# Project

October 15, 2024

## 1 №2

### 1.1 ,

#### 1.1.1 -02

[38]:
```
#jupyter nbconvert --to markdown Project.ipynb
```

### 1.2

[39]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
```

## 2 №1

### 2.1

[40]:
```
exp_sample_100 = np.random.exponential(scale=1, size=100)
exp_sample_1000 = np.random.exponential(scale=1, size=1000)



norm_sample_100 = np.random.normal(loc=0, scale=1, size=100)
norm_sample_1000 = np.random.normal(loc=0, scale=1, size=1000)
```

### 2.2

1.
2.
    1. 1 1 ,
    2. , . .

#### 2.2.1 numpy

[41]:
```
def calculate_mean_variance(sample, distribution_name):
    sample_mean = np.mean(sample)
    sample_variance = np.var(sample, ddof=1)
```

```
    if distribution_name == "exponential":
        theoretical_mean = 1
        theoretical_variance = 1
    elif distribution_name == "normal":
        theoretical_mean = 0
        theoretical_variance = 1


    print(f"              ({distribution_name}): {theoretical_mean}")
    print(f"            ({distribution_name}, n={len(sample)}): {sample_mean:.
↪4f}")
    print(f"              ({distribution_name}): {theoretical_variance}")
    print(f"             ({distribution_name}, n={len(sample)}):␣
↪{sample_variance:.4f}")
    print("")
```

### 2.3

[42]:
```
calculate_mean_variance(exp_sample_100, "exponential")
calculate_mean_variance(exp_sample_1000, "exponential")
calculate_mean_variance(norm_sample_100, "normal")
calculate_mean_variance(norm_sample_1000, "normal")
```

```
              (exponential): 1
            (exponential, n=100): 0.9101
              (exponential): 1
             (exponential, n=100): 0.5744

              (exponential): 1
            (exponential, n=1000): 0.9969
              (exponential): 1
             (exponential, n=1000): 0.9619

              (normal): 0
            (normal, n=100): -0.0815
              (normal): 1
             (normal, n=100): 1.2893

              (normal): 0
            (normal, n=1000): 0.0251
              (normal): 1
             (normal, n=1000): 0.9782
```

### 2.4

[43]:
```
def calculate_quantiles(sample, distribution_name):
    sample_quantiles = np.quantile(sample, [0.5, 0.99])

    if distribution_name == "exponential":
        theoretical_quantiles = stats.expon.ppf([0.5, 0.99], scale=1)
    elif distribution_name == "normal":
        theoretical_quantiles = stats.norm.ppf([0.5, 0.99], loc=0, scale=1)

    print(f"            ({distribution_name}, n={len(sample)}):␣
 ↪{sample_quantiles}")
    print(f"             ({distribution_name}): {theoretical_quantiles}")
    print("\n")
```

[44]:
```
calculate_quantiles(exp_sample_100, "exponential")
calculate_quantiles(exp_sample_1000, "exponential")
calculate_quantiles(norm_sample_100, "normal")
calculate_quantiles(norm_sample_1000, "normal")
```

```
            (exponential, n=100): [0.79599225 3.10469233]
             (exponential): [0.69314718 4.60517019]


            (exponential, n=1000): [0.69396014 4.1181586 ]
             (exponential): [0.69314718 4.60517019]


            (normal, n=100): [-0.11287732  2.37344646]
             (normal): [0.         2.32634787]


