

Министерство образования Республики Беларусь

Учреждение образования  
«Гомельский государственный университет  
имени Франциска Скорины»

**Н. Г. ГАЛИНОВСКИЙ**

**ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ  
НА ЯЗЫКЕ R**

Рекомендовано Учебно-методическим объединением  
по естественно-научному образованию в качестве пособия  
для магистрантов, обучающихся по специальности «Биология»

Гомель  
ГГУ им. Ф. Скорины  
2022

УДК 004.451.9R(076)

ББК 32.973.2я73

Г157

Рецензенты:

кандидат биологических наук В. С. Бирг;

кандидат биологических наук Ж. Е. Мелешко;

кандидат физико-математических наук В. В. Казаченок

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

**Галиновский, Н. Г.**

Г157

Введение в программирование на языке R : пособие / Н. Г. Галиновский ; М-во образования Республики Беларусь, Гомельский гос. ун-т им. Ф. Скорины. – Гомель : ГГУ им. Ф. Скорины, 2022. – 222 с.

ISBN 978-985-577-826-5

Пособие ставит своей целью оптимизировать учебно-познавательную деятельность магистрантов по усвоению материала курса «Введение в программирование на языке R». 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 978-985-577-826-5**

© Галиновский Н. Г., 2022

© Учреждение образования

«Гомельский государственный университет имени Франциска Скорины», 2022

## ОГЛАВЛЕНИЕ

Предисловие.....	4
Тема 1. Создание векторов.....	6
Тема 2. Операции с объектами в R.....	22
Тема 3. Создание графиков различного типа в R.....	55
Тема 4. Характеристика выборки и построение вариационного ряда. Сравнение выборочных средних в R .....	138
Тема 5. Проведение парного параметрического и непараметрического корреляционного анализа в R.....	164
Тема 6. Проведение регрессионного анализа в R (линейная регрессия).....	176
Тема 7. Дисперсионный анализ в R.....	185
Тема 8. Кластерный анализ в R.....	202
Предметный указатель функций R, упомянутых в пособии.....	220
Литература.....	222

## ПРЕДИСЛОВИЕ

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, характеристиках их разнообразия и его структуры, о влияниях разных экологических (и не только) факторов на живые организмы, развивающиеся в различных условиях.

Многие биологические вопросы, поставленные перед исследователем, не могут быть в должной мере решены без применения специальных математических методов. К таким вопросам относятся сравнение выборочных групп по изучаемым показателям и определение достоверности результатов такого сравнения с заданной вероятностью безошибочных прогнозов, оценка взаимосвязи признаков организмов, выявление количественных показателей этих взаимосвязей, выявление силы влияния различных факторов на биологические процессы и явления, а также классификация объектов по совокупности признаков.

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.

Большинство средств для осуществления статистической обработки данных, полученных в результате проведенных исследований из представленных на рынке, преимущественно платные и стоят серьезных денег. В то же время существует ряд программ совершенно бесплатных и обладающих достаточно высоким потенциалом. Это такие программы, как PAST или R. Последний является также средой для программирования со своим собственным языком. Данная особенность делает его особо гибким инструментом, а наличие огромного количества специальных готовых пакетов для анализа или графического отображения данных позволяет провести любой анализ. В связи с этим знание основ данного инструментария является обязательным условием для современного исследователя.

Язык программирования R построен по принципу свободного программного обеспечения, его использование абсолютно бесплатно и доступно всем желающим. Сама среда программирования



под операционной системой Windows доступна по адресу <https://cran.r-project.org/bin/windows/base/>. Для более комфортной работы в среде программирования разработчиками выпущена программная оболочка RStudio. Её также можно использовать абсолютно свободно, и она находится по адресу <https://rstudio.com/products/rstudio/download/>.

Как и в любом языке программирования, работа в R построена на использовании командной строки для написания кода. Однако это не значит, что в языке R нет графического интерфейса пользователя, так называемого GUI (Graphic User Interface). Он реализуется с помощью специальных пакетов. Сама среда представляет собой базовый набор пакетов, который позволяет сразу после установки воспользоваться возможностями статистической обработки данных. Существуют и дополнительные пакеты, значительно расширяющие функционал самой среды программирования. Они также находятся в свободном доступе и устанавливаются из самой среды программирования R, что очень удобно.

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 из восьми тем практических работ. 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 удобный для усвоения элементов математического анализа и облегчает знакомство с новыми программными продуктами у студентов и магистрантов, не обучающихся по IT-специальностям.

Автор благодарит А. Б. Шипунова, С. Э. Мастицкого и В. К. Шитикова, чьи книги открыли ему богатейший мир языка R, а также магистрантов биологического факультета Гомельского государственного университета имени Ф. Скорины, которые помогли в тестировании материала представленного пособия.

# 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).

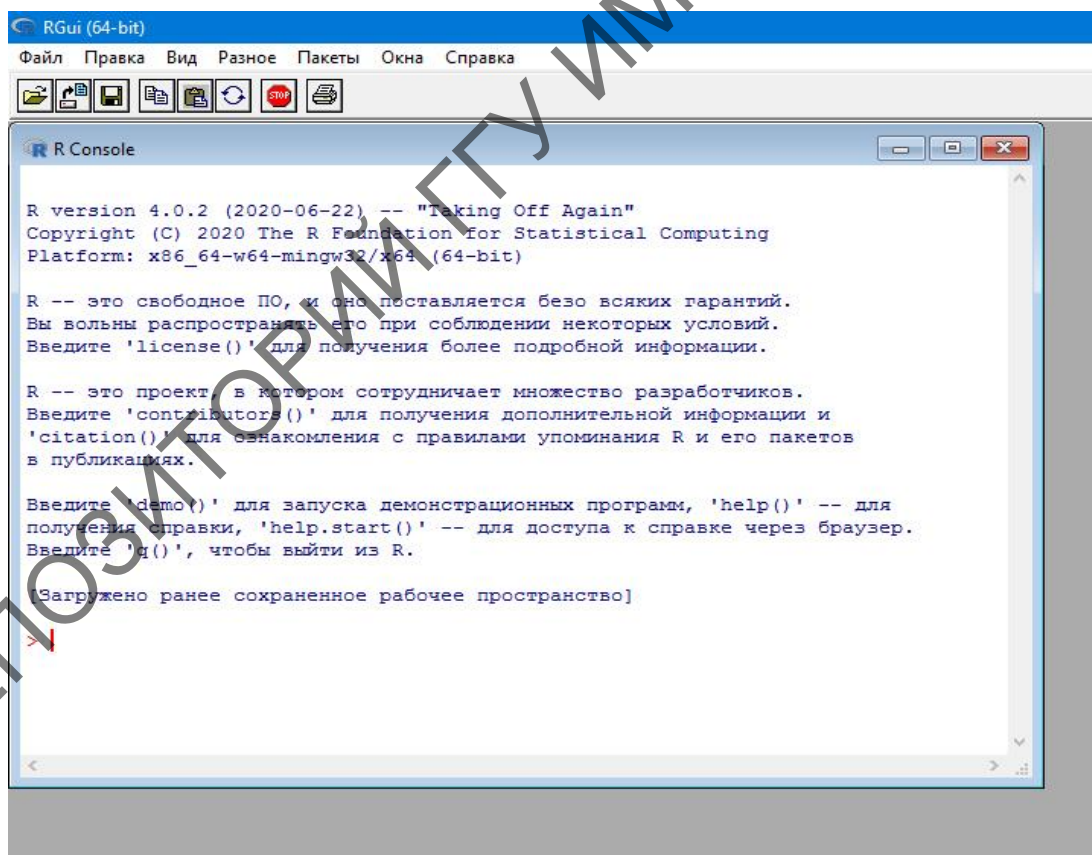


Рисунок 1 – Рабочее окно среды программирования R (консоль)

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 «RCom-mander». 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).

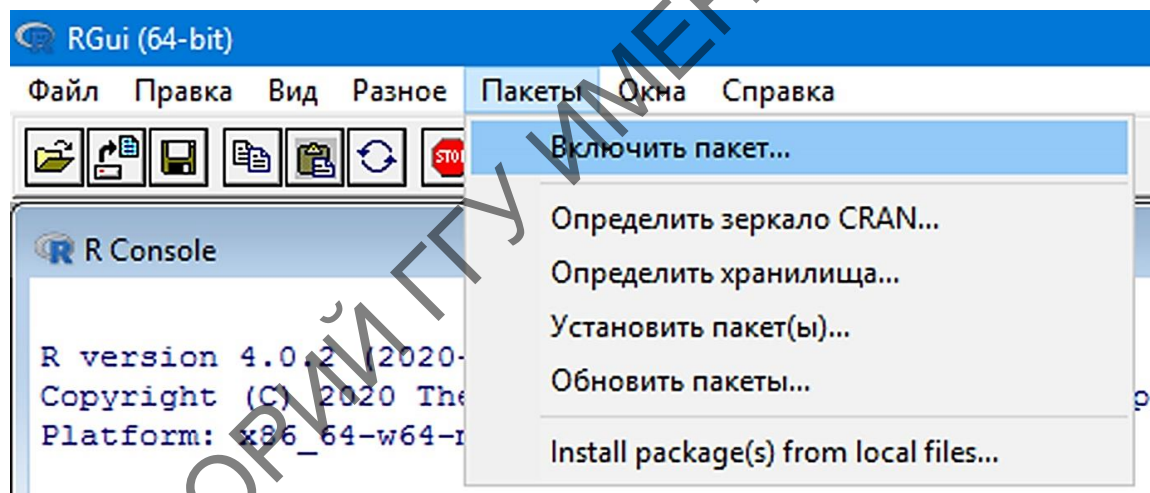


Рисунок 2 – Включение пакета через меню среды R

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 RCom-mander (рисунок 3).

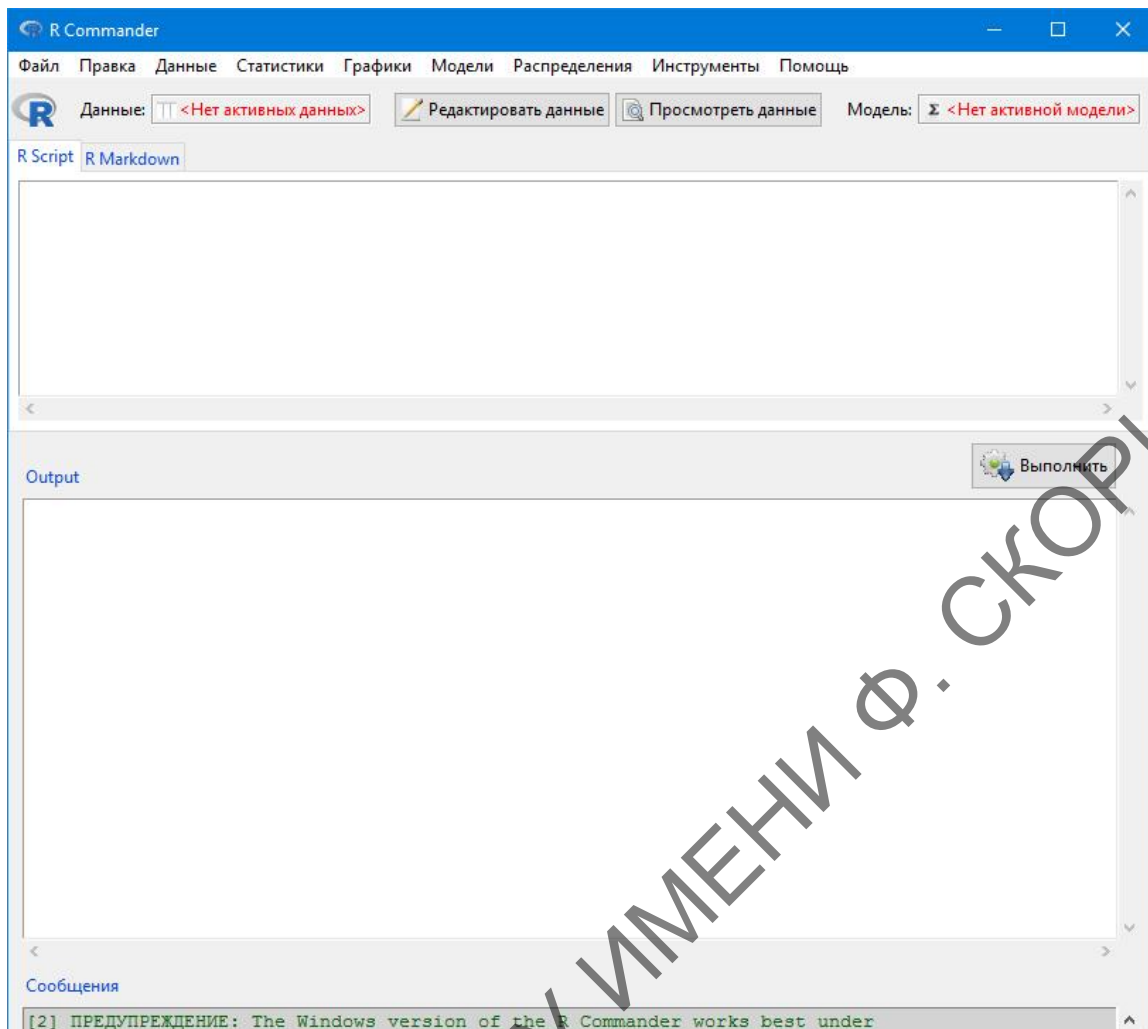


Рисунок 3 – Рабочее окно пакета RCommander

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, несмотря на то, что она достаточно удобна для начинающих пользова-

телей, 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 служит ставший 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. Единственным минусом программы (для русскоговорящих пользователей) можно считать её англоязычность, но у того, кто владеет английским на бытовом уровне, сложностей возникнуть не должно.

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.

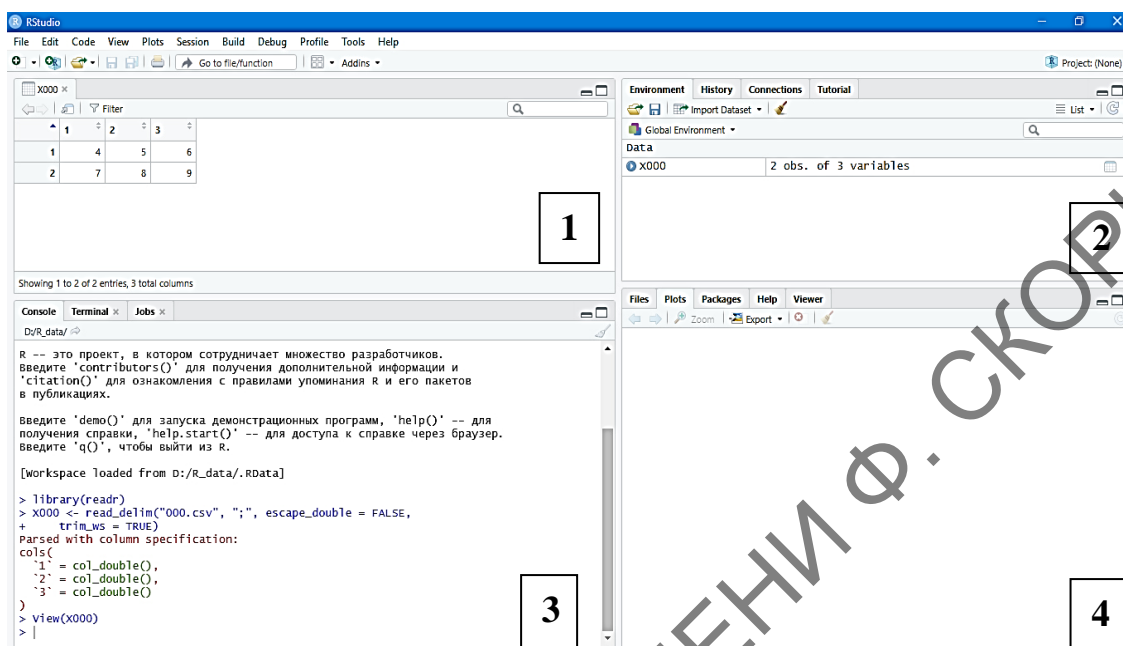


Рисунок 4 – Рабочее окно программного пакета RStudio

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).

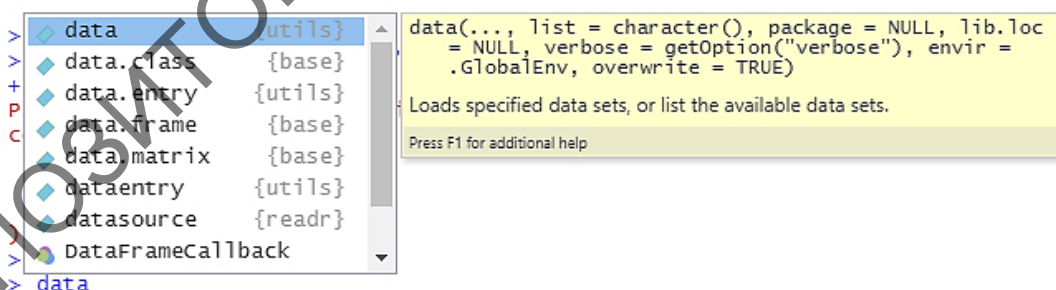


Рисунок 5 – Контекстная подсказка команд в программном пакете RStudio

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. 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("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).

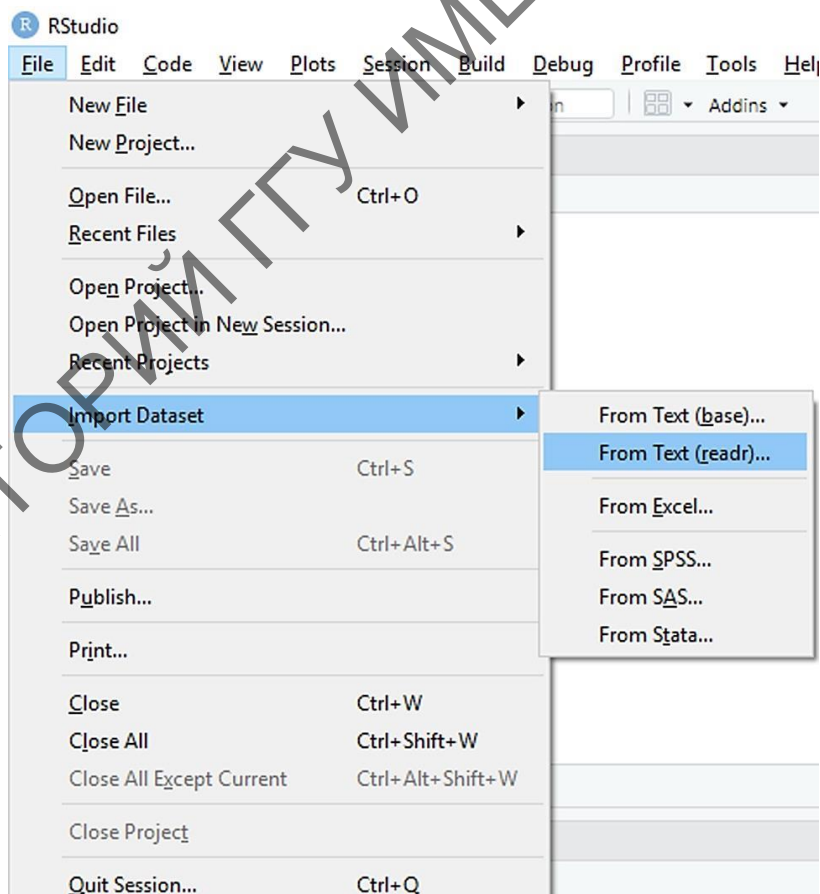


Рисунок 6 – Загрузка данных, используя меню RStudio

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

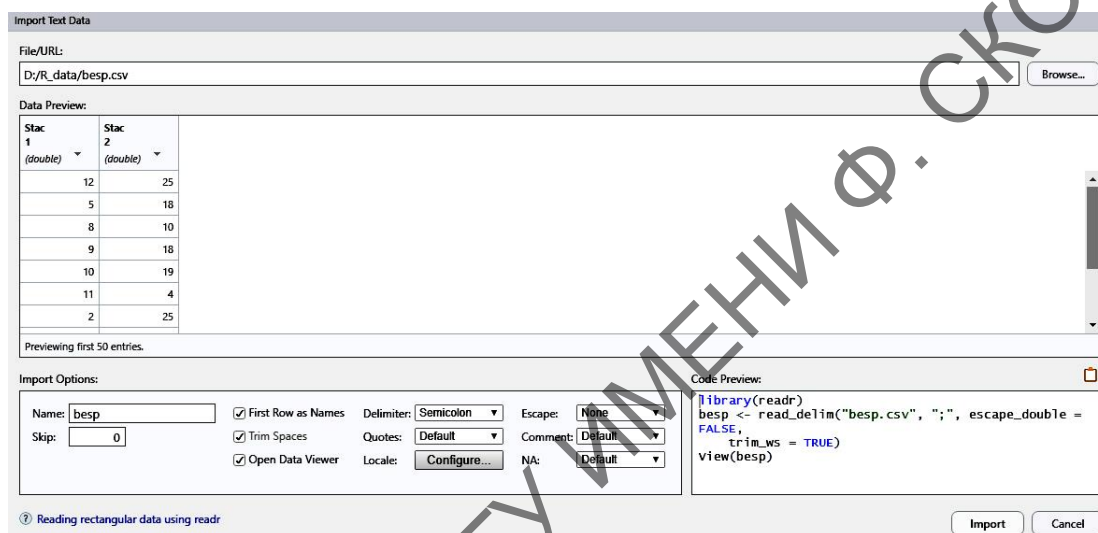


Рисунок 7 – Настройка окна импорта данных RStudio

**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 данных 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).

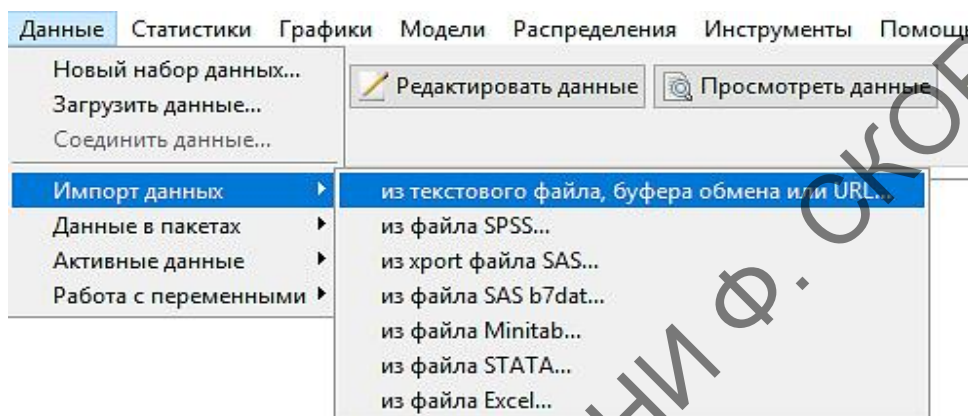


Рисунок 9 – Загрузка данных, используя меню RCommander

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

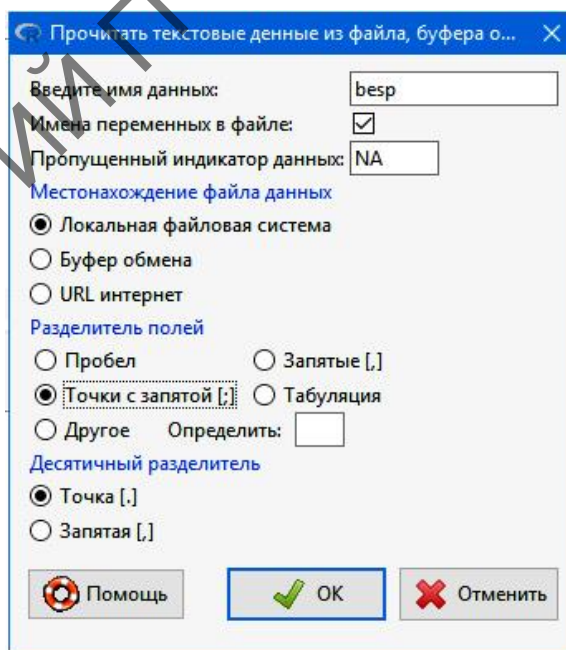
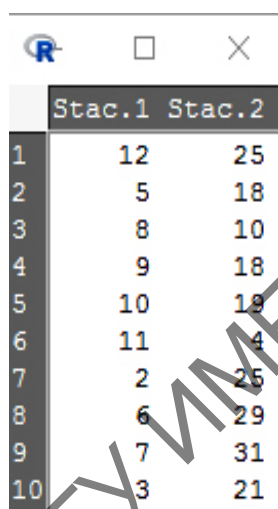


Рисунок 10 – Опции интеграции текстового файла с разделителем в RCommander

**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 этого 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	18
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 программирования R основаны в первую очередь на векторах. Вектор представляет собой объект, содержащий определенную последовательность значений одного типа. В том случае, если в процессе обработки данных возникает необходимость использования в коде скаляров, то их можно заменить единичными векторами (векторами, содержащими только одно значение).

Все элементы вектора могут принадлежать к какому-то одному из типов данных, принятых в языке программирования R (подробнее о типах данных – в теме 2). В то же время вектор является объектом языка программирования 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 программирования 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).

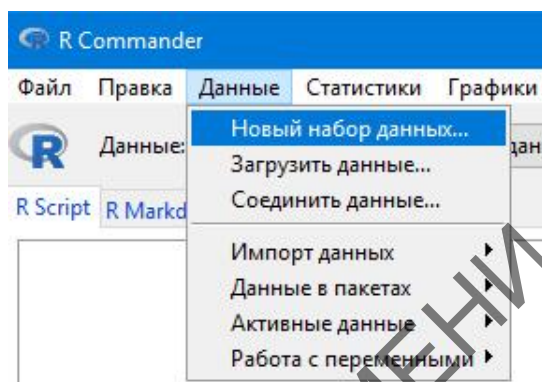


Рисунок 12 – Пункт меню о наборе новых данных в RCommander

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

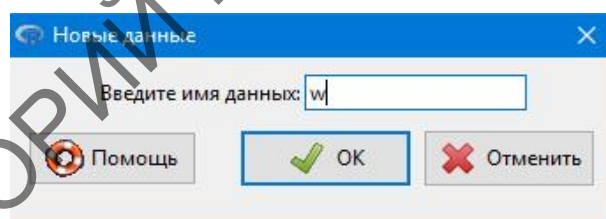


Рисунок 13 – Диалоговое окно с присвоением имени новому вектору в RCommander

**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**. 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.

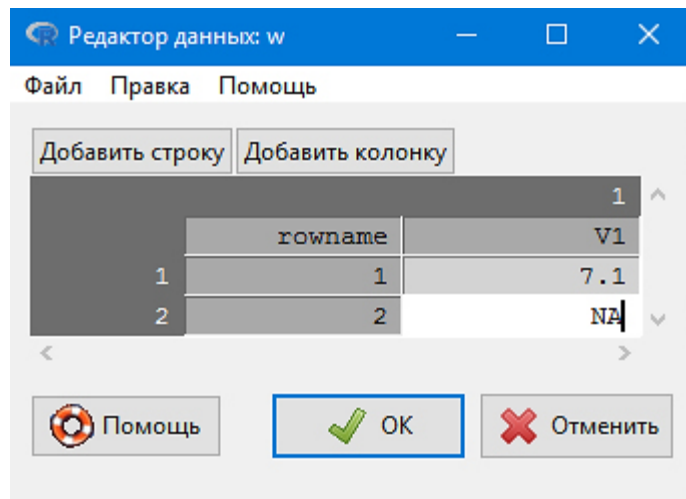


Рисунок 14 – Окно ввода новых данных в RCommander

**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](http://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 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, при этом нужно помнить, что совокупность элементов символьного вектора не является единой строкой);
- **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 вектор  $a$  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).

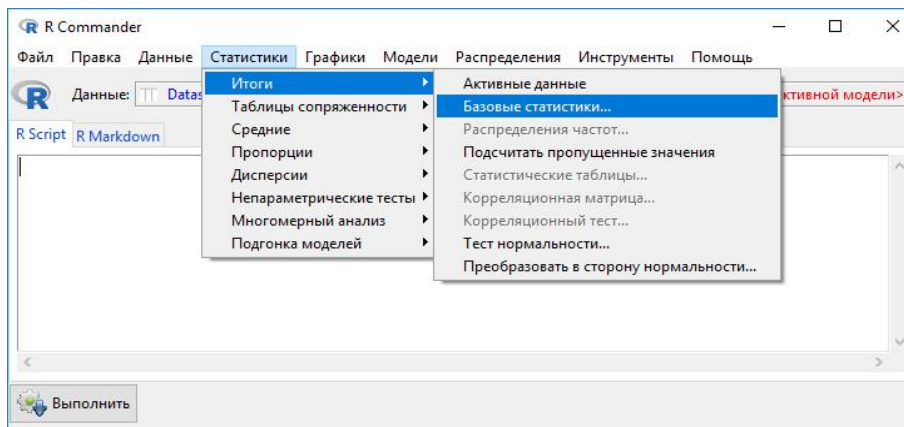


Рисунок 16 – Переход в пункт меню **Базовые статистики...** в R Commander

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

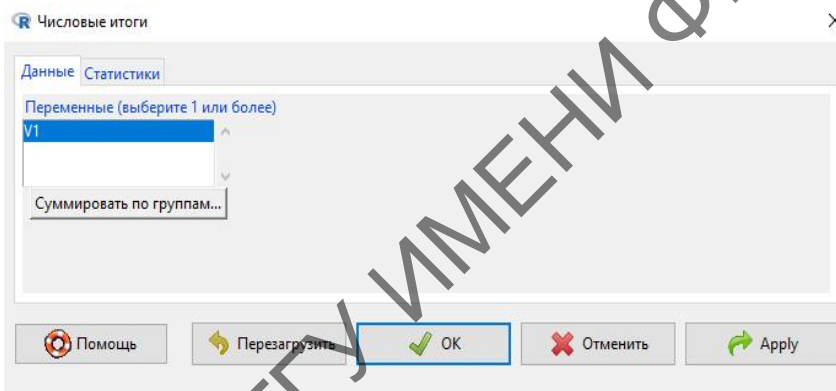


Рисунок 17 – Диалоговое окно **Числовые итоги**

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

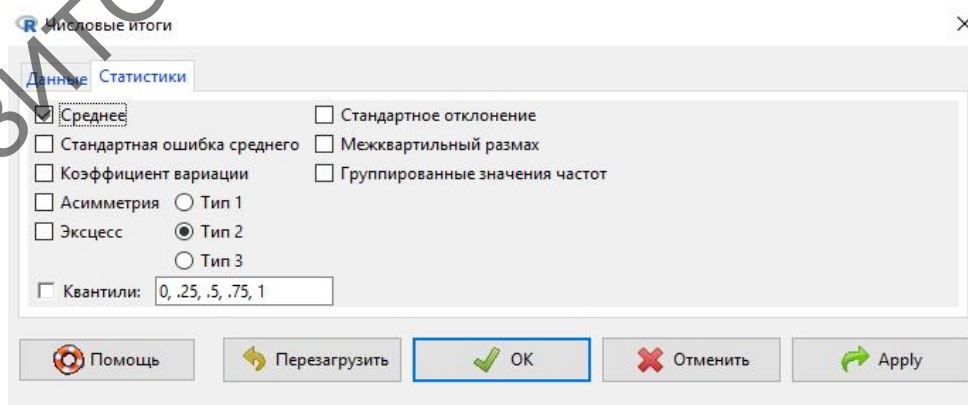


Рисунок 18 – Выделение параметра **Среднее** в закладке **Статистики** диалогового окна **Числовые итоги**

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 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, иг-

ровое поле игры «Морской бой», рабочий лист любой электронной таблицы – это матрица.

В языке программирования R матрицы могут быть разной размерности, даже трехмерные (но для удобства в учебных целях мы рассмотрим только двумерные) и 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 даже 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 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 программирования R позволяют хранить в одной переменной объекты как одного, так и разных типов (в том чис-



ле 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 помощи его 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 переменные 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 программирования 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.

Несмотря на то, что в языке программирования R уже есть тип символьных данных и может показаться, что использование факторов лишнее, данные объекты позволяют эффективно использовать категориальные и порядковые переменные. В отличие от векторов с символьными данными, эти объекты имеют два атрибута: `"levels"` и `"class"`. Атрибут `"class"` содержит название объекта, т. е. фактор (*factor*), а `"levels"` – уровни фактора (все уникальные значения переменной).

Ниже рассмотрим пример работы с факторами. Для начала создадим символьный вектор, содержащий обозначения двух цветов – пять элементов белого цвета *«white»* и шесть 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. чтобы 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 для сравнения, 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	Мышь1	9,3	female	Б2
Мышь2	12,1	male	Б3	Мышь3	10,5	female	Б2
Мышь4	6,7	female	Б1	Мышь2	11,1	male	Б3
Мышь1	8,7	male	Б2	Мышь4	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" "fe-  
male" "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. Делается это 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)
```



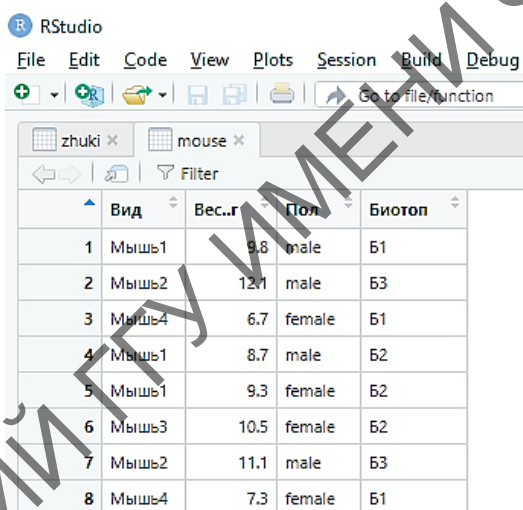
## Шаг 6. Посмотрим на нашу получившуюся таблицу:

```
> 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).

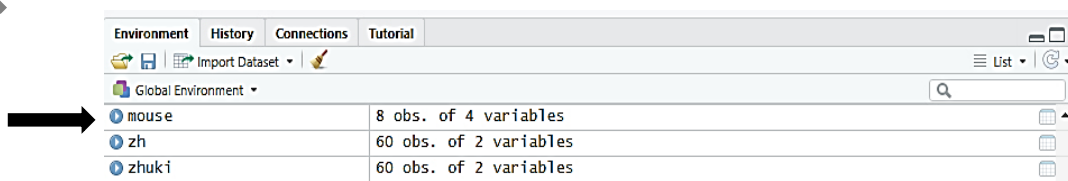


The screenshot shows the RStudio interface with the 'Environment' pane open. The 'mouse' data frame is selected, and its contents are displayed in a table view. The table has four columns: 'Вид', 'Вес..г', 'Пол', and 'Биотоп', and eight rows of data.

	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

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



The screenshot shows the RStudio Environment pane. The 'Global Environment' is expanded, and the 'mouse' data frame is highlighted with a black arrow. The table shows 8 observations of 4 variables.

Object	Class	Attributes
mouse	data.frame	8 obs. of 4 variables
zh	data.frame	60 obs. of 2 variables
zhuki	data.frame	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 среды R и то, что Вы делаете в одной программе, отображается и в другой.

**Шаг 1.** Загружаем среду программирования 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).

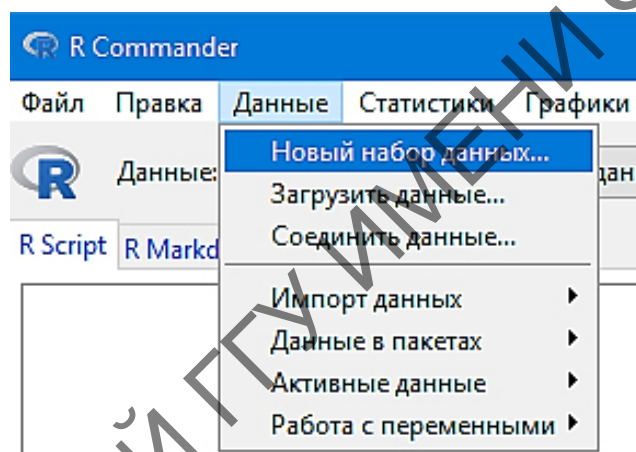


Рисунок 21 – Пункт меню о наборе новых данных в RCommander

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

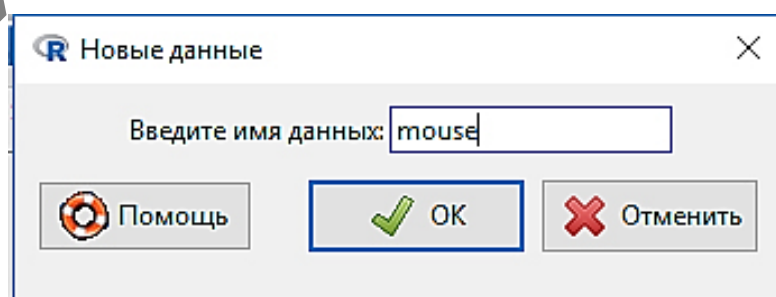


Рисунок 22 – Диалоговое окно с присвоением имени новой таблице данных в RCommander

**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).

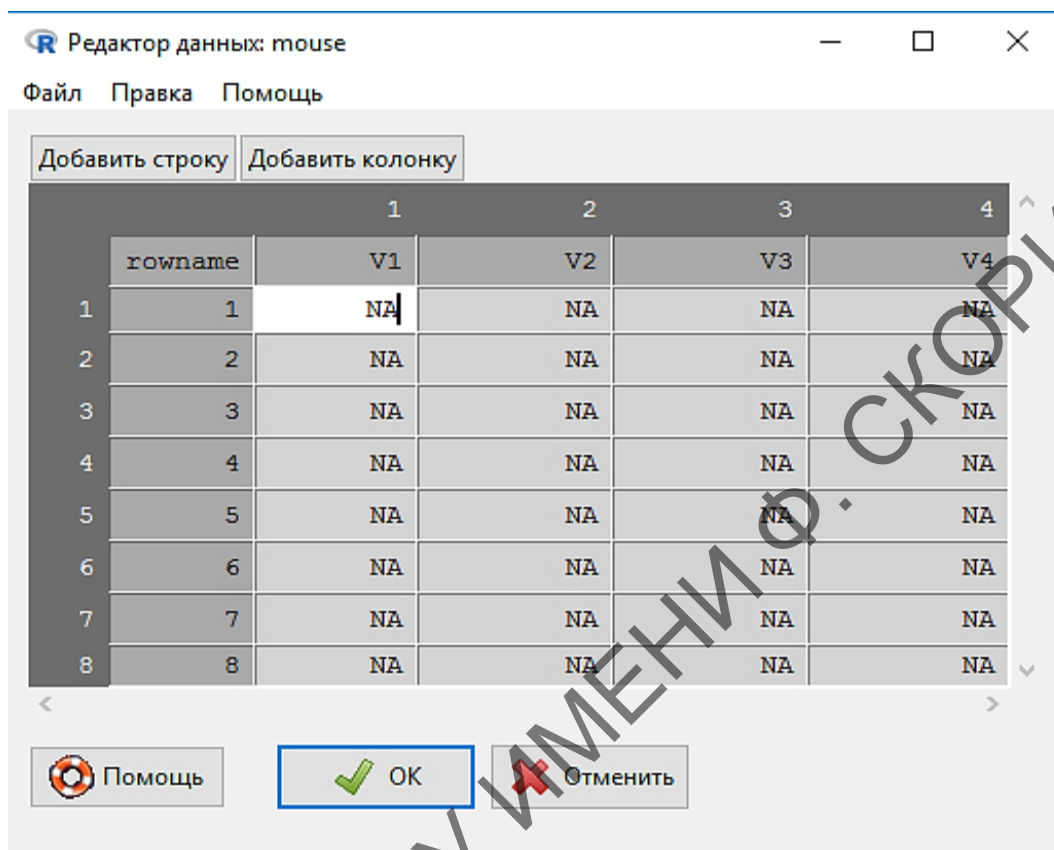


Рисунок 23 – Окно ввода новых данных в RCommander с макетом будущей таблицы

**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. Завершенная таблица будет иметь вид, показанный на рисунке 24. После чего нажмите на кнопку **OK**. Наша таблица готова и загружена в компьютер, а в командной строке среды программирования R будет об этом приведено сообщение, переданное из RCommander:

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

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

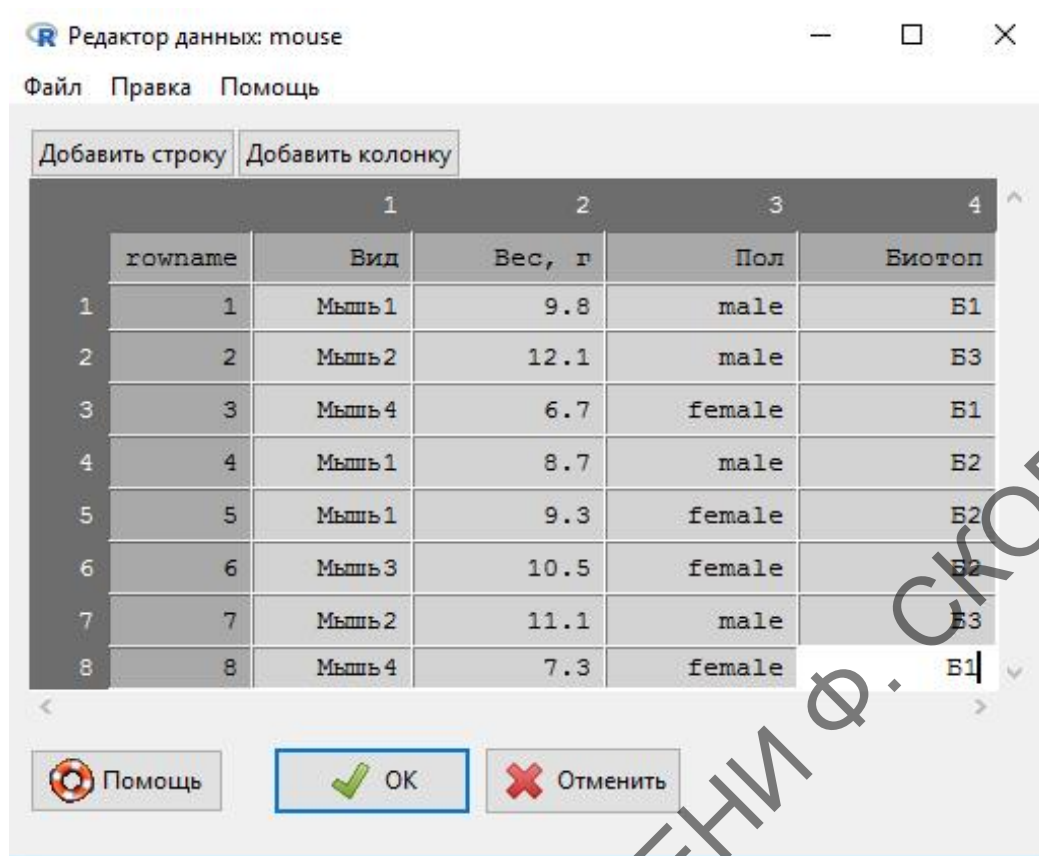


Рисунок 24 – Готовая таблица, созданная в RCommander

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).

## 2.4.3 Индексация данных в таблице, коррекция и сохранение данных

### 2.4.3.1 Индексация данных

В связи с тем, что таблица данных по сути является списком, то к ней также применимы ранее нами рассмотренные методы индексации списков. Датафреймы индексируются как двумерные матрицы. Рассмотрим это на примере про мышей. Допустим, нам нужно для работы 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 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, в нашем случае 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 просмотреть отдельно столбец не получится – при нажатии кнопки **Просмотреть данные** загружается таблица целиком (рисунок 25).

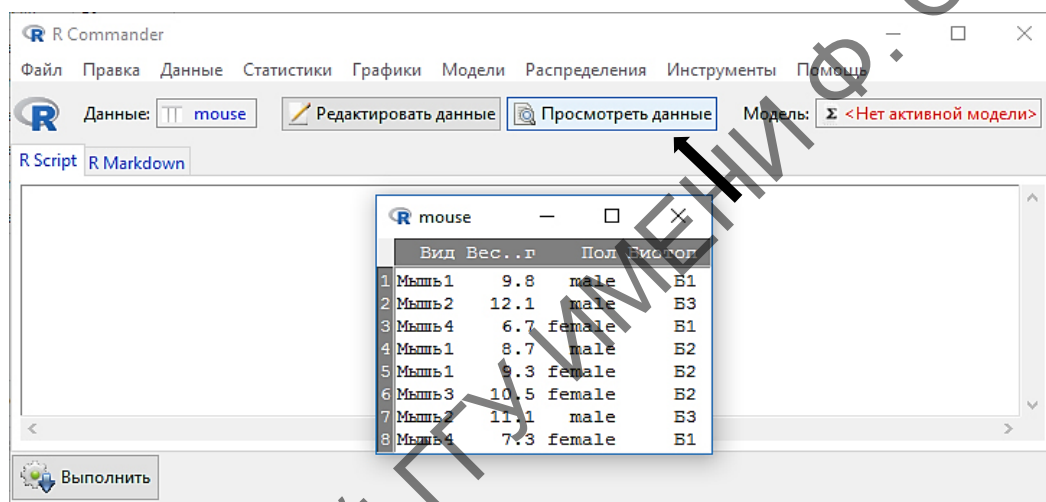


Рисунок 25 – Просмотр таблицы с данными в RCommander

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 просмотреть 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).

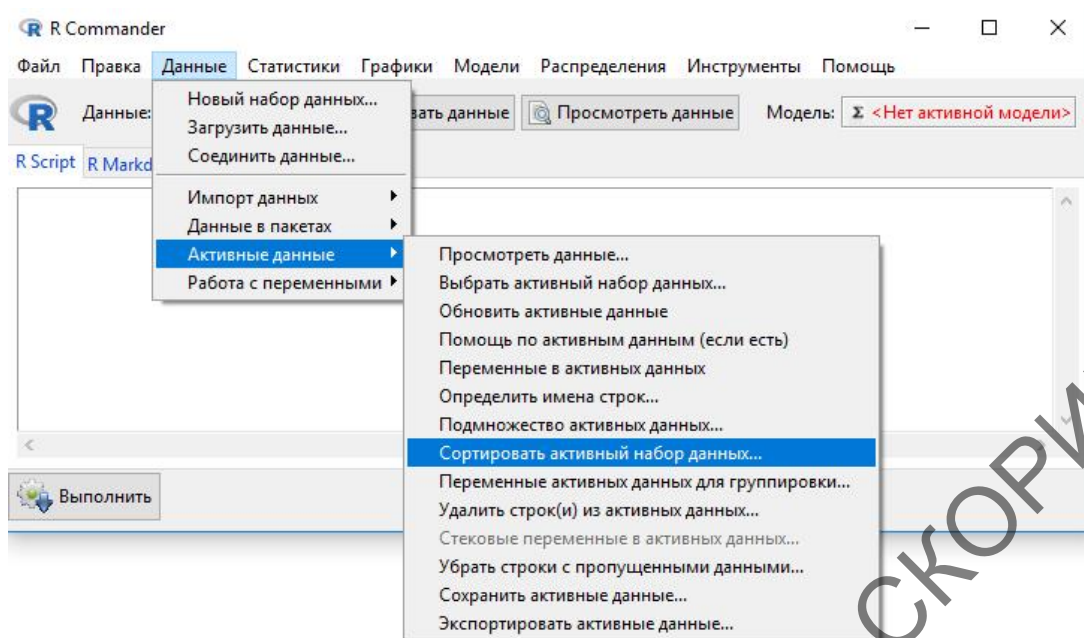


Рисунок 26 – Меню активных данных в R Commander

**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**.

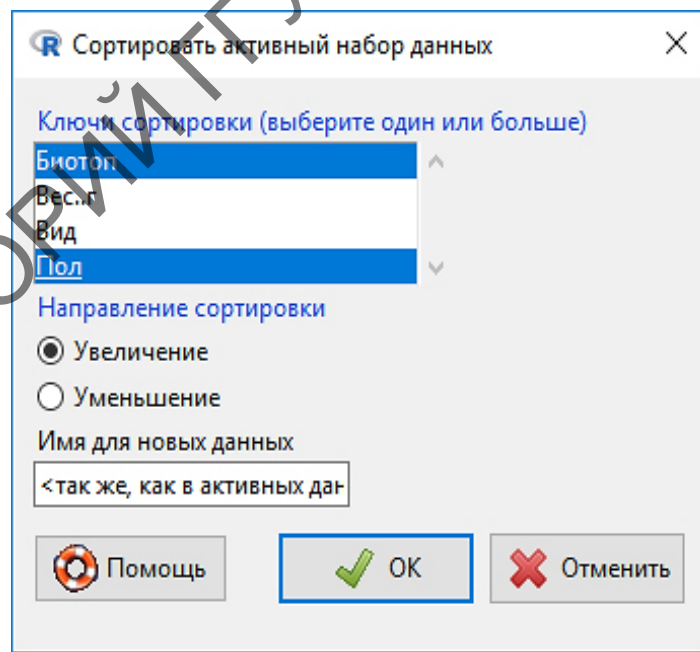


Рисунок 27 – Диалоговое окно **Сортировать активный набор данных...** в R Commander



**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**.

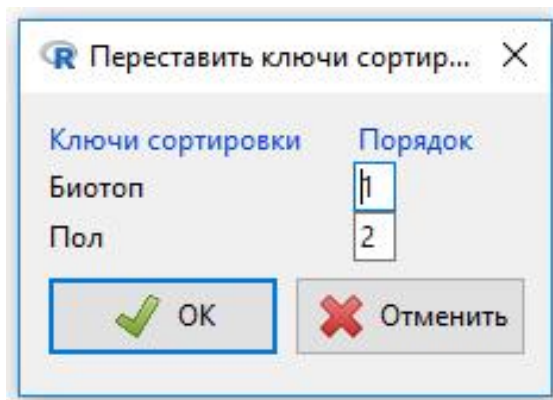
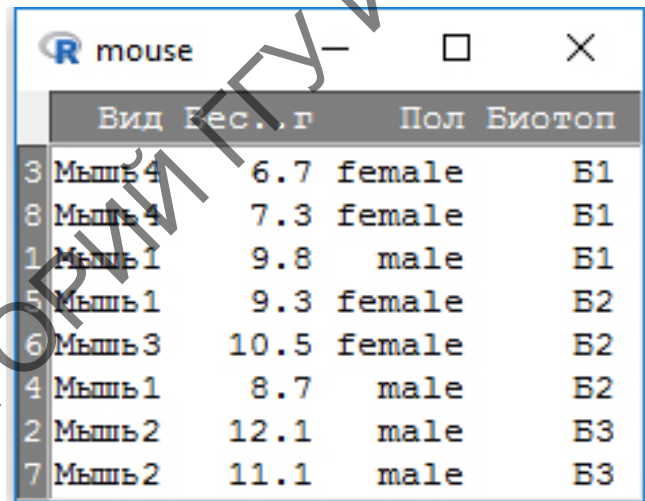


Рисунок 28 – Диалоговое окно **Переставить ключи сортировки...** в RCommander

**Шаг 4.** Таблица отсортирована. Убедиться в этом можем, нажав кнопку **Просмотреть данные**. Полученная таблица отображена 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

### 2.4.3.2 Коррекция данных в таблице

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



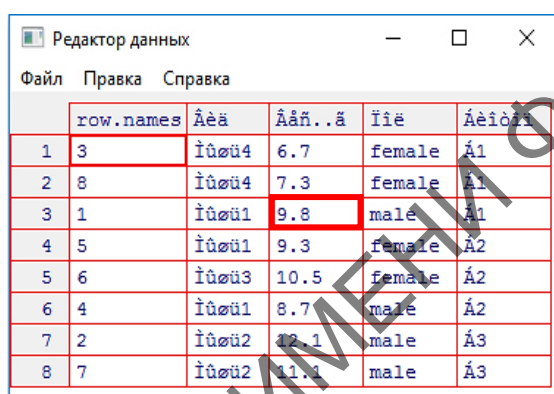
## 1 В RStudio:

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, используя 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	Б1
2	8	Мышь4	7.3	female	Б1
3	1	Мышь1	9.8	male	Б1
4	5	Мышь1	9.3	female	Б2
5	6	Мышь3	10.5	female	Б2
6	4	Мышь1	8.7	male	Б2
7	2	Мышь2	12.1	male	Б3
8	7	Мышь2	11.1	male	Б3

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. 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.

### 2.4.3.3 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).

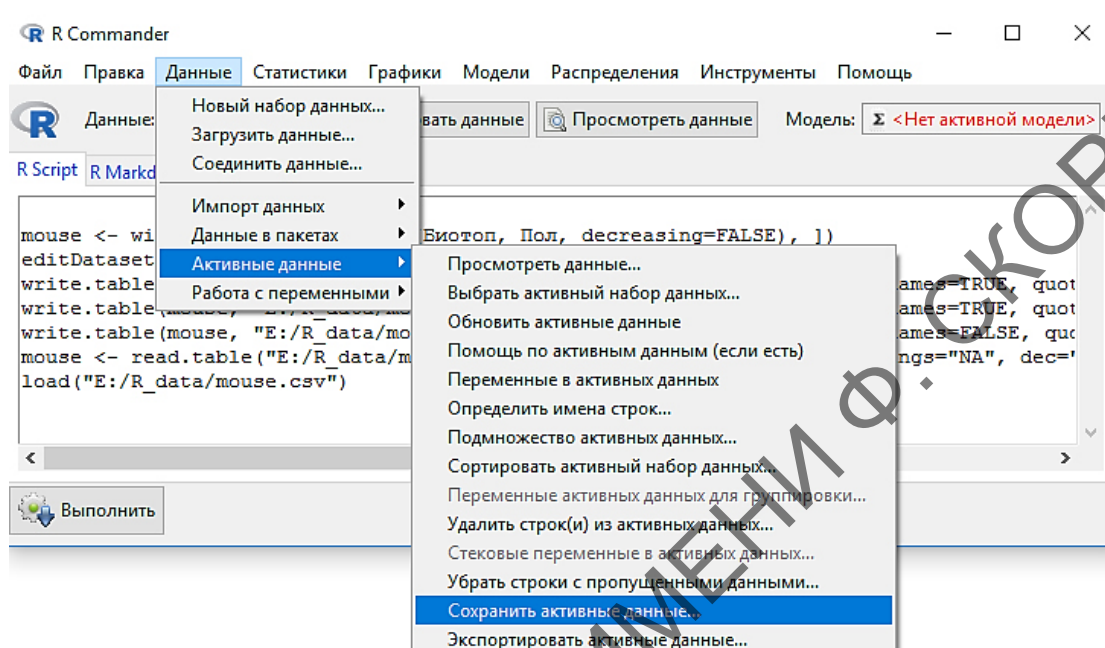


Рисунок 31 – Сохранение данных в RCommander в формате \*.RData

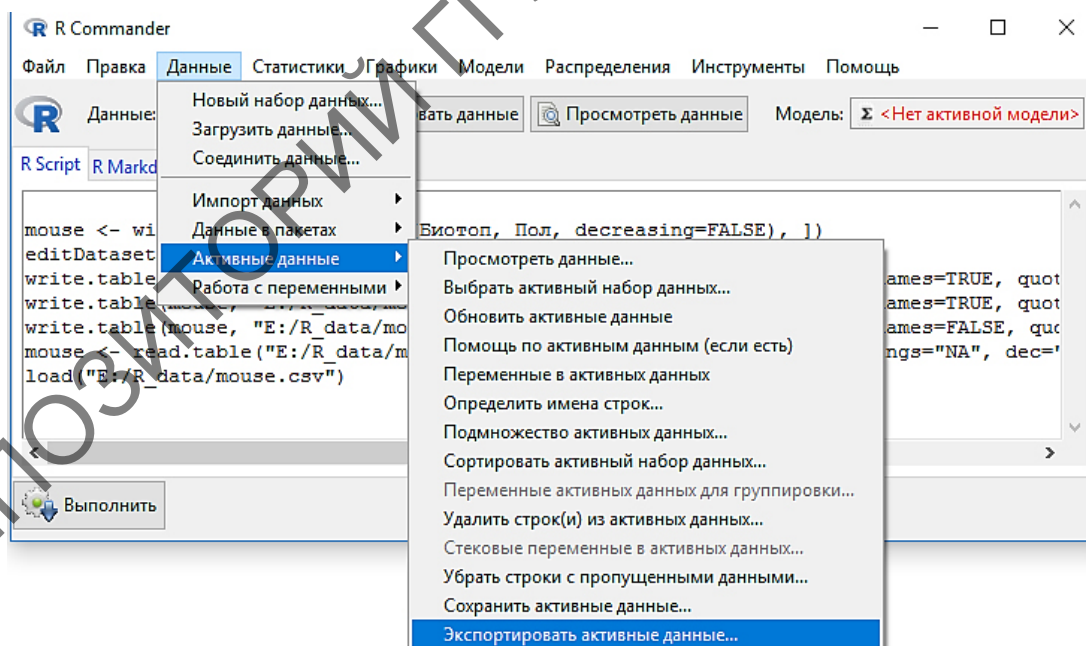


Рисунок 32 – Экспорт данных в RCommander в текстовый формат

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**.

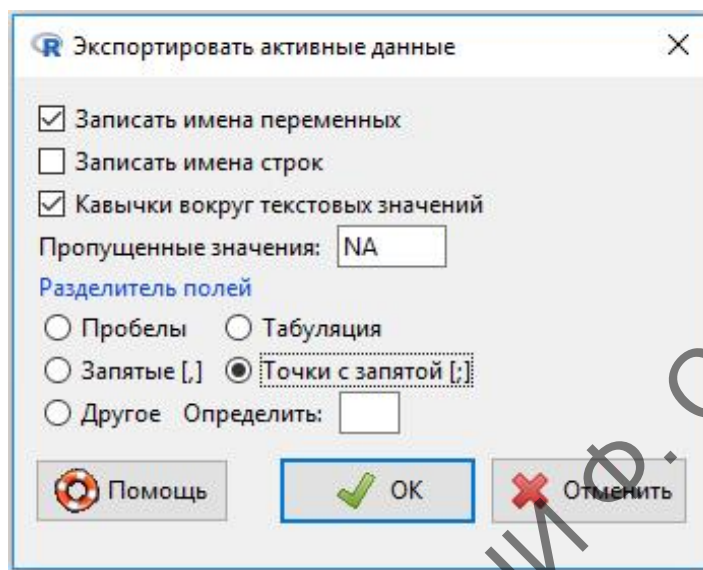


Рисунок 33 – Диалоговое окно **Экспорт активных данных** в R Commander

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 файлами с промежуточными данными папку самой программы. Поэтому логично сменить папку, например, R\_data (данные R), на другом диске, 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 её создать, если ранее не была создана).

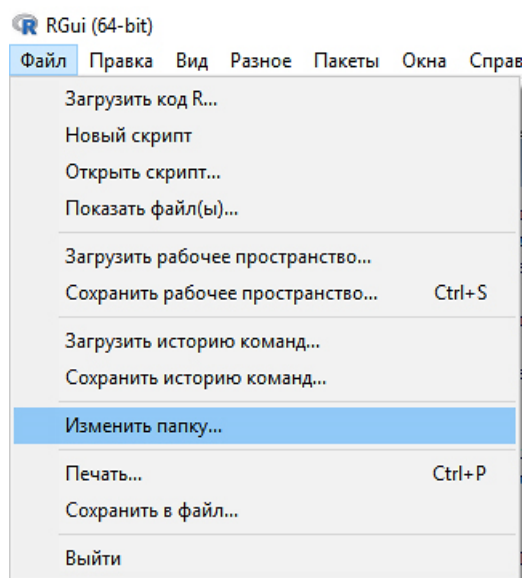


Рисунок 34 – Меню **Изменить папку...** в R

### 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).

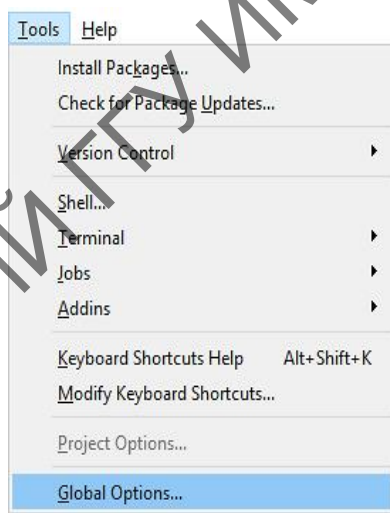


Рисунок 35 – Меню **Global options...** в RStudio

После этого мы попадаем в само диалоговое окно общих настроек, и там 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).



### 3-й способ (при помощи RCommander)

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).

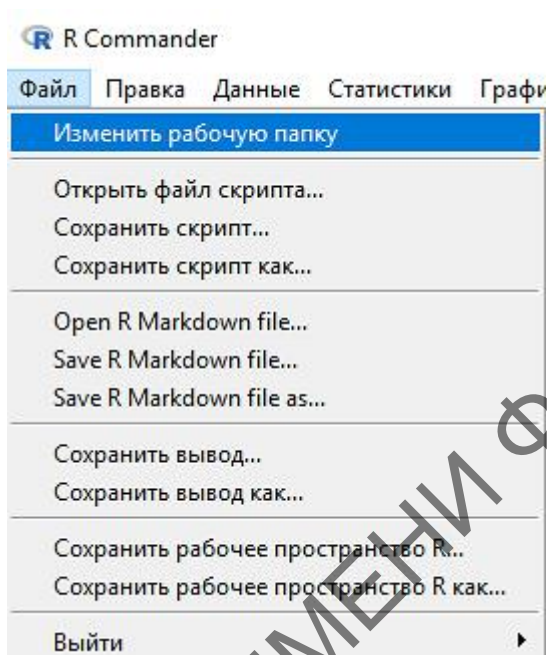


Рисунок 36 – Меню **Изменить рабочую папку** в RCommander

## 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) Имёются 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

«excellent» – 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](http://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.

## ТЕМА 3. СОЗДАНИЕ ГРАФИКОВ РАЗЛИЧНОГО ТИПА В R

- 1 Способы графического отображения результатов в R.
- 2 Управление графическими устройствами в R.

### 1 Способы графического отображения результатов в R

Любой программный пакет, который осуществляет статистический анализ, подразумевает под собой наличие визуализации полученных результатов. Человек физиологически в первую очередь получает информацию именно от органов зрения, поэтому наглядность представления результатов играет немаловажную роль при статистической обработке. Язык программирования R в этом отношении не стоит в стороне и также предоставляет возможность графически отражать полученные в результате анализа выводы.

Графическая информация в языке программирования R реализована в виде отдельных пакетов. Для очень быстрой оценки данных можно использовать функции построения графиков в базе самого R. Они устанавливаются по умолчанию вместе с R в виде базового набора и не требуют установки каких-либо дополнительных пакетов. Функции быстро набираются, просты в использовании в несложных случаях и работают очень быстро. Однако, если вы хотите сделать что-либо, кроме очень простых графиков, обычно лучше переключиться на пакет **ggplot2**. Отчасти это связано с тем, что графический пакет **ggplot2** представляет расширенный набор опций.

В данном пособии мы рассмотрим основные виды графического отображения информации, реализованные в базовом пакете языка программирования R, а также в пакете **ggplot2**. В ряде случаев, как например, при проведении корреляционного анализа, могут быть использованы и другие графические пакеты. Для более подробной информации о графических возможностях языка программирования R рекомендуем книгу Уинстона Чанга «Кулинарная книга графики в R», которая доступна для бесплатного просмотра по адресу <https://r-graphics.org/index.html>.

Для того чтобы скачать и включить дополнительный пакет **ggplot2**, необходимо:

*1-й способ: в RStudio (при помощи командной строки и GUI)*

Набрать сначала команду инсталляции пакета:

```
> install.packages("tidyverse")  
> install.packages("ggplot2")
```

А затем, дождавшись окончания установки, включить пакет, набрав команду:

```
> library(ggplot2)
```

Если не хотите использовать для этого командную строку, то в том же RStudio перейдите в меню **Tools** → **Install Packages** (рисунок 37), а затем в появившемся окне **Install Packages** в боксе **Packages** необходимо набрать название нужного пакета, в данном случае – сначала **tidyverse**, а затем **ggplot2** и нажать кнопку **Install** (рисунок 38).

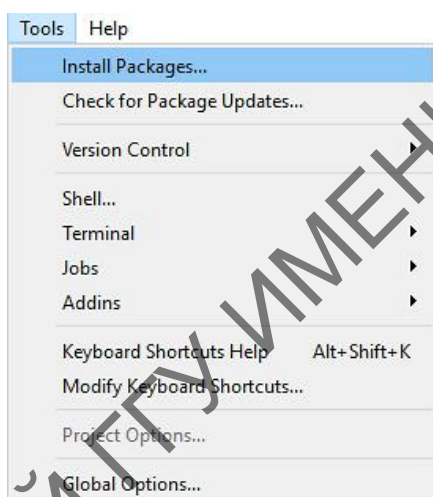


Рисунок 37 – Пункт меню **Install Packages** в RStudio

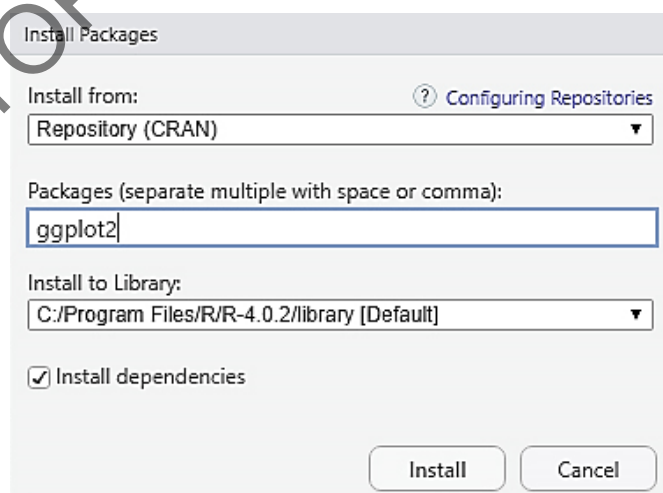


Рисунок 38 – Диалоговое окно **Install Packages** в RStudio

2-й способ: в среде программирования R (при помощи GUI)

Операции с командной строкой такие же, как и в выше рассмотренном примере для RStudio. Что же касается графического интерфейса, то в меню нужно перейти по пути **Пакеты** → **Установить пакет(ы)...** (рисунок 39).

