Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |
| --- |
| Институт информационных технологий  и анализа данных |
| наименование института |

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

по дисциплине «Технологии обработки, анализа и визуализации данных»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИИТм-23-1 |  |  |  | Солопов Д.Д. |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Проверил |  |  |  |  |  | Григорьев С.В. |
|  |  |  |  | подпись |  | И.О. Фамилия |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2024 г

**Содержание**

[Введение 3](#_Toc182849867)

[1 Подготовка данных 4](#_Toc182849868)

[2 Загрузка данных в Processing 6](#_Toc182849869)

[3 Класс визуальных объектов 10](#_Toc182849870)

[4 Построение координатной сетки и добавление управления с помощью мыши 13](#_Toc182849871)

[Заключение 17](#_Toc182849872)

[Список использованных источников 18](#_Toc182849873)

[Приложение 19](#_Toc182849874)

Введение

**Цель работы**: необходимо на основе динамических или сериальных данных (по выбранному варианту) сформулировать задачу для визуализации и разработать интерактивное приложение в Processing для решения этой задачи и его интерпретации.

**Задачи**:

1. Изучить исходные данные, сформулировать задачу для визуализации данных, обосновать выбор методов для подготовки и визуализации данных;
2. Подготовить данные к визуализации;
3. Визуализировать данные в приложении, создать отдельный класс визуализации;
4. Внедрить в приложение методы интерактивного управления визуализацией;
5. Интерпретировать полученную визуализацию и оценить насколько она решает поставленную задачу.

**Поставленная задача визуализации**: визуализировать отдельные видовые группы пингвинов и визуально оценить общие размерные характеристики представителей этих отдельных групп. На основе оценки размеров пингвинов из разных видовых групп сделать заключение о группе с самыми крупными представителями, средними и самыми малыми. Оценка представителей видовых групп будет осуществляться по длине и глубине клюва, а также длине плавника.

1 Подготовка данных

Набор данных представляет собой записи о размерах пингвинов различных видов.

Всего в наборе данных 344 записи, каждая из которых содержит информацию о виде (species), острове обитания (island), длины верхнего края клюва (culmen\_length\_mm), глубины верхнего края клюва (culmen\_depth\_mm), длина плавника (flipper\_length\_mm), масса тела (body\_mass\_g) и пол (sex).

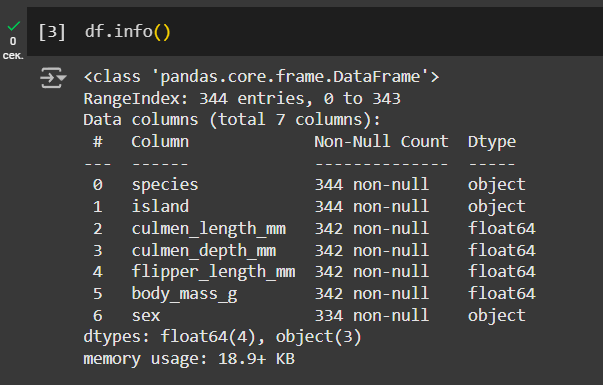


Рисунок 1 – Информация о наборе данных

Отметим, что в наборе данных есть null-значения, которые необходимо удалить. Такие значения находятся во всех столбцах, кроме species и island.

Удалить все null-значения можно с помощью следующего программного кода:

df = df.dropna(axis=0)

После удаления записей с null-значениями в наборе данных остаётся 334 записи:

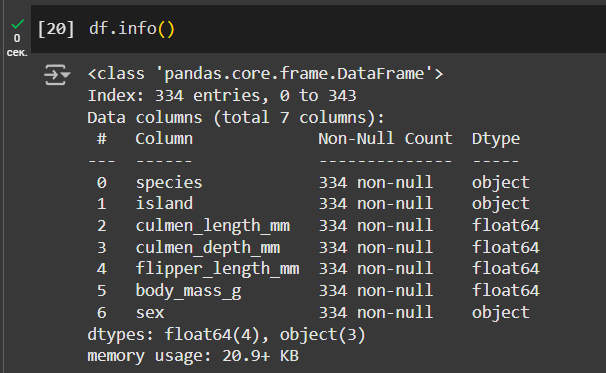


Рисунок 2 – Обработанный набор данных

Также отметим, что в столбце sex встречается значение “.”, что не даёт никакого понимания о поле пингвина. Поэтому строку, где пол пингвина указан как точка, следует удалить.

