МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий

Кафедра алгебраических и информационных систем

«УТВЕРЖДАЮ»
Директор ИМИТ ИГУ
_____ М. В. Фалалеев
«25» мая 2022 г.

Рабочая программа дисциплины (модуля)

Б1.В.07 Очистка данных

Направление подготовки	02.04.02 Фундаментальная информатика и информационные технологии
Направленность (профиль) подготовки	Анализ данных научных исследований и машинное обучение
Квалификация выпускника	магистр
Форма обучения	очная

Иркутск 2022 г.

Согласовано с УМК Института математики и информационных технологий
Протокол № 3 от «04» апреля 2022 г.

Председатель _____
Антоник В.Г.

Рекомендовано кафедрой Алгебраических и информационных систем ИМИТ ИГУ:
Протокол № 9 От «24» марта 2022 г.

Зав. кафедрой _____
Пантелеев В.И.

1. Цели и задачи дисциплины

Цель

Познакомить студентов с современными технологиями получения, подготовки и очистки данных для дальнейшего анализа с помощью современного языка программирования для статистической обработки данных R.

Задачи:

Изучение возможностей языка R, подключение и использование пакетов из различных репозиториев, изучение работы с системой контроля версий, получение и чтение данных из различных источников, подготовка данных для дальнейшего анализа, научить организации, объединению и управлению данными.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП ВО

2.1. Учебная дисциплина (модуль) относится к части программы, формируемой участниками образовательных отношений, и изучается на втором курсе.

2.2. Для изучения данной учебной дисциплины (модуля) необходимы знания, умения и навыки, сформированные: Анализ и визуализация данных.

2.3. Перечень последующих учебных дисциплин, для которых необходимы знания, умения и навыки, формируемые данной учебной дисциплиной: не предусмотрено.

3. ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Процесс освоения дисциплины направлен на формирование компетенций (элементов следующих компетенций) в соответствии с ФГОС ВО по соответствующему направлению подготовки.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с индикаторами достижения компетенций

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-1 Способен осуществлять управление, обработку, визуализацию и анализ данных (включая работу с большими данными), в том числе методами машинного обучения	ИДК ПК1.1 Способен пользоваться методами и инструментами получения, хранения, передачи и обработки данных (в том числе больших)	Знает язык программированию R Умеет работать в интегрированной среде разработки RStudio Владеет навыками программирования на языке программирования R.
	ИДК ПК1.2 Способен разрабатывать системы хранения и обработки данных (в том числе больших)	Знает принципы получения и обработки данных на языке программирования R Умеет получать и обрабатывать данные с помощью языка программирования R Владеет навыками получения и обработки данных из различных источников с помощью языка программирования R
	ИДК ПК1.3 Способен пользоваться методами и	Знает язык программированию R

	инструментальными средствами машинного обучения	Умеет работать в интегрированной среде разработки RStudio Владеет навыками программирования на языке программирования R.

4. СОДЕРЖАНИЕ И СТРУКТУРА ДИСЦИПЛИНЫ

Объем дисциплины составляет 3 зачетных единиц, 108 часов, практическая подготовка 108.
Форма промежуточной аттестации: 3 семестр - зачет.

4.1. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ, СТРУКТУРИРОВАННОЕ ПО ТЕМАМ, С УКАЗАНИЕМ ВИДОВ УЧЕБНЫХ ЗАНЯТИЙ И ОТВЕДЕННОГО НА НИХ КОЛИЧЕСТВА АКАДЕМИЧЕСКИХ ЧАСОВ

№ п/п	Раздел дисциплины/темы	Семестр	Виды учебной работы, включая самостоятельную работу обучающихся и трудоемкость (в часах)				Формы текущего контроля успеваемости
			Контактная работа преподавателя с обучающимися			Самостоятельная работа + контроль	
			Лекции	Семинарские (практические занятия)	Контроль обучения		
1	Знакомство с языком программирования R		2	2	1	6	лаб.
2	Контроль версий и GitHub		2	2	1	6	лаб.
3	Вопросы интеллектуальной обработки данных		2	2	1	8	лаб.
4	Подмножества. Управляющие конструкции. Функции. Область видимости. Дата и время		2	2	1	10	лаб.
5	Циклические функции		2	2	1	10	лаб.
6	Моделирование. Профилирование кода		2	2	1	10	лаб.
7	Получение и чтение данных		2	2	1	10	лаб.
8	Организация, объединение и управление данными		2	2	1	8	лаб.
Итого часов			16	16	8	68	

4.2. ПЛАН ВНЕАУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
3	Знакомство с языком программирования R	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	6	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
3	Контроль версий и GitHub	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	6	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
3	Вопросы интеллектуальной обработки данных	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	8	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
3	Подмножества. Управляющие конструкции. Функции. Область видимости. Дата и время	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	10	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
3	Циклические функции	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	10	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
3	Моделирование. Профилирование кода	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	10	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
3	Получение и чтение данных	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	10	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
3	Организация, объединение и управление данными	Выполнение практической работы	Согласно срокам выполнения заданий в ИОС DOMIC	8	Проверка домашней работы	Материалы курса на платформе ИОС DOMIC
Общая трудоемкость самостоятельной работы по дисциплине (час)				68		

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
Из них объем самостоятельной работы с использованием электронного обучения и дистанционных образовательных технологий (час)				**68**		

4.3. Содержание учебного материала

Знакомство с языком программирования R

Истории R и S, основные типы данных в R, функции для чтения и записи данных, управляющие конструкции и функции, инструменты отладки, имитация данные, профилировщик. Работа в RStudio.