            (normal, n=1000): [-1.7143188e-03  2.2685730e+00]
             (normal): [0.         2.32634787]
```

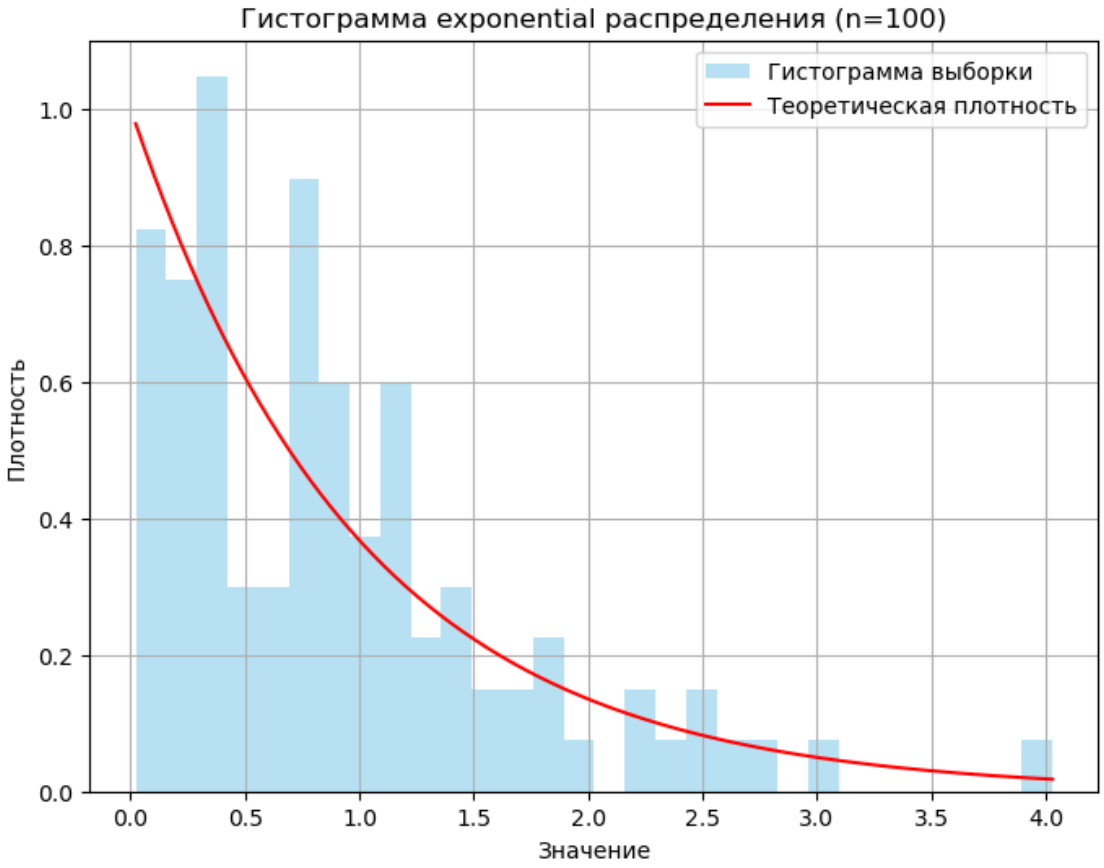
### 2.5

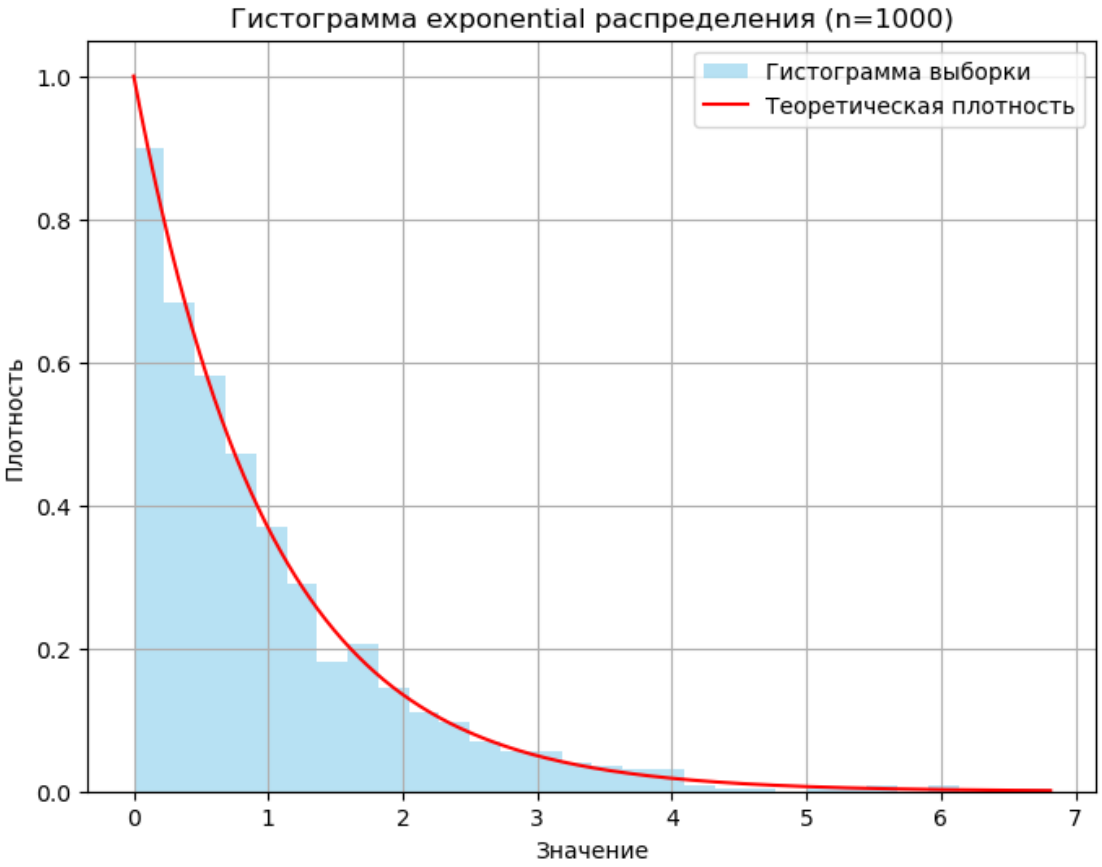
!

