



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

---

**Институт информационных технологий (ИИТ)  
Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**  
по дисциплине «Технологии и инструментарий анализа больших данных»

### **Практическая работа № 3**

Студент группы *ИКБО-01-22 Прокопчук Роман Олегович*

\_\_\_\_\_  
(подпись)

Ассистент *Тетерин Николай Николаевич*

\_\_\_\_\_  
(подпись)

Отчёт представлен «\_\_» октября 2025 г.

Москва, 2025 г.

## **ЦЕЛЬ**

Ознакомиться с инструментами работы со статистикой в Python.

## ХОД РАБОТЫ

### Задание

1. Загрузить данные из файла “insurance.csv”.
2. С помощью метода describe() посмотреть статистику по данным.  
Сделать выводы.
3. Построить гистограммы для числовых показателей. Сделать выводы.
4. Найти меры центральной тенденции и меры разброса для индекса массы тела (bmi) и расходов (charges). Отобразить результаты в виде текста и на гистограммах (3 вертикальные линии). Добавить легенду на графики.  
Сделать выводы.
5. Построить box-plot для числовых показателей. Названия графиков должны соответствовать названиям признаков. Сделать выводы.
6. Используя признак charges или imb, проверить, выполняется ли центральная предельная теорема. Использовать различные длины выборок n. Количество выборок = 300. Вывести результат в виде гистограмм. Найти стандартное отклонение и среднее для полученных распределений. Сделать выводы.
7. Построить 95% и 99% доверительный интервал для среднего значения расходов и среднего значения индекса массы тела.
8. Проверить распределения следующих признаков на нормальность: индекс массы тела, расходы. Сформулировать нулевую и альтернативную гипотезы. Для каждого признака использовать KS-тест и q-q plot. Сделать выводы на основе полученных p-значений.
9. Загрузить данные из файла “ECDCCases.csv”.
10. Проверить в данных наличие пропущенных значений. Вывести количество пропущенных значений в процентах. Удалить два признака, в которых больше всех пропущенных значений. Для оставшихся признаков обработать пропуски: для категориального признака использовать заполнение значением по умолчанию (например, «other»), для числового признака

использовать заполнение медианным значением. Показать, что пропусков больше в данных нет.

11. Посмотреть статистику по данным, используя `describe()`. Сделать выводы о том, какие признаки содержат выбросы. Посмотреть, для каких стран количество смертей в день превысило 3000 и сколько таких дней было.

12. Найти дублирование данных. Удалить дубликаты.

13. Загрузить данные из файла “bmi.csv”. Взять оттуда две выборки. Одна выборка – это индекс массы тела людей с региона northwest, вторая выборка – это индекс массы тела людей с региона southwest. Сравнить средние значения этих выборок, используя t-критерий Стьюдента. Предварительно проверить выборки на нормальность (критерий ШопироУилка) и на гомогенность дисперсии (критерий Бартлетта).

14. Кубик бросили 600 раз, получили следующие результаты: N  
Количество выпадений 1 – 97, 2 – 98, 3 – 109, 4 – 95, 5 – 97, 6 – 104. С помощью критерия Хи-квадрат проверить, является ли полученное распределение равномерным. Использовать функцию `scipy.stats.chisquare()`.

15. С помощью критерия Хи-квадрат проверить, являются ли переменные зависимыми. Создать датафрейм, используя следующий код: `data = pd.DataFrame({'Женат': [89,17,11,43,22,1], 'Гражданский брак': [80,22,20,35,6,4], 'Не состоит в отношениях': [35,44,35,6,8,22]})` `data.index = ['Полный рабочий день','Частичная занятость','Временно не работает','На домохозяйстве','На пенсии','Учёба']` Использовать функцию `scipy.stats.chi2_contingency()`. Влияет ли семейное положение на занятость?

16. Оформить отчет о проделанной работе, написать выводы.

### **Insurance (задания 1-8)**

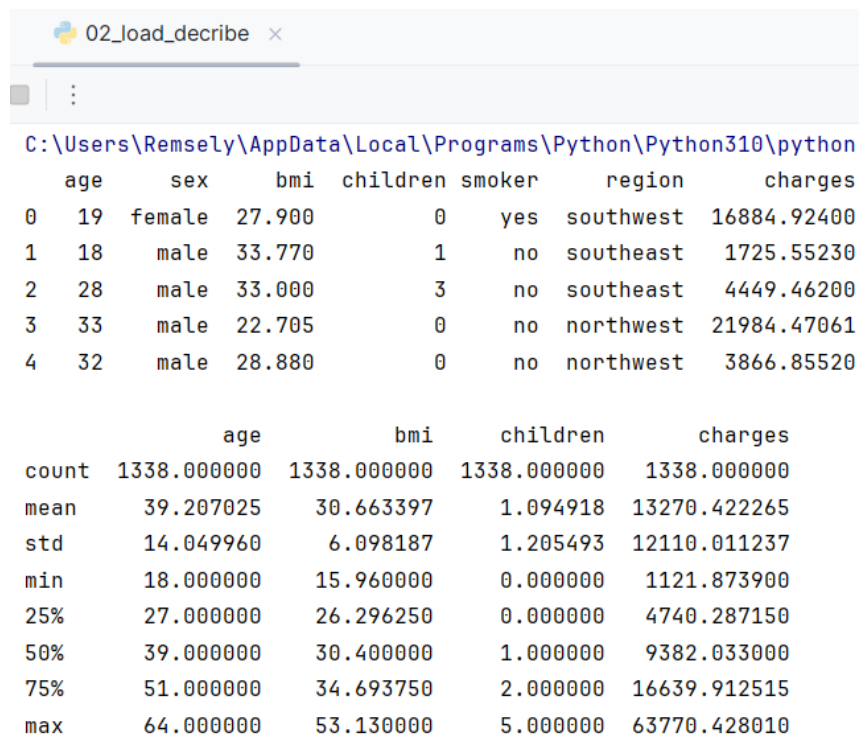
На рисунке 1 представлен скрипт для выгрузки и вывода описания данных.

```
import pandas as pd

insurance_df = pd.read_csv('data/insurance.csv')
print(insurance_df.head())
print()
print(insurance_df.describe())
```

Рисунок 1 – Код загрузки данных

Результат выполнения программы представлен на рисунке 2.



	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Рисунок 2 –Вывод описания данных

Вывод:

- В датасете 1338 записей о застрахованных лицах;
- Возраст варьируется от 18 до 64 лет, средний возраст ~39 лет;
- Индекс массы тела среднее значение 30.66, стандартное отклонение 6.1 указывает на умеренный разброс значений;
- Большинство застрахованных не имеют детей (медиана = 1), максимум -5 детей;
- Сильная вариативность расходов — от \$1,121 до \$63,770, среднее (\$13,270) значительно выше медианы (\$9,382), что указывает на наличие выбросов и правостороннюю асимметрию распределения.

На рисунке 3 представлен код для построения гистограмм для числовых значений. Итоговые диаграммы представлены на рисунке 4.

```

import matplotlib.pyplot as plt
import pandas as pd

insurance_df = pd.read_csv('data/insurance.csv')

numerical_features = ['age', 'bmi', 'children', 'charges']
insurance_df[numerical_features].hist(
    bins=30,
    figsize=(12, 8),
    layout=(2, 2),
    edgecolor='black'
)
plt.suptitle("Гистограммы для числовых показателей", y=1.02)
plt.show()

```

Рисунок 3 – Код для построения гистограмм

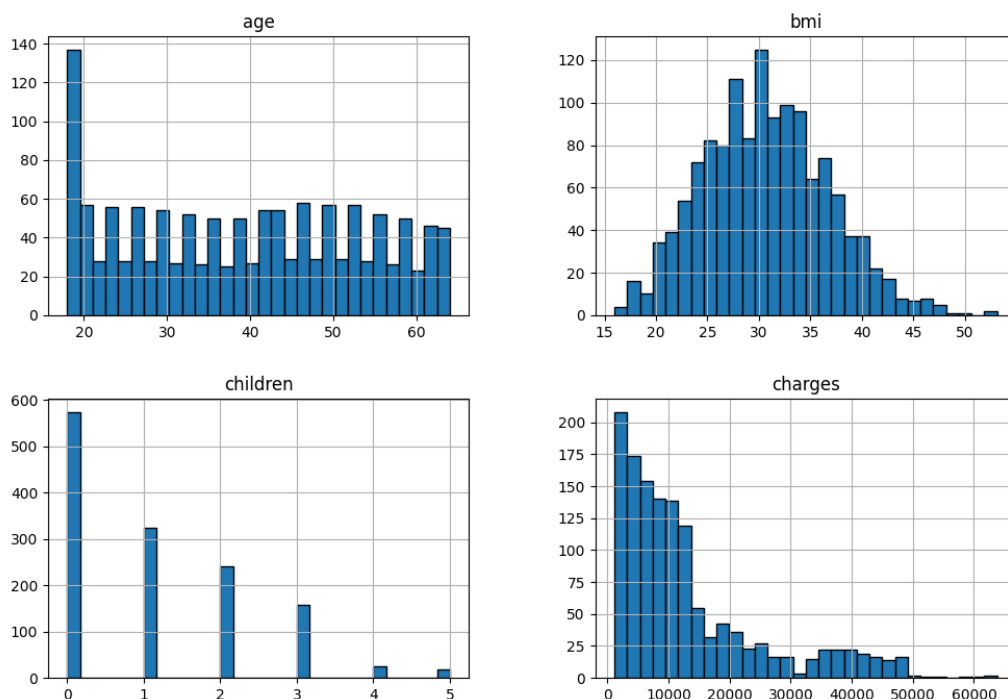


Рисунок 4 – Построенные гистограммы

#### Выводы:

- age: Распределение относительно равномерное с пиком в молодом возрасте;
- bmi: Распределение близко к нормальному с центром около 30;
- children: Преобладают застрахованные без детей, семьи с 4-5 детьми встречаются редко.
- charges: Большинство расходов концентрируются в низком диапазоне (0-20000\$), но есть значительное количество случаев с высокими расходами (35000+ \$)/

Код для выполнения четвертого задания представлен на рисунке 5.

Вывод программы представлен на рисунке 6.

```
import matplotlib.pyplot as plt
import pandas as pd

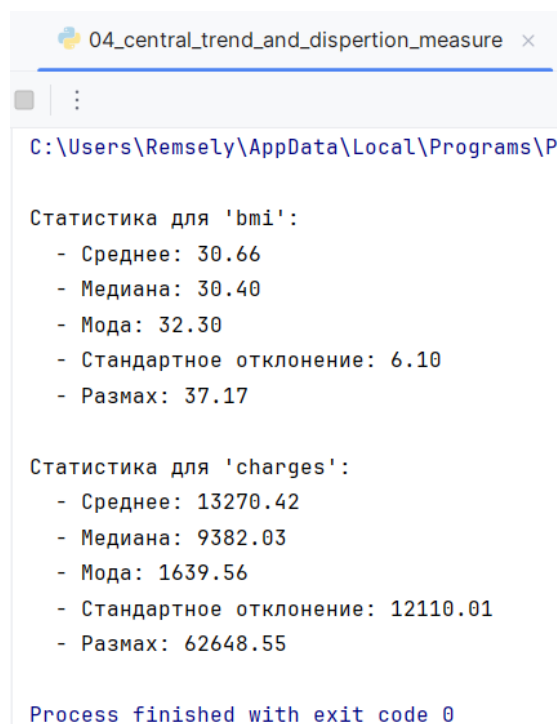
insurance_df = pd.read_csv('data/insurance.csv')

for column in ['bmi', 'charges']:
    mean_val = insurance_df[column].mean()
    median_val = insurance_df[column].median()
    mode_val = insurance_df[column].mode()[0]
    std_val = insurance_df[column].std()
    range_val = insurance_df[column].max() - insurance_df[column].min()

    print(f"\nСтатистика для '{column}':")
    print(f"  - Среднее: {mean_val:.2f}")
    print(f"  - Медиана: {median_val:.2f}")
    print(f"  - Мода: {mode_val:.2f}")
    print(f"  - Стандартное отклонение: {std_val:.2f}")
    print(f"  - Размах: {range_val:.2f}")

    plt.figure(figsize=(10, 6))
    plt.hist(insurance_df[column], bins=50, edgecolor='black', alpha=0.7)
    plt.axvline(mean_val, color='red', linestyle='dashed', linewidth=2, label=f'Среднее: {mean_val:.2f}')
    plt.axvline(median_val, color='green', linestyle='solid', linewidth=2, label=f'Медиана: {median_val:.2f}')
    plt.axvline(mode_val, color='yellow', linestyle='dotted', linewidth=2, label=f'Мода: {mode_val:.2f}')
    plt.title(f'Распределение показателя "{column}"')
    plt.xlabel(column)
    plt.ylabel('Частота')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Рисунок 5 – Код для выполнения четвертого задания



```
04_central_trend_and_dispersion_measure x

C:\Users\Remsely\AppData\Local\Programs\Python\Python38-32\python.exe

Статистика для 'bmi':
  - Среднее: 30.66
  - Медиана: 30.40
  - Мода: 32.30
  - Стандартное отклонение: 6.10
  - Размах: 37.17

Статистика для 'charges':
  - Среднее: 13270.42
  - Медиана: 9382.03
  - Мода: 1639.56
  - Стандартное отклонение: 12110.01
  - Размах: 62648.55

Process finished with exit code 0
```

Рисунок 6 – Вывод программы

Созданные диаграммы представлены на рисунках 7-8

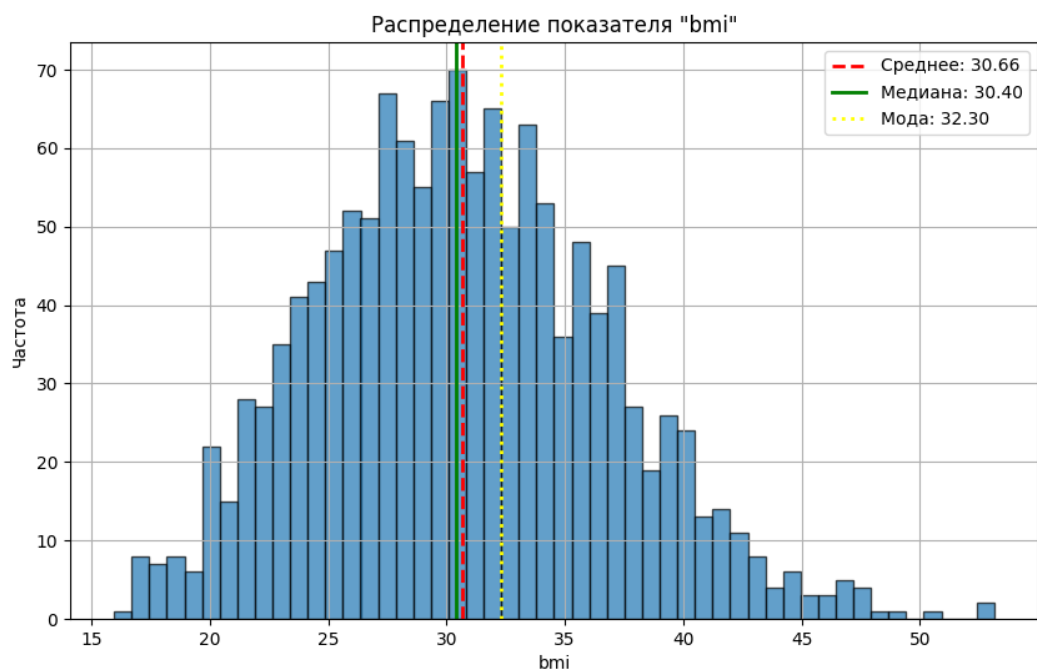


Рисунок 7 – Распределение показателя bmi

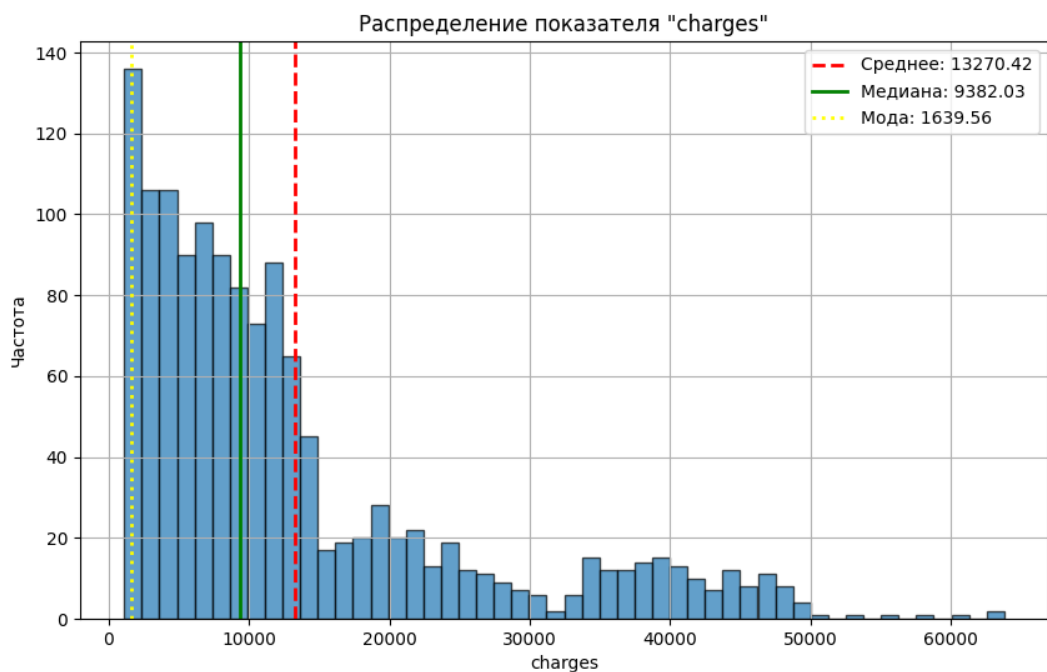


Рисунок 8 – Распределение показателя charges

Выводы для bmi:

- Среднее (30.66), медиана (30.40) и мода (32.30) находятся близко друг к другу, что подтверждает относительную симметричность распределения;
- Стандартное отклонение 6.10 показывает умеренную вариабельность;
- Размах 37.17 (от 15.96 до 53.13).



Выводы для charges:

- Сильное расхождение между средним (13,270) и медианой (9,382) указывает на правостороннюю асимметрию;
- Мода (1,639.56) значительно ниже медианы, что подтверждает скопление данных в нижней части распределения;
- Огромный размах (62,648.55) и очень большое стандартное отклонение (12,110) свидетельствуют о высокой неоднородности расходов;
- Распределение типично для страховых расходов: большинство случаев имеют низкие затраты, но редкие серьезные заболевания создают очень высокие расходы.

На рисунке 9 представлен скрипт для создания box-plot. Сами диаграммы представлены на рисунке 10.

```
import matplotlib.pyplot as plt
import pandas as pd

insurance_df = pd.read_csv('data/insurance.csv')
numerical_features = ['age', 'bmi', 'children', 'charges']

plt.figure(figsize=(12, 8))

for i, column in enumerate(numerical_features, 1):
    plt.subplot(*args: 2, 2, i)
    plt.boxplot(insurance_df[column], vert=False)
    plt.title(f'Box-plot для "{column}"')
    plt.xlabel(column)

plt.tight_layout()
plt.show()
```

Рисунок 9 – Код для создания box-plot

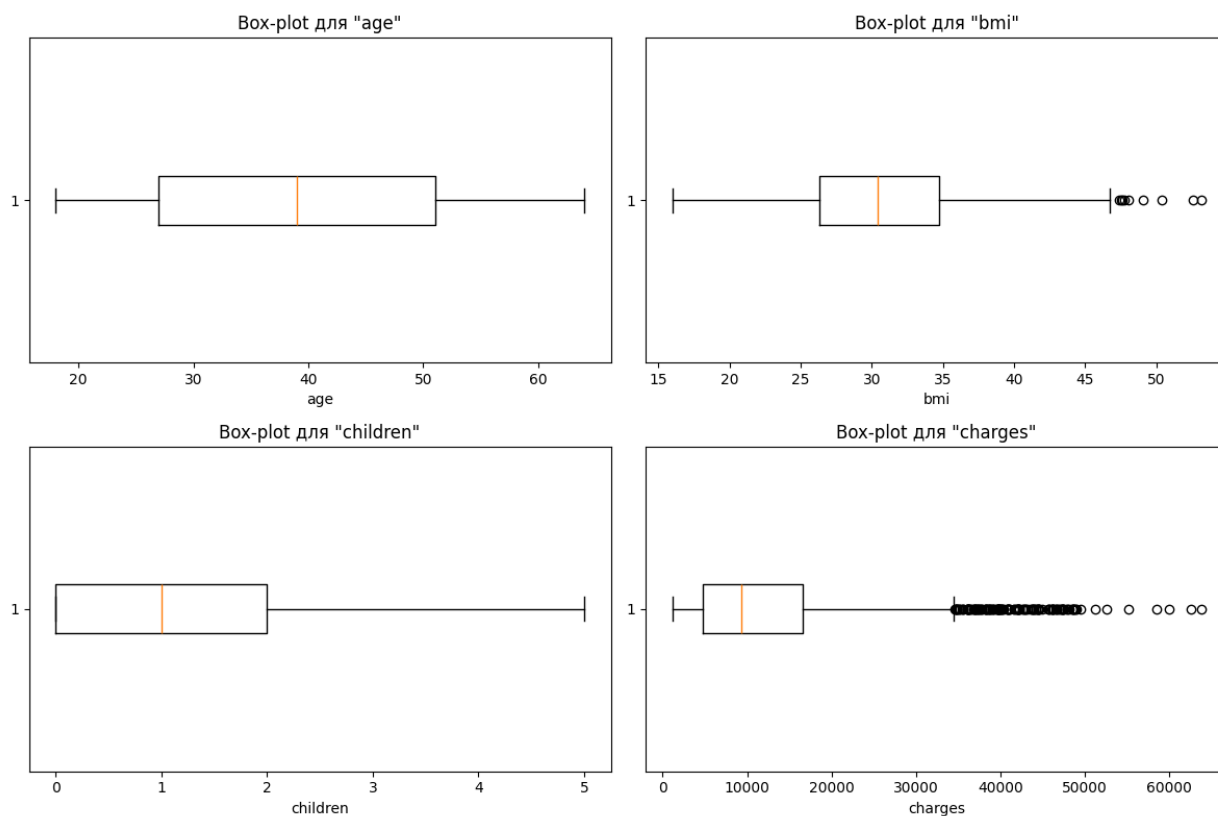


Рисунок 10 – Созданные box-plot

Выводы:

- Age – распределение симметрично;
- bmi – распределение почти симметрично с небольшим количеством выбросов в верхней части (люди с очень высоким ИМТ >45-50);
- children – Сильно асимметричное распределение — медиана смещена к нижней границе, большинство значений концентрируется на 0-2 детей, значения 4-5 являются выбросами;
- charges – наиболее асимметричное распределение с множеством выбросов в верхней части. Медиана смещена к нижней границе коробки, верхний ус значительно длиннее нижнего, что подтверждает правостороннюю асимметрию. Многочисленные выбросы (>\$30,000) соответствуют случаям с серьезными заболеваниями или осложнениями.

Код для проверки выполнения центральной предельной теоремы представлен на рисунке 11. Вывод программы представлен на рисунке 12, получившиеся диаграммы – на рисунке 13.

```

insurance_df = pd.read_csv('data/insurance.csv')
charges_data = insurance_df['charges']

num_samples = 300
sample_sizes = [5, 30, 100, 500]

fig, axes = plt.subplots( nrows= 2, ncols= 2, figsize=(14, 10))
fig.suptitle( t: 'Распределение выборочных средних для "charges" при разных n', fontsize=16)

axes = axes.flatten()

for i, n in enumerate(sample_sizes):
    sample_means = []
    for _ in range(num_samples):
        sample = charges_data.sample(n, replace=True)
        sample_means.append(sample.mean())

    ax = axes[i]

    counts, bins, patches = ax.hist(sample_means, bins=30, edgecolor='black',
                                     alpha=0.7, density=True, label='Выборочные средние')
    mean_of_means = np.mean(sample_means)
    std_of_means = np.std(sample_means)

    x = np.linspace(min(sample_means), max(sample_means), num= 100)
    normal_curve = stats.norm.pdf(x, mean_of_means, std_of_means)
    ax.plot(x, normal_curve, 'r-', linewidth=2, label='Нормальное распределение')

    ax.set_title(f'n = {n}')
    ax.set_xlabel('Среднее расходов')
    if i % 2 == 0:
        ax.set_ylabel('Плотность вероятности')
    ax.legend()
    ax.grid(True, alpha=0.3)

```

Рисунок 11 – Код для проверки центральной предельной теоремы

```

insurance_df = pd.read_csv('data/insurance.csv')
charges_data = insurance_df['charges']

num_samples = 300
sample_sizes = [5, 30, 100, 500]

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
fig.suptitle('Распределение выборочных средних для "charges" при разных n', fontsize=16)

axes = axes.flatten()

for i, n in enumerate(sample_sizes):
    sample_means = []
    for _ in range(num_samples):
        sample = charges_data.sample(n, replace=True)
        sample_means.append(sample.mean())

    ax = axes[i]

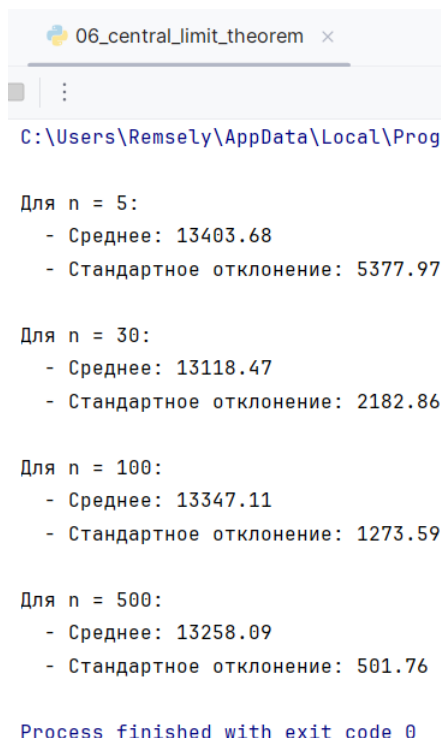
    counts, bins, patches = ax.hist(sample_means, bins=30, edgecolor='black',
                                     alpha=0.7, density=True, label='Выборочные средние')
    mean_of_means = np.mean(sample_means)
    std_of_means = np.std(sample_means)

    x = np.linspace(min(sample_means), max(sample_means), num=100)
    normal_curve = stats.norm.pdf(x, mean_of_means, std_of_means)
    ax.plot(x, normal_curve, 'r-', linewidth=2, label='Нормальное распределение')

    ax.set_title(f'n = {n}')
    ax.set_xlabel('Среднее расходов')
    if i % 2 == 0:
        ax.set_ylabel('Плотность вероятности')
    ax.legend()
    ax.grid(True, alpha=0.3)

```

Рисунок 11 – Код для проверки центральной предельной теоремы



```

06_central_limit_theorem x
C:\Users\Remsely\AppData\Local\Prog

Для n = 5:
- Среднее: 13403.68
- Стандартное отклонение: 5377.97

Для n = 30:
- Среднее: 13118.47
- Стандартное отклонение: 2182.86

Для n = 100:
- Среднее: 13347.11
- Стандартное отклонение: 1273.59

Для n = 500:
- Среднее: 13258.09
- Стандартное отклонение: 501.76

Process finished with exit code 0

```

Рисунок 12 – Вывод программы

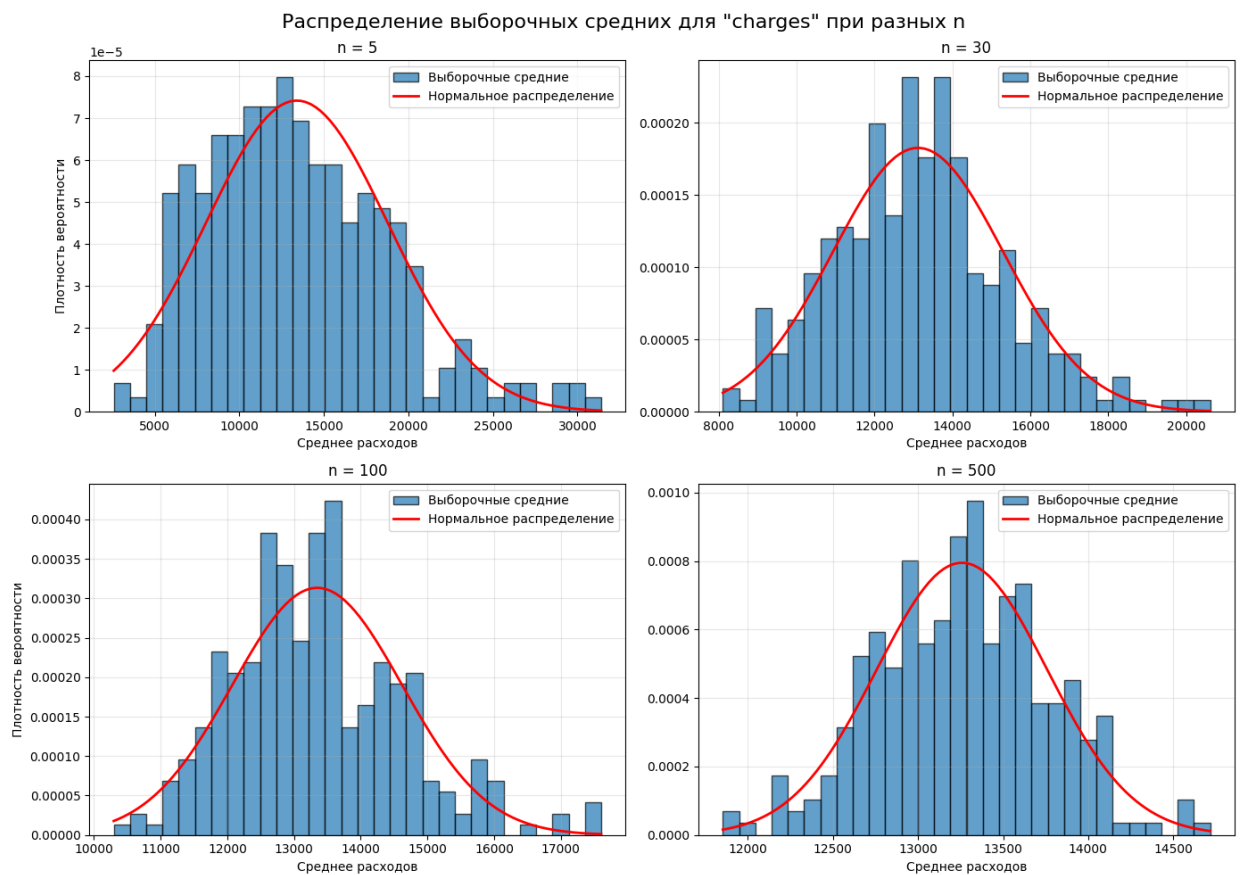


Рисунок 13 – Результаты в виде гистограмм

При увеличении размера выборки  $n$  распределение выборочных средних приближается к нормальному, что полностью подтверждает центральную предельную теорему.

Код для нахождения доверительных интервалов представлен на рисунке 14, результат его работы – на рисунке 15.

```

import numpy as np
import pandas as pd
import scipy.stats as st

insurance_df = pd.read_csv('data/insurance.csv')

for column in ['bmi', 'charges']:
    print(f"\nРасчет доверительных интервалов для '{column}':")

    data = insurance_df[column]

    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    n = len(data)

    se = sample_std / np.sqrt(n)

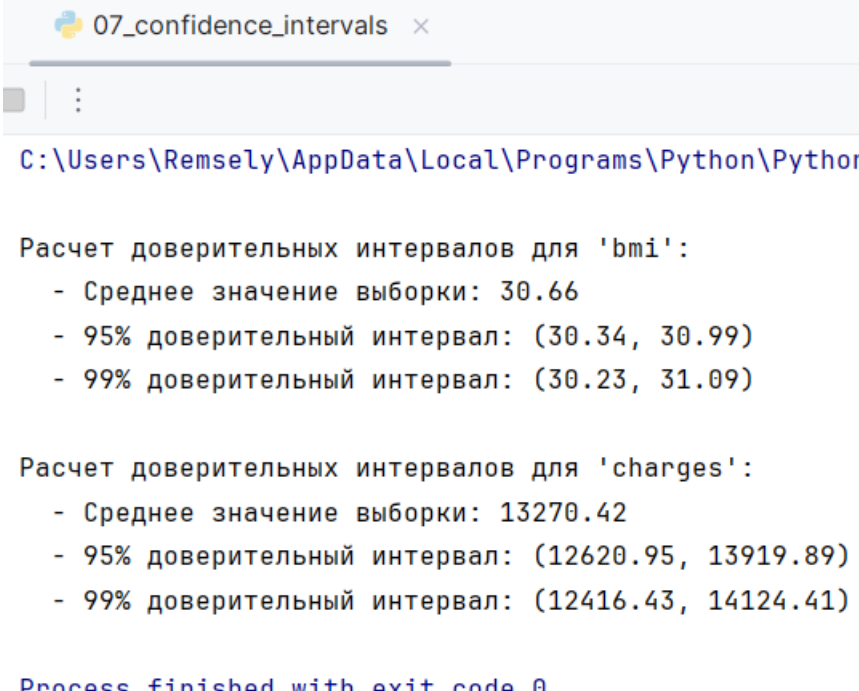
    confidence_level_95 = 0.95
    ci_95 = st.t.interval(confidence_level_95, df=n - 1, loc=sample_mean, scale=se)

    confidence_level_99 = 0.99
    ci_99 = st.t.interval(confidence_level_99, df=n - 1, loc=sample_mean, scale=se)

    print(f" - Среднее значение выборки: {sample_mean:.2f}")
    print(f" - 95% доверительный интервал: ({ci_95[0]:.2f}, {ci_95[1]:.2f})")
    print(f" - 99% доверительный интервал: ({ci_99[0]:.2f}, {ci_99[1]:.2f})")

```

Рисунок 14 – Код для нахождения доверительных интервалов



```

C:\Users\Remsely\AppData\Local\Programs\Python\Pythonor

Расчет доверительных интервалов для 'bmi':
- Среднее значение выборки: 30.66
- 95% доверительный интервал: (30.34, 30.99)
- 99% доверительный интервал: (30.23, 31.09)

Расчет доверительных интервалов для 'charges':
- Среднее значение выборки: 13270.42
- 95% доверительный интервал: (12620.95, 13919.89)
- 99% доверительный интервал: (12416.43, 14124.41)

Process finished with exit code 0

```

Рисунок 15 – Вывод программы

Код для проверки распределений признаков на нормальность представлен на рисунке 16, результат его работы – на рисунке 17, построенные диаграммы – на рисунках 18-19.

```
insurance_df = pd.read_csv('data/insurance.csv')

for column in ['bmi', 'charges']:
    print(f"\n--- Проверка для признака '{column}' ---")
    data = insurance_df[column]

    print("Гипотезы:")
    print(" - H0 (Нулевая гипотеза): Распределение признака не отличается от нормального.")
    print(" - H1 (Альтернативная гипотеза): Распределение признака отличается от нормального.")

    data_standardized = (data - np.mean(data)) / np.std(data)
    ks_statistic, ks_pvalue = st.kstest(data_standardized, cdf='norm')

    print(f"\nРезультаты KS-теста:")
    print(f" - Статистика: {ks_statistic:.4f}")
    print(f" - p-value: {ks_pvalue:.4f}")

    alpha = 0.05
    if ks_pvalue < alpha:
        print(f" - Вывод: p-value ({ks_pvalue:.4f}) < {alpha}. Отвергаем H0. "
              "Распределение не является нормальным.")
    else:
        print(f" - Вывод: p-value ({ks_pvalue:.4f}) >= {alpha}. Не можем отвергнуть H0. "
              "Распределение может быть нормальным.")

    plt.figure(figsize=(8, 6))
    st.probplot(data, dist="norm", plot=plt)
    ax = plt.gca()
    ax.get_lines()[0].set_markersize(2)

    plt.title(f'Q-Q plot для "{column}"')
    plt.grid(visible=True, alpha=0.3)
    plt.show()
```

Рисунок 16 – Код для проверки распределений на нормальность

```

O8_normality_tests x
C:\Users\Remsely\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Remsely\PycharmProje

--- Проверка для признака 'bmi' ---
Гипотезы:
- H0 (Нулевая гипотеза): Распределение признака не отличается от нормального.
- H1 (Альтернативная гипотеза): Распределение признака отличается от нормального.

Результаты KS-теста:
- Статистика: 0.0261
- p-value: 0.3145
- Вывод: p-value (0.3145) >= 0.05. Не можем отвергнуть H0. Распределение может быть нормальным.

--- Проверка для признака 'charges' ---
Гипотезы:
- H0 (Нулевая гипотеза): Распределение признака не отличается от нормального.
- H1 (Альтернативная гипотеза): Распределение признака отличается от нормального.

Результаты KS-теста:
- Статистика: 0.1885
- p-value: 0.0000
- Вывод: p-value (0.0000) < 0.05. Отвергаем H0. Распределение не является нормальным.

Process finished with exit code 0

```

Рисунок 17 – Вывод программы

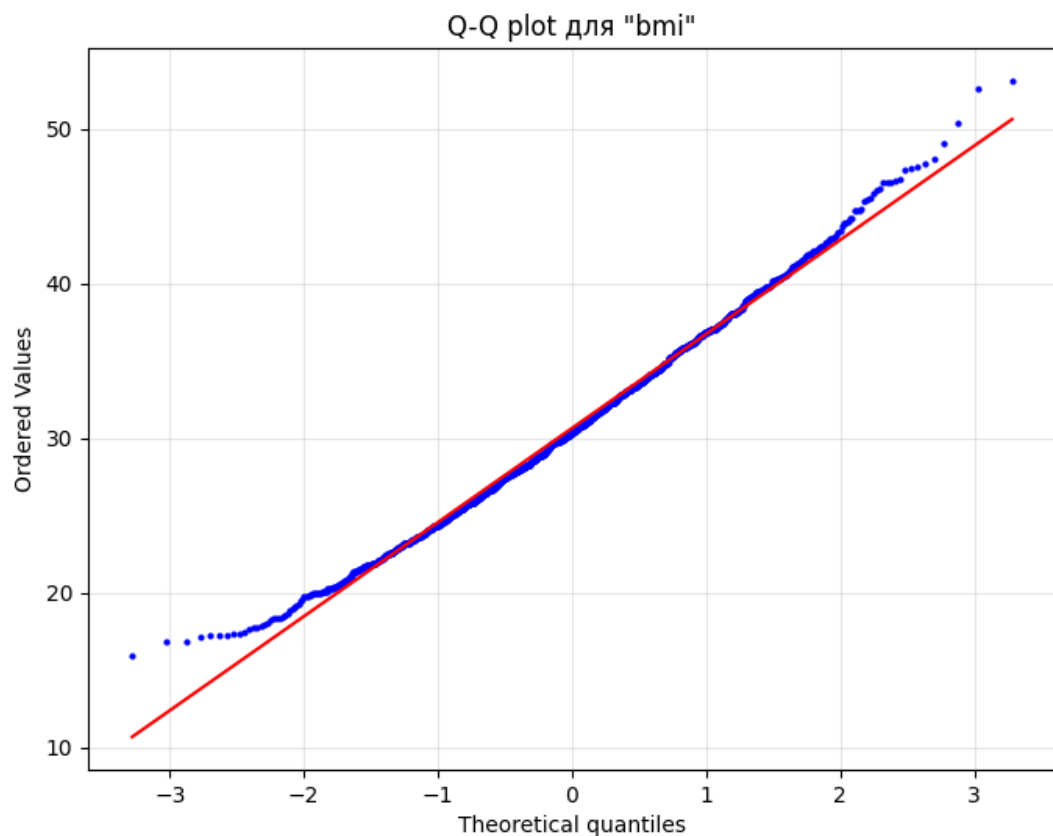


Рисунок 18 – Q-Q plot для bmi



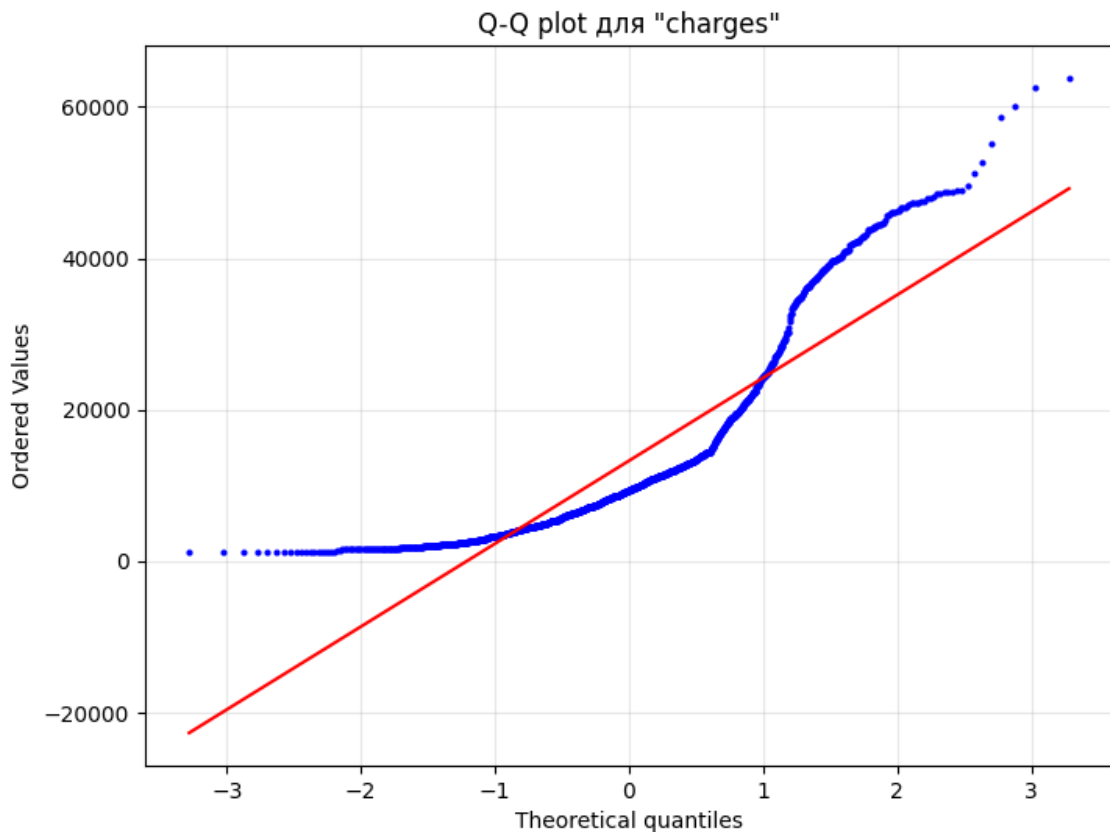


Рисунок 19 – Q-Q plot для charges

bmi: Распределение ИМТ соответствует нормальному закону. Небольшие отклонения на краях Q-Q plot (левый и правый концы) указывают на немного более тяжелые хвосты по сравнению с идеальным нормальным распределением, но эти отклонения не являются критичными

charges: Распределение расходов значительно отличается от нормального. Нарушение нормальности объясняется природой данных: страховые расходы не могут быть отрицательными и имеют редкие, но очень высокие значения.

### **ECDC Cases (задания 9-12)**

Код для удаления пропущенных значений представлен на рисунке 20, результат его работы – на рисунке 21.

```

import pandas as pd

covid_df = pd.read_csv('data/ECDCCases.csv')

missing_percentage = (covid_df.isnull().sum() / len(covid_df)) * 100
print("Процент пропущенных значений по столбцам:")
print(missing_percentage.sort_values(ascending=False))

cols_to_drop = missing_percentage.sort_values(ascending=False).head(2).index.tolist()
covid_df.drop(columns=cols_to_drop, inplace=True)
print(f"\nУдалены столбцы с наибольшим количеством пропусков: {cols_to_drop}")

for col in covid_df.columns:
    if covid_df[col].isnull().sum() > 0:
        if covid_df[col].dtype != 'object':
            median_val = covid_df[col].median()
            covid_df.loc[covid_df[col].isnull(), col] = median_val
            print(f"Пропуски в числовом столбце '{col}' заменены медианой ({median_val}).")
        else:
            covid_df.loc[covid_df[col].isnull(), col] = 'other'
            print(f"Пропуски в категориальном столбце '{col}' заменены на 'other'.")

print("\nПроверка на наличие пропусков после обработки:")
print(covid_df.isnull().sum())

covid_df.to_csv(path_or_buf='data/ECDCCases_cleaned_missing_values.csv', index=False)

```

Рисунок 20 – Код для удаления пропущенных значений

```

C:\Users\Remsely\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Remsely\PycharmProjects\mirea-big-data-analysi
Процент пропущенных значений по столбцам:
Cumulative_number_for_14_days_of_COVID-19_cases_per_100000    4.650750
geoId    0.444236
popData2019    0.198695
countryterritoryCode    0.198695
year    0.000000
month    0.000000
day    0.000000
dateRep    0.000000
countriesAndTerritories    0.000000
deaths    0.000000
cases    0.000000
continentExp    0.000000
dtype: float64

Удалены столбцы с наибольшим количеством пропусков: ['Cumulative_number_for_14_days_of_COVID-19_cases_per_100000', 'geoId']
Пропуски в категориальном столбце 'countryterritoryCode' заменены на 'other'.
Пропуски в числовом столбце 'popData2019' заменены медианой (7169456.0).

