# Математическая статистика

## Практическое задание 1

В данном задании рассматриваются различные способы генерации выборки из некоторых стандартных распределений, а так же рассматриваются некоторые свойства эмпирической функции распределения и ядерной оценки плотности.

**Правила:**

- Баллы за каждую задачу указаны далее. Если сумма баллов за задание меньше 25%, то все задание оценивается в 0 баллов.
- Выполненную работу нужно отправить на почту `probability.diht@yandex.ru`, указав тему письма `"[номер группы] Фамилия Имя - Задание 1"`. Квадратные скобки обязательны.
- Прислать нужно ноутбук и его pdf-версию. Названия файлов должны быть такими: `1.N.ipynb` и `1.N.pdf`, где `N` - ваш номер из таблицы с оценками.
- Никакой код из данного задания при проверке запускаться не будет.

**Баллы за задание:**

- Задача 1 - 13 баллов
- Задача 2 - 3 балла
- Задача 3 - 5 баллов
- Задача 4 - 3 балла
- Задача 5 - 2 балла
- Задача 6 - 1 балл
- Задача 7 - 3 балла

In [2]:

```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt

%matplotlib inline
```

**Задача 1.** Имеется симметричная монета. С ее помощью напишите функцию генерации выборки из многомерного нормального распределения с заданными параметрами.

*Часть 1.* Напишите сначала функцию генерации равномерного распределения на отрезке $[0, 1]$ с заданной точностью. Это можно сделать, записав случайную величину $\xi \sim U[0, 1]$ в двоичной системе системе счисления $\xi = 0, \xi_1\xi_2\xi_3 \ldots$. Тогда $\xi_i \sim Bern(1/2)$ и независимы в совокупности. Приближение заключается в том, что вместо генерации бесконечного количества $\xi_i$ мы полагаем $\xi = 0, \xi_1\xi_2\xi_3 \ldots \xi_n$.

Для получения максимального балла реализовать функцию нужно так, чтобы она могла принимать на вход в качестве параметра `size` объект `tuple` любой размерности, и возвращать объект `numpy.array` соответствующей размерности. Например, если `size=(10, 1, 5)`, то функция должна вернуть объект размера $10 \times 1 \times 5$. Кроме того, функцию `coin` можно вызвать только один раз, и, конечно же, не использовать какие-либо циклы.

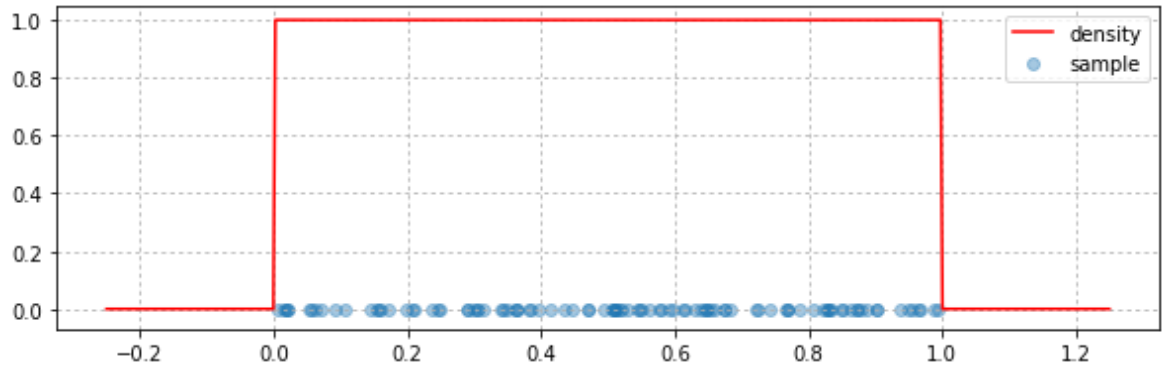
In [152]:

```
coin = sps.bernoulli(0.5).rvs  # симметричная монета
# coin(size=10) --- реализация 10 бросков монеты

def uniform(size=1, precision=30):
    return (coin(precision * np.prod(np.asarray(size))).reshape(np.prod(np.asarray(
            * np.power(2., np.arange(-1, -precision - 1, -1))).sum(axis = 1).reshap
```

Для $U[0, 1]$ сгенерируйте выборку и постройте график плотности.

In [150]:

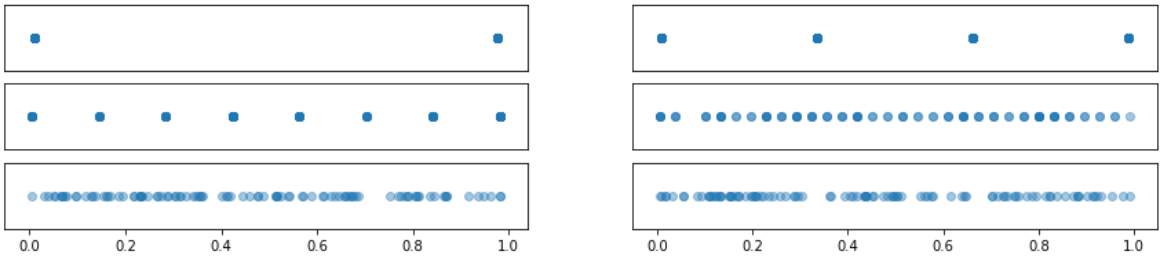
```
size = 100
grid = np.linspace(-0.25, 1.25, 500)

plt.figure(figsize=(10, 3))
plt.scatter(uniform(size, 50),
            np.zeros(size), alpha=0.4, label='sample')
plt.plot(grid,
         sps.uniform.pdf(grid),
         color='red', label='density')
plt.legend()
plt.grid(ls=':')
plt.show()
```




Исследуйте, как меняется выборка в зависимости от precision.

In [151]:

```
size = 100

plt.figure(figsize=(15, 3))
for i, precision in enumerate([1, 2, 3, 5, 10, 30]):
    plt.subplot(3, 2, i + 1)
    plt.scatter(uniform(size, precision),
                np.zeros(size), alpha=0.4)
    plt.yticks([])
    if i < 4: plt.xticks([])
plt.show()
```




**Вывод:**

Мы действительно получили выборку из равномерного распределения. Можно заметить, что равномерность распределения лучше всего видна при precision = {1, 2, 3, 5}

*Часть 2.* Напишите функцию генерации выборки размера `size` (как и раньше, тут может быть `tuple`) из распределения $\mathcal{N}(loc, scale^2)$ с помощью преобразования Бокса-Мюллера (задача 7.12 из книги по теории вероятностей).

Для получения полного балла реализация должна быть без циклов.

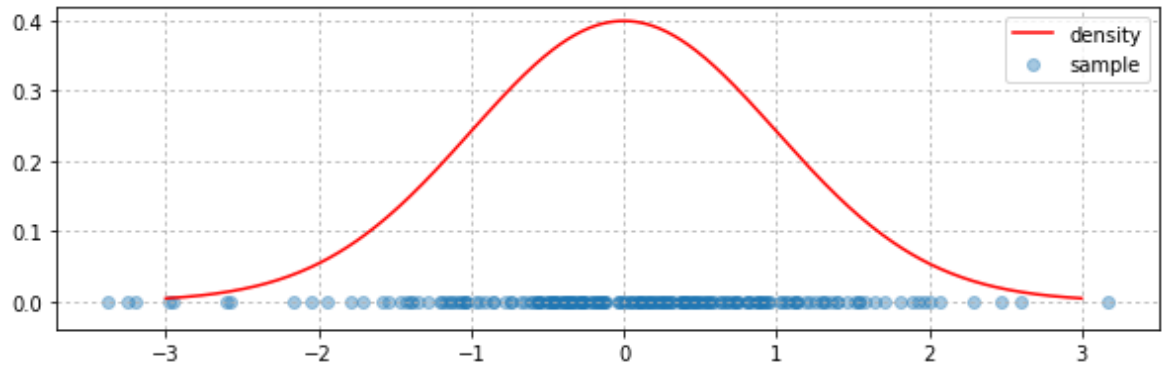
In [184]:

```
def normal(size=1, loc=0, scale=1, precision=30):
    new_size = np.prod(np.asarray(size))
    A = uniform(new_size, precision)
    B = uniform(new_size, precision)
    X = np.cos(2 * np.pi * A) * scale
    Y = np.sqrt(-2 * np.log(B))
    return (X * Y + loc).reshape(size)
```

Для $\mathcal{N}(0, 1)$ сгенерируйте выборку и постройте график плотности этого распределения на отрезке $[-3, 3]$.

In [185]:

```
size = 200
grid = np.linspace(-3, 3, 500)

plt.figure(figsize=(10, 3))
plt.scatter(normal(size, 0, 1, 50),
            np.zeros(size), alpha=0.4, label='sample')
plt.plot(grid,
         sps.norm.pdf(grid),
         color='red', label='density')
plt.legend()
plt.grid(ls=':')
plt.show()
```




Пусть $P$ --- некоторое распределение на $(\mathbb{R}, \mathscr{B}(\mathbb{R}))$. Числа $a$ и $\sigma$ называются параметрами сдвига и масштаба соответственно для семейства распределений $\left\{P_{a,\sigma} \mid a \in \mathbb{R}, \sigma \in \mathbb{R}_+\right\}$, где $P_{a,\sigma}(B) = P\left(\frac{B-a}{\sigma}\right)$ и $\frac{B-a}{\sigma} = \left\{\frac{x-a}{\sigma} \;\middle|\; x \in B\right\}$

**Вопрос:** Найдите плотность $P_{a,\sigma}$, если $P$ имеет плотность $p(x)$.

<...>

**Вопрос:** Пусть $P$ --- стандартное нормальное распределение. Выпишите параметрическое семейство распределений, параметризованное параметрами сдвига и масштаба по отношению к распределению $P$. Какая связь между параметрами и характеристиками распределения (например, математическое ожидание)?

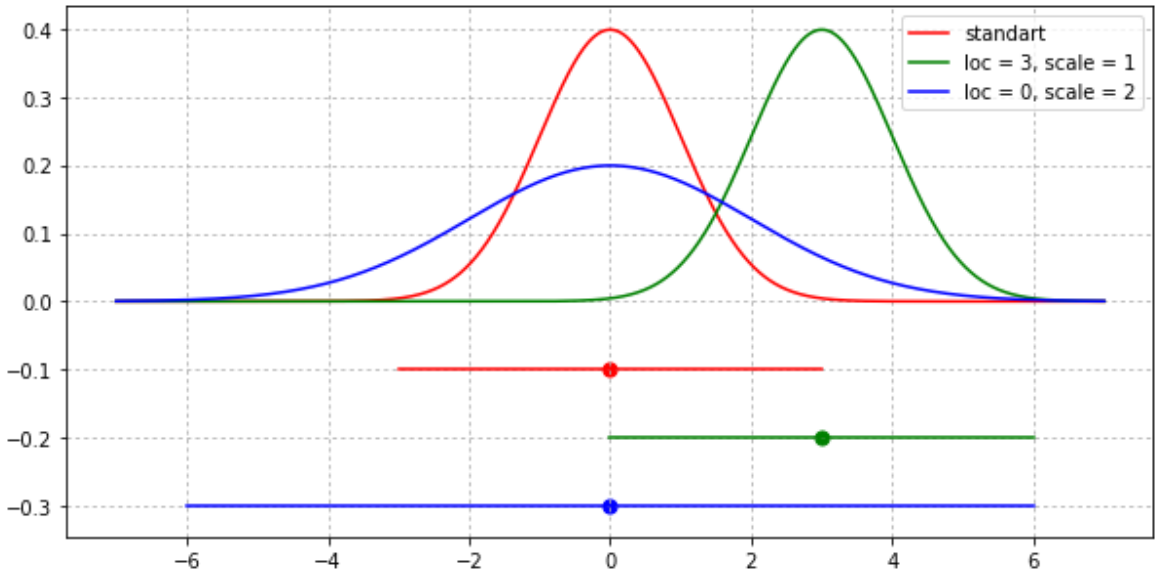
Семейство распределений -- нормальные распределения с параметрами loc и scale соответственно. Следовательно, математическое ожидание данного семейства распределения -- параметр сдвига, дисперсия -- параметр масштаба.

Постройте на одном графике разными цветами плотности стандартного нормального распределения, а так же для параметров $a = 3, \sigma = 1$ и $a = 0, \sigma = 2$. Интеравл по оси икс $[-7, 7]$.

Ниже графика теми же цветами изобразите также точку $a$ и $3\sigma$-интервал, используя шаблон, приведенный ниже.

In [183]:

```
plt.figure(figsize=(10, 5))
grid = np.linspace(-7, 7, 10000)
plt.plot(grid, sps.norm.pdf(grid), color='red', label='standart')
plt.plot(grid, sps.norm.pdf(grid, loc=3, scale=1), color = 'green', label='loc = 3,
plt.plot(grid, sps.norm.pdf(grid, loc=0, scale=2), color = 'blue', label='loc = 0,

plt.plot([-3, 3], [-0.1, -0.1], color='red')
plt.plot([0, 6], [-0.2, -0.2], color='green')
plt.plot([-6, 6], [-0.3, -0.3], color='blue')

plt.scatter(0, -0.1, color='red', s=50)
plt.scatter(3, -0.2, color='green', s=50)
plt.scatter(0, -0.3, color='blue', s=50)

plt.legend()
plt.grid(ls=':')
plt.show()
```




**Вывод:**

Преобразование Бокса-Мюллера позволяет получить стандартное нормальное распределение. Параметризация с помощью параметров сдвига и масштаба позволяет получить класс нормальных распределений с параметрами loc и scale соответственно.

*Часть 3.* Теперь напишите функцию генерации выборки из многомерного нормального распределения с заданным вектором средних `mean` и матрицей ковариаций `cov_matrix`. Помочь в этом может теорема об эквивалентных определениях гауссовского вектора. Для извлечения квадратного корня из матрицы может пригодится следующая функция, которая вычисляет собственные значения и векторы матрицы.

In [24]:

```
from scipy.linalg import eigh
```

На этот раз достаточно, чтобы функция корректно работала в случае, когда `size` является числом.

In [25]:

```
def gauss(mean, cov_matrix, size=1, precision=30):
    # Преобразование типов
    mean = np.array(mean)
    cov_matrix = np.array(cov_matrix)

    # Проверка на корректность входа
    assert mean.ndim == 1 and cov_matrix.ndim == 2
    assert mean.shape[0] == cov_matrix.shape[0]
    assert cov_matrix.shape[0] == cov_matrix.shape[1]

    <...>
```

...

Сгенерируйте выборку размера `size` из двумерного нормального распределения с нулевым вектором средних и матрицей ковариаций $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$. Нанесите точки выборки на график и отметьте цветом значение плотности.

В инструкциях по Питону плотность вычислялась с помощью неэффективного кода. Подумайте, как можно написать эффективный короткий код, не использующий циклы.

In [26]:

```
size = 1000
sample = <...>  # Генерация выборки

grid = np.mgrid[-4:4:0.05, -4:4:0.05]
density = sps.multivariate_normal.pdf(<...>)  # Вычисление плотности

plt.figure(figsize=(10, 10))
plt.pcolormesh(grid[0], grid[1], density, cmap='Oranges')
plt.scatter(sample[:, 0], sample[:, 1], alpha=0.4, label='sample')
plt.legend()
plt.grid(ls=':')
plt.xlim((-4, 4))
plt.ylim((-4, 4))
plt.show()
```

...

**Вывод:**

<...>

**Задача 2.** Вы уже научились генерировать выборку из равномерного распределения. Напишите функцию генерации выборки из экспоненциального распределения, используя результат задачи 6.9 из книги по теории вероятностей.

Для получения полного балла реализация должна быть без циклов, а параметр `size` может быть типа `tuple`.

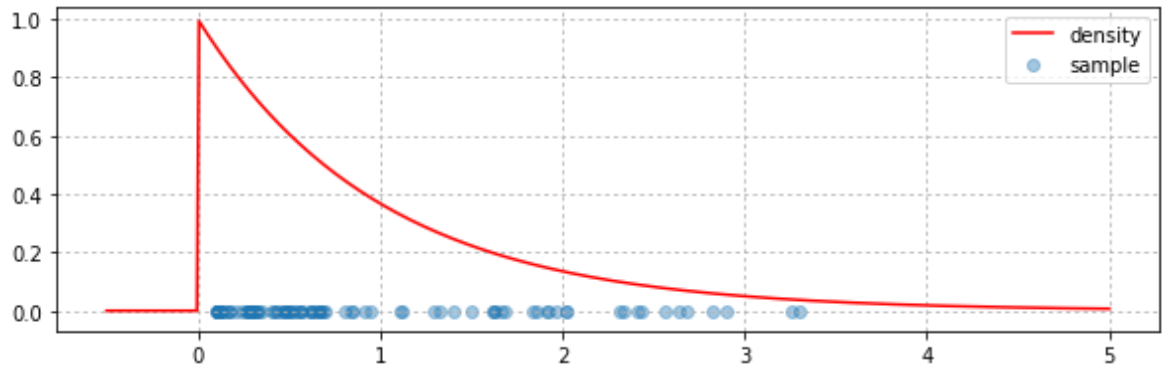
In [72]:

```
def expon(size=1, lambd=1, precision=30):
    return np.log(1 - uniform(size, precision)) / (-lambd)
```

Для $Exp(1)$ сгенерируйте выборку размера 100 и постройте график плотности этого распределения на отрезке $[-0.5, 5]$.

