ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет *Информационных Технологий и Программирования*

Направление (специальность) *Прикладная математика и информатика*

Квалификация (степень) *Бакалавр прикладной математики и информатики*

Кафедра *Компьютерных технологий* Группа *4539*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Проектирование и реализация веб-сервиса для анализа экспрессии генов

Автор квалификационной работы Зайцев К.В. (подпись)

Руководитель Сергушичев А.А. (подпись)

Консультанты:

а) По экономике и организации производства _____ (подпись)

б) По безопасности жизнедеятельности и экологии _____ (подпись)

в) _____ (подпись)

К защите допустить

Зав. кафедрой *Васильев В.Н.* (подпись)

«___» _____ 2015 г.

Санкт-Петербург, 2015 г.

ОГЛАВЛЕНИЕ

Стр.

ВВЕДЕНИЕ

Биоинформатика образована на стыке информатики, математики и статистики. Основной целью биоинформатики является анализ биологических данных и исследований. Зачастую учёным и специалистам в этой сфере приходится прибегать к сложным алгоритмам и математическим методам для анализа тех или иных данных. Чаще всего этот анализ компьютерный – ведь объем данных, который предстоит обработать, достаточно велик для ручной обработки. Зачастую этот анализ приходится совершать биологам, которые могут не иметь соответствующей подготовки. Таким образом, возникает необходимость в простых инструментах для выполнения биоинформатичекого анализа. Однако инструментам недостаточно только совершать анализ: существует необходимость быстро делиться результатами своих исследований (анализа) с другими учёными и специалистами в этой среде и уметь эти исследования воспроизводить. На настоящий момент не существует решения, которое сочетало бы в себе все эти качества. Разработка такого решения является основной задачей этой работы.

ГЛАВА 1

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Биоинформатика

Биоинформатика – междисциплинарная наука, основной целью и задачей которой является создание методов, алгоритмов и программного обеспечения для анализа биологических данных. Биоинформатика – большая отрасль, но в рамках этой работы нас больше всего будет интересовать анализ генетической экспрессии.

1.1.1 Анализ экспрессии генов

Экспрессия генов – процесс преобразования наследственной информации гена (последовательности нуклеотидов ДНК) в функциональный продукт. Функциональный продукт – чаще всего белок или РНК. Процесс экспрессии используется всеми живыми организмами.

Анализ и измерение экспрессии гена является важной частью многих наук (в частности медицинских), ведь понимание того, насколько отдельный ген используется в клетке, ткани или организме, может дать очень много информации, например:

- Возможность отличить инфицированные вирусом клетки от неинфицированных (экспрессия вирусных белков)
- Определение человеческой предрасположенности к раку (экспрессия онкогенов)

1.1.2 Количественный анализ экспрессии

Количественный анализ экспрессии генов – единовременный анализ экспрессии большого числа генов (сотен или тысяч) для того, чтобы создать общую картину клеточной функции. Сам же анализ может в себя включать:

а) Визуализацию (например, тепловые карты);
б) Кластеризацию;
в) Дифференциальную экспрессию.

На рисунке 1.1 показана тепловая карта – графическое представление данных экспрессии генов, где активность гена показана соответствующим цветом.

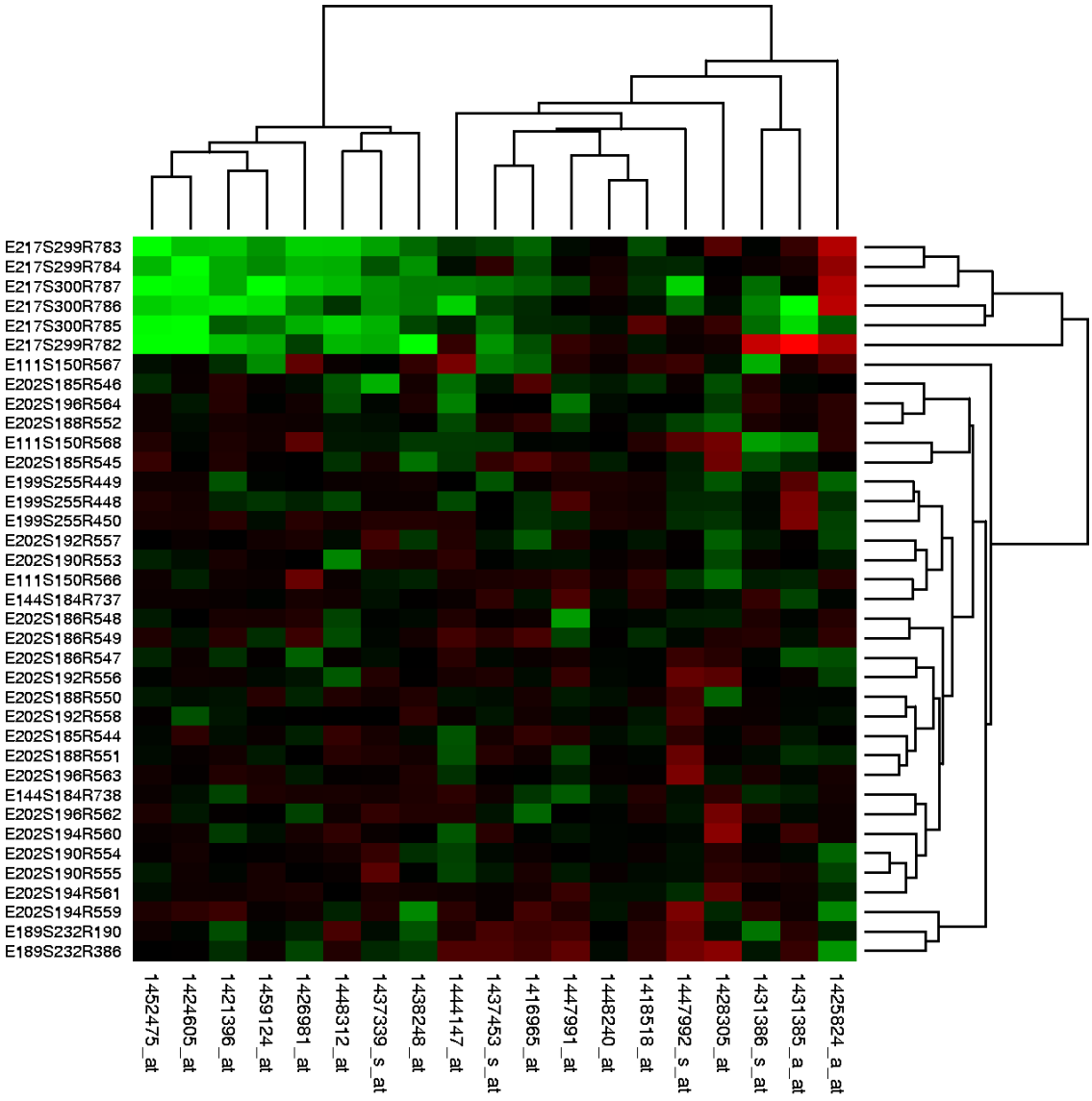

Рис. 1.1 — Пример тепловой карты, построенной на данных ДНК-микрочипа с помощью приложения Cluster от Michael Eisen [1]. Зеленый цвет соответствует большим значениям – красный меньшим.

1.2 Существующие решения для анализа экспрессии генов

Задача создания программного обеспечения для анализа экспрессии генов не нова, поэтому можно рассмотреть преимущества и недостатки некоторых уже существующих продуктов или подходов.