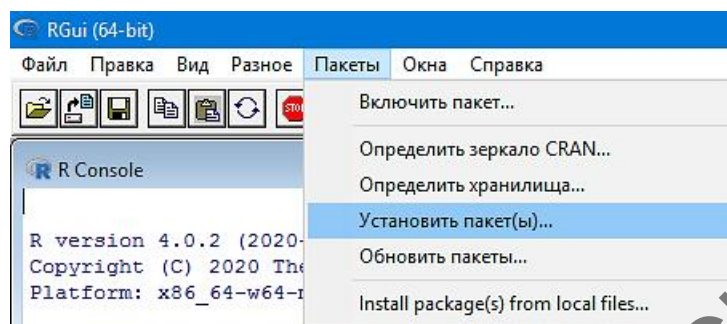


Рисунок 39 – Пункт меню **Установить пакеты** в среде R

После чего в появившемся диалоговом окне **Secure CRAN mirrors** (Безопасные зеркала CRAN) выбрать место расположения зеркала, с которого собираетесь скачивать пакет. Самый простой путь – указать зеркало «Russia (Moscow)», но можно выбрать любой другой, на конечный результат это не повлияет. После выбора зеркала в появившемся диалоговом окне **Packages** нужно выбрать наш пакет, т. е. **ggplot2** (рисунок 40), нажать **ОК** и дождаться окончания установки.

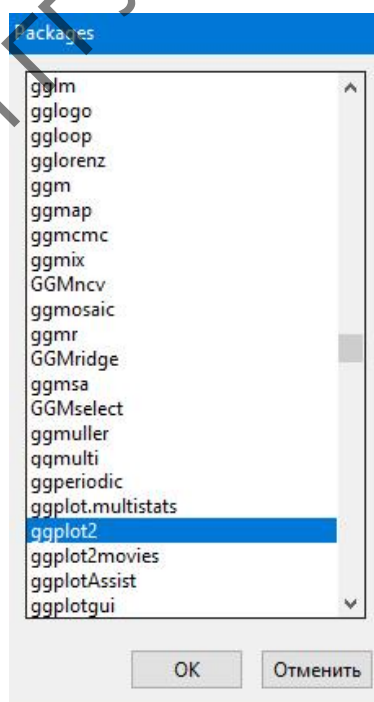


Рисунок 40 – Диалоговое окно **Packages** в R

## 1.1 Диаграмма рассеивания

Диаграмма рассеивания, которая также называется ещё коррелограммой (**scatterplot**), является одним из основных типов диаграмм, используемых в графическом отображении результатов анализа, чаще всего – в корреляционном.

Рассмотрим построение такого типа диаграммы как с использованием командной строки, так и при помощи GUI на ранее рассмотренном примере о соотношении веса мелких млекопитающих и веса потребляемого ими корма (тема 1; таблица 2).

### 1.1.1 Построение диаграммы рассеивания в RStudio (при помощи командной строки и базового пакета)

**Шаг 1.** Создадим вектор  $w$  (вес микромаммалий):

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

**Шаг 2.** Создадим вектор  $f$  (вес потреблённой ими пищи):

```
> f <- c(2.3, 2.3, 4.1, 5.3, 4.4, 5.9, 5.7, 6.2, 6.5, 9.3)
```

**Шаг 3.** Создадим таблицу данных (датафрейм)  $vmm$ , объединив эти два вектора:

```
> vmm <- data.frame("weight_mm"=w, "weight_food"=f)
```

**Шаг 4.** Проверим созданный датафрейм:

```
> vmm
  weight_mm weight_food
1      7.1         2.3
2      7.7         2.3
3      3.6         4.1
4      8.3         5.3
5      8.8         4.4
6     10.4         5.9
7      8.9         5.7
8      9.0         6.2
9      8.9         6.5
10     14.0         9.3
```

Чтобы построить диаграмму рассеивания, используется функция `plot()`, которой в качестве аргумента передаются названия переменных по оси  $x$  и оси  $y$ . В нашем случае по оси  $x$  откладываем значения веса микромаммалий (т. е. `Weight_mm`), а по оси  $y$  – вес потреблённого ими корма (т. е. `Weight_food`).

```
> plot(vmm$weight_mm, vmm$weight_food)
```

На вкладке **Plots** в правом нижнем окне программы появится полученная диаграмма (рисунок 41).

Обратите внимание, что здесь использована индексация, т. е. при указании аргументов по оси  $x$  и оси  $y$  упоминалось, что они части датафрейма *vmm*. Это необходимо делать, если таблица содержит более двух переменных.

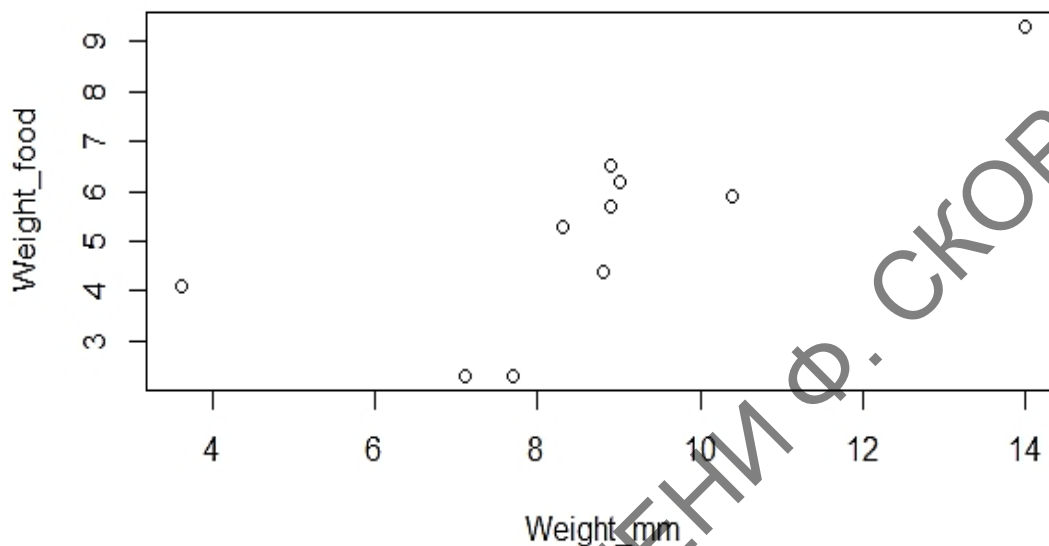


Рисунок 41 – Диаграмма рассеивания соотношения веса микромаммалий и веса потреблённого ими корма

В том случае, если в таблице только две переменные и нужно узнать только их соотношение между собой, можно использовать сокращённый вид этой функции, где в качестве аргумента упоминается только название датафрейма, а программа автоматически рассматривает первый столбец таблицы как аргумент  $x$ , а второй – как аргумент  $y$ . В нашем случае:

```
> plot(vmm)
```

Результат будет тот же.

Есть ещё один способ, чтобы сократить написание аргументов функции `plot()`. Для этого первоначально таблицу нужно «прикрепить» командой `attach()`, где аргумент – название датафрейма. После выполнения этой команды столбцы как бы разъединяются и могут быть проанализированы по отдельности, без необходимости их индексации. В нашем примере:

```
> attach(vmm) # Прикрепляем датафрейм
> plot(weight_mm, weight_food) # Строим диаграмму
> detach(vmm) # Открепляем датафрейм
```

После всех нужных нам операций со столбцами датафрейма, чтобы не запутаться, лучше сразу открепить их обратно командой `detach()`. Прикрепление и открепление в первую очередь используется при работе с таблицами, где количество столбцов больше двух, так как это позволяет значительно сэкономить время.

Однако полученная кореллограмма не вполне информативна, так как не имеет линии средних квадратов. Поэтому добавим её:

```
> abline(lm(f ~ w), col='red')
```

Теперь наша диаграмма рассеивания имеет законченный вид (рисунок 42). В аргументах функции `abline()` указана линейная модель связи переменной  $f$  с переменной  $w$  – `lm(f ~ w)`, а также цвет линии (в нашем случае – красный).

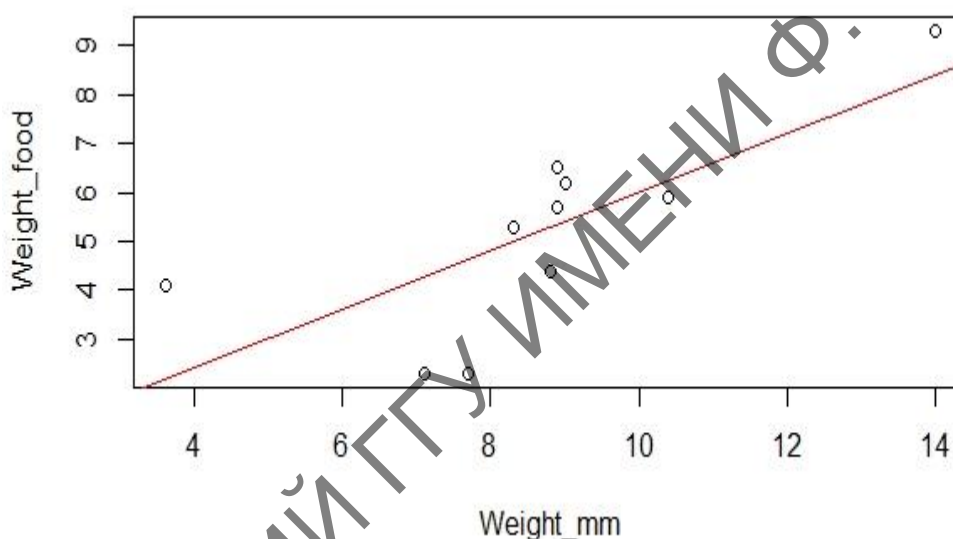


Рисунок 42 – Диаграмма рассеивания с добавленной линией средних квадратов

Сохранить полученную диаграмму также несложно. Графическую информацию можно сохранять как в графическом формате растрового рисунка, например, в `jpg`, так и в формате `pdf` (portable document format) от компании Adobe.

Для того чтобы сохранить рисунок в графическом формате, необходимо зайти в меню по пути, отображённом на рисунке 43, **Plots** → **Save as Image...** (при сохранении в формате `pdf` необходимо выбрать опцию **Save as PDF...**).



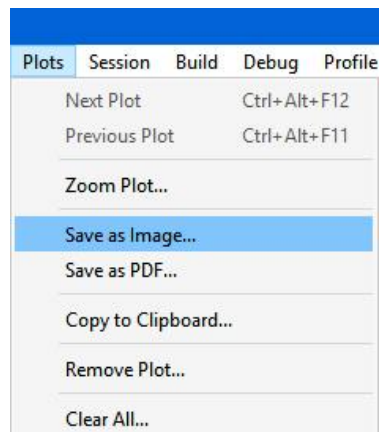


Рисунок 43 – Пункт меню **Save as Image...** в RStudio

Затем в диалоговом окне **Save Plot as Image** (Сохранить диаграмму как рисунок) выбрать формат сохраняемого файла в боксе **Image format** (Формат рисунка), затем строкой ниже указать папку, куда хотите сохранить файл (по умолчанию указана рабочая папка R) и, наконец, в третьей строке в боксе **File name** (Имя файла) необходимо указать имя сохраняемого файла с диаграммой или графиком. Кроме этого, нажав кнопку **Update Preview** (Предпросмотр), можно убедиться в том, что вы сохраняете именно то, что нужно (рисунок 44) и нажать кнопку **Save** (Сохранить).

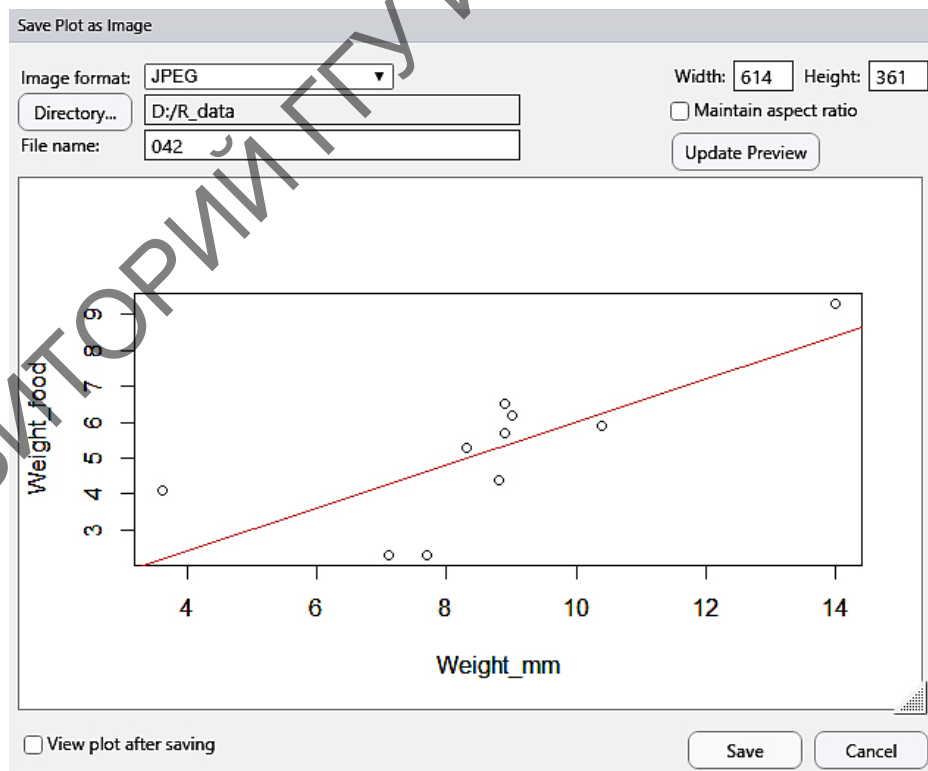


Рисунок 44 – Диалоговое окно **Save Plot as Image...** в RStudio

При сохранении рисунка в формате pdf после выбора пункта меню **Save as PDF...** появляется диалоговое окно **Save Plot as PDF** (Сохранить диаграмму как документ pdf). Оно отличается от рассмотренного нами выше окна сохранения рисунка и имеет свои особенности (рисунок 45).

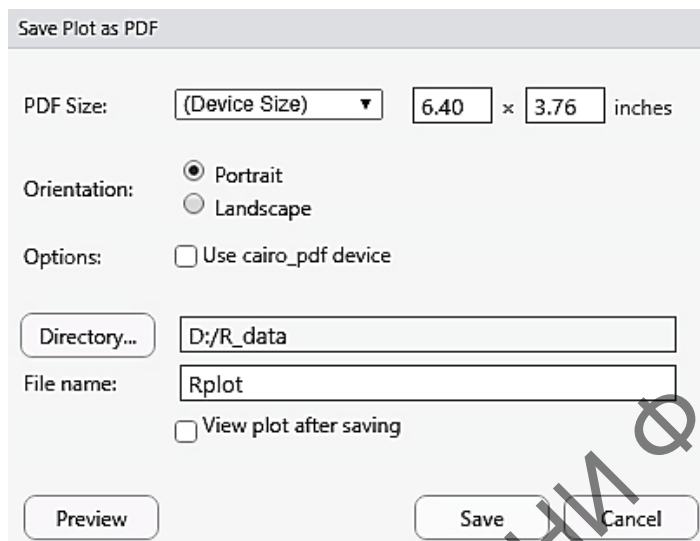


Рисунок 45 – Диалоговое окно **Save Plot as PDF...** в RStudio

Для начала в боксе **PDF size** (Размер документа pdf) нужно указать размеры рисунка в дюймах (или выбрать в выпадающем списке уже готовый размер). После чего в поле **Orientation** (Ориентация) выбрать либо вертикальную (Portrait), либо горизонтальную (Landscape) ориентацию листа с рисунком (рисунок 45). Для уточнения можете использовать кнопку **Preview** (Предпросмотр). Далее необходимо указать папку для размещения сохраняемого файла, присвоить ему название и нажать на кнопку **Save** (Сохранить).

### 1.1.2 Построение диаграммы рассеивания в RStudio (при помощи командной строки и пакета ggplot2)

Для примера создания коррелограммы в пакете **ggplot2** используем те же данные по микромаммалиям, которые были взяты для примера построения диаграммы рассеивания в базовом пакете графики. При использовании этого пакета используется функция `ggplot()` и дополнительная функция размера точки на диаграмме `geom_point()`. В качестве аргументов функции `ggplot()` используются: название датафрейма, а также указываются переменные для осей *x* и *y* в том порядке, как они упоминаются в датафрейме. Аргу-

ментами функции является размер точки (по умолчанию, т. е. без указания конкретного размера – это единица). Строим диаграмму с размером точек по умолчанию:

```
> ggplot(vmm, aes(x = weight_mm, y = weight_food))+  
  geom_point()
```

Отобразится диаграмма, представленная на рисунке 46.

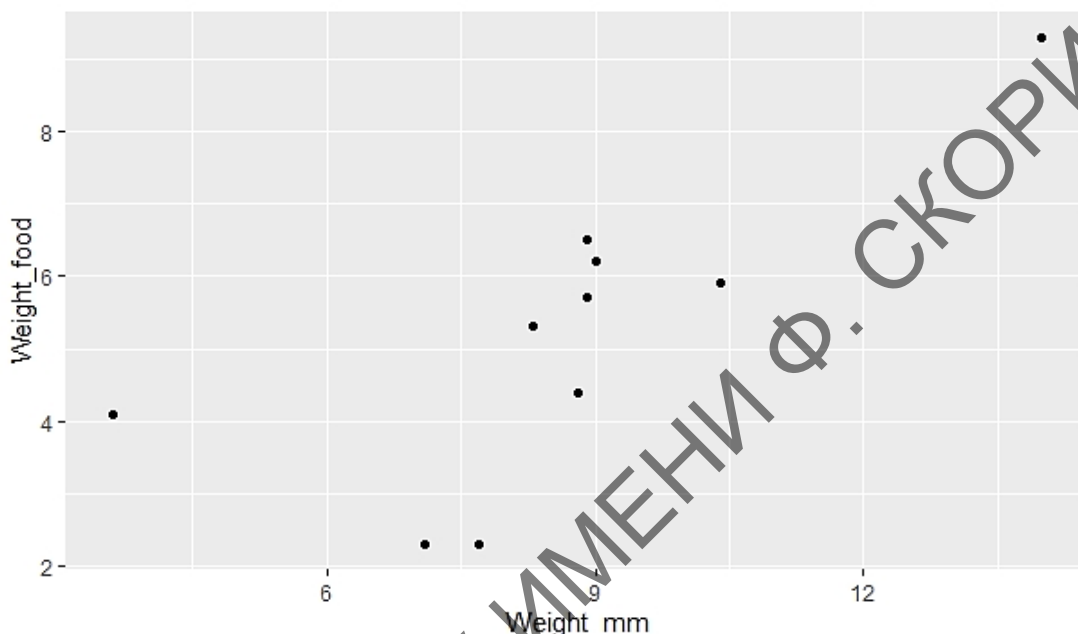


Рисунок 46 – Диаграмма рассеивания, реализованная при помощи графического пакета **ggplot2**

Обычный способ использования функции `ggplot()` – это передать ей датафрейм (в данном случае – `vmm`), а затем сообщить, какие столбцы из него использовать для значений  $x$  и  $y$ . Если необходимо передать ей только два вектора (при датафрейме, где столбцов более двух) для значений  $x$  и  $y$ , то нужно вместо названия датафрейма использовать опцию `data = NULL`, а затем передать ей векторы с полной индексацией (то есть с указанием названия датафрейма через значок «\$» – название вектора). Например, в нашем случае:

```
> ggplot(data = NULL, aes(x = vmm$w, y = vmm$f))+  
  geom_point()
```

Для того чтобы диаграмма имела законченный вид и статистическую ценность, необходимо добавить линию тренда, а также область, куда будут попадать 95 % данных (то есть соответствовать уровню значимости  $p = 0,05$ ). Точки на диаграмме сделаем более крупными, например, 2.5.

**Шаг 1.** Создаем диаграмму с размером точек в 2.5.

```
> ggplot(vmm, aes(x = weight_mm, y = weight_food))+  
  geom_point(size = 2.5)
```

**Шаг 2.** Вводим промежуточную переменную *hw*, присваиваем ей значение нашей диаграммы:

```
> hw <- ggplot(vmm, aes(x = weight_mm, y = weight_food))
```

**Шаг 3.** Используем нашу переменную, ставим знак «+» в конце строки, тем самым говоря программе, что команда не закончена, нажимаем **Enter**, а затем ниже указываем размер точки, ставим знак «+» в конце строки и снова нажимаем **Enter**:

```
> hw+  
  geom_point(size = 2.5)+
```

**Шаг 4.** Еще ниже строим линию наименьших квадратов с областью данных в 95%. Для этого используется функция `stat_smooth()`, где в качестве аргумента указывается метод – линейная модель (*lm* – line model) и уровень значимости – 0,95:

```
stat_smooth(method = lm, level = 0.95)
```

В результате создается коррелограмма, показанная на рисунке 47.

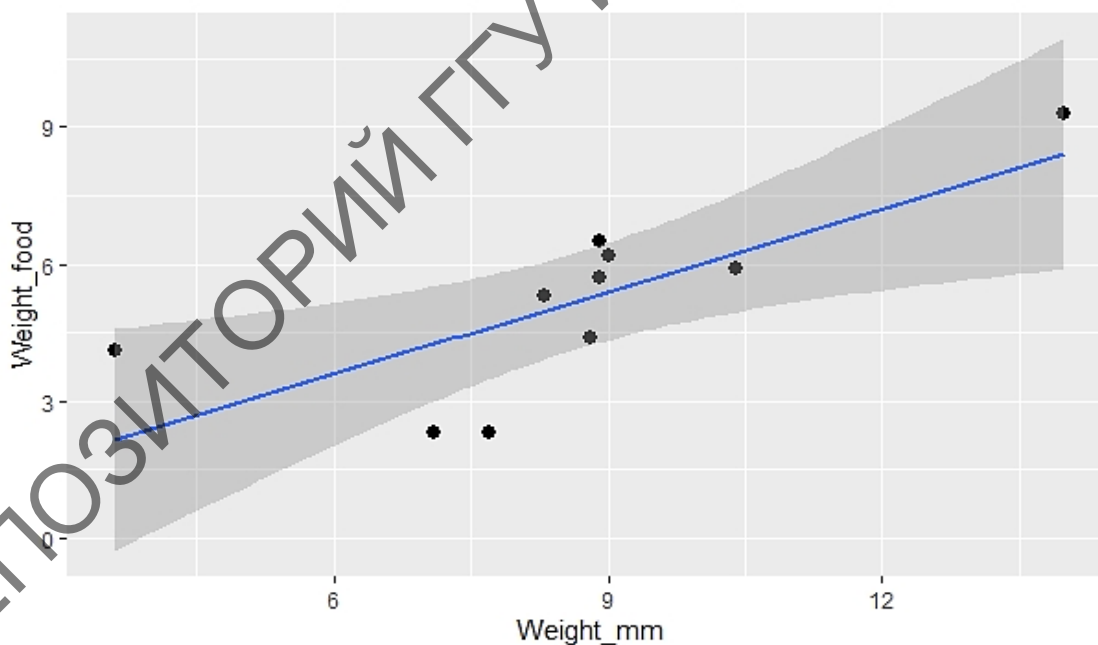


Рисунок 47 – Диаграмма рассеивания, реализованная при помощи графического пакета **ggplot2** с линией тренда и областью точности 0,95

При этом нужно иметь в виду, что пакет **ggplot2** предназначен для работы с датафреймами в качестве источника данных, а не с отдельными векторами, и что его использование подобным образом позволит вам использовать только ограниченную часть его возможностей.

Сохраняется диаграмма так же, как и в предыдущем примере.

### 1.1.3 Построение диаграммы рассеивания в RCommander (при помощи GUI)

Для построения диаграммы рассеивания в RCommander используем те же данные по весу микромаммалей и весу потреблённой ими пищи.

**Шаг 1.** Создаём таблицу данных (как это делается, см. подраздел 2.4.2 темы 2), используя данные по весу микромаммалей и пищи из подраздела 1.1.1 текущей темы.

**Шаг 2.** Переходим в меню создания точечного графика по пути **Графики** → **Точечный график...** (рисунок 48).

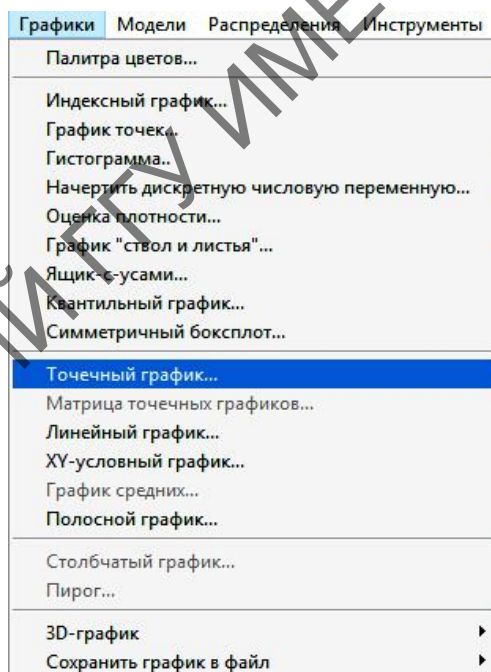


Рисунок 48 – Пункт меню **Точечный график...** в RCommander

**Шаг 3.** В появившемся диалоговом окне **Точечный график** необходимо перейти на закладку **Данные** и указать переменную для оси *x* и оси *y* (рисунок 49).

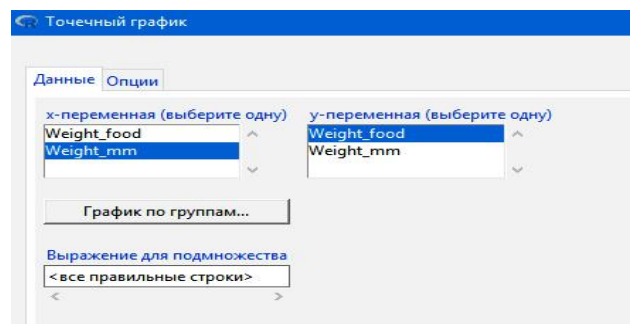


Рисунок 49 – Закладка **Данные** в диалоговом окне **Точечный график**

**Шаг 4.** В этом же диалоговом окне **Точечный график** необходимо перейти на закладку **Опции** и указать настройки графика. В нашем случае необходимо отметить боксы **Линия наименьших квадратов** и **Показать разброс** (рисунок 50). Размер точки при помощи ползунка укажем 1.5, после чего необходимо нажать на кнопку **ОК**. Полученная диаграмма (рисунок 51) будет показана в окне сред программирования R.

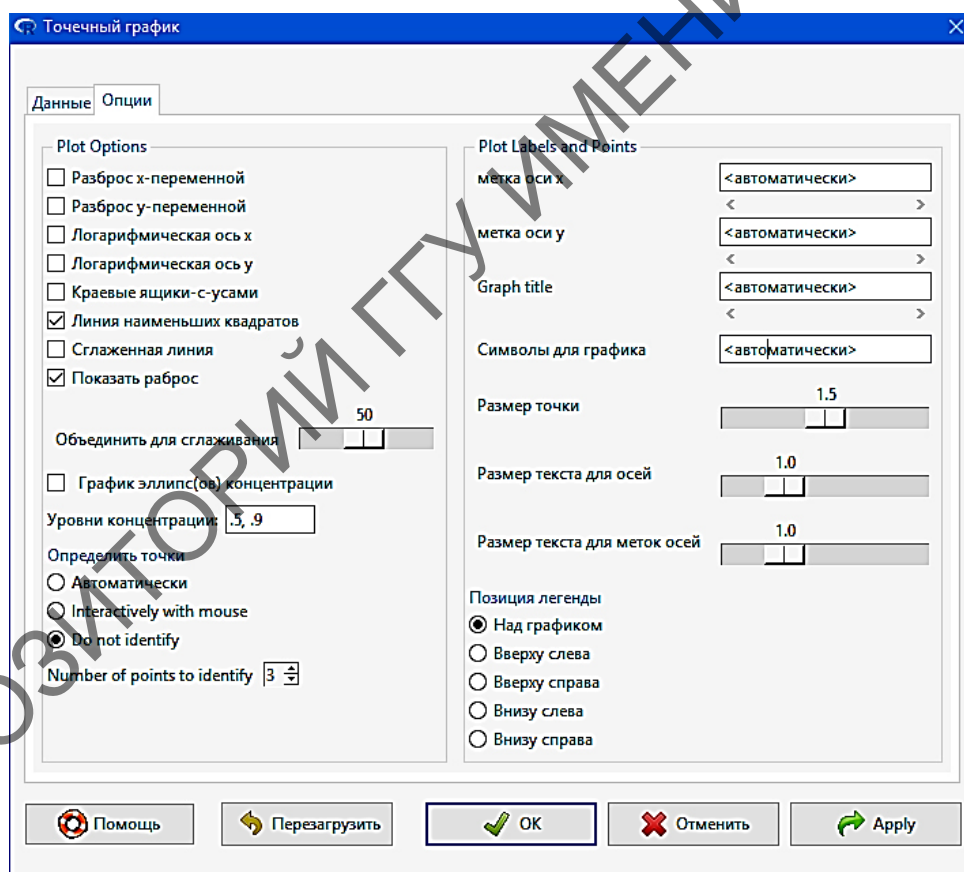


Рисунок 50 – Закладка **Данные** в диалоговом окне **Точечный график**

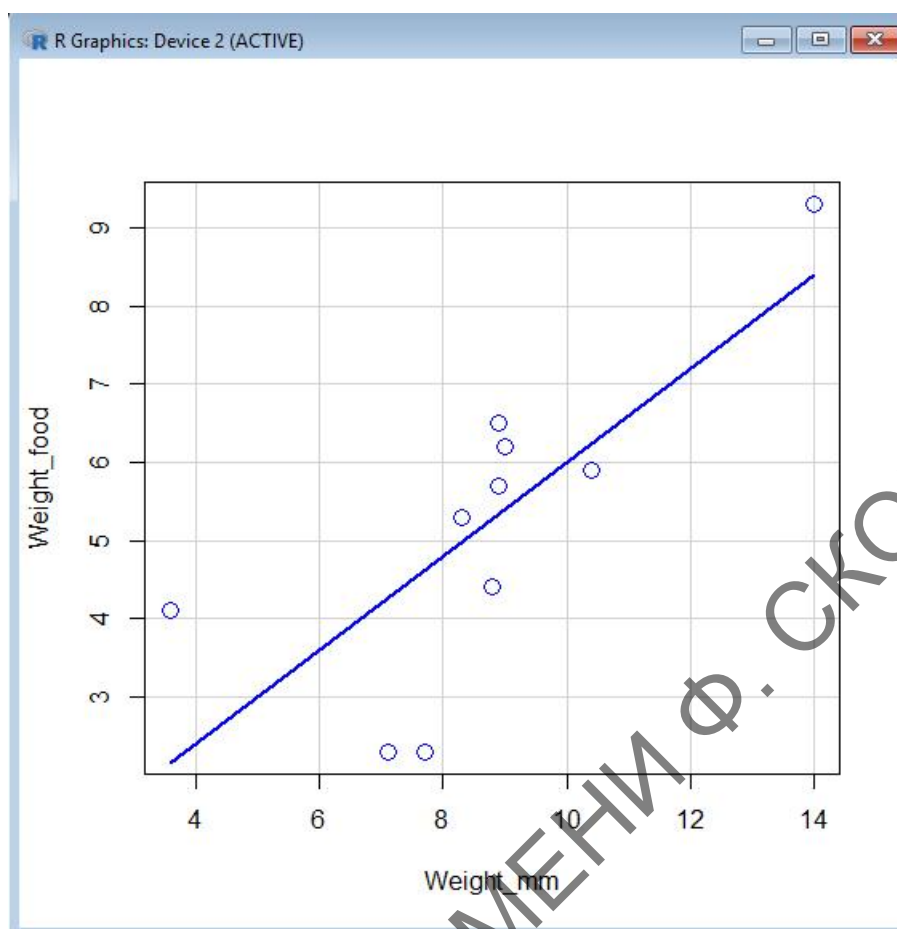


Рисунок 51 – Диаграмма рассеивания, реализованная при помощи пакета RCommander

Сохранить полученную диаграмму также несложно. Для этого достаточно кликнуть правой клавишей мыши по любому месту диаграммы и в контекстном меню выбрать опцию **Сохранить как метафайл...** (рисунок 52).

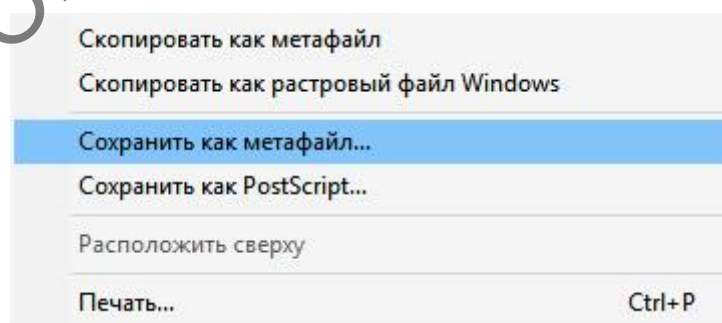


Рисунок 52 – Контекстное меню **Сохранить как метафайл...** в RCommander

После чего необходимо указать в диалоговом окне сохранения папку, куда он будет помещён, и имя нового файла с расшире-



нием .emf. Данный формат позволяет интегрировать сохранённый рисунок в любое офисное приложение.

Как вариант сохранения рисунка можно использовать само меню RCommander.

**Шаг 1.** Необходимо перейти в меню по пути **Графики** → **Сохранить график в файл** → **как растровый рисунок** (рисунок 53).

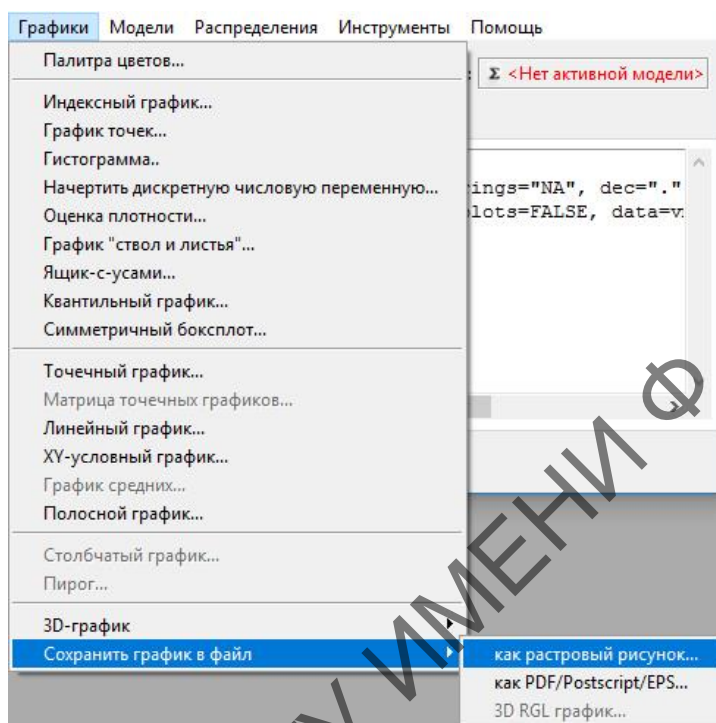


Рисунок 53 – Меню **Сохранить график в файл** в RCommander

**Шаг 2.** В появившемся диалоговом окне (рисунок 54) необходимо выставить настройки, как показано на рисунке, и нажать **ОК**.

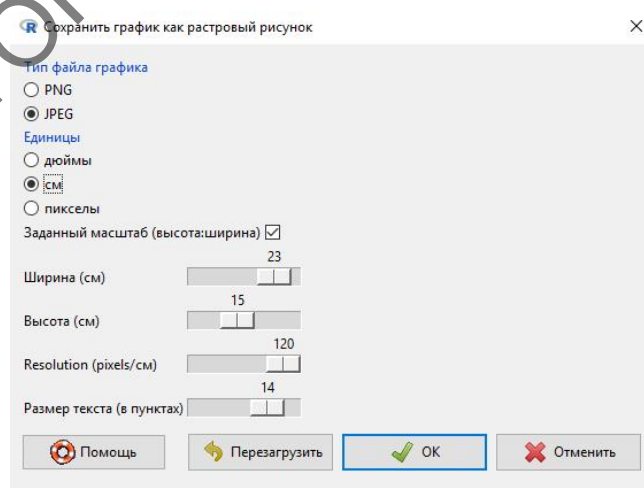


Рисунок 54 – Диалоговое окно **Сохранить график как растровый рисунок**



**Шаг 3.** После чего необходимо указать в диалоговом окне сохранения папку, куда он будет помещён, и имя нового файла.

## 1.2 Гистограмма

Гистограмма в статистическом анализе является очень удобным инструментом для визуализации формы распределения членов выборки и представляет собой распределение частот значений переменной (или частот значений на каждом из интервалов, на которые разбита выбранная переменная). Кривая, огибающая гистограмму, демонстрирует форму функции плотности распределения.

### 1.2.1 Построение гистограммы в RStudio (при помощи командной строки и базового пакета)

Для изучения базовых возможностей языка программирования R при построении гистограмм воспользуемся данными по весу самцов и самок тушканчиков из задания № 5 к теме 2. Гистограмма в базовом наборе языка R строится при помощи функции `hist()`.

**Шаг 1.** Создадим числовой вектор веса самцов тушканчика `wm`:

```
> wm <- c(186, 173, 156, 153, 190, 157, 156, 152, 165, 179, 165, 151, 182, 164, 160, 173, 182, 146, 160, 182, 173, 161, 180, 144, 144)
```

**Шаг 2.** Аналогично создаём числовой вектор веса самок тушканчиков `wf`:

```
> wf <- c(162, 157, 155, 153, 163, 162, 174, 165, 190, 186, 180, 141, 188, 175, 148, 164, 147, 147, 175, 146, 145, 145, 145, 145, 144)
```

**Шаг 3.** Построим гистограмму распределения веса самцов тушканчика, используя аргумент заголовка `main=""`:

```
> hist(wm, main="Вес самцов тушканчиков, г")
```

В результате наших действий в окне диаграмм и графиков RStudio будет отражена гистограмма, приведённая на рисунке 55. В связи с тем, что количество получившихся столбцов имеют достаточно широкий размах (10 значений признака), данная гистограмма получилась малоинформативной. В связи с этим есть смысл уменьшить размах вручную хотя бы до 5 значений признака, что увеличит изначальные 5 столбцов гистограммы до 10. Для этого используется аргумент `breaks =`.

**Шаг 4.** Уплотняем гистограмму:

```
> hist(wm, breaks = 10, main="Вес самцов тушканчиков, г")
```

Результат представлен на рисунке 56.



Рисунок 55 – Образец базовой гистограммы в R

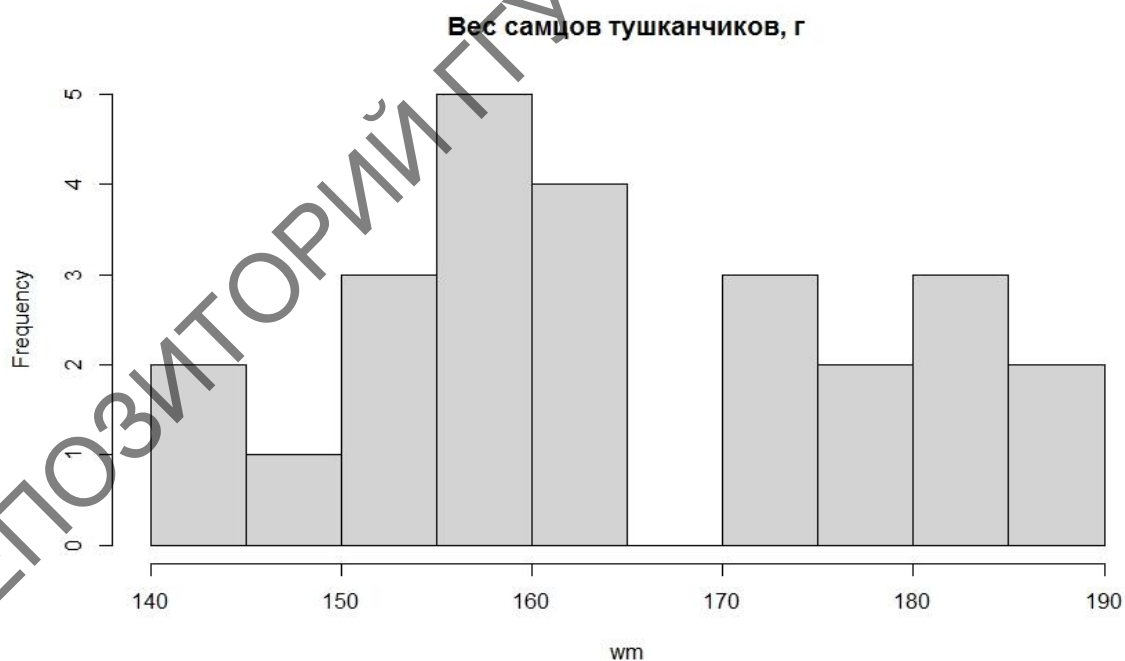


Рисунок 56 – Образец уплотнённой гистограммы в R

Здесь следует заметить, что по умолчанию функция `hist()` показывает по оси  $y$  частоты встречаемости каждого класса значений переменной, размещённой по оси  $x$ . Кроме частоты встречаемости по оси ординат можно отложить показатель плотности вероятности каждого класса (суммарно он равен единице). Для этого используется аргумент `freq` (*frequency* – частота), которому присваивается значение `FALSE`.

**Шаг 5.** Меняем показатель по оси ординат с частоты на плотность:

```
> hist(wm, breaks = 10, freq = FALSE, main="Вес самцов тушканчиков, г")
```

Полученный результат представлен на рисунке 57.

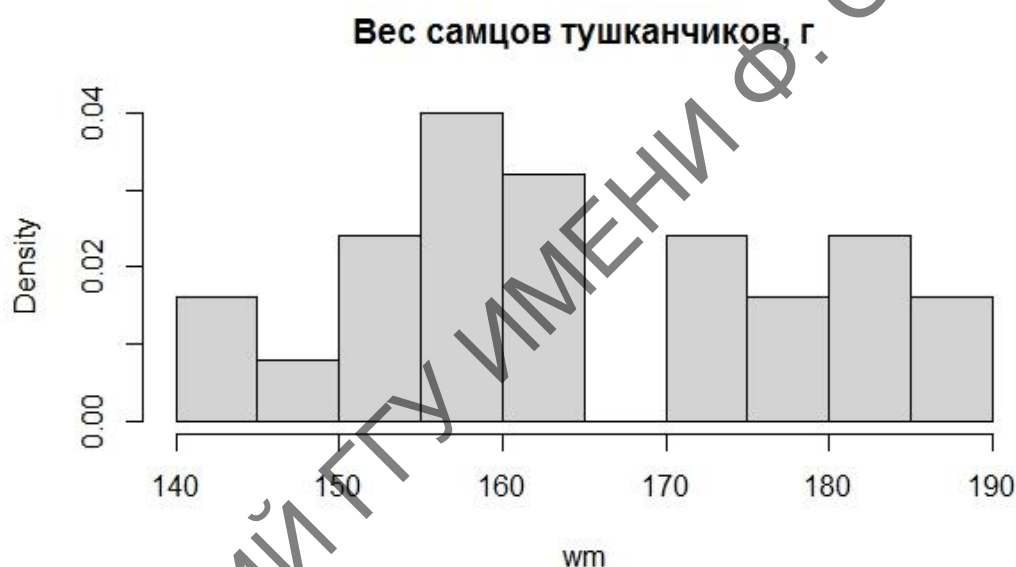


Рисунок 57 – Гистограмма с отображением плотности вероятности по оси ординат в R

Публикации не всегда бывают чёрно-белыми, поэтому часто возникает необходимость показать столбцы гистограммы цветом, отличным от серого, который используется по умолчанию. Для этого следует воспользоваться ещё одним аргументом функции `hist()`, а именно – `col=""` (*color* – цвет). Сделаем цвет наших столбцов, показывающих вес самцов тушканчиков, синим.

**Шаг 6.** Меняем цвет столбцов гистограммы. Заодно вернём и показатель частоты на ось ординат.

```
> hist(wm, breaks = 10, col = "blue", main="Вес самцов тушканчиков, г")
```

Полученный результат отображён на рисунке 58.

Иногда гистограмма сама по себе не всегда даёт значимую информацию о распределении элементов выборки (например, столбцов мало или они расположены с промежутками), поэтому можно для дополнительной характеристики совокупности также использовать кривую плотности вероятности.

Для построения этой кривой используется функция `density()`.



Рисунок 58 – Гистограмма с окрашенными столбцами в R

**Шаг 7.** Построим кривую плотности вероятности для веса самцов тушканчиков. Полученный результат сравните с рисунком 59.

```
> plot(density(wm))
```

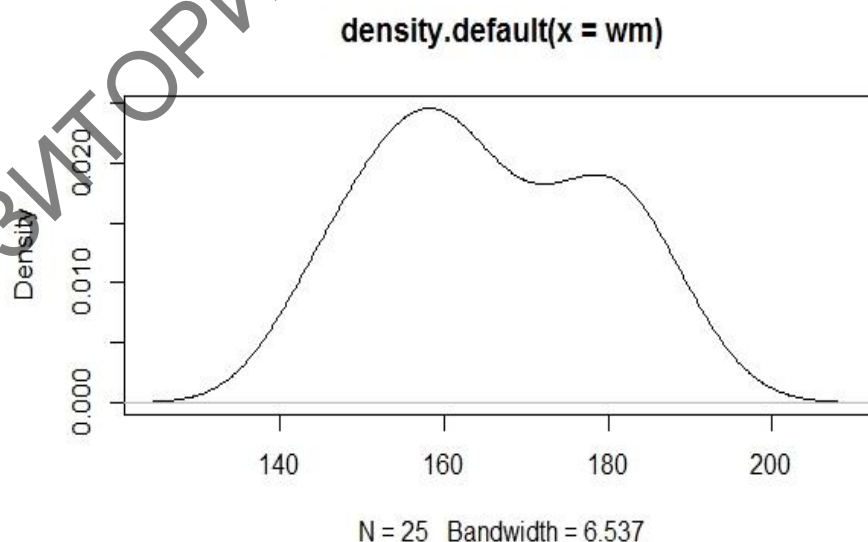


Рисунок 59 – Кривая плотности вероятности веса самцов тушканчиков в R

На данном графике также можно увидеть объём выборки ( $N=25$ ), а также уровень сглаженности (*Bandwith* – полоса пропускания), за который отвечает аргумент `bw` = (сокращённо от *bandwith*).

Особый визуальный эффект достигается при совмещении столбчатой гистограммы и кривой плотности вероятности.

**Шаг 8.** Совместим столбчатую гистограмму распределения веса самцов тушканчиков и кривую плотности вероятности, раскрасив её в красный цвет.

```
> hist(wm, breaks = 10, freq = FALSE, xlab = "Масса, г",  
ylab = "плотность вероятности", col = "blue", main="Вес  
самцов тушканчиков, г") # Строим гистограмму  
> lines(density(wm), col = "red", lwd = 2) # Кривая  
вероятности
```

Сверьте полученный результат с рисунком 60. Обратите внимание, что в данном коде присутствуют аргументы «`xlab =`» и «`ylab =`». Это аргументы подписей соответственно по оси  $x$  (*x label* – подпись  $x$ ) и по оси  $y$  (*y label* – подпись  $y$ ). Используя данные аргументы, можно подписать оси по своему усмотрению.



Рисунок 60 – Гистограмма веса самцов тушканчиков совмещённая с кривой плотности вероятности

**Шаг 9.** Построим аналогичную гистограмму, совмещённую с кривой плотности вероятности по весу самок тушканчиков, сделав цвет столбцов жёлтым, и сверим полученный результат с рисунком 61.

```
> hist(wf, breaks = 10, freq = FALSE, xlab = "Масса, г",
ylab = "Плотность вероятности", col = "yellow",
main="Вес самок тушканчиков, г")
> lines(density(wf), col = "red", lwd = 2)
```

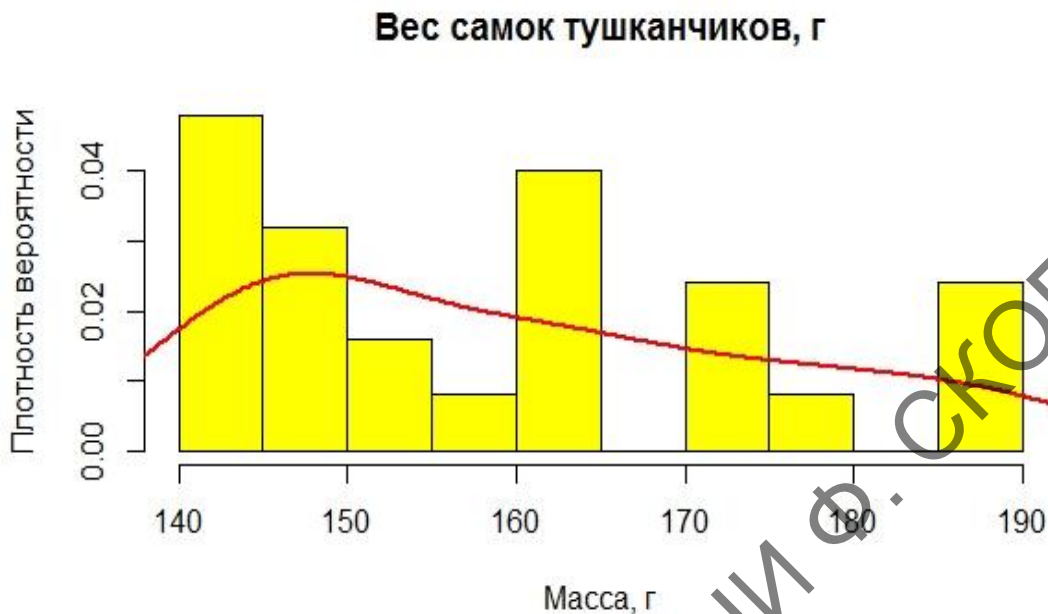


Рисунок 61 – Гистограмма веса самок тушканчиков совмещённая с кривой плотности вероятности

Иногда возникает необходимость совмещения двух графиков на одном рисунке для большей наглядности. Это можно сделать следующим образом.

**Шаг 10.** Разместим обе наши гистограммы по весу самцов и самок тушканчиков на одном рисунке:

```
> par(mfrow=c(2,1))
> hist(wm, breaks = 10, freq = FALSE, xlab = "Масса, г",
ylab = "Плотность вероятности", col = "blue", main="Вес самцов тушканчиков, г") # Строим гистограмму
> lines(density(wm), col = "red", lwd = 2) # кривая вероятности
> hist(wf, breaks = 10, freq = FALSE, xlab = "Масса, г",
ylab = "Плотность вероятности", col = "yellow",
main="Вес самок тушканчиков, г")
> lines(density(wf), col = "red", lwd = 2)
```

Ключевая команда здесь – `par()`. В первой строчке изменяется один из ее параметров, `mfrow`, который регулирует, сколько изображений и каким образом будут размещены на «листе». Значение `mfrow` по умолчанию – `c(1,1)`, то есть один график по вертикали и один по горизонтали.

Не стоит переживать, если получившийся в окне **Plot** график сильно уплощён. При сохранении рисунка как изображение (см. подраздел 1.1.1 данной темы) в правом верхнем углу диалогового окна **Save Plot as Image** в боксах **Width** (Ширина) и **Height** (Высота) укажите значение не менее 1000, и сохранившийся рисунок будет иметь нормальный вид (рисунок 62).

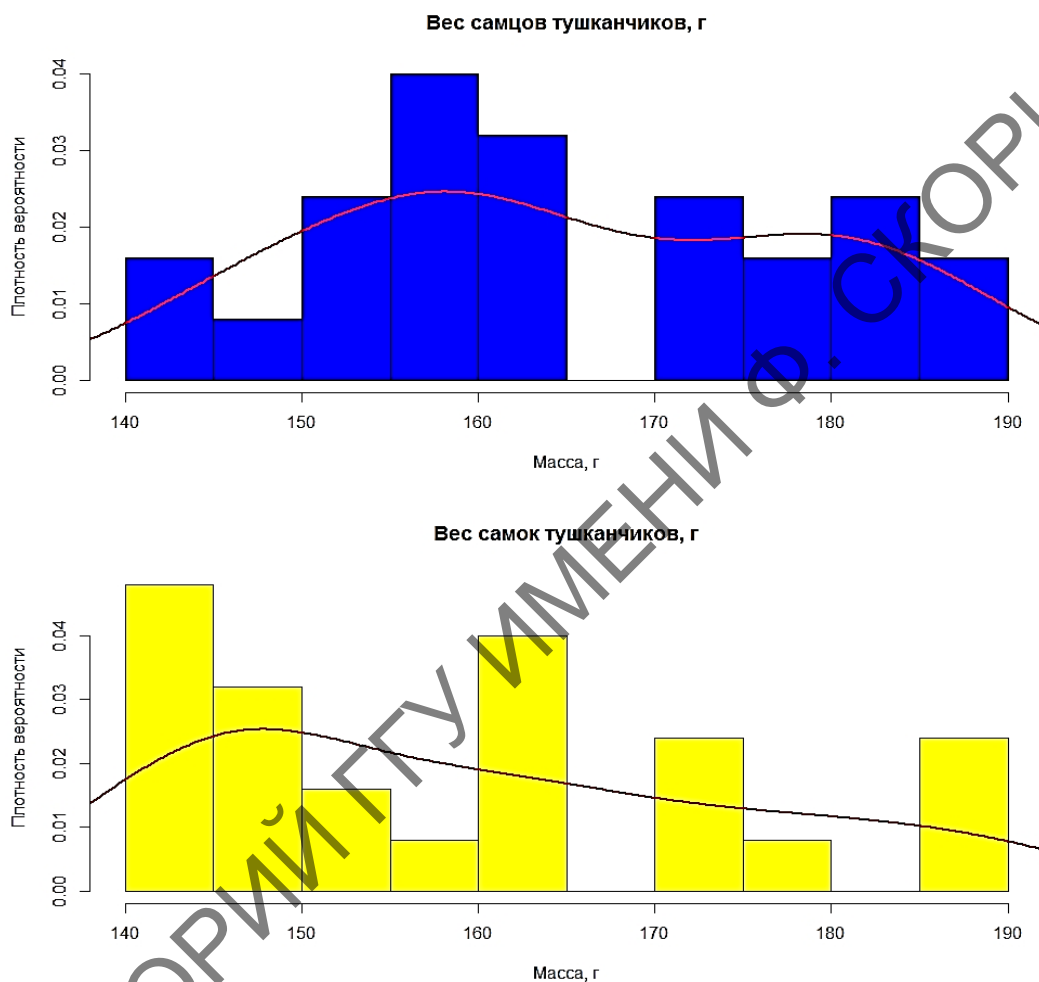


Рисунок 62 – Гистограммы веса тушканчиков, расположенные на одном листе

## 1.2.2 Построение гистограммы в RStudio (при помощи командной строки и пакета ggplot2)

Перед тем как рассматривать построение гистограмм при помощи графического пакета **ggplot2**, убеждаемся, что он загружен и включён. Если нет, то это необходимо сделать (см. п. 1 данной темы). Для учебных целей продолжим использовать данные по весу самцов и самок тушканчиков, которые были нами использованы ранее в разделе 1.1.1.

Следует также напомнить, что пакет **ggplot2** работает не с отдельными числовыми векторами, а с датафреймами, поэтому необходимо из данных по весу тушканчиков – переменная *wm* – *weight of males* (вес самцов) и переменная *wf* – *weight of females* (вес самок) – создать датафрейм *wmf*.

**Шаг 1.** Создаём датафрейм:

```
> wmf <- data.frame("weight_male"=wm, "weight_female"=wf)
```

**Шаг 2.** Создаём гистограмму распределения веса самцов:

```
> ggplot(wmf, aes(x = wm))+  
  geom_histogram()
```

Полученная гистограмма показана на рисунке 63.

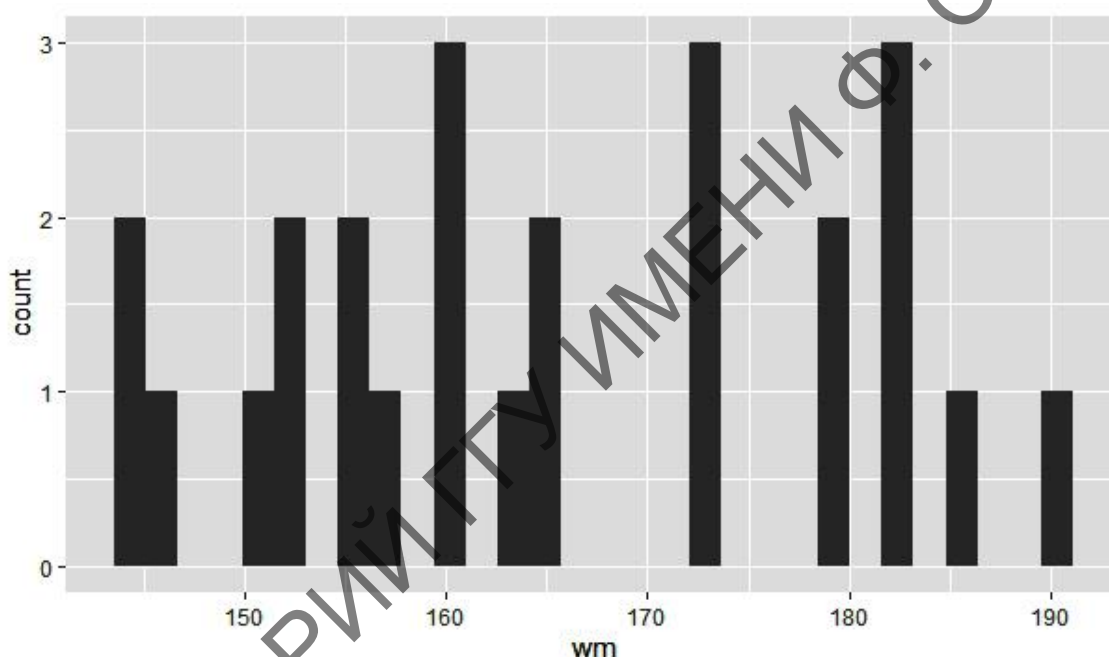


Рисунок 63 – Гистограмма веса самцов тушканчиков с параметрами по умолчанию в **ggplot2**

По умолчанию пакет **ggplot2** выставляет 30 столбцов, и в том случае, если выборка содержит меньше данных, то появляются промежутки между столбцами. Поэтому нужно в параметре `geom_histogram()` указать ширину столбца, за которую отвечает аргумент «`binwidth =`» (ширина столбца). В предыдущем случае мы использовали параметр «5», используем его и сейчас. Одновременно окрасим столбцы нашей гистограммы в голубой цвет (аргумент «`fill =`») с чёрной границей (аргумент «`colour =`»). Обратите



внимание, что название аргумента цвета границы имеет написание, как в британском английском, а не в американском варианте.

**Шаг 3.** Создаём гистограмму распределения веса самцов с заданными параметрами и сверим полученный результат с рисунком 64:

```
> ggplot(wmf, aes(x = wm)) +  
  geom_histogram(binwidth = 5, fill = "lightblue",  
  colour = "black")
```

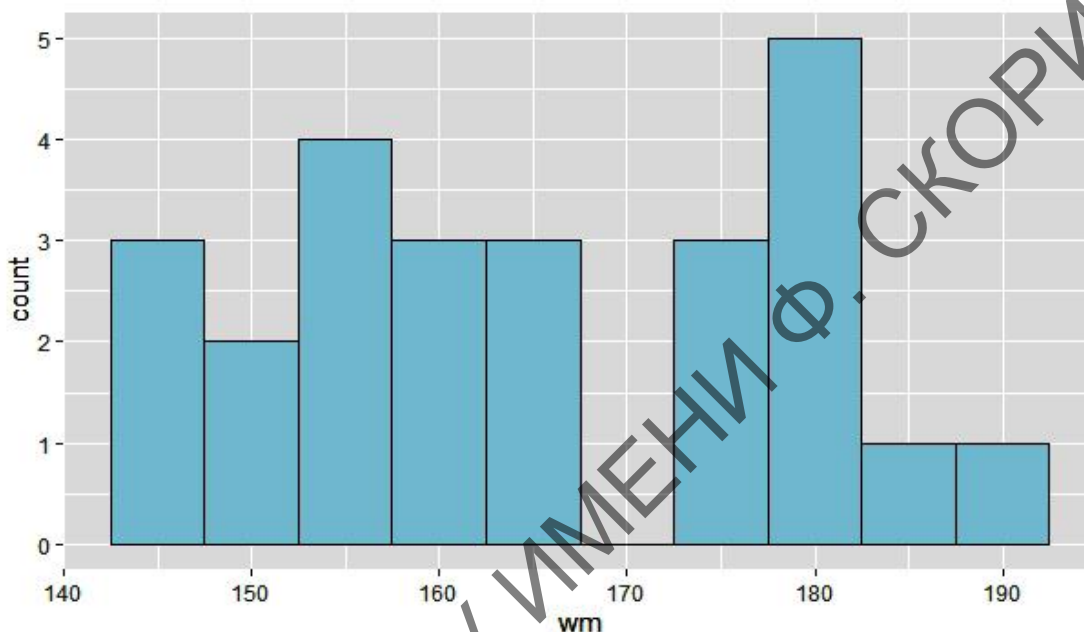


Рисунок 64 – Гистограмма веса самцов тушканчиков с заданными параметрами ширины и цвета столбца в **ggplot2**

Данный графический пакет позволяет также строить кривую плотности вероятности. Используем для построения кривой плотности вероятности данные по весу самцов тушканчиков и функцию `geom_density()`.

**Шаг 4.** Построим кривую плотности вероятности и сверим полученный результат с рисунком 65:

```
> ggplot(wmf, aes(x = wm)) +  
  geom_density()
```

Линию кривой плотности вероятности можно сделать и другим цветом, а также регулировать ее изгиб. Это прописывается как аргументы функции `geom_density()`: «`colour =`» и «`adjust=`» соответственно. Сделаем нашу кривую плотности вероятности более крутой и красного цвета.

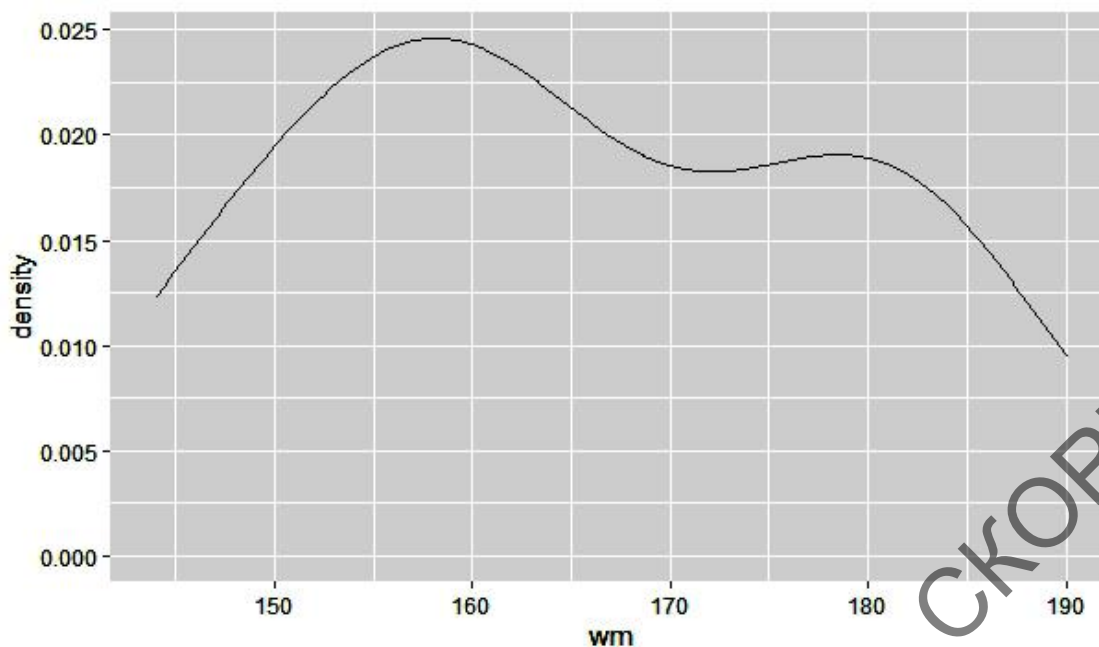


Рисунок 65 – Кривая плотности вероятности веса самцов тушканчиков, выполненная в **ggplot2**

**Шаг 5.** Построим кривую плотности вероятности с заданными параметрами кривизны и цвета, а после сверим полученный результат с рисунком 66:

```
> ggplot(wmf, aes(x = wm)) +  
  geom_density(stat = "density", adjust = .4, colour =  
  "red")
```

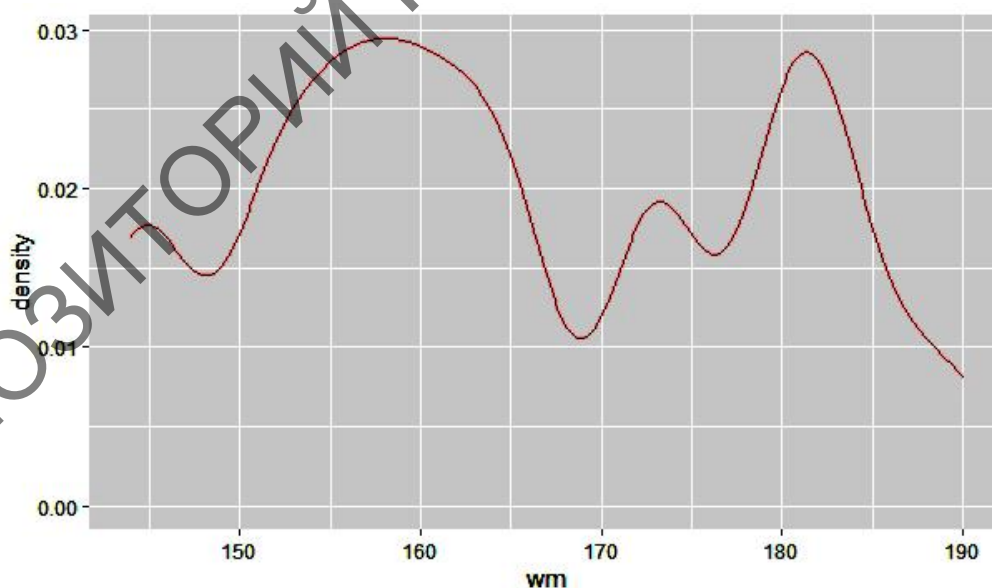


Рисунок 66 – Кривая плотности вероятности веса самцов тушканчиков с заданными параметрами кривизны и цвета в **ggplot2**

Графический пакет **ggplot2** также позволяет совмещать столбчатую гистограмму и кривую плотности вероятности. Для этого используется набор команд, соединённых знаком «+».

