Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

АВТОМАТИЧЕСКОЕ ВЫДЕЛЕНИЕ ФУНКЦИОНАЛЬНЫХ НАБОРОВ ГЕНОВ В БАЗЕ GEO С ПОМОЩЬЮ ДИФФЕРЕНЦИАЛЬНОЙ ЭКСПРЕССИИ

Автор: Беляева Екатерина Сергеевна _____
Направление подготовки (специальность): 09.03.02 Информационные системы и технологии
Квалификация: Бакалавр
Руководитель: Сергушичев А.А., к.т.н. _____

К защите допустить
Зав. кафедрой Лисицына Л.С., д.т.н., проф. _____

«__» _____ 20__ г.

Санкт-Петербург, 2018 г.

Студент Беляева Е.С. **Группа** Р3420 **Кафедра** КОТ **Факультет** ПИиКТ

Направленность (профиль), специализация Автоматизация и управление в образовательных системах

ВКР принята «___» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «18» июня 2018 г.

Секретарь ГЭК *Бутько Е.Ф.*_____
Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

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

УТВЕРЖДАЮ

Зав. кафедрой КОТ

Лисицына Л.С. _____

«___» _____ 20___ г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент Беляева Е.С. **Группа** Р3420 **Кафедра** КОТ **Факультет** ПИиКТ
Руководитель Сергушичев А.А., к.т.н., доцент кафедры КТ

1 Наименование темы: Автоматическое выделение функциональных наборов генов в базе GEO с помощью дифференциальной экспрессии

Направление подготовки (специальность): 09.03.02 Информационные системы и технологии
Направленность (профиль): Автоматизация и управление в образовательных системах
Квалификация: Бакалавр

2 Срок сдачи студентом законченной работы: «31» мая 2018 г.

3 Техническое задание и исходные данные к работе.

В рамках выпускной квалификационной работы основной задачей является построение функциональных наборов генов из базы GEO с помощью анализа дифференциальной экспрессии, а также обеспечение интерфейса доступа к ним

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

а) Обзор предметной области.
б) Анализ существующих сервисов.
в) Загрузка данных и их обработка.
г) Построение модулей генов при помощи анализа дифференциальной экспрессии.
д) Обеспечение интерфейса доступа к модулям генов.
е) Анализ полученных результатов.

5 Перечень графического материала (с указанием обязательного материала)

Не предусмотрено

6 Исходные материалы и пособия

а) Bioconductor. Bioconductor is an open source, open development software project to provide tools for the analysis and comprehension of high-thoughput genomic data. [Электронный ресурс]. URL: https://www.bioconductor.org/;
б) Docker. Docker is the software container platform. [Электронный ресурс]. URL: https://www.docker.com/;

в) Терри А.Браун Геномы / Терри А.Браун. - М.-Ижевск: Институт компьютерных исследований, 2011. - 921 с.

7 Дата выдачи задания: «01» декабря 2017 г.

Руководитель ВКР _____

Задание принял к исполнению _____ «01» декабря 2017 г.

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

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

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент: Беляева Екатерина Сергеевна

Наименование темы работы: Автоматическое выделение функциональных наборов генов в базе GEO с помощью дифференциальной экспрессии

Наименование организации, где выполнена работа: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Получить новую базу функциональных наборов генов.

2 Задачи, решаемые в работе:

а) изучение существующих сервисов для работы с данными экспрессии генов;
б) построение новых функциональных наборов генов из базы GEO при помощи анализа дифференциальной экспрессии;
в) анализ полученных результатов;
г) обеспечение интерфейса доступа к модулям генов.

3 Число источников, использованных при составлении обзора: 15

4 Полное число источников, использованных в работе: 18

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
2	1	0	14	0	1

6 Использование информационных ресурсов Internet: да, число ресурсов: 16

7 Использование современных пакетов компьютерных программ и технологий:

Пакеты компьютерных программ и технологий	**Параграф работы**
R (библиотеки GEOquery, limma, dplyr, stringr, data.table, magrittr, tibble)	Глава 2, Глава 3
Python 2,3 (библиотеки pandas, subprocess, re, csv)	Глава 2, Глава 3
Bash	Глава 2, Глава 3
Kotlin	Глава 3
Django	Глава 3
Java Script	Глава 3
React	Глава 3
HTML	Глава 3
CSS	Глава 3
Docker	Глава 3

8 Краткая характеристика полученных результатов: По итогу работы был реализован сервис с возможностью предоставления доступа к модулям генов конечному потребителю.

9 Гранты, полученные при выполнении работы: Грантов или других форм государственной поддержи и субсидирования в процессе работы не предусматривалось.

10 Наличие публикаций и выступлений на конференциях по теме работы: Публикаций и выступлений на конференциях не было.

Выпускник: Беляева Е.С. _____

Руководитель: Сергушичев А.А. _____

«__» _____ 20__ г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Жизнь, какой мы ее видим, создается геномами мириад организмов, с которыми мы делим нашу планету. Каждый из этих организмов обладает геномом, содержащем биологическую информацию, необходимую для построения и поддержания организма живущего в настоящий момент времени представителя данного вида [1]. Изучением геномов занимается биоинформатика – наука, стоящая на границе биологии и информатики, активно развивающаяся в течении последних десятилетий. Биоинформатика также связана с биохимией, биофизикой, экологией и многими другими областями. Открываются новые направления исследований, появляется всё больше данных, в том числе и открытых.

Геном – хранилище биологической информации, но сам по себе он не способен выдать эту информацию клетке. Использование биологической информации, содержащейся в геноме, возможно лишь благодаря согласованным действиям ферментов и других белков, которые участвуют в сложной серии биохимических реакций, называемой экспрессией генома [1].

Так, одним из направлений биоинформатики являются исследования, связанные с экспрессией генов, т.е. с процессом преобразования последовательности гена в функциональный продукт (обычно белок). Экспрессия гена может быть измерена количественно. Существует множество данных экспрессии генов. Самой известной и часто используемой базой данных экспрессии является база данных Gene Expression Omnibus (GEO). Она содержит данные о тысячах экспериментов, имеет сложную структуру. При этом каждый эксперимент также состоит из нескольких образцов, гены которых проаннотированы определенной пробой. GEO открытая, и именно она и будет использована в дальнейшей работе.

Есть множество сервисов, работающих с данными экспрессии генов. В этой работе был рассмотрен и расширен сервис GeneQuery, который является поисковым порталом, использующим данные из базы GEO. Сервис плохо работал с маленькими по числу образцов датасетами, следовательно существовала проблема изменения принципа обработки данных. Так, целью данной бакалаврской работы являлось построение функциональных наборов генов из базы GEO с помощью дифференциальной экспрессии, т.е. при помощи наблюдения за изменением активности гена в зависимости от биологического состояния. Сервис GeneQuery был расширен для поддержки новых данных. Ожидается, что

используя новую базу функциональных групп генов может быть выявлена потенциально значимая биологическая информация, а сам сервис может быть использован исследователями в области биоинформатики.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Основные понятия и определения

Биоинформатика – это междисциплинарная область, работающая с биологическими данными. Она сочетает в себе информатику, биологию, математику, статистику и ещё множество областей. Также широк и круг исследований биоинформатики, однако данная работа будет сосредоточена на таком направлении, как экспрессия генов.

Ген – единица наследственности живых организмов.

Экспрессия гена – процесс преобразования последовательности гена в функциональный продукт (обычно белок). Она может быть измерена количественно.

Дифференциальная экспрессия гена – изменение уровня экспрессии гена в зависимости от биологического состояния.

Функциональная группа генов – отвечает за конкретное биологическое явление или несколько явлений.

Для примера можно рассмотреть гликолиз – процесс расщепления глюкозы в клетках, сопровождающийся синтезом АТФ. Для протекания этого процесса необходимы ферменты, т.е. ускорители химических реакций. В гликолизе участвуют ферменты гексокиназа, глюкозофосфатизомераза, енолаза и другие. Их структуры определяются генами HK1, PGI1, ENO1 [2] и т.д. Все эти гены являются функциональной группой генов для процесса гликолиз.

Фенотип – это совокупность всех характеристик конкретного организма в определенное время, стадии развития и окружающих факторов. Например, крыса в определенном возрасте с определенным заболеванием и цветом глаз является фенотипом. А крыса того же возраста и цвета глаз, но с другим заболеванием будет уже другим фенотипом.

1.2 База данных GEO

Биоинформатика обладает большим объемом информации, который нужно где-то хранить. Для этой цели существуют множество баз данных. Накопление данных о генах, проявивших себя в биологических экспериментах, т.е. функциональных групп генов, является важной задачей потому, что эта информация может быть полезна в будущем. Самым известным открытым репозиторием для хранения данных экспрессии генов является *Gene Expression Omnibus*. Или же сокращенно *GEO*.

Проект GEO был инициирован в ответ на растущую потребность в общедоступном хранилище данных экспрессии генов [3]. Эксперименты, хранящиеся в репозитории, имеют определенную структуру. Для каждого эксперимента представлены:

— GSM (Geo SaMple) – *образцы* или *сэмплы* – эти данные содержат информацию об организмах, участвовавших в эксперименте, об условиях эксперимента.
— GSE (Geo SEries) – *серия* – объединяет в себе несколько образцов и хранит данные о числовых значениях экспрессии генов каждого образца. Как правило это 20000 генов на каждый обзазец.
— GPL (Geo PLatform) – *платформа*, на которой вычислялась экспрессия. Важным компонентом, хранящимся здесь, является информация для каждого гена о соответствии его entrez к symbol.

Количество данных GEO огромно и постоянно растет, в следствии чего база представляет исключительный интерес в биоинформатическом сообществе. Как было отмечено ранее, данная работа посвящена обработке данных именно из GEO.

1.3 Сервис GeneQuery

Проект GeneQuery [4] - это поисковый портал, основанный на автоматическом выделении наборов генов с помощью кластеризации. Работает с данными экспрессии генов из базы данных GEO для организмов Homo Sapience, Mus Musculus и Rattus Norvegicus, т.е. с человеком, мышью и крысой. На вход сервису подается набор генов. Результатом запроса являются модули генов, перекрывающиеся с запросом пользователя, и проранжированные в порядке статистической значимости. Инструмент также показывает тепловые карты экспрессии генов для каждого найденного модуля. Взаимодействие элементов сервиса представлено на рисунке 1.

