Домашнее задание по матстату

ФИО: Тюльберова Анна

Группа: БЭК212

In [1]:
```
#импорт всех нужных библиотек

import pandas as pd
import numpy as np
import scipy.stats as st
import random

import matplotlib.pyplot as plt
import seaborn as sns

from scipy.optimize import fmin
```

```
C:\Users\annat\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A Num
Py version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected vers
ion 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
alpha = 0.05
```

Задача 1

Функция правдоподобия:

$$L = \frac{(n-1)\cdot(n-2)\cdot(n-3)\cdot(n-4)\cdot(n-5)\cdot(n-6)\cdot(n-7)\cdot(n-8)}{n^9}$$

Метод максимального правдоподобия: максимизируем функцию правдоподобия с помощью scipy.optimize.

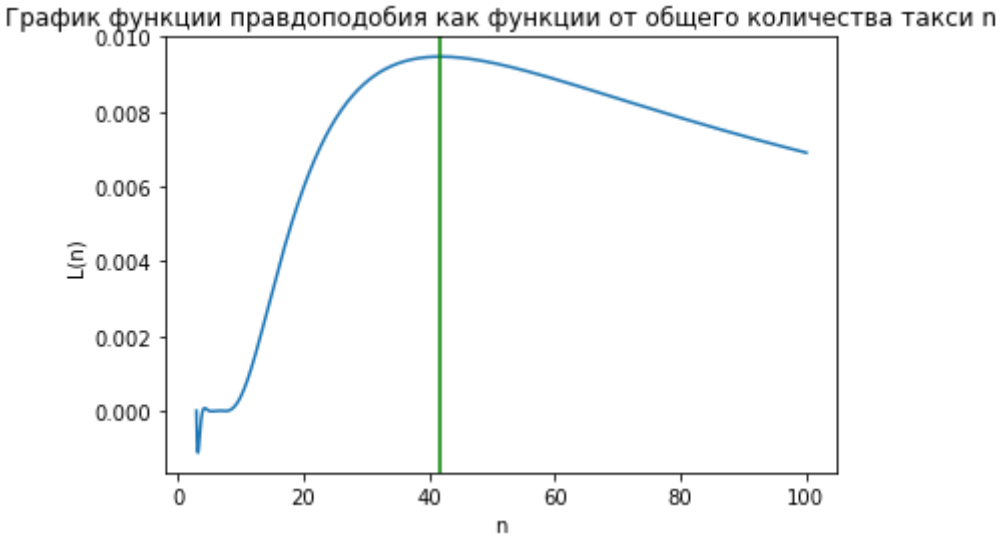
n_optimal - оценка числа n.

In [3]:
```
def func(i):
    return(((i-1)*(i-2)*(i-3)*(i-4)*(i-5)*(i-6)*(i-7)*(i-8)) / (i ** 9))

n_optimal = fmin(lambda x: -func(x), 100)[0]
```

```
Optimization terminated successfully.
         Current function value: -0.009473
         Iterations: 22
         Function evaluations: 44
```

Построим график для функции правдоподобия и отметим на нем n_optimal.

In [4]:
```
n = np.linspace(3, 100, 10000)

plt.plot(n, func(n))
plt.axvline(n_optimal, color='g')
plt.title('График функции правдоподобия как функции от общего количества такси n')
plt.xlabel('n')
plt.ylabel('L(n)');
```



Задание 3

Для того, чтобы использовать бутстрэп, напишем функцию, которая его реализует. Эта функция будет использоваться и в следующих задачах.

In [5]:
```
#функция для бутстрэпа (идея взята из семинара 13 по Анализу данных)

def bootstrap_sample(x, sample_num):
    sample = np.random.choice(x, size=(x.size, sample_num), replace=True)
    return sample
```

In [6]:
```
#это пока не надо
num_naive = 0
num_t = 0

lambda_ = 1

X = np.random.exponential(lambda_, (20, 10**4))

for i in X.T:
    boot = bootstrap_sample(i, 10**4)

    boot_mean = np.mean(boot, axis=0)
    q_left = np.quantile(boot_mean, alpha/2)
    q_right = np.quantile(boot_mean, 1-alpha/2)
    if q_left < lambda_ and q_right > lambda_:
        num_naive += 1


    X_mean = i.mean()
    X_std = i.std()
    boot_t_test = (np.mean(boot - X_mean, axis=0))/np.std(boot, axis=0)
    q_left = X_mean - np.quantile(boot_t_test, 1-alpha/2)*X_std
    q_right = X_mean - np.quantile(boot_t_test, alpha/2)*X_std
    if q_left < lambda_ and q_right > lambda_:
        num_t += 1

num_naive, num_t
```

Out[6]:
```
(9075, 9468)
```

Также напишем функцию, которая будет считать доверительные интервалы и выводить число тех интервалов, которые накрывают мат. ожидание.