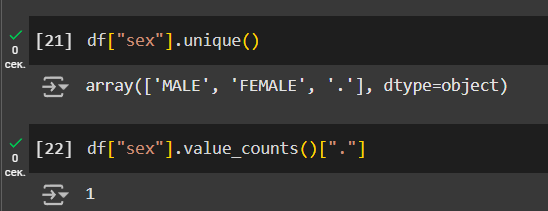


Рисунок 3 – Содержание некорректных данных

Удаление данной записи можно сделать с помощью следующей команды:

df = df.drop(df[df["sex"] == "."].index)

Как и ожидалось, удалится всего лишь одна запись и в наборе данных будет ровно 333 записи:

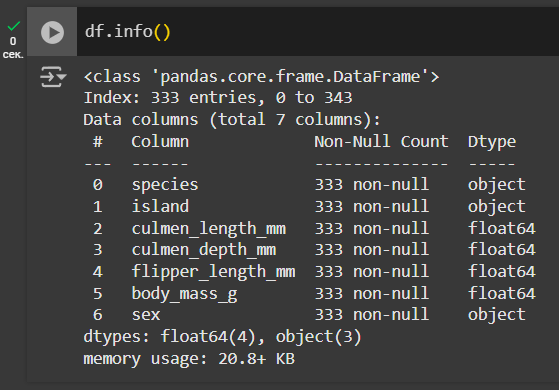


Рисунок 4 – Информация о наборе данных

2 Загрузка данных в Processing

Для загрузки данных в Processing будет использован следующий программный код:

// Объект таблицы

Table table;

// Путь до файла с данными

String data\_filepath = "..\\data\\penguins\_size.csv";

// Массив пингвинов

ArrayList<Penguin> penguins;

// Объект для хранения min/max значений

MaxMinObj maxMin;

// Чтение массива пингвинов из CSV-файла

ArrayList<Penguin> csvReadToArray() {

ArrayList<Penguin> result = new ArrayList<Penguin>();

Table table = loadTable(data\_filepath, "header");

for(TableRow row : table.rows()) {

// Вычисление координат x, y, z

float x = row.getFloat("culmen\_length\_mm");

float y = row.getFloat("culmen\_depth\_mm");

float z = row.getFloat("flipper\_length\_mm");

String species = row.getString("species");

/\*String island = row.getString("island");

float body\_mass\_g = row.getFloat("body\_mass\_g");

String sex = row.getString("sex");\*/

color c;

c = color(0, 128, 255);

// Определение цвета по виду

switch(species) {

case "Adelie": {

c = color(0, 255, 0);

break;

}

case "Chinstrap": {

c = color(255, 0, 0);

break;

}

case "Gentoo": {

c = color(0, 0, 255);

break;

}

default: {

break;

}

}

// Добавление информации о пингвине в массив

result.add(new Penguin(x, y, z, c, 5));

}

float max\_x, min\_x;

float max\_y, min\_y;

float max\_z, min\_z;

// Вычисление маскимальной координаты x

max\_x = Collections.max(result, new MaxX()).getX();

// Вычисление минимальной координаты y

min\_x = Collections.max(result, new MinX()).getX();

// Вычисление максимальной координаты y

max\_y = Collections.max(result, new MaxY()).getY();

// Вычисление минимальной координаты y

min\_y = Collections.max(result, new MinY()).getY();

// Вычисление максимальной координаты z

max\_z = Collections.max(result, new MaxZ()).getZ();

// Вычисление минимальной координаты z

min\_z = Collections.max(result, new MinZ()).getZ();

// Формирование объекта, для хранения максимальных и минимальных значений x, y и z

maxMin = new MaxMinObj(max\_x, min\_x, max\_y, min\_y, max\_z, min\_z);

return result;

}

В программном коде загрузки данных из таблицы мы используем специальный объект table типа Table, который загружает таблицу по пути, представленного в переменной data\_filepath.

Все загруженные интересующие нас данные о пингвинах будут находится в структуре ArrayList result.

В основном цикле загрузки данных из таблицы мы связываем с координатами x, y и z определённые характеристики пингвина. Координата x будет связана с culmen\_length\_mm, координата y будет связана с culmen\_depth\_mm, а координата z будет связана с flipper\_length\_mm.