**Шаг 6.** Соединим столбчатую гистограмму с заданным цветом столбцов (голубой), границ столбцов (серый) и линии плотности вероятности (красный). Полученный результат сравните с рисунком 67.

```
> ggplot(wmf, aes(x = wm, y = ..density..)) +
  geom_histogram(binwidth = 3, fill = "lightblue",
  colour = "grey60", size = .3) +
  geom_density(colour = "red") +
  xlim(140, 200)
```

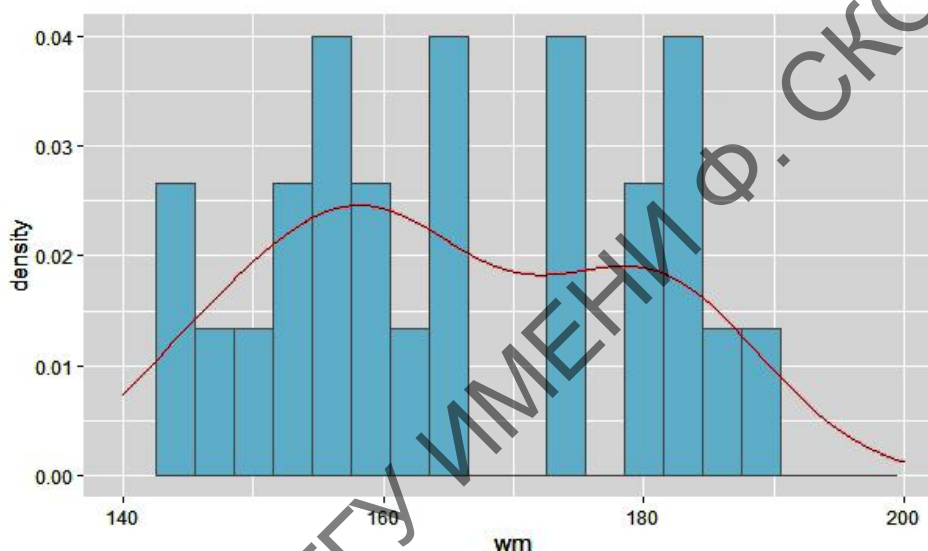


Рисунок 67 – Гистограмма веса самцов тушканчиков совмещённая с кривой плотности вероятности в **ggplot2**

Здесь следует обратить внимание на функцию `xlim()`, аргументами которой служат минимальное и максимальное значения анализируемого признака в выборке, которые откладываются по оси абсцисс.

Графический пакет **ggplot2** может совмещать, как и рассмотренный выше базовый вариант (рисунок 62), 2 графика в теле одного рисунка. Правда, для этого необходима предварительная подготовка датафрейма. Так, в наших данных имеются сведения о поле зверьков и об их весе в зависимости от пола. Допустим, необходимо совместно показать гистограммы распределения веса сначала самок (сверху), а потом самцов (снизу). Поэтому датафрейм должен содержать 2 столбца: пол (*sex*) и вес (*weight*). В столбце «*sex*» надо будет сначала указать 25 значений **male** (самцы), а затем следом 25 значений **female** (самки). Столбец «*weight*» будет содержать данные о весе.

**Шаг 7.** Составляем символьный вектор *sot*:

```
> sot <- c("male", "male", "male", "male", "male",  
"male", "male", "male", "male", "male", "male", "male", "male",  
"male", "male", "male", "male", "male", "male", "male", "fe-  
male", "female", "female", "female", "female", "female", "fe-  
male", "female", "female", "female", "female", "female", "fe-  
male", "female", "female", "female", "female", "female", "fe-  
male", "female", "female", "female", "female", "female", "fe-  
male", "female", "female")
```

**Шаг 8.** Составляем числовой вектор *wot*:

```
> wot <- c(186, 173, 156, 153, 190, 157, 156, 152,  
165, 179, 165, 151, 182, 164, 160, 173, 182, 146, 160,  
182, 173, 161, 180, 144, 144, 162, 157, 155, 153, 163,  
162, 174, 165, 190, 186, 180, 141, 188, 175, 148, 164,  
147, 147, 175, 146, 145, 145, 145, 145, 144)
```

**Шаг 9.** Формируем датафрейм *tsk*:

```
> tsk <- data.frame("Sex" = sot, "weight" = wot)
```

**Шаг 10.** Сформируем рисунок из двух гистограмм со столбцами зелёного цвета, чёрными границами и шириной в 5 значений признака. Полученный результат сверяем с рисунком 68:

```
> ggplot(tsk, aes(x = wot)) +  
  geom_histogram(binwidth = 5, fill = "green", colour =  
  "black") +  
  facet_grid(Sex ~ .)
```

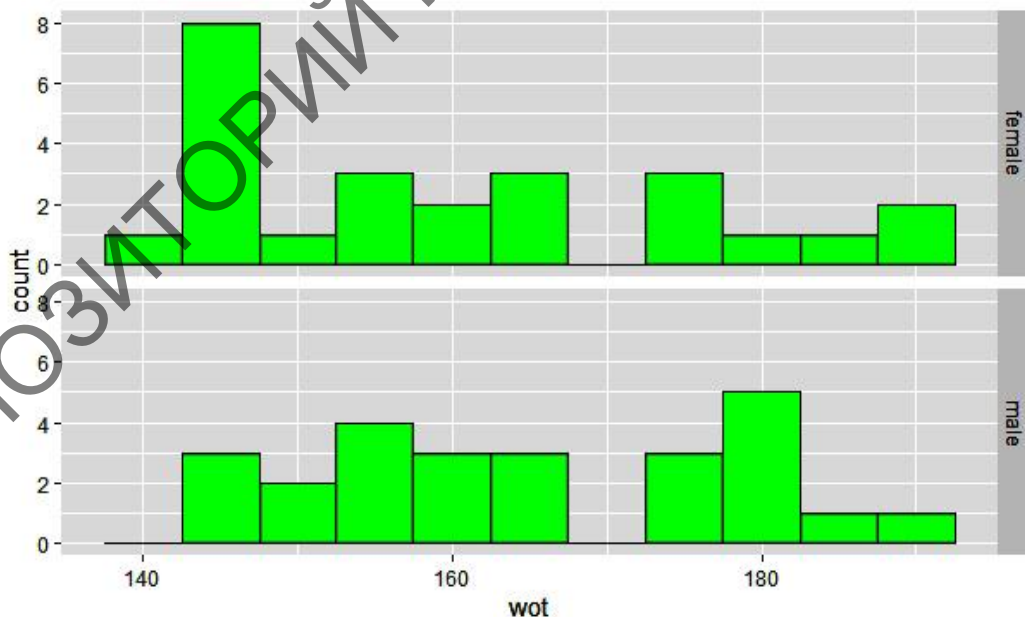


Рисунок 68 – Гистограммы расположенные на одном листе в **ggplot2**

### 1.2.3 Построение гистограммы в RCommander (при помощи GUI)

Графические возможности пакета RCommander хоть и в значительной степени облегчают построение гистограммы, но не отличаются гибкостью и разнообразием настроек. Это позволяет сделать лишь базовую гистограмму, лишённую всех выше рассмотренных возможностей оформления. Рассмотрим построение гистограммы по данным о весе самцов и самок тушканчиков (таблица данных должна содержать два столбца с данными по весу: один озаглавлен как **Male** (Самцы), второй как **Female** (Самки)).

**Шаг 1.** Включаем пакет RCommander. Как это сделать см. раздел 1.1 темы 1.

**Шаг 2.** Загружаем данные, либо подготовленные в сторонней программе электронных таблиц (см. п.1.2.2.1 темы 1), либо набранные заново в самой программе (см. раздел 2.1. темы 1).

**Шаг 3.** Переходим в меню по пути **Графики** → **Гистограмма...** (рисунок 69).

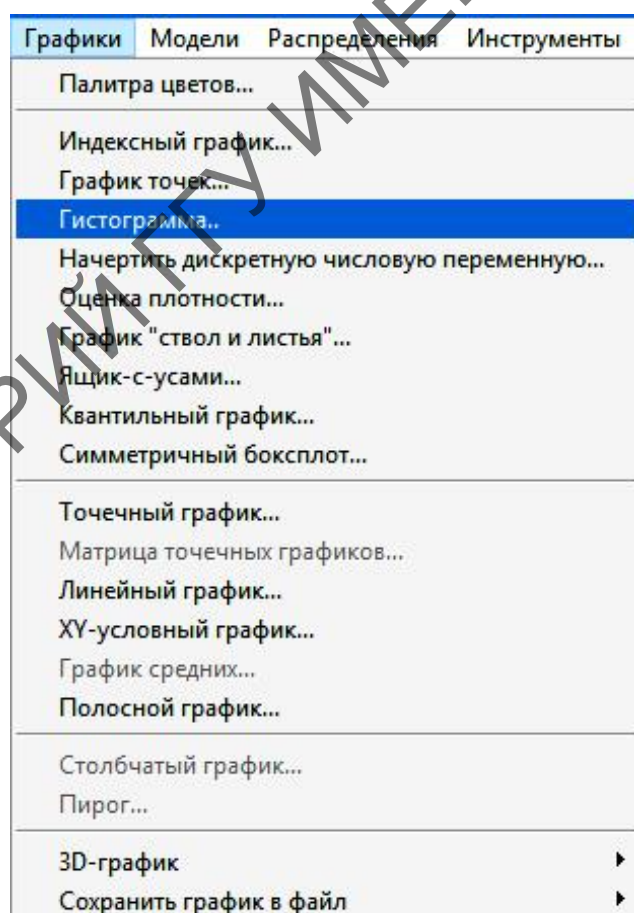


Рисунок 69 – Пункт меню **Гистограмма...** в RCommander

**Шаг 4.** В появившемся диалоговом окне **Гистограмма** необходимо перейти на закладку **Данные** и выбрать одну из переменных, по которой будет построена гистограмма. Выберем, например, самцов (рисунок 70).

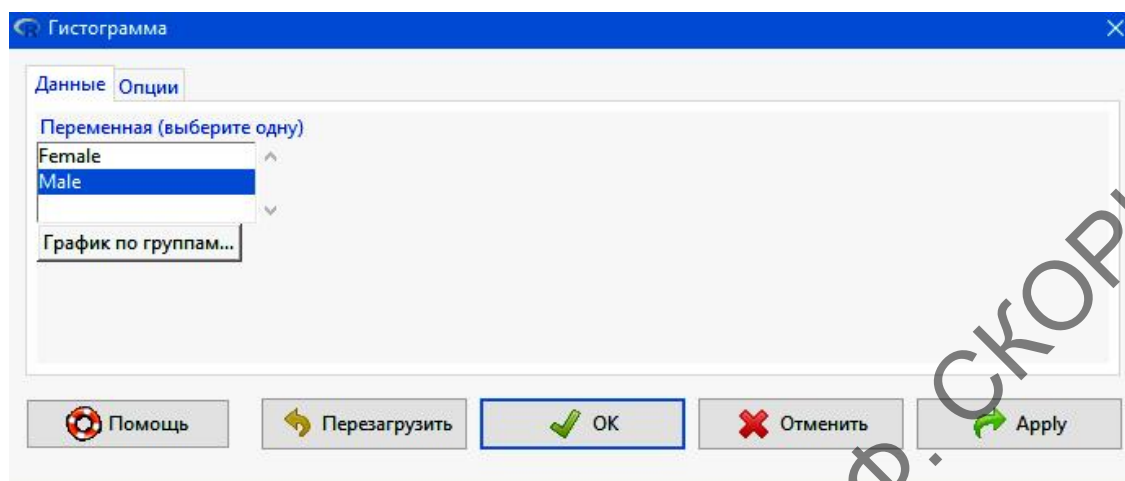


Рисунок 70 – Закладка **Данные** диалогового окна **Гистограмма** в R Commander

**Шаг 5.** Затем необходимо перейти на закладку **Опции** и определить свойства будущей гистограммы. В нашем случае определим опции как на рисунке 71. После чего необходимо нажать **ОК**.

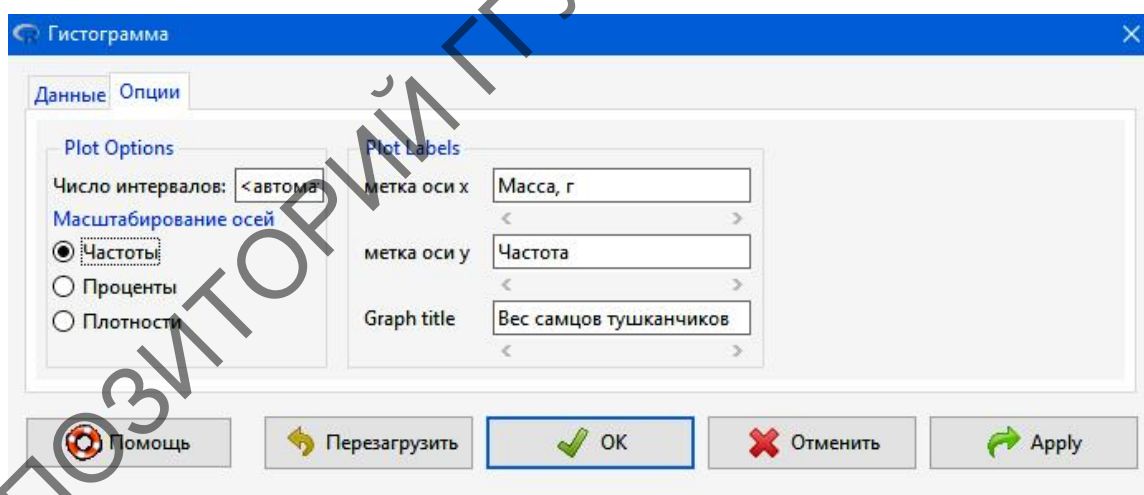


Рисунок 71 – Закладка **Опции** диалогового окна **Гистограмма** в R Commander

Получившийся график отображен на рисунке 72. Он появится или в окне среды программирования R, или в окне графиков RStudio.

Вес самцов тушканчиков

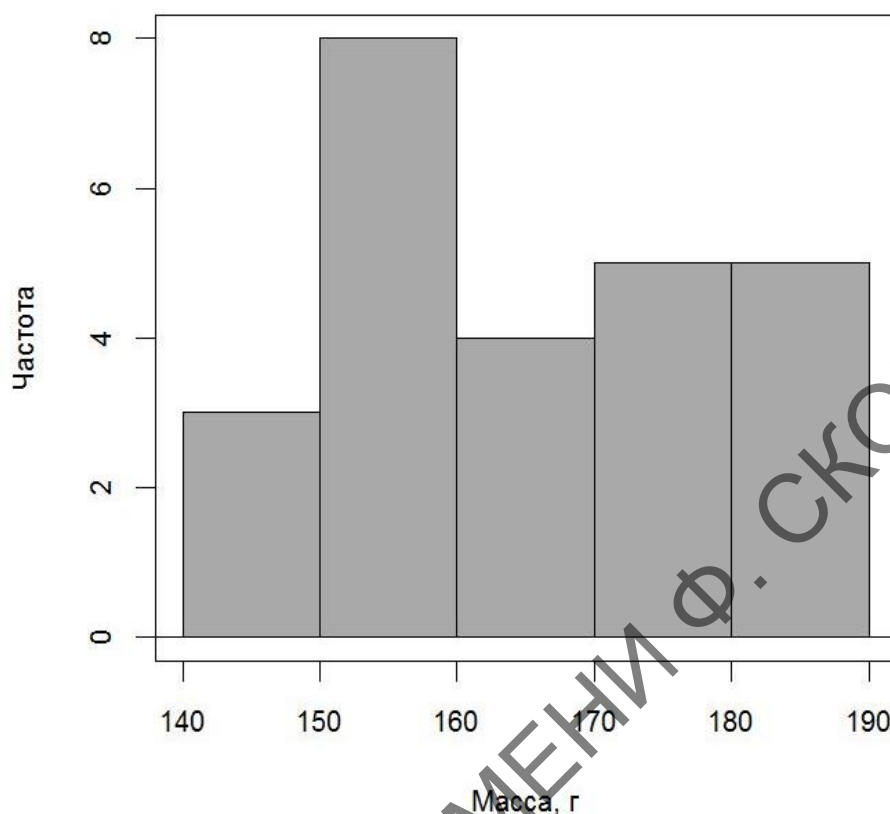


Рисунок 72 – Гистограмма веса самцов тушканчиков с заданными параметрами в RCommander

Данный пакет также может строить график кривой плотности вероятности, который кроме самой кривой показывает зону с наиболее «плотными» значениями переменной (показаны вертикальными чертами на оси абсцисс: чем плотнее, тем ближе они расположены).

**Шаг 6.** Для построения кривой плотности вероятности для самцов необходимо выбрать опцию меню **Оценка плотности** в пункте меню **Графики**.

**Шаг 7.** В появившемся диалоговом окне в закладке **Данные** необходимо выбрать переменную **Male**, а затем, перейдя в закладку **Опции**, настроить нашу будущую кривую согласно рисунку 73.

Аналогично постройте гистограмму и кривую плотности вероятности по весу самок тушканчиков.

Полученная кривая будет отображена на рисунке 74. Отметим, что при создании гистограмм и кривых при помощи GUI недоступны настройки цвета и тонкие настройки столбцов и линий, что значительно ограничивает возможности оформления.



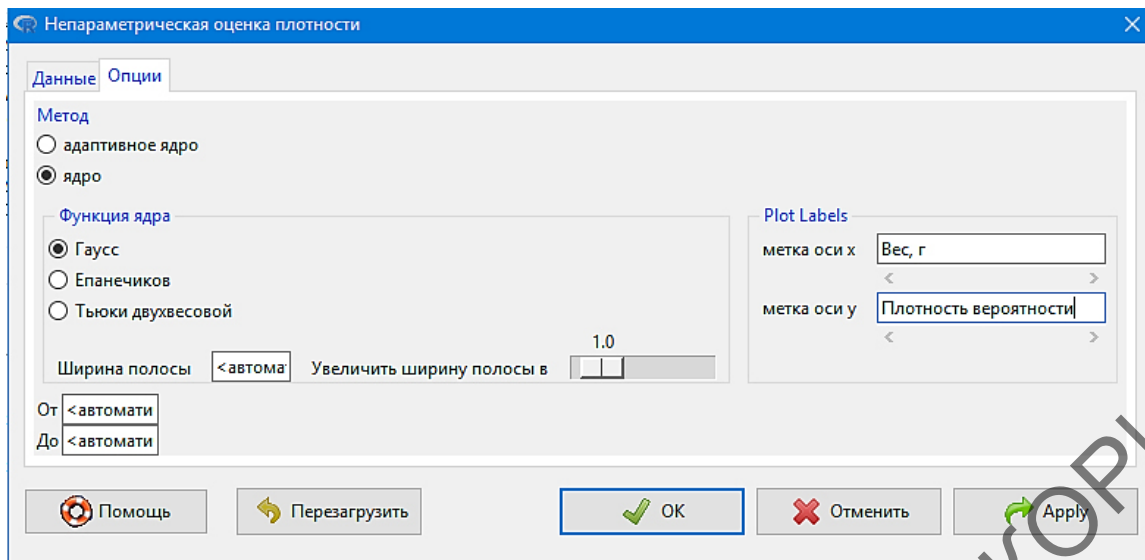


Рисунок 73 – Закладка **Опции** диалогового окна **Непараметрическая оценка плотности** в R Commander

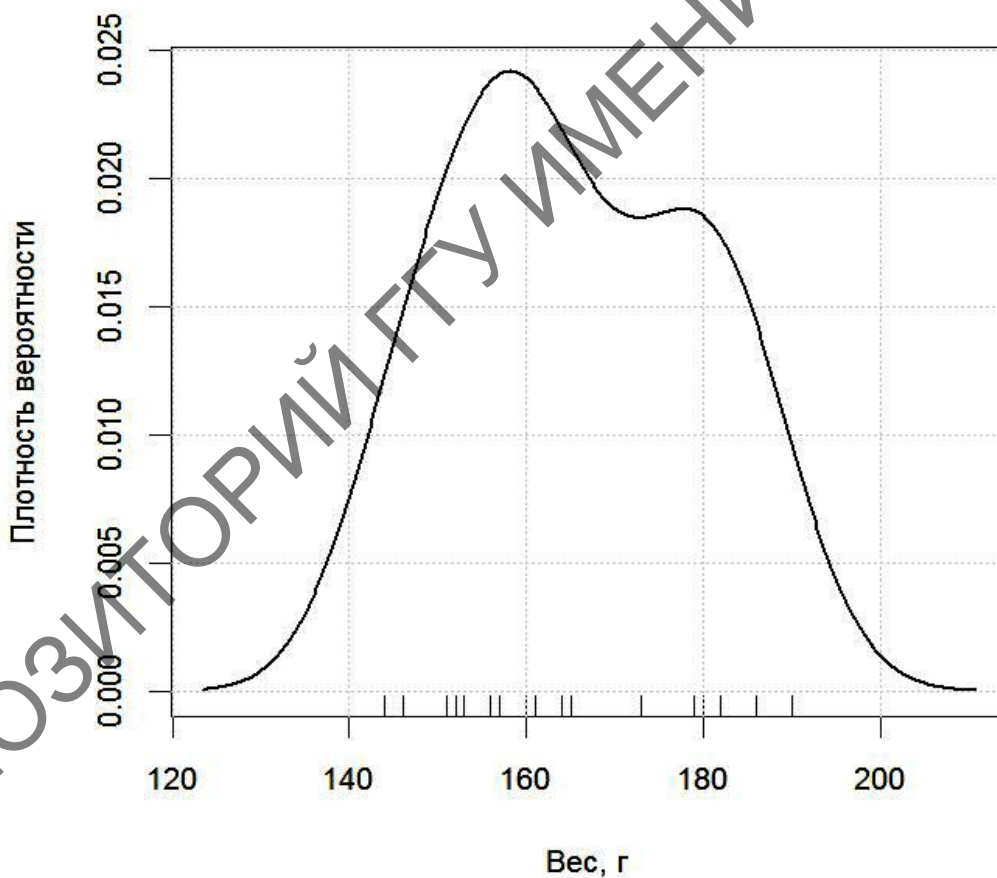


Рисунок 74 – Кривая плотности вероятности веса самцов тушканчиков, сделанная в R Commander



## 1.3 Линейный график

Линейный график представляет собой линию, соединяющую ряд точек данных на координатной сетке значений. Подобного типа графики обычно используются для определения динамики изменений или тенденций в данных.

Для построения линейных графиков в языке программирования R используется несколько функций, основной из которых является `plot()`. Аргументами этой функции являются: название числового вектора с данными; `type = ""` – тип графика (линия – *l*, точка – *p*, точки, соединённые линиями – *o*); `xlab = ""` – это метка для оси *x*; `ylab = ""` – метка для оси *y*; `col = ""` – задание цвета точкам и линиям; `main = ""` – название графика.

### 1.3.1 Построение линейного графика в RStudio (при помощи командной строки и базового пакета)

В качестве учебного примера построения графиков рассмотрим данные численности количества экземпляров в 20 почвенных ловушках, выставленных по мере удаления от нефтяных скважин на лугу и в лесу (таблица 4).

Таблица 4 – Численность беспозвоночных

<i>nt</i>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
<i>nm</i>	10	10	10	11	12	13	15	16	18	20	21	22	23	25	26	27	32	36	39	42
<i>nf</i>	5	5	5	6	7	7	8	8	10	11	12	13	15	17	22	25	30	35	45	56

Поясним таблицу. Здесь *nt* – это номер почвенной ловушки (сокращённо от *number of trap*), *nm* – количество беспозвоночных около скважины, размещённой на лугу (сокращённо от *number of meadow*) и *nf* – количество беспозвоночных около скважины, размещённой в лесу (сокращённо от *number of forest*). Для построения графиков по численности беспозвоночных необходимо создать три числовых вектора.

**Шаг 1.** Создадим вектор количества ловушек. Так как они увеличиваются с шагом через одну, то нет смысла писать все цифры от 1 до 20, достаточно просто задать совокупность.

```
> nt <- c(1:20)
```

**Шаг 2.** Создадим вектор численности беспозвоночных около скважины на лугу.

```
> nm <- c(10, 10, 10, 11, 12, 13, 15, 16, 18, 20, 21, 22, 23, 25, 26, 27, 32, 36, 39, 42)
```

**Шаг 3.** Создадим вектор численности беспозвоночных около скважины в лесу.

```
> nf <- c(5, 5, 5, 6, 7, 7, 8, 8, 10, 11, 12, 13, 15, 17, 22, 25, 30, 35, 45, 56)
```

Теперь всё готово к построению графиков. Графики численности беспозвоночных можно строить и по одному, но в данном случае есть смысл отобразить оба на одной координатной сетке для более полного отображения результатов и сравнения. Также имеет значение и оформление линий графиков и точек, относящихся к разным скважинам разным цветом, ось абсцисс подпишем как **Ловушки**, а ось ординат как **Количество, экз.** Называться график будет **Численность беспозвоночных**.

**Шаг 4.** Строим графики. Изначальным будет график, содержащий наименьшие данные, т. е. «лесной», а затем к нему добавим «луговой». В противном случае первые данные «лесного» графика будут обрезаны. «Лесной» график будет красным, «луговой» – синим.

```
> plot(nf, type = "l", col = "red", xlab = "ловушки", ylab = "Количество, экз", main = "численность беспозвоночных")
> lines(nm, type = "l", col = "blue") # добавляем данные по лесу
> points(nt, nf, col = "red") # добавляем точки по лугу
> points(nt, nm, col = "blue") # добавляем точки по лесу
```

Сверим полученный график с рисунком 75.

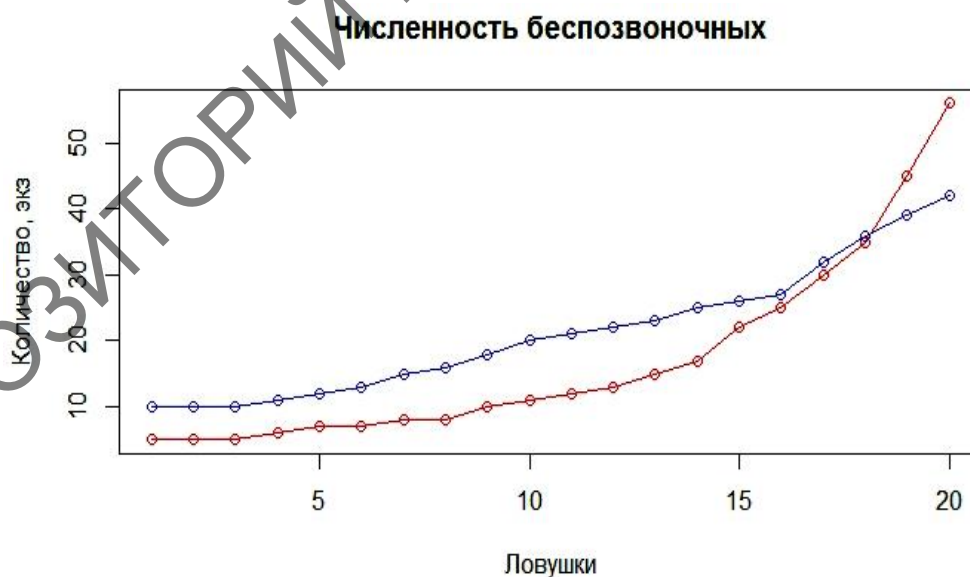


Рисунок 75 – Графики численности беспозвоночных в окрестностях нефтяных скважин

### 1.3.2 Построение линейного графика в RStudio (при помощи командной строки и пакета ggplot2)

Перед тем как начать рассматривать построение линейных графиков при помощи пакета **ggplot2**, следует напомнить, что этот пакет работает не с отдельными векторами, а датафреймами. Рассмотрим построение линейного графика при помощи **ggplot2**, используя данные по беспозвоночным в почвенных ловушках в окрестностях нефтяных скважин (таблица 4) и объединим ранее созданные вектора в датафрейм *inv* (*invertebrate* – беспозвоночные).

**Шаг 1.** Создаём и проверяем датафрейм *inv*:

```
> inv <- data.frame("Trap" = nt, "Meadow" = nm, "Forest" = nf)
> inv
```

	Trap	Meadow	Forest
1	1	10	5
2	2	10	5
3	3	10	5
4	4	11	6
5	5	12	7
6	6	13	7
7	7	15	8
8	8	16	8
9	9	18	10
10	10	20	11
11	11	21	12
12	12	22	13
13	13	23	15
14	14	25	17
15	15	26	22
16	16	27	25
17	17	32	30
18	18	36	35
19	19	39	45
20	20	42	56

**Шаг 2.** Включаем пакет **ggplot2**.

**Шаг 3.** Формируем график, используя данные нашего датафрейма: по оси абсцисс будут расположены номера ловушек, а по оси ординат – численность в ловушках у скважины на лугу.

```
> ggplot(inv, aes(x = nt, y = nm))+
  geom_line()
```

Полученный результат отображён на рисунке 76.

**Шаг 4.** Добавим точки на график и сверим результат с рисунком 77.

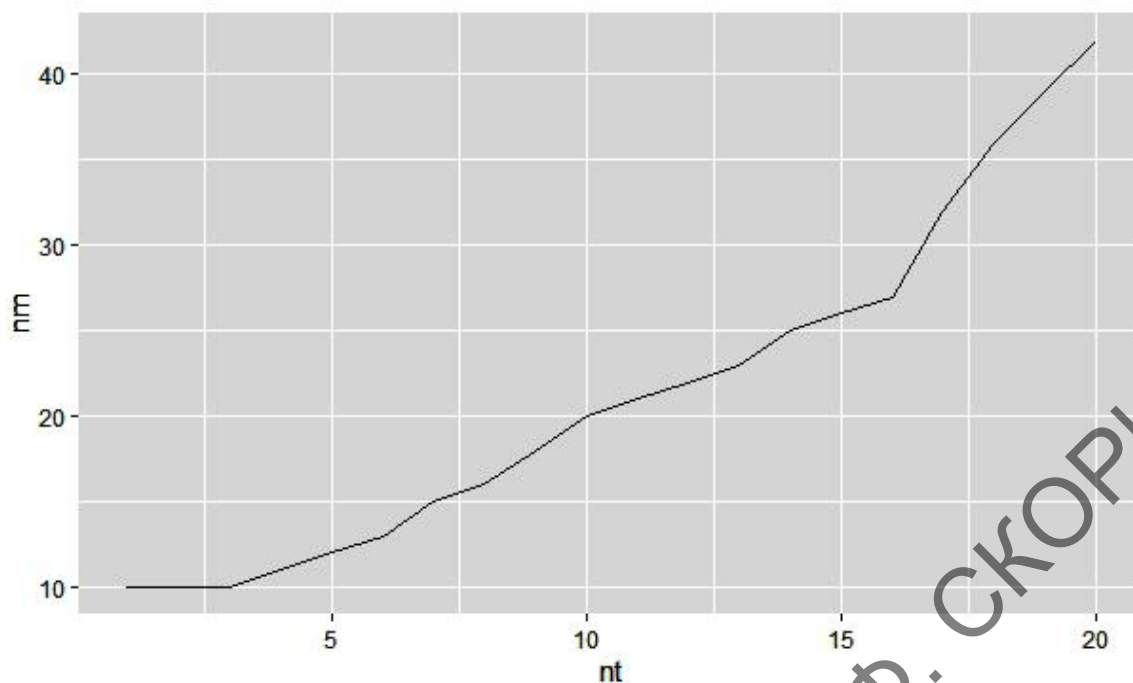


Рисунок 76 – Линейный график численности беспозвоночных в **ggplot2**

```
> ggplot(inv, aes(x = nt, y = nm))+
  geom_line()+
  geom_point()
```

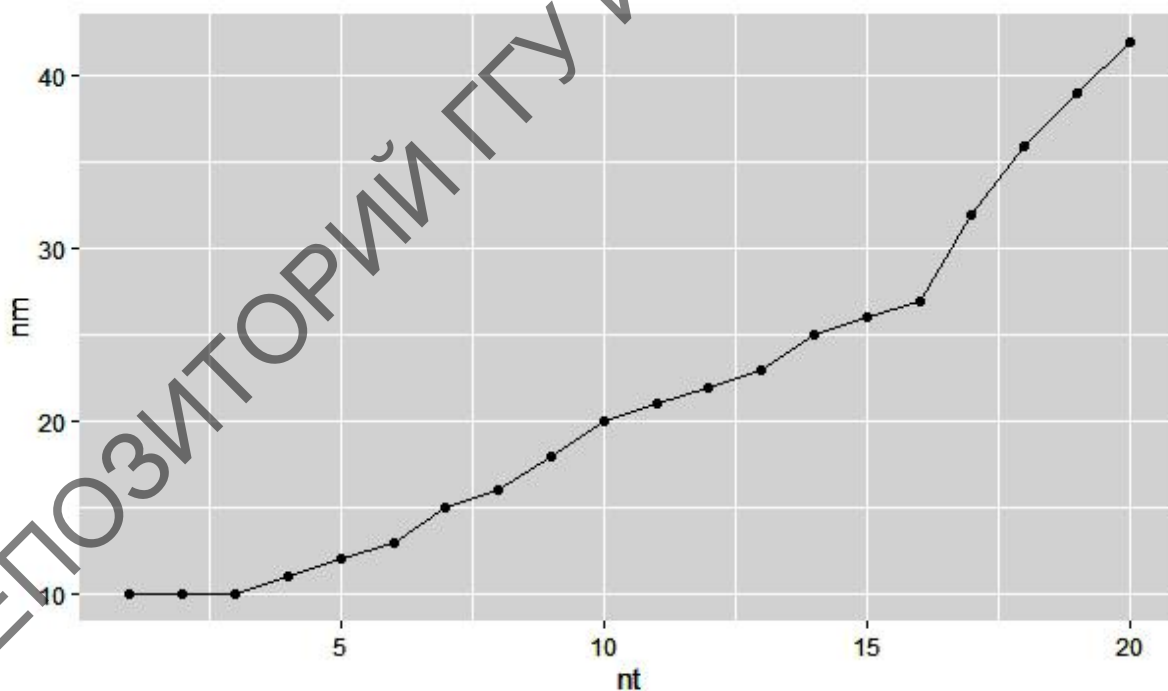


Рисунок 77 – Линейный график численности беспозвоночных с точками данных в **ggplot2**



**Шаг 8.** Создаём числовой вектор *nbr* (*number* – количество).

```
> nbr <- c(10, 10, 10, 11, 12, 13, 15, 16, 18, 20, 21, 22, 23, 25, 26, 27, 32, 36, 39, 42, 5, 5, 5, 6, 7, 7, 8, 8, 10, 11, 12, 13, 15, 17, 22, 25, 30, 35, 45, 56)
```

**Шаг 9.** Объединяем вектора в датафрейм *invow* (*invertibrate of wells* – беспозвоночные скважин).

```
> invow <- data.frame("Traps" = traps, "Place" = pls, "Number" = nbr)
```

**Шаг 10.** Создаём совмещённый график. При этом указываем, что за раскраску линий отвечает переменная **Place**.

```
> ggplot(invow, aes(x = Traps, y = Number, colour = Place)) + geom_line()
```

Полученный результат сверяем с рисунком 79

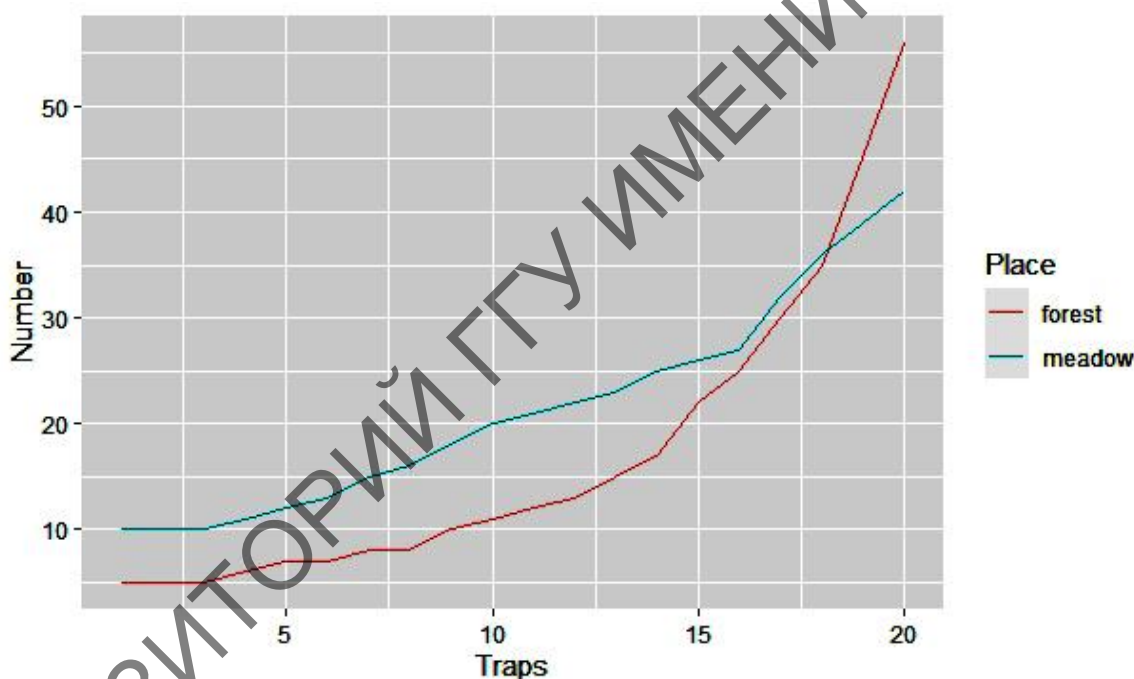


Рисунок 79 – Окрашенный и совмещённый линейный график численности беспозвоночных в **ggplot2**

Кроме цвета (особенно в том случае, когда нет возможности распечатать в цвете) можно изменить тип линии, используя вместо аргумента `colour =` аргумент `linetype =`, который задаёт вид начертания линии. Проверьте это самостоятельно.

**Шаг 11.** Добавим точки данных на наш график и зададим их размер заранее.

```
> ggplot(invw, aes(x = Traps, y = Number, colour = Place))  
+  
  geom_line()+  
  geom_point(size = 4)
```

Полученный результат сравните с рисунком 80.

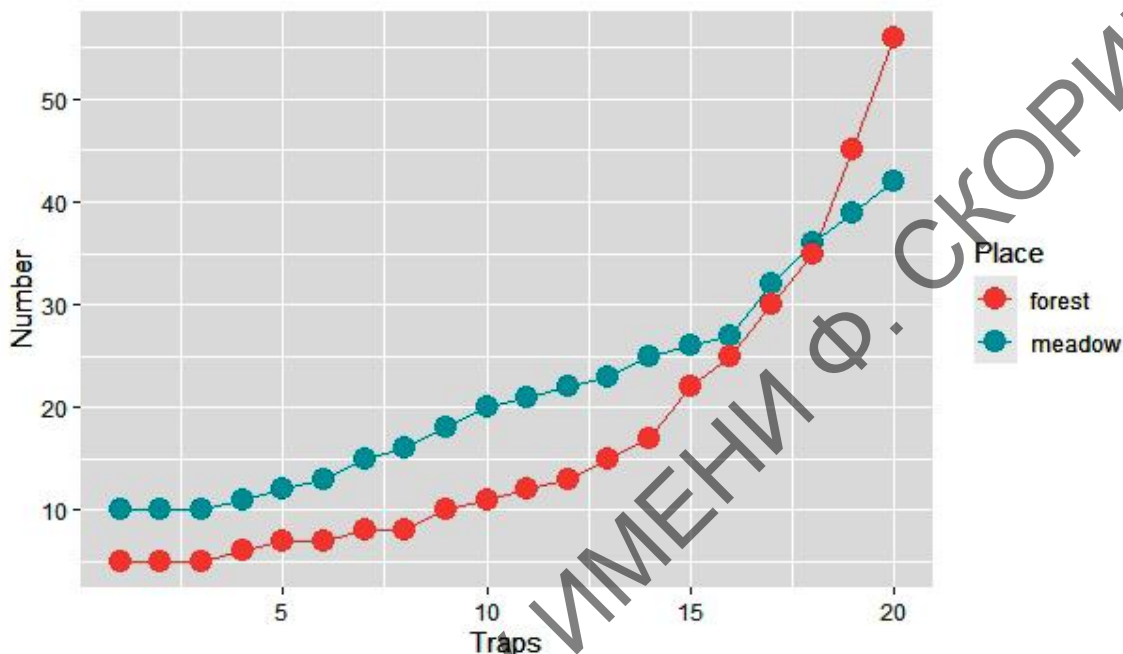


Рисунок 80 – Окрашенный с точками данных совмещённый линейный график численности беспозвоночных в **ggplot2**

Это далеко не все возможности графического пакета **ggplot2** в построении линейных графиков и, к сожалению, объём пособия не позволяет рассмотреть все подробности. Поэтому обращаем ваше внимание на список литературы в конце пособия, которую можно использовать для усовершенствования своих навыков в языке программирования R.

### 1.3.3 Построение линейного графика в RCommander (при помощи GUI)

Для построения линейного графика при помощи пакета RCommander используем те же данные, что и в предыдущих примерах по численности беспозвоночных в окрестностях нефтяных скважин (таблица 4).

**Шаг 1.** Включаем пакет RCommander.

**Шаг 2.** Загружаем данные, подготовленные при помощи электронных таблиц, либо набранные в редакторе самого RCommander.

**Шаг 3.** Для построения линейного графика необходимо перейти в меню по пути **Графики** → **Линейный график...** (рисунок 81).

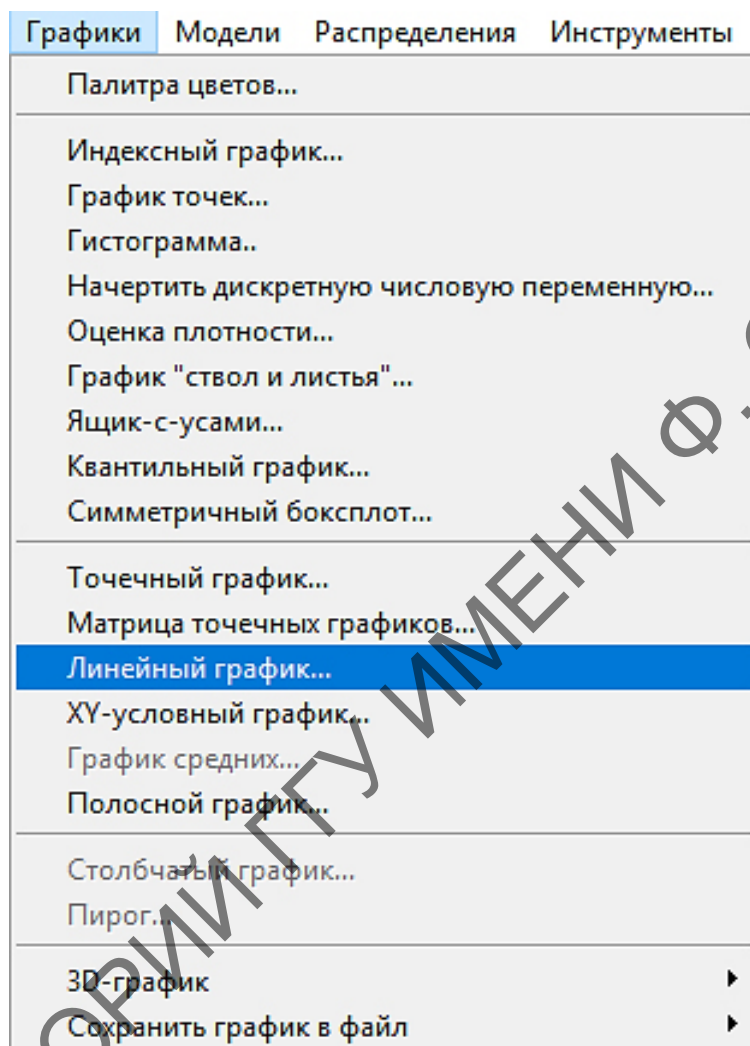


Рисунок 81 – Пункт меню **Линейный график...** в RCommander

**Шаг 4.** В открывшемся диалоговом окне **Линейный график** необходимо выбрать переменные для осей координатной плоскости. В нашем случае  $x$ -переменной будут выступать номера ловушек ( $nt$ ), а в качестве  $y$ -переменной – данные по численности на лугу ( $nw$ ).

Настройки для будущего графика сверьте с рисунком 82. После чего следует нажать **ОК**.

**Шаг 5.** Полученный график сравним с рисунком 83 и сохраним его.



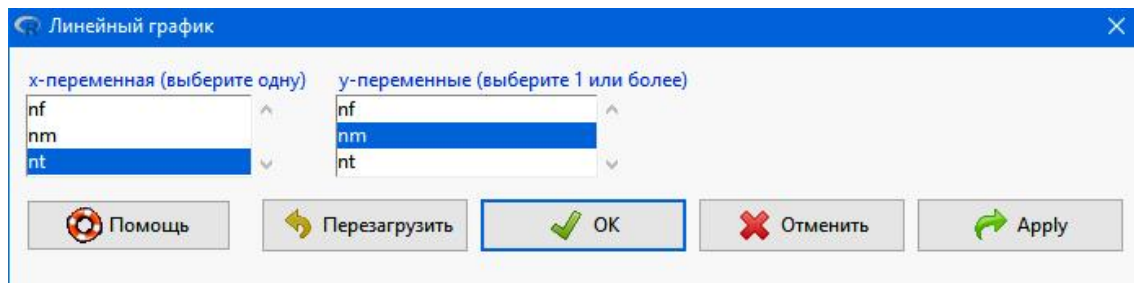


Рисунок 82 – Диалоговое окно **Линейный график** в RCommander

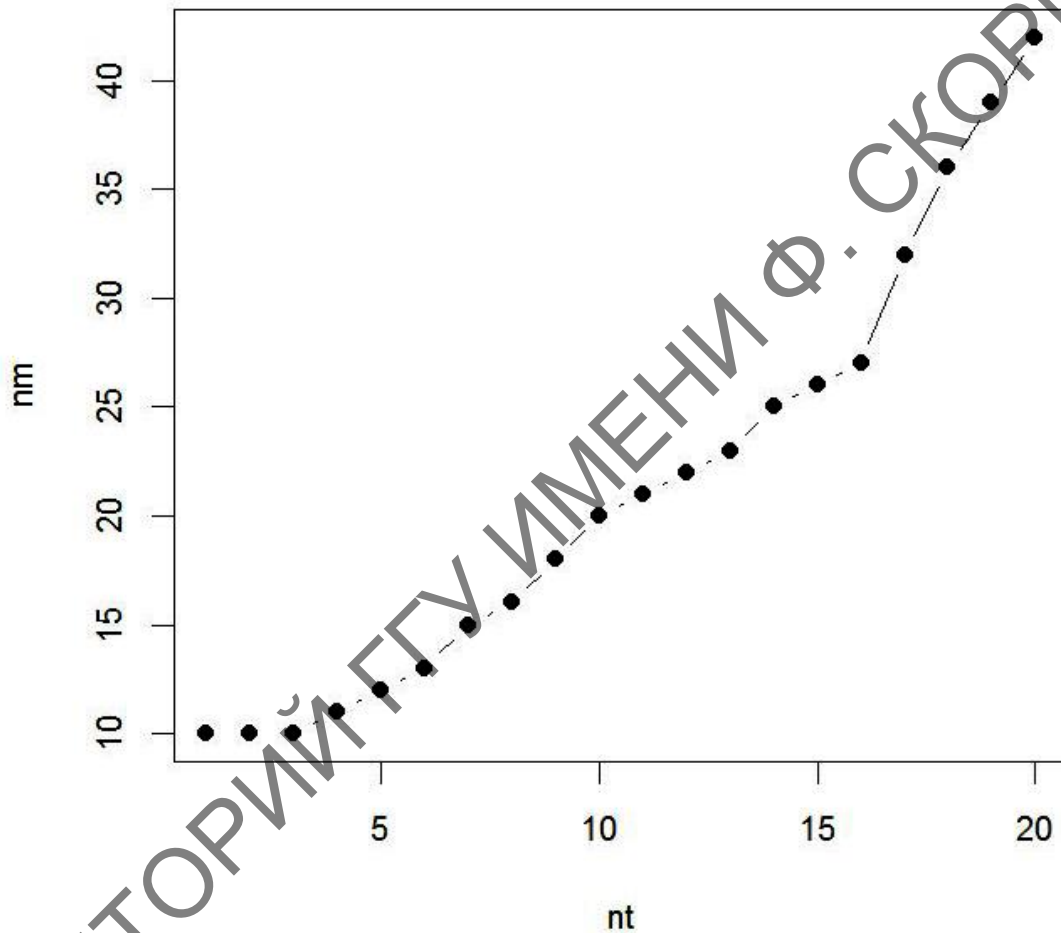


Рисунок 83 – Линейный график, созданный в RCommander

**Шаг 6.** Совместим два графика численности на одном рисунке. Для этого снова повторим путь в меню **Графики** → **Линейный график...** (рисунок 81). В диалоговом окне **Линейный график** переменную  $x$  оставим без изменений и выделим мышью в качестве  $y$ -переменной наши данные по численности беспозвоночных как в лесу, так и на лугу (рисунок 84), и нажмём **ОК**.

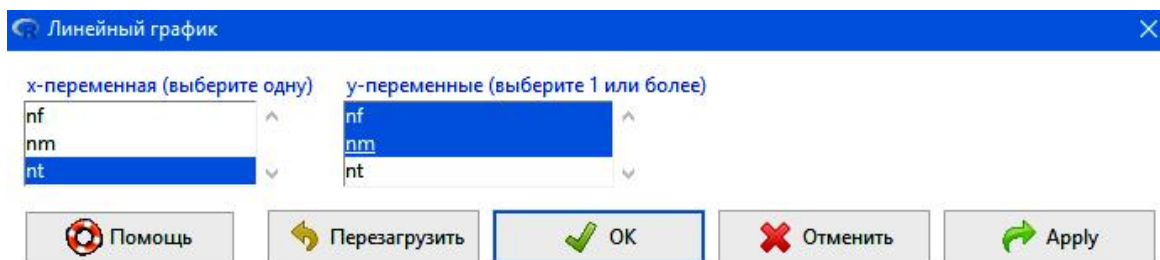


Рисунок 84 – Диалоговое окно **Линейный график** с выделенными двумя y-переменными в RCommander

**Шаг 7.** Полученный график сверим с рисунком 85.

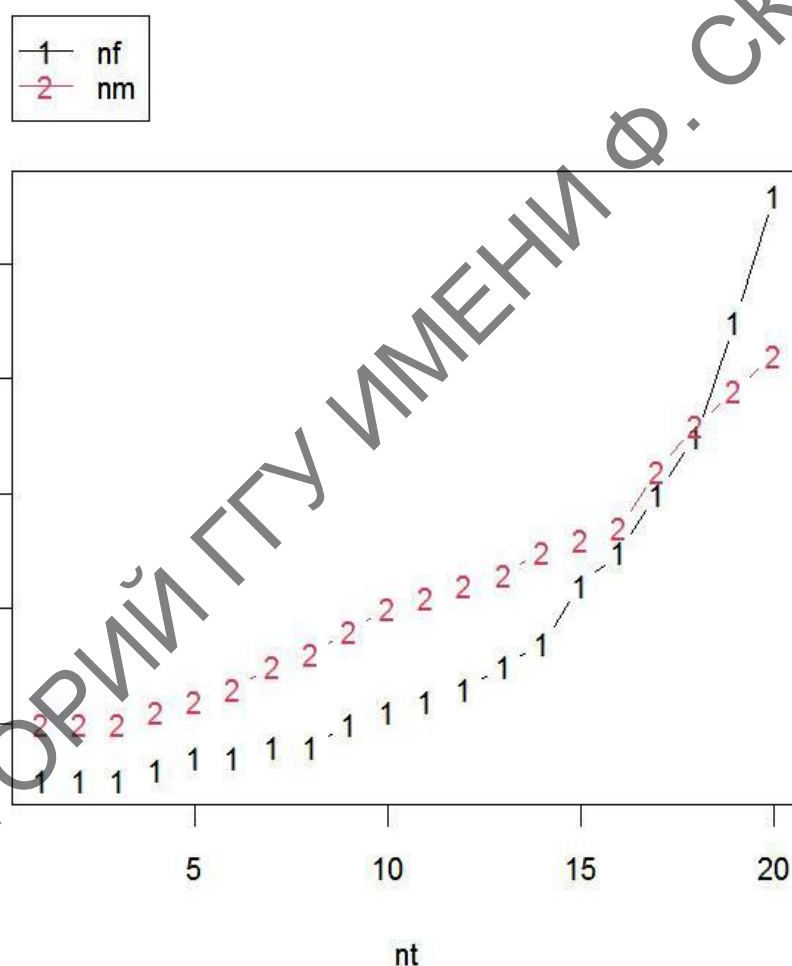


Рисунок 85 – Линейный график с двумя y-переменными, созданный в RCommander

Нетрудно заметить, что графики, созданные в RCommander, подходят лишь для первоначального анализа данных.

## 1.4 Диаграмма «ящички и усы»

Диаграмма «ящички и усы», или как её ещё называют – «боксплот» – служит для более наглядной визуализации распределения данных в выборке. Придумал её Джон Тьюки, поэтому её ещё называют диаграммой Тьюки. Это как бы взгляд на гистограмму распределения сверху. Существует несколько способов изображения боксплота в зависимости от того, что заложено в его основу: показатель средней арифметической или медианы (рисунок 86).

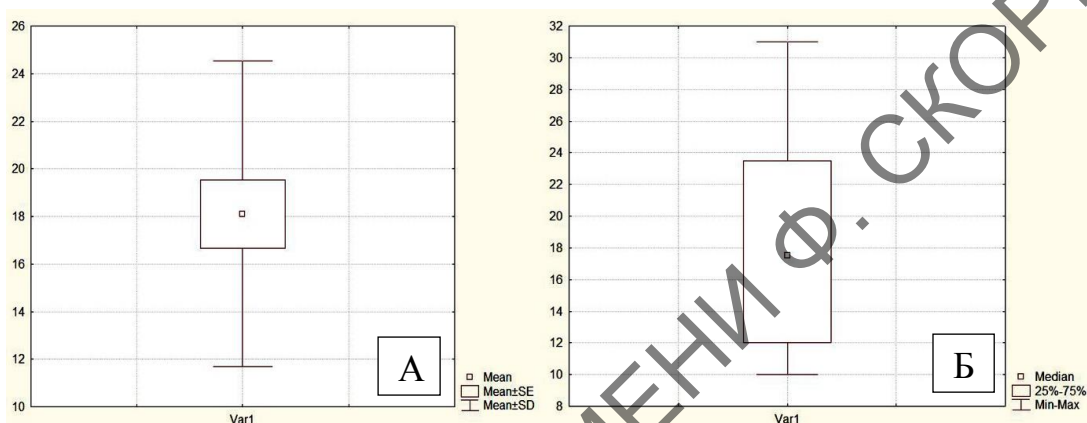


Рисунок 86 – Разные виды боксплотов: на основе средней (А) и медианы (Б)

Наиболее информативным можно назвать боксплот именно на основе более стабильной, т. е. робастной величины – медианы, который наиболее полно раскрывает особенности распределения значений переменной в выборке (рисунок 87).

Рассматривая рисунок, можно заметить, что жирная линия по центру прямоугольника (Median) – это не что иное, как положение медианы. Левая и правая боковые стороны прямоугольника, расположенные параллельно медиане, соответственно 25 и 75 перцентили, расстояние между которыми (длина прямоугольника) – это межквартильный размах (IQR), куда попадают самые значимые показатели признака. Минимальное значение выборки соответствует нижнему «усу» (Minimum), а показатель максимума (Maximum) соответствует не последнему максимальному значению признака, а на самом деле всего лишь полуторное значение межквартильного интервала, отложенного от 75 перцентиле. Особое внимание следует уделить отдельным точкам. Точки – это особый случай, так называемые выбросы. Если таких точек много, то это внесёт серьезное отклонение в распределение. Рекомендуется такие случаи удалять из выборки.

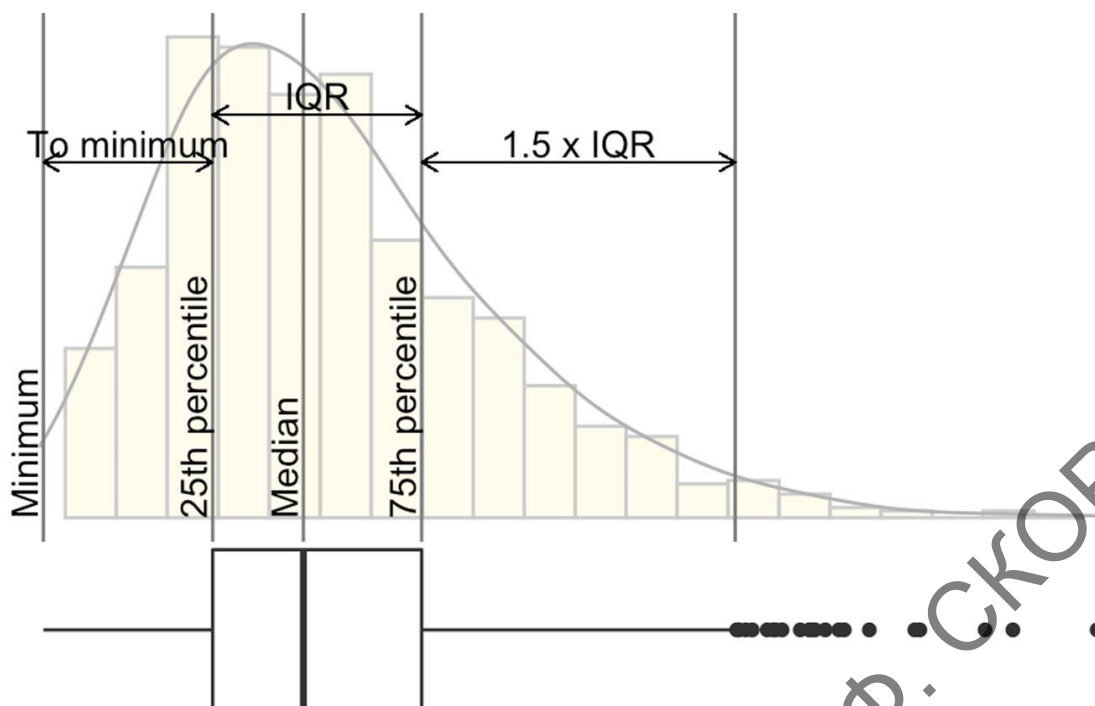


Рисунок 87 – Проекция боксплота по медиане на гистограмму распределения выборки (объяснение в тексте)

Для построения боксплотов воспользуемся для учебных целей уже готовым набором данных, включённых в базовую поставку языка программирования R. Это информация о свойствах 31 дерева по следующим параметрам: **girth** (*обхват*), **height** (*высота*) и **volume** (*объем*). Таблица данных так и называется «*trees*» (деревья). Для того чтобы вызвать включённые таблицы данных, необходимо использовать функцию `data()`. В качестве аргумента используется название таблицы с данными. В том случае, если используется функция без аргументов, то на экране монитора появится список всех таблиц, включённых в базовый набор пакетов R.

Функцией для построения боксплота служит `boxplot()`, где в качестве аргумента употребляется либо имя датафрейма (если нужно показать все переменные), либо проиндексированные переменные, либо просто имена переменных в случае прикрепленного датафрейма.

#### 1.4.1 Построение боксплота в RStudio (при помощи командной строки и базового пакета)

**Шаг 1.** Вызовем готовую таблицу *trees* базового пакета R:

```
> data(trees)
```

**Шаг 2.** «Прикрепим» таблицу, указывая программе, что мы хотим «разъединить» столбцы таблицы:

```
> attach(trees)
```

**Шаг 3.** Строим боксплоты для всех трёх показателей деревьев и сверяем полученный результат с рисунком 88:

```
> boxplot(trees)
```

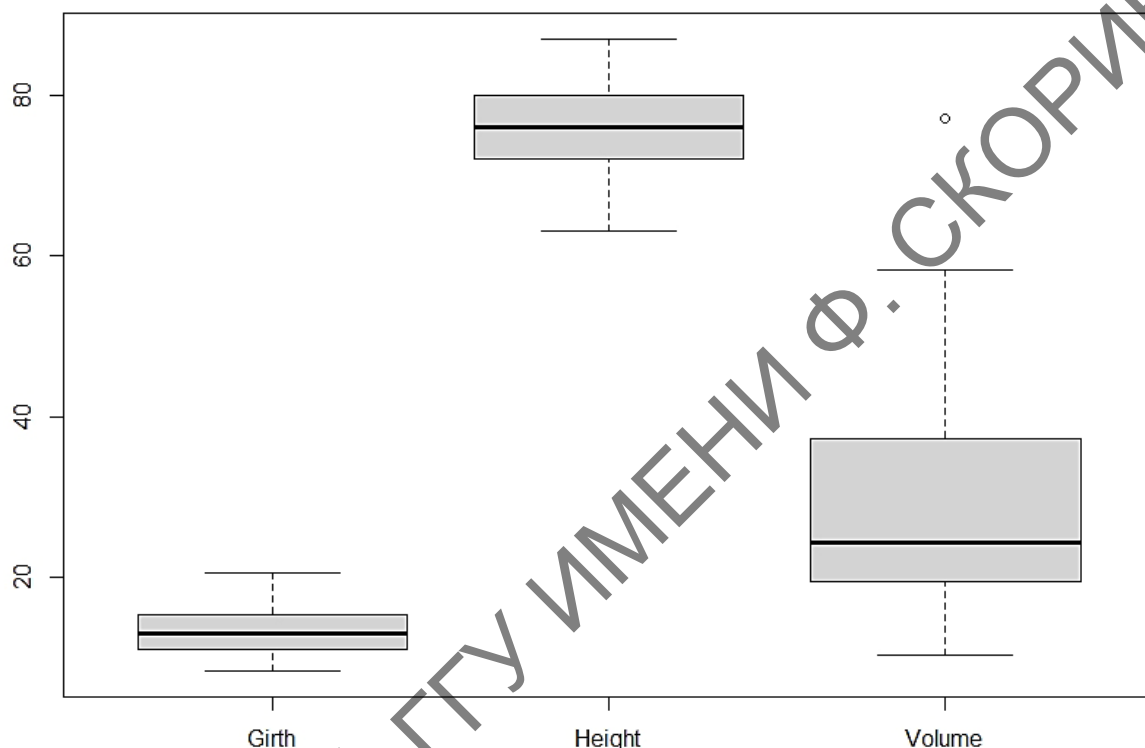


Рисунок 88 – Боксплоты переменных таблицы *trees*

Для достижения дополнительного визуального эффекта и выделения боксплота одной переменной от боксплота другой их можно раскрасить, а сам рисунок озаглавить. Сделаем это.

**Шаг 4.** Построим и раскрасим боксплоты в желтый, красный и синий цвета, а также подпишем наш рисунок:

```
> boxplot(trees, col = c("yellow", "red", "blue"), main = "Параметры деревьев в парке")
```

Полученный результат сверим с рисунком 89. Боксплоты могут быть не только прямоугольными, но и с вырезкой по краям в месте медианы.

Параметры деревьев в парке

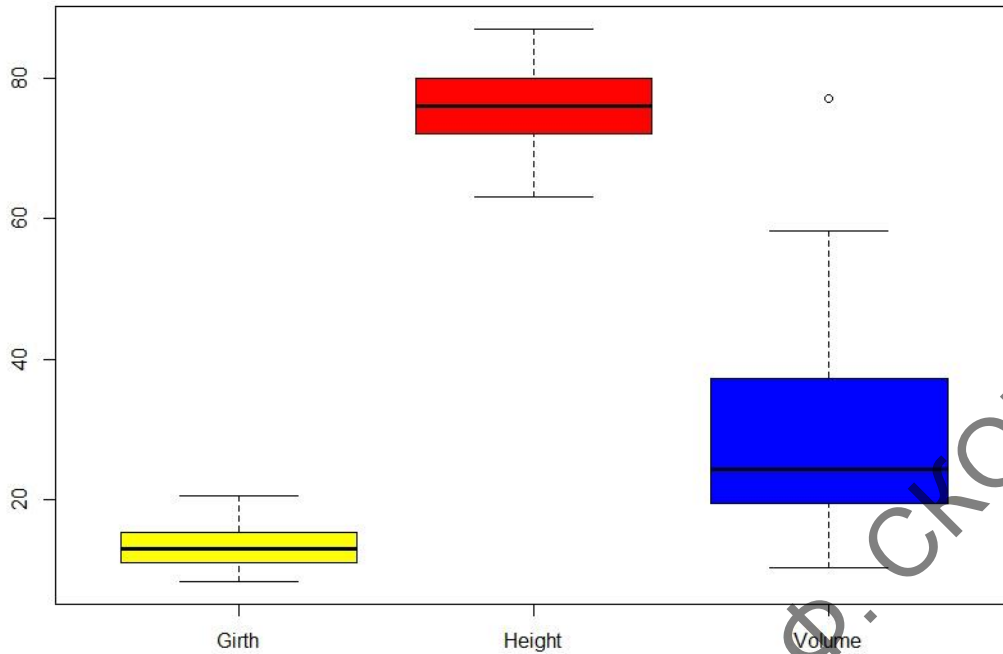


Рисунок 89 – Боксплоты переменных таблицы *trees* в цвете и с заголовком

**Шаг 5.** Для того чтобы сделать боксплоты с вырезами, необходимо использовать аргумент `notch` (выемка) равным значению `TRUE`, и сравним результат с рисунком 90:

```
> boxplot(trees, col = c("yellow", "red", "blue"), notch = TRUE, main = "Параметры деревьев в парке")
```

Параметры деревьев в парке

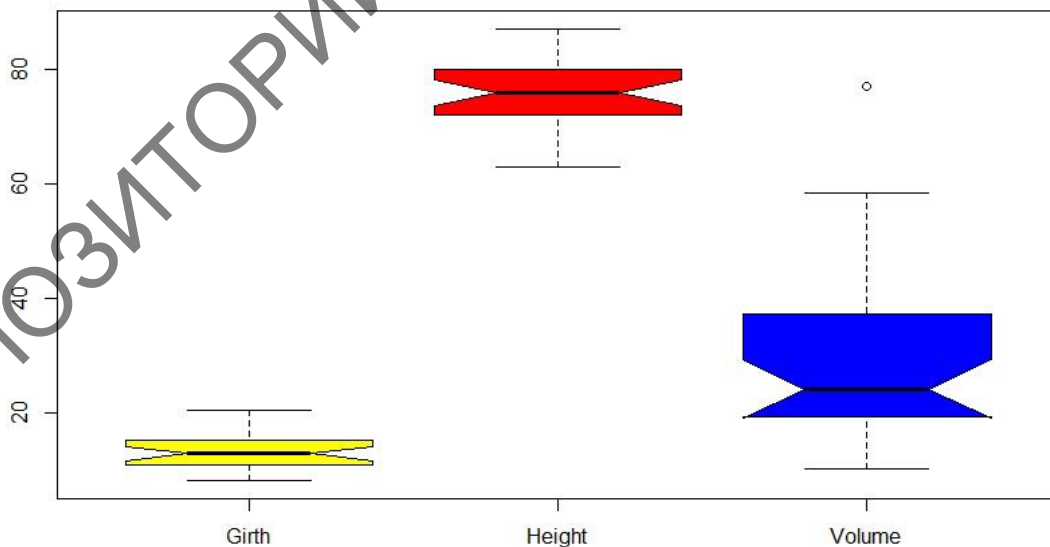


Рисунок 90 – Боксплоты с вырезом

Для того чтобы сделать боксплоты отдельных переменных датафрейма, необходимо использовать индексацию (в случае неприкрепленного датафрейма) или просто названия столбцов (в случае, если датафрейм прикреплен). Допустим, необходимо сделать боксплоты показателей **girth** (*обхват*) и **volume** (*объем*).