1.2.1 Язык R и Bioconductor

Язык R и библиотека Bioconductor [2] являются одними из самых популярных инструментов в биоинформатике. Почти все существующие алгоритмы по анализу экспрессии генов, так или иначе реализованы в библиотеке Bioconductor, что и делает её универсальным инструментом в области биоинформатики. Недостатки подхода заключаются в том, что для совершения анализа необходимо собственноручно писать код на языке R, что может быть попросту невозможно для неимеющих соответствующей подготовки биологов.

1.2.2 GENE-E

GENE-E – “платформа для визуализации данных и анализа, спроектированная для визуального анализа данных” [3]. GENE-E – хороший инструмент для анализа матричных данных, в котором можно совершать некоторые из методов анализа экспрессии генов. Из недостатков GENE-E можно отметить несколько:

а) GENE-E является “коробочным” решением: для того, чтобы им воспользоваться необходимо, чтобы он был предустановлен на вашу машину.
б) GENE-E несколько ограничен в воспроводимости исследования. В GENE-E пользователь может составить отчет, в котором будет указано большое число различных значений параметров необходимый для повтора анализа.

1.2.3 GenePattern

GenePattern – платформа для исследований в области биоинформатики [4]. GenePattern – хороший инструмент в области биоинформатики, также поддерживающий и многие алгоритмы для анализа экспрессии генов. GenePattern поддерживает воспроизводимость исследования. Однако стоит отмтетить большие недостатки GenePattern:

а) Сложность и громоздкость интерфейса – пользователю, который хочет совершить анализ, сначала необходимо найти модуль, который он хочет использовать, в большом списке модулей, затем выбрать большое число аргументов модуля, затем совершить анализ.
б) Разделение визуализации и результатов анализа – совершив в GenePattern какой-либо анализ, вы получите файл – результат анализа. Затем этот файл можно передать в модуль визуализации.

1.3 Инструменты, которые могут быть использованы.

1.3.1 Язык R и Bioconductor

Как уже было упомянуто в обзоре существующих решений и подходов, R и Bioconductor – хорошие инструменты в области анализа экспрессии генов и могут быть использованы в работе как инструменты для совершения анализа.

1.3.2 Фреймворк Shiny

Фреймворк Shiny – веб-фреймворк языка R, для создания веб-приложений, позволяющих проводить анализ в интернете [5]. Отличительной чертой фреймворка является простота использования, не требующая от разработчика пишущего анализ на языке R знаний о веб-разработке. Shiny является хорошим фреймворком для одностраничных приложений со статичной структурой страницы.

1.3.3 OpenCPU

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, использует модель “клиент-сервер”, здесь и далее под HTTP будет подразумеваться протокол версии HTTP/1.1 [6]

HTTP API – интерфейс прикладного программирование, набор процедур и функций которые может нам предоставить сервис/программа. В данном случае вызов процедур и возвращение результата выполняется посредством протокола HTTP.

RPC (Remote Procedure Call) – технология, позволяющая программе выполнять вызов функций или процедур на удалённой машине.

Веб-сервер – программа, обрабатывающая запросы HTTP для распространения информации в интернете.

OpenCPU – это серверное решение предоставляющее RPC посредством HTTP API для вызова функций и получение объектов языка R [7].

В этом разделе мы рассмотрим наиболее важные части API, которые предоставляет OpenCPU. Входной точкой является адрес “/ocpu/”. OpenCPU поддерживает два основных HTTP-метода GET и POST. В зависимости от того, на что указывает ссылка (на файл или на объект языка R) метод GET совершает чтение файла или объекта, а метод POST выполняет скрипт внутри файла или выполняет метод, описываемый объектом языка R. Примеры работы с API можно рассмотрены на таблице 1.1.

Таблица 1.1 — Основные части API OpenCPU

Метод	На что указывает ссылка	Действие	Аргументы	Пример
GET	Файл	Прочитать файл	-	GET /ocpu/library/MASS/NEWS
GET	Объект R	Прочитать объект	Формат для представления объекта	GET /ocpu/library/MASS/R/cats/json
POST	Файл	Выполнить скрипт	Аргументы запуска скрипта	POST /ocpu/library/MASS/scripts/ch01.R
POST	Объект R	Выполнить функцию	Аргументы функции	POST /ocpu/library/stats/R/rnorm

В понимании API объект R может быть, как объектом языка R, так и функцией.

Также, важной частью API являются сессии. Каждый раз, когда OpenCPU вызывает функцию или запускает скрипт, OpenCPU создает для этого отдельную сессию. В этой сессии лежат все сгенерированные файлы, результат выполнения функции (скрипта), графики, а также вывод консоли. Каждая сессия обладает уникальным ключом (далее ключ сессии). Одно из важнейших свойств ключа сессии заключается в том, что его можно передать в качестве аргумента API для вызова функции, и тогда вместо аргумента функции R будет использован результат выполнения сессии, пример рассмотрен на листинге 1.1.

```
POST /ocpu/library/stats/R/rnorm n=15
POST /ocpu/library/base/R/mean x=x01d3f004fc
```

Листинг 1.1 – Пример передачи ключа сессии в качестве аргумента API

OpenCPU также предоставляет API, для работы с сессией, наиболее важные пункты представлены в таблице 1.2.

Таблица 1.2 — Основные части API OpenCPU для работы с сессией

Адрес	Действие/Содержимое
/ocpu/tmp/key/	Перечисляет данные всей сессии
/ocpu/tmp/key/R	Содержит данные о всех объектах сессии
/ocpu/tmp/key/R/.val	Объект R - результат выполнения функции или скрипта
/ocpu/tmp/key/files/*	Содержит все файлы в сессии
/ocpu/tmp/key/graphics/	Содержит все графики сессии
/ocpu/tmp/key/source	Содержит выполненный код сессии
/ocpu/tmp/key/stdout	Содержит стандартный вывод функции/скрипта

1.3.4 Фреймворк Django

Фреймворк Django – высокоуровневый веб-фреймворк с открытым исходным кодом для написания динамических веб-приложений, написан на языке Python [8]. Фреймворк Django может использоваться в качестве основного веб-фреймворка в работе. Отличительные положительные качества Django:

а) Высокая скорость разработки, которая достигается засчет большой библиотеки языка и хорошего проектирования самого фреймворка.
б) Фреймворк поддерживает последние разработки в области веб и защищен от уже известных веб-уязвимостей.

1.3.5 Веб-сервер Nginx

Веб-сервер Nginx – простой, низкоуровневый HTTP-сервер. Основным предзназначением Nginx является работа со статическими страницами или использование в качестве прокси-сервера перед динамическими сайтами. Отличительными чертами Nginx являются скорость работы, распределение нагрузки и отказоустойчивость.

1.3.6 HTML

HyperText Markup Language – (далее просто HTML), язык разметки для создания веб-страниц. Язык HTML является де-факто стандартом веб-разработки для разметки веб-страниц. Здесь и далее под HTML мы понимаем стандарт HMTL5 [9].

1.3.7 JavaScript

JavaScript – прототипно-ориентированный скриптовый язык программирования. Используется в браузерах для выполнения скриптов для придания интерактивности веб-страницами.

AngularJS – JavaScript-фреймворк с открытым исходным кодом [10] для разработки браузерных приложений и клиентской логики. Примечательной особенностью данного фреймворка является двустороннее связывание данных, что означает, что состояние модели и представления идентичны.

1.4 Постановка задачи

После рассмотрения уже существующих наработок в области анализа экспрессии генов, мы можем поставить требования к сервису.

Цель работы можно разбить на три важных пункта:

а) Проектирование сервиса для анализа экспрессии генов;

б) Реализация сервиса;

в) Запуск сервиса для общего пользования.

Также укажем и подробно распишем требования к сервису.

1.4.1 Воспроизводимость исследования

Воспроизводимость исследования – возможность пользователя или другого лица провести это же исследование целиком с тем же результатом. В нашем случае мы будем достигать воспроизводимости, предоставляя возможность пользователю на любом этапе своего исследования получить исполняемый код, воспроизводящий текущее исследование. Воспроизводимость исследования – важный аспект в биоинформатике: специалисты в этой области часто проводят одни и те же исследования с разными входными данными или аргументами.

1.4.2 Доступность

Важной частью нашего сервиса является доступность. Пользователь должен иметь возможность совершить анализ в любой момент времени и потенциально с любой точки планеты. Наиболее удачным решением в плане доступности является реализация сервиса для анализа в качестве веб-сервиса доступного по определенному веб-адресу.

1.4.3 Возможность расширения сервиса новыми способами анализа

Биоинформатика и генетика – стремительно развивающиеся науки, специалисты отрасли постоянно находят новые методы для анализа, а

текущие методы подвергаются оптимизации, поэтому необходимо помнить о том, что результатом работы должен быть сервис, в котором не должно составлять большой проблемы реализовать новый алгоритм или метод для анализа.

1.5 Выводы по главе 1

В данной главе был приведён обзор предметной области и уже существующих решений. По результата обзора были составлены требования к сервису для анализа экспрессии генов, которые можно коротко описать тремя пунктами:

а) воспроизводимость исследования;
б) доступность;
в) возможность расширения сервиса новыми способами анализа.

ГЛАВА 2

АРХИТЕКТУРА ПРОЕКТА

2.1 R, Bioconductor и их необходимость

Так как Bioconductor – на данный момент крупнейшая библиотека по биоинформатике, то выберем R и Bioconductor в качестве инструментов для анализа генетической экспрессии.

Здесь и далее, RBlock – часть кода написанная на R с использованием Bioconductor, ответственная за анализ генетической экспрессии.

2.2 Устройство веб-сервера.

Начальная схема устройства любого веб-сервиса достаточно примитивна и приведена на рисунке 2.1: пользователь делает запрос, а сервер возвращает ему ответ.

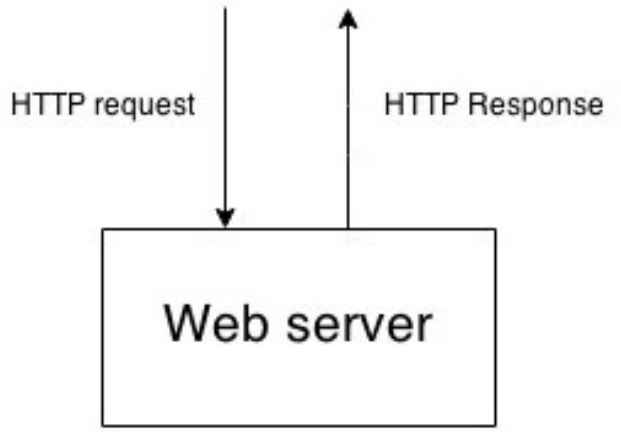

Рис. 2.1 — Схема базового веб-сервиса.

В разделе 2.1 была показана необходимость использовать R и Bioconductor в качестве инструментов для анализа генетической экспрессии. В связи с этим есть несколько подходов к реализации сервиса:

а) Написать всю серверную часть на R.
б) Написать серверную часть на отличном от R языке, и использовать R, только как язык для анализа данных.

Первую опцию тоже можно рассмотреть с двух сторон:

а) Написать свой web-framewrok на языке R с функциональностью, достаточной для совершения всех нужных нам операций.
б) Использовать готовый web-framework на R, и попытаться добиться от него нужной функциональности.

Первый из этих пунктов крайне сложен в реализации, ведь объем нужного от фреймворка функционала достаточно велико, а сам по себе язык недостаточно богат на наработки в области web-разработки.

Второй пункт стоит рассмотреть чуть подробнее. На данный момент существует один фреймворк, который покрыл хотя бы часть, требуемого от него функционала – фреймворк Shiny. Но Shiny оказался невозможным для использования в этих целях, так как он был спроектирован, как инструмент для одностраничных приложений, чаще всего с одной функциональностью, и из-за этого сложно масштабируем. Так же в нем нет поддержки сессий/пользователей (которые нужны для того, чтобы хранить исследования пользователей на сервере).

Поэтому решено было не использовать язык R для серверных целей. Тогда встаёт вопрос, как установить связь между основным веб-сервером и RBlock.

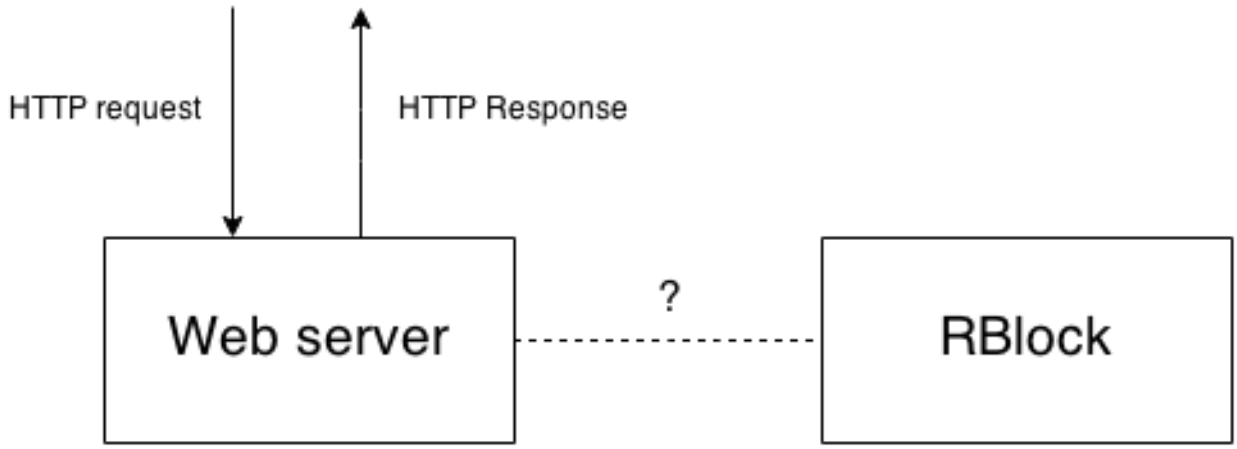

Рис. 2.2 — Вопрос взаимодействия основного веб-сервера и RBlock.

2.2.1 Использование OpenCPU

OpenCPU как было показано в разделе 1.3.3 обладает ёмким API для работы с объектами и функциями языка R, поэтому OpenCPU представляется наиболее удачным способом связать наш основным веб-сервер и RBlock.

Основной веб-сервер не будет работать с объектами языка R – с ними будет общаться сервер OpenCPU. Основной веб-сервер будет только перенаправлять запросы клиента. Сервер OpenCPU будет вызывать методы языка R, описанные в RBlock, и возвращать объекты языка R, созданные в процессе работы. Основной веб-сервер и сервер OpenCPU будут общаться между собой посредством OpenCPU API. На рисунке 2.2 рассмотрена схема решения.

