Отчет по лабораторной работе 7

Петрушов Дмитрий, 1032212287

Содержание

# 1 Цель работы

Основной целью работы является изучение специализированных пакетов Julia для обработки данных.

# 2 Выполнение лабораторной работы

## 2.1 Julia для науки о данных

В Julia для обработки данных используются наработки из других языков программирования, в частности, из R и Python.

## 2.2 Считывание данных

Перед тем, как начать проводить какие-либо операции над данными, необходимо их откуда-то считать и возможно сохранить в определённой структуре.

Довольно часто данные для обработки содержаться в csv-файле, имеющим текстовый формат, в котором данные в строке разделены, например, запятыми, и соответствуют ячейкам таблицы, а строки данных соответствуют строкам таблицы. Также данные могут быть представлены в виде фреймов или множеств.

В Julia для работы с такого рода структурами данных используют пакеты CSV, DataFrames, RDatasets, FileIO (рис. [[1](#fig:001)]):



Figure 1: Установка пакетов

Предположим, что у нас в рабочем каталоге с проектом есть файл с данными programminglanguages.csv, содержащий перечень языков программирования и год их создания. Тогда для заполнения массива данными для последующей обработки требуется считать данные из исходного файла и записать их в соответствующую структуру (рис. [[2](#fig:002)]):

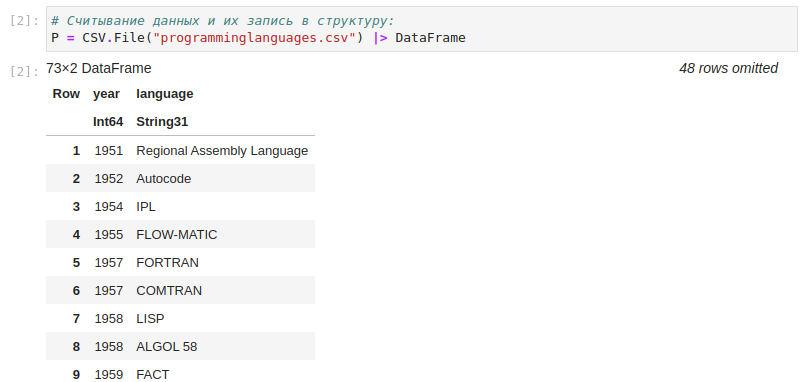


Figure 2: Считывание данных и запись в структуру

Пример для Julia (рис. [[3](#fig:003)]):

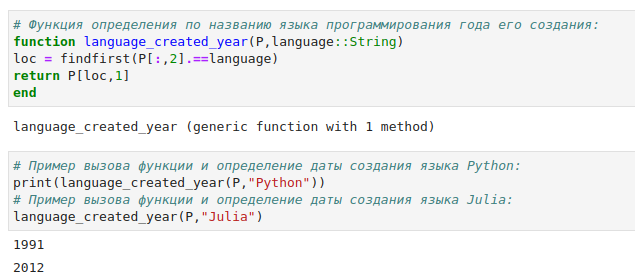


Figure 3: Пример

В следующем примере при вызове функции, в качестве аргумента которой указано слово julia, написанное со строчной буквы (рис. [[4](#fig:004)]):

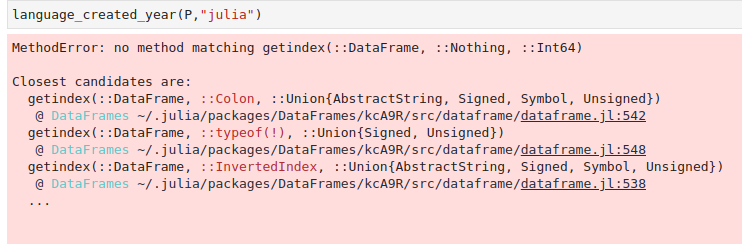


Figure 4: Поиск “julia” со строчной буквы

Для того, чтобы убрать в функции зависимость данных от регистра, необходимо изменить исходную функцию следующим образом (рис. [[5](#fig:005)]):

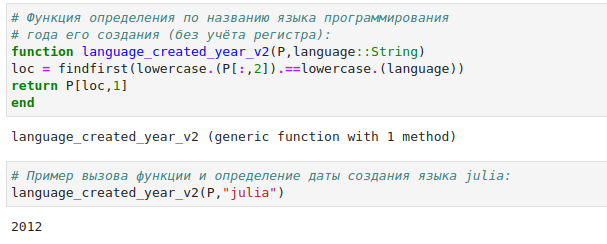


Figure 5: Изменение исходной функции

Можно считывать данные построчно, с элементами, разделенными заданным разделителем (рис. [[6](#fig:006)]):