In [73]:

```
size = 100
grid = np.linspace(-0.5, 5, 500)

plt.figure(figsize=(10, 3))
plt.scatter(expon(size, 1, 50),
            np.zeros(size), alpha=0.4, label='sample')
plt.plot(grid,
         sps.expon.pdf(grid),
         color='red', label='density')
plt.legend()
plt.grid(ls=':')
plt.show()
```




**Вывод:**

Метод обратных функций позволяет получить экспоненциальное распределение из равномерного

**Задача 3.** Для каждого распределения постройте эмпирическую функцию распределения (ЭФР), гистограмму и ядерную оценку плотности. Сделать это помогут следующие функции.

In [3]:

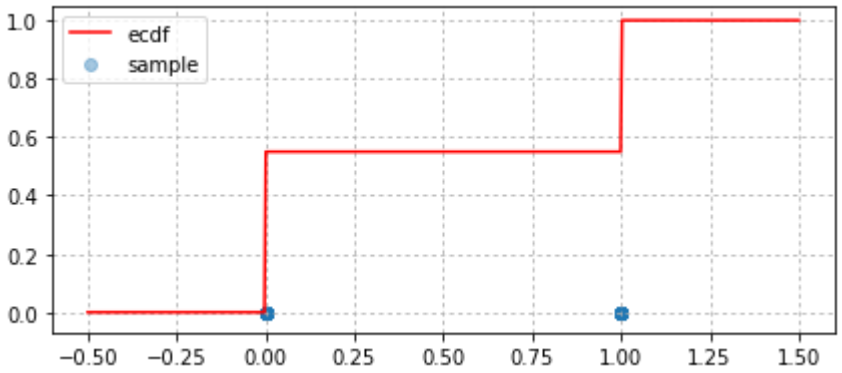
```
from statsmodels.distributions.empirical_distribution import ECDF
from statsmodels.nonparametric.kde import KDEUnivariate
```

*1. Бернуллиевское.*

Тут приведен пример построения ЭФР, просто запустите эту ячейку.

In [75]:

```
sample = coin(size=100)
ecdf = ECDF(sample)
grid = np.linspace(-0.5, 1.5, 500)

plt.figure(figsize=(7, 3))
plt.scatter(sample, np.zeros(size), alpha=0.4, label='sample')
plt.plot(grid, ecdf(grid), color='red', label='ecdf')
plt.legend()
plt.grid(ls=':')
plt.show()
```




Далее, чтобы не копировать несколько раз один и тот же код, напишем некоторую функцию.

В третьей функции нужно построить ядерную оценку плотности, о которой будет рассказано на лекциях. В частности, формула была на презентации на первой лекции. Пример построения можно посмотреть тут http://statsmodels.sourceforge.net/0.6.0/generated/statsmodels.nonparametric.kde.KDEUnivariate.html (http://statsmodels.sourceforge.net/0.6.0/generated/statsmodels.nonparametric.kde.KDEUnivariate.html)

In [76]:

```
def draw_ecdf(sample, grid, cdf=None):
    ''' По сетке grid строит графики эмпирической функции распределения
    и истинной (если она задана) для всей выборки и для 1/10 ее части.
    '''

    plt.figure(figsize=(16, 3))
    for i, size in enumerate([len(sample) // 10, len(sample)]):
        plt.subplot(1, 2, i + 1)

        plt.scatter(sample[:size], np.zeros(size),
                    alpha=0.4, label='sample')

        if cdf is not None:
            plt.plot(grid,
                     cdf(grid),
                     color='green', alpha=0.3, lw=2, label='true cdf')
        plt.plot(grid,
                 ECDF(sample[:size])(grid),
                 color='red', label='ecdf')

        plt.legend()
        plt.grid(ls=':')
        plt.title('sample size = {}'.format(size))
    plt.show()


def draw_hist(sample, grid, pdf=None):
    ''' Строит гистограмму и по сетке grid график истинной плотности
    (если она задана) для всей выборки и для 1/10 ее части.
    '''

    plt.figure(figsize=(16, 3))
    for i, size in enumerate([len(sample) // 10, len(sample)]):
        plt.subplot(1, 2, i + 1)
        plt.hist(sample[:size],
                 bins=20,
                 range=(grid.min(), grid.max()),
                 normed=True)

        if pdf is not None:
            plt.plot(grid,
                     pdf(grid),
                     color='green', alpha=0.3, lw=2)
    plt.show()


def draw_pdf(sample, grid, pdf=None):
    ''' По сетке grid строит графики ядерной оценки плотности
    и истинной плотности (если она задана) для всей выборки и для 1/10 ее части.
    '''

    plt.figure(figsize=(16, 3))
    for i, size in enumerate([len(sample) // 10, len(sample)]):
        plt.subplot(1, 2, i + 1)
        kernel_density = KDEUnivariate(sample[:size])
        kernel_density.fit()

        plt.scatter(sample[:size], np.zeros(size),
                    alpha=0.4, label='sample')
```

```
        if pdf is not None:
            plt.plot(grid,
                     pdf(grid),
                     color='green', alpha=0.3, lw=2, label='true pdf')

        plt.plot(grid,
                 kernel_density.evaluate(grid),
                 color='red', label='kde')

        plt.legend()
        plt.grid(ls=':')
    plt.show()
```

При использовании KDEUnivariate могут возникать разные проблемы. Можно попробовать их решить следующими способами:

**1.** В режиме суперюзера в файле `/usr/local/lib/python3.5/dist-packages/statsmodels/nonparametric/kdetools.py` замените строку 20 на

```
y = X[:int(m/2+1)] + np.r_[0,X[int(m/2+1):],0]*1j
```

В файле `/usr/local/lib/python3.5/dist-packages/statsmodels/nonparametric/kde.py` замените строку 327 на

```
nobs = len(X) # after trim
```

**2.** Попробуйте скачать с гитхаба https://github.com/statsmodels/statsmodels/ (https://github.com/statsmodels/statsmodels/), установить руками. При этом должен быть установлен cython.

