**Лабораторная работа №1**
**ЧАСТЬ 2**

**Цель работы:** изучить библиотеку matplotlib языка программирования python на примере Pie Charts, Box Plots, Scatter Plots, Bubble Plots
.

## Теоретическая часть:

### 1. Предисловие

Все, что тебе нужно знать об избирательных участках Цели обучения
К концу этой лабораторной работы ты сможешь:
Определить наилучший тип графика для данного набора данных и сценария.
Объяснить практику проектирования определенных участков
Дизайн выдающийся, ощутимые визуализации
В этой главе мы познакомимся с основами различных типов графиков.

### 2. Введение

В этой лабораторной работе мы сосредоточимся на различных визуализациях и определим, какая визуализация лучше всего подходит для отображения определенной информации для данного набора данных. Мы подробно опишем каждую рассматриваемую визуализацию и приведем практические примеры, такие как сравнение различных запасов с течением времени или сравнение рейтингов для разных фильмов. Начиная со сравнительных графиков, которые отлично подходят для сравнения нескольких переменных с течением времени, мы рассмотрим их типы, такие как линейные, столбчатые и радиолокационные графики. Графики связей удобны для отображения связей между переменными.

### 3. Bubble Plot

График пузырькового рассеяния расширяет график рассеяния, вводя третью числовую переменную. Значение переменной представлено размером точек. Площадь точек пропорциональна значению. Легенда используется для того, чтобы связать размер точки с действительным числовым значением.
**Uses**:
Показать зависимость между тремя переменными.
**Example**:
На следующей диаграмме показан сюжет с пузырьками, который подчеркивает связь между площадью и параметрами X Y:

```
# libraries
import matplotlib.pyplot as plt
import numpy as np

# create data
x = np.random.rand(5)
y = np.random.rand(5)
```

```
z = np.random.rand(5)

# pimp your plot with the seaborn style
import seaborn as sns
plt.scatter(x, y, s=z*4000, c="green", alpha=0.4, linewidth=6)

# Add titles (main and on axis)
plt.xlabel("the X axis")
plt.ylabel("the Y axis")
plt.title("A bubble plot", loc="left")
```

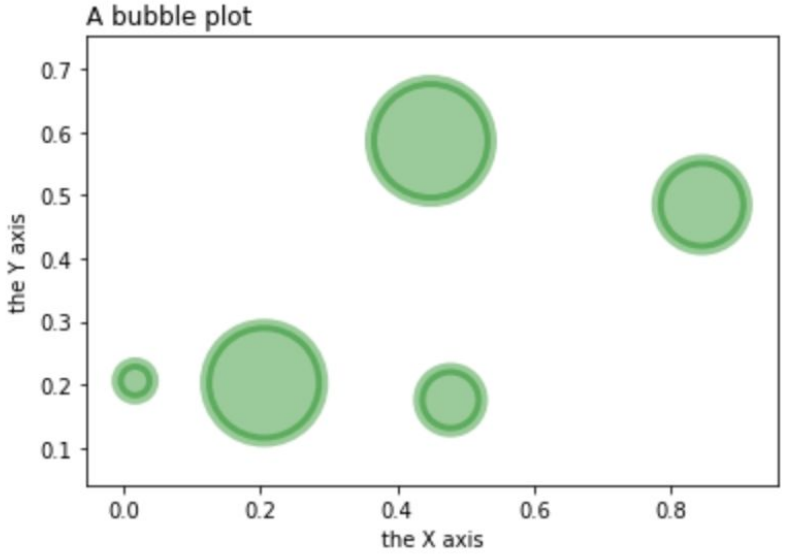


**Figure 2.15: Bubble plot , показывающий связь между X Y**

Практика проектирования:
Практика проектирования для графика рассеяния также применима к графику пузырьков.
Не используйте его для очень больших объемов данных, так как слишком большое количество пузырьков затрудняет чтение графика.

## 4. Scatter Plot

Графики разброса показывают точки данных для двух числовых переменных, отображая переменную на обеих осях

**Uses**:

- Можно определить, существует ли корреляция (связь) между двумя переменными.
- Они позволяют строить отношения для нескольких групп или категорий, используя различные цвета.

- График пузырьков, представляющий собой вариацию графика рассеяния, является отличным инструментом для визуализации корреляции третьей переменной.