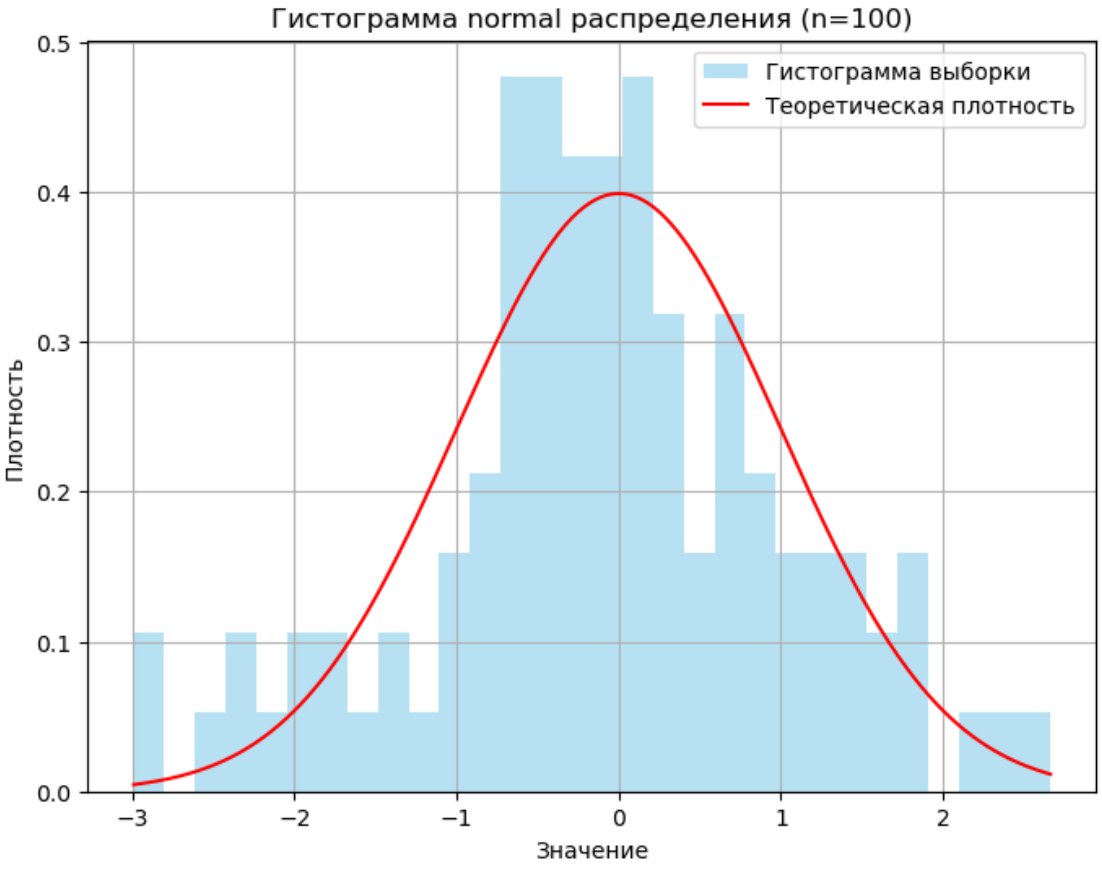
[45]:
```
def plot_histogram(sample, distribution_name, n):
    plt.figure(figsize=(8,6))
    plt.hist(sample, bins=30, density=True, alpha=0.6, color="skyblue",␣
 ↪label="           ")

