

```

In [192]: import pandas as pd
import numpy as np
import scipy.stats as sts

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
%matplotlib inline

from collections import defaultdict

In [193]: # Отключение некоторых лишних предупреждений
import warnings
warnings.filterwarnings("ignore")

In [194]: alpha = 0.05

In [195]: # С семинаров по анализу данных: https://github.com/hse-econ-data-science/andan_2023/blob/main/sem13_bootstrap/sem13_python_bootstrap.ipynb
def get_bootstrap_sample(x, B_sample=1):
    np.random.seed(10)
    N = x.size
    sample = np.random.choice(x, size=(N, B_sample), replace=True)

    if B_sample == 1:
        sample = sample.T[0]
    return sample

```

Номер 1

Пункт а

Заметим, что вероятность встретить первого водителя новым равна 1, вероятность встретить такого же равна $\frac{1}{n}$, а вероятность встретить других водителей равна $\frac{n-1}{n}$, $\frac{n-2}{n}$, $\frac{n-3}{n}$ и т.д. Тогда, функция правдоподобия будет иметь вид

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

```

In [320]: n = np.arange(sts.randint.ppf(0.001, 1, 1000), sts.randint.ppf(0.999, 1, 1000))

```

```

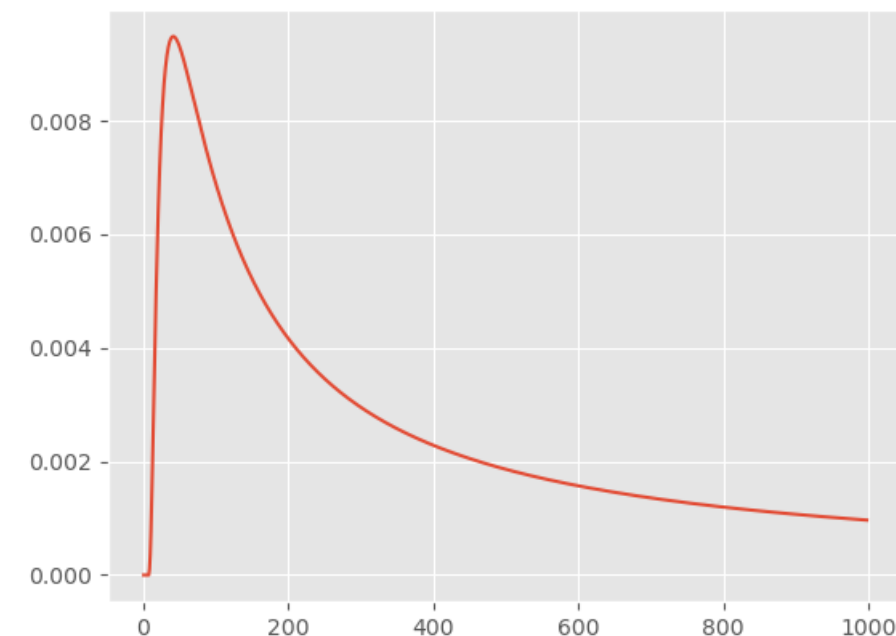
L = ((n-1)*(n-2)*(n-3)*(n-4)*(n-5)*(n-6)*(n-7)*(n-8))/(n**9)
lnL = np.log(L)

```

```

plt.plot(n, L);

```



```

In [345]: import scipy.optimize
def f(x): return ((x-1)*(x-2)*(x-3)*(x-4)*(x-5)*(x-6)*(x-7)*(x-8))/(x**9)
max_x = scipy.optimize.fmin(lambda x: -f(x), 300)
max_x[0]

```

```

Optimization terminated successfully.
Current function value: -0.009473
Iterations: 26
Function evaluations: 52

```

```

Out[345]: 41.771392822265625

```

Оптимальное значение получилось 41.77, однако число таксистов может быть только целым - оптимальный $n = 42$

Номер 3

Пункт а

```
In [196]: np.random.seed(10)
n_obs = 10 ** 4
x = np.random.exponential(1, (20, n_obs))
x_mean = np.mean(x, axis=0)
x_std = np.std(x, axis=0)
sigma = 1
x_std

Out[196]:array([1.28187021, 1.0195338 , 0.85609345, ..., 0.81873017, 1.27613944,
0.79606485])

In [197]: # Классический доверительный интервал
def asci(x_mean, sigma = 1, alpha = 0.05):
    left1 = x_mean - sts.norm.ppf(1 - alpha/2) * sigma / np.sqrt(20)
    right1 = x_mean + sts.norm.ppf(1 - alpha/2) * sigma / np.sqrt(20)
    #print('Асимптотический нормальный ДИ: [{:.4f}, {:.4f}].format(left, right))
    return left1, right1
```

Экспоненциальное распределение (истинное матожидание равно 1)

```
In [198]: ci = asci(x_mean, sigma = x_std)
count = 0
a = (ci[0] < 1) * 1
b = (ci[1] > 1) * 1
for i in range(len(a)):
    if a[i] == 1 and b[i] == 1:
        count += 1
print('Вероятность накрытия МО найденным доверительным интервалом составляет {:.4f}'.format(count / n_obs))
```

