

# Математическая статистика

## Практическое задание 4

В данном задании предлагается провести некоторое исследование доверительных интервалов и байесовских методов.

### Правила:

- Выполненную работу нужно отправить на почту `probability.diht@yandex.ru`, указав тему письма "[номер группы] Фамилия Имя - Задание 4". Квадратные скобки обязательны. Вместо Фамилия Имя нужно подставить свои фамилию и имя.
- Прислать нужно ноутбук и его pdf-версию. Названия файлов должны быть такими: `4.N.ipynb` и `4.N.pdf`, где N - ваш номер из таблицы с оценками.
- Никакой код из данного задания при проверке запускаться не будет.
- Некоторые задачи отмечены символом <sup>\*</sup>. Эти задачи являются дополнительными. Успешное выполнение большей части таких задач (за все задания) является необходимым условием получения бонусного балла за практическую часть курса.
- Баллы за каждую задачу указаны далее. Если сумма баллов за задание меньше 25% (без учета доп. задач), то все задание оценивается в 0 баллов.

### Баллы за задание:

- Задача 1 - 5 баллов
- Задача 2<sup>\*</sup> - 3 балла
- Задача 3<sup>\*</sup> - 3 балла
- Задача 4 - 5 баллов
- Задача 5<sup>\*</sup> - 2 балла
- Задача 6 - 4 балла
- Задача 7 - 1 балл
- Задача 8 - 3 балла
- Задача 9<sup>\*</sup> - 5 баллов
- Задача 10 - 5 баллов
- Задача 11<sup>\*</sup> - 3 балла

```
In [1]: import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt

%matplotlib inline
```

## 1. Доверительные интервалы

**Задача 1.** В этой задаче нужно визуализировать доверительные интервалы для выборок из различных распределений. Чтобы не плодить код, напишите следующую функцию. Пример

построения есть в ячейке №22 ноутбука python\_6.

```
In [61]: def draw_confidence_interval(left, # левая граница интервала
                                     right, # правая граница интервала
                                     estimation=None, # если задана, то рисуется график
                                     sample=None, # если задано, то рисуются точки выборки
                                     ylim=(None, None)): # ограничение по оси y

    plt.figure(figsize=(15, 8))
    n = np.arange(1, 101)
    if sample is not None:
        plt.scatter(n, sample, alpha=0.2, s=40, label='sample') # s - размер точек
    if estimation is not None:
        plt.plot(n, estimation, color='red', linewidth=2.5, label='sample mean $\\mu$')
    # заполняет пространство между двумя функциями
    plt.fill_between(n, left, right, alpha=0.15, label='intervals')
    plt.legend()
    plt.xlim((-2, 102)) # размеры графика по горизонтальной оси (ставим None, если не задано)
    plt.ylim(ylim) # размеры графика по вертикальной оси
    plt.title('Доверительный интервал') # имя графика
    plt.grid() # добавляем сетку
    plt.show()
```

Сгенерируйте выборки и постройте графики доверительных интервалов по следующей схеме.

- Выборка из распределения  $\mathcal{N}(0, 1)$ ; точный доверительный интервал минимальной длины в модели  $\mathcal{N}(\theta, 1)$ ; нужно нанести на график точки выборки.
- Выборка из распределения  $U[0, 1]$ ; точный доверительный интервал минимальной длины в модели  $U[0, \theta]$  на основе статистики  $X_{(n)}$ ; нужно нанести на график точки выборки.
- Выборка из распределения  $\Gamma(3, 2)$ ; точный асимптотический доверительный интервал в модели  $\Gamma(\theta, 2)$ ; точки выборки наносить на график не нужно.
- Выборка из стандартного распределения Коши; точный асимптотический доверительный интервал в модели распределения Коши со сдвигом; нужно нанести на график точки выборки.
- Выборка из стандартного распределения Коши; точный доверительный интервал минимальной длины в модели  $\mathcal{N}(\theta, 1)$ ; нужно нанести на график точки выборки.

Генерировать выборки размера 100, уровень доверия брать  $\alpha = 0.95$ . Для вычисления квантилей у каждого распределения из `scipy.stats` есть функция `ppf`.

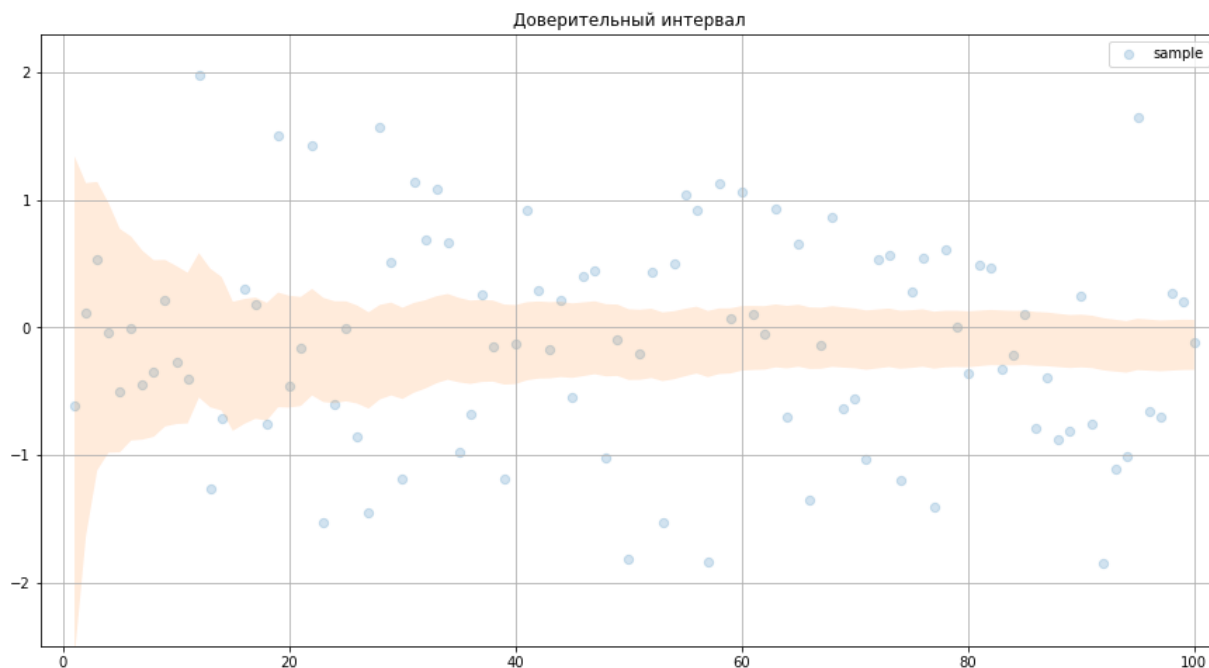
Сделайте вывод. Насколько часто истинное значение параметра попадает в доверительный интервал? Как длина интервала зависит от размера выборки?

Доверительный интервал  $\theta$  в модели  $\mathcal{N}(\theta, 1)$ :

$$\left( \bar{X} - \frac{z_{\frac{\alpha+1}{2}}}{\sqrt{n}}, \bar{X} + \frac{z_{\frac{\alpha+1}{2}}}{\sqrt{n}} \right)$$

In [37]: *# нормальное распределение*

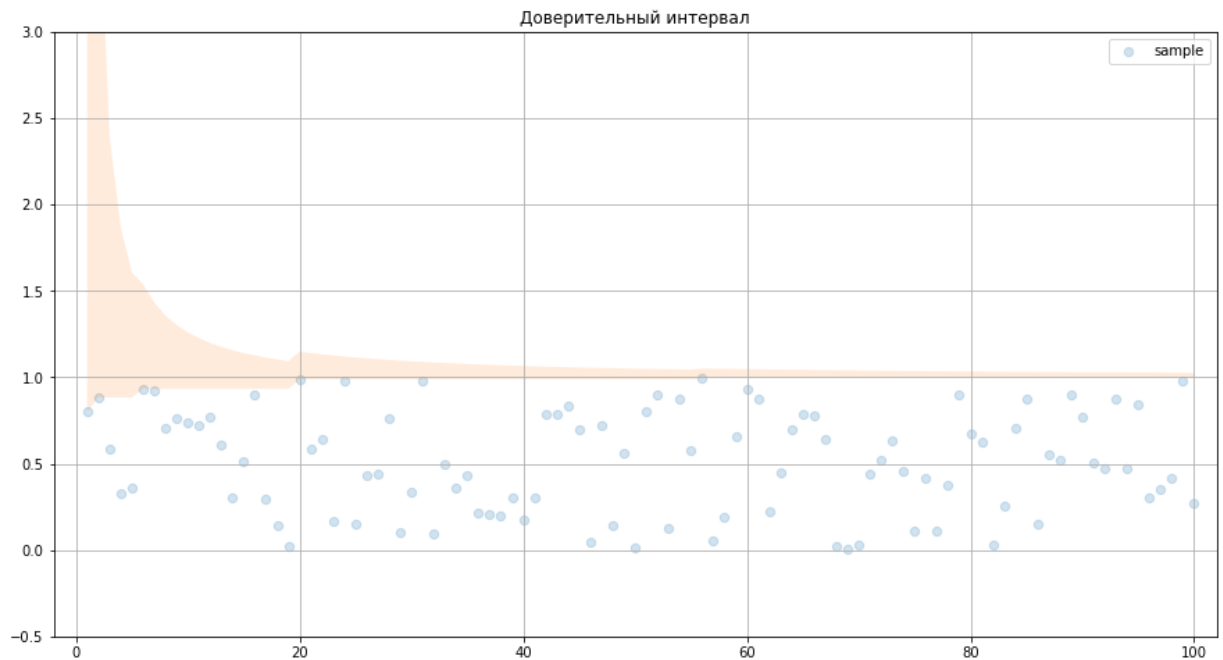
```
sample = sps.norm(loc = 0, scale = 1).rvs(size=100)
alpha = 0.95
means = list(map(lambda n: np.mean(sample[:n]), range(1, 101)))
delta = list(map(lambda n: sps.norm.ppf((alpha + 1) / 2) / np.sqrt(n), range(1, 101)))
left = [means[i] - delta[i] for i in range(0, 100)]
right = [means[i] + delta[i] for i in range(0, 100)]
draw_confidence_interval(left, right, sample=sample, ylim=(-2.5, 2.3))
```



Доверительный интервал  $\theta$  в модели  $U[0, \theta]$ :

$$\left( \frac{X_{(n)}}{\sqrt[n]{1 - \alpha}}, X_{(n)} \right)$$

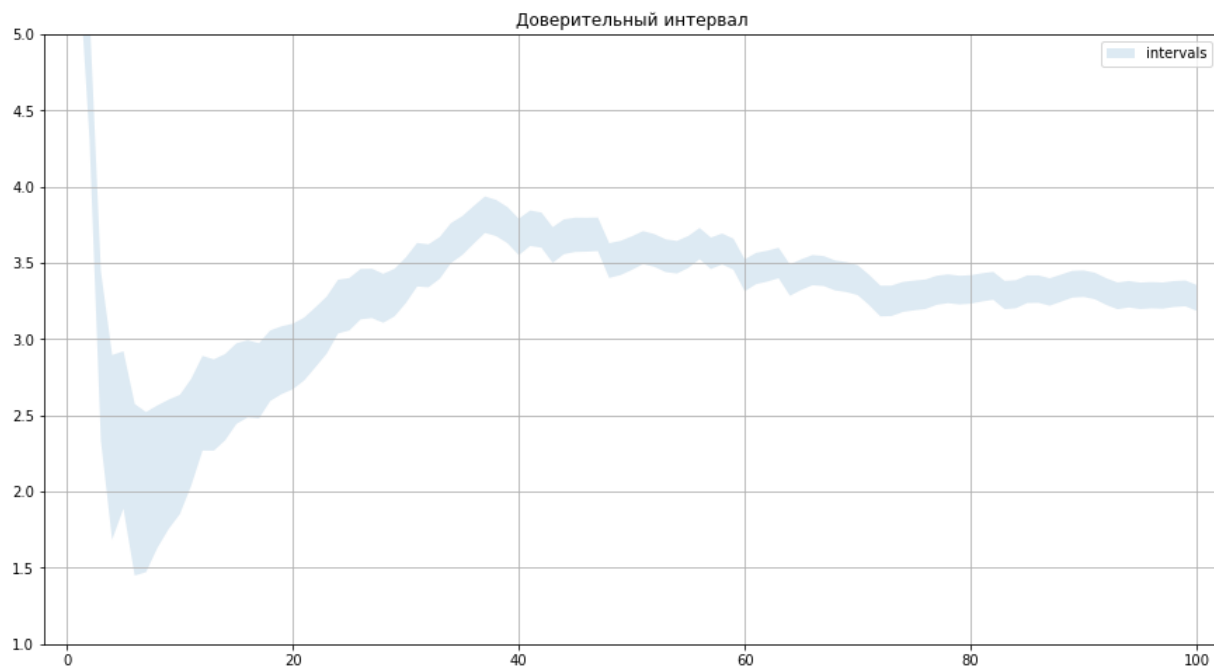
```
In [47]: #равномерное распределение
sps.uniform.rvs(size = 100)
alpha = 0.95
n_th_stat = np.maximum.accumulate(sample)
left = n_th_stat / ((1 - alpha) ** (1 / np.arange(1, 101)))
right = n_th_stat
draw_confidence_interval(left, right, sample=sample, ylim=(-0.5, 3))
```



Доверительный интервал  $\theta$  в модели  $\Gamma(\theta, 2)$ :

$$\left( \frac{2}{\bar{X}} - \frac{\bar{X} * z_{\frac{\alpha+1}{2}}}{\sqrt{2n}}, \frac{2}{\bar{X}} - \frac{\bar{X} * z_{\frac{-\alpha+1}{2}}}{\sqrt{2n}} \right)$$

```
In [62]: #Гамма-распределение
sample = sps.gamma(a = 2, scale = 1/3).rvs(size = 100)
alpha = 0.95
means = np.cumsum(sample) / range(1, 101)
left = 2 / means - means * sps.norm.ppf((alpha + 1) / 2) / (np.sqrt(2) * np.sqrt(
right = 2 / means - means * sps.norm.ppf((-alpha + 1) / 2) / (np.sqrt(2) * np.sqr
draw_confidence_interval(left, right, ylim=(1, 5))
```

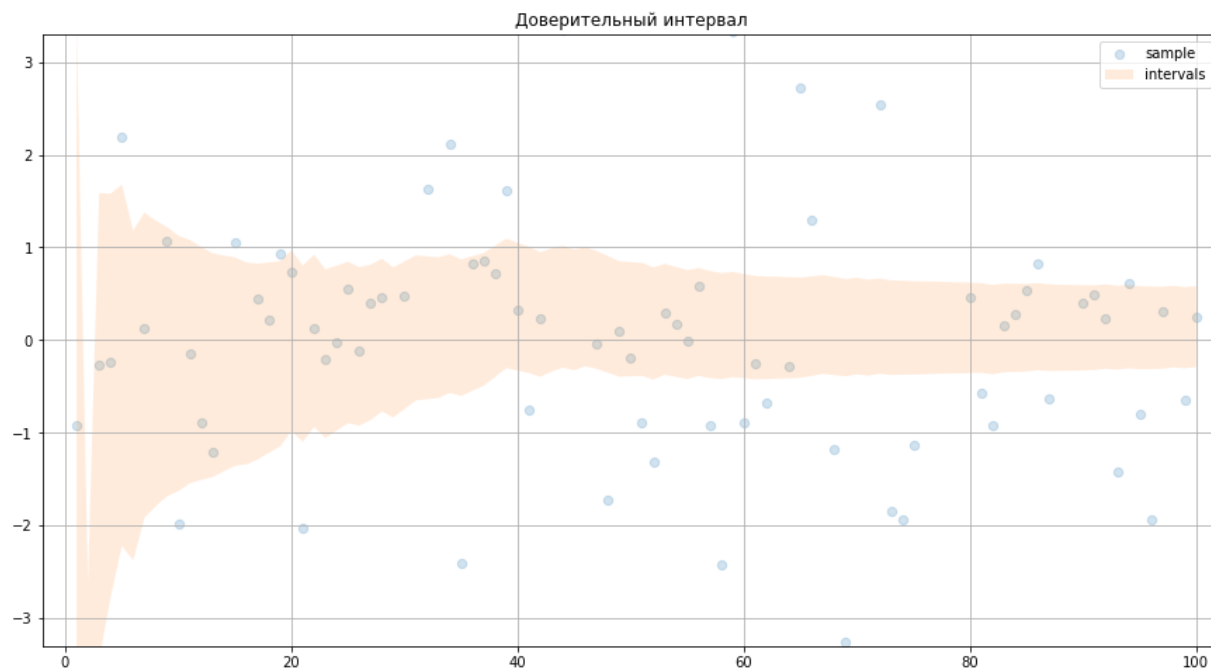


Доверительный интервал  $\theta$  в модели распределения Коши со сдвигом:

$$\left( z_{0.5} - \frac{z_{\frac{\alpha+1}{2} * \pi}}{\sqrt{2n}}, z_{0.5} + \frac{z_{\frac{-\alpha+1}{2} * \pi}}{\sqrt{2n}} \right)$$

In [66]: *# Распределение Коши*

```
sample = sps.cauchy.rvs(size = 100)
alpha = 0.95
medians = np.array(list(map(lambda n: np.median(sample[:n]), range(1, 101))))
left = medians - sps.norm.ppf((alpha + 1) / 2) * np.pi / (np.sqrt(2) * np.sqrt(n))
right = medians + sps.norm.ppf((alpha + 1) / 2) * np.pi / (np.sqrt(2) * np.sqrt(n))
draw_confidence_interval(left, right, sample=sample, ylim=(-3.3, 3.3))
```

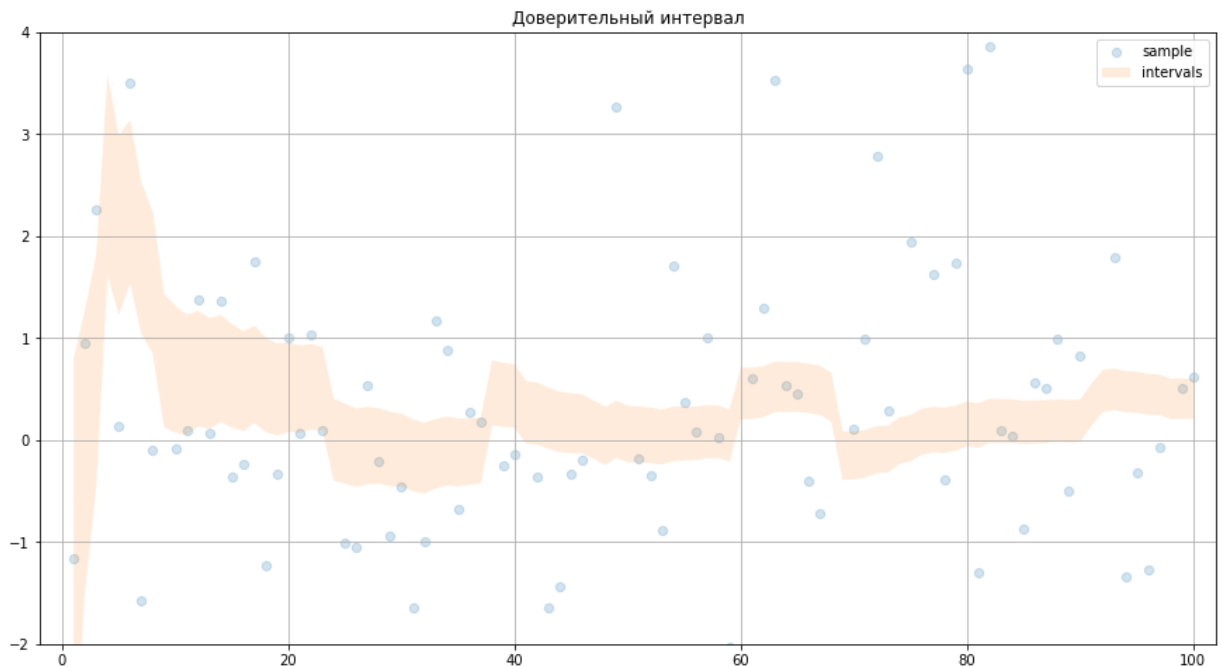


Доверительный интервал для  $\theta$  в модели  $\mathcal{N}(\theta, 1)$ :

$$\left( \bar{X} - \frac{z_{\frac{\alpha+1}{2}}}{\sqrt{n}}, \bar{X} + \frac{z_{\frac{\alpha+1}{2}}}{\sqrt{n}} \right)$$

In [78]: # Распределение Коши

```
sample = sps.cauchy.rvs(size = 100)
alpha = 0.95
means = np.cumsum(sample) / range(1, 101)
left = means - sps.norm.ppf((alpha + 1) / 2) / np.sqrt(range(1, 101))
right = means + sps.norm.ppf((alpha + 1) / 2) / np.sqrt(range(1, 101))
draw_confidence_interval(left, right, sample=sample, ylim=(-2, 4))
```



**Вывод** Я построил графики доверительных интервалов. Чем больше размер выборки, тем меньше доверительный интервал. В данных выборках, истинное значение параметра довольно часто попадало в доверительный интервал.

**Задача 2\***. Аналогично заданию 1 постройте доверительные интервалы для следующих случаев

- Выборка из распределения  $\Gamma(3, 2)$ ; точный асимптотический доверительный интервал для  $\theta$  в модели  $\Gamma(\theta, \beta)$ , причем  $\beta$  неизвестно; точки выборки наносить на график не нужно. Сравните с интервалом для случая известного  $\beta$ .
- Выборка из распределения  $\Gamma(3, 2)$ ; точный асимптотический доверительный интервал для  $\beta$  в модели  $\Gamma(\theta, \beta)$ , причем  $\theta$  неизвестно; точки выборки наносить на график не нужно.

**Задача 3\***. Сгенерируйте выборку размера 200 из распределения  $\mathcal{N}((0, 0)^T, ((2, 1)^T, (1, 3)^T))$ . Постройте точную доверительную область для  $\theta$  в модели  $\mathcal{N}(\theta, ((2, 1)^T, (1, 3)^T))$ . Нанесите на график точки выборки.

**Задача 4.** При использовании асимптотических доверительных интервалов важно понимать, какой размер выборки является достаточным для хорошего приближения. Иначе говоря, пусть  $\xi_n \xrightarrow{d} \mathcal{N}(0, 1)$ . Начиная с какого  $n$  распределение статистики  $\xi_n$  хорошо приближается

нормальным распределением?

Для ответа на этот вопрос проведите следующее исследование. Сгенерируйте  $K = 10^5$  выборок  $(X_{i,k}, i \leq N)$  размера  $N = 300$ , где  $k \leq K$  --- номер выборки. Для каждой подвыборки  $k$ -ой выборки  $(X_{i,k}, i \leq n)$  посчитайте значение статистики  $T_{n,k}$  (определение далее) для всех  $n \leq N$ . Далее для каждого фиксированного  $n$  постройте эмпирическую функцию распределения  $F_n^*$  по выборке  $(T_{n,k}, k \leq K)$  и посчитайте точное значение статистики  $D_n = \sup_{x \in \mathbb{R}} |F_n^*(x) - F(x)|$ , где  $F$  --- функция распределения  $\mathcal{N}(0, 1)$  (см. задачу 4 задания 1). Постройте график зависимости  $D_n$  от  $n$ .

Рассмотрите следующие случаи

- $X_1, \dots, X_n \sim \mathcal{N}(0, 1)$ . Рассмотреть  $T = \sqrt{n} \cdot \bar{X}$  и  $T = \sqrt{n} \cdot \bar{X} / \sqrt{S^2}$ .
- $X_1, \dots, X_n \sim \text{Bern}(p), p = 0.5$ . Рассмотреть  $T = \sqrt{n} \frac{\bar{X} - p}{\sqrt{p(1-p)}}$  и  $T_n = \sqrt{n} \frac{\bar{X} - p}{\sqrt{S^2}}$ .
- $X_1, \dots, X_n \sim \text{Cauchy}$ . Рассмотреть  $T = \sqrt{n} \frac{\hat{\mu}}{\pi/2}$ .

В первых двух пунктах нужно построить две зависимости на одном графике для сравнения. Масштаб графика должен быть таким, чтобы четко можно было увидеть различие между двумя статистиками. Например, поставьте ограничение сверху по оси  $y$  на 0.05. Не забудьте добавить сетку и легенду.

Старайтесь не копировать много кода, пишите вспомогательные функции. Обратите внимание, что оптимальный код для первых двух пунктов выполняется за 30 секунд, для третьего --- за 3 минуты. Неоптимальный код может выполняться более часа.

Сделайте вывод о том, в каком случае распределение статистики быстрее приближается нормальным распределением. Начиная с какого размера выборки можно пользоваться приближением нормальным распределением?

```
In [9]: from statsmodels.distributions.empirical_distribution import ECDF
```

```
In [39]: def Glivenko_Kantelli (sample, ecdf, cdf):
    sample_sort = np.sort(sample)
    ecdf_vector = np.hstack((0, ecdf(sample_sort)))
    cdf_vector = cdf(sample_sort)
    ans = np.max(np.abs(np.hstack((cdf_vector - ecdf_vector[0:len(sample_sort)]),
    return ans
```

- $X_1, \dots, X_n \sim \mathcal{N}(0, 1)$ . Рассмотреть  $T = \sqrt{n} \cdot \bar{X}$  и  $T = \sqrt{n} \cdot \bar{X} / \sqrt{S^2}$ .



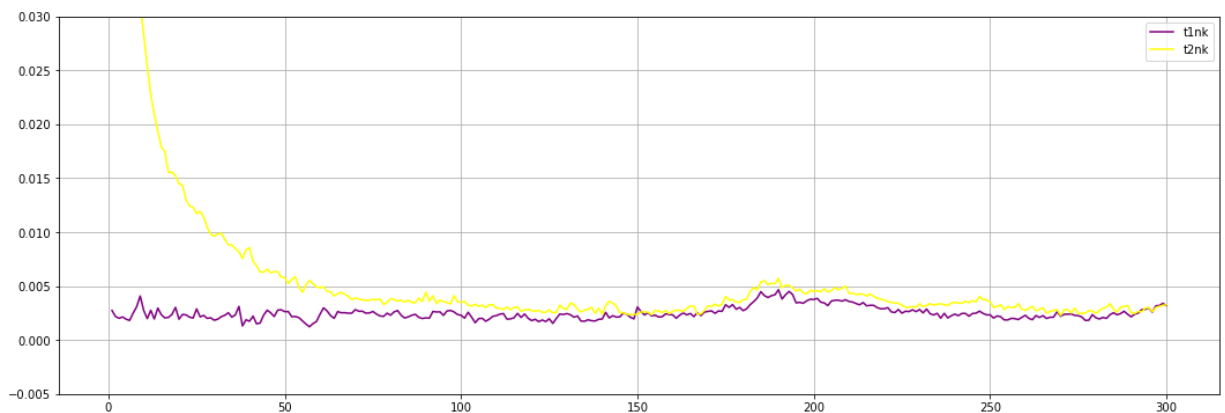
```
In [47]: n = 300
k = 100000
samples = sps.norm.rvs(size=(k, n))
sqr_t = lambda sample: np.cumsum(sample ** 2) / np.arange(1, n + 1) - (np.cumsum(
t1 = lambda sample: np.cumsum(sample) / np.arange(1, n + 1) * np.sqrt(np.arange(1
t2 = lambda sample: t1(sample) / np.sqrt(sqr_t(sample))

t1nk = np.array(list(map(lambda sample: t1(sample), samples)))
t2nk = np.array(list(map(lambda sample: t2(sample), samples)))
```

C:\Users\Maxim\Anaconda3\lib\site-packages\ipykernel\\_\_main\_\_.py:6: RuntimeWarning: divide by zero encountered in true\_divide

```
In [49]: plt.figure(figsize=(18,6))
ecdf1 = list(map(lambda sample: ECDF(sample), t1nk.T))
disperse1 = np.array(list(map(lambda n: Glivenko_Kantelli(t1nk.T[n], ecdf1[n], sp
ecdf2 = list(map(lambda sample: ECDF(sample), t2nk.T))
disperse2 = np.array(list(map(lambda n: Glivenko_Kantelli(t2nk.T[n], ecdf2[n], sp

plt.plot(np.arange(1, n + 1), disperse1, color='purple', label='t1nk')
plt.plot(np.arange(1, n + 1), disperse2, color='yellow', label='t2nk')
plt.ylim(-0.005, 0.03)
plt.legend(loc='best')
plt.grid()
plt.show()
```



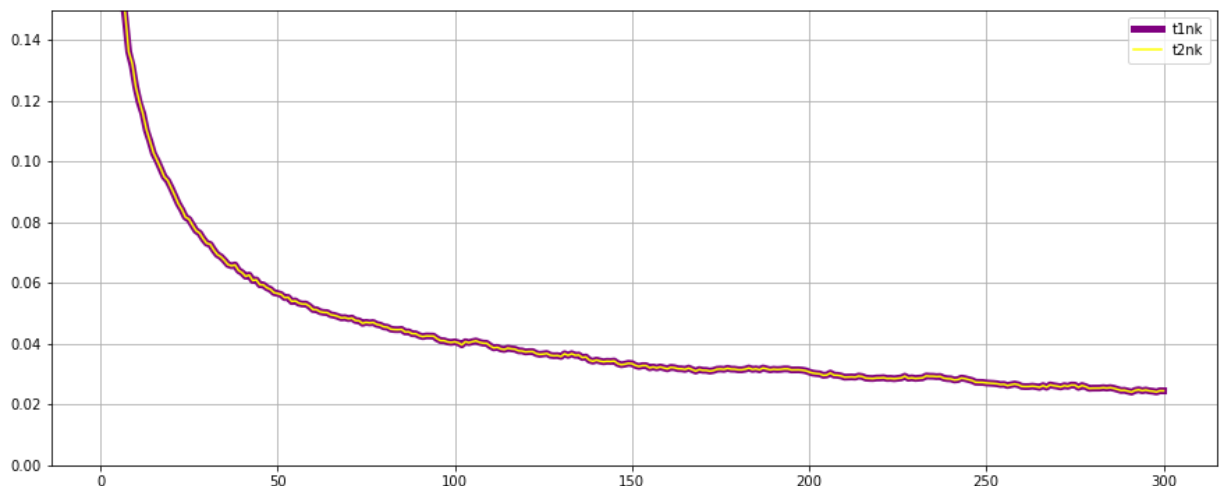
- $X_1, \dots, X_n \sim \text{Bern}(p), p = 0.5$  Рассмотреть  $T = \sqrt{n} \frac{\bar{X} - p}{\sqrt{p(1-p)}}$  и  $T_n = \sqrt{n} \frac{\bar{X} - p}{\sqrt{S^2}}$ .

```
In [26]: p = 0.5
n = 300
k = 100000
samples = sps.bernoulli(p).rvs(size=(k, n))
means = lambda sample: np.cumsum(sample) / np.arange(1, n + 1)
sqr_t = lambda sample: np.cumsum(sample ** 2) / np.arange(1, n + 1) - (np.cumsum(
t1 = lambda sample: np.sqrt(np.arange(1, n + 1)) * (means(sample) - p) / np.sqrt(
t2 = lambda sample: np.sqrt(np.arange(1, n + 1)) * (means(sample) - p) / np.sqrt(
t1nk = np.array(list(map(lambda sample: t1(sample), samples)))
t2nk = np.array(list(map(lambda sample: t2(sample), samples)))
```

C:\Users\Maxim\Anaconda3\lib\site-packages\ipykernel\\_\_main\_\_.py:8: RuntimeWarning: divide by zero encountered in true\_divide

```
In [35]: plt.figure(figsize=(15,6))
ecdf1 = list(map(lambda sample: ECDF(sample), t1nk.T))
disperse1 = np.array(list(map(lambda n: Glivenko_Kantelli(t1nk.T[n], ecdf1[n], sp
ecdf2 = list(map(lambda sample: ECDF(sample), t2nk.T))
disperse2 = np.array(list(map(lambda n: Glivenko_Kantelli(t2nk.T[n], ecdf2[n], sp

plt.plot(np.arange(1, n + 1), disperse1, color='purple', label='t1nk', lw=5)
plt.plot(np.arange(1, n + 1), disperse2, color='yellow', label='t2nk', lw=1.5)
plt.ylim(-0.00005, 0.15)
plt.legend(loc='best')
plt.grid()
plt.show()
```

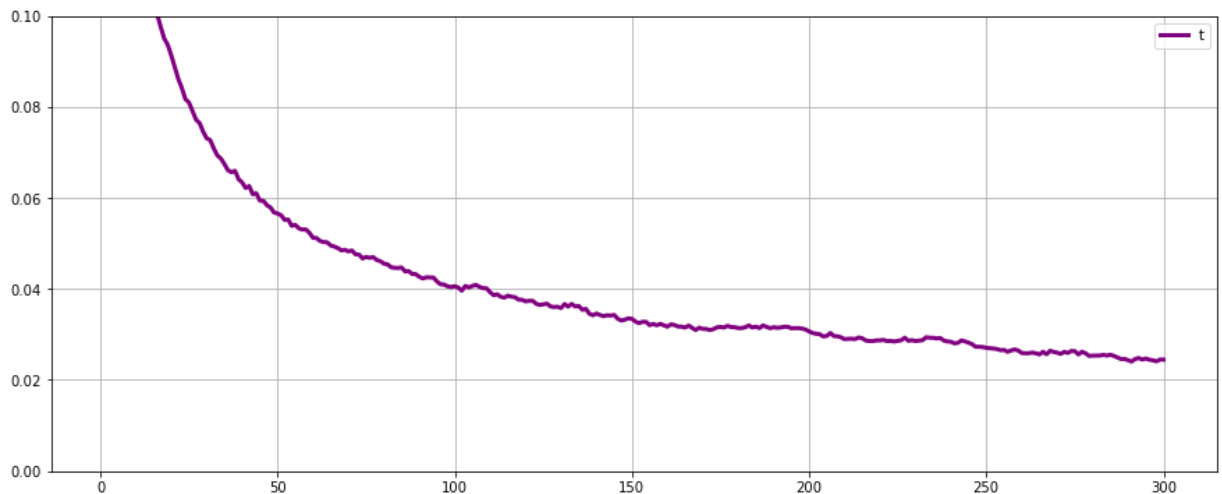


- $X_1, \dots, X_n \sim \text{Cauchy}$ . Рассмотреть  $T = \sqrt{n} \frac{\hat{\mu}}{\pi/2}$ .

```
In [37]: n = 300
k = 100000
samples = sps.cauchy.rvs(size=(k, n))
t = np.array(list(map(lambda n: np.sqrt(n) * np.median(samples[:, :n], axis=1) /
```

```
In [42]: plt.figure(figsize=(15,6))
ecdf = list(map(lambda sample: ECDF(sample), t.T))
disperse = np.array(list(map(lambda n: Glivenko_Kantelli(t1nk.T[n], ecdf1[n], sps

plt.plot(np.arange(1, n + 1), disperse, color='purple', label='t', lw=3)
plt.ylim(-0.00005, 0.10)
plt.legend(loc='best')
plt.grid()
plt.show()
```