In [7]:
```
#функция для подсчета доверительных интервалов
def boot_CI(i, lambda_, alpha):

    #асимптотический интервал
    x_mean = i.mean()
    z_crit = st.norm.ppf(1 - alpha/2)
    x_std = i.std()
    x_n = i.shape[0]

    q_left = x_mean - z_crit * x_std / (x_n ** (1/2))
    q_right = x_mean + z_crit * x_std / (x_n ** (1/2))

    if (lambda_ >= q_left) and (lambda_ <= q_right):
        num_asy = 1
    else:
        num_asy = 0

    boot = bootstrap_sample(i, 10**4)

    #наивный бутстрэп
    boot_mean = np.mean(boot, axis=0)
    q_left_n = np.quantile(boot_mean, alpha/2)
    q_right_n = np.quantile(boot_mean, 1-alpha/2)
    if q_left_n <= lambda_ and q_right_n >= lambda_:
        num_naive = 1
    else:
        num_naive = 0

    #бутстрэп т-тест
    X_mean = i.mean()
    X_std = i.std()
    boot_t_test = (np.mean(boot - X_mean, axis=0))/np.std(boot, axis=0)
    q_left_t = X_mean - np.quantile(boot_t_test, 1-alpha/2)*X_std
    q_right_t = X_mean - np.quantile(boot_t_test, alpha/2)*X_std
    if q_left_t <= lambda_ and q_right_t >= lambda_:
        num_t = 1
    else:
        num_t = 0
    return num_asy, num_naive, num_t
```

Пункт а.

Теперь создадим 10000 симуляций и посчитаем вероятность накрытия, учитывая, что наблюдения распределены экспоненциально с интенсивностью 1 (lambda_ = 1).

In [8]:
```
lambda_ = 1

#10**4 симуляций
X = np.random.exponential(lambda_, (20, 10**4))

all_num = np.apply_along_axis(boot_CI, 1, X.T, lambda_=1, alpha=0.05)

p_cover_asy = all_num.sum(axis=0)[0] / 10**4
p_cover_naive = all_num.sum(axis=0)[1] / 10**4
p_cover_t = all_num.sum(axis=0)[2] / 10**4

print('Ответ пункт а')
print('Вероятность накрытия м. о. (классический асимптотический нормальный интервал)
```

```
print('Вероятность накрытия м. о. (наивный бутстрэп): ', p_cover_naive)
print('Вероятность накрытия м. о. (бутстрэп t-статистики): ', p_cover_t)
```

```
Ответ пункт а
Вероятность накрытия м. о. (классический асимптотический нормальный интервал):  0.90
18
Вероятность накрытия м. о. (наивный бутстрэп):  0.9072
Вероятность накрытия м. о. (бутстрэп t-статистики):  0.9501
```

Пункт б.

Теперь пересчитаем вероятности накрытия, при условии, что наблюдения имеют распределение Стьюдента с тремя степенями свободы.

In [9]:
```
lambda_ = 0

#10**4 симуляций
X = np.random.standard_t(3, (20, 10**4))

all_num = np.apply_along_axis(boot_CI, 1, X.T, lambda_=0, alpha=0.05)

p_cover_asy = all_num.sum(axis=0)[0] / 10**4
p_cover_naive = all_num.sum(axis=0)[1] / 10**4
p_cover_t = all_num.sum(axis=0)[2] / 10**4

print('Ответ пункт б')
print('Вероятность накрытия м. о. (классический асимптотический нормальный интервал)
print('Вероятность накрытия м. о. (наивный бутстрэп): ', p_cover_naive)
print('Вероятность накрытия м. о. (бутстрэп t-статистики): ', p_cover_t)
```

```
Ответ пункт б
Вероятность накрытия м. о. (классический асимптотический нормальный интервал):  0.93
63
Вероятность накрытия м. о. (наивный бутстрэп):  0.9209
Вероятность накрытия м. о. (бутстрэп t-статистики):  0.9258
```

Пункт в.

При экспоненциальном распределении самым лучшим вариантом оказался бутстрэп t-статистики (с вероятностью накрытия 0.947). При распределении Стьюдента получилось, что вероятность накрытия у классического асимптотического нормального интервала - наибольшая. Однако учитывая, что в пункте а данный интервал показал худший результат, то можно сделать вывод о том, что самый лучший способ построения 95% интервала для мат. ожидания - бутстрэп t-статистики.

Ответ в) бутстрэп t-статистики.

Задача 4

df - таблица с результатами экзамена.

results - колонка со всеми результатами.

names - колонка со всеми фамилиями.

