound = np.percentile(bootstrap_means, [2.5, 97.5])
    interval_bootstrap = [lower_bound, upper_bound]
    t = stats.t.ppf(0.975, df=sample_size-1)
    se = np.std(bootstrap_means, ddof=1)
    interval_bootstrap_t = [mean - t * se, mean + t * se]
    if interval_classical[0] <= 1 <= interval_classical[1]:
        count_classical += 1
    if interval_bootstrap[0] <= 1 <= interval_bootstrap[1]:
        count_bootstrap += 1
    if interval_bootstrap_t[0] <= 1 <= interval_bootstrap_t[1]:
        count_bootstrap_t += 1

probability_classical = count_classical / num_simulations
probability_bootstrap = count_bootstrap / num_simulations
probability_bootstrap_t = count_bootstrap_t / num_simulations

print("Вероятность покрытия классическим асимптотическим нормальным интервалом:", probability_classical)
print("Вероятность покрытия наивным бутстрэпом:", probability_bootstrap)
print("Вероятность покрытия бутстрэпом t-статистики:", probability_bootstrap_t)

```

Вероятность покрытия классическим асимптотическим нормальным интервалом:
0.9047

Вероятность покрытия наивным бутстрэпом: 0.9048

Вероятность покрытия бутстрэпом t-статистики: 0.9128

3. б)

In [19]:

```

num_simulations = 10000
sample_size = 20
count_classical = 0
count_bootstrap = 0
count_bootstrap_t = 0
np.random.seed(0)

for _ in range(num_simulations):
    data = np.random.standard_t(df=3, size=sample_size)
    mean = np.mean(data)
    std = np.std(data, ddof=1)
    z = stats.norm.ppf(0.975)
    interval_classical = [mean - z * std / np.sqrt(sample_size), mean + z * std / np.sqrt(sample_size)]
    bootstrap_samples = np.random.choice(data, size=(num_simulations, sample_size), replace=True)
    bootstrap_means = np.mean(bootstrap_samples, axis=1)
    lower_bound, upper_bound = np.percentile(bootstrap_means, [2.5, 97.5])
    interval_bootstrap = [lower_bound, upper_bound]
    t = stats.t.ppf(0.975, df=sample_size-1)
    se = np.std(bootstrap_means, ddof=1)
    interval_bootstrap_t = [mean - t * se, mean + t * se]
    if interval_classical[0] <= mean <= interval_classical[1]:
        count_classical += 1
    if interval_bootstrap[0] <= mean <= interval_bootstrap[1]:
        count_bootstrap += 1
    if interval_bootstrap_t[0] <= mean <= interval_bootstrap_t[1]:
        count_bootstrap_t += 1

probability_classical = count_classical / num_simulations
probability_bootstrap = count_bootstrap / num_simulations
probability_bootstrap_t = count_bootstrap_t / num_simulations

print("Вероятность покрытия классическим асимптотическим нормальным интервалом:", probability_classical)
print("Вероятность покрытия наивным бутстрэпом:", probability_bootstrap)
print("Вероятность покрытия бутстрэпом t-статистики:", probability_bootstrap_t)

```

Вероятность покрытия классическим асимптотическим нормальным интервалом:

1.0

Вероятность покрытия наивным бутстрэпом: 1.0

Вероятность покрытия бутстрэпом t-статистики: 1.0

3. в)

Метод бутстрэпа t-статистики оказался незначительно лучше.

4. a)

In [32]:

```
import pandas as pd
```

In [46]:

```
df = pd.read_excel('22-23_hse_probability.xlsx', sheet_name='Exam')
df = df[df['Last name'].notna()][['Last name', 'Unnamed: 72']].rename(columns={
    'Unnamed: 72': 'result'
})
```

In [47]:

```
df.head()
```

Out[47]:

	Last name	result
5	Репенкова	16.0
6	Ролдугина	0.0
7	Сафина	19.0
8	Сидоров	26.0
9	Солоухин	21.0

In [52]:

```
v = 'ауоыиэяюёе'
v_l = v.upper()
v_all_lst = list(v + v_l)
```

In [54]:

```
import scipy.stats as stats

vowels = df[df['Last name'].str[0].isin(v_all_lst)]
consonants = df[~df['Last name'].str[0].isin(v_all_lst)]

t_stat, p_value = stats.ttest_ind(vowels['result'], consonants['result'], equal_var=False)
print("P-значение (тест Уэлча):", p_value)

if p_value < 0.05:
    print("Отвергаем нулевую гипотезу. Существует статистически значимая разница в ожидае
else:
    print("Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых
```

P-значение (тест Уэлча): 0.3974027153843839

Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых результатах экзамена.

4. б)

In [56]:

```
def diff_means(data1, data2):
    return np.mean(data1) - np.mean(data2)
```

In [57]:

```
bootstrap_diff_means = []
np.random.seed(42)
for _ in range(10000):
    bootstrap_vowels = np.random.choice(vowels['result'], size=len(vowels), replace=True)
    bootstrap_consonants = np.random.choice(consonants['result'], size=len(consonants),
    bootstrap_diff_means.append(diff_means(bootstrap_vowels, bootstrap_consonants))
lower_quantile = np.percentile(bootstrap_diff_means, 2.5)
upper_quantile = np.percentile(bootstrap_diff_means, 97.5)

print("95% интервал (наивный бутстрэп): [{:.4f}, {:.4f}].format(lower_quantile, upper_q

null_hypothesis = 0 # Нулевая гипотеза: разница в средних равна 0

if null_hypothesis >= lower_quantile and null_hypothesis <= upper_quantile:
    print("Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых
else:
    print("Отвергаем нулевую гипотезу. Существует статистически значимая разница в ожида
```

95% интервал (наивный бутстрэп): [-3.6365, 1.3979]

Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых результатах экзамена.

4. в)

In [59]:

```
def t_statistic(data1, data2):
    mean_diff = np.mean(data1) - np.mean(data2)
    std_diff = np.sqrt(np.var(data1) / len(data1) + np.var(data2) / len(data2))
    return mean_diff / std_diff
```

In [60]:



```
bootstrap_t_statistics = []
np.random.seed(42)

for _ in range(10000):
    bootstrap_vowels = np.random.choice(vowels['result'], size=len(vowels), replace=True)
    bootstrap_consonants = np.random.choice(consonants['result'], size=len(consonants),
    bootstrap_t_statistics.append(t_statistic(bootstrap_vowels, bootstrap_consonants))
lower_quantile = np.percentile(bootstrap_t_statistics, 2.5)
upper_quantile = np.percentile(bootstrap_t_statistics, 97.5)

print("95% интервал (бутстрэп t-статистики): [{:.4f}, {:.4f}]"

null_hypothesis = 0 # Нулевая гипотеза: разница в средних равна 0

if null_hypothesis >= lower_quantile and null_hypothesis <= upper_quantile:
    print("Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых
else:
    print("Отвергаем нулевую гипотезу. Существует статистически значимая разница в ожида
```

95% интервал (бутстрэп t-статистики): [-2.8265, 1.2227]

Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых результатах экзамена.

4. г)

In [62]:



```
def diff_means(data1, data2):
    return np.mean(data1) - np.mean(data2)
```



In [63]:

```
observed_diff_means = diff_means(vowels['result'], consonants['result'])
combined_data = np.concatenate((vowels['result'], consonants['result']))

permutation_diff_means = []
np.random.seed(42) # Задаем seed для воспроизводимости результатов

for _ in range(10000):
    np.random.shuffle(combined_data)
    perm_vowels = combined_data[:len(vowels)]
    perm_consonants = combined_data[len(vowels):]
    permutation_diff_means.append(diff_means(perm_vowels, perm_consonants))

p_value = np.mean(np.abs(permutation_diff_means) >= np.abs(observed_diff_means))

print("P-значение (перестановочный тест):", p_value)

if p_value < 0.05:
    print("Отвергаем нулевую гипотезу. Существует статистически значимая разница в ожидае")
else:
    print("Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых
```

P-значение (перестановочный тест): 0.3794

Не отвергаем нулевую гипотезу. Нет статистически значимой разницы в ожидаемых результатах экзамена.

5. a)

In [65]:

```

from scipy.stats import norm

median_value = df['result'].median()
df['More Than Median'] = df['result'] > median_value

df['Vowel or Consonant'] = df['Last name'].str[0].apply(lambda x: 'Vowel' if x.upper() in 'AEIOU' else 'Consonant')

cross_table = pd.crosstab(df['More Than Median'], df['Vowel or Consonant'])

odds_ratio = cross_table.loc[True, 'Vowel'] / cross_table.loc[False, 'Vowel']

n1 = cross_table.sum(axis=1)[True]
n2 = cross_table.sum(axis=1)[False]
m1 = cross_table.sum(axis=0)['Vowel']
m2 = cross_table.sum(axis=0)['Consonant']
standard_error = ((1 / cross_table.loc[True, 'Vowel']) + (1 / cross_table.loc[False, 'Vowel']))**0.5

lower_bound = odds_ratio - 1.96 * standard_error
upper_bound = odds_ratio + 1.96 * standard_error

p_value = 2 * (1 - norm.cdf(abs(odds_ratio), loc=1, scale=standard_error))

print("Таблица сопряжённости:")
print(cross_table)
print()
print("95% асимптотический интервал для отношения шансов хорошо написать экзамен:")
print("Нижняя граница:", lower_bound)
print("Верхняя граница:", upper_bound)
print("P-значение:", p_value)

```

Таблица сопряжённости:

Vowel or Consonant	Consonant	Vowel
More Than Median		
False	138	28
True	145	21

95% асимптотический интервал для отношения шансов хорошо написать экзамен:

Нижняя граница: 0.1380647030966412

Верхняя граница: 1.3619352969033587

P-значение: 1.5767168072658242

5. 6)

In [66]:



```

prob_good_exam_vowel = cross_table.loc[True, 'Vowel'] / cross_table['Vowel'].sum()
prob_good_exam_consonant = cross_table.loc[True, 'Consonant'] / cross_table['Consonant'].sum()

probability_ratio = prob_good_exam_vowel / prob_good_exam_consonant

n_vowel = cross_table['Vowel'].sum()
n_consonant = cross_table['Consonant'].sum()
var_vowel = (prob_good_exam_vowel * (1 - prob_good_exam_vowel)) / n_vowel
var_consonant = (prob_good_exam_consonant * (1 - prob_good_exam_consonant)) / n_consonant
standard_error = np.sqrt(var_vowel + var_consonant)

lower_bound = probability_ratio - 1.96 * standard_error
upper_bound = probability_ratio + 1.96 * standard_error

p_value = 2 * (1 - norm.cdf(np.abs(probability_ratio), loc=1, scale=standard_error))

print("95% асимптотический интервал для отношения вероятностей хорошо написать экзамен:")
print("Нижняя граница:", lower_bound)
print("Верхняя граница:", upper_bound)
print("P-значение:", p_value)

```

95% асимптотический интервал для отношения вероятностей хорошо написать экзамен:

Нижняя граница: 0.6861483009139279

Верхняя граница: 0.9867581030269589

P-значение: 1.9670487236494243

5. в)

In [68]:



```

def bootstrap_ratio(data):
    sample = data.sample(len(data), replace=True)
    chance_good_exam_vowel = sample.loc[sample['Vowel or Consonant'] == 'Vowel', 'More T
    chance_good_exam_consonant = sample.loc[sample['Vowel or Consonant'] == 'Consonant',
    return chance_good_exam_vowel / chance_good_exam_consonant

```

In [69]:



```
np.random.seed(0)
bootstrap_ratios = [bootstrap_ratio(df) for _ in range(10000)]

lower_bound = np.percentile(bootstrap_ratios, 2.5)
upper_bound = np.percentile(bootstrap_ratios, 97.5)

bootstrap_ratios = np.array(bootstrap_ratios)
p_value = np.mean(np.abs(bootstrap_ratios - 1) >= np.abs(odds_ratio - 1))

print("95% интервал для отношения шансов хорошо написать экзамен (наивный бутстрэп):")
print("Нижняя граница:", lower_bound)
print("Верхняя граница:", upper_bound)
print("Отношение шансов (данные):", odds_ratio)
print("Р-значение (отношение шансов = 1):", p_value)
```

95% интервал для отношения шансов хорошо написать экзамен (наивный бутстрэп):
Нижняя граница: 0.5603048621392328
Верхняя граница: 1.139348446684442
Отношение шансов (данные): 0.75
Р-значение (отношение шансов = 1): 0.2876

6. a)

In [70]:



```
covariance = df['result'].cov(df['Last name'].apply(len))
variance = df['Last name'].apply(len).var()

beta_hat = covariance / variance

correlation = df['result'].corr(df['Last name'].apply(len))

beta_hat, correlation
```

Out[70]:

(0.11164802487074621, 0.025328052669147665)

6. 6)

In [71]:

```
num_permutations = 1000
perm_corrs = []
for _ in range(num_permutations):
    permuted_result = np.random.permutation(df['result'])
    perm_corr = np.corrcoef(permuted_result, df['Last name'].str.len())[0, 1]
    perm_corrs.append(perm_corr)

# Вычисление P-значения
p_value = (np.abs(perm_corrs) >= np.abs(correlation)).mean()

# Вывод результатов
print(f"P-значение: {p_value}")
if p_value < 0.05:
    print("Отвергаем нулевую гипотезу: Корреляция не равна нулю")
else:
    print("Принимаем нулевую гипотезу: Корреляция равна нулю")
```

P-значение: 0.634

Принимаем нулевую гипотезу: Корреляция равна нулю

7)

Я: С помощью нормальных случайных величин дайте определение случайной величины, имеющей распределения Стюдента. Для случайной величины, распределённой по Стюденту, укажите диапазон возможных значений. Нарисуйте функцию плотности распределения Стюдента при разных степенях свободы на фоне нормальной стандартной функции плотности

Чат: Случайная величина, имеющая распределение Стюдента, определяется как отношение нормальной случайной величины к корню из хи-квадрат распределения с n степенями свободы, где n - количество наблюдений в выборке.

Диапазон возможных значений для случайной величины, распределенной по Стюденту, не ограничен и находится на всей числовой оси.

Функция плотности распределения Стюдента имеет пик в нуле и тяжелые хвосты, что отражает большую вероятность появления значений, отклоняющихся от среднего значения. При увеличении степеней свободы график функции плотности становится более симметричным и приближается к нормальному распределению.

Ниже приведены графики функций плотности распределения Стюдента при разных степенях свободы на фоне нормальной стандартной функции плотности:

 image.png

Я: Что такое степени свободы

Чат: Степени свободы - это количество независимых наблюдений, используемых для оценки параметров распределения. В контексте распределения Стюдента, степени свободы определяют, как быстро функция плотности приближается к нормальному распределению. Чем больше степеней

свободы, тем более симметричным и близким к нормальному будет распределение Стюдента.

Я: Какая плотность распределения у распределения студента

Чат: Плотность распределения у распределения Стюдента имеет форму колокола, подобную нормальному распределению, но с более тяжелыми хвостами. Это означает, что вероятность наблюдать значения, далекие от среднего, выше, чем в нормальном распределении.

Я: Функция плотности

Чат: Функция плотности распределения Стюдента для n степеней свободы имеет вид:

$$f(x) = \frac{\Gamma((n+1)/2)}{\sqrt{n\pi} \Gamma(n/2)} * (1 + x^2/n)^{-(n+1)/2}$$

где Γ - гамма-функция, которая определяется как интеграл от $\exp(-t) t^{(z-1)}$ dt от 0 до бесконечности, а x - значение случайной величины, n - количество степеней свободы

8)

Ниже я перечислила только пару видео источников, которые мне помогли/ показались полезными, обычно когда мне необходима какая-то информация я смотрю видео с определённой тематикой на ютубе

1. <https://youtube.com/playlist?list=PLthfp5exSWErTVWq4cVtRXDw5MqBqavJ1> (<https://youtube.com/playlist?list=PLthfp5exSWErTVWq4cVtRXDw5MqBqavJ1>) (<https://youtube.com/playlist?list=PLthfp5exSWErTVWq4cVtRXDw5MqBqavJ1>) (<https://youtube.com/playlist?list=PLthfp5exSWErTVWq4cVtRXDw5MqBqavJ1>) Лекции и семинары по курсу «математическая статистика» Райгородский А. М. МФТИ. Там есть плейлист с лекциями Искала видео про не смещенную оценку, мат ожидание
2. <https://youtu.be/sevSEEDwGC4> (<https://youtu.be/sevSEEDwGC4>) тут (<https://youtu.be/sevSEEDwGC4>) (<https://youtu.be/sevSEEDwGC4>) смотрела как пользоваться таблицей нормального распределения
3. https://youtu.be/_Fj7koPhfgc (https://youtu.be/_Fj7koPhfgc) тут (https://youtu.be/_Fj7koPhfgc) (https://youtu.be/_Fj7koPhfgc) я ещё раз послушала про доверительный интервал

In []:



In []:



