ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Биоинформатика образована на стыке биологических направлений и информатики, как реакция на всё увеличивающий объем данных, требующих сложных, быстрых и качественных алгоритмов для обработки и анализа. Разумеется, биоинформатикой занимаются как, непосредственно, информатики, которые обладают навыками программирования и могут реализовывать алгоритмы самостоятельно, так и биологи, которые отлично могут интерпретировать результаты работы алгоритмов и сами данные, но не имеют соответствующей подготовки для использования этих методов. Исследователям для более продуктивной работы нужны удобные и интуитивно понятные инструменты для анализа данных, которые бы хорошо покрывали все их потребности в реализованных методах и алгоритмах. На данный момент, таких инструментов достаточно мало, а в тех, что есть, неполноценный функционал. Таким образом, целью данной работы является разработка инструмента для полноценного анализа биологических данных.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Биоинформатика

Биоинформатика — наука, объединяющая в себе методы прикладной математики, статистики, информатики для создания новых методов и алгоритмов для анализа разного рода биологических данных.

Биоинформатика занимается биохимией, биофизикой, экологией и многими другими областями биологии. Однако фокус в данной работе будет сосредоточен на конкретную задачу биоинформатики — анализ экспрессии генов.

1.1.1. Анализ экспрессии генов

Экспрессия генов — процесс преобразования наследственной информации от гена (в виде последовательности нуклеотидов ДНК) в функциональный продукт (РНК или белок).

Анализ экспрессии генов позволяет выяснить, как ведет себя каждый отдельный ген в разных условиях, тканях или организмах.

Экспрессия гена в образце характеризуется вещественным числом, которое также можно назвать некоторой мерой активности гена в данных условиях.

1.1.2. Используемые методы

Как было сказано ранее, биоинформатика использует в себе математику, информатику и статистику. Соответственно, задача анализа экспрессии генов сводится к исследованию путем статистических методов и алгоритмов числовой двумерной матрицы, где в виде вещественных чисел демонстрируется активность каждого гена в каждом образце. Пример такой матрицы можно увидеть в таблице 1.

Таблица 1 – Срез матрицы GSE14308. Строки матрицы соответствуют генам, столбцы — образцам.

	GSM357839	GSM357841	GSM357842	GSM357843	GSM357844
Rps29	16.32	16.30	16.25	16.32	16.30
Rpl13a	16.27	16.23	16.32	16.30	16.27
Rps3a1	16.23	16.19	16.30	16.25	16.25
Rpl38	16.21	16.25	16.27	16.27	16.21
Tmsb4x	16.30	16.32	16.23	16.21	16.32

Одним из первоочередных методов, применяемых для анализа экспрессии, является *визуальный анализ*. Числовая матрица представляется в виде *тепловой карты*, где цветом показана активность каждого гена.

На рисунке 1 можно увидеть визуализацию матрицы экспрессии из таблицы 1 в виде тепловой карты.

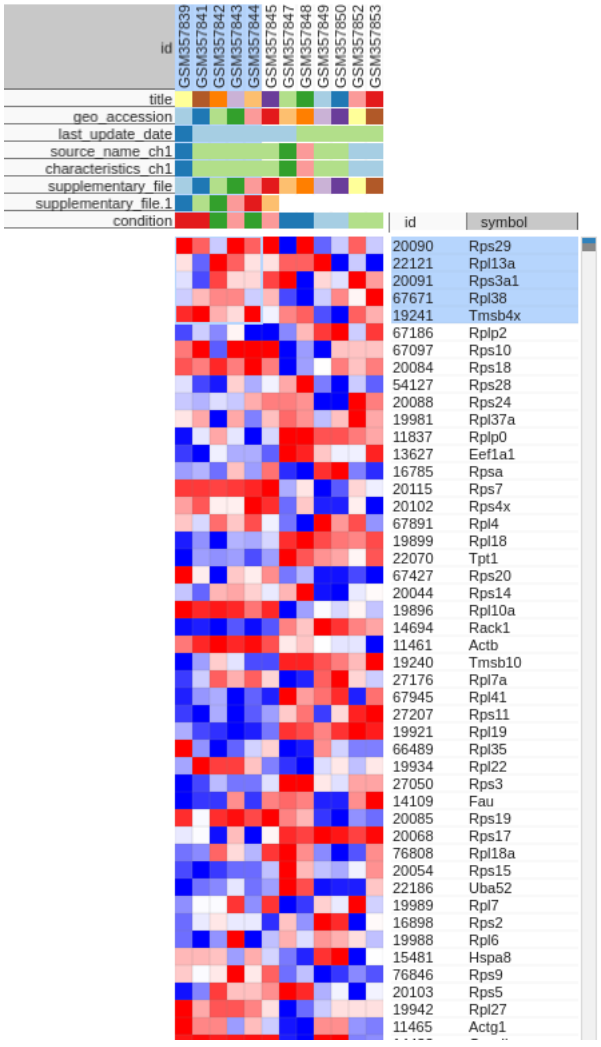

Рисунок 1 – Визуализация матрицы GSE14308 в виде тепловой карты в веб-приложении Morpheus

Также к основным методам анализа относятся:

— кластеризация:
  – иерархическая: метод упорядования данных таким образом, чтобы их можно было визуализировать в виде дерева (дендрограммы);
  – вероятностная: метод разбиения данных на несколько групп (кластеров);

— дифференциальная экспрессия: метод, позволяющий сравнивать поведение генов в разных условиях и искать закономерности;
— метод главных компонент: метод для уменьшения размерности данных с наименьшей потерей информации.

1.2. Существующие решения для анализа экспрессии генов

1.2.1. R/Bioconductor

R — язык программирования для статистического анализа данных и работы с графикой [1].

Bioconductor — библиотека, содержащая в себе множество реализаций биоинформатических алгоритмов и методов обработки биологических данных на *R*. Она постоянно обновляется, пополняется новыми библиотеками, модерируется сообществом [2]. *R* и *Bioconductor* очень популярны в биоинформатической среде ввиду предоставляемых возможностей.

Однако для качественного и полноценного анализа с помощью этих инструментов, нужно иметь навыки программирования на *R*, что весьма неудобно для исследователей биологических специальностей.