Проверка на наличие пропусков после обработки:
dateRep    0
day    0
month    0
year    0
cases    0
deaths    0
countriesAndTerritories    0
countryterritoryCode    0
popData2019    0
continentExp    0
dtype: int64

```

Рисунок 21 – Результат удаления пропущенных значений

Код для нахождения выбросов (задание 11) представлен на рисунке 22, результат его работы – на рисунке 23.

```
import pandas as pd

covid_df = pd.read_csv('data/ECDCCases_cleaned_missing_values.csv')

pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

print("Статистика по данным (describe()):")
print(covid_df.describe())

high_deaths_days = covid_df[covid_df['deaths'] > 3000]

print(f"\nНайдено {len(high_deaths_days)} дней, когда количество смертей превысило 3000.")
print(high_deaths_days[['dateRep', 'countriesAndTerritories', 'cases', 'deaths']])
```

Рисунок 22 – Код для нахождения выбросов

```
11_outlier_alalysis x
```

```
C:\Users\Remsely\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Remsely\Pychar
Статистика по данным (describe()):
```

	day	month	year	cases	deaths	popData2019
count	61904.000000	61904.000000	61904.000000	61904.000000	61904.000000	6.190400e+04
mean	15.629232	7.067104	2019.998918	1155.079026	26.053987	4.091909e+07
std	8.841624	2.954816	0.032881	6779.010824	131.222948	1.529798e+08
min	1.000000	1.000000	2019.000000	-8261.000000	-1918.000000	8.150000e+02
25%	8.000000	5.000000	2020.000000	0.000000	0.000000	1.324820e+06
50%	15.000000	7.000000	2020.000000	15.000000	0.000000	7.169456e+06
75%	23.000000	10.000000	2020.000000	273.000000	4.000000	2.851583e+07
max	31.000000	12.000000	2020.000000	234633.000000	4928.000000	1.433784e+09

```
Найдено 11 дней, когда количество смертей превысило 3000.
```

	dateRep	countriesAndTerritories	cases	deaths
2118	02/10/2020	Argentina	14001	3351
16908	07/09/2020	Ecuador	-8261	3800
37038	09/10/2020	Mexico	4936	3013
44888	14/08/2020	Peru	9441	3935
44909	24/07/2020	Peru	4546	3887
59007	12/12/2020	United_States_of_America	234633	3343
59009	10/12/2020	United_States_of_America	220025	3124
59016	03/12/2020	United_States_of_America	203311	3190
59239	24/04/2020	United_States_of_America	26543	3179
59245	18/04/2020	United_States_of_America	30833	3770
59247	16/04/2020	United_States_of_America	30148	4928

```
Process finished with exit code 0
```

Рисунок 23 – Сравнение времени работы t-SNE и UMAP

## Выводы:

- deaths – Минимальное значение -1918 является явной ошибкой данных (отрицательное количество смертей невозможно), требует исправления. Максимум 4928 смертей в день и стандартное отклонение 131.22 при среднем 26.05 указывают на экстремальные выбросы;
- cases – Аналогично содержит отрицательные значения (вероятно, корректировки данных);
- popData2019 – Огромный разброс — от 815 человек (микрогосударства) до 1.43 млрд (Китай, Индия), что естественно для страновых данных.

Код для удаления дублирующихся данных представлен на рисунке 24, результат его работы – на рисунке 25.

```
import pandas as pd

covid_df = pd.read_csv('data/ECDCCases_cleaned_missing_values.csv')

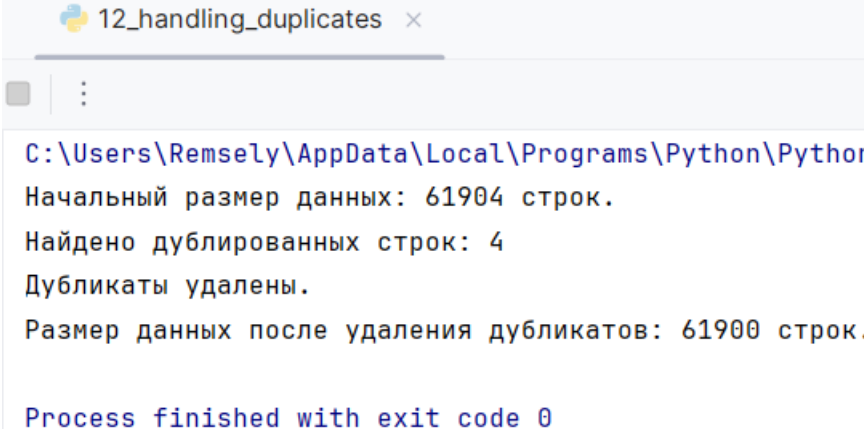
print(f"Начальный размер данных: {covid_df.shape[0]} строк.")

num_duplicates = covid_df.duplicated().sum()
print(f"Найдено дублированных строк: {num_duplicates}")

if num_duplicates > 0:
    covid_df.drop_duplicates(inplace=True)
    print("Дубликаты удалены.")
    print(f"Размер данных после удаления дубликатов: {covid_df.shape[0]} строк.")

covid_df.to_csv(path_or_buf='data/ECDCCases_cleaned_duplicates.csv', index=False)
```

Рисунок 24 – Код для удаления дублей



```
12_handling_duplicates x
C:\Users\Remsely\AppData\Local\Programs\Python\Python
Начальный размер данных: 61904 строк.
Найдено дублированных строк: 4
Дубликаты удалены.
Размер данных после удаления дубликатов: 61900 строк.

Process finished with exit code 0
```

Рисунок 25 – Результат удаления дублей

## Тесты (задания 13-15)

Код для выполнения задания 13 представлен на рисунке 26, его вывод — на рисунке 27.

```
bmi_northwest = bmi_df[bmi_df['region'] == 'northwest']['bmi']
bmi_southwest = bmi_df[bmi_df['region'] == 'southwest']['bmi']

print(f"Размер выборки для Northwest: {len(bmi_northwest)}")
print(f"Размер выборки для Southwest: {len(bmi_southwest)}")

print("\n1. Проверка на нормальность (H0: распределение нормальное)")
shapiro_nw = st.shapiro(bmi_northwest)
shapiro_sw = st.shapiro(bmi_southwest)
print(f" - Northwest: p-value = {shapiro_nw.pvalue:.4f}")
print(f" - Southwest: p-value = {shapiro_sw.pvalue:.4f}")
if shapiro_nw.pvalue > 0.05 and shapiro_sw.pvalue > 0.05:
    print(" -> Обе выборки могут быть из нормального распределения (p > 0.05).")
else:
    print(" -> Одна или обе выборки не распределены нормально. t-тест может быть неточным.")