**Шаг 6.** Строим боксплот с индексированием и разноцветными боксами:

```
> boxplot(trees$Girth, trees$Volume, col = c("lightblue", "red"))
```

Сверим результат с рисунком 91. Попробуем построить боксплоты без индексации, так как ранее (на втором шаге) мы таблицу с данными прикрепили.

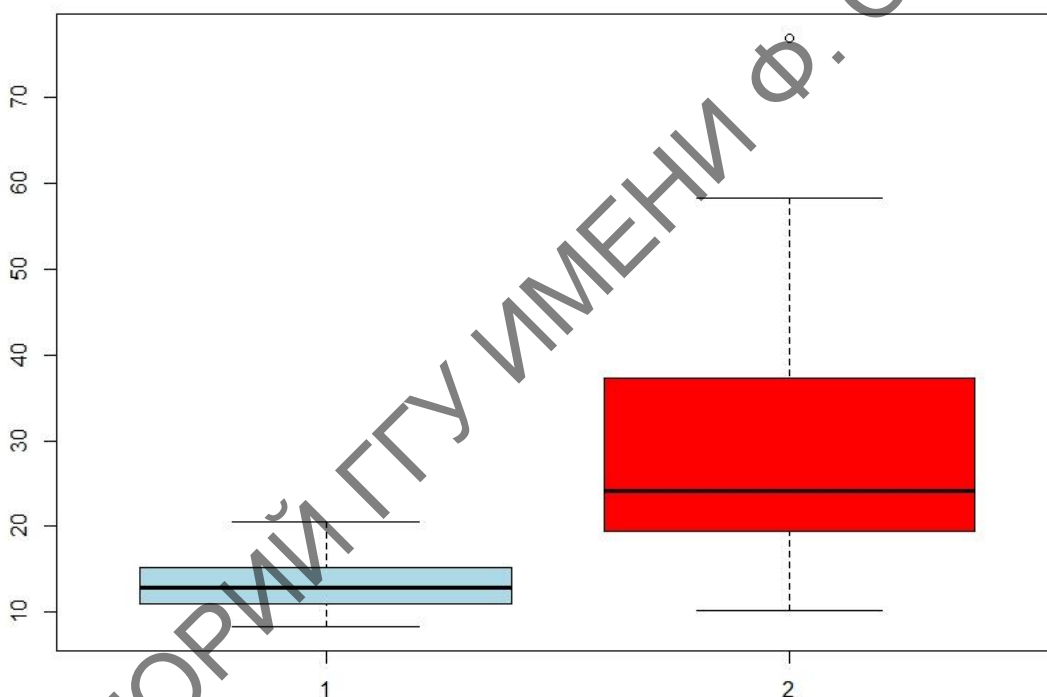


Рисунок 91 – Боксплоты переменных **Girth** (1) и **Volume** (2) таблицы *trees* в цвете

**Шаг 7.** Строим боксплот без индексирования с разноцветными боксами:

```
> boxplot(Girth, volume, col = c("lightblue", "red"))
```

Результат будет тот же, что и на рисунке 91. Не забываем открепить таблицу функцией `detach()`.

## 1.4.2 Построение боксплота в RStudio (при помощи командной строки и пакета `ggplot2`)

**Шаг 1.** Включаем пакет `ggplot2`:

```
> library (ggplot2)
```

Для данного графического пакета построение боксплота является совместным действием двух функций: `ggplot()` и `geom_boxplot()`. Первая содержит аргументы для данных, а вторая – настройки отображения боксплота (цвет бокса, цвет, форму и толщину линии, наличие бокового выреза и многое другое). Все нюансы настройки можно увидеть, ознакомившись с описанием этой функции (как, впрочем, и любой другой), используя команду «?» – `?geom_boxplot()`.

Продолжаем использовать учебную таблицу `trees` в том виде, в котором она представлена. При помощи пакета `ggplot2` нельзя отразить боксплоты трёх переменных таблицы на одном рисунке (как это сделать увидим позже), поэтому для примера построим боксплот переменной `Volume` (Объём).

**Шаг 2.** Строим стандартный боксплот с настройками по умолчанию:

```
> ggplot(trees, aes(y = volume)) +  
  geom_boxplot()
```

В аргументе `aes()` программе нужно указать по какой из осей будет ориентация боксплота. В данном случае ориентация боксплота будет вертикальной (y-ориентированной). Сверим получившийся боксплот с рисунком 92.

Графический пакет `ggplot2` позволяет достаточно гибко настроить нужный боксплот, например, покрасим боксплот в голубой цвет, линии сделаем черными и толстыми, сделаем боковую вырезку.

**Шаг 3.** Добавляем боксплоту визуальных эффектов.

```
> ggplot(trees, aes(y = volume)) +  
  geom_boxplot(fill = "lightblue", colour = "black",  
  size = 2, notch = TRUE)
```

Получившийся боксплот можно посмотреть на рисунке 93 и свериться с ним.

Для того чтобы построить сразу несколько боксплотов параметров деревьев, включённых в таблицу `trees`, её нужно перестроить таким образом, чтобы в одном столбце были показатели, а во втором – их значения.



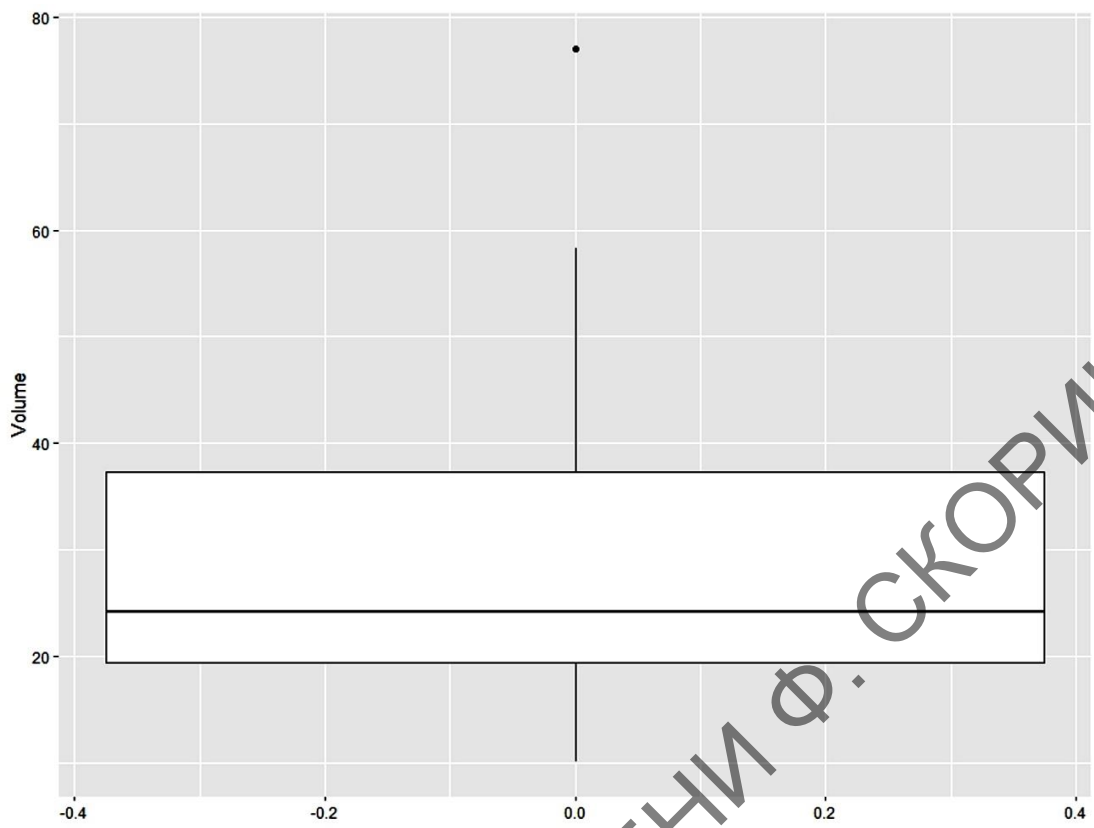


Рисунок 92 – Боксплот переменной **Volume** в **ggplot2**

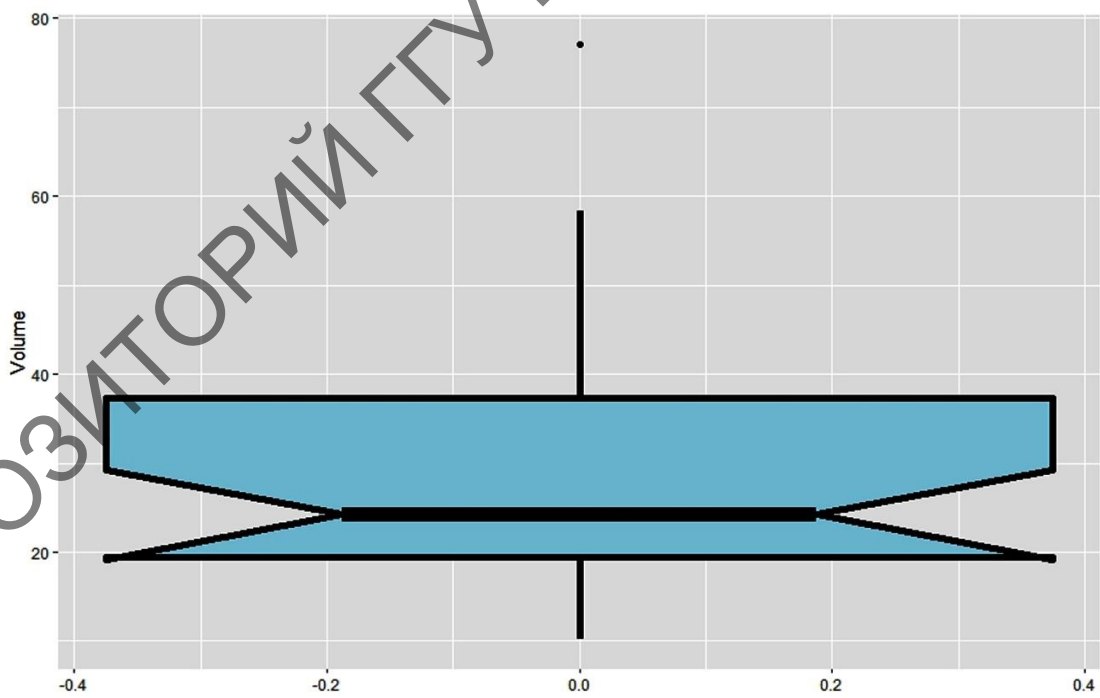


Рисунок 93 – Боксплот переменной **Volume** с заданными параметрами в **ggplot2**

**Шаг 4.** Создадим новый датафрейм *Trees* (напоминаем, что регистр в R имеет значение), который будет содержать 2 столбца: **pot** (*parameters of trees* – параметры деревьев), где перечислим параметры (*Girth*, *Height* и *Volume*) по 31 разу каждый, и значения этих параметров – **number** (количество).

```
> Trees <- data.frame(number=c(8.3, 8.6, 8.8, 10.5,
10.7, 10.8, 11.0, 11.0, 11.1, 11.2, 11.3, 11.4, 11.4,
11.7, 12.0, 12.9, 12.9, 13.3, 13.7, 13.8, 14.0, 14.2,
14.5, 16.0, 16.3, 17.3, 17.5, 17.9, 18.0, 18.0, 20.6,
70, 65, 63, 72, 81, 83, 66, 75, 80, 75, 79, 76, 76, 69,
75, 74, 85, 86, 71, 64, 78, 80, 74, 72, 77, 81, 82, 80,
80, 80, 87, 10.3, 10.3, 10.2, 16.4, 18.8, 19.7, 15.6,
18.2, 22.6, 19.9, 24.2, 21.0, 21.4, 21.3, 19.1, 22.2,
33.8, 27.4, 25.7, 24.9, 34.5, 31.7, 36.3, 38.3, 42.6,
55.4, 55.7, 58.3, 51.5, 51.0, 77.0), pot =
rep(c("Girth", "Height", "Volume"), c(31, 31, 31)))
> Trees #Проверяем полученный датафрейм
  number pot
1     8.3 Girth
2     8.6 Girth
3     8.8 Girth
4    10.5 Girth
5    10.7 Girth
6    10.8 Girth
7    11.0 Girth
8    11.0 Girth
9    11.1 Girth
10   11.2 Girth
11   11.3 Girth
12   11.4 Girth
13   11.4 Girth
14   11.7 Girth
15   12.0 Girth
16   12.9 Girth
17   12.9 Girth
18   13.3 Girth
19   13.7 Girth
20   13.8 Girth
...
82   24.9 volume
83   34.5 volume
84   31.7 volume
85   36.3 volume
86   38.3 volume
87   42.6 volume
88   55.4 volume
89   55.7 volume
90   58.3 volume
91   51.5 volume
92   51.0 volume
93   77.0 volume
```

**Шаг 5.** Строим боксплоты уже трёх параметров деревьев сразу на одной сетке, раскрасим их и зададим толщину линии.

```
> ggplot(Trees, aes(x = pot, y = number)) +  
  geom_boxplot(fill = "lightblue", colour = "black",  
  size = 1)
```

Полученный результат отражён на рисунке 94.

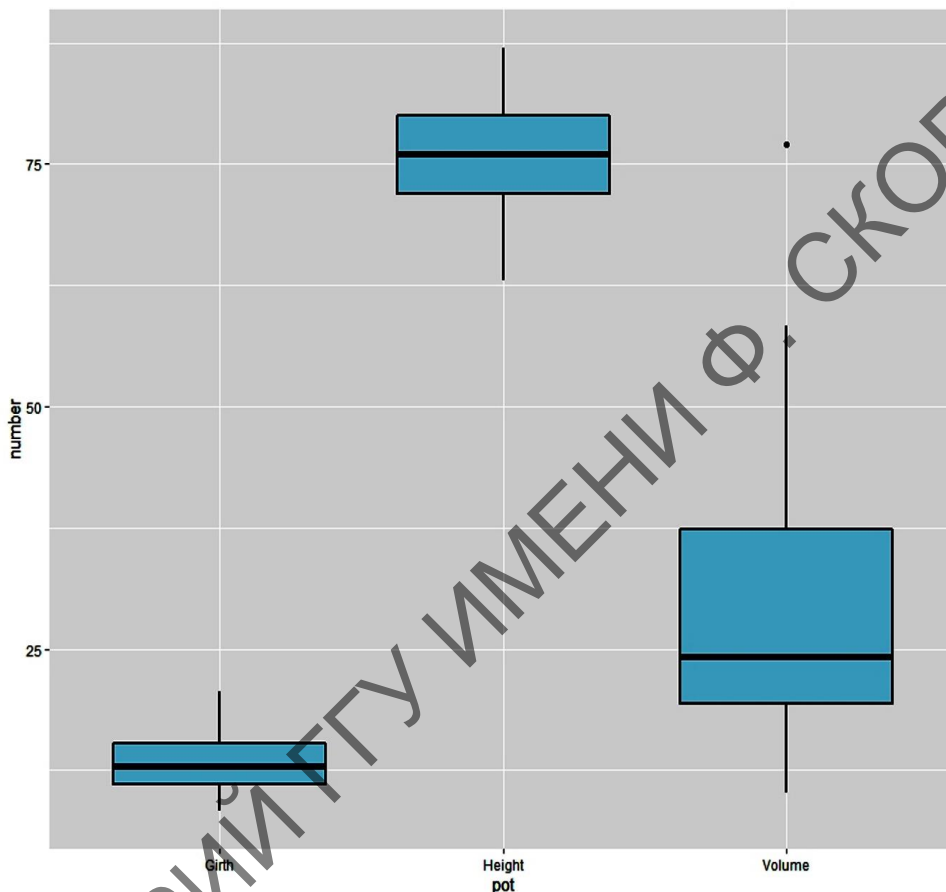


Рисунок 94 – Боксплоты переменных датафрейма *Trees* с заданными параметрами в **ggplot2**

При внимательном рассмотрении боксплотов, выполненных в графическом пакете **ggplot2**, можно заметить, что при минимальных и максимальных значениях отсутствуют поперечные черты. Это сделано намеренно, чтобы не отделять возможные выбросы.

### 1.4.3 Построение боксплотов в RCommander (при помощи GUI)

Для построения линейного графика при помощи пакета RCommander используем те же данные, что и в предыдущих примерах по свойствам деревьев в парке, т. е. набор данных *trees*.

Для этого нужно загрузить данные в RCommander из уже имеющихся пакетов.

**Шаг 1.** Для загрузки данных из имеющейся таблицы в RCommander необходимо перейти в меню по пути **Данные** → **Данные в пакетах** → **Отразить данные из пакетов** (рисунок 95).

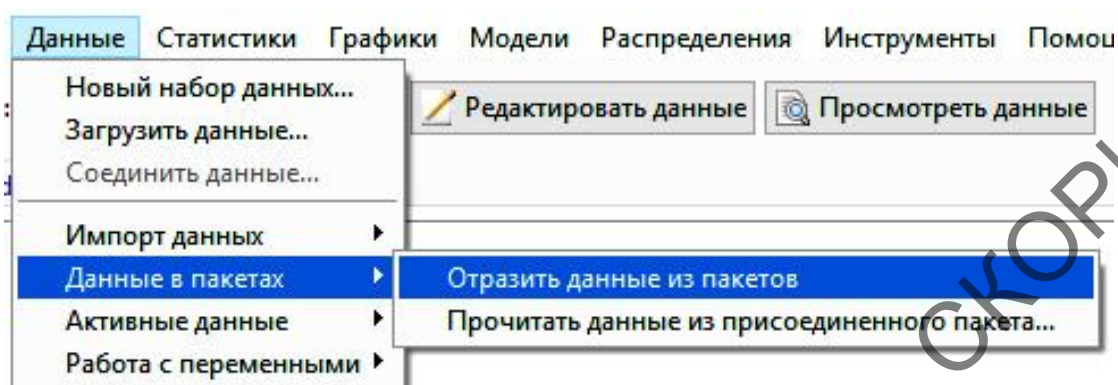


Рисунок 95 – Загрузка данных из имеющихся пакетов в RCommander

**Шаг 2.** Далее следует нажать кнопку **Данные** под строкой меню и из списка выбрать нашу таблицу *trees* (рисунок 96).

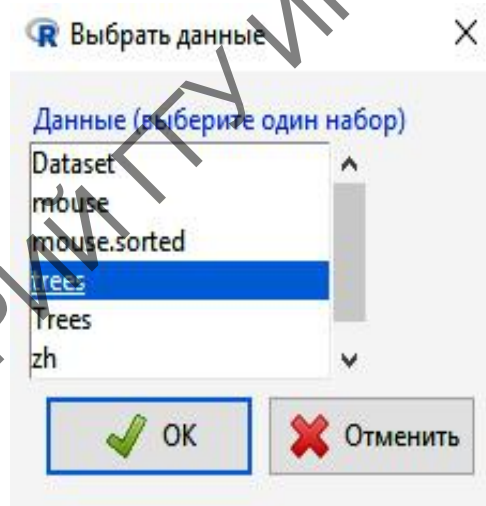


Рисунок 96 – Выбор данных из имеющихся пакетов в RCommander

После этого можно уже переходить непосредственно к построению боксплота.

**Шаг 3.** Для построения боксплота необходимо перейти в меню **Графики** → **Ящик с усами** (рисунок 97).

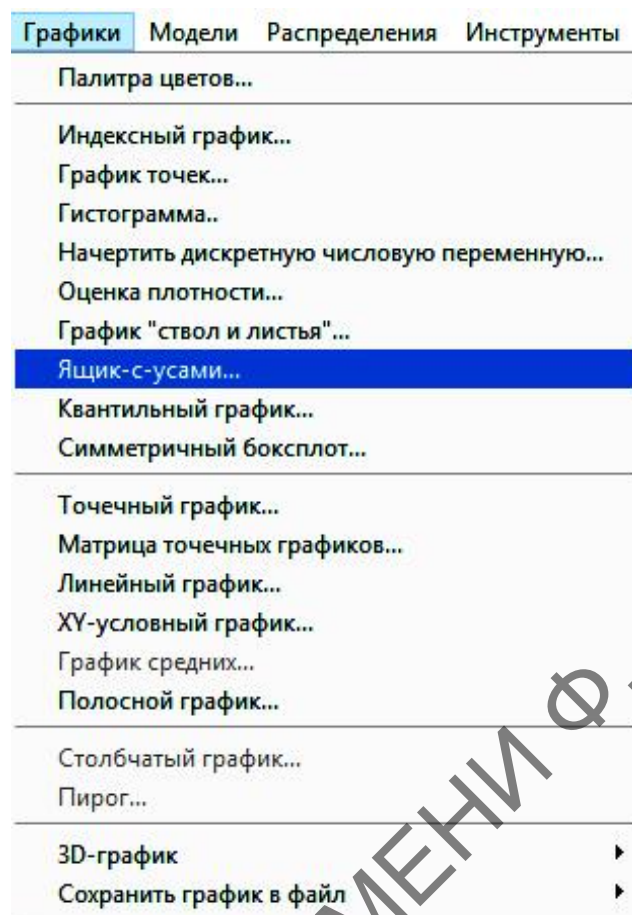


Рисунок 97 – Опция меню **Ящик с усами** в RCommander

**Шаг 4.** В появившемся диалоговом окне **Ящик с усами** перейти на закладку **Данные** и выбрать одну из переменных, например, **Girth** (рисунок 98).

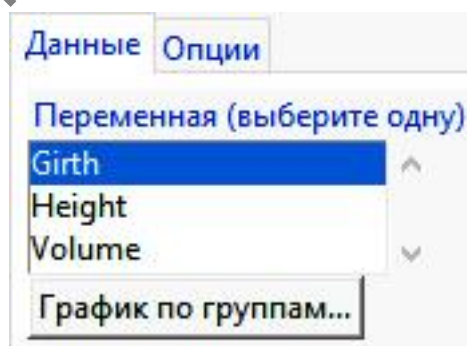


Рисунок 98 – Закладка **Данные** в диалоговом окне **Ящик с усами** в RCommander

**Шаг 5.** Далее необходимо перейти в закладку **Опции** и выбрать опции, как показано на рисунке 99, и нажать кнопку **ОК**.

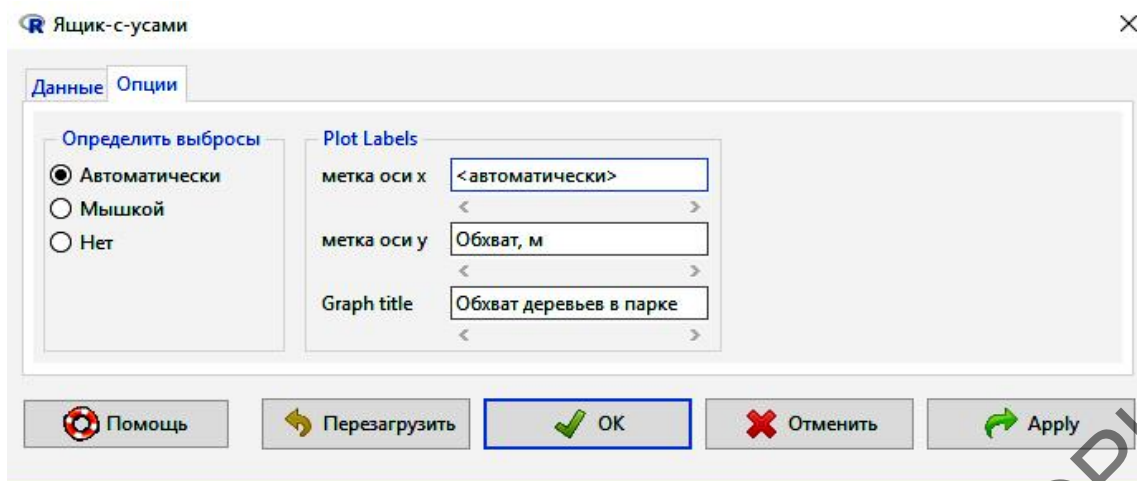


Рисунок 99 – Закладка **Опции** в диалоговом окне **Ящик с усами** в RCommander

Полученный результат сверьте с рисунком 100.

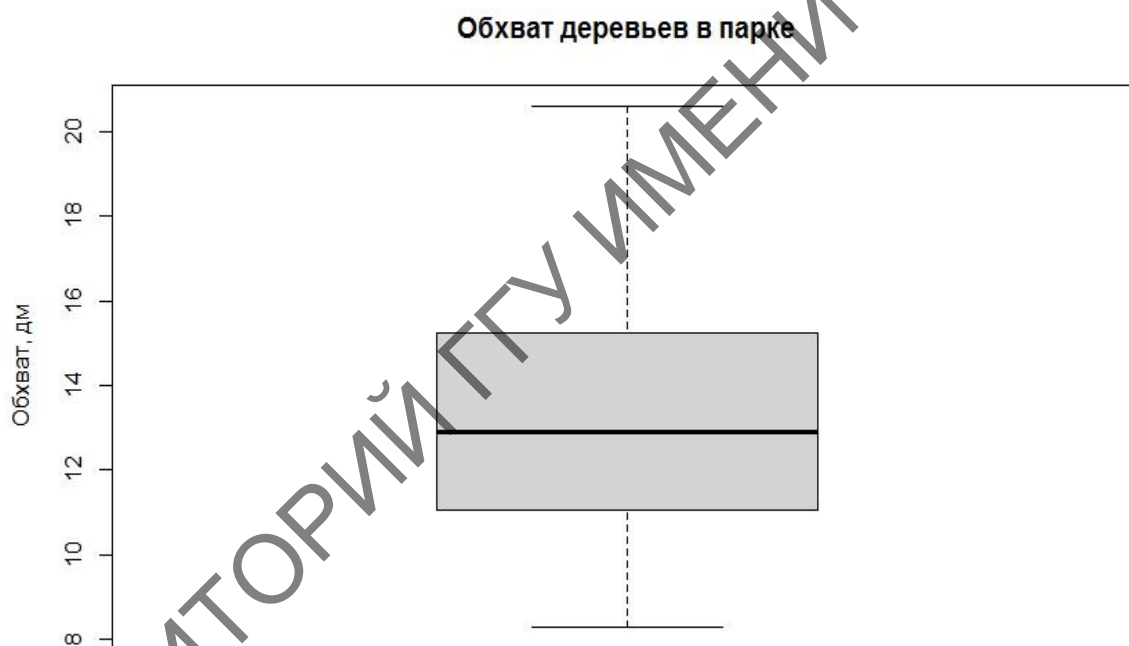


Рисунок 100 – Диаграмма **Ящик с усами**, сделанная в RCommander

Пакет RCommander также позволяет делать обобщённые графики, используя все данные таблицы. Для этого таблица с данными должна быть построена по такому же принципу, как и при обработке пакетом **ggplot2** (см. Шаг 4 подраздела 1.4.2). Загружаем ранее созданные данные таблицы *Trees* так же, как и *trees* (Шаг 1 и 2 данного подраздела).

**Шаг 6.** После загрузки данных переходим через меню *Графики* → *Ящик с усами* в диалоговое окно **Ящик с усами** на закладку **Данные** (рисунок 101).

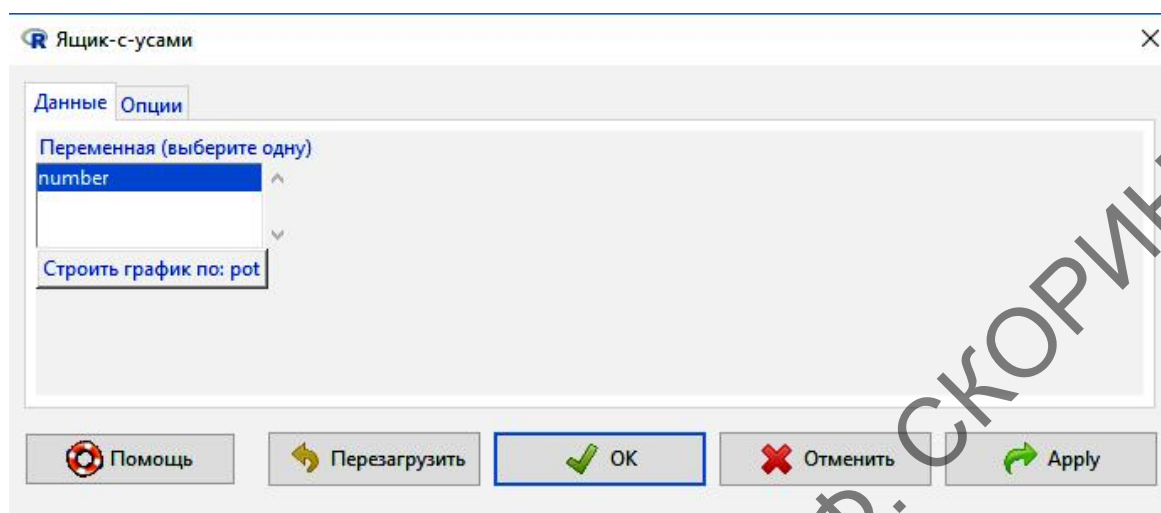


Рисунок 101 – Закладка **Данные** в диалоговом окне **Ящик с усами** в R Commander

**Шаг 7.** Нажать кнопку под боксом и в диалоговом окне **Группы** выбрать группирующую переменную (т. е. такую, которая объединяет все остальные, фактор) – в нашем случае это переменная *rot* (рисунок 102) и нажать **OK**.

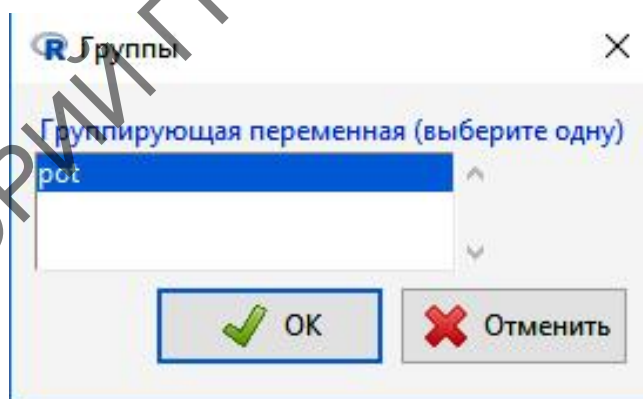


Рисунок 102 – Диалоговое окно **Данные** в R Commander

**Шаг 8.** Далее необходимо перейти на закладку **Опции** и сделать настройки так, как показано на рисунке 103, и нажать **OK**.

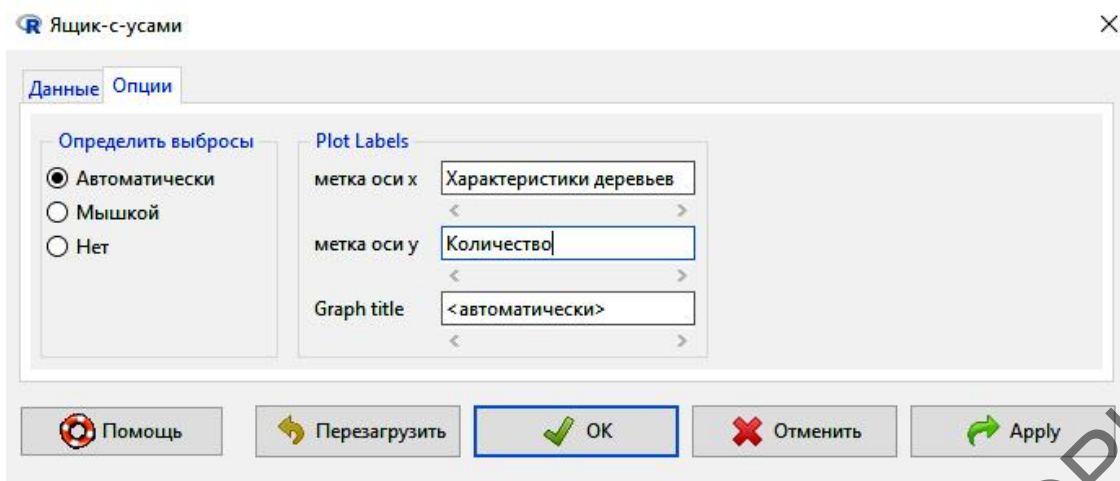


Рисунок 103 – Закладка **Опции** в диалоговом окне **Ящик с усами** в RCommander

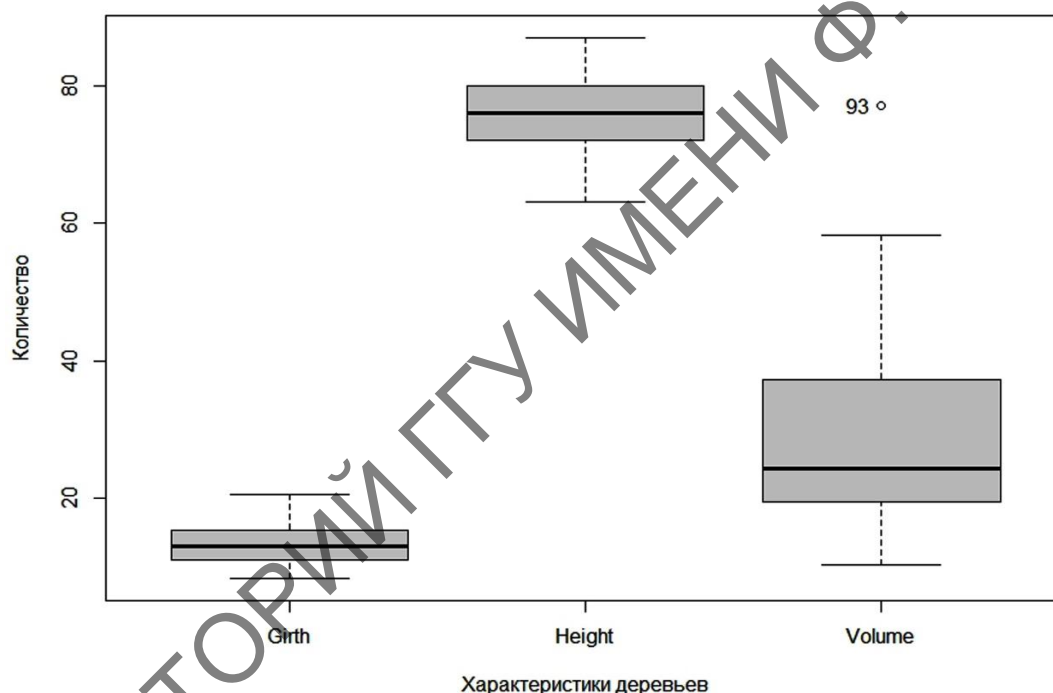


Рисунок 104 – Диаграмма **Ящик с усами** по трем характеристикам на одном листе, сделанная в RCommander

Полученный результат сравните с рисунком 104.

## 1.5 Столбчатая диаграмма

Столбчатые диаграммы, или барплоты используются довольно часто для того, чтобы сравнить несколько показателей между собой.



Они достаточно наглядны и информативны в использовании при проведении исследований и визуализации результатов.

Для создания столбчатых диаграмм в языке программирования R используется специальная функция `barplot()`. При этом следует помнить, что функция работает только при использовании в качестве рабочих данных либо матриц (см. раздел 2.1 темы 2), либо векторов.

### 1.5.1 Создание столбчатой диаграммы в RStudio (при помощи командной строки и базового пакета)

Рассмотрим построение простой столбчатой диаграммы на примере количества видов птиц, обнаруженных в различных экосистемах: сосновый лес (*pine*), еловый лес (*spruce*), смешанный лес (*mixed*), берёзовый лес (*birch*), луг (*meadow*) и сад (*garden*) (таблица 5).

Таблица 5 – Видовое богатство птиц в различных экосистемах

Экосистема	Количество видов
Сосновый лес	6
Еловый лес	5
Смешанный лес	17
Берёзовый лес	12
Луг	4
Сад	7

**Шаг 1.** Создадим символьный вектор *Biotope* (Биотоп) с названиями экосистем.

```
> Biotope <- c("Pine", "Spruce", "Mixed", "Birch",  
"Meadow", "Garden")
```

**Шаг 2.** Создадим числовой вектор *NOS* (*number of species* – количество видов) с числом видов птиц в каждой из экосистем.

```
> NOS <- c(6, 5, 17, 12, 4, 7)
```

**Шаг 3.** Объединим 2 вектора в датафрейм *birds* (птицы).

```
> birds <- data.frame("Place" = Biotope, "Number.of.species" = NOS)
```

**Шаг 4.** Построим столбчатую диаграмму численности птиц в зависимости от местообитания.

```
> barplot(NOS, names.arg = Biotope)
```

Сравните полученную диаграмму с рисунком 105.

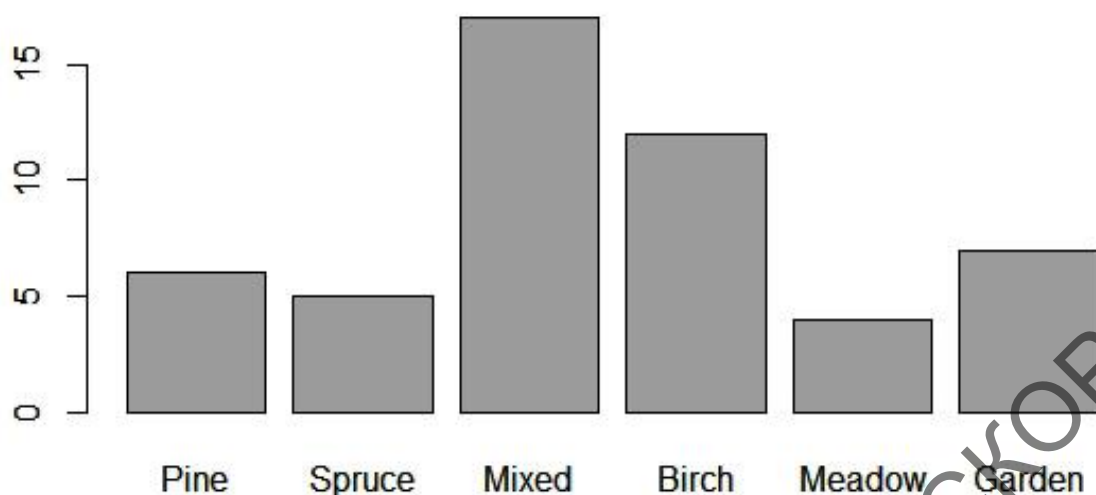


Рисунок 105 – Столбчатая диаграмма видового богатства птиц

Здесь были использованы два аргумента функции `barplot()`. Первый из них – `NOS` в качестве аргумента `height` (высота), т. е. числовой вектор, по значению которого определяется высота того или иного столбца. Второй – `names.arg` – показывает подписи из символьного вектора.

Язык программирования R даже в базовом пакете позволяет столбцы диаграммы заливать серым цветом, использовать штриховку под определенным углом, а также подписать оси определённым образом.

**Шаг 5.** Заштрихуем столбцы диаграммы под углом 30 градусов, подпишем оси диаграммы и сверим полученный результат с рисунком 106.

```
> barplot(NOS, density = 20, angle = 30, ylab = "число видов", xlab = "Экосистема", names.arg = Biotope)
```

Здесь в качестве аргументов выступают: `density` – указывает плотность штриховки; `angle` – угол наклона линий штриховки в градусах (при положительном значении наклон штриховки ориентирован вправо, при отрицательном значении – влево); `ylab` – подпись оси ординат; `xlab` – подпись оси абсцисс.

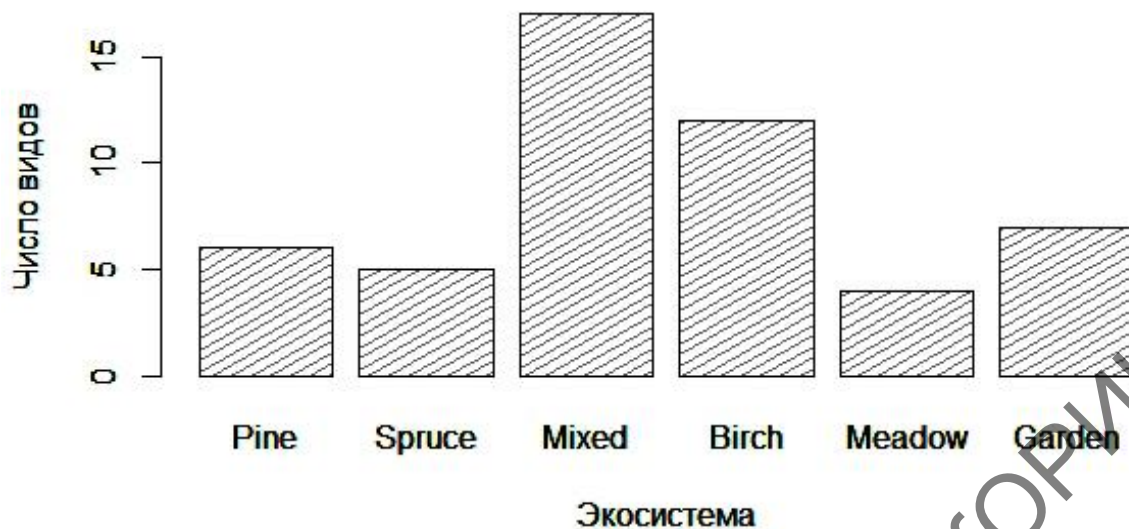


Рисунок 106 – Столбчатая диаграмма видового богатства птиц со штриховкой

В том случае, если предполагается цветное изображение столбцов, то при помощи аргумента можно залить цветом столбики диаграммы, а используя аргумент `border` – окрасить их границу.

**Шаг 6.** Закрасим столбцы диаграммы в голубой цвет и сделаем границу красной (рисунок 107).

```
> barplot(NOS, col = "lightblue", border = "red", ylab = "число видов", xlab = "Экосистема", names.arg = Biotope)
```

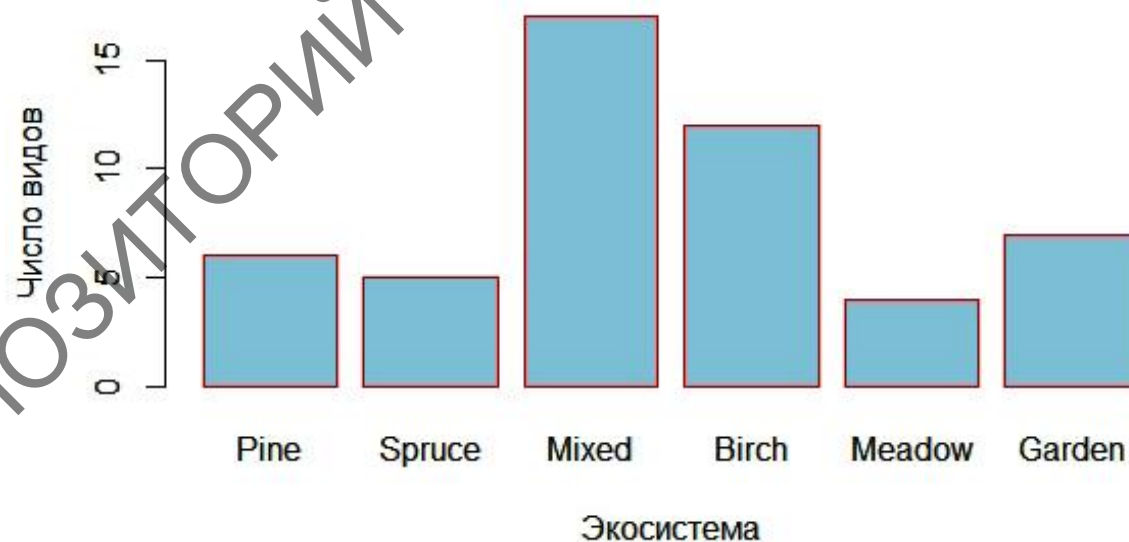


Рисунок 107 – Столбчатая диаграмма видового богатства птиц с использованием цвета

Во время исследований возникают ситуации, когда нужно рядом расположить данные, полученные в разные этапы эксперимента для сравнения. В нашем случае нужно сравнить количество видов птиц, полученные в 2019 и в 2020 годах. Рабочая таблица должна в таком случае иметь следующий вид (таблица 6).

Таблица 6 – Видовое богатство исследованных экосистем в разные годы

Биотоп	Число видов	Год
Сосновый лес	6	2019
Сосновый лес	7	2020
Еловый лес	5	2019
Еловый лес	3	2020
Смешанный лес	17	2019
Смешанный лес	20	2020
Берёзовый лес	12	2019
Берёзовый лес	11	2020
Луг	4	2019
Луг	4	2020
Сад	7	2019
Сад	10	2020

**Шаг 7.** Формируем новый символьный вектор *Biotope*.

```
> Biotope <- c("Pine", "Pine", "Spruce", "Spruce",
"mixed", "mixed", "Birch", "Birch", "Meadow", "Meadow",
"Garden", "Garden")
```

**Шаг 8.** Создаём числовой вектор *NOS*.

```
> NOS <- c(6, 7, 5, 3, 17, 20, 12, 11, 4, 4, 7, 10)
```

**Шаг 9.** Создаём символьный вектор *Year*.

```
> Year <- c("2019", "2020", "2019", "2020", "2019",
"2020", "2019", "2020", "2019", "2020", "2019", "2020")
```

**Шаг 10.** Объединяем все вектора в датафрейм *Birds*.

```
> Birds <- data.frame("Биотоп" = Biotope, "число видов"
= NOS, "Год" = Year)
```

**Шаг 11.** Создаём общую столбчатую диаграмму по исследованиям за 2 года.

```
> barplot(NOS, beside = TRUE, col = topo.colors(2),
names.arg = Biotope, xlab = "Экосистема", ylab = "число
видов")
```

Здесь используются аргументы:

– говорит, что столбцы по годам в каждой экосистеме нужно сгруппировать;

– указывает, что столбцы по годам должны быть раскрашены в 2 цвета (рисунок 108).

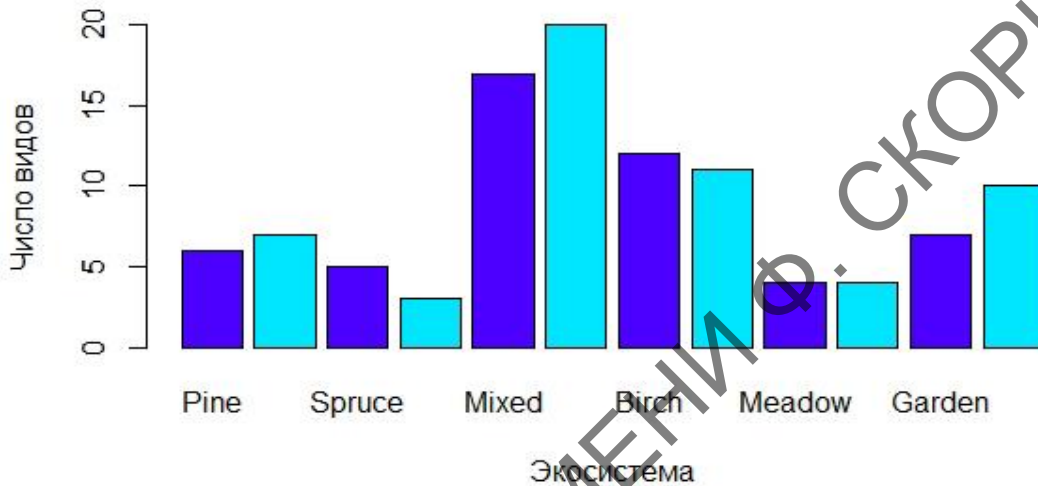


Рисунок 108 – Сгруппированная столбчатая диаграмма видового богатства птиц с использованием цвета в разные годы исследований

Встречается необходимость построить столбчатую диаграмму, где столбцы представляют собой среднее значение параметра, а также отобразить своеобразные «усы» – стандартную ошибку, или чаще – стандартное отклонение. Для учебных целей используем данные по численности птиц в уже известных нам экосистемах за 10 учётов в каждом (таблица 7).

Таблица 7 – Данные учётов численности птиц в различных экосистемах

Экосистема	Учёты									
P	25	30	20	35	25	30	25	20	20	35
S	20	20	15	30	20	25	45	25	20	20
M	105	100	90	95	80	105	110	85	85	90
B	60	55	65	55	55	50	70	60	60	55
Md	20	25	20	30	20	25	25	20	30	20
G	35	35	45	40	40	35	30	45	35	40

Примечание: P – Pine (сосновый лес), S – Spruce (еловый лес), M – Mixed (смешанный лес), B – Birch (берёзовый лес), Md – Meadow (луг), G – Garden (сад)

**Шаг 12.** Создаём рабочую таблицу *Pticy* в программе создания электронных таблиц или датафрейм с двумя переменными: *Ecosys* (*ecosystems* – экосистемы) и *Number* (количество). Переменная *Ecosys* должна будет содержать 60 значений (по 10 на каждую экосистему):

```
Ecosys Number
P          25
P          30
P          20
P          35
P          25
P          30
P          25
P          20
P          20
P          35
...
# ... with 50 more rows
```

**Шаг 13.** Рассчитываем среднее значение численности птиц для каждого вида экосистем в единой таблице при помощи функций *with()* и *tapply()*.

```
> means = with(Pticy, tapply(Number, list(Ecosys),
mean))
```

**Шаг 14.** Вводим дополнительную переменную *wsk* (*wiskers* – усы) для расчёта стандартного отклонения численности птиц для каждого вида экосистем в единой таблице также при помощи функций *with()* и *tapply()*.

```
> wsk = with(Pticy, tapply(Number, list(Ecosys), sd))
```

**Шаг 15.** Вводим промежуточную переменную *P*, которой присваиваем значение построения столбчатой диаграммы средних значений численности птиц.

```
> P <- barplot(means, ylim = c(min(pretty(means-wsk)),
max(pretty(means+wsk))), col = topo.colors(6), xpd =
FALSE, xlab = "Экосистема", ylab = "число видов")
```

**Шаг 16.** Размещаем на диаграмме «усы» при помощи функции *arrows()*.

```
> arrows(P, means+wsk, P, means-wsk, angle = 90, code =
3, length = 0.05)
```

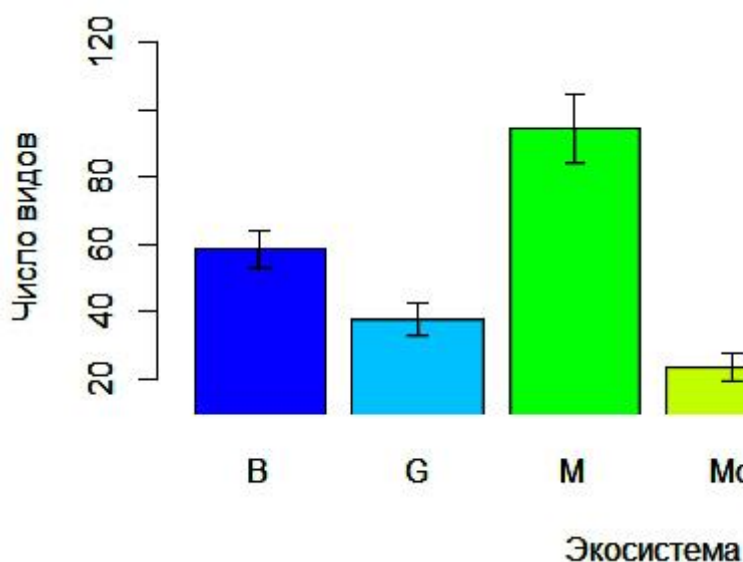


Рисунок 109 – Столбчатая диаграмма средней численности птиц с отображением стандартного отклонения

Полученный результат можно увидеть на рисунке 109. Следует обратить внимание на аргумент `topo.colors()` функции `col =`, который автоматически подбирает и распределяет цвета по указанному количеству столбиков диаграммы.

### 1.5.2 Построение столбчатой диаграммы в RStudio (при помощи командной строки и пакета `ggplot2`)

Для работы со столбчатыми диаграммами в графическом пакете `ggplot2` используются две функции. Основная `ggplot()`, которая ориентирует столбики диаграммы, и дополнительная `geom_col()`, которая отвечает за общее оформление столбиков.

Для построения простого барплота используем ранее созданный датафрейм `birds`, основанный на данных таблицы 5.

**Шаг 1.** Включаем пакет `ggplot2`.

```
> library (ggplot2)
```

**Шаг 2.** Строим простой барplot на основе датафрейма `birds`.

```
> ggplot(birds, aes(x = biot, y = nos)) +  
  geom_col()
```

Полученный результат можно сверить с рисунком 110.

Столбцы диаграммы можно раскрасить, в том числе и разными способами: простым цветом и заливкой разной интенсивности в зависимости от величины признака.

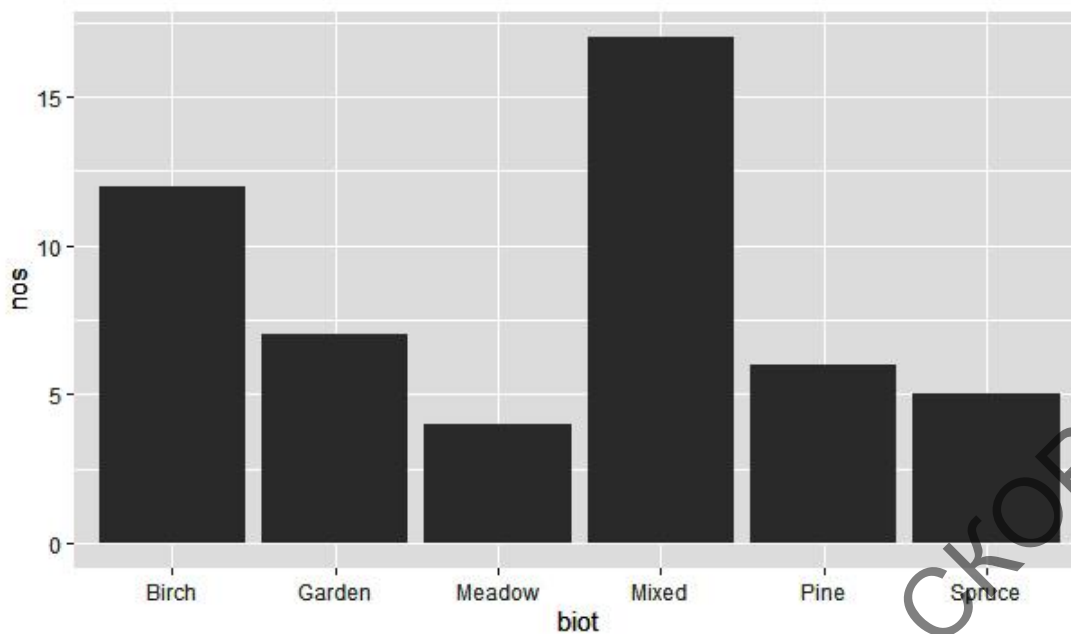


Рисунок 110 – Столбчатая диаграмма видового богатства птиц, выполненная в **ggplot2**

**Шаг 3.** Закрасим столбцы диаграммы, например, красным цветом, и сверим результат с рисунком 111.

```
> ggplot(birds, aes(x = biot, y = nos, fill = "red")) +  
  geom_col()
```

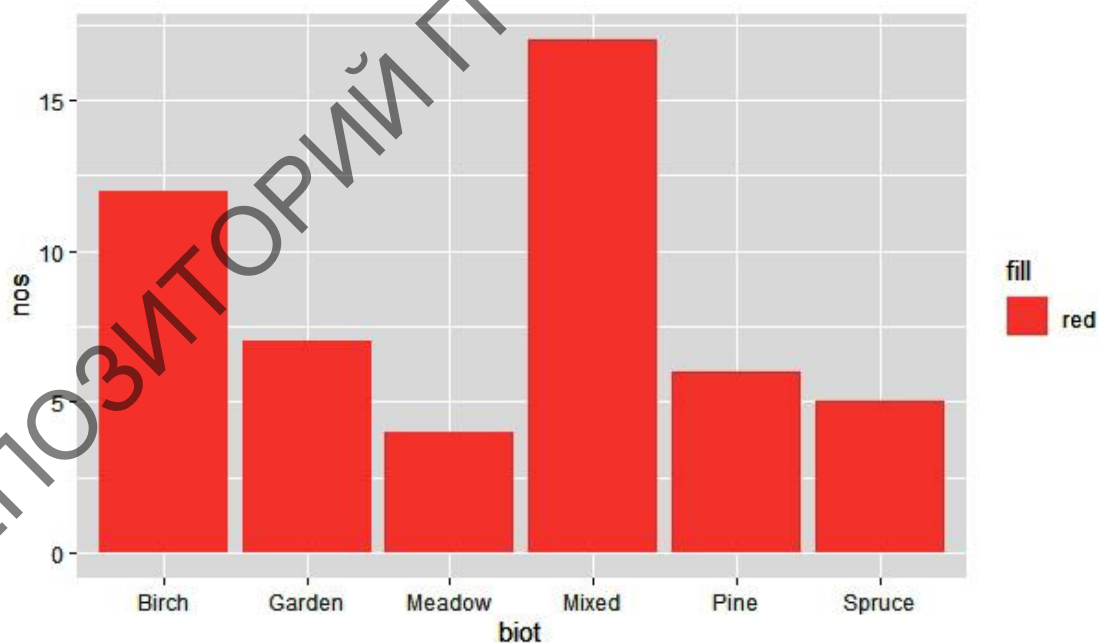


Рисунок 111 – Столбчатая диаграмма видового богатства птиц с окрашенными столбцами, выполненная в **ggplot2**



**Шаг 4.** Закрасим столбцы диаграммы цветом с разной интенсивностью в зависимости от количества видов и сверим результат с рисунком 112.

```
> ggplot(birds, aes(x = biot, y = nos, fill = nos)) +  
  geom_col()
```

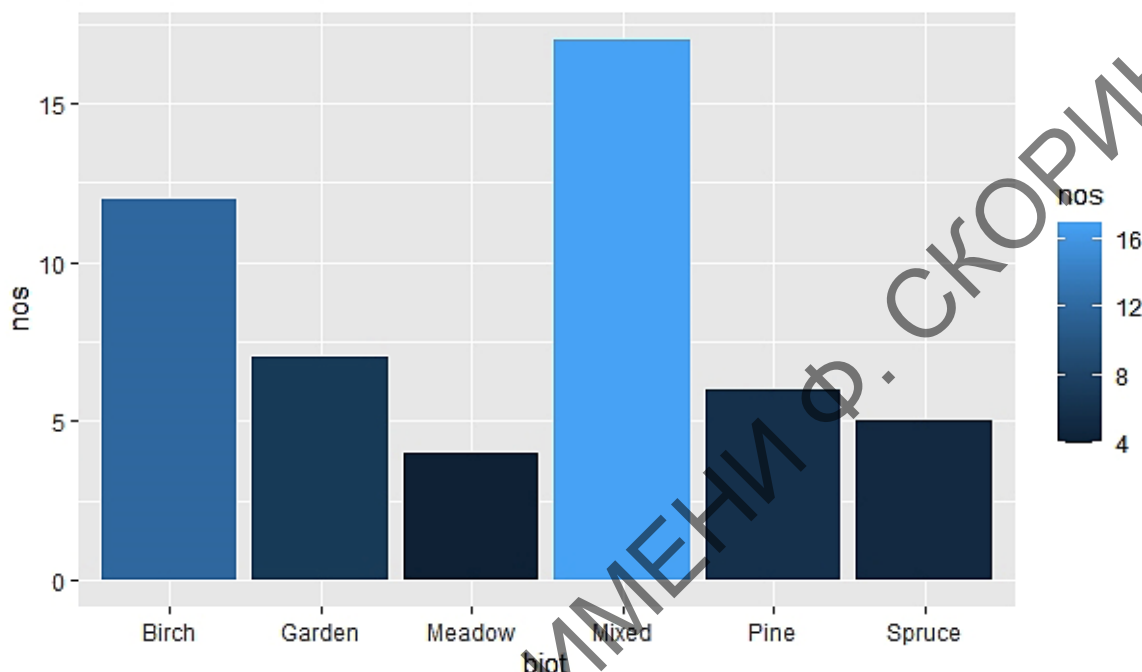


Рисунок 112 – Столбчатая диаграмма видового богатства птиц с окрашенными столбцами разной интенсивности, выполненная в **ggplot2**

В графическом пакете **ggplot2** также можно производить группирование столбцов. Сформируем барплот, основанный на датафрейме *Birds*, созданный на основе данных таблицы 6.

**Шаг 5.** Группируем столбцы столбчатой диаграммы по годам для разных мест учета. В этом случае необходимо указать переменную датафрейма, по которой будет производиться группирование (в нашем случае – *year*).

```
> ggplot(Birds, aes(x = Biotope, y = NOS, fill = Year)) +  
  geom_col(position = "dodge")
```

Полученный результат сравним с рисунком 113. Обращаем внимание на аргумент `position = "dodge"` функции `geom_col`, который говорит программе, что столбцы должны как бы уклоняться друг от друга по горизонтали, иначе вместо барплота получится график с гистограммой.

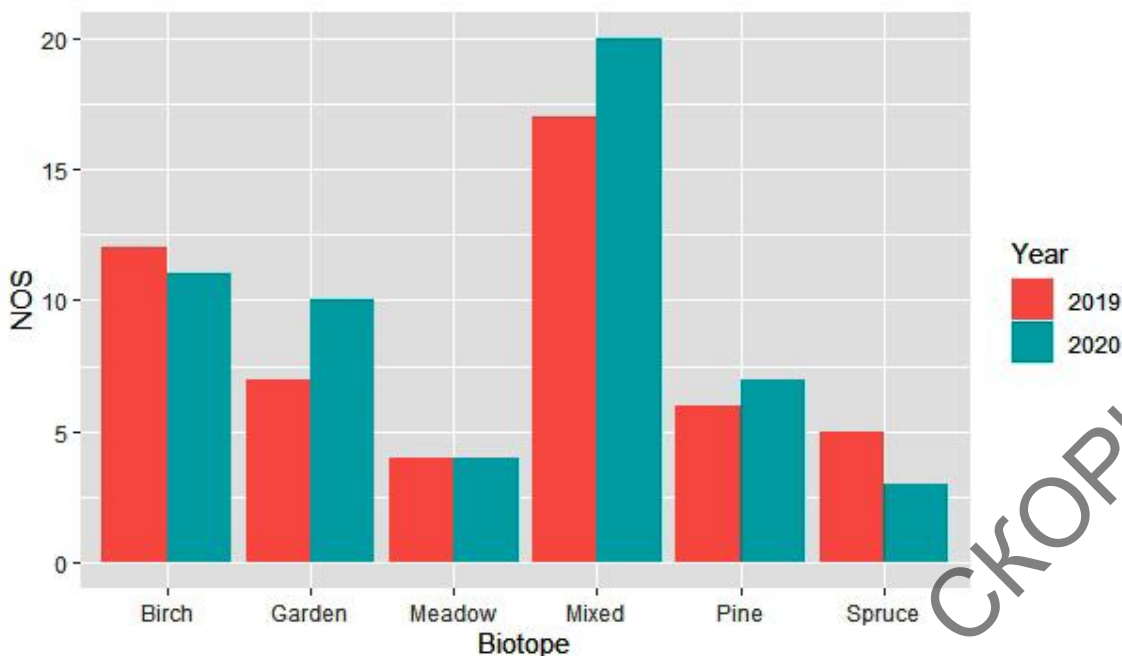


Рисунок 113 – Столбчатая диаграмма видового богатства птиц со сгруппированными столбцами в зависимости от года наблюдений, выполненная в **ggplot2**

Используя графический пакет **ggplot2** можно также отобразить конкретные значения каждого из столбцов диаграммы как снаружи от них, так и внутри столбцов.

**Шаг 6.** Отобразим значения столбцов видового богатства птиц (таблица 5, датафрейм *birds*) в качестве примера.

```
> ggplot(birds, aes(x = biot, y = nos)) +
  geom_col() +
  geom_text(aes(label = nos), vjust = -0.2)
```

Результат сравните с рисунком 114. Обратите внимание, что аргумент `vjust` имеет отрицательное значение, т. е. цифры будут расположены вне площади столбца диаграммы. Для того чтобы он оказался внутри, его нужно изменить на положительный и залить цифры белым цветом, чтобы они были видны на сером фоне.

**Шаг 7.** Отобразим значения столбцов видового богатства птиц (таблица 5, датафрейм *birds*) внутри столбцов и белым цветом.

```
> ggplot(birds, aes(x = biot, y = nos)) +
  geom_col() +
  geom_text(aes(label = nos), vjust = 1.5, colour =
    "white")
```

Результат можно сверить с рисунком 115.