    x = np.linspace(min(sample), max(sample), 1000)
    if distribution_name == "exponential":
```

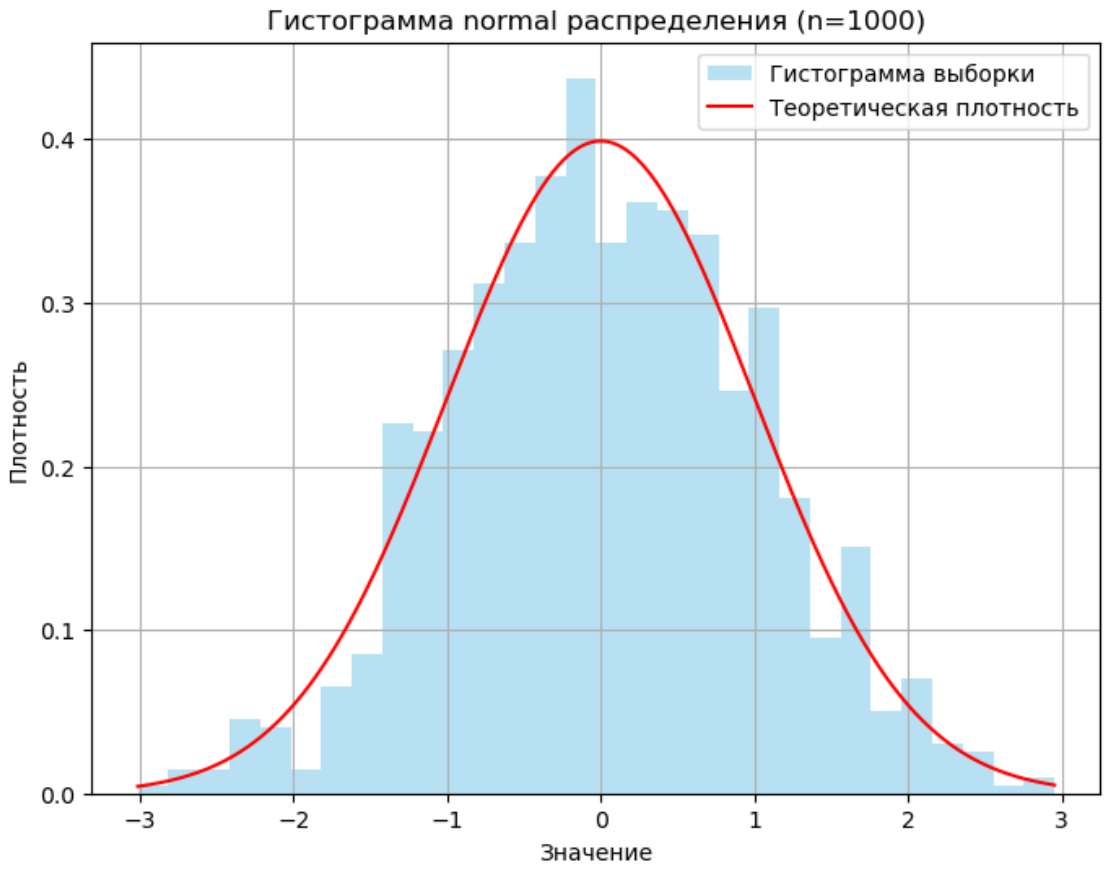
```
        pdf = stats.expon.pdf(x, scale=1)
    elif distribution_name == "normal":
        pdf = stats.norm.pdf(x, loc=0, scale=1)
    plt.plot(x, pdf, "r", label="              ")