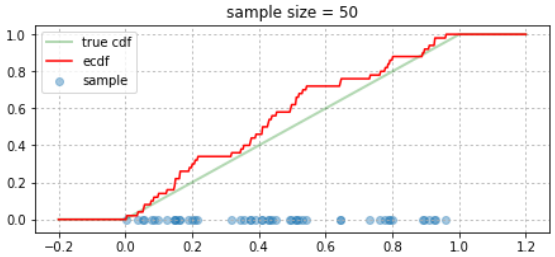
Можно также воспользоваться другой реализацией http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KernelDensity.html#sklearn.neighbors.KernelDensity (http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KernelDensity.html#sklearn.neighbors.KernelDensity)

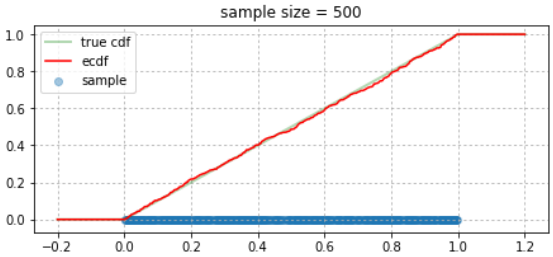
---

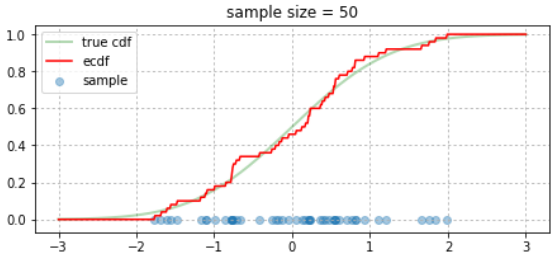
Теперь примените реализованные выше функции к выборкам размера 500 из распределений $U[0, 1]$, $\mathcal{N}(0, 1)$ и $Exp(1)$. Графики (ф.р., плотностей) стройте на интервалах $(-0.2, 1.2)$, $(-3, 3)$ и $(-0.5, 5)$ соответственно.

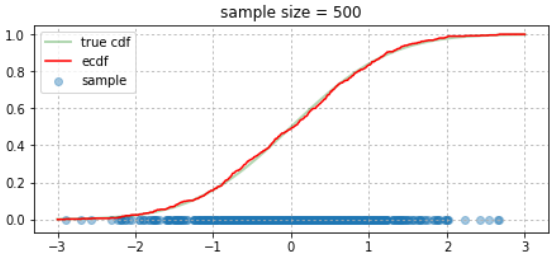
In [77]:

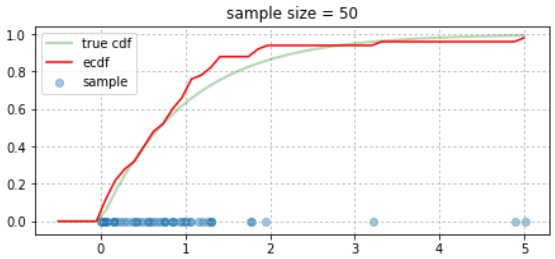
```
size = 500
sample_uniform = sps.uniform.rvs(size=500)
sample_norm = sps.norm.rvs(loc=0, scale=1, size=500)
sample_expon = sps.expon.rvs(size=500)

grid_uniform = np.linspace(-0.2, 1.2, 500)
grid_norm = np.linspace(-3, 3, 500)
grid_expon = np.linspace(-0.5, 5)
```