GeneQuery имеит такие достоинства, как высокая скорость обновления данных (новые данные из GEO можно портировать в GeneQuery), быстрая скорость выдачи результата для пользователя, простота использования. Сервис достаточно хорошо работает для средних по размеру датасетов (по числу образцов), но не подходит для маленьких. Для последних можно использовать анализ дифференциальной экспрессии, что и сделано в альтернативной версии GeneQuery - в GeneQueryDE.

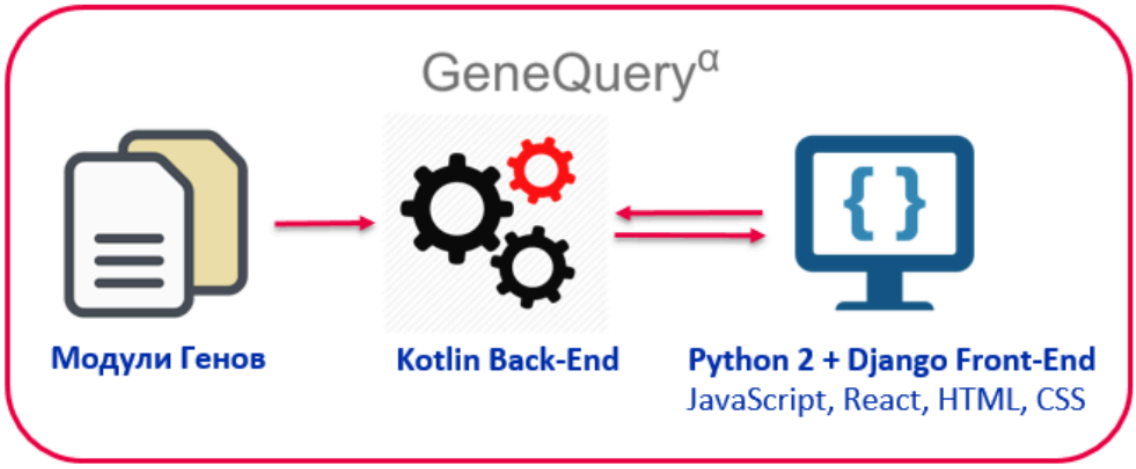


Рисунок 1 – Схема сервиса.

1.4 Сервис GEOracle

GEOracle [5] – инструмент для поиска данных по экспрессии генов по базе GEO, открытый для использования летом 2017 года. Использует технологии машинного обучения для автоматического выделения нужных экспериментов, их объединения в группы и подсчета дифференциальной экспрессии [6]. Сервис получает на вход набор экспериментов, а в результате их обработки, выдает значения экспрессии генов в зависимости от указанного фактора и, также, показывает тепловые карты зависимости экспериментов от генов.

1.5 Сервис ARCHS4

ARCHS4 [7] – сервис с обширным функционалом. Работает с данными видов Homo Sapience и Mus Musculus. Имеет функции поиска, позволяющие просматривать данные с помощью аннотаций метаданных, сходства структур и функциональной представленности [8]. Также сервис позволяет визуализировать данные в 3D представлении. На вход может получать ген либо набор генов в верхней/нижней регуляции (как дополнительные параметры можно указать ткани, органы). Результатом запроса станут данные о гене, средняя экспрессия, отображение данных на 3D модели. Сервис также позволяет загружать полученные результаты.

1.6 Сервис GEO Profiles

GEO Profiles [9] – это инструмент поиска по базе GEO. Обладает широкими возможностями, подразумевая то, что, как и в обычном поисковике, можно

вводить любой текст в надежде получить результат. В данном поисковике запросом может быть свободный текст, но также существуют некие шаблоны ввода. Например, можно искать конкретный ген, эксперимент или организм. А можно какую-то комбинацию выше написанного. В зависимости от запроса пользователь может получить информацию об эксперименте со ссылкой на него, информацию о платформах экспериментов, информацию о генах и многое другое. Также стоит отметить, что GEO Profiles снабжен достаточно полной документацией, что позволяет быстро разобраться в особенностях работы с ним.

1.7 Язык программирования R

R – это бесплатная программная среда для статистических вычислений и графики [10]. Он работает под платформами UNIX, Windows и MacOS. Язык поддерживается большим сообществом разработчиков и активно используется для статистики и анализа данных. Также широко он применяется в биоинформатике. На R написаны десятки библиотек для биоинформатического анализа данных, для работы с базами данных в этой области, а также для представления результата. Проект, объединяющий в себе полезные пакеты для работы с базой GEO называется Bioconductor. Он и был использован в данной работе.

1.8 Тест Фишера

Тест Фишера – это тест статистической значимости, используемый для анализа таблиц сопряженности, т.е. таблиц в матричной форме, отображающих распределения переменных [11]. Тест используется для изучения случайности связи между двумя видами классификации данных и применяется для категориальных данных, представляющих из себя результат классификации.

1.9 P-значение

P-значение (P-value) – величина, которая используется для тестирования статистических гипотез и является одним из возможных методов оценки вероятности ошибки при отклонении нулевой гипотезы.

1.10 Поправка Бонферрони

Одна из мер, позволяющих получить групповую вероятность ошибки первого рода. Данный метод основан на формуле:

$$p_i < \frac{\alpha}{m} \tag{1}$$

, где p_i – вероятность *i*-го события,

m – количество событий,

α – порог значимости – некоторое число (в данной работе = 0.01).

Использование формулы1 позволяет отклонить гипотезы, неудовлетворяющие ей, т.е. уменьшить колличество ложноположительных результатов [12].

1.11 Контейнеризация

Контейнеризация – упаковывание в контейнер, т.е. в тип, позволяющий инкапсулировать в себе объекты других типов [13] и запущенный в изолированном окружении. Одной из самых популярных технологий контейнеризации является Docker [14], предоставляющий удобный интерфейс для управления контейнером.

Docker – это открытый проект для разработки и эксплуатации приложений, разработанная для быстрого выкладывания приложений. При помощи легковесной платформы контейнерной виртуализации, используя процессы и утилиты, помогающие управлять и выкладывать приложения [15], docker отделяет приложение от инфраструктуры и позволяет обращаться с инфраструктурой как с управляемым приложением. Технология docker позволяет запускать приложение, безопасно изолированное в контейнере. Более того, безопасная изоляция позволяет запускать на одном хосте много контейнеров одновременно.

Docker включает в себя три основных компонента: образы(images), реестр(registries) и контейнеры.

Docker-образ – это шаблон read-only, являющийся компонентой сборки. Например, образ может содержать операционную систему Ubuntu с java-машиной и приложением на ней. Образы нужны для создания контейнеров. Docker позволяет создавать новые образы, обновлять те, что есть. Также образы можно скачивать, подразумевая то, что они были созданны другими людьми. Для создания образа пишется Dockerfile.

Docker-реестр – это место хранения образов, являющееся компонентой распростронения. Есть приватные и публичные реестры, из которых можно скачать или в которые загружать образы. Публичный Docker-реестр – это Docker Hub [16], где хранится большая коллекция образов.

Контейнеры похожи на папки с данными – это компонента работы. Так, в контейнерах находится все, что нужно для работы приложения – операционная система, требуемые программные зависимости, само приложение. Каждый

контейнер создается из образа. Контейнеры можно создавать, запускать, останавливать, переносить или удалять. Каждый контейнер изолирован и является безопасной платформой для приложения.

1.12 Цель и актуальность работы

Учитывая то, что сервис GeneQuery плохо работает с маленькими по числу образцов экспериментами, требовалось изменить механизм работы сервиса. Так целью данной работы было построение наборов генов по экспериментам из базы GEO с помощью анализа дифференциальной экспрессии и обеспечить интерфейс доступа к ним.

Портал GeneQuery расширен для поддержки новых данных до версии GeneQueryDE. Сервис может быть использован исследователями в области биоинформатики. Ожидается, что используя новую базу функциональных групп генов может быть выявлена потенциально значимая биологическая информация.

1.13 Задачи

Основываясь на цели, обозначенной в предыдущем пункте, были выделены следующие задачи, решенные в ходе работы:

— Выгружены данные экспериментов и аннотаций из GEO.
— Данные обработаны.
— Построены наборы генов с помощью анализа дифференциальной экспрессии.
— Осуществленна поддержка новых данных сервисом GeneQuery.
— Сервис запущен для массового использования.

Выводы по главе 1

В данной главе был произведен обзор предметной области. А именно, рассмотрены основные биологические понятия, являющиеся базовыми для понимания дальнейших действий. Рассмотрены применненные технологии и сервисы, на которых строится данная работа. Также обозначены цели работы и ее значимость, показаны задачи, решаемые для достижения цели.

ГЛАВА 2. ПОСТРОЕНИЕ МОДУЛЕЙ ГЕНОВ

2.1 Загрузка данных

База данных GEO имеет сложную структуру. Например, датасет GSE61055 находится по адресу *https://ftp.ncbi.nlm.nih.gov/geo/series/GSE61nnn/GSE61055/matrix/GSE61055_series_matrix.txt.gz*, а аннотация к нему лежит на *https://ftp.ncbi.nlm.nih.gov/geo/platforms/GPL6nnn/GPL6887/annot/GPL6887.annot.gz*. При скачивании данных, структура GEO была сохранена. Для загрузки использовались языки Python 3 и Bash.

```
import subprocess
import csv

def download_data(file_name, kind_data):
    with open(file_name,'r') as tsvin:
        tsvin = csv.reader(tsvin, delimiter='\t')
        indexes = []

        for (i, row) in enumerate(tsvin):
            if i == 0:
                if kind_data == 'series':
                    indexes = [row.index(j) for j in ['Series', 'Series_url']]
                else:
                    indexes = [row.index(j) for j in ['Platforms', 'Platforms_url
                        ']]
            else:
                try:
                    dif_string = 'https://ftp.ncbi.nlm.nih.gov/'
                    path = row[indexes[1]].replace(dif_string, '')
                    path = path[:(path.rindex('/')+1)]
                    directory = "../" + path

                    subprocess.call(["mkdir", "-p", directory])
                    subprocess.call(["wget", "-c", "-P", directory, row[indexes
                        [1]]])

                except:
                    pass
```

Листинг 1 – Загрузка данных из GEO.

Основная функция загрузки приведена на листинге 1. Всего было загружено 74177 датасетов типа GSE (с экспрессией генов) и 790 GPL (аннотаций) для видов Homo Sapience, Mus Musculus и Rattus Norvegicus.

Примеры данных типа GSE и GPL приведены на рисунках 2 и 3 соответственно.

Далее GSE были сопоставлены с аннотациями и выяснилось, что 29098 датасетов пригодно для построения таблиц дифференциальной экспрессии.

	GSM1496031	GSM1496032	GSM1496033	GSM1496034	GSM1496035
ILMN_1212602	177.030	222.587	203.629	201.666	360.088
ILMN_1212603	178.454	168.911	169.766	152.873	157.866
ILMN_1212605	528.909	624.454	670.606	413.628	725.952
ILMN_1212607	175.157	157.038	170.438	201.061	216.100
ILMN_1212612	171.559	181.618	176.489	168.857	171.898
ILMN_1212614	150.927	186.845	151.915	166.459	165.649
ILMN_1212619	236.138	259.156	329.527	155.606	290.285
ILMN_1212623	162.638	251.812	170.803	238.335	176.895
ILMN_1212625	868.615	775.291	944.976	557.115	179.897
ILMN_1212626	392.461	342.360	474.882	424.004	279.115
ILMN_1212628	156.348	139.041	166.522	153.823	156.356

Рисунок 2 – Таблица с экспрессией генов для GSE61055.

	ID	ENTREZ_GENE_ID	symbol
ILMN_1243094	ILMN_1243094	21835	Thrsp
ILMN_1238674	ILMN_1238674	212772	Arl14ep
ILMN_2429159	ILMN_2429159	240411	Loxhd1
ILMN_2754227	ILMN_2754227	68988	Prpf31
ILMN_2603725	ILMN_2603725	56378	Arpc3
ILMN_2664884	ILMN_2664884	74257	Tspan17
ILMN_1229397	ILMN_1229397	20510	Slc1a1
ILMN_3003575	ILMN_3003575	338403	Cndp1
ILMN_1257702	ILMN_1257702	107045	Lars
ILMN_2658355	ILMN_2658355	223337	Ugt3a2
ILMN_2521511	ILMN_2521511	76757	Trdn

Рисунок 3 – Аннотация GPL6887.

2.2 Автоматическая обработка

После получения таблицы с матчингом GSE с GPL, для каждого эксперимента надо было построить модули генов с применением анализа дифференциальной экспрессии. Для этого, для каждого датасета были выполнены действия:

— Выявлены уникальные условия каждого эксперимента.
— Сопоставлены пробы и гены.
— Нормализованны данные.
— Построены пары условий.
— Построены таблицы дифференциальной экспрессии.

Для всего вышеперичисленного использовался язык R с применением инструмента Bioconductor. Bioconductor – это проект на R, содержащий множество

библиотек для анализа геномных данных [17]. В том числе имеет библиотеку для взаимодействия с базой данных GEO. Так, работая на R, были использованы пакеты GEOquery и limma.

2.2.1 Выявление уникальных условий каждого эксперимента

GSE датасет содержит таблицу с указанием информации по каждому образцу, участвовавшему в эксперименте. В том числе, содержатся условия эксперимента для каждого образца. Эти данные записаны в столбцах «characterics», которых может быть от 1 до n. Требовалось автоматически выделять биологические состояния.

Поиск нужных для парсинга столбцов на листинге 2.

```
getCharacteristicsColumns <- function(gse) {
  col <- colnames(pData(gse))
  characteristics <- c()
  for (ch in col) {
    if (grepl("characteristics", ch)) characteristics <- c(characteristics, ch)
  }
  return(characteristics)
}
```

Листинг 2 – Поиск столбцов «characterics».

Рассматриваемый столбец не должен был быть пустой или содержать NA, т.ч. сначала были отобраны колонки, удовлетворяющие этим условиям. Далее нужно было оставить только значимые состояния. В колонках «characterics» мог быть написан любой текст, содержащий любые символы и цифры. Прежде всего надо было убедиться, что в столбце написано не простое перечисление номеров образцов. Такие столбцы исключались из рассмотрения. На листинге 3 приведена проверка на значимость цифр. Если массив цифр, отсортированный по возрастанию, содержал целые, не дублирующиеся цифры и имел на концах значения единицы и количества образцов, то такие цифры признавались простым перечислением номеров образцов, и содержащий их столбец дальше в построении условий не участвовал.

Следующая проблема, которую надо было решить, заключалась в работе с длинными последовательностями текста в «characterics», т.е. в обработке сложных условий. Условие признавалось сложным, если в нем было больше некоторого количества определенных знаков (листинг 4). Такие условия обрабатывались определенным способом. Это было сделано не только из-за того,

```
isUsefulNumbers <- function(numbers) {
  maxNumber <- length(numbers)
  numbers <- sort(numbers, decreasing = F)

  isInteger <- T
  tryCatch({
    stopifnot(all(numbers == floor(numbers)))
  }, error = function(e) {
    isInteger <- F
  })

  isUseful <- T
  if (isInteger && numbers[1] == 1 && numbers[maxNumber] == maxNumber
      && length(unique(numbers)) == maxNumber)
    isUseful <- F

  return(isUseful)
}

getConditionsFromUniqValues <- function(charColumns, conditionsList,
                                        explanatoryTable, usedMarkers) {
  values <- sub("^.*: ", "", charColumns[[1]])
  numerics <- is.na(suppressWarnings(as.numeric(values)))

  if ((length(numerics[numerics==FALSE]) == 0) ||
      (length(numerics[numerics==FALSE]) != 0 && isUsefulNumbers(as.numeric(
          values)))) {

    tmpCondition <- c()
    beforeColon <- gsub(" ", ".", sub(":.*", "", charColumns[[1]]))
    values <- paste(beforeColon, ".", values, sep="")

    ...
```

Листинг 3 – Проверка значимости столбца «characterics».

что явное использование некоторых символов приводило к ошибкам заполнения неких структур для построения дифференциальной экспрессии, но также и из-за эстетических соображений и восприятия (т.к. в конечном счете из каждого «простого» условия удалялись все «сомнительные» символы, а пробелы заменялись на точки).

```
isTooComplicatedCondition <- function(condition) {
  if (length(unlist(gregexpr(pattern = "\\+", condition))) > 1
      || length(unlist(gregexpr(pattern = "-", condition))) > 1
      || length(unlist(gregexpr(pattern = " ", condition))) > 1)
    return(TRUE)
  return(FALSE)
}
```

Листинг 4 – Проверка сложности условия.

Так как по результату работы программы создавались файлы модулей генов с подписями условий, по которым была вычисленна дифференциальная экспрессия, то нужно было быть аккуратным с символами, которые можно записывать в название, ведь не все символы допускаются для названия файла. По этой причине и также по причине, описанной выше, некоторые символы удалялись из условия, а пробелы заменялись на точки (листинг 5).

```
v <- gsub("[-\\+\\/;:\\.\\?!,\\\\]", ' ', v) %>%
              gsub("\\s+", ' ', .) %>%
              gsub("(^\\s+)|(\\s+$)", '', .) %>%
              gsub(" ", ".", .)
```

Листинг 5 – Удаление некоторых символов и замена пробелов на точки.

Немного не так обстояла ситуация со «сложными» условиями. Эти условия не подвергались чистке символов, т.к. каждому такому условию присваивалась буква (или комбинация букв), под которой условие сохранялось в специальном файле со списком всех подобных условий (explanatoryTable). Файлы модулей генов в этом случае содержали эти буквы, а не исходный текст биологического состояния. Генерация маркеров для возможных «сложных» условий представлена на листинге 6.

```
createMarkersForComplicatedConditions <- function() {
  usedMarkers <- list()
  for (i in 1:26) usedMarkers[intToUtf8(64+i)] <- 0

  extraMarkers <- permutations(
    n=12, r=2, v=names(usedMarkers)[1:12], set=T, repeats.allowed=T)

  for (i in 1:(length(extraMarkers))/2) {
    cond <- paste(extraMarkers[[i,1]], extraMarkers[[i,2]], sep="")
    usedMarkers[cond] <- 0
  }
  return(usedMarkers)
}
```

Листинг 6 – Генерация маркеров для замены «сложных» условий.

Обработанные условия из каждого столбца «characticstics» для каждого образца затем соединялись вместе через символ «_». Например, как здесь: *ATF3.deficiency_IFNbeta*.

Как можно видеть на листинге 8, возможна ситуация, когда уникальные биологические состояния выделить не удавалось. Это могло быть связано с про-

```
handleComplicatedCondition <- function(cond, explanatoryTable, usedMarkers) {

  newConditionName <- isContainedInExplanatoryTable(explanatoryTable, cond)
  if (newConditionName == "") {
    markerUse <- getFreeMarker(usedMarkers)
    newConditionName <- markerUse$marker
    usedMarkers <- markerUse$usedMarkers

    explanatoryTable <- rbindlist(list(explanatoryTable, data.table(
      "characteristics"=cond, "marking"=newConditionName)))
  }
  return(list("condition"=newConditionName,
              "explanatoryTable"=explanatoryTable,
              "usedMarkers"= usedMarkers))
}

...

if (isTooComplicatedCondition(v)) {
    resultCondition <- handleComplicatedCondition(oldV,
                                explanatoryTable, usedMarkers)
```

Листинг 7 – Добавление условий в explanatoryTable.

пусками данных в столбцах «characterics» или же с тем, что условия не несли в себе универсального значения. Например, в датасете, состоящем из восьми образцов, в одной из колонок «characterics» могли быть написаны цифры от 1 до 8, соответствующие просто номерам образцов. Как было написано ранее, такие случаи игнорировались и не участвовали в дальнейшем анализе.

```
characteristics <- getCharacteristicsColumns(gse)
conditionLists <- list()

...

if (length(characteristics) > 0) {
    message("There are characteristics columns in the samples table.")
    conStructure <- getConditionsFromCharacteristics(gse, characteristics)
    conditionLists <- conStructure$conditionsList
    explanatoryTable <- conStructure$explanatoryTable
  }

  if (length(conditionLists) > 0 && length(unique(unlist(conditionLists))) > 1) {
    pData(gse)$condition <- fillGseConditionColumn(conditionLists)
    ...

  } else
    message("Characteristics columns in the samples table don't exist or were
        unhelpful.")
```

Листинг 8 – Выделение условий.

Все выявленные состояния были сохранены в столбце «condition», из которого затем были построены пары состояний и, основываясь на этом, модули с дифференциальной экспрессией генов.

