МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра теории вероятностей и математической статистики

ОТЧЕТ по лабораторной работе №1

Константинова Николая Андреевича
студента 3 курса, 8 группы

Преподаватель
Полузёров Тимофей Дмитриевич

Минск, 2025

Содержание

Часть 1. Расчёт описательных статистик

Формулировка задания:

Рассчитать основные описательные статистики для всех числовых переменных:

- Минимум, максимум, размах
- Среднее значение
- Дисперсия и стандартное отклонение
- Медиана
- Квантили уровней 0.01, 0.05, 0.95, 0.99

Сначала выбираем только числовые столбцы для анализа:

```
numeric_data = my_data.select_dtypes(include=[np.number])
```

Все статистики будем подсчитывать для каждого столбца отдельно.

Для расчёта описательных статистик из условия будем пользоваться методами из модуля pandas:

```
stats_table['Минимум'] = numeric_data.min()
stats_table['Максимум'] = numeric_data.max()
stats_table['Размах'] = stats_table['Максимум'] - stats_table['Минимум']

stats_table['Среднее'] = numeric_data.mean()

stats_table['Дисперсия'] = numeric_data.var()
stats_table['Ст. откл.'] = numeric_data.std()

stats_table['Медиана'] = numeric_data.median()

stats_table['Q0.01'] = numeric_data.quantile(0.01)
stats_table['Q0.05'] = numeric_data.quantile(0.05)
stats_table['Q0.95'] = numeric_data.quantile(0.95)
stats_table['Q0.99'] = numeric_data.quantile(0.99)
```

Итоговая таблица будет выглядеть так:

```
РАСЧЕТ ОПИСАТЕЛЬНЫХ СТАТИСТИК
==================================================
Таблица описательных статистик:
                      Минимум    Максимум      Размах    Среднее    Дисперсия  Ст. откл.  Медиана    Q0.01    Q0.05     Q0.95     Q0.99
Среднеспис.числ.работн  10.000   28650.000   28640.000  1220.773  6535214.100  2556.406  473.000   36.940   91.000  5152.400  12526.780
k1                       0.248      18.020      17.772     2.002        2.856     1.690    1.474    0.423    0.664     5.258     9.003
k2                       0.000       7.029       7.029     0.238        0.268     0.518    0.056    0.000    0.003     1.129     2.637
k3                       0.008      11.188      11.179     0.825        0.843     0.918    0.538    0.061    0.125     2.545     4.917
k4                      -4.570       0.936       5.506     0.038        0.391     0.625    0.149   -2.450   -0.977     0.756     0.882
k4_new                  -2.599       0.945       3.544     0.336        0.130     0.360    0.367   -0.874   -0.226     0.821     0.896
k5                       0.010       1.084       1.074     0.346        0.039     0.198    0.320    0.037    0.068     0.713     0.860
k6                       0.000       1.000       1.000     0.238        0.045     0.213    0.185    0.000    0.000     0.664     0.860
k7                       0.000       1.000       1.000     0.174        0.050     0.223    0.076    0.000    0.000     0.674     0.890
k8                       0.054       0.990       0.936     0.656        0.038     0.196    0.682    0.161    0.293     0.933     0.962
k9                       0.059     199.606     199.547     6.863       63.095     7.943    4.759    0.456    1.049    19.642    35.671
k10                      0.040      11.834      11.794     0.831        0.643     0.802    0.618    0.088    0.167     2.208     4.030
k11                      0.099      15.865      15.766     1.326        0.289     0.538    1.237    0.579    0.805     2.051     2.649
k12                      0.230      10.809      10.579     1.351        0.356     0.596    1.180    0.722    0.931     2.453     3.685
k13                      0.033       4.983       4.950     1.201        0.490     0.700    1.106    0.135    0.269     2.520     3.209
k14                      0.220      68.830      68.610     9.944       58.310     7.636    8.170    0.800    1.577    24.147    36.195
k15                      0.347     195.042     194.694    13.505      179.138    13.384    9.851    1.429    3.016    34.983    67.004
k16                      0.000      31.560      31.560     2.150        7.937     2.817    1.236    0.165    0.343     6.969    15.233
k17                      0.448  104151.000  104150.552   201.286  5109675.780  2260.459   20.099    2.235    4.059   288.018  4299.428
k18                     -0.975       0.861       1.836     0.055        0.012     0.111    0.059   -0.407   -0.093     0.185     0.277
k19                     -0.321       0.635       0.956     0.079        0.009     0.094    0.056   -0.095   -0.029     0.255     0.390
k20                     -2.671       0.837       3.507     0.065        0.022     0.147    0.044   -0.328   -0.067     0.274     0.433
Year                     5.000      11.000       6.000     8.000        4.001     2.000    8.000    5.000    5.000    11.000    11.000
```