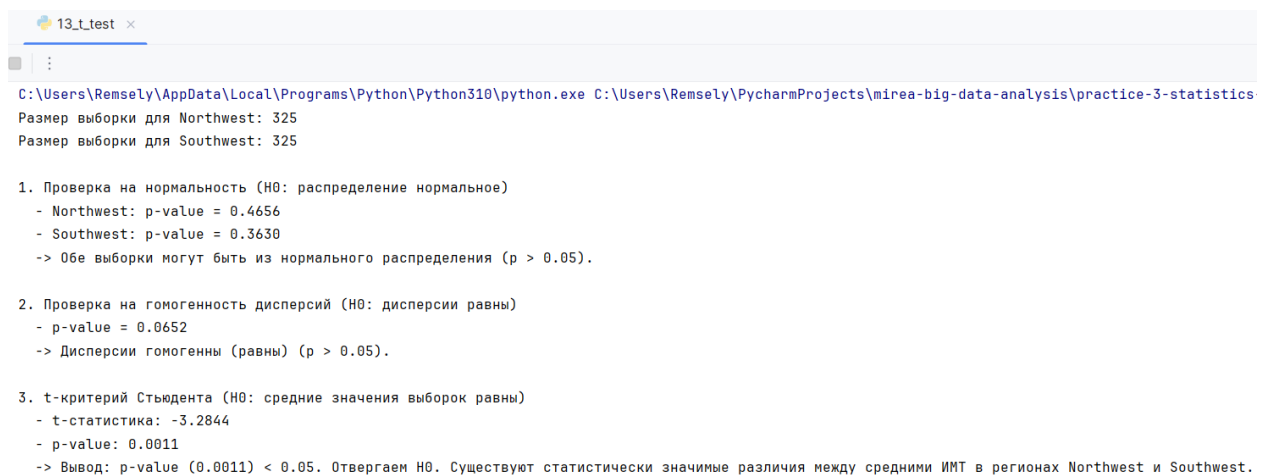
print("\n2. Проверка на гомогенность дисперсий (H0: дисперсии равны)")
bartlett_test = st.bartlett(*samples: bmi_northwest, bmi_southwest)
print(f" - p-value = {bartlett_test.pvalue:.4f}")
if bartlett_test.pvalue > 0.05:
    print(" -> Дисперсии гомогенны (равны) (p > 0.05).")
    equal_var_param = True
else:
    print(" -> Дисперсии не гомогенны (не равны). Используем поправку Уэлча.")
    equal_var_param = False

print("\n3. t-критерий Стьюдента (H0: средние значения выборок равны)")
t_stat, p_value = st.ttest_ind(bmi_northwest, bmi_southwest, equal_var=equal_var_param)

print(f" - t-статистика: {t_stat:.4f}")
print(f" - p-value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print(f" -> Вывод: p-value ({p_value:.4f}) < {alpha}. "
          f"Отвергаем H0. Существуют статистически значимые различия между средними ИМТ в регионах Northwest и Southwest.")
```

Рисунок 26 – Код для выполнения задания 13



```
13_t_test x
C:\Users\Remsel\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Remsel\PycharmProjects\mirea-big-data-analysis\practice-3-statistics
Размер выборки для Northwest: 325
Размер выборки для Southwest: 325

1. Проверка на нормальность (H0: распределение нормальное)
- Northwest: p-value = 0.4656
- Southwest: p-value = 0.3630
-> Обе выборки могут быть из нормального распределения (p > 0.05).

2. Проверка на гомогенность дисперсий (H0: дисперсии равны)
- p-value = 0.0652
-> Дисперсии гомогенны (равны) (p > 0.05).

3. t-критерий Стьюдента (H0: средние значения выборок равны)
- t-статистика: -3.2844
- p-value: 0.0011
-> Вывод: p-value (0.0011) < 0.05. Отвергаем H0. Существуют статистически значимые различия между средними ИМТ в регионах Northwest и Southwest.
```

Рисунок 27 – Результаты проверок

Код для выполнения задания 14 представлен на рисунке 28, его вывод — на рисунке 29.

```

import scipy.stats as st

observed_frequencies = [97, 98, 109, 95, 97, 104]
total_throws = sum(observed_frequencies)

expected_frequencies = [total_throws / 6] * 6

print(f"Наблюдаемые частоты: {observed_frequencies}")
print(f"Ожидаемые частоты: {expected_frequencies}")

print("H0: Наблюдаемое распределение соответствует равномерному (кубик 'честный').")
print("H1: Наблюдаемое распределение не соответствует равномерному.")

chi2_stat, p_value = st.chisquare(f_obs=observed_frequencies, f_exp=expected_frequencies)

print(f"\nРезультаты теста:")
print(f" - Хи-квадрат статистика: {chi2_stat:.4f}")
print(f" - p-value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print(f" -> Вывод: p-value ({p_value:.4f}) < {alpha}. Отвергаем H0. Распределение не является равномерным.")
else:
    print(f" -> Вывод: p-value ({p_value:.4f}) >= {alpha}. Не можем отвергнуть H0. Нет оснований считать кубик 'нечестным'.")

```

Рисунок 28 – Код для выполнения задания 14

```

C:\Users\Remsely\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Remsely\PycharmProjects\
Наблюдаемые частоты: [97, 98, 109, 95, 97, 104]
Ожидаемые частоты: [100.0, 100.0, 100.0, 100.0, 100.0, 100.0]
H0: Наблюдаемое распределение соответствует равномерному (кубик 'честный').
H1: Наблюдаемое распределение не соответствует равномерному.

Результаты теста:
- Хи-квадрат статистика: 1.4400
- p-value: 0.9199
-> Вывод: p-value (0.9199) >= 0.05. Не можем отвергнуть H0. Нет оснований считать кубик 'нечестным'.

Process finished with exit code 0

```

Рисунок 29 – Результаты проверок

Код для выполнения задания 15 представлен на рисунке 30, его вывод – на рисунке 31.

```

data = pd.DataFrame({
    'Женат': [89, 17, 11, 43, 22, 1],
    'Гражданский брак': [80, 22, 20, 35, 6, 4],
    'Не состоит в отношениях': [35, 44, 35, 6, 8, 22]
})
data.index = [
    'Полный рабочий день', 'Частичная занятость', 'Временно не работает',
    'На домохозяйстве', 'На пенсии', 'Учёба'
]

print("Таблица сопряженности (Наблюдаемые частоты):")
print(data)

print("\nH0: Семейное положение и занятость являются независимыми переменными.")
print("H1: Существует зависимость между семейным положением и занятостью.")

chi2, p, dof, expected = st.chi2_contingency(data)

print("\nРезультаты теста:")
print(f" - Хи-квадрат статистика: {chi2:.4f}")
print(f" - p-value: {p:.4f}")
print(f" - Степени свободы: {dof}")

alpha = 0.05

if p < alpha:
    print(
        f" -> Вывод: p-value ({p:.4f}) < {alpha}. "
        f"Отвергаем H0. Существует статистически значимая связь между семейным положением и занятостью.")
else:
    print(f" -> Вывод: p-value ({p:.4f}) >= {alpha}. "
        f"Не можем отвергнуть H0. Нет оснований утверждать о наличии связи между переменными.")

```

Рисунок 30 – Код для выполнения задания 15

15\_variables\_dependencies

C:\Users\Remsely\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Remsely\PycharmProjects\mirea-big-data-analysis\pract

Таблица сопряженности (Наблюдаемые частоты):

	Женат	Гражданский брак	Не состоит в отношениях
Полный рабочий день	89	80	35
Частичная занятость	17	22	44
Временно не работает	11	20	35
На домохозяйстве	43	35	6
На пенсии	22	6	8
Учёба	1	4	22

H0: Семейное положение и занятость являются независимыми переменными.  
H1: Существует зависимость между семейным положением и занятостью.

Результаты теста:

- Хи-квадрат статистика: 122.2965
- p-value: 0.0000
- Степени свободы: 10

-> Вывод: p-value (0.0000) < 0.05. Отвергаем H0. Существует статистически значимая связь между семейным положением и занятостью.

Process finished with exit code 0

Рисунок 31 – Результаты проверок

## **ВЫВОД**

В ходе выполненной практической работы проведено ознакомление с инструментами работы со статистикой в Python.