

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Н. Г. ГАЛИНОВСКИЙ

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ R

Практическое пособие

для магистрантов специальности 1–80 01 01
«Биология»

Гомель
ГГУ им. Ф. Скорины
2021

УДК 004.451.9R(076)
ББК 32.973.2я73
Г157

Рецензенты:

кандидат биологических наук А. А. Саварин,
кандидат биологических наук Н. И. Тимохина

Рекомендовано к изданию научно-методическим советом
учреждения образования «Гомельский государственный
университет имени Франциска Скорины»

Галиновский, Н. Г.

Г157 Введение в программирование на языке R : практическое пособие / Н. Г. Галиновский ; Гомельский гос. ун-т им. Ф. Скорины. – Гомель : ГГУ им. Ф. Скорины, 2021. – 48 с.
ISBN 978-985-577-814-2

Пособие ставит своей целью оптимизировать учебно-познавательную деятельность магистрантов по усвоению материала курса «Введение в программирование на языке R». Пособие может быть использовано как при проведении лабораторных занятий, так и для самостоятельной подготовки при работе над курсовыми и дипломными работами и проектами.

Адресовано магистрантам биологического факультета, биологам-исследователям.

УДК 004.451.9R(076)
ББК 32.973.2я73

ISBN 978-985-577-814-2

© Галиновский Н. Г., 2021
© Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины», 2021

ОГЛАВЛЕНИЕ

Введение.....	4
Тема 1. Создание векторов.....	5
Тема 2. Операции с объектами в R.....	19

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ

ВВЕДЕНИЕ

Современные медико-биологические исследования не могут в должной мере проводиться без применения основ математической статистики. Математическая обработка требуется, прежде всего, для обоснованного доказательства выявленного явления, обнаруженных закономерностей в результате исследований. Математические методы также необходимы для исчерпывающего извлечения информации о выборках, объектах, характеристики их разнообразия и его структуры, о влияниях разных экологических (и не только) факторов на живые организмы, развивающиеся в различных условиях.

Актуальность изучения данной учебной дисциплины связана с необходимостью научной корректности и экономической обусловленности выводов и прогнозов в биологии, сельском хозяйстве и медицине при помощи средств вычислительной техники.

Язык программирования R построен по принципу свободного программного обеспечения. Его использование абсолютно бесплатно и доступно всем желающим.

Цель пособия – ознакомление магистрантов-биологов с основными возможностями и синтаксисом скриптового языка программирования R, а также с методами решения основных прикладных задач статистического анализа данных.

Пособие построено на выполнении конкретных заданий в каждой практической работе. Задания содержат определённый набор данных, взятых как из литературы, так и непосредственно из исследований, проводимых автором пособия. При изучении каждой из тем предварительно подробнейшим образом, по шагам алгоритма, рассматривается проведение того или иного анализа с использованием как командной строки, так и графического интерфейса. Пошаговый алгоритм, как показывает практика, наиболее удобен для усвоения элементов математического анализа и облегчает знакомство с новыми программными продуктами у студентов и магистрантов, не обучающихся по IT-специальностям.

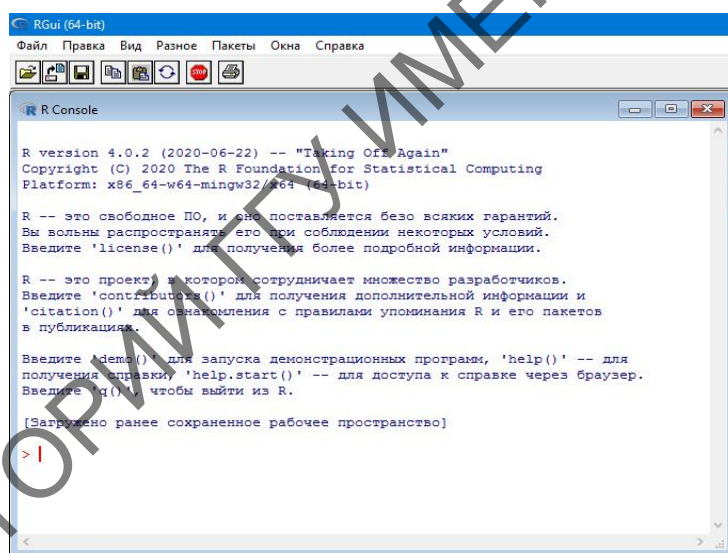
ТЕМА 1. СОЗДАНИЕ ВЕКТОРОВ

- 1 Знакомство со средой языка программирования R.
- 2 Создание векторов в языке программирования R.

1 Знакомство со средой языка программирования R

1.1 Знакомство с интерфейсом среды программирования R

Язык программирования R имеет несколько инструментов для обработки данных. Первый из них – собственно сама среда программирования, которая отображается после загрузки инсталлятора, установки её на компьютер и запуска. Для запуска среды программирования языка R необходимо на рабочем столе кликнуть дважды на значок программы в результате чего будет отображено рабочее окно программы в виде консоли с командной строкой, которая начинается знаком приглашения для ввода команд «>» и мигающим курсором (рисунок 1).



```
RGui (64-bit)
Файл  Правка  Вид  Разное  Пакеты  Окна  Справка

R Console

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R -- это свободное ПО, и оно поставляется безо всяких гарантий.
Вы волены распространять его при соблюдении некоторых условий.
Введите 'license()' для получения более подробной информации.

R -- это проект, в котором сотрудничает множество разработчиков.
Введите 'contributors()' для получения дополнительной информации и
'scitation()' для ознакомления с правилами упоминания R и его пакетов
в публикациях.

Введите 'demo()' для запуска демонстрационных программ, 'help()' -- для
получения справки, 'help.start()' -- для доступа к справке через браузер.
Введите 'q()' чтобы выйти из R.

[Загружено ранее сохраненное рабочее пространство]

>|
```

Рисунок 1 – Рабочее окно среды программирования R (консоль)

Язык меню после установки будет соответствовать языку операционной системы (в нашем случае меню будет русскоязычным).

В языке программирования R существуют инструменты как входящие в базовый пакет, так и разработанные сторонними специалистами и организациями для облегчения работы с командной строкой тем, кто ранее с ней не имел дела. Первый инструмент, который мы рассмотрим, входит в базовый набор пакетов среды программирования R и представляет

собой так называемый GUI (*graphic user interface*) – графический интерфейс пользователя, т. е. вид обычной программы Windows с кнопками и минимальным участием (или совсем без него) командной строки. Этот пакет называется «RCommander» и для того, чтобы его включить, можно воспользоваться любым из приведенных ниже способов.

1-й способ (с использованием меню среды R)

В строке меню выбрать опцию **Пакеты**, а затем в выпадающем списке – опцию **Включить пакет...** (рисунок 2).

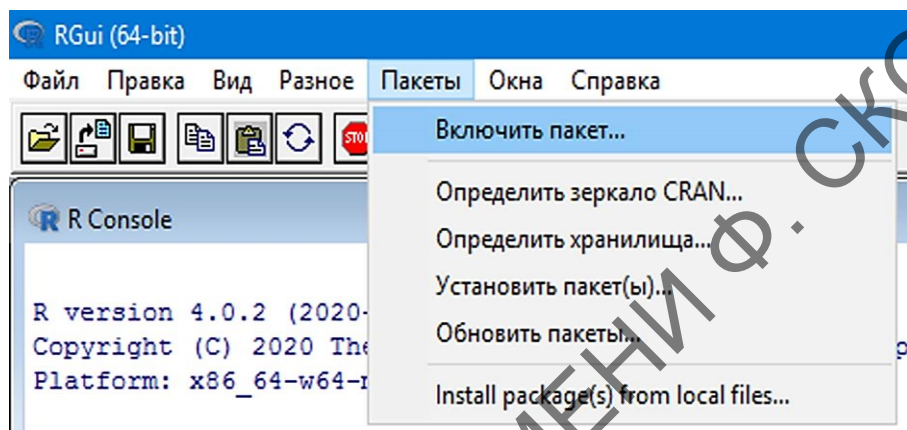


Рисунок 2 – Включение пакета через меню среды R

Затем в появившемся окне **Select one** (Выберите один) перемещаем указателем мыши ползунок, находящийся справа этого окна, пока не найдем пакет под названием **Rcmdr**. После чего устанавливаем на него курсор и нажимаем кнопку **ОК**. В окне среды программирования R будет отражено рабочее окно пакета RCommander (рисунок 3).

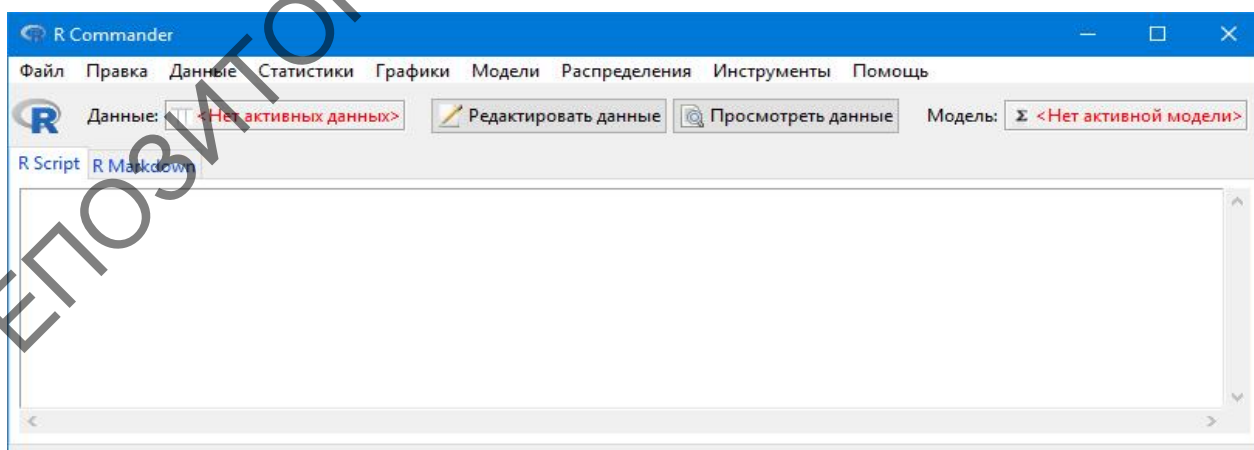


Рисунок 3 – Рабочее окно пакета RCommander

Как можем заметить, оно также имеет русскоязычное меню с набором уже готовых команд, которые можно использовать для обработки имеющихся данных, не прибегая к командной строке.

2-й способ (с использованием командной строки среды R)

Для того чтобы включить уже имеющийся (или скачанный из репозитория) пакет из командной строки, необходимо после символа приглашения набрать команду `library(название пакета)`. В нашем случае:

```
> library(Rcmdr)
```

и нажать клавишу **Enter**. После чего загрузится уже знакомое нам рабочее окно пакета RCommander (рисунок 3).

В то же время работа с чисто графическим интерфейсом не смотря на то, что она достаточно удобна для начинающих пользователей, все же ограничена в функционале и не позволяет ознакомиться со всеми возможностями языка программирования R.

Для облегчения работы с командной строкой и отслеживания в реальном времени загруженных переменных, таблиц, полученных в процессе обработки данных, графиков и диаграмм, а также других объектов служат стандартом для тех, кто использует R в своей повседневной работе программный пакет RStudio. Его абсолютно бесплатно можно скачать и установить на свой компьютер. Программный пакет работает автономно от самой среды языка программирования R и его не нужно предварительно загружать для работы с RStudio.

После загрузки и инсталляции программного пакета необходимо его запустить любым из удобных Вам способов. Рабочее окно программы отображено на рисунке 4.

Окно программы представляет 4 отдельных окна и строку меню над ними. Каждое из окон несет свою определенную функцию:

1 – окно загруженных данных в виде таблиц, наглядно их демонстрирующих;

2 – окно загруженных данных и переменных, имеет 4 закладки:

– **Environment** (Окружение, среда) – отражает все загруженные или созданные в процессе работы переменные и объекты;

– **History** (История) – отображает историю ранее введенных команд;

– **Connections** (Соединения) – отображает активные интернет-соединения со сторонними данными или другими объектами;

– **Tutorial** (Руководство) – отображает ссылку на пакет с руководством пользователя для освоения RStudio (на английском языке);

3 – окно с командной строкой для ввода команд кода;

- 4 – окно визуальной и справочной информации, имеет 5 закладок:
- **Files** (Файлы) – отображает файлы, размещенные в домашней папке среды R;
 - **Plots** (Диаграммы и графики) – отображает графики и диаграммы, полученные при работе с кодом в текущем времени;
 - **Packages** (Пакеты) – отражает список загруженных и включённых пакетов;
 - **Help** (Помощь) – набор файлов помощи по различным вопросам, сгруппированных по разделам;
 - **Viewer** (Просмотрщик) – просмотрщик файлов, которые не являются производными R.

