In [2]:
```
import pandas as pd
import numpy as np
import scipy.stats as sts
```

Номер 1

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

$$lnL = ln(n-1) + ln(n-2) + \ldots + ln(n-8) - 9lnn$$

Максимизируем логарифмическую функцию правдоподобия и получим:

$$\frac{dlnL(n)}{dn} = \frac{1}{n-1} + \frac{1}{n-2} + \ldots + \frac{1}{n-8} - \frac{9}{n} = 0$$

Номер 3

Пункт а

Построим сначала классический асимптотический нормальный доверительный интервал

In [217…
```
def classic_CI(sim=10**4):
    n_in_ci = 0
    for i in range(sim):
        X = sts.expon.rvs(scale=1, size=20)
        X_mean = X.mean()
        X_std = X.std()
        ci_left = X_mean - sts.norm.ppf(0.975)*X_std/np.sqrt(len(X))
        ci_right = X_mean + sts.norm.ppf(0.975)*X_std/np.sqrt(len(X))
        if (ci_left <= 1) and (ci_right >=1):
            n_in_ci += 1
    return n_in_ci/10000
```

In [218…
```
classic_CI(sim=10**4)
```

Out[218…
```
0.8989
```

С помощью наивного бутстрапа

In [200…
```
np.random.seed(13)
def get_bootstrap_sample(x, B_sample=10**4):
    N = x.size
    sample = np.random.choice(x, size=(N, B_sample), replace=True)
    if B_sample == 1:
        sample = sample.T[0]
    return sample
```

In [13]:
```
def naive_bootstrap(sim=10**4):
    n_in_ci = 0
    alpha = 0.05
```

```
    for i in range(sim):
        X = sts.expon.rvs(scale=1, size=20)
        X_sample = get_bootstrap_sample(X, B_sample=10**4)
        X_boot_m = np.mean(X_sample, axis=0)
        left = np.quantile(X_boot_m, alpha/2)
        right = np.quantile(X_boot_m, 1-alpha/2)
        if (left <= 1) and (right >= 1):
            n_in_ci += 1
    return n_in_ci/10000
```

In [14]:
```
naive_bootstrap(sim=10**4)
```

Out[14]: 0.9064

In [15]:
```
def t_bootstrap(sim=10**4):
    n_in_ci = 0
    alpha = 0.05
    for i in range(sim):
        X = sts.expon.rvs(scale=1, size=20)
        X_sample = get_bootstrap_sample(X, B_sample=10**4)

        theta_hat = np.mean(X)
        std_hat = np.std(X)

        x_boot_t = np.mean(X_sample - theta_hat, axis=0)
        x_boot_t = x_boot_t/np.std(X_sample, axis=0)

        left = theta_hat - np.quantile(x_boot_t, 1-alpha/2)*std_hat
        right = theta_hat - np.quantile(x_boot_t, alpha/2)*std_hat

        if (left <= 1) and (right >= 1):
            n_in_ci += 1
    return n_in_ci/10000
```

In [16]:
```
t_bootstrap(sim=10**4)
```

Out[16]: 0.9488

Пункт б

In [205…
```
def classic_CI_t(sim=10**4):
    n_in_ci = 0
    for i in range(sim):
        X = np.random.standard_t(df=3, size=20)
        X_mean = X.mean()
        X_std = X.std()
        ci_left = X_mean - sts.norm.ppf(0.975)*X_std/np.sqrt(len(X))
        ci_right = X_mean + sts.norm.ppf(0.975)*X_std/np.sqrt(len(X))
        if (ci_left <= 0) and (ci_right >=0):
            n_in_ci += 1
    return n_in_ci/10000
```

In [209…
```
np.random.seed(13)
classic_CI_t(sim=10**4)
```

Out[209… 0.9375

In [212…

```
def naive_bootstrap_t(sim=10**4):
    n_in_ci = 0
    alpha = 0.05
    for i in range(sim):
        X = np.random.standard_t(df=3, size=20)
        X_sample = get_bootstrap_sample(X, B_sample=10**4)
        X_boot_m = np.mean(X_sample, axis=0)
        left = np.quantile(X_boot_m, alpha/2)
        right = np.quantile(X_boot_m, 1-alpha/2)
        if (left <= 0) and (right >= 0):
            n_in_ci += 1
    return n_in_ci/10000
```