Контроль версий и GitHub

Контроль версий, использование Git и GitHub для управления контролем версий в проектах по обработке и анализу данных.

Вопросы интеллектуальной обработки данных

Виды анализа данных, проектирование эксперимента, большие данные.

Подмножества. Управляющие конструкции. Функции. Область видимости. Дата и время

Подмножества, векторизованные операции, управляющие конструкции, объявление функций, область видимости, дата и время, стандарты кодирования

Циклические функции

Функции lapply, sapply, apply, mapply, tapply, split

Моделирование. Профилирование кода

Функция str, моделирование, профилирование кода.

Получение и чтение данных

Данные, чистые данные, загрузка, чтение: Excel, XML, JSON, использование data.table, чтение из СУБД MariaDB, чтение из веб.

Организация, объединение и управление данными

Подмножества и сортировка, обобщение данных, создание переменных, изменение формата данных, объединение данных.

4.3.1. Перечень семинарских, практических занятий и лабораторных работ

№ п/н	№ раздела и темы	Наименование семинаров, практических и лабораторных работ	Трудоемкость (час.)	Оценочные средства	

			Всего часов	Из них практическая подготовка		Формируемые компетенции (индикаторы)*
1	**2**	**3**	**4**	**5**	**6**	**7**
1	1	Знакомство с языком программирования R	2	2	Проверка лабораторной работы	ПК-1
2	2	Контроль версий и GitHub	2	2	Проверка лабораторной работы	ПК-1
3	3	Вопросы интеллектуальной обработки данных	2	2	Проверка лабораторной работы	ПК-1
4	4	Подмножества. Управляющие конструкции. Функции. Область видимости. Дата и время	2	2	Проверка лабораторной работы	ПК-1
5	5	Циклические функции	2	2	Проверка лабораторной работы	ПК-1
6	6	Моделирование. Профилирование кода	2	2	Проверка лабораторной работы	ПК-1
7	7	Получение и чтение данных	2	2	Проверка лабораторной работы	ПК-1
8	8	Организация, объединение и управление данными	2	2	Проверка лабораторной работы	ПК-1
			16	16		

4.3.2. Перечень тем (вопросов), выносимых на самостоятельное изучение студентами в рамках самостоятельной работы (СР)

Тема	Задание	Формируемые компетенции
Знакомство с языком программирования R	Решить задачи для самостоятельного выполнения	ПК-1
Контроль версий и GitHub	Решить задачи для самостоятельного выполнения	ПК-1
Вопросы интеллектуальной обработки данных	Решить задачи для самостоятельного выполнения	ПК-1
Подмножества. Управляющие конструкции. Функции. Область видимости. Дата и время	Решить задачи для самостоятельного выполнения	ПК-1
Циклические функции	Решить задачи для самостоятельного выполнения	ПК-1
Моделирование. Профилирование кода	Решить задачи для самостоятельного выполнения	ПК-1
Получение и чтение данных	Решить задачи для самостоятельного выполнения	ПК-1
Организация, объединение и управление данными	Решить задачи для самостоятельного выполнения	ПК-1

4.4. Методические указания по организации самостоятельной работы студентов

Во время изучения дисциплины студент посещает лекции, практические занятия, выполняет лабораторные задания, готовится к тестам, зачетам и экзаменам. Для каждого вида деятельности необходимо правильно организовать самостоятельную работу.

Лекции. В высшем учебном заведении лекция является важной формой учебного процесса. На лекции студенты получают глубокие и разносторонние знания. Лекция способствует развитию творческих способностей, формирует идейную убежденность, позволяет устанавливать связь учебного материала с производством, новейшими научными достижениями. Лекция требует три вида деятельности: подготовку к лекции, работу на лекции и работу после лекции.

После прослушивания лекции студент должен проработать и осмыслить полученный материал. На каждый пример, приведенный на лекции, желательно, (если это возможно) привести свой. Материал, изложенный в лекции, можно просмотреть в других источниках.

