

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Пензенский государственный университет» (ПГУ)

---

С. В. Рындина

БАЗОВЫЕ ВОЗМОЖНОСТИ  
ЯЗЫКА R  
ДЛЯ АНАЛИЗА ДАННЫХ

Учебно-методическое пособие

Пенза  
Издательство ПГУ  
2021

УДК 65.29я7Р  
Р95

Рецензент  
кандидат технических наук, доцент  
*А. А. Масленников*

**Рындина, Светлана Валентиновна.**

Р95      Базовые возможности языка R для анализа данных / С. В. Рындина. – Пенза : Изд-во ПГУ, 2021. – 56 с.

Рассмотрены возможности языка программирования R для анализа данных. Приведены задания для лабораторных работ по разведочному, описательному и исследовательскому анализу данных. Материалы соответствуют программе дисциплины «Анализ данных», могут быть использованы при изучении дисциплин «Бизнес-аналитика на основе больших данных», «Интеллектуальные информационные системы» и при написании выпускной работы бакалавра.

Издание подготовлено на кафедре «Цифровая экономика» ПГУ и предназначено для обучающихся по направлению подготовки 38.03.05 «Бизнес-информатика».

**УДК 65.29я7Р**

© Пензенский государственный  
университет, 2021

## Содержание

Введение .....	4
1. Среда разработки для языка R: RStudio, RStudio.cloud.....	5
2. Данные .....	18
2.1. Типы данных и структуры, которые поддерживает R .....	18
2.2. Импорт данных. Построение срезов .....	24
3. Описательный анализ данных.....	34
4. Разведочный анализ данных .....	39
5. Проверка статистических гипотез .....	47
5.1. Тестирование гипотез для одной выборки .....	49
5.2. Тестирование гипотез для двух выборок.....	50
6. Лабораторные работы .....	52
6.1. Лабораторная работа.....	52
«Импорт и преобразование данных» .....	52
6.2. Лабораторная работа «Описательный анализ данных» .....	52
6.3. Лабораторная работа «Визуализация данных. Разведочный анализ данных» .....	53
6.4. Лабораторная работа «Исследовательский анализ данных» .....	54
Список литературы .....	55

## **Введение**

В соответствии с учебным планом обучающиеся второго курса направления подготовки 38.03.05 «Бизнес-информатика» изучают дисциплину «Анализ данных».

Описательный анализ данных, разведочный анализ данных и проверка статистических гипотез – это базовые методы работы с данными.

Целью учебно-методического пособия является рассмотрение основных типов данных и аналитических приемов работы с ними: преобразование, очистка, изучение распределений данных, тестирование гипотез для ответов на исследовательские вопросы о контексте представления данных, полезные визуализации для получения инсайтов, формулирование исследовательских вопросов, формирование отчетов и презентации результатов анализа.

В пособии рассмотрен язык программирования R, облачный ресурс RStudio.cloud и клиентское приложение RStudio Desktop для написания кода по анализу данных на этом языке программирования.

Приведены задания лабораторных работ по анализу данных, в которых используются представленные методы анализа и инструментальные средства для выполнения анализа данных.

# 1. Среда разработки для языка R: RStudio, RStudio.cloud

Для выполнения анализа данных с помощью языка программирования R, необходимо настроить и установить R и RStudio.

R – это открытый кросс-платформенный язык статистического программирования. На этом языке существует большое число пакетов, которые позволяют реализовать как классические статистические методы работы с данными, так и более современные.

Установочные файлы для языка R под различные операционные системы (Windows, Linux, MacOS) можно найти на официальном сайте проекта R: <https://cran.r-project.org/>. На момент написания пособия актуальной для Windows является версия R-4.1.0.

Среда разработки (integrated development environment, IDE) для языка R – RStudio. Существует две версии RStudio Desktop и RStudio Server. Обе версии существуют в варианте open source edition и commercial license. В пособии используется RStudio Desktop open source edition. Установочные файлы можно найти на официальном сайте продукта <https://rstudio.com/products/rstudio/>.

Начальное окно RStudio представлено на рис. 1.1:

1. Source editor and data view (редактор кода и обзор данных).
2. R console (консоль).
3. Служебная панель 1: Environment, history and connections окружение (данные, функции, история команд и доступные внешние ресурсы).
4. Служебная панель 2: Files, plots, packages, help and viewer (файлы, графики, пакеты, помощь и просмотрщик).

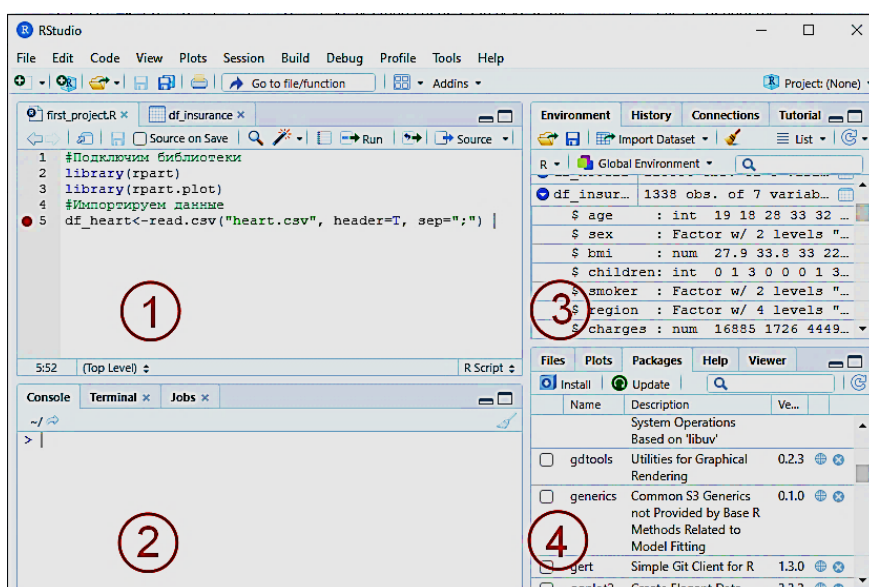



Рис. 1.1. IDE RStudio

Код предпочтительнее набирать в панели Source, открыв новый файл с помощью меню File > New file > R Script. При сохранении скрипта ему необходимо добавить расширение .R. На выполнение код посылается комбинацией клавиш CTRL + ENTER или кнопкой в правом верхнем углу панели  Run.

Для изменения настроек RStudio откроем диалоговое окно с помощью меню Tools > Global Options (рис. 1.2).

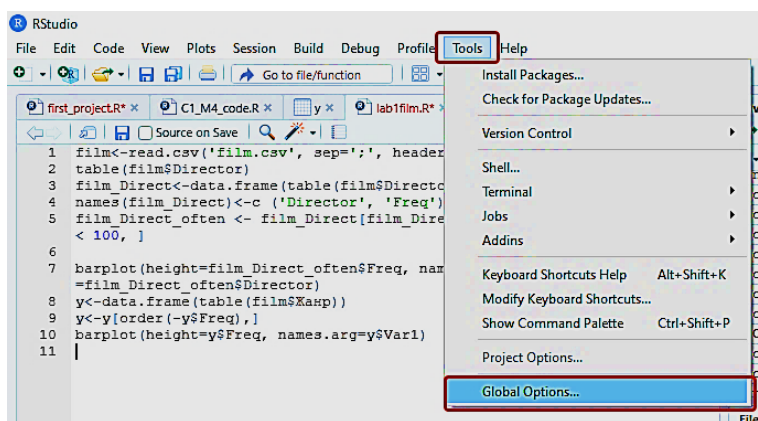


Рис. 1.2. Изменение настроек RStudio

Можно увидеть общие настройки системы (рис. 1.3), в частности, какая версия языка R используется RStudio.

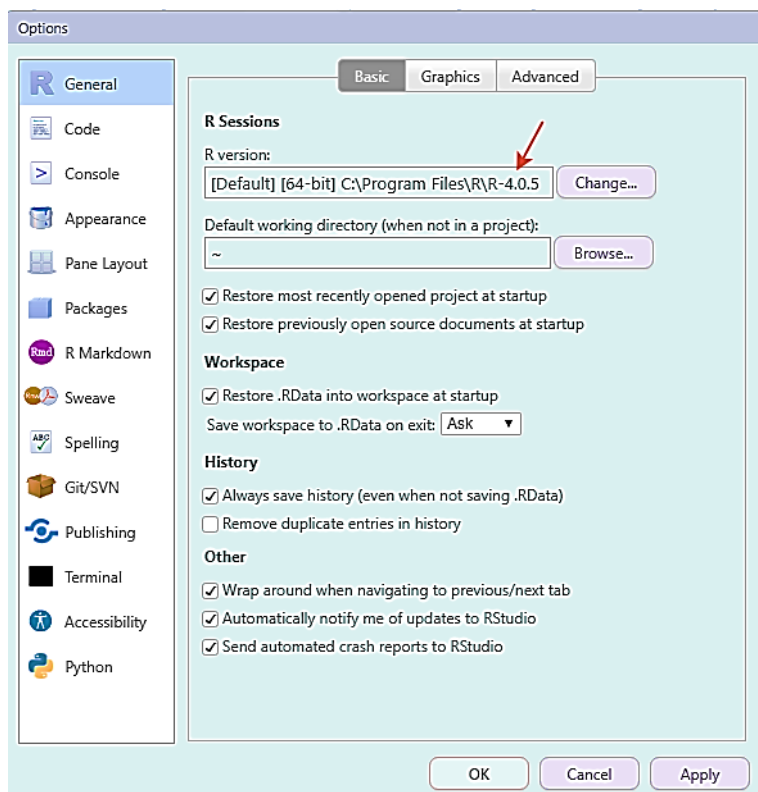


Рис. 1.3. Общие настройки

Перейдем на вкладку Code с помощью левой панели навигации в окне настроек RStudio и установим галочку на опции Soft-wrap R source files (рис. 1.4), которая отвечает за перенос длинных строк в коде (рис. 1.5), что очень удобно, так как в окне Source всегда виден набранный код, без необходимости перемещать ползунок для его просмотра (рис. 1.6).