Вероятность накрытия МО найденным доверительным интервалом составляет 0.8954

```
In [199]: count_boot = 0
count_boot_t = 0
for col in x.T:
    # наивный бутстрап
    boot = get_bootstrap_sample(col, B_sample=10**4)
    boot_m = np.mean(boot, axis=0)
    left = np.quantile(boot_m, alpha/2)
    right = np.quantile(boot_m, 1-alpha/2)
    if left < 1 and right > 1:
        count_boot += 1

    # бутстрап t-статистики
    theta_hat = np.mean(col)
    std_hat = np.std(col)
    x_boot_t = np.mean(boot - theta_hat, axis=0)
    x_boot_t = x_boot_t/np.std(boot, axis=0)
    left_t = theta_hat - np.quantile(x_boot_t, 1-alpha/2)*std_hat
    right_t = theta_hat + np.quantile(x_boot_t, alpha/2)*std_hat
    if left_t < 1 and right_t > 1:
        count_boot_t += 1
print(count_boot)
print(count_boot_t)
```

9040
9479

```
In [200]: print('Вероятность накрытия МО, найденным через наивный бутстрап ДИ, составляет {:.4f}'.format(count_boot / n_obs))
```

Вероятность накрытия МО, найденным через наивный бутстрап ДИ, составляет 0.9040

```
In [201]: print('Вероятность накрытия МО, найденным через t-бутстрап ДИ, составляет {:.4f}'.format(count_boot_t / n_obs))
```

Вероятность накрытия МО, найденным через t-бутстрап ДИ, составляет 0.9479

Распределение Стьюдента с 3 степенями свободы (истинное матожидание равно 0)

```
In [202]: # пункт б
np.random.seed(15)
n_obs = 10 ** 4
x_t = np.random.standard_t(3, (20, n_obs))
xt_mean = np.mean(x_t, axis=0)
xt_std = np.std(x_t, axis=0)
sigma_t = np.sqrt(3 / (3 - 2)) # по формуле корень из дисперсии для распределения стьюдента
xt_std
```

```
Out[202]:array([1.76525292, 1.70810387, 2.15338803, ..., 1.67475835, 2.4928731 ,
1.16968196])
```

```
In [411]: ci_t = asci(xt_mean, sigma = xt_std)
count_t = 0
a = (ci_t[0] < 0) * 1
b = (ci_t[1] > 0) * 1
for i in range(len(a)):
```

```
if a[i] == 1 and b[i] == 1:
    count_t += 1
print('Вероятность накрытия МО найденным доверительным интервалом составляет {:.4f}'.format(count_t / n_obs))
```

Вероятность накрытия МО найденным доверительным интервалом составляет 0.9336

```
In [204]: count_boot_2 = 0
count_boot_t_2 = 0
for col in x_t.T:
    # наивный бутстрап
    boot = get_bootstrap_sample(col, B_sample=10**4)
    boot_m = np.mean(boot, axis=0)
    left = np.quantile(boot_m, alpha/2)
    right = np.quantile(boot_m, 1-alpha/2)
    if left < 0 and right > 0:
        count_boot_2 += 1

    # бутстрап t-статистики
    theta_hat = np.mean(col)
    std_hat = np.std(col)
    x_boot_t = np.mean(boot - theta_hat, axis=0)
    x_boot_t = x_boot_t/np.std(boot, axis=0)
    left_t = theta_hat - np.quantile(x_boot_t, 1-alpha/2)*std_hat
    right_t = theta_hat - np.quantile(x_boot_t, alpha/2)*std_hat
    if left_t < 0 and right_t > 0:
        count_boot_t_2 += 1
print(count_boot_2)
print(count_boot_t_2)
```

9158

9211

```
In [182]: x_boot_t
```

```
Out[182]: array([-0.06562994, -0.10638572, -0.04418342, ..., -0.11054463,
                 0.0615957 , 0.25271073])
```

```
In [205]: print('Вероятность накрытия МО, найденным через наивный бутстрап ДИ, составляет {:.4f}'.format(count_boot_2 / n_obs))
```

Вероятность накрытия МО, найденным через наивный бутстрап ДИ, составляет 0.9158

```
In [206]: print('Вероятность накрытия МО, найденным через t-бутстрап ДИ, составляет {:.4f}'.format(count_boot_t_2 / n_obs))
```

Вероятность накрытия МО, найденным через t-бутстрап ДИ, составляет 0.9211

Выводы

По результатам всех симуляций, для экспоненциального распределения лучше всего сработал бутстрап t-статистики, тогда как асимптотический ДИ оказался самым слабым. Однако вероятность накрытия для наивного бутстрапа незначительно выше, чем для асимптотического интервала.