In [10]:
```
df = pd.read_csv('C://users//annat//Downloads//22-23_hse_probability - Exam.csv')
results = df['Unnamed: 72'].loc[5:]
results = results.reset_index(drop=True)
```

```
names = df['Last name'].loc[5:]
names = names.reset_index(drop=True)
```

Создадим два массива:

results_glas - результаты экзамена тех студентов, у которых фамилия начинается на гласную букву.

results_sogl - результаты экзамена тех студентов, у которых фамилия начинается на согласную букву.

In [11]:
```
#функция, чтобы определеить первую букву в фамилии
def first_letter(x):
    letter = x[0]
    return letter

fisrt_letter_vec = np.vectorize(first_letter)
all_letters = fisrt_letter_vec(np.array(names))
```

In [12]:
```
sogl = ['Б', 'В', 'Г', 'Д', 'Ж', 'З', 'Й', 'К', 'Л', 'М', 'Н', 'П', 'Р', 'С', 'Т', '
glas = ['А', 'И', 'О', 'У', 'Ы', 'Э', 'Е', 'Ё', 'Ю', 'Я']
```

In [13]:
```
glas_letters = np.where(np.isin(all_letters, glas) == True)
sogl_letters = np.where(np.isin(all_letters, sogl) == True)
```

In [14]:
```
results_glas = results.loc[glas_letters]
results_sogl = results.loc[sogl_letters]
```

Гипотезы:

H0: E(results_glas) = E(results_sogl)

H1: E(results_glas) != E(results_sogl)

Пункт а

Критерий Уэлча (предполагаем, что у выборок разные дисперсии).

Воспользуемся функцией ttest_ind из библиотеки scipy.stats.

In [15]:
```
stat, pvalue = st.ttest_ind(results_glas, results_sogl, equal_var=False)

print('P-value равно ', pvalue)

if pvalue > alpha:
    print('H0 не отвергается при уровне значимости 5%.')
else:
    print('H0 отвергается в пользу H1.')
```

```
P-value равно  0.3974027153843839
H0 не отвергается при уровне значимости 5%.
```

Пункт б

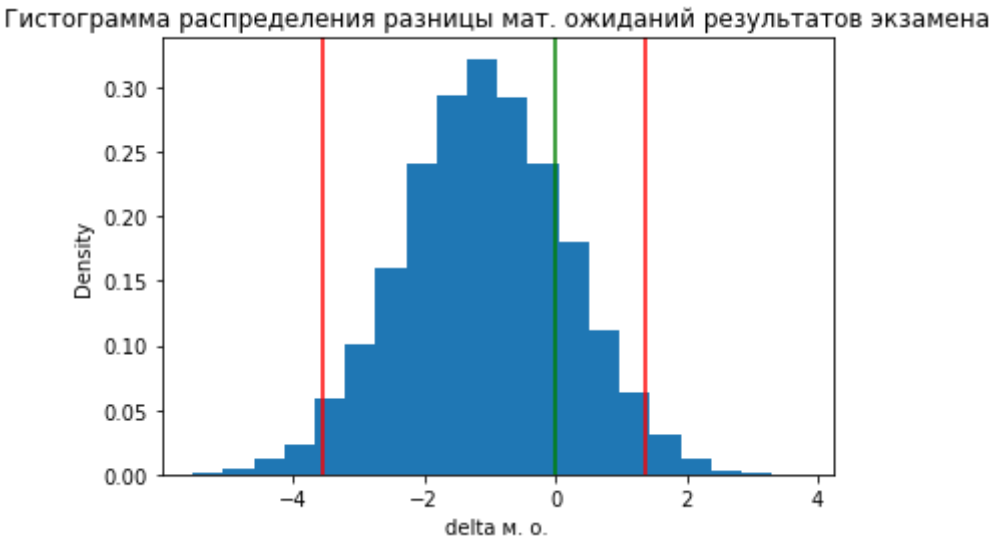
Наивный бутстрэп

In [16]:
```
boot_glas = bootstrap_sample(results_glas, 10**4)
boot_sogl = bootstrap_sample(results_sogl, 10**4)
```

In [17]:
```
#бутстрэп наивный

boot_glas_mean = np.mean(boot_glas, axis=0)
boot_sogl_mean = np.mean(boot_sogl, axis=0)

q_left_n = np.quantile(boot_glas_mean - boot_sogl_mean, alpha/2)
q_right_n = np.quantile(boot_glas_mean - boot_sogl_mean, 1-alpha/2)

# по H0 разница матожиданий == 0

print('ДИ: [', q_left_n, ',', q_right_n, ']')

if q_left_n <= 0 and q_right_n >= 0:
    print('H0 не отвергается при уровне значимости 5%.')
else:
    print('H0 отвергается в пользу H1.')

pvalue = (1 - st.norm.cdf(0))* 2
print('P-value равно ', pvalue)
if pvalue < alpha:
    print('H0 отвергается в пользу H1 при уровне значимости 0.05')
else:
    print('H0 не отвергается при уровне значимости 0.05.')
```

```
ДИ: [ -3.5281856205379656 , 1.368754957813513 ]
H0 не отвергается при уровне значимости 5%.
P-value равно  1.0
H0 не отвергается при уровне значимости 0.05.
```