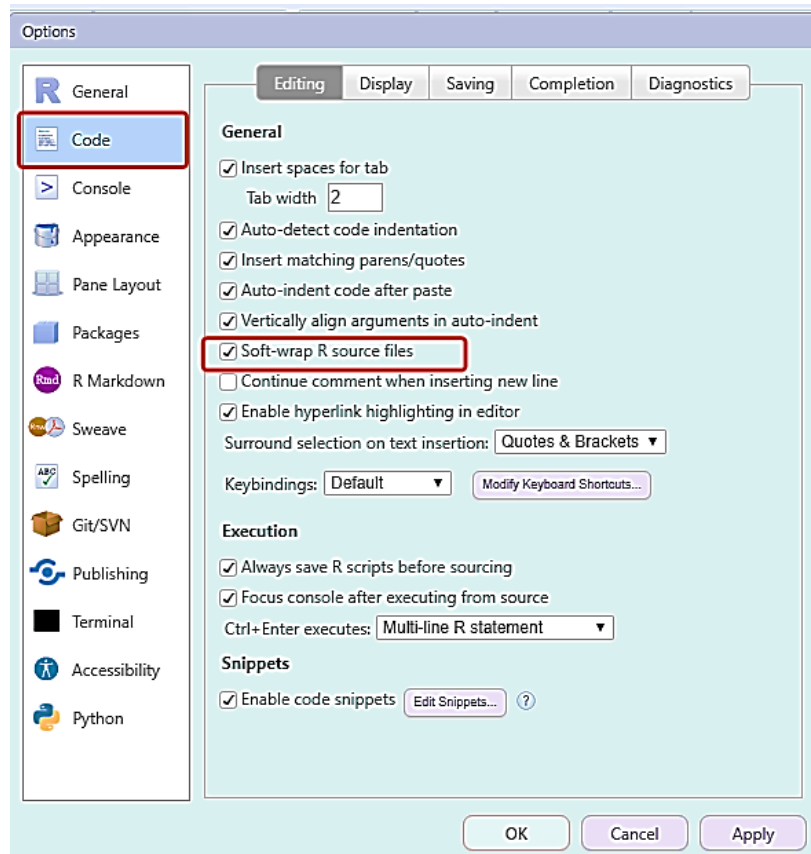


Рис. 1.4. Настройки редактора кода

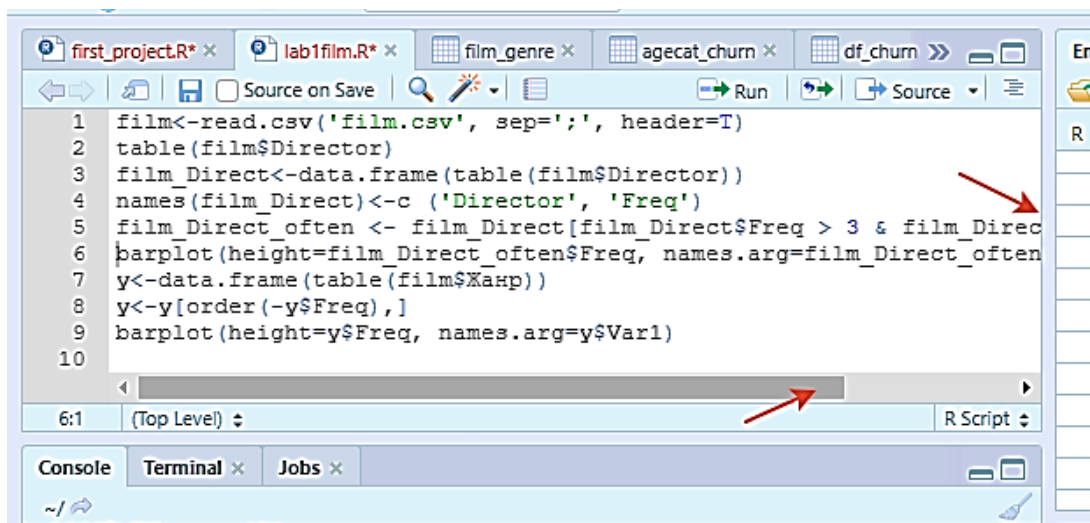


Рис. 1.5. Редактор кода без опции переноса строк

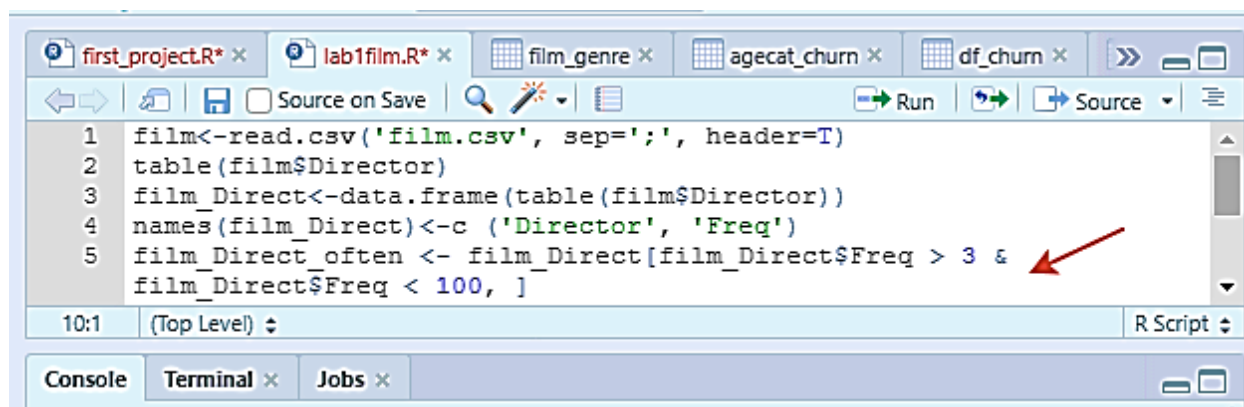


Рис. 1.6. Редактор кода с опцией переноса строк

Можно изменить также настройки внешнего вида IDE RStudio, выбрав вкладку Appearance (рис. 1.7).

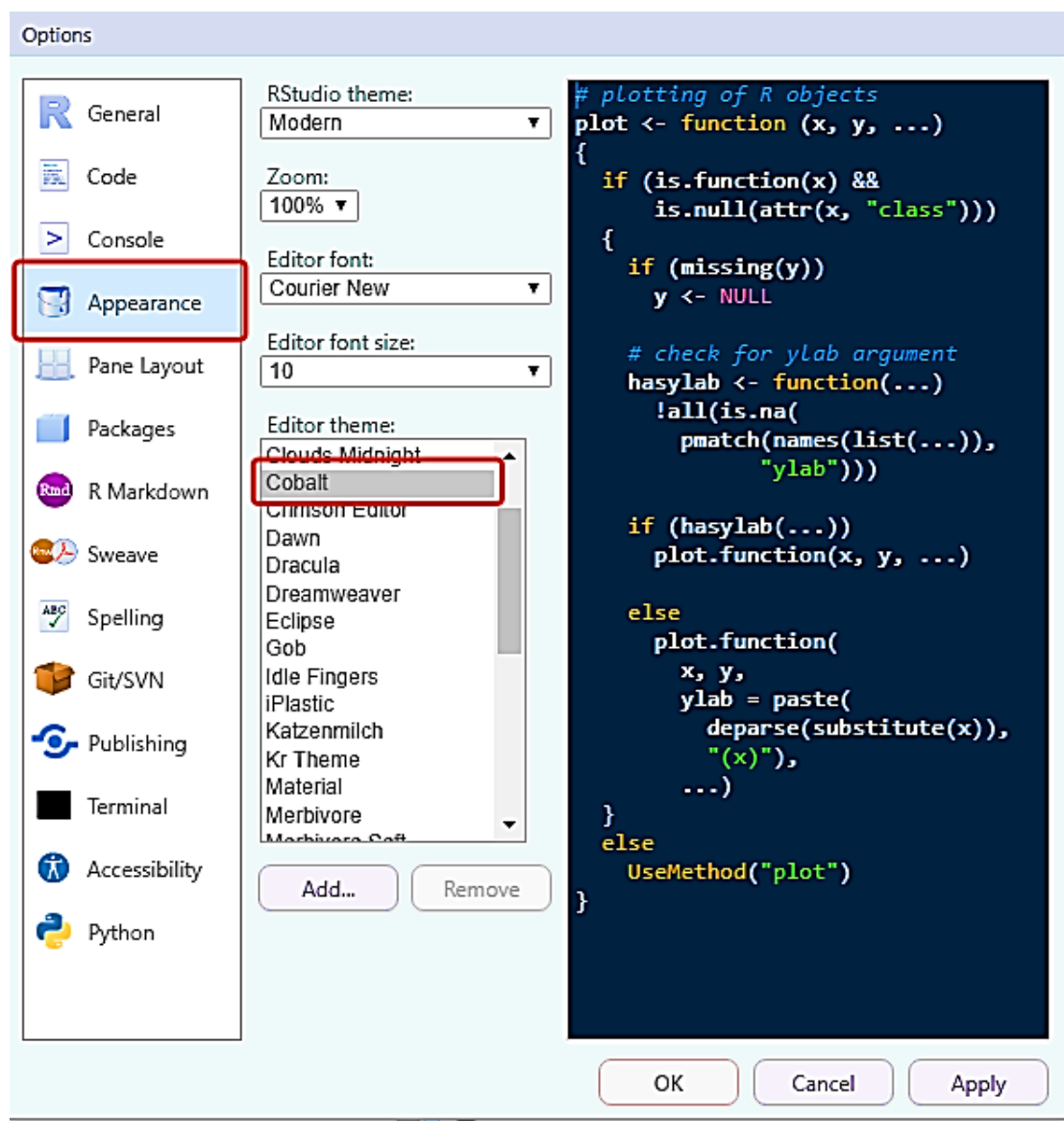


Рис. 1.7. Настройка темы для внешнего вида RStudio



В правом нижнем квадранте (под № 4 на рис. 1.1) есть вкладка **Files**, в которой отображаются файлы рабочей директории.

Команда `getwd()` позволяет узнать текущую рабочую директорию (`get working directory`).

К файлам, которые находятся в рабочей директории, можно обращаться, указывая относительный путь, т.е. только имя файла. Абсолютная адресация выстраивается от корневого каталога.

Так относительный путь к файлу данных в рабочей директории имеет вид: `insurance.csv`, тогда как полный путь к этому файлу: `C:/Users/Svetlana/Documents/insurance.csv` (на компьютере автора пособия).

Рабочую директорию можно сменить с помощью команды `setwd()`, указав в качестве аргумента полный путь к новой папке:

```
setwd("C:/Users/Svetlana/Documents/Working_R")
```

Смена рабочей директории возможна и с помощью командного меню **Session > Set working Directory > Choose Directory...**

При закрытии RStudio программа предлагает три опции на выбор (рис. 1.8):

- сохранить рабочее пространство и выйти;
- не сохранять свое рабочее пространство и все равно выйти;
- выполнить отмену, вернувшись в командную строку, вместо того, чтобы выйти.

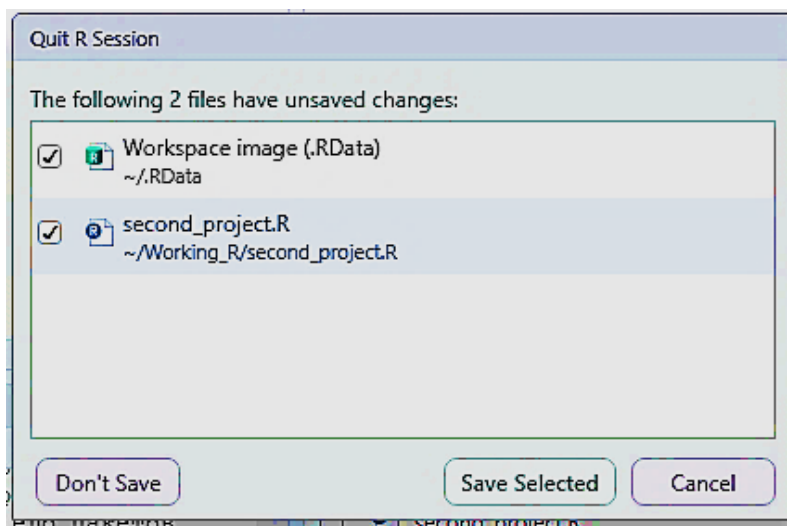


Рис. 1.8. Диалоговое окно выхода из RStudio

Если выбрана опция сохранения рабочего пространства, то оно записывается в файл с именем `.RData` в текущем рабочем каталоге, так же сохраняются любые объекты R, которые были созданы в этой сессии или изменены, но не сохранены. Если рабочая директория с предыдущей

сессии не менялась, то файл `.RData`, сохранивший рабочее пространство предыдущей сессии, будет перезаписан. Если в предыдущей сессии были важные изменения, которые потерялись в ходе текущей, то рабочее пространство лучше не сохранять или сменить директорию.

При открытии RStudio загружается сохраненное рабочее пространство со всеми незакрытыми в прошлой сохраненной сессии файлами, созданными переменными, функциями и историей команд (рис. 1.9).

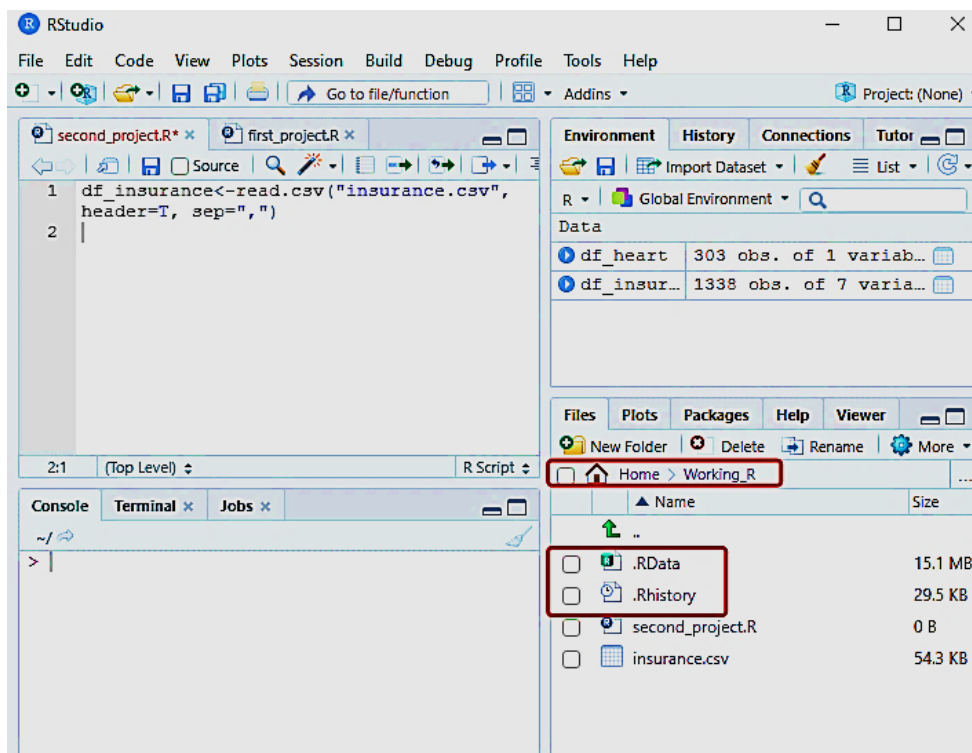


Рис. 1.9. Рабочая директория и сохраненное рабочее пространство

Если необходимо прервать выполнение кода в консоли, но не закрывать RStudio и сохранить в памяти все переменные, то это можно сделать с помощью командного меню `Session > Interrupt R`, можно нажать клавишу `ESC` на клавиатуре или щелкнуть мышкой на кнопке останова в окне консоли, появляющееся в момент выполнения кода (рис. 1.10).

