ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук НИУ ВШЭ-Нижний Новгород

Образовательная программа «Программная инженерия»

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент департамента  программной инженерии  факультета компьютерных наук  канд. техн. наук | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**Пояснительная записка**

**по курсовой работе**

на тему «Алгоритмы бикластеризации: изучение, анализ и применение»

по направлению подготовки бакалавров 09.03.04 «Программная инженерия»

Исполнитель:   
студент группы 20ПИ-1

Лапшина Юлия Алексеевна

(в составе группы также участвуют

Алексанян Р., Румянцев К., Сигал Н., Титова Н.)

Руководитель:

приглашённый преподаватель

Бычков Илья Сергеевич

Нижний Новгород 2022

**Аннотация.** Данная работа, выполненная в составе групповой курсовой работы об изучении алгоритмов бикластеризации, является прикладным проектом с целью разработать графический веб-фреймворк для обеспечения комфорта, понимания и эффективности работы с данными алгоритмами. В работе поставлено техническое задание, проведён анализ существующих технологий подобного класса и обоснованный подбор инструментов разработки, объяснён принцип построения и работы приложения, рассмотрены используемые теоретические концепции (алгоритмы и метрики качества) и, наконец, протестирован и проанализирован получившийся продукт, BCL Web.

**Ключевые слова**: бикластеризация, фреймворк

Оглавление

[1. Введение 4](#_Toc106144343)

[2. Средства и технологии разработки 6](#_Toc106144344)

[2.1 Постановка технического задания 6](#_Toc106144345)

[2.2 Обзор существующих аналогов 7](#_Toc106144346)

[2.3 Фреймворк 8](#_Toc106144347)

[2.4 Библиотека biclustlib 8](#_Toc106144348)

[3. Описание программного продукта 9](#_Toc106144349)

[3.1 Общая структура 9](#_Toc106144350)

[3.2 Веб-интерфейс 9](#_Toc106144351)

[3.3 Алгоритмы 11](#_Toc106144352)

[3.4 Наборы данных и метрики 13](#_Toc106144353)

[4. Тестирование 14](#_Toc106144354)

[4.1 Методика тестирования 14](#_Toc106144355)

[4.2 Выводы и результаты 14](#_Toc106144356)

[5. Заключение 16](#_Toc106144357)

[Источники 17](#_Toc106144358)

[Приложения 18](#_Toc106144359)

[Приложение 1 18](#_Toc106144360)

# Введение

В настоящее время, когда высокие технологии получают широчайшее распространение, у всех на слуху такие слова, как «машинное обучение» и «анализ данных». Услышать их можно не только от людей, профессионально занятых в области ИТ, но и от бизнесменов, желающих извлечь из этих технологий прибыль, журналистов, освещающих новости о них, и других заинтересованных людей. Методы, используемые в данной сфере, действительно оказываются очень полезны при решении ряда практических задач, прямо влияющих на уровень комфорта и качества жизни людей, что делает исследования в ней как нельзя более актуальными. Спрос на профессионалов в этой сфере сейчас велик, поэтому всеобщий интерес достаточно логичен.

В то же время нельзя не заметить, что подобная работа с данными в чётко формализованном виде бывает тяжела и неудобна, особенно для тех, кто не имеет значительного опыта в сфере компьютерных технологий. Именно этот факт подсказал мне проблему, которую я поставила перед собой в рамках своей курсовой работы. Целью моей работы стало разработать продукт, который сделал бы работу с таким видом интеллектуального анализа данных, как бикластеризация, легче и понятнее для пользователя, обеспечив удобство загрузки данных и параметров и наглядность полученных результатов.

Основными этапами по мере выполнения работы были:

* изучение того, что представляет из себя метод бикластеризации, существующих его задач и алгоритмов;
* изучение имеющихся компьютерных реализаций, анализ их корректности и эффективности;
* составление концепции собственного продукта, выбор и изучение средств для реализации;
* разработка программы с учётом поставленной цели и сделанных ранее выводов;
* тестирование, доработка, продумывание способов улучшения.