In [18]:
```
plt.hist(boot_glas_mean - boot_sogl_mean, bins=20, density=True)
plt.title('Гистограмма распределения разницы мат. ожиданий результатов экзамена')
plt.xlabel('delta м. о.')
plt.ylabel('Density');
plt.axvline(q_left_n, color='r')
plt.axvline(q_right_n, color='r')
plt.axvline(0, color='g');
```



Пункт г

Перестановочный тест

df_4 - датафрейм с результатами экзамена (R) и булевыми значениями (B): 0 - если фамилия студента начинается на гласную букву, 1 - если фамилия студента начинается на согласную букву.

In [19]:
```
#перестановочный тест

df_4 = pd.DataFrame()
df_4['R'] = np.concatenate((np.array(results_glas), np.array(results_sogl)))
ones_zeros = np.concatenate((np.zeros(results_glas.shape[0]), np.ones(results_sogl.s
df_4['B'] = ones_zeros
df_4.head()
```

Out[19]:

	R	B
0	25.0	0.0
1	26.0	0.0
2	25.0	0.0
3	29.0	0.0
4	26.0	0.0

"Перемешаем" колонку "B" с помощью функции shuffle из библиотеки numpy.

In [20]:
```
arr = np.array(df_4['B'])
np.random.shuffle(arr)
df_4['B'] = arr
```

Обозначим за R0 и R1 средние результаты экзамена среди студентов с гласной и согласной буквой, соответственно.

In [21]:
```
R1 = df_4.loc[np.where(df_4['B'] == 1)].sum()
R1 = R1.loc['R'] / results_sogl.shape[0]
R0 = df_4.loc[np.where(df_4['B'] == 0)].sum()
R0 = R0.loc['R'] / results_glas.shape[0]
R1, R0
```

Out[21]:
```
(15.703180212014134, 19.102040816326532)
```

Теперь применим этот код 10^4 раз и обозначим за delta разницу R1 и R0:

In [22]:
```
delta_arr = np.array([])

for i in range(10**4):
    df_4['B'] = ones_zeros

    arr = np.array(df_4['B'])
    np.random.shuffle(arr)
    df_4['B'] = arr

    R1 = df_4.loc[np.where(df_4['B'] == 1)].sum()
    R1 = R1.loc['R'] / results_sogl.shape[0]
    R0 = df_4.loc[np.where(df_4['B'] == 0)].sum()
    R0 = R0.loc['R'] / results_glas.shape[0]
```

```
    delta = R1 - R0
    delta_arr = np.append(delta, delta_arr)

delta_arr
```

Out[22]:
```
array([ 1.50919449, -0.43008582,  0.6951756 , ..., -2.27359919,
       -0.21461023,  1.17401024])
```

Считаем квантили:

In [23]:
```
q_left = np.quantile(delta_arr, alpha/2)
q_right = np.quantile(delta_arr, 1-alpha/2)
q_left, q_right
```

Out[23]:
```
(-2.346022932141053, 2.371096848633446)
```

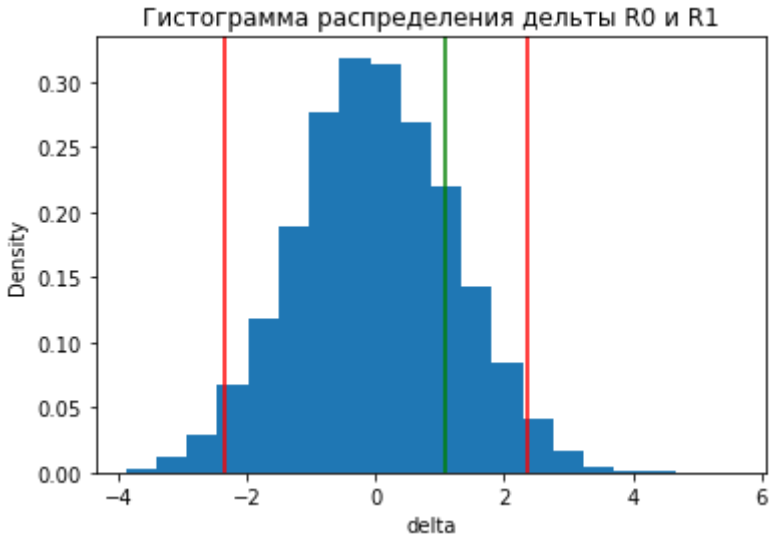
Считаем дельту (delta_1) по изначальной выборке (до перестановки).