**Examples**:

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np

x = np.linspace(0, 10, 30)
y = np.sin(x)

plt.plot(x, y, 'o', color='black');
```

На следующей диаграмме показан график рассеяния синусоиды:

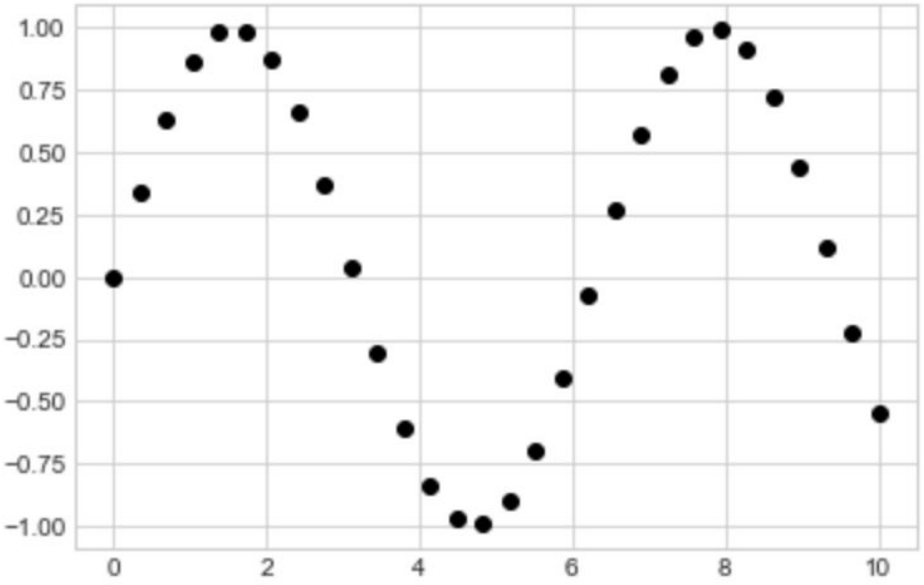


**Figure 2.11: График рассеяния с одной переменной (одна группа)**

В данном случае у нас разные группы: маркеры: ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np


rng = np.random.RandomState(0)
for marker in ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']:
    plt.plot(rng.rand(5), rng.rand(5), marker,
             label="marker='{0}'".format(marker))
plt.legend(numpoints=1)
plt.xlim(0, 1.8);
```

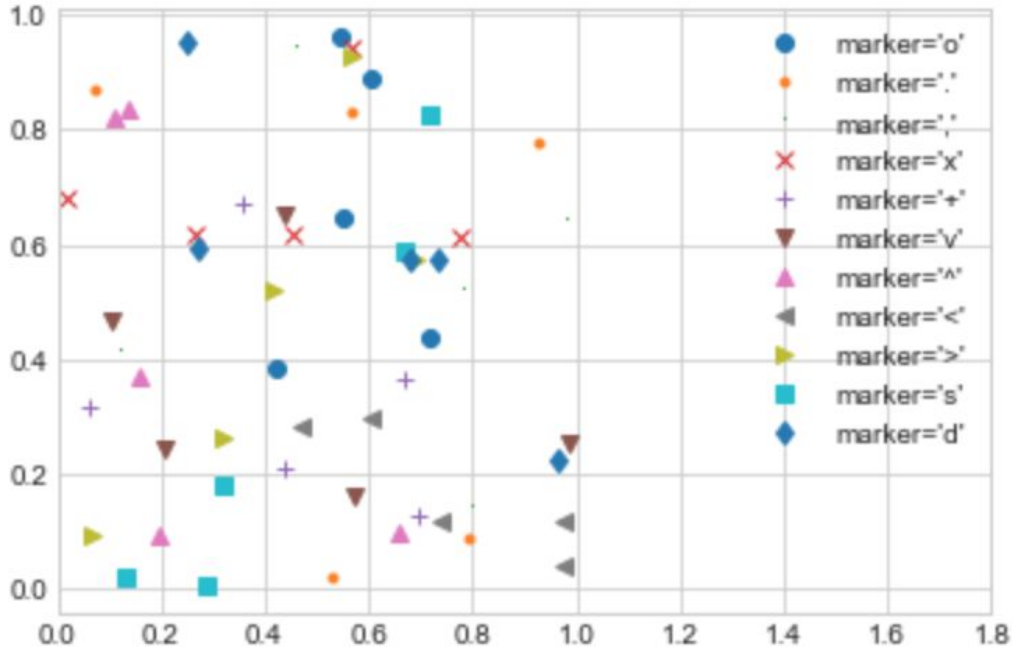


**Figure 2.12: График рассеяния с многочисленными переменными (одиннадцать групп)**

Практика проектирования:

Начать обе оси с нуля для точного представления данных.