В процессе лекционного занятия студент должен выделять важные моменты, выводы, анализировать основные положения. Недостаточно только «слушать» лекцию. Возможности памяти человека не универсальны. Как бы внимательно студент ни слушал лекцию, большая часть информации вскоре после восприятия будет забыта. Чтобы восстановить лекционный материал, его нужно повторить, а для этого лекцию необходимо конспектировать. Конспект лекций должен быть в отдельной тетради, в которой не должно быть ничего, кроме

лекции. Не надо стремиться подробно слово в слово записывать всю лекцию. Конспектируйте только самое важное в рассматриваемой теме: ключевые слова и их значения, примеры использования конструкций, что старается выделить лектор, на чем акцентирует внимание студентов.

Тетрадь для конспекта лекций также требует особого внимания. Ее нужно сделать удобной, практичной и полезной, ведь именно она является основным информативным источником при подготовке к различным отчетным занятиям, зачетам, экзаменам. Конечно, оформление лекционной тетради – это дело вкуса. Но целесообразно отделить поля, где студент мог бы изложить свои мысли, вопросы, появившиеся в ходе лекции. Полезно одну из страниц оставлять свободной. Она потребуется потом, при самостоятельной подготовке. Сюда можно будет занести дополнительную информацию по данной теме, полученную из других источников: рисунки, схемы, примеры кода и т.д.

Лабораторное занятие. Лабораторные занятия по решению задач существенно дополняют лекции. В процессе анализа и решения задач студенты расширяют и углубляют знания, полученные из лекционного курса и учебников, приобретают умение применять общие закономерности к конкретным случаям.

Необходимо, чтобы студенты готовили теоретический материал, т.к. именно невыполнение этого требования приводит к неудаче при решении задач.

Несмотря на различие в видах задач, их решение можно проводить по следующему общему плану (некоторые пункты плана могут выпадать в некоторых конкретных случаях): а) прочесть внимательно условие задачи; б) посмотреть, все ли термины в условиях задачи известны и понятны (если что-то неясно, следует обратиться к учебнику, просмотреть решения предыдущих задач, посоветоваться с преподавателем); в) произвести анализ задачи, (нужно чётко понимать, в чем будет заключаться решение задачи); г) решить задачу; д) протестировать полученные решение на данных из примеров к задаче, а также на дополнительных данных.

Если задача не решена или «не решается», то необходимо еще раз вернуться к пунктам а) и б). Сколько раз нужно возвращаться к этим пунктам? Практика показывает, что не более десяти раз. Если и после этого задача «не решается», то можно попытаться найти решение этой или похожей задачи в различных источниках.

Домашнее задание. При выполнении домашнего задания необходимо просмотреть текст лекции, разобраться с новыми определениями, посмотреть задания, которые были выполнены на лабораторной работе и применить полученные знания для выполнения домашней работы.

Тест. В первую очередь постарайтесь узнать чего ждать от теста, какие примерно там будут задания. Если вам доступны образцы теста (как, например, при сдаче ЕГЭ), необходимо этим воспользоваться и ежедневно тренироваться.

Не оставляйте все на самый последний момент. Если будете постоянно готовиться к тесту, вы наверняка улучшите свои знания. Для этого составьте план на каждый день, чтобы правильно распределять свое время.

Делайте небольшие перерывы во время учебы. В промежутках можно дать себе небольшую физическую нагрузку. Мозг лучше всего работает, когда умственный труд сменяется физическим. Прогуляйтесь, побегайте, поиграйте в баскетбол, попинайте мяч – помимо стимуляции умственной деятельности, это снимет стресс.

Отдых и контроль над волнением — одни из главных составляющих успеха при подготовке к тесту. Часто ошибки совершаются только из-за стресса, который мешает сконцентрироваться и собраться. Чтобы быть отдохнувшим и расслабленным, соблюдайте составленный режим и старайтесь высыпаться.

Экзамен. На экзамене оцениваются: 1) понимание и степень усвоения теории; 2) методическая подготовка; 3) знание фактического материала; 4) знакомство с обязательной литературой; 5) умение приложить теорию к практике, решать практические задачи и т. д.; 6) логика,

структура и стиль ответа, умение защищать выдвигаемые положения. Но значение экзаменов не ограничивается проверкой знаний. Являясь естественным завершением работы студента, они способствуют обобщению и закреплению знаний и умений, приведению их в строгую систему, а также устранению возникших в процессе занятий пробелов.

Студенты готовятся к экзаменам по-разному. Одни из них прорабатывают лишь некоторые вопросы, выбранные наугад, другие стремятся запомнить весь материал подряд, не вникая глубоко в его суть. Работа при этом концентрируется на одном стремлении – сдать экзамен. Недостатки такой системы очевидны. Очевидно также, что подготовка не должна ограничиваться чтением лекционных записей. Первоначальные необработанные конспекты студента содержат факты, определения, выводы, сделанные преподавателем, но в них, как правило, слабо просматривается связующая идея курса, так как студент, записывая каждую лекцию в отдельности, редко способен сразу и достаточно точно уловить общую направляющую мысль. Поэтому конспект требует дополнительной обработки на основе использования учебников и рекомендованной литературы.