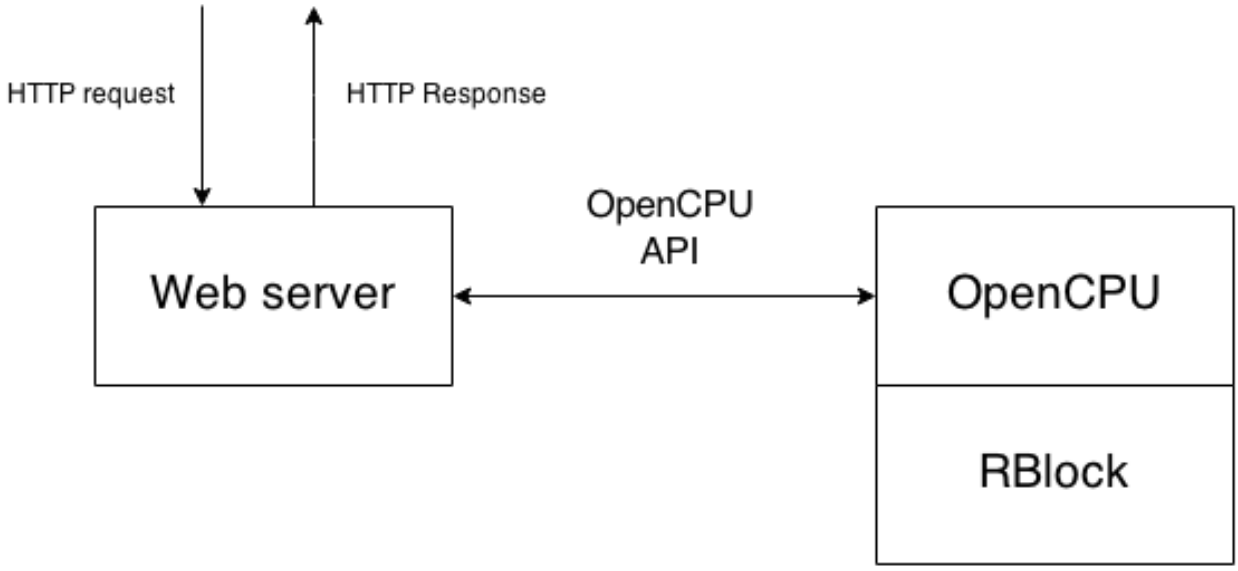

Рис. 2.3 — OpenCPU API связывает основной веб-сервер, обрабатывающий запросы клиента, и сервер OpenCPU, работающий с методами и объектами языка R из RBlock.

2.2.2 Устройство RBlock

В этой части рассмотрим разработанный формат для устройства кода на R, позволяющий поддерживать нужную функциональность. Введём абстракции Dataset и Presentation.

Абстракция Dataset хранит в себе данные и позволяет совершать над ними операции посредством методов.

Абстракция Presentation – представление Dataset в том или ином формате. Абтракция Presentation также хранит данные для представления, но не позволяет совершать над ними операции.

Здесь и далее для описания формата возвращаемых объектов языка R мы будем использовать формат JSON [11]. Мы будем делать это по двух причинам:

а) OpenCPU API может возвращать объекты языка R в формате JSON.

б) Формат JSON является удобным форматом для описания такого рода объектов.

Опишем структуру Dataset. Дескриптор Dataset – структура языка R, эквивалентная следующей структуре в формате JSON, описанной на листинге 2.1.

```
{
   "name": "Dataset name",
   "class": "Dataset class name",
   "constructor": "Name of function which returns object-descriptor of
      Dataset's constructor function",
   "methods": "Name of function which returns object-descriptor of Dataset's
      methods function"
}
```

Листинг 2.1 – Описание дескриптора Dataset в формате JSON.

Поле “name” – название Dataset. Поле “class” – строковое имя класса языка R, описывающего датасет. Поле “constructor” – единственное необязательное поле в списке, имя функции возвращающей дескриптор конструктора. Поле “methods” – имя функции возращающей дескриптор функции, ответственной за методы Dataset.

Теперь рассмотрим устройство класса языка R описывающего датасет (поле ‘class’ дескриптора). В R очень мало конструкций для поддержания шаблонов объектно-ориентированного программирования, понятие “класс” там достаточно размыто относительно других языков. Мы воспользуемся языковой структурой refClass [12] для описания класса Dataset. Интерфейс любого класса Dataset приведен на листинге 2.2. Интерфейсом он является в том понимании, что любой класс Dataset должен иметь соответствующие поля и методы.

Поле log – должно содержать в себе код, соответствующий тому, который должен быть исполнен для получения этого объекта класса Dataset. Поле name – имя Dataset, которое должен видеть конечный пользователь сервиса. Метод initialize – инициализатор класса. Метод showKnit – функция

```
DatasetInterface <- setRefClass(
   "DatasetInterface",
   fields = list(
   log="character",
   name="character",…
    #any other class fields
   ),
   methods = list(
      initialize = function(...) {...},
      showKnit = function(...) {...},…
       #any other class methods
   )
)
```

Листинг 2.2 – Интерфейс для класса Dataset, описанный с помощью языковой структуры refClass.

возвращающая HTML-представление датасета в данный момент. Это может быть любая функция возвращающая HTML, но, как показала практика, очень удобно использовать библиотеку knitr [13], для генерации HTML в среде R.

Теперь рассмотрим функцию methods и формат объекта, который она должна возвращать. Сигнатура функции приведена на листинге 2.3.

```
example_methods <- function(dataset) {
   ...
}
```

Листинг 2.3 – Сигнатура функции methods дескриптора класса Dataset.

Функция methods принимает единственный аргумент – объект класса Dataset. Это обусловлено тем, что для каждого отдельного объекта dataset мы можем выбрать различные параметры-аргументы функций, или даже автоматически конфигурировать некоторые аргументы за пользователя.

Рассмотрим структуру результата функции methods в эквивалентной ей структуре в формате JSON, описанной на листинге 2.4.

Поле method_name1 – строковый идентификатор метода (например “differential_expression”). Поле external_function_name – строка содержащее название внешней функции производящей вызов данного метода. Функция является внешней в том понимании, что она не является методом класса. Необходимость запуска внешней функции обусловлена тем, что API OpenCPU не позволяет запуск метода объекта. Формально структура внешней функции требует принимать первым аргументом объект класса Dataset, а затем аргументы из дескриптора метода, также внешняя функция

```
{
   ”method_name1”: {
            ”exec”: external_function_name,
            ”description”: method description,
            ”modificator”: true/false,
            ”args” :{
                  ”arg_name”: arg_description
               }
             },
   ”method_name2”: {...},
   ...
}
```

Листинг 2.4 – Описание дескриптора methods в формате JSON.

должна возвращать Dataset или Presentation. В связи с этим можно увидеть два подхода в написании функциональности в RBlock:

а) Писать методы внутри класса Dataset и писать внешнюю функцию, вызывающую методы класса, пример на листинге 2.5;

б) Не писать функциональность внутри класса Dataset и писать внешние функции содержащие функционал, пример на листинге 2.6.

```
external_method_example1 <– function(dataset, arg1, arg2) {
   dataset\$method_example(arg1, arg2)
}
```

Листинг 2.5 – Пример внешней функции, вызывающей метод класса.

```
external_method_example2 <– function(dataset, arg1, arg2) {
   #functionality
}
```

Листинг 2.6 – Пример внешней функции с функционалом.

Оба подходы эквивалентны, так как в дальнейшем для применения операций к объектам класса Dataset мы будем вызывать именно внешние функции.

Поле modificator – истинно если возвращаемое методом значение – объект класса Dataset, ложно – если возвращаемое значение – объект класса Presentation. Поле args – аргументы метода. Дескриптор arg_description – дескриптор аргумента метода структура эквивалентная структуре в формате JSON, описанной на листинге 2.7.