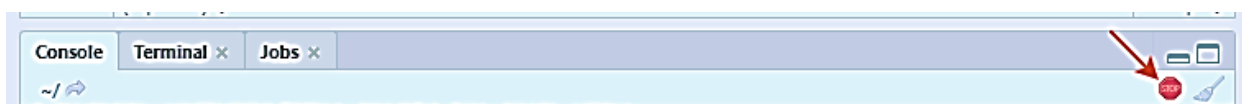
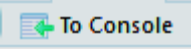
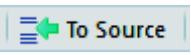


Рис. 1.10. Прерывание выполнения кода

Вкладка `History` (рис. 1.11) содержит журнал всех ранее выполненных в рабочем пространстве команд (в окне `Console`). Они сохраняются и после очистки окна `Console`. Эти команды можно передавать на вы-

полнение в Console кнопкой , команды выполняются после нажатия ENTER. Или скопировать в открытый файл в окне Source (если открытого файла нет, то создается новый) кнопкой . Для работы с несколькими командами их нужно предварительно выделить.

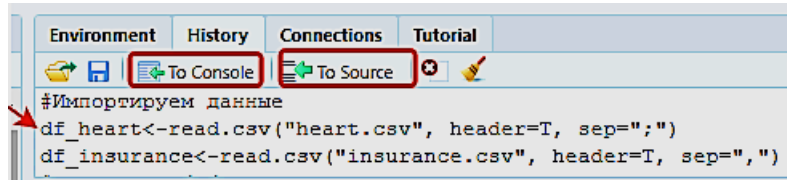


Рис. 1.11. Вкладка History

Вкладка Environment (рис. 1.12) содержит описание всех объектов (это могут быть пользовательские функции, переменные), которые были созданы ранее в рабочем пространстве.

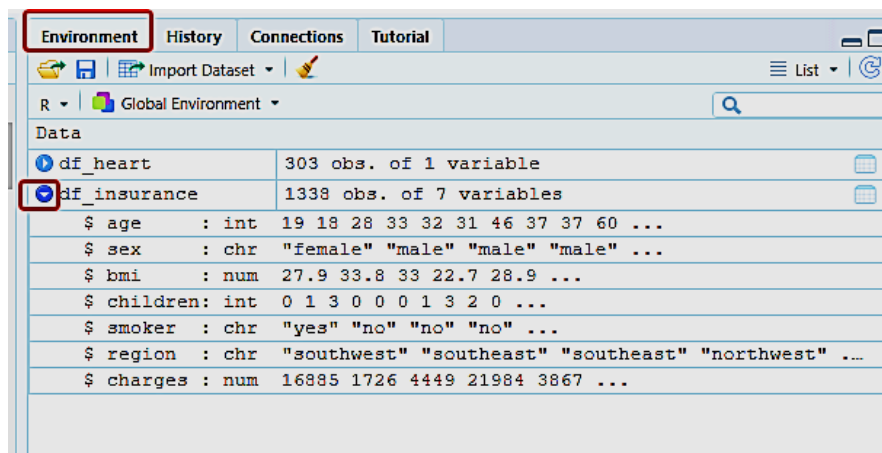


Рис. 1.12. Вкладка Environment

Установка новых пакетов, необходимых в данной рабочей сессии, возможна с помощью командного меню Tools > Install Packages..., на вкладке Packages – с помощью кнопки Install (рис. 1.13).

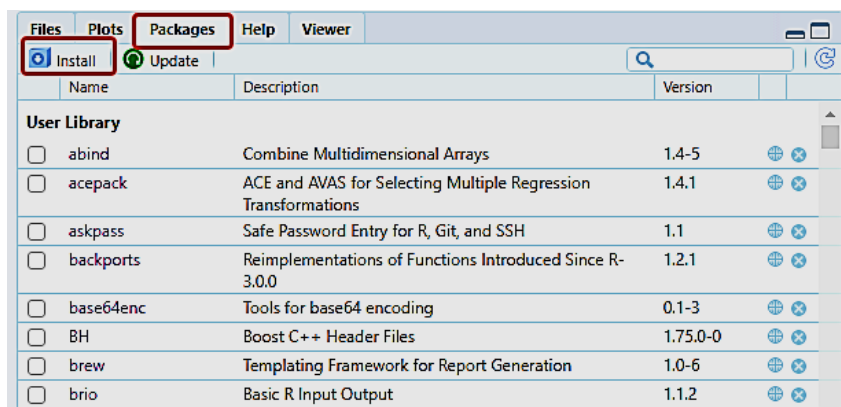


Рис. 1.13. Вкладка Packages

При наборе имени пакета подключается автозаполнение. Флаг на опции `Install dependencies` означает, что установятся и все связанные пакеты, на функции которых опирается нужный пакет (рис. 1.14).

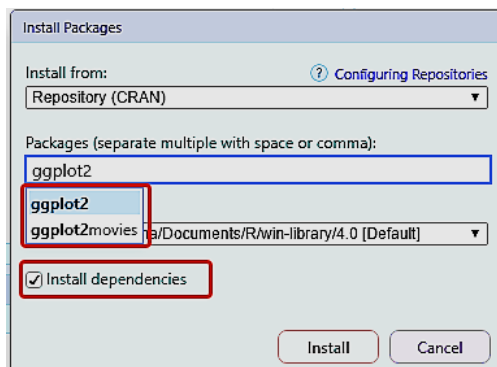


Рис. 1.14. Диалоговое окно загрузки пакетов

Установка новых пакетов возможна и с помощью команды, аргументы которой – это имя пакета (например, `'ggplot2'`) и значение `TRUE` для переменной `dependencies`, чтобы установились связанные пакеты:

```
install.packages('ggplot2', dependencies = TRUE)
```

Установленные пакеты нужно подключить в текущей рабочей сессии, чтобы можно было использовать определенные в них функции и имеющиеся в пакетах наборы данных. Это можно сделать на вкладке `Packages`, установив флаг на нужном пакете (рис. 1.15).

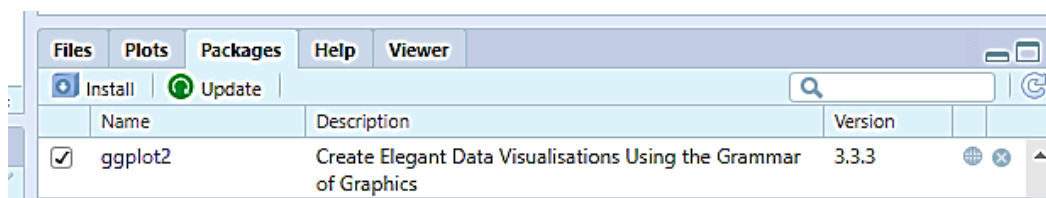


Рис. 1.15. Подключение пакетов в текущей рабочей сессии

Или с помощью команды

```
library(ggplot2)
```

Но подключение пакета с помощью команды `library()` не гарантирует, что такой пакет уже был установлен (и подключение не выполнится), тогда как в первом варианте недоступные для подключения пакеты просто не отображаются.

Важно, что при начале новой сессии инсталляция ранее установленных пакетов не требуется, а подключение должно быть проведено обязательно, если в рабочей сессии предполагается обращение к функциям этих пакетов или их данным.

На вкладке **Help** можно получить справку по пакетам, функциям, наборам данных и т.п. Помощь доступна и с помощью знака вопрос перед объектом, о котором нужна справка:

```
?library()
```

Для поиска справки по возможностям языка **R** можно использовать ресурс **Stack Overflow** (<https://stackoverflow.com>). В большинстве случаев на этом ресурсе есть вопросы, аналогичные тем, что возникли у вас и на них уже ответили.

На вкладке **Plots** отображаются построенные в текущей сессии графики. Между графиками можно перемещаться и экспортировать график в файлы различных форматов (рис. 1.16).

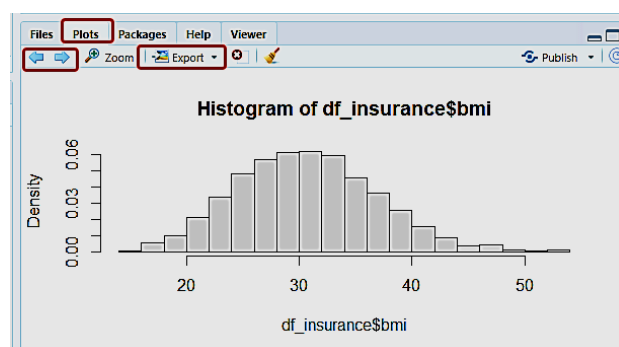


Рис. 1.16. Вкладка Plots

Запуск на выполнение кода, сохраненного в файле **R Script** (например, `second_project.R`), возможен из консоли с помощью команды `source()`, файл с расширением `.R` должен быть в рабочей директории:

```
source("second_project.R")
```

Также рассмотрим возможность работы с **RStudio** в облаке (рис. 1.17). Веб-ресурс расположен по адресу <https://rstudio.cloud/>.

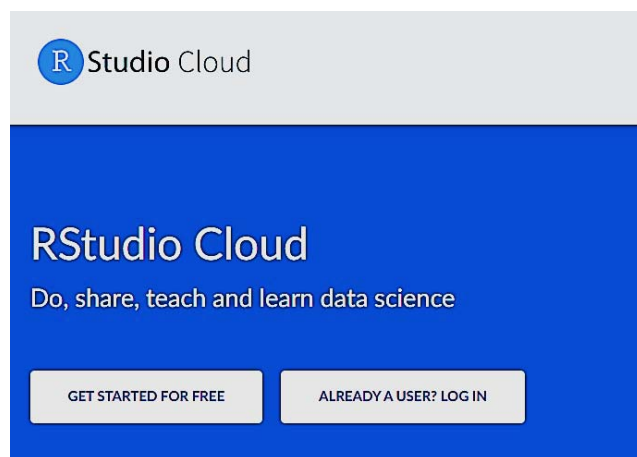


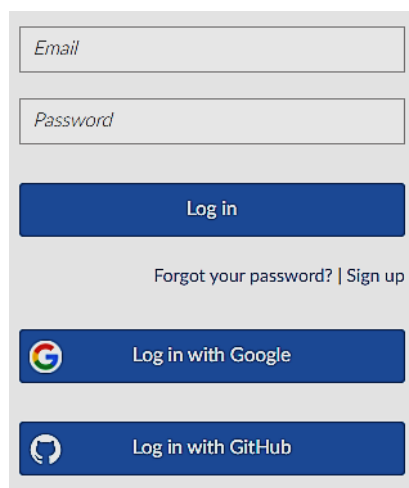
Рис. 1.17. RStudio Cloud

Бесплатный персональный план использования включает следующие возможности:

- до 15 проектов;
- одно общее пространство (максимум 5 участников и 10 проектов);
- 15 проектных часов в месяц;
- до 1 ГБ оперативной памяти на проект;
- до 1 процессора на проект;
- время выполнения в фоновом режиме до 1 часа.

Для тестирования возможностей облачной среды и небольших проектов вполне достаточно ограниченных возможностей бесплатного плана.

Регистрация происходит либо с помощью учетной записи Google или GitHub, либо с помощью новой учетной записи, созданной для веб-сервиса (рис. 1.18).



The image shows a login interface for RStudio Cloud. It features two input fields at the top: 'Email' and 'Password'. Below these is a blue 'Log in' button. Underneath the button is a link that says 'Forgot your password? | Sign up'. Further down are two more blue buttons: 'Log in with Google' (with the Google logo) and 'Log in with GitHub' (with the GitHub logo).