Если delta_1 находится в интервале от левого квантиля до правого, то H0 не отвергается.

In [24]:
```
df_4['B'] = ones_zeros
R1 = df_4.loc[np.where(df_4['B'] == 1)].sum()
R1 = R1.loc['R'] / results_sogl.shape[0]
R0 = df_4.loc[np.where(df_4['B'] == 0)].sum()
R0 = R0.loc['R'] / results_glas.shape[0]

delta_1 = R1 - R0

print('Дельта равна ', delta_1)
if delta_1 > q_left and delta_1 < q_right:
    print('H0 не отвергается при уровне значимости 5%.')
else:
    print('H0 отвергается в пользу H1.')
```

```
Дельта равна  1.0782433114588574
H0 не отвергается при уровне значимости 5%.
```

In [25]:
```
plt.hist(delta_arr, bins=20, density=True)
plt.title('Гистограмма распределения дельты R0 и R1')
plt.xlabel('delta')
plt.ylabel('Density');
plt.axvline(q_left, color='r')
plt.axvline(q_right, color='r')
plt.axvline(delta_1, color='g');
```



Задача 5

Составим таблицу сопряженности.

Для начала создадим датафрейм df_5, в котором будут данные о результатах экзамена results и фамилиях студентов names.

In [26]:
```
df_5 = pd.DataFrame({'results': results, 'names': names})
df_5.head()
```

Out[26]:

	results	names
0	16.0	Репенкова
1	0.0	Ролдугина
2	19.0	Сафина
3	26.0	Сидоров
4	21.0	Солоухин

Добавим столбик с информацией о первой букве фамилии (1 - согл, 0 - глас).

In [27]:
```
df_5['letter'] = np.isin(all_letters, sogl) * 1
df_5.head()
```

Out[27]:

	results	names	letter
0	16.0	Репенкова	1
1	0.0	Ролдугина	1
2	19.0	Сафина	1
3	26.0	Сидоров	1
4	21.0	Солоухин	1

Посчитаем медиану результатов (median_res) и добавим новый столбец median в датафрейм: 1 - если результат студента выше медианы, 0 - иначе.

In [28]:
```
median_res = df_5['results'].median()

df_5['median'] = (df_5['results'] > median_res) * 1
df_5.head()
```

Out[28]:

	results	names	letter	median
0	16.0	Репенкова	1	0
1	0.0	Ролдугина	1	0
2	19.0	Сафина	1	1
3	26.0	Сидоров	1	1
4	21.0	Солоухин	1	1

Теперь создадим таблицу сопряженности (contingency_table).

In [29]:
```
sb = np.where((df_5['letter'] == 1) & (df_5['median'] == 1))[0].shape[0]
gb = np.where((df_5['letter'] == 0) & (df_5['median'] == 1))[0].shape[0]
sl = np.where((df_5['letter'] == 1) & (df_5['median'] == 0))[0].shape[0]
gl = np.where((df_5['letter'] == 0) & (df_5['median'] == 0))[0].shape[0]

contingency_table = pd.DataFrame({'result': ['больше', 'меньше'], 'согласная': [sb,
contingency_table = contingency_table.set_index('result')
contingency_table
```

Out[29]:

result	согласная	гласная
больше	145	21
меньше	138	28

Пункт а

На основе таблицы сопряженности построим 95% асимптотический интервал для отношения шансов хорошо написать экзамен.

OR_est - оценка Odds Ratio (отношение шансов).

se_log_OR - стандартная ошибка логарифма OR.

Гипотезы:

H0: OR == 1

H1: OR != 1

In [30]:
```
odds_glas = contingency_table['гласная'].loc['больше'] / contingency_table['гласная'
odds_sogl = contingency_table['согласная'].loc['больше'] / contingency_table['соглас

OR_est = odds_glas / odds_sogl
print('Оценка OR:', OR_est)

se_log_OR = (((1/((21/49)*(28/49)*49)) + (1/((145/283)*(138/283)*283)))) ** (1/2)
```

```
z_crit = st.norm.ppf(1 - alpha/2)

left = OR_est * np.exp(-z_crit*se_log_OR)
right = OR_est * np.exp(z_crit*se_log_OR)
print(f'95% асимптотический интервал: [', left, ',', right, ']')
```

```
Оценка OR: 0.7137931034482758
95% асимптотический интервал: [ 0.38709459582547806 , 1.3162172761513564 ]
```

Посчитаем p-value и проверим гипотезу.

In [31]:

```
z_obs = np.log(OR_est) / se_log_OR

pvalue = (1 - st.norm.cdf(OR_est))* 2
print('Значение p-value равно ', pvalue)

if z_obs < -z_crit or z_obs > z_crit:
    print('H0 отвергается в пользу H1 при уровне значимости 0.05')
else:
    print('H0 не отвергается при уровне значимости 0.05.')
```

```
Значение p-value равно  0.47535512483042774
H0 не отвергается при уровне значимости 0.05.
```

Пункт б

На основе таблицы сопряженности построим 95% асимптотический интервал для вероятностей хорошо написать экзамен.