# 2. Средства и технологии разработки

## 2.1 Постановка технического задания

На начальном этапе планирования своего продукта я определила список целей, которых хотела добиться при его реализации, он выглядит следующим образом:

* использование бикластеризации – продукт должен представлять собой программную платформу (фреймворк, framework), предоставляющую возможность работать с алгоритмами, решающими задачи бикластеризации;
* наглядность ввода – пользователю легко понять, какие параметры работы алгоритмов он может варьировать и каким именно образом;
* наглядность вывода – пользователю легко интерпретировать результаты работы алгоритмов;
* переносимость – пользователю легко установить продукт, запустить его и удалить, а также загрузить данные или выгрузить результаты;
* эффективность работы – пользователь не тратит лишнего времени, чтобы разобраться или найти нужный ему нюанс;
* удовлетворённость – пользователь не должен получать неприятных впечатлений от работы с продуктом.

Эти цели трансформировались в следующие решения и следующий список задач.

* Для наглядности продукт должен, это почти очевидно, иметь графический интерфейс, поскольку работа с данными в консольном текстовом формате очень утомительна и провоцирует на ошибки.
* Наглядность ввода обеспечивается графическими элементами в виде окошек для ввода текста, раскрывающихся списков, кнопок единичного (radio button) и множественного (checkbox) выбора. Наглядность вывода – построением графиков и таблиц.
* Интерфейс должен выглядеть по возможности минималистично, без отвлекающих элементов, фоновые цвета – спокойные. Цвета элементов интерфейса сигнализируют об их назначении и, насколько возможно, интуитивно понятны; в необходимых местах присутствуют пояснения о назначении.
* Функциональность основана на уже реализованных алгоритмах бикластеризации и метриках. Пользователь имеет возможность выбрать алгоритмы, которые он хочет запустить, отрегулировать их параметры, выбрать набор данных и результаты, которые хочет увидеть.
* Пользователь имеет возможность импортировать собственный файл с данными, а также экспортировать файлы результатов.

## 2.2 Обзор существующих аналогов

Чтобы представлять себе ситуацию с разработкой подобных приложений в мире на данный момент, я совершила небольшой поиск. Воспользовавшись собранной статистикой по ряду существующих фреймворков [1], я оценила запланированный функционал собственного продукта, который получил рабочее название BCL Web (см. Приложение 1) (подробности, указанные мной в таблице, объясняются ниже).

Из анализа данного сравнения можно увидеть, что, хотя по разнообразию обработки результатов (post processing) другие проекты выглядят привлекательнее, по основному функционалу мой проект вполне с ними сравним – в самом деле, моей задачей является больше визуальная доступность, а не глубина анализа, на которой фокусируются более сложные реализации.

## 2.3 Фреймворк

Для разработки интерфейса было принято использовать язык Python, поскольку, в отличие от более низкоуровневых C++ и Java или интерфейсного JavaScript, он многогранен и позволяет с одинаковым качеством реализовать программную логику и графический слой, причём бесшовно, то есть в рамках одного проекта, и в одном стиле. Также на нём уже реализованы необходимые мне инструменты для бикластеризации.

Между самостоятельным графическим интерфейсом и веб-интерфейсом (страницей в браузере) был выбран второй вариант, потому что такое приложение может работать и, благодаря встроенным графическим элементам системы в HTML, выглядеть универсально и гармонично вне зависимости от платформы.

Основной выбор фреймворка для разработки был между Django и Flask. Изучение информации о них дало понять, что Django обеспечивает более цельную интеграцию частей приложения за счёт собственной «жёсткой» структуры, а также предоставляет более полную документацию и поддержку [2], поэтому проект был построен на нём.

## 2.4 Библиотека biclustlib

В качестве вычислительной основы для своего продукта я использовала библиотеку biclustlib, над которой работал мой коллега по курсовой работе Сигал Никита [3]. Её функционал включает ряд алгоритмов бикластеризации, функции для их совместного запуска и сравнения метрик, несколько наборов данных, а также метод анализа обогащения онтологии генов – функцию более специфичную, чем классическое сравнение, применяемую в рамках прикладной задачи об обогащении генов. Всё это и было преобразовано мной в BCL Web.

# 3. Описание программного продукта

## 3.1 Общая структура

Код BCL Web в открытом доступе: [4]

Так как проект разрабатывался как приложение Django, файловая структура его в основном определена и создана самим этим фреймворком для обеспечения комфортной работы.