In [213…

```
np.random.seed(13)
naive_bootstrap_t(sim=10**4)
```

Out[213…

```
0.9179
```

In [89]:

```
def t_bootstrap_t(sim=10**4):
    n_in_ci = 0
    alpha = 0.05
    for i in range(sim):
        X = np.random.standard_t(df=3, size=20)
        X_sample = get_bootstrap_sample(X, B_sample=10**4)

        theta_hat = np.mean(X)
        std_hat = np.std(X)

        x_boot_t = np.mean(X_sample - theta_hat, axis=0)
        x_boot_t = x_boot_t/np.std(X_sample, axis=0)

        left = theta_hat - np.quantile(x_boot_t, 1-alpha/2)*std_hat
        right = theta_hat - np.quantile(x_boot_t, alpha/2)*std_hat

        if (left <= 0) and (right >= 0):
            n_in_ci += 1
    return n_in_ci/10000
```

In [214…

```
np.random.seed(13)
t_bootstrap_t(sim=10**4)
```

Out[214…

```
0.9233
```

Пункт в

Сравним полученные результаты:

Для экспоненциального распределения наилучшие результаты показал бутстрэп t-статистики, а наихудшие - классический асимптотический доверительный интервал

Для распределения Стьюдента наилучшие результаты показал классический доверительный интервал, а наихудший - наивный бутстрэп

Номер 4

In [190…
```
prob = pd.read_csv('probability_exam.csv', sep=';')
```

In [111…
```
prob.head()
```

Out[111…

	id	surname	name	score
0	1	Репенкова	Полина Александровна	16
1	2	Ролдугина	Софья Александровна	0
2	3	Сафина	Алия Линаровна	19
3	4	Сидоров	Иван Максимович	26
4	5	Солоухин	Иван Владимирович	21

Разделим таблицу на две выборки: в первой выборке фамилии будут начинаться с согласной, а во второй - с гласной

In [118…
```
sogl = ('Б', 'В', "Г", "Д", "Ж", "З", "й", "К", "Л", "М", "Н", "П", "Р", "С", "Т", "
glas = ("А", "Е", "Ё", "И", "О", "У", "Ы", "Э", "Ю", "Я")
prob.loc[prob['surname'].str.startswith(glas), 'part'] = 2
prob.loc[prob['surname'].str.startswith(sogl), 'part'] = 1
```

In [120…
```
prob
```

Out[120…

	id	surname	name	score	part
0	1	Репенкова	Полина Александровна	16	1.0
1	2	Ролдугина	Софья Александровна	0	1.0
2	3	Сафина	Алия Линаровна	19	1.0
3	4	Сидоров	Иван Максимович	26	1.0
4	5	Солоухин	Иван Владимирович	21	1.0
...	...	...	...	...	...
327	328	Сенников	Александр -	19	1.0
328	329	Ся	Юйцянь -	0	1.0
329	330	Сятова	Альфия -	0	1.0
330	331	Темиркулов	Дастан Автандилович	0	1.0
331	332	Эшмеев	Павел Владиславович	16	2.0

332 rows × 5 columns

$H_0 : \mu_{sogl} = \mu_{glas}$

$H_1 : \mu_{sogl} \neq \mu_{glas}$

Пункт а

Используем тест Уэлча для проверки гипотезы о равенстве математических ожиданий баллов за экзамен для двух полученных выборок

In [122...
```
welch = sts.ttest_ind(prob['score'].loc[prob['part'] == 1], prob['score'].loc[prob['
welch
```

Out[122...
```
Ttest_indResult(statistic=0.8519661870595602, pvalue=0.3974027153843839)
```

Заметим, что $p - value > \alpha$, соответственно, нет оснований для опровержения гипотезы

