

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	6
1. Обзор предметной области .....	7
1.1. Биоинформатика .....	7
1.1.1. Анализ экспрессии генов .....	7
1.1.2. Используемые методы .....	7
1.2. Существующие решения для анализа экспрессии генов .....	9
1.2.1. R/Bioconductor .....	9
1.2.2. GENE-E .....	9
1.2.3. morpheus.js .....	10
1.2.4. ProjectX .....	10
1.3. Инструменты, которые могут быть применены .....	10
1.3.1. Язык R и библиотека Bioconductor .....	10
1.3.2. JavaScript и Node.js .....	11
1.3.3. R shiny.....	11
1.3.4. OpenCPU .....	11
1.3.5. Gene Expression Omnibus .....	13
1.3.6. Docker .....	14
1.3.7. JSON .....	14
1.3.8. Protocol Buffers .....	15
1.3.9. Apache.....	15
1.3.10.HTML.....	15
1.4. Постановка задачи.....	15
1.4.1. Цель работы .....	15
1.4.2. Основные задачи.....	16
1.4.3. Требования к веб-приложению phantastus.....	16
Выводы по главе 1.....	16
2. Архитектура проекта phantastus.....	17
2.1. morpheus.js .....	17
2.1.1. Чтение данных .....	17
2.1.2. Класс Dataset.....	18
2.1.3. Класс SlicedDatasetView.....	18
2.1.4. Класс HeatMap .....	19
2.1.5. Реализованные методы .....	19

2.1.6. Класс DatasetUtil.....	19
2.2. phantasus.js.....	20
2.2.1. Поддержка Protocol Buffers — protobuf.js .....	20
2.2.2. Клиентская сторона OpenCPU — opencpu.js .....	20
2.2.3. Интерактивные графики — plotly.js.....	21
2.2.4. Поддержка ExpressionSet .....	21
2.2.5. Инструмент PcaPlotTool.....	22
2.2.6. Инструмент KmeansTool .....	23
2.2.7. Инструмент LimmaTool .....	23
2.3. R-пакет phantasus.....	25
2.3.1. Biobase и ExpressionSet.....	25
2.3.2. Поддержка Protocol Buffers — protolite.....	27
2.3.3. Создание ExpressionSet из данных.....	27
2.3.4. Загрузка данных из GEO — loadGEO .....	28
2.3.5. Дифференциальная экспрессия — limmaAnalysis .....	29
2.3.6. Статистические функции — stats.....	29
Выводы по главе 2.....	30
3. Реализация и использование .....	32
3.1. Структура git-репозитория.....	32
3.2. Кэш для данных из GEO .....	32
3.3. Единый R-пакет phantasus .....	32
3.4. Docker-образ phantasus.....	32
3.4.1. Запуск Docker-контейнера.....	33
3.5. Настройка с помощью Apache .....	33
3.5.1. Переадресация OpenCPU-сервера.....	33
3.5.2. Балансировщик для multi-user соединения .....	34
3.6. Статистика использования.....	34
Выводы по главе 3.....	34
ЗАКЛЮЧЕНИЕ.....	35
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	36
ПРИЛОЖЕНИЕ А. Протокол сериализации в ProtoBuf .....	39
ПРИЛОЖЕНИЕ Б. Dockerfile.....	40

## **ВВЕДЕНИЕ**

Биоинформатика образована на стыке биологических направлений и информатики, как реакция на всё увеличивающийся 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.