Элементы пакета, предназначенные для взаимодействия с пользователем, - созданные по стандартной схеме *README.md* и *requirements.txt* (пояснения к установке программы), а также *manage.py* – управляющий скрипт пакета, который запускается непосредственно пользователем.

Среди элементов пакета, игравших значительную роль в моей разработке, можно выделить *biclustering/views.py*, *biclustering/preprocessing.py*, *biclustering/urls.py*, а также *biclustering/static* и *biclustering/templates*. Роль и содержание каждого из них будут раскрыты ниже. Что касается остальных файлов, в большинстве случаев роль их сводится к вспомогательной.

## 3.2 Веб-интерфейс

Для построения графического интерфейса приложение использует веб-сервер, который обновляет страницу согласно запросам пользователя. Страницы строятся по шаблонам (templates), собранным в *biclustering/templates*. Эти файлы представляют собой шаблоны Django, то есть стандартный язык разметки HTML, расширенный тэгами (tags) - инструментами, функционирующими подобно базовым элементам кода, такими как переменные, сравнения, циклы и другие [5]. Это делает их гибкими и функциональными. В визуальном оформлении задействован также *biclustering/static/styles.css*, содержащий визуальные характеристики стиля элементов, объявленных в шаблонах на HTML. Вынесение стилей в отдельный файл позволяет сразу нескольким шаблонам обращаться к этому коду и таким образом создаёт единство дизайна по всему проекту.

Программная обработка шаблонов происходит в видах (views) – функциях, расположенных в *biclustering/views.py*. Стандартным способом они принимают запрос от сервера, загружают шаблон и контекстные переменные, рендерят (отрисовывают) его и возвращают серверу ответ. Так выглядят эти стандартные действия:

|  |
| --- |
| def main\_view(request): |
|  |

|  |
| --- |
| template = loader.get\_template('biclustering/main.html') |
| … |
|  |

|  |
| --- |
| context = { |
|  |

|  |
| --- |
| … |

|  |
| --- |
| } |
|  |

return HttpResponse(template.render(context, request))

*run\_view* отрисовывает главную страницу. Шаблону главной страницы передаются все списки, которые потом предлагаются пользователю, а также осуществляется проверка операционной системы – для действий с рядом алгоритмов (освещено ниже). Главная страница представляет собой списки наборов данных, алгоритмов, возможных графиков и таблиц – все регулируемые параметры запуска алгоритмов бикластеризации, удобно сгруппированные и заполненные значениями по умолчанию. В зависимости от типа необходимых данных параметры представлены текстовым окошком, кнопкой единичного или множественного выбора, выпадающим списком или полем для файла.

*error\_view* служит для отлова возникающих ошибок, выдавая лаконичное и аккуратное сообщение об ошибке вместо системной информации.

*save\_view* не отрисовывает непосредственно стилизованного окна, она системными средствами предлагает пользователю открыть архив с отчётом программы, предварительно его составив.

*table* не имеет собственного шаблона, она служит для отображения csv-таблиц с результатами работы, которые можно представить как самостоятельный HTML-файл силами функции *to\_html()* библиотеки *pandas*.

*run\_view* выполняет самые сложные задачи, то есть разбирает ввод параметров с главной страницы методом *request.GET.get()*, передаёт их нужным алгоритмам, собирает их набор (бенчмарк, benchmark) и запускает его, и наконец, отрисовывает запрошенные пользователем графики и таблицы на новой, пустой странице. По этой причине в функции создаётся огромное количество потенциальных ошибок, поэтому их отлов и разбор просто необходим (и, собственно, присутствует). В следующем разделе детали работы функции будут обрисованы чуть подробнее.

## 3.3 Алгоритмы

Данный фреймворк предоставляет основные алгоритмы решения задачи бикластеризации [6], реализованные в библиотеке *biclustlib* и подробнее рассмотренные в других частях групповой курсовой работы. Это:

* Cheng and Church’s Algorithm;
* Conserved Gene Expression Motifs;
* Large Average Submatrices;
* Binary Inclusion-Maximal Biclustering Algorithm;
* Bit-Pattern Biclustering Algorithm;
* Plaid;
* Spectral.