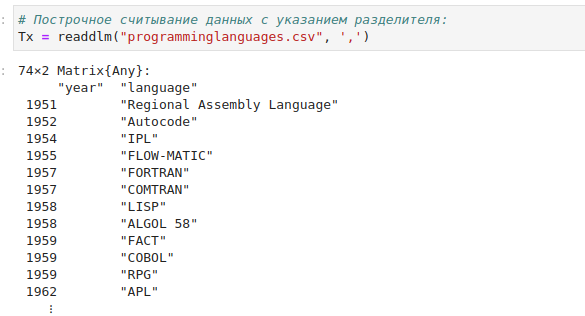


Figure 6: Построчное считывание данных

## 2.3 Запись данных в файл

Предположим, что требуется записать имеющиеся данные в файл. Для записи данных в формате CSV можно воспользоваться следующим вызовом (рис. [[7](#fig:007)]):

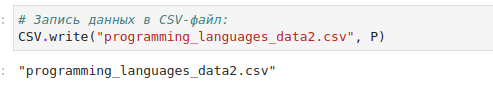


Figure 7: Запись данных в файл

Можно задать тип файла и разделитель данных (рис. [[8](#fig:008)]):

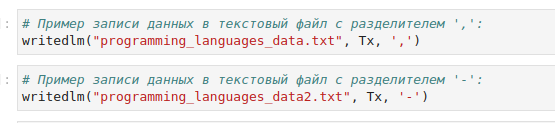


Figure 8: Пример с указанием типа данных и разделителем данных

Можно проверить, используя readdlm, корректность считывания созданного текстового файла (рис. [[9](#fig:009)]):

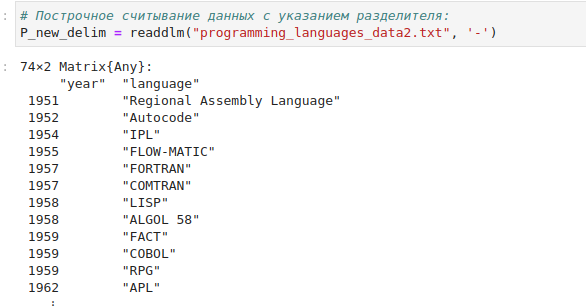


Figure 9: Проверка корректности считывания созданного текстового файла

## 2.4 Словари

При работе с данными бывает удобно записать их в формате словаря.

Предположим, что словарь должен содержать перечень всех языков программирования и года их создания, при этом при указании года выводить все языки программирования, созданные в этом году.

При инициализации словаря можно задать конкретные типы данных для ключей и значений (рис. [[10](#fig:010)]):

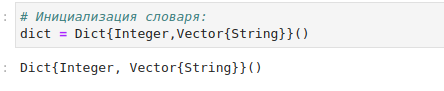


Figure 10: Инициализация словаря

Можно инициировать пустой словарь, не задавая строго структуру (рис. [[11](#fig:011)]):

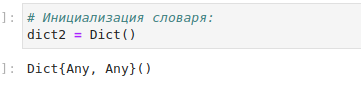


Figure 11: Инициализация пустого словаря

Далее требуется заполнить словарь ключами и годами, которые содержат все языки программирования, созданные в каждом году, в качестве значений (рис. [[12](#fig:012)]):

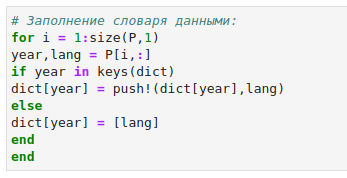


Figure 12: Заполнение словаря данными

В результате при вызове словаря можно, выбрав любой год, узнать, какие языки программирования были созданы в этом году (рис. [[13](#fig:013)]):

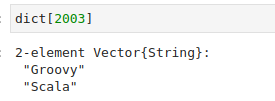


Figure 13: Пример работы словаря

## 2.5 DataFrames

Работа с данными, записанными в структуре DataFrame, позволяет использовать индексацию и получить доступ к столбцам по заданному имени заголовка или по индексу столбца.