Существенные недостатки имеет и такой способ подготовки к экзаменам, как беглый просмотр всего материала. Он эффективен только на некоторых этапах планирования и закрепляющего повторения. Более надежный и целесообразный путь – это тщательная систематизация материала при вдумчивом повторении, установлении внутрипредметных связей, увязке различных тем и разделов, закреплении путем решения задач.

Перед экзаменом назначается консультация. Цель ее – дать ответы на вопросы, возникшие в ходе самостоятельной подготовки. Хотелось бы обратить особое внимание на важность предэкзаменационных консультаций. Здесь студент имеет полную возможность получить ответ на все неясные ему вопросы. А для этого он должен проработать до консультации весь курс. Кроме того, преподаватель будет отвечать на вопросы других студентов, что будет для вас повторением и закреплением знаний. И еще очень важное обстоятельство: лектор на консультации, как правило, обращает внимание на те разделы, по которым на предыдущих экзаменах ответы были неудовлетворительными, а также фиксирует внимание на наиболее трудных разделах курса. Некоторые студенты не приходят на консультации либо потому, что считают, что у них нет вопросов к лектору, либо полагают, что у них и так мало времени и лучше самому почитать материал по конспекту или в учебнике. Это глубокое заблуждение. Никакая другая работа не сможет принести столь значительного эффекта накануне экзамена, как консультация преподавателя.

Подготовку к экзамену следует начинать с первого дня изучения дисциплины. Как правило, на лекциях подчеркиваются наиболее важные и трудные вопросы или разделы курса, требующие внимательного изучения и обдумывания. Нужно эти вопросы выделить и обязательно постараться разобраться в них, не дожидаясь экзамена, проработать их, готовясь к лабораторным занятиям, попробовать самостоятельно решить несколько типовых задач. И если, несмотря на это, часть материала осталась неусвоенной, ни в коем случае нельзя успокаиваться, надеясь на то, что это не попадется на экзамене. Факты говорят об обратном: если те или другие вопросы курса не вошли в экзаменационный билет, преподаватель может их задать (и часто задает) в виде дополнительных вопросов. Точно такое же отношение должно быть выработано к вопросам и задачам, перечисленным в экзаменационной программе, выдаваемой студентам еще до экзамена. Обычно эти же вопросы и аналогичные задачи содержатся в экзаменационных билетах. Не следует оставлять без внимания ни одного раздела курса; если не удалось в чем-то разобраться самому, нужно обратиться к товарищам; если и это не помогло выяснить какой-либо вопрос до конца, нужно обязательно задать этот вопрос преподавателю на предэкзаменационной консультации. Чрезвычайно важно приучить себя к умению самостоятельно мыслить, учиться думать, понимать суть дела. Очень полезно после проработки каждого раздела восстановить в памяти

содержание изученного материала, кратко записав это на листе бумаги. Если этого не сделать, то большая часть материала останется не понятой, а лишь формально заученной, и при первом же вопросе экзаменатора студент убедится в том, насколько поверхностно он усвоил материал.

4.5. Примерная тематика курсовых работ (проектов)

Не предусмотрено.

5. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

а) перечень литературы

основная литература:

1 Гришин, В. А. Основы программирования на языке R : учебно-методическое пособие / В. А. Гришин. — Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2021. — 67 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/191498.

2. дополнительная литература:

1. Альтман, Е. А. Система контроля версий GIT : учебно-методическое пособие / Е. А. Альтман, А. В. Александров, Т. В. Васеева. — Омск : ОмГУПС, 2021. — 26 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/190155.

2. Гришин, В. А. Методы обработки данных и моделирование на языке R : учебно-методическое пособие / В. А. Гришин, М. С. Тихов. — Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2019. — 54 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/144653.

3. Роберт, И. R в действии. Анализ и визуализация данных в программе R : руководство / И. Роберт, Кабаков ; перевод с английского Полины А. Волковой. — Москва : ДМК Пресс, 2014. — 588 с. — ISBN 978-5-97060-077-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/58703.

б) периодические издания

в) список авторских методических разработок:

Материалы курса, опубликованные в ИОС «DOMIC».

г) базы данных, информационно-справочные и поисковые системы

https://www.r-project.org/other-docs.html

https://cran.r-project.org/web/views/

https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

https://r-pkgs.org/

6. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

6.1. Учебно-лабораторное оборудование:

Для проведения лекционных занятий необходима аудитория с презентационным оборудованием, для проведения лабораторных занятий необходима аудитория на 15-20 рабочих мест (в зависимости от численности учебной группы), оборудованная доской, презентационной техникой, компьютерами.