Для распределения Стьюдента самым лучшим оказался асимптотический ДИ, а бутстрап t-статистики лучше, чем наивный бутстрап, однако для обоих бутстрапов метрики почти равны

Номер 4

```
In [207]: df_students = pd.read_excel('probability.xlsx')
df_students.head()
```

```
Out[207]:
```

	Last name	Score
0	Абдугаффорова	17
1	Абдулаева	21
2	Аврамидис	0
3	Аврамчук	29
4	Авсеенко	26

```
In [31]: df_students.shape
```

```
Out[31]: (332, 2)
```

```
In [77]: score_median = df_students.median()
score_median[0]
```

```
Out[77]: 17.5
```

Разделим выборку на две по первым буквам фамилии

И посмотрим на распределение оценок в каждой из них

```
In [208]: glas = ['А', 'Е', 'Ё', 'И', 'О', 'У', 'Ы', 'Э', 'Ю', 'Я']
sogl = ['Б', 'В', 'Г', 'Д', 'Ж', 'З', 'Й', 'К', 'Л', 'М', 'Н', 'П', 'Р', 'С', 'Т', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ']
stud_glas = df_students.loc[df_students['Last name'].str.startswith(tuple(glas))]
stud_sogl = df_students.loc[df_students['Last name'].str.startswith(tuple(sogl))]
```

```
In [209]: stud_glas.shape # всего 49 человек, у которых фамилия начинае тся на гласную
```

```
Out[209]: (49, 2)
In [210]: stud_sogl.shape # 283 человека, у которых фамилия начинается на согласную
```

```
Out[210]: (283, 2)
```

```
In [43]: stud_glas.plot.hist(bins=15)
plt.xlabel('Количество баллов', fontsize=10)
plt.ylabel('Количество студентов', fontsize=10);
plt.title('Распределение баллов у студентов, с фамилий на гласную', fontsize=10);
```



```
In [44]: stud_sogl.plot.hist(bins=25)
plt.xlabel('Количество баллов', fontsize=10)
plt.ylabel('Количество студентов', fontsize=10);
plt.title('Распределение баллов у студентов, с фамилий на согласную', fontsize=10);
```



Тест Уэлча

```
In [113]: from scipy.stats import ttest_ind
res1 = ttest_ind(stud_glas['Score'], stud_sogl['Score'], equal_var = False)
res1

Out[113]: Ttest_indResult(statistic=-0.8519661870595602, pvalue=0.39740271538438365)
$alpha < pvalue$, значит, нулевая гипотеза не отвергается
```

Наивный бутстрап

```
In [211]: gl_boot = get_bootstrap_sample(stud_glas['Score'], 10 ** 6)
so_boot = get_bootstrap_sample(stud_sogl['Score'], 10 ** 6)
```

```
In [397]: gl_boot_m = np.mean(gl_boot, axis=0)
so_boot_m = np.mean(so_boot, axis=0)
obs_mean = np.mean(stud_sogl)['Score'] - np.mean(stud_glas)['Score']
```

```
In [213]: diff_boot_m = so_boot_m - gl_boot_m
left4b = np.quantile(diff_boot_m, alpha/2)
right4b = np.quantile(diff_boot_m, 1-alpha/2)
```

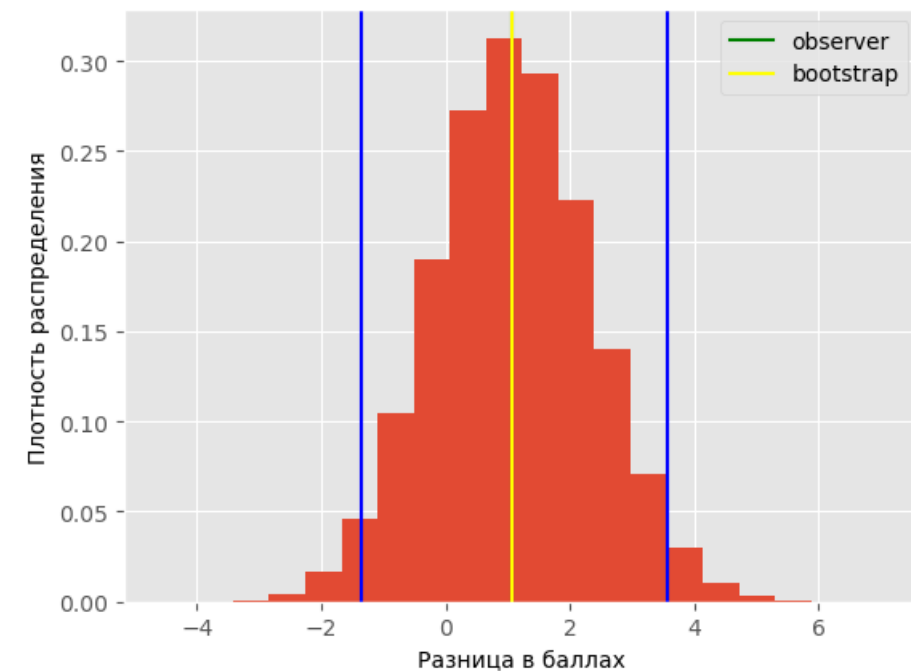
```
In [403]: obs_mean
```

```
Out[403]: 1.0782433114588574
```

```
In [404]: diff_boot_m.mean()
```

```
Out[404]: 1.0787444207831547
```

```
In [412]: plt.hist(diff_boot_m, bins=20, density=True)
plt.axvline(left4b, color='b')
plt.axvline(right4b, color='b')
plt.axvline(obs_mean, color='g', label='observer')
plt.axvline(diff_boot_m.mean(), color='yellow', label='bootstrap')
plt.xlabel('Разница в баллах', fontsize=10)
plt.ylabel('Плотность распределения', fontsize=10);
plt.legend();
```



```
In [425]: p_value = 2 * len(diff_boot_m[diff_boot_m < 0]) / 10 ** 6
print('p_value равно {:.4f}'.format(p_value))
```

