

▼ Определения

- **Параметрический бутстреп.**

Пусть $\hat{\theta}$ – оценка параметра θ по выборке X_1, \dots, X_n , которая получена из распределения P_θ . Бутстрепная выборка размера N в параметрическом бутстрепе – это выборка из распределения $P_{\hat{\theta}}$.

- **Непараметрический бутстреп.**

Пусть дана выборка X_1, \dots, X_n из распределения P и пусть P^* – эмпирическое распределение, построенное по этой выборке. Бутстрепная выборка размера N в непараметрическом бутстрепе – это выборка из распределения P^* . Легко видеть, что если $i_1, \dots, i_N \sim R\{1, \dots, N\}$ – независимые случайные величины, то X_{i_1}, \dots, X_{i_N} – бутстрепная выборка размера N в непараметрическом бутстрепе (построенная по выборке X_1, \dots, X_n из некоторого распределения P).

- **Бутстрепная оценка дисперсии.**

Пусть дана выборка X_1, \dots, X_n из распределения P_θ , а $\hat{\theta} = \hat{\theta}(X_1, \dots, X_n)$ – оценка параметра θ . Сгенерировано k бутстрепных выборок $X^1 = (X_1^1, \dots, X_N^1), \dots, X^k = (X_1^k, \dots, X_N^k)$ (при этом все эти выборки можно генерировать как на основе параметрического бутстрепа, так и на основе непараметрического, но эти серии выборок должны быть сгенерированы одним и тем же способом) и для каждой из них посчитана оценка параметра $\hat{\theta}_i = \hat{\theta}(X^i)$, $i \in \{1, \dots, k\}$. Далее по выборке $\{\hat{\theta}(X^i)\}_{i \geq 1}$ строится выборочная дисперсия $s^2(\hat{\theta}) = s^2(\hat{\theta}(X^1), \dots, \hat{\theta}(X^k))$, которая и называется бутстрепной оценкой дисперсии оценки $\hat{\theta}$.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats
import warnings
```

```
warnings.simplefilter('ignore')
```

```
%matplotlib inline
```

▼ Задача №1

(К теоретической задаче 3.1)

Сгенерируйте выборки X_1, \dots, X_N из всех распределений из задачи 3.1 ($N = 1000$). Для всех $n \leq N$ посчитайте значения полученных оценок (по выборке X_1, \dots, X_n) методом моментов. Оцените дисперсию каждой оценки, сгенерировав для каждой из них $K = 1000$ бутстрепных выборок а) с помощью параметрического бутстрепа, б) с помощью непараметрического бутстрепа. Проведите эксперимент для разных значений параметров распределений (рассмотрите не менее трех различных значений).

Распределения:

a) $\mathcal{N}(a, \sigma^2)$

b) $\Gamma(\alpha, \lambda)$

c) $\mathcal{R}(a, b)$

d) $Poiss(\lambda)$

e) $Bin(m, p)$

f) $Geom(p)$

g) $Beta(\lambda_1, \lambda_2)$

N, K = 1000, 1000

Аналитически было получено, что оценки полученные методом моментов равны:

a) $(\bar{X}; s^2)$

b) $(\frac{\bar{X}^2}{s^2}; \frac{s^2}{\bar{X}})$

c) $(\bar{X} - \sqrt{3 \cdot s^2}; \bar{X} + \sqrt{3 \cdot s^2})$

d) (\bar{X})

e) $(\frac{\bar{X}^2}{\bar{X} - s^2}; \frac{\bar{X} - s^2}{\bar{X}})$

$$f)\left(\frac{1}{X}\right)$$

$$g)\left(\overline{X} \cdot \frac{\overline{X}-\overline{X^2}}{s^2}; (1-\overline{X}) \cdot \frac{\overline{X}-\overline{X^2}}{s^2}\right)$$

Напишем функцию для подсчета выборочной дисперсии:

```
def calc_sample_variance(sample):
    return (sample**2).mean() - sample.mean()**2

s = calc_sample_variance
```

▼ Для упрощения генерации выборок напишем следующие функции:

▼ Нормальное распределение

```
def gen_normal_rvs(sample_size, param):
    """
    arg:    param = (\alpha, \sigma^2)
    usage: norm(loc=\alpha, scale=\sigma)
    """
    return stats.norm(
        loc=param[0],
        scale=param[1]**0.5
    ).rvs(size=sample_size)
```

▼ Гамма распределение

```
def gen_gamma_rvs(sample_size, param):
    """
    arg:    param = (\alpha, \beta)
    usage: gamma(a=\beta, scale=\frac{1}{\alpha})
    """
```

```

return stats.gamma(
    a=param[1],
    scale=1/param[0]
).rvs(size=sample_size)

```

▼ Непрерывное равномерное распределение

```

def gen_uniform_rvs(sample_size, param):
    """
    arg:    param = (\alpha, \beta)
    usage:  uniform(loc=\alpha, scale=\beta-\alpha)
    """
    return stats.uniform(
        loc=param[0],
        scale=param[1]-param[0]
    ).rvs(size=sample_size)

```

▼ Пуассоновское распределение

```

def gen_poisson_rvs(sample_size, param):
    """
    arg:    param = \lambda
    usage:  poisson(mu=\lambda)
    """
    return stats.poisson(mu=param).rvs(size=sample_size)

```

▼ Биномиальное распределение

```

def gen_bin_rvs(sample_size, param):
    """
    arg:    param = (n, p)
    usage:  binom(n, p)
    """

```

```
return stats.binom(param[0], param[1]).rvs(size=sample_size)
```

▼ Геометрическое распределение

```
def gen_geom_rvs(sample_size, param):
    """
    arg:    param = p
    usage:  geom(p)
    """
    return stats.geom(param).rvs(size=sample_size)
```

▼ Бета распределение

```
def gen_beta_rvs(sample_size, param):
    """
    arg:    param = (\alpha, \beta)
    usage:  beta(a=\alpha, b=\beta)
    """
    return stats.beta(
        a=param[0],
        b=param[1]
    ).rvs(size=sample_size)
```

▼ И словарь с этими же функциями-генераторами

```
rvs_generator = {
    "normal":  gen_normal_rvs,
    "gamma":   gen_gamma_rvs,
    "uniform": gen_uniform_rvs,
    "poiss":   gen_pois_rvs,
    "bin":     gen_bin_rvs,
    "geom":    gen_geom_rvs,
    "beta":    gen_beta_rvs
}
```

▼ Напишем функции подсчета оценки параметра методом моментов:

▼ Нормальное распределение

```
def calc_normal_est(sample):  
    return (sample.mean(), s(sample))
```

▼ Гамма распределение

```
def calc_gamma_est(sample):  
    mean, var = sample.mean(), s(sample)  
    return (mean**2 / var, var / mean)
```

▼ Непрерывное равномерное распределение

```
def calc_uniform_est(sample):  
    mean, var = sample.mean(), s(sample)  
    return (mean - (3*var)**0.5, mean + (3*var)**0.5)
```

▼ Пуассоновское распределение

```
def calc_poiss_est(sample):  
    return sample.mean()
```

▼ Биномиальное распределение

```
def calc_bin_est(sample):  
    mean, var = sample.mean(), s(sample)  
    return (mean**2 / (mean-var), (mean-var) / mean)
```

▼ Геометрическое распределение

```
def calc_geom_est(sample):
    return 1 / sample.mean()
```

▼ Бета распределение

```
def calc_beta_est(sample):
    mean, mean2, var = sample.mean(), (sample**2).mean(), s(sample)
    mult = (mean-mean2) / var
    return (mean * mult, (1-mean) * mult)
```

▼ И словарь с этими же функциями-оценками