Рис. 1.18. Авторизованный вход в сервис

Загружается рабочая среда, в которой можно создать новый проект (рис. 1.19)

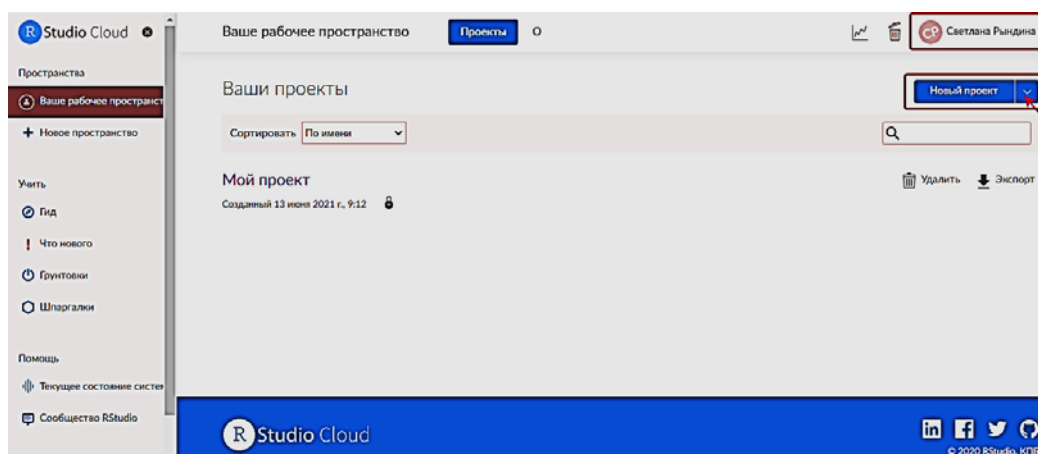


Рис. 1.19. Создание проекта в RStudio Cloud

Рабочее пространство облачной версии RStudio (рис. 1.20), аналогично десктопной версии (см. рис. 1.1).

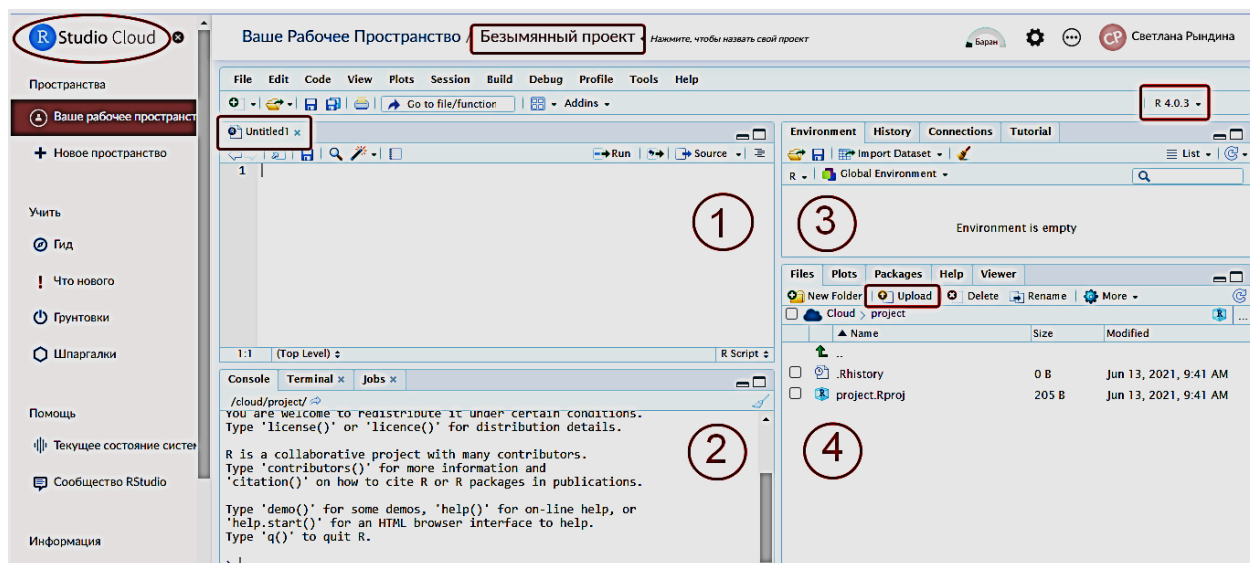


Рис. 1.20. Рабочая среда в RStudio

Для доступа к файлам с данными и кодом в десктопной версии RStudio они перемещались в рабочую директорию. По аналогии в рабочую директорию в облаке нужно загрузить файлы. В четвертом квадранте на вкладке Files есть кнопка Upload, которая позволяет загружать в рабочую директорию облака нужные файлы (с кодом, с данными). Загрузим файл с данными insurance.csv, который есть на компьютере (рис. 1.21).

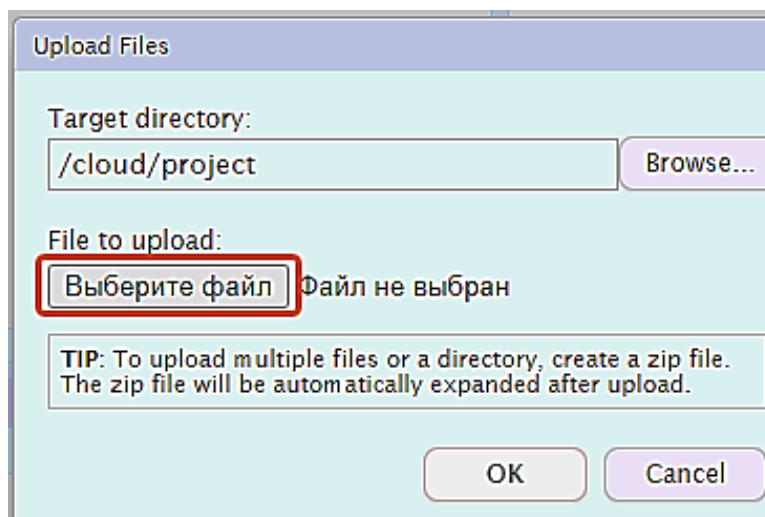


Рис. 1.21. Загрузка файла с данными в рабочую директорию в облаке

Выберем файл в Проводнике и одобрим его загрузку (рис. 1.22)



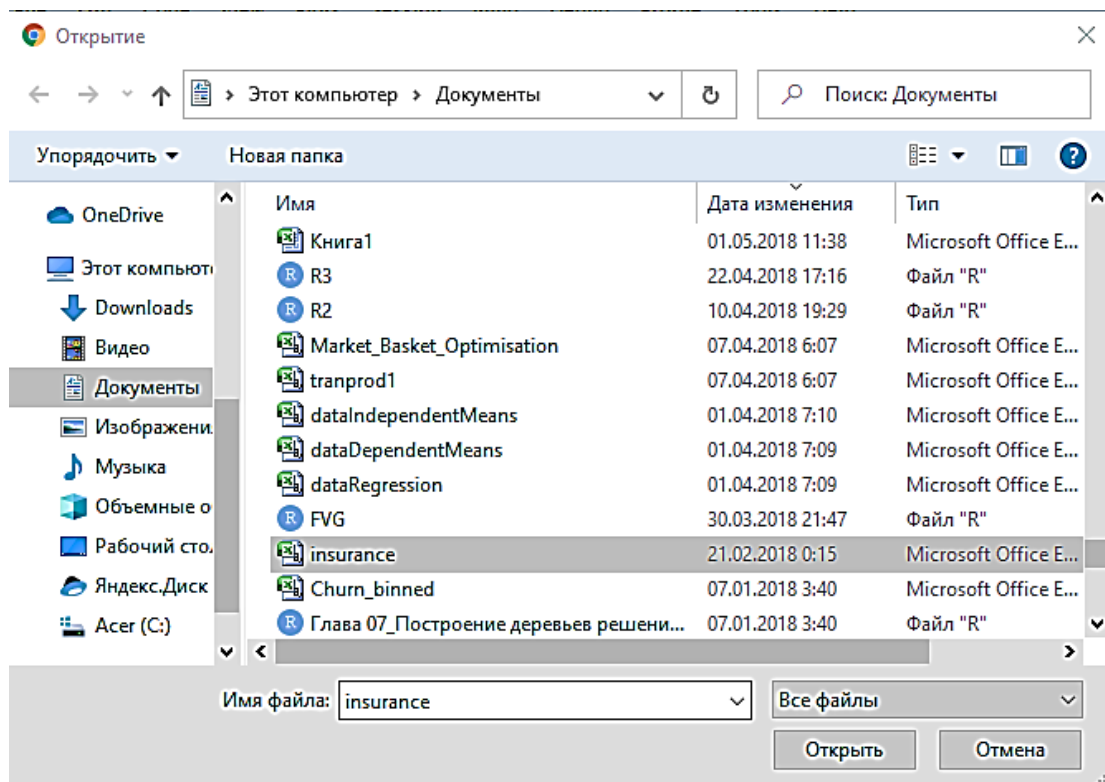


Рис. 1.22. Выбор файла для загрузки

Работа с кодом, т.е. создание кода на вкладке **Source**, выполнение в **Console**, отображение файлов рабочей директории на вкладке **Files**, где теперь отображается доступный для обращения к его содержимому файл с данными `insurance.csv`, не имеет отличий от десктопной версии (рис. 1.23).

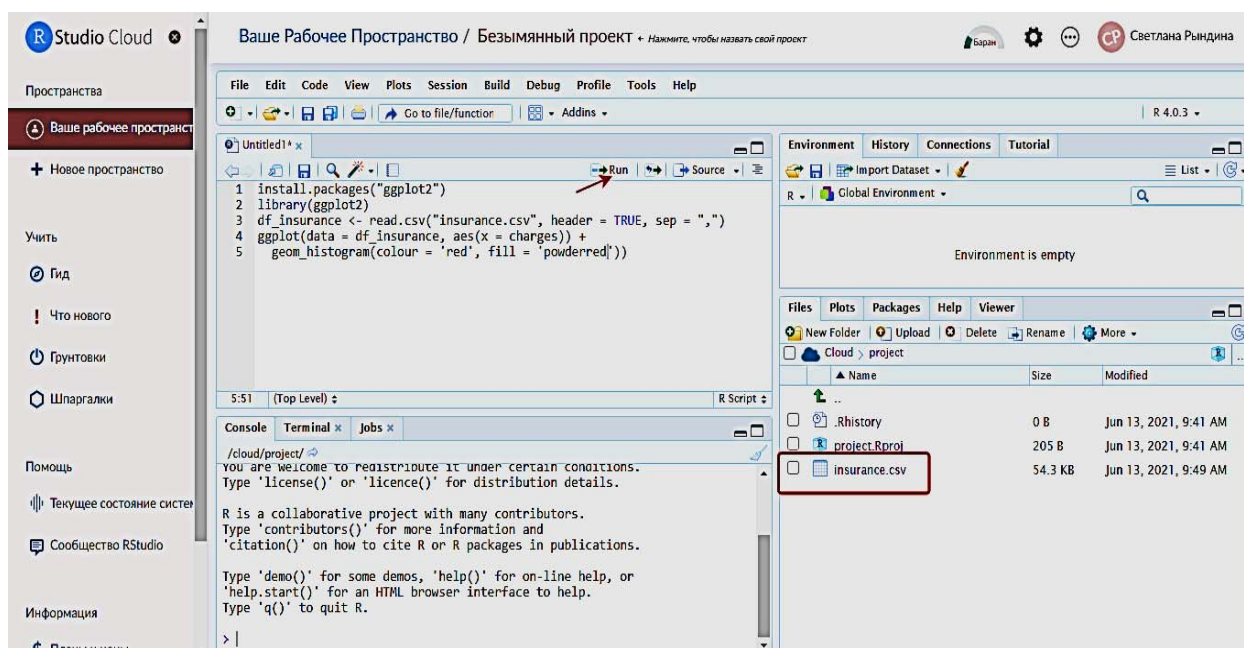


Рис. 1.23. Рабочая среда в RStudio Cloud



Загрузка и подключение пакетов в облачной версии происходят аналогично десктопной. Выполнение кода в веб-сервисе представлено на рис. 1.24.