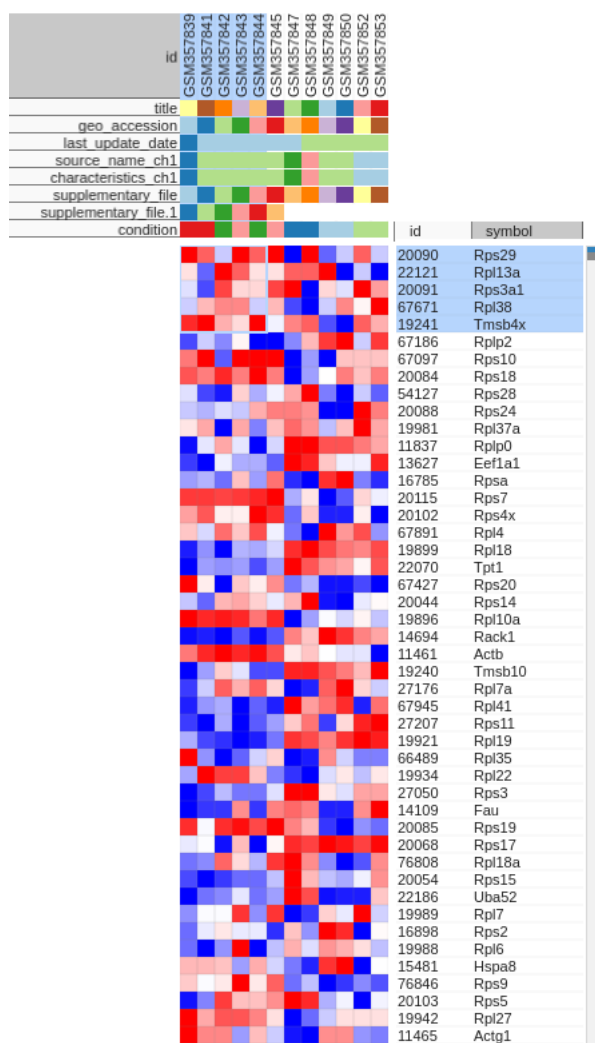


Рисунок 1 – Визуализация матрицы GSE14308 в виде тепловой карты в веб-приложении Morpheus

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 воспроизводимость 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 результата 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 `/оспу/`. 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 /оспу/library/MASS/R/cats/json
POST	вызвать функцию	аргументы функции	POST /оспу/library/stats/R/rnorm
GET	прочитать файл	-	GET /оспу/library/MASS/NEWS
POST	запустить скрипт	аргументы запуска скрипта	POST /оспу/library/MASS?scripts/ch01.R

На *OpenCPU*-сервере могут быть доступны:

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

```
1  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.