    plt.title(f"       {distribution_name}          (n={n})")
    plt.xlabel("     ")
    plt.ylabel("     ")
    plt.legend()
    plt.grid(True)
    plt.show()
```

[46]:
```
plot_histogram(exp_sample_100, "exponential", 100)
plot_histogram(exp_sample_1000, "exponential", 1000)
plot_histogram(norm_sample_100, "normal", 100)
plot_histogram(norm_sample_1000, "normal", 1000)
```




Гистограмма exponential распределения (n=1000)
Гистограмма выборки
Теоретическая плотность
Плотность
Значение
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7


Гистограмма normal распределения (n=100)
Гистограмма выборки
Теоретическая плотность
Плотность
Значение
0.0
0.1
0.2
0.3
0.4
0.5
−3
−2
−1
0
1
2

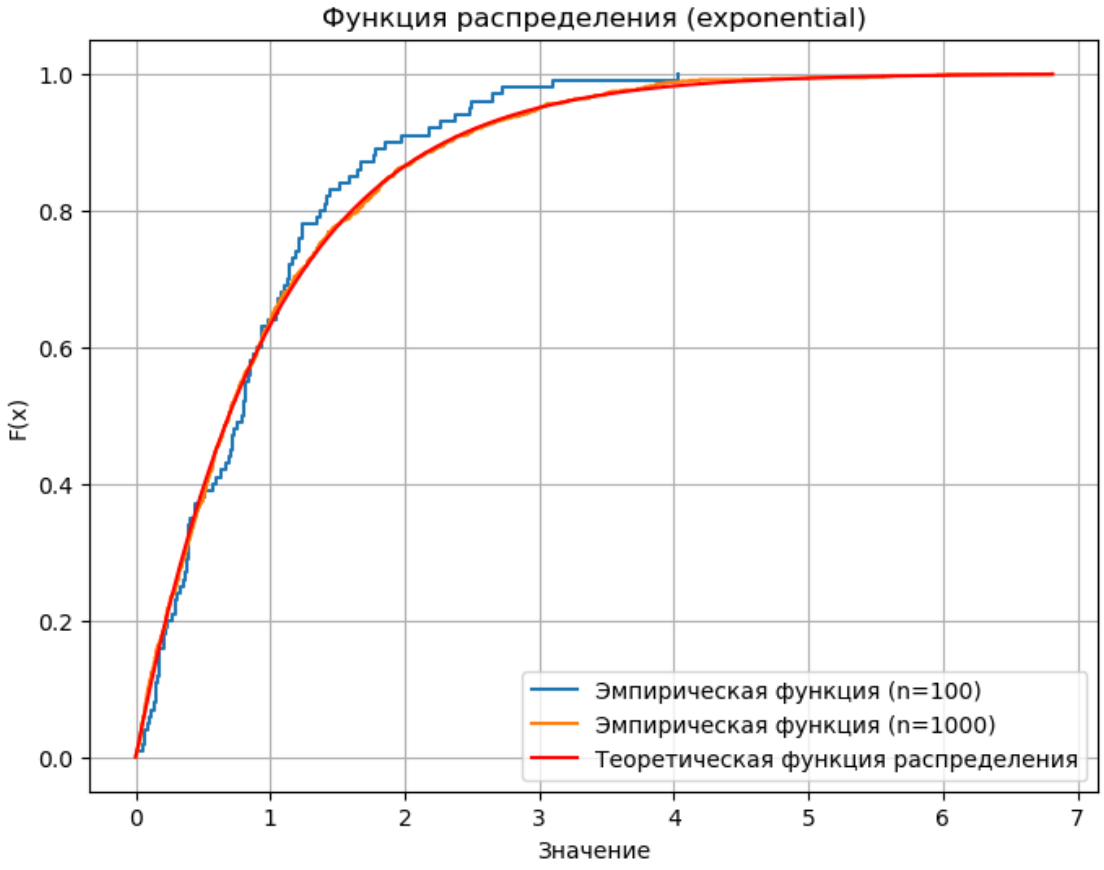


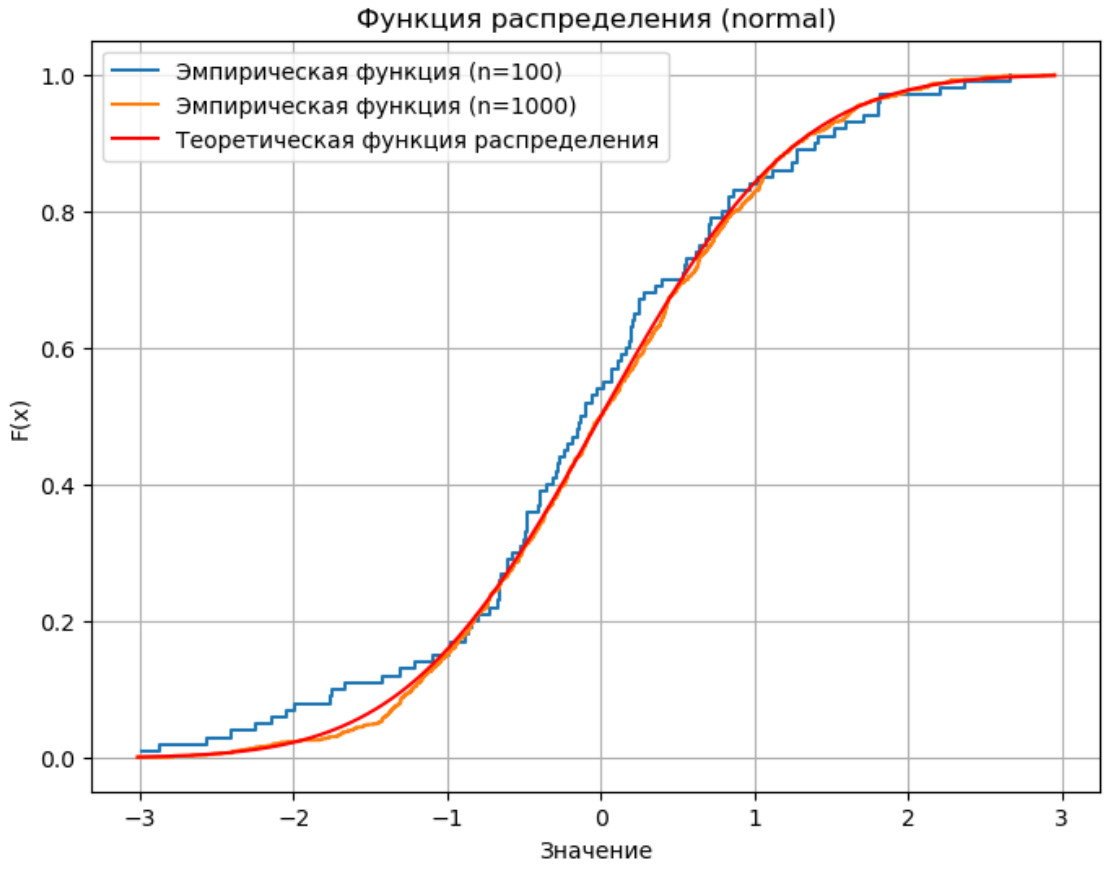

### 2.6

[47]:
```
def plot_cdf(sample_100, sample_1000, distribution_name):
    sorted_sample_100 = np.sort(sample_100)
    sorted_sample_1000 = np.sort(sample_1000)

