МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Дзержинский филиал ННГУ

В. А. Гришин, М. С. Тихов

МЕТОДЫ ОБРАБОТКИ ДАННЫХ И МОДЕЛИРОВАНИЕ НА ЯЗЫКЕ R

Учебно-методическое пособие

Рекомендовано Объединенной методической комиссией Института открытого образования и филиалов университета для студентов филиалов ННГУ, обучающихся по направлению подготовки 09.03.03 «Прикладная информатика»

Нижний Новгород

2019

УДК - 004.6 (004.4)
ББК – 32.97
Г-85

Г-85 Гришин В.А., Тихов М. С.: МЕТОДЫ ОБРАБОТКИ ДАННЫХ И МОДЕЛИРОВАНИЕ НА ЯЗЫКЕ R: Учебно-методическое пособие — Нижний Новгород: Нижегородский госуниверситет, 2019. – 54 с.

Рецензент: заведующий кафедрой социально-экономических дисциплин Дзержинского филиала ННГУ, д.э.н., профессор М. Н. Павленков

Учебно-методическое пособие предназначено для методической поддержки самостоятельной работы студентов, обучающихся по направлению подготовки 09.03.03 «Прикладная информатика», при изучении дисциплин «Математическое и имитационное моделирование», «Программное обеспечение обработки данных» и «Эконометрика». Пособие включает изучение языка программирования статистической обработки данных R. При освоении R студенты выступают в роли исследователей, которым нужно обрабатывать информацию, решая задачи статистического анализа, визуализации и моделирования больших массивов данных. В пособии также приведены результаты программных кодов обработки различных наборов данных на языке R, что значительно облегчит самостоятельную работу студентов.

Методическое пособие предназначено для студентов Дзержинского филиала ННГУ, обучающихся по направлению 09.03.03 «Прикладная информатика», а также для преподавателей, магистрантов, аспирантов и широкого круга специалистов, проявляющих интерес к изучению курсов «Математическое и имитационное моделирование», «Программное обеспечение обработки данных» и «Эконометрика».

Ответственный за выпуск:
председатель Объединенной методической комиссии Института открытого образования и филиалов университета В.В. Недорослова

УДК 004.06 (004.04)
ББК 32.97

Содержание

Введение

Учебно-методическое пособие предназначено для методической поддержки самостоятельной работы студентов, обучающихся по направлению подготовки 09.03.03 «Прикладная информатика», при изучении дисциплин «Математическое и имитационное моделирование», «Программное обеспечение обработки данных» и «Эконометрика».

Содержание пособия, как и содержание указанных курсов, разработано в соответствии с требованиями ФГОС ВО по направлению 09.03.03 «Прикладная информатика» в области формирования необходимых компетенций.

В пособии даётся представление об одном из самых мощных, профессиональных и современных средств работы с числами, начиная от простых вычислительных задач и заканчивая статистической обработкой больших массивов данных – языком R. R - это свободно распространяемая программная среда с открытым кодом, развиваемая в рамках проекта GNU, и язык программирования для статистической обработки данных и работы с графикой. R можно применять везде, где нужна работа с данными. Это и сама математическая статистика во всех её приложениях, и первичный анализ данных, и математическое моделирование. С помощью R можно подготовить данные для исследования, которое может быть осуществлено с помощью реализованных в различных функциях статистических методов, а затем вывести полученные результаты для дальнейшего анализа.

Самостоятельная работа студентов должна поддерживать различные виды деятельности, чтобы полноценно дополнять аудиторную подготовку. В настоящее издание включены некоторые темы по дисциплинам «Математическое и имитационное моделирование», «Программное обеспечение обработки данных» и «Эконометрика», а также включены программные коды обработки и визуализации различных наборов данных.

Учебно-методическое пособие предназначено для студентов высших учебных заведений, обучающихся по направлению 09.03.03 «Прикладная информатика». Пособие поможет студентам сориентироваться при организации текущей аудиторной и внеаудиторной самостоятельной работы, при подготовке к проверочным работам и экзаменам.

1. Из истории языка R

R – это язык программирования и среда для статистических вычислений и графического анализа, сходный с языком S, первоначально разработанным в1976 году Белл Лабораториз (Bell Labs). Начало пути относится к 1993 г., когда двое молодых новозеландских учёных Росс Ихака (Ross Ihaka) и Роберт Джентльмен (Robert Gentleman), анонсировали свою новую разработку, которую назвали R. Они разработали бесплатную свободную реализацию с легко расширяемой модульной архитектурой. В скором времени возникла распределенная система хранения и распространения пакетов к R, известная под аббревиатурой "CRAN". Это программа для анализа данных с открытым кодом.

Язык R в последнее время стал активно конкурировать с коммерческими статистическими пакетами такими как SAS, SPSS, Stata.

Среди пользователей языка R такие компании как: Boeing, Google, Pfizer, Merck, Bank of America, the InterContinental Hotels Group и Shell.

Ключевая особенность языка R — это его открытость, что дает возможность специалистам самостоятельно создавать собственные алгоритмы. По сути, это framework для разработки. В настоящее время существует огромное количество (более 7000) готовых, бесплатных пакетов для R.

В январе 2015 года фирма Microsoft выкупила компанию Revolution Analytics, которая была основана 2007 году учёными Йельского университета. Помимо непосредственного участия в разработке R компания известна своими продуктами для разработчиков на языке программирования R, предназначенного для решения задач в области статистики и анализа данных, что сегодня особенно актуально в контексте обработки Big Data.

Известно, что Microsoft и раньше применяла свободный язык R, а эта покупка, очевидно, свидетельствует о интересе софтверного гиганта к данному направлению. Джозеф Сирош (Joseph Sirosh), вице-президент Microsoft по машинному обучению: «В Microsoft есть сообщество ученых, специализирующихся на работе с данными и использующими R для анализа бизнес-данных для разных задач, а также создающих модели для некоторых применений». В январе 2016 года Microsoft объявила о появлении языка R в Visual Studio.

В СУБД Oracle для анализа больших объемов данных используется надстройка Oracle Enterprise R**,** которая является частью опции Oracle Advanced Analytics. Oracle R Enterprise специально разработан, чтобы справляться с огромными массивами данных, для этого он использует масштабируемость и параллелизм СУБД Oracle, а также встроенные функции СУБД для оптимизации работы скриптов, написанных на R.

2. Преимущества и недостатки R

R применяется везде, где необходима работа с данными. Более всего его применяют для статистического анализа – от вычисления средних до временных рядов.

Достоинства языка R заключаются в:

- наличии бесплатной кроссплатформенной среды для простого написания программ;
- богатом арсенале статистических методов. В R реализованы сложные статистические процедуры, еще недоступные в других программах;
- качественной векторной графике, с помощью которой реализованы самые разнообразные и мощные методы визуализации данных из доступных;
- получении данных из разных источников в пригодном для использования виде. R может импортировать данные из самых разных источников, включая текстовые файлы, системы управления базами данных, другие статистические программы и специализированные хранилища данных. R может также записывать данные в форматах всех этих систем;
- существовании более 7000 проверенных прикладных пакетов;
- установке на разные операционные системы, включая Windows, Unix, Linux и Mac OS.

У R есть и немало недостатков. Самый главный из них — это трудность обучения программе. Команд много, вводить их надо вручную, запомнить все трудно, а привычной системы меню нет. Поэтому порой очень трудно найти, как именно сделать какой-нибудь анализ. Удобным средством вычислений в R является RStudio. RStudio представляет собой бесплатную интегрированную среду разработки (IDE) для R. Благодаря ряду своих особенностей, этот активно развивающийся программный продукт делает работу с R очень удобной.

Второй недостаток R — относительная медлительность. Некоторые функции, особенно использующие циклы, и виды объектов, особенно списки и таблицы данных, работают в десятки раз медленнее, чем их аналоги в коммерческих пакетах. Но этот недостаток преодолевается, хотя и медленно. Новые версии R умеют делать параллельные вычисления, создаются оптимизированные варианты подпрограмм, работающие много быстрее, память в R используется все эффективнее, а вместо циклов рекомендуется применять векторизованные вычисления.

Что же касается трудности обучения, то мы надеемся, что это пособие послужит одним из средств его преодоления.

3. Установка R и RStudio под Windows

R можно бесплатно скачать из «всеобъемлющего сетевого архива R»(Comprehensive R Archive Network, CRAN) по адресу https://cran.r-project.org/bin/windows/base/. На время инсталляции отключите антивирус.

Следуйте следующим инструкциям по установке:

1) Установите RStudio.
2) Настройте RStudio. Запустите RStudio. Зайдите в раздел Tools — Global options.

В разделе General: а) уберите галочку у Restore .Rdata into workspace in startup; б) выберите “Never” у Save workspace to .Rdata on exit.

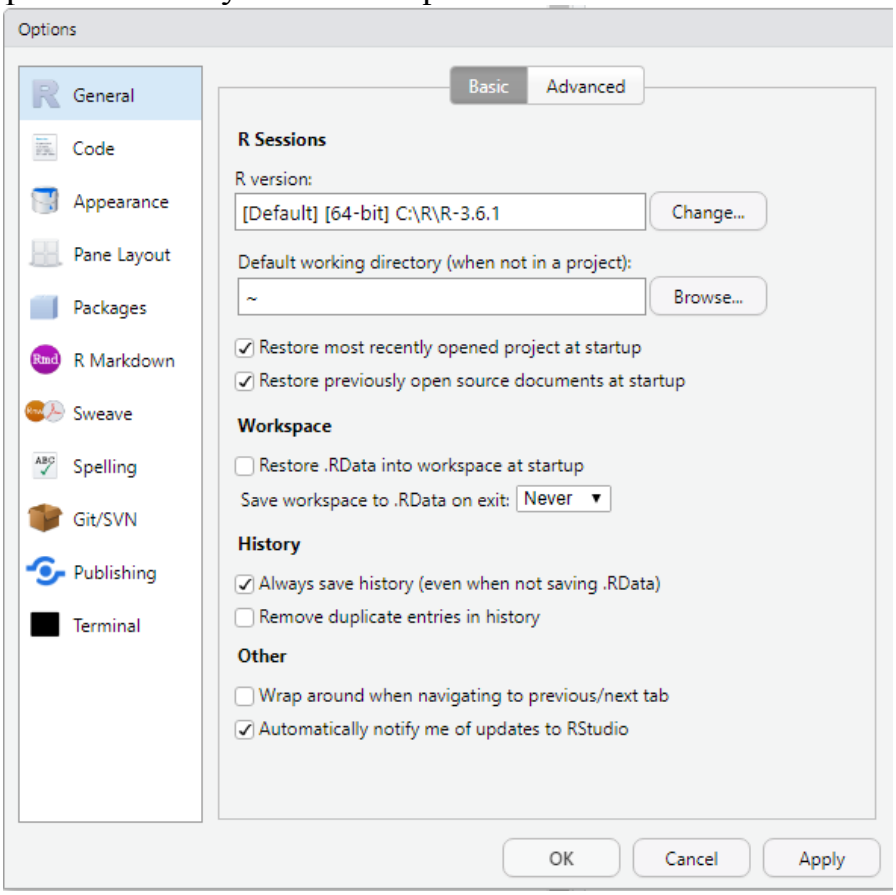


Рис.1 Раздел General

В разделе Sweave: "Weave .Rnw files using" выберите knitr.

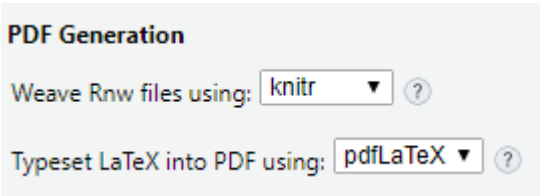


Рис.2 Раздел Sweave

В разделе Code - Diagnostics: выставьте все галочки.

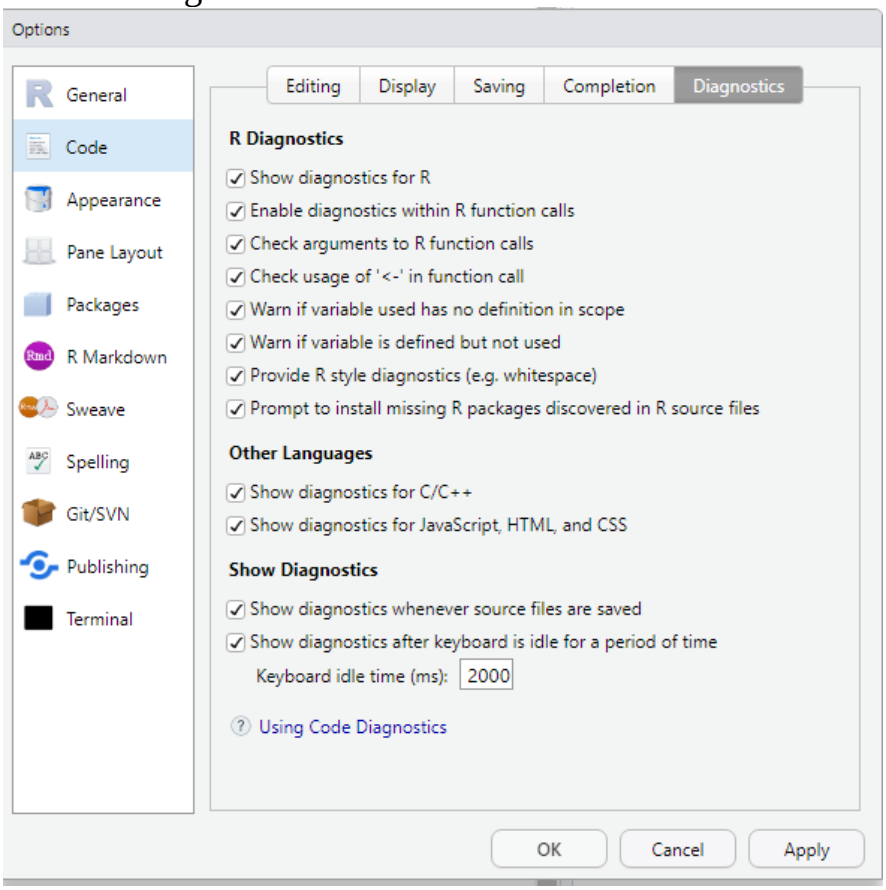


Рис.3 Раздел Code – Diagnostics

3) Установите свежую версию **Rtools**.

Это дополнительные программы, которые позволяют, в частности, из R создавать экселевские файлы.

4) Создайте папку для установки пакетов без русских букв и пробелов, например, **C:/Rlib**.
5) Выполните в консоли команду: **system ("setx R_LIBS C:/ Rlib ")** Вместо C:/Rlib должно быть имя папки, созданной для установки пакетов.
6) Перезапустите RStudio.
7) Проверьте, что R знает, куда ему ставить пакеты. Для этого выполните в консоли RStudio команду: **.libPaths()**. Она должна указать путь к папке **C:/Rlib**. После этого все пакеты будут ставиться в папку **C:/Rlib**.
8) Установите все необходимые пакеты R анализа данных. Чтобы увидеть установленные пакеты на пользовательской инсталляции, введите команду: **library()**.

4. Получение помощи по функциям и средствам

R обладает обширными справочными материалами. Встроенная система помощи содержит подробные разъяснения, ссылки на литературу и примеры для каждой функции из установленных пакетов. Справку можно вызвать при помощи функций, перечисленныхв табл. 1.