RR_est - оценка Risk Ratio (отношение вероятностей).

se_log_RR - стандартная ошибка логарифма RR.

Гипотезы:

H0: RR == 1

H1: RR != 1

In [32]:

```
RR_est = (21/49)/(145/283)
print('Оценка RR:', RR_est)

se_log_RR = ((1/21) - (1/49) + (1/145) - (1/283)) ** (1/2)

left = np.exp(np.log(RR_est) - z_crit * se_log_RR)
right = np.exp(np.log(RR_est) + z_crit * se_log_RR)
print(f'95% асимптотический интервал: [', left, ',', right, ']')
```

```
Оценка RR: 0.8364532019704434
95% асимптотический интервал: [ 0.5937529565040844 , 1.1783586951819993 ]
```

Посчитаем p-value и проверим гипотезу.

In [33]:

```
z_obs = np.log(RR_est) / se_log_RR

pvalue = (1 - st.norm.cdf(RR_est))* 2
print('Значение p-value равно ', pvalue)

if z_obs < -z_crit or z_obs > z_crit:
    print('H0 отвергается в пользу H1 при уровне значимости 0.05')
```

```
else:
    print('H0 не отвергается при уровне значимости 0.05.')
```

```
Значение p-value равно  0.4028999934344841
H0 не отвергается при уровне значимости 0.05.
```

Пункт в

На основе таблицы сопряженности построим 95% интервал для отношения шансов хорошо написать экзамен с помощью наивного бутстрэпа.

Гипотезы:

H0: OR == 1

H1: OR != 1

Чтобы использовать бутстрэп, разделим выборку на две части: результаты студентов с первой согласной и гласной буквой в фамилии (df_5_c_sogl и df_5_c_glas соответственно). "Пробутстрапируем" каждую из них по отдельности, а затем соединим.

In [34]:
```
df_5_c = df_5[['results', 'letter']]
df_5_c_sogl = df_5_c.loc[np.where(df_5_c['letter'] == 1)[0]]
df_5_c_glas = df_5_c.loc[np.where(df_5_c['letter'] == 0)[0]]

boot_sogl_c = bootstrap_sample(np.array(df_5_c_sogl['results']), 10**4)
boot_glas_c = bootstrap_sample(np.array(df_5_c_glas['results']), 10**4)

boot_5_c = np.concatenate((boot_sogl_c, boot_glas_c), axis=0).T
```

Для каждой бутстрэп выборки найдем OR.

In [35]:
```
all_OR_est = np.array([])

for i in boot_5_c:
    median_res = np.median(i)

    s = i[:df_5_c_sogl.shape[0]]
    g = i[df_5_c_sogl.shape[0]+1:]
    sb = s[np.where(s > median_res)].shape[0]
    sl = s[np.where(s <= median_res)].shape[0]
    gb = g[np.where(g > median_res)].shape[0]
    gl = g[np.where(g <= median_res)].shape[0]
    n_g = gb + gl
    n_s = sb + sl

    OR_est_ = (gb / gl) / (sb / sl)
    all_OR_est = np.append(all_OR_est, OR_est_)
```

Посчитаем квантили на основе всех OR.

In [36]:
```
q_left = np.quantile(all_OR_est, alpha/2)
q_right = np.quantile(all_OR_est, 1-alpha/2)
print(f'95% интервал: [', q_left, ',', q_right, ']')
```

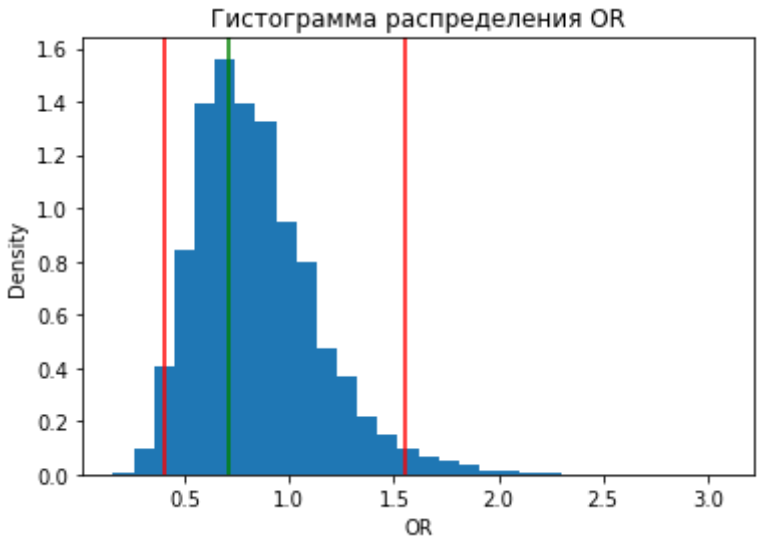
```
95% интервал: [ 0.4031262854792267 , 1.556919642857143 ]
```

Теперь определим p-value и проверим гипотезу.