Поле name – строковый идентификатор аргумента. Поле required – истинно, если аргумент обязателен для заполнения, поле description –

```
{
   ”name”: ”argument_id”,
   ”required”: true/false,
   ”description”: ”argument description”,
   ”type”: ”TYPE”,
   ”*choices”: [value1, value2 …valueN],
   ”*default”: value_default
}
```

Листинг 2.7 – Описание дексриптора аргумента в формате JSON.

строковое описание аргумента, поле type – тип аргумента, поле type может быть элементом из списка TYPE, TYPE на данный момент поддерживает “integer”, “file”, “character”, “boolean”, “select”. Если поле type имеет значение select, то необходимо предоставить поле choices – лист возможных принимаемых значений. В других случаях поле choices игнорируется. Поле default – значение аргумента по умолчании. Все другие поля игнорируются. Суть дескриптора аргумента – это описание аргумента на стороне клиента. Каждый TYPE будет сопоставлен соответствующему типу элементу input в HTML, подробнее об этом рассмотрено в пункте 2.3.1.

Таким образом функция methods из дескриптора Dataset содержит всю необходимую информацию для применения методов и отображения их на стороне клиента. Кажущаяся на первый взгляд громоздкой структура необходима для достаточного задания любой функциональности в RBlock без необходимости менять код основного веб-сервера и код frontend-сотавляющей.

Функция constructor возвращает информацию о конструкторе класса Dataset; это тот конструктор, которым может воспользоваться клиент, в отличие от initialize – контруктора класса языка R.

Constructor имеет структуру эквивалентную структуре в формате JSON, описанной на листинге 2.8.

```
{
   ”exec”: external_function_name,
   ”args” :{
         ”arg_name”: arg_descriptor
      }
}
```

Листинг 2.8 – структура дескриптора конструктора.

Поля этой структуры эквивалентны полям дескриптора метода, за исключением того, что внешняя функция не принимает дополнительного первого аргумента, а принимает лишь аргументы, описанные дескрипторами аргументов. Построенная структура для описания Dataset, его методов и конструктора достаточна для использования.

Presentation – облегчённая версия Dataset, хранит в себе только результат применения метода, и предполагается, что к ней нельзя применить другие методы. Всё что нужно от Presentation – это интерфейс класса. Интерфейс описан на листинге 2.9.

```
PresentationInterface <– setRefClass(
   ”PresentationInterface”,
   fields = list(
   log=”character”,…
    #any other class fields
   ),
   methods = list(
      initialize = function(...) {...},
      showKnit = function(...) {...},…
       #any other class methods
   )
)
```

Листинг 2.9 – Интерфейс для класса Presentation, описанный с помощью языковой структуры refClass.

На построенной структуре, можно легко описать механизм взаимодействия:

а) Получить все доступные дескрипторы датасетов;
б) Для каждого дескриптора датасета получить дескриптор конструктора;
в) Построить объект-датасет;
г) Получить дескриптор методов данного объекта-датасета;
д) По необходимости применять методы.

Механизм взаимодействия продемонстрирован на рисунке 2.4.

2.2.3 Использование фреймворка Django

Для основного веб-сервера, обрабатывающего клиентские запросы будем использовать django-framework. Django-framework является универсальным инструментом для backend веб-разработки.

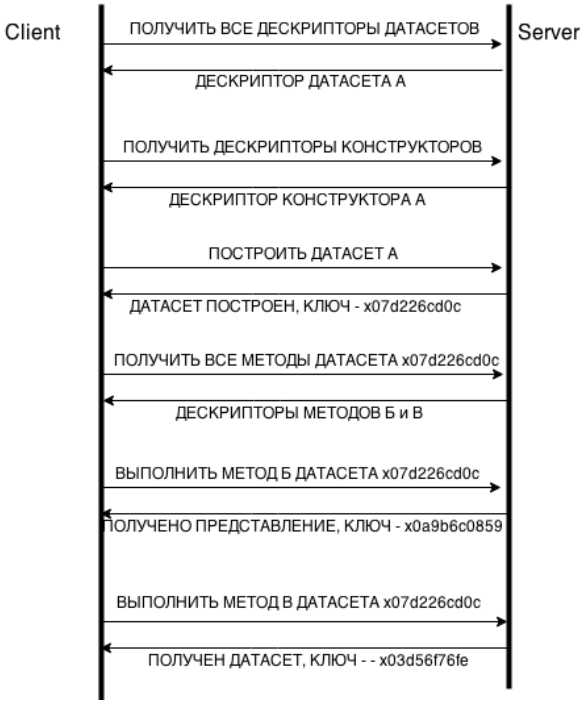

Рис. 2.4 — Механизм взаимодействия с RBlock, здесь client – это основной веб-сервер, а server – сервер OpenCPU. Стрелками изображены запросы и ответы, совершаемые с помощью OpenCPU API.

Можно было бы попробовать построить весь сервис на связке OpenCPU + frontend (html + javascript), но есть ряд вопросов, которые нельзя решить только этими средствами:

а) Ограничение доступа к серверу OpenCPU.

б) Хранение информации о сессиях с привязкой к пользователям.

OpenCPU – проект, основная задача которого RPC. Если разместить сервер публично, то любой человек, узнавший его адрес, мог проводить на нем ресурсоемкие вычисления. Мы же заинтересованы только в вычислениях, которые делает пользователь нашего сервиса. Поэтому хочется изолировать наш сервер OpenCPU от внешнего вмешательства.

Необходимо запоминать OpenCPU сессии и привязывать их к пользователям сервиса, для того, чтобы посетители могли не терять свои исследования после того, как вышли из сервиса или просто закрыли браузер.

Таким образом, Django-приложение будет служить нескольким основным целям:

а) Работа с OpenCPU API.

б) Работа с пользователями и хранение текущей сессии.

в) Работа с сессиями OpenCPU (привязка ключей сессии к пользователям).
г) Любая другая функциональность сервиса кроме функциональности для анализа данных.

2.2.4 Конечная схема backend

В этом разделе на рисунке 2.5 приведена схема, которая содержит в себе все аспекты устройства backend-части нашего сервиса. Nginx в данной схеме служит не только для переадресации запросов на django-приложение, а также для того, чтобы отдавать статические элементы-файлы нашего сервиса, такие как таблицы стилей css и javascript-файлы.