    ecdf_100 = np.arange(1, len(sorted_sample_100)+1) / len(sorted_sample_100)
    ecdf_1000 = np.arange(1, len(sorted_sample_1000)+1) /␣
↪len(sorted_sample_1000)

    plt.figure(figsize=(8,6))
    plt.step(sorted_sample_100, ecdf_100, where="post", label="                ␣
↪(n=100)")
    plt.step(sorted_sample_1000, ecdf_1000, where="post", label="                ␣
↪(n=1000)")

    x = np.linspace(min(np.min(sorted_sample_100), np.min(sorted_sample_1000)),
                    max(np.max(sorted_sample_100), np.max(sorted_sample_1000)),␣
↪1000)
```

```
    if distribution_name == "exponential":
        cdf = stats.expon.cdf(x, scale=1)
    elif distribution_name == "normal":
        cdf = stats.norm.cdf(x, loc=0, scale=1)
    plt.plot(x, cdf, "r", label="                    ")

    plt.title(f"             ({distribution_name})")
    plt.xlabel("    ")
    plt.ylabel("F(x)")
    plt.legend()
    plt.grid(True)
    plt.show()
```

[48]:
```
plot_cdf(exp_sample_100, exp_sample_1000, "exponential")
plot_cdf(norm_sample_100, norm_sample_1000, "normal")
```



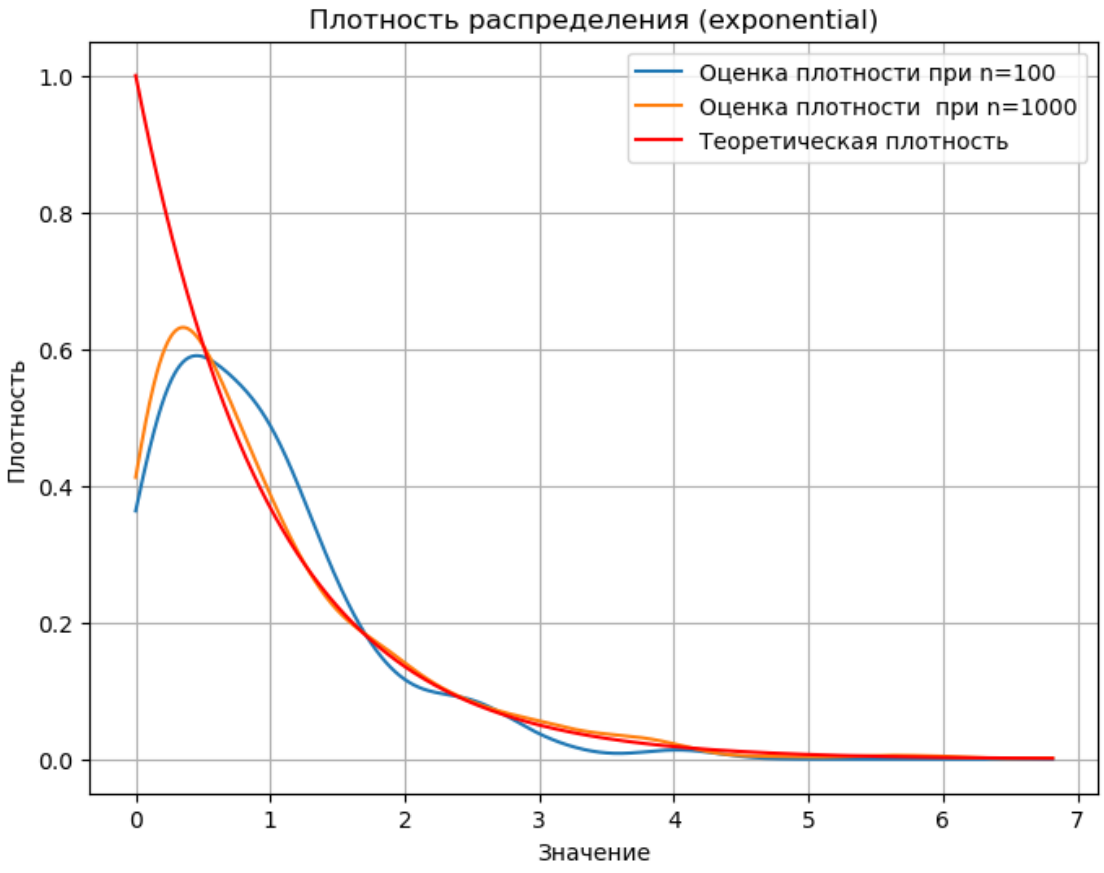


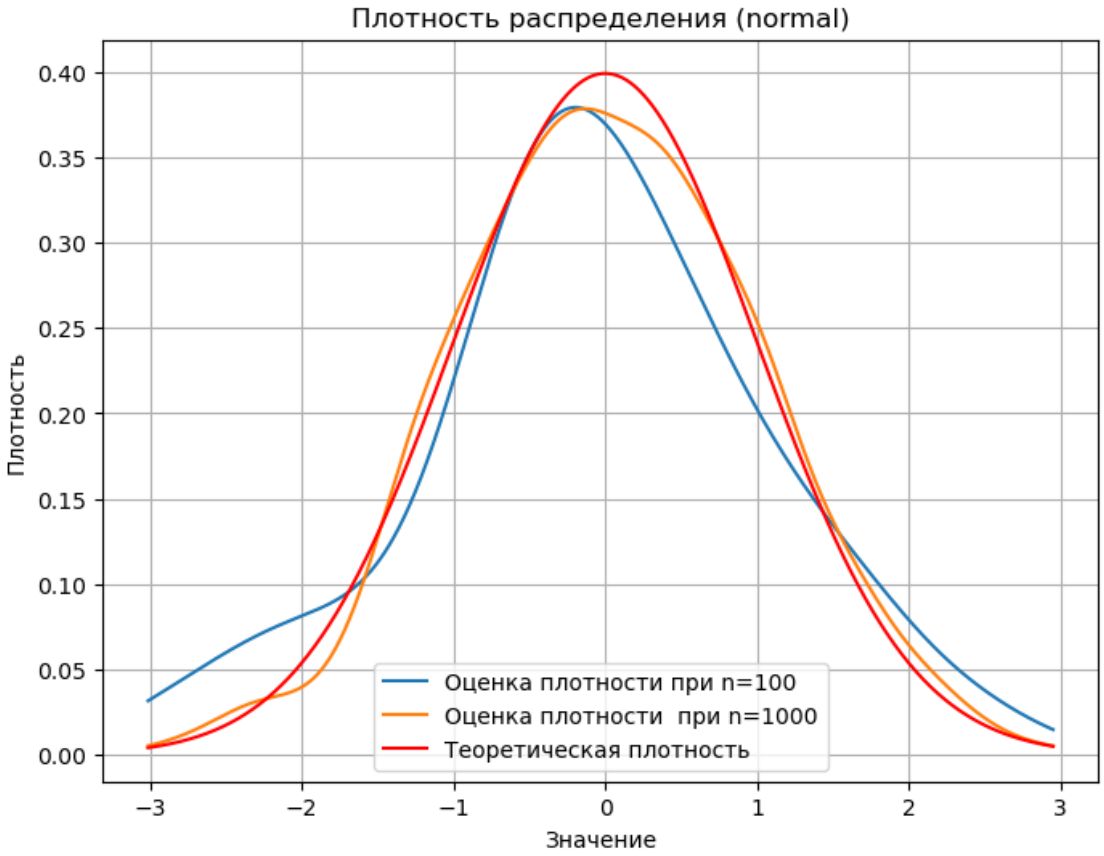

### 2.7

[49]:
```python
def plot_density(sample_100, sample_1000, distribution_name):
    from scipy.stats import gaussian_kde