```
1  var req = ocpu.rpc("function.name", arguments, callback(session) {
2    \\ Handling result
3  });
```

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 конфигурацией. 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 выполняется 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 следующих структур:

- пара *ключ: значение*, где ключом может быть только регистрозависимая строка, а в качестве значения может выступать 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 следующие составляющие проекта:

- исходное веб-приложение *Morpheus* (*morpheus.js*), содержащиеся в нем методы и классы, которые будут необходимы для дальнейших модификаций;
- созданный и использующийся на серверной стороне проекта *R*-пакет *phantasus*;
- *OpenCPU* как связь между модифицированным *Morpheus* (*phantasus.js*) на стороне клиента и *R*-пакетом *phantasus* на стороне сервера.

Также будут описаны 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* — 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 одним из следующих путей:

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

Допустимые форматы загружаемых файлов:

- *txt*-файл с *tab*-разделителями;
- *Excel*-таблица;
- *Mutation Annotation Format* (*MAF*) [16];
- *Gene Cluster Text* (*GCT*) [17];
- *Gene Matrix Transposed* (*GMT*) [18].

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

Также, *morpheus.js* предлагает набор предзагруженных данных из базы *The Cancer Genome Atlas (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 и в каком порядке.

#### 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 обработки и анализа данных:

- 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;
- CalculatedAnnotation — добавление вычисленной аннотации для строк или столбцов;
- Similarity Matrix — построение матрицы соответствия строк или столбцов друг другу;
- Transpose — транспонирование матрицы;
- t-SNE — реализация алгоритма снижения размерности;
- ChartTool — построение графиков.

Также присутствуют фильтрация и сортировка.

#### 2.1.6. Класс DatasetUtil

Класс DatasetUtil содержит в себе утилиты для обработки и чтения данных в Dataset:



- обработка входных файлов с данными и отправка их на соответствующий класс чтения в зависимости от формата;
- поиск по данным;
- преобразование данных в *JSON* и обратно.

## 2.2. *phantasus.js*

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

### 2.2.1. 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, технология *Protocol Buffers* позволяет 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*, который компилируется из \*.proto-файлов и позволяет получить доступ к сериализованным данным, можно закодировать соответствующий *JSON*-объект в *Uint8Array*, чтобы после пересылать его в сжатом виде на сервер.

Для преобразования объекты классов *Dataset* и *SlicedDatasetView* в сериализованный *Uint8Array* с помощью протокола описанного в приложении А.1. Соответственно, предварительно данные из *Dataset* с помощью добавленных в *DatasetUtil* методов преобразовываются в соответствующий *JSON*-объект, а после с помощью *Builder* сериализуется.

### 2.2.2. Клиентская сторона OpenCPU — *opencpu.js*

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

Каждый из инструментов, реализованных в *phantasus.js* строится по следующему принципу:

- а) обработка аргументов, полученных в интерфейсе;
- б) подготовка данных и аргументов к отправке на сервер;
- в) *RPC*-вызов функции, как на листинге 2;
- г) обработка содержащегося в полученной сессии результата;
- д) передача ответа в интерфейс.

### 2.2.3. Интерактивные графики — *plotly.js*

Для отображения интерактивных графиков используется библиотека *plotly.js* [21], которая предоставляет удобное *API*, где 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 *openspu.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 /ospu/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).

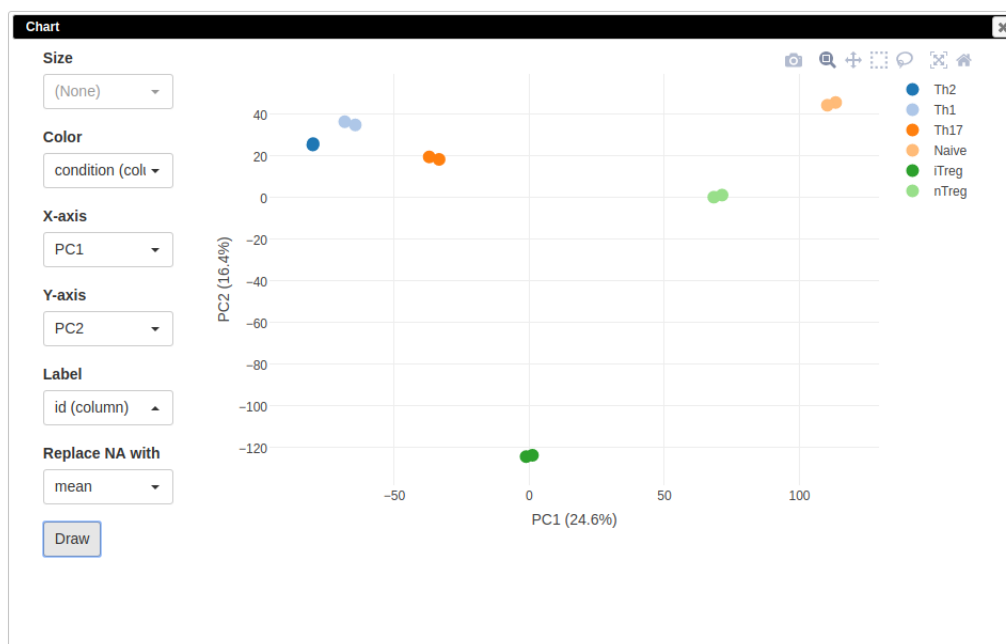


Рисунок 2 – Инструмент PcaPlotTool и отрисованный график по данным GSE14308

Далее по алгоритму, описанному в разделе 2.2.2, на *OpenCPU*-сервер отправляется *RPC*-вызов с аргументами: ключ сессии, содержащий актуальный *ExpressionSet*, и функция замены *NA*. На клиент в *JSON*-формате приходит вычисленная матрица *PCA*.

После, по дополнительным аргументам и вычисленной матрице, строится интерактивный график с помощью *plotly.js*, пример которого можно увидеть на рисунке 2.

### 2.2.6. Инструмент KmeansTool

Этот инструмент осуществляет разбиение генов на указанное пользователем число кластеров по алгоритму *kmeans*.

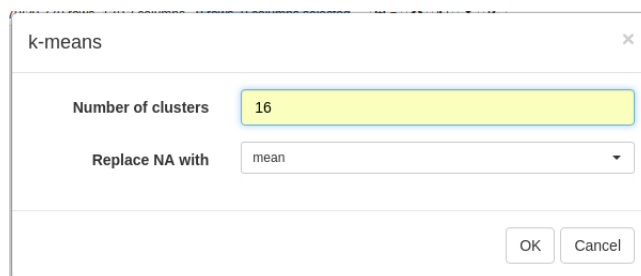


Рисунок 3 – Графический интерфейс инструмента KmeansTool

На клиенте в инструменте KmeansTool, который показан на рисунке 3, считываются следующие аргументы:

- количество кластеров, на которые нужно разбить данные;
- функция замены *NA* в данных.

Данные аргументы и ключ сессии актуального *ExpressionSet* отправляются на сервер. На сервер возвращается список соответствия каждого гена определенному кластеру, который отрисовывается как новая цветовая аннотация к строкам, пример такой аннотации можно увидеть на рисунке 4.

### 2.2.7. Инструмент LimmaTool

Инструмент предназначен для анализа дифференциальной экспрессии: экспрессия генов сравнивается в двух группах образцов, и вычисляются несколько статистических характеристик, показывающих, насколько случайны различия этих групп.

На клиенте, в инструменте, показанном на рисунке 5, осуществляется получение следующих аргументов:

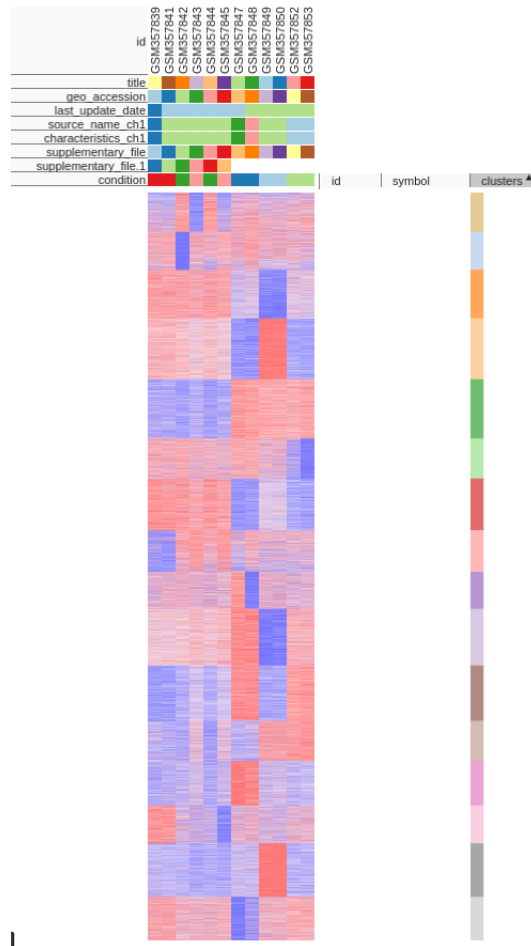


Рисунок 4 – Результат работы инструмента KmeansTool на данных GSE14308

- какие аннотации образцов участвуют в сравнении;
- какая комбинация значений указанных выше аннотаций обозначает класс А;
- аналогично для класса В.

Далее происходит подготовка аргументов к отправке на сервер: образцы разбиваются по выбранным аннотациям на три группы: А, В и неучаствующие в сравнении. Список соответствия образцов классам и ключ сессии, содержащей актуальный ExpressionSet, отправляются на сервер.

От сервера приходит файл с сериализованной в *ProtoBuf* матрицей результатов, которые с помощью *protobuf.js* разбираются и отрисовываются в виде аннотации к строкам. Результат работы можно увидеть на рисунке 6.



Рисунок 5 – Графический интерфейс инструмента LimmaTool

### 2.3. R-пакет phantus

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] также 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;

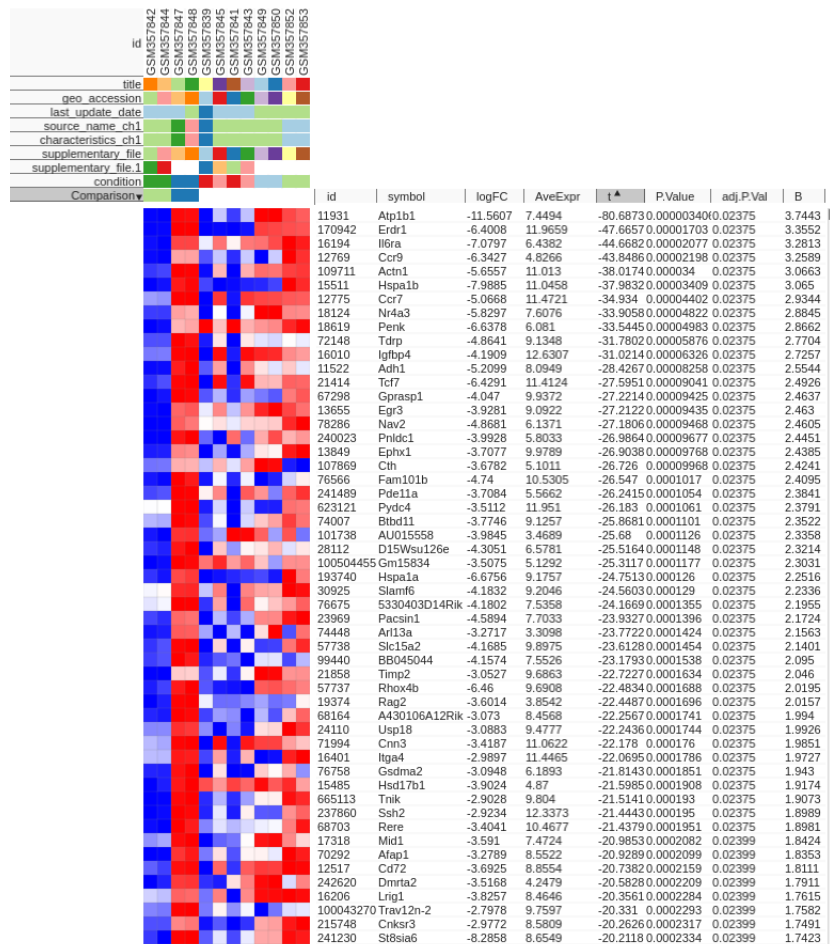


Рисунок 6 – Результат работы инструмента LimmaTool на данных GSE14308

- 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 *protolit* [24]. В реализованных функциях данный пакет используется каждый раз, когда необходимо вернуть на клиент матрицу больших размеров. Вместо отправки ее в исходном виде или в *JSON*, предпочтительнее сериализовать результат и сохранить в файл, который будет содержаться в директории текущей *OpenCPU*-сессии и 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.