In [38]:
```
print('Значение p-value равно ', (1 - st.norm.cdf(OR_est))* 2)

if q_left <= OR_est and q_right >= OR_est:
    print('H0 не отвергается при уровне значимости 5%.')
else:
    print('H0 отвергается в пользу H1.')
```

```
Значение p-value равно  0.47535512483042774
H0 не отвергается при уровне значимости 5%.
```

In [39]:
```
plt.hist(all_OR_est, bins=30, density=True);
plt.title('Гистограмма распределения OR')
plt.xlabel('OR')
plt.ylabel('Density');
plt.axvline(q_left, color='r');
plt.axvline(q_right, color='r');
plt.axvline(OR_est, color='g');
```



Задача 6

Для работы в задаче 6 будем использовать датафрейм df_6 (он идентичен df_5).

In [40]:
```
df_6 = pd.DataFrame({'results': results, 'names': names})
```

Функция length считает число букв в фамилии студента.

In [41]:
```
def length(x):
    return len(x)
```

Применим эту функцию и создадим новый столбик в датафрейме 'len', в котором будет информация о числе букв в фамилии студента.

In [42]:
```
length_func = np.vectorize(length)
name_length = length_func(df_6['names'])
df_6['len'] = name_length
df_6.head()
```

Out[42]:

	results	names	len
0	16.0	Репенкова	9
1	0.0	Ролдугина	9
2	19.0	Сафина	6
3	26.0	Сидоров	7
4	21.0	Солоухин	8

Пункт а

Как оценить бета (b)?

$$E(Y_i) = b \cdot F_i$$

$$E(Y_i) = E(Y_{mean})$$

$$\beta \cdot F_i/n = E(Y_{mean})$$

$$\beta(mm) = Y_{mean}/F_{mean}$$

In [43]:
```
b_mm = df_6['results'].mean() / df_6['len'].mean()
print(f'Оценка параметра beta методом моментов равна ', b_mm)
```

```
Оценка параметра beta методом моментов равна  2.0613026819923372
```

Посчитаем выборочную корреляцию (r_sample).

In [44]:
```
r_up_1 = ((df_6['results'] - df_6['results'].mean())*(df_6['len'] - df_6['len'].mean
r_down_1 = (((df_6['results'] - df_6['results'].mean()) ** 2).sum()*((df_6['len'] -

r_sample = r_up_1 / r_down_1
print('Выборочная корреляция равна', r_sample)
```

```
Выборочная корреляция равна 0.025328052669147696
```

Пункт б

Гипотезы:

H0: r_sample == 0

H1: r_sample != 0

Воспользуемся перестановочным тестом 10000 раз, все значения выборочной корреляции сохраним в массиве all_r:

In [45]:
```
df_6 = df_6[['results', 'len']]
all_r = np.array([])

for i in range(10**4):
    len_shuf = np.array(df_6['len'])
    np.random.shuffle(len_shuf)

    r_up = ((df_6['results'] - df_6['results'].mean())*(len_shuf - len_shuf.mean()))
    r_down = (((df_6['results'] - df_6['results'].mean()) ** 2).sum()*((len_shuf - l
```

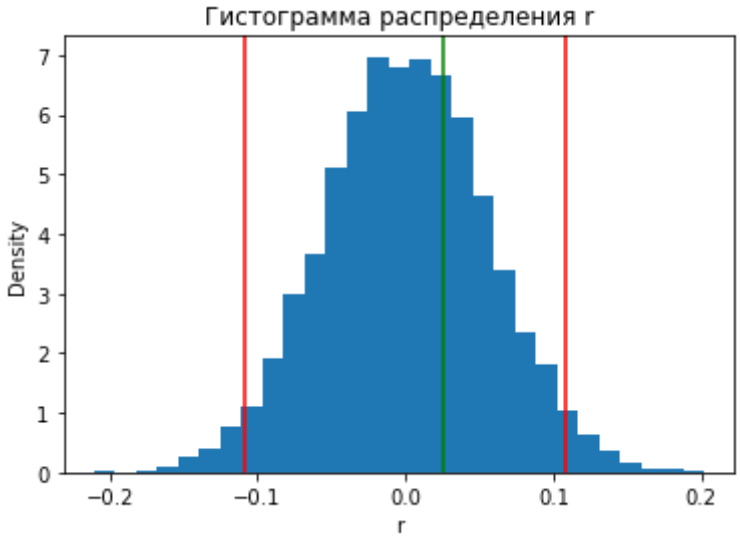
```
    r = r_up / r_down
    all_r = np.append(all_r, r)
```