1.2.2. GENE-E

GENE-E — Платформа для анализа данных и визуального исследования данных, созданная на *Java* и *R* [3]. Содержит в себе множество полезных для исследования инструментов: тепловые карты, кластеризацию, фильтрацию, построение графиков и т.д. Позволяет исследовать любые данные в виде матрицы. К тому же, содержит дополнительные инструменты для данных генетической экспрессии.

Недостатки:

— чтобы использовать, необходимо устанавливать на свой компьютер;
— поддержка данного приложения прекратилась в связи с созданием *morpheus.js* [4];
— не имеет открытого исходного кода, а только *API* для взаимодействия и создания новых приложений на его основе.

1.2.3. morpheus.js

Morpheus.js — веб-приложение для визуализации и анализа матриц от создателя *GENE-E* [4]. В отличие от предшественника, создано на *JavaScript* и с открытым исходным кодом. Удобно для использования исследователями без навыков программирования и так же, как и *GENE-E*, применимо к любым матрицам.

Недостатки:

— ограниченный набор функций, которых недостаточно для полноценного анализа;
— для расширения биоинформатическими алгоритмами, не прибегая к дополнительным инструментам, требуется реализовывать их заново на *JavaScript*;

1.2.4. ProjectX

ProjectX — веб-приложение, созданное в рамках выпускной квалификационной работы [**projectx**], которое соединяло в себе возможности, предоставляемые *API GENE-E* и методы из библиотеки *Bioconductor* с помощью *OpenCPU*. Реализовано было данное приложение во фреймворке *Django*. Преимуществом данного приложения перед *GENE-E* была вопроизводимость исследований (на каждом этапе пользователь мог скачать исполняемый *R*-код, эквивалентный коду, выполненному в сервисе), также он содержал в себе большее число методов анализа и обработки данных.

Недостатки:

— проект использовал в своей базе приложение, которое больше не поддерживается разработчиками;
— работа над проектом была завершена до того, как у него появились пользователи, так что оно осталось невостребованным.

1.3. Инструменты, которые могут быть применены

1.3.1. Язык R и библиотека Bioconductor

Как было сказано ранее, *Bioconductor* полон актуальными и широко используемыми биоинформатическими алгоритмами, в том числе и для анализа экспрессии генов. Соответственно, реализовывать их заново обычно нет необходимости и можно использовать их для достижения целей этой работы.

1.3.2. JavaScript и Node.js

JavaScript — мультипарадигменный скриптовый язык программирования, широко используемый для создания веб-приложений.

Node.js — программная платформа, превращающая JavaScript в язык общего назначения, добавляя возможность взаимодействовать с устройствами ввода-вывода, использовать внешние библиотеки, написанные на других языках. В основе Node.js лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом-выводом. [5]

1.3.3. R shiny

1.3.4. OpenCPU

HTTP (*HyperText Transfer Protocol*) — протокол прикладного уровня передачи данных на основе технологии «клиент-сервер» [**http**].

HTTP API - набор процедур и функций, вызов которых и возвращение их результа осуществляется посредством протокола *HTTP*.

RPC (*Remote Procedure Call*) — класс технологий, позволяющих компьютерным программам вызывать функции или процедуры в другом адресном пространстве.

Веб-сервер — сервер, принимающий *HTTP*-запросы от клиентов, обычно веб-браузеров, и выдающий им *HTTP*-ответы.

OpenCPU — система для встроенных научных вычислений и воспроизводимых исследований, предоставляющая *HTTP API* для вызова R-функций и взаимодействия с R-объектами с помощью `POST` и `GET` запросов [6].

1.3.4.1. OpenCPU-сервер

OpenCPU-сервер можно запустить одним из следующих способов:

— однопользовательский сервер: сервер запускается из активной *R*-сессии и предназначен в основном для разработки и локального использования. Такой сервер не поддерживает параллельных запросов, так как *R*-сессии работают в одном потоке;

— облачный сервер: этот сервер можно запустить на *Ubuntu 16.04* и выше. Запуск и настройка облачного сервера осуществляется с помощью *apache* или *nginx*. В отличие от однопользовательского сер-

вера, здесь поддерживаются параллельные запросы и настройка безопасности.

1.3.4.2. OpenCPU API

Входной точкой *API* является `/ocpu/`. `GET`-запрос используется для получения некоторого ресурса, а `POST`-запрос используется для *RPC*. На таблице 2 можно увидеть более подробное описание запросов.

Таблица 2 – Запросы к *OpenCPU*-серверу, их аргументы и действия

Запрос	Действие	Аргументы	Пример
`GET`	посмотреть объект	формат представления объекта	GET /ocpu/library/MASS/R/cats/json
`POST`	вызвать функцию	аргументы функции	POST /ocpu/library/stats/R/rnorm
`GET`	прочитать файл	-	GET /ocpu/library/MASS/NEWS
`POST`	запустить скрипт	аргументы запуска скрипта	POST /ocpu/library/MASS?scripts/ch01.R

На *OpenCPU*-сервере могут быть доступны:

— библиотеки и содержащиеся в них пакеты: `/ocpu/library/pkgname/`;
— пакеты из *git*-репозиториев: `/ocpu/apps/gituser/reponame/`;
— пакеты, установленные в домашней директории *Linux*-пользователя: `/ocpu/user/username/library/pkgname/`;
— временные сессии, содержащие вывод от запуска функции или скрипта: `/ocpu/tmp/key/`.

1.3.4.3. Opencpu.js

Так как зачастую *OpenCPU* используется разработчиками для использования *R* в веб-приложениях для анализа и визуализации данных, для удобной интеграции *JavaScript* и *R* существует библиотека *opencpu.js*, которая реалазует *RPC*-вызовы по принципу *Asynchronous JavaScript and XML* (*Ajax* [7]). Таким образом запросы обрабатываются в фоновом режиме, тем самым не замедляя работу графического интерфейса и вычислений, осуществляемых на стороне клиента.

В данной библиотеке реализован класс `Session`, содержащий в себе ключ сессии, адреса на ссылки, файлы и переменные, существующие внутри сессии.