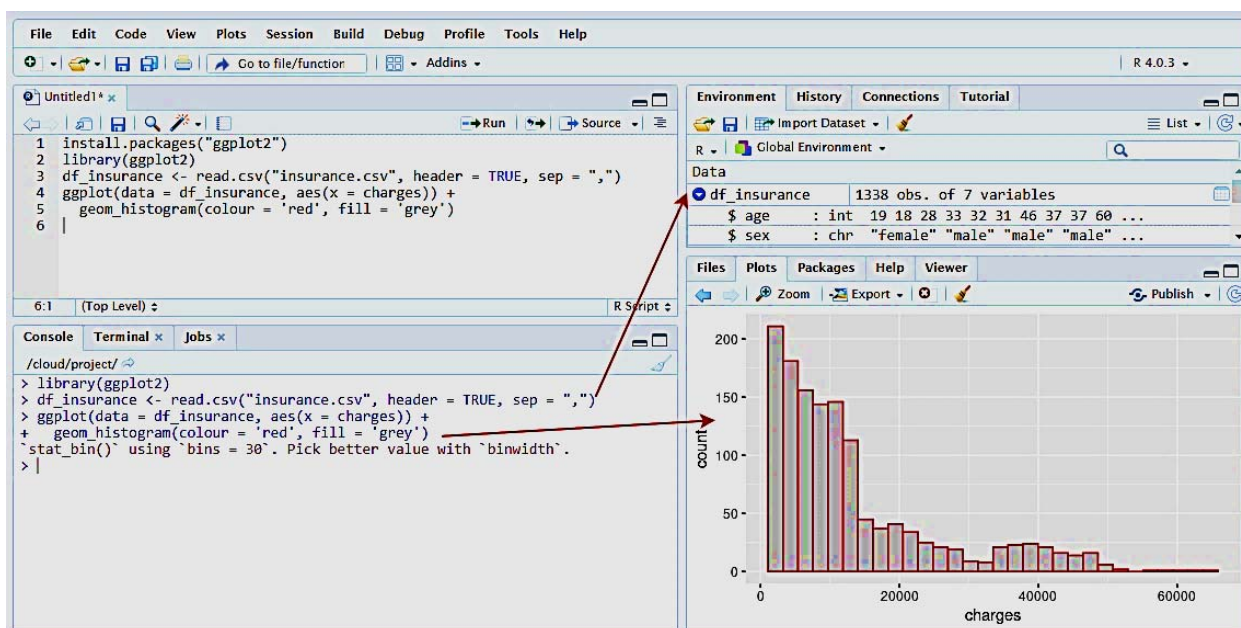


Рис. 1.24. Результат выполнения кода в RStudio Cloud

По щелчку мыши на профиле (рис. 1.25) открывается информация об используемых ресурсах из предоставленного объема (так, согласно статистике в профиле создано уже два проекта из доступных 15, истрачен почти час из 15 доступных каждый месяц часов работы в сервисе).

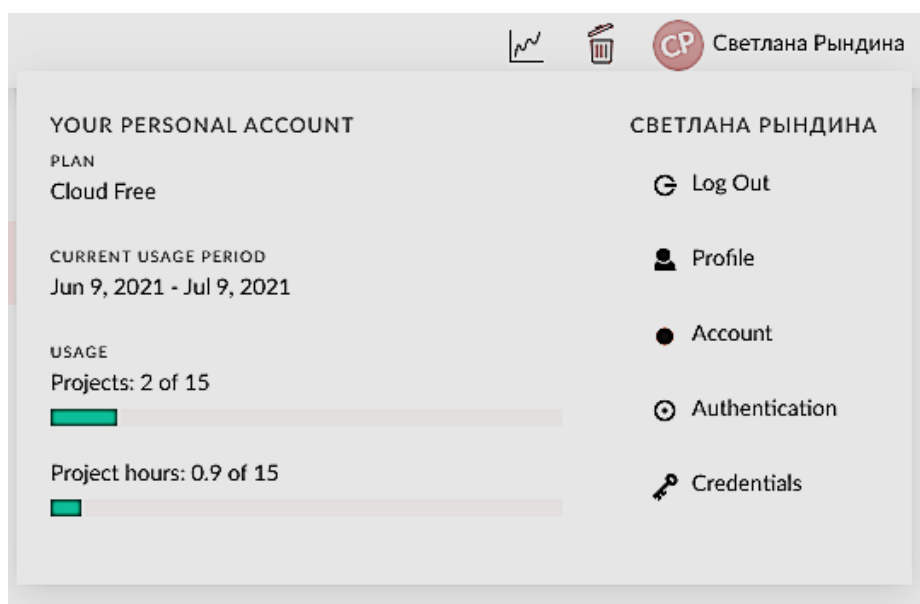


Рис. 1.25. Статистика по использованию возможностей бесплатного плана для RStudio Cloud

## 2. Данные

Любой проект по анализу данных начинается с данных. У данных есть такие характеристики, как тип и структура.

### 2.1. Типы данных и структуры, которые поддерживает R

#### Типы данных в R

Числовые данные (numeric) – это целочисленные данные (integer), действительные числа (double), комплексные числа (complex).

Символьные данные (character) – это текстовые данные, которые при вводе и выводе заключаются в одинарные или двойные кавычки.

Логические данные (logical) – принимают два значения TRUE и FALSE.

В R зарезервированы отдельные обозначения для особых данных («вырожденных» значений).

NaN – ноль-подобное значение (Not a Number, не число), применяется к числовым векторам (чаще всего возникает в результате неудачных вычислений, например, при попытке деления на ноль).

NA – логическая константа, определяющая отсутствующее значение (Not Available, отсутствующее значение), применяется к любым векторам. Например, при попытке вычислить среднее значение для символьных данных (рис. 2.1)

```
> mean(df_insurance$sex)
[1] NA
Предупреждение:
В mean.default(df_insurance$sex) :
  аргумент не является числовым или логическим: возвращаю NA
```

Рис. 2.1. Возвращение NA

NULL – ноль-подобное значение (ничто или пустое значение), обычно возвращается функциями в случае незадаанных значений. Также используется для удаления столбцов в таблицах данных.

#### Структуры данных в R

Основная структура – вектор. Все элементы вектора имеют один и тот же тип.

Скаляры, т.е. обычные числа, также считаются векторами длины 1.

Создать вектор можно с помощью функции `c()` (для названия функции использована первая буква слов concatenate и combine).

Создадим два вектора перечислением входящих в них элементов и присвоим их переменным `x` и `y`:

```
x <- c(1, 2, 3, 5)
y <- c("female", "male", 1)
```

После выполнения кода в **Console** (код при выполнении всегда начинается с приглашающего знака “>” интерпретатора, который выполняет код построчно, возвращая результат выполнения каждой строки в консоль), во вкладке **Environment** (левый верхний квадрант) появятся две переменные, числовая `x` и символьная `y`. Причем числовое значение 1 будет заключено в кавычки и работать с ним можно будет только как с текстом, а не как с числом (рис. 2.2).

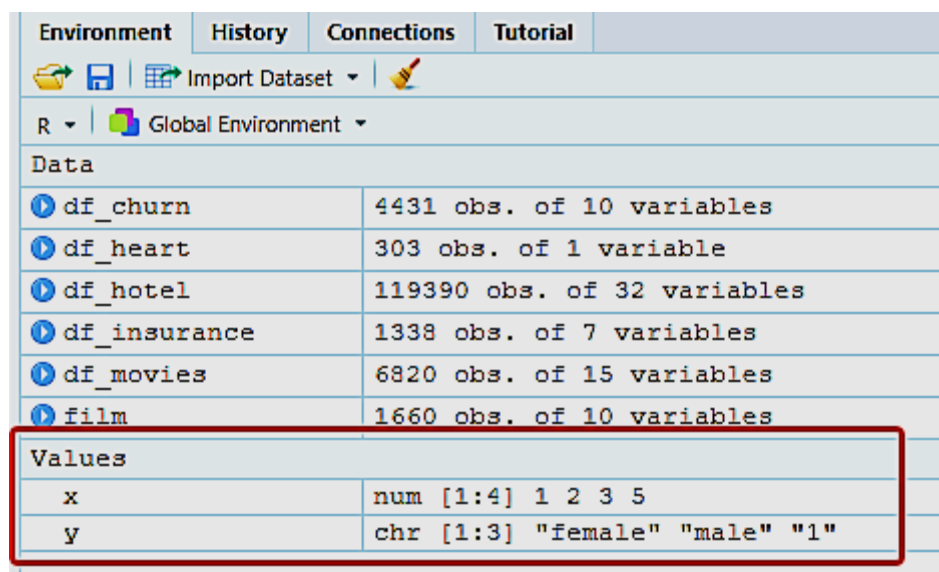


Рис. 2.2. Вкладка Environment

Обращаться к элементам вектора можно по индексу позиции, который указывается в квадратных скобках (рис. 2.3).

```
> y[2]
[1] "male"
> y[c(2,3)]
[1] "male" "1"
```

Рис. 2.3. Обращение к элементам вектора по индексу

Значениям вектора можно присвоить имена (рис. 2.4), и, хотя, на отображение вектора во вкладке **Environment** это не повлияет, вектор останется одномерной структурой, тем не менее, к элементам вектора можно обращаться теперь не только по индексу, но и по имени элемента. При выводе значений вектора будут отображаться и имена элементов.

```

> names(y) <- c("V1", "V2", "V3")
> y["V3"]
V3
"1"
> y
      V1      V2      V3
"female" "male"  "1"

```

Рис. 2.4. Присвоение элементам вектора имен

Заданные имена можно удалить с помощью присвоения значения NULL либо с помощью присвоения пустого вектора c(), результат будет аналогичным (рис. 2.5).

```

> names(y) <- NULL
> y
[1] "female" "male"  "1"

```

Рис. 2.5. Удаление имен элементов из вектора

Генерировать последовательности значений можно с помощью функций seq() и rep(). Функция seq() имеет три аргумента: начальное значение последовательности, конечное значение и шаг (или длина последовательности, т.е. число элементов). Функция rep() имеет два аргумента: вектор, который подлежит репликации, и сколько раз нужно выполнить повтор (рис. 2.6).

```

> seq(1, 7, by=3)
[1] 1 4 7
> seq(5, -2, by = -0.5)
[1] 5.0 4.5 4.0 3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0 -0.5 -1.0 -1.5 -2.0
> seq(5, 12, length = 4)
[1] 5.000000 7.333333 9.666667 12.000000
> rep(y, 2)
[1] "female" "male"  "1"      "female" "male"  "1"
> |

```

Рис. 2.6. Использование функций seq() и rep()

Списки – это специальный тип векторов, в которых элементы могут быть различных типов. Создать список можно с помощью функции list(). Создадим переменную z, в которую поместим те же значения, что и в y, но передадим их не функцией c(), а функцией list() (рис. 2.7).

```

> z <- list("female", "male", 1)
> z
[[1]]
[1] "female"

[[2]]
[1] "male"

[[3]]
[1] 1

```

Рис. 2.7. Создание списка

Для извлечения отдельных значений из списка используются индексы, указываемые в двойных квадратных скобках. Каждый элемент в переменной *z* считается вектором, имеющим единичную длину.

Создадим новый список в переменной *z1* из двух векторов *x* и *y*, и с помощью указания номера элемента в двойных квадратных скобках получим доступ к элементу списка – вектору, а с помощью указания в одинарных квадратных скобках индекса элемента вектора, входящего в список, получим доступ к конкретному элементу (рис. 2.8). Использовать в двойных квадратных скобках можно только один индекс (последняя операция завершилась неудачей).

```
> z1<- list(x,y)
> z1
[[1]]
[1] 1 2 3 5

[[2]]
[1] "female" "male" "1"

> z1[[2]]
[1] "female" "male" "1"
> z1[[2]] [2]
[1] "male"
> z1[[2]] [c(2,3)]
[1] "male" "1"
> z1[[c(1,2)]] [c(2,3)]
[1] NA NA
```

Рис. 2.8. Обращение к элементам списка

В виде списков в *R* возвращается результат применения статистических методов к набору данных. Именно поэтому им уделено такое внимание.

Массивы – структура, позволяющая хранить многомерные данные одного типа. Создаются массивы функцией *array()*.

Матрицы – это двумерные массивы, могут создаваться функцией *matrix()*.

И матрицы, и массивы можно считать векторами, которые имеют размер, определяемый функцией *dim()*.

Создадим вектор *z2*, зададим его размерность как 3×4 (получим матрицу из трех строк и двух столбцов), изменим размерность на 2×3×2 (получим массив соответствующей размерности) (рис. 2.9).

```

> z2<- rep(y,4)
> z2
[1] "female" "male"    "1"      "female"
[5] "male"    "1"      "female" "male"
[9] "1"      "female" "male"    "1"
> dim(z2) <- c(3,4)
> z2
      [,1] [,2] [,3] [,4]
[1,] "female" "female" "female" "female"
[2,] "male"    "male"    "male"    "male"
[3,] "1"      "1"      "1"      "1"
> dim(z2) <- c(2,3,2)
> z2
, , 1
      [,1] [,2] [,3]
[1,] "female" "1"    "male"
[2,] "male"    "female" "1"

, , 2
      [,1] [,2] [,3]
[1,] "female" "1"    "male"
[2,] "male"    "female" "1"

```

Рис. 2.9. Преобразование вектора в массив и матрицу

У R есть особая структура – фактор, который похож на символьный вектор, но с особыми свойствами. Каждое уникальное значение в таком векторе представляет собой градацию или уровень фактора. Эта структура используется для представления категориальных переменных, как номинальных, так и порядковых. Создадим два символьных вектора `gender` и `size` и преобразуем их в факторы:

```

gender <- c("female", "male", "male", "female", "male", "female", "female", "male" )
size <- c("XS", "L", "M", "XL", "XS", "S")
gender <- as.factor(gender)
size <- as.factor(size)

```

На вкладке `Environment` мы видим структуру переменных (рис. 2.10): `gender` имеет два уровня “female” с числовой меткой 1, “male” с числовой меткой 2 (произошло упорядочение уровней по алфавиту). Аналогично для фактора `size` числовые метки распределились по уровням следующим образом: размер “L” метка – 1, размер “M” – метка 2, размер “S” – метка 3, размер “XL” – метка 4, размер “XS” – метка 5. Эти метки противоречат размерному ряду, так как буквенные размеры одежды упорядочиваются не по алфавиту.