На примере с данными о языках программирования и годах их создания зададим структуру DataFrame (рис. [[14](#fig:014)]):

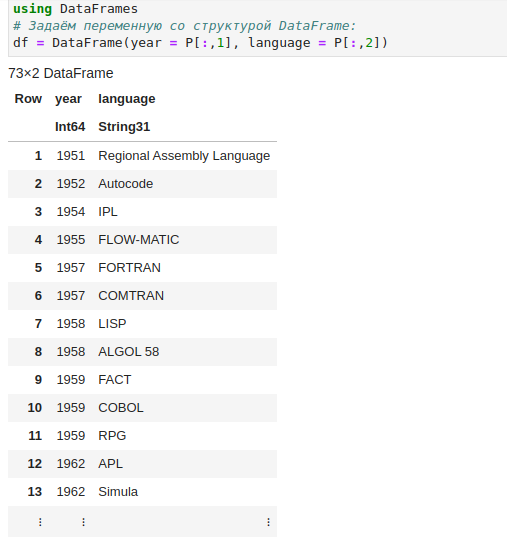


Figure 14: Пример создания структуры DataFrame

## 2.6 RDatasets

С данными можно работать также как с наборами данных через пакет RDatasets языка R (рис. [[15](#fig:015)]):

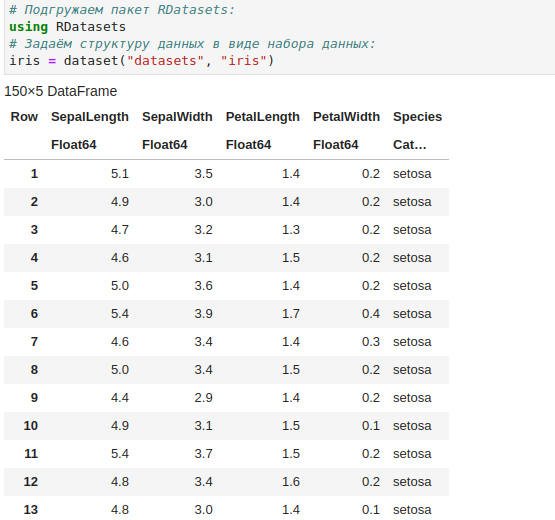


Figure 15: Работа с пакетом RDatasets

Пакет RDatasets также предоставляет возможность с помощью description получить основные статистические сведения о каждом столбце в наборе данных (рис. [[16](#fig:016)]):

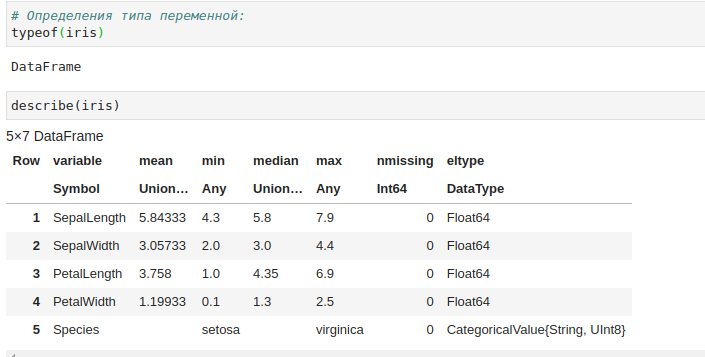


Figure 16: Получение основных статических сведений о каждом столбце в наборе данных

## 2.7 Работа с переменными отсутствующего типа (Missing Values)

Пакет DataFrames позволяет использовать так называемый «отсутствующий» тип (рис. [[17](#fig:017)]):

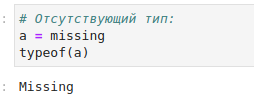


Figure 17: Использование “отсутствующего” типа

В операции сложения числа и переменной с отсутствующим типом значение также будет иметь отсутствующий тип (рис. [[18](#fig:018)]):

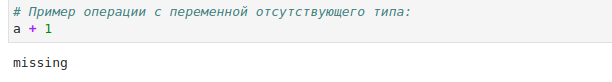


Figure 18: Операция сложения числа и переменной с отсутствующим типом

Приведём пример работы с данными, среди которых есть данные с отсутствующим типом.

Предположим есть перечень продуктов, для которых заданы калории. В массиве значений калорий есть значение с отсутствующим типом (рис. [[19](#fig:019)]):

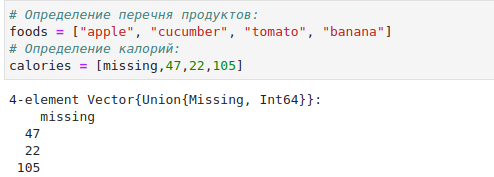


Figure 19: Пример работы с данными, среди которых есть данные с отсутствующим типом

При попытке получить среднее значение калорий, ничего не получится из-за наличия переменной с отсутствующим типом.