Для подключения к *R*-пакету на *OpenCPU*-сервере удобно использовать код, представленный на листинге 1. Для успешного подключения *R*-пакет должен быть предварительно установлен на *host*-машину, на которой располагается сервер.

```
ocpu.seturl("//hostname/ocpu/library/{pkgname}/R");
```

Листинг 1 – Подключение к *R*-пакету

После этого можно вызывать и запускать функции, содержащиеся в данном *R*-пакете, например, как в листинге 2.

```
var req = ocpu.rpc("function.name", arguments, callback(session) {
  \\ Handling result
});
```

Листинг 2 – Шаблон вызова *R*-функции из *JavaScript*

1.3.5. Gene Expression Omnibus

Gene Expression Omnibus (GEO) — международный публичный репозиторий, агрегирующий и распространяющий различные формы геномных данных от исследовательского сообщества [8].

GEO предоставляет устойчивую базу данных для эффективного хранения геномных данных, содержит их в качественно аннотированном формате и дает возможность удобно как добавлять новые данные для публикации, так и запрашивать интересующие данные для исследований.

В *GEO* содержатся следующие типы и форматы данных:

— информация о платформе, на которой производилось секвенирование генов (*Platform records*): `GPLxxx`;

— информация об образцах и условиях, в которых производились исследования этих образцов (*Sample records*): `GSMxxx`;

— информация о непосредственно исследованиях, полученные данные и выводы (*Series records*): `GSExxx`.

— обработанная и подготовленная для дальнейшего статистического анализа *GEO*-кураторами информация об исследованиях ((DataSet records)): `GDSxxx`.

В библиотеке *Bioconductor* есть *R*-пакет *GEOquery* для удобной загрузки данных из *GEO* [9].

1.3.6. Docker

Docker — программное обеспечение для автоматизации запуска и внедрения приложений внутри контейнеров [10].

Для дальнейшего описания данного инструмента введем несколько определений.

Образ — отдельный исполняемый пакет, включающий себя все необходимое для запуска единицы программного обеспечения, в том числе исходный код, библиотеки, переменные окружения, конфигурационные файлы. Зачастую образ построен на основе другого образа с дополнительной конфигураций. Образ компилируется по *Dockerfile*, каждая команда в котором соответствует новому слою. При перекомпиляции обновляются только те слои, которые изменились.

Контейнер — запущенный экземпляр образа. Контейнер обычно исполняется изолированно от окружения, имея доступ к файлам или портам хост-системы только при наличии соответствующей конфигурации.

В отличие от виртуальных машин, которые запускают гостевую операционную систему в каждом экземпляре, контейнеры могут разделять общее ядро, и вся информация, которая должна быть в контейнере, это исполняемый процесс и его зависимости. Исполняемые процессы из контейнеров работают как нативные процессы, и могут управляться по отдельности.

Для контроля версий и хранения образов в открытом доступе используется *Docker Hub* [11]. В этом хранилище можно как добавлять репозитории, управляемые вручную, так и поддерживать автоматические сборки (*Automated Build*), которые привязаны к репозиториям на в популярных системах контроля версий: GitHub [12] и Bitbucket [13].

1.3.7. JSON

JSON — текстовый формат представления данных, основанный на *JavaScript*, который, среди прочих достоинств, легко читается человеком [**json**].

JSON-текст представляет собой одну из следущих структур:

— пара *ключ*: *значение*, где ключом может быть только регистрозависимая строка, а в качестве значения может выступата массив, число, строка, литералы или другой *JSON*-объект;
— набор значений.

1.3.8. Protocol Buffers

Protocol Buffers (Protobuf) — гибкий, универсальный и автоматизированный механизм для сериализации структурированных данных [14].

Структура информации задается с помощью `*.proto`-файлов в форме сообщений (`message`).

ProtoBuf-формат не является человекочитаемым, данные хранятся в двоичном формате. Для десериализации и дальнейшего чтения необходим соответствующий `*.proto`-файл. Файл с форматом компилируется соответствующим выбранному языку программирования компилятором, таким образом будет создан класс доступа к данным. С помощью этого класса уже можно сериализовать/десериализовать данные, получать данные с помощью `get`/`set`-методов и пр.

1.3.9. Apache

Apache HTTP Server Project — устойчивый, полностью открытый *HTTP*-сервер [15].

1.3.10. HTML

1.4. Постановка задачи

Рассмотрев существующие решения для анализа экспрессии генов и инструментов, которые могли бы пригодиться для будущих решений, можно сформулировать цель и основные задачи данной работы

1.4.1. Цель работы

Создать веб-приложение, интегрирующее существующие возможности веб-приложения morpheus.js и методы анализа, реализованные в Bioconductor.

1.4.2. Основные задачи

а) Разработать способ взаимодействия между js-клиентом и R и встроить его в morpheus.js, чтобы избежать реализации с нуля уже существующих алгоритмов;
б) Реализовать графический интерфейс в js-клиенте и серверную реализацию в R-пакете;
в) Соединить все составляющие в одном веб-приложении phantasus;
г) Запустить веб-приложение в открытый доступ для исследователей.

1.4.3. Требования к веб-приложению phantasus

1.4.3.1. Доступность

Необходимо, чтобы веб-приложение phantasus было доступно для исследователей независимо от их местоположения и времени суток. Варианты действий:

а) Сделать его доступным по определенному веб-адресу, и тогда пользователь сможет продолжать исследования из любой точки, где есть подключение к интернету;
б) Предоставить возможность запускать приложение локально, например, с помощью Docker или внутри R.

1.4.3.2. Возможность дальнейшего расширения функционала

Как уже было сказано выше, библиотека Bioconductor постоянно обновляется и пополняется новыми алгоритмами, а исследователи находят новые методы для анализа экспрессии генов, так что необходимо не только реализовать дополнительные методы, но также отладить и описать алгоритм действий для добавления новых.

Выводы по главе 1

В данной главе были введены понятия биоинформатики и анализа экспрессии генов, обозначена цель работы и задачи, выполнение которых необходимо для ее достижения, представлены существующие решения с их достоинствами и недостатками, а также перечислен ряд инструментов, которые могут быть использованы для решения поставленных задач работы.

ГЛАВА 2. АРХИТЕКТУРА ПРОЕКТА PHANTASUS

В этой главе будут подробно рассмотрены составляющие проекта:

— morpheus.js;
— R-пакет phantasus;
— OpenCPU.