p_value равно 0.3923

Вывод: для $\alpha = 0.05$ нулевая гипотеза не отвергается

Бутстреп t-статистики

```
In [391]: theta_hat = np.mean(stud_sogl)['Score'] - np.mean(stud_glas)['Score']
var_ = np.var(stud_glas['Score']) / stud_glas.shape[0] + np.var(stud_sogl['Score']) / stud_sogl.shape[0]
std_hat = np.sqrt(var_)
x_boot_t = diff_boot_m - theta_hat
x_boot_t = x_boot_t / std_hat
```

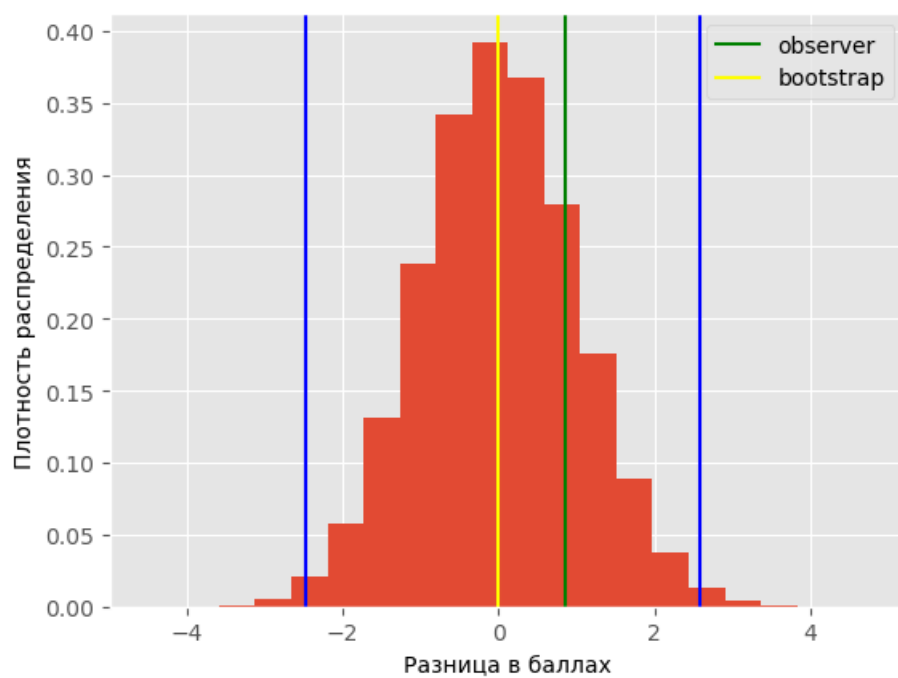
```
left_t_4c = np.quantile(x_boot_t, 1 - alpha/2)
right_t_4c = np.quantile(x_boot_t, alpha/2)
```

```
t_obs = theta_hat / (std_hat)
```

```
left_t_4c, right_t_4c
t_obs
```

```
Out[391]: 0.8597828196161024
```

```
In [414]: plt.hist(x_boot_t, bins=20, density=True)
plt.axvline(left_t_4c, color='b')
plt.axvline(right_t_4c, color='b')
plt.axvline(t_obs, color='g', label='observer')
plt.axvline(x_boot_t.mean(), color='yellow', label='bootstrap')
plt.xlabel('Разница в баллах', fontsize=10)
plt.ylabel('Плотность распределения', fontsize=10)
plt.legend();
```



```
In [394]: t_larger = len(x_boot_t[x_boot_t > t_obs]) / 10 ** 6
t_smaller = len(x_boot_t[x_boot_t < t_obs]) / 10 ** 6
p_value = 2 * min(t_larger, t_smaller)
print('p_value равно {:.4f}'.format(p_value))

p_value равно 0.3898
Вывод: для  $\alpha = 0.05$  нулевая гипотеза не отвергается
```