Для решения этой проблемы необходимо игнорировать отсутствующий тип (рис. [[20](#fig:020)]):

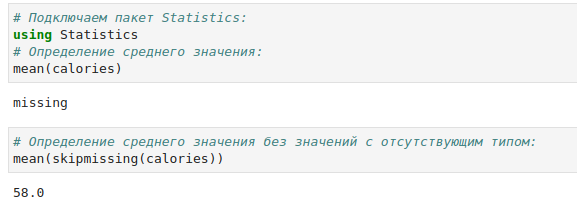


Figure 20: Игнорирование отсутствующего типа

Далее показано, как можно сформировать таблицы данных и объединить их в один фрейм (рис. [[21](#fig:021)]):

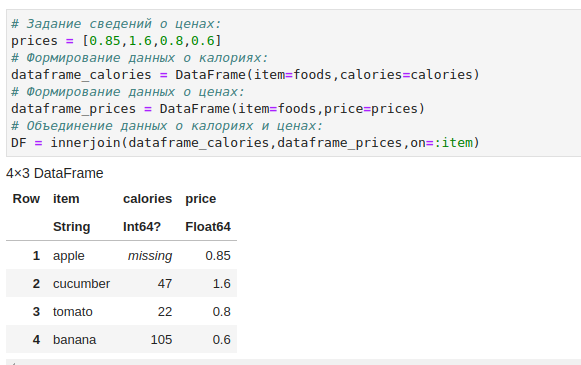


Figure 21: Формирование таблиц данных и их объединение в один фрейм

## 2.8 FileIO

В Julia можно работать с так называемыми «сырыми» данными, используя пакет FileIO.

Попробуем посмотреть, как Julia работает с изображениями.

Подключим соответствующий пакет (рис. [[22](#fig:022)]):



Figure 22: Подключение пакетов

Загрузим изображение (в данном случае логотип Julia) (рис. [[23](#fig:023)]):

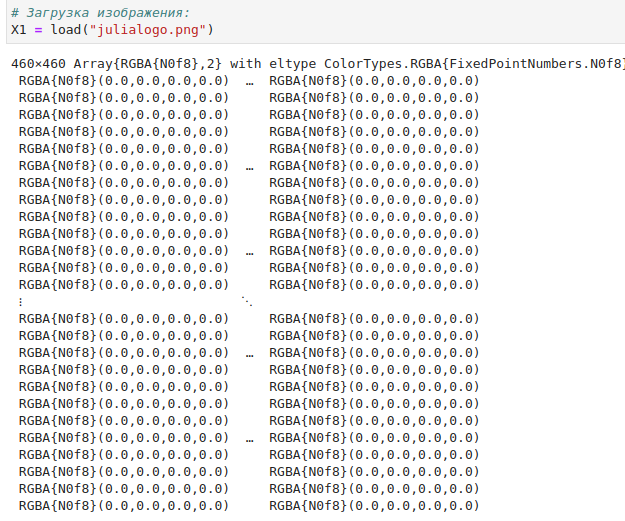


Figure 23: Загрузка изображения

Julia хранит изображение в виде множества цветов (рис. [[24](#fig:024)]):

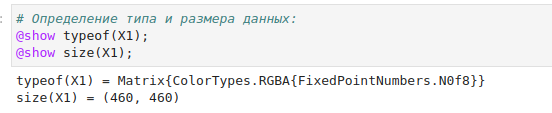


Figure 24: Определение типа и размера данных

## 2.9 Обработка данных: стандартные алгоритмы машинного обучения в Julia. Кластеризация данных. Метод k-средних

Задача кластеризации данных заключается в формировании однородной группы упорядоченных по какому-то признаку данных.

Метод k-средних позволяет минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров.

Рассмотрим задачу кластеризации данных на примере данных о недвижимости. Файл с данными houses.csv содержит список транзакций с недвижимостью в районе Сакраменто, о которых было сообщено в течение определённого числа дней.

Сначала подключим необходимые для работы пакеты (рис. [[25](#fig:025)]):



Figure 25: Подключение нужных пакетов

Затем загрузим данные (рис. [[26](#fig:026)]):

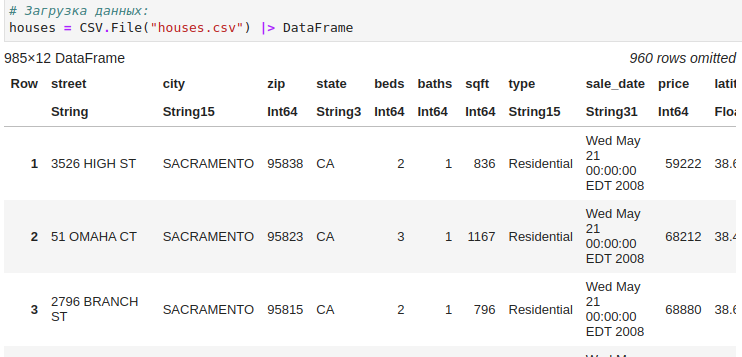


Figure 26: Загрузка данных

Построим график цен на недвижимость в зависимости от площади (рис. [[27](#fig:027)]):