Также будут описаны взаимосвязи между компонентами, сопутствующие инструменты и их предназначение в системе и ключевые для архитектуры выдержки из исходного кода.

2.1. morpheus.js

Как уже было рассказано в обзоре, morpheus.js — веб-приложение, полностью созданное на JavaScript, для визуализации и анализа матриц.

В этом разделе будут описаны основные классы и функции, реализованные в исходном коде morpheus.js, которые будут в дальнейшем необходимы для расширения функционала.

2.1.1. Чтение данных

В morpheus.js данные могут быть загружены из файла, полученного одним из следующих путей:

— Из компьютера;
— По URL-ссылке;
— Из Dropbox.

Допустимые форматы загружаемых файлов:

— txt-файл с tab-разделителями;
— Excel-таблица;
— MAF [16];
— GCT [17];
— GMT [18].

Для каждого формата файла в исходном коде morpheus.js присутствует соответствующий обработчик данных.

Также, morpheus.js предлагает набор предзагруженных данных из базы TCGA [19].

2.1.2. Класс Dataset

Одним из ключевых классов всего веб-приложения является класс Dataset. В каждом экземпляре этого класса хранится вся необходимая информация о данных, в которую входят:

- Числовая матрица, характеризующая уровень экспрессии всех генов во всех образцах;
- Количество строк и столбцов в матрице;
- Аннотация к образцам, например:
  * пол, возраст, контактную информацию испытуемых, если образцы были взяты с людей;
  * есть или нет инфекция в данном образце;
  * способ лечения;
  * контакты ответственного за взятие данного образца и пр.;
- Аннотация к генам, например:
  * Идентификатор гена в том или ином стандарте;
  * Числовые характеристики гена (средний уровень экспрессии по образцам, номер кластера) и пр.

Аннотация реализована в классе MetadataModel, который представляет собой не что иное, как набор именованных векторов с характеристиками. В каждом векторе хранятся:

- Название;
- Формат (строка, число);
- Массив значений.

Для векторов так же предусмотрены утилиты для визуализации. Так, например, есть возможность показать аннотацию в виде текста и/или цветом, что удобно для категориальных характеристик.

2.1.3. Класс SlicedDatasetView

Чаще всего во время работы программы экземпляры класса Dataset становятся обернуты в оболочку из SlicedDatasetView. Этот дополнительный класс дает возможность не пересоздавать каждый раз Dataset, а просто добавляет к данным информацию о том, какие индексы строк и столбцов выбраны и используются в данный момент.

2.1.4. Класс HeatMap

Данный класс предназначен для визуализации данных, обернутых в класс Dataset или SlicedDatasetView. Он дает возможность выбирать, какая аннотация будет представлена на экране, цветовой код, выбирать строки и столбцы, с которыми будут работать те или иные инструменты.

2.1.5. Реализованные методы

В morpheus.js имеются реализации следующих методов:

- Adjust — инструмент для корректировки данных:
  - $\log_2$;
  - $\log_2^{-1}$;
  - Квантиль-нормализация;
  - Z-тест;
  - Устойчивый Z-тест;
- Collapse — инструмент, позволяющий агрегировать строки или столбцы с одинаковыми значениями с помощью функции: $\min$, $\max$, $mean$, $median$, sum, максимум 25-го и 75-го перцентилей;
- Создать вычисленную аннотацию для строк или столбцов;
- Similarity Matrix;
- Transpose;
- t-SNE;
- Построение графиков.

Также присутствуют фильтрация и сортировка.

2.2. phantasus.js

В этом разделе будет рассмотрен модифицированный вариант morpheus.js и потребовавшиеся для его расширения компоненты.

2.2.1. Клиентская сторона OpenCPU — opencpu.js

В обзоре было рассказано об OpenCPU и его необходимости. В данной работе он нужен для связи JavaScript-клиента и R-сервера.

2.2.2. Поддержка Protocol Buffers — protobuf.js

Чаще всего размеры обрабатываемых матриц 10000-40000 строк на 12-40 столбцов. Соответственно, пересылать их между клиентом и сервером в JSON-формате слишком долго.

Как было сказано в обзоре, Protocol Buffers позволяют лучше сериализовать данные, чтобы уменьшить размер пересылаемого пакета.

К сожалению, Google Developers официально поддерживают только Java, Python, C++, Go, Objective-C, Ruby, JavaNano и C#. Для JavaScript сообщество создает поддержку самостоятельно. После анализа существующих решений, было решено выбрать библиотеку ProtoBuf.js [20].

С помощью класса Builder, обрабатывающего файлы с протоколом структуры (`*.proto`), можно закодировать соответствующий JSON объект в Uint8Array, чтобы после пересылать его на сервер.

2.2.3. Интерактивные графики — plotly.js

Для отображения интерактивных графиков используется библиотека plotly.js [21], которая предоставляет удобное API, в котором описание графика строится в JSON-формате. Соответственно, вся графическая работа лежит на клиенте.

2.2.4. Поддержка ExpressionSet

В phantasus.js в Dataset добавлено дополнительное поле esSession, в котором находится объект класса Promise для асинхронного обновления ключа сессии в этом поле.

При загрузке или обновлении Dataset осуществляется следующий ряд действий:

а) В поле dataset.esSession записывается экземпляр класса Promise, который позволяет продолжать загрузку данных в фоновом режиме, а также ждать, когда данные будут обработаны прежде чем запускать функции использующие ExpressionSet в качестве аргумента (pcaPlot, kmeans, limma). При создании Promise в аргументах указывается две функции: `reject` и `resolve`;

б) Актуальное содержимое экземпляра класса Dataset вместе с матрицей и аннотацией сериализуется в ProtoBuf по протоколу, описанному в приложении А.1;

в) С помощью opencpu.js отправляется RPC за функцией `createES` с аргументом в виде сериализованных данных;

г) Данные поступают на сервер, десериализуются там автоматически и функция `createES` создает ExpressionSet, являющийся копией Dataset из клиента;

д) Функция `createES` объявляет данный ExpressionSet глобальной переменной, таким образом имеется доступ к этому объекту по API-entrypoint `/ocpu/tmp/key/R/es`;
е) По завершении RPC получает ключ временной сессии, содержащий созданный ExpressionSet и завершает Promise с `resolve(session)`;
ж) Если во время одного из этапов произошла ошибка, то Promise завершается с `reject(error)` с текстом ошибки.