```

1 createES <- function(data, pData, varLabels, fData, fvarLabels) {
2   exprs <- t(data)
3   phenoData <- AnnotatedDataFrame(data.frame(pData))
4   varLabels(phenoData) <- varLabels
5
6   featureData <- AnnotatedDataFrame(data.frame(fData))
7   varLabels(featureData) <- fvarLabels
8
9   es <- ExpressionSet(assayData = exprs, phenoData=phenoData, featureData =
10     featureData)
11   assign("es", es, envir = parent.frame())
12   es
13 }
```

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*. Чаще всего изменения происходят в результате работы одного из следующих инструментов: *Adjust*, *Collapse*, *new HeatMap*, *Transpose*. Изменные данные, точно так же, как и новые, 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 phantasmus 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 считывает и обрабатывает клиент

```

1 getGSE <- function(name, destdir = tempdir()) {
2   es <- getGEO(name, AnnotGPL = T, destdir = destdir)[[1]]
3   featureData(es) <- featureData(es)[,grepl("symbol", fvarLabels(es), ignore.case
4     = T)]
5   phenoData(es) <- phenoData(es)[,grepl("characteristics", varLabels(es), ignore.
6     case = T)
7     | (varLabels(es) %in% c("title", "id", "geo_
8       accession"))]
9   chr <- varLabels(es)[grepl("characteristics", varLabels(es), ignore.case = T)]
10  take <- function(x, n) {
11    sapply(x, function(x) { x[[n]] })
12  }
13  rename <- function(prevName, x) {
14    splitted <- strsplit(x, ": ")
15    sumlength <- sum(sapply(as.vector(splitted), length))
16    if (sumlength != 2 * length(x)) {
17      return(list(name = prevName, x = x))
18    }
19    splittedFirst <- unique(take(splitted, 1))
20    if (length(splittedFirst) == 1) {
21      res = list(name = splittedFirst[1], x = take(splitted, 2))
22    }
23    else {
24      res = list(name = prevName, x = x)
25    }
26    res
27  }
28  renamed <- lapply(chr, function(x) { rename(x, as.vector(pData(es)[,x])) })
29  phenoData(es) <- phenoData(es)[, !(varLabels(es) %in% chr)]
30  pData(es)[,take(renamed,1)] <- take(renamed,2)
31  es
32 }
```

Листинг 4 – Загрузка данных типа GSE из Gene Expression Omnibus

Код и его дополнительные рутины можно увидеть на листинге 6.