Values	
gender	Factor w/ 2 levels "female","male": 1 2 2 1 2 1 1 2
size	Factor w/ 5 levels "L","M","S","XL",...: 5 1 2 4 5 3

Рис. 2.10. Отображение факторов на вкладке `Environment`

Для номинальной переменной порядок присвоения меток неважен, а вот для ординальной его нужно задать:

```
size <- factor(size, levels = c("XS", "S", "M", "L", "XL"))
```

На вкладке Environment теперь порядок присвоения меток соответствует упорядочению размеров от наименьшего “XS” к наибольшему “XL” (рис. 2.11).

Values	
gender	Factor w/ 2 levels "female","male": 1 2 2 1 2 1 1 2
size	Factor w/ 5 levels "XS","S","M","L",...: 1 4 3 5 1 2

Рис. 2.11. Отображение факторов на вкладке Environment

Рассмотрим таблицы данных – наиболее востребованную структуру в анализе данных.

Таблица данных состоит из строк и столбцов. Значения в столбце должны быть одного типа, но сами столбцы в таблице могут иметь разный тип. Эта структура данных создается функцией `data.frame()`. Создадим три вектора одинаковой длины и объединим их в таблицу данных (иногда говорят фрейм данных, это словосочетание будет использоваться далее как синонимичное):

```
gender <- c("female", "male", "male", "female", "male", "female" )
size <- c("XS", "L", "M", "XL", "XS", "S")
totalsum <- c( 3000, 2456, 8965, 4567, 3487, 9555)
df <- data.frame(gender, size, totalsum)
```

Созданную таблицу данных можно открыть на вкладке Source и просмотреть на вкладке Environment (рис. 2.12).

	gender	size	totalsum
1	female	XS	3000
2	male	L	2456
3	male	M	8965
4	female	XL	4567
5	male	XS	3487
6	female	S	9555

Variable	Class	Levels
gender	chr	"female" "male" "male" "female" ...
size	chr	"XS" "L" "M" "XL" ...
totalsum	num	3000 2456 8965 4567 3487 ...

Рис 2.12. Просмотр отображения фрейма данных

К элементам таблицы можно обращаться разными способами (рис. 2.13): к отдельному столбцу как к вектору (`df$gender`), по индексу столбца или по имени столбца (можно использовать сразу несколько индексов, например, `c(1,3)` или несколько имен `c("gender", "totalsum")`),



используя размерность таблицы, когда число в квадратных скобках до запятой – это индекс строки, а после запятой – это индекс столбца (отсутствие числа до запятой означает выбор всех строк, отсутствие числа после – выбор всех столбцов, можно также использовать несколько индексов, объединяя их в вектор). С помощью функции `str()` можно вывести информацию о структуре объекта, передаваемого в функцию в качестве аргумента.

```
> df$gender
[1] "female" "male"   "male"   "female" "male"   "female"
> str(df$gender)
chr [1:6] "female" "male" "male" "female" "male" ...
> df[1]
  gender
1 female
2   male
3   male
4 female
5   male
6 female
> str(df[1])
'data.frame':   6 obs. of  1 variable:
 $ gender: chr  "female" "male" "male" "female" ...
> df["gender"]
  gender
1 female
2   male
3   male
4 female
5   male
6 female
> str(df["gender"])
'data.frame':   6 obs. of  1 variable:
 $ gender: chr  "female" "male" "male" "female" ...
> df[,2]
[1] "XS" "L"  "M"  "XL" "XS" "S"
> df[3,]
  gender size totalsum
3   male    M      8965
> df[3,2]
[1] "M"
```

Рис. 2.13. Обращение к элементам фрейма данных и запрос их структуры

## 2.2. Импорт данных. Построение срезов

Так как в большинстве случаев в анализе данных имеют дело именно с табличным представлением данных и не вводят их вручную, а загружают данные из различных источников, то далее будем рассматривать импортированные наборы данных.

Текстовый формат табличных данных часто представлен расширением `.csv` (Comma-Separated Values, значения разделенные запятыми).



Для его загрузки в рабочее пространство используется функция `read.csv()`

Таблица 2.1

Аргументы `read.csv()`

Аргумент	Описание
<code>file</code>	Имя файла, из которого должны быть прочитаны данные. Каждая строка таблицы отображается как одна строка файла. Если он не содержит абсолютного пути, имя файла указывается относительно текущего рабочего каталога
<code>header</code>	Логическое значение, указывающее, содержит ли файл имена переменных в качестве первой строки
<code>sep</code>	Символ разделителя полей (начальные буквы <code>separator</code> , т.е. разделитель). Значения в каждой строке файла разделяются этим символом
<code>dec</code>	Символ, используемый в файле для десятичных знаков
<code>row.names</code>	Вектор имен строк. Это может быть вектор, дающий фактические имена строк, или одно число, дающее столбец таблицы, который содержит имена строк, или символьную строку, дающую имя столбца таблицы, содержащего имена строк. Если есть заголовок и первая строка содержит на одно поле меньше, чем количество столбцов, первый столбец во входных данных используется для имен строк
<code>col.names</code>	Вектор необязательных имен для переменных. По умолчанию используется значение «V», за которым следует номер столбца
<code>fileEncoding</code>	Символьная строка. Объявляет кодировку, используемую в файле, поэтому символьные данные могут быть перекодированы

Распространенные кодировки:

- “utf-8” – стандарт кодирования символов, позволяющий более компактно хранить и передавать символы Юникода, используя переменное количество байт;

- “cp1251” – стандартная восьмибитная кодировка для русских версий Microsoft Windows до 10 версии.

Если в импортированном наборе символы имеют странное представление, то необходимо отработать вопрос с кодировкой.

Используемые в примерах данные – расходы на медицинское обслуживание тех, кто имеет медицинскую страховку, доступны на ресурсе kaggle: URL: <https://www.kaggle.com/mirichoi0218/insurance>.

Описание переменных набора:

- `age`: возраст основного бенефициара;
- `sex`: пол застрахованного;
- `bmi`: индекс массы тела;

- children: число детей, охваченных медицинским страхованием / число иждивенцев;
- smoker: курит ли застрахованный;
- region: жилой район получателя страховых выплат в США (северо-восток, юго-восток, юго-запад, северо-запад).
- charges: индивидуальные медицинские расходы, оплачиваемые страховкой.

Используем функцию `read.csv()` для загрузки в рабочее пространство набора данных из файла `insurance.csv` предварительно размещенного в рабочей директории (рис. 2.14):

```
df_insurance<-read.csv("insurance.csv", header=TRUE, sep=",")
```

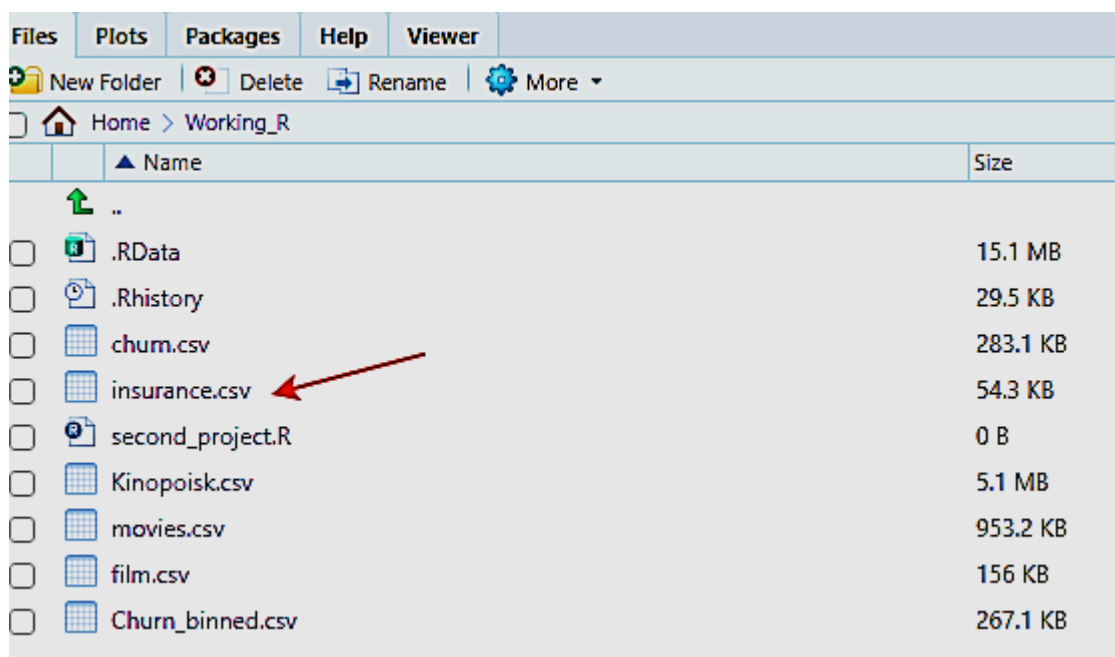


Рис. 2.14. Загрузка файла с данными из рабочей директории

Построим срезы на основе отбора строк и столбцов и на основе фильтров на загруженном наборе данных:

```
#Вектор данных из столбца age
insurance_age <- df_insurance$age

#Срез таблицы по столбцу age
df_insurance_age <- df_insurance["age"]

#Срез таблицы по столбцам с первого по четвертый и седьмому
df_insurance_12347 <- df_insurance[, c(1:4,7)]
```

```

#Срез таблицы по первым 30-ти строкам
df_insurance_1_30 <- df_insurance[1:30, ]

#Срез таблицы по первым 30-ти строкам и 100-ой строке
df_insurance_1_30_100 <- df_insurance[c(1:30, 100), ]

#Фильтрация: отбор строк по условию region == "southeast"
df_insurance_southeast <- df_insurance[df_insurance$region == "southeast",]

#Фильтрация: отбор строк по условию region != "southeast"
df_insurance_no_southeast <- df_insurance[df_insurance[6] != "southeast",]

#Фильтрация: отбор строк по условию charges > 20000
df_insurance_big_charges <- df_insurance[df_insurance[7] > 20000,]

#Фильтрация: отбор строк по составному условию
#charges > 20000 & region == "southeast"
df_southeast_big_charges <- df_insurance[
  df_insurance[7] > 20000 & df_insurance[6] == "southeast",]

#Фильтрация: отбор строк по условию значения region из вектора
df_insurance_south <- df_insurance[df_insurance$region %in%
  c("southeast", "southwest"), ]

```

В Environment созданные переменные срезов для исходного набора имеют структуру как на рис. 2.15.

Environment	
R   Global Environment	
Data	
df_insurance	1338 obs. of 7 variables
df_insurance_1_30	30 obs. of 7 variables
df_insurance_1_30_100	31 obs. of 7 variables
df_insurance_12347	1338 obs. of 5 variables
df_insurance_age	1338 obs. of 1 variable
df_insurance_big_charges	273 obs. of 7 variables
df_insurance_no_southeast	974 obs. of 7 variables
df_insurance_south	689 obs. of 7 variables
df_insurance_southeast	364 obs. of 7 variables
df_southeast_big_charges	88 obs. of 7 variables
Values	
insurance_age	int [1:1338] 19 18 28 33 32 31 46 37 ...

Рис. 2.15. Вкладка Environment после выполнения кода

Логические операторы, которые используются в условиях, представлены в табл. 2.2.

Таблица 2.2

### Логические операторы

Оператор	Описание
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно
&	Логическое И
	Логическое ИЛИ
!	Логическое НЕ

Функция `order()` используется для сортировки строк таблицы по значению какого-либо столбца (табл. 2.3).