2.2.5. Инструмент PcaPlotTool

Данный инструмент предназначен для построения графиков в соответствие с методом главных компонент. В качестве аргументов на вход к инструменту подается:

— Номера образцов для сравнения;
— Категориальная аннотация для различения точек по цвету (если не указана, то стандартный цвет);
— Числовая аннотация для различения точек по размеру (если не указана, то стандартный размер);
— Аннотация для подписей к точкам (если не указана, то без подписи);
— Функция замены NA в данных при вычислении матрицы PCA (mean или median).

На сервер отправляется только ключ сессии и функция замены, содержащей актуальный ExpressionSet. После чего на клиент возвращается матрица PCA, по которой уже на клиенте отрисовывается интерактивный график с учетом графических аргументов в PcaPlot.

2.2.6. Инструмент KmeansTool

На клиенте в инструменте KmeansTool осуществляется только сбор значений аргументов:

— Количество кластеров, на которые нужно разбить данные;
— Функция замены NA в данных.

Данные аргументы и ключ сессии актуального ExpressionSet отправляются на сервер, после чего, получив список Ген-Номер кластера на клиенте отрисовывается новая аннотация к строкам.

2.2.7. Инструмент LimmaTool

На клиенте осуществляется получение аргументов:

— Какие аннотации образцов участвуют в сравнении;
— Какая комбинация значений указанных выше аннотаций обозначает класс A;
— Аналогично для класса B.

После чего образцы разбиваются на классы A и B, либо обозначаются пустыми, в соответствии с полученными аргументами. Список Образец-Класс и ключ сессии актуального ExpressionSet после отправляются на сервер, где происходит вычисление. От сервера приходит файл с сериализованной матрицей результатов, которые с помощью protobuf.js разбираются и отрисовываются в виде аннотации к строкам.

2.3. R-пакет phantasus

Весь реализованный функционал должен иметь клиентскую часть в виде графического интерфейса и серверную в виде R-функции. Прежде чем рассматривать созданные функции, будут представлены имеющиеся необходимые элементы.

2.3.1. Biobase и ExpressionSet

Необходимый минимум функций для работы с геномными данными содержится в R-пакете Biobase [22].

Класс ExpressionSet [23] так же содержится в Biobase. Он помогает представлять данные об экспрессии генов в удобном формате:

— assayData — описание матрицы:
  – features — количество генов;
  – samples — количество образцов;
  – exprs — числовая матрица экспрессии;
— phenoData — аннотация к образцам:
  – sampleNames — идентификаторы образцов;
  – varLabels — названия характеристик;
  – varMetadata — описание характеристик;
  – pData — матрица характеристик;
— featureData — аннотация к генам:
  – featureNames — идентификаторы генов;

- fvarLabels — названия характеристик;
- fvarMetadata — описание характеристик;
- fData — матрица характеристик.

Для доступа к каждому из элементов есть одноименная функция, что позволяет удобно взаимодействовать с экземплярами класса. Также многие из функций обработки данных в Bioconductor и в Biobase в частности завязаны на использование ExpressionSet.

Все реализованные в R-пакете phantasus функции принимают на вход в качестве одного из аргументов экземпляр класса ExpressionSet.

2.3.2. Поддержка Protocol Buffers — protolite

Поддержка ProtoBuf со стороны R-сервера осуществляется с помощью R-пакета protolite [24]. В реализованных функциях данный пакет используется каждый раз, когда необходимо вернуть матрицу больших размеров. Вместо отправки ее в исходном виде или в JSON предпочтительнее сериализовать результат и сохранить в файл, который после будет прочитан на клиенте.

2.3.3. Создание ExpressionSet из данных

В начале работы с phantasus необходимо загрузить данные. Если данные загружены из файла, то они будут сначала обработаны на клиенте, а после пересланы на сервер для создания ExpressionSet из них с помощью кода на листинге 3

```
createES <- function(data, pData, varLabels, fData, fvarLabels) {
  exprs <- t(data)
  phenoData <- AnnotatedDataFrame(data.frame(pData))
  varLabels(phenoData) <- varLabels

  featureData <- AnnotatedDataFrame(data.frame(fData))
  varLabels(featureData) <- fvarLabels

  es <- ExpressionSet(assayData = exprs, phenoData=phenoData, featureData =
      featureData)
  assign("es", es, envir = parent.frame())
  es
}
```

Листинг 3 – Функция создания ExpressionSet из исходных данных

По завершении функция отправляет `es` в глобальные переменные, чтобы ExpressionSet был доступен по адресу: /ocpu/tmp/session-key/R/es.

Таким образом, получив ключ данной сессии, можно иметь доступ и к ExpressionSet, находящемуся в ней.

Ключ сессии обновляется каждый раз при изменении Dataset в phantasus.js (Adjust, Collapse, new HeatMap, Transpose). Изменные данные пересылаются на сервер и ключ сессии обновляется в поле esSession в Dataset.

2.3.4. Загрузка данных из GEO — loadGEO

В обзоре были описаны форматы данных в репозитории Gene Expression Omnibus. В phantasus загрузка данных из GEO осуществляется следующим образом:

а) Функция loadGEO принимает на вход идентификатор GEO;
б) В зависимости от его вида (GSE или GDS) запускаются дополнительные функции (getGSE на листинге 4 и getGDS на листинге 5);
в) После с помощью `GEOquery::getGEO` загружаются данные с аннотацией (или подгружаются из кэша, если их уже загружали);
г) Результат обрабатывается, создается ExpressionSet и отправляется в глобальные переменные;
д) В файл записываются сериализованные в ProtoBuf данные, которые после считает и обработает клиент

Код и его дополнительные рутины можно увидеть на листинге 6.

2.3.5. Дифференциальная экспрессия — limmaAnalysis

R-пакет limma [25] предоставляет методы для вычисления дифференциальной экспрессии. Данная функция помогает увидеть, насколько случайны или нет различия между образцами в генах в разных условиях.

Соответственно, было решено включить в phantasus поддержку такой функции.