Перестановочный тест

```
In [66]: glas_perm = pd.DataFrame({'Score': stud_glas['Score'], 'num': np.zeros(stud_glas.shape[0])}).reset_index(drop = True)
sogl_perm = pd.DataFrame({'Score': stud_sogl['Score'], 'num': np.ones(stud_sogl.shape[0])}).reset_index(drop = True)

In [67]: df_perm = glas_perm
df_perm = df_perm.append(sogl_perm)
```

```
In [418]: df_perm.head()
```

Out[418]:

	Score	num
0	17	0.0
1	21	0.0
2	0	0.0
3	29	0.0
4	26	0.0

```
In [419]: df_perm.tail()
```

Out[419]:

	Score	num
278	23	1.0
279	11	1.0
280	23	1.0
281	15	1.0
282	26	1.0

```
In [68]: def mean_perm(x, nx = stud_glas.shape[0], ny = stud_sogl.shape[0]):
diff = 0
diff = x[x['num'] == 1]['Score'].sum()/ny - x[x['num'] == 0]['Score'].sum()/nx
return diff
```

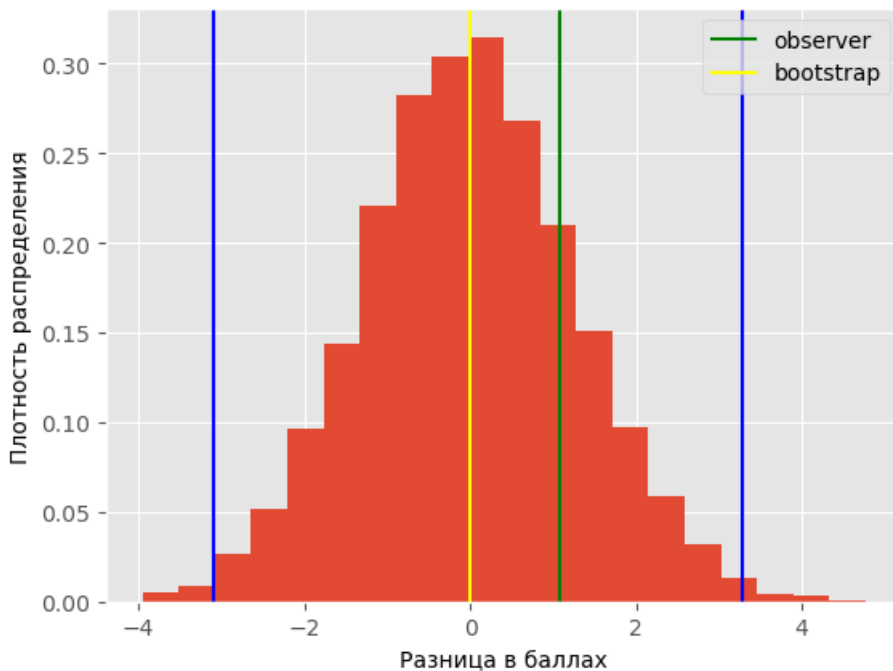
```
In [420]: diff_hat = mean_perm(df_perm)
diff_hat
```

Out[420]:1.0782433114588574

```
In [69]: import random
n = list(df_perm['num'])
sc = df_perm['Score']
diff_perm = [0] * 10 ** 4
d = 0
for i in range(10 ** 4):
    random.seed(10)
    random.shuffle(n)
    perm = pd.DataFrame({'Score': sc, 'num': n})
```

```
d = mean_perm(perm)
diff_perm[i] = d
```

```
In [422]: left4d = np.quantile(diff_perm, alpha/2)
right4d = np.quantile(diff_perm, 1-alpha/2)
plt.hist(diff_perm, bins=20, density=True)
plt.axvline(left4d, color='b')
plt.axvline(right4d, color='b')
plt.axvline(diff_hat, color = 'g', label = 'observer')
plt.axvline(np.mean(diff_perm), color = 'yellow', label = 'bootstrap')
plt.xlabel('Разница в баллах', fontsize=10)
plt.ylabel('Плотность распределения', fontsize=10);
plt.legend();
```



```
In [423]: p_value = 2 * len([i for i in diff_perm if i > diff_hat]) / 10 ** 4
print('p_value равно {:.4f}'.format(p_value))
```

p_value равно 0.3892

Вывод: для $\alpha = 0.05$ нулевая гипотеза не отвергается

№5

```
In [226]: gl_larger = len(stud_glas[stud_glas['Score'] > score_median[0]])
gl_smaller = len(stud_glas[stud_glas['Score'] < score_median[0]])
sogl_larger = len(stud_sogl[stud_sogl['Score'] > score_median[0]])
sogl_smaller = len(stud_sogl[stud_sogl['Score'] < score_median[0]])
table = pd.DataFrame({'Имя': ['Гласная', 'Согласная'], 'Больше': [gl_larger, sogl_larger], 'Меньше': [gl_smaller, sogl_smaller]})
table
```

```
Out[226]:
```