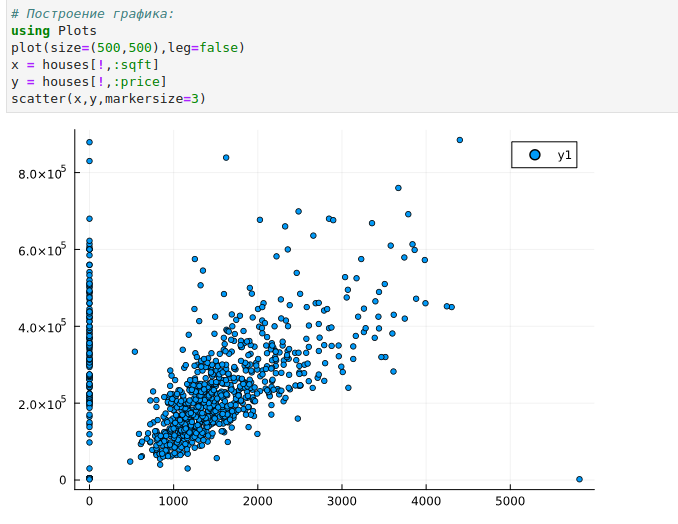


Figure 27: Построение графика цен на недвижимость в зависимости от площади

Для того чтобы избавиться от “артефактов”, можно отфильтровать и исключить такие значения, получить более корректный график цен (рис. [[28](#fig:028)]):

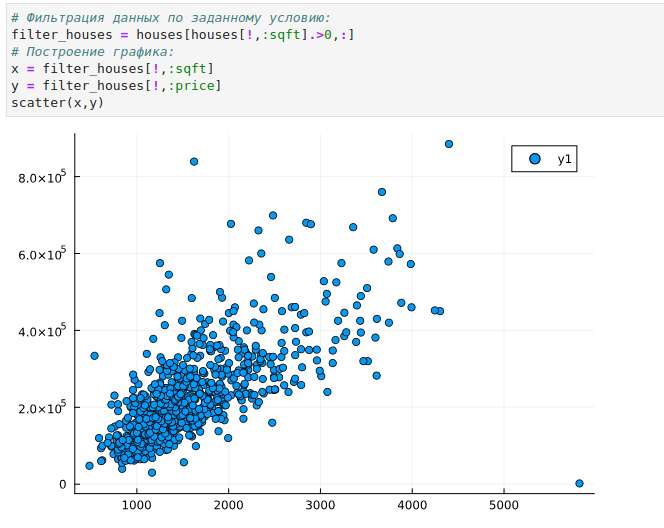


Figure 28: Построение графика без “артефактов”

Построим график, обозначив каждый кластер отдельным цветом (рис. [[29](#fig:029)]):

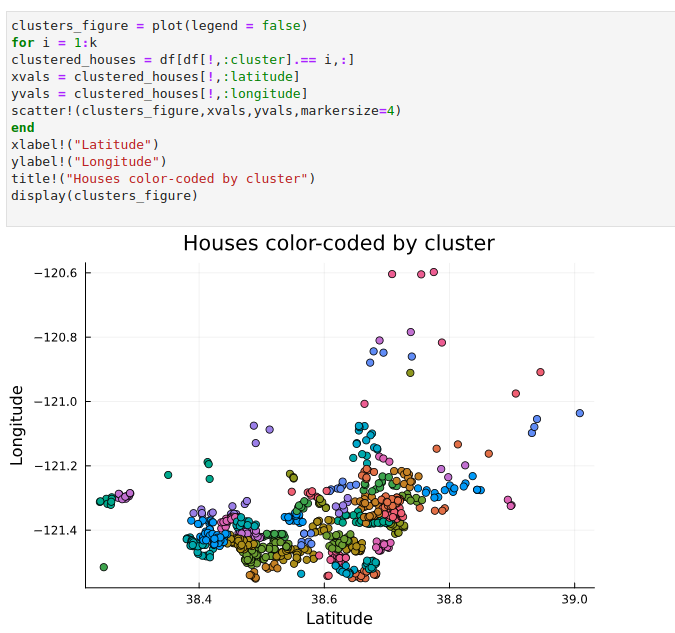


Figure 29: Построение графика с кластерами разных цветов

Построим график, раскрасив кластеры по почтовому индексу (рис. [[30](#fig:030)]):