Используйте контрастные цвета для точек данных и избегайте использования символов для графиков рассеяния с несколькими группами или категориями.

Варианты: диаграммы рассеяния с маргинальными гистограммами.

В дополнение к графику рассеяния, который визуализирует корреляцию между двумя числовыми переменными, можно построить предельное распределение для каждой переменной в виде гистограмм, чтобы лучше понять, как каждая переменная распределена.

**Examples**:
На следующей диаграмме показана корреляция между массой тела и максимальной продолжительностью жизни животных класса Aves. Также показаны предельные гистограммы, что помогает лучше понять обе переменные:

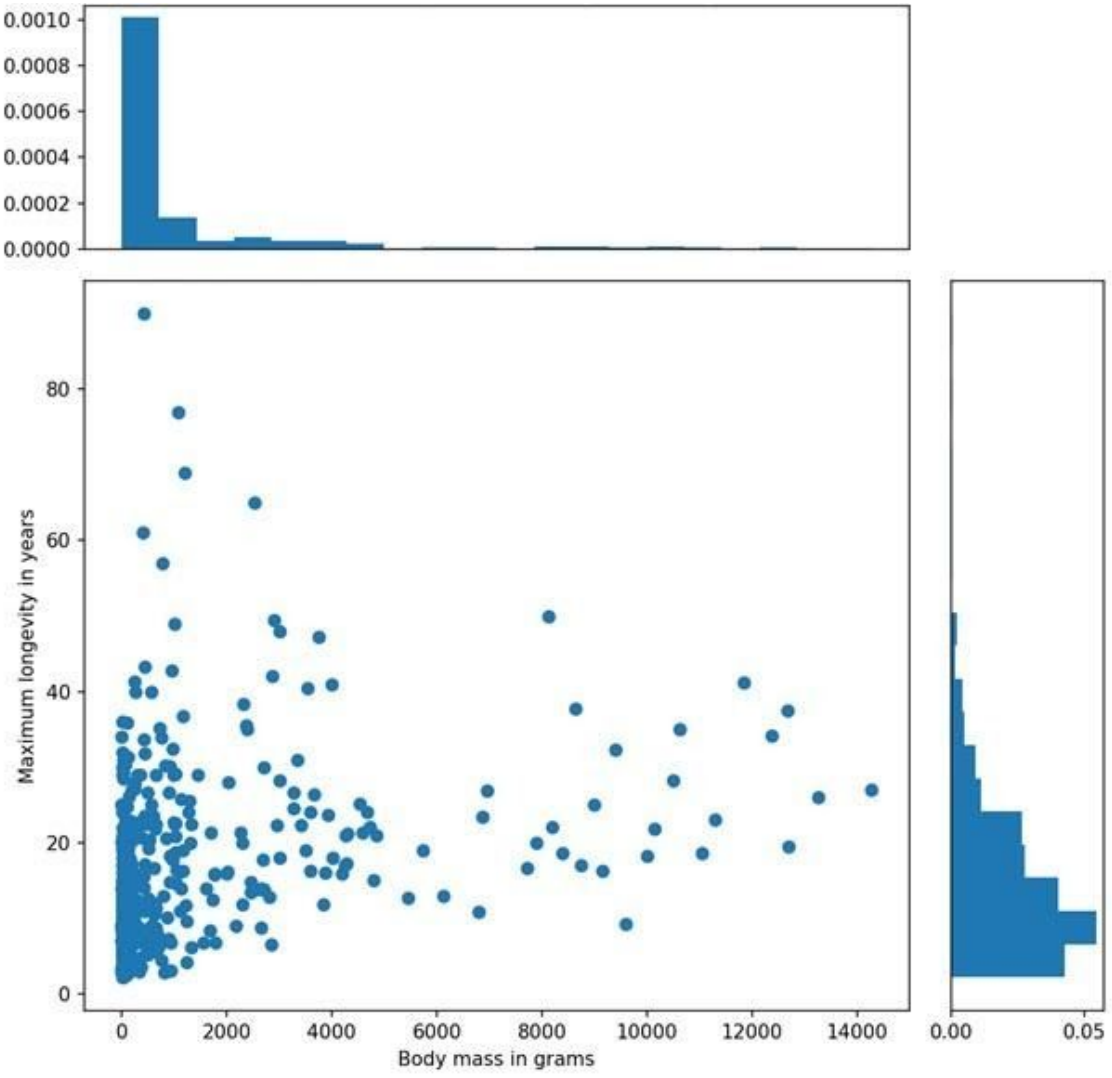


**Figure 2.14: Взаимосвязь между массой тела и максимальной продолжительностью жизни класса Aves с предельными гистограммами**

## 5. Pie Chart

Круговые диаграммы иллюстрируют числовую пропорцию, разделяя окружность на срезы. Каждая длина дуги представляет собой пропорцию категории. Полная окружность равна 100%. Людям легче сравнивать гистограммы, чем длины дуг, поэтому большую часть времени рекомендуется использовать гистограммы или уложенные в стопку гистограммы.