6.2. Программное обеспечение:

1. Программная среда вычислений R 4.2.0 (распространяется бесплатно, GNU GPL 2);
2. Среда разработки программного обеспечения RStudio Open Source Edition 2022.02.3 (распространяется бесплатно, GNU Affero General Public License v3).
3. Распределённая система управления версиями Git 2.37.0 (распространяется бесплатно, GNU GPL 2).
4. Браузер Google Chrome 100.0 (распространяется бесплатно, proprietary freeware, based on open source components)

6.3. Технические и электронные средства:

ИОС EDUCA, DOMIC, презентационное оборудование, персональный компьютер с возможностью демонстрации презентаций в формате pdf.

7. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

При реализации данного курса используются следующие образовательные технологии: технологии традиционного обучения, технологии проблемного обучения, технологии контекстного обучения, интерактивные технологии, технологии дистанционного обучения.

8. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Разноуровневые задания к лабораторному практикуму

8.1. Оценочные материалы для входного контроля

Не предусмотрены.

8.2. Оценочные материалы текущего контроля

Разноуровневые задания к лабораторному практикуму, доклад.

Примеры оценочных средств текущего контроля

1. Лабораторная работа по теме «Управляющие конструкции. Функции»

Отслеживание загрязнения воздуха

Для выполнения задания нужно написать три функции, которые предназначены для взаимодействия с набором данных определенного формата. Набор данных содержится в представленном zip-архиве.

Данные

Zip-архив включает 332 файла с разделителями-запятыми (формат CSV), в которых содержатся данные мониторинга загрязнения воздуха мелкими твердыми частицами (PM = particulate matter) в 332 пунктах наблюдений. Каждый файл содержит данные с одного монитора, идентификационный номер каждого монитора указан в имени файла. Например, данные для монитора 200 находятся в файле «200.csv». Каждый файл содержит четыре переменные:

- `Date`: дата наблюдения в формате ГГГГ-ММ-ДД (год-месяц-день);
- `sulfate`: уровень сульфата в воздухе на эту дату (измеряется в микрограммах на кубический метр);
- `nitrate`: уровень нитратов в воздухе на эту дату (измеряется в микрограммах на кубический метр);
- `ID`: идентификатор монитора.

Для выполнения задания нужно распаковать архив и создать каталог data. После того, как вы распаковали zip-файл, не вносите никаких изменений в файлы из каталога data. В каждом файле вы заметите, что есть дни, когда данные о сульфатах или нитратах отсутствуют (обозначены как `NA`). Это довольно типичная ситуация при работе с мониторингом загрязнения воздуха.

Задание 1

Напишите функцию под названием `getMean`, которая вычисляет среднее значение загрязнителя (сульфата или нитрата) по указанному списку мониторов. Функция `getMean`

принимает три аргумента: `folder`, `type` и `id`. Функция `getMean` должна для заданного вектора идентификаторов мониторов id прочитать данные из каталога `folder` и вернуть среднее значение загрязняющего вещества типа `type` по этим мониторам, игнорируя любые пропущенные значения, закодированные как `NA`. Прототип функции выглядит следующим образом

```
getMean <- function(folder, type, id=1:332){
        # folder - символьный вектор длины 1, который указывает расположение
CSV-файлов
        # type - символьный вектор длины 1, который указывает тип загрязнителя
воздуха "sulfate" или "nitrate"
        # id - целочисленный вектор, указывающий идентификаторы мониторов,
которые нужно обработать

        # Функция должна вычислить среднее значение по указанному загрязнителю
по всем мониторам указанным в векторе id, игнорируя значения NA.
        # Не используйте округление для результата
}
```

Ниже представлены результаты работы данной функции. Сравните эти результаты с результатами работы вашей функции, они должны совпадать.

```
getMean("data", "sulfate", 1:10)
## [1] 4.064128

getMean("data", "nitrate", 70:72)
## [1] 1.706047

getMean("data", "nitrate", 23)
## [1] 1.280833
```

Задание 2

Напишите функцию, которая читает каталог с файлами данных и сообщает количество полных наблюдаемых случаев (для которых указаны значения обоих видов загрязнений) в каждом файле данных. Функция должна возвращать дата фрейм, в котором первый столбец – это имя файла (по сути идентификатор монитора), а второй столбец – количество полных наблюдений. Прототип функции выглядит следующим образом

```
getCompleteObservation <- function(folder, id = 1:332) {
        # folder - символьный вектор длины 1, который указывает расположение
CSV-файлов
        # id - целочисленный вектор, указывающий идентификаторы мониротов,
которые нужно обработать

        # Функция возвращает дата фрейм вида
        #   id count
        #    1   117
        #    2   1041
        #   ...
        # где id - идентификатор монитора, count - количество полных наблюдаемых
случаев для этого монитора.
}
```