	Имя	Больше	Меньше
0	Гласная	21	28
1	Согласная	145	138

Пункт а

На семинаре 25.04 была выведена формула для расчета ДИ отношения шансов

```
In [230]: chance_gl = gl_larger / gl_smaller
chance_sogl = sogl_larger / sogl_smaller
OR = chance_gl / chance_sogl
se = np.sqrt(1 / gl_larger + 1 / gl_smaller + 1 / sogl_larger + 1 / sogl_smaller)
left5a = OR * np.exp(- 1.96 * se)
right5a = OR * np.exp(1.96 * se)
left5a, right5a
```

```
Out[230]: (0.38709024318230967, 1.3162320763800786)
```

```
In [128]: print('Асимптотический ДИ: [{:.4f}, {:.4f}].format(left5a, right5a))
```

Асимптотический ДИ: [0.3871, 1.3162]

Вывод: 1 попадает в доверительный интервал, значит, гипотеза не отвергается

```
In [232]: p_value = (1 - sts.norm.cdf(OR)) * 2
print('p_value равно {:.4f}'.format(p_value))
```

p_value равно 0.4754

Данное значение p-value подтверждает вывод о том, что при $\alpha = 0.05$ нулевая гипотеза не отвергается

Пункт б

Аналогично выводится интервал для отношения вероятностей (отношения рисков)
(https://influentialpoints.com/Training/confidence_intervals_of_risk_ratio_odds_ratio_and_rate_ratio-principles-properties-assumptions.htm)

```
In [103]: p_gl = gl_larger / (gl_larger + gl_smaller)
          p_sogl = sogl_larger / (sogl_larger + sogl_smaller)
          RR = p_gl / p_sogl
          RR

Out[103]:0.8364532019704434

In [104]: se_rr = np.sqrt(1 / gl_larger - 1 / (gl_larger + gl_smaller) + 1 / sogl_larger - 1 / (sogl_larger + sogl_smaller))
          se_rr

Out[104]:0.17485384517729596

In [129]: left5b = RR * np.exp(- 1.96 * se_rr)
          right5b = RR * np.exp(1.96 * se_rr)
          left5b, right5b

Out[129]:(0.5937492173912553, 1.1783661158504992)

In [130]: print('Асимптотический ДИ: [{:.4f}, {:.4f}'].format(left5b, right5b))

Асимптотический ДИ: [0.5937, 1.1784]
1 принадлежит ДИ, значит, нулевая гипотеза не отвергается

In [220]: p_value = (1 - sts.norm.cdf(RR)) * 2
          print('p_value равно {:.4f}'.format(p_value))

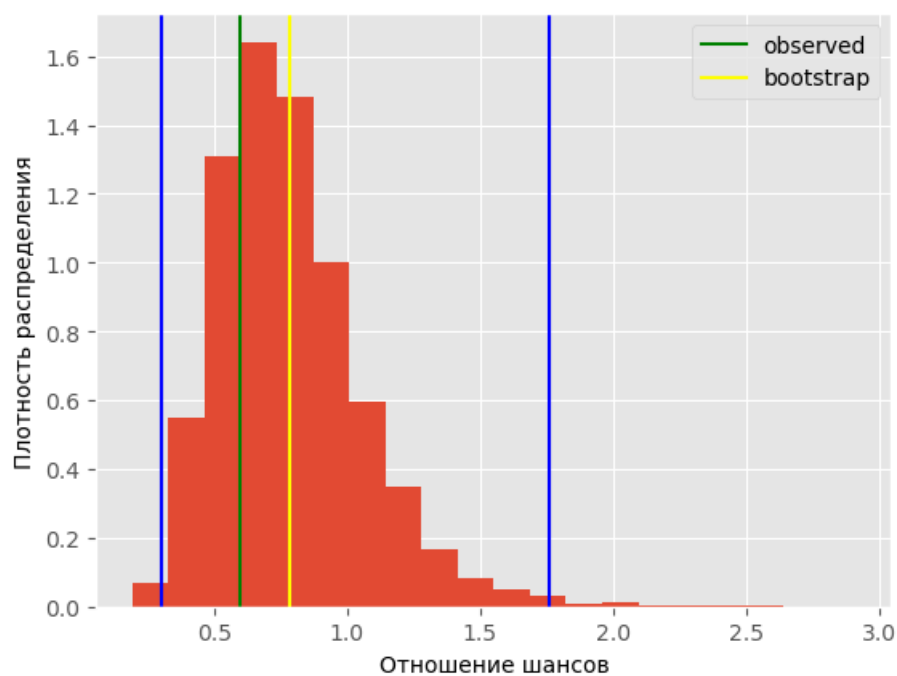
p_value равно 0.4029
p-value $> \alpha$, это подтверждает вывод о том, что нулевая гипотеза не отвергается
```