Таблица 1

Функции вызова справки в R

Функция	Действие
help.start()	Общая справка
help (“*предмет*”) или ? *предмет*	Справка по функции *предмет* (кавычки необязательны)
help.search(“*предмет*”) или ??*предмет*	Поиск в справке записей, содержащих *предмет*
example(“*предмет* ”)	Примеры использования функции *предмет* (кавычки необязательны)
RSiteSearch(“*предмет* ”)	Поиск записей, содержащих *предмет* в онлайн-руководствах и заархивированных рассылках
apropos(“*предмет* ”, mode=”function”)	Список всех доступных функций, в названии которых есть *предмет*
data()	Список всех демонстрационных данных, содержащихся в загруженных пакетах
vignette()	Список всех доступных руководств по загруженным пакетам
vignette(“*предмет* ”)	Список руководств по теме *предмет*
library(help="библиотека")	Справка о библиотеке

Функция **help.start()** открывает окно браузера с перечнем доступных руководств разного уровня сложности, часто задаваемых вопросов и ссылок на источники. Функция **RSiteSearch()** осуществляет поиск на заданную тему в онлайн-руководствах и архивах рассылок и представляет результаты в окне браузера. Функция **vignette()** вызывает список вводных статей в формате PDF. Такие статьи написаны не для всех пакетов.

5. Начало работы с RStudio

RStudio представляет собой бесплатную интегрированную среду разработки (IDE) для R. Структура пользовательского интерфейса RStudio с подсказками приведена на рис.4.

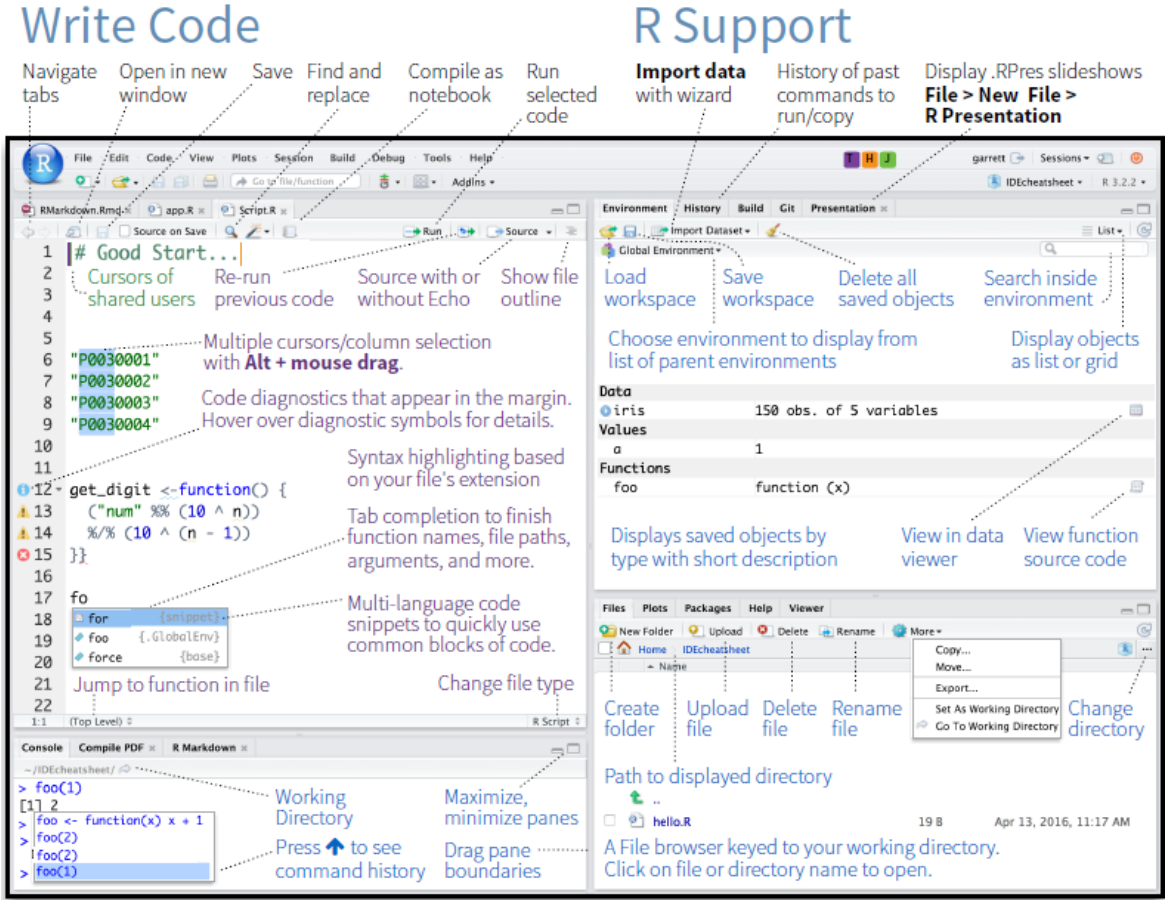


Рис. 4 RStudio IDE

Консоль Rstudio **(Console)** предоставляет целый ряд опций, делающих работу с R простой и продуктивной. Rstudio поддерживает автоматическое завершение кода при помощи клавиши **Tab**. Например, если в рабочем пространстве имеется объект с именем **matrix()**, то можно набрать на клавиатуре **matr**, нажать **Tab**, и **Rstudio** автоматически завершит название этого объекта. Аналогично можно получать подсказки по функциям при введении их имен. Например, введя название функции app и нажав на Tab, получим следующее:

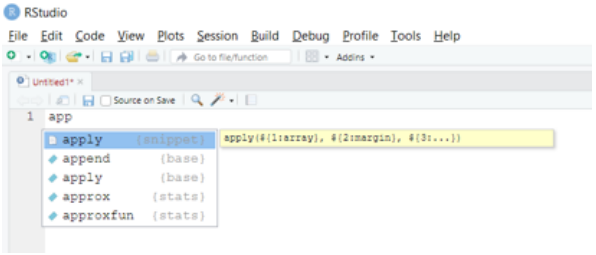


Рис. 5 Автоматическое завершение кода

При работе с R постоянно возникает необходимость выполнить заново ту или иную команду, которая уже была выполнена ранее. Как и стандартная R-консоль, консоль RStudio поддерживает возможность навигации по ранее выполненным командам, используя клавиши со стрелками.

Для просмотра списка недавно выполненных команд и для выбора определенной команды из этого списка можно использовать также сочетание клавиш **Ctrl+Вверх**.

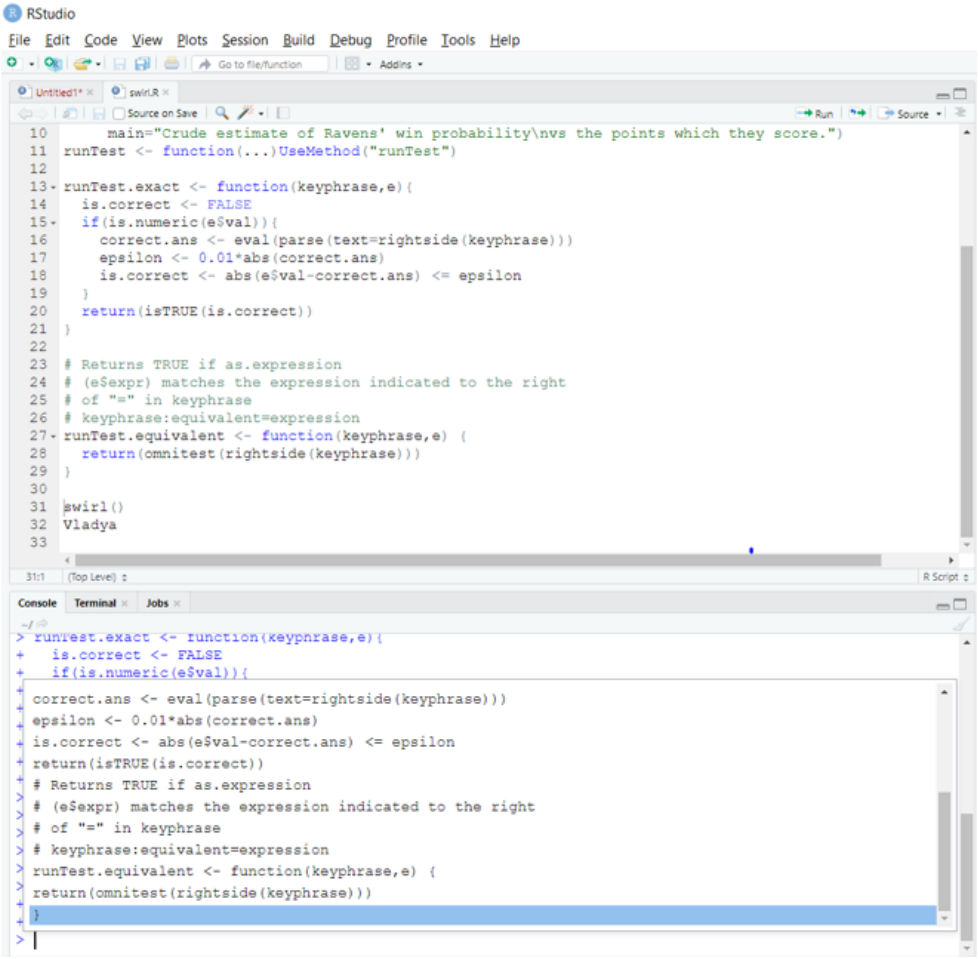


Рис. 6 Просмотр списка выполненных команд

Такое же сочетание клавиш подходит и для быстрого поиска ранее вызванных функций, в имени которых имеется определенный префикс. Например, для поиска функций, в имени которых есть **plot**, следует просто ввести **plot** и нажать **Ctrl+Вверх.**

Горячие клавиши консоли:

Ctrl+1 - перемещает курсор в окно Редактора кода

Ctrl+2 - перемещает курсор в Консоль

Ctrl+L - очищает окно Консоли от текста

Esc - прерывает вычисления

Редактор кода RStudio включает ряд опций для продуктивной работы, в частности подсветку кода, автоматическое завершение кода и др.

Кроме того, в RStudio имеются гибкие возможности по выполнению кода непосредственно из окна редактора. Для многих пользователей это является предпочтительным способом работы с R. Выполнение команд из окна Редактора кода вместо командной строки Консоли облегчает воспроизведение одних и тех же команд и позволяет "упаковать" такие команды в одну функцию для последующего использования.

RStudio поддерживает подсветку синтаксиса и другие специализированные опции по работе с кодом следущих типов файлов:

— R Script;
— R Notebook;
— Text File;
— C++ File и др.

Для создания нового файла используется меню File→ New File:

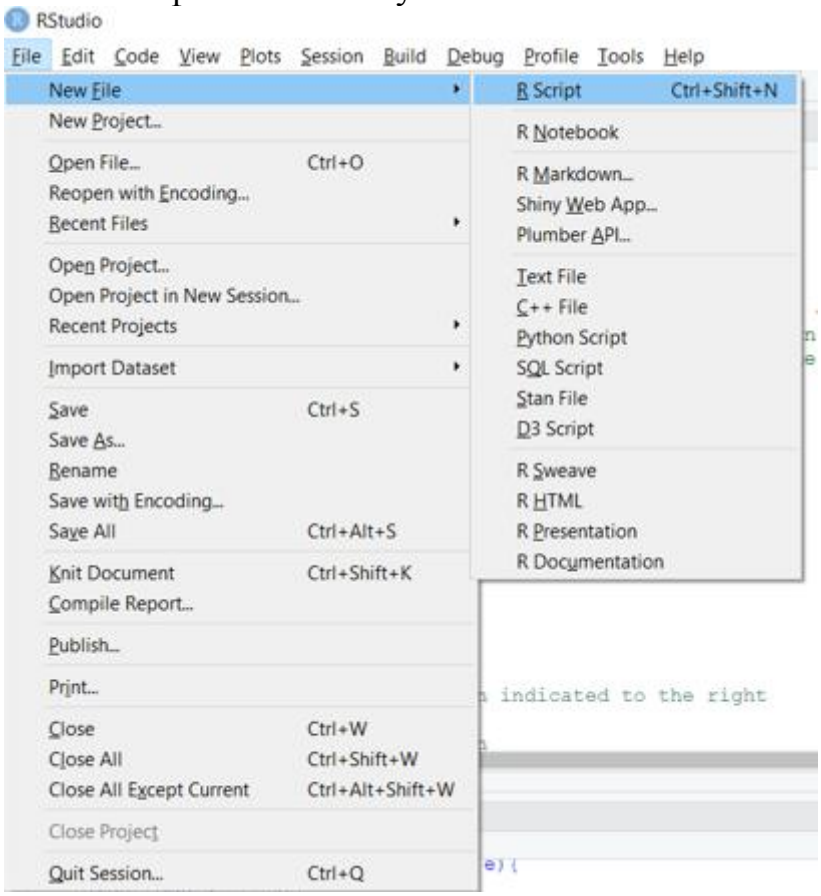


Рис. 7 Меню File

Для открытия существующего файла необходимо воспользоваться меню **File → Open** или **Open Recent** (для открытия файла, с которым работа осуществлялась недавно). Если открыто несколько файлов одновременно, быстрый переход от одного документа к другому выполняется при помощи соответствующих закладок в верхней части окна редактора кода.

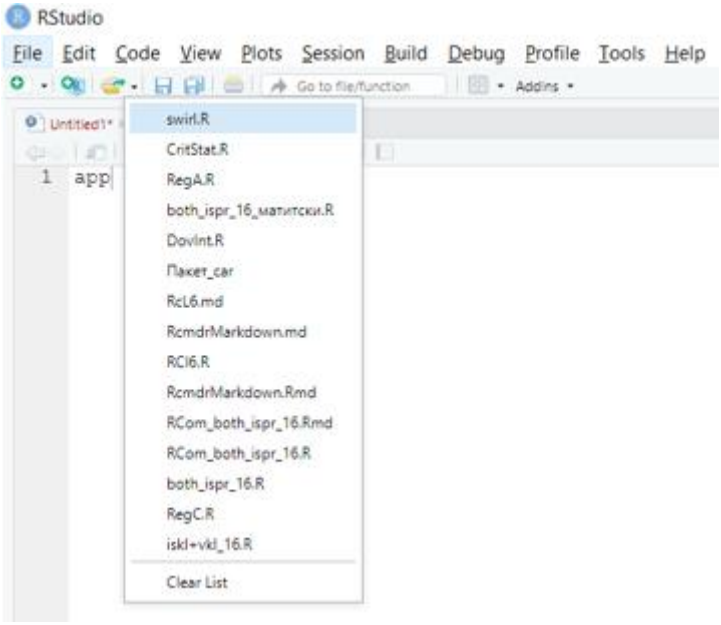


Рис. 8 Меню Open Recent

RStudio поддерживает выполнение кода непосредственно из окна Редактора (выполняемые команды посылаются в Консоль, где появляется также результат их выполнения).

Для выполнения текущей строки кода можно воспользоваться сочетанием клавиш **Ctrl+Enter** или кнопкой **Run Line(s)**, расположенной в верхней части окна Редактора (рис.9).