Примеры вывода этой функции представлены ниже. Сравните эти результаты с результатами работы вашей функции, они должны совпадать.

```
getCompleteObservation("data", 1)

##   id count
## 1  1   117
```

```
getCompleteObservation("data", c(2, 4, 8, 10, 12))
```

```
##   id count
## 1  2  1041
## 2  4   474
## 3  8   192
## 4 10   148
## 5 12    96
```

```
getCompleteObservation("data", 30:25)
```

```
##   id count
## 1 30   932
## 2 29   711
## 3 28   475
## 4 27   338
## 5 26   586
## 6 25   463
```

```
getCompleteObservation("data", 3)
```

```
##   id count
## 1  3   243
```

Задание 3

Напишите функцию, которая берет каталог файлов с данными и пороговое значение количества полных случаев (случаев, когда указаны оба показателя) и вычисляет корреляцию между сульфатом и нитратом для мест мониторинга, в которых количество полностью наблюдаемых случаев (по всем переменным) превышает пороговое значение. Функция должна возвращать вектор корреляций для мониторов, соответствующих пороговому значению. Если ни один из мониторов не соответствует пороговому значению, функция должна возвращать числовой вектор длины 0. Прототип функции выглядит следующим образом

```
getCorrelation < function(folder, limen = 0) {
        # folder - символьный вектор длины 1, который указывает расположение
CSV-файлов
        # limen - целочисленный вектор длины 1, указывающий пороговое значение
количества полных случаев, при котором нужно вычислять корреляцию.

        # Функция возвращает числовой вектор корреляций
}
```

Для реализации этой функции вам нужно будет использовать встроенную функцию `cor`, которая вычисляет корреляцию между двумя векторами. Прочтите страницу справки для этой функции с помощью `?cor`.

Ниже представлены результаты работы данной функции. Сравните эти результаты с результатами работы вашей функции, они должны примерно соответствовать этому выводу. Обратите внимание, что из-за того, как R округляет и представляет числа с плавающей точкой, результаты работы вашей функции могут немного отличаться от приведенных примеров.

```
res <- getCorrelation("data", 150)
head(res)
```

```
## [1] -0.01895754 -0.14051254 -0.04389737 -0.06815956 -0.12350667 -0.07588814

summary(res)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.21057 -0.04999  0.09463  0.12525  0.26844  0.76313

res <- getCorrelation("data", 400)
head(res)

## [1] -0.01895754 -0.04389737 -0.06815956 -0.07588814  0.76312884 -0.15782860

summary(res)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.17623 -0.03109  0.10021  0.13969  0.26849  0.76313

res <- getCorrelation("data", 5000)
summary(res)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##

length(res)

## [1] 0

res <- getCorrelation("data")
summary(res)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.00000 -0.05282  0.10718  0.13684  0.27831  1.00000

length(res)

## [1] 323
```

2. Лабораторная работа по теме «Моделирование. Профилирование кода»

Рейтинг больниц

Загрузите файл, содержащий данные для выполнения задания. Разархивируйте файл в каталог, который будет вашим рабочим каталогом. Данные для задания взяты с веб-сайта Министерства здравоохранения и социальных служб США. Цель веб-сайта – предоставить данные и информацию о качестве обслуживания в более чем 4000 больницах США, сертифицированных программой Medicare. Этот набор данных охватывает все основные больницы США, он используется для различных целей, в том числе для определения того, должны ли больницы быть оштрафованы за то, что они не предоставляют пациентам высококачественный уход.
Данные

Zip-архив для этого задания содержит два файла:

results_of_care.csv: содержит информацию о 30-дневной смертности и частоте повторных госпитализаций от сердечных приступов (heart attacks), сердечной недостаточности (heart failure) и пневмонии (pneumonia) для более чем 4000 больниц. Описание структуры файла
hospital.csv – содержит информацию о каждой больнице. Описание структуры файла

Описание переменных в каждом из файлов находится в описании структры файла, см. ссылки выше. В частности, номера переменных для каждой таблицы указывают индексы

столбцов в каждой таблице (например, название больницы (Hospital Name) – это столбец 2 в файле results_of_care.csv).

Задание 1

Гистограмма 30-дневной смертности от сердечного приступа

Считайте данные из results_of_care с помощью функции read.csv и просмотрите первые несколько строк.

```
> res <- read.csv("results_of_care.csv", colClasses = "character")
> head(outcome)
```

В этом наборе данных много столбцов. Вы можете узнать количетво столбцов, набрав ncol(res) (также можно получить количество строк с помощью функции nrow). Кроме того, вы можете получить имена каждого столбца с помощью names(res) (имена также есть в описании файлов).

Чтобы построить гистограмму 30-дневных показателей смертности от сердечного приступа (столбец 11 в наборе данных), запустите следующий код

```
> res[, 11] <- as.numeric(res[, 11])
> ## Вы можете получить предупреждение о введении неопределенного значения, игнорируйте его
> hist(res[, 11])
```

Поскольку мы изначально считываем данные в виде символов (указав colClasses = "character", нам нужно заставить столбец быть числовым. Вы можете получить предупреждение о вводе NA, но это нормально.