Таким образом координаты (x, y, z) будут характеризовать размеры пингвина определённого вида.

Вид пингвина мы записываем в переменную species, на основе которой определяется цвет точки, которая будет характеризовать одного пингвина (одну запись из набора данных).

Если пингвин вида “Adelie”, то цвет характеризующей его точки будет зелёным, если вида “Chinstrap” – красным, а если “Gentoo” – синим.

После добавления пингвинов в ArrayList мы вычисляем максимальные и минимальные значения по координатам x, y и z, которые затем добавляются в объект MaxMinObj:

// Вычисление маскимальной координаты x

max\_x = Collections.max(result, new MaxX()).getX();

// Вычисление минимальной координаты y

min\_x = Collections.max(result, new MinX()).getX();

// Вычисление максимальной координаты y

max\_y = Collections.max(result, new MaxY()).getY();

// Вычисление минимальной координаты y

min\_y = Collections.max(result, new MinY()).getY();

// Вычисление максимальной координаты z

max\_z = Collections.max(result, new MaxZ()).getZ();

// Вычисление минимальной координаты z

min\_z = Collections.max(result, new MinZ()).getZ();

// Формирование объекта, для хранения максимальных и минимальных значений x, y и z

maxMin = new MaxMinObj(max\_x, min\_x, max\_y, min\_y, max\_z, min\_z);

Данные вычисления необходимы для будущей нормировки координат в пространстве.

Класс MaxMinObj выглядит следующим образом:

class MaxMinObj {

public float max\_x;

public float min\_x;

public float max\_y;

public float min\_y;

public float max\_z;

public float min\_z;

public MaxMinObj(float max\_x, float min\_x, float max\_y, float min\_y, float max\_z, float min\_z){

this.max\_x = max\_x;

this.min\_x = min\_x;

this.max\_y = max\_y;

this.min\_y = min\_y;

this.max\_z = max\_z;

this.min\_z = min\_z;

}

}

Это простой DTO (для передачи данных) класс, который содержит максимальны и минимальные значения определённых координат.

Реализация компораторов (функций, которые сравнивают различные значения в ArrayList) выглядит следующим образом:

import java.util.Comparator;

class MaxX implements Comparator<Penguin> {

public int compare(Penguin a, Penguin b) {

if (a.getX() < b.getX())

return -1;

if (a.getX() == b.getX())

return 0;

return 1;

}

}

class MinX implements Comparator<Penguin> {

public int compare(Penguin a, Penguin b) {

if (a.getX() > b.getX())

return -1;

if (a.getX() == b.getX())

return 0;

return 1;

}

}

class MaxY implements Comparator<Penguin> {

public int compare(Penguin a, Penguin b) {

if (a.getY() < b.getY())

return -1;

if (a.getY() == b.getY())

return 0;

return 1;

}

}

class MinY implements Comparator<Penguin> {

public int compare(Penguin a, Penguin b) {

if (a.getY() > b.getY())

return -1;

if (a.getY() == b.getY())

return 0;

return 1;

}

}

class MaxZ implements Comparator<Penguin> {

public int compare(Penguin a, Penguin b) {

if (a.getZ() < b.getZ())

return -1;

if (a.getZ() == b.getZ())

return 0;

return 1;

}

}

class MinZ implements Comparator<Penguin> {

public int compare(Penguin a, Penguin b) {

if (a.getZ() > b.getZ())

return -1;

if (a.getZ() == b.getZ())

return 0;

return 1;

}

}

Для каждой координаты и каждой задачи (нахождение максимума / минимума) реализован свой компаратор.

Каждый компоратор определяет, каким образом необходимо сортировать элементы в ArrayList.

3 Класс визуальных объектов

При визуализации будет построена трёхмерная диаграмма рассеивания, в качестве точек на диаграмме будут использованы сферы, которые будут раскрашены в определённый цвет определяемый видом пингвина.

На диаграммах рассеяния ряда точек, размещённых в декартовой системе координат, отображает значения по трём переменным. Присвоив каждой оси переменную, можно определить, существуют ли отношения или корреляции между этими тремя переменными.