Пункт 6

Наивный бутстрэп

In [124...
```
alpha = 0.05
```

In [177...
```
sogl_sample = get_bootstrap_sample(prob['score'].loc[prob['part'] == 1], B_sample=10
glas_sample = get_bootstrap_sample(prob['score'].loc[prob['part'] == 2], B_sample=10
sogl_boot_m = np.mean(sogl_sample, axis=0)
glas_boot_m = np.mean(glas_sample, axis=0)
diff_boot_m = sogl_boot_m - glas_boot_m #найдем разницу средних бутстрапированных вы
sogl_m = prob['score'].loc[prob['part'] == 1].mean()
glas_m = prob['score'].loc[prob['part'] == 2].mean()
diff_m_real = sogl_m - glas_m #найдем разницу средних существующих выборок
left = np.quantile(diff_boot_m, alpha/2)
right = np.quantile(diff_boot_m, 1-alpha/2)
```

Заметим, что для того, чтобы не было оснований опровергнуть гипотезу H_0, 0 должен входить в получившийся доверительный интервал и p-value должна быть больше, чем α

In [179...
```
if (left < 0) and (right > 0):
    print('Нет оснований для опровержения H0')
    print(left, right)
```

```
Нет оснований для опровержения H0
-1.2840700944688828 3.604040167303668
```

Номер 5

Добавим признак о том, набрал ли студент больше медианы баллов

In [127...
```
med = prob['score'].median()
med
```

Out[127...
```
17.5
```

In [128...
```
prob.loc[prob['score'] > med, 'more_than_med'] = True
prob.loc[prob['score'] < med, 'more_than_med'] = False
prob
```

Out[128...

	id	surname	name	score	part	more_than_med
0	1	Репенкова	Полина Александровна	16	1.0	False
1	2	Ролдугина	Софья Александровна	0	1.0	False
2	3	Сафина	Алия Линаровна	19	1.0	True

	id	surname	name	score	part	more_than_med
3	4	Сидоров	Иван Максимович	26	1.0	True
4	5	Солоухин	Иван Владимирович	21	1.0	True
...	...	...	...	...	...	...
327	328	Сенников	Александр -	19	1.0	True
328	329	Ся	Юйцянь -	0	1.0	False
329	330	Сятова	Альфия -	0	1.0	False
330	331	Темиркулов	Дастан Автандилович	0	1.0	False
331	332	Эшмеев	Павел Владиславович	16	2.0	False

332 rows × 6 columns

Теперь создадим таблицу сопряженности, в которой будет указано количество студентов в каждой из четырех групп, где good - количество студентов, которые набрали за экзамен больше медианы баллов, а bad - количество студентов, которые набрали меньше медианы баллов

In [130...
```
df_5 = pd.DataFrame({'surname': ['consonant', 'vowel'],
                     'good': [len(prob.loc[(prob['part'] == 1) & (prob['more_than_me
                    'bad': [len(prob.loc[(prob['part'] == 1) & (prob['more_than_med'
df_5
```

Out[130...

	surname	good	bad
0	consonant	145	138
1	vowel	21	28

Пункт а

Построим доверительный интервал для отношения шансов хорошо написать контрольную студентов, чьи фамилии начинаются на гласную к студентам, чьи фамилии начинаются на согласную

In [146...
```
cons_good = len(prob.loc[(prob['part'] == 1) & (prob['more_than_med'] == True)])
cons_bad = len(prob.loc[(prob['part'] == 1) & (prob['more_than_med'] == False)])
vow_good = len(prob.loc[(prob['part'] == 2) & (prob['more_than_med'] == True)])
vow_bad = len(prob.loc[(prob['part'] == 2) & (prob['more_than_med'] == False)])
```

Посчитаем шансы

In [137...
```
odds_cons = cons_good/cons_bad
odds_vow = vow_good/vow_bad
OR = odds_vow/odds_cons
print(OR)
stand_error = np.sqrt(1/cons_bad + 1/cons_good + 1/vow_bad + 1/vow_good)
print(stand_error)
```

```
0.7137931034482758
0.3122118861751831
```

Найдем 95% доверительный интервал

In [145…

```
left_odds_1 = OR * np.exp(-sts.norm.ppf(0.975) * stand_error)
right_odds_1 = OR * np.exp(sts.norm.ppf(0.975) * stand_error)
if (left_odds_1 <= 1) and (right_odds_1 >= 1):
    print('Нет оснований отвергать гипотезу, так как 1 попадает в доверительный инте
else:
    print('1 не попадает в доверительный интервал, поэтому гипотеза отвергается')
print('Доверительный интервал: (' + str(left_odds_1) + '; ' + str(right_odds_1) + ')
```

```
Нет оснований отвергать гипотезу, так как 1 попадает в доверительный интервал
Доверительный интервал: (0.38709459582547806; 1.3162172761513564)
```