Посчитаем квантили:

In [46]:
```
left = np.quantile(all_r, alpha/2)
right = np.quantile(all_r, 1-alpha/2)
print('Интервал: [', left, ',', right, ']')
```

```
Интервал: [ -0.10871233583093422 , 0.10846702781476811 ]
```

In [47]:
```
plt.hist(all_r, bins=29, density=True);
plt.title('Гистограмма распределения r')
plt.xlabel('r')
plt.ylabel('Density');
plt.axvline(r_sample, color='g');
plt.axvline(left, color='r');
plt.axvline(right, color='r');
```



Найдем p-value и протестируем гипотезу.

In [48]:
```
pvalue = (1 - st.norm.cdf(r_sample))* 2
print('Значение p-value равно ', pvalue)
if pvalue < alpha:
    print('H0 отвергается в пользу H1 при уровне значимости 0.05')
else:
    print('H0 не отвергается при уровне значимости 0.05.')
```

```
Значение p-value равно  0.9797932983108961
H0 не отвергается при уровне значимости 0.05.
```

Задача 7

Решалась вторая задача из первой контрольной работы по теории вероятностей для ИП.

В таверне «Крутой ковбой» разбавленный ром подают с вероятностью 1/2. Джо немного сменил свой характер и теперь устраивает скандал, если две пинты рома подряд разбавлены. Какова вероятность того, что Джо сможет выпить 100 пинт подряд без скандалов?

Поскольку вероятность того, что ром разбавленный, равна 1/2, то вероятность того, что он крепкий, также равна 1/2.

Для того, чтобы Джо выпил 100 пинт подряд без скандалов, каждая из 100 пинт должна быть либо крепкой, либо разбавленной, но не две разбавленные подряд.

Вероятность того, что первая пинта будет крепкой, равна 1/2. Вероятность того, что вторая пинта будет крепкой, если первая была крепкой, также равна 1/2. Вероятность того, что вторая пинта будет крепкой, если первая была разбавленной, равна 1/2. Однако, если первая была разбавленной, то третья пинта должна быть крепкой, чтобы избежать скандала.

Таким образом, вероятность того, что Джо выпьет 100 пинт подряд без скандалов, равна:

(1/2) * (1/2) * (1/2) * (1/2) * ... * (1/2) * (1/2) = (1/2)^99 = 7.9 x 10^-31

То есть, эта вероятность крайне мала, практически равна нулю.

Попробуй решить через задачу Коши

Для решения задачи через задачу Коши нам нужно определить вероятность p_n того, что Джо выпьет n пинт подряд без скандалов.

Если первая пинта крепкая, то Джо может выпить еще $n-1$ пинт без скандалов, что происходит с вероятностью $1/2 \cdot p_{n-1}$.

Если первая пинта разбавленная, то вторая пинта должна быть крепкой, а затем Джо может выпить еще $n-2$ пинт без скандалов, что происходит с вероятностью $1/2 \cdot 1/2 \cdot p_{n-2}$.

Таким образом, мы можем записать задачу Коши:

$$\begin{cases} p_1 = 1/2 \\ p_2 = 1/2 \\ p_n = 1/2 \cdot p_{n-1} + 1/4 \cdot p_{n-2}, \quad n \geq 3 \end{cases}$$

Теперь мы можем решить эту задачу рекурсивно, используя формулу для p_n:

$$p_3 = 1/2 \cdot p_2 + 1/4 \cdot p_1 = 5/16$$
$$p_4 = 1/2 \cdot p_3 + 1/4 \cdot p_2 = 11/32$$
$$p_5 = 1/2 \cdot p_4 + 1/4 \cdot p_3 = 21/64$$
$$p_6 = 1/2 \cdot p_5 + 1/4 \cdot p_4 = 43/128$$
$$\vdots$$
$$p_{100} = 1/2 \cdot p_{99} + 1/4 \cdot p_{98} \approx 3.5 \cdot 10^{-31}$$

Таким образом, вероятность того, что Джо сможет выпить 100 пинт подряд без скандалов, очень мала и составляет примерно $3.5 \cdot 10^{-31}$.

Задача 8

Очень полезный источник, которым я пользовалась на протяжении всего прошлого полугодия во время прохождения тервера - это HTML-версия лекций Натальи Черновой, доцента кафедры теории вероятностей и математической статистики НГУ.

Ссылка на лекции: https://tvims.nsu.ru/chernova/tv/lec/lec.html

На эти конспекты я наткнулась случайно, когда готовилась к первой контрольной работе, и именно из них я брала всю информацию, которая нужна была для ответа на задачи минимума. Более того, этот источник стал отличным дополнением к лекциям Елены Владимировны.