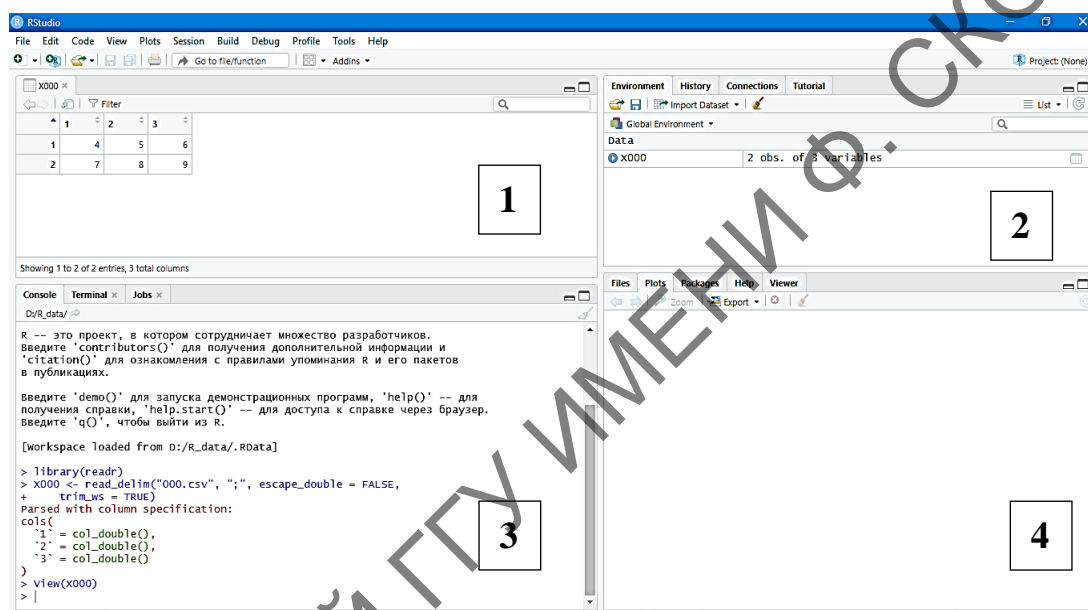


Рисунок 4 – Рабочее окно программного пакета RStudio

Очень полезной особенностью RStudio является то, что при вводе какой-либо команды в командной строке при правильном наборе символов появляется контекстная подсказка, которая позволяет при выборе нужного сочетания команды и аргумента и последующем нажатии клавиши **Enter** сразу отобразить необходимую функцию в строке кода, сокращая время набора команды (рисунок 5).

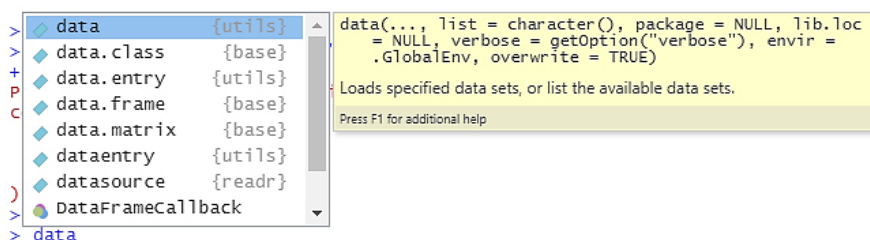


Рисунок 5 – Контекстная подсказка команд в программном пакете RStudio

Для того чтобы ознакомиться на начальном этапе с каждой из перечисленных выше программ и пакетов, попробуем провести в каждой из них расчет выражений $4+1*5$ и $(2+3)*5$ как на калькуляторе.

Шаг 1. Загружаем среду языка программирования R.

Шаг 2. В командной строке набираем первое наше выражение и нажимаем клавишу **Enter**. Результат будет выведен на экран на следующей строке.

```
> 4+1*5  
[1] 9
```

Шаг 3. В следующей строке набираем выражение $(2+3)*5$ и нажимаем клавишу **Enter**. Результат, как и в предыдущем примере, будет выведен на экран на следующей за выражением в строке.

```
> (2+3)*5  
[1] 25
```

Шаг 4. По аналогии сделайте то же самое в RCommander и RStudio.

В дальнейшем, при рассмотрении нами примеров с командной строкой для удобства будет использоваться программный пакет RStudio как более гибкий и наглядный инструмент в отличие от обычной среды R.

1.2 Загрузка данных

1.2.1 Загрузка данных при помощи командной строки

Существует несколько способов загрузки и сохранения данных. Однако в учебных целях мы познакомим только с несколькими из них, наиболее используемыми. Если необходимо ознакомиться с большим количеством способов загрузки и сохранения данных, мы рекомендуем обратиться к руководствам, приведённым в списке литературы в конце пособия.

Допустим, у нас имеются данные по количеству беспозвоночных в 10 почвенных ловушках на 2 участках учета (таблица 1). Их мы подготовим заранее, используя либо Excel из пакета Microsoft Office, либо Calc из пакета Open Office, либо любую программу по работе с электронными таблицами на Ваше усмотрение.

Таблица 1 – Количество беспозвоночных в почвенных ловушках

	1	2	3	4	5	6	7	8	9	10
Уч. 1	12	5	8	9	10	11	2	6	7	3
Уч. 2	25	18	10	18	19	4	25	29	31	21

Главное при создании файла учесть, что данные должны быть размещены в два столбца и первая строчка столбцов – это заголовок. В нашем случае: **Stac 1** и **Stac 2**. В данном пособии для удобства и единообразия в учебных целях будет использоваться Microsoft Excel.

Шаг 1. Создаем таблицу в Excel из двух столбцов с указанными выше заголовками и данными для каждого участка, как в таблице 1, и сохраняем её в рабочую папку R под именем «besp» в формате .csv (MS-DOS), соглашаясь на то, что в файле сохранятся только данные на первом листе книги Excel (*это всегда следует помнить*).

Шаг 2. В командной строке набираем команду `read.table()` (Прочитать таблицу) с необходимыми атрибутами:

```
> read.table("besp.csv", sep=";", head=TRUE)
```

В этой команде сначала указывается путь к файлу. Но так как наш файл находится в рабочей папке R, то никаких подробностей не нужно, в противном случае нужно будет указать полный путь к файлу. Далее, аргумент `sep=";"` (*separate* – разделитель) показывает, что является разделителем столбцов (в нашем случае – точка с запятой, но может быть и запятая, и двоеточие, и знак табуляции). Следующий аргумент – `head` (заголовок), который показывает, есть ли заголовок у таблицы (`true`) или отсутствует (`false`). Кстати, в данном случае (и в большинстве других) можно не писать полное название `true` или `false` – достаточно указать лишь первую букву. **Но имейте в виду, что в R регистр букв имеет значение.** Обратите внимание, что аргументы отделены друг от друга запятой и пробелом.

В итоге, на экране будет отражена наша таблица:

	Stac.1	Stac.2
1	12	25
2	5	18
3	8	10
4	9	18
5	10	19
6	11	4
7	2	25
8	6	29
9	7	31
10	3	21

Команда `read.table()` является достаточно универсальной. Для чтения файлов в формате .csv есть своя собственная команда `read.csv()` с похожими аргументами и ее можно использовать для нашего примера, но для начала уберем из памяти компьютера уже ранее загруженную нашу

таблицу. Для этого используется команда `rm()` (*remove* – перемещение, удаление).

Шаг 3. Удаляем ранее загруженную таблицу из памяти компьютера.

```
> rm(besp)
```

Шаг 4. Загружаем нашу таблицу, используя команду `read.csv()`.

```
> read.csv("besp.csv", sep = ";", header = TRUE)
```

В итоге на экране появится наша таблица в том же виде, как и на шаге 2.

1.2.2 Загрузка данных при помощи GUI

Для примера будем также использовать тот же созданный нами ранее файл `besp.csv`.

1.2.2.1 Загрузка данных в RStudio

Шаг 1. Интегрируем сохраненную таблицу в R. Для этого в RStudio переходим в меню: **File** → **Import Dataset** → **From Text (readr)...** (рисунок 6).

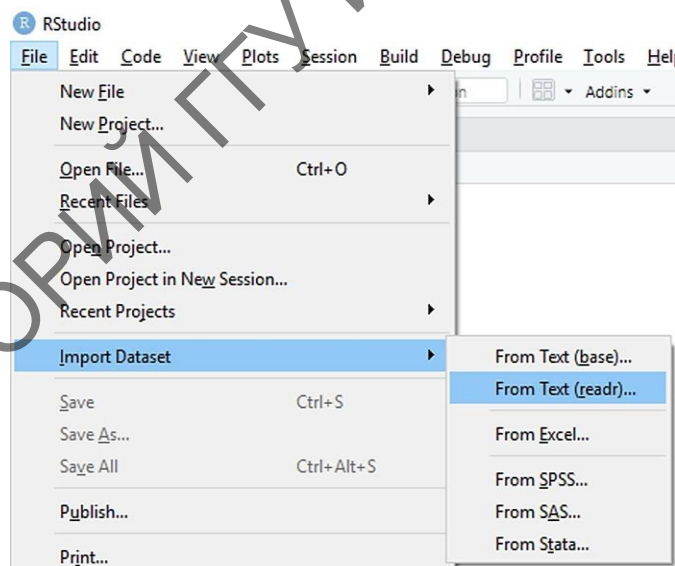


Рисунок 6 – Загрузка данных, используя меню RStudio

Шаг 2. В открывшемся диалоговом окне указываем название (и путь, если он находится не в домашней папке R) файла в строке **File/URL**, нажав кнопку **Browse...** Далее, так как у нас первая строка таб-

лицы – это названия столбцов, то внизу необходимо отметить бокс **First Row as Names**, а в качестве разделителя столбцов (**Delimiter**) в выпадающем списке указать точку с запятой – **Semicolon**. В результате в центральной области окна мы увидим нашу таблицу в том виде, в котором она загрузится в программу. Следует отметить, что в правой нижней части окна дублируется программный код, который можно ввести в консоли для получения точно такого же эффекта (рисунок 7).

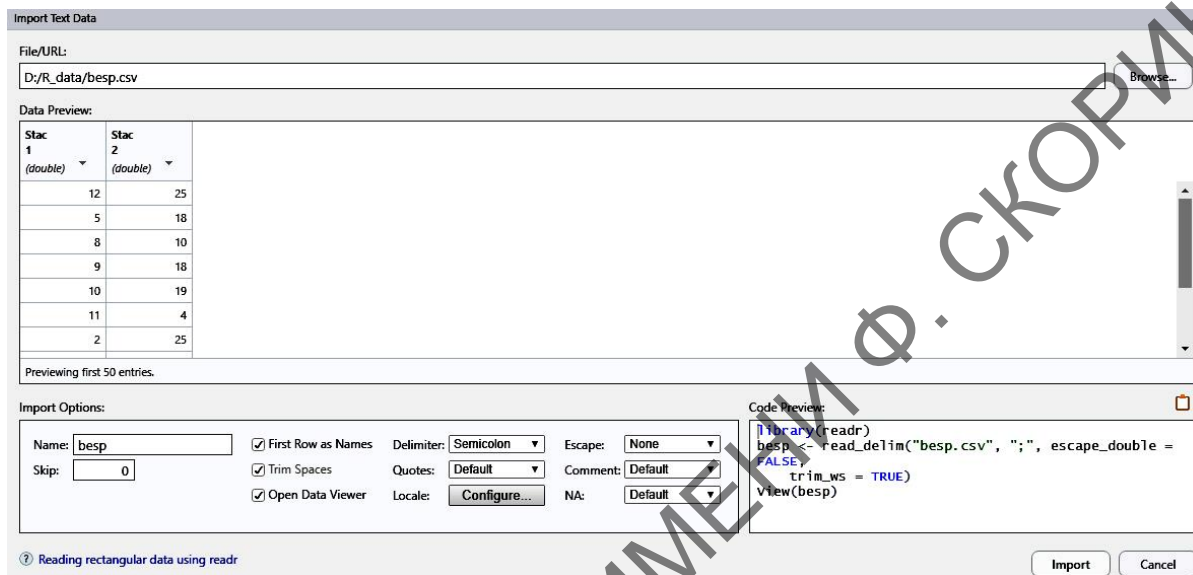


Рисунок 7 – Настройка окна импорта данных RStudio

Шаг 3. После нажатия клавиши **Import** (Преобразование) наша таблица отобразится в левом верхнем окне рабочей среды RStudio (рисунок 8).

	Stac 1	Stac 2
1	12	25
2	5	18
3	8	10
4	9	18
5	10	19
6	11	4
7	2	25

Рисунок 8 – Таблица данных в окне рабочей среды RStudio

1.2.2.2 Загрузка данных в RCommander

Шаг 1. Загружаем среду языка программирования R и включаем пакет RCommander (см. раздел 1.1.).

Шаг 2. Интегрируем сохраненную таблицу в R при помощи RCommander. Для этого в RCommander переходим в меню: **Данные** → **Импорт данных** → **из текстового файла, буфера обмена или URL...** (рисунок 9).