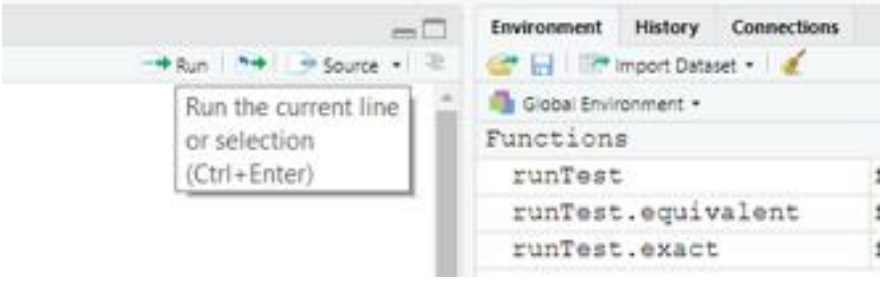


Рис. 9 Кнопка Run Line(s)

6. Пакеты

R в базовой установке уже обладает обширными возможностями. Однако некоторые наиболее впечатляющие опции программы реализованы в дополнительных модулях, которые можно скачать и установить. Существует более 7000 созданных пользователями модулей, называемых пакетами (packages), которые можно скачать с https://www.r-project.org/.

Пакеты – это собрания функций R, данных и скомпилированного программного кода в определенном формате. Директория, в которой пакеты хранятся на компьютере, называется библиотекой.

Все пакеты R относятся к одной из трех категорий: базовые ("base"), рекомендуемые ("recommended") и прочие, установленные пользователем. Получить их список на конкретном компьютере можно, подав команду **library()** или:

- **installed.packages(priority = "base")**
- **installed.packages(priority = "recommended")**
- **packlist<rownames(installed.packages());write.table(packlist,"clipboard",sep="\t", col.names=NA)** – полный список пакетов в формате Excel.
- **CRAN Task Views** – тематический каталог пакетов.

Команда **search()** выводит на экран названия загруженных и готовых к использованию пакетов.

Для установки пакета используется команда **install.packages()**. Например, пакет **MASS** содержит набор данных и статистических функций. Этот пакет можно скачать и установить при помощи команды **install.packages("MASS")**. Пакет нужно установить только один раз. Однако, как и любые другие программы, пакеты часто обновляются их разработчиками. Для обновления всех установленных пакетов используется команда **update.package()**. Для получения подробной информации об установленных пакетах можно использовать команду **installed.packages()**. Она выводит на экран список всех имеющихся пакетов с номерами их версий, названиями пакетов, от которых они зависят, и другой информацией. Установить, какие пакеты загружены в каждый момент проводимой сессии, можно, подав **команду sessionInfo()**.

Для использования пакета в текущей сессии программы нужно загрузить его при помощи команды **library()**. Например, для того чтобы использовать пакет **MASS**, надо ввести команду **library(MASS)**.

Получить список функций каждого пакета можно, например, подав команду **ls(pos = "package:MASS")**.

Получить список аргументов входящих параметров любой функции загруженного пакета можно, подав команду **args()**. Например, **args(lm)**, где **lm()** функция получения линейной регрессионной модели.

7. Создание набора данных

R работает с самыми разными структурами данных, включая скаляры, векторы, массивы данных, таблицы данных и списки. Такое большое разнообразие поддерживаемых структур дает языку R большую гибкость в работе с данными.

Типы данных в R бывают числовыми (numeric), текстовыми (character), логическими (logical), комплексными (complex) и необработанными (байты).

Операторами присваивания в R выступают либо =, либо <- (состоящий из двух символов: < и -) — присваивание справа, либо -> — присваивание слева, как поодиночке, так и в последовательности. Присваивание справа принято среди профессионалов.

7.1. Организация данных в R

Вектор представляет собой поименованный одномерный объект, содержащий набор однотипных элементов (числовые, логические, либо текстовые значения – никакие их сочетания не допускаются). Для создания векторов небольшой длины в R используется функция конкатенации **c()** (от "concatenate" – объединять, связывать). В качестве аргументов этой функции через запятую перечисляют объединяемые в вектор значения, например:

```
> a<-c(0,1,3,2,5,4,7)
> a
[1] 0 1 3 2 5 4 7
>
```

Вектор можно создать также при помощи функции **scan()**, которая "считывает" последовательно вводимые с клавиатуры значения:

```
> q = scan()
1: 1.1
2: 1.2
3: 1.3
4: 1.4
5: 1.5
6: # выполнение команды scan завершают введением пустой строки
Read 5 items# программа сообщает о считывании 5 значений
> q
[1] 1.1 1.2 1.3 1.4 1.5
>
```

Для создания векторов, содержащих последовательную совокупность чисел, удобна функция **seq()** (от "sequence" – последовательность). Так, вектор

с именем V, содержащий совокупность целых чисел от 10 до 20, можно создать следующим образом:

> V = seq(10,20)

> V

[1] 10 11 12 13 14 15 16 17 18 19 20

Идентичный результат будет получен при помощи команды

> V = 10:20

> V

[1] 10 11 12 13 14 15 16 17 18 19 20

>

В качестве дополнительного аргумента функции **seq()** можно задать шаг приращения чисел:

> b<-seq(from = 1,to = 11,by = 2)

> b

[1] 1 3 5 7 9 11

В R несколько векторов можно объединить в один, используя уже рассмотренную выше функцию конкатенации:

> v1 <- c(1, 2, 3)

> v2 <- c(4, 5, 6)

> V <- c(v1, v2)

> V

[1] 1 2 3 4 5 6

Отдельные элементы вектора можно вызывать при помощи числового вектора, состоящего из номеров элементов и заключенного в квадратные скобки. Например, **a[c(2, 4)]** обозначает второй и четвертый элементы вектора **a**:

> a[c(2, 4)]

[1] 1 2

Другие примеры:

> a[5]

[1] 5

> a[2:5]

[1] 1 3 2 5

Для упорядочения значений вектора по возрастанию или убыванию используют функцию **sort()** в сочетании с аргументом **decreasing = FALS**E (по умолчание) или **decreasing = TRUE**:

> sort(a)

[1] 0 1 2 3 4 5 7

> sort(a,decreasing =TRUE)

[1] 7 5 4 3 2 1 0

Матрица (matrix) – это двумерный массив данных, в котором каждый элемент имеет одинаковый тип (числовой, текстовый или логический). Матрицы создают при помощи функции **matrix**:

> mymat <- matrix(seq(1, 9), nrow = 3, ncol = 3)

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

Для придания пользовательских заголовков строкам и столбцам матриц используют функции **rownames()** и **colnames()** соответственно. Например, для обозначения строк матрицы **mymat** буквами A, B, C и D необходимо выполнить следующее:

```
> rownames(mymat) <- c("A", "B", "C")
> mymat
  [,1] [,2] [,3]
A    1    2    3
B    4    5    6
C    7    8    9
>
```

Все векторные операции применимы в отношении матриц. Индексированием можно извлекать из матриц необходимые элементы:

```
> y=matrix(9:17,ncol = 3,byrow = TRUE)
> y
     [,1] [,2] [,3]
[1,]    9   10   11
[2,]   12   13   14
[3,]   15   16   17
> y[5]
[1] 13
> y[,2]
[1] 10 13 16
> y[2,]
[1] 12 13 14
> y[1,c(2,3)]
[1] 10 11
```

Списки (list). Каждый компонент списка может являться переменной, вектором, матрицей, другим списком. Кроме того, эти элементы могут принадлежать к различным типам: числа, строки символов, булевы переменные. Списки являются наиболее общим средством хранения

внутрисистемной информации: в частности, результаты большинства статистических анализов в программе R хранятся в объектах списках. Для создания списков в R служит одноименная функция **list()**. Рассмотрим пример:

```
vector1 <- c("A", "B", "C")
vector2 <- seq(1, 3, 0.5)
vector3 <- c(FALSE, TRUE)
my.list <- list(Text=vector1, Number=vector2, Logic=vector3)
```

Просмотрим содержимое созданного списка:

```
> my.list
$Text
[1] "A" "B" "C"
$Number
[1] 1.0 1.5 2.0 2.5 3.0
$Logic
[1] FALSE  TRUE
```

К элементам списка можно получить доступ посредством трех различных операций индексации. Для обращения к поименованным компонентам применяют знак **$**. Так, для извлечения компонентов Text, Number и Logic из созданного списка **my.list** необходимо последовательно ввести следующие команды:

```
> my.list$Text
[1] "A" "B" "C"
> my.list$Number
[1] 1.0 1.5 2.0 2.5 3.0
> my.list$Logic
[1] FALSE  TRUE
```

Имеется возможность извлекать из списка не только его поименованные компоненты-векторы, но и отдельные элементы, входящие в эти векторы:

```
> my.list$Text[2]
[1] "B"
> my.list$Number[3:5]
[1] 2.0 2.5 3.0
> my.list$Logic[1]
[1] FALSE
```

Извлечение компонентов списка можно осуществлять также с использованием двойных квадратных скобок, в которые заключается номер компонента списка:

```
> my.list[[1]]
[1] "A" "B" "C"
> my.list[[2]]
[1] 1.0 1.5 2.0 2.5 3.0
> my.list[[3]]
[1] FALSE  TRUE
> my.list[[1]][2]
```

```
[1] "B"
> my.list[[2]][3:5]
[1] 2.0 2.5 3.0
> my.list [[3]][1]
[1] FALSE
```

Для выяснения структуры объектов в языке R имеется специальная функция **str()**:

```
> str(my.list)
List of 3
 $ Text  : chr [1:3] "A" "B" "C"
 $ Number: num [1:5] 1 1.5 2 2.5 3
 $ Logic : logi [1:2] FALSE TRUE
```

Таблица данных (data frame) – представляет собой объект R, по структуре напоминающий лист электронной таблицы Microsoft Excel. Каждый столбец таблицы является вектором, содержащим данные определенного типа. Таблица данных создается при помощи функции **data.frame()**:

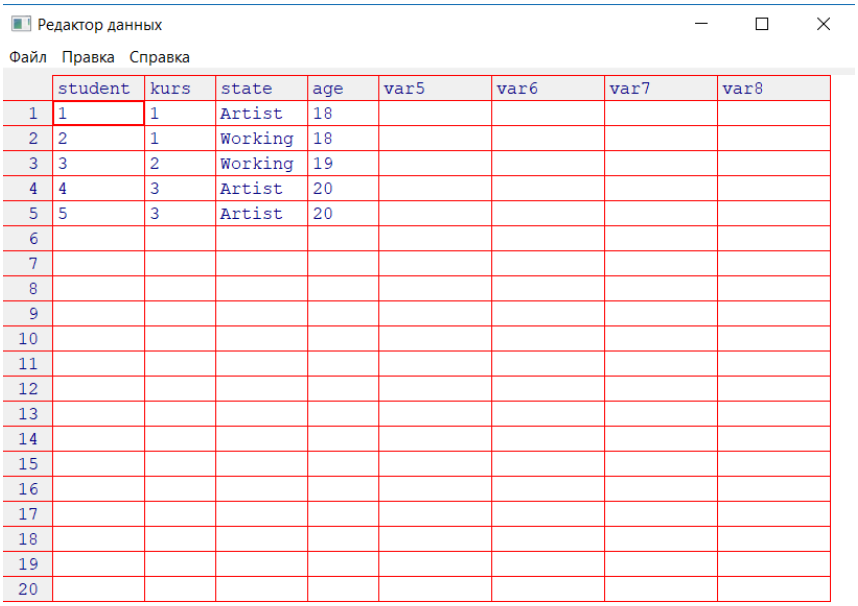
mydata <- data.frame(col1, col2, col3,…),

где – col1, col2, col3,… это векторы любого типа (текстового, числового или логического), которые станут столбцами таблицы. Названия каждого столбца можно прочитать при помощи функции **names()**. Проиллюстрируем сказанное при помощи следующего программного кода:

```
> student=c(1,2,3,4,5)
> age=c(18,18,19,20,20)
> kurs=c(1,1,2,3,3)
> state=c("Artist","Working","Working","Artist","Artist")
> studentdata=data.frame(student,kurs,state,age)
> studentdata
  student kurs   state age
1       1    1  Artist  18
2       2    1 Working  18
3       3    2 Working  19
4       4    3  Artist  20
5       5    3  Artist  20
> studentdata[2:3]
  kurs   state
1    1  Artist
2    1 Working
3    2 Working
4    3  Artist
5    3  Artist
> studentdata["state"]
    state
1  Artist
2 Working
```

3 Working
4 Artist
5 Artist
> studentdata$age
[1] 18 18 19 20 20

При необходимости внесения исправлений в таблицу можно воспользоваться встроенным в R редактором данных. Внешне этот редактор напоминает обычный лист Excel, однако имеет весьма ограниченные функциональные возможности. Все, что он позволяет делать, это добавлять новые или исправлять уже введенные значения переменных, изменять заголовки столбцов, а также добавлять новые строки и столбцы. Работая в стандартной версии R, редактор данных можно запустить выполнив команду **fix()** из командной строки консоли R (например, **fix(studentdata)**). После внесения исправлений редактор просто закрывают – все изменения будут сохранены автоматически.



	student	kurs	state	age	var5	var6	var7	var8
1	1	1	Artist	18				
2	2	1	Working	18				
3	3	2	Working	19				
4	4	3	Artist	20				
5	5	3	Artist	20				
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								

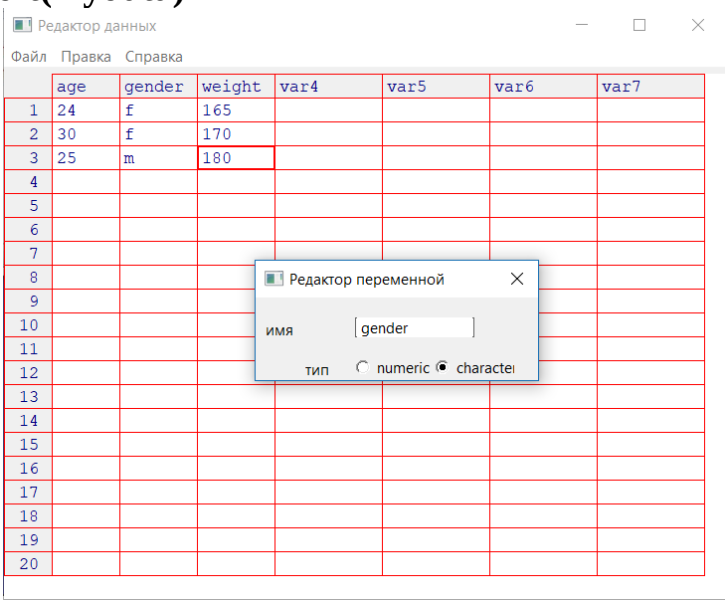
Рис. 10 Редактор данных

7.2. Ввод данных

Самый простой способ введения данных – это ввод с клавиатуры. Для этого необходимо создать пустую таблицу данных (или матрицу), указав названия и типы переменных. Затем открывается текстовый редактор функцией **edit()** с этим объектом, вводятся данные и сохраняется результат в виде объекта с данными.

В приведенном ниже примере создается таблица данных с названием **mydata** с тремя переменными: **age** (возраст, числовая), **gender** (пол, текстовая) и **weight** (вес, числовая). Затем откроется текстовый редактор, вносятся данные и сохраняется результат.

```
> mydata <- data.frame(age=numeric(0),
+                      gender=character(0), weight=numeric(0))
> mydata <- edit(mydata)
```



	age	gender	weight	var4	var5	var6	var7
1	24	f	165				
2	30	f	170				
3	25	m	180				
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

Рис.11 Копия объекта **mydata** открытая в редакторе данных функцией **edit()**

Этот метод хорошо работает для небольших объемов данных. Для наборов данных большего размера выполняется импорт данных из существующих текстовых файлов, электронных таблиц Excel и статистических пакетов.

Импорт данных из текстовых файлов с разделителями возможен при помощи команды **read.table()**, функции, которая сохраняет данные в виде таблицы.