2.2.2 Сопоставление проб и генов

Как можно видеть на рисунке 3 аннотация к чипу содержит id проб, генов и названия генов. Однако данные в таблице могут дублироваться. Чтобы избежать неточностей, было решено удалить дубликаты данных и рассматривать только уникальные значения (листинг 9).

```
createGenesSymbolsTable <- function(gpl) {
  gpl <- read.table(gpl, header = T, sep = "\t",
                    colClasses = c("character", "character", "integer"))
  gpl <- gpl[!duplicated(gpl$id),]
  gpl <- gpl[!duplicated(gpl$gene_id),]

  gpl <- gpl %>%
    set_rownames(.$id) %>%
    select(ID = id, ENTREZ_GENE_ID = gene_id, symbol = gene_symbol)

  return(gpl)
}
```

Листинг 9 – Выделение условий.

Используя данные типа GSE (как на рисунке 2) и GPL (рисунок 3), на листинге 10 выделяются 7000 генов с наибольшей экспрессией, а также к таблице добавляются данные о entrez генов и их symbol, основываясь на сопоставлении проб. На основе этого набора генов в 7000 в дальнейшем строются модули с дифференциальной экспрессией.

2.2.3 Нормализация данных

В процессе обработки данных были совершены следующие действия:

— Убраны дублирующиеся данные.
— Убраны данные с пропусками.
— Отобраны 7000 генов с самой высокой экспрессией (листинг 10).
— Логарифмизация данных при большом размахе значений экспрессии (листинг 11).

Как правило, после удаления дублирующихся данных и данных с пропусками в рассмотрении оставался набор, состоящий примерно из 20000 генов.

```
collapseData <- function(gse, gpl, FUN=median) {
  ranks <- apply(exprs(gse), 1, FUN)
  ranks <- data.frame(r = ranks, i = seq_along(ranks))
  table <- inner_join(rownames_to_column(gpl), rownames_to_column(ranks),
           by="rowname") %>%
           mutate(j = seq_along(symbol))
  t <- table[order(table$r, decreasing=T), ][1:7000,]
  keep <- t$i
  res <- gse[keep, ]
  rownames(res) <- table$ENTREZ_GENE_ID[t$j]
  fData(res)$symbol <- t$symbol
  return(res)
}

...

es <- collapseData(gse, gpl)
```

Листинг 10 – Выделение 7000 генов с максимальной экспрессией.

```
if (max(exprs(es)) - min(exprs(es)) > 100)
        exprs(es) <- normalizeBetweenArrays(log2(exprs(es)+1), method="quantile")
```

Листинг 11 – Логарифмизация данных.

2.2.4 Построение пар условий

Так как дифференциальная экспрессия – это изменение активности гена в зависимости от конкретного биологического состояния, то на данном этапе требовалось составить пары условий с различием в одно состояние. Для этого обрабатывался столбец *pData(gse)$condition*, получение которого показано на листинге 8.

Так, например парой условий является пара *ATF3.deficiency_IFNbeta* против *ATF3.deficiency_untreated*, где различно биологическое состояние *IFNbeta* и *untreated*.

2.2.5 Построение таблиц дифференциальной экспрессии

Для каждой пары условий, полученной по результатам предыдущего пункта, требовалось построить таблицу с дифференциальной экспрессией. Построение показано на листинге 12. При помощи функции *lmFit* заполняется линейная модель на основе данных об экспрессии генов и дизайн-матрицы. Дизайн-матрица – это таблица с образцами в строчках и условиями в столбцах. В ячейке стоит 1, если условие соответствует данному образцу и 0, если в данном образце его нет.

Далее вызывалась функция *fitLinearModel*, на вход которой подавались линейная модель, пары условий, дизайн-матрица и количество обрабатываемых генов. Для каждой пары условий строилась контрастная матрица, на основе нее вычислялись расчетные коэффициенты линейной модели. Далее для модели расчитывалась эмпирическая Баесова статистика, включающая t-статистику, p-value и т.д. По просшествии данных преобразований, модель представлялась в виде таблицы с данными дифференциальной экспрессией генов.

```
es.design <- model.matrix(~0+condition, data=pData(es))
fit <- lmFit(es, es.design)
deSize <- dim(exprs(es))[1]

...

fitLinearModel <- function(fit, conditions, design, deSize) {
  deList <- list()
  for (i in 1:length(conditions)) {

    contrasts <- makeContrasts2(
              c("condition", conditions[[i]]$firstCon, conditions[[i]]$secondCon
                  ),
              levels=design)

    fit2 <- contrasts.fit(fit, contrasts)

    df.residual <- unique(fit2$df.residual)
    if ((length(df.residual) == 1 && df.residual[1] != 0) ||
        (length(df.residual) != 1)) {

      fit2 <- eBayes(fit2)

      deList[[i]] <- data.table(
        topTable(fit2, adjust.method="BH", number=deSize, sort.by = "B"),
        keep.rownames = T)
    }
  }
  return(deList)
}
```

Листинг 12 – Построение модулей генов при помощи дифференциальной экспрессии.

Пример таблицы приведен на рисунке 4. Модуль содержит 7000 генов и включает в себя информацию о entrez гена, его названии, средней экспрессии и различные статистические метрики.

Модули получилось построить для 15956 экспериментов.

	A	B	C	D	E	F	G	H
1	rn	symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B
2	20715	Serpina3g	6.030547534	10.6892139601	63.5082547013	5.52118363962044E-16	3.86482854773431E-12	25.902659987
3	17329	Cxcl9	6.3356914437	11.1081628067	50.3345597005	7.89521658726984E-15	2.76332580554445E-11	23.9421695132
4	433470	AA467197	4.8914732002	10.1720346836	48.1240467197	1.31922974800399E-14	3.07820274534264E-11	23.5316561252
5	16153	Il10	4.2295908369	10.1817500872	45.4194949439	2.55467298396242E-14	4.47067772193424E-11	22.9893089008
6	21939	Cd40	4.9775575043	11.5357658018	43.8679282341	3.79952707161957E-14	5.3193379002674E-11	22.6563788376
7	238393	Serpina3f	4.4428201703	9.8144074483	42.9611188811	4.82280357789186E-14	5.40983082981626E-11	22.4538784664
8	667373	Ifit1bl1	6.4006078219	10.8477655629	42.5309949304	5.40983082981626E-14	5.40983082981626E-11	22.3556956694
9	215900	Fam26f	4.9573906927	10.6700218457	38.0833579053	1.90701284723663E-13	1.66863624133205E-10	21.2525205748
10	20293	Ccl12	4.8845394113	10.2466374648	33.9997416566	6.93922387707301E-13	5.39717412661234E-10	20.0762550528
11	57444	Isg20	4.4238649325	9.8502322878	33.6162522149	7.89498185435406E-13	5.52648729804784E-10	19.9564774078
12	20306	Ccl7	5.0046765276	10.3295731048	33.1902740896	9.12726442872727E-13	5.80825918191736E-10	19.8213859834
13	14469	Gbp2	6.1048408097	12.3552471869	30.4561553354	2.42406699001782E-12	1.12203333713762E-09	18.8995639495
14	20440	St6gal1	-3.0680562242	9.9700985417	-30.3521433526	2.52004868904069E-12	1.12203333713762E-09	18.8625045566
15	434341	Nlrc5	3.5299944374	10.1806904955	30.2039436948	2.66400828460823E-12	1.12203333713762E-09	18.8094339419
16	20296	Ccl2	3.2613845656	9.3364759317	30.1773683194	2.69075540828832E-12	1.12203333713762E-09	18.7998838549
17	384009	Glipr2	3.4399433181	9.3336980256	30.1570985832	2.71135219609879E-12	1.12203333713762E-09	18.7925929003
18	55932	Gbp3	5.7307955453	11.0316049778	30.139529113	2.72934372101617E-12	1.12203333713762E-09	18.7862684181
19	623121	Pydc4	3.4673625155	9.1939253204	29.9923720691	2.88522858121102E-12	1.12203333713762E-09	18.7331204748
20	107607	Nod1	3.6845924408	10.2540842923	29.433783429	3.57115125319917E-12	1.31434428741991E-09	18.5284760793
21	14190	Fgl2	4.3861676633	9.9567512421	29.3036076285	3.75526939262832E-12	1.31434428741991E-09	18.4801135337
22	17858	Mx2	4.9663078784	10.6244240359	28.4180215472	5.31776893282039E-12	1.77258964427346E-09	18.1441434238
23	15945	Cxcl10	5.0658835144	10.2232050734	28.1323356722	5.9627956451799E-12	1.89725315982997E-09	18.0331002325
24	54199	Ccrl2	3.3698242532	9.651375616	27.9820867281	6.33573059337887E-12	1.89888106503496E-09	17.974164357
25	81913	Bambi-ps1	3.292733862	9.3821770387	27.8279828643	6.74470365158551E-12	1.89888106503496E-09	17.9133279097
26	74481	Batf2	3.7166394137	9.5759254503	27.8145391522	6.78171808941057E-12	1.89888106503496E-09	17.9080018755
27	16365	Acod1	5.1400474315	10.5609573344	27.21930329	8.66390684701144E-12	2.33259030496462E-09	17.6691132718
28	22035	Tnfsf10	3.7883540799	9.4726720517	26.8191061274	1.02449835098398E-11	2.65610683588439E-09	17.5050462442

Рисунок 4 – Таблица дифференциальной экспрессии генов.

2.3 Верхняя и нижняя регуляции генов

Верхняя и нижняя регуляция генов – это сравнение их активностей относительно друг друга. На предыдущем этапе были постоены таблицы генов при противопоставлении двух разных биологических состояний. Так, на данном этапе для каждого из модулей дифференциальной экспрессии генов требовалось построить дескриптивную выборку. По максимальному значению t-статистики отбиралось 200 генов, что соответствует верхней регуляции и по минимальному значению t-статистики отбиралось 200 генов, соответствующих нижней регуляции (листинг 13). Полученные модули сохранялись.

```
gse <- read.table(nameTables, header=T, sep="\t",
                          fill = TRUE, quote = "", colClasses=colTypes)

up <- gse[order(gse$t, decreasing=TRUE),][1:200,]
down <- gse[order(gse$t),][1:200,]
```

Листинг 13 – Получение модулей по 200 генов.