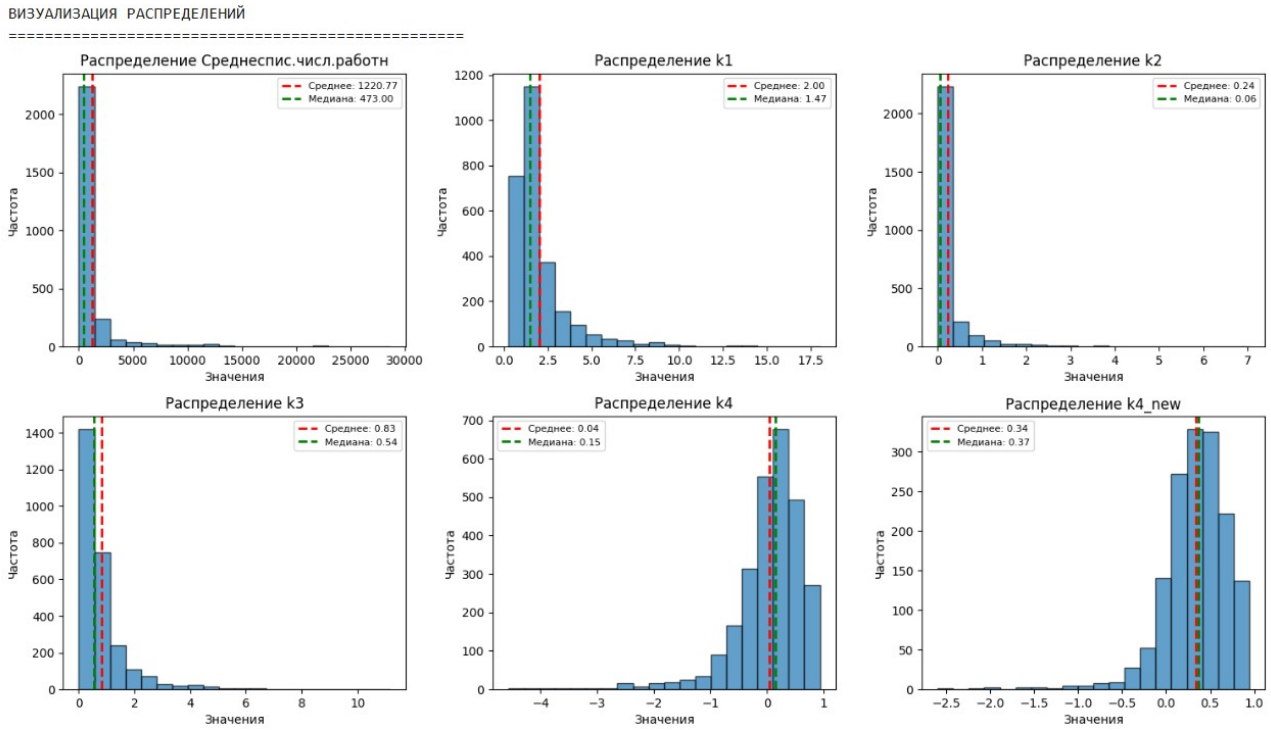
Часть 2. Визуализация распределений

Формулировка задания:

Построить гистограммы распределения для каждой числовой переменной и проанализировать форму распределений.

```python
#гистограммы для каждого столбца
for i, column in enumerate(numeric_data.columns):
    if i < len(axes):
        data = numeric_data[column].dropna()

        #гистограмма
        n, bins, patches = axes[i].hist(data, bins=20, alpha=0.7, edgecolor='black')

        axes[i].set_title(f'Распределение {column}')
        axes[i].set_xlabel('Значения')
        axes[i].set_ylabel('Частота')
        #линии для медианы и среднего
        mean_val = data.mean()
        median_val = data.median()
        axes[i].axvline(mean_val, color='red', linestyle='--', linewidth=2,
                        label=f'Среднее: {mean_val:.2f}')
        axes[i].axvline(median_val, color='green', linestyle='--', linewidth=2,
                        label=f'Медиана: {median_val:.2f}')
        axes[i].legend(fontsize=8)
```

Скриншот результата (часть):



Выводы:

Встречаются распределения величин, напоминающие нормальное (k18, k19, k20). Графики распределения многих СВ убывают (Среднеспис. Числ. Работн., k2, k3, k6 и др.). Также все значения СВ year встречаются с одинаковой частотой.

Часть 3. Корреляционный анализ

Формулировка задания:

* Вычислить парные корреляции Пирсона. Проверить значимость корреляций.

Результаты представить в **виде таблицы**

* Для сильно коррелирующих пар построить график рассеивания (scater plot).

* Сделать выводы о наличии корреляций.

```
corr_matrix = numeric_data.corr()
```

Вывод по 10 столбцов, чтобы значения не перекрывали друг друга:

было

```
КОРРЕЛЯЦИОННЫЙ АНАЛИЗ
==========================================================
МАТРИЦА КОРРЕЛЯЦИЙ ПИРСОНА:
==========================================================
                        Среднеспис.числ.работн     k1     k2     k3     k4  k4_new     k5
k13    k14    k15    k16    k17    k18    k19    k20   Year
Среднеспис.числ.работн                  1.000  0.035  0.040  0.058  0.009   0.087 -0.045
-0.062  0.002 -0.067 -0.087  0.056  0.096  0.019  0.017 -0.002
k1                                      0.035  1.000  0.733  0.883  0.453   0.658 -0.412
0.048  0.541  0.023 -0.256  0.004  0.227  0.314  0.187  0.114
k2                                      0.040  0.733  1.000  0.813  0.328   0.437 -0.318
0.041  0.374  0.093 -0.130  0.013  0.203  0.318  0.186  0.078
k3                                      0.058  0.883  0.813  1.000  0.398   0.582 -0.356
0.078  0.493 -0.084 -0.304  0.017  0.237  0.345  0.213  0.117
k4                                      0.009  0.453  0.328  0.398  1.000   0.639 -0.557
0.279  0.339  0.065 -0.264  0.016  0.415  0.442  0.367  0.011
k4_new                                  0.087  0.658  0.437  0.582  0.639   1.000 -0.472
0.095  0.374 -0.031 -0.353  0.026  0.301  0.403  0.344  0.025
k5                                     -0.045 -0.412 -0.318 -0.356 -0.557  -0.472  1.000
0.026 -0.319 -0.234  0.055 -0.047 -0.179 -0.245 -0.152  0.040
k6                                     -0.149 -0.114 -0.124 -0.111 -0.090  -0.101  0.037
-0.131 -0.154 -0.073  0.075 -0.015 -0.243 -0.226 -0.219 -0.117
k7                                     -0.064 -0.248 -0.208 -0.227 -0.263  -0.260  0.166
-0.246 -0.325 -0.142  0.152  0.007 -0.293 -0.330 -0.280 -0.084
```