Если файл сценария находится в текущем каталоге, то , например, набирается следующий программный код

gdp17 <- read.table("GDP.txt", header = TRUE, sep = "\t"),

где опция header = TRUE указывает, что считывается и строка заголовков, опция sep = "\t", указывает на то, что столбцы таблицы разделены символом табуляции.

Смена каталога выполняется командой **setwd(“D:/”)**, команда **getwd()** выводит текущий рабочий каталог.

Лучший способ прочесть файл в формате **Excel** – это сохранить его в формате текстового файла с разделителями и импортировать в R, как это описано выше.

Аналогом **read.table()** для считывания **csv**-файлов является функция **read.csv()**:

gdp18<- read.csv("GDP.csv", header = TRUE, sep = ";")

8. Визуализация данных с помощью функций plot() и hist()

Особенность R — возможность создания качественной графики с широким спектром графических возможностей R. На сайте **R Graph Gallery** (https://www.r-graph-gallery.com/all-graphs) представлены не только примеры всевозможных графиков, но и исходный R-код, написанный для их построения.

Базовая графика построена на функциях высокого уровня. Часто используются графические функции **plot()** и **hist()**.

Функция **plot()** – функция общего назначения, которая используется для построения графиков. С помощью **plot()** можно создавать диаграммы рассеяния, точечные графики с гладкими линиями и маркерами, точечные графики с отрезками линий и маркерами и др..

Новая диаграмма, которая создается при помощи команды высокого уровня, обычно заменяет предыдущую диаграмму. Для создания ещё одной диаграммы, сохранив предыдущую, в стандартной графической оболочке **RGui** можно открыть новое графическое устройство функцией **dev. new()**. Каждая новая диаграмма будет появляться в последнем открытом окне.

Можно использовать функции **dev.next()**, **dev.prev(), dev.set()** и **dev.off()** для одновременного открытия нескольких окон графики, выбора необходимой диаграммы и закрытия окон.

В R очень легко объединить несколько диаграмм в одну общую, используя функции **par()** или **layout()**. Имеется возможность добавить графический параметр **mfrow=c(nrows, ncols)** в функцию **par()** для создания матрицы из диаграмм размером nrows×ncols, которая будет заполнена по рядам. Для заполнения этой матрицы по столбцам нужно использовать параметр mfcol=c(nrows, ncols).

Например, следующий программный код позволяет создать две диаграммы и расположить их в две строки и один столбец:

```
> library(car)
Загрузка требуемого пакета: carData
> attach(mtcars)
> opar <- par(no.readonly=TRUE)
> par(mfcol=c(2,1))
> plot(wt,mpg, main = "Диаграмма рассеяния для расхода топлива и веса машины")
> plot(wt,disp, main="Диаграмма рассеяния для объёма двигателя и веса машины")
> par(opar)
> detach(mtcars)
```

Функция **layout()** позволяет делать более сложное разделение: она делит активное графическое окно на несколько частей, где графики будут отображены последовательно. Например, деление окна на четыре равные части:

> layout (matrix (c (1,2,3,4), 2, 2))

где вектор дает числа подокон и два числа 2 указывают на то, что это окно будет разделено на две строки и два столбца.

Исходные данные и график (по умолчанию - точки, опция **pch** определяет вид точек):

> x <- 0:10
> y <- sin (x)
> plot(x,y,pch =15)
>

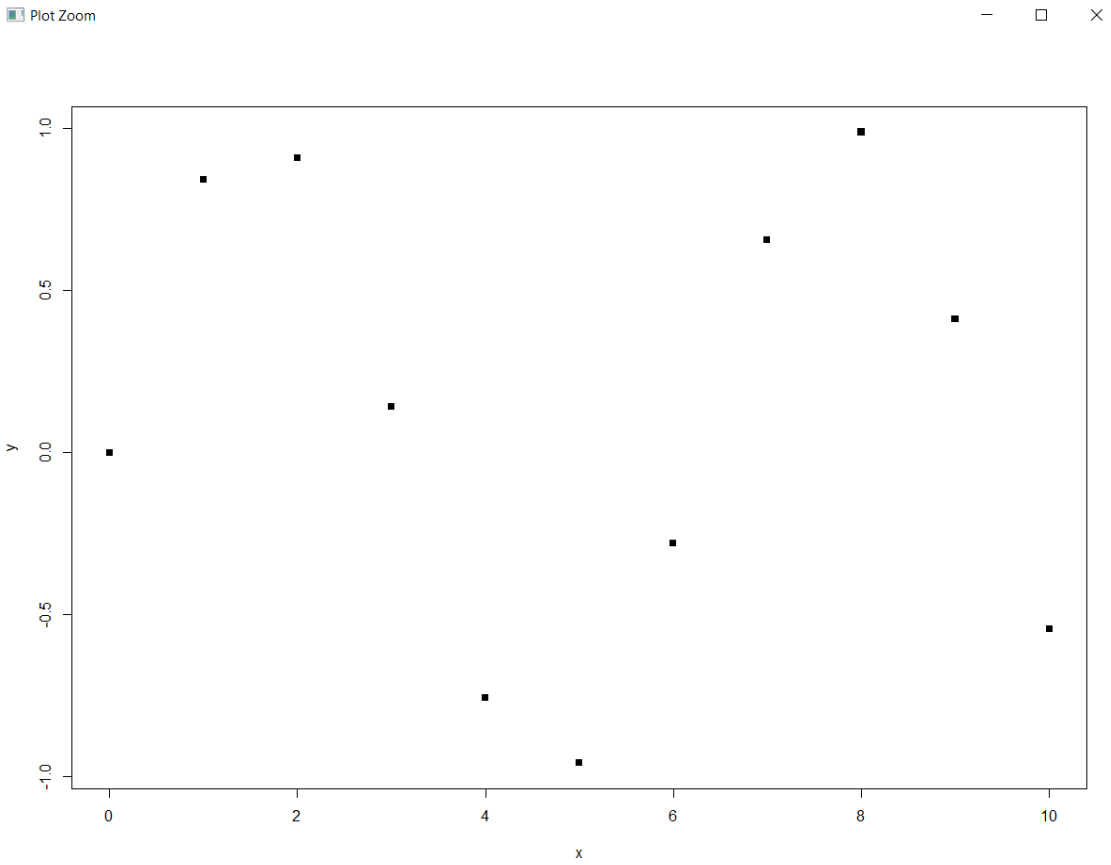


Рис. 12 Диаграмма синусоиды с точками в виде закрашенных квадратов

Чтобы добавить новый график к уже существующему рисунку используются функции **lines** (опция **col** задаёт цвет, опция **lwd** — толщину линии) и **points**:

> t <-seq (from=min (x),to=max (x),by =0.01)

> lines (t, sin (t), col ="red ", lwd =3)

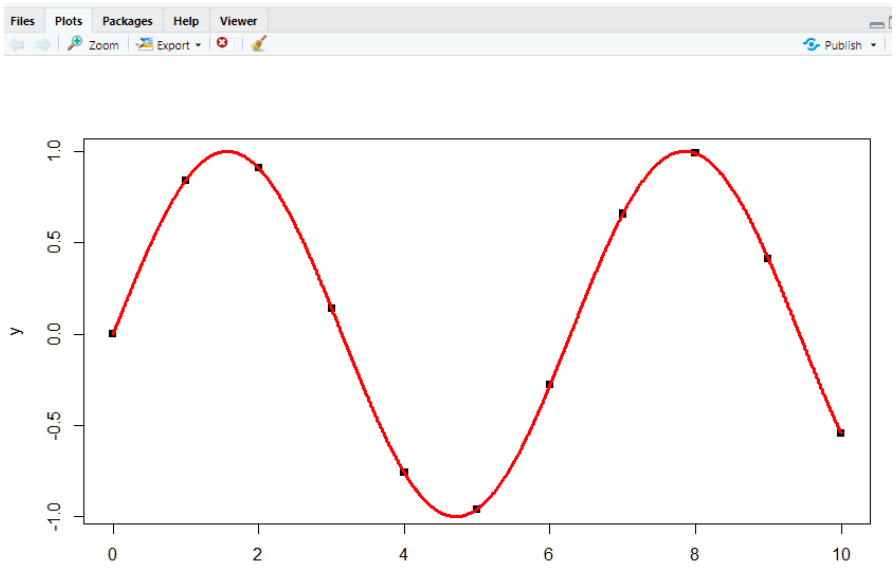


Рис. 13 Линия синусоиды добавленная к рис. 12

Параметры функции **plot**:

type — тип графика ("p" — точки (по умолчанию), "l" — линия, "b"— точки и отрезки, "o"— точки и линия, "n" — ничего не рисуется).

col — цвет ("red", "blue", "green", "cyan", "magenta", "black" и др.). Также можно определить цвет с помощью функции rgb(r,g,b), аргументы которой могут иметь значения в диапазоне от 0 до 1.

xlab, ylab — названия осей абсцисс и ординат.

xlim, ylim — векторы из двух элементов, определяющие размеры диапазонов по x и y. Если второй элемент вектора меньше первого, то ось меняет направление.

main, sub — основное и дополнительное названия (вверху и внизу рисунка).

lty — тип линии ("blank" — нет линии, "solid" — сплошная, "dashed" — пунктирная, "dotted" — точки, "dotdash" — штрихпунктирная, "longdash" — длинные штрихи, "twodash" — короткие и длинные штрихи).

lwd — толщина линии.

pch — вид точек (0 — квадратики, 1 — кружки, 2 — треугольники, 3 — крестики и до 24).

log — задание логарифмического масштаба для указанной оси ("x", "y" или "xy").

asp — число, задающее пропорции окна (y/x).

Гистограммы рисуются с помощью функции **hist**. Параметры идентичны функции **plot()**.

В качестве примера создадим нормально распределенную совокупность X из 100 наблюдений со средним значением 0 и стандартным отклонением 1:

> X <- rnorm (100) # N(0, 1)

Строим гистограмму совокупности функцией **hist**:

> hist(X)

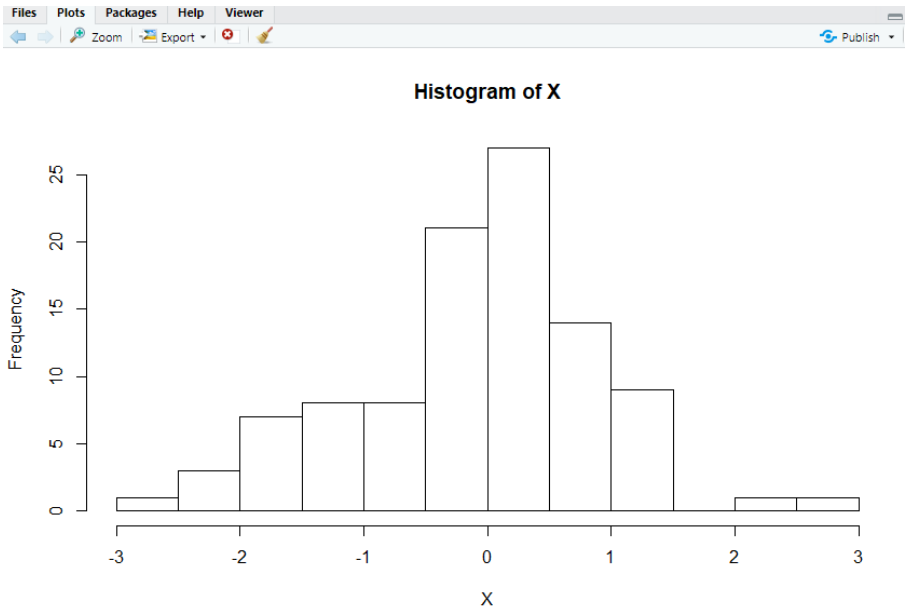


Рис. 14 Гистограмма совокупности с автоматически выбранными настройками

Для детального изучения совокупности можно увеличить количество столбцов аргументом **breaks**. При необходимости залить столбцы цветом аргументом **col**:

> hist(X, breaks = 20,col = "blue")

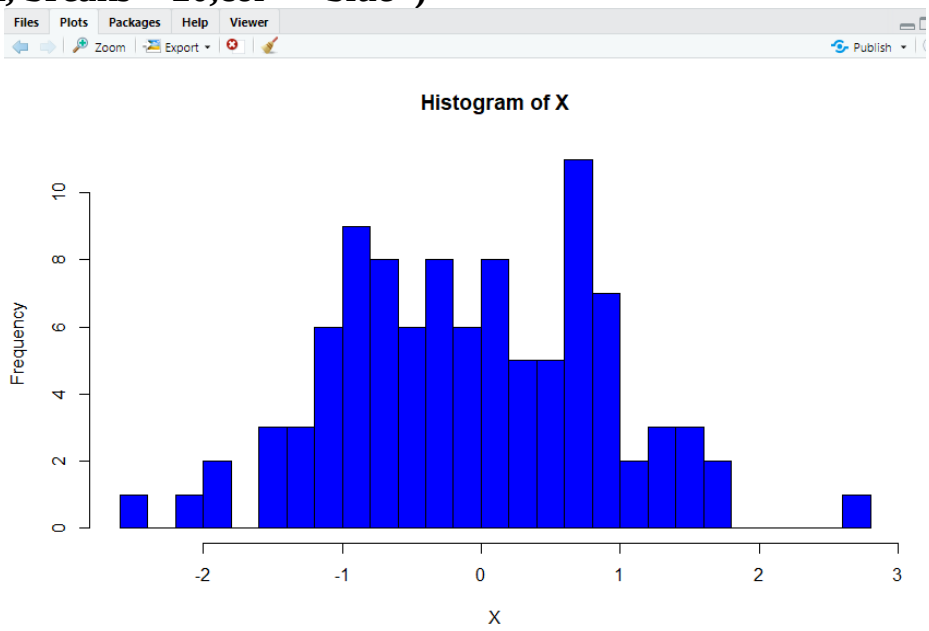


Рис. 15 Гистограмма совокупности со значениями аргументов breaks = 20, col = "blue"

Для отражения на оси ординат значение плотности вероятности для каждого интервала надо аргументу **probability** присвоить значение **TRUE**, тогда суммарная площадь под гистограммой составит 1:

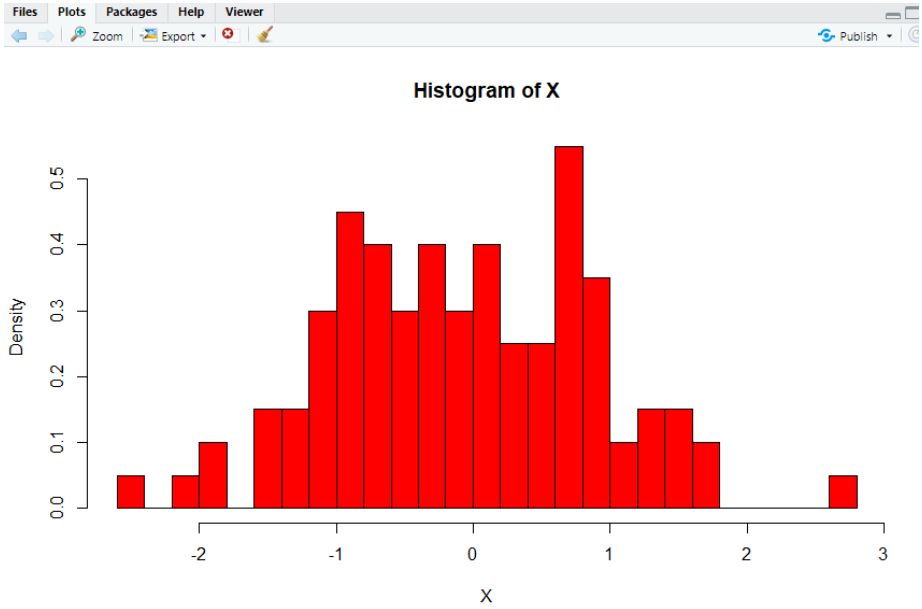


Рис. 16 Гистограмма с осью ординат «Density»