На основе модулей строились *.gmt* файлы (рисунок 5) для каждого вида (человек, мышь, крыса). Файлы содержали гены из экспериментов. А именно, для каждого эксперимента было представленно название, универсум (7000 генов), номера модулей и соответствующие выборки по 200 генов. Для каждого

.*gmt* файла был построен файл с аннотацией (рисунок 6), содержащий название эксперимента с номером модуля и название модуля.

```
GSE56963-GPL6885#0
66402,56291,14545,15388,75686,18679,66943,108097,16905,214951,19122,66397,112407,14181,26
GSE56963-GPL6885#1
66402,15388,75686,66943,16905,19122,112407,432940,104886,231252,11363,18845,11370,269023,
GSE56963-GPL6885#2
56291,14545,18679,108097,214951,66397,14181,268490,244923,231872,11692,67956,12322,69260,
GSE56963-GPL6885#3
66402,75686,18036,13039,103551,16009,216805,231130,217039,79555,68738,80744,84095,71732,5
```

Рисунок 5 – Пример .*gmt* файла.

```
GSE56963-GPL6885#0      universe
GSE56963-GPL6885#1      up in 'genotype.variation.Tg' compared to 'genotype.variation.wt' on 'treatment.Veh'  background
GSE56963-GPL6885#2      up in 'genotype.variation.wt' compared to 'genotype.variation.Tg' on 'treatment.Veh'  background
GSE56963-GPL6885#3      up in 'genotype.variation.Tg' compared to 'genotype.variation.wt' on 'treatment.4B'  background
```

Рисунок 6 – Пример аннотации к .*gmt* файлу.

По итогам выборок по 200 генов, всего было получено 844324 новых модуля, которые затем были встроены в обработку сервиса GeneQueryDE.

Выводы по главе 2

В данной главе был описан основной этап работы, который заключался в загрузке данных из базы GEO и построении модулей генов. Приведены цифры обработки данных и итоговые результаты.

ГЛАВА 3. WEB-СЕРВИС ДЛЯ ПОИСКА ФЕНОТИПОВ В МОДУЛЯХ ПО ДИФФЕРЕНЦИАЛЬНОЙ ЭКСПРЕССИИ

После получения модулей генов, *.gmt* файлов и файлов с аннотациями требовалось расширить сервис GeneQuery для возможности работы с новыми данными. Так была создана версия проекта GeneQueryDE.

3.1 Схема сервиса

Взаимодействие элементов системы представлено на рисунке 7.

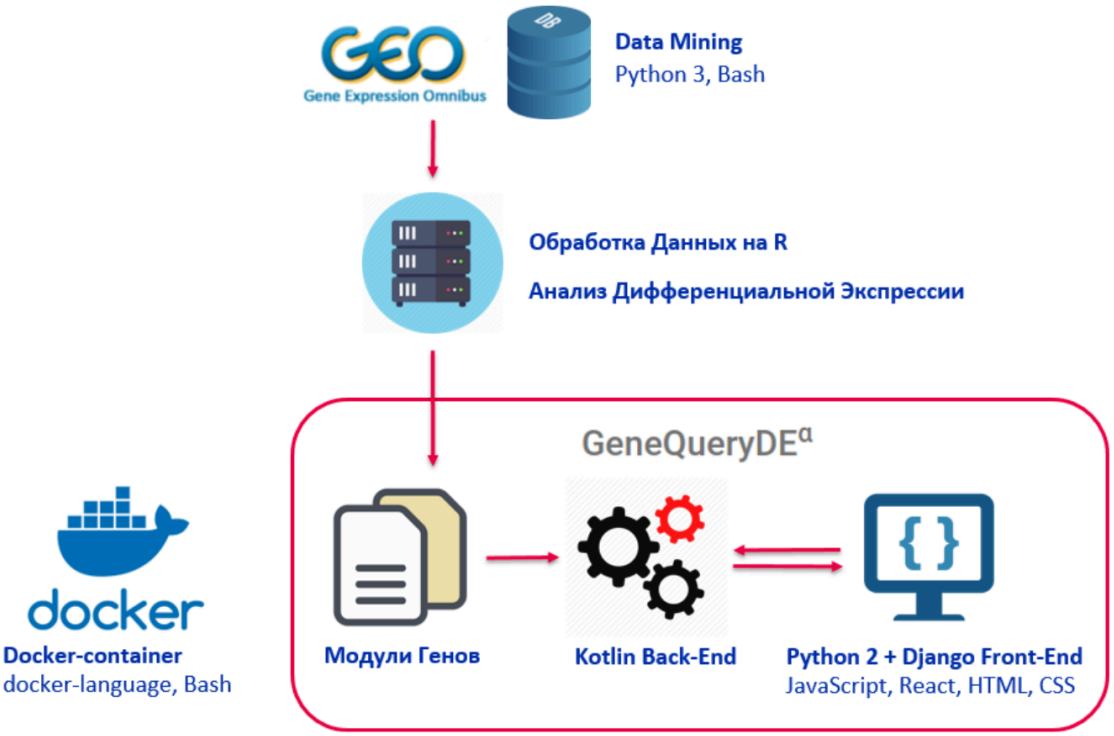


Рисунок 7 – Схема сервиса GeneQueryDE.

Подводя итог всему выше сделанному, с начала данные были загружены из базы GEO на сервер. Для этого использовались языки Python 3 и Bash. Далее на сервере происходила их обработка, и при помощи анализа дифференциальной экспрессии строились модули генов. Для этого использовался язык R.

После того, как модули генов были построены, сервис GeneQuery был расширен для поддержки новых данных до версии GeneQueryDE. Сам сервис состоит из двух репозиториев - front-end части на Python 2 + Django и back-end части на Kotlin. После, GeneQueryDE с новыми данными был упакован в docker-контейнер и запущен. Сейчас сервис доступен по адресу *http://genome.ifmo.ru/genequery-de*.

3.2 Back-end часть

На рисунке 8 показаны основные элементы структуры back-end репозитория GeneQueryDE.

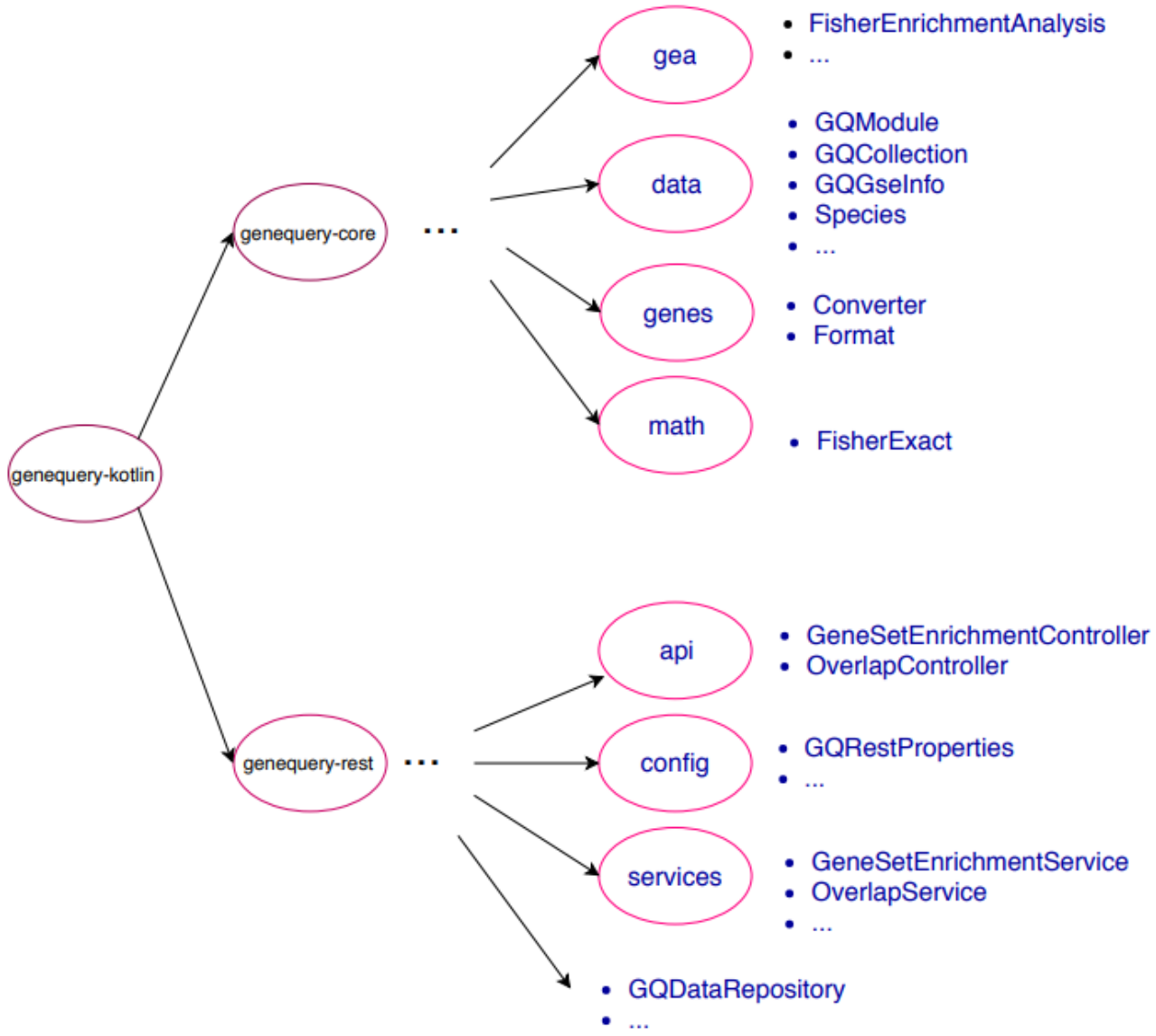


Рисунок 8 – Back-end часть GeneQueryDE.

Как можно видеть, внутри себя проект разделяется на две логические составляющие – *genequert-rest* и *genequery-core*, связанные соответственно с взаимодействием с front-end частью и обеспечивающие базовый функционал. Каждая составляющая имеет сложную иерархию пакетов. Основные из них показаны на рисунке. Также на схему вынесены главные файлы, содержащиеся в данных пакетах.

3.2.1 Пакет *data*

Пакет содержит такие файлы как *GQModule*, *GQCollection*, *GQGseInfo*, *Species* и т.д., отвечающие за хранение и доступ к разным структурам вроде модулей генов, экспериментов, видов организмов.

Класс *GQModule* представляет объект модуля. Модули инициализируются при старте приложения и чтении *.gmt* файлов с аннотациями. Класс модуля хранит информацию об эксперименте, из которого он получен(gse), о соответствующей эксперименту аннотации (gpl), о номере модуля, о виде организма, которому модуль принадлежит. Также модуль включает в себя информацию о своем имени и id всех генов, составляющих модуль. Класс в той или иной мере предоставляет доступ ко всем хранящимся данным.

Файл *GQCollection* занимается непосредственно чтением *.gmt* файлов и аннотаций к ним, т.е. содержит методы для чтения файлов и в процессе чтения создает объекты модулей. По мимо этого содержит класс *GQModuleCollection*, позволяющий выполнять некоторые действия над коллекцией модулей – например, сгруппировать модули по организмам.