Класс визуализации объектов выглядит следующим образом:

class Penguin {

private float \_x;

private float \_y;

private float \_z;

private color \_color;

private float \_radius;

// Конструктор с параметрами

public Penguin(float x, float y, float z, color c, float r){

this.\_x = x;

this.\_y = y;

this.\_z = z;

this.\_color = c;

this.\_radius = r;

}

public void display(MaxMinObj maxMin, float size) {

// Вычисление нормированных координат

float x = (this.\_x - maxMin.min\_x) \* size/ maxMin.max\_x;

float y = (this.\_y - maxMin.min\_y) \* size/ maxMin.max\_y;

float z = (this.\_z - maxMin.min\_z) \* size/ maxMin.max\_z;

pushMatrix();

noStroke();

lights();

translate(x, y, z);

fill(this.\_color);

sphere(this.\_radius);

popMatrix();

}

public float getX() {

return this.\_x;

}

public float getY() {

return this.\_y;

}

public float getZ() {

return this.\_z;

}

}

Класс визуализации имеет метод display, в которй передаётся объект MaxMinObj и размер сферы, которая будет отображать одну запись из набора данных в трёхмерном пространстве.

Перед отрисовкой по трёхмерным координатам для начала эти координаты нормируются по следующей формуле:

где coord – полученная координата, min(coords) – минимальная координата (по x, y, или z), size – определённое значение для нормировки (по умолчанию равна 1400), max(coords) – максимальная координата (по x, y или z).

Перед визуализацией объектов класса Pinguin необходимо добавить предварительные настройк окна, в котором будет происходить вся визуализация в функции setup:

void setup(){

// Перемещение окна на координату (50, 50) от левого верхнего угла экрана

windowMove(50, 50);

// Установка размера окна с поддержкой 3D

size(1000, 800, P3D);

background(0);

noStroke();

// Чтение набора данных из CSV

penguins = csvReadToArray();

}

В функции setup сначала происходит перемещение в область окна компьютера (50 пикселей сверху и 50 пикселей слева). Это делается с целью предотвращения сокрытия верхнего меню окна, которое может пропасть, если переместить окно за пределы экрана, тогда из-за кэширования Processing’ом положения окна оно навсегда может прикрепится вверху и не будет возможности его переместить вниз с помощью компьютерной мыши. Вызов функции windowMove решает данную проблему.

После перемещения окна происходит установка его размеров и включение поддержки работы с 3D-визуализацией. Также добавляем стили окна (например, наполнение заднего фона).

После настроек окна происходит считывание данных из набора данных.

Теперь визуализируем каждый объект Penguin в пространстве:

void draw() {

background(255);

translate(width / 3, height / 2, 0);

for(int i = penguins.size() - 1; i >= 0; i--){

Penguin penguin = penguins.get(i);

penguin.display(maxMin, customSize);