Добавьте на гистограмму заголовок и соответствующие подписи осей.

Задание 2

Поиск лучшей больницы в штате

Напишите функцию getBestHospital, которая принимает два аргумента: двухсимвольное сокращенное соответствующее названию штата и имя показателя. Функция считывает файл results_of_care.csv и возвращает вектор символов с названием больницы, которая имеет лучшую (то есть самую низкую) 30-дневную смертность для указанного показателя в этом штате. Название больницы – это имя, указанное в переменной Hospital.Name. Показателем может быть "heart attack" (сердечный приступ), "heart failure" (сердечная недостаточность) или "pneumonia" (пневмония). Больницы, у которых нет данных по конкретному показателю, должны быть исключены из набора больниц при определении рейтинга.

Обработка одинаковых результатов. Если у вас получились одинаковые результаты для заданного показателя, то названия больниц следует отсортировать в алфавитном порядке и выбрать первую больницу в этом наборе (т. е. если больницы с названиями «b», «c» и «f» имеют равный лучший результат, то нужно вернуть больницу «b»).

Шаблон для функции

```
getBestHospital <- function(state, criteria) {
## Прочитать данные по показателю
## Убедитесь, что штат и показатель корректны
## Выберите названия больницы в штате с наименьшей 30-дневной смертностью.
## При необходимости отсортируйте результаты по наименованию больницы
```

}

Функция должна проверять правильность аргументов. Если передается недопустимое значение штата, функция должна выдать ошибку с помощью функции stop с сообщением «недопустимый штат». Если в функцию передается несуществующий показатель, функция должна выдать ошибку с помощью функции stop с сообщение «недопустимый показатель».

Примеры вызовов

```
> getBestHospital("TX", "heart attack")
[1] "CYPRESS FAIRBANKS MEDICAL CENTER"
> getBestHospital("TX", "heart failure")
[1] "FORT DUNCAN MEDICAL CENTER"
> getBestHospital("MD", "heart attack")
[1] "JOHNS HOPKINS HOSPITAL, THE"
> getBestHospital("MD", "pneumonia")
[1] "GREATER BALTIMORE MEDICAL CENTER"
> getBestHospital("BB", "heart attack")
Error in getBestHospital("QWE", "heart attack") : недопустимый штат
> getBestHospital("NY", "hert attack")
Error in getBestHospital("NY", "het atack") : недопустимый показатель
```

Задание 3
Ранжирование больниц по показателям в штате

Напишите функцию под названием getHospitalRating, которая принимает три аргумента: двухсимвольное сокращенное название штата (state), показатель (criteria) и рейтинг больницы в этом штате по этому критерию (n). Функция считывает файл results_of_care.csv и возвращает вектор символов с названием больницы, рейтинг которой указан в аргументе n. Например, вызов getHospitalRating("MD", "heart failure", 5) вернет вектор символов, содержащий название больницы с 5-м наименьшим 30-дневным уровнем смертности от сердечной недостаточности ("heart failure"). Аргумент n может принимать значения «best», «worst» или целое число, указывающее рейтинг (чем меньшие число, тем выше рейтинг). Если число, заданное параметром n, больше, чем количество больниц в этом штате, функция должна вернуть NA. Больницы, у которых нет данных о конкретном показателе, должны быть исключены из набора больниц при определении рейтинга.

Обработка одинаковых результатов. Может случиться, что несколько больниц имеют одинаковый 30-дневный уровень смертности по тому или иному показателю смерти. В таких случаях нужно поступать так же как и в функции getBestHospital, используя наименование больницы.

Шаблон для функции

```
getHospitalRating <- function(state, criteria, n = "best") {
## Прочитать данные по показателю
## Убедится, что штат и показатель корректны
## Вернуть название больницы в заданном штате с заданным рейтингом 30-дневной
смертности
}
```

Функция должна проверять правильность переданных аргументов. Если передается недопустимое значение штата, функция должна выдать ошибку с помощью функции stop с сообщением «недопустимый штат». Если в функцию передается несуществующий показатель, функция должна выдать ошибку с помощью функции stop с сообщение «недопустимый показатель».

Примеры вызовов

```
> getHospitalRating("TX", "heart failure", 4)
[1] "DETAR HOSPITAL NAVARRO"
> getHospitalRating("MD", "heart attack", "worst")
[1] "HARFORD MEMORIAL HOSPITAL"
> getHospitalRating("MN", "heart attack", 5000)
[1] NA
```

Задание 4
Рейтинг больниц во всех штатах

Напишите функцию с именем getRaiting, которая принимает два аргумента: название показателя (criteria) и рейтинг больницы (n). Функция считывает файл results_of_care.csv и возвращает дата фрейм с двумя столбцами, содержащий больницу в каждом штате, рейтинг которой указан в n. Например, вызов функции getRaiting("heart attack", "best") вернет дата фрейм, содержащий названия больниц, которые являются лучшими в своих штатах по 30-дневным показателям смертности от сердечных приступов ("heart attack"). Функция должна возвращать значение для каждого штата (некоторые могут быть NA). Первый столбец в дата фрейме называется hospital и содержит название больницы, а второй столбец называется state и содержит двухсимвольное сокращение для названия штата. Больницы, у которых нет данных о показателе, должны быть исключены из набора больниц при определении рейтинга.