Реализация функции `limmaAnalysis` в R-пакете phantasus принимает в себя ExpressionSet и вектор с соответствием Образец-Класс. Класс в этом векторе может быть A, B или пустым. Необходимо сравнить классы A и B, а для этого нужно дополнить фенотипические данные ExpressionSet полученным вектором. После дизайн сравнения передается в функцию, которая возвращает матрицу статистических характеристик к каждому гену. Эта матрица далее сериализуется в ProtoBuf, записывается в файл, который будет в дальнейшем разобран на клиенте.

```
getGSE <- function(name, destdir = tempdir()) {
  es <- getGEO(name, AnnotGPL = T, destdir = destdir)[[1]]
  featureData(es) <- featureData(es)[,grepl("symbol", fvarLabels(es), ignore.case = T)]
  phenoData(es) <- phenoData(es)[,grepl("characteristics", varLabels(es), ignore.case = T)
                                 | (varLabels(es) %in% c("title", "id", "geo_accession"))]
  chr <- varLabels(es)[grepl("characteristics", varLabels(es), ignore.case = T)]
  take <- function(x, n) {
    sapply(x, function(x) { x[[n]] })
  }
  rename <- function(prevName, x) {
    splitted <- strsplit(x, ": ")
    sumlength <- sum(sapply(as.vector(splitted), length))
    if (sumlength != 2 * length(x)) {
       return(list(name = prevName, x = x))
    }
    splittedFirst <- unique(take(splitted, 1))
    if (length(splittedFirst) == 1) {
       res = list(name = splittedFirst[1], x = take(splitted, 2))
    }
    else {
       res = list(name = prevName, x = x)
    }
    res
  }
  renamed <- lapply(chr, function(x) { rename(x, as.vector(pData(es)[,x])) })
  phenoData(es) <- phenoData(es)[, !(varLabels(es) %in% chr)]
  pData(es)[,take(renamed,1)] <- take(renamed,2)
  es
}
```

Листинг 4 – Загрузка данных типа GSE из Gene Expression Omnibus

Код функции `limmaAnalysis` представлен на листинге 7.

2.3.6. Статистические функции — stats

R-пакет stats [26] содержит в себе большинство современных используемых методов статистического анализа. В данной работе используются два: PCA (метод главных компонент) и kmeans (кластеризация).

Выводы по главе 2

В данной главе были рассмотрены основные составляющие проекта:

— morpheus.js — база для расширения, поэтому были описаны важные компоненты, с которыми необходимо взаимодействовать во время дополнения проекта новым функционалом;

```
getGDS <- function(name, destdir = tempdir()) {
 l <- getGEO(name, destdir = destdir)
 table <- slot(l, 'dataTable') # extracting all useful information on dataset
 data <- Table(table)  # extracting table ID_REF | IDENTIFIER/SAMPLE | SAMPLE1 |
     ...
 columnsMeta <- Columns(table) # phenoData
 sampleNames <- as.vector(columnsMeta[["sample"]])
 rownames <- as.vector(data[["ID_REF"]])
 symbol <- as.vector(data[["IDENTIFIER"]])
 data <- data[,sampleNames] # expression data
 exprs <- as.matrix(data)
 row.names(exprs) <- rownames
 row.names(columnsMeta) <- sampleNames
 # columnsMeta <- columnsMeta[,!(colnames(columnsMeta) %in% c('sample'))]
 pData <- AnnotatedDataFrame(data.frame(columnsMeta, check.names = F))
 fData <- data.frame(matrix(symbol, nrow(exprs), 1));
 colnames(fData) <- "symbol"
 fData <- AnnotatedDataFrame(fData)
 featureNames(fData) <- rownames
 ExpressionSet(assayData = exprs, phenoData = pData, featureData = fData)
}
```

Листинг 5 – Загрузка данных типа GDS из Gene Expression Omnibus

— phantasus.js — расширенный morpheus.js. Были описаны дополнения: новые инструменты, поддержка ProtoBuf, поддержка актуального состояния ключа сессии ExpressionSet для каждого Dataset;
— R-пакет phantasus — R-пакет, содержащий в себе серверные реализации всех добавленных методов и инструментов.

```
loadGEO <- function(name, type = NA) {
  es <- getES(name, type, destdir = "/var/phantasus/cache")
  assign("es", es, envir = parent.frame())
  data <- as.matrix(exprs(es)); colnames(data) <- NULL; row.names(data) <- NULL

  pdata <- as.matrix(pData(es)); colnames(pdata) <- NULL; row.names(pdata) <-
     NULL

  participants <- colnames(es)
  rownames <- rownames(es)

  fdata <- as.matrix(fData(es))
  colnames(fdata) <- NULL
  row.names(fdata) <- NULL

  res <- list(data = data, pdata = pdata,
              fdata = fdata, rownames = rownames,
              colMetaNames = varLabels(phenoData(es)),
              rowMetaNames = varLabels(featureData(es)))

  f <- tempfile(pattern = "gse", tmpdir = getwd(), fileext = ".bin")
  writeBin(protolite::serialize_pb(res), f)
  f
}
getES <- function(name, type = NA, destdir = tempdir()) {
  if (is.na(type)) {
     type = substr(name, 1, 3)
  }
  if (type == 'GSE') {
    es <- getGSE(name, destdir)
  }
  else if (type == 'GDS') {
    es <- getGDS(name, destdir)
  }
  else {
    stop("Incorrect name or type of the dataset")
  }
  es
}
```