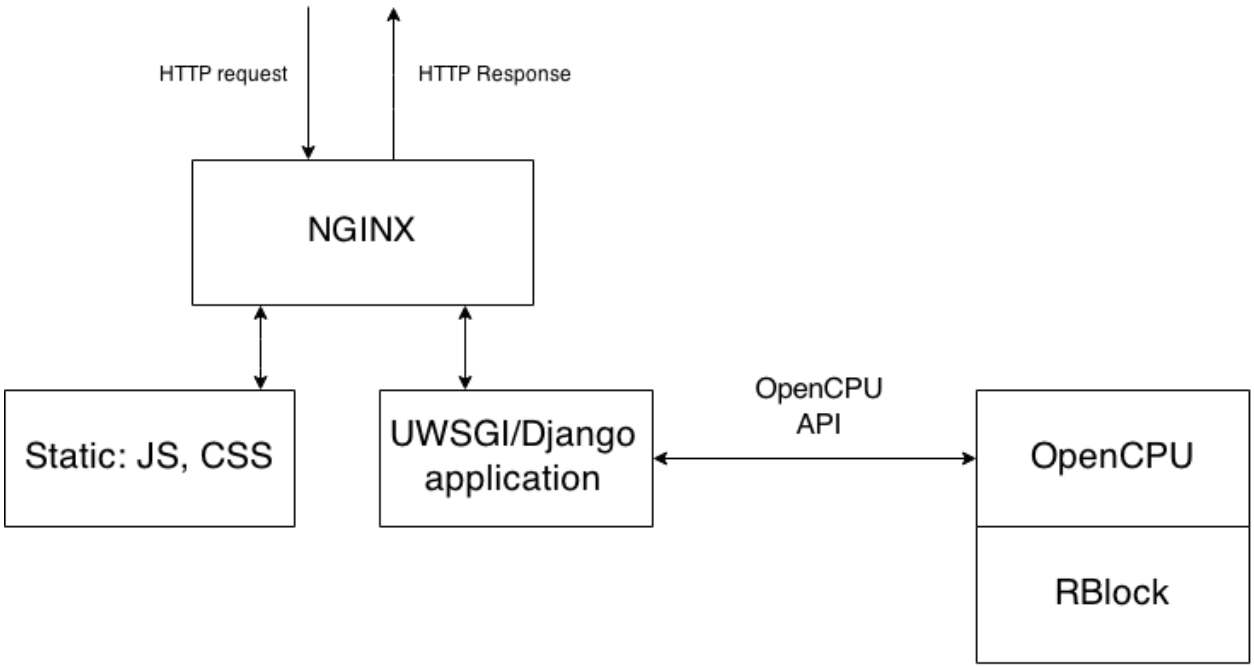

Рис. 2.5 — Backend-схема проекта. Входящий запрос обрабатывается NGINX, если это запрос статического файла – он просто его отдает, если нет перенаправляет запрос на Djagno-приложение. Django-приложение в случае надобности общается с сервером OpenCPU с помощью OpenCPU API.

2.3 Frontend-составляющая

Frontend нашего сервиса построен на связке HTML5 + JavaScript, используя при этом AngularJS и его преимущества. AgnularJS является

удобным фреймворком для одностраничных приложений. Несмотря на то, что наш сервис не является одностраничным, существует главная страница, где пользователь может создавать датасеты и совершать над ними операции. Именно в разработке этой страницы мы будет использовать AngularJS.

Определим основные функции, которые нам понадобятся на этой странице:

а) Функция получения датасетов и их конструкторов. Эта функция запускается каждый раз, при загрузке страницы.
б) Функция запуска конструктора и добавления датасета-результата в общий список датасетов.
в) Функция применения метода датасета и добавления датасета-результата в общий список датасетов или результата-представления в список результатов.

Двухстороннее связывание данных помогает не писать много кода для добавления/удаления HTML-элемнтов при добавлении датасетов.

2.3.1 Типы input-элементов HTML

Стоит также уделить отдельное внимание какие элементы из TYPE (дескриптора аргумента) соответствуют каким input-элементам HMTL:

Таблица 2.1 — Соответствие типов из TYPE элементам HTML.

TYPE	Html element
file	<input type=”file” ... >
integer	<input type=”number” ...>
character	<input type=”text” ...>
boolean	<input type=”checkbox” ...>
select	<select ...> <option ...> ... </select>

2.4 Выводы по главе 2

В данной главе, используя средства, описанные в обзоре, был спроектирован веб-сервис, обладающий нужной нам функциональностью. Были спроектированы и описаны такие компоненты:

а) Rblock и интерфейсы внутри него.
б) Основной веб-сервер, которым является Django-приложение.
в) NGINX – веб-сервер для взаимодействия основного веб-сервера, и отдачи статических файлов.
г) Frontend-составляющая.

Также было в данной главе было описано взаимодействие этих компонент.

ГЛАВА 3

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ И РЕЗУЛЬТАТЫ

3.1 Реализация RBlock

На данный момент RBlock представляет из себя один файл, содержащий R-код. Данный файл оборачивается в пакет, который можно установить стандартными средствами языка R. Укажем, какие именно датасеты, алгоритмы и представления были реализованы.

Кроме того были реализованы некоторые примитивы для поддержания валидности логов исполнения, для того, чтобы возвращать корректный исполняемый код.

3.1.1 Gene Expression Dataset

Прежде всего в ходе работы был реализован датасет для хранения данных о генетической экспрессии. Части реализации которые мы рассмотрим:

а) конструктор;
б) построение тепловой карты;
в) построение дифференциальной экспрессии;
г) метод showKnit.

Конструктор принимает аргументами два файла:

а) файл содержащий данные экспресии в формате tsv;
б) файл содержащий аннотацию в формате tsv.

Данные экспрессии представляют из себя таблицу, в которой строки соответствуют строковым или числовым идентификаторам генов, а столбцы соответствуют идентификаторам образцов. Элементы таблицы – числовые характеристики экспрессии.

Файл аннотации также представляет из себя таблицу, первый столбец которой – образец (sample), а остальные столбцы именуются специалистом и по сути являются именами отдельных аннотаций. Аннотация по своей сути есть разбиение всего множества образцов на меньшие множества образцов схожими друг с другом некоторыми общими условиями.

Конструктор проверяет являются ли данные в файле логарифмированными/нормализованными и при необходимости логарифмирует/нормализует данные. На данном этапе аннотации просто прикрепляются к датасету. Метод возвращает объект класса GeneExpressionDataset.

Метод построения тепловой карты принимает три аргумента:

а) Название аннотации (из файла аннотации), которую использовать при отображении датасета;
б) Число n – число генов, которые будут отображены на тепловой карте;
в) Число k – число кластеров в кластеризации k-means.

Данный метод подготавливает данные для построения тепловой карты. Он выбирает n строк таблицы экспрессии с максимальной медианой, затем кластеризует их k-мерами и возвращает объект класса-представления HeatMap.

Метод построения дифференциальной экспрессии принимает три аргумента:

а) Выбранная аннотация;
б) Первый элемент множества для сравнения из множества выбранной аннотации;
в) Второй элемент множества для сравнения из множества выбранной аннотации.

Данный метод строит датасет дифференциальной экспрессии. Здесь используется библиотекой limma (из библиотеки Bioconductor) для построения дифференциальной экспрессии генов, сравнивая два разных набора образцов. Данный метод возвращает объект класса DifferentialExpressionDataSet.

Метод showKnit возвращает html-представление датасета, содержащее:

а) Количество строк;
б) Идентификаторы образцов;
в) Был ли датасет логарифмирован/нормализован.

3.1.2 HeatMap Presentation

Метод showKnit этого представления возвращает html-представление тепловой карты. В нашем случае это html содержащий изображение тепловой карты с выбранной аннотацией. На рисунке 3.1 продемонстрирована тепловая карта – результат выполнения метода showKnit.

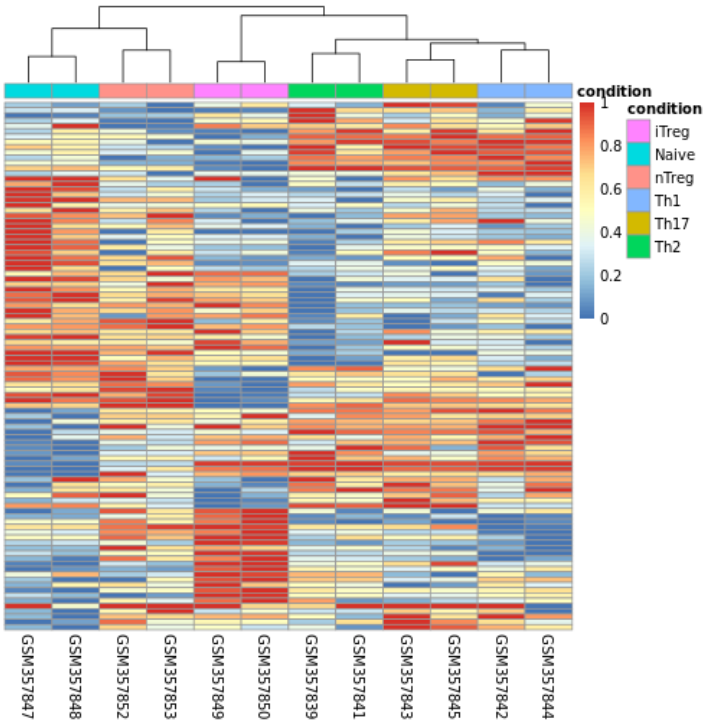

Рис. 3.1 — Пример построенного представления класса HeatMap для GeneExpressionDataset.