    density_100 = gaussian_kde(sample_100)
    density_1000 = gaussian_kde(sample_1000)

    x = np.linspace(min(np.min(sample_100), np.min(sample_1000)),
                    max(np.max(sample_100), np.max(sample_1000)), 1000)

    plt.figure(figsize=(8,6))
    plt.plot(x, density_100(x), label="              n=100")
    plt.plot(x, density_1000(x), label="               n=1000")

    if distribution_name == "exponential":
        pdf = stats.expon.pdf(x, scale=1)
```

```
    elif distribution_name == "normal":
        pdf = stats.norm.pdf(x, loc=0, scale=1)
    plt.plot(x, pdf, 'r', label="              ")

    plt.title(f"              ({distribution_name})")
    plt.xlabel("     ")
    plt.ylabel("     ")
    plt.legend()
    plt.grid(True)
    plt.show()
```

[50]:
```
plot_density(exp_sample_100, exp_sample_1000, "exponential")
plot_density(norm_sample_100, norm_sample_1000, "normal")
```






## 3 №2

*100, 1000 10000* ,
*10 30.* ,
. .

### 3.1 2

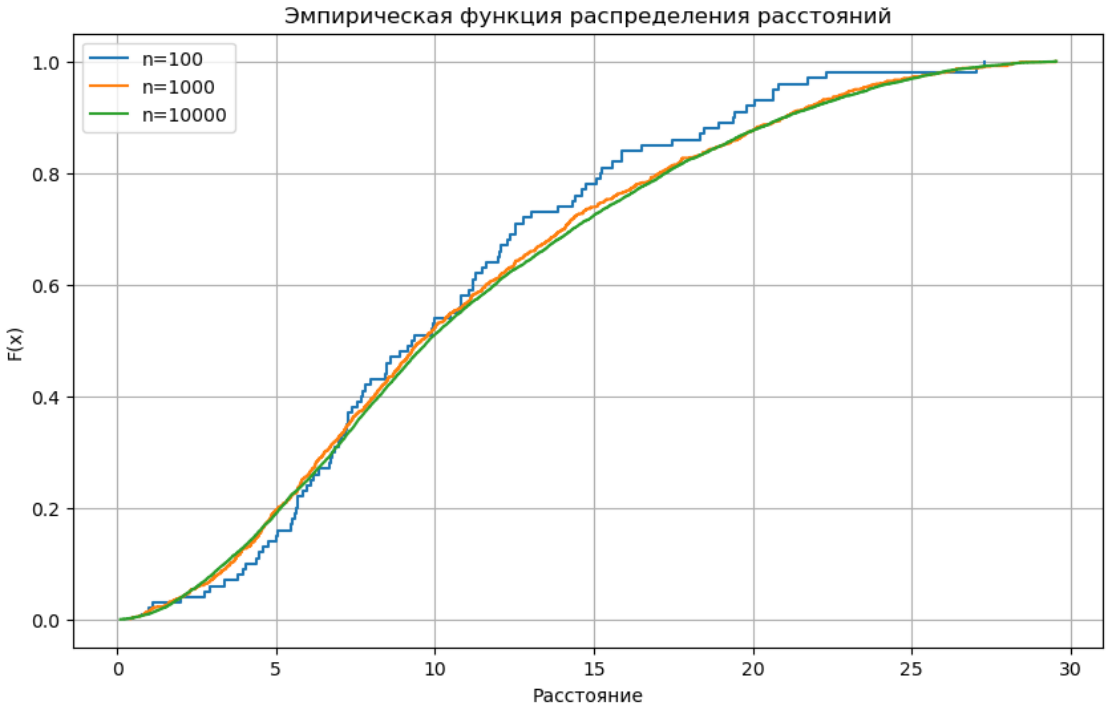
[51]:
```
def generate_points(sample_size, width=10, height=30):
    x1 = np.random.uniform(0, width, sample_size)
    y1 = np.random.uniform(0, height, sample_size)
    x2 = np.random.uniform(0, width, sample_size)
    y2 = np.random.uniform(0, height, sample_size)
    distances = np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
    return distances
```

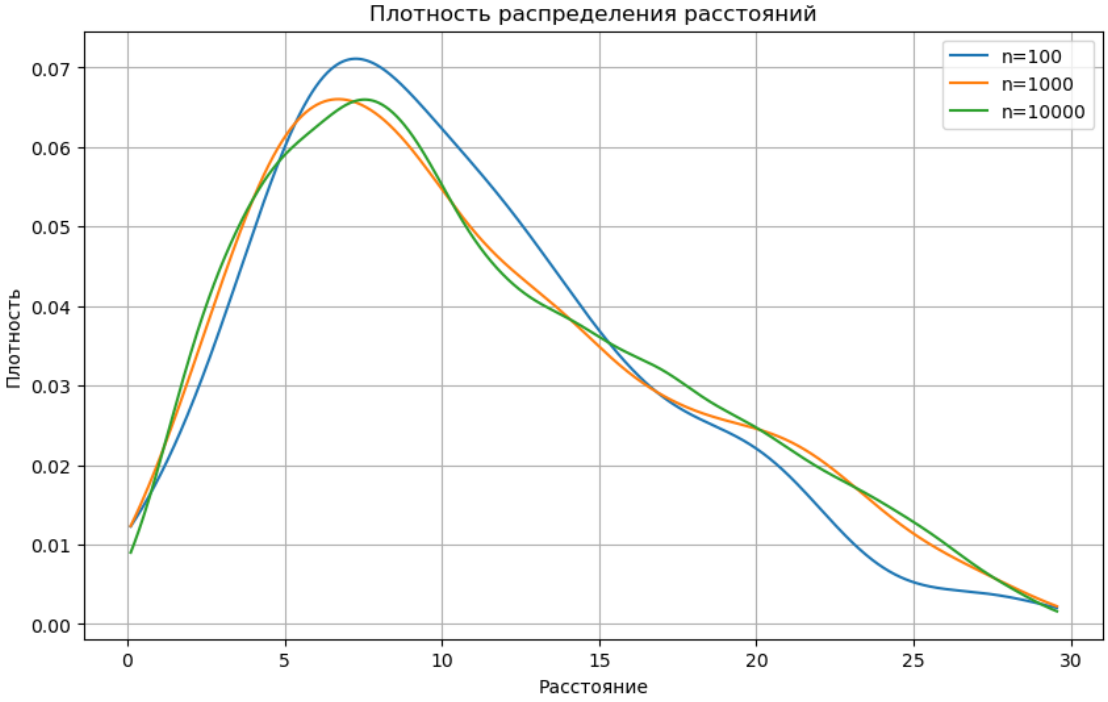
### 3.2 3 ,

[52]:
```
sizes = [100, 1000, 10000]
distance_samples = {}