Листинг 6 – Загрузка данных из Gene Expression Omnibus

```
limmaAnalysis <- function(es, rows = c(), columns = c(), fieldValues) {
  assertthat::assert_that(length(columns) == length(fieldValues) || length(
     columns) == 0)
  rows <- getIndicesVector(rows, nrow(exprs(es)))
  columns <- getIndicesVector(columns, ncol(exprs(es)))
  fieldName <- "Comparison"
  fieldValues <- replace(fieldValues, fieldValues == '', NA)
  new.pdata <- pData(es)[columns,]
  new.pdata[[fieldName]] <- as.factor(fieldValues)
  new.pdata <- new.pdata[!is.na(new.pdata[[fieldName]]),]
  new.sampleNames <- row.names(new.pdata)
  es.copy <- es[rows, new.sampleNames]
  pData(es.copy) <- new.pdata
  fData(es.copy) <- data.frame(row.names=rownames(es.copy))
  es.design <- model.matrix(~0 + Comparison, data = pData(es.copy))
  colnames(es.design) <- gsub(pattern = fieldName,
                              replacement = '',
                              x = make.names(colnames(es.design)))
  fit <- lmFit(es.copy, es.design)
  fit2 <- contrasts.fit(fit, makeContrasts(B - A,
                                           levels=es.design))
  fit2 <- eBayes(fit2)
  de <- topTable(fit2, adjust.method="BH", number=Inf)
  de <- de[row.names(fData(es.copy)),]
  f <- tempfile(pattern = "de", tmpdir = getwd(), fileext = ".bin")
  writeBin(protolite::serialize_pb(as.list(de)), f)
  f
}
```

Листинг 7 – Реализация дифференциальной экспрессии в R-пакете phantasus

ГЛАВА 3. РЕАЛИЗАЦИЯ И ИСПОЛЬЗОВАНИЕ

3.1. Структура git-репозитория

Как следует из главы об архитектуре проекта, проект состоит из двух составляющих:

— phantasus.js — fork репозитория morpheus.js [4];

— phantasus — репозиторий для R-пакета.

Внутри репозитория phantasus находится подмодуль для репозитория phantasus.js. Соответственно, чтобы загрузить целиком весь проект достаточно вызвать команду из листинга 8.

```
1  git clone --recursive https://github.com/ctlab/phantasus.git
```

Листинг 8 – Клонирование репозитория проекта phantasus

3.2. Кэш для данных из GEO

Независимо от способа запуска, данные, загруженные из GEO, кэшируются в определенной папке, чтобы не было необходимости перескачивать их заново.

3.3. Единый R-пакет phantasus

Так как проект теперь существует в виде почти единого git-репозитория, его легко можно использовать как полноценный R-пакет, содержащий в себе в том числе и файлы для веб-приложения. С помощью функции, представленной на листинге 9, можно запускать веб-приложение phantasus непосредственно из R.

3.4. Docker-образ phantasus

На hub.docker.com существует автоматический репозиторий, привязанный к git-репозиторию phantasus. Для каждой перекомпиляции он использует Dockerfile с листинга Б.1, расположенный в репозитории.

На данный момент существуют две ветки Docker-образа:

— master — компиляция происходит из master-веток составляющих проекта, чаще всего эти скомпилированные образы стабильны и отправляются в открытый доступ для использования;

```
#' Starts http server handling morpheus static files and opencpu
#' @param host host to listen
#' @param port port to listen
#' @param morpheusRoot path to static files with morpheus (on local file system)
#' @import opencpu
#' @import httpuv
#' @import Rook
#' @export
serveMorpheus <- function(host, port, morpheusRoot) {
  app <-
    Rook::URLMap$new(
      "/ocpu"=opencpu:::rookhandler("/ocpu"),
      "/?"=Rook::Static$new(
        urls = c('/'),
        root = morpheusRoot
      ))

  httpuv::runServer(host,
                    port,
                    app=app)

}
```

Листинг 9 – Функция для запуска приложения из R

— develop — компиляция происходит из develop-веток составляющих проекта, эти образы используются для тестирования всего приложения в целом, тестирования нового функционала и не предназначены для использования на серверах.

Чтобы загрузить Docker-образ, нужно воспользоваться командой с листинга 10

```
  docker pull dzenkova/phantasus
```

Листинг 10 – Загрузка Docker-образа phantasus

3.4.1. Запуск Docker-контейнера

3.5. Настройка с помощью Apache

3.5.1. Переадресация OpenCPU-сервера

После того, как сравнили скорость работы с использованием single-user OpenCPU-сервера и multi-user, пришли к выводу, что скорость доступа к первому выше из-за того, что RApache работает достаточно медленно.

Соответственно, происходит переадресация запросов с /ocpu на //localhost:8001/ocpu.

3.5.2. Балансировщик для multi-user соединения

Из-за того, что используется single-user OpenCPU-сервер, несколько людей, использующих веб-приложение phantasus одновременно, вынуждены ждать, пока закончится запрос для одного.

Чтобы такого не происходило, запускается четыре экземпляра OpenCPU-сервера и с помощью Apache-балансировщика можно получать доступ к R-серверу параллельно.

3.6. Статистика использования

Выводы по главе 3

В данной главе были рассмотрены технические подробности реализации веб-приложения: инструкции для запуска, варианты использования и подробности настройки веб-приложения на сервере.

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Gentleman R.*, *Ihaka R.* R project. — URL: https://www.r-project.org/. [Электронный ресурс].

2 Bioconductor / A. Sonali [и др.]. — URL: https://www.bioconductor.org/. [Электронный ресурс].

3 *Gould J.* GENE-E. — URL: http://www.broadinstitute.org/cancer/software/GENE-E/. [Электронный ресурс].

4 *Gould J.* morpheus.js. — URL: https://clue.io/morpheus.js/. [Электронный ресурс].

5 *Foundation N.* Node.js. — URL: https://nodejs.org/. [Электронный ресурс].

6 *Jeroen O.* OpenCPU. — URL: https://www.opencpu.org/. [Электронный ресурс].

7 Asynchronous JavaScript and XML. — URL: http://api.jquery.com/jquery.ajax/. [Электронный ресурс].

8 *Biotechnology Information N. C. for.* Gene Expression Omnibus. — URL: https://www.ncbi.nlm.nih.gov/geo/. [Электронный ресурс].

9 *Davis S.*, *Meltzer P.* GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor // Bioinformatics. — 2007. — Т. 14. — С. 1846–1847.

10 *Hykes S.* Docker. — URL: https://www.docker.com. [Электронный ресурс].

11 *Docker I.* Docker Hub. — URL: https://hub.docker.com/. [Электронный ресурс].

12 *GitHub*. GitHub. — URL: https://github.com/. [Электронный ресурс].

13 *Atlassian*. Bitbucket. — URL: https://bitbucket.org/. [Электронный ресурс].

14 *Developers G.* Protocol Buffers. — URL: https://developers.google.com/protocol-buffers/. [Электронный ресурс].

