Задача 3

In [1]:
```
import numpy as np
import scipy.stats as sts
```

a) $X_1, \dots, X_{20} \sim i.i.d.,\ X_i \sim Exp(1)$

In [2]:
```
alpha = 0.05  # Уровень значимости
lam = 1  # Интенсивность показательного распределения
```

1 способ:

In [3]:
```
def does_norm_CI_cover(sample):
    q_l = np.mean(sample) - sts.norm.ppf(1 - alpha / 2) * np.std(sample, ddo
    q_r = np.mean(sample) + sts.norm.ppf(1 - alpha / 2) * np.std(sample, ddo
    return q_l <= 1 / lam and 1 / lam <= q_r
```

In [4]:
```
# Создадим 10**4 выборок размера 20
samples = sts.expon.rvs(size=(10**4, 20), scale=1 / lam, random_state=222)
print(samples.shape)
samples
```

Out[4]:
```
(10000, 20)
array([[0.74660793, 1.15619008, 1.80590416, ..., 0.28984257, 2.72612982,
        0.08608175],
       [0.55425976, 1.38121152, 0.28823428, ..., 2.75471494, 0.3378933 ,
        0.29786844],
       [0.3621953 , 0.2221918 , 0.34847724, ..., 0.60011926, 0.51124433,
        3.52871164],
       ...,
       [2.13482072, 3.52043236, 0.88118601, ..., 0.1026552 , 2.71131173,
        1.68439971],
       [2.4833841 , 1.47532361, 0.83892948, ..., 1.31164588, 6.56467774,
        0.31202082],
       [1.63514573, 1.79093836, 0.50246544, ..., 0.402314  , 0.06868973,
        2.35401496]])
```

In [5]:
```
# Получим оценку вероятности накрытия доверительным интервалом мат. ожидания
arr = np.array([does_norm_CI_cover(sample) for sample in samples])
print(f"Оценка вероятности накрытия доверительным интервалом мат. ожидания равна {np.me
```

Оценка вероятности накрытия доверительным интервалом мат. ожидания равна 0.9024

2 способ:

In [6]:
```
# Сгенерируем базовую выборку
base_sample = sts.expon.rvs(size=20, scale=1/lam, random_state=222)
print(base_sample.shape)
base_sample
```

Out[6]:
```
(20,)
array([0.74660793, 1.15619008, 1.80590416, 1.06063829, 0.03772467,
       0.19864079, 1.42949175, 0.25399538, 1.32853529, 0.33896198,
       1.31968301, 0.11389402, 0.09855281, 1.52872961, 0.73313077,
       2.3548982 , 0.50706333, 0.28984257, 2.72612982, 0.08608175])
```

In [7]:
```
does_naive_CI_cover = []

# Проделаем 10**4 процедур создания Д.И.
```

```
for i in range(10**4):
    np.random.seed(i)

    # Создадим 25 бутстрэп-выборок размера 20, не слишком много, чтобы можно было с
    iteration = np.random.choice(base_sample, size=(25, 20))

    # Создадим список средних, посчитанных по всем бутстрэп-выборкам
    theta_hat_stars = [np.mean(sample) for sample in iteration]

    # Посчитаем левый и правый квантили
    q_l = np.quantile(theta_hat_stars, q=alpha / 2)
    q_r = np.quantile(theta_hat_stars, q=1 - alpha / 2)

    # Добавим в итоговый список результат: накрывает ли найденный ДИ истинное матема
    cond = (q_l <= 1 / lam) & (1 / lam <= q_r)
    does_naive_CI_cover.append(cond)
```

In [8]:
```
# Получим оценку вероятности накрытия доверительным интервалом мат. ожидания
print(f"Оценка вероятности накрытия доверительным интервалом мат. ожидания равна {np.me
```

Оценка вероятности накрытия доверительным интервалом мат. ожидания равна 0.9996

3 способ:

In [9]:
```
# Сгенерируем базовую выборку
base_sample = sts.expon.rvs(size=20, scale=1/lam, random_state=222)
print(base_sample.shape)
base_sample
```