3.1.3 Differential Expression Dataset

В ходе работы был реализован датасет для хранения данных о дифференциальной экспрессии. Методы которые мы рассмотрим:

а) Посмотреть верхних значений таблицы;
б) Добавить к таблице имена в соответствии с базами данных Entrez (The Entrez Global Query Cross-Database Search System – поисковая система, позволяющая проивзодить поиск внутри баз данных National Center for Biotechnological Information – национальный центр биотехнологической информации США);
в) Метод showKnit.

Метод просмотра верхних значений таблицы принимает два аргумента:

а) Строковый параметр определяющий, что мы ищем (максимум или минимум) и в каком столбце таблицы дифференциальной экспрессии, например параметр “-P.value” означает брать первые значения из таблицы, отсортированной по убыванию значения P.Value, а “+B” означает брать первые значения из таблицы, отсортированной по возрастанию значения B-статистики анализа дифференциальной экспресии;
б) Число n – число верхних/нижних строк таблицы.

Данный метод ищет n максимальных/минимальных по значению выбранного столбца строк таблицы дифференциальной экспрессии и возвращает объект класса Stat.

Метод добавления к таблице имён в соответствии с Entrez не принимает аргументов и возвращает новый объект класса DifferentialExpressionDataset с новым столбцом symbol, со строковыми представлениями имён генов, соответствующих Entrez.

Метод showKnit не принимает аргументов и возвращает html-представление первых 20 строк таблицы, отсортированной по B-статистике анализа дифференциальной экспресии.

Реализация приведена в приложении А на листинге.

3.1.4 Stat Presentation

Единственный реализованный метод этого представления – метод showKnit, он возвращает html-представление любого примитива языка R.

3.1.5 Поддержание корректности log

Поддержание корректности поля log интерфейсов Dataset и Presentation является важной составляющей реализации этих интерфейсов. В во всех реализованных классах мы применили следующий подход: сначала генерируется код, необходимый для создания датасета/представления, затем это код исполняется. Этот же самый код дописывается в поле log. Таким

образом достигается точное соответствие между реальными объектами языка R и исполняемым кодом, которым может скачать пользователь сервиса. Таким образом достигается воспроизводимость исследования.

3.2 Реализация Django-приложения

Django-приложение на данный момент состоит из следующих утилит для работы с API OpenCPU:

- утилиты для создания запроса
- утилиты для разбора ответа
- утилиты для работы с файлами внутри OpenCPU-сессий

Реализованная возможность загрузки файлов из OpenCPU позволяет легко получить доступ к файлам из интерфейса пользователя. Это удобно, потому что не все результаты операций над датасетами можно легко и удобно расположить на странице. Например, таблицы хранящаяся в датасете дифференциальной экспрессии достаточно велика, поэтому показываются только верхние 20 значений, однако можно скачать файл, содержащий дифференциальную экспрессию целиком.

3.3 Реализация Frontend и пользовательского интерфейса

Задачи, которые ставились перед пользовательским интерфейсом на данном этапе: предоставление доступа к функциональности и удобство пользования. Для удобства пользования рабочая зона главной страницы сервиса разделена на два блока: в одном можно выбрать датасет для работы или выбрать создание датасета, во втором пользователь может работать над текущим датасетом (в том числе просматривая представления – результаты работы над текущим датасетом) или создать датасет.

Второй блок также состоит из некоторых составляющих:

а) Отображение showKnit текущего датасета.
б) Блок с загрузкой файлов сессии (если он присутствуют).
в) Блок с кодом для получения текущего датасета (изначально скрыт, можно раскрыть кнопкой).

г) Блок предоставляющий возможность совершений операций над датасетом.

Интерфейс и его блоки можно посмотреть в приложении на рисунке в приложении Б.

3.3.1 Использование JavaScript

На языке Javascript была реализована большая часть клиентской логики:

а) Прежде всего, был реализован указанный на рисунке 2.4 механизм взаимодействия.

б) Получаемые в формате JSON-данные сериализуются в JavaScript-объекты, которые с помощью двойного связывания данных отображаются в интерфейсе пользователя.

Также JavaScript, в частности библиотека highlight.js [14], был использован для подсветки синтаксиса исходного R-кода, получаемого от сервера.

3.4 Остальные важные детали реализации

3.4.1 Конфигурация аргументов методов

Как указывалось в пункте 2.2.2 данной работы, мы оставили возможность при реализации интерфейса указывать аргументы по умолчанию или автоматически конфигурировать некоторые аргументы методов за пользователя. В реализации данных интерфейсов были указаны значения по умолчанию для числовых характеристик метода построения тепловой карты.

Но также стоит привести пример того, как была использована возможность конфигурации аргумента. В функции differential_expression, строящей датасет дифференциальной экспрессии, мы предлагаем пользователю аннотацию по умолчанию из соображений “интересности” данной аннотации в контексте датасета экспрессии генов. Мы рассматриваем иерархическую кластеризацию образцов (столбцы в таблице экспрессии

генов) и проверяем, что каждое подмножество образцов из аннотации целиком является поддеревом иерархической кластеризации, тогда можно говорить, что данная аннотация “интересна” и предложить её пользователю в качестве аннотации по умолчанию в методе построения дифференциальной экспрессии.

3.4.2 Хранение сессий OpenCPU

На данный момент сложно определить оптимальную политику для определения времени хранения сессии OpenCPU на сервере, ведь проект находится в стадии альфа-тестирования. Поэтому текущее время хранения любой сессии на сервере установлено равным 48 часам: этого времени достаточно пользователю, чтобы поработать с датасетом и вернуться к нему на следующий день. В дальнейшем будут выбрана более гибкая политика для хранения сессий OpenCPU.

3.5 Выводы по главе 3

В данной главе были рассмотрены основные детали реализации:

а) реализация RBlock, ответственного за анализ эекспрессии генов;
б) реализация Django-приложения;
в) реализация frontend-составляющей и пользовательского интерфейса;
г) реализация остальных важных деталей: время жизни сессии OpenCPU и автоматическая конфигурация аргументов методов анализа.

ЗАКЛЮЧЕНИЕ

В ходе данной работы был спроектирован, реализован и запущен в режиме альфа-тестирования веб-сервис для вопспроизводимого биоинформатического анализа данных экспрессии.

Полученный сервис удовлетворяет всем предъявленным к нему требованиям, а именно:

а) Сервис предоставляет возможность проводить исследования в области анализа экспрессии генов.

б) Сервис поддерживает воспроизводимость исследования: на любой стадии своего исследования пользвотель может скачать исполняемый R-код, эквивалентный коду выполненному в сервисе.

в) Функционал сервиса легко расширять: для этого достаточно реализовать DatasetInterface и другие регламенты принятые для кода внутри RBlock, при этом корректировка django- и frontend-составляющих проекта не требуется.