Файл *GQGseInfo* хранит номера экспериментов (напр. GSE61055) и их цифровую часть – id (напр. 61055). Также при запуске сервиса он читает и сохраняет названия экспериментов из соответствующего файла. Класс *GQGseInfoCollection* принимает итератор с объектами GSE и предоставляет методы получения эксперимента по id, по номеру и т.д.

Класс *Species* хранит организмы, с которыми работает сервис (human, mouse и rat) и проверяет, что поданный на вход организм действительно может обработаться проектом.

3.2.2 Пакет *genes*

Пакет содержит файлы *Converter* и *Format*.

Файл *Converter* содержит функции для чтения файлов, в которых гены представлены разными способами (entrez, symbol, refseq, ensembl). Функции читают файлы при старте сервиса, затем матчат все форматы гена и сохраняют полную информацию по каждому гену в *OrthologyMapping* класс.

Файл *Format* содержит класс перечислений *GeneFormat* с допустимыми представлениями генов, методами определения формата и проверками на нужный формат.

3.2.3 Пакеты *gea* и *math*

Данные пакеты содержат файлы *FisherEnrichmentAnalysis* и *FisherExact*, отвечающие за расчет статистической значимости модулей генов. Класс *FisherExact* представляет собой непосредственно реализацию формулы теста Фишера, в то время как функция *findBonferroniSignificant* из файла *FisherEnrichmentAnalysis* подает аргументы для формулы. Файл *FisherEnrichmentAnalysis* также содержит класс *EnrichmentResultItem*, хранящий итоговую структуру данных, которая отдается во front-end представление. Структура содержит модули генов, связанные с ними gse и gpl, расчетное p-value, полученное p-value после поправки Бонферрони, название и номер модулей, число перекрытий генов из модулей с запросом пользователя и т.д.

3.2.4 Пакет *api*

Пакет *api* – это пакет контроллеров, общающихся с front-end частью сервиса по средством матчинга url и принимающих и отправляющих json-структуры. Пакет содержит 2 контроллера – это классы *GeneSetEnrichmentController* и *OverlapController*.

Класс *GeneSetEnrichmentController* – основной. Когда пользователь отправляет запрос с набором генов на сервер, именно данный контроллер принимает его и выдает список модулей генов, соответствующий структуре, описанной в предыдущем пакете.

Запрос на контроллер *OverlapController* может прийти только после того, как пользователь получил модули генов, они отобразились в табличном представлении, и пользователь решил посмотреть на перекрытые гены в каком-либо модуле. Для того, чтобы получить подробный список перекрытых генов, происходит обращение к классу *OverlapController*.

3.2.5 Пакет *config*

В данном пакете находится только один класс – это *GQRestProperties*, который содержит информацию о том, какие файлы надо читать при запуске сервиса. Здесь представленны пути и названия файлов *.gmt*, аннотаций к ним, файлы с различными форматами генов, названиями экспериментов и т.д.

3.2.6 Пакет *services*

Содержит классы *GeneSetEnrichmentService* и *OverlapService*. Данные классы вызываются из соответствующих контроллеров и являются прослойками для сбора информации, которая затем отправляется пользователю.

Сервис *GeneSetEnrichmentService* конвертирует гены в entrez представление и ищет подходящие к запросу модули генов, заполняет информацию об экспериментах.

Класс *OverlapService* получает набор генов пользователя и конкретный модуль гена, для которого сервис ищет перекрытия.

3.2.7 Класс *GQDataRepository*

Вызывается при старте сервиса и отвечает для инициализацию форматов генов, чтение *.gmt* файлов и т.д. – т.е. вызывает все эти функции.

3.2.8 Изменения в GeneQueryDE по сравнению с версией GeneQuery

Помимо изменившихся входных данных, также были осуществленны изменения в коде проекта. Они затронули класс *GQRestProperties*, в который было добавленно распознавание файлов с аннотациями (т.е. с названиями модулей) к *.gmt* файлам. Далее добавлен вызов чтения аннотаций в класс *GQDataRepository*. Класс *GQCollection* расширен для чтения названий модулей и сохранения их в структуру модуля, соответствующего данной аннотации. Т.е., соответственно, был изменен *GQModule* класс.

Названия модулей были добавлены в структуру, возвращаемую на front-end из класса *EnrichmentResultItem*. Также функция *findBonferroniSignificant* из файла *FisherEnrichmentAnalysis* была изменена, а именно вычисление параметров теста Фишера, который описан в следующем пункте.

3.3 Метод поиска модулей генов

GeneQueryDE со стороны пользователя принимает название организма и некий набор генов. Затем на основании статистической значимости выдаются подобранные модули. Для этого отдельно оценивается значимость каждого модуля, используя тест Фишера. При этом обозначаются две гипотезы:

— *нулевая* – связи между двумя наборами генов нет, пересечение случайно.
— *альтернативная* – пересечение получено не случайно, наблюдается статистически значимая связь, гены связанны одним процессом.

Таблица 1 – Тест Фишера

	в запросе	вне запроса	всего
в модуле	a	b	a + b
не в модуле	c	d	c + d
всего	a + c	b + d	n

, где *a, b, c, d* - количества генов в соответствующих категориях;

n - размер универсума (= 7000 генов)

$$p = \binom{a+b}{a}\binom{c+d}{c} / \binom{n}{a+c} = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!} \quad (2)$$

Используя формулу 2 производится расчет таблицы 1. Чем меньше полученное *p-value*, тем значимее модуль. Таким образом, число выражает нашу «неуверенность» в связи модуля и запроса.

Далее происходит корректировка на множественные сравнения. Для этого применяется поправка Бонферрони – полученное *p-value* умножается на количество модулей, т.е. на число произведенных сравнений. Полученное значение после поправки, *adj.p-value*, равно вероятности отклонить хотя бы одну нулевую гипотезу из всех проверенных, т.е. вероятности совершить хотя бы одну ошибку первого рода. Если $adj.p\text{-}value \leq 0.01$, то модуль признается статистически значимым и выводится в качестве результата запроса.

3.4 Front-end часть

Front-end репозиторий сервиса имеет достаточно стандартную для django-проекта структуру, поэтому на рисунке 9 показаны только основные элементы, в которых были осуществлены изменения по сравнению с версией GeneQuery.

Классы из пакета *searcher* были расширены для получения и обработки названий модулей генов. В пакеты *js* и *css*, отвечающие за представление данных, также были добавленны изменения – реализовано само представление модулей, выделены жирным шрифтом значимые части названий, отрегулирована ширина итоговой таблицы и т.д. Пару шаблонов было изменено в пакете *templates* – например, название новой версии сервиса.

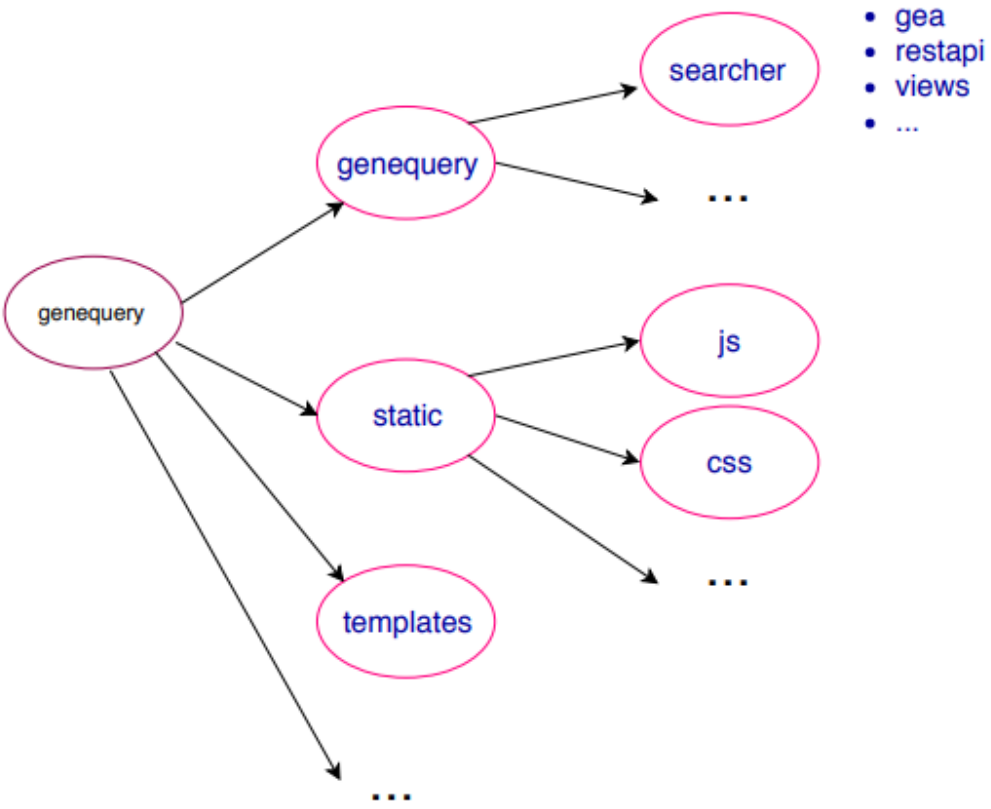


Рисунок 9 – Front-end часть GeneQueryDE.

3.5 Представление результатов пользователю

Пользователь выбирает организм и вводит соответствующий ему набор генов, как показано на рисунке 10. Система выдает ответ в виде статистически значимых модулей (рисунки 11 и 12), в которых найдены совпадающие гены. Список отсортирован в порядке возрастания логарифма *adj.p-value* ≤ 0.01. Как можно видеть, информация, которую получает пользователь по каждому модулю, также включает в себя название эксперимента, название модуля, число перекрытий генов (рисунок 13), ссылку на эксперимент и возможность загрузки таблицы с дифференциальной экспрессией данного модуля.

Название модуля включает в себя два биологических состояния, на основании которых была вычисленна дифференциальная экспрессия, и условия, которые не менялись.

Например, *up in* **‘oxygen.3%.O2’** *compared to* **‘oxygen.21%.O2’** *on ‘tissue.fetal.cortex_Cortex.primary.culture_13.days’ background* – ответ на гипоксию. Здесь гены при **‘oxygen.3%.O2’** реагируют против **‘oxygen.21%.O2’** на основании условий, присутствующих в обоих экспериментах.