**Uses**:
Сравните предметы, которые являются частью целого.

**Examples**:

```
import matplotlib.pyplot as plt

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]

fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.show()
```

На следующей диаграмме показана круговая диаграмма, показывающая различные положения поля:

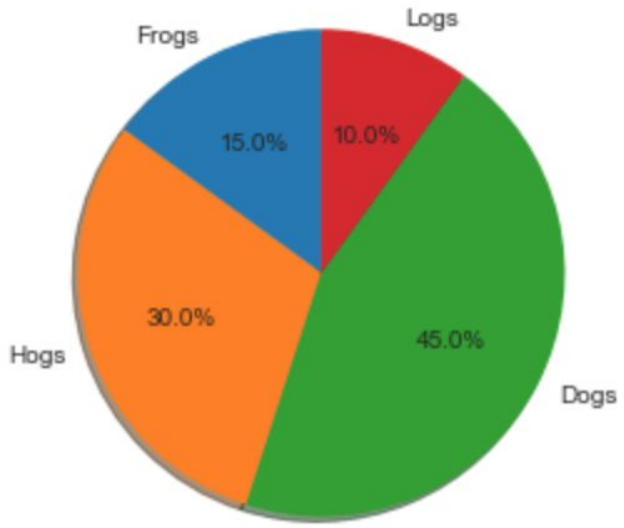


**Figure 2.21: Круговая диаграмма, показывающая положение поля в крикетной площадке**

```
import matplotlib.pyplot as plt

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.show()
```

На следующей диаграмме показано использование воды во всем мире:

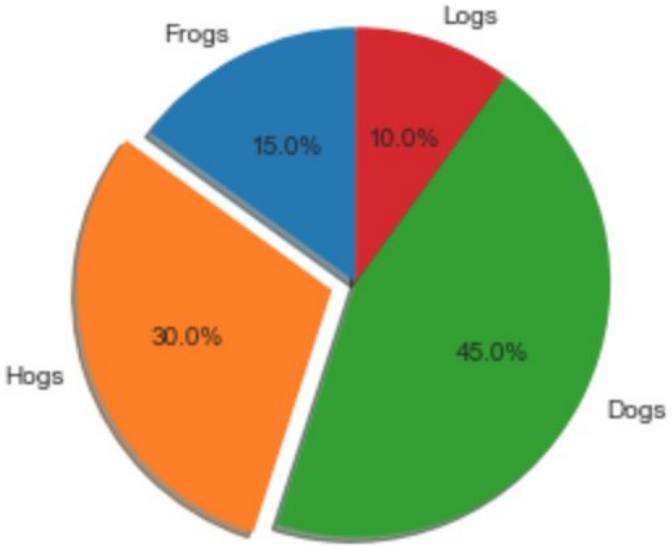


**Figure 2.22: Круговая диаграмма для животных, где выделена одна часть**

**Практика проектирования:**

Располагайте ломтики в соответствии с их размерами в порядке возрастания/уменьшения, по часовой или против часовой стрелки.

Убедитесь, что каждый кусочек имеет свой цвет.

Варианты: пончиковая диаграмма

Альтернативой круговой диаграмме является пончиковая диаграмма. В отличие от круговой диаграммы, легче
сравнить размер фрагментов, так как читатель больше концентрируется на чтении длины дуг, а не площади. Понятные диаграммы также более эффективны с точки зрения использования пространства, так как центр вырезан, поэтому его можно использовать для отображения информации или дальнейшего разделения групп на подгруппы.