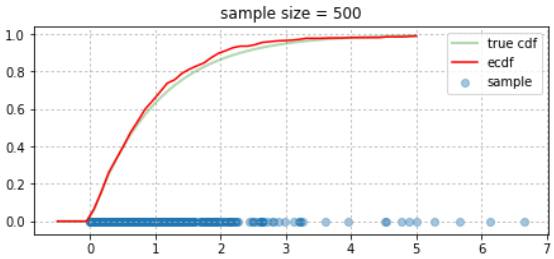










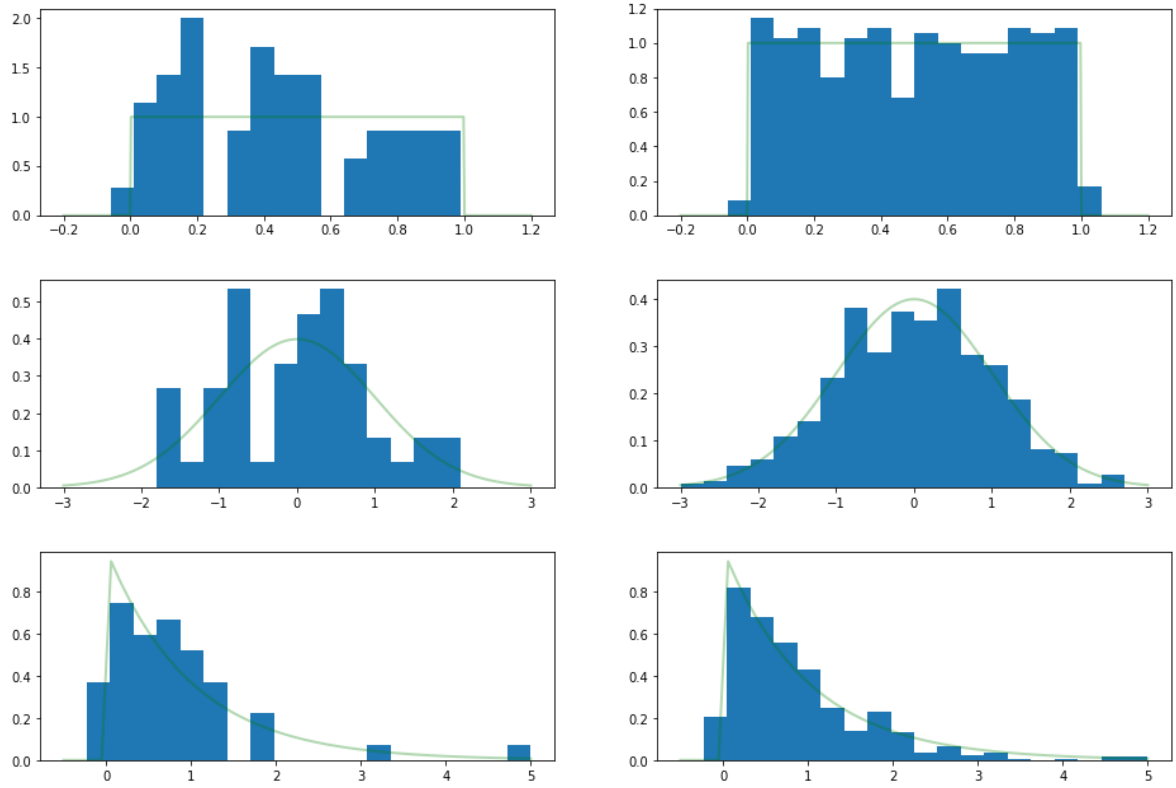


In [78]:

```
draw_ecdf(sample_uniform, grid_uniform, sps.uniform.cdf)
draw_ecdf(sample_norm, grid_norm, sps.norm.cdf)
draw_ecdf(sample_expon, grid_expon, sps.expon.cdf)
```

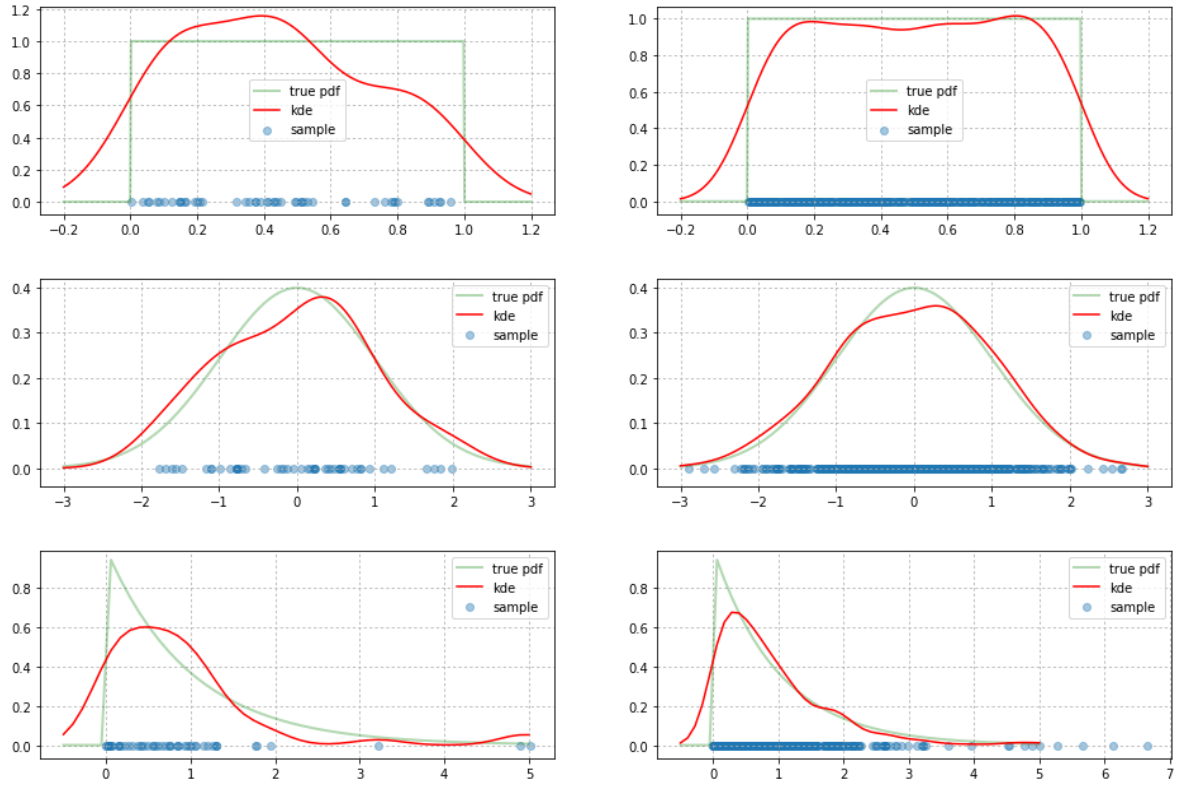
In [79]:

```
draw_hist(sample_uniform, grid_uniform, sps.uniform.pdf)
draw_hist(sample_norm, grid_norm, sps.norm.pdf)
draw_hist(sample_expon, grid_expon, sps.expon.pdf)
```



In [80]:

```
draw_pdf(sample_uniform, grid_uniform, sps.uniform.pdf)
draw_pdf(sample_norm, grid_norm, sps.norm.pdf)
draw_pdf(sample_expon, grid_expon, sps.expon.pdf)
```



**Вывод:**

Чем больше выборка, тем точнее получается оценка

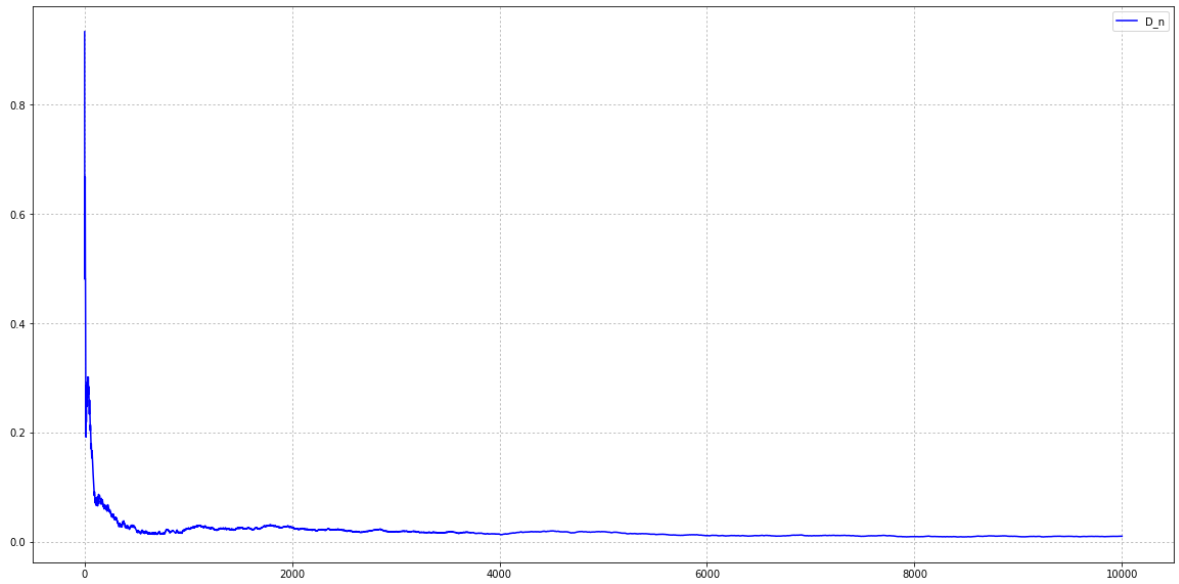
**Задача 4.** Сгенерируйте выборку $X_1, \ldots, X_{10000}$ из стандартного нормального распределения. Для каждого $n \leqslant 10000$ постройте эмпирическую функцию распределения $F_n^*$ и посчитайте **точное** значение статистики