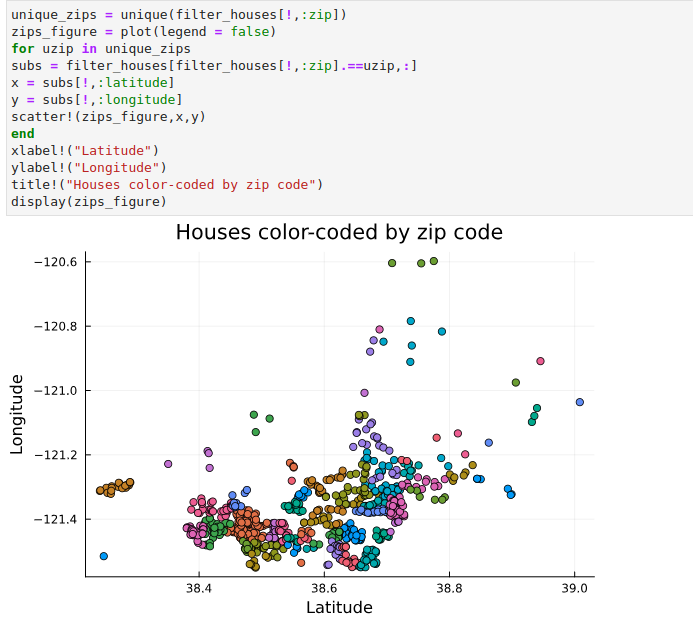


Figure 30: Построение графика с кластерами разных цветов по почтовому индексу

## 2.10 Кластеризация данных. Метод k ближайших соседей

Отобразим на графике соседей выбранного объекта недвижимости (рис. [[31](#fig:031)]):

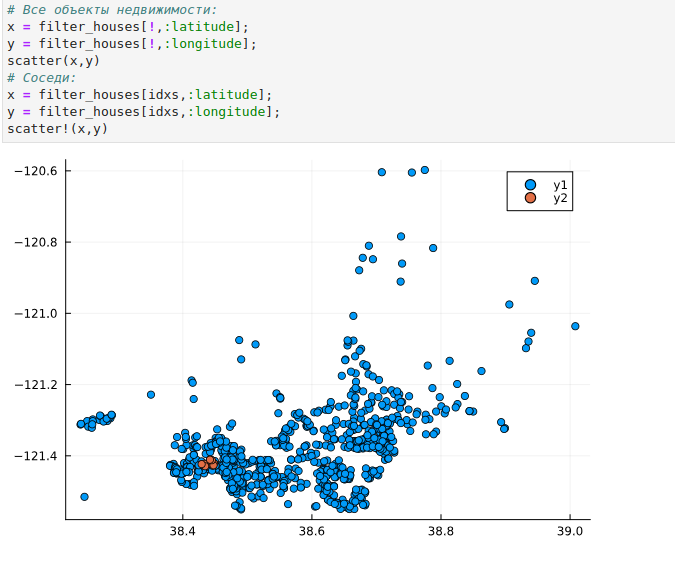


Figure 31: Отображение на графике соседей выбранного объекта недвижимости

Используя индексы idxs и функцию :city для индексации в DataFrame filter\_houses, можно определить районы соседних домов (рис. [[32](#fig:032)]):

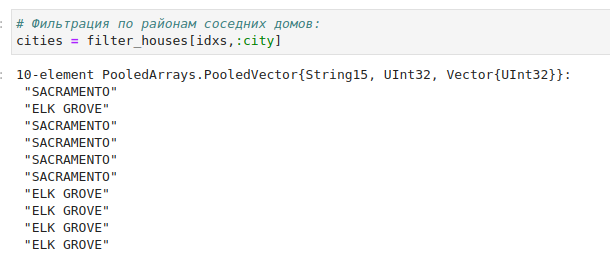


Figure 32: Определение районов соседних домов

## 2.11 Обработка данных. Метод главных компонент

Метод главных компонент (Principal Components Analysis, PCA) позволяет уменьшить размерность данных, потеряв наименьшее количество полезной информации. Метод имеет широкое применение в различных областях знаний, например, при визуализации данных, компрессии изображений, в эконометрике, некоторых гуманитарных предметных областях, например, в социологии или в политологии.

На примере с данными о недвижимости попробуем уменьшить размеры данных о цене и площади из набора данных домов (рис. [[33](#fig:033)]):

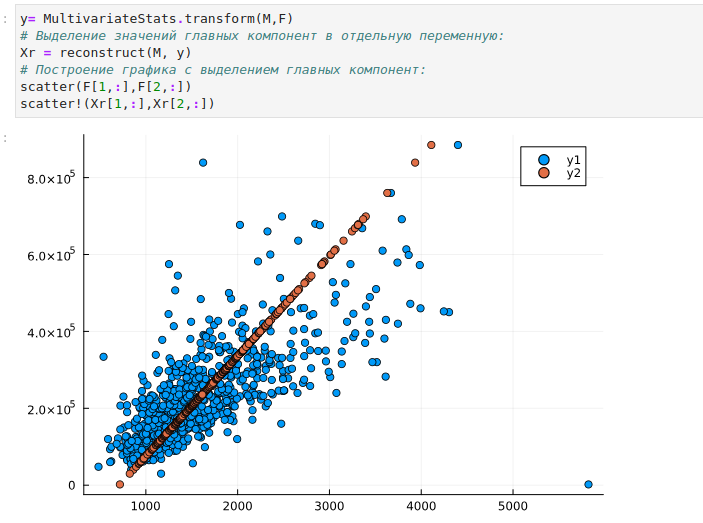


Figure 33: Попытка уменьшения размера данных о цене и площади из набора данных домов