На следующем рисунке показана базовая пончиковая диаграмма:

```
import matplotlib.pyplot as plt

names='groupA', 'groupB', 'groupC', 'groupD',
size=[12,11,3,30]

# Create a circle for the center of the plot
my_circle=plt.Circle( (0,0), 0.7, color='white')

plt.pie(size, labels=names, colors=['red','green','blue','skyblue'])
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```

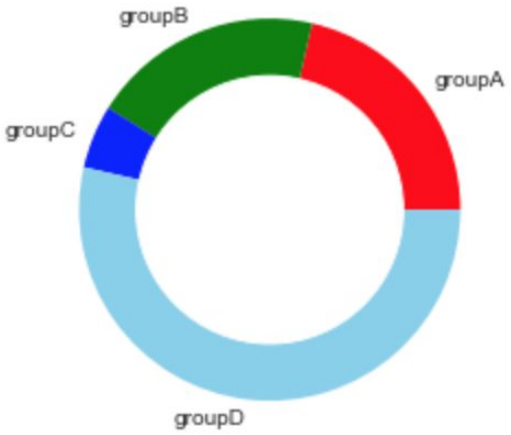


**Figure 2.23: Donut chart**

На следующем рисунке показан график donut chart с подгруппами:

```
import matplotlib.pyplot as plt
import numpy as np

fig, ax = plt.subplots()

size = 0.3
vals = np.array([[60., 32.], [37., 40.], [29., 10.]])

cmap = plt.get_cmap("tab20c")
outer_colors = cmap(np.arange(3)*4)
inner_colors = cmap(np.array([1, 2, 5, 6, 9, 10]))

ax.pie(vals.sum(axis=1), radius=1, colors=outer_colors,
       wedgeprops=dict(width=size, edgecolor='w'))

ax.pie(vals.flatten(), radius=1-size, colors=inner_colors,
       wedgeprops=dict(width=size, edgecolor='w'))

ax.set(aspect="equal", title='Pie plot with `ax.pie`')
plt.show()
```


**Figure 2.24: Donut chart с подгруппами**

**Практика проектирования:**

- Используйте тот же цвет (который используется для категории) для подкатегорий. Используйте различные уровни яркости для различных подкатегорий.

## 6. Box Plot

На графике в рамке показаны многочисленные статистические измерения. Окно простирается от нижнего к верхнему квартилю значений данных, что позволяет нам визуализировать интерквартильный диапазон. Горизонтальная линия внутри окошка обозначает медиану. Вискеры, простирающиеся от окошка, показывают диапазон данных. Кроме того, можно отобразить пропуски данных, как правило, в виде кругов или ромбов, мимо конца усов.

**Uses**:

Если вы хотите сравнить статистические показатели для нескольких переменных или групп, вы можете просто построить несколько ящиков рядом друг с другом.

**Examples**:

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(19680801)

spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))
```

```
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
plt.show()
```

На следующем рисунке показан основной график коробки:

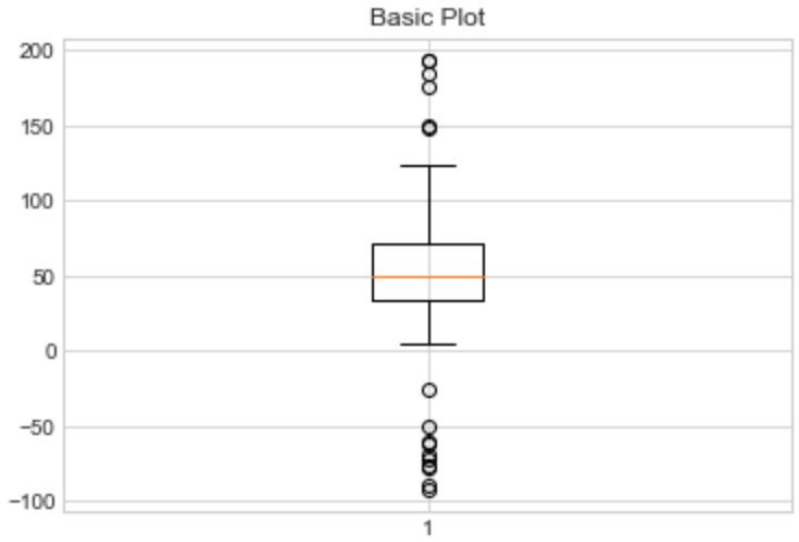


**Figure 2.34: Box plot , показывающий одну переменную**

На следующей диаграмме показаны базовые графики violinplot для нескольких переменных:

```
import matplotlib.pyplot as plt

np.random.seed(10)
collectn_1 = np.random.normal(100, 10, 200)
collectn_2 = np.random.normal(80, 30, 200)
collectn_3 = np.random.normal(90, 20, 200)
collectn_4 = np.random.normal(70, 25, 200)

data_to_plot = [collectn_1, collectn_2, collectn_3, collectn_4]

fig = plt.figure()

ax = fig.add_axes([0,0,1,1])

bp = ax.violinplot(data_to_plot)
plt.show()
```