$$D_n = \sup_{x \in \mathbb{R}} |F_n^*(x) - F(x)|.$$

Постройте график зависимости статистики $D_n$ от $n$. Верно ли, что $D_n \to 0$ и в каком смысле? Не забудьте сделать вывод.

In [181]:

```
grid = np.linspace(1, 10000, 10000)
sample = sps.norm.rvs(size = 10000)
data = [sps.kstest(sample[:x], 'norm').statistic for x in range(1, 10001)]

plt.figure(figsize=(20, 10))
plt.plot(grid, data, color='blue', label='D_n')
plt.legend()
plt.grid(ls=':')
plt.show()
```




**Вывод:** При n $\to \infty$ эмпирическая функция распределения сходится к настоящей почти наверное по критерию сходимости почти наверное

**Задача 5.** Исследуйте вид ядерной оценки плотности в зависимости от вида ядра и его ширины.

Для этого сгенерируйте выборку $X_1, \ldots, X_{200}$ из распределения $U[0, 1]$ и постройте серию графиков для различной ширины гауссовского ядра, а затем другую серию графиков для различных типов ядер при фиксированной ширине. На каждом графике на отрезке $[-0.2, 1.2]$ должны быть изображены истинная плотность (полупрозрачным цветом) и ее ядерная оценка, а так же с нулевой $y$-координатой должны быть нанесены точки выборки. Для экономии места стройте графики в два столбца.

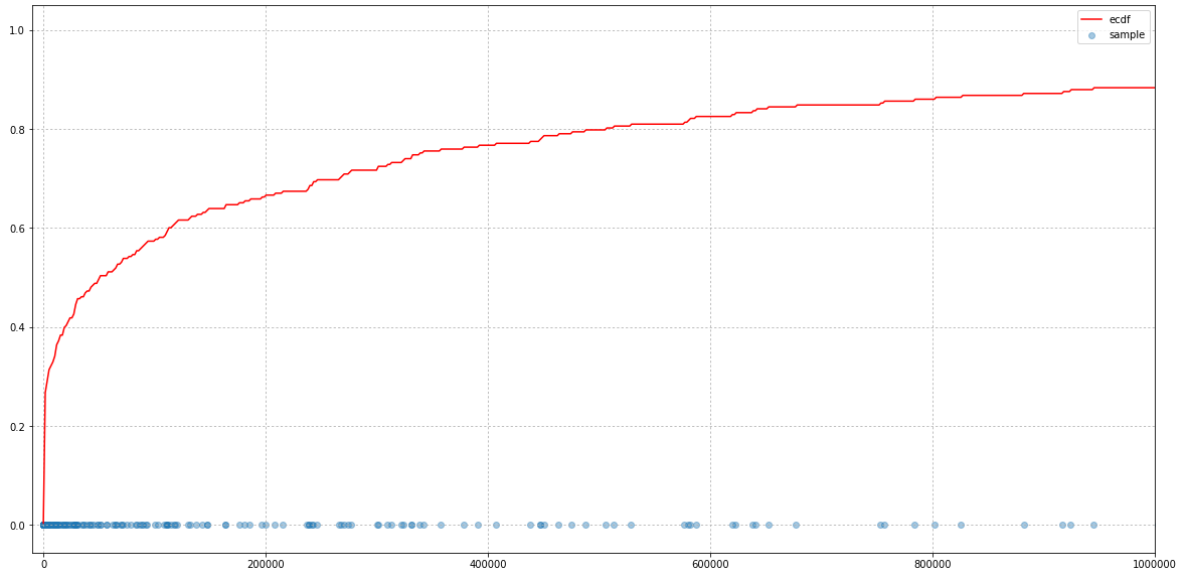
Не забудьте сделать вывод.

**Задача 6.** В файле `countries.csv` дан список стран и территорий с указанием их площади. Нанести

значения площади на график и постройте эмпирическую функцию распределения и ядерную оценку плотности. Поскольку некоторые страны слишком большие, ограничьте график по оси икс. Не забудьте сделать вывод.