}

}

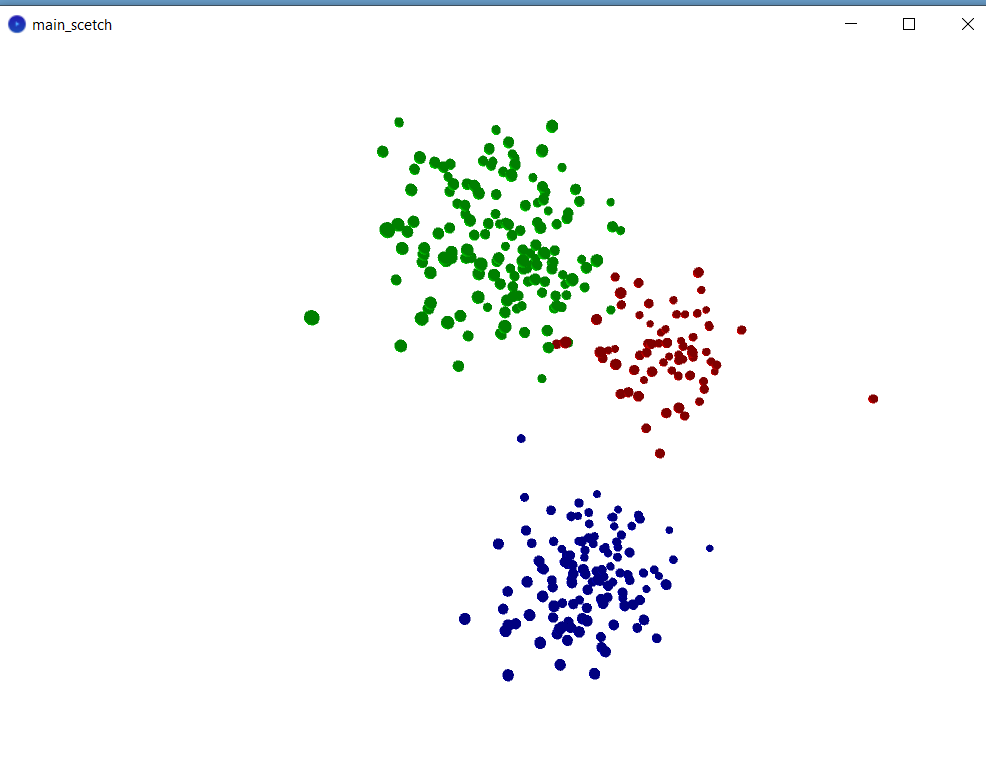


Рисунок 5 – Результат визуализации

Как видно из рисунка 5 мы отобразили в пространстве все размеры разных пингвинов из разных групп. Однако оценить из какой группы пингвины обладают наиболее крупными элементами (клюв и плавник) сейчас нельзя. Необходимо добавить координатную сетку, чтобы можно было дать оценку каждой группе исходя из положения элементов этих групп в пространстве.

4 Построение координатной сетки и добавление управления с помощью мыши

Для начала добавим управление с помощью мыши. Чтобы комфортно оценивать положение групп в пространстве необходима возможность увеличивать и уменьшать масштаб, а также изменять положение камеры.

Для изменения масштаба при помощи прокрутки колёсика мыши определим соответствующий обработчик Processing и добавим туда следующую логику:

float scale = 1;

void mouseWheel(MouseEvent e) {

float diff = e.getCount();

scale = scale + diff / 10;

}

Аналогично переопределим обработчик для удерживания кнопки мыши (правой или левой) для того, чтобы изменять положение камеры:

float x\_coord, y\_coord;

float new\_x\_coord, new\_y\_coord;

void mouseDragged() {

new\_x\_coord = mouseX / float(width) \* TWO\_PI;

new\_y\_coord = mouseY / float(width) \* TWO\_PI;

float diff = (x\_coord - new\_x\_coord);

if(abs(diff) > 0.01){

x\_coord -= diff / 2.0;

}

diff = (y\_coord - new\_y\_coord);

if(abs(diff) > 0.01) {

y\_coord -= diff / 2.0;

}

}

Теперь изменим функцию update таким образом, чтобы была отрисована координатная сетка, применён определённый масштаб и перемещение к координатам, которые были получены при движении мыши с задержкой правой или левой её кнопки:

void draw() {

background(255);

translate(width / 3, height / 2, 0);

scale(scale);

rotateX(-y\_coord);

rotateY(-x\_coord);

float posZ = -maxMin.min\_z;

stroke(0, 0, 255, 100);

line(0, 0, posZ, 0, 0, sizeWindow);

fill(255, 0, 0);

stroke(255, 0, 0, 100);

line(0, 0, posZ, 0, sizeWindow, posZ);

fill(0, 255, 0);

stroke(0, 0, 255, 100);

line(0, 0, posZ, sizeWindow, 0, posZ);

fill(0, 0, 255);

stroke(200);

int iter = sizeWindow / 10;

int delta = 1;

for(int i = iter; i <= sizeWindow; i = i + iter) {

line(0, 0, posZ + i, 0, sizeWindow, posZ + i);

line(i, 0, posZ , i, 0, sizeWindow);

line(0, i, posZ, 0, i, sizeWindow);

line(0, 0, posZ + i, sizeWindow, 0, posZ + i);

pushMatrix();

textSize(14);

text(delta, i, 0, posZ);

text(delta, 0, i, posZ);

text(delta, 0, 0, posZ + i);

popMatrix();

delta++;

}

pushMatrix();

fill(200);

translate(0, 0, posZ);

rect(0, 0, sizeWindow, sizeWindow);

popMatrix();

for(int i = penguins.size() - 1; i >= 0; i--){

Penguin penguin = penguins.get(i);

penguin.display(maxMin, customSize);