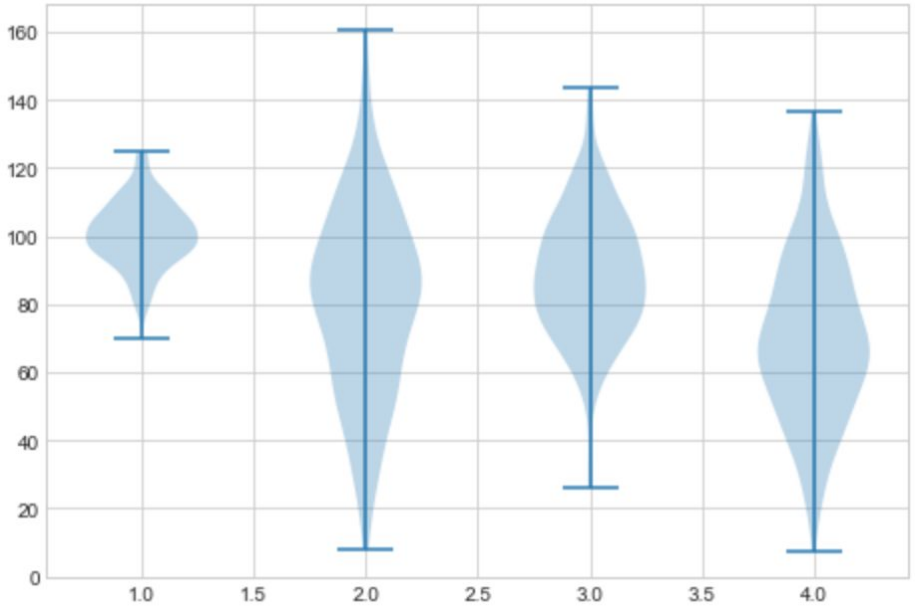


**Figure 2.35: Box plot для нескольких переменных**

**Практическая часть:**

1. **Задание 1:** Pie Charts
    a. Добавляем интересующие нас библиотеки для реализации данного задания:

```
1. import numpy as np
2. import matplotlib.pyplot as plt
```

    b. Создаем массивы для дальнейшей работы программы

```
1. vals = [24, 17, 53, 21, 35]
2. labels = [“Ford”, “Toyota”, “BMV”, “AUDI”, “Jaguar”]
3. explode = (0.1, 0, 0.15, 0, 0)
```

    c. Создаем область Figure, а после добавляем область Axes:

```
1. fig, ax = plt.subplots()
```

    d. Создаем круговую диаграмму на основе созданных нами данных (добавьте в отчет что обозначают параметры в ax.pie):

```
1. ax.pie(vals, labels=labels, autopct=’%1.1f%%’, shadow = True, explode = explode, wedgeprors = {‘lw’:1, ‘ls’:’—‘, ‘edgecolor’:”k”}, rotatelabels=True)
```

    e. Преобразуем вид круговой диаграммы.

```
1. ax.axis(“equal”)
```

f. Демонстрируем полученный Figure.

```
1. fig.show()
```

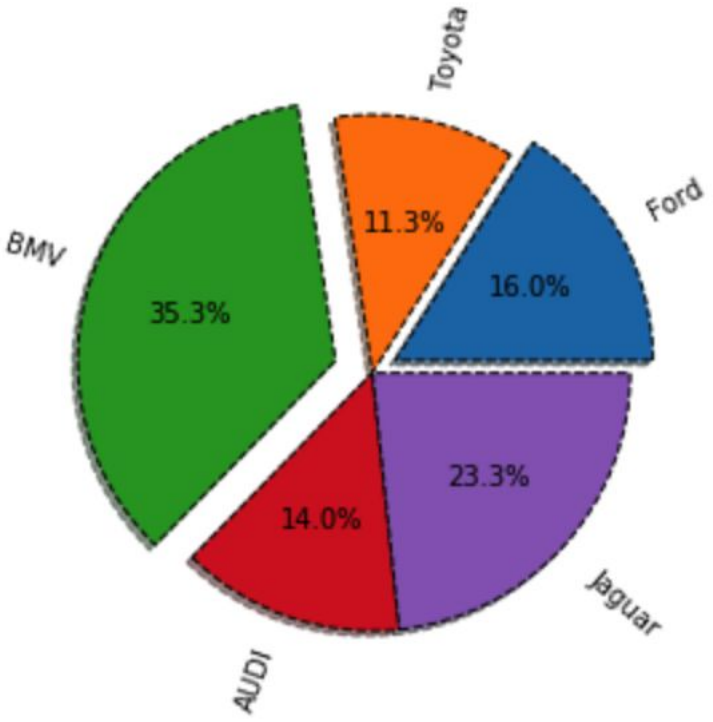


Figure 6: Результат выполнения задания 1

2. **Задание 2:** Box Plots

a. Добавляем интересующие нас библиотеки для реализации данного задания:

```
1. import numpy as np
2. import matplotlib.pyplot as plt
```