Каждый из них можно запустить с любыми угодными пользователю параметрами, причём для удобства у них всех проставлены оптимальные значения по умолчанию. За один запуск алгоритмы работают на одном и том же наборе данных, но в зависимости от алгоритма они принимают разный вид: в той же функции *run\_view* из оригинального массива данных в виде чисел с плавающей точкой создаются подобные ему целочисленный и бинарный массив, то есть данные подвергаются дискретизации. Здесь стоит особо отметить передачу данных в алгоритм Spectral: чтобы избежать пограничного для алгоритма случая, когда сумма ряда данных равно нулю, к ряду прибавляется его минимум (в общем случае это совсем незначительно изменяет показания алгоритма) [7].

Особенно важная часть алгоритмической составляющей – наличие алгоритма анализа обогащения онтологии генов, который, как мы видели ранее, являлся одним из важных критериев оценки качества и полезности такого приложения. Его не рекомендуется запускать по умолчанию и, скажем так, при отсутствии необходимости, поскольку он гораздо сложнее и требовательнее остальных в плане вычислительной мощности. Но для тех, у кого есть прямая потребность в этом инструменте, он, несомненно, будет очень полезен.

Ещё одна практическая деталь, которую я не могу не отметить в этом разделе – реализация трёх из этих семи алгоритмов на языке программирования R в оригинальной библиотеке. В процессе разработки выяснилось, что как минимум в текущей версии продукта эти алгоритмы ловят фатальные ошибки при работе на операционной системе Windows 10, хотя адекватно работали на одной из версий MacOs. За неимением технической возможности провести тесты на большем количестве систем и точно вычислить проблему не удалось, но вероятно, что здесь играет роль одно из множества несоответствий систем Windows и Unix-подобных. Как бы то ни было, приходится признать неполноценность адаптации части алгоритмов в данном фреймворке, по крайней мере на текущий момент времени.

## 3.4 Наборы данных и метрики

Как и в случае с алгоритмами, фреймворк предоставляет пользователям возможность воспользоваться одним из имеющихся в библиотеке 19 наборов данных - Tavazoie, Prelic, Jaskowiak (17 таблиц) - а также загрузить свои, приведённые к подобному, удобному для машинной разборки, виду в формате .csv. Как упоминалось выше, данные дискретизируются непосредственно перед подачей в алгоритм, а вот их предварительная загрузка происходит в файле *biclustering/preprocessing.py*.

Что касается метрик оценки качества алгоритмов, пользователь может выбирать из следующих [8]:

* Average Spearman’s Rho;
* Mean Squared Residue;
* Scaling Mean Squared Residue;
* Virtual Error;
* Virtual Error transposed.

Каждая из метрик доступна в виде графика (максимально наглядное сравнение) и таблицы (чуть менее наглядно, но более подробно). Кроме того, после выполнения предлагаются общие отчёты о количестве найденных бикластеров и затраченном на работу времени, а также, разумеется, отчёты об анализе обогащения, если он проводился.

# Тестирование

## 4.1 Методика тестирования

На этапе тестирования не применялось никаких автоматизированных систем тестирования по той причине, что довольно значительную часть времени оценки требовали не какие-либо численные результаты, а внешний вид и дизайн продукта, удобство расположения элементов, понятность действий и т.д. Другим аспектом тестов стало общение пользователя со средой, иными словами, как разбирать и передавать все параметры так, чтобы допустить минимальное количество несоответствий типов. Это была довольно долгая и не очень приятная работа, и я не могу гарантировать, что результат можно назвать красивым, но я постаралась обеспечить максимальную надёжность выполнения. Большая часть всего, что относится к вычислениям собственно библиотеки, была уже протестирована и известна – собственно, именно имевшиеся результаты работы алгоритмов помогали мне оценить правильность моих действий.

## 4.2 Выводы и результаты

Проведённые по составленному плану тесты вовремя помогли исправить недочёты в работе. Усовершенствовалась обработка пользовательских значений параметров, для такого необычного по сравнению с другими алгоритма, как Spectral, были найдены оптимальные способы форматировать данные и ещё некоторые параметры, средствами мультипроцессинга была улучшена производительность. Последняя, конечно, очень сильно зависит от параметров машины, на которой происходит работа; если запуск стандартных алгоритмов даже не на очень сильном компьютере занимает секунд 20 (что я считаю вполне допустимым временем для такой сложной задачи и реальных наборов данных), то анализ обогащения на этой же машине занимал у меня на тестах 45 минут. Ещё раз подчеркну, что это всего лишь слабость системы, потому что результаты при этом выдаются вполне корректные.