стало

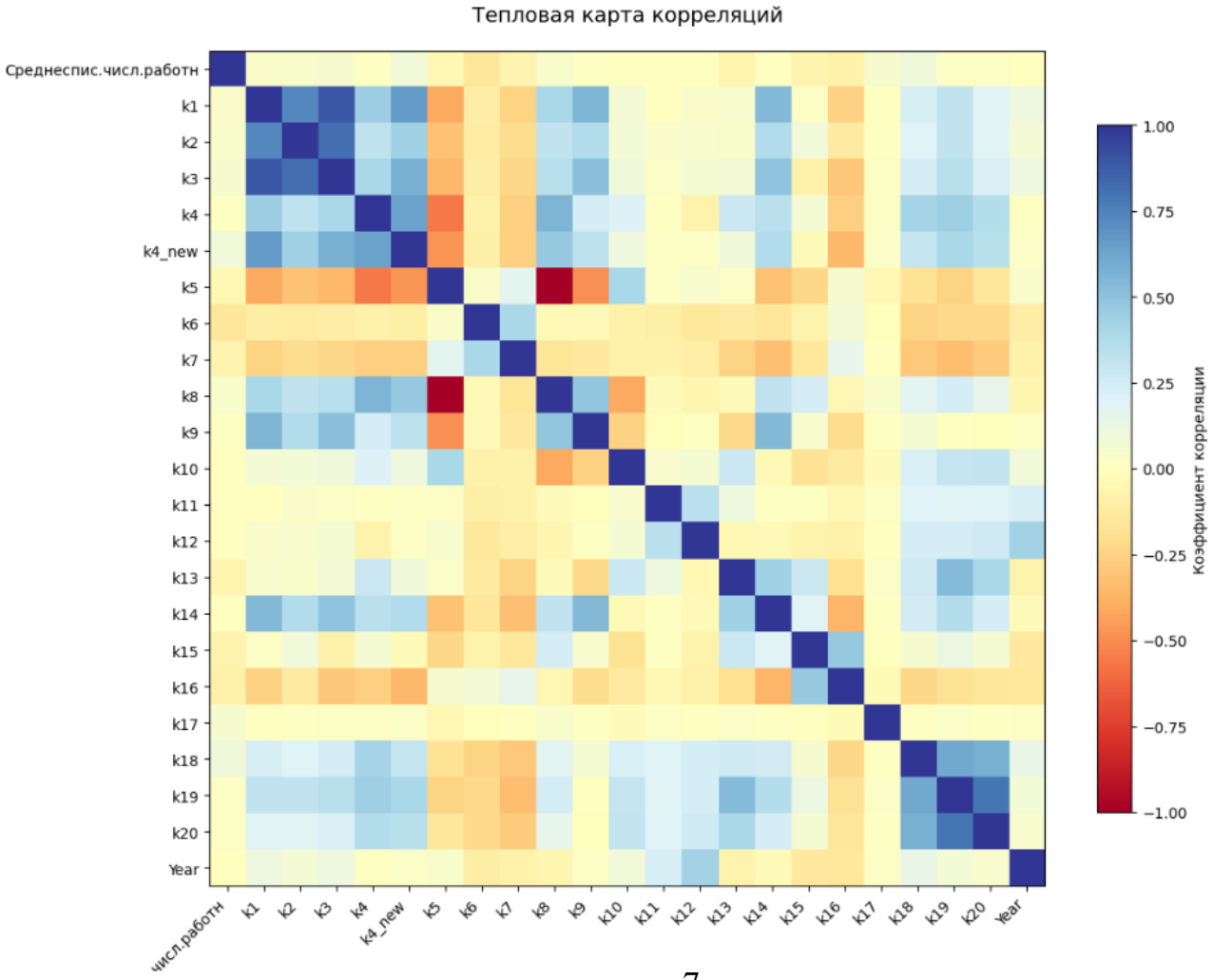
```
Среднеспис.числ.работн   1.000  0.035  0.040  0.058
k1                       0.035  1.000  0.733  0.883
k2                       0.040  0.733  1.000  0.813
k3                       0.058  0.883  0.813  1.000
k4                       0.009  0.453  0.328  0.398
k4_new                   0.087  0.658  0.437  0.582
k5                      -0.045 -0.412 -0.318 -0.356
k6                      -0.149 -0.114 -0.124 -0.111
k7                      -0.064 -0.248 -0.208 -0.227
k8                       0.044  0.406  0.317  0.351
k9                       0.014  0.558  0.368  0.510
k10                      0.007  0.071  0.079  0.096
k11                     -0.002 -0.003  0.033  0.028
k12                      0.005  0.039  0.051  0.068
k13                     -0.062  0.048  0.041  0.078
```

Код для вывода по 10 столбцов:

```python
for start_idx in range(0, n_cols, chunk_size):
    end_idx = min(start_idx + chunk_size, n_cols)
    current_cols = cols[start_idx:end_idx]

    print("-" * 80)

    corr_chunk = corr_matrix[current_cols]
    print(corr_chunk.round(3))
    print()
```