```
est_calculator = {
    "normal": calc_normal_est,
    "gamma": calc_gamma_est,
    "uniform": calc_uniform_est,
    "poiss": calc_poiss_est,
    "bin": calc_bin_est,
    "geom": calc_geom_est,
    "beta": calc_beta_est
}
```

▼ Оценим дисперсию каждой оценки, сгенерировав для каждой из них $K = 1000$ бутстрепных выборок:

```
import functools
```

```
# декоратор для подсчета бутстрепной оценки дисперсии
```

```
def bootstrap(calc_bootstrap_var):
```

```
    @functools.wraps(calc_bootstrap_var)
```

```

def wrapper(*args, **kwargs):
    bootstrap_ests = calc_bootstrap_var(*args, **kwargs)
    param = kwargs['param']
    param_size = len(param) if type(param) is tuple else 1

    vars = np.zeros(param_size)

    if param_size > 1:
        for dim in range(param_size):
            vars[dim] = s(bootstrap_ests.T[dim])
    else:
        vars[0] = s(bootstrap_ests.T)

    return vars

return wrapper

```

▼ а) с помощью параметрического бутстрепа

```

@bootstrap
def calc_param_bootstrap_var(gen_rvs, sample_size, param, calc_est):
    param_bootstrap_samples = np.array([
        gen_rvs(sample_size, param)
        for _ in range (K)
    ])
    return np.array([calc_est(param_bootstrap_samples[i]) for i in range (K)])

```

▼ б) с помощью непараметрического бутстрепа

```

@bootstrap
def calc_non_param_bootstrap_var(gen_rvs, sample_size, param, calc_est):
    take_choices = np.random.randint(sample_size, size=(K, sample_size))
    sample = gen_rvs(sample_size, param)
    non_param_bootstrap_samples = np.array([
        np.take(sample, take_choices[i])

```



```

        for i in range (K)
    ])
    return np.array([
        calc_est(non_param_bootstrap_samples[i])
        for i in range (K)
    ])

```

▼ Проведем сами эксперименты

Эксперименты будем проводить для разных значений параметров распределений (например, для трёх), для это определим следующую коллекцию параметров `params_collection`:

```

params_collection = {
    "normal": [(0, 0.04), (0, 1), (0, 25)],
    "gamma": [(1, 1), (2, 1), (3, 1)],
    "uniform": [(0, 1), (0, 42), (-1, 1)],
    "poiss": [0.1, 0.2, 0.3, 0.5],
    "bin": [(20, 0.5), (20, 0.1), (40, 0.5)],
    "geom": [0.1, 0.2, 0.3, 0.5],
    "beta": [(0.5, 0.5), (5, 1), (1, 3)]
}

```

И сами серии экспериментов:

```

for distrib_nm in params_collection.keys():
    for param in params_collection[distrib_nm]:
        print(f"""
            Распределение: {distrib_nm}
            Параметр:      {param}
            Оценка дисперсии
            \tпараметрическим бутстрепом: {calc_param_bootstrap_var(rvs_generator[distrib_nm], N, param=param, calc_est=est_calculato
            \тнепараметрическим бутстрепом: {calc_non_param_bootstrap_var(rvs_generator[distrib_nm], N, param=param, calc_est=est_calcu
            """)

```



Распределение: normal
Параметр: (0, 0.04)
Оценка дисперсии
 параметрическим бутстрепом: [4.01196968e-05 2.97489084e-06]
 непараметрическим бутстрепом: [3.80451198e-05 2.59671846e-06]

Распределение: normal
Параметр: (0, 1)
Оценка дисперсии
 параметрическим бутстрепом: [0.00091755 0.00203285]
 непараметрическим бутстрепом: [0.00098447 0.00187921]

Распределение: normal
Параметр: (0, 25)
Оценка дисперсии
 параметрическим бутстрепом: [0.02599688 1.2725864]
 непараметрическим бутстрепом: [0.02482682 1.16074928]

Распределение: gamma
Параметр: (1, 1)
Оценка дисперсии
 параметрическим бутстрепом: [0.00394979 0.00480283]
 непараметрическим бутстрепом: [0.00446773 0.00637369]

Распределение: gamma
Параметр: (2, 1)
Оценка дисперсии
 параметрическим бутстрепом: [0.00382667 0.00117843]
 непараметрическим бутстрепом: [0.00366263 0.0011357]

Распределение: gamma
Параметр: (3, 1)
Оценка дисперсии
 параметрическим бутстрепом: [0.003789 0.00055188]
 непараметрическим бутстрепом: [0.00267971 0.00033225]

Распределение: uniform
Параметр: (0, 1)
Оценка дисперсии
 параметрическим бутстрепом: [0.00013541 0.00013372]
 непараметрическим бутстрепом: [0.00012233 0.00013838]

Распределение: uniform
Параметр: (0, 42)
Оценка дисперсии
 параметрическим бутстрепом: [0.22428116 0.25483221]
 непараметрическим бутстрепом: [0.25101208 0.2172571]

Распределение: uniform
Параметр: (-1, 1)
Оценка дисперсии
 параметрическим бутстрепом: [0.00052509 0.00053433]
 непараметрическим бутстрепом: [0.00055872 0.00046601]

Распределение: poiss
Параметр: 0.1
Оценка дисперсии
 параметрическим бутстрепом: [9.5312624e-05]
 непараметрическим бутстрепом: [8.4534816e-05]

Распределение: poiss
Параметр: 0.2
Оценка дисперсии
 параметрическим бутстрепом: [0.0002015]
 непараметрическим бутстрепом: [0.00023204]

Распределение: poiss
Параметр: 0.3
Оценка дисперсии
 параметрическим бутстрепом: [0.00032556]
 непараметрическим бутстрепом: [0.000356]

Распределение: `poiss`
Параметр: `0.5`
Оценка дисперсии
 параметрическим бутстрепом: `[0.00054497]`
 непараметрическим бутстрепом: `[0.00055058]`

Распределение: `bin`
Параметр: `(20, 0.5)`
Оценка дисперсии
 параметрическим бутстрепом: `[7.81138024e-01 4.98460913e-04]`
 непараметрическим бутстрепом: `[1.38978235e+00 5.87263041e-04]`

Распределение: `bin`
Параметр: `(20, 0.1)`
Оценка дисперсии
 параметрическим бутстрепом: `[2.80749064e+03 1.48351500e-03]`
 непараметрическим бутстрепом: `[1.14426518e+01 1.46312438e-03]`

Распределение: `bin`
Параметр: `(40, 0.5)`
Оценка дисперсии
 параметрическим бутстрепом: `[3.07071101e+00 4.75112108e-04]`
 непараметрическим бутстрепом: `[3.72968748e+00 4.92220942e-04]`

Распределение: `geom`
Параметр: `0.1`
Оценка дисперсии
 параметрическим бутстрепом: `[9.29256101e-06]`
 непараметрическим бутстрепом: `[1.01962578e-05]`

Распределение: `geom`
Параметр: `0.2`
Оценка дисперсии
 параметрическим бутстрепом: `[3.12180382e-05]`
 непараметрическим бутстрепом: `[3.65948932e-05]`

```
Распределение: geom
Параметр:      0.3
Оценка дисперсии
    параметрическим бутстрепом: [6.29562754e-05]
    непараметрическим бутстрепом: [7.05886367e-05]
```

```
Распределение: geom
Параметр:      0.5
Оценка дисперсии
    параметрическим бутстрепом: [0.00013283]
    непараметрическим бутстрепом: [0.00011182]
```

```
Распределение: beta
Параметр:      (0.5, 0.5)
Оценка дисперсии
    параметрическим бутстрепом: [0.00061576 0.00064493]
    непараметрическим бутстрепом: [0.00059119 0.00059942]
```