**Вывод** Распределение статистики хорошо приближается к нормальному во втором случае уже начиная с маленького размера выборки. В первом случае только начиная с выборки размера 100-150

**Задача 5\***. Проведите исследование аналогичное задаче 4 для статистик из задачи 2.

**Задача 6.** Реализуйте следующую функцию для выборки из нормального распределения

```
In [ ]: def normal_summary(sample):
    print('size: %d' % ...)
    print('sample mean: %.2f' % ...)
    print('sample median: %.2f' % ...)
    print('sample std: %.2f' % ...) # стандартное отклонение = корень из дисперс
    print('0.95 confidence interval: (%.2f, %.2f)' % (...))
    print('KS-stat: %.3f' % ...) # значение статистики из теоремы Колмогорова-См
                                # взяв в качестве F функцию распределения норма
                                # распределения с оцененными выше параметрами
```

Протестируйте функцию на выборках из нормального распределения и на выборках из других распределений. Какой вывод можно сделать о поведении статистики Колмогорова-Смирнова?

Скачайте данные <http://archive.ics.uci.edu/ml/datasets/Wine>  
(<http://archive.ics.uci.edu/ml/datasets/Wine>), файл wine.data. Что вы можете сказать про

столбцы 1, 4, 8 (нумерация с нуля), соответствующие 'Alcohol', 'Alcalinity of ash', 'Nonflavanoid phenols'?

## 2. Байесовские методы

**Задача 7.** Пусть  $X_1, \dots, X_n \sim \mathcal{N}(\theta, 1)$  и  $\theta$  имеет априорное распределение Коши. Как было сказано на лекции, аналитически интеграл в знаменателе формулы Байеса посчитать не удастся. Однако, поскольку в данном случае параметр один, можно его посчитать с помощью приближенного интегрирования.

В качестве метода приближенного интегрирования можно использовать следующую модификацию известного метода Монте-Карло. В качестве оценки интеграла  $\int_{\mathbb{R}} f(x)p(x)dx$ ,

где  $p(x)$  --- некоторая плотность, можно взять величину  $\sum_{j=1}^k f(Y_j)$ , где  $Y_1, \dots, Y_k$  ---

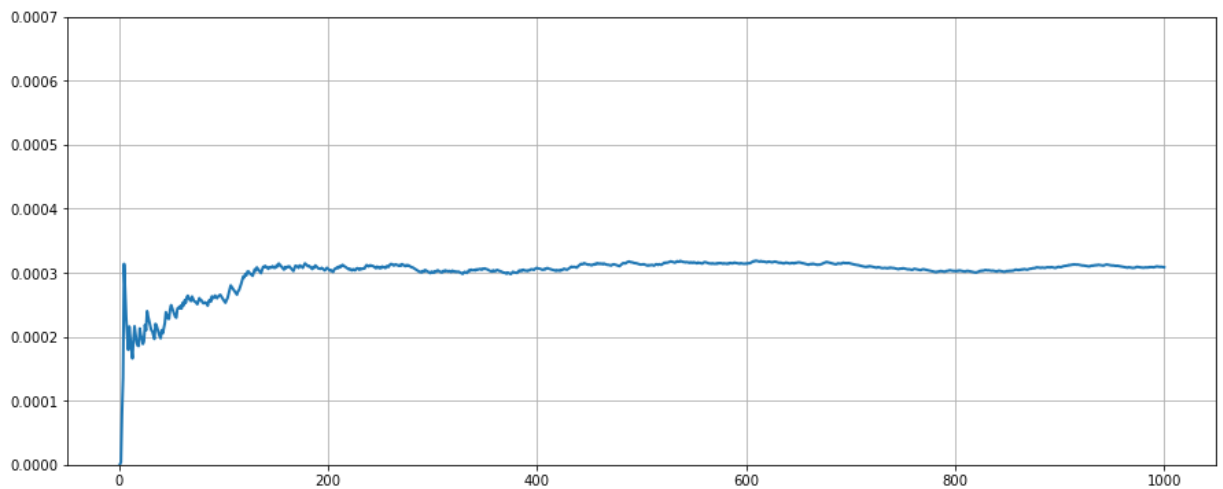
сгенерированная выборка из распределения, имеющего плотность  $p(x)$ .

Сгенерируйте выборку размера 5 из стандартного нормального распределения. Посчитайте для нее  $c$  --- знаменатель в формуле Байеса. Какой размер вспомогательной выборки в методе приближенного интегрирования необходим, чтобы с большой точностью посчитать значение  $c$ ?

Нарисуйте график плотности апостериорного распределения. Посчитайте математическое ожидание по апостериорному распределению.

```
In [80]: sample = sps.norm.rvs(size=5)
n = 1000
cauchy_sample = sps.cauchy.rvs(size=n)
func = np.vectorize(lambda t: 1 / (np.pi * 2) ** (5 / 2) * np.exp(-1/2 * np.sum((
approximate_values = np.cumsum(func(cauchy_sample)) / np.arange(1, n + 1)
```

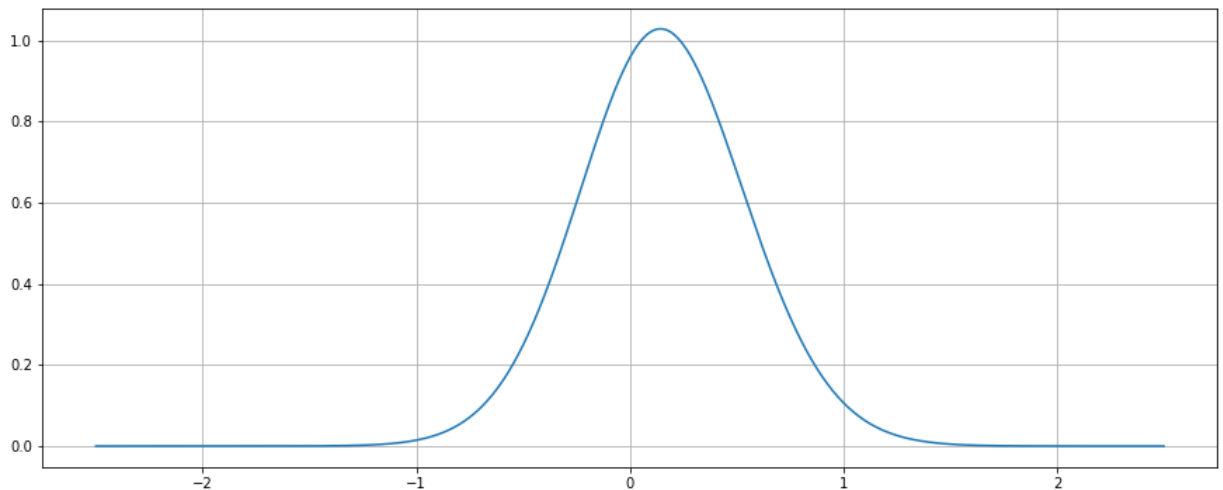
```
In [81]: plt.figure(figsize=(15,6))
plt.plot(np.arange(1, n + 1), approximate_values, lw=2)
plt.ylim(0, 0.0007)
plt.grid()
plt.show()
```



```
In [83]: # найдём знаменатель в формуле Байеса:

c = approximate_values[-1]
x = np.linspace(-2.5, 2.5, 1000)
plt.figure(figsize=(15, 6))
plt.plot(x, func(x) * sps.cauchy.pdf(x) / c)
plt.grid()
plt.show()

expect = 1 / (n * c) * np.sum(cauchy_sample * func(cauchy_sample))
print("Expectation = ", expect)
```



Expectation = 0.15513803956

**Вывод** Я смог посчитать неберущийся в привычном смысле интеграл с помощью метода Монте-Карло

**Задача 8.** Рассмотрим схему испытаний Бернулли (т.е. броски монет) с вероятностью успеха  $p$ .

Постройте несколько графиков сопряженного распределения для разных параметров и охарактеризуйте, как значения параметров его соотносятся с априорными знаниями о монете. Это могут быть, например, знания вида

- монета скорее честная (при таком априорном распределении наиболее вероятны значения  $p$  в окрестности 0.5)
- монета скорее нечестная, перевес неизвестен (наименее вероятны значения  $p$  в окрестности 0.5)
- монета скорее нечестная, перевес в сторону герба (наиболее вероятны значения  $p$  в окрестности 1)
- монета скорее честная, либо с небольшим перекосом вправо (наиболее вероятны значения  $p$  в окрестности  $\sim 0.6$ )
- ничего не известно (все значения равновероятны)

Для каждого случая из перечисленных выше постройте график плотности сопряженного распределения (на одной фигуре).

Постройте несколько графиков сопряженного распределения для разных параметров и охарактеризуйте, как значения параметров его соотносятся с априорными знаниями о монете.

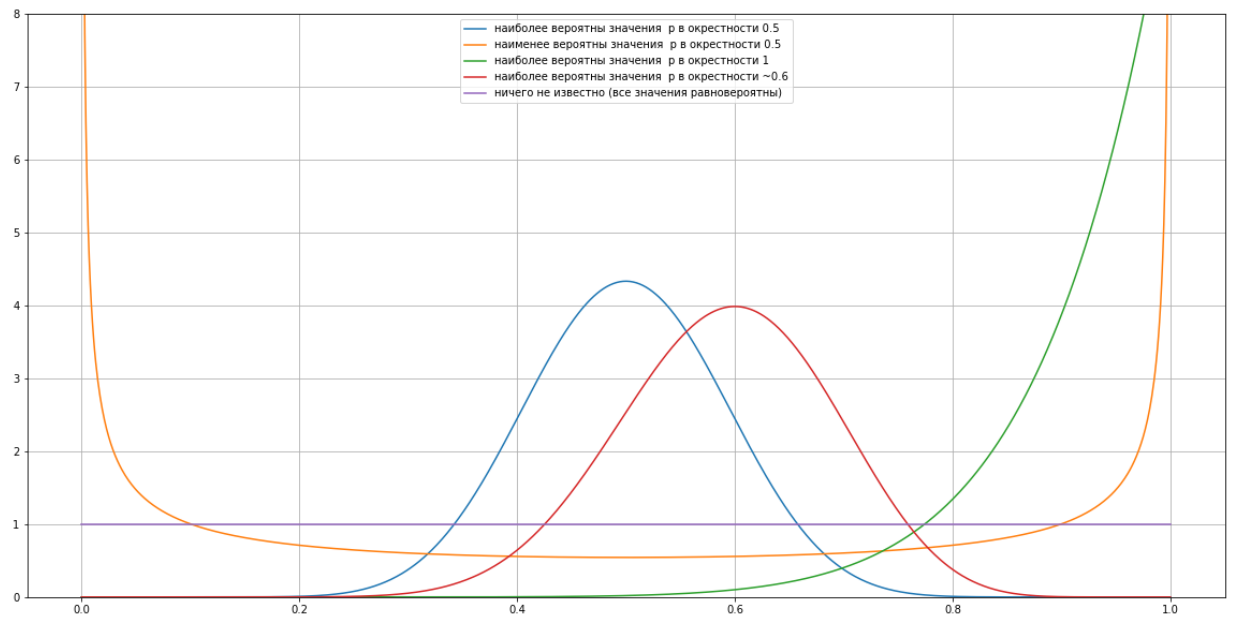
Знаем, что сопряжённым к бернуллиевскому распределению, является  $B$ -распределение.