Пункт в

```
In [364]: boot_sogl = get_bootstrap_sample(stud_sogl['Score'], 10 ** 4)
          boot_gl = get_bootstrap_sample(stud_glas['Score'], 10 ** 4)
          new_df = np.concatenate ((boot_sogl, boot_gl))
          score_median_new = np.median(new_df, axis = 0)

          OR_boot = [0] * 10 ** 4
          for i in range(10 ** 4):
              gl_larger = len(boot_gl[:, i][boot_gl[:, i] > score_median_new[i]])
              gl_smaller = len(boot_gl[:, i][boot_gl[:, i] < score_median_new[i]])
              sogl_larger = len(boot_sogl[:, i][boot_sogl[:, i] > score_median_new[i]])
              sogl_smaller = len(boot_sogl[:, i][boot_sogl[:, i] < score_median_new[i]])
              chance_gl = gl_larger / gl_smaller
              chance_sogl = sogl_larger / sogl_smaller
              OR = chance_gl / chance_sogl
              OR_boot[i] = OR

In [427]: left5c = np.quantile(OR_boot, alpha/2)
          right5c = np.quantile(OR_boot, 1-alpha/2)
          plt.hist(OR_boot, bins=20, density=True)
          plt.axvline(left5c, color='b')
          plt.axvline(right5c, color='b')
          plt.axvline(OR, color = 'g', label = 'observed')
          plt.axvline(np.mean(OR_boot), color = 'yellow', label = 'bootstrap')
          plt.xlabel('Отношение шансов', fontsize=10)
          plt.ylabel('Плотность распределения', fontsize=10)
          plt.legend();
```

```
In [168]: print('Бутстреп ДИ: [{:.4f}, {:.4f}'].format(left5c, right5c))
```

Бутстреп ДИ: [0.3781, 1.2895]
 Нулевая гипотеза не отвергается, так как 1 принадлежит ДИ

```
In [377]: p_value = len([i for i in OR_boot if i > OR]) / 10 ** 4
print('p_value равно {:.4f}'.format(p_value))
```

p_value равно 0.7453
 Вывод о неотвержении нулевой гипотезы подтверждается

Номер 6

Пункт а

```
In [108]: Y = df_students['Score']
Y_mean = df_students['Score'].mean()
Y_mean

Out[108]:16.204819277108435

In [109]: len_sum = df_students['Last name'].apply(lambda x: len(x))
mean_len = len_sum.mean()
sum(len_sum)
```

Out[109]:2610
 Известно, что $E(Y_i) = E(Y_{\text{mean}})$ и $E(Y_{\text{mean}}) = Y_{\text{mean}}$

Значит, $E(Y_i) = \frac{1}{n} * E(\sum_1^n Y_i) = \frac{1}{n} * \sum_1^n E(Y_i) = \frac{1}{n} * \beta * \sum F_i$
 $\frac{1}{n} * \beta * \sum_1^n F_i = Y_{\text{mean}}$

```
In [110]: beta = Y_mean * df_students.shape[0] / sum(len_sum)
beta
```

```
Out[110]:2.0613026819923372

In [111]: corr = sum((len_sum - mean_len) * (Y - Y_mean)) / np.sqrt(sum((len_sum - mean_len) ** 2) * sum((Y - Y_mean) ** 2))
corr
```