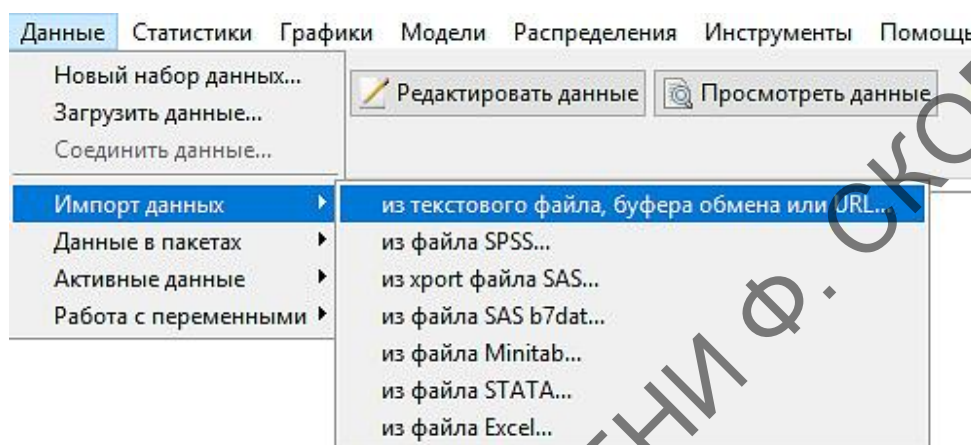


Рисунок 9 – Загрузка данных RCommander

Шаг 3. В появившемся диалоговом окне выставляем опции, как показано на рисунке 10.

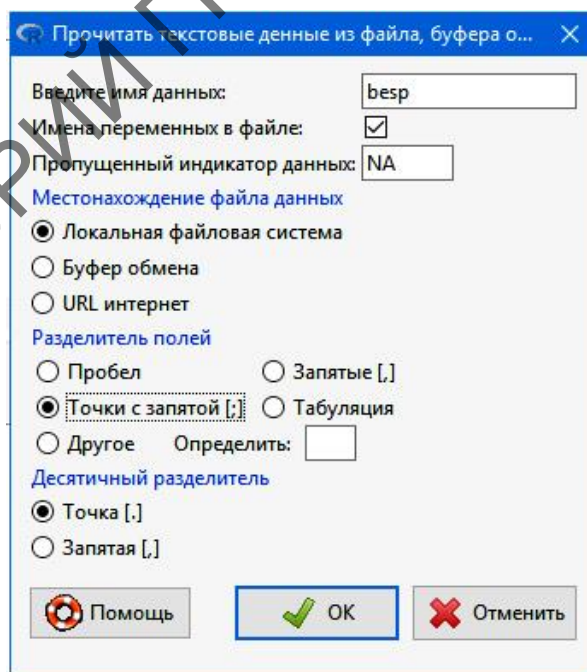
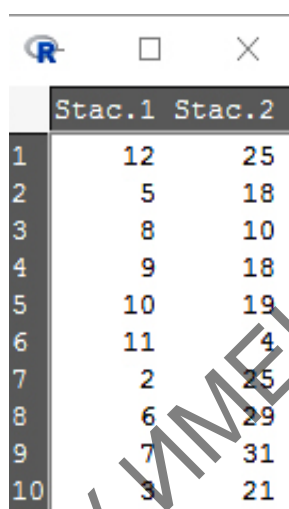


Рисунок 10 – Опции интеграции текстового файла с разделителем в RCommander

Шаг 4. Нажимаем кнопку **ОК**, оказываемся в окне выбора файла, указываем путь к файлу и нажимаем **ОК**. Наш файл загружен в память компьютера.

Шаг 5. Для проверки правильности загрузки данных необходимо первоначально нажать на кнопку в области **Данные**, которая находится под строкой меню справа, и в списке объектов с данными выбрать наш объект под именем «besp», сделать по нему двойной клик мышью и нажать кнопку **ОК**. После чего нажимаем кнопку **Просмотреть данные** и убеждаемся, что появившаяся таблица (рисунок 11) полностью совпадает с той, которую вы вводили при помощи электронных таблиц.



	Stac.1	Stac.2
1	12	25
2	5	18
3	8	10
4	9	18
5	10	19
6	11	4
7	2	25
8	6	29
9	7	31
10	3	21

Рисунок 11 – Таблица с данными, отраженная в RCommander

2 Создание векторов в языке программирования R

Все операции с данными в языке R основаны в первую очередь на векторах. Вектор представляет собой объект, содержащий определенную последовательность значений одного типа.

Все элементы вектора могут принадлежать какому-то одному из типов данных, принятых в языке R (подробнее о типах данных – в теме 2). В то же время вектор является объектом языка R и, как уже было сказано выше, может содержать компоненты только одного типа, а также пропущенные значения, обозначаемые как «NA», которые могут быть в векторе абсолютно любого типа. В языке R есть и другие объекты: матрицы, списки, таблицы, функции, факторы (подробнее о них рассказано в теме 2). Тем не менее, в основе всех этих типов объектов лежит базовый объект – вектор.

2.1 Создание векторов при помощи командной строки

В результате проведенного исследования на участке смешанного леса были получены данные по весу 10 бурозубок в мг (w), весу потребленного ими корма в мг (f) (таблица 2).

Таблица 2 – Вес бурозубок смешанного леса и вес их корма

w , мг	7.1	7.7	3.6	8.3	8.8	10.4	8.9	9.0	8.9	14.0
f , мг	2.3	2.3	4.1	5.3	4.4	5.9	5.7	6.2	6.5	9.3

Необходимо обратить внимание, что при работе с десятичными дробями дробное от целого отличается точкой, а не запятой.

Для создания в будущем рабочей таблицы (датафрейма), содержащего в себе данные по весу микромаммалий и количеству съеденного, необходимо создать 2 вектора. Создадим вектор, содержащий данные по весу самих зверьков.

Шаг 1. Создадим вектор w , используя команду $c()$:

```
> w <- c(7.1, 7.7, 3.6, 8.3, 8.8, 10.4, 8.9, 9.0, 8.9, 14.0)
```

Рассмотрим структуру команд (или как ещё их называют – функций) языка программирования R.

В нашем примере « w » – это имя объекта R (или проще – переменная), « $<-$ » – функция присвоения, $c()$ – функция создания вектора (*concatenate* – собрать). Собственно, R и работает в основном с объектами и функциями. У объекта может быть своя структура, которую мы сейчас и проверим.

Шаг 2. Проверим структуру вектора командой $str()$ для того, чтобы убедиться, что вектор набран верно:

```
> str(w)
num [1:10] 7.1 7.7 3.6 8.3 8.8 10.4 8.9 9 8.9 14
```

В приведенном примере структура вектора представлена числами (num – сокращённо от *numeric*), вектор имеет 10 значений – $[1:10]$ и перечислены члены вектора от первого до десятого.

Шаг 3. Аналогично создаём и проверяем структуру вектора f .

2.2 Создание векторов при помощи RCommander

При создании векторов при помощи GUI (в данном случае – пакета RCommander) для набора данных используется сильно упрощенная электронная таблица. Рассмотрим это на примере выше упомянутых бурозубок.

Шаг 1. Загружаем среду программирования R и включаем пакет RCommander (раздел 1.1).

Шаг 2. Создаем новый вектор с данными, зайдя в меню **Данные** → **Новый набор данных...** (рисунок 12).

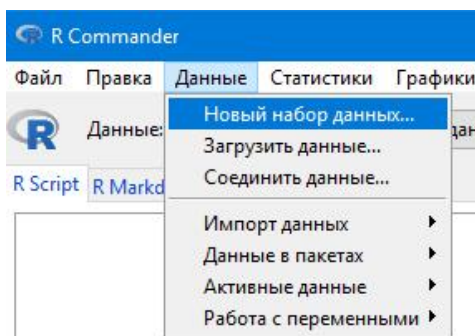


Рисунок 12 – Пункт меню о наборе новых данных в RCommander

Шаг 3. В открывшемся диалоговом окне выбираем название для вектора. В нашем случае – w, жмём кнопку **ОК** (рисунок 13).

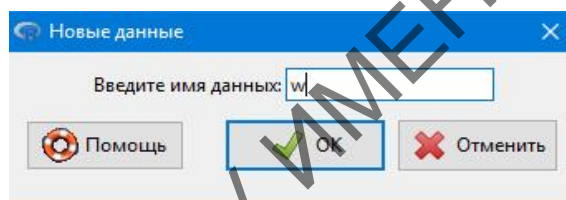


Рисунок 13 – Диалоговое окно с присвоением имени новому вектору в RCommander

Шаг 4. В появившемся окне набора данных (рисунок 14) в колонке под названием «V1» (variable 1 – переменная 1) вместо аббревиатуры «NA» набираем первое значение нашего вектора – 7.1, затем нажимаем клавишу со стрелкой вниз и **Enter**.

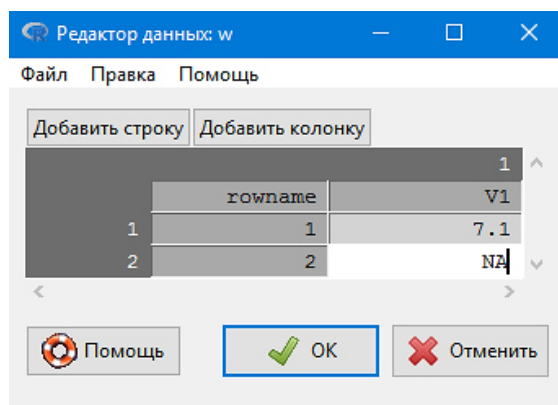
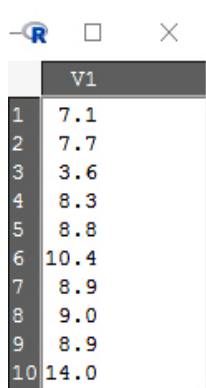


Рисунок 14 – Окно ввода новых данных в RCommander

Курсор перемещается на строку ниже. Повторяем процедуру до тех пор, пока не введем значения всего вектора, после чего необходимо нажать кнопку **ОК**. Наш вектор будет загружен в компьютер.

Шаг 5. Проверяем наш вектор при помощи процедуры, описанной выше (шаг 5 подпункта 1.2.2.1), и увидим результаты, отображенные на рисунке 15.



	v1
1	7.1
2	7.7
3	3.6
4	8.3
5	8.8
6	10.4
7	8.9
8	9.0
9	8.9
10	14.0

Рисунок 15 – Результаты проверки данных вектора w в RCommander

Шаг 6. Аналогично создаём и проверяем структуру вектора f .

Вопросы для самоконтроля

- 1 Назовите основные особенности языка программирования R.
- 2 При помощи чего в языке программирования R реализован ввод команд?
- 3 Как реализован графический интерфейс в языке программирования R? Какие возможности он предоставляет?
- 4 Что такое вектор в языке программирования R? Назовите основные функции вектора.

Задания для самоконтроля

1) Были получены следующие данные о весе тушканчиков (*Dipus aegyptius*):

Самцы	186	190	165	182	182	182	180	153	152
	173	157	179	164	146	173	144	151	173
	156	156	165	160	160	161	144		
Самки	162	163	190	188	147	146	145	153	165
	157	162	186	175	147	145	145	141	164
	155	174	180	148	175	145	144		

Сформируйте 2 вектора по весу самцов и самок тушканчиков при помощи командной строки и GUI.

2) Были изучены две выборки численности жесткокрылых на участке до посева газонной травы (А) и после (Б):

А		2	0	1	0	19	2	11	16	0	0	3	0	0	0	5	1	0
Б		1	1	14	1	11	3	3	30	1	20	5	1	2	1	16	1	5

Сформируйте 2 вектора по численности жуков при помощи командной строки и GUI.

3) Была изучена длина двухнедельных проростков кукурузы (в см) на участке до внесения удобрений (а) и после (б):

a		24	16	20	17	17	15	21	18	17	12	12	12	15	30	33	15	32	40	22	25
b		22	23	17	21	8	19	29	21	20	13	24	20	19	30	26	23	32	11	21	14

Сформируйте 2 вектора по проросткам кукурузы при помощи командной строки и GUI.

Литература по теме

1 Зарядов, И. С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика / И. С. Зарядов. – М. : Из-во Российского университета дружбы народов, 2010. – 207 с.

2 Мастицкий, С. Э. Статистический анализ и визуализация данных с помощью R / С. Э. Мастицкий, В. К. Шитиков. – М. : ДМК-пресс, 2015. – 496 с.

3 Наглядная статистика. Используем R! / А. Б. Шипунов [и др.]. – М. : ДМК-Пресс, 2017. – 296 с.

4 Dalgaard, P. Introductory Statistics with R / P. Dalgaard. – New York : Springer, 2008. – 370 p.

5 Hervé, M. Aide-mémoire de statistique appliquée à la biologie. Construire son étude et analyser les résultats à l'aide du logiciel R / M. Hervé [Электронный ресурс]. – Режим доступа: cran.r-project.org/doc/contrib/Herve-Aide-memoire-statistique.pdf. – Дата доступа: 16.02.2021.