Сравним их плотности: Для  $B$ :  $\frac{x^{a-1} * (1-x)^{b-1}}{B(a,b)}$ ,

для Бернулли:  $p^{\sum_{i=1}^n x_i} * (1-p)^{n-\sum_{i=1}^n x_i}$ .

Видно, что параметры  $a$  и  $b$  отвечают за количество успехов и неудач соответственно в испытаниях Бернулли. Поэтому построение графиков сводится к следующему

```
In [94]: plt.figure(figsize=(20, 10))
x = np.linspace(0, 1, 1000)
plt.plot(x, sps.beta(a=15, b=15).pdf(x), label='наиболее вероятны значения p в о
plt.plot(x, sps.beta(a=0.4, b=0.4).pdf(x), label='наименее вероятны значения p в
plt.plot(x, sps.beta(a=10, b=1).pdf(x), label='наиболее вероятны значения p в ок
plt.plot(x, sps.beta(a=14.5, b=10).pdf(x), label='наиболее вероятны значения p в
plt.plot(x, sps.beta(a=1, b=1).pdf(x), label='ничего не известно (все значения ра
plt.grid()
plt.legend(loc='best', fontsize=10)
plt.ylim(0, 8)
plt.show()
```



Ниже приведена реализация некоторых вспомогательных функций.

```

In [97]: def draw_posteriori(grid, distr_class, post_params, xlim=None):
    ''' Рисует серию графиков апостериорных плотностей.
        grid --- сетка для построения графика
        distr_class --- класс распределений из scipy.stats
        post_params --- параметры апостериорных распределений
                                shape=(размер выборки, кол-во параметров)
    ...

    size = post_params.shape[0] - 1

    plt.figure(figsize=(12, 7))
    for n in range(size+1):
        plt.plot(grid,
                 distr_class(post_params[n]).pdf(grid) if np.isscalar(post_params[n])
                 else distr_class(*post_params[n]).pdf(grid),
                 label='n={}: {}'.format(n, post_params[n]),
                 lw=2.5,
                 color=(1-n/size, n/size, 0))
    plt.grid(ls=':')
    plt.legend()
    plt.xlim(xlim)
    plt.show()

def draw_estimations(ml, distr_class, post_params, confint=True, ylim=None):
    ''' Рисует графики байесовской оценки (м.о. и дов. инт.) и ОМП.
        ml --- Оценка максимального правдоподобия для 1 <= n <= len(sample)
        distr_class --- класс распределений из scipy.stats
        post_params --- параметры апостериорных распределений
                                shape=(размер выборки, кол-во параметров)
    ...

    size = len(ml)
    distrs = []
    for n in range(size+1):
        distrs.append(distr_class(post_params[n]) if np.isscalar(post_params[n])
                     else distr_class(*post_params[n]))

    plt.figure(figsize=(12, 4))
    plt.plot(np.arange(size+1), [d.mean() for d in distrs], label='Bayes', lw=1.5)
    plt.fill_between(np.arange(size+1), [d.ppf(0.975) for d in distrs],
                    [d.ppf(0.025) for d in distrs], alpha=0.1)
    plt.plot(np.arange(size)+1, ml, label='ML', lw=1.5)
    plt.grid(ls=':')
    plt.ylim(ylim)
    plt.legend()
    plt.show()

```

Реализуйте следующую функцию



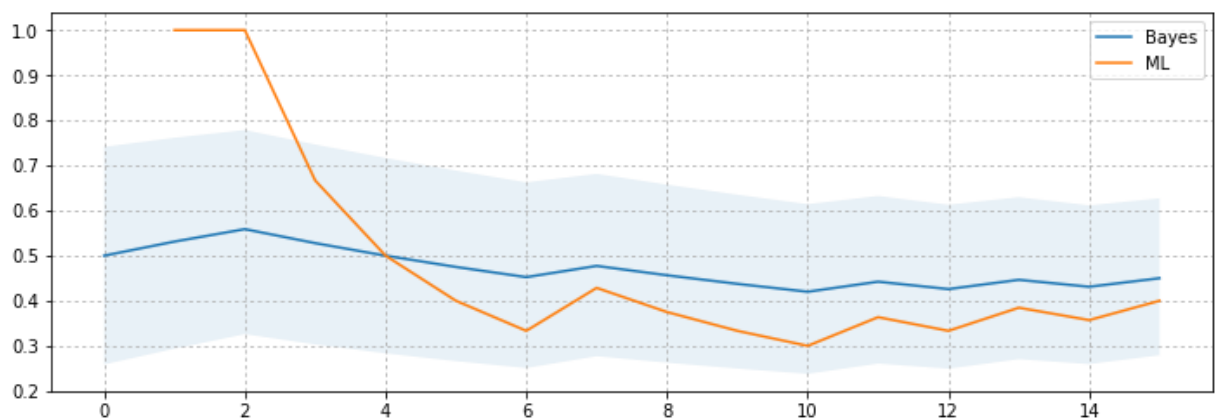
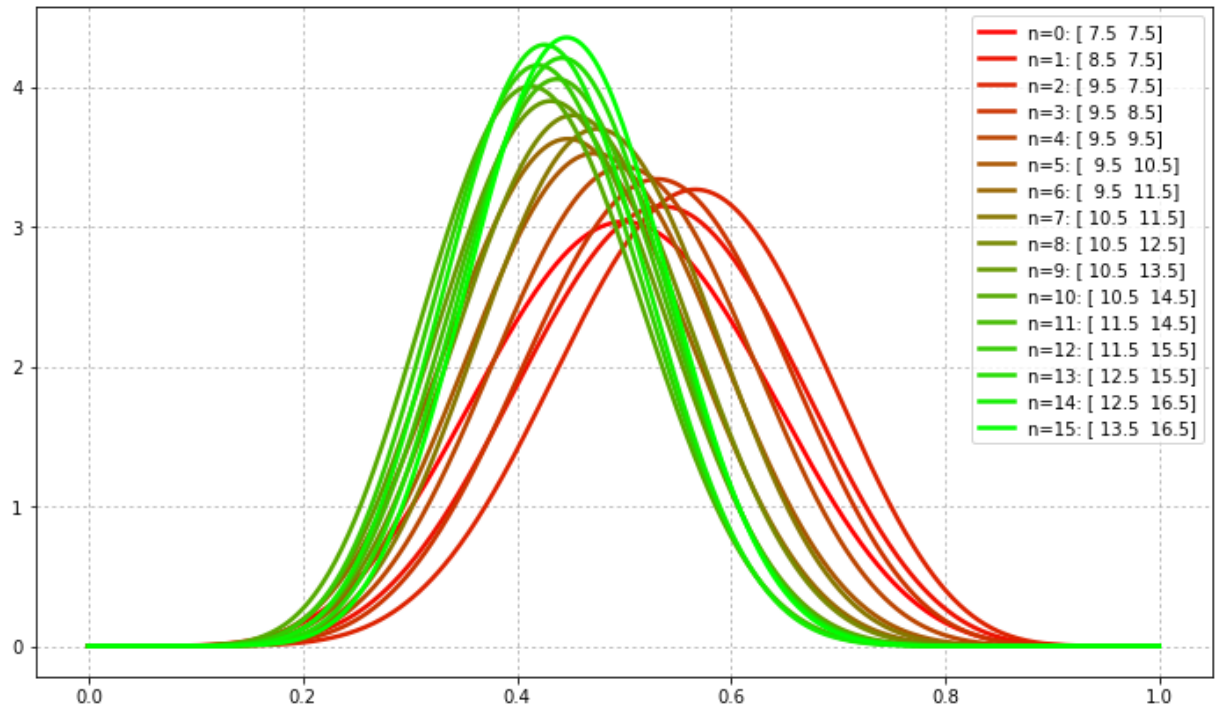
```
In [99]: def bern_posterior_params(sample, a, b):  
    ''' Возвращает параметры апостериорного распределения для всех  $0 \leq n \leq \text{len}(\text{sample})$   
        a, b --- параметры априорного распределения.  
    ...  
    params = np.array(list(map(lambda n: np.array([a + np.sum(sample[:n]), b + 1 - np.sum(sample[:n])]), range(1, len(sample) + 1)))  
    return params
```

Проведите по 15 бросков симметричной и несимметричной монет (можно сгенерировать) и рассмотрите для каждой из них два случая --- параметры априорного распределения подобраны правильно или неправильно. Постройте графики, воспользовавшись функциями `draw_posteriori` и `draw_estimations`.

Сделайте вывод. Что можно сказать про зависимость от параметров априорного распределения? Сравните байесовские оценки с оценкой максимального правдоподобия.