В ряде случаев, в частности при небольшом числе наблюдений, гистограммы могут давать неверное представление о свойствах совокупности, например, из-за небольшого числа редко расположенных столбцов. В таких случаях, одновременно с гистограммой, рекомендуется воспользоваться кривой плотности вероятности. Оценка плотности вероятности выполняется при помощи функции **density()**, которую можно применить в качестве аргумента функции **lines()** для графического изображения результата:

```
> X <- rnorm(50)# N(0, 1)
> hist(X, breaks = 20, probability =TRUE, col = "lightblue",
+      xlab = "Переменная X",
+      ylab = "Плотность вероятности",
+      main = "Гистограмма, совмещенная с кривой плотности")
> lines(density(X), col = "red", lwd = 3)
> lines(density(X, bw = 0.8), col = "blue", lwd = 2)
>
```

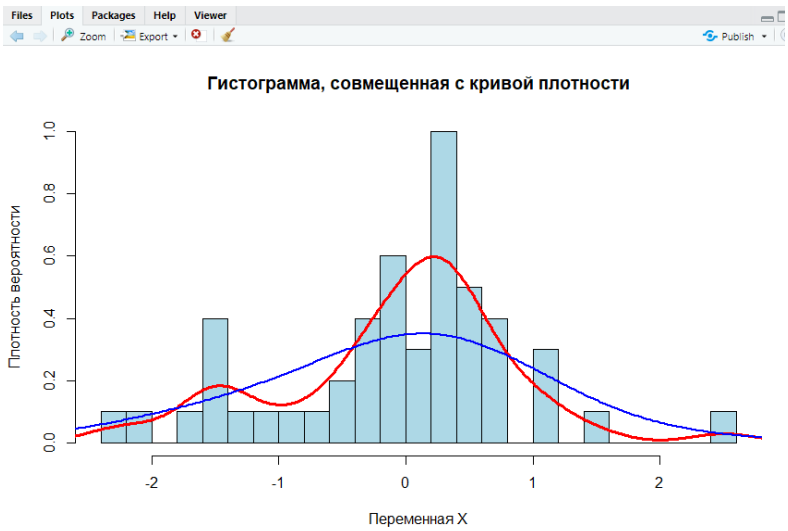


Рис. 17 Гистограмма и кривые плотности вероятности

Гладкость кривой плотности вероятности регулируется при помощи аргумента **bw** (от bandwidth – ширина окна) функции **density**.

9. Базисные методы статистической обработки данных

В этом разделе разберём числовые характеристики центральной тенденции, разброса и типа распределения значений непрерывных переменных на примере переменных для встроенного набора данных в R **mtcars** о характеристиках разных марок автомобилей []. Рассмотрим расход топлива (в милях на галлон – miles per gallon, **mpg**), мощность (лошадиных сил – horsepower, **hp**) и вес (weight, **wt**). Затем вычислим описательные статистики отдельно для каждого типа коробки передач (**am**) и числа цилиндров (**cyl**).

9.1. Вычисление описательных статистик

В базовой версии реализована функция **summary()**, которая позволяет вычислить некоторые описательные статистики. Вычислим их для расхода топлива, мощности и веса по следующему алгоритму:

```
> mars <- c("mpg", "hp", "wt")
> options(digits=2)
> summary(mtcars[mars])
      mpg             hp             wt
 Min.   :10   Min.   : 52   Min.   :1.5
 1st Qu.:15   1st Qu.: 96   1st Qu.:2.6
 Median :19   Median :123   Median :3.3
 Mean   :20   Mean   :147   Mean   :3.2
 3rd Qu.:23   3rd Qu.:180   3rd Qu.:3.6
 Max.   :34   Max.   :335   Max.   :5.4
```

Функция **summary()** позволяет вычислить минимум, максимум, квартили и среднее для числовых переменных. Для расчёта стандартного отклонения воспользуемся векторной функцией **sapply**:

```
> sapply(mtcars[mars], sd)
  mpg    hp    wt
 6.03 68.56  0.98
```

где **sd** – функция вычисления стандартного отклонения.

Для вычисления большего количества описательных статистик можно использовать функции входящие в пакеты **Hmisc**, **pastecs** и **psych**.

В пакете **pastecs** есть функция **stat.desc()**, которая позволяет вычислить множество описательных статистик. Формат ее применения таков: **stat.desc(x, basic=TRUE, desc=TRUE, norm=FALSE, p=0.95)** где x – это таблица данных или временной ряд. Если basic=TRUE (по умолчанию), то вычисляется число значений, число нулей и пропущенных значений, минимум, максимум, размах и сумма. Если desc=TRUE (тоже по умолчанию), то вычисляются медиана, среднее арифметическое, стандартная ошибка среднего, 95% доверительный интервал для среднего, дисперсия, стандартное отклонение и коэффициент

вариации. Наконец, если norm = TRUE (не по умолчанию),вычисляются статистики нормального распределения, включая асимметрию и эксцесс (и их достоверность), и проводится тест Шапиро-Уилка (Shapiro-Wilk test) на нормальность распределения. Опция p используется при вычислении доверительного интервала для среднего арифметического (по умолчанию она равна 0.95).

```
> library(pastecs)
> options(digits=2)
> stat.desc(mtcars[mars])
                mpg      hp     wt
nbr.val        32.0   32.00  32.00
nbr.null        0.0    0.00   0.00
nbr.na          0.0    0.00   0.00
min            10.4   52.00   1.51
max            33.9  335.00   5.42
range          23.5  283.00   3.91
sum           642.9 4694.00 102.95
median         19.2  123.00   3.33
mean           20.1  146.69   3.22
SE.mean         1.1   12.12   0.17
CI.mean.0.95    2.2   24.72   0.35
var            36.3 4700.87   0.96
std.dev         6.0   68.56   0.98
coef.var        0.3    0.47   0.30
```

>

Функция **describeBy()** из пакета **phych()** позволяет вычислить некоторые описательные статистики отдельно для каждой группы:

```
> library(psych)
> describeBy(mtcars[mars], mtcars$am)

 Descriptive statistics by group
group: 0
    vars  n   mean    sd median trimmed   mad   min    max  range  skew kurtosis    se
mpg    1 19  17.15  3.83  17.30   17.12  3.11 10.40  24.40  14.00  0.01    -0.80  0.88
hp     2 19 160.26 53.91 175.00  161.06 77.10 62.00 245.00 183.00 -0.01    -1.21 12.37
wt     3 19   3.77  0.78   3.52    3.75  0.45  2.46   5.42   2.96  0.98     0.14  0.18
------------------------------------------------------------------------------
group: 1
    vars  n   mean    sd median trimmed   mad   min    max  range skew kurtosis    se
mpg    1 13  24.39  6.17  22.80   24.38  6.67 15.00  33.90  18.90 0.05    -1.46  1.71
hp     2 13 126.85 84.06 109.00  114.73 63.75 52.00 335.00 283.00 1.36     0.56 23.31
wt     3 13   2.41  0.62   2.32    2.39  0.68  1.51   3.57   2.06 0.21    -1.17  0.17
```

>

9.2. Корреляции

Корреляция отражает меру степени и направления связи между значениями двух переменных. Количественным показателем связи служит

коэффициент корреляции (**r**), который принимает значения в диапазоне -1 до +1.

Ковариация выражает силу зависимости между двумя множествами данных. Если она отлична от нуля, то наборы данных зависимы.

Рассмотрим разные коэффициенты корреляции наряду с тестами на достоверность. Будем использовать набор данных **state.x77**, входящий в базовую версию R [].

Коэффициент корреляции Пирсона отражает степень линейной связи между двумя количественными переменными. Коэффициент ранговой корреляции Спирмена – мера связи между двумя ранжированными переменными.

Функция **cor()** позволяет вычислить эти два коэффициента, а функция **cov()** рассчитывает ковариации. У этих функций есть много опций. Упрощенный формат вычисления корреляций имеет следующий вид: **cor(**таблица данных**,** method=**"spearman"** (“**pearson"**) **)**. Пример программного кода:

```
> n<-1:6
> states<- state.x77[,n]
> cov(states)
             Population      Income   Illiteracy
Population 19931683.7588 571229.7796  292.8679592
Income       571229.7796 377573.3061 -163.7020408
Illiteracy      292.8680   -163.7020    0.3715306
Life Exp       -407.8425    280.6632   -0.4815122
Murder         5663.5237   -521.8943    1.5817755
HS Grad       -3551.5096   3076.7690   -3.2354694
              Life Exp      Murder      HS Grad
Population -407.8424612 5663.523714 -3551.509551
Income      280.6631837 -521.894286  3076.768980
Illiteracy   -0.4815122    1.581776    -3.235469
Life Exp      1.8020204   -3.869480     6.312685
Murder       -3.8694804   13.627465   -14.549616
HS Grad       6.3126849  -14.549616    65.237894

> cor(states,method = "pearson")
            Population     Income Illiteracy    Life Exp
Population  1.00000000  0.2082276  0.1076224 -0.06805195
Income      0.20822756  1.0000000 -0.4370752  0.34025534
Illiteracy  0.10762237 -0.4370752  1.0000000 -0.58847793
Life Exp   -0.06805195  0.3402553 -0.5884779  1.00000000
Murder      0.34364275 -0.2300776  0.7029752 -0.78084575
HS Grad    -0.09848975  0.6199323 -0.6571886  0.58221620
               Murder     HS Grad
Population  0.3436428 -0.09848975
Income     -0.2300776  0.61993232
Illiteracy  0.7029752 -0.65718861
Life Exp   -0.7808458  0.58221620
Murder      1.0000000 -0.48797102
HS Grad    -0.4879710  1.00000000

>
```

Частная корреляция – это корреляция между двумя количественными переменными, зависящими, в свою очередь, от одной или более других количественных переменных. Для вычисления коэффициентов частной корреляции можно использовать функцию **pcor()** из пакета **ggm**.

Формат применения этой функции:

pcor(u, S)

где **u** – это числовой вектор, в котором первые два числа – номера переменных, для которых нужно вычислить коэффициент, а остальные числа – номера «влияющих» переменных (воздействие которых должно быть отделено). **S** – это ковариационная матрица для всех этих переменных:

```
> library(ggm)
> pcor(c(1,5,2,3,6), cov(states))
[1] 0.3462724
```

Частный коэффициент корреляции между численностью населения и уровнем преступности без влияния дохода, доли неграмотного населения и долей людей со средним образованием равен 0.346.

Проверка значимости отдельных корреляционных коэффициентов Пирсона, Спирмена и Кэнделла выполняется с помощью функцию **cor.test()**. Формат её применения следующий:

cor.test(x, y, alternative = , method =)

где x и y – это переменные, корреляция между которыми исследуется, опция alternative определяет тип теста (“two.side”,“less” или “greater”), опция method задает тип корреляции (“pearson”, “kendall” или “spearman”). Опция alternative = ”less” используется для проверки гипотезы о том, что в генеральной совокупности коэффициент корреляции меньше нуля, а опция alternative = ”greater” – для проверки того, что он больше нуля. По умолчанию alternative = ”two.side” - не равен нулю). Пример программного кода:

```
> cor.test(states[,4], states[,5])

        Pearson's product-moment correlation

data:  states[, 4] and states[, 5]
t = -8.6596, df = 48, p-value = 2.26e-11
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.8700837 -0.6420442
sample estimates:
      cor
-0.7808458
```

Проверяется нулевая гипотеза о том, что коэффициент корреляции Пирсона между средней продолжительностью жизни и уровнем преступности равен нулю. Вероятность принятия нулевой гипотезы очень мала (p-value =

2.26e-11) и принимается альтернативная – о том, что значение этого коэффициента для генеральной совокупности равно: **-0.7808458**.

В пакете **psych** есть функция **corr.test()**, которая позволяет вычислить коэффициенты корреляции Пирсона, Спирмена и Кэнделла между несколькими переменными и оценить их достоверность.

9.3. Проверка на нормальность распределения

Проверка исследуемых переменных на нормальность распределения является важной составной частью разведочного анализа данных. Существуют графические и тестовые методы такой проверки.

Самый простой графический способ проверки характера распределения данных – это построение гистограммы. Если гистограмма имеет колоколообразный симметричный вид, можно сделать заключение о том, что анализируемая переменная имеет примерно нормальное распределение (рис.18). Однако при интерпретации гистограмм следует соблюдать осторожность, поскольку их внешний вид может сильно зависеть как от числа наблюдений, так и от шага, выбранного для разбиения данных на классы. Примерный программный код:

```
> #Построение гистограммы
> set.seed(1953)
> x1=rnorm(100,5,2)
> h=hist(x1,breaks=7,main = "Гистограмма с числом интервалов 7 и
+ кривой нормального распределения",col="blue",xlab="Независимая переменная X", ylab = "Частота")
> xfit=seq(min(x1), max(x1), length=15)
> yfit=dnorm(xfit, mean=mean(x1), sd=sd(x1))
> yfit=yfit*diff(h$mids[1:2])*length(x1)
> lines(xfit, yfit, col="red", lwd=2)
> box()
```

Очень часто используемым графическим способом проверки характера распределения данных является построение так называемых графиков квантилей (q-q plots, quantile-quantile plots). На таких графиках изображаются квантили двух распределений – эмпирического (т.е. построенного по анализируемым данным) и теоретически ожидаемого стандартного нормального распределения. При нормальном распределении проверяемой переменной точки на графике квантилей должны выстраиваться в прямую линию, исходящую под углом 45 градусов из левого нижнего угла графика.

Интерпретация квантильных графиков, получаемых с использованием распространенных функций **qqnorm()** и **qqplot()**, носит в значительной мере субъективный характер.