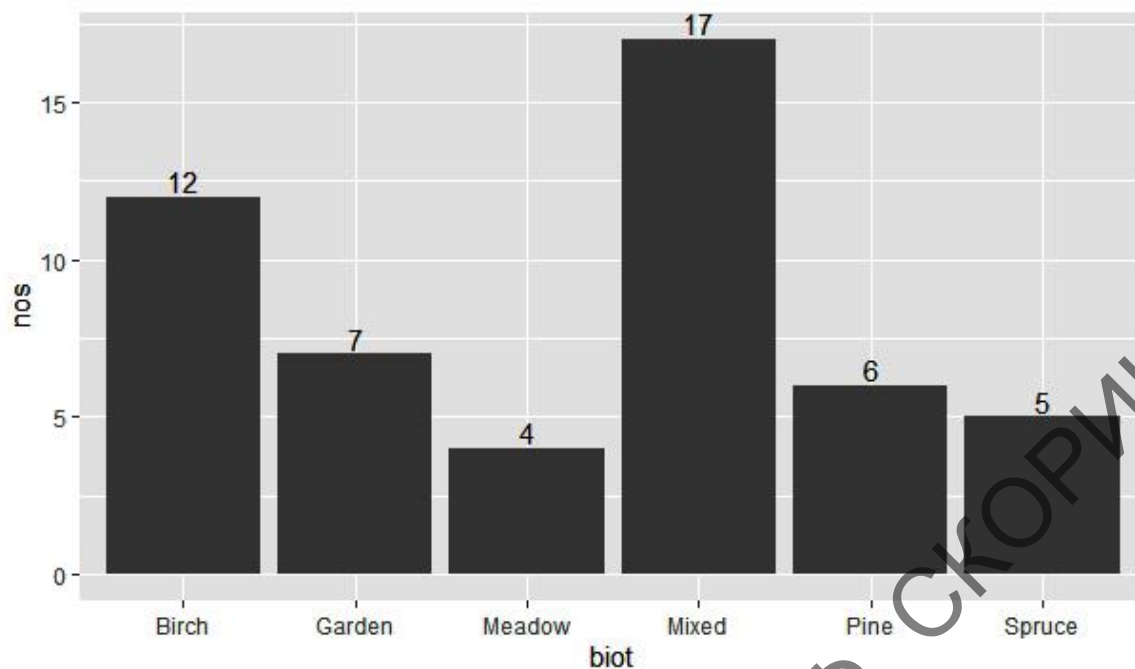


Рисунок 114 – Столбчатая диаграмма видового богатства птиц с подписанными значениями данных над столбцами

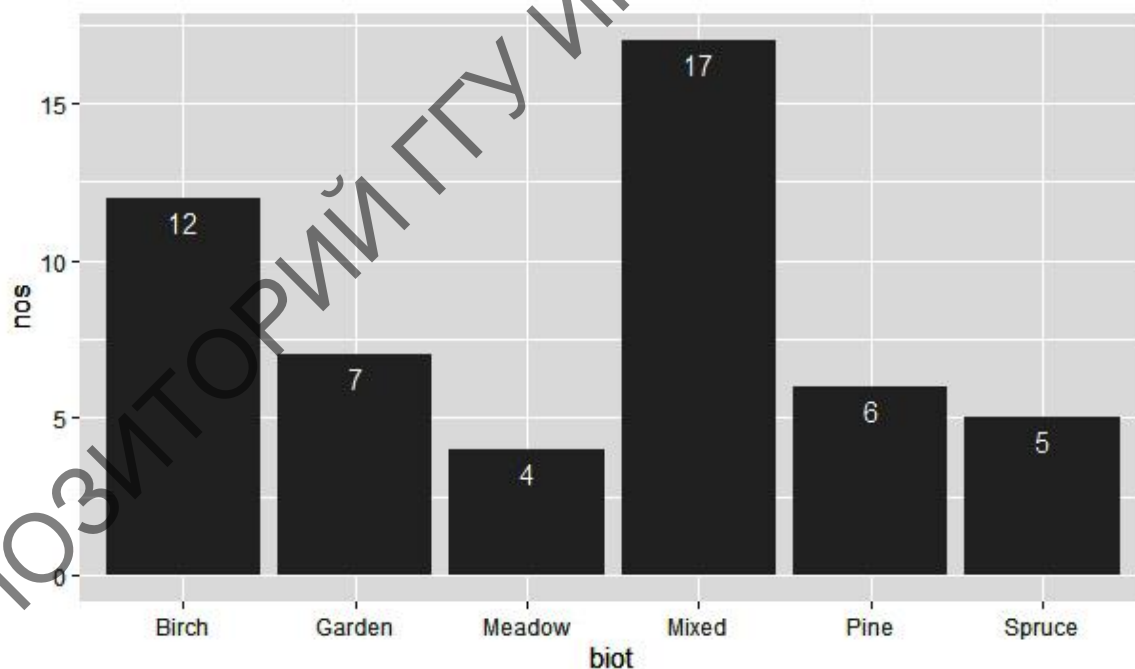


Рисунок 115 – Столбчатая диаграмма видового богатства птиц с подписанными значениями данных внутри столбцов

### 1.5.3 Построение аналогов столбчатой диаграммы в RCommander

В пакете RCommander есть опция построения столбчатой диаграммы, однако в ней есть один существенный минус – в качестве у-шкалы используются лишь частоты и проценты, а как показывает практика, чаще всего при демонстрации данных с помощью столбцов используется либо абсолютное значение признака, либо среднее с ошибкой или стандартным отклонением (т. н. «усы»). Поэтому здесь мы рассмотрим использование именно подобного рода аналогов. Для первого случая используется полосной график, а для второго – график средних.

Для примера построения полосного графика используем данные по видовому богатству птиц из таблицы 5, предварительно создав файл в программе электронных таблиц (лучше – в формате .csv), в котором место обитания птиц обозначим как *Biotore*, а количество видов – *NOS*.

**Шаг 1.** Включаем пакет RCommander.

**Шаг 2.** Загружаем в него файл с данными.

**Шаг 3.** Заходим в меню по пути **Графики** → **Полосной график...** (рисунок 116).

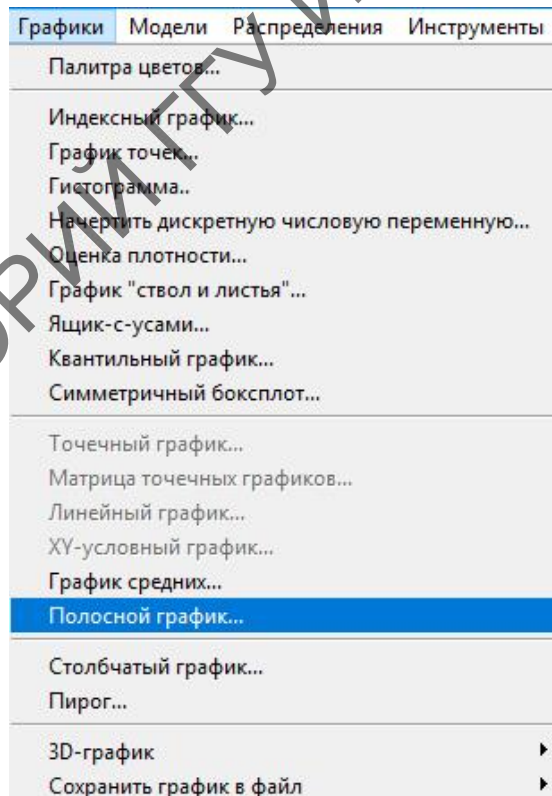


Рисунок 116 – Пункт меню **Полосной график** в RCommander

**Шаг 4.** В открывшемся диалоговом окне **Полосной график** необходимо перейти на закладку **Данные** и выделить слева фактор, т. е. в данном случае переменную, признаки которой будут демонстрироваться, а в боксе справа выделить зависимую переменную, т. е. ту, которая отвечает за значения признака (рисунок 117).

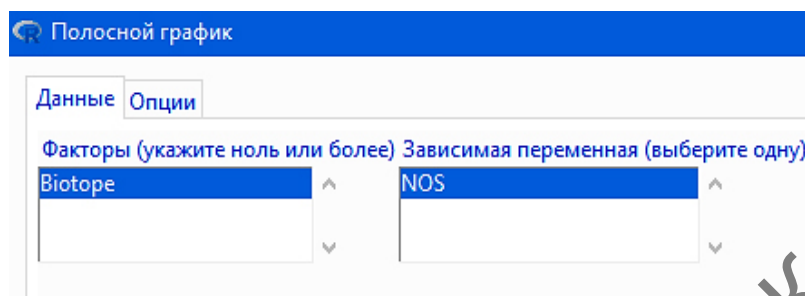


Рисунок 117 – Закладка **Данные** в диалоговом окне **Полосной график** RCommander

**Шаг 5.** Затем следует перейти на закладку **Опции**, указать их так, как отображено на рисунке 118, и после нажать на кнопку **ОК**. Получившаяся диаграмма показана на рисунке 119.

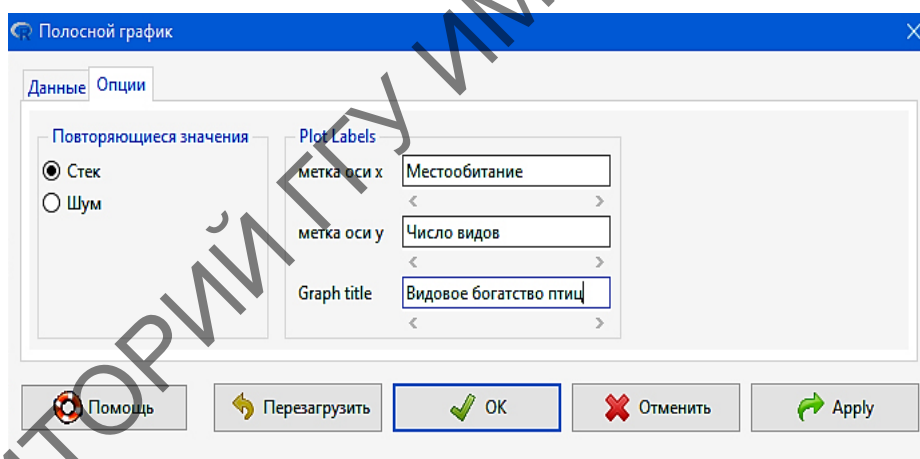


Рисунок 118 – Закладка **Опции** в диалоговом окне **Полосной график** RCommander

Для того чтобы построить график средних, воспользуемся данными таблицы 7, которые для ускорения процесса создаём в программе электронных таблиц. Создадим файл **Pticy**, данные по местообитаниям назовём **Ecosys** (*ecosystems* – экосистемы), а данные по численности – **Number** (количество).

## Видовое богатство птиц

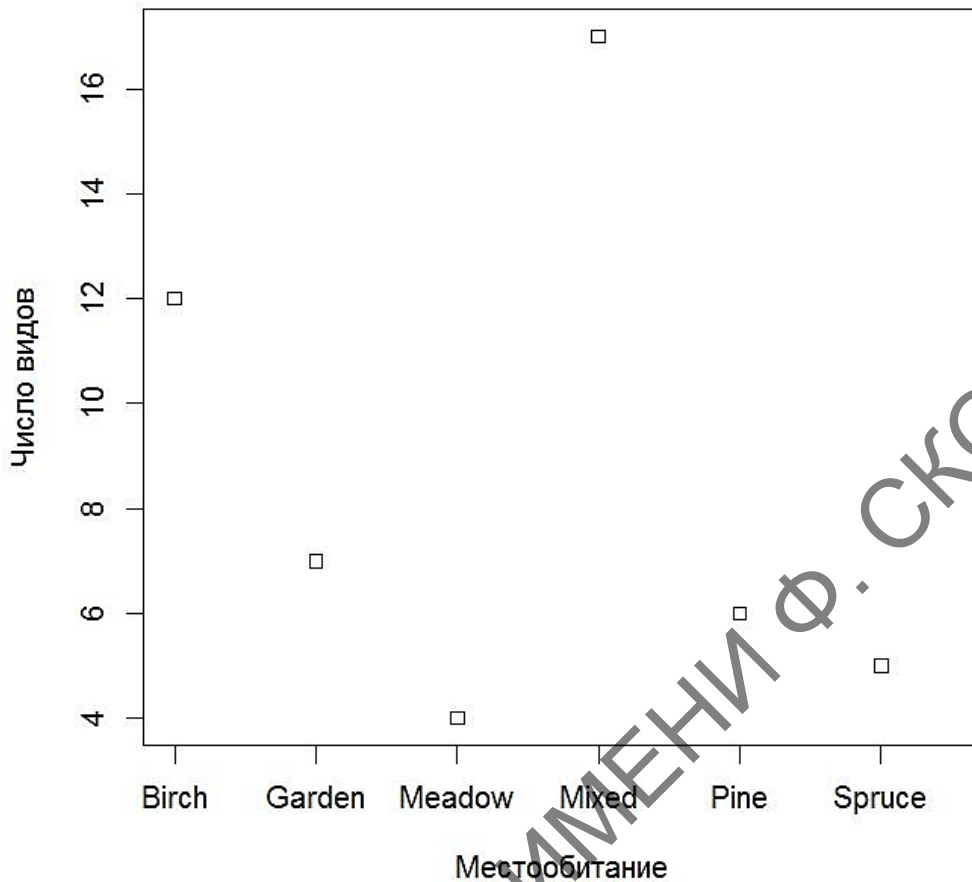


Рисунок 119 – Полосной график RCommander

**Шаг 1.** Включаем пакеты RCommander.

**Шаг 2.** Загружаем в него файл с данными.

Следует учесть одно немаловажное условие. После того как данные будут загружены в RCommander, то переменную, которая обозначает место обитания птиц, нужно конвертировать в фактор.

**Шаг 3.** Перекодируем символьную переменную *Ecosys* в фактор *Biotope*. Для этого необходимо перейти в меню по пути **Данные** → **Работа с переменными** → **Перекодировать переменные** (рисунок 120).

**Шаг 4.** Далее, в диалоговом окне **Перекодировать данные** в боксе **Переменные для перекодировки** необходимо выбрать переменную *Ecosys*, а в боксе **Новое имя переменной или префикс для множественных перекодировок:** указать новое имя – *Biotope*. В поле **Введите правила перекодирования** следует написать «as.factor» (рисунок 121) и нажать **ОК**. Теперь можно приступить к созданию графика.

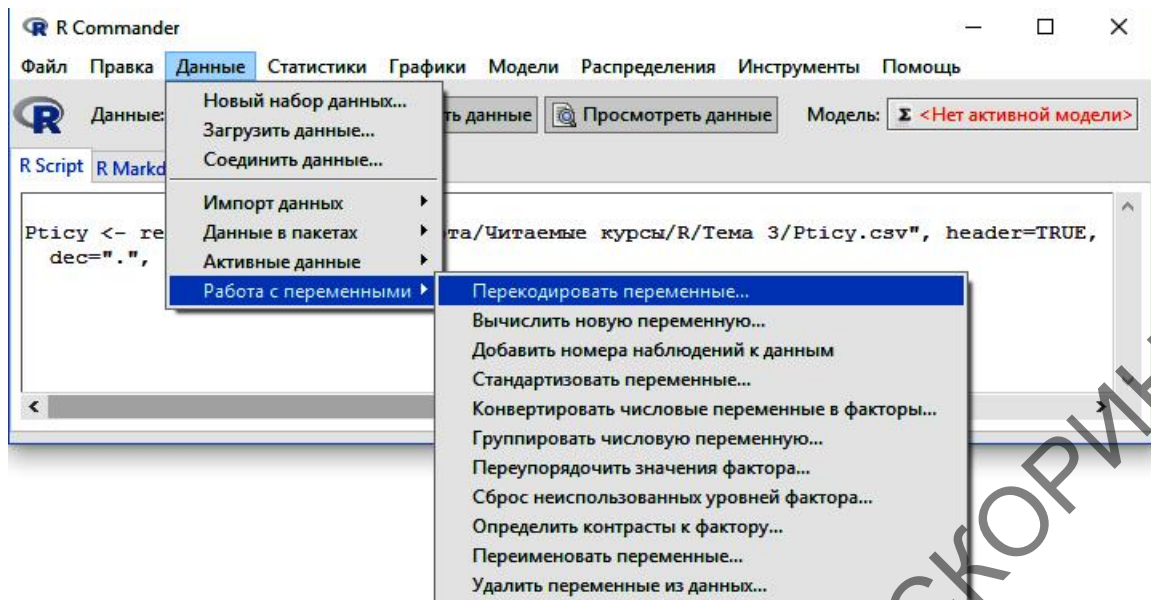


Рисунок 120 – Пункт меню **Перекодировать переменные...** в RCommander

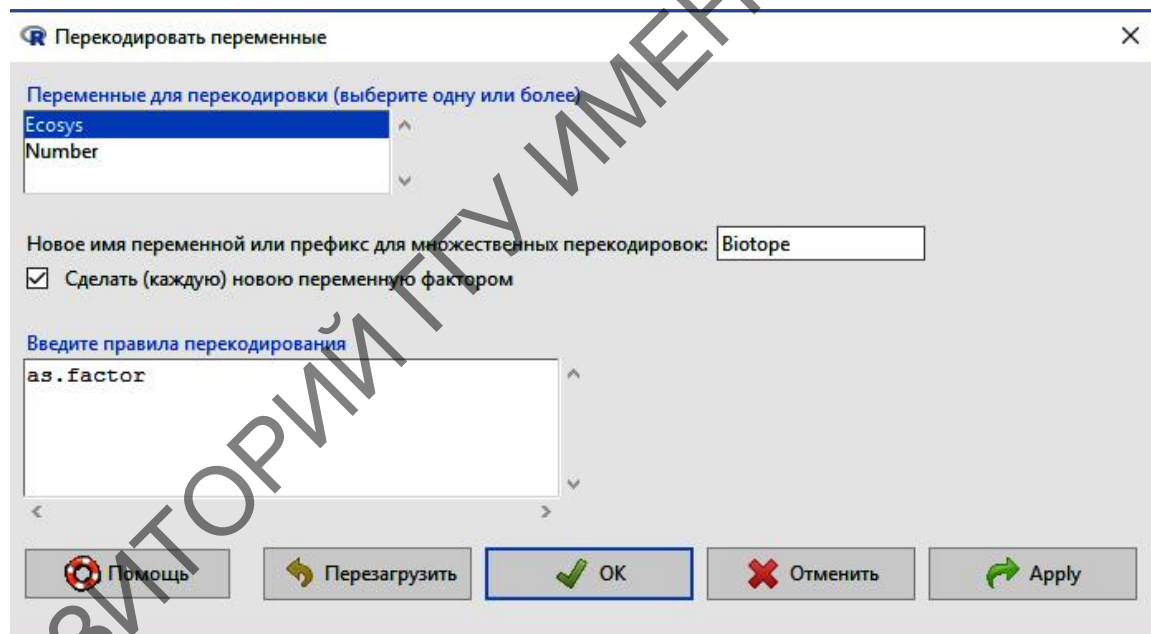


Рисунок 121 – Диалоговое окно **Перекодировать переменные** в RCommander

**Шаг 5.** Для создания графика средних необходимо перейти в меню по пути **Графики** → **График средних...** (рисунок 122).



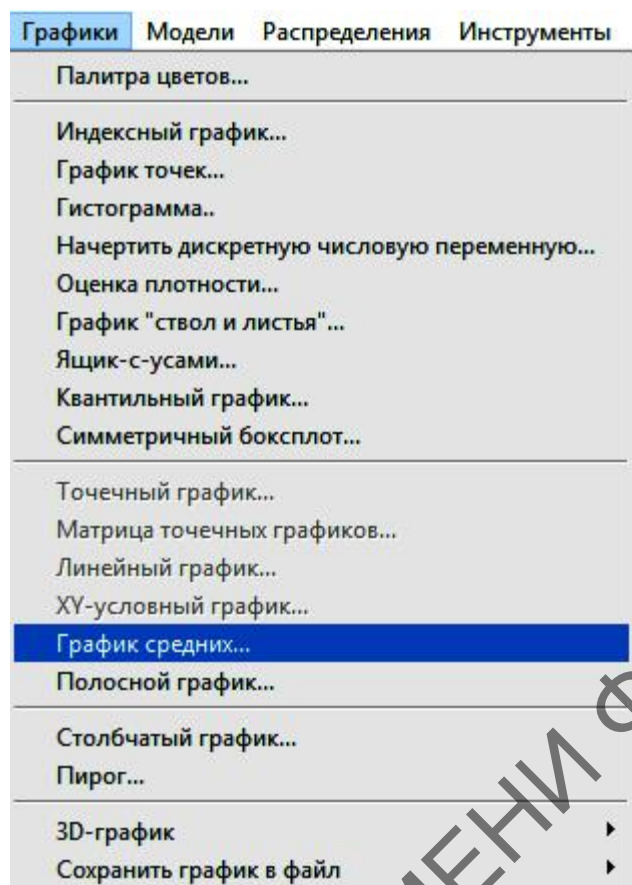


Рисунок 122 – Пункт меню **График средних...** в RCom Commander

**Шаг 6.** Далее, в диалоговом окне **График средних** на закладке **Данные** необходимо слева выбрать фактор (в нашем случае это перекодированная в фактор переменная *Biotope*), а справа – зависимую переменную (в нашем случае – *Number*). Сверьте с рисунком 123.

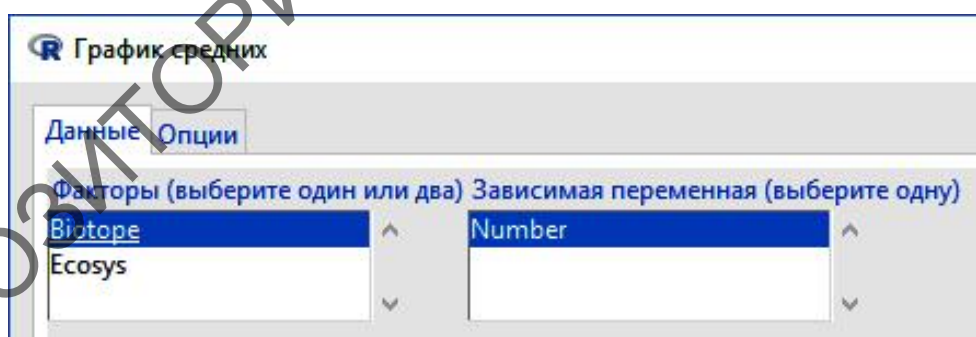


Рисунок 123 – Закладка **Данные** в диалоговом окне **График средних** RCom Commander

**Шаг 7.** После этого в диалоговом окне **Опции** необходимо выставить значения, как показано на рисунке 124, и нажать **ОК**.



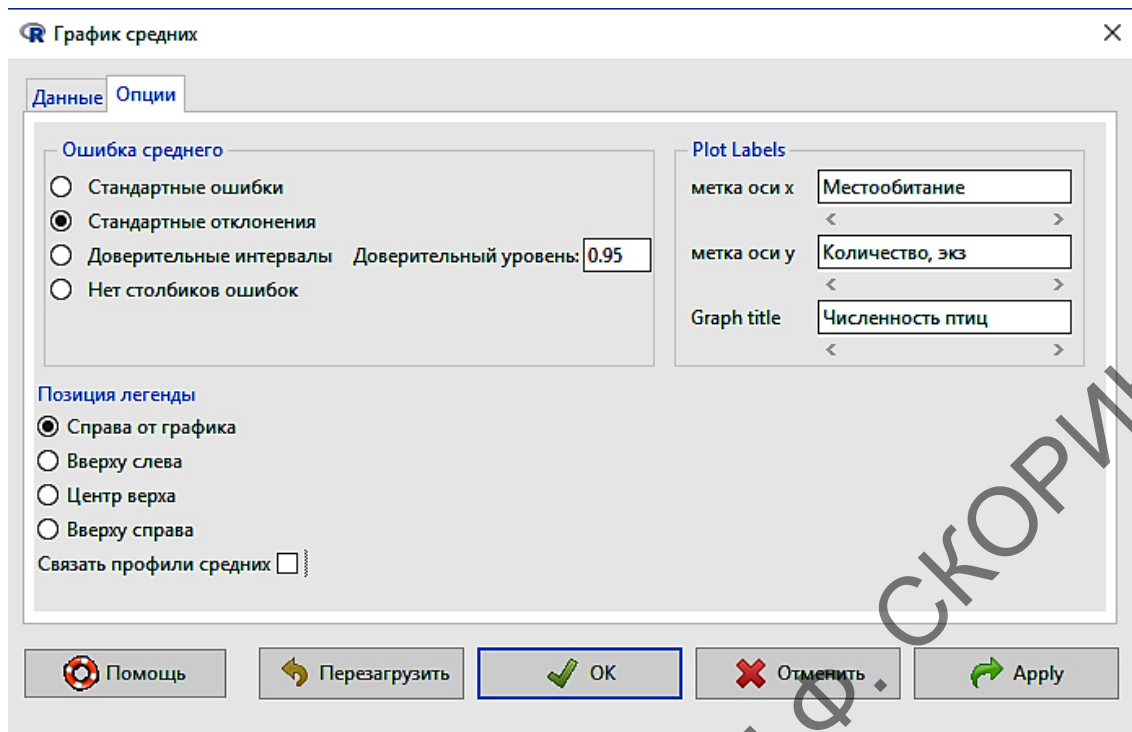


Рисунок 124 – Закладка **Опции** в диалоговом окне **График средних** RCommander

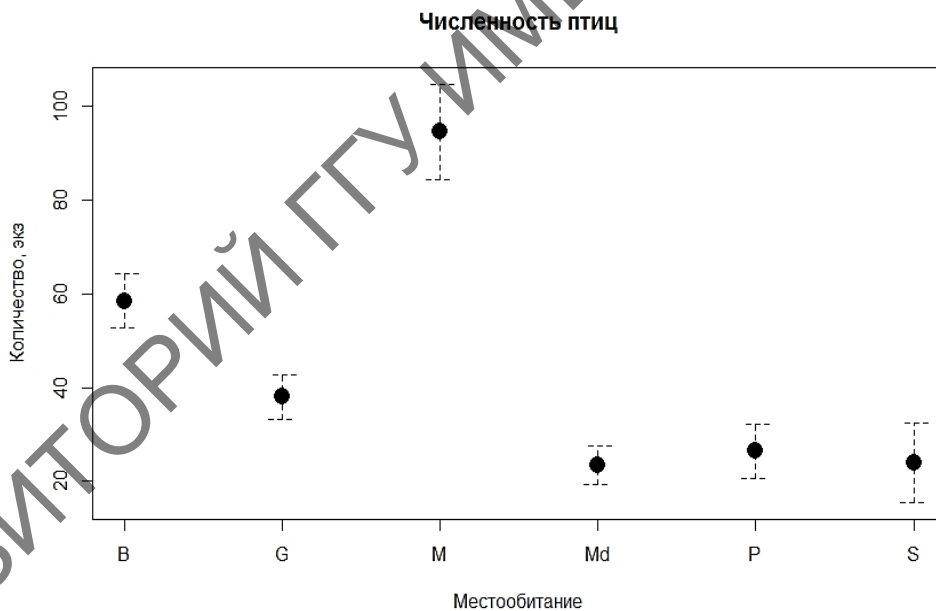


Рисунок 125 – График средних RCommander

Полученный график показан на рисунке 125. В том случае, если в закладке **Опции** оставить отмеченным бокс напротив пункта **Связать профили средних**, то график будет иметь вид, показанный на рисунке 126.

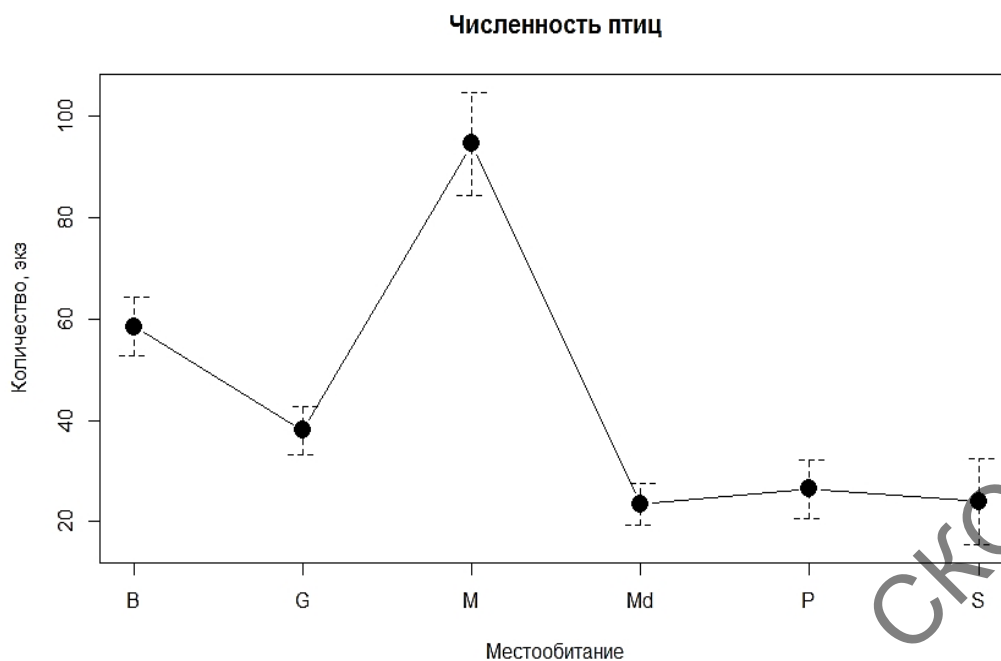


Рисунок 126 – График средних со связанными профилями средних в RCommander

## 1.6 Круговая диаграмма

Круговая диаграмма в научных исследованиях используется довольно редко (поэтому в начальных версиях языка R способы её построения вообще отсутствовали), но бывают случаи, когда она необходима.

Для построения такого рода диаграмм в языке программирования R используется специальная функция `pie()`, т. е. «пирог», которая имеет свои собственные аргументы, регламентирующие характеристики секторов и надписей: `pie(x, labels, radius, main, col, clockwise)`. Поясним суть каждого из аргументов: `x` – название числового вектора, данные которого используются при построении диаграммы; `labels` – метки секторов диаграммы; `radius` – радиус круга диаграммы (в диапазоне от  $-1$  до  $+1$ ); `main` – заголовок диаграммы; `col` – цвета секторов диаграммы; `clockwise` – направление подписей секторов диаграммы (по часовой или против часовой стрелки).

Наиболее полно создание круговой диаграммы реализовано через командную строку, поэтому здесь рассмотрим только использование для этих целей RStudio, так как в данном случае пакет RCommander хоть и может сделать круговую диаграмму (опция меню **Пирог...**), но на достаточно примитивном уровне.

## 1.6.1 Создание столбчатой диаграммы в RStudio (при помощи командной строки и базового пакета)

Для примера в учебных целях воспользуемся данными таблицы 5 по видовому богатству птиц.

**Шаг 1.** Для начала необходимо создать числовой вектор количества видов птиц.

```
> nos <- c(6, 5, 17, 12, 4, 7)
```

**Шаг 2.** Далее необходимо создать подписи секторам диаграммы.

```
> labels <- c("Сосновый лес", "Еловый лес", "Смешанный лес", "Берёзовый лес", "Луг", "Сад")
```

**Шаг 3.** Используя данные предыдущих шагов, строим простую круговую диаграмму (рисунок 127).

```
> pie(nos, labels)
```

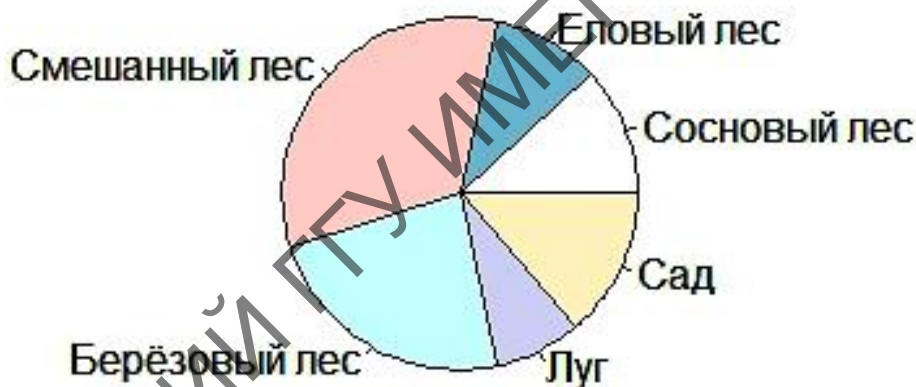


Рисунок 127 – Простая круговая диаграмма в R

Данная диаграмма хороша только для предварительного анализа. Внесём дополнительное оформление и подпишем диаграмму.

**Шаг 3.** Изменим цвета секторов и дадим название диаграмме (рисунок 128).

```
> pie(nos, labels, main = "Видовое богатство птиц", col = rainbow(length(nos)))
```

Обратим внимание на дополнительную функцию `rainbow()`, которая раскрашивает нашу диаграмму в цвета радуги, а площадь секторов регулирует параметр `length()`, где в качестве аргумента указан вектор `nos`.

## Видовое богатство птиц



Рисунок 128 – Круговая диаграмма в R с произвольными цветами и названием диаграммы

Кроме случаев, когда достаточно визуального обилия секторов (как рассмотрено выше), чаще всего возникает необходимость представления относительного обилия (доли) того или иного признака и наличия легенды. В качестве учебного примера рассмотрим это на диаграмме численности беспозвоночных в почвенных ловушках, выявленных в разных экосистемах (таблица 8).

Таблица 8 – Численность беспозвоночных в различных экосистемах

Экосистема	Численность, экз
Сосновый бор	60
Еловый лес	48
Смешанный лес	152
Берёзовая роща	146
Заливной луг	271
Суходольный луг	133
Поле	29
Сад	34

**Шаг 1.** Создадим числовой вектор *ch*, который будет включать данные по численности беспозвоночных.

```
> ch <- c(60, 48, 152, 146, 271, 133, 29, 34)
```

**Шаг 2.** Создадим символьный вектор *labels*, который будет включать названия экосистем.

```
> labels <- c("Сосновый бор", "Еловый лес", "Смешанный лес", "Берёзовая роща", "Заливной луг", "Суходольный луг", "Поле", "Сад")
```

**Шаг 3.** Введём дополнительную переменную *prct* (*percent* – процент), которой присвоим значение расчёта доли каждого сегмента диаграммы.

```
> prct <- round(100*ch/sum(ch), 1)
```

**Шаг 4.** Проверим расчёты.

```
> prct
[1] 6.9 5.5 17.4 16.7 31.0 15.2 3.3 3.9
```

**Шаг 5.** Строим диаграмму.

```
> pie(ch, labels = prct, radius = 0.9, main = "численность беспозвоночных", col = rainbow(length(ch)))
```

**Шаг 6.** Добавляем к диаграмме легенду. Полученный результат сравните с рисунком 129.

```
> legend("left", c("Сосновый бор", "Еловый лес", "Смешанный лес", "Берёзовая роща", "Заливной луг", "Суходольный луг", "Поле", "Сад"), cex = 0.5, fill = rainbow(length(ch)))
```

### Численность беспозвоночных

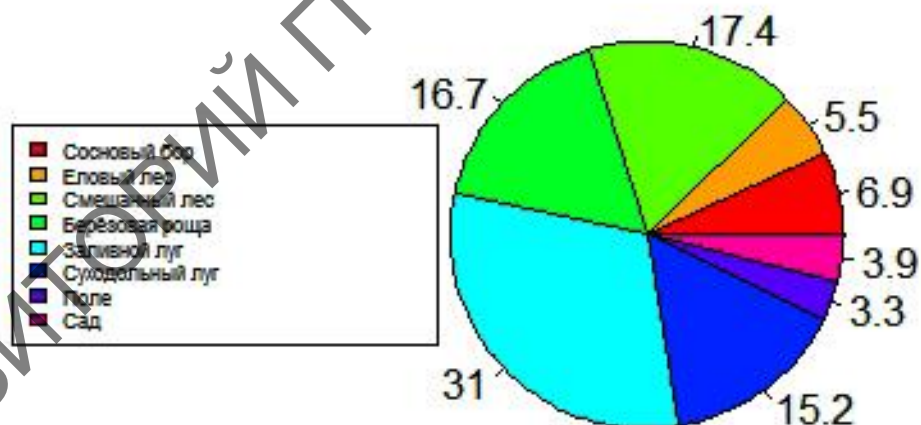


Рисунок 129 – Круговая диаграмма в R с произвольными цветами, названием диаграммы и легендой

Круговую диаграмму можно сделать и трехмерной. Для этого используются дополнительные пакеты, например, пакет **plotrix**. Сделаем трёхмерную диаграмму численности беспозвоночных.

**Шаг 1.** Загружаем и включаем пакет **plotrix**.

```
> install.packages("plotrix")
> library(plotrix)
```

**Шаг 2.** Создаём числовой вектор *ch* с данными по численности беспозвоночных

```
> ch <- c(60, 48, 152, 146, 271, 133, 29, 34)
```

**Шаг 3.** Создадим символьный вектор *lbl*, который будет включать названия экосистем.

```
> lbl <- c("СБ", "ЕЛ", "СЛ", "БР", "ЗЛ", "СхЛ", "П", "С")
```

**Шаг 4.** Строим диаграмму.

```
> pie3D(ch, labels = lbl, explode = 0.1, main = "Численность беспозвоночных")
```

Полученный результат отображён на рисунке 130.

**Численность беспозвоночных**

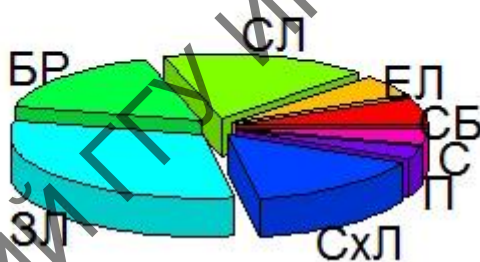


Рисунок 130 – Трёхмерная круговая диаграмма в R

### 1.6.2 Создание столбчатой диаграммы в RStudio (при помощи командной строки и пакета **ggplot2**)

Для построения круговой диаграммы в графическом пакете **ggplot2** для учебных целей воспользуемся данными таблицы 5, а также ранее созданными переменными *nos*, *biot*, а также датафреймом *birds*. Напоминаем, что пакет **ggplot2** работает не с отдельными векторами, а с датафреймами.

**Шаг 1.** Создаём промежуточный барплот для визуализации данных.

```
> bdp <- ggplot(birds, aes(x="", y=nos, fill = biot))+
  geom_bar(width = 1, stat = "identity")
```

**Шаг 2.** Посмотрим получившийся барплет (рисунок 131).

> bdp

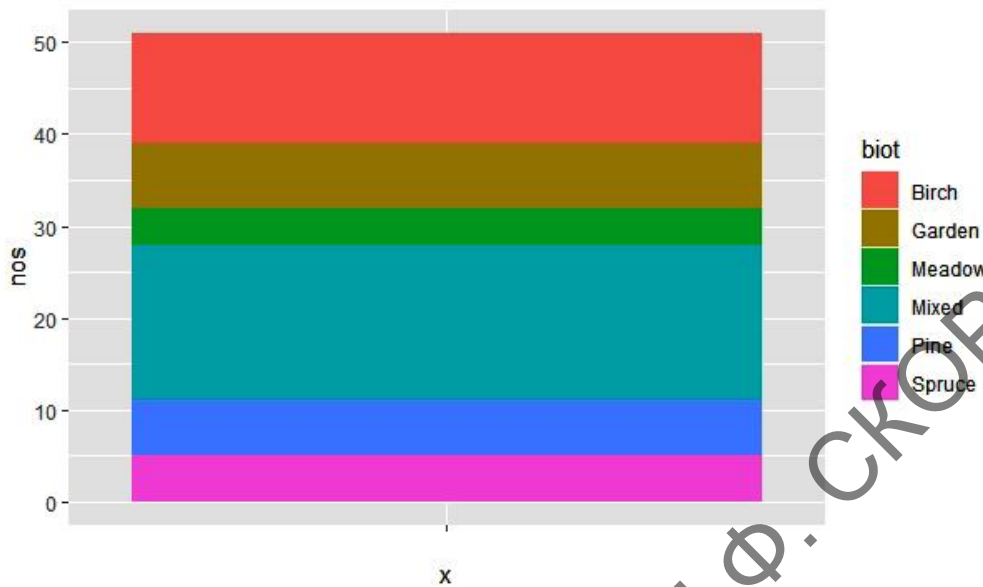


Рисунок 131 – Вертикальный барплет видового богатства птиц

**Шаг 3.** Строим круговую диаграмму

> pie <- bdp + coord\_polar("y", start=0)

**Шаг 4.** Просматриваем получившуюся круговую диаграмму (рисунок 132).

> pie

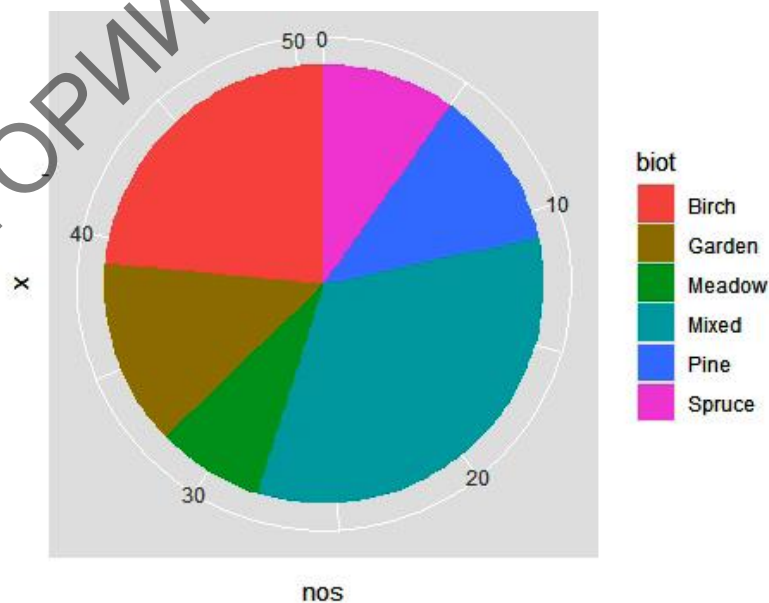


Рисунок 132 – Простая круговая диаграмма в ggplot2



Для более тонкой настройки диаграммы лучше использовать заранее сформированный шаблон и дополнительный пакет **scales**.

**Шаг 5.** Создаём шаблон оформления, введя дополнительную переменную `shab_theme`.

```
> blank_theme <- theme_minimal()+  
  theme(  
    axis.title.x = element_blank(),  
    axis.title.y = element_blank(),  
    panel.border = element_blank(),  
    panel.grid=element_blank(),  
    axis.ticks = element_blank(),  
    plot.title=element_text(size=14, face="bold"))
```

**Шаг 6.** Включаем пакет **scales**.

```
> library(scales)
```

**Шаг 7.** Настраиваем диаграмму.

```
> pie + shab_theme +  
  theme(axis.text.x=element_blank())+  
  geom_text(aes(y = nos/6 + c(0, cumsum(nos)[-  
length(nos)]),  
label = percent(nos/100)), size=3)
```

Полученную диаграмму сравним с рисунком 133.

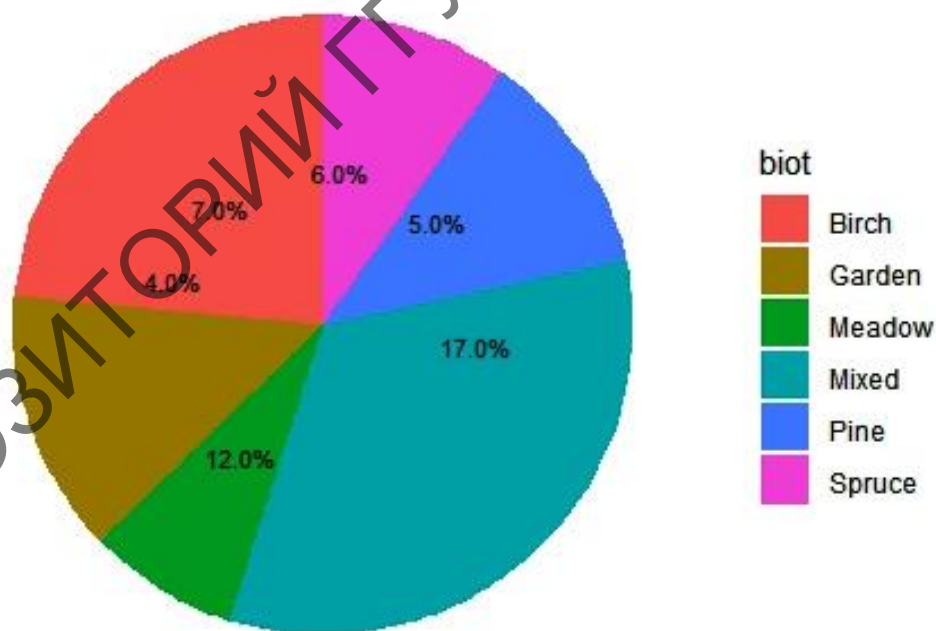


Рисунок 133 – Круговая диаграмма с подписями и легендой в **scales**



## 2 Управление графическими устройствами в R

При работе с графической информацией в языке программирования R при создании диаграммы или графика новый графический объект затирает предыдущий и он становится недоступным. Для того чтобы все создаваемые рисунки автоматически сохранялись, используются так называемые графические устройства.

Для включения устройства, т. е. для того, чтобы указать программе, что планируется создание и автоматическое сохранение входящих растровых графиков, используется команда `png()`, которая открывает одноименное графическое устройство. Для векторных изображений используется команда `pdf()`. Отключается графическое устройство командой `dev.off()`.

Посмотрим, как это работает, на конкретном примере, построив простейший график точек, и автоматически сохраним рисунок.

```
> png(file="dots.png")  
> plot(1:15)  
> dev.off
```

Получившийся график (рисунок 134) будет сохранён в указанном месте (или в рабочей папке R, если не указан другой путь).

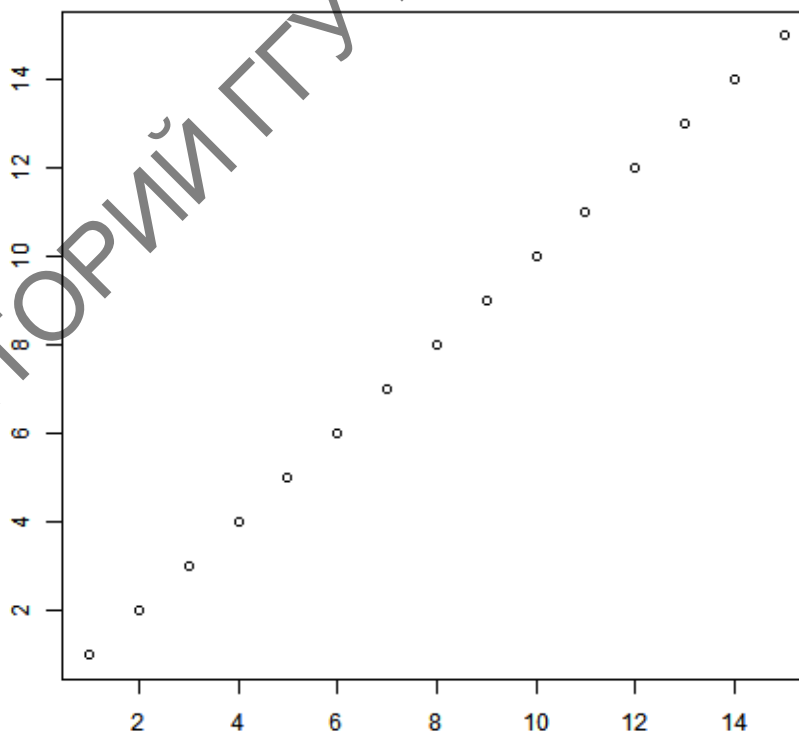


Рисунок 134 – График точек

Графический формат .png позволяет сразу делать прозрачный фон (что очень удобно для дизайнеров). Для этого в аргументах функции `png()` нужно добавить аргумент `bg="transparent"`. В нашем примере:

```
> png(file="dots.png", bg="transparent")
> plot(1:15)
> dev.off
```

Создадим также векторный вариант нашего графика точек в формате .pdf. Однако, нужно всегда учитывать возможные проблемы с кириллическими шрифтами, поэтому сделаем следующим образом, учитывая озвученную проблему.

```
> pdf("dots.pdf", family="NimbusSan", encoding="CP1251.enc")
> plot(1:15, main="График точек")
> dev.off()
```

Обычно этого достаточно (рисунок 135).

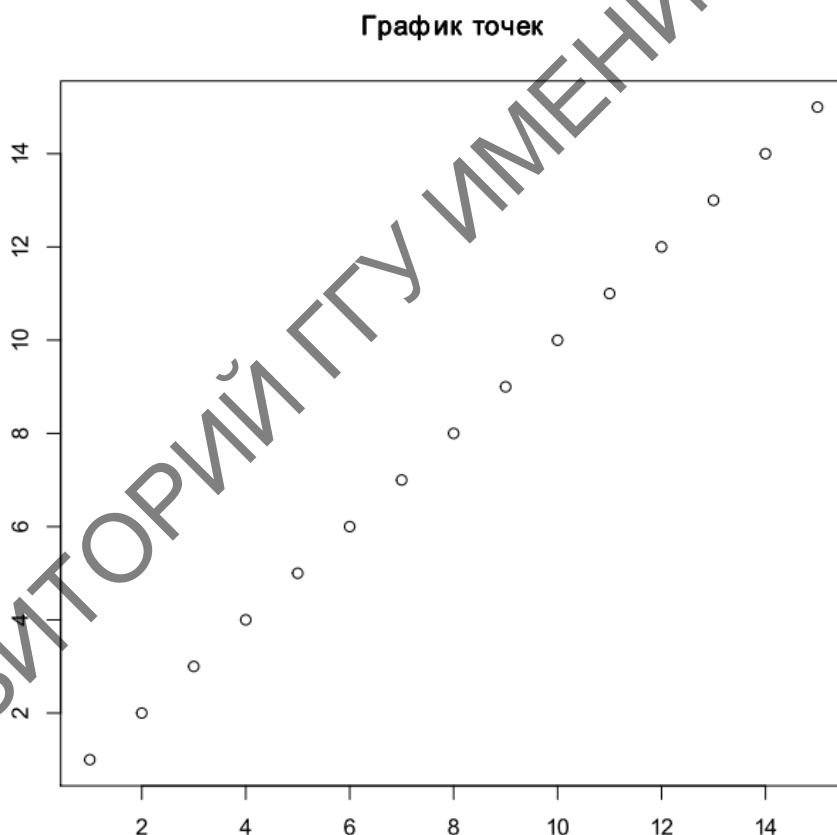


Рисунок 135 – График точек, сохранённый в формате .pdf

Здесь в качестве аргумента указывается название шрифта – `NimbusSan` и вид кодировки – `CP1251.enc`.

## Вопросы для самоконтроля

1 Охарактеризуйте особенности построения диаграммы рассеивания в языке программирования R с использованием различных пакетов.

2 Опишите процесс построения гистограмм с использованием различных графических пакетов и функции построения гистограмм в языке программирования R.

3 Какие основные функции и аргументы используются при создании линейных графиков в языке программирования R. Опишите их синтаксис.

4 Опишите условия для создания боксплотов в языке программирования R. Что представляет собой диаграмма Тьюки?

5 Охарактеризуйте особенности создания столбчатых диаграмм (барплотов) в языке программирования R.

6 Для каких целей может быть использована круговая диаграмма? Как реализуется создание круговых диаграмм в языке программирования R?

## Задания для самоконтроля

1) Имеются данные по величине ростков пеларгонии, в см ( $x$ ) и концентрации гормона роста, г/л ( $y$ ) в водном растворе:

$x$	0,15	0,17	0,20	0,22	0,29	0,42	0,69	0,70	0,94	1,21	1,48	1,75	2,20	2,68
$y$	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0	1,1	1,2	1,3	1,4

Постройте диаграмму рассеивания соотношения величины ростков с концентрацией гормона роста при помощи командной строки и GUI. Добавьте на неё линию средних квадратов.

2) Провели промеры 20 экземпляров плотвы одного из прудов рыбхоза (в см):

7,2	14,3	12,2	10,8	8,9	25,6	20,9	14,7	15,3	11,4
18,3	19,5	21,2	9,6	7,9	10,6	10,8	12,7	16,1	13,3

Постройте гистограмму распределения длины плотвы и совместите ее с кривой плотности вероятности при помощи командной строки и GUI.

3) Провели учёты плотности ( $\text{г/м}^3$ ) планктона на 15 створах реки (верхний ряд) весной (средний ряд) и осенью (нижний ряд):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
23,5	15,7	22,3	18,9	16,1	15,0	16,0	17,2	21,6	15,9	16,2	19,4	20,6	20,4	17,8
33,1	19,2	36,1	25,3	21,6	18,0	21,4	24,2	29,8	19,9	23,6	28,6	31,9	38,4	27,1

Постройте совмещённый линейный график изменения плотности планктона в разный период сезона при помощи командной строки и GUI. Добавьте на него точки данных.

4) Получены данные по численности жесткокрылых в 10 ловушках из 3 стационаров в прибрежных экосистемах с различной степенью рекреационной нагрузки:

C1	5	8	4	7	1	3	4	4	8	6
C2	10	12	11	12	15	13	12	12	11	9
C3	22	24	28	30	33	27	29	35	31	25

Постройте боксплоты численности жесткокрылых при помощи командной строки и GUI.

5) На 5 полях картофеля провели уборочные работы и рассчитали урожайность:

Поле	1	2	3	4	5
Урожайность, ц/га	280	320	370	365	340

Постройте столбчатую диаграмму урожайности картофеля на полях при помощи командной строки и GUI.

6) Имеются следующие данные по населению областей Республики Беларусь и г. Минску в тыс. человек [4]:

Регион	Численность населения
Брестская область	1 347,0
Витебская область	1 133,4
Гомельская область	1 386,6
Гродненская область	1 025,8
Минская область	1 472,0
Могилевская область	1 023,0
г. Минск	2 020,6

Постройте круговую диаграмму численности населения регионов Беларуси при помощи командной строки.

## 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 Chang, W. R Graphics Cookbook / W. Chang [Электронный ресурс]. – Режим доступа: <https://r-graphics.org/index.html>. – Дата доступа: 10.01.2021.

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.

# ТЕМА 4. ХАРАКТЕРИСТИКА ВЫБОРКИ И ПОСТРОЕНИЕ ВАРИАЦИОННОГО РЯДА. СРАВНЕНИЕ ВЫБОРОЧНЫХ СРЕДНИХ В R

- 1 Описательная статистика.
- 2 Проверка достоверности выборки.
- 3 Сравнение выборочных средних.

## 1 Описательная статистика

Описательная статистика позволяет обобщать первичные результаты, полученные при наблюдении или в эксперименте. Основная суть описательной статистики сводится к группировке значений признака в выборке или генеральной совокупности, построению распределения их частот, выявлению центральных тенденций распределения (средние значения признака), а также к оценке разброса данных по отношению к найденной центральной тенденции.

Основными показателями описательной статистики являются среднее арифметическое, её ошибка, среднеквадратичное (стандартное) отклонение, пределы (минимум и максимум), мода, медиана, асимметрия, эксцесс, коэффициент вариации и др.

Язык программирования R позволяет реализовать данные возможности при оценке структуры набора данных. В качестве учебного примера рассмотрим их на показателях обхвата деревьев в парке (столбец *Girth* из готовой таблицы данных *trees*).

### 1.1 Расчёт показателей выборки в RStudio (при помощи командной строки)

**Шаг 1.** Загружаем таблицу данных *trees* из базового пакета R с использованием функции `data()` (раздел 1.4 темы 3).

```
> data(trees)
```

Для того чтобы проводить операции и расчеты для каждого столбца таблицы отдельно, необходимо «прикрепить» таблицу командой `attach()`.

**Шаг 2.** «Прикрепим» таблицу, указывая программе, что мы хотим «разъединить» столбцы таблицы:

```
> attach(trees)
```

**Шаг 3.** Для того чтобы рассчитать среднее арифметическое данных в колонке обхвата ствола, используем функцию `mean()`.

```
> mean(Girth)
[1] 13.24839
```

**Шаг 4.** Аналогично рассчитываем медиану, среднее квадратичное отклонение, дисперсию, минимальное и максимальное значения, используя команды `median()`, `sd()`, `var()`, `min()` и `max()` соответственно.

**Шаг 5.** Рассчитаем стандартную ошибку выборки обхвата ствола, используя формулу расчета стандартной ошибки:

$$S_x = \frac{\sigma}{\sqrt{n}}$$

```
> SEgirth= sd(Girth)/sqrt(length(Girth))
> SEgirth
[1] 0.5636263
```

Здесь задействована также функция `length()`, которая показывает объём выборки. Таким образом, средний обхват ствола этих деревьев составляет  $13,2 \pm 0,56$ .

Следующий показатель выборки – мода. Она демонстрирует то значение признака, которое встречается чаще всех и встречается довольно редко. Для расчётов моды используем столбец с высотой деревьев (*Height*), так как он имеет более разнообразные значения по сравнению со столбцом *Girth*.

**Шаг 6.** Перед расчётом моды выясняем частоту встречаемости каждого из признаков высоты деревьев (вариационный ряд), используя функцию `table()` и промежуточную переменную `t.height`.

```
> t.height <- table(height)
> t.height
Height
63 64 65 66 69 70 71 72 74 75 76 77 78 79 80 81 82 83 85
86 87
  1  1  1  1  1  1  1  2  2  3  2  1  1  1  5  2  1  1  1
  1  1
```

**Шаг 7.** Теперь всё готово для расчёта моды. Для этого введём дополнительную переменную `mode`, которой присвоим значение, наиболее часто встречающееся в выборке, – функция `which.max()`.

```
> mode <- t.height[which.max(t.height)]
> mode
80
5
```

В результате выясняем, что модой является значение высоты «80», которое встречалось 5 раз.

Важными показателями выборки являются показатели кривой распределения – квантили (квартили, децили и перцентили) и межквартильное расстояние.

**Шаг 8.** Рассчитаем квартили столбца охвата ствола. Для этого необходимо использовать функцию `quantile()`.

```
> quantile(Girth)
 0%   25%   50%   75%  100%
8.30 11.05 12.90 15.25 20.60
```

**Шаг 9.** Выясним межквартильный интервал (между 25 и 75 квартилями):

```
> IQR(Girth)
[1] 4.2
```

**Шаг 10.** Завершим первый анализ выборки общими данными в «одном пакете» командой `summary()`.

```
> summary(Girth)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 8.30   11.05   12.90   13.25   15.25   20.60
```

**Шаг 11.** В качестве завершения построим гистограмму распределения уже знакомой нам командой `hist()` (раздел 1.2 темы 3) и сверим её с рисунком 136.

```
> hist(Girth, xlab = "обхват", ylab = "частота", main = "Обхват ствола, дм")
```

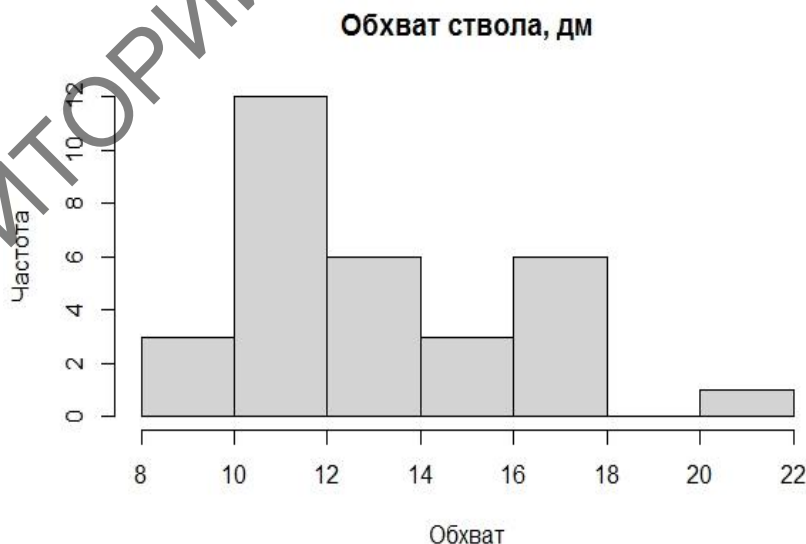


Рисунок 136 – Гистограмма данных по обхвату ствола деревьев в парке



**Шаг 12.** Для полного завершения построим гистограмму распределения, совмещённую с кривой плотности вероятности (раздел 1.2 темы 3), и сверим полученный результат с рисунком 137.

```
> hist(Girth, freq = FALSE, xlab = "Обхват", ylab = "ча-  
стота", main = "Обхват ствола, дм")  
> lines(density(Girth), col = "red", lwd = 2)
```

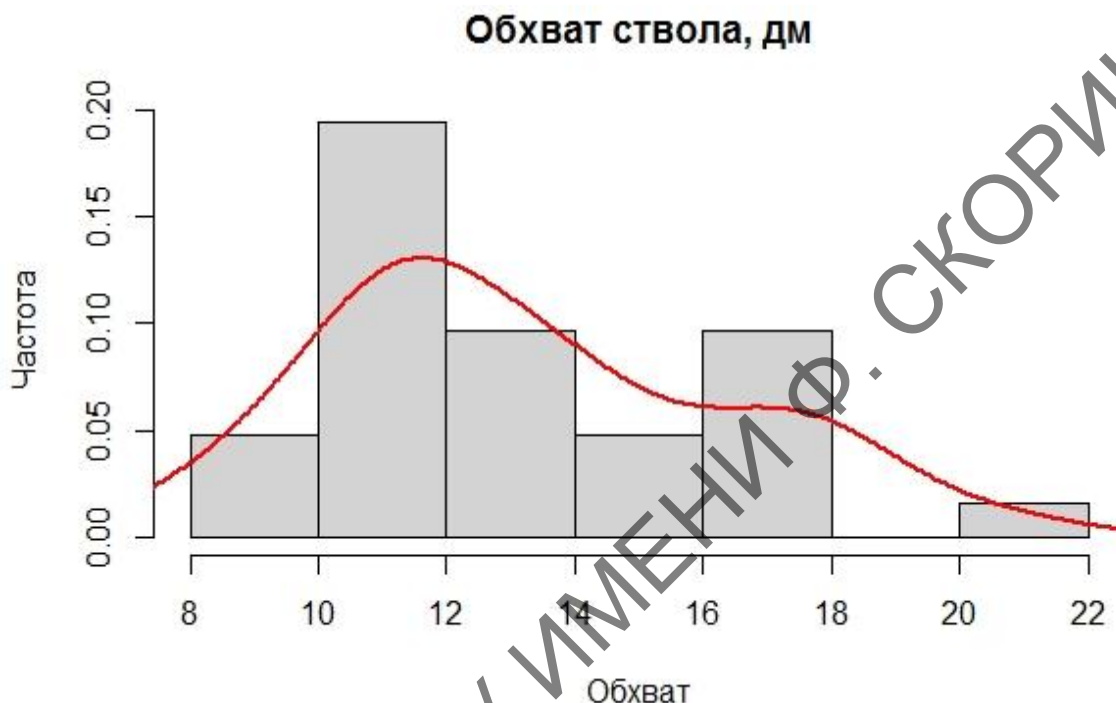


Рисунок 137 – Гистограмма данных по обхвату ствола деревьев в парке, совмещённая с кривой плотности вероятности

**Шаг 13.** Аналогично проводим расчеты для двух других колонок таблицы.

**Шаг 14.** Необходимо не забыть открепить столбцы таблицы *trees* командой `detach()`:

```
> detach(trees)
```

Немаловажным показателем характеристик разброса данных является коэффициент вариации. Он позволяет как безразмерная величина сравнить данные признаков, измеренных в разных единицах (например, вес коров в килограммах и удои в литрах). По своей сути коэффициент вариации – это отношение стандартного отклонения к среднему, взятое в процентах. Воспользуемся для примера данными для всех показателей деревьев таблицы *trees*.

**Шаг 15.** Рассчитаем коэффициент вариации для трёх показателей таблицы данных *trees*.

```
> 100*sapply(trees, sd)/colMeans(trees)
      Girth      Height      Volume
23.686948  8.383964  54.482331
```

Здесь использованы функции `sapply()` – упрощённый вывод функции `lapply()` и `colMeans()` – среднее значение колонок таблицы. Похожие функции есть и для расчёта среднего значения строк таблицы – `rowMeans()`, и для суммирования значений колонок – `colSums()`, и строк, естественно, – `rowSums()`. Это лишний раз свидетельствует о том, что таблицы данных в R – ни что иное, как электронные таблицы.

## 1.2 Расчёт показателей выборки в RCommander (при помощи GUI)

Для учебных целей воспользуемся всё той же таблицей данных о параметрах 31 дерева в парке, т. е. *trees*.

**Шаг 1.** Загружаем таблицу из базового пакета в число активных данных RCommander. Для этого необходимо перейти в меню по пути **Данные** → **Данные в пакетах** → **Прочитать данные из присоединенного пакета...** (рисунок 138).

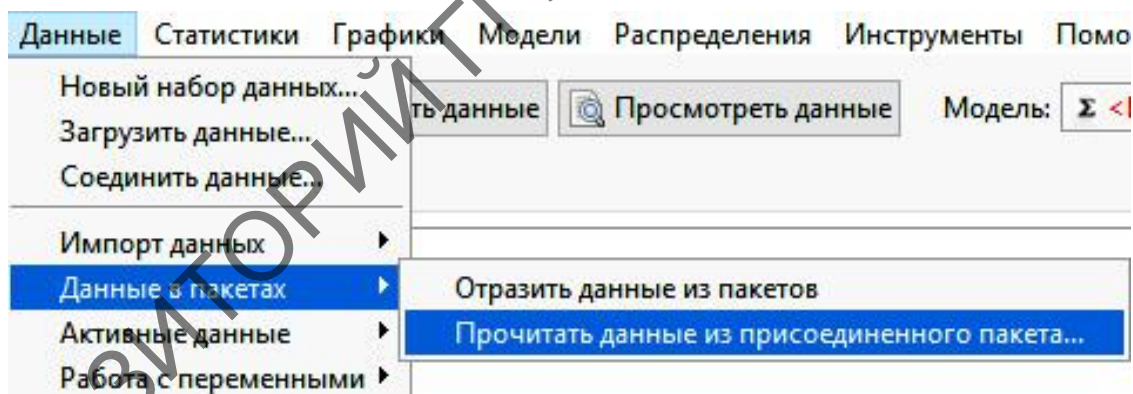


Рисунок 138 – Меню RCommander для загрузки данных из присоединённых пакетов

**Шаг 2.** В диалоговом окне **Прочитать данные из пакета** в том случае, если нужный пакет не отражается в боксе **Пакет** в левой части окна, то нужно ввести название пакета с данными в боксе **Введите имя данных** и нажать **ОК** (рисунок 139).

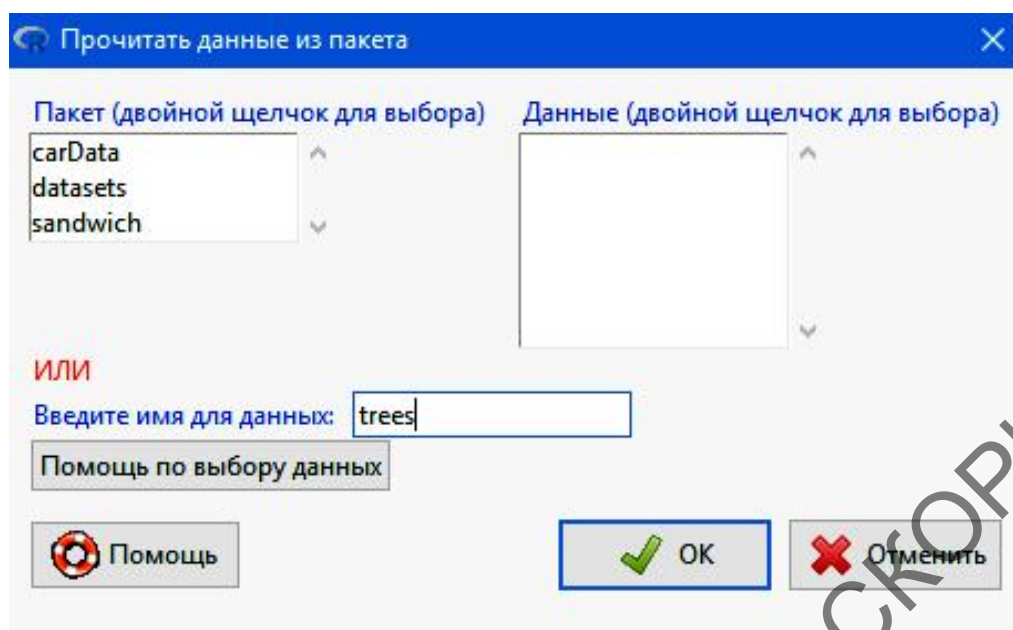


Рисунок 139 – Диалоговое окно **Прочитать данные из пакета**

После этого название таблицы данных *trees* будет отображено в боксе **Данные** под строкой меню в основном окне RCommander. После этого можно приступить к расчёту показателей описательной статистики данных деревьев в парке.