}

}

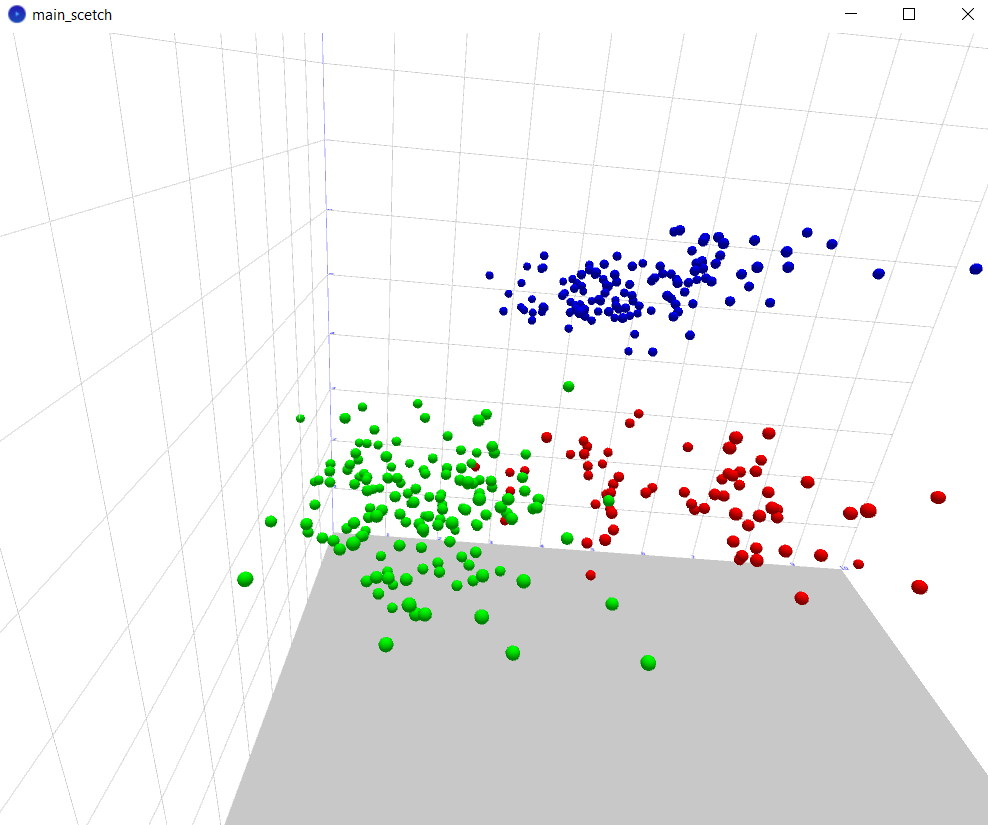


Рисунок 6 – Результат визуализации с координатными сетками

Теперь можно дать оценку каждой группе пингвинов исходя из цветового выделения каждой группе и положению элементов каждой группы относительно осей координат.

Группа пингвинов, обозначенная синим цветом (“Gentoo”) выделяется наиболее крупными представителями (см. рис. 7), поскольку именно в данной группе координаты x, y и z больше всех, а значит и соответствующие характеристики (culmen\_length\_mm, culmen\_depth\_mm, flipper\_length\_mm) являются самыми большими.

Группа пингвинов, обозначенная красным цветом (“Adelie”) не превосходит по размерам представителей вида “Gentoo”, но превосходит представителей вида “Chinstrap”.

Представители группы пингвинов, обозначенная зелёным цветом (“Chinstrap”), обладают наименьшими общими размерами рассматриваемых характеристик из всех групп.