b. Создаем данные для дальнейшей работы с графиком

```
1. np.random.seed(10)
2. data_1 = np.random.normal(100, 10, 200)
3. data_2 = np.random.normal(90, 20, 200)
4. data_3 = np.random.normal(80, 30, 200)
5. data_4 = np.random.normal(70, 40, 200)
```

```
6. data = [data_1, data_2, data_3, data_4]
```

c. Создаем область Figure, а после добавляем область Axes. Создаем плот на основе добавленных данных

```
1. fig = plt.figure(figsize =(10, 7))
2. ax = fig.add_subplot(111)
3. bp = ax.boxplot(data, patch_artist = True,
4.                 notch ='True', vert = 0)
```

g. Переименовываем значения Y

```
1. ax.set_yticklabels(['data_1', 'data_2',
2.                     'data_3', 'data_4'])
```

h. Демонстрируем полученный Figure.

```
1. plt.show(bp)
```

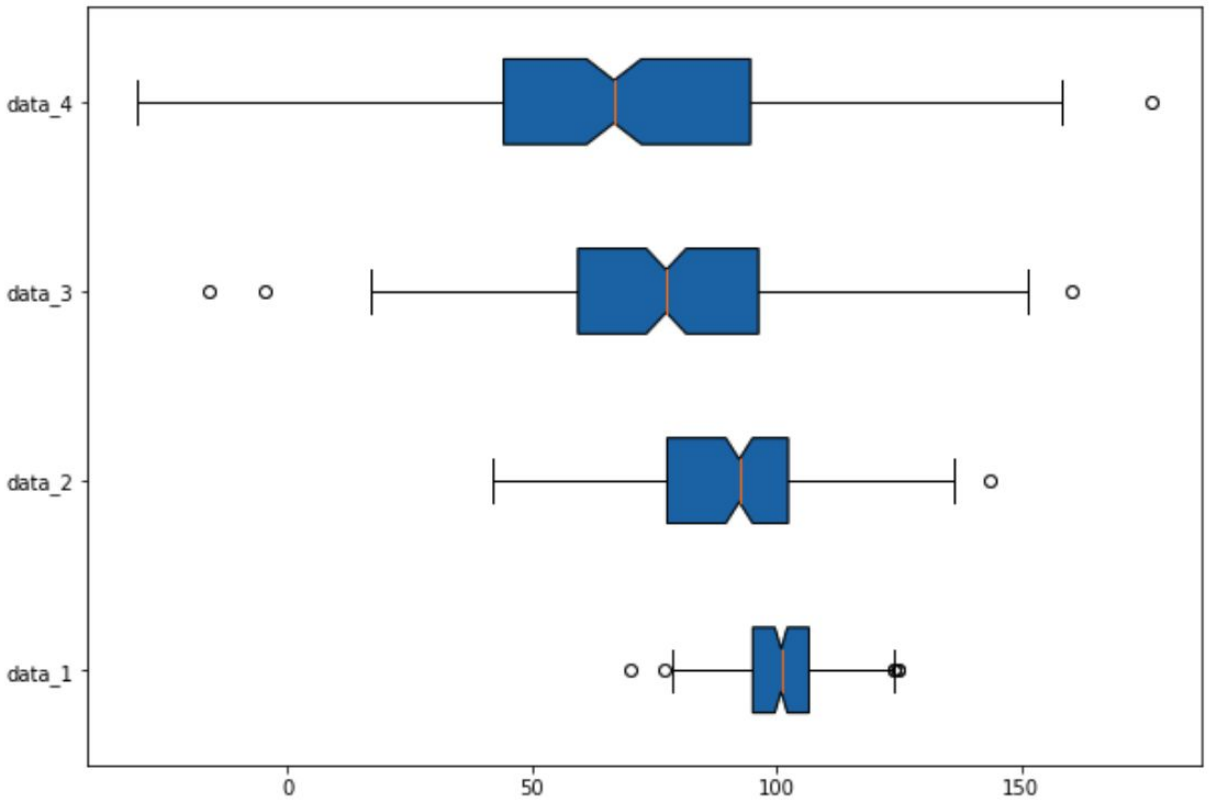


Figure 7: Результат выполнения задания 2

3. **Задание 3:** Scatter Plots

a. Добавляем интересующие нас библиотеки для реализации данного задания:

```
1. import numpy as np
2. import matplotlib.pyplot as plt
```