**Шаг 3.** Для перехода в диалоговое окно, содержащее показатели описательной статистики необходимо перейти в меню по пути **Статистики → Итоги → Базовые статистики** (рисунок 140).

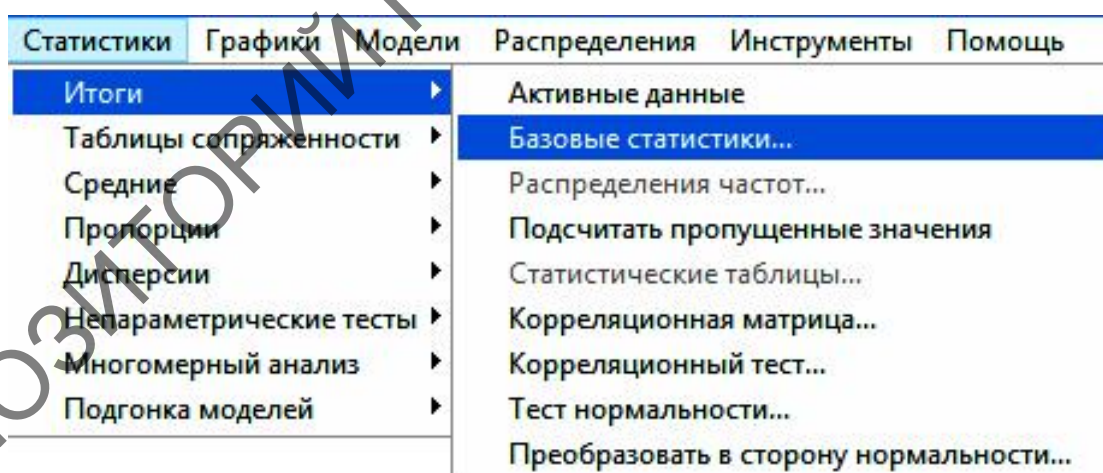


Рисунок 140 – Пункт меню **Базовые статистики...**

**Шаг 4.** Затем в диалоговом окне **Числовые итоги** необходимо в закладке **Данные** выбрать переменную *Girth* (рисунок 141).

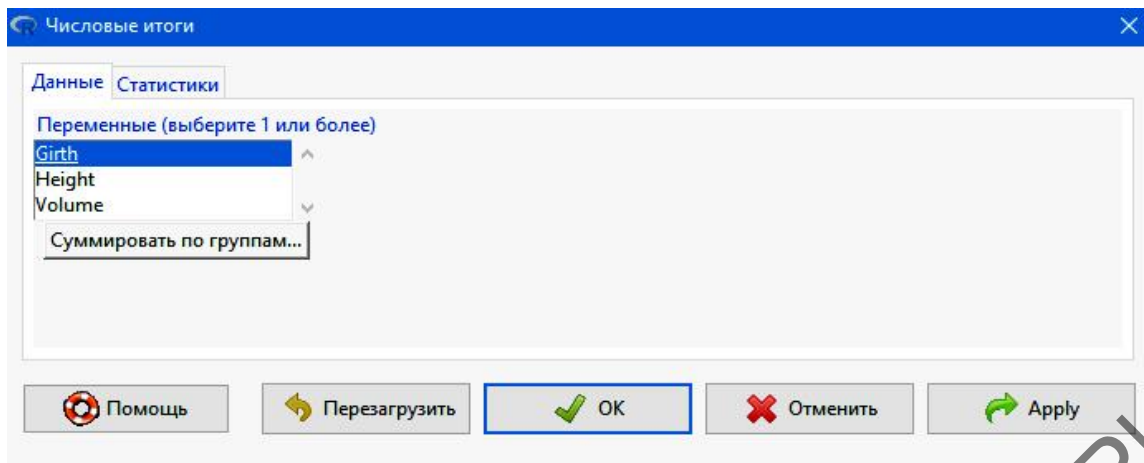


Рисунок 141 – Закладка **Данные** в диалоговом окне **Числовые итоги**

**Шаг 5.** После выбора переменной в закладке **Статистики** необходимо обозначить боксы с нужными параметрами описательной статистики. Выберите их все (рисунок 142) после чего нажмите кнопку **OK**.

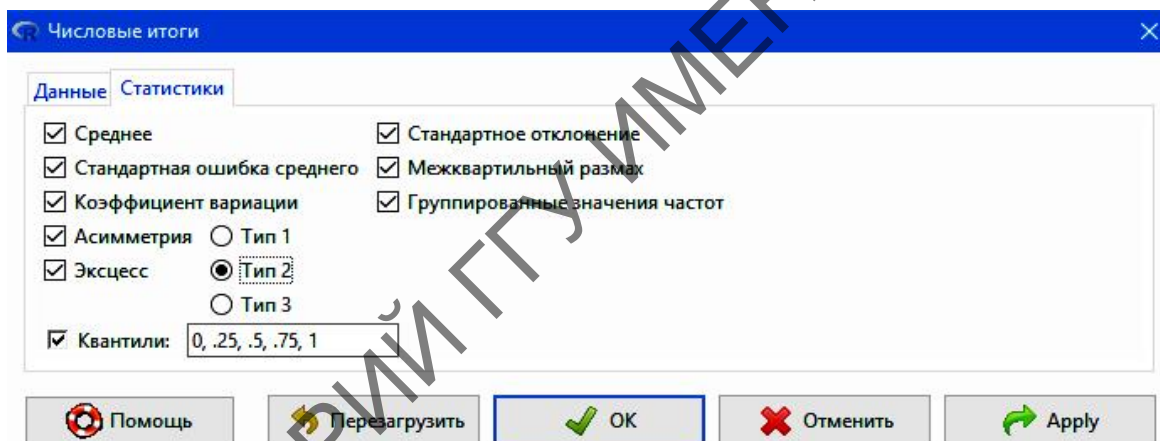


Рисунок 142 – Закладка **Статистики** в диалоговом окне **Числовые итоги**

Как результат в окне **Output** (Вывод) будут отражены результаты расчётов под строками командного кода (рисунок 143):

- mean** – средняя арифметическая;
- sd** – стандартное отклонение (*standard deviation*);
- se(mean)** – стандартная ошибка средней арифметической (*standard error*);

```

+  type="2")
  mean      sd se(mean) IQR      cv skewness  kurtosis  0%   25%  50%
13.24839 3.138139 0.5636263 4.2 0.2368695 0.5534652 -0.4354214 8.3 11.05 12.9
  75% 100%  n
 15.25 20.6 31

> binnedCounts(trees[,"Girth", drop=FALSE])
Binned distribution of Girth
      Count Percent
[8, 10]      3    9.68
(10, 12]     12   38.71
(12, 14]      6   19.35
(14, 16]      3    9.68
(16, 18]      6   19.35
(18, 20]      0    0.00
(20, 22]      1    3.23
Total        31  100.00

```

Рисунок 143 – Окно **Output** в основном окне пакета RCommander

**IQR** – межквартильный размах (*interquartile range*);

**cv** – коэффициент вариации (*coefficient of variation*);

**skewness** – асимметрия;

**kurtosis** – эксцесс;

0 %–100 % – значения квартилей (с первого по пятый);

**n** – объём выборки.

Таким образом, основные показатели переменной *Girth* отображены. Аналогично рассчитайте показатели и двух других переменных таблицы *trees*.

## 2 Проверка достоверности выборки

Для проверки достоверности средних показателей выборки воспользуемся данными по учёту численности остромордой лягушки в 20 кварталах лесного массива за год (таблица 9).

Таблица 9 – Численность остромордой лягушки в лесном массиве

К	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Ч	56	180	141	97	302	155	247	231	250	184	166	197	201	103	333	192	154	155	143	84
Примечание: К – квартал лесного массива, Ч – численность остромордой лягушки, экз																				

## 2.1 Проверка достоверности выборки в RStudio (при помощи командной строки)

Подготовим наши данные для проверки.

**Шаг 1.** Создаем числовой вектор численности остромордой лягушки, используя переменную *nra* (*number of Rana arvalis* – численность остромордой лягушки), и проверяем его.

```
> nra <- c(56, 180, 141, 97, 302, 155, 247, 231, 250, 184, 166, 197, 201, 103, 333, 192, 154, 155, 143, 84)
> nra
[1] 56 180 141 97 302 155 247 231 250 184 166 197 201 103
[15] 333 192 154 155 143 84
```

**Шаг 2.** Рассчитываем среднее значение численности остромордой лягушки в лесном массиве.

```
> mean(nra)
[1] 178.55
```

Для проверки достоверности среднего показателя выборки могут использоваться как параметрические (в том случае, если распределение признака в выборке соответствует закону нормального распределения), так и непараметрические тесты.

### 2.1.1 Параметрический тест Стьюдента

При проверке достоверности параметрическими тестами наиболее часто используется параметрический тест Стьюдента, который в языке программирования R рассчитывается при помощи функции `t.test()`.

**Шаг 3.** Рассчитываем критерий Стьюдента для данных по остромордой лягушке.

```
> t.test(nra, mu=mean(nra))
      One Sample t-test

data:  nra
t = 0, df = 19, p-value = 1
alternative hypothesis: true mean is not equal to 178.55
95 percent confidence interval:
 145.7094 211.3906
sample estimates:
mean of x
 178.55
```

Остановимся на результатах подробнее. Сначала отметим, что аргументами функции `t.test()` послужила сама переменная – `nra`, а также `mu` – число, указывающее истинное значение среднего (т. е. мы должны указать, откуда необходимо взять данные о реальном среднем значении). Далее рассмотрим результаты самого теста. В первой строке результата указывается переменная, которая анализировалась. На следующей строке показаны данные по результату расчёта критерия Стьюдента, и он равен нулю ( $t = 0$ ); число степеней свободы ( $df = 19$ ) и значение уровня значимости  $p$ , равный единице, что соответствует 100 % (т. е. вычисленный показатель на 100 % соответствует реальному среднему значению). Ниже указана альтернативная гипотеза, с которой проводилось сравнение. Она говорит о том, что истинное среднее не равно значению 178,55 экз. В данном случае это не так, альтернативная гипотеза не подтвердилась, следовательно, верна нулевая гипотеза о том, что рассчитанная средняя достоверно не отличается от истинной. На следующей строке указаны пределы доверительного интервала (*confidence interval*) при 95 % точности ( $p = 0,05$ ).

### 2.1.2 Непараметрический тест Уилкоксона

Данный вид теста используется в том случае, когда распределение признаков в выборке не соответствует нормальному или оно неизвестно. Для расчёта непараметрического теста Уилкоксона в языке программирования *R* используется функция `wilcox.test()`, синтаксис которой схож с ранее рассмотренной функцией критерия Стьюдента для одномерной выборки.

**Шаг 4.** Рассчитываем показатель Уилкоксона для данных по остромордой лягушке.

```
> wilcox.test(nra, mu=median(nra), conf.int=TRUE)
wilcoxon signed rank test with continuity correction
data: nra
V = 109, p-value = 0.896
alternative hypothesis: true location is not equal to 173
95 percent confidence interval:
 144.5000 208.5001
sample estimates:
(pseudo)median
 174.5556
```

Как и в предыдущем шаге, прокомментируем итоги. Они несколько схожи с результатами критерия Стьюдента. Основное внимание надо уделить показателю уровня значимости  $p$ , который равен



0,896, что значительно выше порогового 0,005 и, следовательно, альтернативная гипотеза отвергается. В данном случае также нужно иметь в виду, что тест Уилкоксона работает не со средней арифметической, как предыдущий, а с медианой, как более устойчивой (робастной) величиной. Отсюда, кстати, и другие, более узкие значения доверительного интервала.

### 2.1.3 Проверка распределения на нормальность

Для того чтобы определить, какой метод проверки выбрать, нужно знать, соответствует ли распределение в выборке закону нормального распределения. Для этого можно воспользоваться гистограммой распределения, но проще всего использовать тест Шапиро – Уилка. В языке программирования R за проведение этого теста отвечает функция `shapiro.test()`. Воспользуемся им для определения соответствия нормальному распределению данных по численности остромордой лягушки (таблица 9).

**Шаг 5.** Проведём тест Шапиро – Уилка для проверки соответствия распределения численности остромордой лягушки в лесном массиве закону нормального распределения.

```
> shapiro.test(nra)
      Shapiro-Wilk normality test

data:  nra
W = 0.97312, p-value = 0.8188
```

Нулевая гипотеза (т. е. соответствие распределения закону нормального распределения) в данном случае не соответствует действительности – уровень значимости  $p$  значительно выше порогового 0,05.

Другая функция, при помощи которой можно проверить распределение выборки на нормальность, – это тест Колмогорова – Смирнова – `ks.test()`. При этом, в отличие от теста Шапиро – Уилка необходимо в качестве аргумента дополнительно указать параметр "pnorm" (т. е. проверка на нормальность).

**Шаг 6.** Проведём тест Колмогорова – Смирнова для проверки соответствия распределения численности остромордой лягушки в лесном массиве закону нормального распределения.

```
> ks.test(nra, "pnorm")
      one-sample kolmogorov-smirnov test

data:  nra
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```



В данном случае, в отличие от предыдущего теста уже упомяну- та альтернативная гипотеза, в качестве которой принимается воз- можность нормального распределения. Так как нулевая гипотеза (отсутствие нормального распределения) подтверждается –  $p$  мень- ше, чем  $2,2 \times 10^{-16}$  (и, соответственно, значительно меньше, чем 0,001), то можно сказать, что распределение лягушки остромордой в 20 кварталах лесного массива не соответствует нормальному рас- пределению и для расчёта статистических параметров для этой вы- борки нельзя использовать параметрические методы.

Ещё один действенный способ – графический. При данном спосо- бе используется расположение точек из реальных данных (выбороч- ный квантиль) в соответствии с тем положением, каким оно должно быть при нормальном распределении (теоретический квантиль). Сама прямая проводится через квантили, и если точки лежат на самой этой прямой или близко к ней, то распределение нормальное.

**Шаг 7.** Создадим диаграмму расположения точек по линиям ре- альных и теоретических данных.

```
> qqnorm(nra, main="Распределение лягушек")
```

**Шаг 8.** Добавляем прямую распределения (рисунок 144).

```
> qqline(nra, col=2)
```

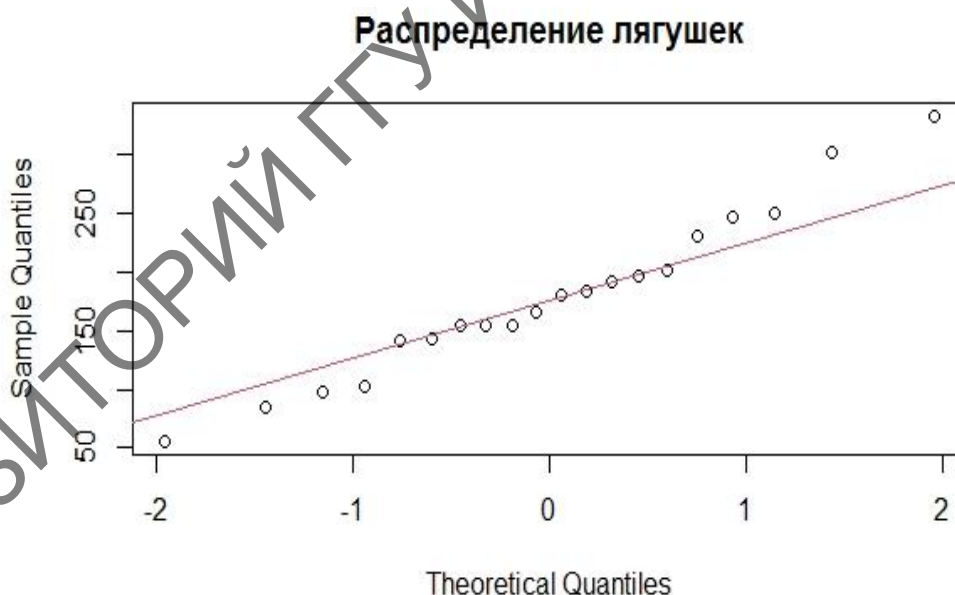


Рисунок 144 – График проверки соответствия распределения на нормальность

В связи с тем, что точки на концах линии стоят достаточно да- леко от самой прямой, то можно сделать вывод о том, что распреде- ление не соответствует закону нормального распределения.

## 2.2 Проверка достоверности выборки RCommander (при помощи GUI)

Пакет RCommander также позволяет провести проверку как достоверности расчёта средней, так и соответствия распределения значений выборки закону нормального распределения.

### 2.2.1 Проверка достоверности выборки

Для проверки достоверности выборки с использованием пакета RCommander воспользуемся теми же данными по численности остромордой лягушки, чтобы сравнить получившиеся результаты с предыдущими шагами.

**Шаг 1.** Готовим и загружаем данные в RCommander. Как это делается рассмотрено ранее (п. 1.2.2.1 темы 1; п. 2.4.2 темы 2).

**Шаг 2.** Далее необходимо выбрать в меню проведение параметрического одновыборочного теста по пути **Статистики** → **Средние** → **Одновыборочный t-тест...** (рисунок 145).

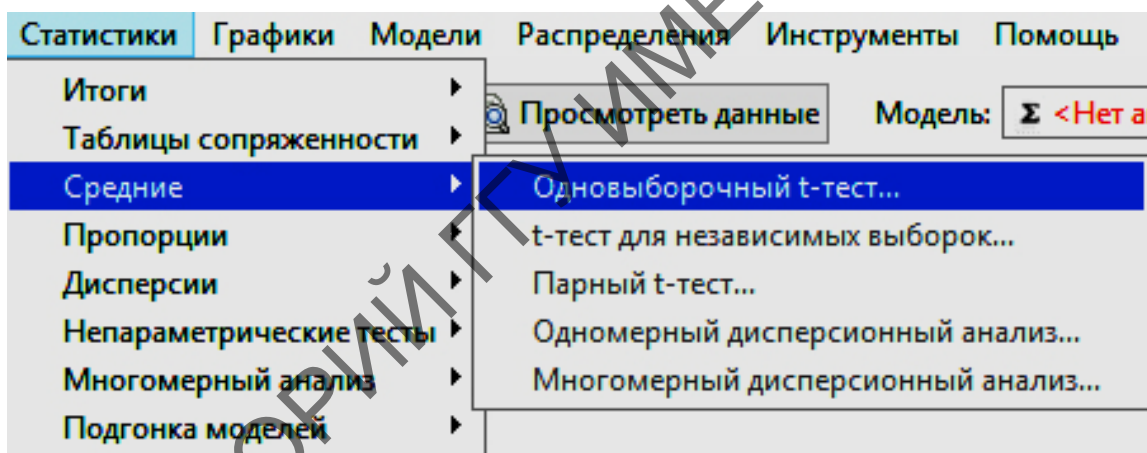


Рисунок 145 – Пункт меню **Одновыборочный t-тест**

**Шаг 3.** В появившемся диалоговом окне **Одновыборочный тест Стьюдента** необходимо выбрать нужную переменную и указать альтернативную гипотезу (в нашем случае – среднее в популяции) и нажать кнопку **ОК** (рисунок 146).

Результат теста, а также строка кода будут показаны либо в нижнем окне самого RCommander, либо в окне RStudio (если он открыт), либо в основном окне R.

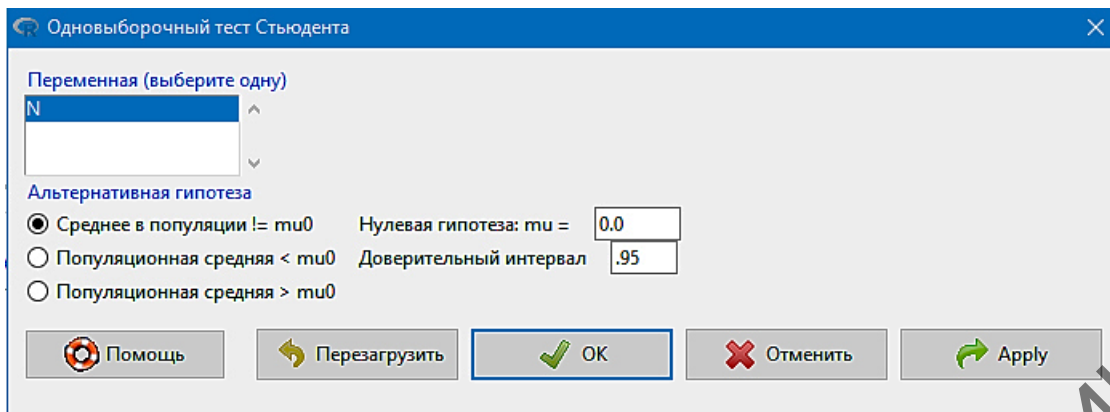


Рисунок 146 – Диалоговое окно **Одновыборочный тест Стьюдента**

```
> with(Rana, (t.test(N, alternative='two.sided', mu=0.0,
conf.level=.95)))
```

One Sample t-test

```
data: N
t = 11.379, df = 19, p-value = 6.318e-10
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 145.7094 211.3906
sample estimates:
mean of x
 178.55
```

Результат теста говорит о том, что рассчитанная средняя верна и статистически достоверна.

Далее проверим расчёт средней арифметической непараметрическим методом Уилкоксона.

**Шаг 4.** Для использования теста Уилкоксона необходимо перейти в меню по пути **Статистики** → **Непараметрические тесты** → **Одновыборочный тест Уилкоксона...** (рисунок 147).

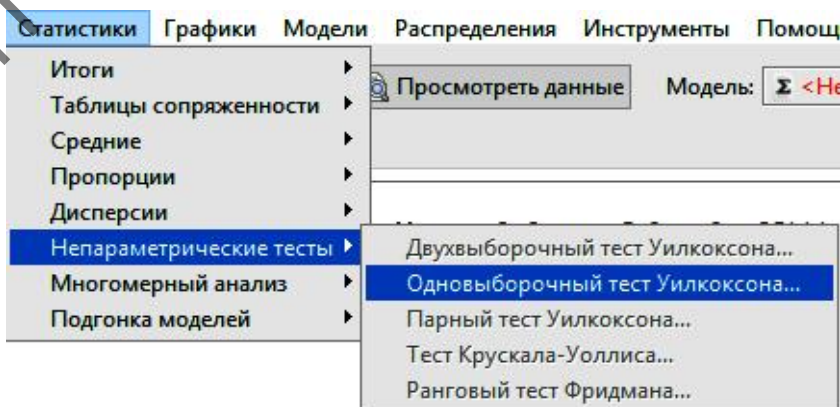


Рисунок 147 – Пункт меню **Одновыборочный тест Уилкоксона**

**Шаг 5.** Затем в диалоговом окне **Одновыборочный тест Уилкоксона** в закладке **Данные** необходимо выбрать переменную для оценки, а в закладке **Опции** выставить необходимые условия сравнения (рисунок 148).

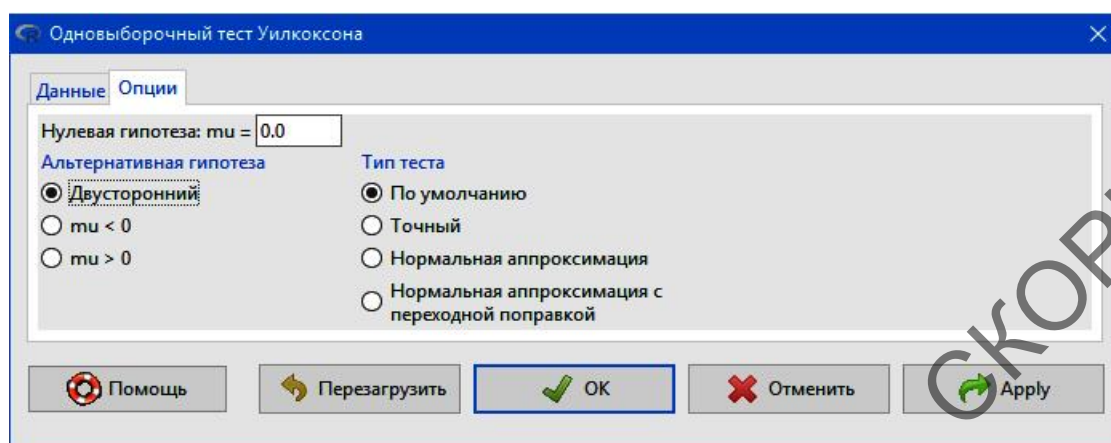


Рисунок 148 – Диалоговое окно **Одновыборочный тест Стьюдента**

Результат теста, а также и строка кода будут показаны либо в нижнем окне самого RCommander, либо в окне RStudio (если он открыт), либо в основном окне R.

```
> with(Rana, wilcox.test(N, alternative='two.sided',
mu=0.0))
```

```
wilcoxon signed rank test with continuity correction
data: N
V = 210, p-value = 0.00009556
alternative hypothesis: true location is not equal to 0
```

Напомним, что тест Уилкоксона работает только с медианой, и расчёты показали, что она статистически достоверна.

### 2.2.2 Проверка распределения на нормальность

Для того чтобы с использованием RCommander проверить распределение выборки на нормальность, необходимо использовать меню.

**Шаг 1.** Перейти в меню RCommander по пути **Статистики** → **Итоги** → **Тест нормальности...** (рисунок 149).

**Шаг 2.** Далее, в диалоговом окне **Тест нормальности** в боксе **Переменная** необходимо выбрать переменную (в нашем случае – N) и затем, ниже этого бокса среди перечня тестов нормальности следует выбрать пункт **Shapiro-Wilk** (рисунок 150) и нажать **ОК**.

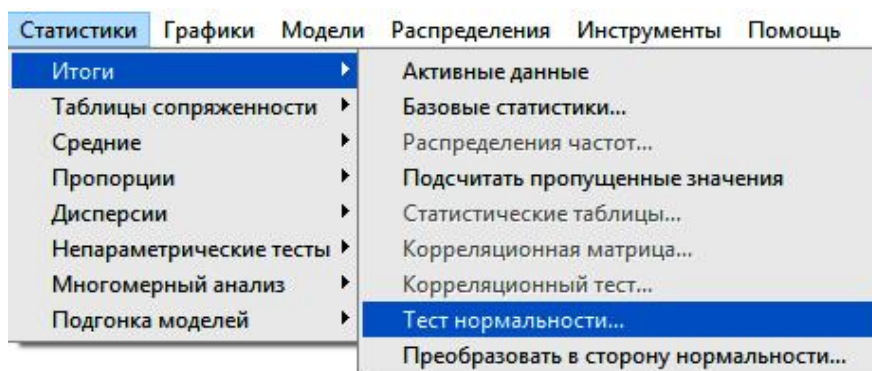


Рисунок 149 – Пункт меню **Тест нормальности...**

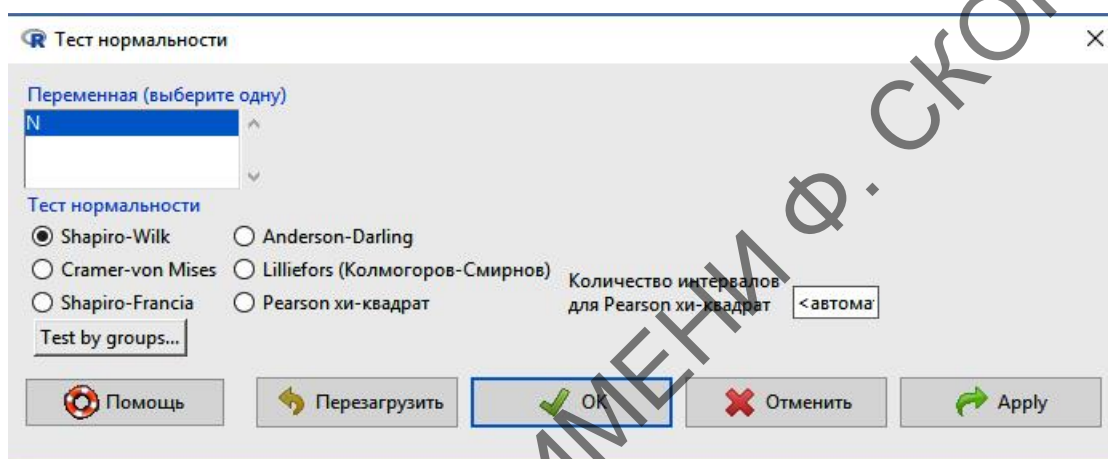


Рисунок 150 – Диалоговое окно **Тест нормальности**

В нижнем окне двухоконного RCommander или в окне R (при однооконном RCommander) будут изображены строка кода и результат теста:

```
> normalityTest(~N, test="shapiro.test", data=Rana)
```

```
shapiro-wilk normality test
```

```
data: N
W = 0.97312, p-value = 0.8188
```

Тест Колмогорова – Смирнова проведите самостоятельно.

Пакет RCommander также позволяет отобразить графически результаты теста на нормальность в виде квантильного графика. Посмотрим, как это сделать, продолжая рассматривать те же данные по численности остромордой лягушки.

**Шаг 3.** Для построения квантильного графика необходимо перейти по меню **Графики** → **Квантильный график...** (рисунок 151).



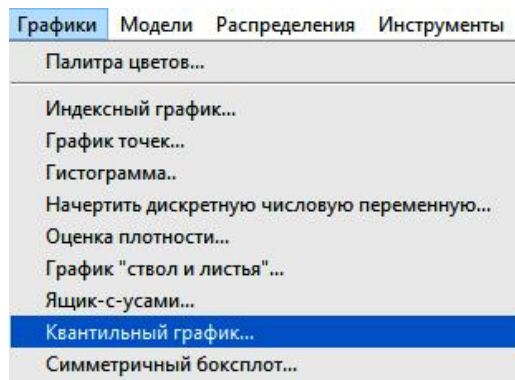


Рисунок 151 – Пункт меню **Квантильный график...**

**Шаг 4.** Далее, в диалоговом окне **Квантильный (QQ) график** (рисунок 152) в закладке **Данные** нужно выбрать переменную для анализа (в нашем случае – N), а в закладке **Опции** необходимо выставить опции так, как показано на рисунке 152, и нажать **ОК**.

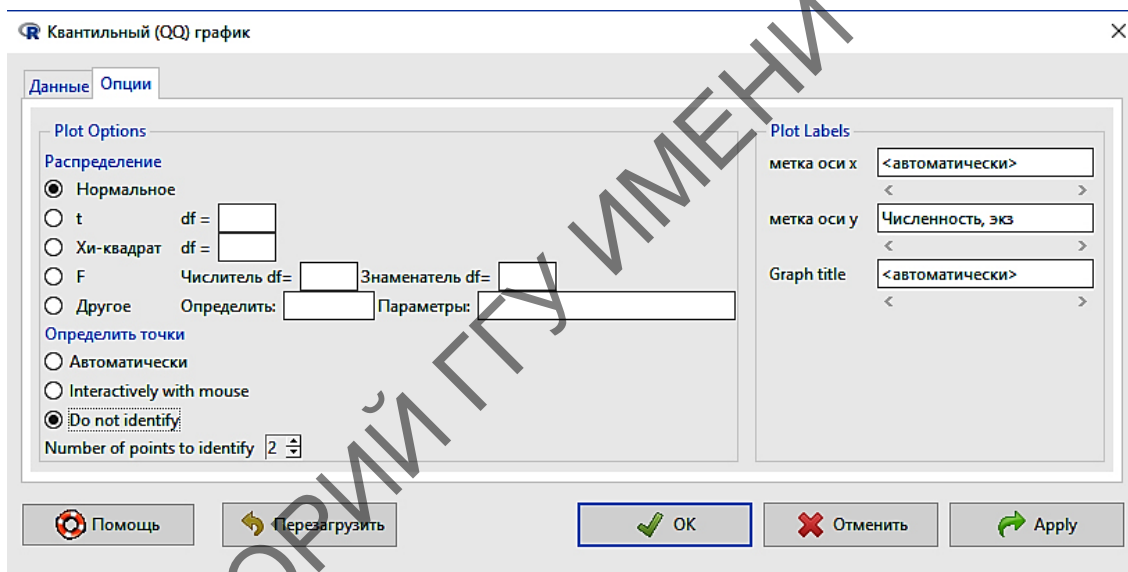


Рисунок 152 – Диалоговое окно **Квантильный (QQ) график**

Получившийся график сравните с рисунком 153. Следует отметить тот момент, что на графике, сделанном в пакете R Commander, в отличие от базового пакета можно отобразить диапазон, который включает область с 95 % точности (ограничена штриховой линией). И то, что только несколько значений вышли за этот диапазон, не делает распределение нормальным, но сильно приближает к нему (например, в том случае, если увеличить выборку, то оно станет нормальным).

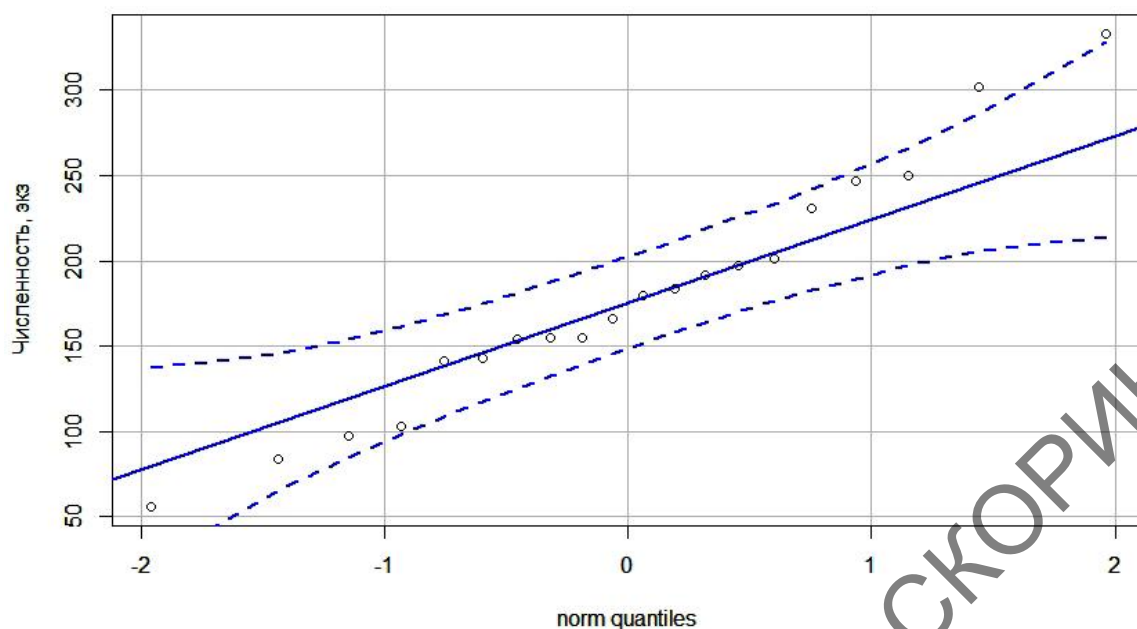


Рисунок 153 – График проверки соответствия распределения на нормальность

### 3 Сравнение выборочных средних

Для того чтобы изучить процесс сравнения двух выборочных средних, воспользуемся для учебных целей данными по исследованию численности жесткокрылых, попавших в почвенные ловушки, в зависимости от удаления от зоны отчуждения нефтяной скважины (таблица 10).

Таблица 10 – Численность жесткокрылых вблизи нефтескважины

a	5	5	7	8	9	4	11	12	14	13	18	15	19	19	19	21	25	28	30	33
b	3	3	2	1	6	4	5	8	6	7	7	6	8	9	9	12	13	14	17	22
d	12	12	14	24	26	12	13	14	21	15	15	13	14	17	18	18	28	28	38	31

Примечание: a – скважина весной; b – скважина летом; d – контроль

Для проведения статистического теста нужно выдвинуть две статистические гипотезы. Если распределение данных в выборке подчиняется закону нормального распределения, то нужно провести параметрический *t*-тест (тест Стьюдента). При этом следует также выяснить вид теста Стьюдента. В том случае, если переменные, которые мы хотим сравнить, были получены на разных объектах, мы будем использовать двухвыборочный *t*-тест для независимых

переменных (two sample  $t$ -test), который запускается при помощи функции `t.test()`. Например, если две сравниваемые выборки записаны в первом и втором столбцах таблицы данных `data`, то команда `t.test(data[,1], data[,2])` по умолчанию выдаст нам результаты двухвыборочного  $t$ -теста для независимых переменных.

Если же пары сравниваемых характеристик были получены на одном объекте, то есть переменные зависимы (например, частота пульса до и после физической нагрузки измерялась у одного и того же человека), надо использовать парный  $t$ -тест (paired  $t$ -test). Для этого в функции `t.test()` надо указать аргумент `paired=TRUE`. В нашем примере, если данные зависимы, надо использовать команду вида `t.test(data[,1], data[,2], paired=TRUE)`.

Если мы имеем дело с непараметрическими данными, то нужно провести непараметрический двухвыборочный тест Вилкоксона, «Wilcoxon test» (см. раздел 2.1.2 этой темы). Для этого в случае с независимыми переменными надо использовать функцию `wilcox.test()`. В случае с зависимыми выборками, аналогично  $t$ -тесту, в функции `wilcox.test()` надо использовать аргумент `paired=TRUE`.

### 3.1 Проведение сравнения выборочных средних в RStudio (при помощи командной строки)

**Шаг 1.** Необходимо создать три вектора:  $a$ ,  $b$  и  $d$ . В данном случае вектора  $a$  и  $b$  будут зависимыми векторами (так как учеты проводились в разное время года, но вблизи одной скважины), а вектор  $d$  будет независимым по отношению к ним.

**Шаг 2.** Объединяем все 3 вектора в один датафрейм `beetles`.

**Шаг 3.** Сравним независимые вектора  $a$  и  $d$  параметрическим критерием Стьюдента:

```
> t.test(beetles[,1], beetles[,3])
```

```
welch Two Sample t-test
```

```
data: beetles[, 1] and beetles[, 3]
```

```
t = -1.3319, df = 37.364, p-value = 0.191
```

```
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
```

```
-8.570683  1.770683
```

```
sample estimates:
```

```
mean of x mean of y  
15.75      19.15
```



Как результат, мы видим, что общая численность жесткокрылых у скважины и на контроле статистически значимо не отличались: критерий Стьюдента составил  $t = -1.3319$  при невысоком уровне значимости  $p\text{-value} = 0.191$ , который значительно больше даже минимального  $0,05$ .

**Шаг 4.** Сравним зависимые вектора  $a$  и  $b$  параметрическим критерием Стьюдента:

```
> t.test(beetles[,1], beetles[,2], paired=TRUE)
```

```
Paired t-test
```

```
data: beetles[,1] and beetles[,2]
t = 8.5173, df = 19, p-value = 0.00000006527
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 5.770107 9.529893
sample estimates:
mean of the differences
              7.65
```

Общая численность жесткокрылых около скважины достоверно отличалась по временам года: критерий Стьюдента составил  $t = 8.5173$  при очень высоком уровне значимости  $p\text{-value} = 0.00000006527$ , который значительно меньше даже максимального  $0,001$ .

**Шаг 5.** Проведем сравнение независимых векторов  $a$  и  $d$  непараметрическим тестом Вилкоксона:

```
> wilcox.test(beetles[,1], beetles[,3])
```

```
wilcoxon rank sum test with continuity correction
```

```
data: beetles[, 1] and beetles[, 3]
W = 154.5, p-value = 0.2226
alternative hypothesis: true location shift is not equal
to 0
```

Как результат, мы видим, что общая численность жесткокрылых у скважины и на контроле статистически значимо не отличались: показатель Вилкоксона составил  $w = 154.5$  при невысоком уровне значимости  $p\text{-value} = 0.2226$ , который значительно больше даже минимального  $0,05$ .

**Шаг 6.** Сравним зависимые вектора  $a$  и  $b$  непараметрическим тестом Вилкоксона:

```
> wilcox.test(beetles[,1], beetles[,2], paired=TRUE)
wilcoxon signed rank test with continuity correction

data:  beetles[, 1] and beetles[, 2]
V = 190, p-value = 0.0001412
alternative hypothesis: true location shift is not equal
to 0
```

Общая численность жесткокрылых около скважины достоверно отличалась по временам года: показатель Вилкоксона составил  $V = 190$  при очень высоком уровне значимости  $p\text{-value} = 0.0001412$ , который значительно меньше даже максимального  $0,001$ .

**Шаг 7.** Проверим вектор  $a$  на нормальность.

```
> qqnorm(a)
> qqline(a, col=2)
```

И сверим полученный результат с рисунком 154.

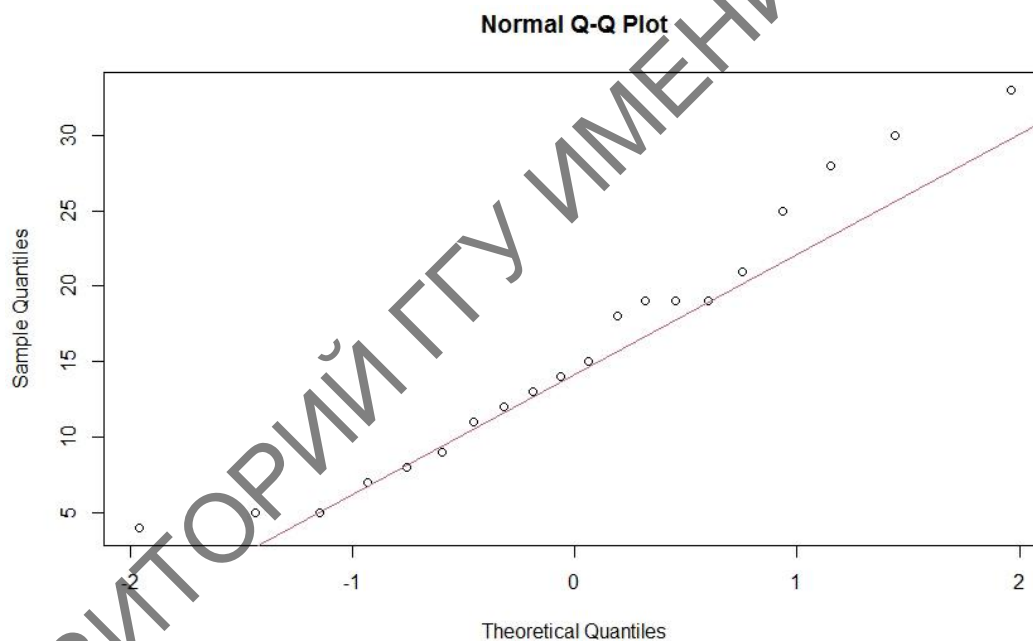


Рисунок 154 – График проверки соответствия распределения переменной  $a$  на нормальность

**Шаг 8.** Проверим каждый из оставшихся векторов на нормальность и определим наиболее адекватный метод сравнения.

```
> qqnorm(b)
> qqline(b, col=2)
> qqnorm(d)
> qqline(d, col=2)
```

Таким образом, исходя из полученных результатов наиболее адекватными в данном случае будут непараметрические тесты.

### 3.2 Проведение сравнения выборочных средних в RCommander (при помощи GUI)

Для того чтобы сравнить две выборочные средние между собой в пакете RCommander, необходимо использовать меню. Воспользуемся в учебных целях данными по жесткокрылым (таблица 10). Предварительно подготовим и загрузим данные в пакет RCommander (п. 1.2.2.1 темы 1; п. 2.4.2 темы 2).

**Шаг 1.** Для сравнения двух зависимых выборок параметрическим  $t$ -критерием Стьюдента нужно перейти по пути в меню **Статистики** → **Средние** → **Парный  $t$ -тест...** (рисунок 155).

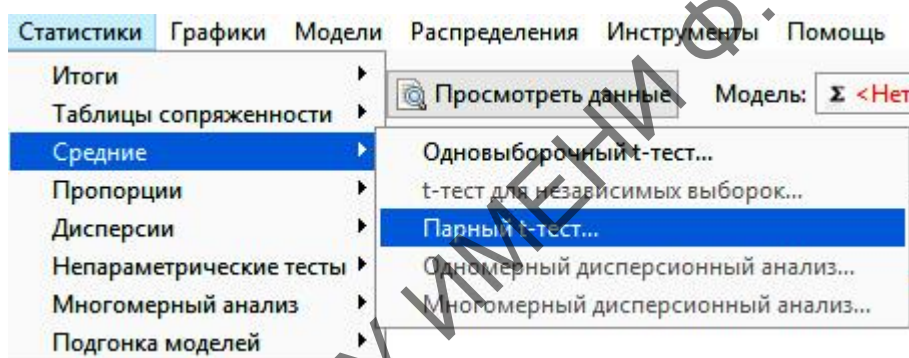


Рисунок 155 – Пункт меню **Парный  $t$ -тест...**

**Шаг 2.** В диалоговом окне **Парный тест Стьюдента** в закладке **Данные** выберите две переменные для сравнения между собой (рисунок 156).

**Шаг 3.** Далее, в закладке **Опции** диалогового окна **Парный тест Стьюдента** выберите альтернативную гипотезу и уровень значимости для сравнения (рисунок 157) и нажмите кнопку **ОК**. Результаты теста будут выданы либо в нижнее окно самого RCommander, либо в окно непосредственно R.

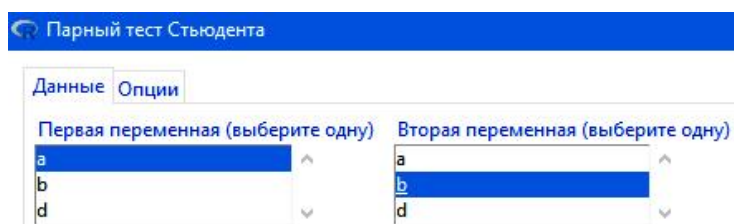


Рисунок 156 – Закладка **Данные** диалогового окна **Парный тест Стьюдента**

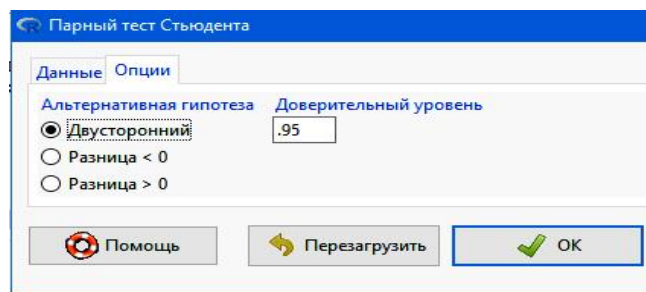


Рисунок 157 – Закладка **Опции** диалогового окна **Парный тест Стьюдента**

Сравнение независимых выборок делается схожим образом.

**Шаг 4.** Для сравнения независимых выборок необходимо перейти по пути в меню **Статистики** → **Средние** → **t-тест для независимых выборок...**

**Шаг 5.** В появившемся диалоговом окне в закладке **Данные** необходимо выбрать независимую переменную в левом боксе и зависимую, с которой проводится сравнение – в правом.

**Шаг 6.** В закладке **Опции** нужно выбрать альтернативную гипотезу, уровень значимости, а также указать, учитывать ли при сравнении равенство дисперсий (в том случае, если используется классический тест Стьюдента, то надо отметить мышкой бокс **Да**, если же планируется использовать метод Уэлча, то надо отметить бокс **Нет**). После чего нажать кнопку **ОК**. Результат будет отображён там же. Где и в предыдущем случае.

Если необходимо сравнить выборки, которые не подчиняются нормальному распределению, используется тест Уилкоксона.

**Шаг 7.** В случае несоответствия нормальному распределению при сравнении зависимых выборок необходимо перейти по пути в меню **Статистики** → **Непараметрические тесты** → **Парный тест Уилкоксона...** (рисунок 158).

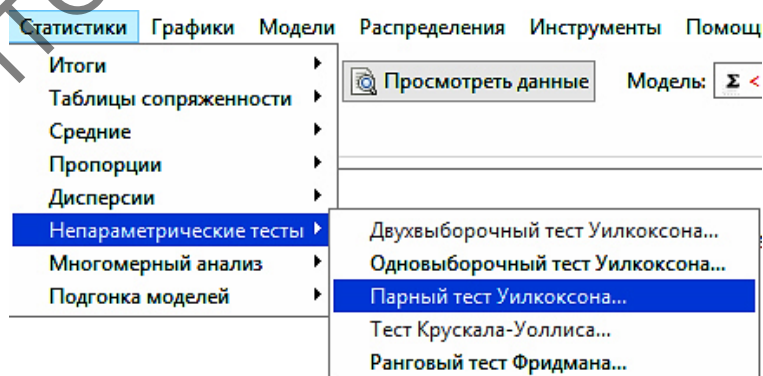


Рисунок 158 – Пункт меню **Парный тест Уилкоксона...**

**Шаг 8.** Далее, в диалоговом окне **Парный тест Уилкоксона** в закладках **Данные** и **Опции** необходимо выбрать переменные для сравнения и выставить необходимые параметры сравнения. Настройки похожи на таковые при парном тесте Стьюдента, поэтому сложности не вызывают.

**Шаг 9.** При сравнении двух независимых выборок, не подчиняющихся закону нормального распределения, используется двухвыборочный тест Уилкоксона, который выбирается в меню по пути **Статистики** → **Непараметрические тесты** → **Двухвыборочный тест Уилкоксона...** Настройки окна схожи с таковыми при подобном тесте Стьюдента.

Результаты тестов будут отражены либо в нижнем окне самого RCommander, либо в окне R.

## Вопросы для самоконтроля

1 Какие функции используются языком программирования R для описательной статистики параметров выборки?

2 Опишите функции для проверки выборочного среднего выборки при нормальном распределении.

3 Как в языке программирования R проверяется достоверность выборки при распределении отличном от нормального?

4 Какая функция в языке программирования R используется для сравнения независимых выборок между собой при нормальном распределении обоих?

5 Опишите синтаксис функции сравнения зависимых выборок с нормальным распределением.

6 Как в языке программирования R можно сравнить две независимые выборки, не подчиняющиеся закону нормального распределения?

7 Какая функция в языке программирования R используется для сравнения зависимых выборок между собой при распределении, отличном от нормального в обоих выборках?

8 Каким образом можно проверить распределение данных в выборке на соответствие закону нормального распределения при помощи командной строки и GUI?

## Задания для самоконтроля

1) Были получены следующие данные о численности травяной лягушки в кварталах двух лесных массивов:

Лесной	186	190	165	182	182	182	180
массив	173	157	179	164	146	173	144
№ 1	156	156	165	160	160	161	144
	153	152	151	173			
Лесной	162	163	190	188	147	146	145
массив	157	162	186	175	147	145	145
№ 2	155	174	180	148	175	145	144
	153	165	141	164			

Сформируйте датафрейм по численности травяной лягушки. Рассчитайте описательную статистику для каждого из лесных массивов. Сравните эти две независимые выборки между собой различными способами. Выясните, есть ли различия в численности лягушек из двух лесных массивов?

2) Были изучены две выборки численности жесткокрылых на песчаном 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 датафрейм по численности жуков. Рассчитайте описательную статистику для численности жуков до и после посадок. Сравните эти две зависимые выборки между собой различными способами. Выясните, есть ли различия в численности жесткокрылых до и после посадки травы?

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

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

Сформируйте датафрейм по проросткам ячменя. Рассчитайте описательную статистику для проростков ячменя до и после внесения удобрений. Сравните эти две зависимые выборки между собой различными способами. Выясните, есть ли различия в длине проростков ячменя до и после внесения удобрений?

## Литература по теме

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

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

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

4 Field, A. Discovering statistics using R / A. Field, J. Miles, Z. Field. – London : SAGE Publications Ltd, 2012. – 1193 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](http://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 5. ПРОВЕДЕНИЕ ПАРНОГО ПАРАМЕТРИЧЕСКОГО И НЕПАРАМЕТРИЧЕСКОГО КОРРЕЛЯЦИОННОГО АНАЛИЗА В R

- 1 Понятие о корреляционном анализе.
- 2 Параметрический коэффициент корреляции Пирсона.
- 3 Непараметрический коэффициент корреляции Спирмена.

## 1 Понятие о корреляционном анализе

**Корреляционный анализ** – статистический метод изучения взаимосвязи между двумя и более случайными величинами. В качестве случайных величин в эмпирических исследованиях выступают значения переменных, измеряемые свойства исследуемых объектов наблюдения. Суть корреляционного анализа заключается в расчете коэффициентов корреляции. Коэффициенты корреляции могут принимать, как правило, положительные и отрицательные значения (от  $-1$  до  $+1$ ). Знак коэффициента корреляции позволяет интерпретировать направление связи, а абсолютное значение – силу связи.

### *Виды корреляционной связи:*

- 1) параметрическая (линейный коэффициент корреляции Пирсона);
- 2) непараметрическая (ранговая):
  - а) коэффициент корреляции Спирмена;
  - б) коэффициент ранговой корреляции Кендалла;
  - в) коэффициент корреляции знаков Фехнера;
  - г) коэффициент множественной ранговой корреляции (коэффициент Конкордации);
  - д) меры оценки связи между дихотомическими переменными.

## 2 Параметрический коэффициент корреляции Пирсона

### 2.1 Расчёт параметрического коэффициента корреляции Пирсона в RStudio (при помощи командной строки)

В качестве учебного примера воспользуемся данными, полученными на звероферме о весе  $x$  (в кг) и длине туловища  $y$  (в см) 17 серебристо-чёрных лисиц (таблица 11).



Таблица 11 – Вес и длина туловища серебристо-чёрных лисиц

<i>x</i>	4,7	4,6	5,2	5,1	5,3	5,3	4,6	4,8	5,8	5,7	4,5	5,7	5,0	4,8	4,7	5,2	4,6
<i>y</i>	70	65	69	70	66	68	65	71	69	68	57	73	65	67	71	62	69

Необходимо определить, есть ли корреляционная связь между весом и длиной туловища у лисиц?

**Шаг 1.** Перед проведением расчётов необходимо составить 2 вектора: *x* и *y*, соответствующие весу и длине туловища соответственно:

```
> x <- c(4.7, 4.6, 5.2, 5.1, 5.3, 5.3, 4.6, 4.8, 5.8,
5.7, 4.5, 5.7, 5.0, 4.8, 4.7, 5.2, 4.6)
```

```
> y <- c(70, 65, 69, 70, 66, 68, 65, 71, 69, 68, 57, 73,
65, 67, 71, 62, 69)
```

**Шаг 2.** Далее нужно создать датафрейм под названием *foxies* (лисы) из ранее созданных векторов:

```
> foxies <- data.frame(weight_kg=x, Length_sm=y)
```

**Шаг 3.** Проверим получившийся датафрейм на предмет ошибок и в случае необходимости исправим их:

```
> foxies
  weight_kg Length_sm
1         4.7         70
2         4.6         65
3         5.2         69
4         5.1         70
5         5.3         66
6         5.3         68
7         4.6         65
8         4.8         71
9         5.8         69
10        5.7         68
11        4.5         57
12        5.7         73
13        5.0         65
14        4.8         67
15        4.7         71
16        5.2         62
17        4.6         69
```

**Шаг 4.** Для быстрого расчёта коэффициента корреляции в языке программирования R существует функция `cor()`. Рассчитаем коэффициент корреляции Пирсона для нашего датафрейма:

```
> cor(x, y, method = "pearson")
[1] 0.3422742
```

Отмечаем, что данная функция выводит только коэффициент корреляции без всяких подробностей.

**Шаг 5.** Для большей наглядности рассчитаем корреляционную матрицу для всего нашего датафрейма:

```
> cor(foxies)
      weight_kg Length_sm
weight_kg 1.0000000 0.3422742
Length_sm 0.3422742 1.0000000
```

**Шаг 6.** В том случае, если необходима более подробная информация о корреляционной связи и её достоверности, используется функция `cor.test()`. Рассчитаем при помощи её подробный коэффициент корреляции Пирсона для нашего датафрейма:

```
> cor.test(x, y, method = "pearson")
Pearson's product-moment correlation
data:  x and y
t = 1.4108, df = 15, p-value = 0.1787
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.1656165  0.7066640
sample estimates:
      cor
0.3422742
```

**Шаг 7.** Далее построим коррелограмму для нашего датафрейма (рисунок 159).

```
> plot(x, y)
```

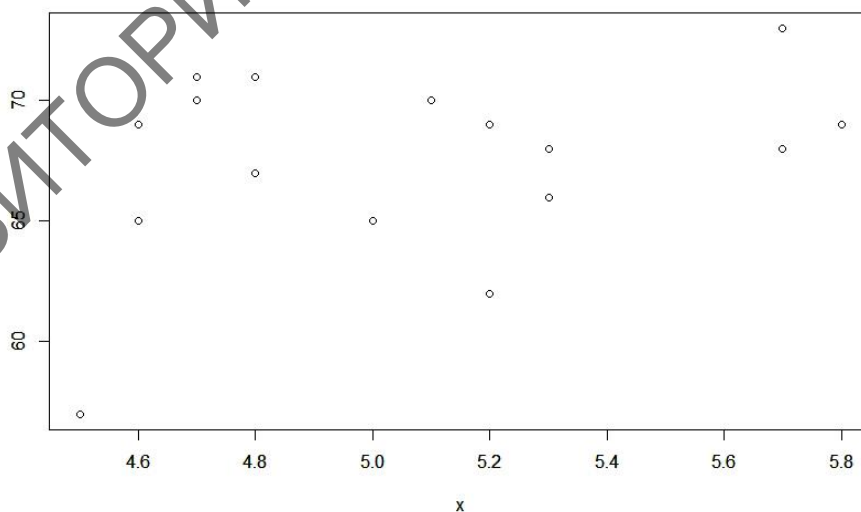


Рисунок 159 – Коррелограмма соотношения веса и длины серебристо-чёрных лисиц

**Шаг 8.** После чего для наглядности добавим на нее линию регрессии (рисунок 160).

```
> abline(lm(y ~ x), col='red')
```

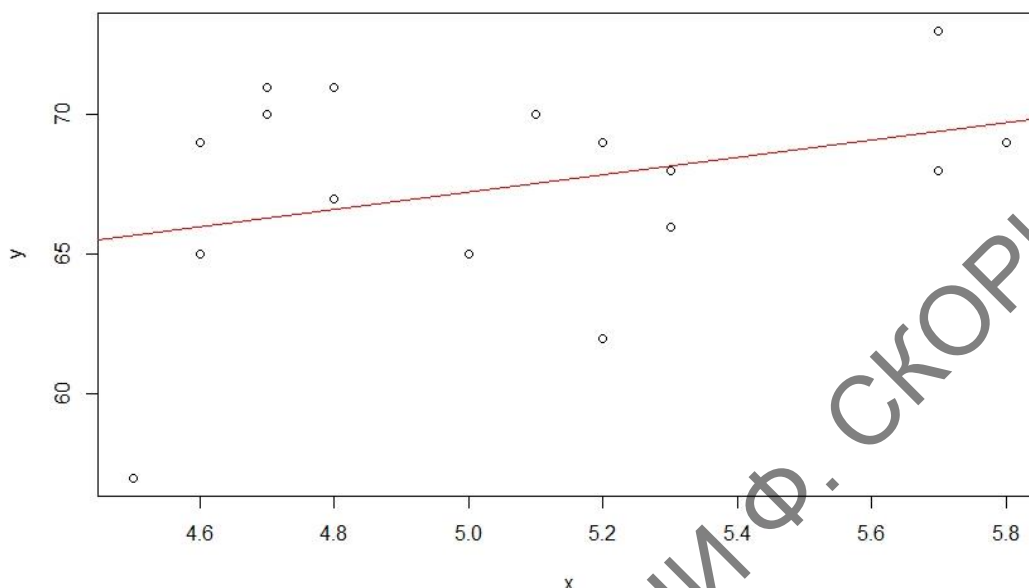


Рисунок 160 – Коррелограмма соотношения веса и длины серебристо-чёрных лисиц с линией регрессии

С учетом того, что коэффициент корреляции низкий (+0,34), то можно сделать предварительный вывод, что вес лисиц очень слабо коррелирует с длиной хвоста. Однако, в то же время уровень значимости *p-value* в 0,1787 говорит о том, что полученный коэффициент корреляции недостоверен. В связи с чем необходимо проверить каждый из наших векторов на нормальность. Это можно сделать при помощи функции `qqnorm()` – каждый вектор в отдельности (см. тему 4), а используя пакет **car** можно это определить визуально сразу по каждой переменной датафрейма.

**Шаг 9.** Загружаем и включаем пакет **car**.

```
> install.packages("car")  
> library(car)
```

**Шаг 10.** Далее строим комплексный график с простой линией регрессии, используя функцию `scatterplotMatrix()` (рисунок 161).

```
> scatterplotMatrix(foxies, diagonal = "histogram",  
smooth = FALSE)
```

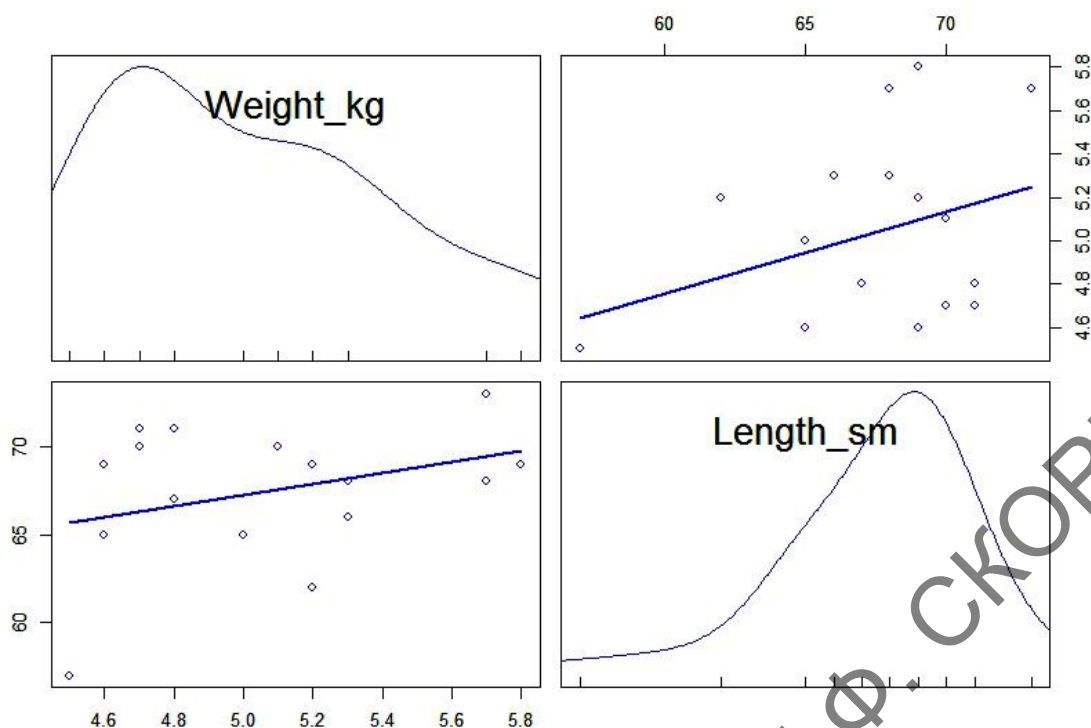


Рисунок 161 – Комплексный график переменных веса и длины серебристо-чёрных лисиц

По графику каждой переменной видно, что распределение далеко от нормального, в связи с чем имеет смысл использовать непараметрический (ранговый) коэффициент корреляции Спирмена.

## 2.2 Расчёт параметрического коэффициента корреляции Пирсона в RCommander (при помощи GUI)

Расчёт параметрического коэффициента корреляции Пирсона в пакете RCommander не составляет труда. Используем всё те же данные о весе и росте серебристо-белых лисиц из таблицы 11.

**Шаг 1.** Загружаем активные данные в пакет. Если ранее они были созданы и загружены в RStudio, и эта программа открыта, то их загрузка делается в меню по пути **Данные** → **Активные данные** → **Выбрать активный набор данных...** с указанием названия активных данных (в нашем случае – *foxies*). Если же вы работаете в RCommander с нуля, то данные нужно создать и загрузить любым из рассмотренных ранее способов (п. 1.2.2.1 темы 1; п. 2.4.2 темы 2).

**Шаг 2.** После загрузки активных данных необходимо перейти в меню по следующему пути **Статистики** → **Итоги** → **Корреляционный тест...** (рисунок 162).

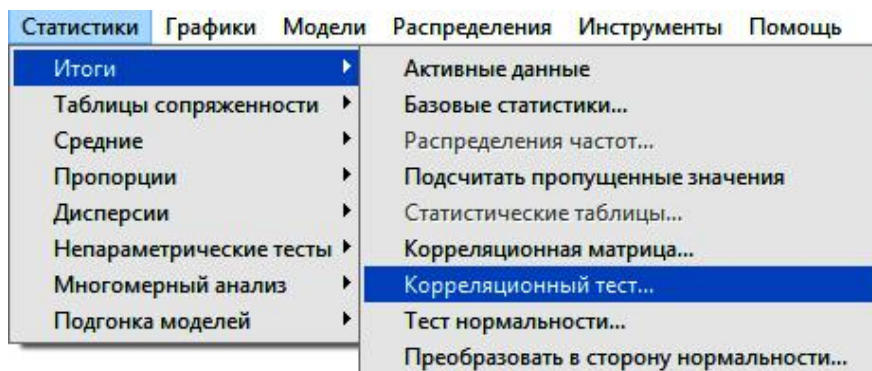


Рисунок 162 – Пункт меню **Корреляционный тест...**

**Шаг 3.** Далее в диалоговом окне **Корреляционный тест** необходимо выбрать две переменные для анализа (в нашем случае – по весу и росту) и указать тип корреляции – Пирсона (рисунок 163). После чего нажать кнопку **ОК**. Результат отразится в окне пакета или в среде R.

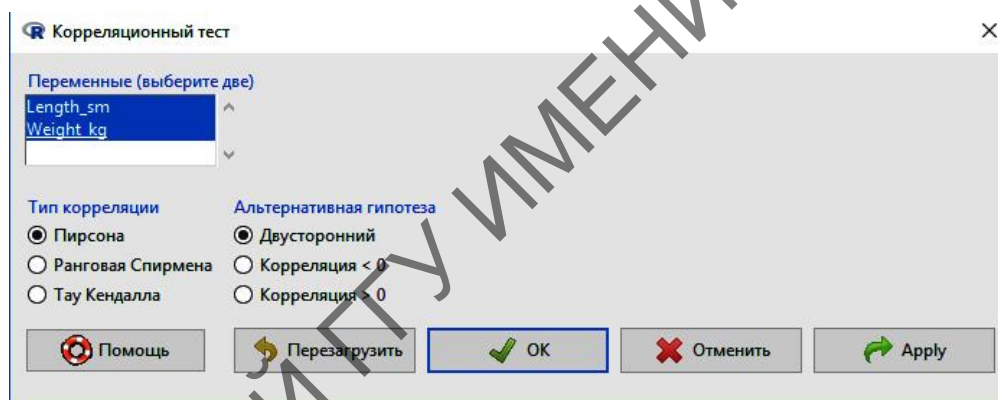


Рисунок 163 – Диалоговое окно **Корреляционный тест**

```
> with(foxies, cor.test(Length_sm, weight_kg, alternative="two.sided", method="pearson"))
```

Pearson's product-moment correlation

data: Length\_sm and weight\_kg

t = 1.4108, df = 15, p-value = 0.1787

alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval:

-0.1656165 0.7066640

sample estimates:

cor  
0.3422742

Полученный результат соответствует расчёту корреляционного теста Пирсона функцией `cor.test()`.

Если необходимо отразить корреляционную матрицу, то следует сделать следующее.

**Шаг 4.** Перейти в меню по пути **Статистики** → **Итоги** → **Корреляционная матрица...** (рисунок 164).

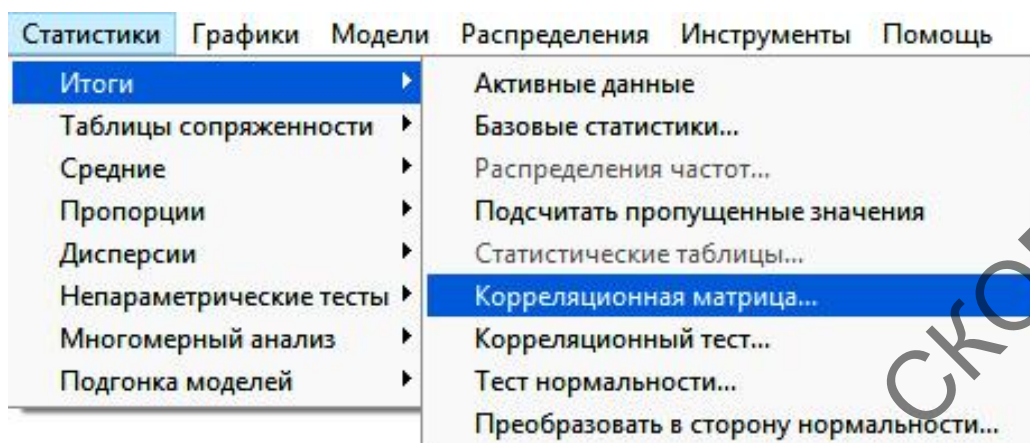


Рисунок 164 – Пункт меню **Корреляционная матрица...**

**Шаг 5.** Далее в диалоговом окне **Корреляционная матрица** (рисунок 165) необходимо выбрать две переменные и указать тип корреляции (в нашем случае – Пирсона). После чего нажать кнопку **ОК**.

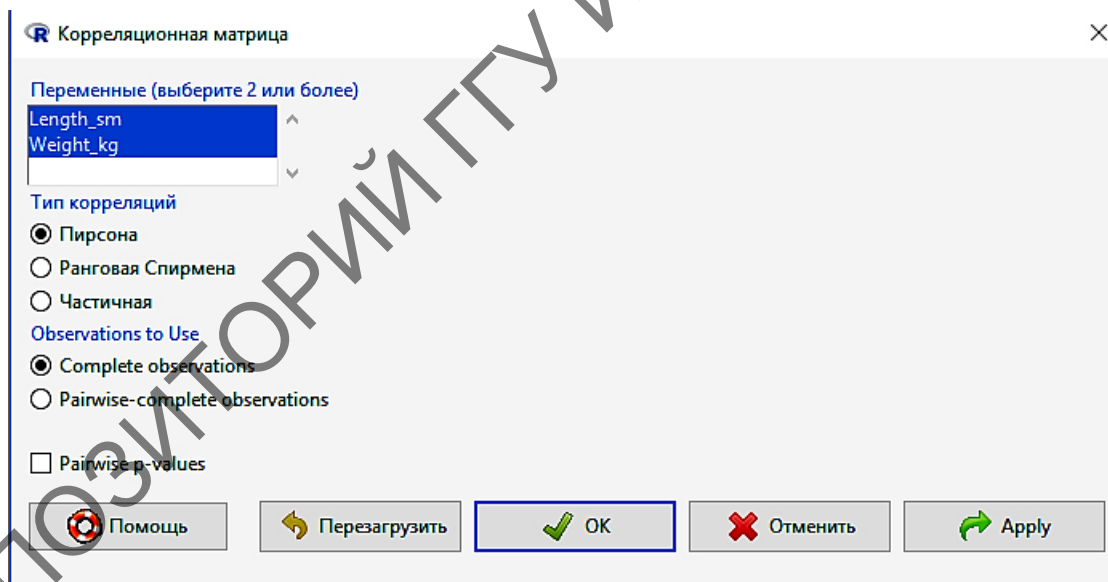


Рисунок 165 – Диалоговое окно **Корреляционная матрица**

В результате расчётов будет создана и отображена корреляционная матрица полностью идентичная, созданной в RStudio.

## 3 Непараметрический коэффициент корреляции Спирмена

### 3.1 Расчёт непараметрического коэффициента корреляции Спирмена в RStudio (при помощи командной строки)

В качестве учебного примера воспользуемся теми же данными о весе и длине туловища серебристо-чёрных лисиц (таблица 11).

**Шаг 1.** Рассчитываем сразу подробный коэффициент корреляции Спирмена, используя функцию `cor.test()`.