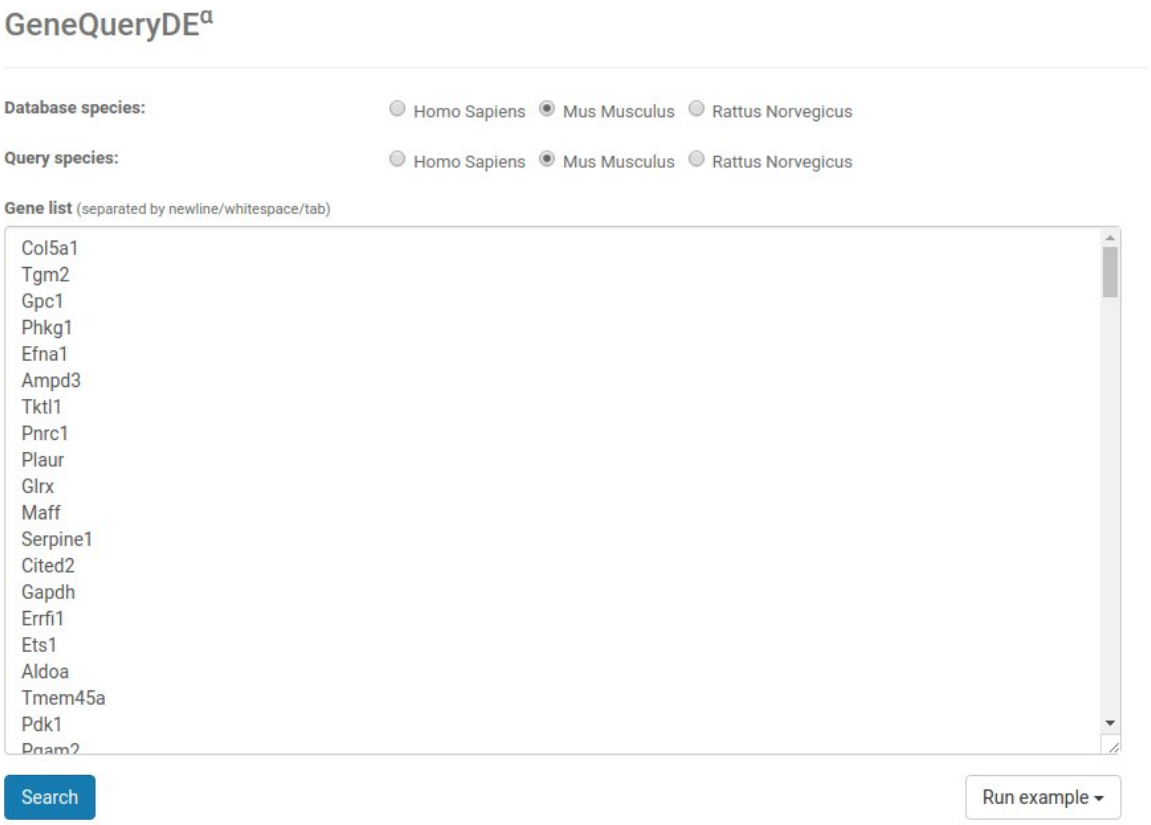


Рисунок 10 – Пример поиска для молекулярного пути ответа на гипоксию.

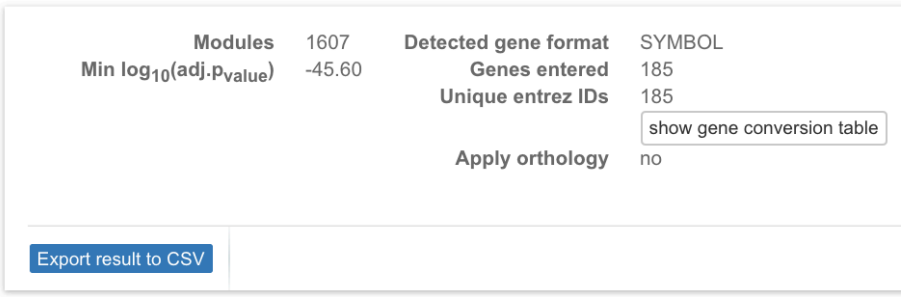
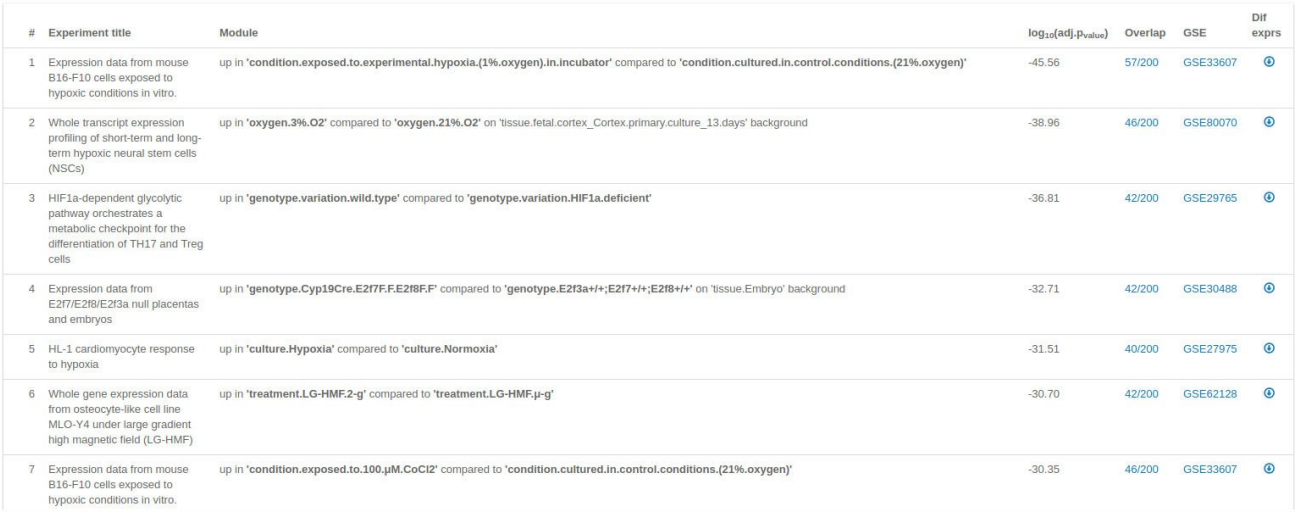


Рисунок 11 – Результат поиска – количество найденных модулей генов для молекулярного пути ответа на гипоксию.

Как можно видень на рисунке 12 первый найденный модуль называется *up in 'condition.exposed.to.experimental.hypoxia(1%.oxygen).in.incubator' compared to 'condition.cultured.in.control.conditions.(21%.oxygen)'*. Учитывая, что гипоксия – это недостаток кислорода, логично наблюдать в найденных модулях генов, модули, в названиях которых фигурирует разница в кислороде. Последующие модули тоже так или иначе включают эту информацию.

3.6 Контейнеризация

В соответствии с документацией docker [18] был создан Dockerfile для построения образа и впоследующем контейнера с сервисом. Используя данную



#	Experiment title	Module	log_{10}(adj.p_{value})	Overlap	GSE	Dif exprs
1	Expression data from mouse B16-F10 cells exposed to hypoxic conditions in vitro.	up in **'condition.exposed.to.experimental.hypoxia.(1%.oxygen).in.incubator'** compared to **'condition.cultured.in.control.conditions.(21%.oxygen)'**	-45.56	57/200	GSE33607	
2	Whole transcript expression profiling of short-term and long-term hypoxic neural stem cells (NSCs)	up in **'oxygen.3%.O2'** compared to **'oxygen.21%.O2'** on 'tissue.fetal.cortex_Cortex.primary.culture_13.days' background	-38.96	46/200	GSE80070	
3	HIF1a-dependent glycolytic pathway orchestrates a metabolic checkpoint for the differentiation of TH17 and Treg cells	up in **'genotype.variation.wild.type'** compared to **'genotype.variation.HIF1a.deficient'**	-36.81	42/200	GSE29765	
4	Expression data from E2f7/E2f8/E2f3a null placentas and embryos	up in **'genotype.Cyp19Cre.E2f7F.F.E2f8F.F'** compared to **'genotype.E2f3a+/+;E2f7+/+;E2f8+/+'** on 'tissue.Embryo' background	-32.71	42/200	GSE30488	
5	HL-1 cardiomyocyte response to hypoxia	up in **'culture.Hypoxia'** compared to **'culture.Normoxia'**	-31.51	40/200	GSE27975	
6	Whole gene expression data from osteocyte-like cell line MLO-Y4 under large gradient high magnetic field (LG-HMF)	up in **'treatment.LG-HMF.2-g'** compared to **'treatment.LG-HMF.μ-g'**	-30.70	42/200	GSE62128	
7	Expression data from mouse B16-F10 cells exposed to hypoxic conditions in vitro.	up in **'condition.exposed.to.100.μM.CoCl2'** compared to **'condition.cultured.in.control.conditions.(21%.oxygen)'**	-30.35	46/200	GSE33607	

Рисунок 12 – Результат поиска – список модулей для молекулярного пути ответа на гипоксию.