Корреляцию будем считать значимой, если P-value < 0,05

```
significant_pairs = []
for i in range(len(corr_matrix.columns)):
    for j in range(i+1, len(corr_matrix.columns)):
        col1 = corr_matrix.columns[i]
        col2 = corr_matrix.columns[j]
        r = corr_matrix.iloc[i, j]

        pair_data = numeric_data[[col1, col2]].dropna()
        n = len(pair_data)
        if n > 2:
            p_value = calculate_p_value(r, n)
            if p_value < 0.05:
                significant_pairs.append((col1, col2, r, p_value))

#сортируем
significant_pairs.sort(key=lambda x: abs(x[2]), reverse=True)
```



Часть 4. Цензурирование данных

Код для построения ящичных диаграмм:

```
plt.figure(figsize=(15, 8))
numeric_data.boxplot()
plt.xticks(rotation=45)
plt.title('Ящичные диаграммы для выявления аномальных наблюдений')
plt.tight_layout()
plt.show()
```

Для выявления экстремальных наблюдений используем тест Хампеля:

```
def hampel_test(data, k=5.2):
    med = np.median(data)
    mad = np.median(np.abs(data - med))
    lower_bound = med - k * mad
    upper_bound = med + k * mad
    outliers = data[(data < lower_bound) | (data > upper_bound)]
    return outliers, lower_bound, upper_bound, len(outliers)/len(data)*100
```

И правило 6 сигм:

```
def six_sigma_test(data):
    mean = np.mean(data)
    std = np.std(data)
    lower_bound = mean - 6 * std
    upper_bound = mean + 6 * std
    outliers = data[(data < lower_bound) | (data > upper_bound)]
    return outliers, lower_bound, upper_bound, len(outliers)/len(data)*100
```

Найдем множитель, удовлетворяющий условию (не более 5-8% данных экстремальные):

```python
def find_optimal_k(data, target_pct_min=5, target_pct_max=8):
    k_values = np.arange(1.0, 10.0, 0.1)
    optimal_k = 5.2
    for k in k_values:
        _, _, _, pct = hampel_test(data, k)
        if target_pct_min <= pct <= target_pct_max:
            optimal_k = k
            break
    return optimal_k
```

```
2. ВЫЯВЛЕНИЕ ЭКСТРЕМАЛЬНЫХ НАБЛЮДЕНИЙ
--------------------------------------------------
ПЕРЕМЕННАЯ                МЕТОД        ВЫБРОСЫ  ДОЛЯ,%   ГРАНИЦЫ
-------------------------------------------------------------------------------
Среднеспис.числ.работн    Хампель      213      7.90     [-1959.00, 2905.00]
                          6 сигм       14       0.52     [-14114.82, 16556.37]
k1                        Хампель      214      7.94     [-1.29, 4.24]
                          6 сигм       11       0.41     [-8.14, 12.14]
k2                        Хампель      509      18.89    [-0.20, 0.31]
                          6 сигм       13       0.48     [-2.87, 3.34]
k3                        Хампель      214      7.94     [-0.90, 1.97]
                          6 сигм       9        0.33     [-4.68, 6.33]
k4                        Хампель      208      7.72     [-0.75, 1.05]
                          6 сигм       7        0.26     [-3.71, 3.79]
k4_new                    Хампель      122      7.92     [-0.16, 0.89]
                          6 сигм       4        0.26     [-1.82, 2.50]
k5                        Хампель      193      7.16     [-0.03, 0.67]
                          6 сигм       0        0.00     [-0.84, 1.53]
k6                        Хампель      205      7.61     [-0.23, 0.60]
                          6 сигм       0        0.00     [-1.04, 1.52]
k7                        Хампель      214      7.94     [-0.41, 0.56]
                          6 сигм       0        0.00     [-1.17, 1.51]
```

Цензурирование:

```python
def censor_data(data, lower_bound, upper_bound):
    censored = data.copy()
    censored[censored < lower_bound] = lower_bound
    censored[censored > upper_bound] = upper_bound
    return censored
```

Нормировка:

```python
def normalize_data(data, direct_relation=True):
    min_val = data.min()
    max_val = data.max()
    if direct_relation:
        normalized = (data - min_val) / (max_val - min_val)
    else:
        normalized = (max_val - data) / (max_val - min_val)
    return normalized
```