В целом, подводя итог прикладной работе, хочется взглянуть на сравнение существующих фреймворков с планируемым, которое было составлено в главе 2, и оценить, все ли запланированные детали присутствуют в конечном продукте. Да, это действительно так; и также можно сказать, что разработанный продукт удовлетворяет требованиям и целям, которые я ставила себе в начале – алгоритмическая база практически полностью реализована и при этом удобна и приятна для использования в лаконичном и дружелюбном интейфейсе. Разумеется, продукт сложно назвать готовым или завершённым, но он абсолютно точно работоспособен, эффективен, и, главное, имеет потенциал. При желании проект можно развивать дальше, причём в разных направлениях: можно углубиться в более качественную бикластеризацию с детальной оценкой параметров, можно, наоборот, расширить область применения, включив алгоритмы кластеризации, или классификации или любой другой области анализа данных. В конце концов, уж чему никогда не стать совершенством, так это графическому интерфейсу: для создания более дружелюбной среды тем, кто мало знаком с теорией этой области, можно добавить больше пояснений и подсказок к уже существующим.

# Заключение

Интеллектуальный анализ данных, и задача бикластеризации как его подвид, многогранны и имеют огромный прикладной потенциал. Изучать и применять их на благо человечества очень ценно, но любая теоретическая работа будет незавершённой, пока не будет воплощена в практике. Познакомиться с теорией алгоритмов кластеризации и в это же время построить для них удобную и понятную оболочку, так упрощающую использование, было для меня увлекательной работой, и я думаю, что результаты можно назвать вполне удовлетворительными, если не совсем хорошими. Фреймворк BCL Web несомненно имеет потенциал развития параллельно с другими практическими воплощениями и исследованиями в этой области, и лучшим результатом этой работы как моего прикладного курсового проекта для меня стало бы, если бы кто-то продолжал пользоваться им по назначению.

# Источники

[1] Comparison between features provided by JBiclustGE and other relevant biclustering frameworks. 2021. <https://jbiclustge.github.io/>

[2] Flask vs Django in 2022: Which Framework to Choose? 2022. <https://hackr.io/blog/flask-vs-django>

[3] nikitasigal/biclustlib <https://github.com/nikitasigal/biclustlib>

[4] yualapshina/bcl\_web <https://github.com/yualapshina/bcl_web>

[5] The Django template language. <https://docs.djangoproject.com/en/4.0/ref/templates/language/>

[6] Victor A. Padilha, Ricardo J. G. B. Campello. A systematic comparative evaluation of biclustering techniques. BMC Bioinformatics. 2017

[7] Kluger Y, Basri R, Chang JT, Gerstein M. Spectral biclustering of microarray data: coclustering genes and conditions. Genome Res. 2003

[8] Beatriz Pontes, Ral Girldez, Jess S. Aguilar-Ruiz. Quality Measures for Gene Expression Biclusters. Plos One. 2015

# Приложения

## Приложение 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature | JBiclustGE | BiMAX | biclust | BiBench | MTBA | BCL Web |
| Programming language | Java | Java | R | Python | Matlab | Python |
| Number Biclustering algorithms | 21 | 6 | 6 | 12 | 12 | 8 |
| Graphical Interface | yes | yes | no | no | no | yes |
| Requires programming expertise | no | no | yes | yes | yes | no |
| Visualization features | yes | yes | yes | yes | yes | yes |
| Synthetic data | yes | no | no | yes | no | no |
| Built-in gene enrichment analysis | yes | no | no | no | no | yes |
| Post-Processing: |  |  |  |  |  |  |
| Filtering | yes | yes | no | yes | no | yes |
| Overlap | yes | yes | yes | yes | yes | no |
| Similarity | yes | no | yes | yes | yes | no |
| Pairwise analysis | yes | yes | no | yes | no | no |
| Coherence | yes | no | yes | no | no | no |
| Coverage | yes | no | no | no | no | no |