for size in sizes:
    distances = generate_points(size, width=10, height=30)
    distance_samples[size] = distances
```

[55]:
```
from scipy.stats import gaussian_kde
def plot_cdf(samples_dict):
    plt.figure(figsize=(10,6))
    for size, distances in samples_dict.items():
        sorted_distances = np.sort(distances)
        ecdf = np.arange(1, len(sorted_distances)+1) / len(sorted_distances)
        plt.step(sorted_distances, ecdf, where="post", label=f"n={size}")

    plt.title("                          ")
    plt.xlabel("     ")
    plt.ylabel("F(x)")
    plt.legend()
    plt.grid(True)
    plt.show()

def plot_pdf(samples_dict):
    plt.figure(figsize=(10,6))
    x_min = min([min(distances) for distances in samples_dict.values()])
    x_max = max([max(distances) for distances in samples_dict.values()])
    x = np.linspace(x_min, x_max, 1000)

    for size, distances in samples_dict.items():
        density = gaussian_kde(distances)
        plt.plot(x, density(x), label=f'n={size}')

    plt.title("                   ")
    plt.xlabel("     ")
    plt.ylabel("     ")
    plt.legend()
    plt.grid(True)
    plt.show()
```

[56]:
```
plot_cdf(distance_samples)
plot_pdf(distance_samples)
```


Эмпирическая функция распределения расстояний
n=100
n=1000
n=10000
F(x)
Расстояние
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30


Плотность распределения расстояний
n=100
n=1000
n=10000
Плотность
Расстояние
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0
5
10
15
20
25
30