```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
!pip install arch
```

```
Requirement already satisfied: arch in /usr/local/lib/python3.10/dist-packages (6.2.0)
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.10/dist-packages (from arch) (1.23.5)
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.10/dist-packages (from arch) (1.11.3)
Requirement already satisfied: pandas>=1.1 in /usr/local/lib/python3.10/dist-packages (from arch) (1.5.3)
Requirement already satisfied: statsmodels>=0.12 in /usr/local/lib/python3.10/dist-packages (from arch) (0.14.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1->arch) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1->arch) (2023.3.post1)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.12->arch) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.12->arch) (23.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.12->arch) (1.16.0)
```

```
from arch.bootstrap import IIDBootstrap, IndependentSamplesBootstrap
```

```
rng = np.random.default_rng(131123)
x= rng.normal(loc=5, scale=4, size=20)
```

```
x
```

```
array([ 9.37822494,  1.42644792,  4.0469715 ,  8.30606264,  4.91599978,
        7.4402057 ,  0.38310218,  6.01956045,  7.64807502,  6.87581191,
        3.42976811,  2.51190751, 10.81304314,  0.82566469,  1.03971287,
        8.51201933,  3.47244996,  5.02142554, -1.47430039,  3.94698933])
```

$E(x_1) = 5$ $Var(X_i) = 4^2$, X_i независимы.

```
np.mean(x)
```

```
4.726957107341645
```

Поиграем в настоящего иссоелователя, сделаем вид, что мы не знаем $E(X_i)$. Мы можем получить точечную оценку для математического ожидания. Естественная формула, $\bar{x} = \frac{x_1 + \ldots + x_n}{n}$

```
mu_hat = np.mean(x)
mu_hat
```

```
4.726957107341645
```

Как построит доверительный интервал для неизвестного $\mu = E(X_i)$? Хочу точечную оценку превратить в интервальную. Достаточно "размножить" точечную оценку.

Вывод: из наших $n = 20$ наблюдений случайно выберем 20 с возможностью повтора.

```
x_star1 = rng.choice(x, size=len(x))
```

```
x_star1
```

```
array([ 1.03971287, -1.47430039,  8.30606264, 10.81304314,  3.47244996,
        5.02142554,  7.64807502,  2.51190751,  1.03971287,  3.94698933,
        7.4402057 ,  9.37822494,  6.87581191,  1.03971287,  3.47244996,
        6.87581191,  6.87581191,  0.38310218,  0.82566469,  7.64807502])
```

```
np.mean(x_star1)
```

```
4.6569974785301635
```

```
x_star2 = rng.choice(x, size=len(x))
np.mean(x_star2)
```

```
2.925412241598031
```

```
x_star2
```

```
array([ 9.37822494,  0.38310218,  3.42976811,  0.82566469,  1.42644792,
        3.47244996,  2.51190751,  3.47244996,  9.37822494, -1.47430039,
        3.42976811,  1.03971287,  0.82566469,  3.47244996, -1.47430039,
       10.81304314, -1.47430039,  0.38310218,  0.38310218,  8.30606264])
```

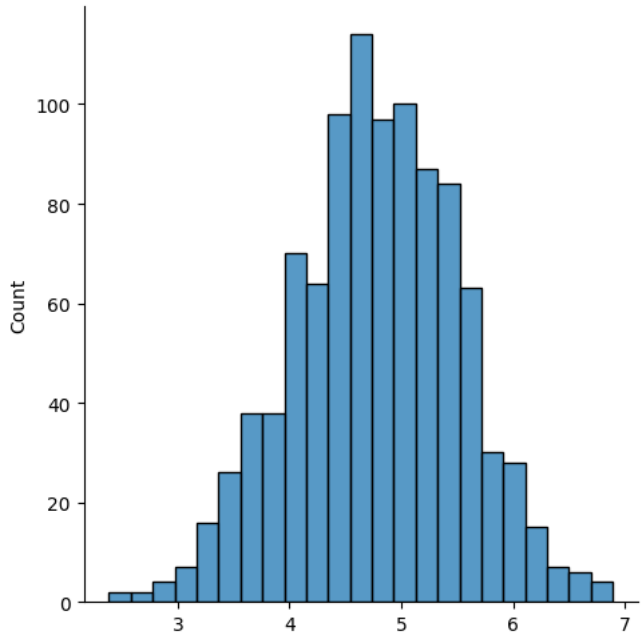
```
n_boot = 1000
mu_hat_star = [np.mean(rng.choice(x, size=len(x))) for i in range(n_boot)]
```

```
mu_hat_star[1:10]
```

```
[3.050223510385478,
 5.438397406179908,
 4.586355083948829,
 4.756568981525143,
 5.128541643733957,
 5.68474586422387,
 4.666978871724915,
 6.115400245949409,
 5.063112930329381]
```

```
sns.displot(x=np.array(mu_hat_star))
```

```
<seaborn.axisgrid.FacetGrid at 0x788a7650b850>
```



Наивный бутстрэп доверительный интервал для $\mu = E(X_i)$

```
[np.quantile(mu_hat_star, 0.025), np.quantile(mu_hat_star, 0.975)]
```

```
[3.293853021868897, 6.1863875400958594]
```

Плюсы:

1. Мы можем не знать формулы для дисперсии оценки.