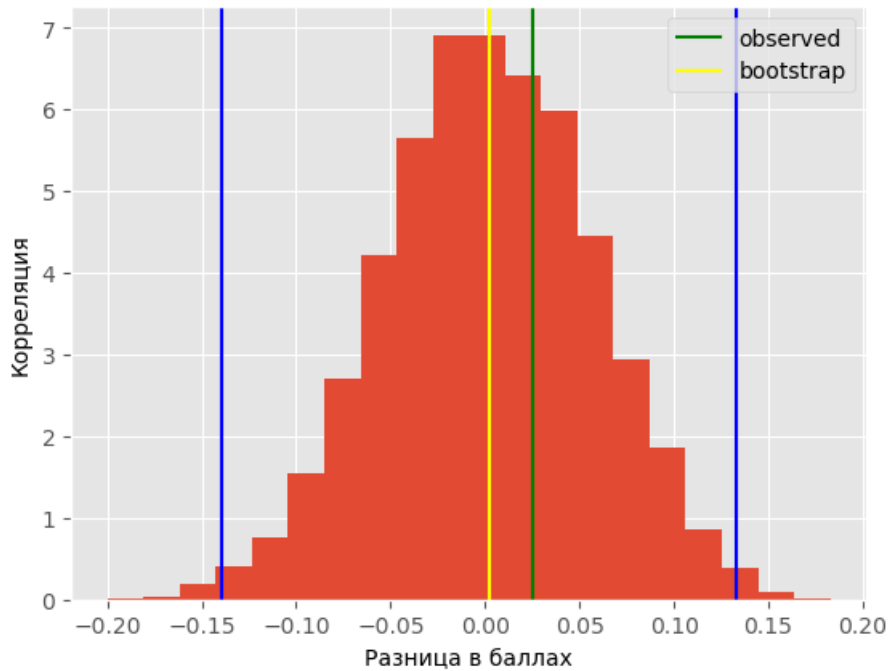
Out[111]:0.025328052669147626

Пункт б

```
In [235]: import random
n = list(len_sum)
sc = Y
corr_perm = [0] * 10 ** 4
d = 0
for i in range(10 ** 4):
    random.seed(10)
    random.shuffle(n)
    n = np.array(n)
    d = sum((n - mean_len) * (Y - Y_mean)) / np.sqrt(sum((n - mean_len) ** 2) * sum((Y - Y_mean) ** 2))
    corr_perm[i] = d
```

```
In [429]: left6c = np.quantile(corr_perm, alpha/2)
right6c = np.quantile(corr_perm, 1-alpha/2)
plt.hist(corr_perm, bins=20, density=True)
plt.axvline(left6c, color='b')
```

```
plt.axvline(right6с, color='b')
plt.axvline(corr, color = 'g', label = 'observed')
plt.axvline(np.mean(corr_perm), color = 'yellow', label = 'bootstrap')
plt.xlabel('Разница в баллах', fontsize=10)
plt.ylabel('Корреляция', fontsize=10)
plt.legend();
```



```
In [120]: print('ДИ перестановочный тест: [{:.4f}, {:.4f}].format(left, right))
```

ДИ перестановочный тест: [-0.1066, 0.1072]

```
In [240]: p_value = len([i for i in corr_perm if i > corr]) * 2 / 10 ** 4
print('p_value равно {:.4f}'.format(p_value))
```

p_value равно 0.6946

Вывод: нет оснований отвергать нулевую гипотезу

Номер 7 - разговор с чатом GPT

Вопрос:

Среди курьеров проводится соревнование по скоростной доставке заказов. Каждый участник имеет 20 попыток, лучшая из которых (минимальное время) засчитывается в качестве итогового результата. Время, за которое случайно выбранный участник (в пределах одной попытки) доставляет заказ, хорошо описывается экспоненциальным распределением с параметром λ и не зависит ни от результатов предыдущих попыток, ни от результатов других участников. Имеется выборка X_1, \dots, X_{500} из итоговых (лучших) результатов 500 участников.

Оцените вероятность того, что худший результат среди первых 50 участников соревнований окажется меньше 1.

Ответ чата

Заметим, что время доставки заказа X_i имеет экспоненциальное распределение с параметром λ и функцией распределения $F(x) = 1 - e^{-\lambda x}$.

Тогда, если X_1, \dots, X_{50} - лучшие результаты первых 50 участников, то вероятность того, что худший результат среди них будет меньше 1, равна:

$$P(\min\{X_1, \dots, X_{50}\} < 1) = 1 - P(\min\{X_1, \dots, X_{50}\} \geq 1)$$

Заметим, что $\min\{X_1, \dots, X_{50}\}$ также имеет экспоненциальное распределение с параметром 50λ .

Тогда:

$$P(\min\{X_1, \dots, X_{50}\} \geq 1) = e^{-50\lambda}$$

И, следовательно:

$$P(\min\{X_1, \dots, X_{50}\} < 1) = 1 - e^{-50\lambda}$$

Для оценки вероятности $P(\min\{X_1, \dots, X_{50}\} < 1)$ нужно оценить параметр λ . Для этого воспользуемся выборкой X_1, \dots, X_{50} и методом максимального правдоподобия.

Функция правдоподобия для выборки X_1, \dots, X_{50} имеет вид:

$$L(\lambda) = \prod_{i=1}^{50} \lambda e^{-\lambda X_i} = \lambda^{50} e^{-\lambda \sum_{i=1}^{50} X_i}$$

Логарифмируя функцию правдоподобия, получаем:

$$\ln L(\lambda) = 50 \ln \lambda - \lambda \sum_{i=1}^{50} X_i$$

Дифференцируя логарифм функции правдоподобия по λ и приравнявая производную к нулю, получаем оценку параметра λ :

$$\hat{\lambda} = \frac{50}{\sum_{i=1}^{50} X_i}$$

Теперь можем оценить вероятность $P(\min\{X_1, \dots, X_{50}\} < 1)$:

$$P(\min\{X_1, \dots, X_{50}\} < 1) = 1 - e^{-50\hat{\lambda}} = 1 - e^{-\frac{25}{\bar{X}}}$$

где $\bar{X} = \frac{1}{50} \sum_{i=1}^{50} X_i$ - выборочное среднее.

Номер 8

Максимально милое и полезное видео на понимание p-value, после его просмотра стало чуточку понятнее, что это такое

<https://www.youtube.com/watch?v=9jW9G8MO4PQ>

А также целый сайт для объяснения матстата и тервера для чайников, просто погружение в материал, было отличным подспорьем при подготовке к контрольным, особенно минимумам http://www.mathprofi.ru/matematicheskaya_statistika.html

Ну и, конечно же, лучший курс по андану для эконома, без которого выполнение этой домашки было бы гораздо менее приятным :)

https://github.com/hse-econ-data-science/andan_2023/blob/main/sem13_bootstrap/sem13_python_bootstrap.ipynb