Таблица 2.3

### Аргументы `order()`

Аргумент	Описание
x	Вектор, значения которого нужно упорядочить. Нельзя передавать столбец таблицы (например, <code>df_insurance[1]</code> ), только вектор ( <code>df_insurance\$age</code> )
decreasing	Логическое значение, указывающее, в каком порядке нужно упорядочивать ( <code>decreasing</code> – убывающий). <code>decreasing = FALSE</code> (упорядочение по возрастанию, это значение по умолчанию, используется при незаданном параметре). <code>decreasing = TRUE</code> (упорядочение по убыванию)

Проведем упорядочение данных по столбцу `age` по возрастанию/по убыванию:

```
#Сортировка по столбцу age (по возрастанию)
df_insurance_ordered <- df_insurance[order(df_insurance$age), ]

#Сортировка по столбцу age (по убыванию)
df_insurance_ordered <- df_insurance[order(df_insurance$age, decreasing = TRUE), ]
```

Пример выполнения кода приведен на рис. 2.16.

```

> #Сортировка по столбцу age и вывод на экран первых десяти строк
> df_insurance_ordered <- df_insurance[order(df_insurance$age, decreasing
= FALSE),]
> df_insurance_ordered[1:10,]
  age  sex  bmi children smoker  region  charges
2   18 male 33.770      1    no southeast 1725.552
23  18 male 34.100      0    no southeast 1137.011
32  18 female 26.315      0    no northeast 2198.190
47  18 female 38.665      2    no northeast 3393.356
51  18 female 35.625      0    no northeast 2211.131
58  18 male 31.680      2   yes southeast 34303.167
103 18 female 30.115      0    no northeast 21344.847
122 18 male 23.750      0    no northeast 1705.624
158 18 male 25.175      0   yes northeast 15518.180
162 18 female 36.850      0   yes southeast 36149.484
> #Сортировка по столбцу age и вывод на экран первых десяти строк
> df_insurance_ordered <- df_insurance[order(df_insurance$age, decreasing
= TRUE),]
> df_insurance_ordered[1:10,]
  age  sex  bmi children smoker  region  charges
63   64 male 24.700      1    no northwest 30166.62
95   64 female 31.300      2   yes southwest 47291.06
200  64 female 39.330      0    no northeast 14901.52
329  64 female 33.800      1   yes southwest 47928.03
336  64 male 34.500      0    no southwest 13822.80
379  64 female 30.115      3    no northwest 16455.71
399  64 male 25.600      2    no southwest 14988.43
403  64 female 32.965      0    no northwest 14692.67
419  64 male 39.160      1    no southeast 14418.28
421  64 male 33.880      0   yes southeast 46889.26

```

Рис. 2.16. Упорядочение строк в таблицах по значениям одного из столбцов по убыванию/по возрастанию

Для вывода нескольких строк в таблице данных можно использовать функции `head()` и `tail()`:

```

#Вывод на экран первых семи строк набора данных df_insurance
head(df_insurance, 7)

#Вывод на экран последних семи строк набора данных df_insurance
tail(df_insurance, 7)

```

При импортировании данных можно использовать функции пакета “readr”.

Щелчком левой кнопки мыши на файле данных в рабочей директории на вкладке **Files** вызываем контекстное меню и выбираем **Import Dataset...** (рис. 2.17).

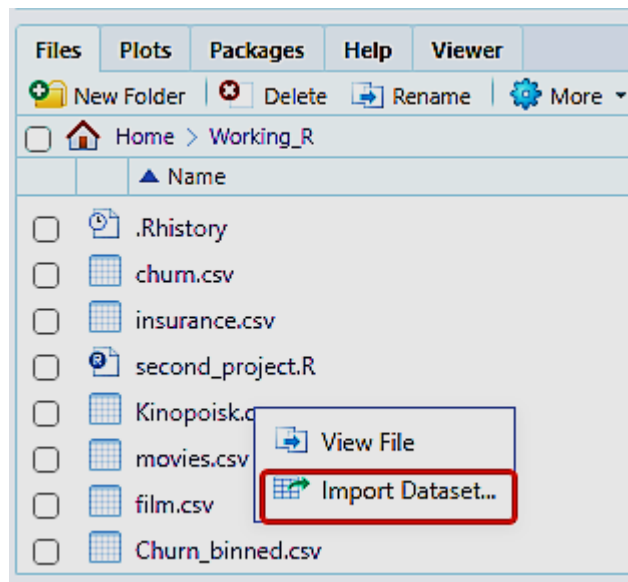


Рис. 2.17. Загрузка файла с данными

Откроется диалоговое окно работы с файлом данных (рис. 2.18).

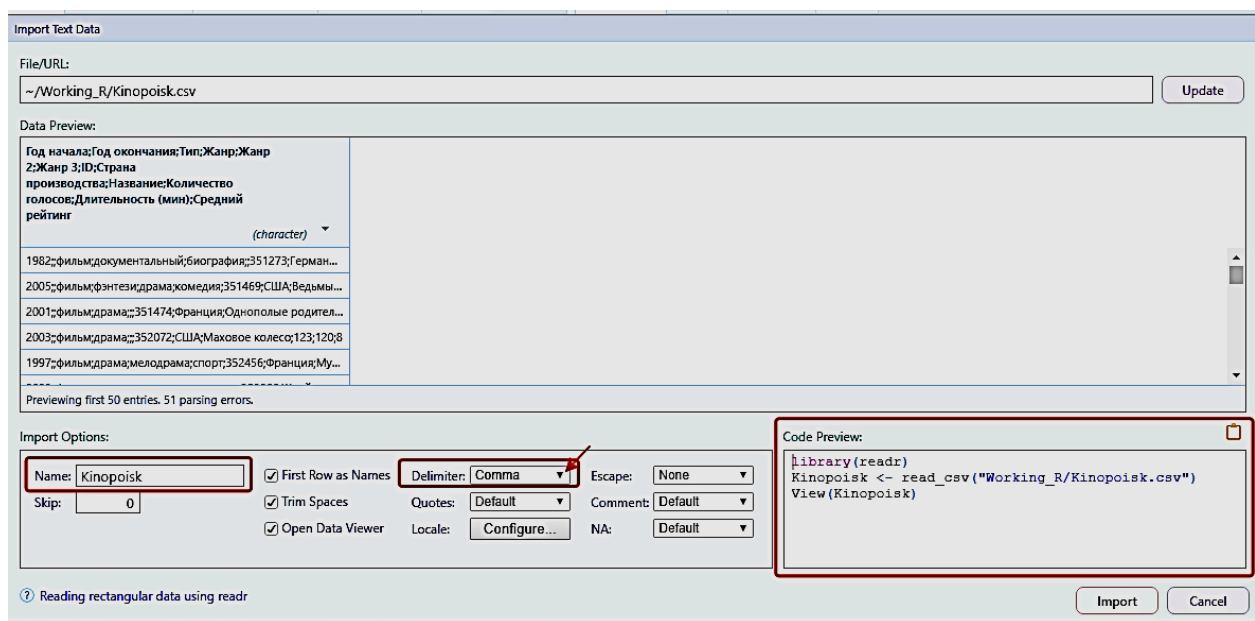


Рис. 2.18. Настройка параметров загрузки

В диалоговом окне импорта можно изменить имя переменной, которой будет присвоен загруженный набор (по умолчанию имя переменной совпадает с именем файла). Также можно сменить разделитель (delimiter), если он определился неверно, как в нашем примере (на просмотре сразу отобразится несколько столбцов вместо одного). В правом нижнем углу отображается код, который генерируется для действий в этом окне: подключение нужной библиотеки, чтение данных из файла и их просмотр (рис. 2.19).

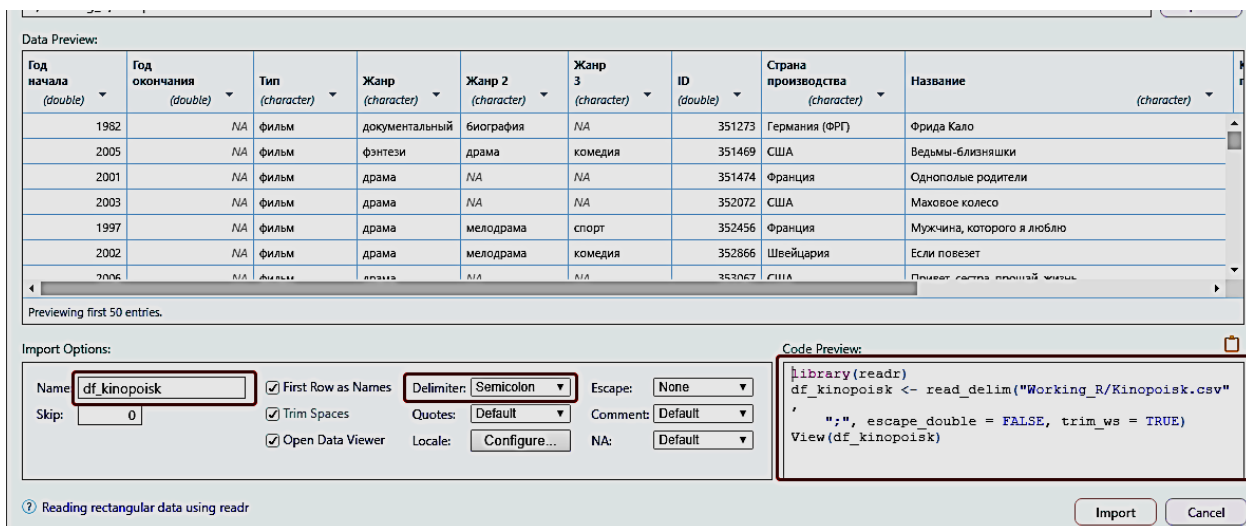


Рис. 2.19. Изменение настроек, выставленных по умолчанию

Стрелка справа от имени столбца позволяет изменять тип данных (рис. 2.20).

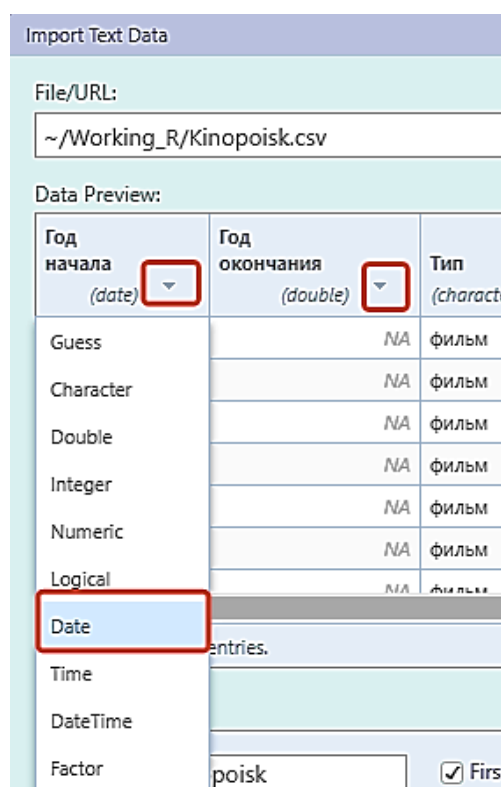


Рис. 2.20. Изменение типа загружаемых данных

В результате выполнения автоматически сгенерированного кода по выбору операций для работы с данными в диалоговом окне загруженный набор данных отобразится в окне Source, а в окне Environment появится фрейм данных, в котором кроме столбцов есть дополнительные атрибуты (рис. 2.21).



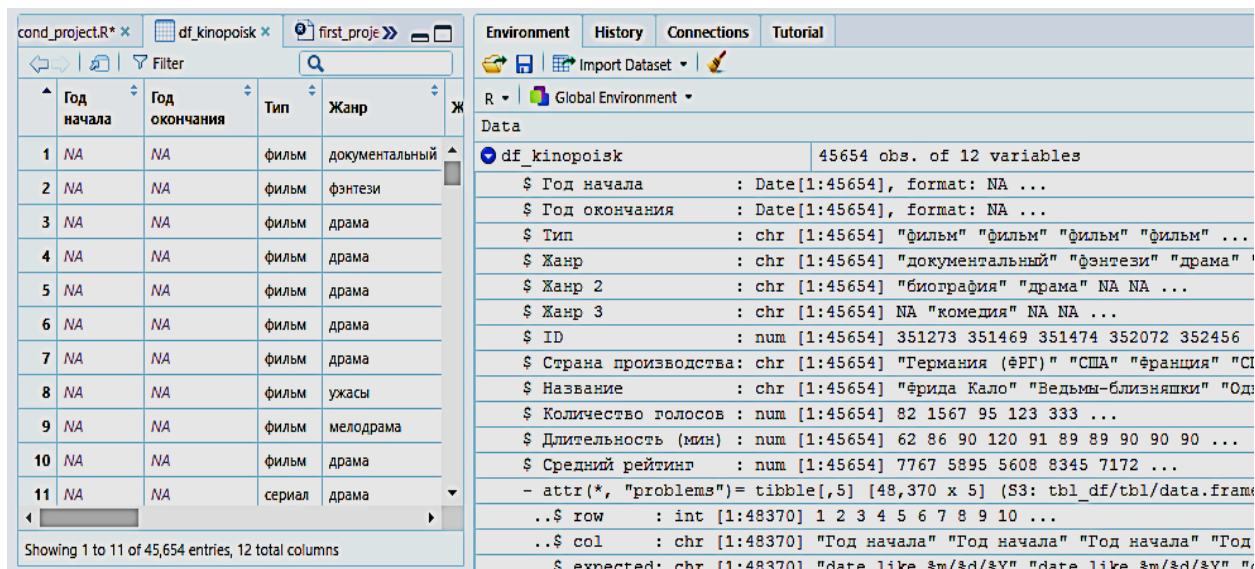


Рис. 2.21. Загруженный набор данных

Чтобы они нам не мешали при анализе данных, можно их удалить из переменной:

```
attr(df_kinopoisk, "problems")<-NULL
```

Загрузим файл с данными из репозитория URL: <https://github.com/Smeilz/ML-Class/tree/master/R/Trees> имя файла Churn\_binned.csv [1, с. 319–320]. Перенесем файл в рабочую директорию и вызовем в RStudio контекстное меню Import Dataset... Сменим имя файла, выберем корректный разделитель для данных и столкнемся с проблемой кодировки, когда кириллица не считывается (рис. 2.22).