Out[9]:
```
(20,)
array([0.74660793, 1.15619008, 1.80590416, 1.06063829, 0.03772467,
       0.19864079, 1.42949175, 0.25399538, 1.32853529, 0.33896198,
       1.31968301, 0.11389402, 0.09855281, 1.52872961, 0.73313077,
       2.3548982 , 0.50706333, 0.28984257, 2.72612982, 0.08608175])
```

In [10]:
```
# Посчитаем среднее значение и стандартную ошибку по исходной выборке
theta_hat = np.mean(base_sample)
se_theta_hat = np.std(base_sample, ddof=1) / np.sqrt(len(base_sample))
```

In [11]:
```
def t_star_counter(sample):
    return (np.mean(sample) - theta_hat) / (np.std(sample, ddof=1) / np.sqrt
```

In [12]:
```
does_t_bootstap_CI_cover = []
for i in range(10**4):
    np.random.seed(i)

    # Создадим 25 бутстрэп-выборок размера 20, не слишком много, чтобы можно было с
    iteration = np.random.choice(base_sample, size=(25, 20))

    # Создадим список t-статистик, посчитанных по всем бутстрэп-выборкам
    t_stars = np.array([t_star_counter(sample) for sample in iteration])

    # Посчитаем левый и правый квантили
    q_l = np.quantile(t_stars, q=alpha/2)
    q_r = np.quantile(t_stars, q=1-alpha/2)

    cond = (theta_hat - q_r * se_theta_hat <= 1 / lam) & (1 / lam <= theta_h
    does_t_bootstap_CI_cover.append(cond)
```

In [13]:
```
# Получим оценку вероятности накрытия доверительным интервалом мат. ожидания
print(f"Оценка вероятности накрытия доверительным интервалом мат. ожидания равна {np.me
```

Оценка вероятности накрытия доверительным интервалом мат. ожидания равна 0.9998

б)

1 способ:

In [14]:
```
def does_norm_CI_cover(sample):
    q_l = np.mean(sample) - sts.norm.ppf(1 - alpha / 2) * np.std(sample, ddo
    q_r = np.mean(sample) + sts.norm.ppf(1 - alpha / 2) * np.std(sample, ddo
    return q_l <= 0 and 0 <= q_r
```

In [15]:
```
# Создадим 10000 выборок
np.random.seed(111)
samples = np.random.standard_t(df=3, size=(10**4, 20))
print(samples.shape)
samples
```

Out[15]:
```
(10000, 20)
array([[-1.08669864e+00, -5.74089037e-01, -7.98119348e-02, ...,
        -3.62640210e-03, -8.50206406e-01, -8.86044880e-01],
       [-2.78706043e+00, -8.50150788e-01,  4.50058763e-01, ...,
        -1.06382198e-01, -3.25162308e-01,  1.45758236e+00],
       [ 3.23789516e-01,  1.64488594e+00, -2.18188566e+00, ...,
         6.62300497e-01,  5.99714079e-01, -6.42638238e-01],
       ...,
       [-6.65070368e-01, -1.93785060e-01, -5.53176705e-01, ...,
        -6.48768253e+00, -4.21151019e-02, -6.77056852e-01],
       [ 1.34310331e+00,  8.45396151e-01, -7.92699574e-01, ...,
         5.61866638e-01,  3.17609257e+00, -3.61733677e-01],
       [ 1.88724641e+00, -1.41306151e-01,  4.18548984e-01, ...,
        -3.20680641e-01,  4.68659648e-01, -2.47280150e-01]])
```

In [16]:
```
arr = np.array([does_norm_CI_cover(sample) for sample in samples])
print(f"Оценка вероятности накрытия доверительным интервалом мат. ожидания равна {np.me
```

Оценка вероятности накрытия доверительным интервалом мат. ожидания равна 0.9442

2 способ:

In [17]:
```
np.random.seed(111)
base_sample = np.random.standard_t(df=3, size=20)
print(base_sample.shape)
base_sample
```