Обработка одинаковых результатов. Функция getRaiting должна обрабатывать одинаковые рейтинги так же, как функция getHospitalRating.

Шаблон для функции

```
getRaiting <- function(criteria, n = "best") {
## Прочитать данные по показателю
## Убедится, что штат и показатель корректны
## Для каждого штата найти больницу с указанным рейтингом
## Вернуть дата фрейм с названиями больниц и названием штата
}
```

Внимание! Для повышения эффективности работы программы функция НЕ должна вызывать функцию getHospitalRating из задания 3.

Функция должна проверять правильность своих аргументов. Если в функцию передается несуществующий показатель, функция должна выдать ошибку с помощью функции stop с сообщение «недопустимый показатель». Переменная n может принимать значения «best», «worst» или целое число, указывающее рейтинг (чем меньшие число, тем выше рейтинг). Если число, заданное параметром n, больше, чем количество больниц в этом штате, функция должна вернуть NA.

Примеры вызовов

```
> head(getRaiting("pneumonia", 15), 10)
                                hospital state
1                                          AK
2                 JACK HUGHSTON MEMORIAL HOSPITAL    AL
3                   DE QUEEN MEDICAL CENTER, INC    AR
4  UNIVERSITY OF ARIZONA MEDICAL CTR-UNIVERSITY, THE    AZ
5                        ST MARY MEDICAL CENTER    CA
6                      KEEFE MEMORIAL HOSPITAL    CO
7                   NORWALK HOSPITAL ASSOCIATION    CT
8                                          DC
9                                          DE
10                  WESTCHESTER GENERAL HOSPITAL    FL

> tail(getRaiting("heart failure", "worst"), 5)
                    hospital state
1       FAIRBANKS MEMORIAL HOSPITAL    AK
2    DEKALB REGIONAL MEDICAL CENTER    AL
3     NEA BAPTIST MEMORIAL HOSPITAL    AR
4 MT GRAHAM REGIONAL MEDICAL CENTER    AZ
5   CLOVIS COMMUNITY MEDICAL CENTER    CA
```

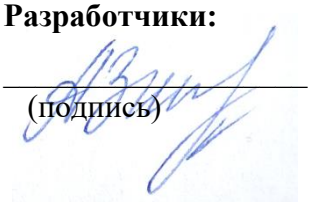
8.3. Оценочные материалы промежуточного контроля

Список вопросов для промежуточной аттестации:

Наука о данных
Данные и их характеристики
Процесс анализа и обработки данных. Этапы
Язык программирования R
Пакеты в R
Контроль версий
Репозиторий Git, терминология
Виды анализа данных
Этапы проектирования эксперимента
Большие данные и их характеристики
Язык программирования R: особенности, недостатки, дизайн
Объекты в R
Векторы и списки, имена для списков
Матрицы
Факторы
Отсутствующие значение
Дата фреймы
Чтение и запись данных. Расчет требований к потребляемой памяти
Подмножества
Векторизованные операции
Управляющие конструкции
Функции
Область видимости
Работа с датой и временем
Функция lapply
Функция sapply
Функция apply

Функция mapply
Функция tapply
Функция split
Методы отладки в R
Функция str
Генерация случайных чисел: функции плотности распределения, распределения, генерации случайных чисел, квантиля.
Оптимизация кода. Использование sistem.time()
Профилировщик
Неструктурированные и структурированные данные
Компоненты чистых данных
Чтение локальных файлов
Чтение Excel-файлов
Чтение XML, язык XPath
Чтение JSON
Функция data.table
Чтение из СУБД
Чтение с web-страниц
Получение подмножества данных
Сортировка
Обобщение данных
Создание переменных
Изменение формата данных
Объединение данных

Разработчики:



_____ _____доцент____ _____ Зинченко А.С._____
(подпись) (занимаемая должность) (Ф.И.О.)

Программа составлена в соответствии с требованиями ФГОС ВО по направлению подготовки 02.04.02 «Фундаментальная информатика и информационные технологии» (уровень магистратуры), утвержденный приказом Министерства образования и науки Российской Федерации от «23» августа 2017 г. № 811, зарегистрированный в Минюсте России «13» сентября 2017 г. № 48168 с изменениями и дополнениями от 26 ноября 2020 г., 8 февраля 2021 г.

Программа рассмотрена на заседании кафедры Алгебраических и информационных систем ИМИТ ИГУ «24» марта 2022 г.

Протокол № 9 Зав. кафедрой_____ Пантелеев В.И.