Рис. 18 Графическая проверка на нормальность распределения

Алгоритм получения геометрического места точек квантилей заложен в функции **qqPlot()** пакета **car**:

```
> library(car)
> qqPlot(x1, dist= "norm", col=palette()[1] , pch=19,
+       xlab="Квантили нормального распределения",
+       ylab="Наблюдаемые квантили",
+       main="Сравнение квантилей ЭР и НР")
```

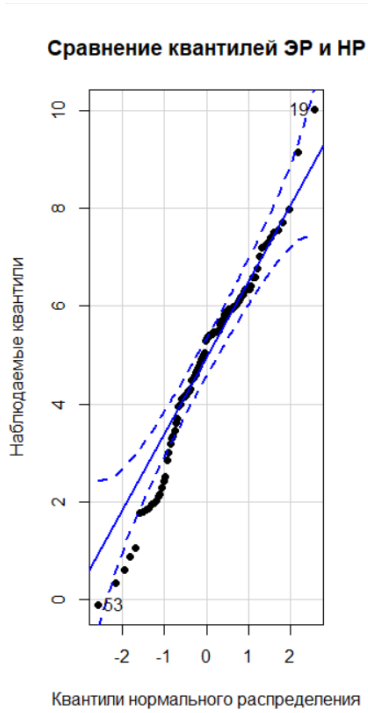


Рис. 19 Сравнение квантилей двух распределений: экспериментального и нормального

Альтернативой квантильным графикам **qqPlot()** является использование функции sm.density() из пакета sm. Если, например, указать для функции sm.density() параметр model = "Normal", то получим на графике такую же доверительную полосу, но относительно теоретической функции плотности распределения при выборочных значениях параметров. На этом же графике показывается кривая ядерной плотности для эмпирических данных и можно оценить насколько правдоподобно предположение о нормальности и в каких диапазонах анализируемой переменной наблюдаются отклонения:

```
> library(sm)
> sm.density(x1, model = "Normal", xlab="Имитированная выборка",
+          ylab="Функция плотности распределения")
```

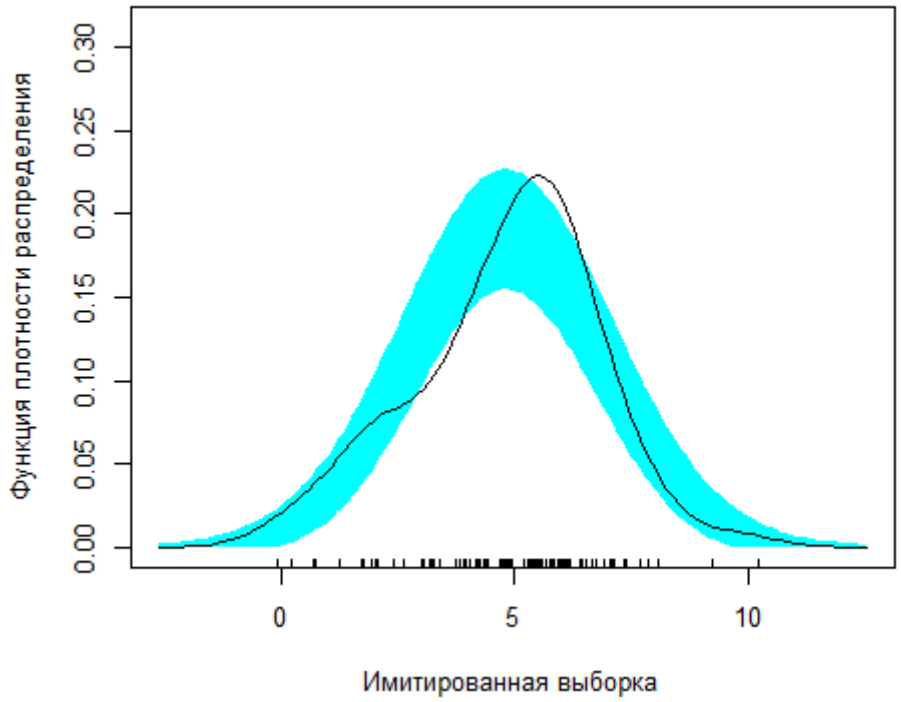


Рис. 20 Кривая ядерной плотности ЭР и доверительная полоса НР.

Существует целый ряд статистических тестов, специально разработанных для проверки нормальности распределения данных. В общем виде проверяемую при помощи этих тестов нулевую гипотезу можно сформулировать так: "анализируемая выборка происходит из генеральной совокупности, имеющей нормальное распределение". Если получаемая при помощи того или иного теста вероятность ошибки **p** оказывается меньше некоторого заранее принятого уровня значимости (например, 0.05), нулевая гипотеза отклоняется.

В R реализованы практически все имеющиеся тесты на нормальность – либо в виде стандартных функций, либо в виде функций, входящих в состав

подгружаемых пакетов. Примером базовой функции является **shapiro.test()**, при помощи которой можно выполнить широко используемый тест Шапиро-Уилка:

```
> shapiro.test(x1)
        Shapiro-Wilk normality test
data:  x1
W = 0.97319, p-value = 0.03899
```

В пакете **nortest** существуют дополнительные функции, реализующие другие распространенные тесты на нормальность:

- ad.test() - тест Андерсона-Дарлинга;
- cvm.test() - тест Крамера фон Мизеса;
- lillie.test() - тест Колмогорова-Смирнова в модификации Лиллиефорса;
- sf.test() - тест Шапиро-Франсия.

10. Регрессионное моделирование

Классическая практическая задача: определить наличие и вид зависимости в некоторой системе данных.

Предположим, что наблюдают значение пары переменных X и Y и необходимо найти зависимость между ними.

Переменная X называется независимой переменной или фактором (предиктором), переменная Y — зависимой переменной или результативным признаком (откликом).

Итак, есть фактор $X = \{X_1, X_2, \ldots, X_n\}$ и результативный признак $Y = \{Y_1, Y_2, \ldots, Y_n\}$. Нас интересует связь Y и X, которую можно представить и описать с помощью построения некоторой регрессионной модели.

Линейная регрессионная модель записывается следующим образом:

$$y_i = a + b \cdot x_i + e_i, \qquad (10.1)$$

где a, b — неизвестные константы, а e_i, i = 1, …, n случайные величины, характеризующие отклонения реального значения результативного признака от теоретического, найденного по уравнению регрессии.

Общая задача регрессионного моделирования на примере класса линейных регрессионных моделей состоит в том, чтобы по имеющимся наблюдениям $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$:

- оценить наилучшим образом параметры a и b;
- построить доверительные интервалы для a и b;
- проверить гипотезу о значимости регрессии;
- оценить адекватность модели.

10.1. Линейная регрессия

Рассмотрим простейшую модель парной регрессии – линейную регрессию. Линейная регрессия находит широкое применение в эконометрике ввиду четкой экономической интерпретации ее параметров.

Линейная регрессия сводится к нахождению уравнения вида

$$\hat{y}_i = a + b \cdot x_i \qquad (10.2)$$

Уравнение вида (10.2) позволяет по заданным значениям фактора x находить теоретические значения результативного признака, подставляя в него фактические значения фактора x.

Рассмотрим построение модели линейной регрессии на следующей задаче. Имеются данные по 20 туристическим фирмам: затраты на рекламу и количество туристов, воспользовавшихся услугами фирмы. Требуется построить регрессионную модель и дать прогноз количества туристов при затратах на рекламу, равных 15 000 у.е. Данные представлены таблицей 2.

Таблица 2 Статистика по турфирмам

№, п/п	Затраты на рекламу, тыс. у.е.	Кол-во туристов
	Reclama	Turist
1	8	800
2	8	850
3	8	720
4	9	850
5	9	800
6	9	880
7	9	950
8	9	820
9	10	900
10	10	1000
11	10	920
12	10	1060
13	10	950
14	11	900
15	11	1200
16	11	1150
17	11	1000
18	12	1200
19	12	1100
20	12	1000

Создаем таблицу в MS Excel. Сохраняем файл в формате «csv»-Doreg.csv. Считываем исходные данные и контролируем ввод:

```
> getwd()
[1] "D:/Rdata/Ecmetr"
> linreg <- read.table("Doreg.csv", header = TRUE,sep = ";")
> head(linreg)
  Reclama Turist
1       8    800
2       8    850
3       8    720
4       9    850
5       9    800
6       9    880
> tail(linreg)
   Reclama Turist
15      11   1200
16      11   1150
17      11   1000
18      12   1200
19      12   1100
```

```
20      12   1000
> str(linreg)
'data.frame':       20 obs. of  2 variables:
 $ Reclama: int  8 8 8 9 9 9 9 9 10 10 ...
 $ Turist : int  800 850 720 850 800 880 950 820 900 1000 ...
```

Строим диаграмму рассеяния введённых данных:

```
> plot(linreg,cex = 2, main = "Диаграмма рассеяния")
```

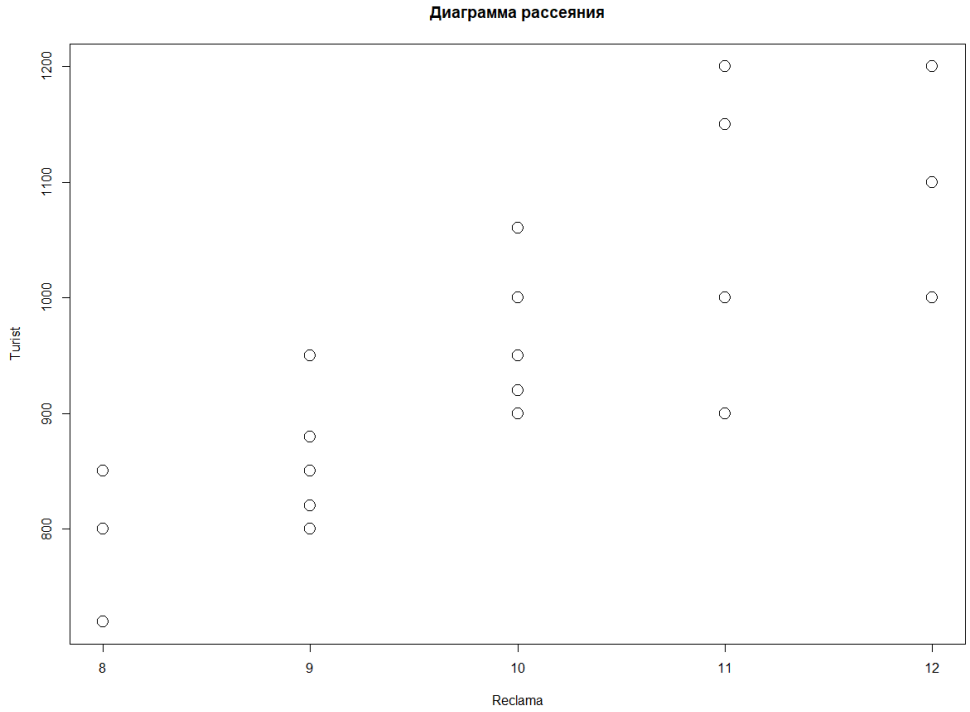


Рис .21 Диаграмма рассеяния в координатах «затраты на рекламу — количество туристов»

Визуальное изучение диаграммы рассеяния на рис. 21 показывает, что связь между затратами на рекламу и количеством обратившихся на фирму туристов есть. Дальнейший шаг состоит в выборе вида линии, которая будет описывать эту связь. В качестве такой линии возьмем прямую линию вида:

Turist = a + b × **Reclama**. (10.3)

В R основная функция для подгонки регрессионных моделей – это **lm()**. Формат ее применения **: myreg <- lm(formula, data)** где **formula** описывает вид модели, которую нужно подогнать, а **data** – это таблица с данными, которые используются для создания модели.

Выполняем подгонку модели и оцениваем её качество с помощью функции **summary()**:

```
> reg<- lm(Turist ~ Reclama,data = linreg)
> summary(reg)

Call:
lm(formula = Turist ~ Reclama, data = linreg)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-140.53  -53.81  -14.77   64.65  159.47

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   118.30     143.28   0.826     0.42
Reclama        83.84      14.28   5.870 1.47e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 81.98 on 18 degrees of freedom
Multiple R-squared:  0.6569,    Adjusted R-squared:  0.6378
F-statistic: 34.46 on 1 and 18 DF,  p-value: 1.471e-05
```

Из полученных результатов следует, что уравнение для предсказания количества туристов от затрат на рекламу имеет следующий вид:

Turist = 118.30 + 83.84 × **Reclama**. (10.4)

Из столбца Pr(>|t|) ясно, что коэффициент регрессии (83.84) статистически значимо отличается от нуля ($p < 0.001$) и означает, что на каждую тысячу условных единиц затрат на рекламу ожидается увеличение туристов на 83.84 единицы. Множественный коэффициент детерминации (0.6569) означает, что модель объясняет 65.69 % дисперсии значений количества туристов. Этот коэффициент представляет собой квадрат коэффициента корреляции между реальными и предсказанными значениями. Другими словами, туристический бизнес, согласно данной небольшой статистике, на 65,69% состоит из рекламы. Стандартную ошибку остатков (81.98) можно интерпретировать как усредненную ошибку предсказания числа туристов по затратам на рекламу с использованием данной модели. F-значение позволяет установить, являются ли предсказанные значения зависимой переменной только случайными.

Построим доверительные интервалы для коэффициентов регрессии с помощью функции **confint()** с уровнем значимости 0.05 (по умолчанию):

```
> confint(reg)
                2.5 %   97.5 %
(Intercept) -182.71621 419.3171
Reclama       53.83484 113.8435


>
```

Коэффициент регрессии ***a*** попадает в диапазон [-182.71621; 419.3171], т. е. может принимать как положительные так и отрицательные значения. Это означает, что нулевую гипотезу по оценке коэффициента ***a*** отбросить не удается – он статистически незначимо отличается от нулевого значения на уровне значимости 0,05, что подтверждается значением Pr(>|t|) = 0.42.

Коэффициент регрессии ***b*** попадает в интервал [53.83484;113.8435], поэтому нулевую гипотезу по оценке коэффициента можно отбросить на уровне значимости 0.05, что подтверждается значением Pr(>|t|) =1.47e-05.

Выполним дисперсионный анализ подогнанной модели:

```
> anova(reg)
Analysis of Variance Table

Response: Turist
          Df Sum Sq Mean Sq F value    Pr(>F)
Reclama    1 231606  231606  34.462 1.471e-05 ***
Residuals 18 120969    6721
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Объяснённая сумма квадратов остатков RSS = 231606, необъяснённая – ESS = 120969.

Построим график линейной регрессионной модели вместе с диаграммой рассеяния:

```
> plot(linreg,cex = 2, main = "Линейная регрессия")
> abline(reg, lwd = 2)
```

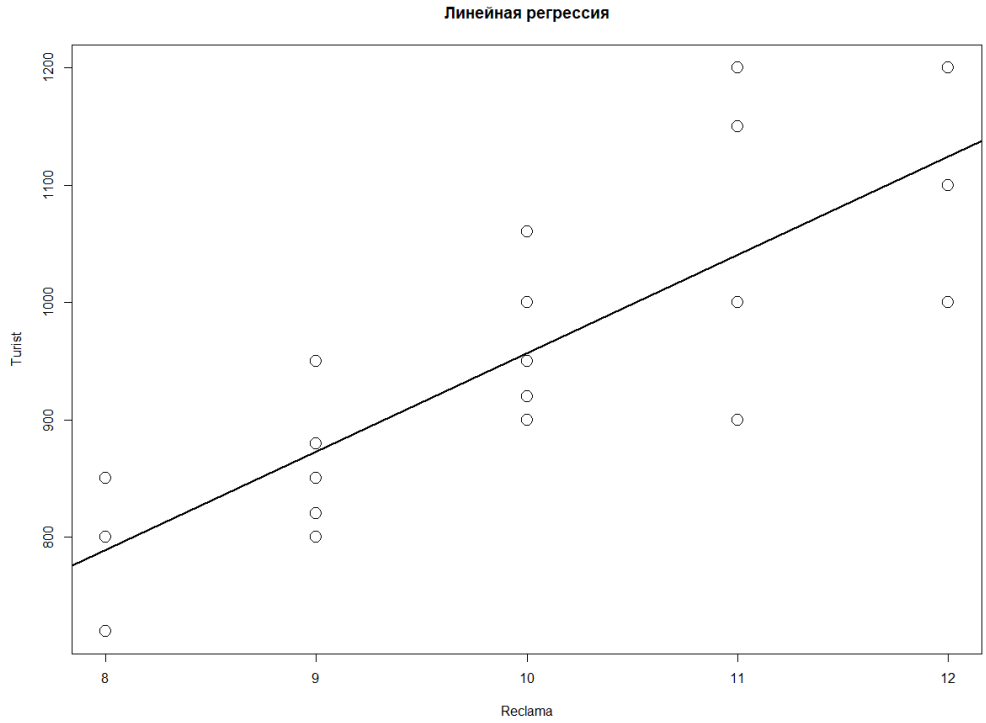


Рис. 22 Линейная регрессия для предсказаний количества туристов от затрат на рекламу

Для получения необходимого прогноза используем следующий программный код:

```
> nov<-data.frame(Reclama = 15)
```

```
> predict(reg,newdata = nov)
       1
1375.888
```

Ожидается около 1376 туристов при затратах на рекламу 15000 условных единиц. Построим доверительные интервалы для среднего "confidence" и индивидуального "prediction" прогнозных значений, выполняя следующий программный код:

```
> predict(reg,newdata = nov,interval = "confidence")
       fit      lwr      upr
1 1375.888 1219.548 1532.227
> predict(reg,newdata = nov,interval = "prediction")
       fit      lwr      upr
1 1375.888 1143.282 1608.494


>
```

10.2. Нелинейные модели парной регрессии

Нелинейная регрессия в целом характеризуется нелинейными соотношениями, которые выражаются с помощью соответствующих нелинейных функций.

Кривая зависимости безработицы в РФ от среднедушевых доходов имеет тенденцию к насыщению (табл.3), что отвечает исследованиям А.В. Филлипса. Регрессионная кривая Филлипса может быть записана в виде:

$$\hat{y}_i = a + b/x_i \qquad (10.5)$$

Равносторонняя гипербола $\hat{y} = a + b/x$ приводится к линейному уравнению простой заменой: $z = 1/x$.

Таблица 3 Данные по безработице и заработной плате работников в РФ

Year	2000	2001	2002	2003	2004	2005
Unemployed (тыс. чел.)	1037	1122	1500	1639	1920	1830
Salary (руб.)	2281	3062	3947	5170	6383	8112

Сначала посмотрим на график зависимости численности безработных от номинальной заработной платы работников:

```
> setwd("D:/Rdata/Ecmetr")
> nonlin <- read.table("fill.csv", header = TRUE,sep = ";")
> head(nonlin)
  Salary Unem
1   2281 1037
2   3062 1122
3   3947 1500
4   5170 1639
5   6383 1920
6   8112 1830
```

```
> str(nonlin)
'data.frame':   6 obs. of  2 variables:
 $ Salary: int  2281 3062 3947 5170 6383 8112
 $ Unem  : int  1037 1122 1500 1639 1920 1830
> plot(nonlin,type='b',cex = 2,lwd =2,ylab="Unemployed", main = "Кривая Филлипса")


>
```

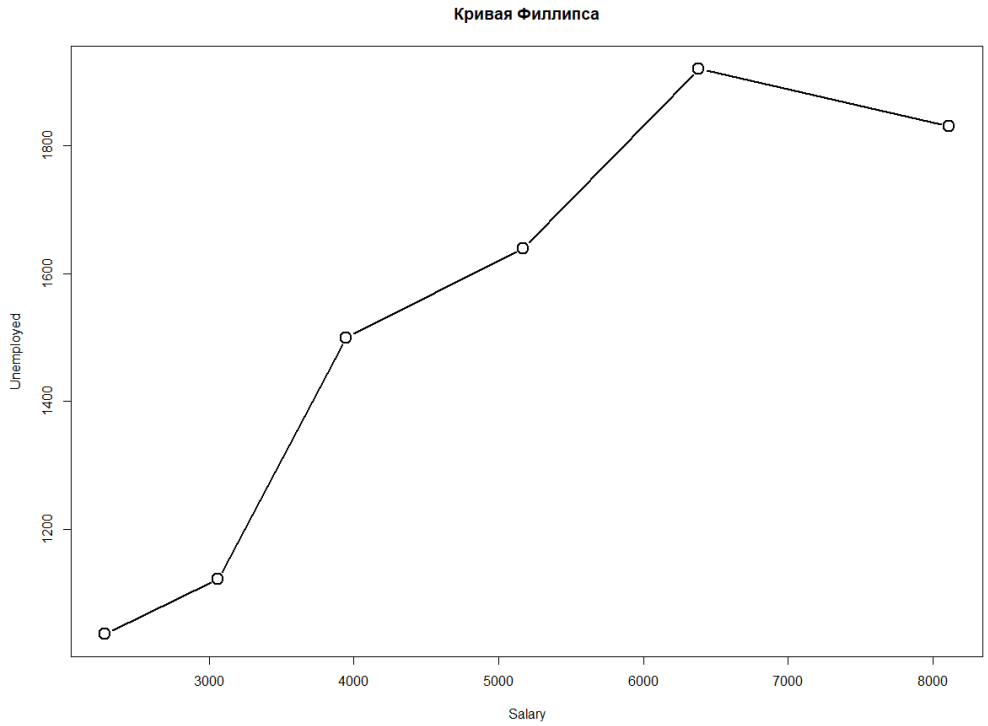


Рис. 23 График зависимости числа безработных в РФ от заработной платы работников

Найдем коэффициенты нелинейной регрессии a, b по данным о безработице и среднедушевых доходах в РФ согласно процедуре, опробованной в предыдущей задаче и оценим качество модели:

```
> z<-1/nonlin$Salary
> nongip<- lm(Unem ~ z,data = nonlin)
> summary(nongip)

Call:
lm(formula = Unem ~ z, data = nonlin)

Residuals:
       1        2        3        4        5        6
  90.145 -155.477    6.011  -32.195  140.123  -48.608

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)     2243        121  18.534 4.99e-05 ***
z           -2956725     446822  -6.617   0.0027 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 117.7 on 4 degrees of freedom
Multiple R-squared:  0.9163,   Adjusted R-squared:  0.8954
F-statistic: 43.79 on 1 and 4 DF,  p-value: 0.002704
```

```
> confint(nongip)
                 2.5 %       97.5 %
(Intercept)     1907.07     2579.121
z           -4197302.20 -1716147.847
> anova(nongip)
Analysis of Variance Table

Response: Unem
          Df Sum Sq Mean Sq F value   Pr(>F)
z          1 606121  606121  43.788 0.002704 **
Residuals  4  55369   13842
---
Signif. codes:  0 ‘***’ 0.
```

Оба коэффициента нелинейной регрессии высокозначимы, что подтверждают данные столбца **Pr(>|t|)**, в котором приведены вероятности того, что коэффициенты регрессии не отличаются от нуля.

Подставляя в (10.5) вычисленные значения коэффициентов *a* и *b*, получим рассчитанную нелинейную регрессионную модель:

Unemployed = 2243− 2956725/**Salary**. (10.6)

Качество построенной регрессионной модели определяем по коэффициенту множественной детерминации R^2 =0.9163, *F*-критерию Фишера *F* = *F*1,4 = 43.788, данному значению статистики отвечает минимальный уровень значимости, равный α_{min} = 0.002704, что значительно меньше α = 0,05. Всё это свидетельствует о общей значимости построенной регрессионной модели.

Построим совместный график статистических данных задачи и значений регрессионной модели:

```
> unemn<-2243 - 2956725/nonlin$Salary
> linnon<-data.frame(nonlin$Salary,unemn)
> plot(nonlin,type='b',cex = 2,lwd = 2,col = "green", ylab="Unemployed",
main = "Зависимость безработицы от заработной платы работников")
> par(new = TRUE)
> plot(linnon,type = 'b',xlab = "",ylab = "",axes = FALSE,
cex = 2,lwd =2,lty = 5,pch = 2,col = "red")
> legend("bottomright" ,title ="Кривая Филлипса",
c("Наблюдения", "Модель"),
+lty = c(1,5),pch  = c(1,2),col = c("green","red")  )
```

В следующей задаче рассмотрим взаимосвязь доли расходов на товары длительного пользования и общих сумм расходов (или доходов). Математическое описание такой взаимосвязи получило название кривых Энгеля. В 1857 г. немецкий статистик Э. Энгель на основе исследования семейных расходов сформулировал закономерность: с ростом доходов их доля, расходуемая на продовольствие, уменьшается. Соответственно, с увеличением дохода доля расходов на непродовольственные товары будет возрастать.

Обратимся к данным структуры потребительских расходов домашних хозяйств с сайта Федеральной службы государственной статистики (табл.4). В

таблице 4 строка **Food** обозначает расходы на покупку продуктов для домашнего питания (в %), строка **Thing** — расходы на покупку непродовольственных товаров (в %), а переменная **Salary** обозначает среднемесячную номинальную начисленную заработную плату работников организаций по всем видам экономической деятельности.

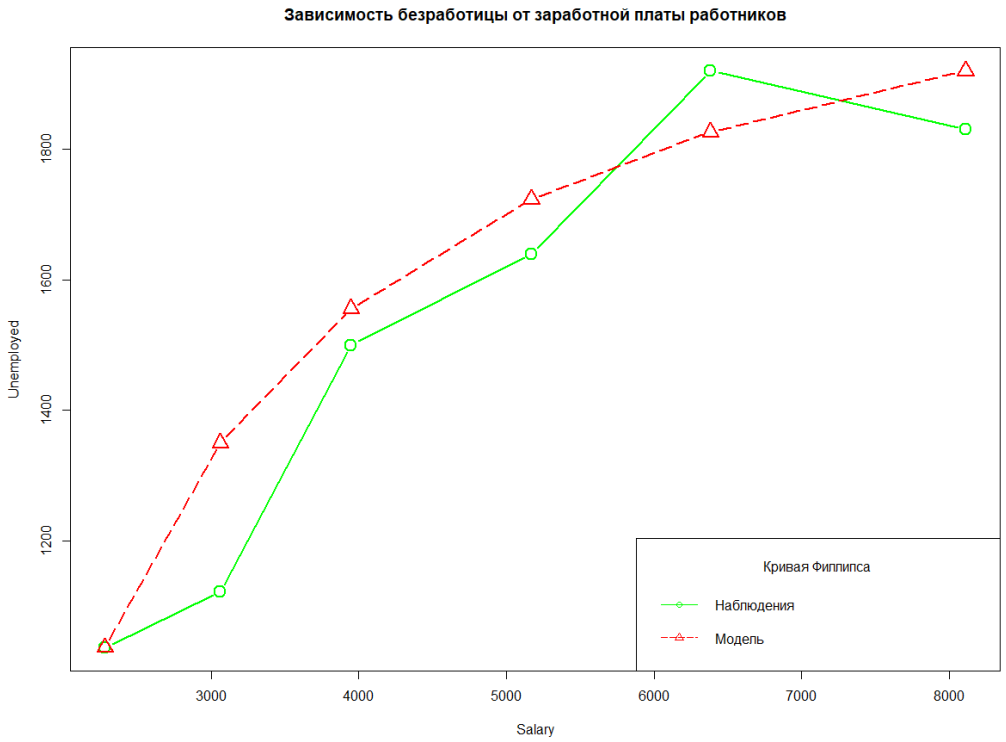


Рис. 24 График зависимости безработицы от заработной платы работников и нелинейная регрессионная кривая (10.6)

Таблица 4. Данные о потребительских расходах домашних хозяйств и заработной плате работников в РФ

Year	2000	2001	2002	2003	2004	2005
Food (%)	47.6	45.9	41.7	37.7	36.0	33.2
Thing (%)	34.3	34.4	36.2	37.3	37.2	38.5
Salary (руб.)	2281	3062	3947	5170	6383	8112

Для описания примера кривых Энгеля рассмотрим нелинейную регрессию следующего вида:

$$y_i = a + b \cdot \ln x_i, \qquad (10.7)$$

Выполним ввод и проверку статистических данных:

```
> setwd("D:/Rdata/Ecmetr")
> food <- read.csv("food.csv",sep = ";")
> thing <- read.csv("thing.csv",sep = ";")
> head(food)
```

```
  Salary Food
1   2281 47.6
2   3062 45.9
3   3947 41.7
4   5170 37.7
5   6383 36.0
6   8112 33.2
> str(food)
'data.frame':   6 obs. of  2 variables:
 $ Salary: int  2281 3062 3947 5170 6383 8112
 $ Food  : num  47.6 45.9 41.7 37.7 36 33.2
> head(thing)
  Salary Thing
1   2281  34.3
2   3062  34.4
3   3947  36.2
4   5170  37.3
5   6383  37.2
6   8112  38.5
> str(thing)
'data.frame':   6 obs. of  2 variables:
 $ Salary: int  2281 3062 3947 5170 6383 8112
 $ Thing : num  34.3 34.4 36.2 37.3 37.2 38.5


>
```

Реализуя следующий программный код:

```
> plot(food,type='o',ylab = "Food;Thing, (%)",cex = 2,lwd = 3,col = "red",
 main= "Кривые Энгеля")
> lines(thing$Salary,thing$Thing,type='o',cex = 2,lwd =3,pch = 2,col = "green")
> legend("topright" ,title ="Наблюдаемые данные", c("Food", "Thing"),
+        lty = c(1,1),pch  = c(1,2),col = c("red","green"))


>
```

сформируем совместный график статистических данных, используя функции **plot()** и **lines()** (рис. 25). Функция **lines()** добавляет информацию на существующую диаграмму, но не может сама создать график. Из-за этого функция **lines()** обычно применяется после того, как диаграмма уже создана посредством команды **plot()**.

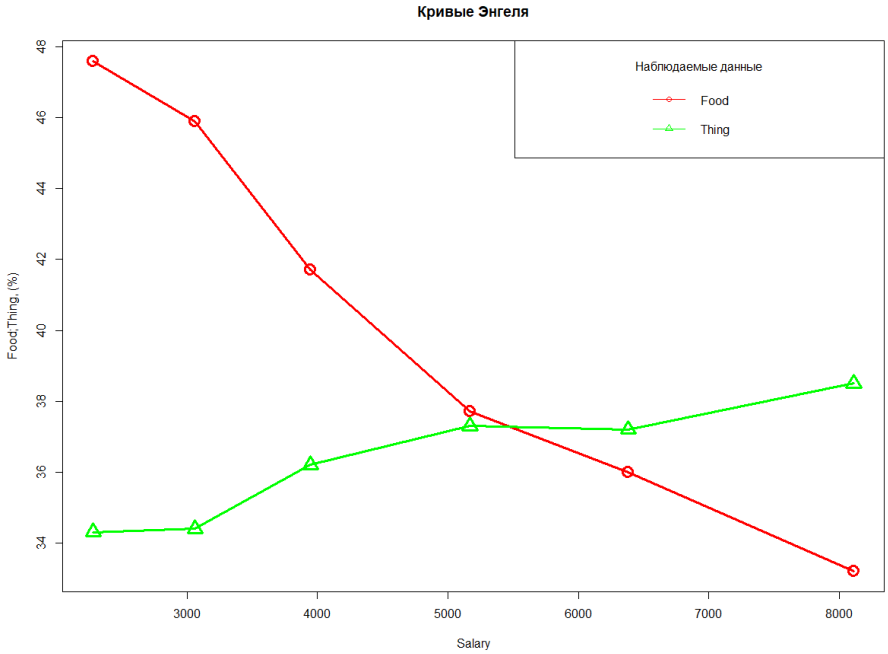


Рис. 25 Сопоставление наблюдаемых статистических данных

Найдем коэффициенты нелинейной регрессии **a**, **b** формулы (10.7) согласно процедуре, опробованной в предыдущей задаче, и оценим качество модели:

```
> lnfood<- lm(Food ~ log(Salary),data = food)
> summary(lnfood)

Call:
lm(formula = Food ~ log(Salary), data = food)

Residuals:
       1        2        3        4        5        6
-0.65574  1.17241  0.01449 -0.75137  0.07403  0.14619