Мы не использовали $Var(\hat{\mu})$

2. В большинстве случаев формулы верны при больших n. И бутстрэп тоже требует больших n.

Часто оказывается, что большое n, нужное для "правильного" варианта бутстрэпа, гораздо меньше, чем больное n, нужное для центральной предельной теоремы. Здесь мы строим бутстрэп доверительный интервал для μ;

```
boot_x = IIDBootstrap(x, seed=131123)
boot_x.conf_int(np.mean, method='basic', reps=10000, size=0.95)
```

```
array([[3.29911297],
       [6.16044048]])
```

А здесь доверительный интервал для σ:

```
boot_x.conf_int(np.std, method='basic', reps=10000, size=0.95)
```

```
array([[2.62463275],
       [4.17933285]])
```

Tim Hestergerg, What teacher should know about bootstrap?

```
w = x + rng.normal(loc=1, scale=3, size=len(x))
```

```
np.corrcoef(x, w)
```

```
array([[1.        , 0.77905422],
       [0.77905422, 1.        ]])
```

```
def corr(x, y):
  corr_mat = np.corrcoef(x,y)
  return corr_mat[0,1]
```

```
corr(x, w)
```

```
0.7790542234649814
```

```
x
```

```
array([ 9.37822494,  1.42644792,  4.0469715 ,  8.30606264,  4.91599978,
        7.4402057 ,  0.38310218,  6.01956045,  7.64807502,  6.87581191,
        3.42976811,  2.51190751, 10.81304314,  0.82566469,  1.03971287,
        8.51201933,  3.47244996,  5.02142554, -1.47430039,  3.94698933])
```

```
w
```

```
array([ 1.45268099e+01,  3.35189500e+00,  6.37035315e+00,  1.20621021e+01,
        3.47422243e+00,  5.31293830e+00,  7.30495615e+00,  1.27090061e+01,
        1.05884655e+01,  4.64039688e+00,  3.87618414e+00,  1.28298066e+00,
        1.33589223e+01,  4.74954206e+00,  4.92262233e+00,  1.21749339e+01,
        3.88188779e+00,  6.65207158e+00, -2.17281783e-03,  4.01069644e+00])
```

```
obs_id = rng.choice(range(len(x)), size=len(x))
obs_id
```

```
array([13,  7, 18, 13,  3, 13,  4,  7, 15, 19,  4,  5,  0,  5, 11, 17,  5,
       12,  2,  8])
```

```
x_star1 = x[obs_id]
x_star1
```

```
array([ 1.03971287, -1.47430039,  8.30606264, 10.81304314,  3.47244996,
        5.02142554,  7.64807502,  2.51190751,  1.03971287,  3.94698933,
        7.4402057 ,  9.37822494,  6.87581191,  1.03971287,  3.47244996,
        6.87581191,  6.87581191,  0.38310218,  0.82566469,  7.64807502])
```

```
w_star1 = w[obs_id]
w_star1
```

```
array([ 4.74954206e+00,  1.27090061e+01, -2.17281783e-03,  4.74954206e+00,
        1.20621021e+01,  4.74954206e+00,  3.47422243e+00,  1.27090061e+01,
        1.21749339e+01,  4.01069644e+00,  3.47422243e+00,  5.31293830e+00,
        1.45268099e+01,  5.31293830e+00,  1.28298066e+00,  6.65207158e+00,
        5.31293830e+00,  1.33589223e+01,  6.37035315e+00,  1.05884655e+01])
```

```
corr(x_star1, w_star1)
```

```
-0.40898798150523846
```

```
boot_xw = IIDBootstrap(x, w, seed=131123)
boot_xw.conf_int(corr, method='basic', reps=10000, size=0.95)
```

```
array([[0.64452662],
       [1.02272886]])
```

```
4 - 3 == 1
```

```
True
```

```
0.4 - 0.3 == 0.1
```

```
False
```

```
y = rng.normal(loc=9, scale=3, size=25)
```

```
y
```

```
array([11.30682425,  5.79565967,  8.01503085,  5.35149433, 12.05612684,
       12.64778535, 11.75660882, 10.78256859, 11.98463813,  7.00434741,
       10.87857563, 11.93132004,  7.49922477,  9.02554332, 12.54405301,
        8.88598087,  9.01322336,  9.62972101,  7.57213223, 13.09290891,
       10.88642535,  7.96234663, 13.17257248, 10.4388024 , 10.19316908])
```

```
np.mean(y) - np.mean(x)
```

```
5.250126226677068
```

```
def mean_diff(x, y):
  return np.mean(y) - np.mean(x)
```

```
boot_xy = IndependentSamplesBootstrap(x, y, seed=131123)
boot_xy.conf_int(mean_diff, reps=10000, size=0.95, method='basic')
```

```
array([[3.5834364 ],
       [6.91985932]])
```