6 Navarro, D. Learning statistics with R: A tutorial for psychology students and other beginners / D. Navarro. – Sydney : University of New South Wales, 2013. – 542 p.

ТЕМА 2. ОПЕРАЦИИ С ОБЪЕКТАМИ В R

1 Типы данных языка программирования R.

2 Объекты языка программирования R.

1 Типы данных и объекты языка программирования R

Каждый язык программирования, в том числе и рассматриваемый нами R, предполагает наличие данных определенного типа. Каждый тип данных характеризует некоторое множество значений и операции, которые можно применять к этим значениям. Любые данные, используемые в программе, относятся к тому или иному типу. В языке программирования R приняты следующие типы данных:

- **numeric** – числовой тип (включает в себя и целые числа, и дроби);
- **integer** – целочисленный тип (включает в себя только целые числа);
- **character** – символьный тип данных (каждый элемент в таком векторе является последовательностью из одного или более символов);
- **complex** – комплексный тип (содержит комплексные числа);
- **logical** – логический тип, принимает значения TRUE (правда) или FALSE (ложь).

Расскажем подробнее о каждом из этих типов данных.

1.1 Числовые данные (numeric)

Объекты этого типа могут содержать только числа. Эквивалентные обозначения (синонимы): **double** или **real**. Последнее существует только для сохранения обратной совместимости со старым кодом. То есть в языке программирования R имеют место три варианта обозначения векторов, содержащих числа с плавающей точкой. Объекты данного типа созданы для выполнения математических операций. Проверка на принадлежность переменной типу **numeric** производится путем выполнения функции (команды) `is.numeric()`. Например, у нас есть переменная *a*, которой мы присвоили значение 0,25 и нужно проверить, относится ли этот тип данных к числовым. Воспользуемся RStudio:

```
> a <- 0.25
> is.numeric(a)
[1] TRUE
```

1.2 Целочисленные данные (integer)

Этот тип данных создан для того, чтобы обеспечивать совместимость кода языка программирования R с кодом на языках программирования C или Fortran таким образом, чтобы представить целочисленные данные наиболее компактно. Следует отметить, что в актуальной на сегодня реализации интерпретатора языка программирования R используется 32-битный целочисленный тип данных, разброс значений которого ограничен от $-2 \cdot 10^9$ до $+2 \cdot 10^9$. В свою очередь, объекты типа **double** могут вмещать целые числа из более широкого диапазона. Чтобы убедиться, что необходимый нам вектор содержит целые числа, можно использовать функцию `is.integer()` из предыдущего примера:

```
> a <- 0.25
> is.integer(a)
[1] FALSE
```

1.3 Символьные данные (character)

Данный тип данных создан для выполнения операций с символами. Символом может быть что угодно: буквы алфавита в той или иной кодировке, цифры, а также любой другой символ, который пользователь сможет найти на своей клавиатуре или в сочетании клавиш. Понятно, что с векторами, состоящими из строк символов, невозможно проводить никаких вычислений, даже если в них содержатся цифры.

В тех случаях, когда с цифрами, содержащимися в строковых векторах, необходимо провести вычисления, их следует преобразовать в числовой тип функциями `as.integer()` или `as.numeric()`. Соответственно, если мы хотим преобразовать число в строку, то используется функция `as.character()`, а для выяснения типа переменной – `is.character()`. Посмотрим на конкретных примерах как это делается. Используем RStudio:

Шаг 1. Создадим и проверим символьный вектор *a*:

```
> a <- c("Вася", "5", "7", "male", "female")
> is.character(a)
[1] TRUE
```

Шаг 2. Преобразуем аргументы созданного вектора *a* в целые числа:

```
> as.integer(a)
[1] NA 5 7 NA NA
```

Аналогично конвертируйте символьный вектора a в числовой.

Здесь необходимо сделать небольшое отвлечение. Обращаем внимание на то, что значения, которые были не цифрами, а словами, программа заменила на выражение «NA». Таким образом, программа говорит нам, что эти данные не удалось конвертировать в числа и они «пропущены» и не будут учитываться. Такое может наблюдаться и в других случаях. Например, в процессе учёта беспозвоночных почвенными ловушками были получены данные только для семи ловушек из десяти, размещенных в ловушко-линии (три из них отсутствовали по непонятным причинам). Предполагаемый числовой вектор l будет иметь вид (для создания вектора используем RStudio или RCommander):

```
> l <- c(3, 5, NA, 6, NA, 4, 5, 1, NA, 2)
> l
[1] 3 5 NA 6 NA 4 5 1 NA 2
```

Теперь попробуем посчитать среднюю арифметическую значений вектора.

1 В RStudio:

```
> mean(l)
[1] NA
```

Программа отказывается нам считать среднее арифметическое и это естественно, так как неясно, как учитывать пропущенные данные. Для этого программе необходимо либо указать, что данные, обозначенные как «NA», должны быть проигнорированы:

```
> mean(na.omit(l))
[1] 3.714286
```

либо разрешить функции `mean()` принимать пропущенные данные:

```
> mean(l, na.rm=TRUE)
[1] 3.714286
```

2 В RCommander:

После набора нового вектора (например, по умолчанию – *Dataset*) и загрузки его в память необходимо перейти в меню: **Статистики** → **Итоги** → **Базовые статистики...** (рисунок 16).

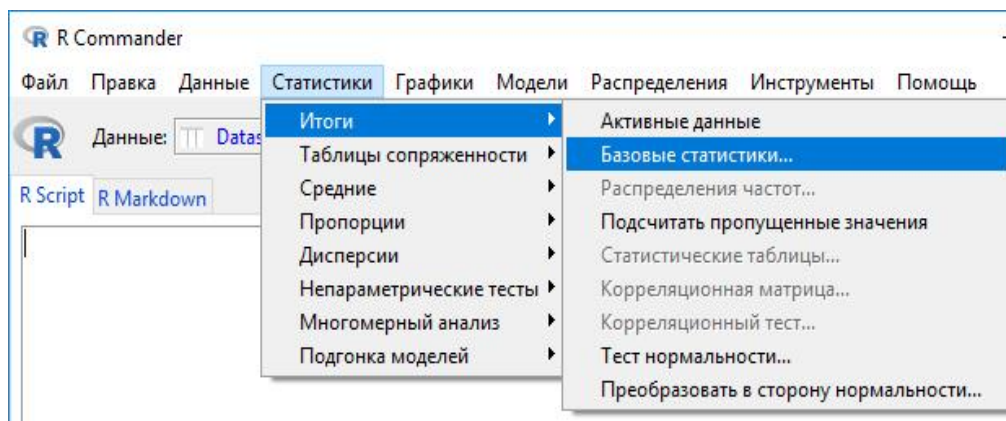


Рисунок 16 – Переход в пункт меню **Базовые статистики...** в RCommander

После этого на экране появится диалоговое окно описательной статистики **Числовые итоги** (рисунок 17).

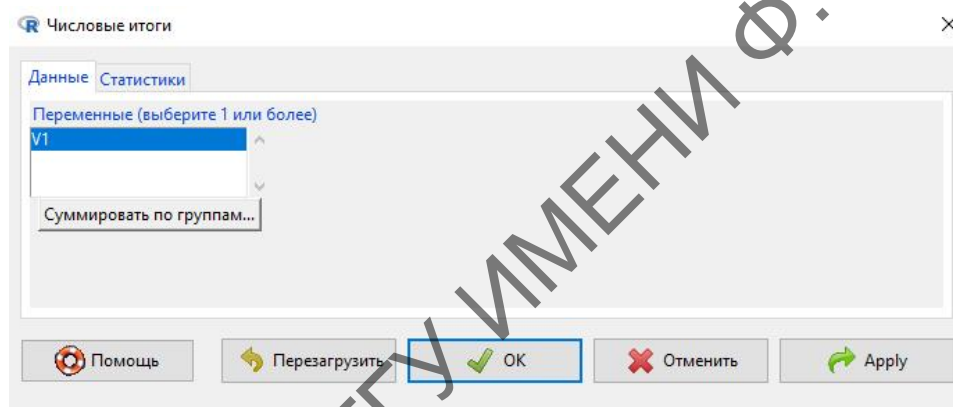


Рисунок 17 – Диалоговое окно **Числовые итоги**

Необходимо перейти на закладку **Статистики** (рисунок 18), поставить галочку на боксе **Среднее** и нажать **OK**.

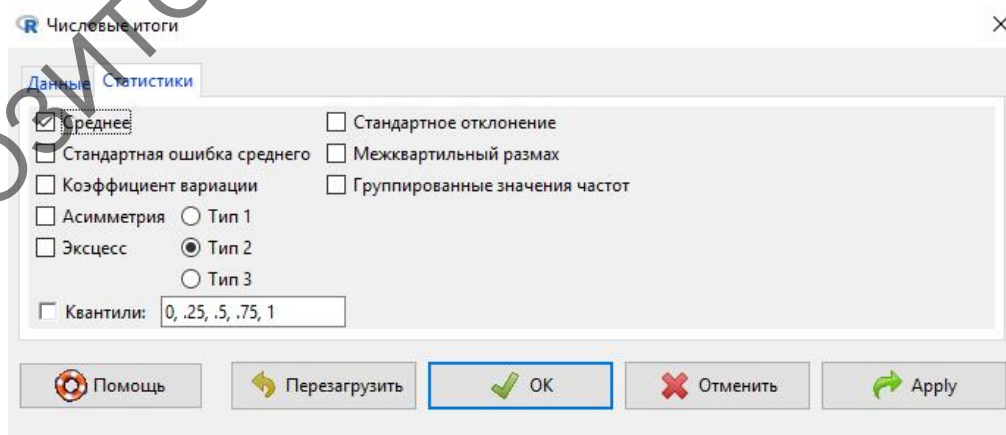


Рисунок 18 – Выделение параметра **Среднее** в закладке **Статистики** диалогового окна **Числовые итоги**

В командной строке среды программирования R появится следующее сообщение:

```
Rcmdr> numSummary(Dataset[, "V1", drop=FALSE], +
+statistics=c("mean"), quantiles=c(0, .25, .5, .75, 1))
      mean      n      NA
3.714286 7.000000 3.000000
```

Таким образом, пакет сам определяет, что пропущенные данные нужно игнорировать, при этом указывая, что учитывались 7 элементов вектора, а 3 – пропущенные значения.

1.4 Комплексные данные (complex)

Комплексные данные содержат комплексные числа, т. е. числа вида $a + bi$, где a , b – вещественные числа, i – мнимая единица (число, для которого выполняется равенство: $i^2 = -1$).

1.5 Логические данные (logical)

Логические переменные могут принимать только два противоположных по смыслу значения: TRUE (истина) и FALSE (ложь). TRUE и FALSE – это зарезервированные слова языка программирования R, которые обозначают логические константы. Эти обозначения при наборе программного кода (правда, не всегда) можно заменять заглавными символами «Т» и «F». При конвертации логического вектора в целочисленный вектор все значения TRUE будут заменены на 1, а FALSE – на 0. Например:

```
> b <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
> is.logical(b)
[1] TRUE
> as.integer(b)
[1] 1 1 1 0 1 0
```

2 Объекты языка программирования R

2.1 Матрицы

Матрицы – это довольно распространенная форма представления данных, организованных в форме таблицы. По своей сути матрица – это таблица, строки и столбцы, которые имеют свое обозначение. Любой ячейке матрицы можно присвоить «адрес», состоящий из маркера строки и маркера столбца. Например, шахматная доска с нотацией, игровое поле игры «Морской бой», рабочий лист любой электронной таблицы – это матрица.

В языке R матрицы могут быть разной размерности, даже трехмерные (но для удобства в учебных целях мы рассмотрим только двумерные). По своей сути, это всего лишь специальный тип вектора, который имеет добавочные свойства – атрибуты, позволяющие интерпретировать его как совокупность строк и столбцов.

Для примера: создадим простейшую матрицу размером в 3 строки и 3 столбца, итого – 9 ячеек. Используем для этого RStudio.

Шаг 1. Создаем числовой вектор:

```
> m <- 1:9 #Создаем вектор целых значений от 1 до 9
> m
[1] 1 2 3 4 5 6 7 8 9
```

Обратим внимание на строку, идущую после знака #. Это комментарий. Его можно использовать в процессе написания текста кода для пояснения действий, чтобы не забыть в дальнейшем. Программой он не учитывается.