## 2.12 Обработка данных. Линейная регрессия

Регрессионный анализ представляет собой набор статистических методов исследования влияния одной или нескольких независимых переменных (регрессоров) на зависимую (критериальная) переменную. Терминология зависимых и независимых переменных отражает лишь математическую зависимость переменных, а не причинноследственные отношения.

Наиболее распространённый вид регрессионного анализа — линейная регрессия, когда находят линейную функцию, которая согласно определённым математическим критериям наиболее соответствует данным.

Зададим случайный набор данных (можно использовать и полученные экспериментальным путём какие-то данные). Попробуем найти для данных лучшее соответствие (рис. [[34](#fig:034)]):

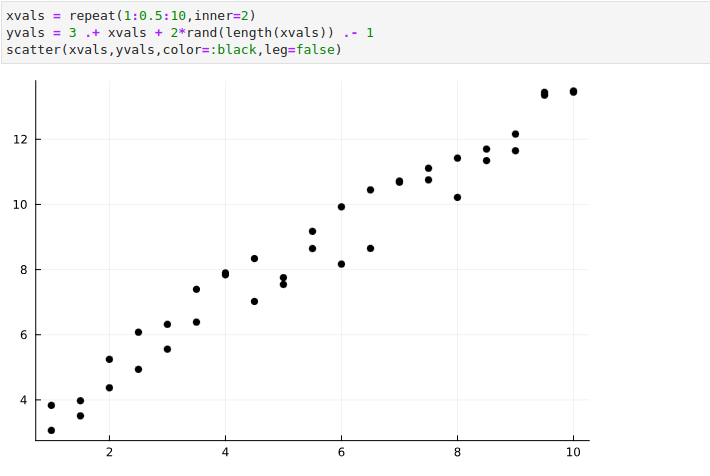


Figure 34: Исходные данные

Определим функцию линейной регрессии. Применим функцию линейной регрессии для построения соответствующего графика значений (рис. [[35](#fig:035)]):

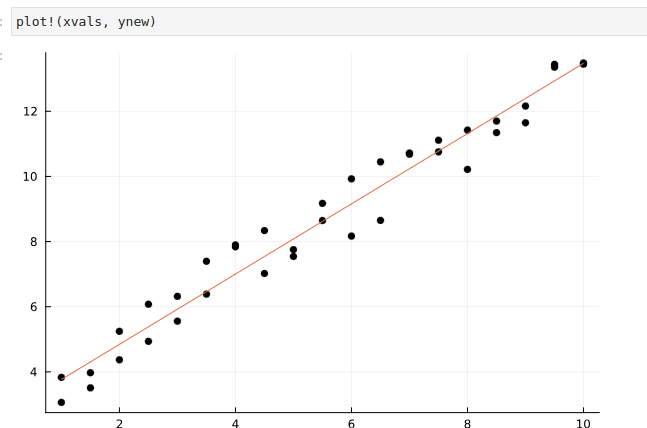


Figure 35: Применение функции для построения графика

Сгенерируем больший набор данных. Определим, сколько времени потребуется, чтобы найти соответствие этим данным. Для сравнения реализуем подобный код на языке Python. Используем пакет для анализа производительности, чтобы провести сравнение (рис. [[36](#fig:036)]):



Figure 36: Сравнение

## 2.13 Самостоятельное выполнение

Выполнение задания №1 (рис. [[37](#fig:037)] - рис. [[39](#fig:039)]):