```

1 getGDS <- function(name, destdir = tempdir()) {
2   l <- getGEO(name, destdir = destdir)
3   table <- slot(l, 'dataTable') # extracting all useful information on dataset
4   data <- Table(table) # extracting table ID_REF | IDENTIFIER/SAMPLE | SAMPLE1 |
5   ...
6   columnsMeta <- Columns(table) # phenoData
7   sampleNames <- as.vector(columnsMeta[["sample"]])
8   rownames <- as.vector(data[["ID_REF"]])
9   symbol <- as.vector(data[["IDENTIFIER"]])
10  data <- data[, sampleNames] # expression data
11  exprs <- as.matrix(data)
12  row.names(exprs) <- rownames
13  row.names(columnsMeta) <- sampleNames
14  # columnsMeta <- columnsMeta[,!(colnames(columnsMeta) %in% c('sample'))]
15  pData <- AnnotatedDataFrame(data.frame(columnsMeta, check.names = F))
16  fData <- data.frame(matrix(symbol, nrow(exprs), 1));
17  colnames(fData) <- "symbol"
18  fData <- AnnotatedDataFrame(fData)
19  featureNames(fData) <- rownames
20  ExpressionSet(assayData = exprs, phenoData = pData, featureData = fData)
}

```

Листинг 5 – Загрузка данных типа GDS из Gene Expression Omnibus

### 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 А, В или пустым. Необходимо сравнить классы А и В, 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.

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).

```

1 loadGEO <- function(name, type = NA) {
2   es <- getES(name, type, destdir = "/var/phantasus/cache")
3   assign("es", es, envir = parent.frame())
4   data <- as.matrix(exprs(es)); colnames(data) <- NULL; row.names(data) <- NULL
5
6   pdata <- as.matrix(pData(es)); colnames(pdata) <- NULL; row.names(pdata) <-
   NULL
7
8   participants <- colnames(es)
9   rownames <- rownames(es)
10
11  fdata <- as.matrix(fData(es))
12  colnames(fdata) <- NULL
13  row.names(fdata) <- NULL
14
15  res <- list(data = data, pdata = pdata,
16             fdata = fdata, rownames = rownames,
17             colMetaNames = varLabels(phenoData(es)),
18             rowMetaNames = varLabels(featureData(es)))
19
20  f <- tempfile(pattern = "gse", tmpdir = getwd(), fileext = ".bin")
21  writeBin(protolite::serialize_pb(res), f)
22  f
23 }
24 getES <- function(name, type = NA, destdir = tempdir()) {
25   if (is.na(type)) {
26     type = substr(name, 1, 3)
27   }
28   if (type == 'GSE') {
29     es <- getGSE(name, destdir)
30   }
31   else if (type == 'GDS') {
32     es <- getGDS(name, destdir)
33   }
34   else {
35     stop("Incorrect name or type of the dataset")
36   }
37   es
38 }

```

Листинг 6 – Загрузка данных из Gene Expression Omnibus

## 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;
- phantasus.js — расширенный morpheus.js. Были описаны дополнения: новые инструменты, поддержка ProtoBuf, поддержка актуального состояния ключа сессии ExpressionSet для каждого Dataset;

```

1 limmaAnalysis <- function(es, rows = c(), columns = c(), fieldValues) {
2   assertthat::assert_that(length(columns) == length(fieldValues) || length(
3     columns) == 0)
4   rows <- getIndicesVector(rows, nrow(exprs(es)))
5   columns <- getIndicesVector(columns, ncol(exprs(es)))
6   fieldName <- "Comparison"
7   fieldValues <- replace(fieldValues, fieldValues == '', NA)
8   new.pdata <- pData(es)[columns,]
9   new.pdata[[fieldName]] <- as.factor(fieldValues)
10  new.pdata <- new.pdata[!is.na(new.pdata[[fieldName]]),]
11  new.sampleNames <- row.names(new.pdata)
12  es.copy <- es[rows, new.sampleNames]
13  pData(es.copy) <- new.pdata
14  fData(es.copy) <- data.frame(row.names=row.names(es.copy))
15  es.design <- model.matrix(~0 + Comparison, data = pData(es.copy))
16  colnames(es.design) <- gsub(pattern = fieldName,
17    replacement = '',
18    x = make.names(colnames(es.design)))
19  fit <- lmFit(es.copy, es.design)
20  fit2 <- contrasts.fit(fit, makeContrasts(B - A,
21    levels=es.design))
22  fit2 <- eBayes(fit2)
23  de <- topTable(fit2, adjust.method="BH", number=Inf)
24  de <- de[row.names(fData(es.copy)),]
25  f <- tempfile(pattern = "de", tmpdir = getwd(), fileext = ".bin")
26  writeBin(protolite::serialize_pb(as.list(de)), f)
27 }

```

Листинг 7 – Реализация дифференциальной экспрессии в R-пакете phantastus

— R-пакет phantastus — R-пакет, содержащий в себе серверные реализации всех добавленных методов и инструментов.

## 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;