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 140.9050     6.1636   22.86 2.17e-05 ***
log(Salary) -11.9820     0.7335  -16.34 8.22e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.774 on 4 degrees of freedom
Multiple R-squared:  0.9852,   Adjusted R-squared:  0.9815
F-statistic: 266.9 on 1 and 4 DF,  p-value: 8.219e-05
```

Получили следующую регрессионную кривую по данным, представленным в таблице 4:

Food = 140.9050− 11.9820·ln(**Salary**). (10.8)

Рассчитаем значения расходов на покупку продуктов домашнего питания по формуле (10.8) и сформируем таблицу :

```
> Foodmd<- 140.9050 - 11.9820*log(food$Salary)
> lnfood1<-data.frame(food$Salary,Foodmd)
```

Построим совместный график регрессионной зависимости переменной **Food** от переменной **Salary** и данных наблюдений (рис. 26):

```
> plot(food,type='o',ylab = "Food, (%)",cex = 2,lwd = 3,col = "red", main=
"Кривые Энгеля")
> lines(lnfood1$food.Salary,lnfood1$Food,type='o',cex = 2,lwd =3,pch = 1,col =
"green")
> legend("topright" , c("Наблюдения", "Модель"),
+        lty = c(1,1),pch  = c(1,1),col = c("red","green"))
```

Степень уверенности в параметрах регрессии зависит от того, насколько модель удовлетворяет ограничениям МНК. Хотя функция **summary()** описывает модель, она ничего не сообщает о том, насколько модель соответствует лежащим в ее основе МНК статистическим ограничениям – условиям Гаусса - Маркова.

В пакете **car** реализован ряд функций, которые значительно расширяют возможности оценки регрессионных моделей и допущений МНК – регрессии:

— нормальность;

— независимость;

— линейность;

— гомоскедастичность.

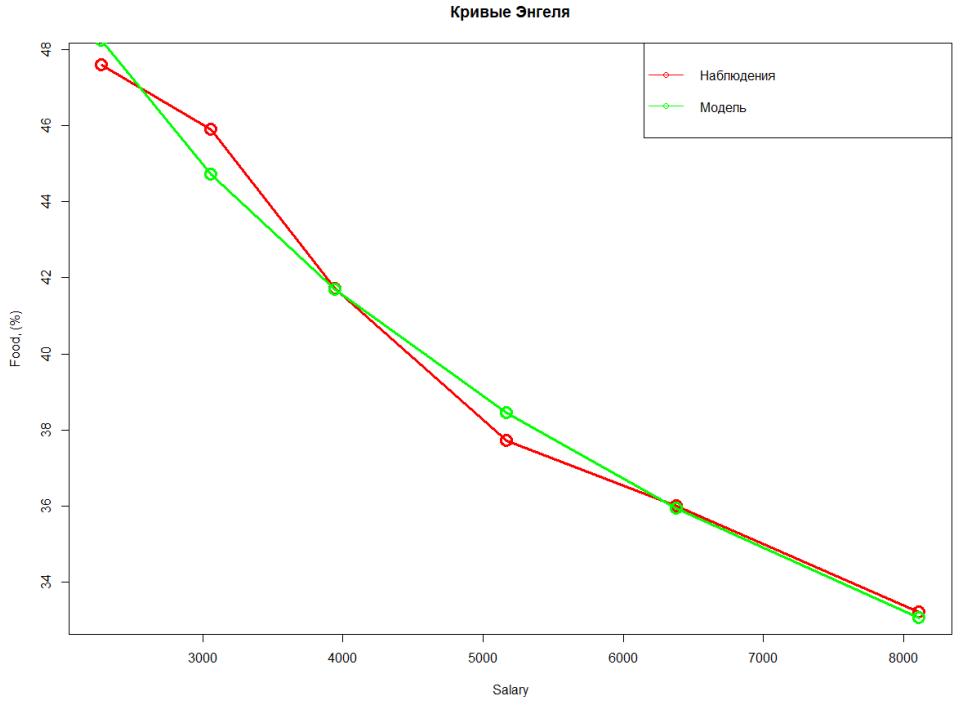


Рис. 26 Кривые Энгеля зависимости доли затрат на продовольствие от среднемесячной зарплаты работников

С помощью функции qqPlot()выполним проверку предположения о нормальности. Она изображает связь между остатками Стьюдента и квантилями распределения Стьюдента с n – p – 1 степенями свободы, где n – это объем выборки, а p – число параметров регрессии (включая свободный член).

Программный код:

```
> qqPlot(lnfood, cex = 2,labels=row.names(states), id.method="identify",
+         simulate=TRUE, main="Нормальная Q-Q диаграмма",
ylab = "Остатки Стьюдента",
+         xlab = "Квантили распределения Стьюдента")
[1] 1 2


>
```

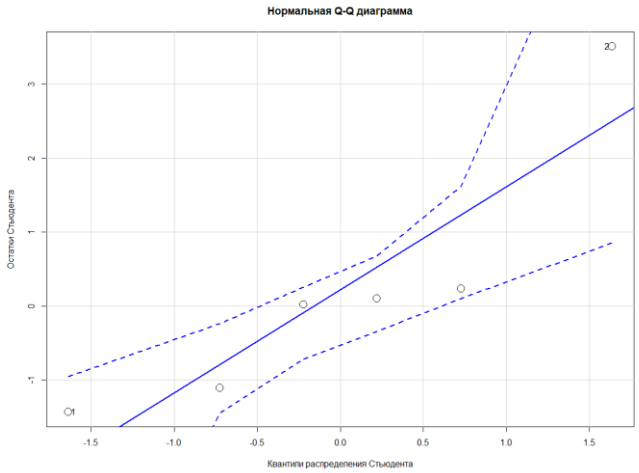


Рис. 27 Графическая проверка нормальности распределения (Q-Q-диаграмма)

Допущение о нормальном распределении остатков выполняется, точки на этой диаграмме ложатся на прямую с углом наклона в 45° с учетом 95% - ных доверительных границ.

Независимость остатков проверим с помощью теста Дарбина-Уотсона:

```
> durbinWatsonTest(lnfood)
 lag Autocorrelation D-W Statistic p-value
   1      -0.3369978       2.485623   0.922
 Alternative hypothesis: rho != 0
```

Высокое значение вероятности статистической ошибки первого рода (p = 0.922) свидетельствует об отсутствии автокорреляции и, следовательно, о независимости остатков.

Наличие нелинейной связи между зависимой (Food) и независимой (ln(Salary)) переменными можно проверить при помощи диаграмм компонент и остатков. Диаграммы создаются при помощи функции **crPlots()**:

```
> crPlots(lnfood)
```

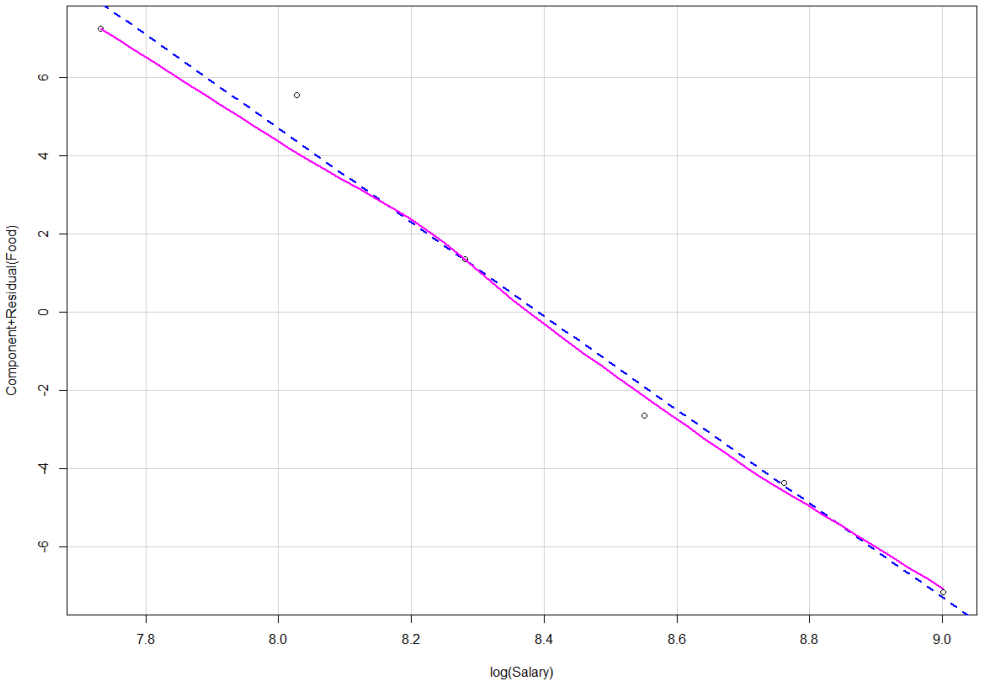


Рис. 28 Диаграмма компонент и остатков для регрессии
Food = 140.9050− 11.9820·ln(**Salary**)

Диаграмма компонент и остатков подтверждает, что требование линейности соблюдено. Вид линейной модели подходит для данного набора данных.

Функцией **ncvTest()** проверяем гипотезу о постоянстве дисперсии остатков. Статистически значимый результат свидетельствует о гетероскедастичности (неоднородности дисперсии остатков):

```
> ncvTest(lnfood)
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 1.307396, Df = 1, p = 0.25287
```

Результат теста незначим (p = 0.25287), что свидетельствует о выполнении условия однородности дисперсии. Это также можно увидеть на диаграмме (рис. 29), которая строится функцией **spreadLevelPlot()**. Точки беспорядочно располагаются в виде кривой вдоль регрессионной прямой.

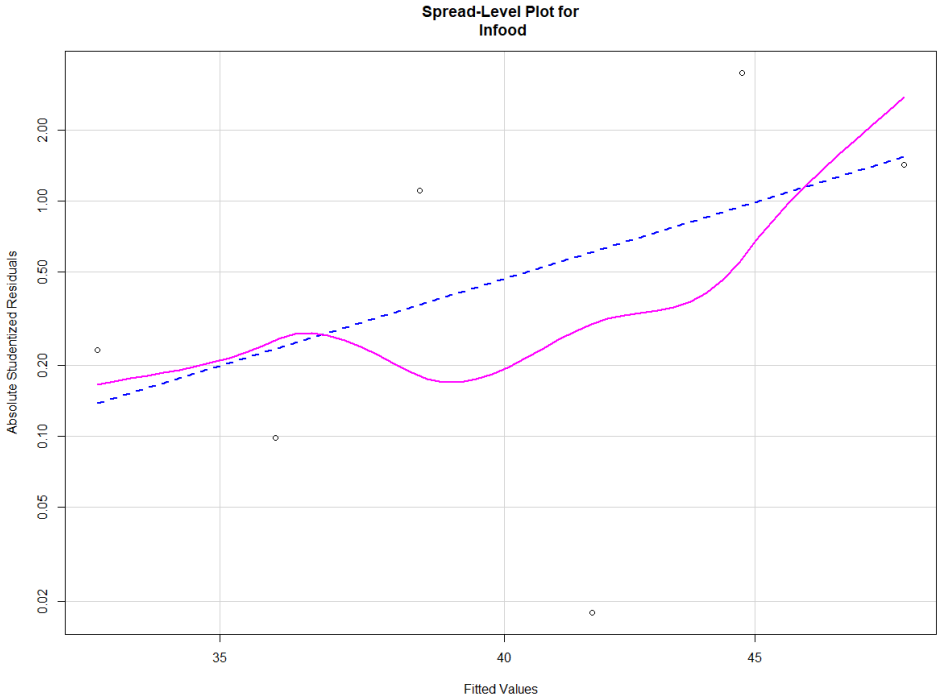


Рис. 29 Диаграмма для проверки однородности дисперсии остатков

Осуществим общую проверку выполнения требований, предъявляемых к линейным моделям наряду с отдельной оценкой асимметрии, эксцесса и гомоскедастичности с помощью функции **gvlma()** из пакета **gvlma**:

```
> library(gvlma)
> gvmodel <- gvlma(lnfood)
> summary(gvmodel)

Call:
lm(formula = Food ~ log(Salary), data = food)

Residuals:
       1        2        3        4        5        6
-0.65574  1.17241  0.01449 -0.75137  0.07403  0.14619

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 140.9050     6.1636   22.86 2.17e-05 ***
log(Salary) -11.9820     0.7335  -16.34 8.22e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.774 on 4 degrees of freedom
Multiple R-squared:  0.9852,   Adjusted R-squared:  0.9815
F-statistic: 266.9 on 1 and 4 DF,  p-value: 8.219e-05


ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = lnfood)
```

```
                     Value p-value                Decision
Global Stat         1.9325  0.7482 Assumptions acceptable.
Skewness            0.3603  0.5484 Assumptions acceptable.
Kurtosis            0.0622  0.8031 Assumptions acceptable.
Link Function       0.1126  0.7372 Assumptions acceptable.
Heteroscedasticity 1.3975  0.2371 Assumptions acceptable.
```

>

Из результатов теста (строка Global Stat) следует, что данные удовлетворяют всем статистическим допущениям, лежащим в основе МНК-регрессии (p = 0.7482). Общая проверка не противоречит результатам пошаговой проверки выполнений условий Гаусса – Маркова.

Заключение

В этом учебно – методическом пособии рассмотрены самые разнообразные темы, включая основные, такие как устройство среды R и интегрированной среды разработки (IDE) RStudio, создание данных, базисные статистические методы анализа и визуализации данных. Приведены также некоторые традиционные статистические модели и статистические графики. Показано замечательное свойство R – в этой программе всегда есть чему научиться. R – это обширная, мощная и постоянно развивающаяся статистическая среда и язык программирования, которые с успехом могут быть использованы при изучении таких дисциплин, как «Математическое и имитационное моделирование», «Программное обеспечение обработки данных» и «Эконометрика».

В пособии рассмотрены конструкции самого языка R, которые были отобраны в результате длительной практической обработки большого количества разных наборов данных и необходимость в которых выявляется в первую очередь. Продемонстрированы также примеры построения регрессионных моделей и оценки их качества.

Настоящее учебно – методическое пособие является методической поддержкой указанных выше курсов. Оно будет полезно преподавателям и студентам высших учебных заведений. Правильная организация своей образовательной траектории поможет студентам получить надежную теоретическую и практическую основу для дальнейшего овладения и использования современных методов анализа больших массивов данных по выбранному направлению «Прикладная информатика».

Список литературы

1. Кабаков Р.И. R в действии. Анализ и визуализация данных в программе R / Кабаков Р.И. - М.:ДМК Пресс, 2014. - 588 с.
2. Мастицкий С.Э. Статистический анализ и визуализация данных с помощью R / Мастицкий С. Э., Шитиков В. К. – М.: ДМК Пресс, 2015. – 401 с.
3. Джеймс Г. Введение в статистическое обучение с примерами на языке R / Джеймс Г., Уиттон Д., Хасти Т., Тибширани Р. - М.: ДМК Пресс, 2017. – 456 с.
4. Зарядов И. С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика / Зарядов И. С. – М.: Издательство Российского университета дружбы народов, 2010. – 207 с.
5. Зорин А. В. Введение в прикладной статистический анализ в пакете R / Зорин А. В., Федоткин М. А. – Нижний Новгород: ННГУ, 2010. – 50 с.

Владимир Анатольевич **Гришин**
Михаил Семенович **Тихов**

МЕТОДЫ ОБРАБОТКИ ДАННЫХ И МОДЕЛИРОВАНИЕ НА ЯЗЫКЕ R

Учебно – методическое пособие

Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Нижегородский государственный
университет им. Н.И. Лобачевского»

603950, Нижний Новгород, пр. Гагарина, 23