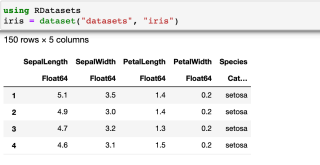


Figure 37: Решение задания №1

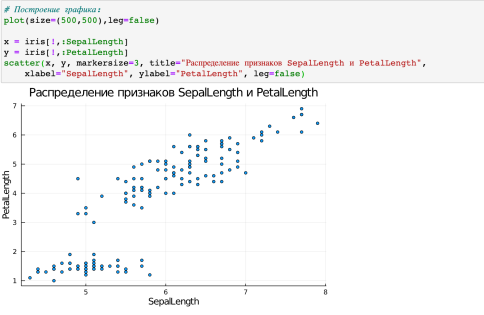


Figure 38: Решение задания №1

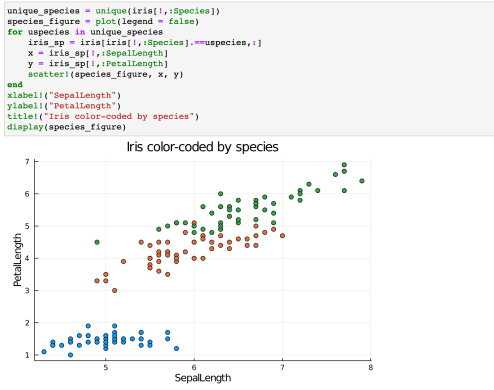


Figure 39: Решение задания №1

Выполнение задания №2 (рис. [[40](#fig:040)] - рис. [[41](#fig:041)]):

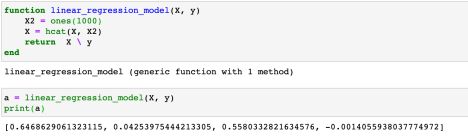


Figure 40: Решение задания №2

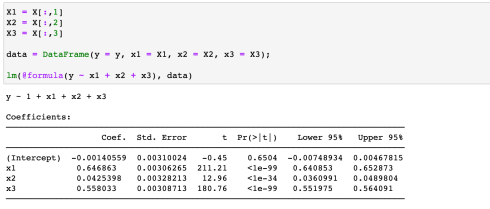


Figure 41: Решение задания №2

Выполнение задания №2-2 (рис. [[42](#fig:042)] - рис. [[45](#fig:045)]):

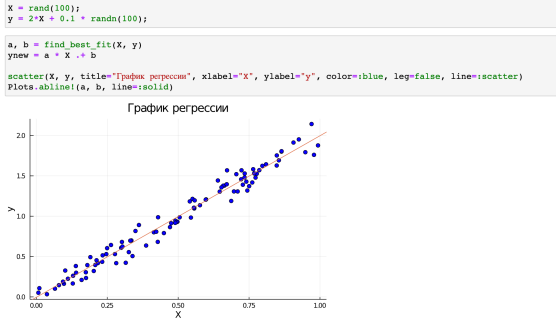


Figure 42: Решение задания №2-2

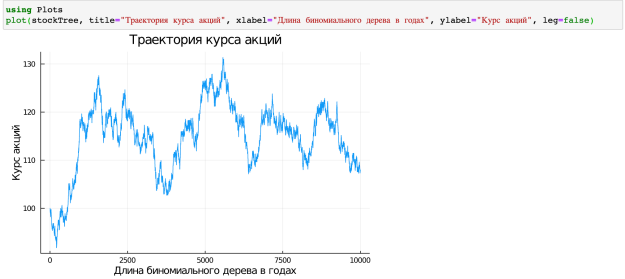


Figure 43: Решение задания №2-2a

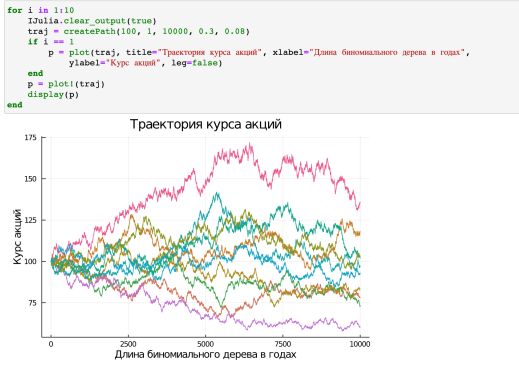


Figure 44: Решение задания №2-2b

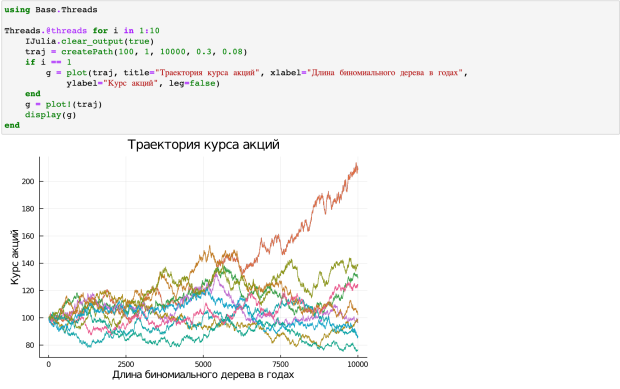


Figure 45: Решение задания №2-2c

# 3 Вывод

В ходе выполнения лабораторной работы были изучены специализированные пакеты Julia для обработки данных.

# Список литературы