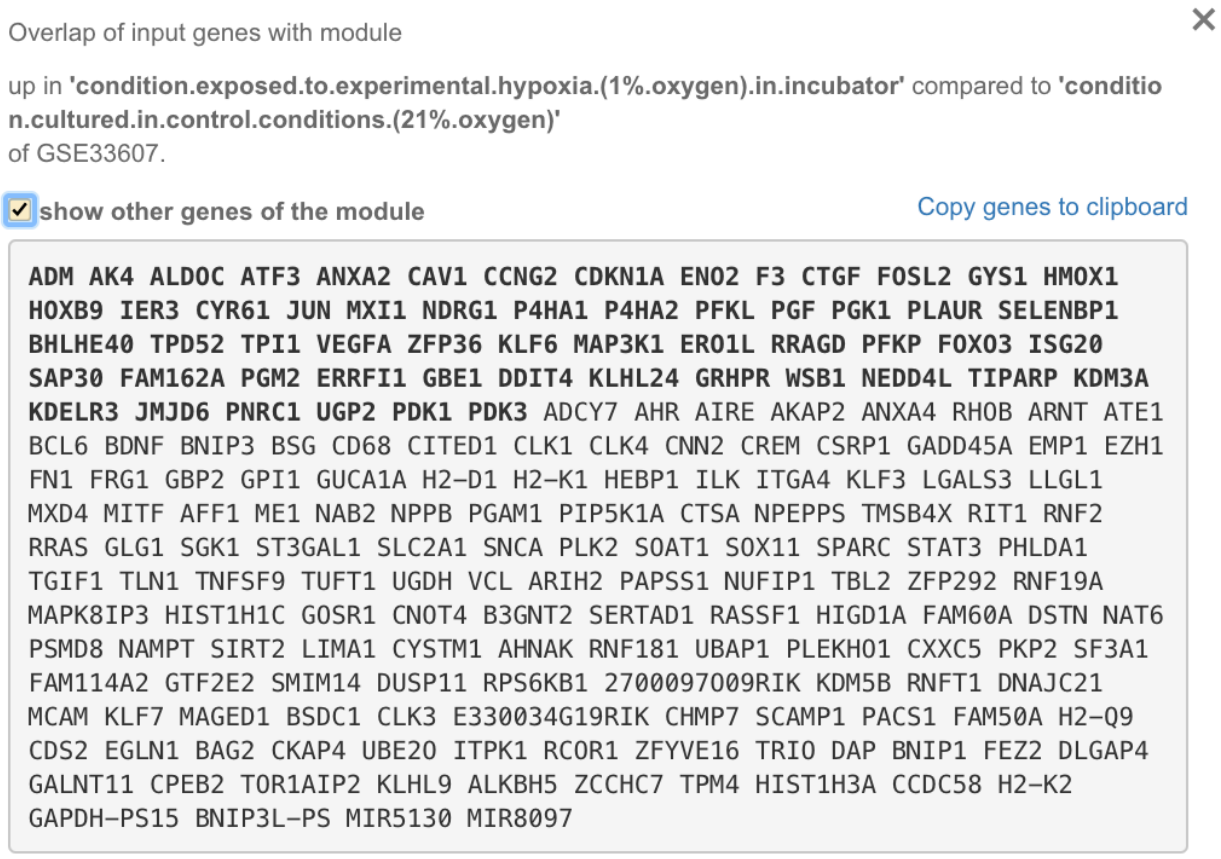


Рисунок 13 – Результат поиска – перекрытие генов первого модуля, найденного для молекулярного пути ответа на гипоксию.

технологию, проект GeneQueryDE, состоящий из front-end и back-end частей был упакован в docker-контейнер с доступом к данным с сервера – к модулям ге-

нов, *.gmt* файлам и аннотациям к ним. Контейнер запущен, т.ч. сервис доступен по адресу *http://genome.ifmo.ru/genequery-de*.

Выводы по главе 3

В данной главе были описаны структура и интерфейс сервиса GeneQueryDE, способ выдачи результата и пример запроса-ответа. Сказано о различиях, внесенных в версию GeneQueryDE по сравнению с GeneQuery. Также обозначена возможность доступа к сервису посредством интернета.

ГЛАВА 4. СРАВНЕНИЕ И АНАЛИЗ РЕЗУЛЬТАТА

4.1 Сравнение GeneQuery с GeneQueryDE по числу данных

На рисунке 14 представленно сравнение сервисов по количеству пар GSE+GPL, для которых в последствии были построены модули генов. Как можно увидеть, помимо совпадающих экспериментов, проанализированных в обоих сервисах, также много уникальных GSE, обработанных только в одном из проектов. Причем в GeneQueryDE обработано больше. Если для вида Rattus Norvegicus количество уникальных GSE примерно равно в обоих проектах, то, например, для Mus Musculus, сервис GeneQueryDE предоставляет в 2.8 раза больше экспериментов, доступных для поиска, чем версия GeneQuery.

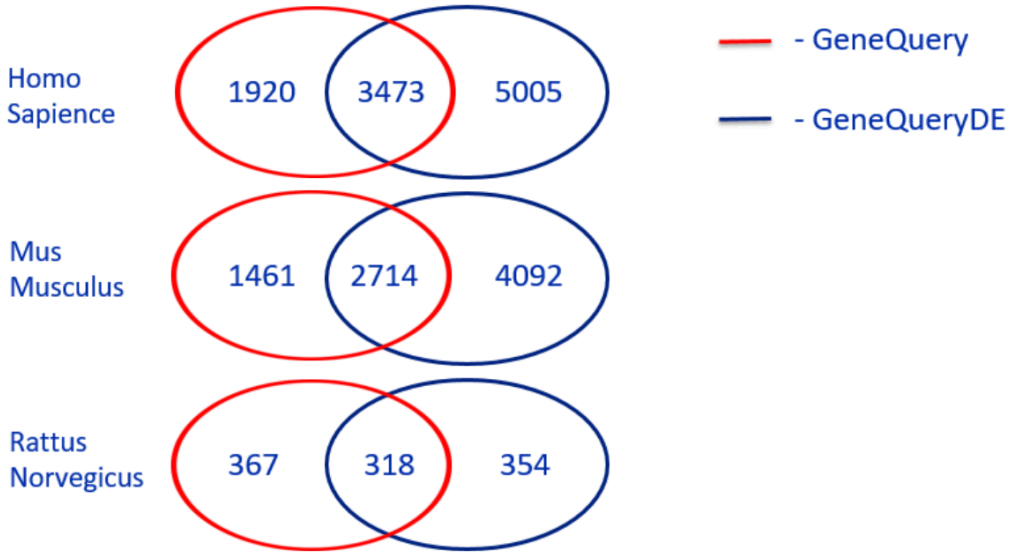


Рисунок 14 – Сравнение по парам GSE+GPL.

В таблице 2 представлены различия в итоговом числе модулей, полученных для двух сервисов. Видно, что сервис GeneQueryDE работает с гораздо большим количеством данных, чем альтернативная версия.

Таблица 2 – Сравнение по модулям генов

	Модули экспрессии генов	
	GeneQuery	GeneQueryDE
Homo Sapience	117497	620216
Mus Musculus	82371	201240
Rattus Norvegicus	13560	22868

4.2 Сравнение сервисов

В данном разделе приведено сравнение существующих проектов с GeneQueryDE. А именно, сервисов GeneQuery, GEOracle, ARCHS4 и GEO Profiles. Как видно из таблицы 3 все сервисы выполняют разные функции, принимая на вход различные аргументы. Сервис GeneQueryDE делает свою уникальную работу – ищет модули дифференциальной экспрессии генов, перекрывающиеся с запросом пользователя, и выдает их, ранжируя в порядке статистической значимости.

Выводы по главе 4

В данной главе были подведены итоги созданного сервиса GeneQueryDE, а именно произведено его сравнение с альтернативной версией по количеству обрабатываемых данных и сравнение с существующими проектами.

Таблица 3 – Сравнение сервисов

	Входные данные	Обрабатываемые виды	Результаты работы
GeneQuery	Набор генов	— Homo Sapience — Mus Musculus — Rattus Norvegicus	— Перекрытия генов — Тепловые карты экспрессии генов
GeneQueryDE	Набор генов	— Homo Sapience — Mus Musculus — Rattus Norvegicus	— Перекрытия генов — Модули дифференциальной экспрессии
GEOracle	Набор экспериментов	Без ограничений	— Числовые значения дифференциальной экспрессии — Тепловые карты зависимости экспериментов от генов
ARCHS4	— Ген — Набор генов в верхней/-нижней регуляции — (доп.параметры: ткани, органы)	— Homo Sapience — Mus Musculus	— Информация о гене — Средняя экспрессия гена
GEO Profiles	— Ген — Эксперимент — Организм — Свободный текст	Без ограничений	— Эксперименты с аннотациями — Информация о генах

ЗАКЛЮЧЕНИЕ

В результате данной работы:

— Построены модули дифференциальной экспрессии генов для видов Homo Sapience, Mus Musculus, Rattus Norvegicus.

— Сервис GeneQuery расширен до версии GeneQueryDE, обеспечивающей интерфейс доступа к модулям генов.

— Число экспериментов, доступных для поиска, увеличено по сравнению с GeneQuery.

— Разработанное решение было упаковано в контейнер.

— Сервис GeneQueryDE был развернут по адресу: *http://genome.ifmo.ru/genequery-de*.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *А.Браун Т.* Геномы. — М.-Ижевск : Институт компьютерных исследований, 2011. — С. 921.

2 *N.Yu. Oparina A.V. Snezhkina A. S.* Differential expression of genes that encode glycolysis enzymes in kidney and lung cancer in humans // Russian Journal of Genetics. — 2013. — Vol. 49. — P. 707–716.

3 *Edgar R Domrachev M L. A.* Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. // Nucleic Acids Res. — 2002. — Vol. 30. — P. 207–210.

4 GeneQuery. — 2016. — URL: `http://genome.ifmo.ru/genequery/searcher/`. [Электронный ресурс].

5 GEOracle. — 2017. — URL: `http://georacle.victorchang.edu.au/`. [Электронный ресурс].

6 *Djordje Djordjevic Y. X. C.* GEOracle: Mining perturbation experiments using free text metadata in Gene Expression Omnibus. — 2017. — URL: `https://doi.org/10.1101/150896`. [Электронный ресурс].

7 ARCHS4: Massive Mining of Publicly Available RNA-seq Data from Human and Mouse. — 2018. — URL: `https://amp.pharm.mssm.edu/archs4/index.html`. [Электронный ресурс].

8 ARCHS4 Help. — 2018. — URL: `https://amp.pharm.mssm.edu/archs4/help.html`. [Электронный ресурс].

9 GEO Profiles. — 2018. — URL: `https://www.ncbi.nlm.nih.gov/geoprofiles/`. [Электронный ресурс].

10 The R Project for Statistical Computing. — 2018. — URL: `https://www.r-project.org/`. [Электронный ресурс].

11 Fisher's exact test. — 2018. — URL: `https://en.wikipedia.org/wiki/Fisher%27s_exact_test`. [Электронный ресурс].

12 Bonferroni correction. — 2018. — URL: `https://en.wikipedia.org/wiki/Bonferroni_correction`. [Электронный ресурс].

13 Контейнеризация. — 2017. — URL: `https://ru.wikipedia.org/wiki/`. [Электронный ресурс].

14 Docker. — 2018. — URL: https://www.docker.com/. [Электронный ресурс].

15 Понимая Docker. — 2015. — URL: https://habr.com/post/253877/. [Электронный ресурс].

16 DockerHub. — 2016. — URL: https://hub.docker.com/. [Электронный ресурс].

17 Bioconductor. — 2018. — URL: https://www.bioconductor.org/. [Электронный ресурс].

18 Dockerfile reference. — 2018. — URL: https://docs.docker.com/engine/reference/builder/. [Электронный ресурс].