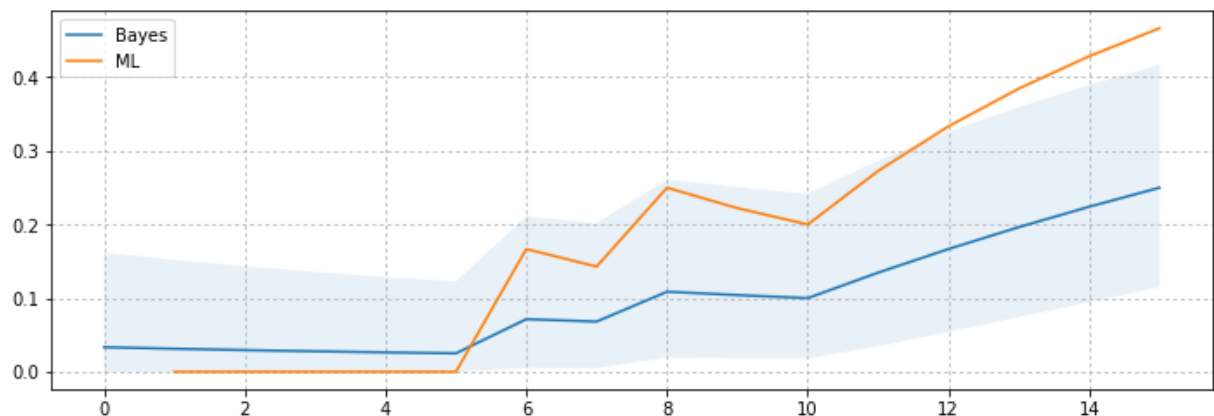
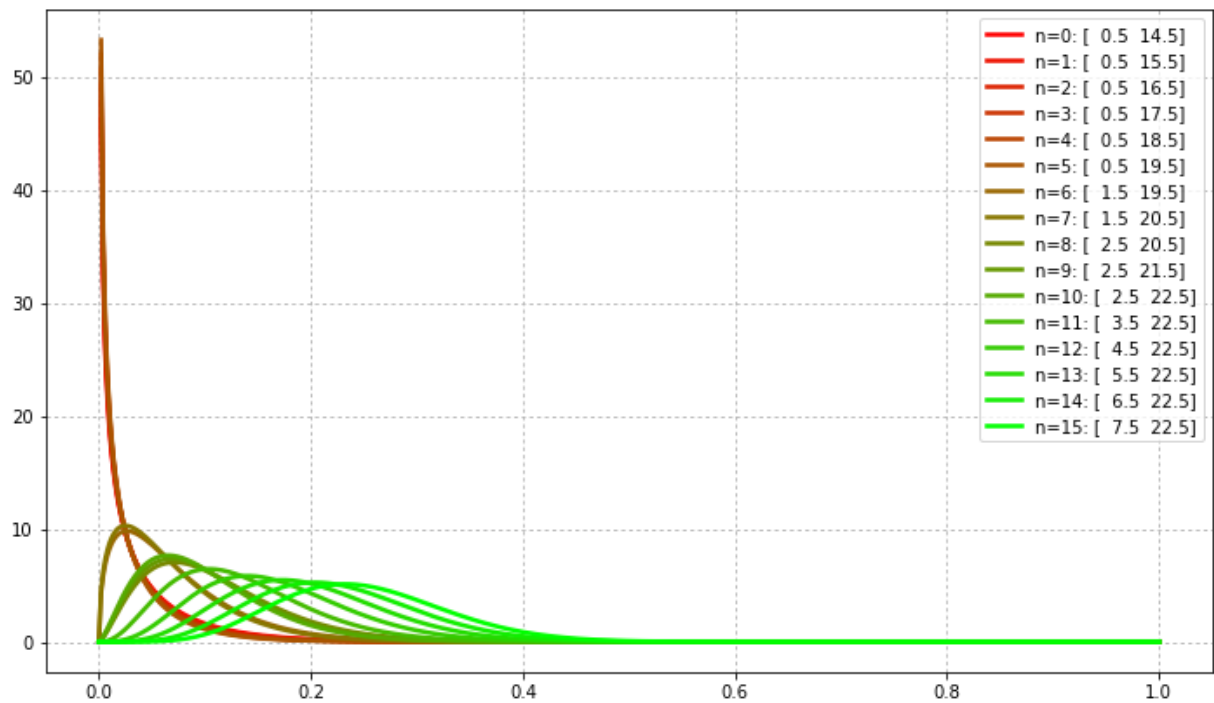
In [101]: *#симметричная монета и правильные параметры*

```
sample = sps.bernoulli(0.5).rvs(15)
grid = np.linspace(0, 1, 500)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 7.5, 7.5))
max_credibility = np.cumsum(sample) / np.arange(1, 16)
draw_estimations(ml=max_credibility, distr_class=sps.beta, post_params=bern_poste
```



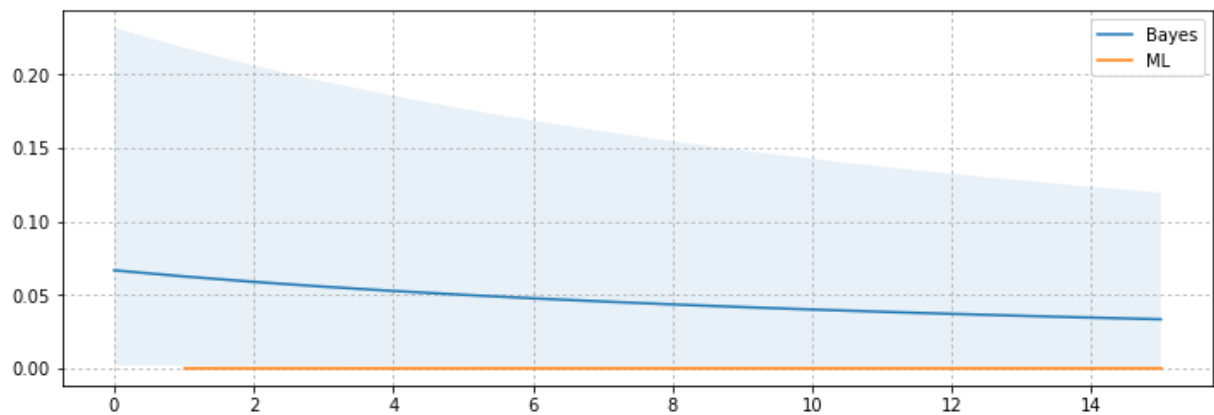
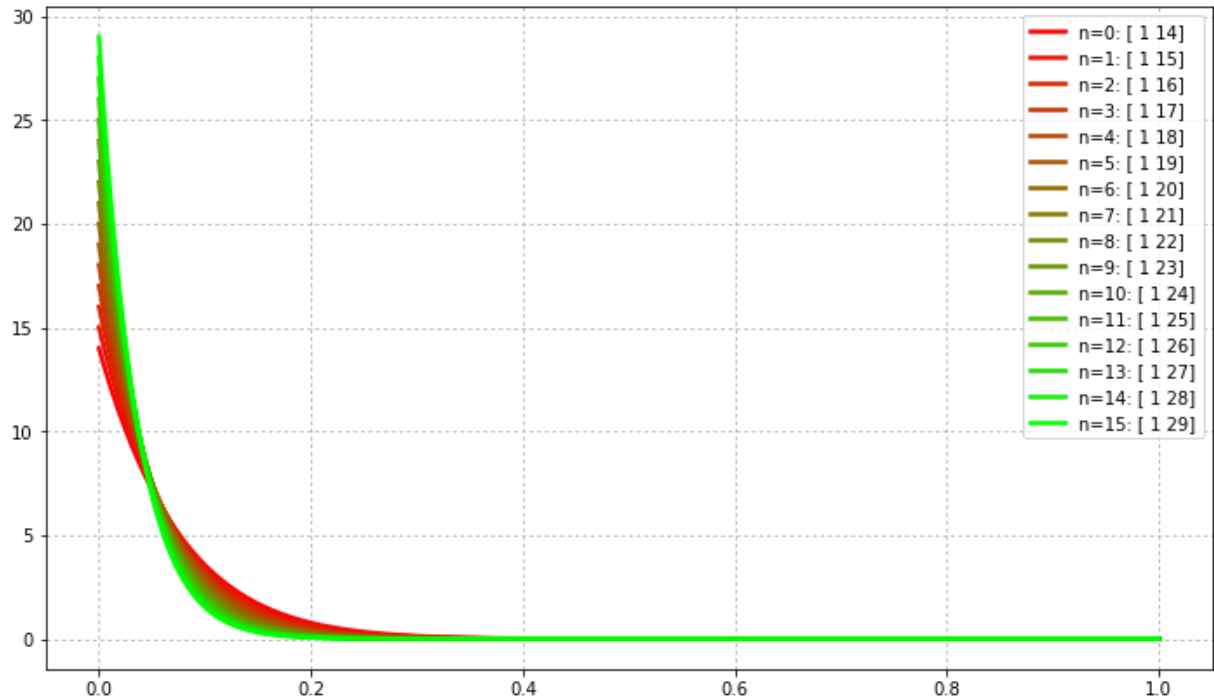
In [103]: *#симметричная монета, неправильные параметры*

```
sample = sps.bernoulli(0.5).rvs(15)
grid = np.linspace(0, 1, 500)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 0.5, 14.5))
max_credibility = np.cumsum(sample) / np.arange(1, 16)
draw_estimations(ml=max_credibility, distr_class=sps.beta, post_params=bern_poste
```



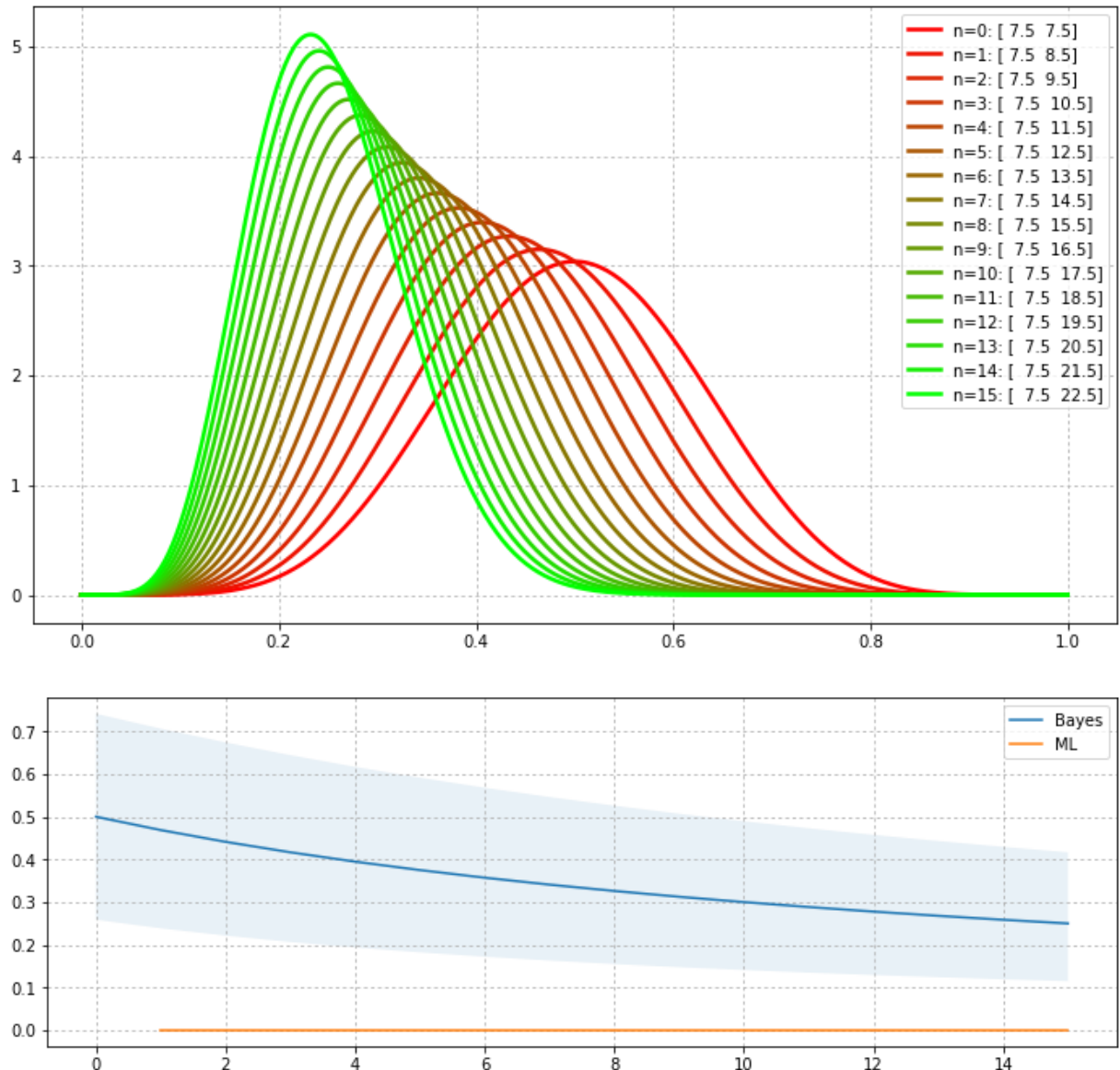
In [108]: # несимметричная монета, правильные параметры

```
sample = sps.bernoulli(1/15).rvs(15)
grid = np.linspace(0, 1, 500)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 1, 14))
max_credibility = np.cumsum(sample) / np.arange(1, 16)
draw_estimations(ml=max_credibility, distr_class=sps.beta, post_params=bern_poste
```