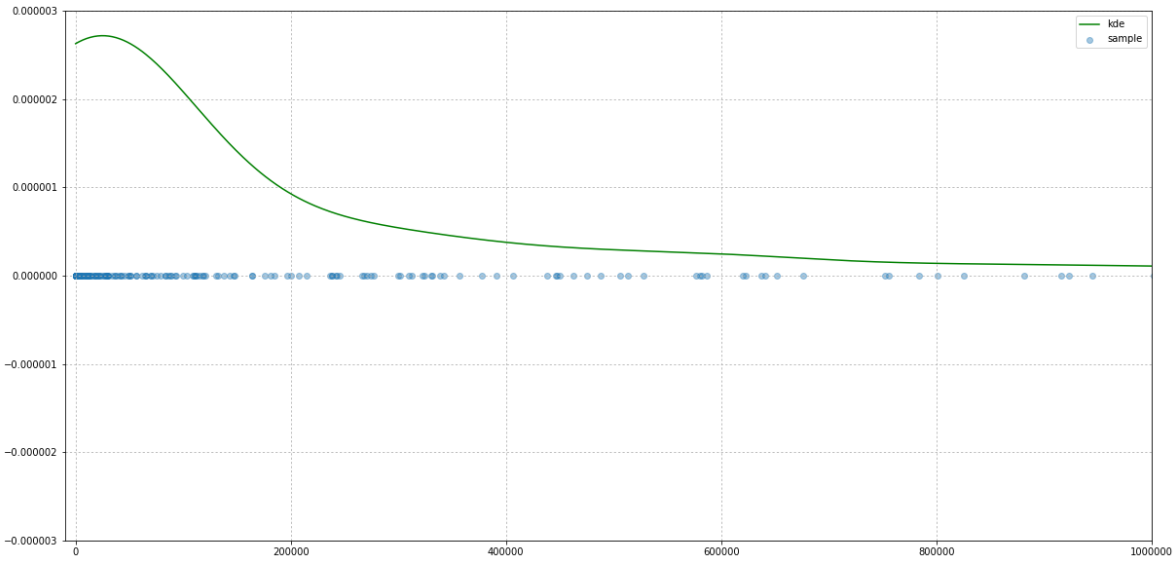
In [179]:

```
data = []
with open("countries.csv", "r") as file:
    for line in file:
        pieces = line.split('\t')
        data.append(float(pieces[2].split('\n')[0]))

grid = np.linspace(1, data[0], 10000)

plt.figure(figsize=(20,10))
plt.xlim([-10000, 1000000])
plt.scatter(data, np.zeros(len(data)), alpha=0.4, label='sample')
plt.plot(grid, ECDF(data)(grid), color='red', label='ecdf')
plt.legend()
plt.grid(ls=':')
plt.show()
```



In [180]:

```
plt.figure(figsize=(20,10))
plt.xlim([-10000, 1000000])
plt.ylim([-0.000003, 0.000003])
plt.scatter(data, np.zeros(len(data)), alpha=0.4, label='sample')
kernel_density = KDEUnivariate(data)
kernel_density.fit()
plt.plot(grid, kernel_density.evaluate(grid), color='green', label='kde')
plt.legend()
plt.grid(ls=':')
plt.show()
```



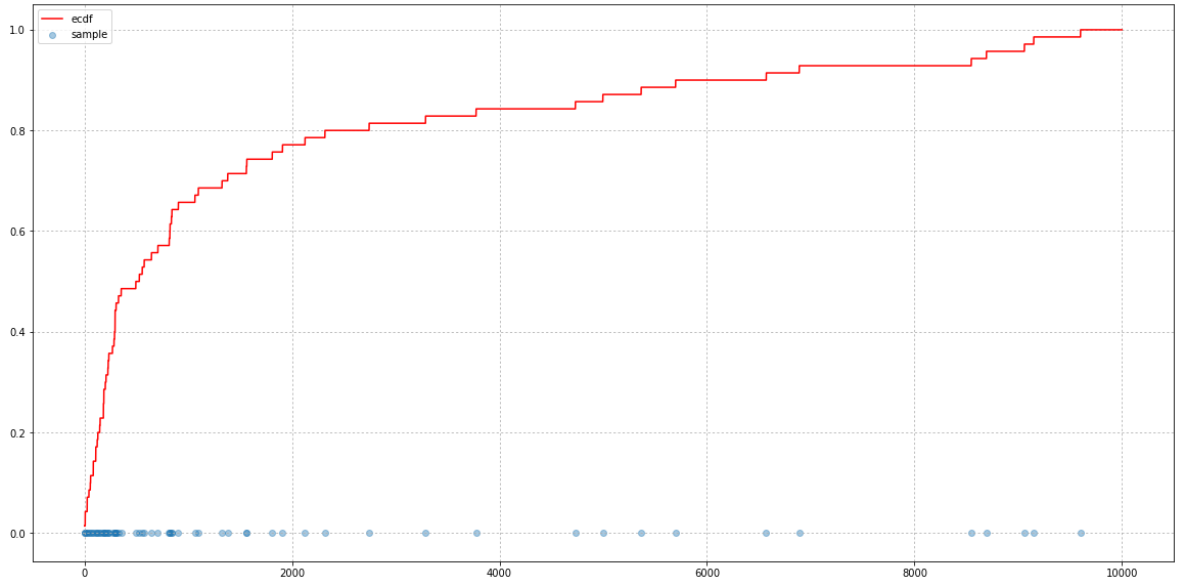

**Вывод:** Графики говорят о том, что выборка была из экспоненциального распределения, т.к. соответствуют графикам, полученным в задаче 3

**Задача 7.** Проведите небольшое исследование. Выберите случайных $n$ человек в социальной сети. Вы можете выбирать их случайно из всех зарегистрированных в этой социальной сети, либо по какому-то *одному* критерию (укажите его). Составьте выборку $X_1, \ldots, X_n$, где $X_i$ --- количество друзей у $i$-го человека. Постройте по этой выборке эмпирическую функцию распределения. Можете ли вы сказать, какому закону подчиняется распределение количества друзей?

Выборка должна быть из не менее 30 человек, ограничений сверху нет. Вы можете также написать программу, которая будет автоматически собирать данные. Не забудьте сделать вывод.

In [10]:

```
data = []
with open("vk.txt", "r") as file:
    for line in file:
        pieces = line.split(' ')
        data.append(float(pieces[2]))

grid = np.linspace(1, 10000, 70000)

plt.figure(figsize=(20,10))
plt.scatter(data, np.zeros(len(data)), alpha=0.4, label='sample')
plt.plot(grid, ECDF(data)(grid), color='red', label='ecdf')
plt.legend()
plt.grid(ls=':')
plt.show()
```




**Вывод:** Выборка производилась с параметром "город -- Санкт-Петербург", размер выборки -- 70. По эмпирической функции распределения можно сказать, что выборка была из экспоненциального распределения