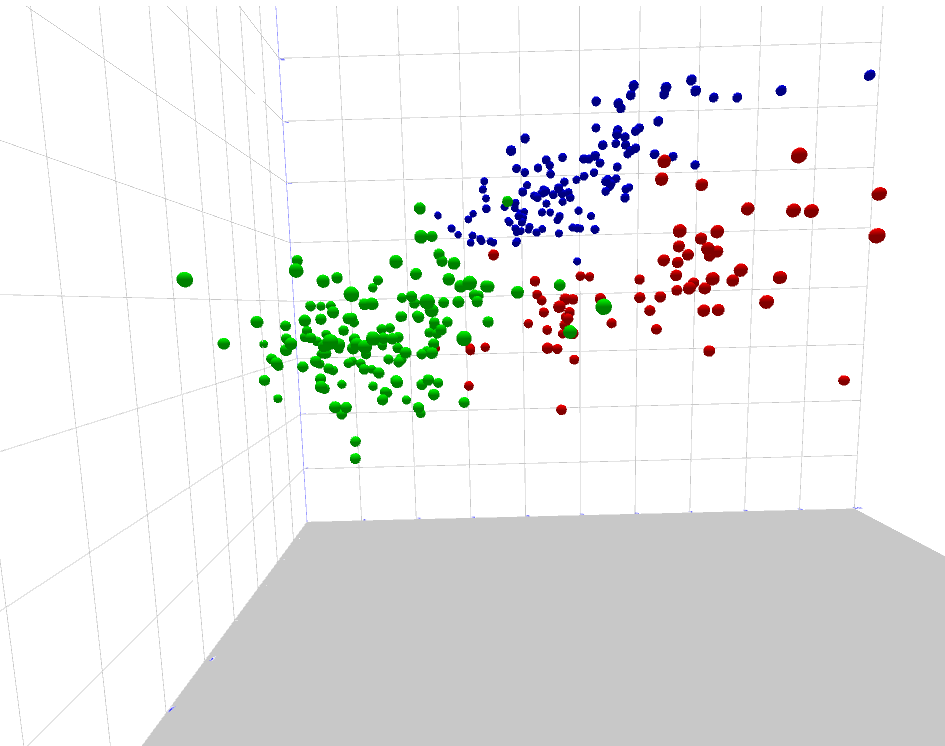


Рисунок 7 – Пингвины вида “Gentoo” выделяются наибольшими размерами своих представителей на фоне других видов пингвинов

Подведём итоги:

1. Пингвины вида “ Gentoo” самые крупные (самые большие);
2. Пингвины вида “Adelie” не крупнее пингвинов группы “Gentoo”, однако крупнее пингвинов вида “ Chinstrap” (средние по размерам);
3. Пингвины вида “Chinstrap” менее крупны, чем представители пингвинов из группы “Adelie” (самые маленькие).

Заключение

В рамках данной лабораторной работы на основе сериальных данных была сформулирована задача для визуализации и разработано интерактивное приложение в Processing для решения поставленной задачи визуализации.

Поставленная задача визуализации состояла в оценке общих размерных характеристик отдельных видовых групп пингвинов с целью выявления группы с самыми крупными представителями, средними и самыми маленькими.

В ходе выполнения лабораторной работы были изучены исходные данные и проведена их предварительная подготовка для визуализации. В ходе подготовки были выявлены некорректные данные (например, запись с точкой вместо указания пола пингвина) и удалены записи с null-значениями.

После подготовки данных был реализован механизм по их загрузке в среду Processing с последующей визуализацией через класс визуализации Penguin, который был реализован исходя из требования визуализации пингвина как объекта в трёхмерной среде по координатам x, y и z, каждая из которой была представлена в виде отдельной физической характеристики пингвина.

Каждая видовая группа пингвинов была окражена в отдельный цвет: Gentoo – синий, Adelie – красный и Chinstrap – зелёный.

Были также разработаны механизмы интерактивного взаимодействия с приложением: добавлена возможность изменения масштаба с помощью колёсика мыши и перемещения с помощью задержки кнопки мыши.

В ходе визуального анализа видовых групп пингвинов было выяснено, что группа Gentoo обладает самыми группными представителями, группа Adelie менее крупными представителями, а группа Chinstrap обладает самыми маленькими представителями.

Поставленная цель лабораторной работы достигнута, а поставленные задачи успешно решены.

Список использованных источников

1. Редактируем CSV-файлы, чтобы не сломать данные (URL: <https://habr.com/ru/companies/hflabs/articles/432906/>)
2. Seaborn (URL: <https://blog.skillfactory.ru/glossary/seaborn/>)
3. Каталог Визуализации Данных (URL: <https://datavizcatalogue.com/RU/metody/diagramma_razmaha.html>)
4. 10 полезных Python-библиотек для Data Science (URL: <https://www.reg.ru/blog/10-poleznyh-python-bibliotek-dlya-data-science/>)
5. Каталог Визуализации Данных (URL: <https://datavizcatalogue.com/RU/metody/teplovaja_karta.html>)

Приложение

1. Ссылка на исходный код проекта: <https://github.com/DanSoW/INRTU/tree/main/data-processing-analysis-and-visualization-technologies/lab6/main_scetch>