15 *Foundation A. S.* Apache HTTP Server Project. — URL: https://httpd.apache.org. [Электронный ресурс].

16 *Institute N. C.* Mutation Annotation Format. — URL: https://wiki.nci.nih.gov/display/TCGA/Mutation+Annotation+Format+%28MAF%29+Specification/. [Электронный ресурс].

17 GCT. — URL: http://software.broadinstitute.org/cancer/software/genepattern/file-formats-guide#GCT/. [Электронный ресурс].

18 Gene Matrix Transposed file format. — URL: http://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats#GMT:_Gene_Matrix_Transposed_file_format_.28.2A.gmt.29/. [Электронный ресурс].

19 *Insitute N. C.* The Cancer Genome Atlas. — URL: https://cancergenome.nih.gov/. [Электронный ресурс].

20 *dcode*. Protocol Buffers for JavaScript (& TypeScript). — URL: https://github.com/dcodeIO/ProtoBuf.js/. [Электронный ресурс].

21 *Plotly*. plotly. — URL: https://plot.ly/company/team/. [Электронный ресурс].

22 Orchestrating high-throughput genomic analysis with Bioconductor / Huber [и др.] // Nature Methods. — 2015. — Т. 12, № 2. — С. 115–121. — URL: http://www.nature.com/nmeth/journal/v12/n2/full/nmeth.3252.html.

23 *Falcon S.*, *Morgan M.*, *Gentleman R.* An Introduction to Bioconductor's ExpressionSet Class. — 2006. — URL: https://www.bioconductor.org/packages/devel/bioc/vignettes/Biobase/inst/doc/ExpressionSetIntroduction.pdf. [Электронный ресурс].

24 *Ooms J.* protolite: Fast and Simple Object Serialization to Protocol Buffers. — URL: https://cran.r-project.org/web/packages/protolite/index.html/. [Электронный ресурс].

25 limma powers differential expression analyses for RNA-sequencing and microarray studies / M. E. Ritchie [и др.] // Nucleic Acids Research. — 2015. — Т. 43, № 7. — e47.

26 *R Core Team*. R: A Language and Environment for Statistical Computing / R Foundation for Statistical Computing. — Vienna, Austria, 2017. — URL: https://www.R-project.org/.

ПРИЛОЖЕНИЕ А. ПРОТОКОЛ СЕРИАЛИЗАЦИИ В PROTOBUF

R-пакет protolite использует стандартный протокол сериализации, представленный на листинге А.1. Этот же протокол было решено использовать и при сериализации на клиенте для однообразия и для корректного разбора сообщений как на клиенте от сервера, так и на сервере от клиента.

```
package rexp;

option java_package = "org.godhuli.rhipe";
option java_outer_classname = "REXPProtos";

message REXP {
    enum RClass {
        STRING = 0;
        RAW = 1;
        REAL = 2;
        COMPLEX = 3;
        INTEGER = 4;
        LIST = 5;
        LOGICAL = 6;
        NULLTYPE = 7;
        NATIVE = 8;
    }
    enum RBOOLEAN {
        F=0;
        T=1;
        NA=2;
    }

    required RClass rclass = 1;
    repeated double realValue = 2 [packed=true];
    repeated sint32 intValue = 3 [packed=true];
    repeated RBOOLEAN booleanValue = 4;
    repeated STRING stringValue = 5;

    optional bytes rawValue = 6;
    repeated CMPLX complexValue = 7;
    repeated REXP rexpValue = 8;

    repeated string attrName = 11;
    repeated REXP attrValue = 12;
    optional bytes nativeValue = 13;
}
message STRING {
    optional string strval = 1;
    optional bool isNA = 2 [default=false];
}
message CMPLX {
    optional double real = 1 [default=0];
    required double imag = 2;
}
```

Листинг А.1 – Протокол сериализации R-пакета protolite

ПРИЛОЖЕНИЕ Б. DOCKERFILE

```
FROM ubuntu
RUN apt-get -y update && apt-get -y dist-upgrade && apt-get -y install \
        software-properties-common \
        git \
        libcairo2-dev \
        libxt-dev \
        libssl-dev \
        libssh2-1-dev \
        libcurl4-openssl-dev \
        apache2 \
        locales && \
    apt-add-repository -y ppa:opencpu/opencpu-1.6 && \
    apt-get -y update && apt-get -y install opencpu-lib

RUN touch /etc/apache2/sites-available/opencpu2.conf
RUN printf "ProxyPass /ocpu/ http://localhost:8001/ocpu/\nProxyPassReverse /ocpu/
    http://localhost:8001/ocpu\n" >> /etc/apache2/sites-available/opencpu2.conf
RUN a2ensite opencpu2

RUN sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu trusty/" >> /etc/
   apt/sources.list'
RUN gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9
RUN gpg -a --export E084DAB9 | apt-key add -
RUN apt-get -y update && apt-get -y install \
    r-base \
    libprotobuf-dev \
    protobuf-compiler \
    r-cran-xml

RUN git clone --recursive https://github.com/ctlab/phantasus /root/phantasus
RUN cp -r /root/phantasus/inst/www/phantasus.js /var/www/html/phantasus
RUN R -e 'source("https://bioconductor.org/biocLite.R"); install.packages("XML",
   repo = "http://cran.gis-lab.info"); biocLite("Biobase"); biocLite("limma");
   biocLite("org.Mm.eg.db")'
RUN R -e 'install.packages("devtools", repo = "http://cran.gis-lab.info");
   library(devtools); install_github("hadley/scales"); install_github("assaron/
   GEOquery"); install("/root/phantasus")'

RUN a2enmod proxy_http
EXPOSE 80
EXPOSE 443
EXPOSE 8004
RUN locale-gen en_US.UTF-8
ENV LANG en_US.UTF-8
ENV LANGUAGE en_US:en
ENV LC_ALL en_US.UTF-8

RUN mkdir -p /var/phantasus/cache
VOLUME ["/var/phantasus/cache"]

CMD service apache2 start && \
    R -e 'opencpu::opencpu$start(8001)' && \
    tail -F /var/log/opencpu/apache_access.log
```

Листинг Б.1 – Dockerfile для Docker-образа веб-приложения phantasus