```
Распределение: beta
Параметр:      (5, 1)
Оценка дисперсии
    параметрическим бутстрепом: [0.07490786 0.00243359]
    непараметрическим бутстрепом: [0.08503042 0.00273205]
```

```
Распределение: beta
Параметр:      (1, 3)
Оценка дисперсии
    параметрическим бутстрепом: [0.00230771 0.02382598]
    непараметрическим бутстрепом: [0.00273445 0.02538831]
```

▼ Вывод №1

В бутстрепе мы не получаем новой информации, но разумно используем имеющиеся данные, исходя из поставленной задачи. Оценки дисперсии с помощью параметрического и непараметрического бутстрепа в нашей задаче численно близки: отличаются на малых порядках. Также было установлено, что если выборочное распределение статистической функции не включает в себя неизвестные по популяции, бутстреп-распределение позволяет получить довольно хорошее приближение к выборочному

Следующие задачи используют данные из .csv файла, находящегося на Google Drive. Т.к. этот ноутбук разрабатывался в Google Colab, то загрузку данных сделаем через официальные API.

```
!pip3 install -U -q PyDrive
```

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

```
def collect_csv_data_via_gd(link: str, filename: str) -> pd.core.frame.DataFrame:
    auth.authenticate_user()
    gauth = GoogleAuth()
    gauth.credentials = GoogleCredentials.get_application_default()
    drive = GoogleDrive(gauth)
    fluff, id = link.split('=')
    downloader = drive.CreateFile({'id':id})
    downloader.GetContentFile(filename)
    return pd.read_csv(filename)
```

▼ Задача №2

На высоте 1 метр от поверхности Земли закреплено устройство, которое периодически излучает лучи на поверхность Земли (считайте, что поверхность Земли представляет из себя прямую). Пусть l – перпендикуляр к поверхности Земли, опущенный из точки, в которой закреплено устройство. Угол к прямой l (под которым происходит излучение) устройство выбирает случайно из равномерного распределения на отрезке $(-\pi/2, \pi/2)$ (все выборы осуществляются независимо). В этих предположениях точки

пересечения с поверхностью имеют распределение Коши с плотностью $p(x) = \frac{1}{\pi(1+(x-x_0)^2)}$. Неизвестный параметр сдвига x_0 соответствует проекции точки расположения устройства на поверхность Земли (направление оси и начало координат на поверхности Земли выбраны заранее некоторым образом независимо от расположения устройства). В файле *Cauchy.csv* находятся координаты точек пересечения лучей с поверхностью Земли. Оцените параметр сдвига методом максимального правдоподобия а) по половине выборки (первые 500 элементов выборки, т.е. выборка состоит из 1000 наблюдений); б) по всей выборке. Оценку произведите по сетке (т.е. возьмите набор точек с некоторым шагом и верните ту, на которой достигается максимум функции правдоподобия). Известно, что параметр масштаба принадлежит интервалу $[-1000, 1000]$. Выберите шаг равным 0.01. Если получается долго или не хватает памяти, то уменьшите интервал поиска и поясните (в комментариях), почему берете именно такой интервал

Получим данные из файла Cauchy.csv прямо из Google Drive

```
df = collect_csv_data_via_gd(
    link="https://drive.google.com/open?id=1Qt0g3nsNfkmZCKUk6861emuPEbsoun3I",
    filename='Cauchy.csv')
cauchy_sample = np.array(df.iloc[:,0])
cauchy_half_sample = cauchy_sample[:500]
```

Функция максимального правдоподобия параметра θ ($\theta = x_0$) для распределения Коши с плотностью $p(x) = \frac{1}{\pi(1+(x-\theta)^2)}$ выборки из n элементов есть:

$$\mathcal{L}_X(\theta) = \prod_{i=1}^n \frac{1}{\pi(1+(X_i-\theta)^2)} = \frac{1}{\pi^n} \cdot \prod_{i=1}^n \frac{1}{1+(X_i-\theta)^2}$$

Логарифмическая функция максимального правдоподобия:

$$L_X(\theta) = -n \cdot \ln(\pi) - \sum_{i=1}^n \ln(1+(X_i-\theta)^2)$$