Найдем p-value

In [141…

```
p_value = (1 - sts.norm.cdf(OR)) * 2
print(p_value)
if p_value > alpha:
    print('Нет оснований отвергать гипотезу H0, так как p-value больше значения alph
else:
    print('Гипотеза H0 отвергается')
```

```
0.47535512483042774
Нет оснований отвергать гипотезу H0, так как p-value больше значения alpha
```

Пункт 6

Построим теперь доверительный интервал для отношения вероятностей хорошо написать контрольную студентов, чьи фамилии начинаются на гласную к студентам, чьи фамилии начинаются на согласную

Посчитаем вероятности

In [143…

```
prob_cons = cons_good/(cons_bad + cons_good)
prob_vow = vow_good/(vow_bad + vow_good)
RR = prob_vow/prob_cons
print(RR)
stand_error_p = np.sqrt(1/cons_good - 1/(cons_good+cons_bad) + 1/vow_good - 1/(vow_g
print(stand_error_p)
```

```
0.8364532019704434
0.17485384517729596
```

Найдем доверительный интервал

In [160…

```
left_risk = RR * np.exp(-sts.norm.ppf(0.975) * stand_error_p)
right_risk = RR * np.exp(sts.norm.ppf(0.975) * stand_error_p)
if (left_risk <= 1) and (right_risk >= 1):
    print('Нет оснований отвергать гипотезу, так как 1 попадает в доверительный инте
else:
    print('1 не попадает в доверительный интервал, поэтому гипотеза отвергается')
print('Доверительный интервал: (' + str(left_risk) + '; ' + str(right_risk) + ')')
```

```
Нет оснований отвергать гипотезу, так как 1 попадает в доверительный интервал
Доверительный интервал: (0.5937529565040844; 1.1783586951819993)
```

Посчитаем p-value

In [148…

```
p_value = (1 - sts.norm.cdf(RR)) * 2
print(p_value)
if p_value > alpha:
    print('Нет оснований отвергать гипотезу H0, так как p-value больше значения alph
```

```
else:
    print('Гипотеза H0 отвергается')
```

```
0.4028999934344841
Нет оснований отвергать гипотезу H0, так как p-value больше значения alpha
```

Пункт в

In [172...

```
cons_sample = get_bootstrap_sample(prob['score'].loc[prob['part'] == 1], B_sample=10
vow_sample = get_bootstrap_sample(prob['score'].loc[prob['part'] == 2], B_sample=10*
median_boot = pd.concat([pd.DataFrame(cons_sample), pd.DataFrame(vow_sample)], join=
```

In [173...

```
OR_all = []
for i in range(10**4):
    cons_good = len(cons_sample[:, i][cons_sample[:, i] > median_boot[i]])
    cons_bad = len(cons_sample[:, i][cons_sample[:, i] < median_boot[i]])
    vow_good = len(vow_sample[:, i][vow_sample[:, i] > median_boot[i]])
    vow_bad = len(vow_sample[:, i][vow_sample[:, i] < median_boot[i]])
    odds_cons = cons_good/cons_bad
    odds_vow = vow_good/vow_bad
    OR = odds_vow/odds_cons
    OR_all.append(OR)
```

In [174...

```
left_odds_2 = np.quantile(OR_all, alpha/2)
right_odds_2 = np.quantile(OR_all, 1-alpha/2)
if (left_odds_2 <= 1) and (right_odds_2 >= 1):
    print('Нет оснований отвергать гипотезу, так как 1 попадает в доверительный инте
else:
    print('1 не попадает в доверительный интервал, поэтому гипотеза отвергается')
print('Доверительный интервал: (' + str(left_odds_2) + '; ' + str(right_odds_2) + ')
```

```
Нет оснований отвергать гипотезу, так как 1 попадает в доверительный интервал
Доверительный интервал: (0.37543626581838413; 1.411764705882353)
```