Шаг 2. Создаем матрицу из вектора при помощи функции `matrix()`:

```
> ma <- matrix(m, ncol=3, byrow=TRUE)
> ma
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Атрибут `byrow=TRUE` указывает на то, что матрица будет заполняться построчно.

Шаг 3. Проверяем структуру полученной матрицы:

```
> str(ma)
int [1:3, 1:3] 1 4 7 2 5 8 3 6 9
```

Видно, что структура матрицы и нашего ранее созданного вектора `ma` мало отличаются.

Матрицы можно создавать и с заполнением по столбцам. Есть несколько способов.

1-й способ (с использованием атрибута `byrow=FALSE`):

```
> ma <- matrix(m, ncol=3, byrow=FALSE)
> ma
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```


2-й способ (при помощи функции `attr()` и атрибута “dim”):

```
> mb <- m
> mb
[1] 1 2 3 4 5 6 7 8 9
> attr(mb, "dim") <- c(3,3)
> mb
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

В данном случае атрибут “dim” (*dimensions* – размерность) указывает, что мы будем устанавливать размерность, и указываем значение этого атрибута в `c(3, 3)`, то есть 3 строки и 3 столбца.

В том случае, если необходимо матрицу перевернуть, т. е. транспонировать, то используется функция `t()` (работает и для таблиц). Развернем нашу матрицу `mb`:

```
> t(mb)
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

В том случае, когда нужно для каких-либо целей выбрать один из элементов матрицы, в квадратных скобках указывается номер столбца и номер строки – **именно в такой последовательности**. Например, из нашей перевернутой матрицы `mb` выберем значение под адресом (2; 3, т. е. пересечение второго столбца и третьей строки):

```
> mb[2, 3]
[1] 8
```

Трехмерные и многомерные матрицы в языке программирования R обозначаются как *array*.

2.2 Списки

Списки являются одним из важнейших объектов языка R. Несмотря на это, вам вряд ли в начале освоения этого достаточно мощного языка понадобится создавать списки для обработки данных, но в учебных целях мы познакомимся с ними, так как многие функции R после обработки выдают как результат именно списки.

Списки в языке R позволяют хранить в одной переменной объекты как одного, так и разных типов (в том числе и другие списки). Кроме того, объекты, входящие в список, могут быть не только разных типов, но и разного размера.

Создадим список «ls», содержащий все перечисленные выше особенности при помощи RStudio:

```
> ls <- list(colors = c("red", "green", "blue"), +
+ hours=1:24, c(TRUE, FALSE, FALSE), list("r", 4))
> ls
$colors
[1] "red" "green" "blue"

$hours
 [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24

[[3]]
[1] TRUE FALSE FALSE

[[4]]
[[4]][[1]]
[1] "r"

[[4]][[2]]
[1] 4
```

Для того чтобы выбрать элемент списка (это также называется индексированием), можно воспользоваться тремя способами.

1-й способ (используя квадратные скобки).

Выберем первый элемент списка:

```
> ls[1]
```

На экране отобразится результат:

```
$colors
[1] "red" "green" "blue"
```

Обратите внимание на символ «\$». При его помощи можно выбирать элементы не только списка, но и таблиц, что мы и увидим далее при рассмотрении таблиц данных.

2-й способ (используя двойные квадратные скобки).

```
> ls[[1]]
```

На экране отобразится результат:

```
[1] "red" "green" "blue"
```

Обратите внимание, что в данном случае не отображается название элемента списка – `$colors`, а только его содержимое.

3-й способ (используя имена членов списка).

Прежде чем использовать имена членов списка, необходимо их для начала создать. Под этими именами мы закодируем каждый элемент нашего списка. Так, в нашем списке 4 члена: цвета, время, логические значения и ещё один список. Назовем их по порядку от первого до четвертого:

```
> names(l1) <- c("first", "second", "third", "fourth")
```

А теперь проиндексируем, например, третий член нашего списка `l1`:

```
> l1$third
```

На экране отобразится результат:

```
[1] TRUE FALSE FALSE
```

Для выбора по имени также употребляется указанный выше символ «`$`», а полученный результат будет таким же, как при использовании во втором способе двойной квадратной скобки. Проиндексируйте самостоятельно остальные элементы списка.

2.3 Факторы

Фактор – это тип объектов, который используется для работы с категориальными или порядковыми данными. *Категориальные данные*, или, как их ещё называют, *номинальные данные* – это данные, измеренные в номинальной шкале, не имеющие количественного выражения и неупорядоченные. Примером таких значений могут быть названия населенных пунктов, географических объектов, пола, имена людей или клички животных. Ни один из приведенных примеров не может быть выше или ниже относительно друг друга. В принципе, можно обозначать различные номинальные показатели не буквами, а цифрами или даже целыми словами и специальными значками – все равно суть от этого не изменится.

Порядковые данные отличаются от категориальных тем, что в отличие от них позволяют упорядочивать объекты – то есть значения такой переменной могут быть больше или меньше относительно друг друга. В качестве примера порядкового объекта можно привести уровень доходов человека: низкий, средний, высокий. В данном случае понятно,

что низкий уровень ниже среднего, а средний – ниже высокого. В то же время конкретную величину различий в цифрах порядковая шкала измерений, как и номинальная, установить не позволяет.

Факторы в языке R получают из числового или символьного вектора с помощью функции `factor()`. При этом нужно отдавать себе отчет, что после конвертации в фактор числовые векторы преобразуются в символы и выполнение каких-либо арифметических действий с символьными факторами, полученными из чисел, невозможно, так как данные выражены уже в качественной, а не количественной шкале измерений.

В отличие от векторов с символьными данными, эти объекты имеют два атрибута: `"levels"` – название объекта, т. е. фактор (*factor*), и `"class"` – уровни фактора (все уникальные значения переменной).

Ниже рассмотрим пример работы с факторами. Для начала создадим символьный вектор, содержащий обозначения двух цветов – 5 элементов белого цвета «*white*» и 6 элементов черного цвета «*black*»:

```
> color <- rep(c("white", "black"), c(5, 6))
```

Посмотрим результат:

```
> color
[1] "white" "white" "white" "white" "white" "black"
[7] "black" "black" "black" "black" "black"
```

Проверим тип нашего полученного вектора *color*, и является ли он фактором:

```
> is.character(color)
[1] TRUE
> is.factor(color)
[1] FALSE
```

В результате выясняется, что полученный нами вектор *color* – это не фактор, а обычный символьный вектор без атрибутов.

Для того чтобы он был фактором, необходимо его преобразовать в фактор при помощи функции `factor()`. Для этого введем дополнительную переменную *color.f*:

```
> color.f <- factor(color)
> color.f
[1] white white white white white black black black black
[10] black black
Levels: black white
> is.factor(color.f)
[1] TRUE
```

Теперь у нас есть фактор *color.f* с двумя уровнями значений: «*black*» и «*white*». Для того чтобы узнать сколько раз встречается каждый из уровней, необходимо использовать функцию `summary()`, для нашего примера:

```
> summary(color.f)
black white
  6      5
```

Таким образом, мы видим, что уровню «*black*» соответствует 6 значений, а уровню «*white*» – 5.

В ряде случаев полученные факторы необходимо упорядочить. В качестве примера можно предложить измерение роста студентов в аудитории по шкале «высокий-средний-низкий», закодированные под обозначения «*t*», «*a*» и «*s*» соответственно, т. е. *tall* – высокий, *average* – средний и *short* – низкий. Это и будут уровни фактора. По умолчанию уровни фактора сортируются в алфавитном порядке. Мы введём дополнительную переменную «*rost.f.bad.order*», которой присвоим значение фактора. В названии переменной закодированы наши действия: название вектора, то, что это фактор, и то, что это неправильный для нас порядок. Рассмотрим этот наш пример подробнее при помощи RStudio.

```
> rost <- c("s", "s", "a", "t", "a", "a", "t", "a", "a", "a",
"s", "t", "s", "t", "t")
> rost.f.bad.order <- factor(rost)
> levels(rost.f.bad.order)
[1] "a" "s" "t"
```

Таким образом, видим, что уровни распределились в алфавитном порядке, а не по росту респондентов (так, как нам было нужно). Для того чтобы распределить уровни по нашему желанию, необходимо задать последовательность самостоятельно:

```
> rost.f <- factor(rost, levels = c("s", "a", "t"))
```

После чего проверяем, как распределились наши уровни:

```
> levels(rost.f)
[1] "s" "a" "t"
```

То есть уровни расположились так, как нам нужно – от низкого роста через средний до высокого. В то же время наша переменная «*rost.f*.» всё ещё остаётся номинальной, то есть её уровни нельзя сравнить между собой по величине. В этом легко убедиться, если проверить, например, первый уровень и второй:

```
> rost.f[1]<rost.f[2]
```

На экране появится сообщение об ошибке:

```
[1] NA
Предупреждение:
В Ops.factor(rost.f[1], rost.f[2]): '<' не значимо для факторов
```

Для того чтобы наша переменная «*rost.f*» стала ординальной (от слова *order* – порядок), т. е. её уровни были взаимозначимы и могли сравниваться между собой, необходимо упорядочить уровни фактора по значимости. Делается это при помощи атрибута `ordered=`. Введем дополнительную переменную «*rost.f.ordered*» и присвоим ей конвертированный в фактор вектор *rost*, а затем проверим:

```
> rost.f.ordered <- factor(rost, levels = c("s", "a", "t"),
ordered=TRUE)
> rost.f.ordered
[1] s s a t a a t a a a s t s t t
Levels: s < a < t
```

Сейчас мы видим, что уровни нашего фактора разместились по величине и их уже можно сравнивать между собой. Воспользуемся командой, которую мы уже использовали выше, и сравним первый уровень с третьим:

```
> rost.f.ordered[1]<rost.f.ordered[3]
[1] TRUE
```

Видим, что программа правильно определила, что первый уровень меньше третьего.

Иногда возникает необходимость добавить уровень в фактор. Но напрямую его добавить невозможно, программа выдаст ошибку. Например, в нашу импровизированную группу студентов добавился новичок-баскетболист и нам понадобился ещё один уровень – «*vt*» (*very tall* – очень высокий). Перед тем как добавлять его в фактор *rost.f*, уточним его размер, т. е. «длину»:

```
> length(rost.f)
[1] 15
```

Таким образом, мы видим, что добавляемый нами элемент нового уровня будет шестнадцатым. Пробуем добавить, указывая `R`, что шестнадцатый элемент будет «*vt*»:

```
> rost.f[16] <- "vt"
```

И получим предупреждение об ошибке:

```
предупреждение в `[<-.factor`(`*tmp*`, 16, value = "vt") :  
invalid factor level, NA generated
```

Программа говорит нам, что уровень фактора неверный и вместо его элемента будет сгенерирован «NA». Убедимся в этом непосредственно:

```
> rost.f  
[1] s s a t a a t a a a s t s t t <NA>  
Levels: s a t
```

После чего мы можем отметить, что наш элемент не вставлен (вместо него, как и было обещано, сгенерировано значение «NA») и количество уровней не изменилось. Выше показанным способом можно добавить элемент только уже имеющегося уровня. Например, ещё одного новичка среднего роста:

```
> rost.f[16] <- "a"  
> length(rost.f)  
[1] 16
```

Убеждаемся, что наш фактор вырос до 16 элементов. Для того чтобы добавить элемент другого уровня, и надо добавить этот уровень фактора:

```
> rost.f <- factor(rost.f, levels=c(levels(rost.f), "vt"))  
> rost.f[17] <- "vt" #добавляем элемент нового уровня  
> rost.f #проверяем наш фактор  
[1] s s a t a a t a a a s t s t t a vt  
Levels: s a t vt
```

Теперь видим, что добавлен и уровень, и элемент этого уровня. В то же время кроме добавления могут возникнуть случаи, когда необходимо удалить элементы уровня или даже сам уровень фактора. Сначала попробуем удалить первые три элемента фактора *rost.f*:

```
> rost.f.trunc <- rost.f[1:3]  
> rost.f.trunc  
[1] s s a  
Levels: s a t vt
```

Программа показала нам, что удалила элементы под названием «s», «s» и «a», а также продемонстрировала, что количество уровней не уменьшилось. Допустим, что наш новичок-баскетболист перешел в другую группу и его рост нужно удалить из общего фактора группы. Делается это через использование специального аргумента `drop=`:

```
> rost.f.trunc <- rost.f [1:16, drop=TRUE]  
> rost.f.trunc  
[1] s s a t a a t a a a s t s t t a  
Levels: s a t
```

Теперь видим, что был удалён последний, семнадцатый элемент, а так как он принадлежал уровню «vt», то с ним был удалён и сам уровень фактора.

2.4 Таблицы данных

Таблицы данных, или датафреймы (*data frame* – набор данных) – это электронные таблицы с данными, в которых хранят собранную в процессе исследования информацию, и те таблицы, в которые собственно оформляют данные для той или иной статистической обработки. Таблица данных по своей сути является набором векторов, каждый из которых представлен столбцом таблицы. Для создания датафрейма необходимо соблюдение двух условий:

1 Размер векторов, составляющих датафрейм, должен быть одинаковой длины (то есть содержать одинаковое количество элементов).

2 Данные, содержащиеся в векторе, должны быть одного типа.

Рассмотрим создание таблиц данных как с помощью командной строки в RStudio, так и при помощи пакета RCommander. Предположим, что у нас имеется набор данных об особенностях восьми экземпляров 4 видов мышей, обитавших в трех разных биотопах (таблица 3).