In [109]: *#несимметричная монета и неправильные параметры*

```
sample = sps.bernoulli(1/15).rvs(15)
grid = np.linspace(0, 1, 500)
draw_posteriori(grid, sps.beta, bern_posterior_params(sample, 7.5, 7.5))
max_credibility = np.cumsum(sample) / np.arange(1, 16)
draw_estimations(ml=max_credibility, distr_class=sps.beta, post_params=bern_poste
```



**Вывод** Из анализа графиков, можно увидеть, что при правильно подобранных параметрах, апостериорная плотность похожа на искомую. Байесовские оценки слабо совпадают с оценками максимального правдоподобия на маленьких выборках

**Задача 9\***. Один экзаменатор на экзамене по теории вероятностей при выставлении оценки студенту пользуется следующей схемой. В течении экзамена экзаменатор задает студенту некоторое количество вопросов, получая тем самым выборку  $X_1, \dots, X_n \sim \text{Bern}(p)$  --- индикаторы того, что студент на вопрос ответил правильно. При этом сначала он подбирает некоторое априорное распределение на основе его знаний о студенте к моменту начала ответа. После каждого ответа студента экзаменатор вычисляет апостериорное

распределение и строит байесовский доверительный интервал для  $p$  уровня доверия 0.95. Если после очередного ответа студента доверительный интервал содержит лишь одно число  $i/10$ , где  $i \in \{0, \dots, 10\}$ , то экзаменатор выставляет студенту оценку  $i + 1$ .

Ответьте на следующие вопросы:

- Квантили какого уровня нужно выбирать экзаменатору при построении доверительного интервала, чтобы задавать студенту как можно меньше вопросов? Какие оценки будет выставлять экзаменатор в таком случае?
- Как зависит оценка студента и среднее количество заданных вопросов у различных студентов (по уровню знаний) при различных априорных представлений экзаменатора?
- Нужно ли дружить с таким экзаменатором?

**Задача 10.** Проведите исследование, аналогичное задаче 8 для выборок из распределений

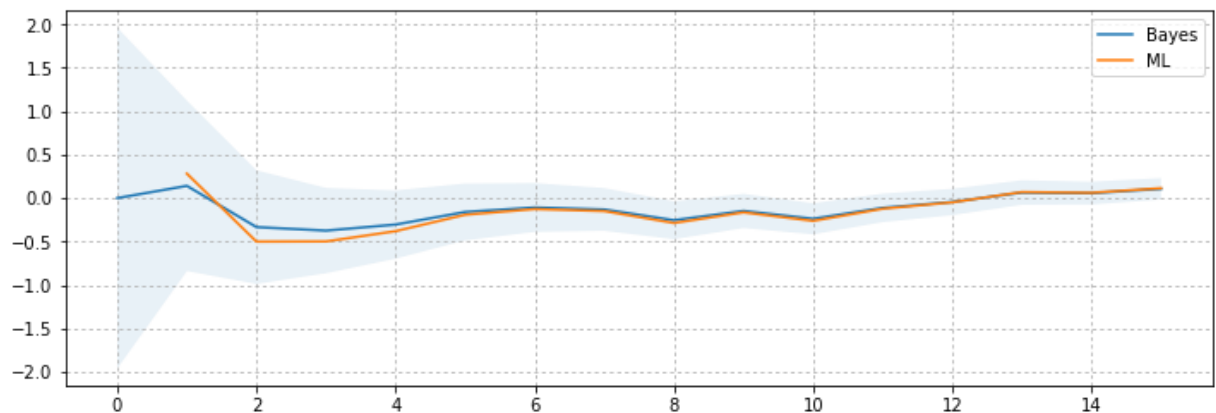
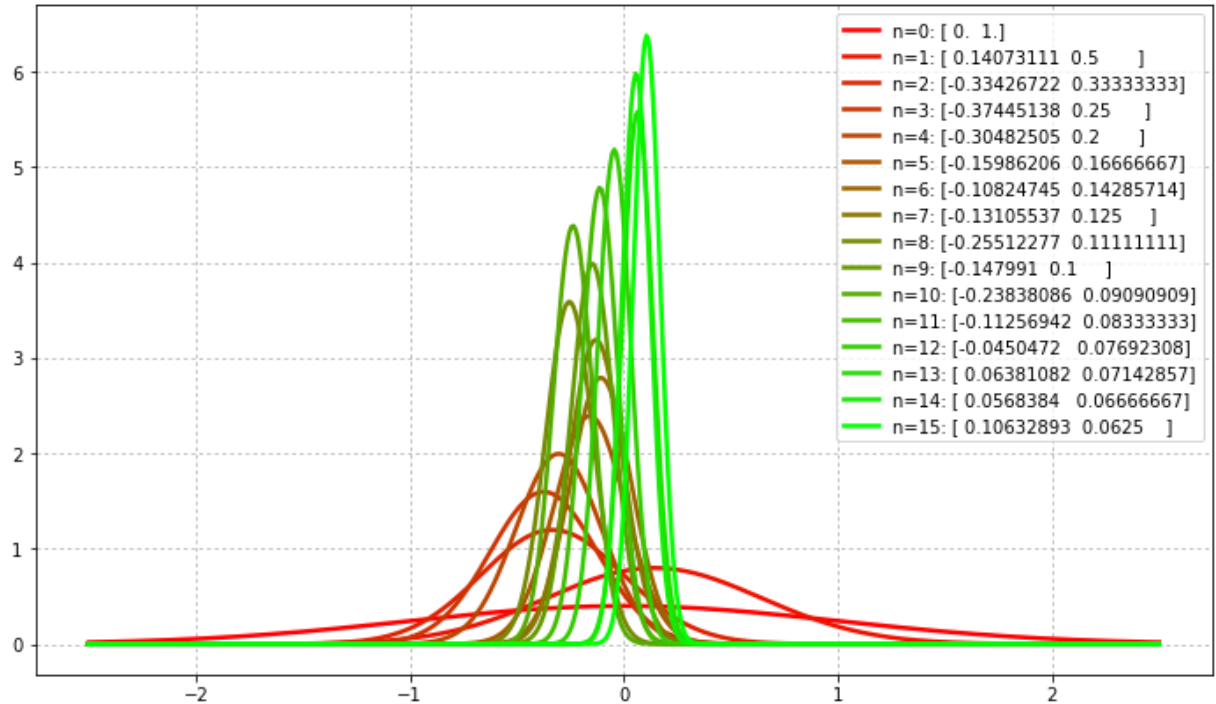
- $\mathcal{N}(\theta, 1)$
- $Exp(\theta)$

Нормальное распределение с дисперсией 1: сопряженным к нему является также нормальное распределение. Если априорное распределение  $\theta$  - это  $N(\mu, \sigma^2)$ , то апостериорным распределением  $\theta$  является  $N(\frac{\sum_{i=1}^n x_i + \mu}{n\sigma^2 + 1}, \frac{\sigma^2}{n\sigma^2 + 1})$

```
In [111]: def norm_posterior_params(sample, mu, sigma):
           params = np.array(list(map(lambda n: np.array([(mu + np.sum(sample[:n])) / (1
           return params
```

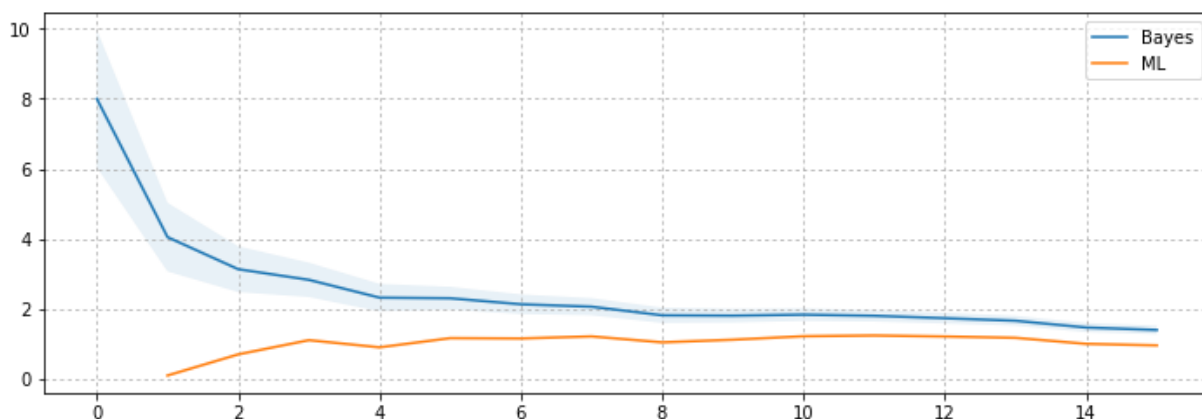
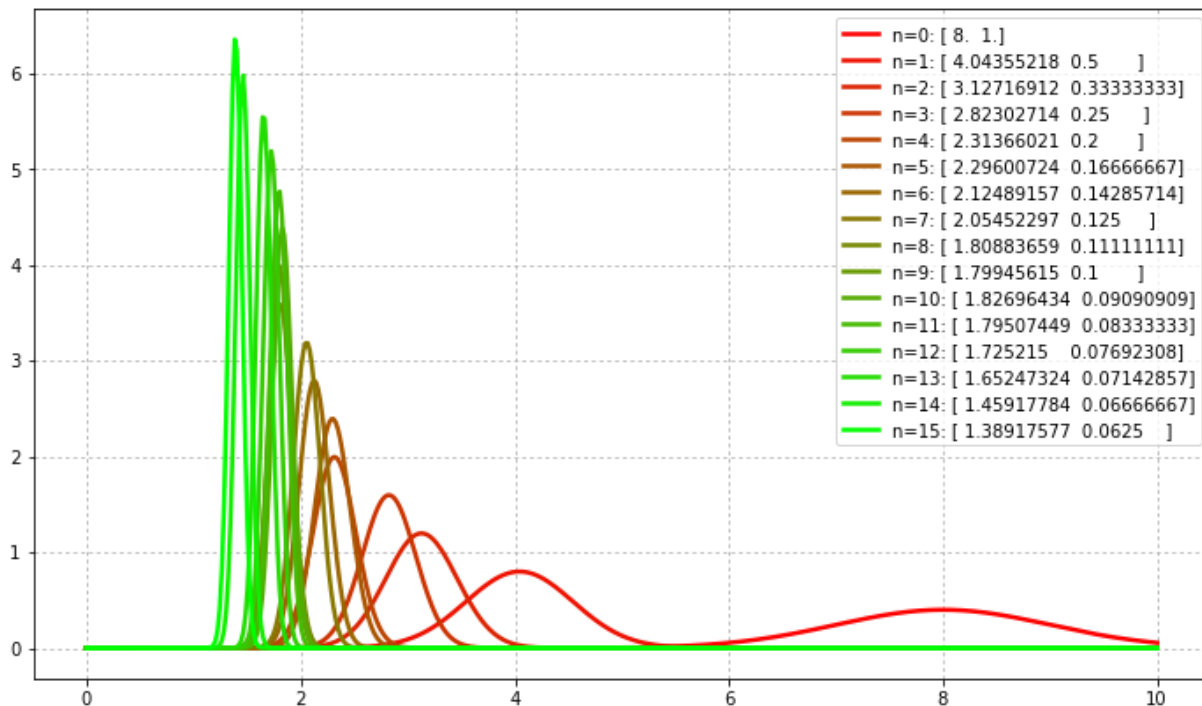
In [119]: *# правильные параметры*

```
sample = sps.norm(0.5).rvs(15)
grid = np.linspace(-2.5, 2.5, 500)
draw_posteriori(grid, sps.norm, norm_posterior_params(sample, 0, 1))
max_credibility = np.cumsum(sample) / np.arange(1, 16)
draw_estimations(ml=max_credibility, distr_class=sps.norm, post_params=norm_poste
```