Out[17]:
```
(20,)
array([-1.08669864e+00, -5.74089037e-01, -7.98119348e-02, -2.20083002e-01,
       -1.18474575e+00,  9.79129257e-02,  1.26531716e+00, -6.24767768e-02,
        4.45462511e-01,  6.27476072e-01,  2.45541848e+00, -3.05525889e+00,
        5.96614266e-01,  7.12012381e-01, -4.40014695e-01,  1.92665032e+00,
        5.03694196e+00, -3.62640210e-03, -8.50206406e-01, -8.86044880e-01])
```

In [18]:
```
does_naive_CI_cover = []
for i in range(10**4):
    np.random.seed(i)
    # Создадим 25 бутстрэп-выборок размера 20
    iteration = np.random.choice(base_sample, size=(25, 20))

    # Создадим список средних, посчитанных по всем бутстрэп-выборкам
    theta_hat_stars = [np.mean(sample) for sample in iteration]

    # Посчитаем левый и правый квантили
    q_l = np.quantile(theta_hat_stars, q=alpha/2)
    q_r = np.quantile(theta_hat_stars, q=1-alpha/2)

    # Добавим в итоговый список результат: накрывает ли найденный ДИ истинное матема
```

```
    cond = (q_l <= 0) & (0 <= q_r)
    does_naive_CI_cover.append(cond)
```

In [19]:
```
# Получим оценку вероятности накрытия доверительным интервалом мат. ожидания
print(f"Оценка вероятности накрытия доверительным интервалом мат. ожидания равна {np.me
```

Оценка вероятности накрытия доверительным интервалом мат. ожидания равна 0.9986

3 способ:

In [20]:
```
np.random.seed(111)
base_sample = np.random.standard_t(df=3, size=20)
print(base_sample.shape)
base_sample
```

Out[20]:
```
(20,)
array([-1.08669864e+00, -5.74089037e-01, -7.98119348e-02, -2.20083002e-01,
       -1.18474575e+00,  9.79129257e-02,  1.26531716e+00, -6.24767768e-02,
        4.45462511e-01,  6.27476072e-01,  2.45541848e+00, -3.05525889e+00,
        5.96614266e-01,  7.12012381e-01, -4.40014695e-01,  1.92665032e+00,
        5.03694196e+00, -3.62640210e-03, -8.50206406e-01, -8.86044880e-01])
```

In [21]:
```
# Посчитаем среднее значение и стандартную ошибку по исходной выборке
theta_hat = np.mean(base_sample)
se_theta_hat = np.std(base_sample, ddof=1) / np.sqrt(len(base_sample))
```

In [22]:
```
def t_star_counter(sample):
    return (np.mean(sample) - theta_hat) / (np.std(sample, ddof=1) / np.sqrt
```

In [23]:
```
np.random.seed(222)

does_t_bootstap_CI_cover = []
for i in range(10**4):
    np.random.seed(i)

    # Создадим 25 бутстрэп-выборок размера 20, не слишком много, чтобы можно было с
    iteration = np.random.choice(base_sample, size=(25, 20))

    # Создадим список t-статистик, посчитанных по всем бутстрэп-выборкам
    t_stars = np.array([t_star_counter(sample) for sample in iteration])

    # Посчитаем левый и правый квантили
    q_l = np.quantile(t_stars, q=alpha/2)
    q_r = np.quantile(t_stars, q=1-alpha/2)

    cond = (theta_hat - q_r * se_theta_hat <= 0) & (0 <= theta_hat - q_l * s
    does_t_bootstap_CI_cover.append(cond)
```

In [24]:
```
# Получим оценку вероятности накрытия доверительным интервалом мат. ожидания
print(f"Оценка вероятности накрытия доверительным интервалом мат. ожидания равна {np.me
```

Оценка вероятности накрытия доверительным интервалом мат. ожидания равна 0.9977

в)

Вывод: В п. **а** лучше всех себя показал бутстрэп t-статистики, а в п. **б** - наивный бутстрэп.