Таблица 3 – Информация о собранных в ловушки мышах

Вид	Вес, г	Пол	Биотоп	Вид	Вес, г	Пол	Биотоп
Мышь1	9,8	male	Б1	Мышь 1	9,3	female	Б2
Мышь 2	12,1	male	Б3	Мышь 3	10,5	female	Б2
Мышь 4	6,7	female	Б1	Мышь 2	11,1	male	Б3
Мышь 1	8,7	male	Б2	Мышь 4	7,3	female	Б1

Таким образом, наша таблица данных будет состоять из 4 векторов: вид, вес, пол и биотоп.

2.4.1 Создание таблиц данных

2.4.1.1 Создание таблицы данных в RStudio

Шаг 1. Для начала создадим символьный вектор, обозначающий совокупность видов мышей, и проверим его структуру:

```
> sp <- c("Мышь1", "Мышь2", "Мышь4", "Мышь1", "Мышь1",
"Мышь3", "Мышь2", "Мышь4")
> str(sp)
chr [1:8] "Мышь1" "Мышь2" "Мышь4" "Мышь1" "Мышь1" "Мышь3"
"Мышь2" "Мышь4"
```


Шаг 2. Далее создадим числовой вектор, обозначающий массу мышей, и проверим его структуру:

```
> ves <- c(9.8, 12.1, 6.7, 8.7, 9.3, 10.5, 11.1, 7.3)
> str(ves)
num [1:8] 9.8 12.1 6.7 8.7 9.3 10.5 11.1 7.3
```

При создании вектора обратите внимание на отделение дробной части от целого, здесь это точка, а не запятая.

Шаг 3. Создадим ещё один символьный вектор, обозначающий пол мышей, и проверим его структуру:

```
> s <- c("male", "male", "female", "male", "female", "female", "male", "female")
> str(s)
chr [1:8] "male" "male" "female" "male" "female" "female" "male" "female"
```

Шаг 4. Наконец, создадим символьный вектор, обозначающий номер биотопа, где обитали мыши и проверим его структуру:

```
> b <- c("Б1", "Б3", "Б1", "Б2", "Б2", "Б2", "Б3", "Б1")
> str(b)
chr [1:8] "Б1" "Б3" "Б1" "Б2" "Б2" "Б2" "Б3" "Б1"
```

Работа по созданию векторов закончена. Осталось лишь конвертировать их в таблицу при помощи функции `data.frame()`.

Шаг 5. Объединяем созданные ранее вектора в датафрейм *mouse*:

```
> mouse <- data.frame("Вид"=sp, "Вес, г"=ves, "Пол"=s, "Биотоп"=b)
```

Шаг 6. Посмотрим на нашу получившуюся таблицу:

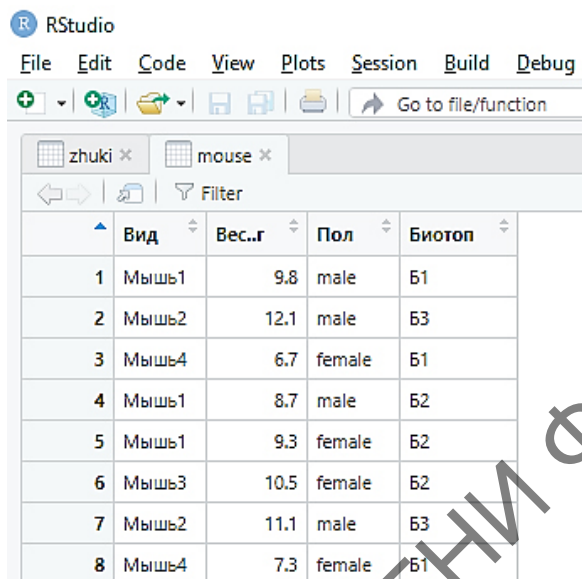
```
> mouse
  Вид Вес, г Пол Биотоп
1 Мышь1  9.8 male   Б1
2 Мышь2 12.1 male   Б3
3 Мышь4  6.7 female  Б1
4 Мышь1  8.7 male   Б2
5 Мышь1  9.3 female  Б2
6 Мышь3 10.5 female  Б2
7 Мышь2 11.1 male   Б3
8 Мышь4  7.3 female  Б1
```

Посмотреть полученную таблицу можно также при помощи команды `view()`. В нашем случае:

```
> view(mouse)
```

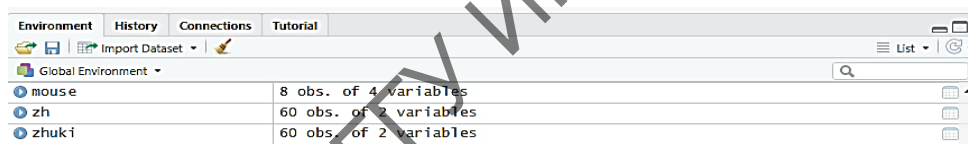
Созданная нами таблица отобразится в левом верхнем окне RStudio (рисунок 19).

Загрузить данные в окно отображения данных можно не набирая команду `view()`, а нажав кнопку в окне загруженных переменных (рисунок 20), найдя название нашего датафрейма.



	Вид	Вес..г	Пол	Биотоп
1	Мышь1	9.8	male	Б1
2	Мышь2	12.1	male	Б3
3	Мышь4	6.7	female	Б1
4	Мышь1	8.7	male	Б2
5	Мышь1	9.3	female	Б2
6	Мышь3	10.5	female	Б2
7	Мышь2	11.1	male	Б3
8	Мышь4	7.3	female	Б1

Рисунок 19 – Созданная таблица в окне отображения загруженных данных RStudio



Object	Details
mouse	8 obs. of 4 variables
zh	60 obs. of 2 variables
zhuki	60 obs. of 2 variables

Рисунок 20 – Список созданных таблиц в окне отображения данных в RStudio

2.4.2 Создание таблицы данных при помощи пакета RCommander

Если перед тем, как работать в RCommander, Вы делали датафрейм в RStudio и он остался загруженным в память среды программирования R, то его необходимо выгрузить командой `rm()`.

```
> rm(mouse)
```

Дело в том, что и RStudio, и RCommander всего лишь надстройки над самой средой R и то, что Вы делаете в одной программе, отображается и в другой.

Шаг 1. Загружаем среду R и включаем пакет RCommander (тема 1; раздел 1.1).

Шаг 2. Создаем новую таблицу с данными, зайдя в меню **Данные** → **Новый набор данных...** (рисунок 21).

Шаг 3. В открывшемся диалоговом окне выбираем название для таблицы. В нашем случае – *mouse*, и жмём кнопку **ОК** (рисунок 22).

Шаг 4. В появившемся окне набора данных поочередно нажимаем кнопки **Добавить строку** и **Добавить колонку** пока не создадим макет нашей таблицы для ввода данных (рисунок 23).

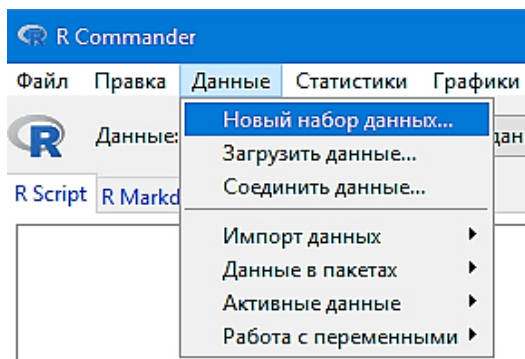


Рисунок 21 – Пункт меню о наборе новых данных в RCommander

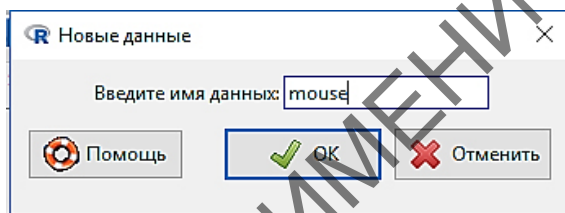


Рисунок 22 – Диалоговое окно с присвоением имени новой таблице данных в RCommander

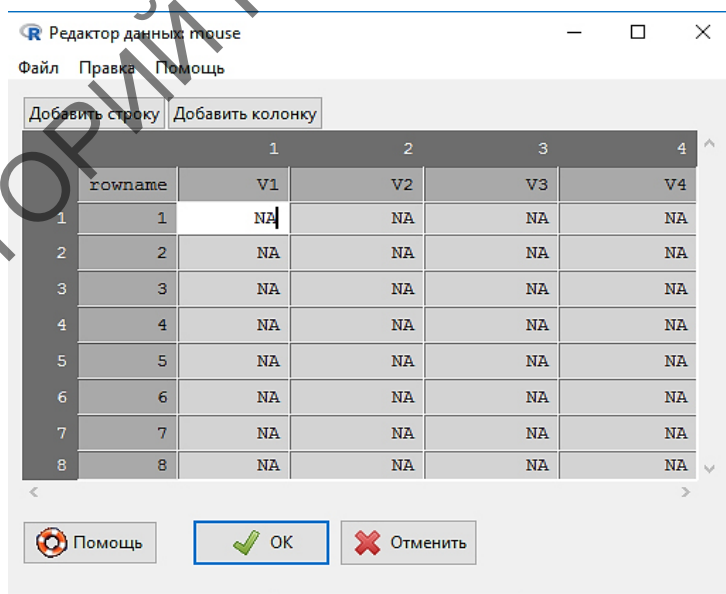


Рисунок 23 – Окно ввода новых данных в RCommander с макетом будущей таблицы

Шаг 5. В ячейках под названием «V1» (variable 1 – переменная 1) замените его на название первой нашей колонки, т. е. **Вид**. Аналогично поменяйте все названия колонок на нужные, а затем заполните таблицу данными вместо аббревиатур «NA», перемещаясь по ячейке таблицы при помощи мыши или клавиш стрелок на клавиатуре. Законченная таблица будет иметь вид, показанный на рисунке 24. После чего нажмите на кнопку **ОК**. Наша таблица готова и загружена в компьютер, а в командной строке среды R будет об этом приведено сообщение, переданное из RCommander:

RcmdrMsg: [2] ЗАМЕЧАНИЕ: данные mouse: 8 строк(а) и 4 колонк(а).

	1	2	3	4
rowname	Вид	Вес, г	Пол	Биотоп
1	Мышь1	9.8	male	B1
2	Мышь2	12.1	male	B3
3	Мышь4	6.7	female	B1
4	Мышь1	8.7	male	B2
5	Мышь1	9.3	female	B2
6	Мышь3	10.5	female	B2
7	Мышь2	11.1	male	B3
8	Мышь4	7.3	female	B1

Рисунок 24 – Готовая таблица, созданная в RCommander

Можете убедиться в этом самостоятельно, проверив в RStudio при помощи команды `view()`.

Существует также и третий способ набора данных – просто создать таблицу в сторонней программе электронных таблиц, например в Excel, и затем загрузить их в R. Как это делается, мы рассмотрели ранее в предыдущей теме (тема 1; раздел 2.1).

2.4.2 Индексация данных в таблице, коррекция и сохранение данных

2.4.2.1 Индексация данных

В связи с тем, что таблица данных по сути является списком, к ней также применимы ранее нами рассмотренные методы индексации списков. Датафреймы индексируются как двумерные матрицы. Рассмотрим это на нашем примере про мышей. Допустим, для работы нам нужно просмотреть только столбец с весом мышей. Вот способы, как это можно сделать.

1-й способ:

```
> mouse$Вес..г  
[1] 9.8 12.1 6.7 8.7 9.3 10.5 11.1 7.3
```

2-й способ:

```
> mouse[[2]]  
[1] 9.8 12.1 6.7 8.7 9.3 10.5 11.1 7.3
```

3-й способ:

```
> mouse[,2]  
[1] 9.8 12.1 6.7 8.7 9.3 10.5 11.1 7.3
```

4-й способ:

```
> mouse[, "Вес..г"]  
[1] 9.8 12.1 6.7 8.7 9.3 10.5 11.1 7.3
```

Возникает в ряде случаев также необходимость просмотреть несколько столбцов одновременно. Например, в нашей таблице нам нужно проиндексировать столбцы с полом мышей и где они обитают, т. е. третий и четвертый. Проще всего это сделать при помощи следующей команды:

```
> mouse[,3:4]  
  Пол Биотоп  
1  male     Б1  
2  male     Б3  
3  female   Б1  
4  male     Б2  
5  female   Б2  
6  female   Б2  
7  male     Б3  
8  female   Б1
```

Далее, допустим, нам нужно выяснить, кто из наших отловленных мышей самки. Это делается при помощи логического выражения:

```
> mouse[mouse$Пол=="female",]  
 Вид Вес..г Пол Биотоп  
3 Мышь4 6.7 female Б1  
5 Мышь1 9.3 female Б2  
6 Мышь3 10.5 female Б2  
8 Мышь4 7.3 female Б1
```

Таким образом, программа отобразила нам из выборки только мышей женского пола и их характеристики по остальным столбцам.