In [123]: *# неправильные параметры*

```
sample = sps.norm(0.5).rvs(15)
grid = np.linspace(0, 10, 500)
draw_posteriori(grid, sps.norm, norm_posterior_params(sample, 8, 1))
max_credibility = np.cumsum(sample) / np.arange(1, 16)
draw_estimations(ml=max_credibility, distr_class=sps.norm, post_params=norm_poste
```



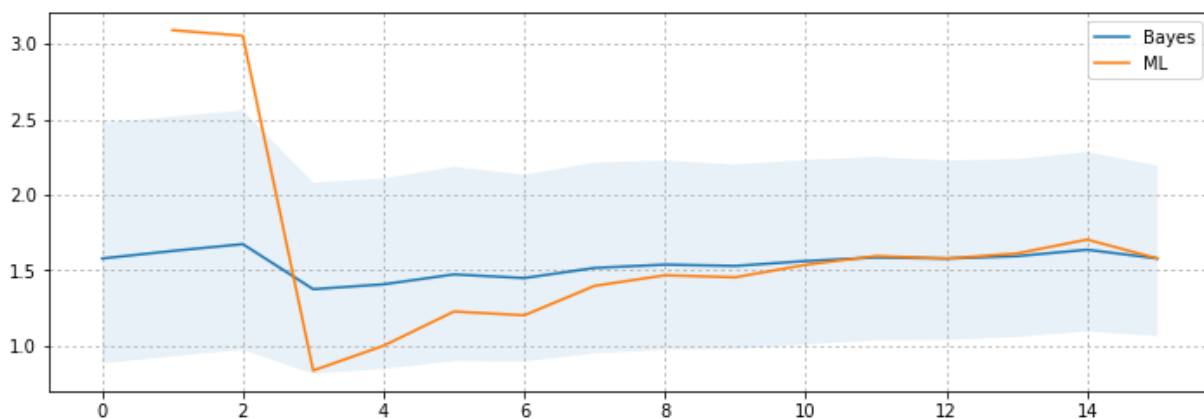
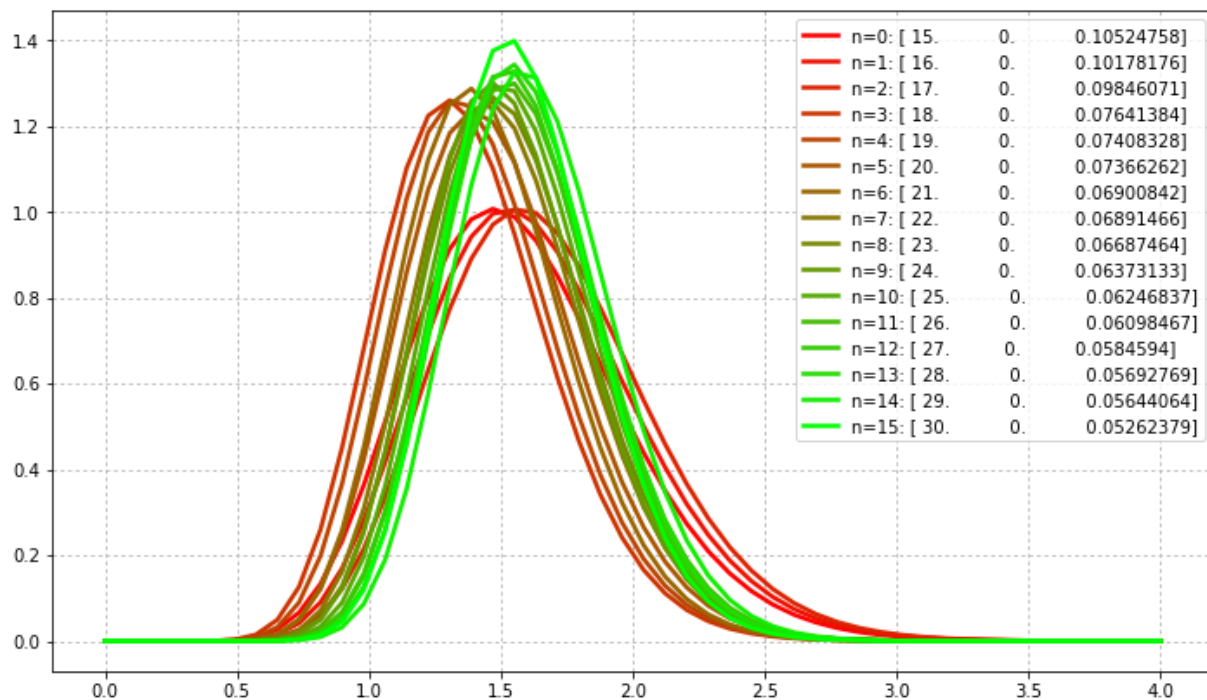
Экспоненциальное распределение с параметром  $\theta$ : сопряженным к нему является гамма-распределение. Если  $\Gamma(a, b)$  - априорное распределение  $\theta$ , то апостериорным распределением  $\theta$  является  $\Gamma(a + \sum_{i=1}^n x_i + a, n + b)$

```
In [124]: def exp_posterior_params(sample, a, b):
           params = np.array(list(map(lambda n: np.array([n + b, 0, 1/(a + np.sum(sample
                                                       range(len(sample)+1))]))
           return params
```



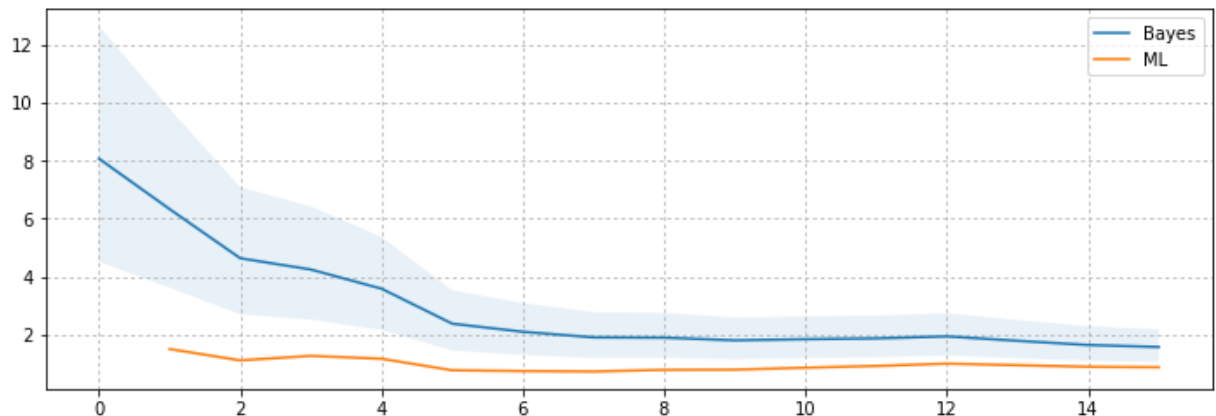
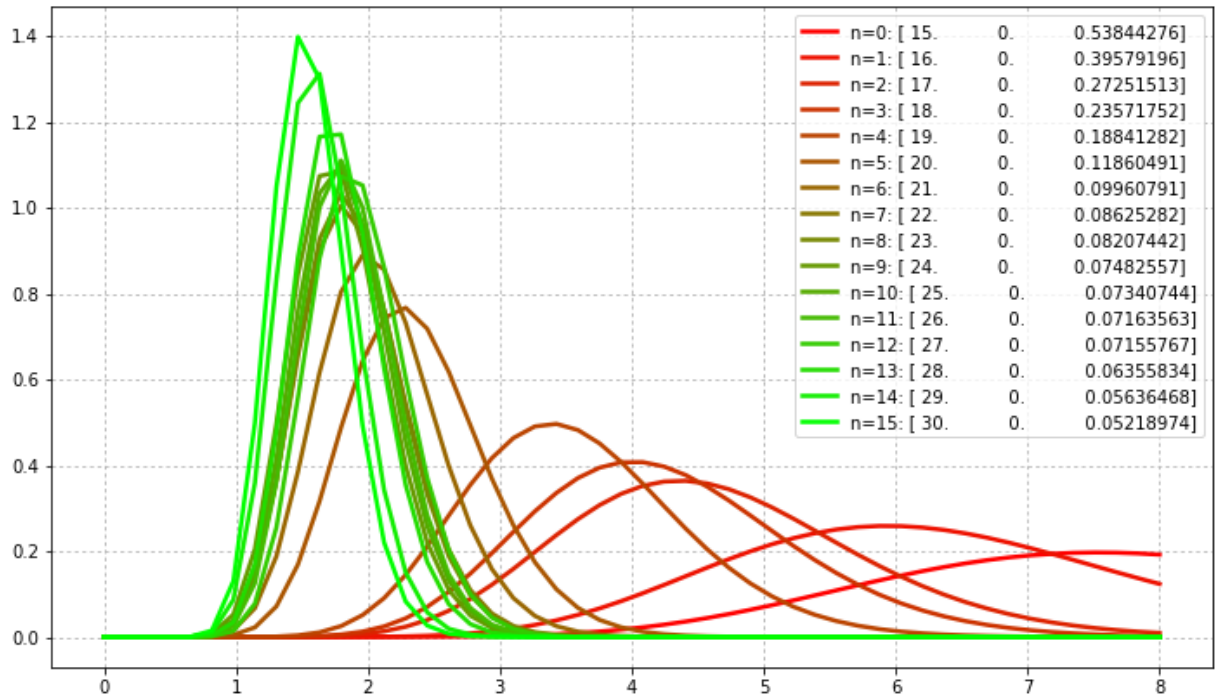
In [125]: *# правильные параметры*

```
sample = sps.expon.rvs(size=15)
grid = np.linspace(0, 4)
max_credibility = np.arange(1, 16) / np.cumsum(sample)
draw_posteriori(grid, sps.gamma, exp_posterior_params(sample, np.sum(sample), 15)
draw_estimations(ml=max_credibility, distr_class=sps.gamma, post_params=exp_poste
```



In [130]: # неправильные параметры

```
sample = sps.expon.rvs(size=15)
grid = np.linspace(0, 8)
sec = sps.expon(scale=1/8).rvs(size=15)
max_credibility = np.arange(1, 16) / np.cumsum(sample)
draw_posteriori(grid, sps.gamma, exp_posterior_params(sample, np.sum(sec), 15))
draw_estimations(ml=max_credibility, distr_class=sps.gamma, post_params=exp_poste
```



**Вывод** по сути, повторили задачу 8 для двух других распределений. Ещё раз убедились в силе и качестве Байесовских оценок

**Задача 11\***. Проведите исследование, аналогичное задаче 8 для выборки из распределения  $\mathcal{N}(\theta_1, \theta_2^{-1})$ .