b. Создаем данные для дальнейшей работы, используя библиотеку numpy

```
1. N = 500
2. x = np.random.rand(N)
3. y = np.random.rand(N)
4. colors = (0,0,0)
5. area = np.pi*3
```

c. Создаем собрание точек, сразу применяя к ней свойства цвета и размера

```
1.  plt.scatter(x, y, s=area, c=colors, alpha=0.5)
```

i. Изменяем названия лейблов и заголовка

```
1. plt.title('Scatter plot')
2. plt.xlabel('x')
3. plt.ylabel('y')
```

j. Демонстрируем полученный Figure.

```
1. plt.show()
```

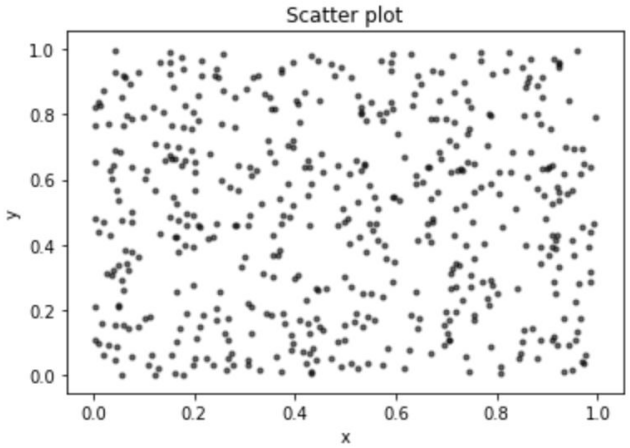


Figure 8: Результат выполнения задания 3

4. **Задание 3:** Bubble Plots

d. Добавляем интересующие нас библиотеки для реализации данного задания:

```
1. import numpy as np
2. import matplotlib.pyplot as plt
```

e. Создаем данные для дальнейшей работы, используя библиотеку numpy

```
1. x = np.random.rand(40)
2. y = np.random.rand(40)
3. z = np.random.rand(40)
4. colors = np.random.rand(40)
```

f. Создаем гистограмму, сразу применяя к ней свойства цвета

```
1.  plt.scatter(x, y, s=z*1000,c=colors)
```

k. Демонстрируем полученный Figure.

```
2. plt.show()
```

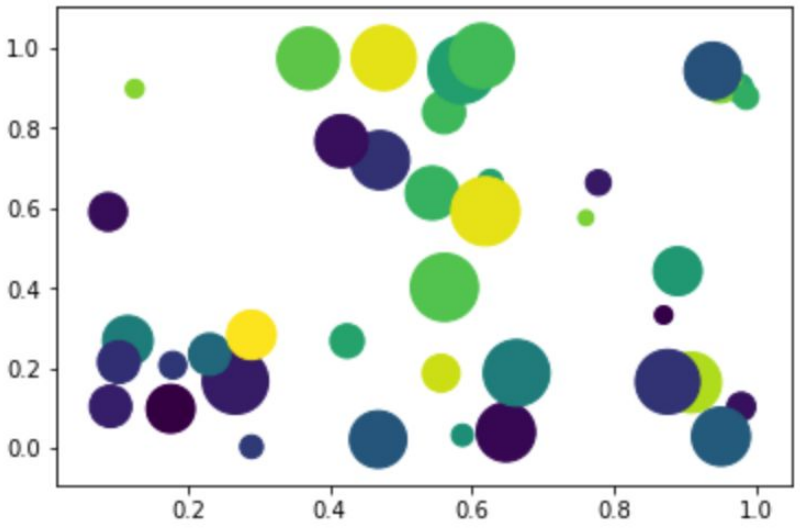


Figure 8: Результат выполнения задания 3

5. ЗАДАНИЕ: Постройте лестницу из случайного полученного количества ступенек при помощи Scatter Plots, при том, что длина каждой ступени увеличивается в 2 раза по отношению к предыдущей.

6. ЗАДАНИЕ: При помощи Pie Charts создайте график, который отражает континенты планеты и как подгруппы их страны и города.

7. ЗАДАНИЕ: Покажите зависимость между ростом, весом и возрастом ваших однокурсников с помощью Bubble plot и Box plot.