В том случае, если возникает необходимость отсортировать столбцы таблицы данных по какому-либо признаку, используется функция `order()`. Например, необходимо отсортировать датафрейм сначала по месту обитания мышей и одновременно по полу.

```
> mouse.sorted <- mouse[order(mouse$Биотоп, mouse$Пол), ]
> mouse.sorted
  Вид Вес..г Пол Биотоп
3 Мышь4 6.7 female Б1
8 Мышь4 7.3 female Б1
1 Мышь1 9.8 male Б1
5 Мышь1 9.3 female Б2
6 Мышь3 10.5 female Б2
4 Мышь1 8.7 male Б2
2 Мышь2 12.1 male Б3
7 Мышь2 11.1 male Б3
```

Отмечаем, что все отсортировано так, как мы и просили. Попробуйте отсортировать самостоятельно сначала по биотопу, а потом по весу, и наоборот.

В пакете RCommander посмотреть отдельно столбец не получится. При нажатии кнопки **Просмотреть данные** загружается таблица целиком (рисунок 25).

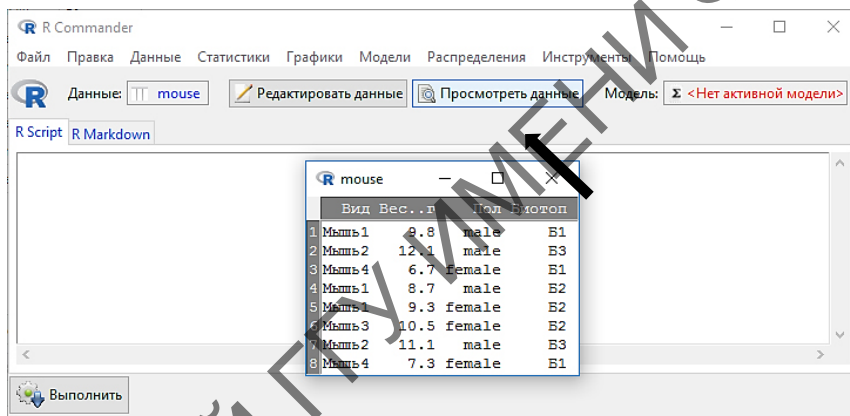


Рисунок 25 – Просмотр таблицы с данными в RCommander

В то же время существует много возможностей по операции как со всеми данными таблицы, так и с отдельными столбцами, т. е. переменными. Для того чтобы посмотреть виды операций с переменными в пакете RCommander, необходимо перейти в меню по пути: **Данные** → **Активные данные...** (рисунок 26).

В этом пункте меню перечислены операции, которые можно сделать с данными, отраженными в загруженной таблице. Но нужно понимать, что это никак не связано со статистической обработкой. Это всего лишь подготовка данных к работе. Для примера отсортируем наши данные по мышам, как и ранее – по биотопам и полу.

Шаг 1. Для этого в RCommander заходим в меню по пути: **Данные** → **Активные данные...** → **Сортировать активный набор данных...** (рисунок 26).

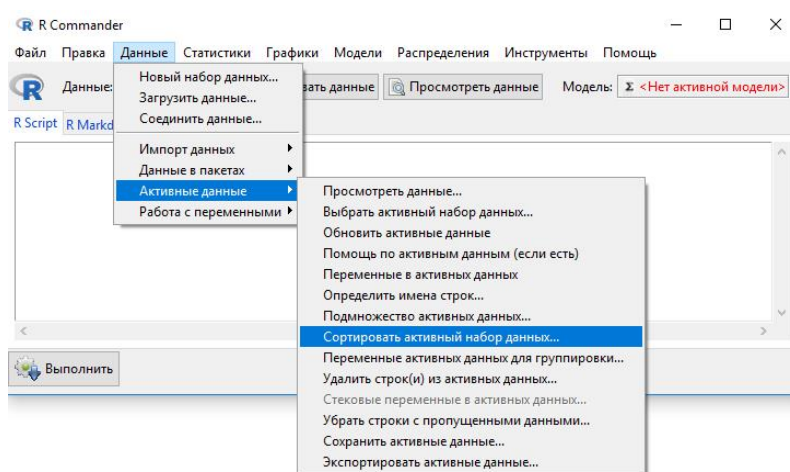


Рисунок 26 – Меню активных данных в RCommander

Шаг 2. После этого, в появившемся диалоговом окне **Сортировать активный набор данных...** в боксе **Ключи сортировки** при нажатой клавише **Ctrl** указателем мыши кликните сначала биотоп, а потом пол. Диалоговое окно примет вид, как на рисунке 27. После чего нажмите **ОК**.

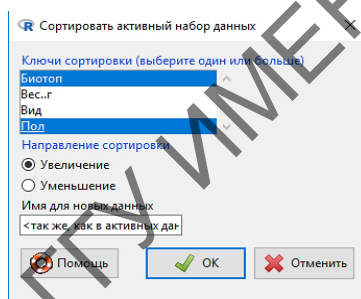


Рисунок 27 – Диалоговое окно **Сортировать активный набор данных...** в RCommander

Шаг 3. В появившемся диалоговом окне **Переставить ключи сортировки...** укажите порядок сортировки. В нашем случае Биотоп – это 1, а Пол – это 2. Диалоговое окно примет вид, как на рисунке 28. После чего нажмите **ОК**.

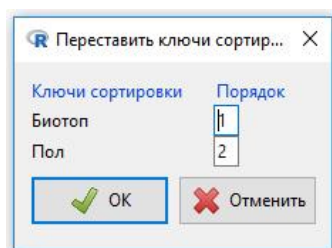


Рисунок 28 – Диалоговое окно **Переставить ключи сортировки...** в RCommander

Шаг 4. Таблица отсортирована. Убедимся в этом, нажав кнопку **Просмотреть данные**. Полученная таблица – на рисунке 29.

	Вид	Вес..г	Пол	Биотоп
3	Мышь4	6.7	female	B1
8	Мышь4	7.3	female	B1
1	Мышь1	9.8	male	B1
5	Мышь1	9.3	female	B2
6	Мышь3	10.5	female	B2
4	Мышь1	8.7	male	B2
2	Мышь2	12.1	male	B3
7	Мышь2	11.1	male	B3

Рисунок 29 – Отсортированная в RCommander таблица данных

2.4.2.2 Коррекция данных в таблице

Иногда возникает необходимость исправлять уже введенные в таблицу данные из-за невнимательности, набранные ошибочно. Исправить их можно несколькими путями.

1 В RStudio:

Для правки данных в RStudio можно воспользоваться функциями `edit()` и `fix()`. Какая в них разница, сейчас вы увидите. Допустим, в нашем датафрейме о мышах, вводя данные по весу, мы ошиблись и вес самца мыши1, обитавшем в биотопе 1, составил не 9,8 г, как мы внесли в таблицу, а 9,5 г. Исправить эту ошибку мы можем, использовав для начала функцию `edit()`:

Шаг 1. Загружаем редактор данных:

```
> edit(mouse)
```

На экране монитора появится очень упрощённый аналог электронной таблицы, где будут показаны наши данные (рисунок 30).

	row.names	Вид	Вес..г	Пол	Биотоп
1	3	Мышь4	6.7	female	B1
2	8	Мышь4	7.3	female	B1
3	1	Мышь1	9.8	male	B1
4	5	Мышь1	9.3	female	B2
5	6	Мышь3	10.5	female	B2
6	4	Мышь1	8.7	male	B2
7	2	Мышь2	12.1	male	B3
8	7	Мышь2	11.1	male	B3

Рисунок 30 – Редактор данных в RStudio

Обращаем внимание на тот факт, что в этом редакторе не всегда корректно могут отображаться надписи на кириллице. Это надо иметь в виду, и если Вы часто планируете работать с данным редактором, то символьные данные (и заголовки таблиц) лучше заранее делать в английской раскладке.

Шаг 2. Меняем значение «9.8» на «9.5», нажимаем **Enter** и закрываем редактор. Тут же на экране появляется датафрейм с изменёнными данными:

Вид	Вес..г	Пол	Биотоп
3 Мышь4	6.7	female	Б1
8 Мышь4	7.3	female	Б1
1 Мышь1	9.5	male	Б1
5 Мышь1	9.3	female	Б2
6 Мышь3	10.5	female	Б2
4 Мышь1	8.7	male	Б2
2 Мышь2	12.1	male	Б3
7 Мышь2	11.1	male	Б3

Но если посмотреть в окно загруженных таблиц RStudio (левое верхнее окно), то можно отметить, что там изменений не произошло. То есть мы изменили данные только на текущий сеанс для конкретных расчётов и не более. Глобальной замены не произошло. Для этого необходимо воспользоваться функцией `fix()`.

Шаг 3. Загружаем окно редактора данных:

```
> fix(mouse)
```

На экране монитора появится точно такое же окно редактора данных, как и в шаге 1 (рисунок 30).

Шаг 4. Меняем значение «9.8» на «9.5», нажимаем **Enter** и закрываем редактор. Проверяем полученные изменения:

```
> mouse
```

Вид	Вес..г	Пол	Биотоп
3 Мышь4	6.7	female	Б1
8 Мышь4	7.3	female	Б1
1 Мышь1	9.5	male	Б1
5 Мышь1	9.3	female	Б2
6 Мышь3	10.5	female	Б2
4 Мышь1	8.7	male	Б2
2 Мышь2	12.1	male	Б3
7 Мышь2	11.1	male	Б3

Если мы посмотрим в окно загруженных таблиц, то убедимся, что данные изменились и там.

2. В RCommander:

В этом пакете все проще. При загруженных данных о наших мышах необходимо нажать кнопку **Редактировать данные** и появится тот редактор, в котором мы их набирали ранее (рисунки 23, 24). На этот раз поменяем значение «9.5» на «9.8» после чего нажмем на **ОК**. Изменения сохранены.

2.4.2.3 Сохранение таблицы данных на диск

Перед тем как сохранить свои данные на диск компьютера, нужно определиться, будете ли Вы в дальнейшем использовать их в работе только в среде R или в какой-нибудь ещё программе, например, в среде обработки электронных таблиц. От этого зависит функция, которой необходимо воспользоваться при сохранении.

В том случае, если Вам нужен только язык программирования R, то наиболее удобно сохранять (а затем и загружать) данные в его собственном формате. Для этого используются функции `save()` – для сохранения и `load()` – для загрузки. Как можете увидеть, расширение собственного формата файлов языка программирования R – `*.rd`.

Рассмотрим это в RStudio на примере нашей таблицы с мышами:

```
> save(mouse, file="mouse.rd") #Собственно операция сохранения
> exists("mouse") #Проверка того, что объект удалён
[1] TRUE
> load("mouse.rd") #Загрузить объект "mouse"
```

Обращаем внимание на то, что здесь нет полного указания пути к файлу, так как файл сохраняется в рабочем каталоге R.

Для того чтобы сохранить таблицу, а затем при необходимости открыть её в Excel или подобной программе, используется функция `write.table()`. В нашем примере:

```
> write.table(mouse, file="mouse.csv", quote=T, sep=";", dec=".",
+row.names=F, col.names=T)
```

Здесь необходимо пояснить аргументы функции `write.table()`:

- **mouse** – название таблицы с данными;
- **file="mouse.csv"** – имя файла с таблицей;
- **quote=F** – наличие – T (или отсутствие – F) в таблице логических переменных;
- **sep=";"** – разделитель столбцов (*separate* – разделять), в данном случае – точка с запятой, бывает также пробел, запятая, знак табуляции, двоеточие;

- **dec="."** – знак того, что является отделителем целого от дробного: точка или запятая;
- **row.names=F** – наличие – T (или отсутствие – F) названия строк таблицы;
- **col.names=T** – наличие – T (или отсутствие – F) названия столбцов таблицы.

Эта функция является более универсальной и может сохранять данные в разных форматах.

Можно также воспользоваться функциями `write.csv()` и `write.csv2()`. Они имеют схожую структуру с функцией `write.table()`:

```
> write.csv(mouse, file="mouse.csv", sep=";", dec=".", +row.names=F,
col.names=T)
> write.csv2(mouse, file="mouse.csv", sep=";", dec=".",
+row.names=F, col.names=T)
```

Средствами RCommander можно сохранить таблицу с данными либо свой собственный формат R, перейдя по меню: **Данные** → **Активные данные** → **Сохранить активные данные...** (рисунок 31) и затем указать адрес и имя файла либо конвертировать в текстовый формат, перейдя по меню: **Данные** → **Активные данные** → **Экспортировать активные данные...** (рисунок 32).