```

1 #' Starts http server handling morpheus static files and opencpu
2 #' @param host host to listen
3 #' @param port port to listen
4 #' @param morpheusRoot path to static files with morpheus (on local file system)
5 #' @import opencpu
6 #' @import httpuv
7 #' @import Rook
8 #' @export
9 serveMorpheus <- function(host, port, morpheusRoot) {
10   app <-
11     Rook::URLMap$new(
12       "/ocpu"=opencpu:::rookhandler("/ocpu"),
13       "/?"=Rook::Static$new(
14         urls = c('/'),
15         root = morpheusRoot
16       ))
17
18   httpuv::runServer(host,
19                     port,
20                     app=app)
21 }
22

```

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

```

1 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 phantasmus 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/](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.

```

1 package rexp;
2
3 option java_package = "org.godhuli.rhipe";
4 option java_outer_classname = "REXPProtos";
5
6 message REXP {
7     enum RClass {
8         STRING = 0;
9         RAW = 1;
10        REAL = 2;
11        COMPLEX = 3;
12        INTEGER = 4;
13        LIST = 5;
14        LOGICAL = 6;
15        NULLTYPE = 7;
16        NATIVE = 8;
17    }
18    enum RBOOLEAN {
19        F=0;
20        T=1;
21        NA=2;
22    }
23
24    required RClass rclass = 1;
25    repeated double realValue = 2 [packed=true];
26    repeated sint32 intValue = 3 [packed=true];
27    repeated RBOOLEAN booleanValue = 4;
28    repeated STRING stringValue = 5;
29
30    optional bytes rawValue = 6;
31    repeated CMPLX complexValue = 7;
32    repeated REXP rexpValue = 8;
33
34    repeated string attrName = 11;
35    repeated REXP attrValue = 12;
36    optional bytes nativeValue = 13;
37 }
38 message STRING {
39     optional string strval = 1;
40     optional bool isNA = 2 [default=false];
41 }
42 message CMPLX {
43     optional double real = 1 [default=0];
44     required double imag = 2;
45 }

```

Листинг А.1 – Протокол сериализации R-пакета protolite

## ПРИЛОЖЕНИЕ Б. DOCKERFILE

```

1 FROM ubuntu
2 RUN apt-get -y update && apt-get -y dist-upgrade && apt-get -y install \
3     software-properties-common \
4     git \
5     libcairo2-dev \
6     libxt-dev \
7     libssl-dev \
8     libssh2-1-dev \
9     libcurl4-openssl-dev \
10    apache2 \
11    locales && \
12    apt-add-repository -y ppa:opencpu/opencpu-1.6 && \
13    apt-get -y update && apt-get -y install opencpu-lib
14
15 RUN touch /etc/apache2/sites-available/opencpu2.conf
16 RUN printf "ProxyPass /ocpu/ http://localhost:8001/ocpu/\nProxyPassReverse /ocpu/\n" >> /etc/apache2/sites-available/opencpu2.conf
17 RUN a2ensite opencpu2
18
19 RUN sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu trusty/" >> /etc/apt/sources.list'
20 RUN gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9
21 RUN gpg -a --export E084DAB9 | apt-key add -
22 RUN apt-get -y update && apt-get -y install \
23     r-base \
24     libprotobuf-dev \
25     protobuf-compiler \
26     r-cran-xml
27
28 RUN git clone --recursive https://github.com/ctlab/phantasus /root/phantasus
29 RUN cp -r /root/phantasus/inst/www/phantasus.js /var/www/html/phantasus
30 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")'
31 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")'
32
33 RUN a2enmod proxy_http
34 EXPOSE 80
35 EXPOSE 443
36 EXPOSE 8004
37 RUN locale-gen en_US.UTF-8
38 ENV LANG en_US.UTF-8
39 ENV LANGUAGE en_US:en
40 ENV LC_ALL en_US.UTF-8
41
42 RUN mkdir -p /var/phantasus/cache
43 VOLUME ["/var/phantasus/cache"]
44
45 CMD service apache2 start && \
46     R -e 'opencpu::opencpu$start(8001)' && \
47     tail -F /var/log/opencpu/apache_access.log

```

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