Номер 6

Пункт а

Посчитаем средний балл студентов за экзамен

In [181...

```
mean_score = prob['score'].mean()
```

Посчитаем длину фамилии студентов

In [191...

```
prob['len_surname'] = prob['surname'].apply(lambda x: len(x))
prob
```

Out[191...

	id	surname	name	score	len_surname
0	1	Репенкова	Полина Александровна	16	9
1	2	Ролдугина	Софья Александровна	0	9
2	3	Сафина	Алия Линаровна	19	6
3	4	Сидоров	Иван Максимович	26	7
4	5	Солоухин	Иван Владимирович	21	8
...	...	...	...	...	...

	id	surname	name	score	len_surname
327	328	Сенников	Александр -	19	8
328	329	Ся	Юйцянь -	0	2
329	330	Сятова	Альфия -	0	6
330	331	Темиркулов	Дастан Автандилович	0	10
331	332	Эшмеев	Павел Владиславович	16	6

332 rows × 5 columns

Найдем β с помощью метода моментов

Заметим, что $E(Y_i) = E(Y_{mean}) = Y_{mean}$

$E(Y_i) = \frac{1}{n} * E(\sum Y_i) = \frac{1}{n} * \sum(F_i * \beta) = Y_{mean}$

Таким образом, $\beta = \frac{Y_{mean} * n}{\sum(F_i)}$

In [194...
```
beta = mean_score * len(prob['score']) / prob['len_surname'].sum()
beta
```

Out[194...
```
2.0613026819923372
```

Теперь рассчитаем выборочную корреляцию

In [197...
```
corr = np.corrcoef(prob['score'], prob['len_surname'])[0][1]
corr
```

Out[197...
```
0.025328052669147665
```

Номер 7

Вопрос:

Производитель молочных продуктов выпустил новый низкокалорийный йогурт Fit и утверждает, что он вкуснее его более калорийного аналога Fat. Четырем независимым экспертам предлагают выбрать наиболее вкусный йогурт из трёх, предлагая им в одинаковых стаканчиках в случайном порядке два Fat и один Fit. Предположим, что йогурты одинаково привлекательны. Величина ξ — число экспертов, отдавших предпочтение Fit. а) Какова вероятность, что большинство экспертов выберут Fit?

б) Постройте функцию распределения величины ξ;

в) Каково наиболее вероятное число экспертов, отдавших предпочтение йогорту Fit?

г) Вычислите математическое ожидание и дисперсию ξ

Ответ:

а) Вероятность того, что большинство экспертов выберут Fit, равна сумме вероятностей того, что 3 эксперта выберут Fit и 1 выберет Fat, и того, что 2 эксперта выберут Fit и 2

выберут Fat: P(большинство выберут Fit) = P(3 выберут Fit и 1 выберет Fat) + P(2 выберут Fit и 2 выберут Fat)

Для первого случая вероятность равна: P(3 выберут Fit и 1 выберет Fat) = C(4,3) *(1/3)^3* (2/3)^1 = 8/81

Для второго случая вероятность равна: P(2 выберут Fit и 2 выберут Fat) = C(4,2) *(1/3)^2* (2/3)^2 = 24/81

Суммируя эти вероятности, получим: P(большинство выберут Fit) = 8/81 + 24/81 = 32/81

Ответ: вероятность того, что большинство экспертов выберут Fit, равна 32/81.

б) Функция распределения величины ξ:

ξ 0 1 2 3 F(ξ) 0 8/81 32/81 41/81 в) Наиболее вероятное число экспертов, отдавших предпочтение йогурту Fit, равно 2, так как вероятность этого события наибольшая (32/81).

г) Математическое ожидание и дисперсия ξ:

Математическое ожидание: E(ξ) = 0 *(49/81)* + *1* (8/81) + 2 *(32/81)* + *3* (41/81) = 2

Дисперсия: D(ξ) = E(ξ^2) - (E(ξ))^2

E(ξ^2) = 0^2 *(49/81)* + *1^2* (8/81) + 2^2 *(32/81)* + *3^2* (41/81) = 6

D(ξ) = 6 - 2^2 = 2

Ответ: математическое ожидание ξ равно 2, а дисперсия равна 2.

Номер 8

Мне были очень полезны семинары из курса по анализу данных, в частности, семинар №13 про то, как написать программу для бутстрэпа:

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