На данный момент сервис доступен по адресу http://genome.ifmo.ru/projectx/ .

ПРИЛОЖЕНИЕ А

```
DifferentialExpression <- setRefClass(
   "DifferentialExpression",
   fields = list(
      diff_exp = "data.frame",
      log = "character",
      name = "character"
   ),
   methods = list(
      initialize = function(diff_exp, log, name) {
         diff_exp <<- diff_exp
         log <<- log
         name <<- name
      },
      showKnit = function() {
         library(knitr)
         opts_knit\$set(width=120)
         strings = c(
         "<!--begin.rcode",
         "head(diff_exp, 20)",
         "end.rcode-->"
         )
         string = paste(strings, collapse="\n")
         val = knit2html(text=string, fragment.only=TRUE)
         val
      },
      topValues = function(n, order_property) {
         to_exec = c(
            definition(n), definition(order_property),
            "stat = head(diff_exp[with(diff_exp, order(eval(parse(text=order_property)))), ], n)"
         )
         code = paste(to_exec, collapse="\n")
         eval(parse(text=to_exec))
         Stat(stat, add_log(log, code))

      },
      getEntrezNames = function() {
         to_exec = c(
            "library(data.table)",
            "load(system.file(\"reflink.rda\", package=\"GeneExprDataSet\"))",
            "diff_exp$symbol <- reflink[match(rownames(diff_exp), reflink$Entrez), \"symbol\"]"
         )
         code = paste(to_exec, collapse="\n")
         eval(parse(text=code))
         DifferentialExpression(diff_exp, add_log(log, code), paste(name, "Entrez"))
      },
      perform = function() {
         head(diff_exp, 20)
      }
   )
)
```

Листинг – частная реализация интерфейса Dataset – класс DifferentialExpression.

ПРИЛОЖЕНИЕ Б

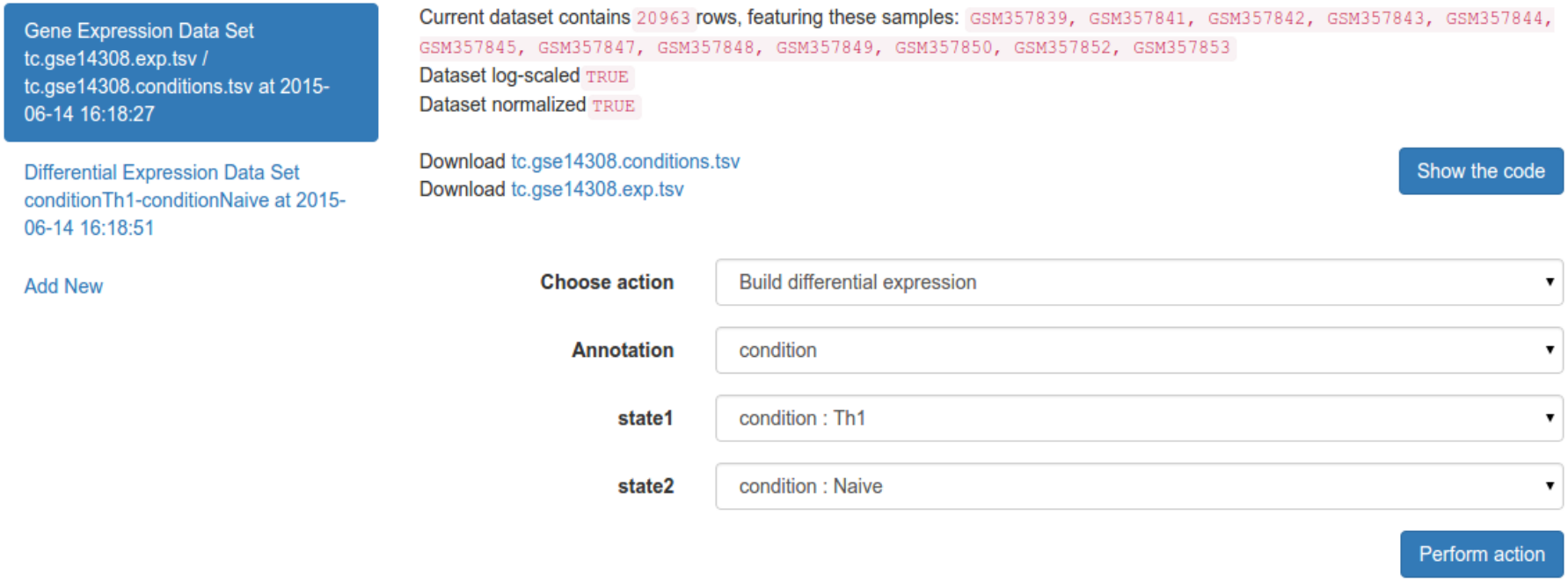

Рис. 3.1 — Рисунок – Фрагмент реализованного интерфейса. Слева расположен блок выбора текущего датасета и кнопка создания нового датасета. Остальная зона является рабочей: здесь приведена основная информация о датасете, возможность загрузить файлы текущей сессии, возможность просмотра исполняемого R-кода, а также блок предоставляющий возможность выполнения методов над датасетом.

СПИСОК ИСТОЧНИКОВ

1. Heat map. [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Heat_map.
2. Arora Sonali, Carlson Marc, Hayden Nate [и др.]. Bioconductor is an open source, open development software project to provide tools for the analysis and comprehension of high-throughput genomic data. [Электронный ресурс]. URL: http://bioconductor.org/.
3. Gould Joshua. GENE-E is a matrix visualization and analysis platform designed to support visual data exploration. [Электронный ресурс]. URL: http://www.broadinstitute.org/cancer/software/GENE-E/.
4. GenePattern 2.0 / Michael Reich, Ted Liefeld, Joshua Gould [и др.] // Nature genetics. 2006. Т. 38. С. 500–501.
5. RStudio. Shiny A web application framework for R. [Электронный ресурс]. URL: http://shiny.rstudio.com/.
6. Hypertext Transfer Protocol – HTTP/1.1: Tech. Rep.: / Fielding R., Gettys J., Mogul J. [и др.]: IETF, 1999. июнь. URL: http://www.rfc-editor.org/info/rfc2616.
7. Ooms Jeroen. OpenCPU is a system for embedded scientific computing and reproducible research. [Электронный ресурс]. URL: https://www.opencpu.org/.
8. Foundation Django Software. Django. The web framework for perfectionists with deadlines. [Электронный ресурс]. URL: https://www.djangoproject.com/start/overview/.
9. HTML5: Tech. Rep.: / под ред. Robin Berjon, Travis Leithead, Erika Doyle Navara [и др.]: W3C, 2012. дек. URL: http://www.w3.org/TR/html5.
10. LLC Brat Tech, Google, community. AngularJS — Superheroic JavaScript MVW Framework. [Электронный ресурс]. URL: https://angularjs.org/.
11. The JavaScript Object Notation (JSON) Data Interchange Format: Tech. Rep.: / под ред. Tim Bray: IETF. URL: https://tools.ietf.org/html/rfc7159.
12. Objects With Fields Treated by Reference (OOP-style): Tech. Rep.: / под ред. John Chambers. URL: https://stat.ethz.ch/R-manual/R-devel/library/methods/html/refClass.html.
13. Xie Yihui. knitr. Elegant, flexible and fast dynamic report generation with R. [Электронный ресурс]. URL: http://yihui.name/knitr/.
14. Sagalaev Ivan, Hull Jeremy, Efimov Oleg. highlight.js Syntax highlighting for the Web. [Электронный ресурс]. URL: https://highlightjs.org/.