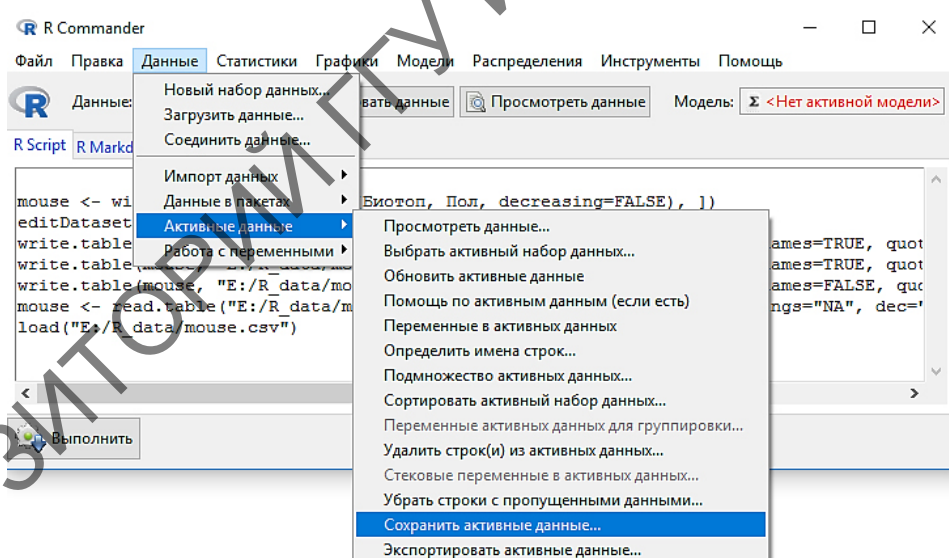


Рисунок 31 – Сохранение данных в RCommander в формате *.RData

После выбора экспорта активных данных в текстовый формат появится диалоговое окно **Экспортировать активные данные** (рисунок 33), где необходимо будет указать необходимые настройки будущего файла после конвертации и нажать **ОК**.

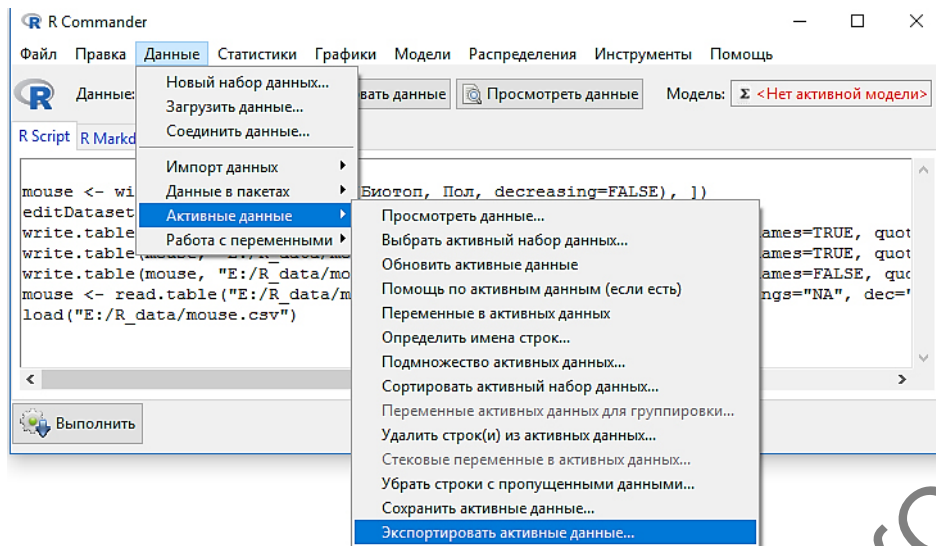


Рисунок 32 – Экспорт данных в RCommander в текстовый формат

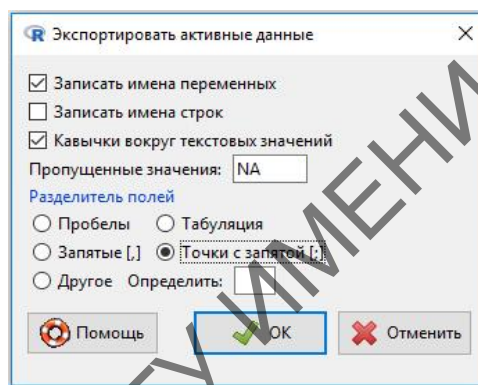


Рисунок 33 – Диалоговое окно Экспорт активных данных в RCommander

Необходимо сказать также пару слов о рабочей папке среды программирования R. По умолчанию эта папка – каталог с установленными рабочими файлами R. Это не совсем удобно, и не всегда хочется захламлять файлы с промежуточными данными папку самой программы. Поэтому логично сменить папку, например, на *R_data* (данные R) на другом логическом диске, заодно и страхуясь от возможной их потери.

Сменить рабочую папку можно через командную строку в RStudio. На нашем примере:

```
> setwd("e:/R_data") #Установка пути к новой рабочей папке
> getwd()# Проверка расположения рабочей папки
[1] "e:/R_data"
```

Сменить месторасположение и имя рабочей папки можно и используя GUI. Вот несколько способов:

1-й способ (в самой среде программирования R)

В среде программирования R перейти в меню: **Файл** → **Изменить папку...** (рисунок 34) и в появившемся диалоговом окне указать имя и адрес новой рабочей папки (или создать её, если ранее не была создана).

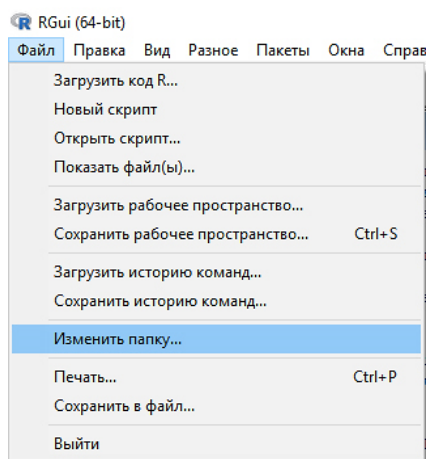


Рисунок 34 – Меню **Изменить папку...** в R

2-й способ (при помощи RStudio)

Здесь также используется меню, но настраивается место и имя рабочей папки в окне настроек после перехода в меню: **Tools** → **Global options** (рисунок 35).

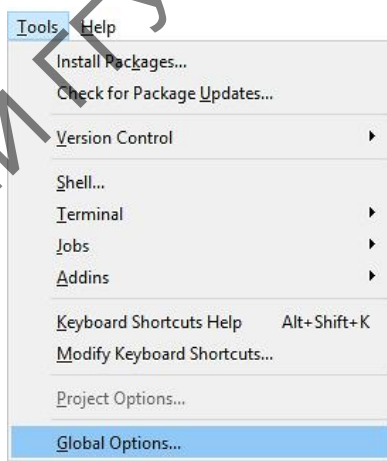


Рисунок 35 – Меню **Global options...** в RStudio

После этого мы попадаем в само диалоговое окно общих настроек и в разделе **General** (Наиболее общие настройки) на закладке **Basic** (Основные) в строке бокса **Default working directory** (Рабочая директория по умолчанию) необходимо выбрать путь к новой рабочей папке, вручную или предварительно нажав рядом кнопку **Browse...** (Обзор).

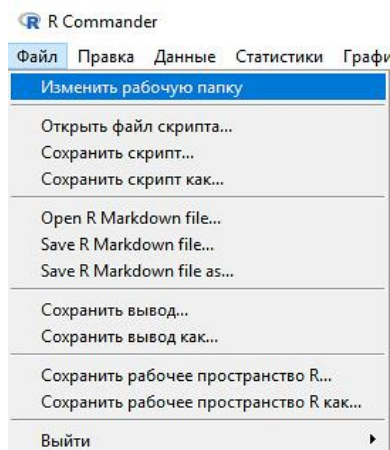


Рисунок 36 – Меню Изменить рабочую папку в RCommander

3-й способ (при помощи RCommander)

Здесь все просто. Необходимо перейти в меню: **Файл** → **Изменить рабочую папку** (рисунок 36), а затем в появившемся диалоговом окне указать имя и адрес новой рабочей папки (или её создать, если ранее она не была создана).

Вопросы для самоконтроля

- 1 Какие типы данных встречаются в языке программирования R?
- 2 Какие числовые и целочисленные типы данных существуют в языке программирования R?
- 3 Какие особенности символьных и логических данных в языке программирования R?
- 4 Как реализованы матрицы в языке программирования R?
- 5 Каким образом составляются списки в языке программирования R и проводится их индексация?
- 6 Что представляют из себя факторы как объекты в языке программирования R?
- 7 Как отражены такие объекты, как таблицы данных, или датафреймы в языке программирования R?
- 8 Как создаются таблицы данных и проводится индексация переменных в датафреймах?
- 9 Каким образом можно сохранить таблицы данных в R?

Задания для самоконтроля

1) Имеются данные по весу (в г) микромаммалий в окрестностях нефтяной скважины:

7,2 8,3 9,0 8,1 6,9 12,0 11,6 9,2 7,3 7,1 9,0 7,9 8,2 6,3 9,1 10,0 7,0

Создайте вектор из приведенных данных. Выясните, является он числовым или целочисленным?

2) Создайте символьный вектор из чисел на циферблате часов. Преобразуйте его в числовой.

3) Имеются данные по численности яиц в гнёздах синицы:

№ гнезда	1	2	3	4	5	6	7	8	9	10
Количество	4	5	–	5	3	–	4	–	4	3

Создайте числовой вектор, используя обозначение «NA» на месте пропущенных данных. Рассчитайте среднее число яиц в гнёздах синицы, используя командную строку и GUI.

4) Имеются данные о количестве зёрен в 16 бобах гороха:

5 4 4 6 5 5 4 5 6 6 7 4 3 7 5 5

Создайте матрицу 4x4 с заполнением ячеек матрицы построчно. Поменяйте построчное заполнение ячеек матрицы на заполнение по столбцам различными способами, включая транспонирование.

5) Создайте список, содержащий следующие данные о членах учебной группы: имена; возраст в годах; цвет волос; список, содержащий номер учебной группы и количество изучаемых иностранных языков членами этой группы.

6) Проиндексируйте список, созданный в предыдущем задании, выбрав из него: только имена членов учебной группы; только их возраст.

7) Пронумеруйте членов списка, подготовленного в задании № 5, и проиндексируйте третий элемент списка.

8) Отметки в учебной ведомости академической группы по результатам экзамена распределились по следующим категориям номинальной шкалы:

«poor» – 4

«average» – 10

«good» – 14

«exellent» – 2

Создайте символьный вектор, содержащий данные по результатам экзамена, и конвертируйте его в фактор. Отсортируйте уровни фактора сначала по алфавиту, а затем – по значимости. Упорядочите уровни фактора, отсортированные по значимости.

9) На белых крысах была показана следующая зависимость между температурой внешней среды x (в град.) и количеством поглощенного кислорода y (в мл/г веса):

x	0	5	10	15	20	25	28	29	30	31	32	33	34	35	40
y	3,83	3,35	2,60	2,02	1,69	1,42	1,39	1,38	1,29	1,39	1,39	1,45	1,65	1,61	2,40

Создайте датафрейм по количеству поглощенного кислорода белыми крысами.

10) Было проведено внесение удобрений разных смесей (b , c , d) под посевы ячменя, на одно поле удобрения не вносились (a) для увеличения урожайности (в ц/га):

a 9,4 8,4 4,9 13,4 11,1 3,6 15,1 13,2 12,0 11,6 7,3
 b 21,1 8,2 16,6 10,5 18,4 11,3 18,3 11,2 19,4 10,0 18,2
 c 12,5 12,1 12,1 15,4 2,0 9,5 12,1 11,0 5,8 16,0 8,1
 d 21,0 5,0 24,1 14,8 29,1 20,1 11,2 19,5 20,5 22,9 20,0

Создайте датафрейм по урожайности ячменя.

Литература по теме

1 Зарядов, И. С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика / И. С. Зарядов. – М.: Из-во Российского университета дружбы народов, 2010. – 207 с.

2 Мастицкий, С. Э. Статистический анализ и визуализация данных с помощью R / С. Э. Мастицкий, В. К. Шитиков. – М.: ДМК-пресс, 2015. – 496 с.

3 Наглядная статистика. Используем R! / А. Б. Шипунов [и др.]. – М.: ДМК-Пресс, 2017. – 296 с.

4 Hervé, M. Aide-mémoire de statistique appliquée à la biologie. Construire son étude et analyser les résultats à l'aide du logiciel R / M. Hervé [Электронный ресурс]. – Режим доступа: cran.r-project.org/doc/contrib/Herve-Aide-memoire-statistique.pdf. – Дата доступа: 16.02.2021.

5 Navarro, D. Learning statistics with R: A tutorial for psychology students and other beginners / D. Navarro. – Sydney : University of New South Wales, 2013. – 542 p.

Производственно-практическое издание

Галиновский Николай Геннадьевич

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ R

Практическое пособие

Редактор А. А. Негодина
Корректор В. В. Калугина

Подписано в печать 03.12.2021. Формат 60x84 1/16.

Бумага офсетная. Ризография.

Усл. печ. л. 3,02. Уч.-изд. л. 3,30.

Тираж 25 экз. Заказ 637.

Издатель и полиграфическое исполнение:
учреждение образования

«Гомельский государственный университет имени Франциска Скорины».

Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1452 от 17.04.2017.

Специальное разрешение (лицензия) № 02330 / 450 от 18.12.2013.

Ул. Советская, 104, 246028, Гомель