```
> cor.test(x, y, method = "spearman")
Spearman's rank correlation rho
data:  x and y
S = 597.83, p-value = 0.2995
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.2673699
```

Как итог, полученный коэффициент корреляции ещё ниже – 0.2673699, и к тому же результат также не показал достоверного коэффициента корреляции ( $p\text{-value} = 0.2995$ , что намного больше, чем 0,05). Это может свидетельствовать как об отсутствии связи, так и о недостаточном количестве данных.

**Шаг 2.** Отображаем графически результаты расчетов, как и для коэффициента корреляции Пирсона (см. шаги с 7 по 9).

**Шаг 3.** Создадим коррелограмму с областью 95 % достоверности (п. 1.1.2 темы 3). Для этого предварительно загрузим и включим графический пакет **ggplot2**.

```
> install.packages("ggplot2")
> library(ggplot2)
```

**Шаг 4.** Введем новую переменную  $f$  и присвоим ей значение графика с областью 95 % достоверности:

```
> f <- ggplot(foxies, aes(x = x, y = y))
```

**Шаг 5.** Строим график с областью 95 % достоверности и величиной точки в 2.5 пункта (рисунок 166).

```
> f+
+geom_point(size = 2.5)+
+stat_smooth(method = "lm)
```

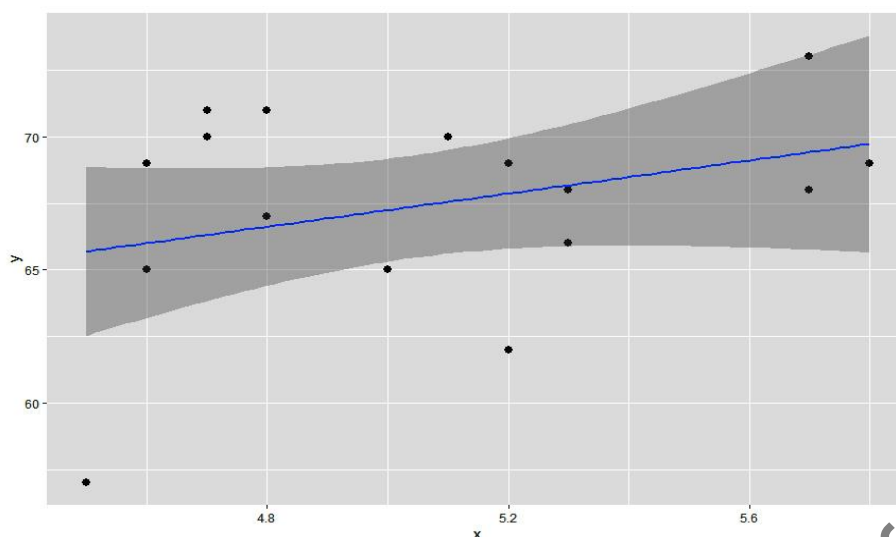


Рисунок 166 – Коррелограмма зависимости роста серебристо-чёрных лисиц от их веса с зоной 95 % достоверности

В том случае, если есть предположение, что зависимость имеет криволинейный вид, то это можно проверить с использованием того же графического пакета **ggplot2**. Для учебных целей используем те же данные о весе и длине тела лисиц и промежуточную переменную  $f$ , а в качестве метода будет уже не *lm* (*line model* – линейная модель), а *loess* – криволинейная модель.

**Шаг 6.** Строим криволинейную коррелограмму с областью 95 % достоверности и величиной точки в 2.5 пункта (рисунок 167).

```
> f+
+geom_point(size = 2.5)+
+stat_smooth(method = loess)
```

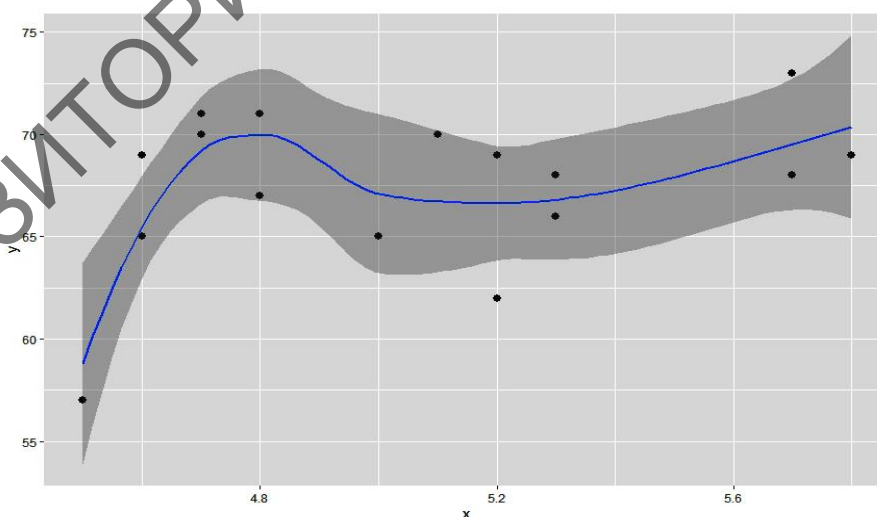


Рисунок 167 – Криволинейная коррелограмма



Функция `stat_smooth()` позволяет также разнообразить график в цветовой гамме при помощи аргументов `fill = ""` (заполнение зоны 95% вероятности) и `colour = ""` (цвет линии тренда). Например, если привести аргументы `fill = "yellow"` и `colour = "red"`, то линия тренда станет красной, а область с 95 % вероятностью – желтой. Проверьте это самостоятельно (не забудьте отделить аргументы друг от друга запятой с пробелом).

### 3.2 Расчёт непараметрического коэффициента корреляции Спирмена в RCommander (при помощи GUI)

Расчёт рангового (непараметрического) коэффициента корреляции Спирмена в пакете RCommander также осуществляется через пункт меню **Корреляционный тест**.

**Шаг 1.** Для расчёта коэффициента корреляции Спирмена необходимо в меню перейти по пути **Статистики** → **Итоги** → **Корреляционный тест...** (рисунок 162).

**Шаг 2.** В диалоговом окне **Корреляционный тест** необходимо выбрать переменные для анализа и сам тип анализа. В нашем случае – это **Ранговая Спирмена** (рисунок 168), и нажать кнопку **ОК**.

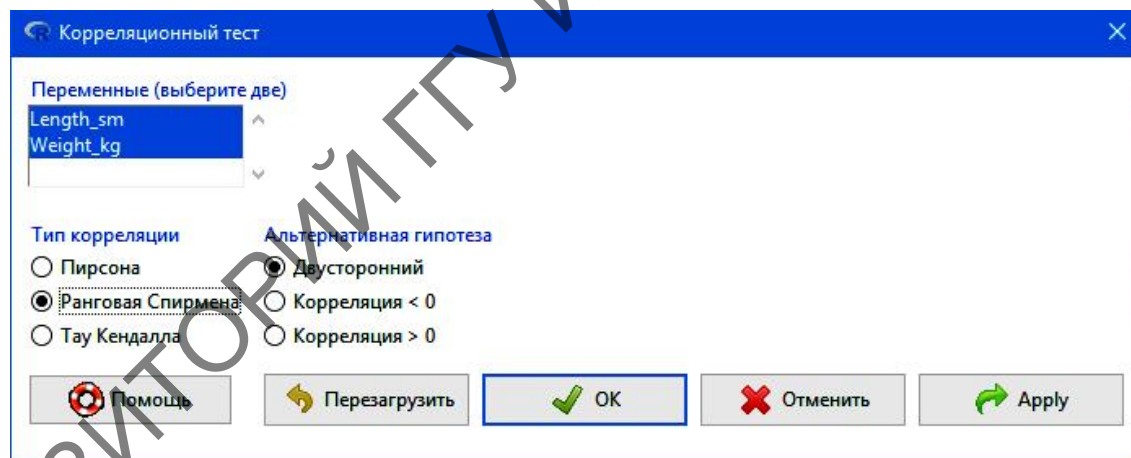


Рисунок 168 – Диалоговое окно **Корреляционный тест** при выборе типа корреляции Спирмена

Строка кода и полученный результат будет отражён либо в окне самого пакета, либо в окне среды R.

## Вопросы для самоконтроля

- 1 Что представляет собой коррелятивная связь и каковы её отличия от функциональной?
- 2 Какими свойствами обладает коррелятивная связь?
- 3 Какие виды коэффициента корреляции используются в математической статистике?
- 4 Опишите синтаксис функции расчёта параметрического коэффициента корреляции Пирсона в языке программирования R.
- 5 Охарактеризуйте функцию расчёта рангового коэффициента корреляции Спирмена, укажите её синтаксис.
- 6 Как можно рассчитать коэффициент корреляции при помощи средств GUI в пакете RCommander?

## Задания для самоконтроля

- 1) Получены данные о весе (в г) левой камеры сердца ( $x$ ) и длине ядер (в  $\mu$ ) в мышцах сердца ( $y$ )

$x$	206	222	255	261	274	288	292	291	305	327	373	398	459	633
$y$	16,7	17,9	16,0	20,6	19,5	19,7	11,8	20,9	23,1	13,5	19,7	22,8	19,5	28,3

Определите необходимый коэффициент корреляции и рассчитайте его в кратком и подробном виде. Постройте график рассеяния. Сделайте вывод.

- 2) Необходимо установить, есть ли корреляция между высотой головы ( $x$ ) и длиной 3-го членика усика ( $y$ ) *Drosophila funebris*:

$x$	16	17	14	16	15	17	16	19	19	18	18	18	14	17	16	16	14	18
$y$	29	31	32	33	32	33	33	36	36	35	35	35	35	33	31	31	31	35
$x$	15	13	15	14	17	15	16	15	15	16	15	16	15	16	18	17	14	15
$y$	33	30	32	31	35	33	33	32	30	33	33	33	30	31	34	34	31	33

Определите необходимый коэффициент корреляции и рассчитайте его в кратком и подробном виде. Постройте график рассеяния. Сделайте вывод.

3) Получены следующие данные о продолжительности беременности у кроликов при различных размерах помета (число крольчат в помете ( $x$ ) и длительность беременности в днях ( $y$ )):

$x$	1	8	3	5	7	8	4	8	3	4	4	8	8	5	7	6	6	5	6	6
$y$	33	30	31	31	31	32	31	31	32	33	32	31	31	31	31	30	31	32	32	31
$x$	6	5	7	8	10	6	7	6	7	6	5	10	7	8	8	6	5	6	5	4
$y$	32	32	31	32	31	31	30	31	31	32	31	30	32	32	31	31	31	32	30	31

Определите необходимый коэффициент корреляции и рассчитайте его в кратком и подробном виде. Постройте график рассеяния. Сделайте вывод.

4) Были получены следующие данные о весе ягнят-баранчиков ( $y$ ) и весе баранов – их отцов ( $x$ ) (в кг).

$x$	76,6	72,2	67,0	66,5	63,3	65,4	63,9	63,1	63,0	62,5	62,2
$y$	4,56	4,79	4,49	4,32	4,59	4,32	4,67	4,29	4,57	4,20	4,12

Есть ли корреляция между весом баранчиков и весом их отцов? Определите необходимый коэффициент корреляции и постройте график рассеяния. Сделайте вывод.

### Литература по теме

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

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

3 Field, A. Discovering statistics using R / A. Field, J. Miles, Z.Field. – London : SAGE Publications Ltd, 2012. – 1193 p.

4 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.

5. Owen, J. Scientific Programming and Simulation Using R / J. Owen, R. Maillardet, A. Robinson. – London, New York : CRC-Press, 2009. – 455 p.

## ТЕМА 6. ПРОВЕДЕНИЕ РЕГРЕССИОННОГО АНАЛИЗА В R (ЛИНЕЙНАЯ РЕГРЕССИЯ)

- 1 Понятие о регрессионном анализе.
- 2 Выполнение линейного регрессионного анализа в R.

### 1 Понятие о регрессионном анализе

Если корреляционный анализ позволяет определить, зависимы ли переменные, и вычислить силу этой зависимости, то для того чтобы определить тип зависимости и вычислить ее параметры, используется регрессионный анализ. Наиболее известна двумерная линейная регрессионная модель:

$$m = b_0 + b_1 \times x,$$

где  $m$  – модельное (predicted) значение зависимой переменной  $y$ ,  
 $x$  – независимая переменная,  
 $b_0$  и  $b_1$  – параметры модели (коэффициенты).

Такая модель позволяет оценить среднее значение переменной  $y$  при известном значении  $x$ . Разность между истинным значением и модельным называется ошибкой («error») или остатком («residual»):

$$E = y - m$$

В идеальном варианте остатки имеют нормальное распределение с нулевым средним и неизвестным, но постоянным разбросом  $\sigma^2$ , который не зависит от значений  $x$  и  $y$ . В этом случае говорят о гомогенности остатков. Если же разброс зависит еще от каких-либо параметров, то остатки считаются гетерогенными. Кроме того, если обнаружилось, что средние значения остатков зависят от  $x$ , то это значит, что между  $y$  и  $x$  имеется нелинейная зависимость.

Для того чтобы узнать значения параметров  $b_0$  и  $b_1$ , применяют метод наименьших квадратов. Затем вычисляют среднеквадратичное отклонение  $R^2$ :

$$R^2 = 1 - \sigma_m^2 / \sigma_y^2,$$

где  $\sigma_y^2$  – разброс переменной  $y$ .

Для проверки гипотезы о том, что модель значительно отличается от нуля, используется так называемая F-статистика (статистика Фишера). Как обычно, если  $p$ -значение меньше уровня значимости (обычно 0,05), то модель считается значимой.

## 2 Выполнение линейного регрессионного анализа в R

### 2.1 Проведение регрессионного анализа в RStudio (при помощи командной строки)

Для регрессионного анализа воспользуемся в учебных целях данными о весе при рождении  $w1$  (в кг) и суточном привесе  $w2$  (в г) десяти бычков (таблица 12).

Таблица 12 – Вес бычков и их суточный прирост

$w1$	38,5	46,0	43,0	43,0	40,5	44,0	38,0	35,0	40,5	54,0
$w2$	694	901	736	1005	841	743	896	863	855	830

Необходимо определить коэффициент регрессии, чтобы выяснить, как влияет исходный вес телят на их будущий прирост; построить график и уравнение линейной регрессии.

**Шаг 1.** Предварительно необходимо составить 2 вектора:  $w1$  и  $w2$ , соответствующие весу бычков при рождении и суточному привесу соответственно.

```
> w1 <- c(38.5, 46, 43, 43, 40.5, 44, 38, 35, 40.5, 54)
> w2 <- c(694, 901, 736, 1005, 841, 743, 896, 863, 855, 830)
```

**Шаг 2.** Далее нужно сформировать датафрейм под названием *bulls* из вновь созданных векторов.

```
> bulls <- data.frame(ves_kg=w1, prives_g=w2)
```

**Шаг 3.** Проверьте получившийся датафрейм на предмет ошибок и в случае необходимости – исправьте их.

```
> bulls
  ves_kg prives_g
1  38.5      694
2  46.0      901
3  43.0      736
4  43.0     1005
5  40.5      841
6  44.0      743
7  38.0      896
8  35.0      863
9  40.5      855
10 54.0      830
```

**Шаг 4.** После создания датафрейма необходимо вычислить параметры уравнения регрессии.

```
> lm.bulls <- lm(formula = prives_g ~ ves_kg, data =
bulls)
> lm.bulls$coefficients
(Intercept)    ves_kg
818.9614568    0.4127466
```

Таким образом, мы получили значения коэффициентов  $b_0 = 819$  и  $b_1 = 0,413$ . Здесь функция  $lm()$  (*line model*) – это линейная модель взаимоотношения веса и привеса бычков.

**Шаг 5.** Далее необходимо провести вывод модельных значений.

```
> b0 <- lm.bulls$coefficient[1]
> b1 <- lm.bulls$coefficient[2]
> x1 <- min(bulls$ves_kg)
> x2 <- max(bulls$ves_kg)
> x <- seq(from = x1, to = x2, length.out = 100)
> y <- b0 + b1*x
```

Здесь нужно обратить внимание на функцию  $seq()$ , которая задает произвольное множество данных в заданном диапазоне (в нашем случае – всего сто значений от минимального ( $x1$ ) до максимального ( $x2$ ) значения переменной веса телят).

**Шаг 6.** Для визуализации результата следует построить коррелограмму полученной зависимости (рисунок 169).

```
> plot(bulls$ves_kg, bulls$prives_g, main = "y = 819 +
0.413 * ves", xlab = "Вес (кг)", ylab = "привес (г)")
```

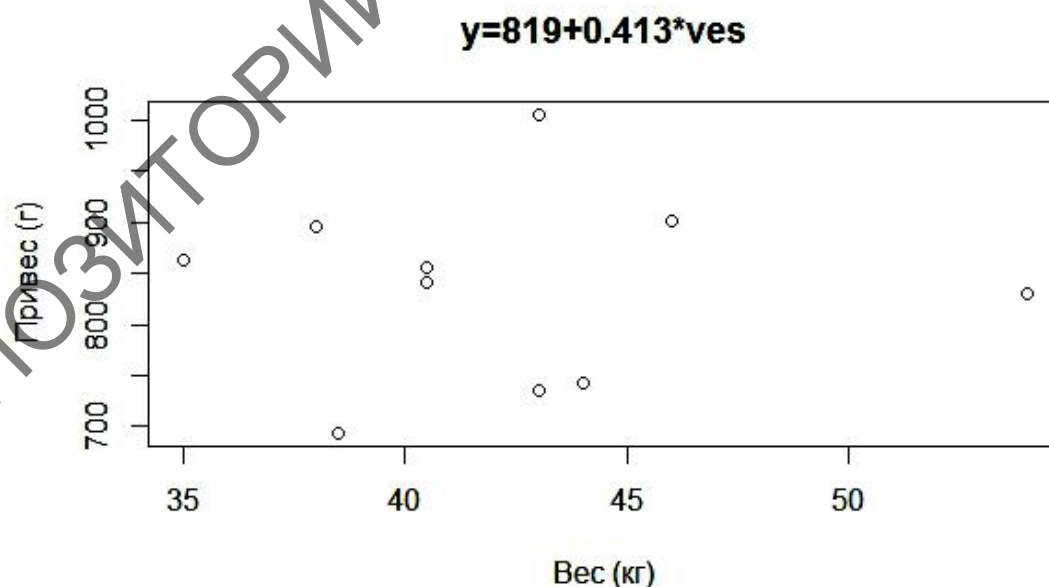


Рисунок 169 – Коррелограмма соотношения веса и суточного привеса бычков

**Шаг 7.** На коррелограмму добавить координатную сетку, используя функцию `grid()` (рисунок 170).

```
> grid()
```

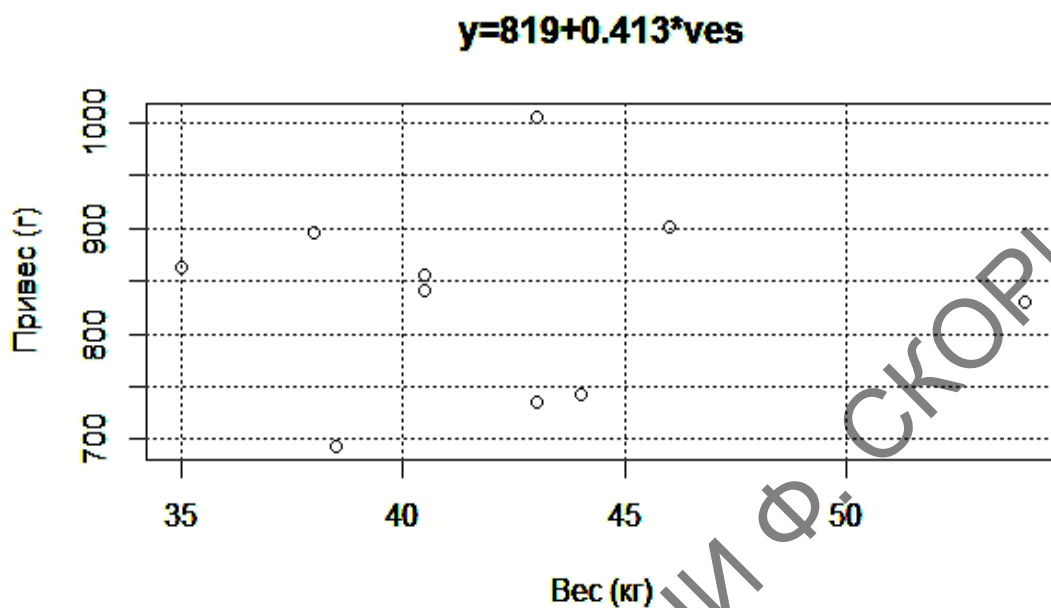


Рисунок 170 – Коррелограмма с добавленной на неё координатной сеткой

**Шаг 8.** Добавляем линию регрессии красного цвета (рисунок 171).

```
> lines(x, y, col="red")
```

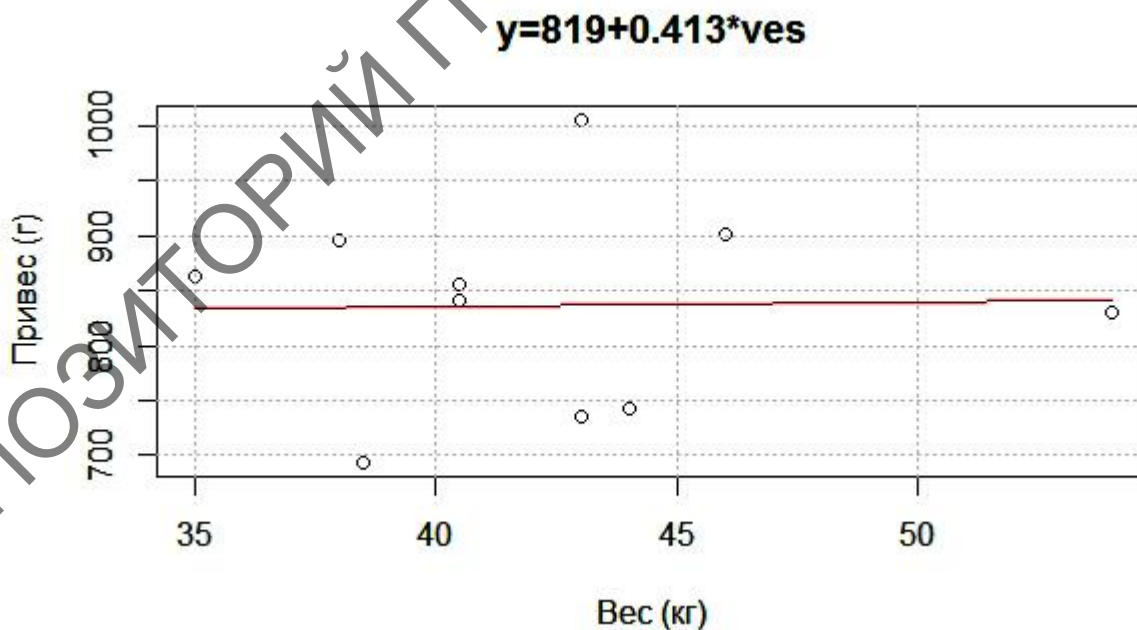


Рисунок 171 – Коррелограмма с добавленной на неё координатной сеткой и линией регрессии

**Шаг 9.** Для целостного представления о регрессии необходимо вывести на экран полные данные по зависимости, используя функцию `summary()`.

```
> summary(lm.bulls)
```

```
call:
```

```
lm(formula = prives_g ~ ves_kg, data = bulls)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max
-140.85  -73.40   12.32   53.41  168.29
```

```
Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  818.9615    264.2951   3.099   0.0147 *
ves_kg        0.4127     6.2126   0.066   0.9487
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 97.66 on 8 degrees of freedom
Multiple R-squared:  0.0005514, Adjusted R-squared:  -0.1244
F-statistic: 0.004414 on 1 and 8 DF, p-value: 0.9487
```

Кратко прокомментируем полученные итоги:

1) В результате расчётов получена линейная регрессионная модель вида:

$$\text{Привес(мод)} = 819 + 0,413 * \text{Вес},$$

то есть увеличение веса на 5 кг соответствует увеличению привеса примерно на 10 г.

2) Наибольшее положительное отклонение истинного значения отклика от модельного составляет 168,29 г, наибольшее отрицательное – минус 140,85 г.

3) Почти половина остатков находится в пределах от первого квартиля (1Q = -73,4 г) до третьего (3Q = 53,41 кг).

4) Один из коэффициентов ( $b_0$ ) значим на уровне  $p\text{-value} < 0,05$ ; это показывает знак звёздочки «\*» в строке *Coefficients* и само значение  $p\text{-value}$   $\text{Pr}(>|t|)$ : 0,0147 для  $b_0$  («Intercept») и коэффициент  $b_1$  статистически не значим даже на уровне  $p\text{-value} < 0,05$  – 0,9487 («ves\_kg»).

5) Среднеквадратичное отклонение (*Adjusted R-squared*) для данной модели составляет  $R^2 = -0,1244$ . Его значение, а также значение коэффициента регрессии  $R$  (*Multiple R* = 0,0005514) говорит об отсутствии регрессионной зависимости.



б) Отсутствие значимости среднеквадратичного отклонения подтверждает и крайне низкое значение F-статистики, равное 0,004414, и общий уровень значимости (определяемый по этой статистике)  $p$ -value: 0,9487, что много больше 0,05.

7) Если на основе этого анализа будет составлен отчет, то следует указать еще и значения степеней свободы  $df$ : 1 и 8 (по колонкам и по строкам данных соответственно).

## 2.2 Проведение регрессионного анализа в RCommander (при помощи GUI)

Расчёт линейной корреляции в пакете RCommander также, как и корреляционной связи, не составляет труда. Для учебных целей используем данные о весе и привесе бычков из таблицы 12.

**Шаг 1.** Загружаем активные данные в пакет. Если ранее они были созданы и загружены в RStudio, а эта программа открыта и пакет RCommander также запускался из неё, то их загрузка делается в меню по пути **Данные** → **Активные данные** → **Выбрать активный набор данных...** с указанием названия активных данных (в нашем случае – bulls). Если же вы работаете в RCommander с нуля, то данные нужно создать и загрузить любым из рассмотренных ранее способов (п. 1.2.2.1 темы 1; п. 2.4.2 темы 2).

**Шаг 2.** После загрузки активных данных необходимо перейти в меню по следующему пути **Статистики** → **Подгонка моделей** → **Линейная регрессия...** (рисунок 172).

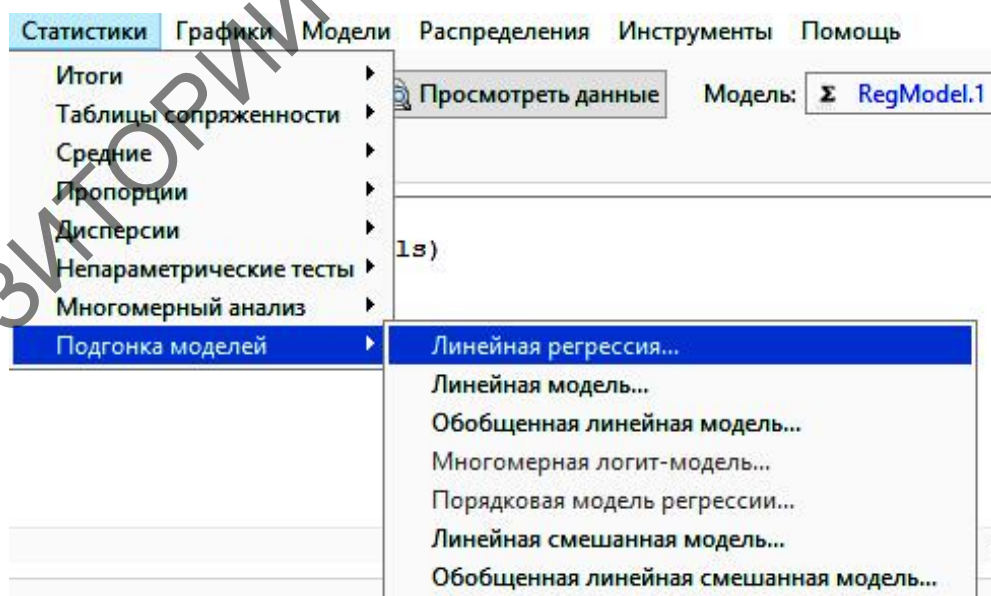


Рисунок 172 – Пункт меню **Линейная регрессия**

**Шаг 3.** Далее в диалоговом окне **Линейная регрессия** необходимо выбрать две переменные для анализа: в левом боксе – зависимую (привес бычков), в правом боксе – независимую (вес бычков). В поле имени модели нужно ввести название модели, которое мы хотим ей придать, например, *bulls* (рисунок 173). После чего нажать кнопку **ОК**. Результат отразится в окне пакета или в среде R.

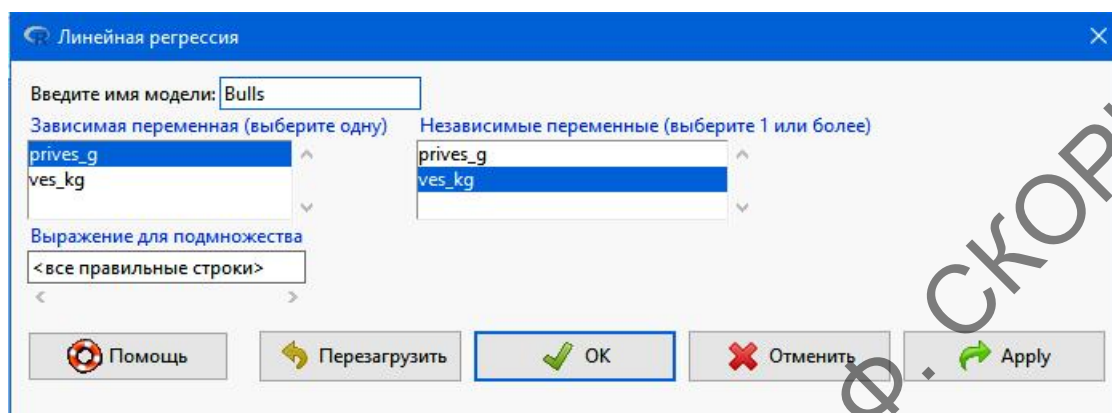


Рисунок 173 – Диалоговое окно **Линейная регрессия**

Результаты будут теми же, что и при обработке через командную строку.

Для графического сопровождения результатов регрессионного анализа в пакете RCommander можно воспользоваться диаграммой рассеяния (коррелограммой). Принцип её создания в этом пакете рассматривался ранее (см. п. 1.1.3 темы 3).

## Вопросы для самоконтроля

- 1 Что такое регрессионный анализ и каково его ключевое отличие от корреляционного анализа?
- 2 Охарактеризуйте функцию прямолинейного регрессионного анализа, опишите значения её аргументов.
- 3 Опишите способ построения линейной модели при регрессионном анализе при помощи командной строки. Какие функции используются при этом?
- 4 Расскажите, каким образом можно провести регрессионный анализ при помощи графического интерфейса. Какой способ реализации регрессионного анализа быстрее, нагляднее?

## Задания для самоконтроля

1) У 20 взрослых мужчин были измерены высота (длина тела)  $x$  (в см) и вес  $y$  (в кг):

$x$	165	176	175	168	167	172	175	180	179	173
$y$	56	75	70	61	61	63	72	80	76	68
$x$	166	178	169	169	170	176	180	169	177	176
$y$	58	76	60	64	63	71	78	63	75	71

Определите коэффициент регрессии, постройте график и уравнение линейной регрессии. Сделайте вывод о возможном соответствии признаков.

2) Предполагается, что между количеством настриженной шерсти ( $y$ ) и живым весом овец ( $x$ ) имеется зависимость. Для 10 овец были получены следующие данные (в кг):

$x$	50	55	60	50	65	60	50	55	50	65
$y$	4,0	4,2	4,1	4,2	4,5	4,3	4,1	4,4	4,0	4,2

Определите коэффициент регрессии, постройте график и уравнение линейной регрессии. Сделайте вывод о возможном соответствии признаков.

3) Для 10 петушков леггорнов 15-дневного возраста были получены следующие данные о весе их тела  $x$  (в г) и весе гребня  $y$  (в мг):

$x$	83	72	69	90	90	95	95	91	75	70
$y$	56	42	18	84	56	107	90	68	31	48

Определите коэффициент регрессии, постройте график и уравнение линейной регрессии. Сделайте вывод о возможном соответствии признаков.

4) Для установления связи между содержанием фосфора в почве ( $x$ ) и содержанием фосфора в злаковых растениях ( $y$ ) было проведено 9 анализов со следующими результатами:

$x$	1	4	5	9	13	11	23	23	28
$y$	64	71	54	81	93	76	77	95	109

Определите коэффициент регрессии, постройте график и уравнение линейной регрессии. Сделайте вывод о возможном соответствии признаков.

5) Было учтено среднесуточное количество перевариваемых веществ корма  $x$  (в кг), съеденного коровой за 12 месяцев лактации:

Месяц	1	2	3	4	5	6	7	8	9	10	11	12
$x$	14,3	12,8	13,2	13,6	13,4	13,2	12,9	12,8	12,5	12,2	11,9	11,5

Определите коэффициент регрессии, постройте график и уравнение линейной регрессии. Сделайте вывод о возможном соответствии признаков.

### Литература по теме

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

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

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

4 Field, A. Discovering statistics using R / A. Field, J. Miles, Z. Field. – London : SAGE Publications Ltd, 2012. – 1193 p.

5 Filzmoser, P. Linear and nonlinear methods for regression and classification and applications in R / P. Filzmoser. – Vienna University of Technology. CS. – №3, 2008. – PP. 1–52.

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 7. ДИСПЕРСИОННЫЙ АНАЛИЗ В R

- 1 Понятие о дисперсионном анализе.
- 2 Проведение однофакторного дисперсионного анализа.
- 3 Проведение двухфакторного дисперсионного анализа.

### 1 Понятие о дисперсионном анализе

**Дисперсионный анализ**, или *ANOVA (analysis of variance)* – это статистический метод изучения влияния одного или нескольких факторов на результивный признак. **Результивный признак** ( $Y$ ) – это элементарное качество или свойство объектов, изучаемое как результат влияния факторов: организованных в исследовании ( $X$ ) и всех остальных, неорганизованных в данном исследовании, случайных ( $Z$ ). Результивными признаками, или зависимые переменные, отклики могут быть:

- точно измеряемые особенности объектов (длина, ширина, рост, сила, содержание гемоглобина, артериальное давление и т. д.);
- неточно измеряемые признаки (густота раствора в баллах, умственные способности и т. д.);
- комбинированные признаки (отношение размеров тела, индексы продуктивности, средние из нескольких данных для одного объекта и т. д.);
- качественные признаки (масть, цвет глаз, болезнь, выздоровление, смерть и т. д.).

**Фактор** – это любое влияние, воздействие или состояние, разнообразие которых может, так или иначе, отражаться в разнообразии результивного признака.

Факторами, или независимыми переменными, причинами, могут быть:

- физические влияния (температура, влажность, радиационное излучение);
- химические влияния (питание, стимуляторы, мутагены, алкоголь);
- биологические влияния (здоровье и болезни, биостимуляторы, наследственность, талантливость, идиотизм);
- признаки объектов (возраст, пол, сорт, порода, национальность, ареал обитания, условия жизни);
- отдельные признаки, принимаемые за аргумент при изучении других признаков – функций (длина ног рысака как один из факторов, определяющих его резвость).

При проведении дисперсионного анализа основной целью является исследование значимости различия между средними значениями зависимой количественной переменной по группам фактора. При этом общая дисперсия результативного признака (зависимой переменной) раскладывается на составляющие: дисперсию за счет разбиения на группы (межгрупповая дисперсия – та, что учитывается в исследовании) и дисперсию за счет остальных, случайных факторов, не учитываемых в исследовании (внутригрупповая дисперсия). Одновременно решаются две задачи: оценка доли воздействия каждого фактора (учитываемого и случайных) на результативный признак (зависимую переменную) и выявление того, за счет каких именно групп идет различие средних значений результативного признака.

Существует несколько моделей дисперсионного анализа в зависимости от числа факторов: однофакторные, двух- и многофакторные дисперсионные модели.

Среди методов дисперсионного анализа выделяют параметрические и непараметрические (критерий Крускала – Уоллиса и критерий Джонкхиера – Терпстры для независимых выборок, критерий Фридмана и критерий Пейджа для повторных измерений).

В том случае, если помимо категориальных факторов (предикторов) в модель вводятся количественные независимые переменные, то получаемая модель носит название *ковариационного анализа*.

Дисперсионный анализ применяется в планировании эксперимента, в управлении производством и в ряде областей экономических, социологических, медико-биологических исследований.

## **2 Проведение однофакторного дисперсионного анализа**

В качестве учебных данных воспользуемся результатами проведенных исследований численности жесткокрылых в окрестностях нефтескважин № 127, № 174 нефтяного месторождения и в том числе на контрольном участке по результатам учётов в 20 почвенных ловушках на каждом из участков (таблица 13).

Проведя однофакторный дисперсионный анализ, необходимо выяснить, влияет ли место обитания (окрестности нефтескважин) на численность обитающих там жесткокрылых.

Таблица 13 – Численность жесткокрылых в окрестностях нефтескважин

Номер ловушки	Номер скважины		Контроль
	127	174	
1	38	240	114
2	13	205	35
3	97	139	98
4	68	116	55
5	16	27	200
6	67	39	101
7	39	79	140
8	75	109	58
9	38	60	39
10	14	150	56
11	19	431	37
12	36	328	24
13	50	115	23
14	135	88	13
15	44	66	1
16	48	27	15
17	31	81	142
18	52	103	121
19	86	59	82
20	16	24	191

Перед проведением анализа нужно правильно подготовить таблицу с данными. Для подготовки таблицы нужно определить 2 переменные, одна из которых будет предиктором, т. е. влияющим фактором. Эту переменную назовем «*location*» и она под собой будет подразумевать место сборов жесткокрылых (номера скважин и контроль). Вторая переменная (назовем ее «*number*») будет подразумевать количество особей, обнаруженных жесткокрылых в том или ином месте сбора. Таким образом, нам необходимо создать вектор переменной *number* со всеми числовыми данными, а затем указать программе, что вектор с данными необходимо разбить на 3 условные части по 20 значений в каждой из частей, назначить вектор переменной *location* с названием точек сбора и количеством значений из каждого подвекторов.

## 2.1 Однофакторный дисперсионный анализ в RStudio (при помощи командной строки)

**Шаг 1.** Примем за переменную *zh* всю совокупность наших данных и создадим датафрейм одновременно с созданием векторов:

```
> zh <- data.frame(number=c(38, 13, 97, 68, 16, 67, 39, 75,
38, 14, 19, 36, 50, 135, 44, 48, 31, 52, 86, 16, 240,
205, 139, 116, 27, 39, 79, 109, 60, 150, 431, 328, 115,
88, 66, 27, 81, 103, 59, 24, 114, 35, 98, 55, 200, 101,
140, 58, 39, 56, 37, 24, 23, 13, 1, 15, 142, 121, 82,
191), location = rep(c("127", "174", "контроль"), c(20,
20, 20)))
```

**Шаг 2.** Проверяем получившийся датафрейм:

```
> zh
  number location
1      38      127
2      13      127
3      97      127
4      68      127
5      16      127
6      67      127
7      39      127
8      75      127
9      38      127
10     14      127
11     19      127
12     36      127
13     50      127
14    135      127
15     44      127
16     48      127
17     31      127
18     52      127
19     86      127
20     16      127
21    240      174
22    205      174
23    139      174
24    116      174
25     27      174
26     39      174
27     79      174
28    109      174
29     60      174
30    150      174
31    431      174
32    328      174
33    115      174
34     88      174
35     66      174
36     27      174
37     81      174
38    103      174
39     59      174
40     24      174
41    114 контроль
```



42	35	контроль
43	98	контроль
44	55	контроль
45	200	контроль
46	101	контроль
47	140	контроль
48	58	контроль
49	39	контроль
50	56	контроль
51	37	контроль
52	24	контроль
53	23	контроль
54	13	контроль
55	1	контроль
56	15	контроль
57	142	контроль
58	121	контроль
59	82	контроль
60	191	контроль

**Шаг 3.** Прикрепляем получившийся датафрейм для возможности использования данных столбцов по отдельности:

```
> attach(zh)
```

Датафрейм готов к работе.

Таблицу с данными можно подготовить и в программе по созданию электронных таблиц, например, в Excel и загрузить её потом в RStudio (п.1.2.2 темы 1).

### 2.1.1 Проведение дисперсионного анализа при соответствии выборки нормальному распределению (параметрический анализ)

**Шаг 1.** Проведем дисперсионный анализ линейной модели влияния численности (переменная *number*) от места положения (переменная *location*), используя функцию `anova()`:

```
> anova(lm(number ~ location))
```

```
Analysis of Variance Table
```

```
Response: number
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
location	2	57741	28870.5	<u>5.5672</u>	<u>0.006187</u> **
Residuals	57	295590	5185.8		

```
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Таким образом, по результатам анализа можно утверждать, что место сбора достоверно влияет на численность жесткокрылых – кри-

терий Фишера (F value) равен 5,57 ( $p = 0,006187$ , что значительно меньше даже 0,01 – две звезды из нижней строчки кодов).

**Шаг 2.** Построим график результатов дисперсионного анализа в виде боксплотов («ящичков и усов»), отложив по оси  $x$  – место сборов, а по оси  $y$  – численность жесткокрылых, окрасив боксы в разные цвета (жёлтый, красный и стальной голубой).

```
> boxplot(number ~ location, col = c("yellow", "red",  
"steelblue"), xlab = "Место сборов", ylab =  
"численность, экз")
```

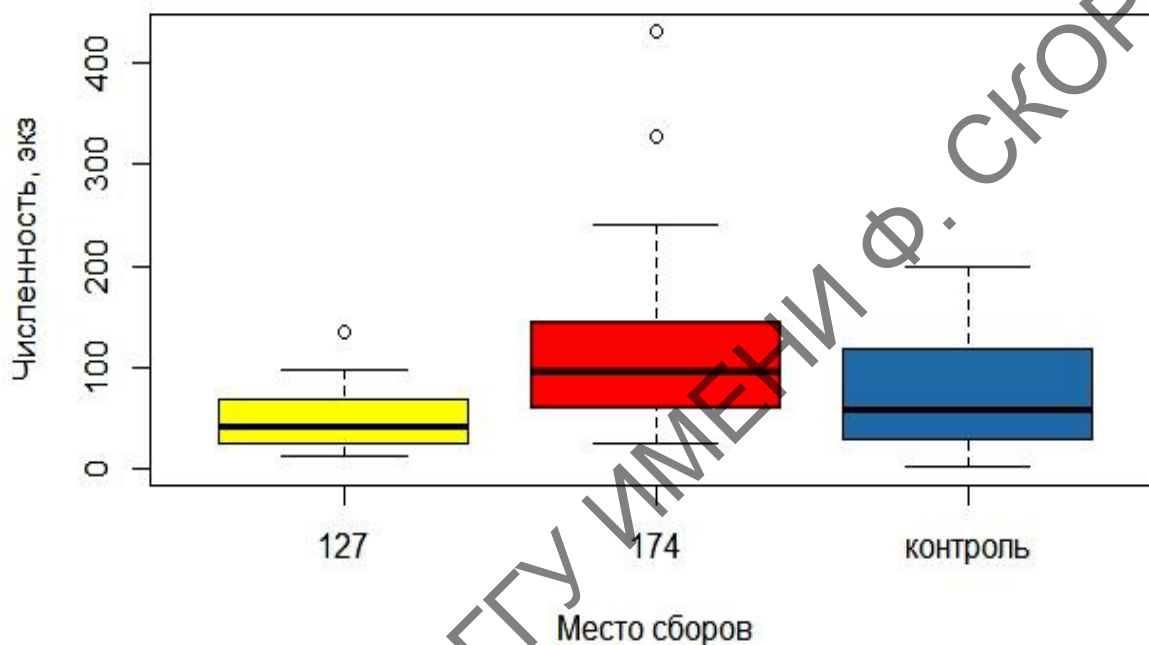


Рисунок 174 – Боксплоты численности жесткокрылых около скважин

Полученный результат сравните с рисунком 174.

### 2.1.2 Проведение дисперсионного анализа при несоответствии выборки нормальному распределению (непараметрический анализ)

В том случае, если есть сомнения в параметричности исходных данных, то лучше всего использовать непараметрический аналог ANOVA, тест Крускала – Уоллиса (Kruskal – Wallis test). Для нашего примера:

```
> kruskal.test(number ~ location)  
kruskal-wallis rank sum test  
data: number by location
```

kruskal-wallis chi-squared = 9.5075, df = 2, p-value = 0.008619

Результат аналогичен «классическому» ANOVA, а p-значение чуть больше, как и положено для непараметрического теста.

### 2.1.3 Проверка распределения на нормальность

Для уверенности в том, какой метод проведения дисперсионного анализа выбрать, необходимо проверить распределения каждой из переменных на нормальность. Это можно сделать отдельно для каждой из переменной по отдельности, но проще и нагляднее объединить все кривые распределения переменных в одном рисунке. Для этого используется пакет **car**. Предварительно необходимо переформатировать таблицу с данными, приведя её в «классический» вид из трёх столбцов по каждой переменной в столбцах. Название таблицы во избежание путаницы лучше сделать отличным от названия ранее созданного датафрейма, например, «*zhuki*».

**Шаг 1.** Включаем пакет **car**.

```
> library(car)
```

**Шаг 2.** Далее выводим результаты на экран (рисунок 175) при помощи функции `scatterplotMatrix()`.

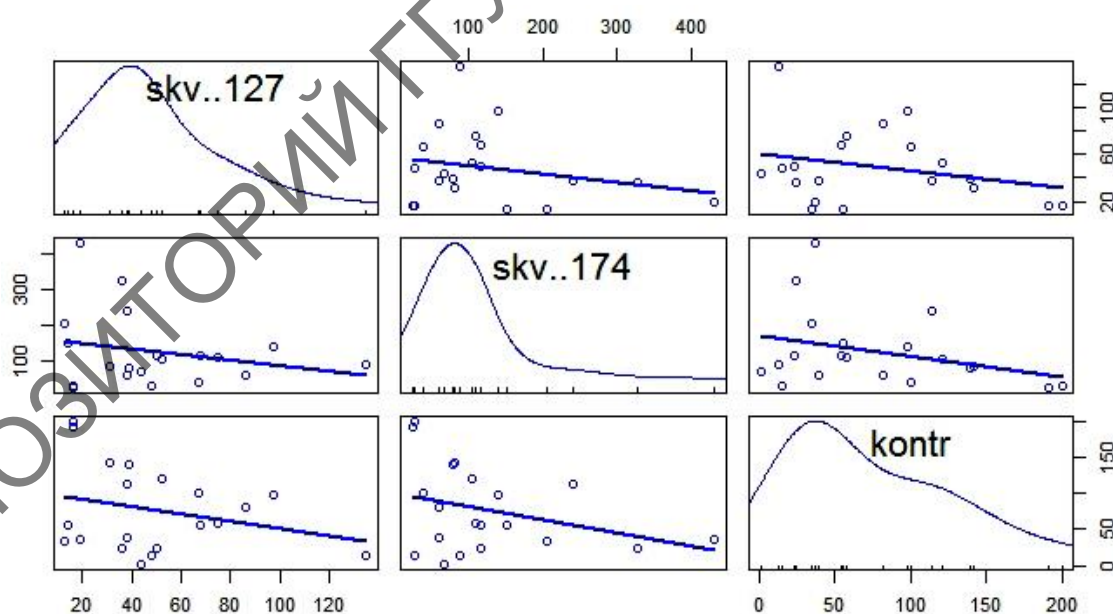


Рисунок 175 – Кривые распределения переменных по численности жуков в окрестностях скважин

В результате распределение по всем трём переменным далеки от нормального и более предпочтительным будет использование непараметрического теста Крускала – Уоллиса при оценке влияния местообитания на численность жесткокрылых в окрестностях нефтескважин. Результаты теста, проведённого нами ранее, отвергают нулевую гипотезу, следовательно, можно сделать вывод о том, что расположение нефтескважин достоверно влияет на численность жесткокрылых.

## 2.2 Однофакторный дисперсионный анализ в RCommander (при помощи GUI)

Проведение однофакторного дисперсионного анализа в пакете RCommander не сложно. Для учебных целей также воспользуемся данными по численности жесткокрылых в окрестностях нефтескважин из таблицы 13.

**Шаг 1.** Загружаем активные данные в пакет. В том случае, если ранее они были созданы и уже загружены в RStudio, а эта программа открыта и пакет RCommander также запускался из неё, то их загрузка делается в меню по пути **Данные** → **Активные данные** → **Выбрать активный набор данных...** с указанием названия активных данных (в нашем случае – *zh*). Если же вы работаете в RCommander с нуля (загружен из среды R), то данные нужно создать и загрузить любым из рассмотренных ранее способов (п. 1.2.2.1 темы 1; п. 2.4.2 темы 2).

**Шаг 2.** После загрузки активных данных необходимо перейти в меню по следующему пути **Статистики** → **Средние** → **Одномерный дисперсионный анализ...** (рисунок 176).

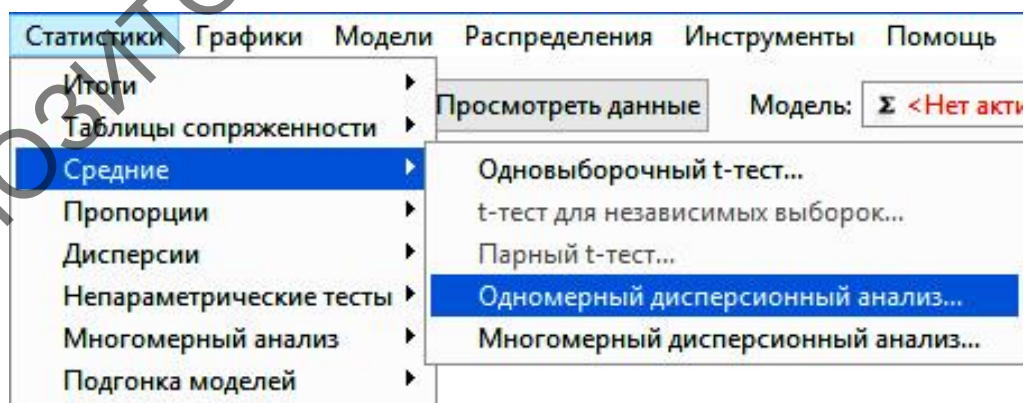


Рисунок 176 – Пункт меню Одномерный дисперсионный анализ

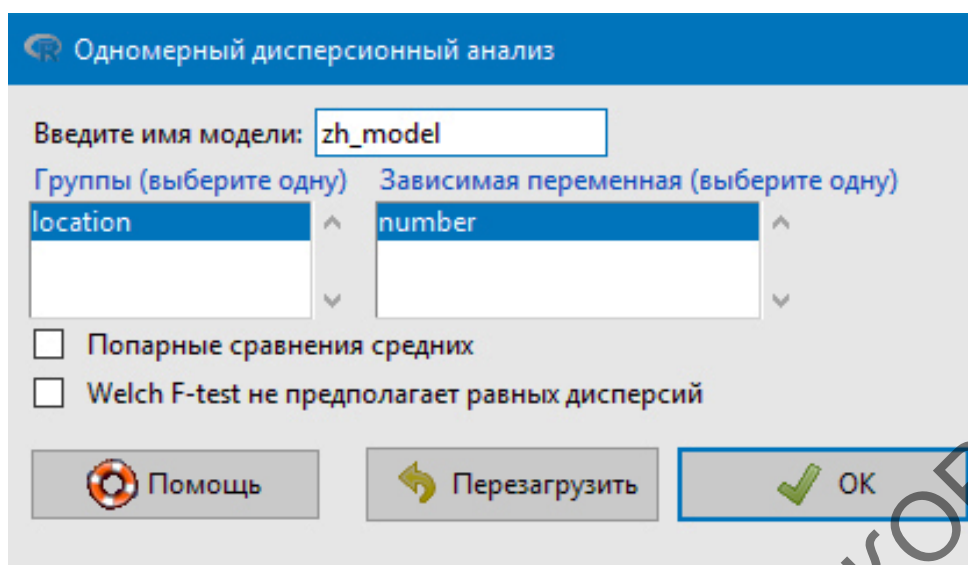


Рисунок 177 – Пункт меню **Одномерный дисперсионный анализ**

**Шаг 3.** Далее в диалоговом окне **Одномерный дисперсионный анализ** (рисунок 177) необходимо для начала ввести название модели, например, «*zh\_model*», а затем в соответствующих боксах выбрать две переменные для анализа: группирующую (т. е. переменную, отражающую фактор) – слева и зависимую (т. е. переменную, отражающую значения фактора) – справа. Далее следует нажать кнопку **ОК**. Результат отразится в окне пакета или в среде R.

Результаты будут теми же, что и при обработке через командную строку.

Для графического сопровождения результатов дисперсионного анализа в пакете RCommander можно воспользоваться построением тех же боксплотов. Принцип её создания в этом пакете рассматривался ранее (см. п. 1.4.3 темы 3, начиная с шестого шага).

Кроме боксплота в пакете RCommander, после проведённого однофакторного анализа и создания в результате него активной модели (то, чьё имя вводили в диалоговом окне **Одномерный дисперсионный анализ**) можно также графически отобразить средние арифметические с ошибкой по каждой из зависимой переменной (в примере со скважинами – по каждой скважине и контролю).

**Шаг 4.** Для создания графика эффекта предиктора необходимо перейти в меню по пути **Модели** → **Графики** → **Графики эффекта предиктора...** (рисунок 178). Полученный график сравните с рисунком 179.

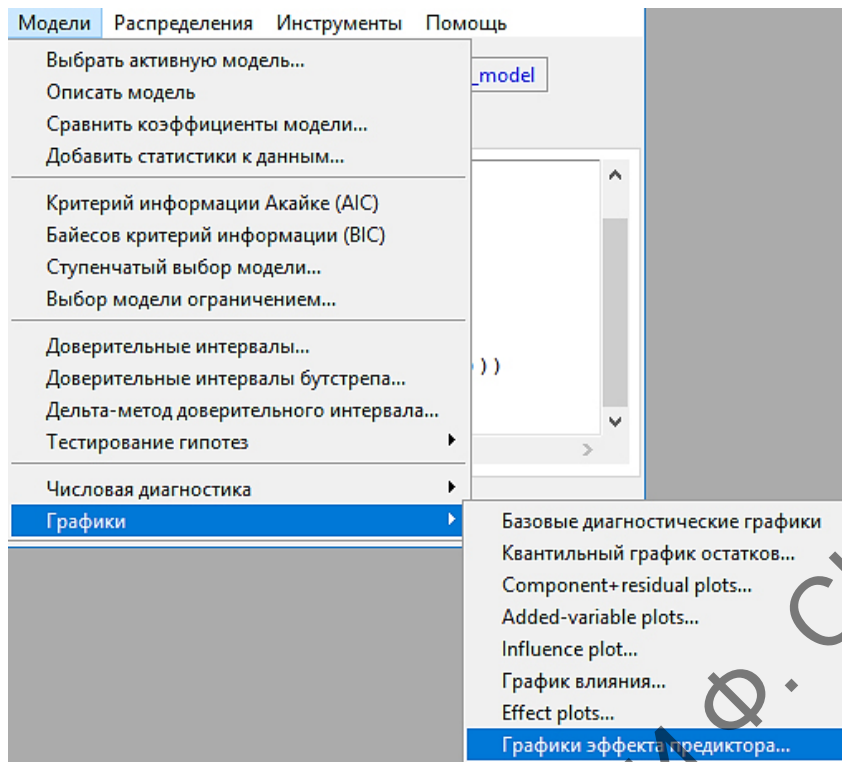


Рисунок 178 – Пункт меню **График эффекта предиктора**

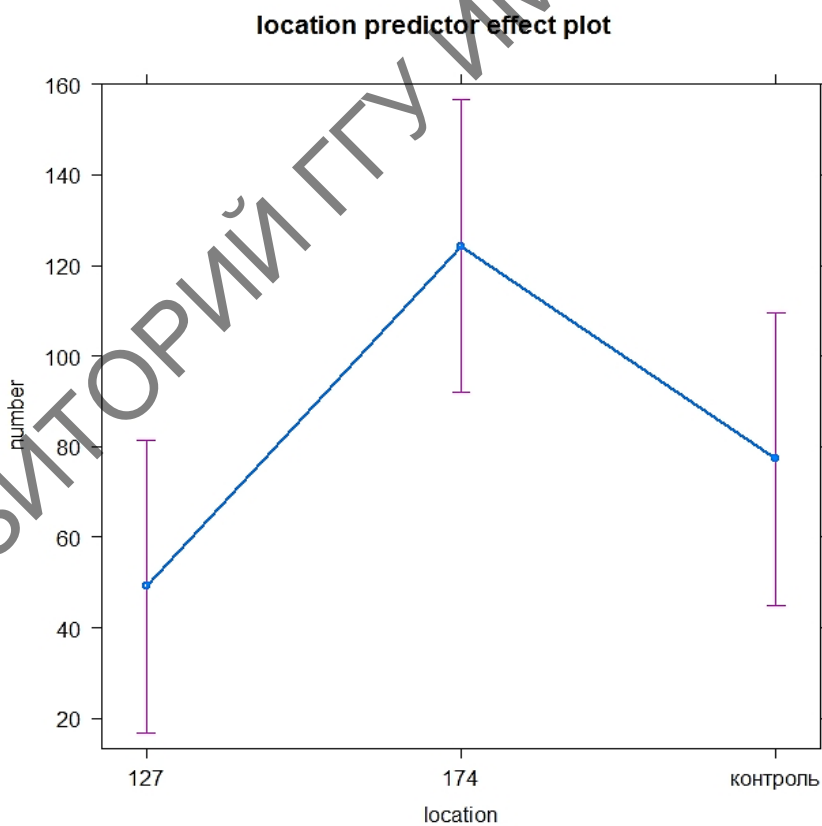


Рисунок 179 – График эффекта предиктора

### 3 Проведение двухфакторного дисперсионного анализа

Двухфакторный дисперсионный анализ предполагает наличие влияния на признак не одного фактора, а двух. В качестве примера рассмотрим уже известную нам по первому разделу урожайность ржи, но учтём тот факт, что повторности – это разные сорта этого растения (таблица 14), в связи чем будем оценивать влияние на урожайность не только вид удобрения, но и сорт.

Таблица 14 – Урожайность ржи после внесения различных смесей удобрений  
в т/га

Сорт	Удобрения			
	Контроль	Смесь 1	Смесь 2	Смесь 3
Рожь 01	24,2	26,6	27	32,2
Рожь 02	31	25,6	33	31
Рожь 03	34,2	31	32,2	30

#### 3.1 Двухфакторный дисперсионный анализ в RStudio (при помощи командной строки)

Перед проведением анализа необходимо подготовить таблицу с данными соответствующим образом. Предполагается оценка влияния на урожайность уже двух факторов – смесей удобрений и сорта ржи. Следовательно, первые две переменные будут факторами. Назовем их, например, «*factor*» и «*sort*». Третья переменная – это урожайность, конкретные значения в цифрах, поэтому назовём её, например, «*numb*» (*number* – количество). Кроме того, следует учесть тот аспект, что на каждое значение фактора удобрения будет приходиться по три значения фактора сорта ржи.

**Шаг 1.** Составляем символьный вектор первого фактора – переменной *factor*.

```
> factor <- c("k", "k", "k", "k", "k", "k", "m1", "m1",  
"m1", "m1", "m1", "m1", "m2", "m2", "m2", "m2", "m2",  
"m2", "m3", "m3", "m3", "m3", "m3", "m3")
```

**Шаг 2.** Далее создаём символьный вектор второго фактора – переменной *sort*.

```
> sort <- c("rozh01", "rozh02", "rozh03", "rozh01",  
"rozh02", "rozh03", "rozh01", "rozh02", "rozh03",
```

```
"rozh01", "rozh02", "rozh03", "rozh01", "rozh02",  
"rozh03", "rozh01", "rozh02", "rozh03", "rozh01",  
"rozh02", "rozh03", "rozh01", "rozh02", "rozh03")
```

**Шаг 3.** После делаем числовой вектор зависимой переменной – переменной *numb*.

```
> numb <- c(24.2, 31, 34.2, 22.2, 31.3, 34.4, 25.1,  
26.4, 29.8, 26.6, 25.6, 31, 27, 33, 32.2, 32.3, 33.1,  
31.9, 31, 32, 30, 32.2, 31, 30)
```

**Шаг 4.** В конце подготовки формируем датафрейм *urozh*.

```
> urozh <- data.frame("Factor" = factor, "Sort" = sort,  
"Urozhajnost" = urozh)
```

**Шаг 5.** Проверяем получившийся датафрейм.

```
> urozh  
  Factor Sort Urozhajnost  
1      k rozh01      24.2  
2      k rozh02      31.0  
3      k rozh03      34.2  
4      k rozh01      22.2  
5      k rozh02      31.3  
6      k rozh03      34.4  
7     m1 rozh01      25.1  
8     m1 rozh02      26.4  
9     m1 rozh03      29.8  
10    m1 rozh01      26.6  
11    m1 rozh02      25.6  
12    m1 rozh03      31.0  
13    m2 rozh01      27.0  
14    m2 rozh02      33.0  
15    m2 rozh03      32.2  
16    m2 rozh01      32.3  
17    m2 rozh02      33.1  
18    m2 rozh03      31.9  
19    m3 rozh01      31.0  
20    m3 rozh02      32.0  
21    m3 rozh03      30.0  
22    m3 rozh01      32.2  
23    m3 rozh02      31.0  
24    m3 rozh03      30.0
```

**Шаг 6.** После создания таблицы следует ввести промежуточную переменную, например, «*rozh\_ur*», которой будет присвоен результат расчёта дисперсионного анализа. Сам анализ проводится при помощи функции `aov()` (*aov* – **a**nalysis **o**f **v**ariance).