```
def compute_log_maximum_likelihood_fun(theta, sample):
    n = sample.size
    return -n*np.log(np.pi) - np.sum([np.log(1+(rv-theta)**2) for rv in sample])
```

Сгенерируем сетку:

```
cauchy_probe_params = np.arange(-1000, 1000, .01)
```


Оценим параметр сдвига методом максимального правдоподобия по половине выборки (первые 500 элементов выборки):

```
max_index = np.argmax([
    compute_log_maximum_likelihood_fun(theta, cauchy_half_sample)
    for theta in cauchy_probe_params
])
print(cauchy_probe_params[max_index])
```

 662.039999999831

А также по всей выборке:

```
max_index = np.argmax([
    compute_log_maximum_likelihood_fun(theta, cauchy_sample)
    for theta in cauchy_probe_params
])
print(cauchy_probe_params[max_index])
```

 662.049999999831

▼ Вывод №2

Как видим, из полученных выше результатов вычислений, что при части выборки, что при всей, получилось почти тот же результат. Поэтому от значительного увеличения выборки результат сильно не меняется. Поэтому достаточно рассмотреть половину. Также заметно значительное уменьшение скорости вычислений при меньшем размере выборки.

▼ Задача №3

В банке каждую минуту подсчитывается баланс по сравнению с началом дня (6 часов утра). В полночь работники банка измеряют две величины: X^1 – максимальное значение баланса за день, X^2 – значение баланса в полночь. Считается, что величина $X = X^1 - X^2$ имеет распределение Вейбулла с функцией распределения $F(x) = 1 - e^{-x^\gamma}$ ($x > 0$), где $\gamma > 0$ – параметр формы. В течение 10 лет каждый день банк проводил измерение величины X , получив в результате выборку X_1, \dots, X_{3652} . В файле Weibull.csv находятся соответствующие измерения. Оцените параметр формы методом максимального правдоподобия а) по первым 4 годам; б) по всей выборке. Оценку произведите по сетке (в логарифмической шкале). Известно, что $\log_{10} \gamma \in [-2, 2]$. Выберите шаг равным 10^{-3} .

Получим данные из файла Weibull.csv прямо из Google Drive. Также обработаем нули по входных данных, заменив на достаточно малое число, например, 10^{-8} .

```
from scipy.stats import invweibull as weibull

df = collect_csv_data_via_gd(
    link="https://drive.google.com/open?id=16nS4rd0NIecdWF4UrfzM8d7GRVyMq6Mu",
    filename='Weibull.csv')
weibull_sample = np.array(df.iloc[:,0]).map(
    lambda x: 10**-8 if np.abs(x) < 10**-8 else x
))

weibull_subsample = weibull_sample[:365*3 + 366 + 1] # 1 высокосный год
```

Найдем плотность, функцию максимального правдоподобия и логарифмическую функцию максимального правдоподобия:

$$p(x) = F'(x) = \gamma \cdot x^{\gamma-1} \cdot e^{-x^\gamma}$$

$$\mathcal{L}_X(\theta) = \prod_{i=1}^n \gamma \cdot X_i^{\gamma-1} \cdot e^{-X_i^\gamma} = \gamma^n \cdot \prod_{i=1}^n X_i^{\gamma-1} \cdot e^{-X_i^\gamma}$$

$$L_X(\theta) = n \cdot \ln(\gamma) + (\gamma - 1) \cdot \sum_{i=1}^n \ln(X_i) - \sum_{i=1}^n X_i^\gamma$$