```
> rozh_ur <- aov(Urozhajnost ~ Factor + Sort + Fac-  
tor:Sort, data = urozh)
```



Здесь необходимо обратить внимание на аргументы функции. В данном примере мы указываем, что хотим оценить влияние на зависимую переменную `urozhajnost` по отдельности первого фактора (переменная `Factor`) и второго фактора (переменная `Sort`), которую мы добавляем к первой знаком «+» и затем ещё добавляем изучение взаимного влияния двух факторов сразу на зависимую переменную, объединяя их знаком «:».

**Шаг 7.** Рассмотрим результаты анализа, используя функцию `summary()`.

```
> summary(rozh_ur)
      Df Sum Sq Mean Sq F value Pr(>F)
Factor   3  62.44   20.81   12.78 0.000480 ***
Sort     2  71.01   35.51   21.80 0.000101 ***
Factor:sort 6 102.03   17.00   10.44 0.000360 ***
Residuals 12  19.55    1.63
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Прокомментируем полученные результаты. Выявлено, что оба фактора – и смеси удобрений, и сорт – достоверно влияют на урожайность ржи – критерий Фишера составил 12,78 и 21,80 соответственно при уровне значимости 0,000480 для смеси удобрений и 0,000101 для сорта. Это также выделено тремя звёздами справа от этих цифр, что подтверждает значение  $p < 0,001$ .

Совокупность факторов также достоверно влияет на урожайность (третья строка таблицы результатов) – критерий Фишера 10,44 при  $p = 0,000360$ .

### 3.2 Двухфакторный дисперсионный анализ в RCommander (при помощи командной GUI)

Проведение двухфакторного дисперсионного анализа в пакете RCommander подобно однофакторному. Для учебных целей также воспользуемся данными по урожайности ржи из таблицы 14.

**Шаг 1.** Загружаем активные данные в пакет. В том случае, если ранее они были созданы и уже загружены в RStudio, а эта программа открыта и пакет RCommander также запускался из неё, то их загрузка делается в меню по пути **Данные** → **Активные данные** → **Выбрать активный набор данных...** с указанием названия активных данных (в нашем случае – `urozh`). Если же вы работаете в RCommander с нуля (загружен из среды R), то данные нужно со-

здать и загрузить любым из рассмотренных ранее способов (п. 1.2.2.1 темы 1; п. 2.4.2 темы 2).

**Шаг 2.** После загрузки активных данных необходимо перейти в меню по следующему пути **Статистики** → **Средние** → **Многомерный дисперсионный анализ...** (рисунок 180).

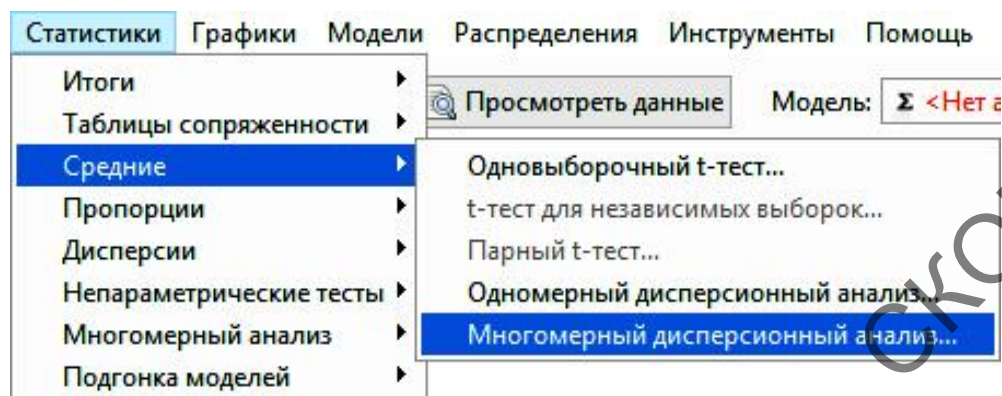


Рисунок 180 – Пункт меню **Многомерный дисперсионный анализ**

**Шаг 3.** Далее, в диалоговом окне **Многомерный дисперсионный анализ** (рисунок 181) необходимо для начала ввести название модели, например, «urozh», а затем в соответствующих боксах выбрать факторы, влияние которых исследуется – слева, и зависимую переменную – справа. Далее следует нажать кнопку **ОК**.

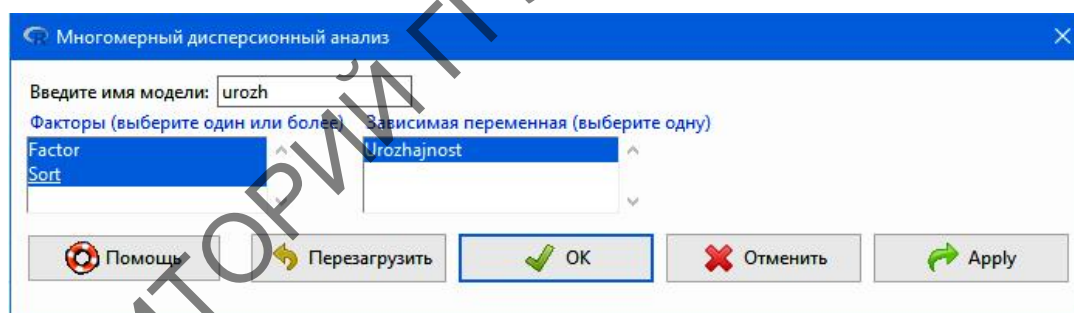


Рисунок 181 – Диалоговое окно **Многомерный дисперсионный анализ**

Результат отразится в окне пакета или в среде R. Результаты будут теми же, что и при обработке через командную строку.

## Вопросы для самоконтроля

- 1 Чем по своей сути является дисперсионный анализ?
- 2 Что может являться фактором в дисперсионном анализе?

3 Какие явления могут выступать в качестве результативного признака?

4 Какие виды дисперсионного анализа существуют?

5 Какая функция используется в языке программирования R для проведения параметрического однофакторного дисперсионного анализа?

6 Какие функции используются в языке программирования R для проведения рангового однофакторного дисперсионного анализа?

7 Опишите синтаксис функции в языке программирования R для проведения двухфакторного анализа.

8 Охарактеризуйте проведение дисперсионного анализа при помощи графического интерфейса.

### Задания для самоконтроля

1) Были получены данные по внесению смеси удобрений ( $a$ ) и их влиянию на прирост ели в питомнике ( $b$ ), в см.

$a$	12,5	12,1	12,1	15,4	2,0	9,5	12,1	11,0	5,8	16,0	12,1
$b$	21,0	5,0	24,1	14,8	29,1	20,1	11,2	19,5	20,5	22,9	17,7
$a$	27,8	19,8	18,1	28,2	43,7	19,1	28,0	20,1	15,0	20,7	12,6
$b$	1,6	0,9	1,1	0,7	1,6	0,7	1,0	1,1	1,1	0,8	0,9
$a$	16,6	15,5	11,5	9,3	14,4	12,2	13,2	6,2	8,1	31,3	10,8
$b$	13,8	9,3	20,4	16,7	18,3	29,1	13,1	19,5	20,0	1,1	0,7

Выясните параметрическим и непараметрическим способами проведения дисперсионного анализа, влияет ли фактор внесения данной смеси удобрений на прирост ели. Постройте график «ящички и усы» для наглядной демонстрации модели.

2) Были получены данные по внесению смеси кормов ( $a$ ) и их влиянию на привес телят ( $b$ ), в кг.

$a$	24,4	15,8	19,5	16,5	16,5	15,0	21,4	17,5	17,2	11,5	11,5	12,0	15,3	30,2	33,3	15,4
$b$	21,5	23,2	16,8	20,5	8,2	19,4	28,6	21,3	19,8	12,7	23,5	20,2	19,1	29,8	25,9	22,5
$a$	17,3	22,8	11,0	17,8	23,5	11,9	15,0	24,6	19,9	20,8	22,8	29,2	25,4	19,8	25,6	10,8
$b$	33,0	14,2	18,3	27,2	5,6	22,6	31,4	26,4	33,9	26,6	14,1	20,7	25,8	21,7	29,4	27,0

Выясните как параметрическим и непараметрическим способами проведения дисперсионного анализа, влияет ли фактор внесения данных кормов на прирост телят. Постройте график «ящички и усы» для наглядной демонстрации модели.

3) 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):

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

Определите, влияет ли внесение данных марок удобрений на урожайность ячменя. Постройте графики «ящички и усы». Дайте обоснованный ответ.

4) На опытном агрокомбинате было проведено комплексное исследование новых видов удобрений для повышения урожайности различных сортов ячменя и кукурузы. Кроме того, учитывалась также глубина заделки семян при посадке:

Сорт	Удобрение 1		Удобрение 2		Удобрение 3		Глубина заделки семян
	Повт. 1	Повт. 2	Повт. 1	Повт. 2	Повт. 1	Повт. 2	
Ячмень 01	9,4	9,7	21,1	21	12,5	12,9	5
Ячмень 01	8,4	8,7	8,2	8,1	12,1	12,5	10
Ячмень 01	4,9	5,2	16,6	16,5	12,1	12,5	15
Ячмень 02	13,4	13,7	10,5	10,4	15,4	15,8	5
Ячмень 02	11,1	11,4	18,4	18,3	2	2,4	10
Ячмень 02	3,6	3,9	11,3	11,2	9,5	9,9	15
Ячмень 03	15,1	15,4	18,3	18,2	12,1	12,5	5
Ячмень 03	13,2	13,5	11,2	11,1	11	11,4	10
Ячмень 03	12	12,3	19,4	19,3	5,8	6,2	15
Кукуруза 01	11,6	11,9	10	9,9	16	16,4	5
Кукуруза 01	18,2	18,5	11,2	11,1	12,1	12,5	10
Кукуруза 01	12,4	12,7	11,1	11	16,6	17	15
Кукуруза 02	15,2	15,5	6,3	6,2	15,5	15,9	5
Кукуруза 02	9,8	10,1	16,3	16,2	11,5	11,9	10
Кукуруза 02	8	8,3	18,8	18,7	9,3	9,7	15
Кукуруза 03	12	12,3	10,1	10	14,4	14,8	5
Кукуруза 03	13,3	13,6	17,6	17,5	12,2	12,6	10
Кукуруза 03	12	12,3	6,6	6,5	13,2	13,6	15

Проведите многофакторный дисперсионный анализ и определите степень влияния каждого фактора в отдельности, а также их совместное влияние на урожайность культур и сортов культур. Сделайте обоснованный вывод.

## Литература по теме

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

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

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

4 Field, A. Discovering statistics using R / A. Field, J. Miles, Z.Field. – London : SAGE Publications Ltd, 2012. – 1193 p.

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.

## ТЕМА 8. КЛАСТЕРНЫЙ АНАЛИЗ В R

- 1 Понятие о кластерном анализе.
- 2 Проведение кластерного анализа методом  $k$ -средних в R.
- 3 Проведение дендрограммного кластерного анализа в R.

### 1 Понятие о кластерном анализе

**Кластерный анализ** – это метод проведения классификации без обучения. В результате применения этих процедур исходная совокупность объектов разделяется на *кластеры* или *группы* (классы) схожих между собой объектов.

**Кластер** – это группа объектов, обладающая свойством плотности (плотность объектов внутри кластера выше, чем вне его), дисперсией, отделимостью от других кластеров, формой (например, кластер может иметь очертания гиперсферы или эллипсоида), размером.

Наиболее часто методы кластерного анализа используются в социологии, маркетинговых исследованиях, экономике, биологии, медицине, археологии.

В целом *методы кластеризации* делятся на *агломеративные* (от слова агломерат – скопление) и *итеративные дивизивные* (от слова division – деление, разделение).

В агломеративных, или объединительных методах происходит последовательное объединение наиболее близких объектов в один кластер. Процесс такого последовательного объединения можно показать на графике в виде дендрограммы, или дерева объединения. Это удобное представление позволяет наглядно представить кластеризацию агломеративными алгоритмами.

Исходными данными для анализа могут быть собственно объекты и их параметры. Данные для анализа могут быть также представлены матрицей расстояний между объектами, в которой на пересечении строки с номером  $i$  и столбца с номером  $j$  записано расстояние между  $i$ -м и  $j$ -м объектом (пример – таблица расстояний между городами в автомобильном атласе).

Если расстояния не даны сразу, то агломеративные алгоритмы начинаются с вычисления расстояний между объектами. Расстояние между объектами – одна из мер сходства. В качестве методов измерения расстояния чаще всего используют евклидову метрику, манхэттенское расстояние и расстояние Чебышёва.

*Евклидово расстояние, или евклидову метрику* используют в том случае, если объект описывается двумя параметрами. Тогда он может быть изображен точкой на плоскости, а расстояние между объектами – это расстояние между точками, вычисленное по теореме Пифагора. Вы просто возводите в квадрат расстояния по каждой координате, суммируете их и из полученной суммы извлекаете квадратный корень (рисунок 182).

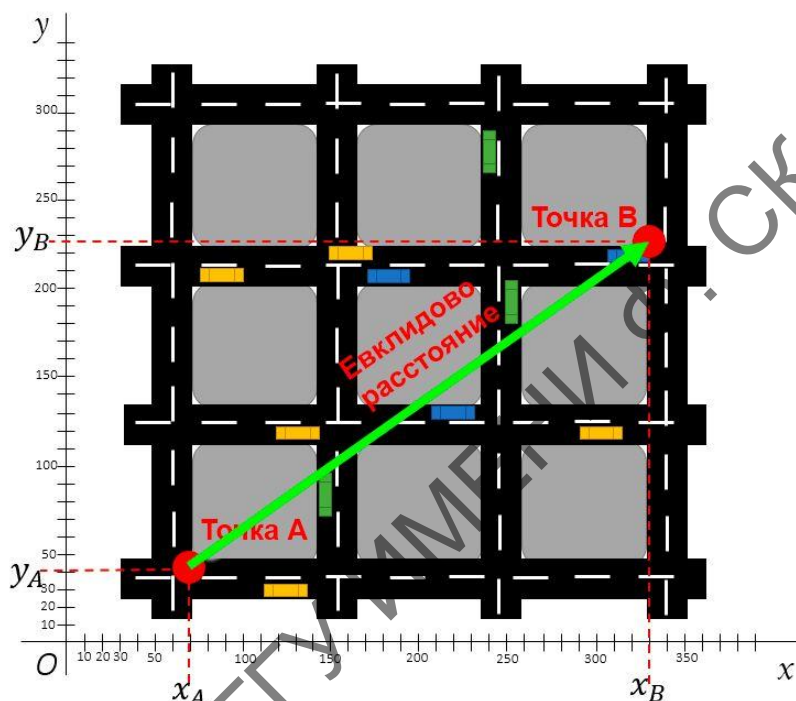


Рисунок 182 – Схематическое изображение евклидова расстояния

Если не возводить в квадрат по координатные расстояния, а просто использовать их абсолютные значения и просуммировать, то получится так называемое *манхэттенское расстояние*, или «расстояние городских кварталов».

В качестве примера можно представить перемещение человека по улицам города, а не движение по ровной местности. В городе существуют определенные правила перемещения и, соответственно, правила вычисления пройденного расстояния (рисунок 183). Перемещаться можно только по улицам (нельзя, например, пересечь квартал или дом по диагонали). Аналогия в декартовой плоскости приводит к перемещениям только по линиям, параллельным осям координат, и, соответственно, к манхэттенскому расстоянию.

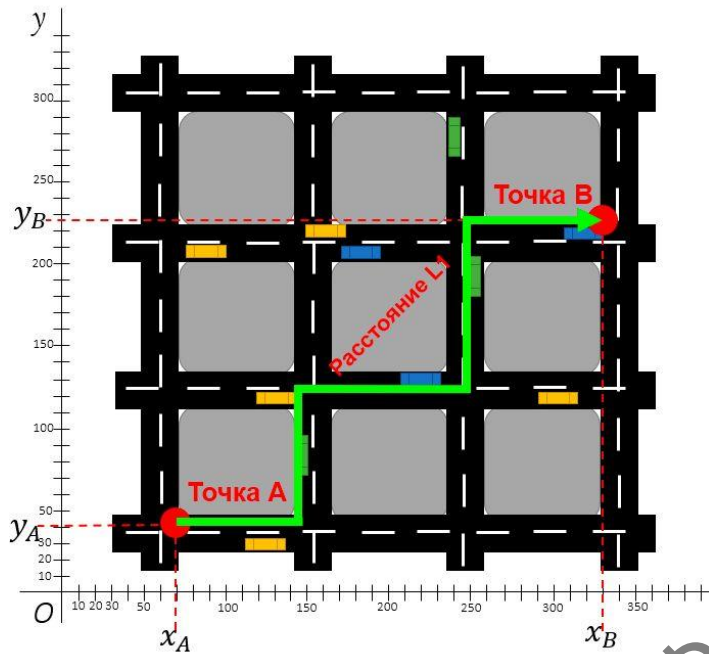


Рисунок 183 – Схематическое изображение манхэттенского расстояния

Разница между манхэттенским расстоянием и *расстоянием Чебышёва* (рисунок 184) в том (используя аналогию движения короля по шахматной доске), что при переходе на одну клетку, расположенную по диагонали в первом случае засчитывается два хода (вперед и в сторону), а во втором случае засчитывается всего один ход.

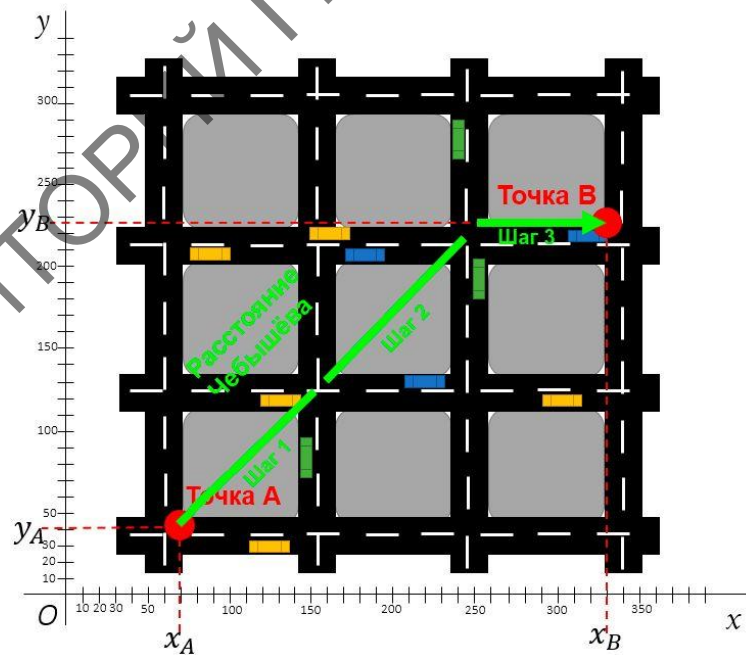


Рисунок 184 – Схематическое изображение расстояния Чебышёва



Эти расстояния отличаются от Евклидова расстояния тем, что у Евклидова движение по диагонали рассчитывается по теореме Пифагора (квадрат гипотенузы прямоугольного треугольника равен сумме квадратов катетов), где гипотенуза и есть искомое расстояние.

В практике чаще реализуются агломеративные методы кластеризации. Можно выбрать следующие правила иерархического объединения кластеров:

- метод одиночной связи;
- метод полной связи;
- невзвешенный метод «средней связи»;
- взвешенный метод «средней связи»;
- взвешенный центроидный метод;
- метод Уорда.

Данные алгоритмы различаются правилами объединения объектов в кластеры.

В *методе одиночной связи* на первом шаге объединяются два объекта, имеющие между собой максимальную меру сходства. На следующем шаге к ним присоединяется объект с максимальной мерой сходства с одним из объектов кластера. Таким образом, процесс продолжается далее. Итак, для включения объекта в кластер требуется максимальное сходство лишь с одним членом кластера. Отсюда и название метода одиночной связи: нужна только одна связь, чтобы присоединить объект к кластеру (связь нового элемента с кластером определяется только по одному из элементов кластера). Недостатком этого метода является образование слишком больших «продолговатых» кластеров.

*Метод полных связей* позволяет устранить указанный недостаток. Здесь мера сходства между объектом – кандидатом на включение в кластер и всеми членами кластера – не может быть меньше некоторого порогового значения.

В *методе средней связи* мера сходства между кандидатом и членами кластера усредняется, например, берется просто среднее арифметическое мер сходства.

Идея *метода Уорда* состоит в том, чтобы проводить объединение, дающее минимальное приращение внутригрупповой суммы квадратов отклонений. Отметим, что метод Уорда приводит к образованию кластеров примерно равных размеров и имеющих форму гиперсфер.

*Итеративный метод группировки k-средних* работает непосредственно с объектами, а не с матрицей сходства.

В методе k-средних объект относится к тому классу, расстояние до которого минимально. Расстояние понимается как евклидово расстояние, то есть объекты рассматриваются как точки евклидова пространства.

## 2 Проведение кластерного анализа методом $k$ -средних в R

### 2.1 Проведение кластерного анализа методом $k$ -средних в RStudio (при помощи командной строки)

Для проведения кластерного анализа методом  $k$ -средних воспользуемся учебными данными по численности 10 видов животных в десяти кварталах лесничества (таблица 15).

Таблица 15 – Численность животных в кварталах лесничества

	kv6	kv8	kv11	kv12	kv15	kv18	kv19	kv21	kv28	kv32
Sp1	5	7	6	12	36	34	28	22	9	16
Sp2	6	8	7	14	42	40	34	28	10	18
Sp3	14	16	15	30	90	88	82	76	18	34
Sp4	12	14	13	26	78	76	70	64	16	30
Sp5	4	6	5	10	30	28	22	16	8	14
Sp6	10	12	11	22	66	64	58	52	14	26
Sp7	23	25	24	48	144	142	136	130	27	52
Sp8	2	4	3	6	18	16	10	4	6	10
Sp9	7	9	8	16	48	46	40	34	11	20
Sp10	3	5	4	8	24	22	16	10	7	12

**Шаг 1.** Первоначально необходимо создать 11 векторов с данными (1 – вектор символьный с названным видом, а остальные 10 – числовые с численностью по кварталам).

```
> species <- c("Sp1", "Sp2", "Sp3", "Sp4", "Sp5", "Sp6",  
"Sp7", "Sp8", "Sp9", "Sp10")  
> kv6 <- c(5, 6, 14, 12, 4, 10, 23, 2, 7, 3)  
> kv8 <- c(7, 8, 16, 14, 6, 12, 25, 4, 9, 5)  
> kv11 <- c(6, 7, 15, 13, 5, 11, 24, 3, 8, 4)  
> kv12 <- c(12, 14, 30, 26, 10, 22, 48, 6, 16, 8)  
> kv15 <- c(36, 42, 90, 78, 30, 66, 144, 18, 48, 24)  
> kv18 <- c(34, 40, 88, 76, 28, 64, 142, 16, 46, 22)  
> kv19 <- c(28, 34, 82, 70, 22, 58, 136, 10, 40, 16)  
> kv21 <- c(22, 28, 76, 64, 16, 52, 130, 4, 34, 10)  
> kv28 <- c(9, 10, 18, 16, 8, 14, 27, 6, 11, 7)  
> kv32 <- c(16, 18, 34, 30, 14, 26, 52, 10, 20, 12)
```

**Шаг 2.** Далее необходимо создать датафрейм *kv* (сокращённо от «кварталы»).

```
> kv <- data.frame("species" = species, "kv6" = kv6,
"kv8" = kv8, "kv11" = kv11, "kv12" = kv12, "kv15" =
kv15, "kv18" = kv18, "kv19" = kv19, "kv21" = kv21,
"kv28" = kv28, "kv32" = kv32
```

**Шаг 3.** Проверяем получившийся датафрейм *kv* на возможные ошибки и при необходимости исправляем их.

```
> kv
  Species kv6 kv8 kv11 kv12 kv15 kv18 kv19 kv21 kv28 kv32
1    Sp1    5  7    6   12   36   34   28   22    9   16
2    Sp2    6  8    7   14   42   40   34   28   10   18
3    Sp3   14 16   15   30   90   88   82   76   18   34
4    Sp4   12 14   13   26   78   76   70   64   16   30
5    Sp5    4  6    5   10   30   28   22   16    8   14
6    Sp6   10 12   11   22   66   64   58   52   14   26
7    Sp7   23 25   24   48  144  142  136  130   27   52
8    Sp8    2  4    3    6   18   16   10    4    6   10
9    Sp9    7  9    8   16   48   46   40   34   11   20
10   Sp10   3  5    4    8   24   22   16   10    7   12
```

В качестве примера кластеризации попробуем объединить в кластеры виды по их численности в кварталах (то есть кластеризацию нужно будет провести по строкам таблицы). Предварительно нужно подготовить таблицу для анализа.

**Шаг 4.** Перед проведением кластерного анализа из названий видов в первом столбце необходимо сделать заголовки строк.

```
> rownames(kv) <- make.names(kv[,1], unique = TRUE)
```

**Шаг 5.** Таблица с данными будет иметь уже другой вид.

```
> kv
  Species kv6 kv8 kv11 kv12 kv15 kv18 kv19 kv21 kv28 kv32
Sp1    Sp1    5  7    6   12   36   34   28   22    9   16
Sp2    Sp2    6  8    7   14   42   40   34   28   10   18
Sp3    Sp3   14 16   15   30   90   88   82   76   18   34
Sp4    Sp4   12 14   13   26   78   76   70   64   16   30
Sp5    Sp5    4  6    5   10   30   28   22   16    8   14
Sp6    Sp6   10 12   11   22   66   64   58   52   14   26
Sp7    Sp7   23 25   24   48  144  142  136  130   27   52
Sp8    Sp8    2  4    3    6   18   16   10    4    6   10
Sp9    Sp9    7  9    8   16   48   46   40   34   11   20
Sp10   Sp10   3  5    4    8   24   22   16   10    7   12
```

**Шаг 6.** Далее следует убрать первый столбец таблицы из обработки, чтобы программа работала с заголовками строк.

```
> kv <- kv[, -1]
```

**Шаг 7.** Проверим получившуюся таблицу с данными.

```
> kv
  kv6 kv8 kv11 kv12 kv15 kv18 kv19 kv21 kv28 kv32
Sp1   5   7   6  12  36  34  28  22   9  16
Sp2   6   8   7  14  42  40  34  28  10  18
Sp3  14  16  15  30  90  88  82  76  18  34
Sp4  12  14  13  26  78  76  70  64  16  30
Sp5   4   6   5  10  30  28  22  16   8  14
Sp6  10  12  11  22  66  64  58  52  14  26
Sp7  23  25  24  48 144 142 136 130 27  52
Sp8   2   4   3   6  18  16  10   4   6  10
Sp9   7   9   8  16  48  46  40  34  11  20
Sp10  3   5   4   8  24  22  16  10   7   1
```

**Шаг 8.** После этого необходимо ввести дополнительную переменную (например, *kl*) для расчёта данных по каждой из строк таблицы.

```
> k1 <- (nrow(kv)-1)*sum(apply(kv,2,var))
```

**Шаг 9.** В связи с тем, что количество возможных кластеров достоверно неизвестно (предположим, что их пять – половина от числа всех видов), рассчитаем центр каждого кластера, используя оператор цикла `for()`.

```
> for (i in 2:5) k1[i] <- sum(kmeans(kv, centers=i)$withinss)
```

Здесь следует обратить внимание на аргумент цикла *i*, который является счётчиком вариантов, и функцию `kmeans()`, определяющую центр каждого из возможных кластеров.

**Шаг 10.** После проведённой подготовки можно определить количество кластеров для объединения. Для этого необходимо построить график распределения кластеров, используя в качестве аргумента переменную *kl*. Получившийся график необходимо сверить с рисунком 185.

```
> plot(1:5, k1, type="b", xlab="число кластеров",
ylab="Сумма квадратов расстояний внутри кластеров")
```

На рисунке следует отметить точку резкого излома графика. Она соответствует числу кластеров (ось абсцисс), равную двум. Следовательно, это и есть число искомых кластеров.

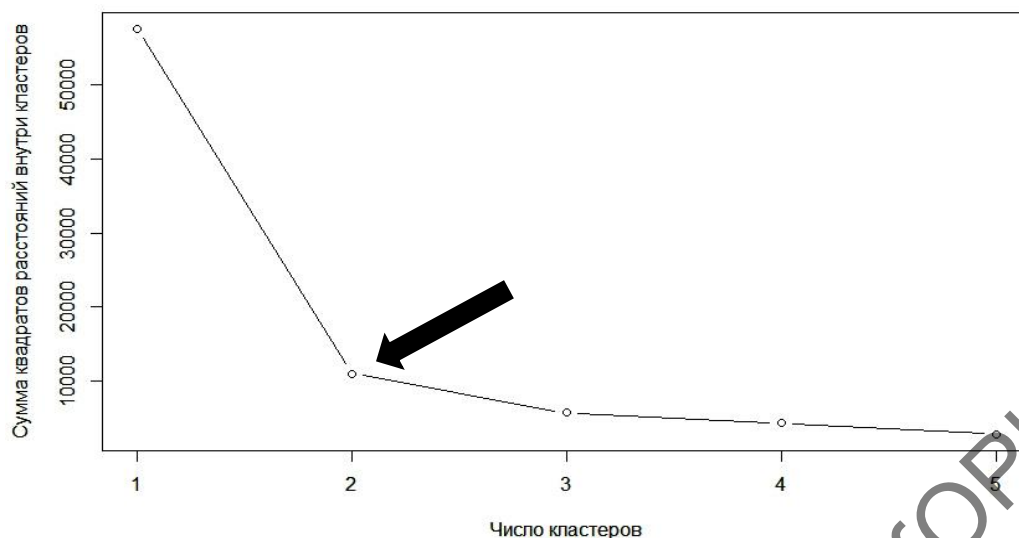


Рисунок 185 – Определение количества кластеров графически

**Шаг 11.** После определения количества кластеров проводим собственно кластерный анализ методом  $k$ -средних с использованием функции `kmeans()`, указывая в качестве аргументов функции название таблицы данных и количество кластеров. Для присвоения значения группы кластеров необходимо использовать промежуточную переменную (например, `kcl` – сокращённо от `k-cluster`).

```
> kcl <- kmeans(kv, 2)
```

**Шаг 12.** Теперь можно посмотреть виды, попавшие в каждый кластер.

```
> aggregate(kv, by=list(kcl$cluster), FUN=mean)
```

**Шаг 13.** Для лучшей визуализации состава кластеров более наглядно будет объединение их в датафрейм с указанием конкретного кластера для каждого вида.

```
> kv_c1 <- data.frame(kv, kcl$cluster)
```

Здесь также вводится дополнительная переменная (в нашем случае `kv_c1`), которой присваивается таблица данных с указанием номера кластера.

**Шаг 14.** Просматриваем получившуюся таблицу.

```
> kv_c1
kv6 kv8 kv11 kv12 kv15 kv18 kv19 kv21 kv28 kv32 kc.cluster
Sp1 5 7 6 12 36 34 28 22 9 16 2
Sp2 6 8 7 14 42 40 34 28 10 18 2
Sp3 14 16 15 30 90 88 82 76 18 34 1
Sp4 12 14 13 26 78 76 70 64 16 30 1
Sp5 4 6 5 10 30 28 22 16 8 14 2
Sp6 10 12 11 22 66 64 58 52 14 26 2
```

Sp7	23	25	24	48	144	142	136	130	27	52	1
Sp8	2	4	3	6	18	16	10	4	6	10	2
Sp9	7	9	8	16	48	46	40	34	11	20	2
Sp10	3	5	4	8	24	22	16	10	7	12	2

Таким образом, судя по полученным результатам к первому кластеру относятся виды под номерами 3, 4 и 7; ко второму – все остальные.

**Шаг 15.** Большое значение имеет также визуализация результатов. Создадим рисунок, показывающий членов каждого из кластеров. Для этого воспользуемся пакетом **cluster**.

```
> library(cluster)
```

**Шаг 16.** После включения пакета **cluster** можно строить диаграмму кластеров при помощи функции **clusplot()**, указав в качестве аргументов название переменной с таблицей данных, а также особенности оформления диаграммы.

```
> clusplot(kv, kcl$cluster, color=TRUE, shade=TRUE,
labels=2, lines=0)
```

Полученную диаграмму сравните с рисунком 186.

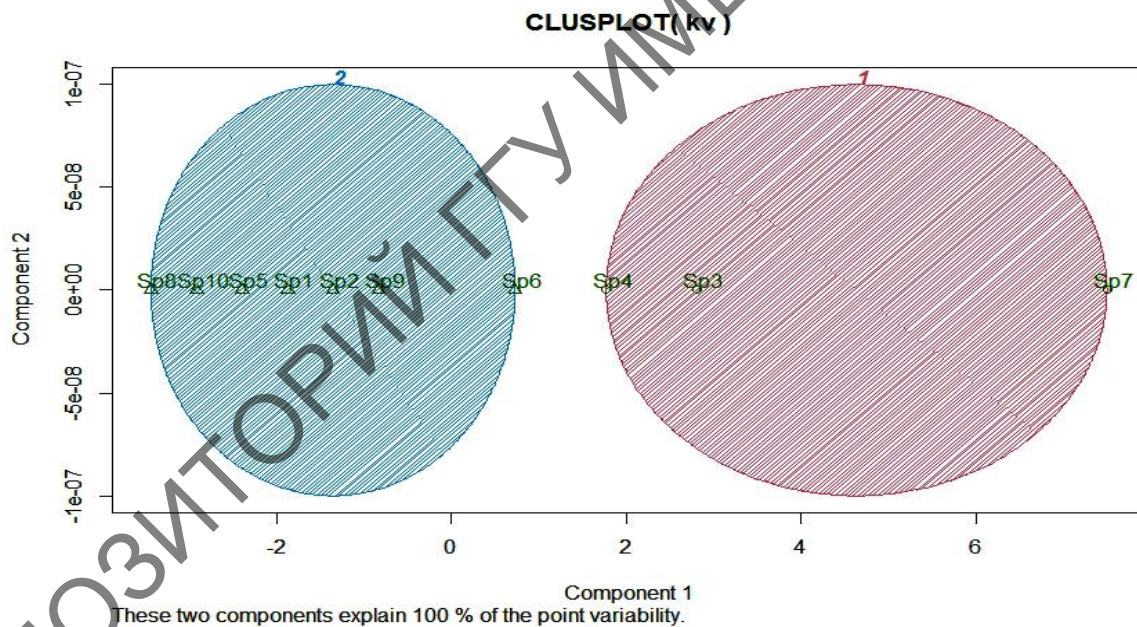


Рисунок 186 – Диаграмма кластеров

**Шаг 17.** Визуализация результатов также позволяет отобразить расстояние членов кластеров от центра кластера. Для этого вычисляем расстояние между объектами в кластерах и формируем шкалу расстояний.

```
> kv_dist <- dist(kv)
> cmd <- cmdscale(kv_dist)
```

**Шаг 18.** Для графика расстояний используется пакет **vegan**.

```
> install.packages("vegan")
> library(vegan)
```

**Шаг 19.** Вводим дополнительную переменную *groups* (группы), которой будут присвоены значения уровней кластеров.

```
> groups <- levels(factor(kc1$cluster))
```

**Шаг 20.** Начинаем постепенно строить график.

```
> plot(cmd, type = "n")
```

**Шаг 21.** Далее выберем цвета для определения кластеров. Например, синий и красный.

```
> cols <- c("blue", "red")
```

**Шаг 22.** Далее постепенно формируем рисунок, используя несколько функций (в том числе и оператор цикла).

```
> for(i in seq_along(groups)){points(cmd[factor(kc1$cluster)
== groups[i], ], col = cols[i], pch = 16)}
> ordispider(cmd, factor(kc1$cluster), label = TRUE)
> ordihull(cmd, factor(kc1$cluster), lty = "dotted")
```

Полученный результат можно сравнить с рисунком 187.

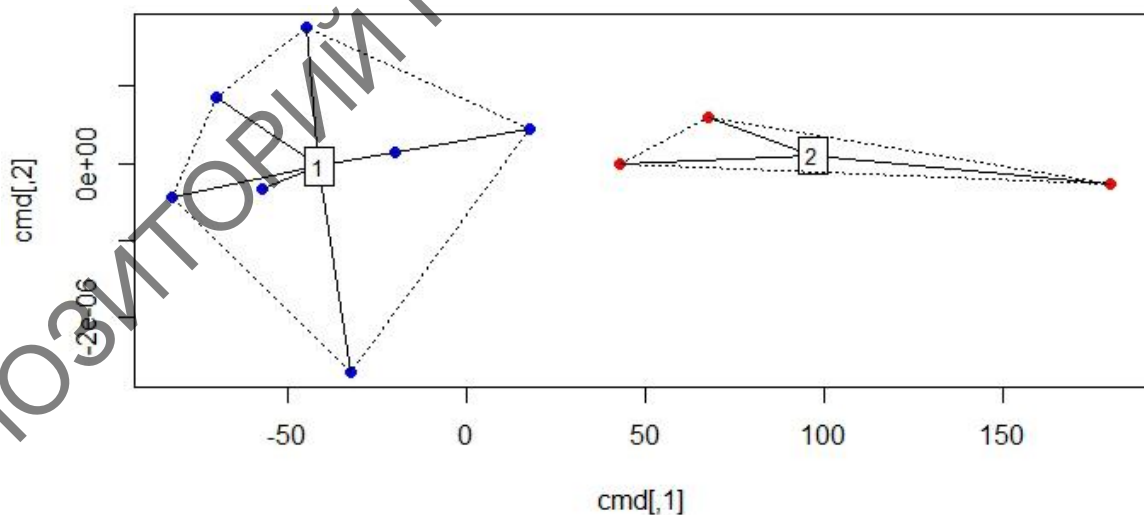


Рисунок 187 – График расстояний от центра каждого кластера к его членам



## 2.2 Проведение кластерного анализа методом $k$ -средних в RCommander (при помощи GUI)

Для проведения кластерного анализа методом  $k$ -средних в RCommander для учебных целей воспользуемся уже созданными ранее в RStudio данными по численности животных в десяти кварталах лесничества (таблица данных  $kv$ ).

**Шаг 1.** Включаем пакет RCommander любым из удобных способов (п. 1.1 темы 1).

**Шаг 2.** Загружаем активные данные в пакет. Используем созданную ранее в RStudio таблицу данных  $kv$ , перейдя в меню по пути **Данные** → **Активные данные** → **Выбрать активный набор данных...** с указанием названия активных данных (в нашем случае –  $kv$ ).

**Шаг 3.** После загрузки активных данных можно перейти непосредственно к проведению кластерного анализа. Для этого необходимо перейти в меню по пути **Статистики** → **Многомерный анализ** → **Кластерный анализ** → **Кластерный анализ К-средних...** (рисунок 188).

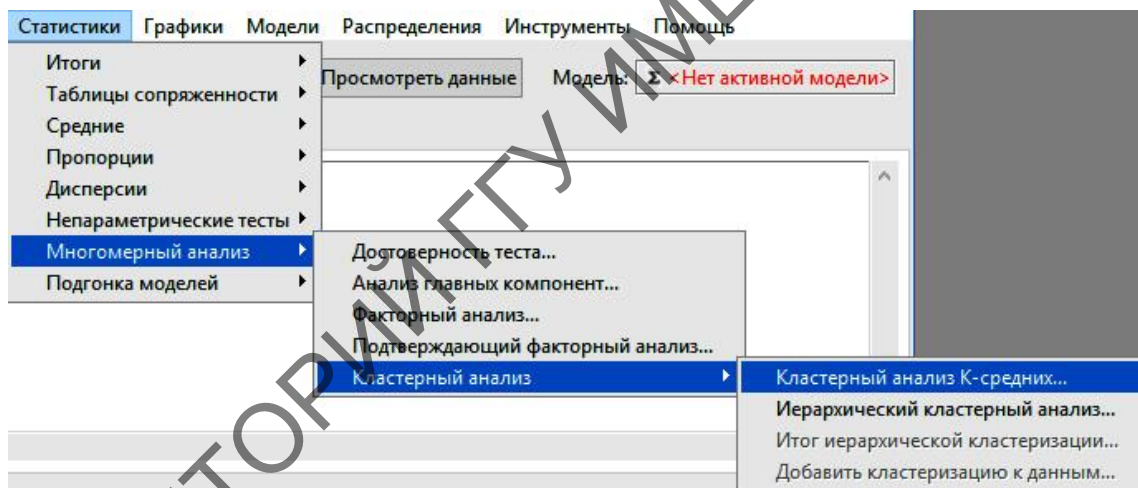


Рисунок 188 – Пункт меню «Кластерный анализ К-средних»

**Шаг 4.** Далее, в диалоговом окне **Кластерный анализ К-средних** в закладке **Данные** нужно выделить все переменные и перейти на закладку **Опции**.

**Шаг 5.** В закладке **Опции** нужно определить количество кластеров (как уже выяснили ранее, их в данном случае 2), а также выставить количество стартовых единиц и количество итераций (проходов). Можно оставить по умолчанию – 10. Можно также присвоить



уникальное имя переменной результатам кластерного анализа (или оставить название, данное по умолчанию). Остальное оставить так, как на рисунке 189.

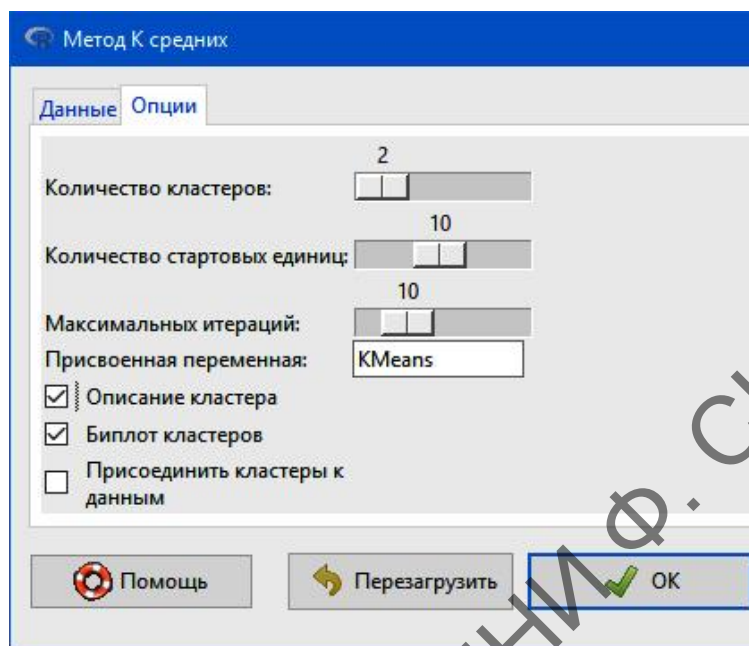


Рисунок 189 – Диалоговое окно **Кластерный анализ К-средних**

В результате в окне среды программирования (или в RStudio) будут отражены строка кода и результаты анализа, подобные результатам, приведённым после шага 12 п. 2.1 данной темы, и будет добавлен рисунок – биplot (двойной график) кластеров (рисунок 190).

```
Rcmdr> .cluster <- kMeans(model.matrix(~-1 + kv6 +
kv8 + kv11 + kv12 + kv15 +
Rcmdr+ kv18 + kv19 + kv21 + kv28 + kv32, kv), centers =
2, iter.max = 10,
Rcmdr+ num.seeds = 10)
Rcmdr> .cluster$size # Cluster Sizes
[1] 3 7
Rcmdr> .cluster$centers # Cluster Centroids
new.x.kv6 new.x.kv8 new.x.kv11 new.x.kv12 new.x.kv15
new.x.kv18 new.x.kv19
1 16.333333 18.333333 17.333333 34.66667
104.00000 102.00000 96.00000
2 5.285714 7.285714 6.285714 12.57143
37.71429 35.71429 29.71429
new.x.kv21 new.x.kv28 new.x.kv32
1 90.00000 20.333333 38.66667
2 23.71429 9.285714 16.57143
Rcmdr> .cluster$withinss # within Cluster sum of
Squares
[1] 10712.000 6774.857
```

```

Rcmdr> .cluster$tot.withinss # Total within sum of
Squares
[1] 17486.86
Rcmdr> .cluster$betweenss # Between cluster sum of
Squares
[1] 39983.54
Rcmdr> biplot(princomp(model.matrix(~-1 + kv6 + kv8 +
kv11 + kv12 + kv15 + kv18 +
Rcmdr+ kv19 + kv21 + kv28 + kv32, kv)), xlabs =
as.character(.cluster$cluster))

```

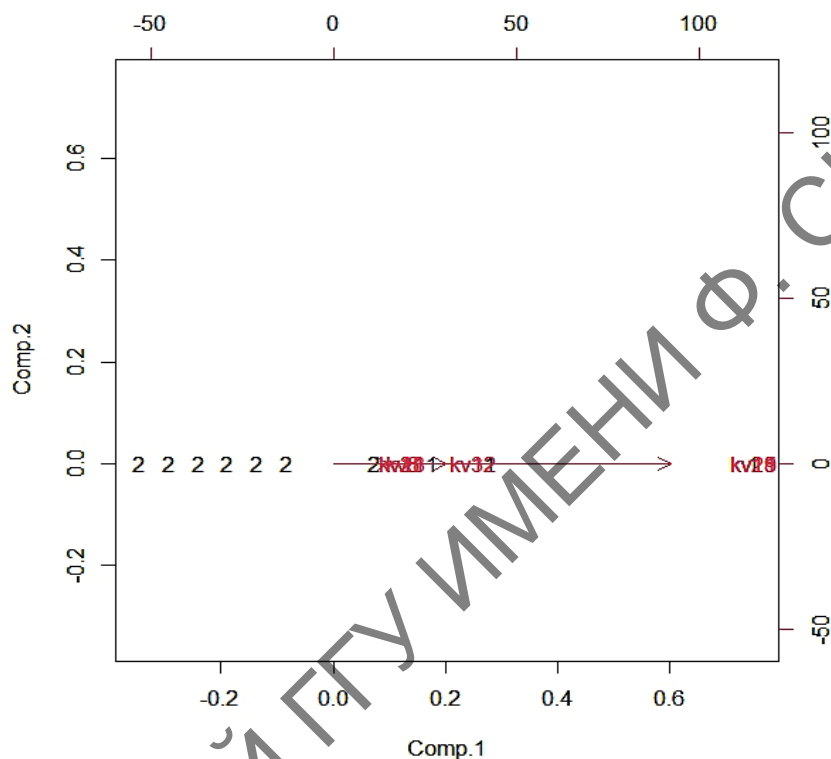


Рисунок 190 – Биplot (двойной график) кластеров в RCommander

### 3 Проведение дендрограммного кластерного анализа в R

#### 3.1 Проведение дендрограммного кластерного анализа в RStudio (при помощи командной строки)

Дендрограммный анализ относится к агломеративным методам кластерного анализа, и его проведение не составляет труда. Главное, что нужно помнить при проведении подобного анализа, – опреде-

лить для себя те правила иерархического объединения кластеров, которые вам необходимы (см. раздел 1 данной темы).

Для проведения иерархического кластерного анализа воспользуемся теми же данными из таблицы 15 и датафреймом *kv*.

**Шаг 1.** Введём промежуточную переменную *d* и присвоим ей значения расчёта квадратов расстояний между объектами датафрейма *kv* (евклидова метрика), который рассчитаем при помощи функции `dist()`.

```
> d <- dist(scale(kv))^2
```

**Шаг 2.** Введём промежуточную переменную *hc* и присвоим ей значения иерархического дендрограммного кластерного анализа. Сам анализ проводится при помощи функции `hclust()`, аргументами которой является переменная с рассчитанными ранее квадратами расстояний и правилом агломерации, например, Уорда.

```
> hc <- hclust(d, method = "ward.D")
```

Кроме правила Уорда, синтаксис функции поддерживает также следующие правила:

- **single** – метод одиночной связи;
- **complete** – метод полной связи;
- **average** – метод средней связи;
- **centroid** – центроидный метод.

**Шаг 3.** Завершает анализ графического изображения дендрограммы связей объектов.

```
> plot(hc, xlab = "Виды", ylab = "Расстояние")
```

Полученную дендрограмму сравните с рисунком 191. На данном рисунке наиболее близкие объекты расположены внизу. Так, например, можно выделить 4 начальных кластера с наиболее близкими по численности видами: 2 и 9, 1 и 5, 8 и 10, 3 и 4. Вид номер 7 стоит особняком от остальных.

Далее кластера видов 1 и 5 и 8 и 10 объединяются в более крупный кластер, который, в свою очередь, объединяется с кластером видов 2 и 9. И так далее, пока не останется два кластера – левое скопление и правое скопление.

Попробуйте самостоятельно проверить каждое их правил агломерации и как они отображаются графически в итоге.

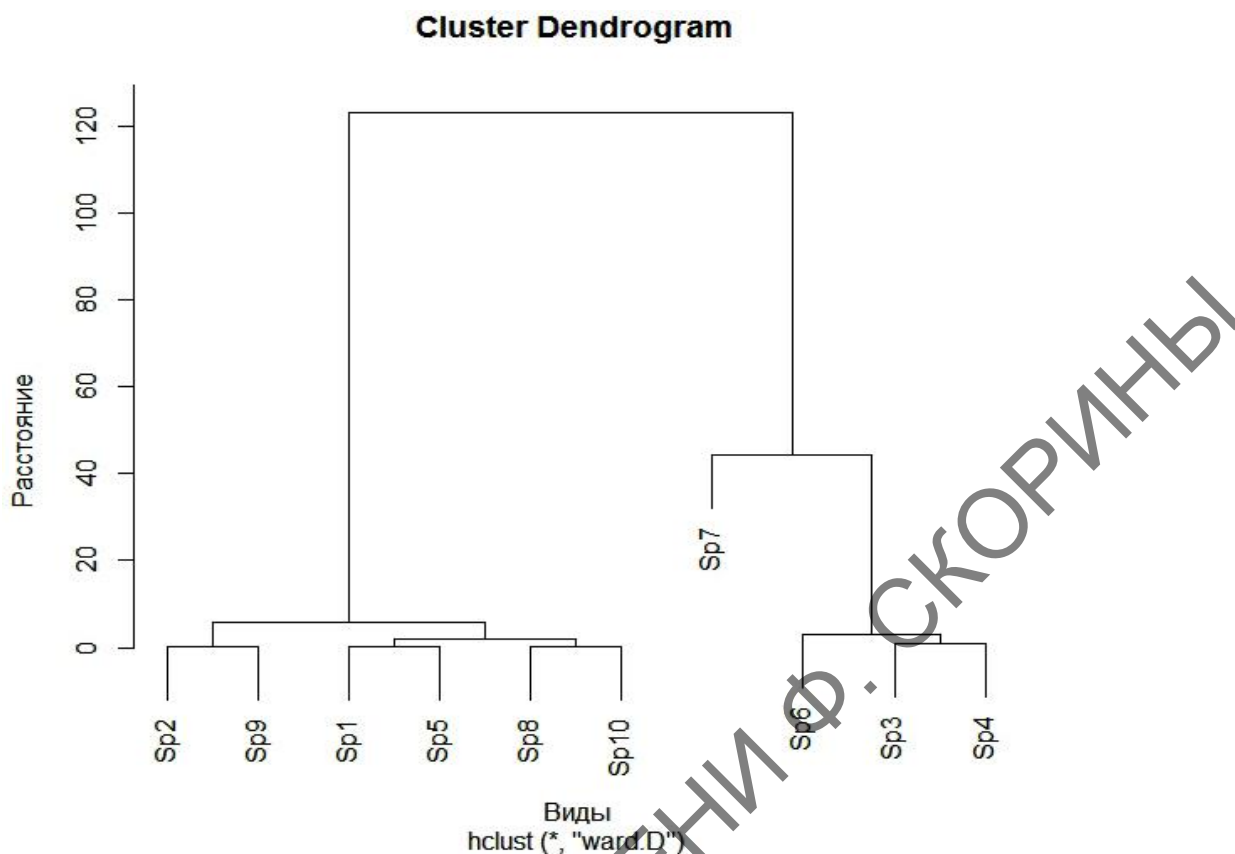


Рисунок 191 – Дендрограмма иерархического кластерного анализа

### 3.2 Проведение дендрограммного кластерного анализа в RCommander (при помощи GUI)

Для проведения иерархического кластерного анализа при помощи средств графического интерфейса пакета RCommander для учебных целей воспользуемся ранее созданной таблицей данных *kv* и загрузим данные в пакет.

**Шаг 1.** Включаем пакет RCommander любым из удобных способов (п. 1.1 темы 1).

**Шаг 2.** Загружаем активные данные в пакет. Используем созданную ранее в RStudio таблицу данных *kv* перейдя в меню по пути **Данные** → **Активные данные** → **Выбрать активный набор данных...** с указанием названия активных данных (в нашем случае – *kv*).

**Шаг 3.** После загрузки активных данных можно перейти непосредственно к проведению кластерного анализа. Для этого необходимо перейти в меню по пути **Статистики** → **Многомерный анализ** → **Кластерный анализ** → **Иерархический кластерный анализ...** (рисунок 192).

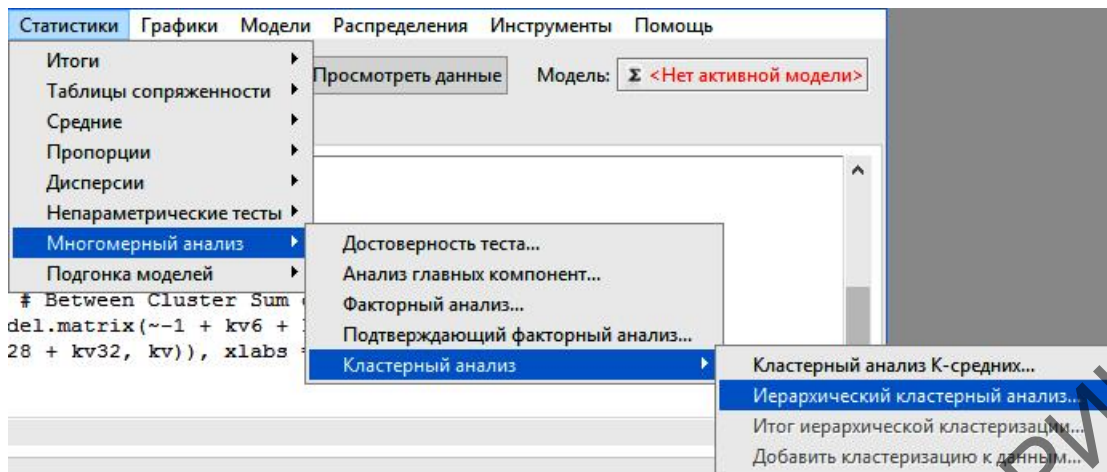


Рисунок 192 – Пункт меню **Иерархический кластерный анализ**

**Шаг 4.** Далее, в диалоговом окне **Иерархический кластерный анализ** в закладке **Данные** нужно выделить все переменные и перейти на закладку **Опции** (рисунок 193).

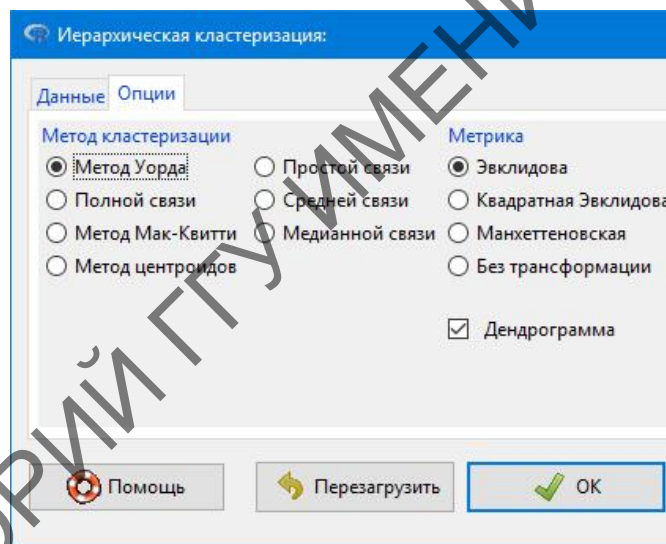


Рисунок 193 – Диалоговое окно **Иерархический кластерный анализ**

**Шаг 5.** В закладке **Опции** нужно выставить метод кластеризации (в нашем случае – Уорда), а также выбрать метрику (в нашем случае – эвклидову). Также следует оставить «галочку» в боксе **Дендрограмма** и тогда по окончании анализа в графическом окне будет отображена итоговая диаграмма. После чего нажать **ОК**.

В результате в окне среды программирования (или в RStudio) будут отражены строка кода и результаты анализа, а также итоговая диаграмма (рисунок 194).

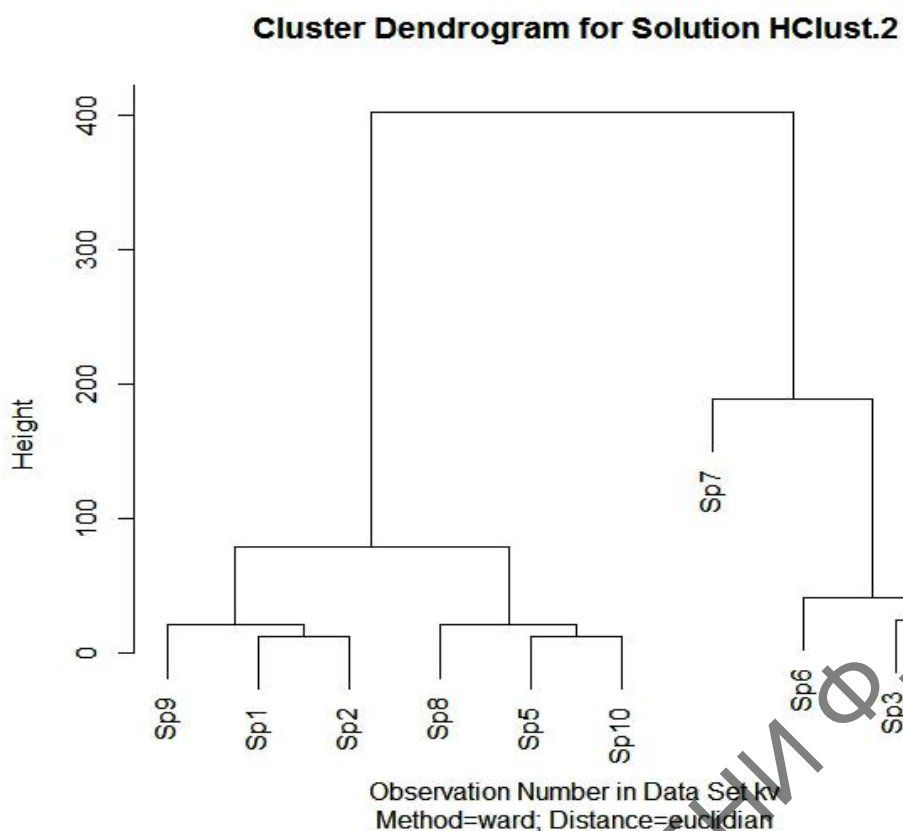


Рисунок 194 – Дендрограмма иерархического кластерного анализа

Попробуйте самостоятельно проверить каждое из правил агломерации и как они отображаются графически в итоге.

## Вопросы для самоконтроля

- 1 В чем суть кластерного анализа? Что представляют собой кластеры?
- 2 Какие методы используются для проведения кластерного анализа?
- 3 Охарактеризуйте метрики, которые используются для расчёта расстояний между центрами кластеров.
- 4 Какую предварительную работу нужно провести для подготовки данных к кластерному анализу в языке программирования R?
- 5 Как можно определить количество необходимых кластеров?
- 6 Опишите синтаксис функций, используемых для проведения кластерного анализа итеративным способом.
- 7 Опишите синтаксис функций, используемых для проведения кластерного анализа агломеративным способом.
- 8 Охарактеризуйте проведение кластерного анализа при помощи графического интерфейса пользователя.

## Задание для самоконтроля

Имеются данные социально-экономического развития и загрязнённости радионуклидами (средние значения суммарной годовой эффективной индивидуальной дозы и удельной активности молока) 36 населённых пунктов.

№ НП	Соц.-экон.	Радиолог.	№ НП	Соц.-экон.	Радиолог.	№ НП	Соц.-экон.	Радиолог.
1	0,271	0,325	13	1,059	0,368	25	0,497	0,373
2	0,540	0,545	14	0,880	0,245	26	0,515	0,471
3	0,845	0,264	15	0,911	0,765	27	0,934	0,328
4	0,922	0,390	16	1,100	0,402	28	0,837	0,277
5	1,013	0,352	17	0,787	0,389	29	0,328	0,221
6	0,921	0,449	18	0,974	0,630	30	0,903	0,482
7	0,706	0,340	19	0,724	0,306	31	0,975	0,277
8	0,830	0,317	20	0,816	0,696	32	0,837	0,587
9	0,929	0,374	21	0,752	0,393	33	0,910	0,397
10	0,476	0,266	22	0,752	0,651	34	0,983	1,100
11	1,014	0,318	23	0,563	0,471	35	0,419	0,505
12	0,906	0,267	24	0,839	0,855	36	0,825	0,331

Классифицируйте их при помощи кластерного анализа (при помощи командной строки и графического интерфейса) на группы (количество необходимых групп определите самостоятельно) и дайте рекомендации о дальнейшей судьбе этих населённых пунктов. Также постройте дендрограмму сходства населённых пунктов по показателям социально-экономического развития.

## Литература по теме

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

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

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

4. Field, A. Discovering statistics using R / A. Field, J. Miles, Z. Field. – London : SAGE Publications Ltd, 2012. – 1193 p.

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.

## ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ ФУНКЦИЙ R, УПОМЯНУТЫХ В ПОСОБИИ

- ? 100
- abline() 60, 167
- anova() 189
- aov() 196
- arrows() 114
- as.integer() 23, 26
- attach() 59, 97, 138, 189
- barplot() 109–111, 113, 114
- boxplot() 96–99, 109–111, 113, 114, 190
- c() 17, 23, 24, 26, 28–34, 36, 58, 69, 74, 80, 85, 86, 89, 90, 97–99, 102, 109, 112, 114, 127–130, 132, 146, 165, 177, 188, 190, 195, 196, 206, 211
- clusplot() 210
- cmdscale() 211
- colMeans() 142
- colSums() 142
- coord\_polar() 131
- cor() 165, 166
- cor.test() 166, 169, 171
- data() 96, 138
- data.frame() 36, 58, 76, 80, 87, 90, 102, 109, 112, 165, 177, 188, 196, 207, 209
- density() 72–74, 141
- detach() 59, 99, 141
- dev.off 133, 134
- dist() 211, 215
- edit() 45
- exists() 47
- facet\_grid() 80
- factor() 31–34, 211
- fix() 45, 46
- for() 208, 211
- geom\_bar() 130
- geom\_boxplot() 100, 103
- geom\_col() 115–118
- geom\_density() 77–79
- geom\_histogram() 76, 77, 79, 80
- geom\_line() 87–91
- geom\_point() 62–64, 88, 89, 91, 171, 172
- geom\_text() 118, 132
- ggplot() 62–64, 76–80, 87–91, 100, 103, 115–118, 130, 171
- getwd() 49
- grid() 179
- hclust() 215
- hist() 69–71, 73, 74, 140, 141
- install.packages() 56, 130, 167, 171, 211
- is.character() 23, 31
- is.factor() 31, 32
- is.integer() 23
- is.logical() 26
- is.numeric() 23
- kmeans() 208, 209, 213
- kruskal.test() 190
- ks.test() 148
- lapply() 142
- legend() 129
- length() 33, 34, 127, 129, 132, 139
- levels() 32, 33, 211
- library() 8, 56, 100, 130, 132, 167, 171, 191, 210
- lines() 73, 86, 141, 179
- list() 29, 114, 209
- lm() 60, 167, 178, 180, 188, 189
- load() 47
- make.names() 207
- matrix() 27
- max() 139, 178



mean() 24, 139, 146  
median() 139, 147  
min() 139, 178  
ordihull() 211  
ordispider() 211  
par() 74  
pdf() 133, 134  
pie() 126, 127, 129  
plot() 58, 59, 72, 85, 86, 132–134,  
166, 178, 208, 211, 215  
png() 133, 134  
points() 86, 211  
qqline() 149, 158  
qqnorm() 149, 158  
quantile() 140  
rainbow() 127, 129  
read.csv() 12, 13  
read.table() 12  
rep() 31, 102, 188  
rm() 13, 38  
round() 129  
rowMeans() 142  
rowSums() 142  
sapply() 142  
save() 47  
scatterplotMatrix() 167, 191  
sd() 139  
setwd() 49  
shapiro.test() 148  
sqrt() 139  
stat\_smooth() 63, 170–172  
str() 17, 27, 36  
sum() 129, 208  
summary() 32, 140, 180, 197  
t() 28  
t.test() 146, 151, 156, 157  
table() 139  
tapply() 114  
var() 139  
view() 37  
which.max() 139  
wilcox.test() 147, 152, 156, 157,  
158  
with() 114, 151, 152, 169  
write.csv() 47  
write.csv2() 47  
write.table() 47  
xlim() 79

## ЛИТЕРАТУРА

1 Гланц, С. Медико-биологическая статистика / С. Гланц. – М. : Практика, 1999. – 459 с.

2 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.

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

4 Численность населения на 1 января 2020 г. по областям и г. Минску с учетом итогов переписи населения 2019 года [Электронный ресурс]. – Режим доступа: [https://www.belstat.gov.by/ofitsialnaya-statistika/ssrd-mvf\\_2/natsionalnaya-stranitsa-svodnyh-dannyh/naselenie\\_6/chislennost-naseleniya1\\_yan\\_poobl](https://www.belstat.gov.by/ofitsialnaya-statistika/ssrd-mvf_2/natsionalnaya-stranitsa-svodnyh-dannyh/naselenie_6/chislennost-naseleniya1_yan_poobl). – Дата доступа: 15.02.2021.

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

6 Chang, W. R Graphics Cookbook / W. Chang [Электронный ресурс]. – Режим доступа: <https://r-graphics.org/index.html>. – Дата доступа: 10.01.2021.

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

8 Field, A. Discovering statistics using R / A. Field, J. Miles, Z. Field. – London : SAGE Publications Ltd, 2012. – 1193 p.

9 Filzmoser, P. Linear and nonlinear methods for regression and classification and applications in R / P. Filzmoser. – Vienna University of Technology – CS, № 3, 2008. – PP. 1–52.

10 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.

11 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.

12 Owen, J. Scientific Programming and Simulation Using R / J. Owen, R. Maillardet, A. Robinson. – London, New York : CRC-Press, 2009. – 455 p.

Учебное издание

**Галиновский** Николай Геннадьевич

**ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ R**

Пособие

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

Подписано в печать 02.02.2022. Формат 60x84 1/16.

Бумага офсетная. Цифровая печать.

Усл. печ. л.13,02. Уч.-изд. л. 14,24.

Тираж 50 экз. Заказ 61.

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

РЕПОЗИТОРИЙ ГГУ ИМЕНИ Ф. СКОРИНЫ