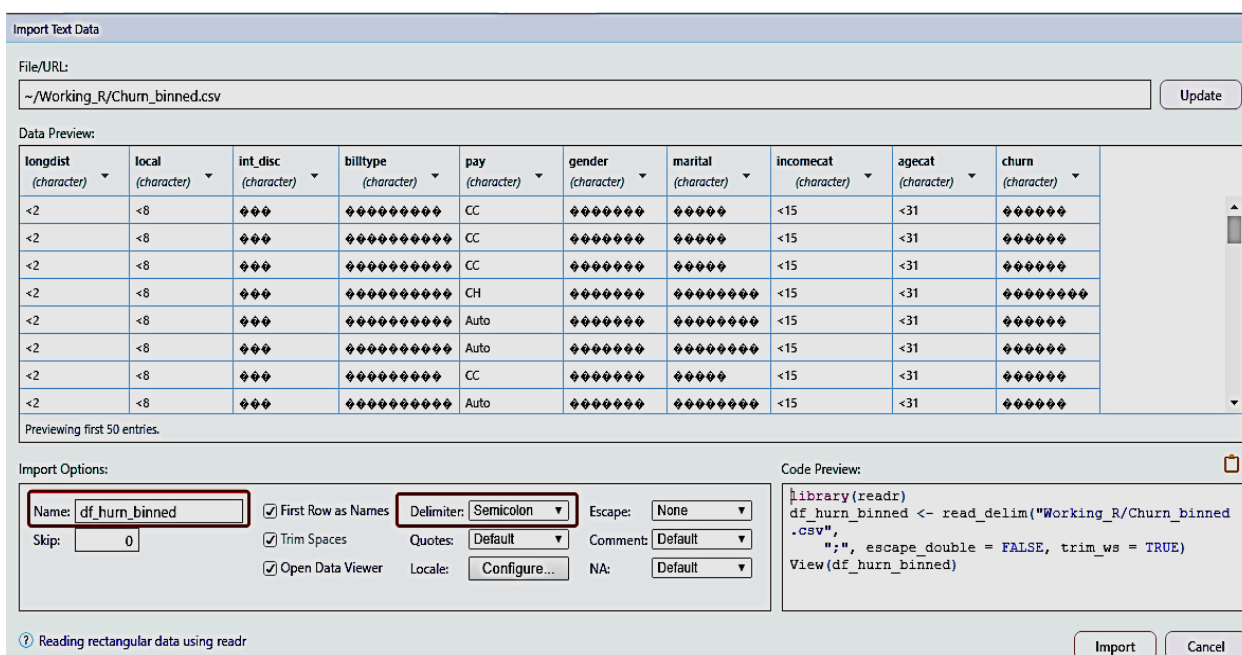


Рис. 2.22. Проблемы с кодировкой в загружаемом файле



Поработаем с кодировкой (нажмем на Configure... для Locale, выберем в списке Encoding опцию Other... и введем кодировку “cp1251” (характерную для текстовых файлов с кириллицей, созданных в операционной системе Windows до 10 версии)) (рис. 2.23).

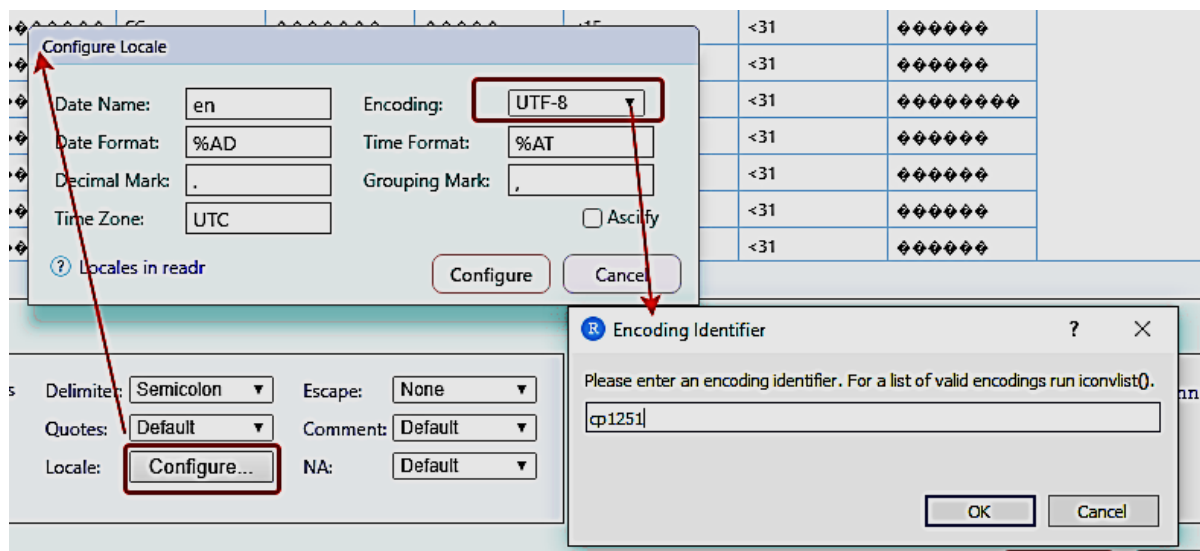


Рис. 2.23. Изменение кодировки загружаемого файла

Если кодировка выбрана верно, то на просмотре отобразится корректное содержание таблицы и можно принимать автоматически сгенерированный код на выполнение (рис. 2.24).

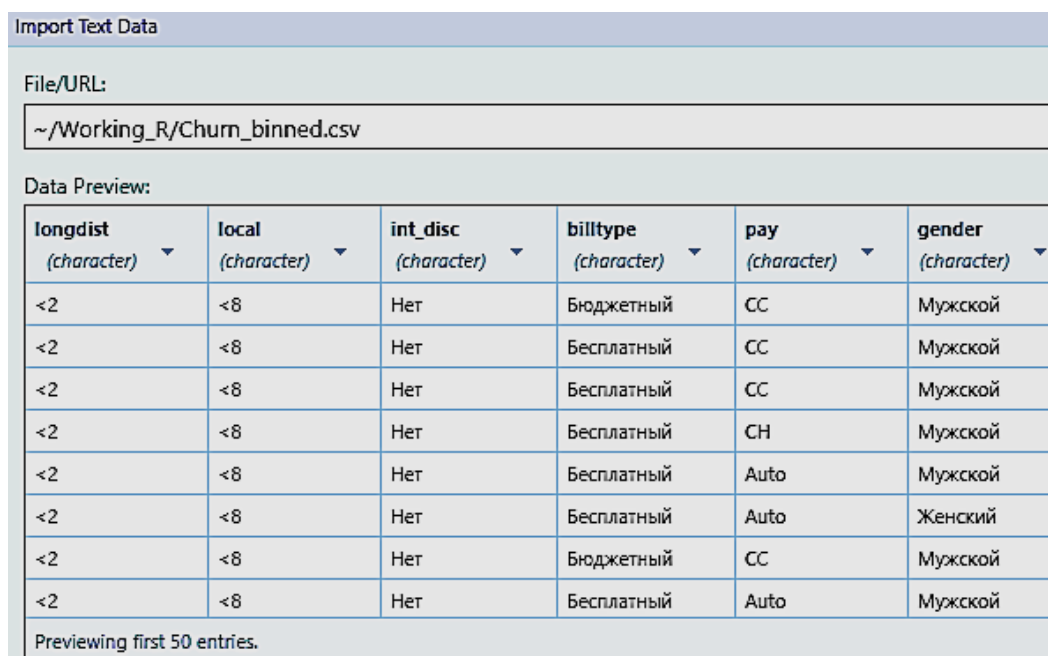


Рис. 2.24. Просмотр содержимого файла с измененными настройками кодировки

### 3. Описательный анализ данных

Базовые функции для расчета параметров описательной статистики приведены в табл. 3.1.

Таблица 3.1

Параметры описательной статистики

Функция	Описание
mean()	Среднее значение
median()	Медиана
var()	Дисперсия
sd()	Стандартное отклонение
min()	Минимальное значение
max()	Максимальное значение
quantile()	Квантили (по умолчанию рассчитывается максимальное и минимальное значения, а также квантили)
IQR()	Интерквартильный размах (межквартильный размах)

Среди данных могут быть пропуски, которые не позволят рассчитать статистические параметры для столбцов количественных переменных с пропусками без специальных настроек.

Функция `is.na()` возвращает набор данных (вектор, таблицу и т.п.) заполненный значениями `TRUE` (если значение отсутствует) и `FALSE` (при его наличии). Так как `TRUE` маркируется 1, а `FALSE` маркируется нулем, то функция `sum()` примененная к такому преобразованному набору данных дает число пропусков. В загруженном наборе данных `df_insurance` нет пропусков (рис. 3.1).

```
> sum(is.na(df_insurance))  
[1] 0
```

Рис. 3.1. Подсчет числа пропусков в данных

В копии исходного набора принудительно создадим пропуски в данных (рис. 3.2):

```
#Скопируем загруженный набор данных в новую переменную  
df_insurance_NA <- df_insurance  
  
#В сотой строке заменим значения в трех столбцах на NA  
df_insurance_NA[100,c(1, 3, 7)] <- c(NA, NA, NA)  
  
#Проконтролируем, что замена произошла  
df_insurance_NA[100, ]
```

```
#Подсчитаем общее число пропусков в данных
sum(is.na(df_insurance_NA))

#Подсчитаем общее число пропусков в каждом столбце
colSums(is.na(df_insurance_NA))

#Подсчитаем общее число пропусков в столбце age
sum(is.na(df_insurance_NA$age))
```

```
> #Скопируем загруженный набор данных в новую переменную
> df_insurance_NA <- df_insurance
> #В сотой строке заменим значения в трех столбцах на NA
> df_insurance_NA[100,c(1, 3, 7)] <- c(NA, NA, NA)
> #Проконтролируем, что замена произошла
> df_insurance_NA[100, ]
  age sex bmi children smoker  region charges
100 NA male  NA      0    yes southwest    NA
> #Подсчитаем общее число пропусков в данных
> sum(is.na(df_insurance_NA))
[1] 3
> #Подсчитаем общее число пропусков в каждом столбце
> colSums(is.na(df_insurance_NA))
  age      sex      bmi children  smoker  region  charges
    1         0         1         0        0         0         1
> #Подсчитаем общее число пропусков в столбце age
> sum(is.na(df_insurance_NA$age))
[1] 1
```

Рис. 3.2. Работа с пропусками

Попробуем вычислить средний возраст для наблюдений в наборе данных `df_insurance` без пропусков и с пропусками (рис. 3.3)

```
> mean(df_insurance$age)
[1] 39.20703
>
> mean(df_insurance_NA$age)
[1] NA
>
> mean(df_insurance_NA$age, na.rm = TRUE)
[1] 39.20793
```

Рис. 3.3. Вычисление среднего значения в данных `age`

Как видим, чтобы в данных с пропусками сработали функции для вычисления статистических параметров, необходимо дополнить функцию аргументом `na.rm = TRUE` (для игнорирования пропущенных значений).

Для большой выборки влияние одного пропущенного значения на статистический параметр ничтожно. В этом случае можно просто усечь данные (рис. 3.4), исключив строки с пропущенными значениями с помощью функции `na.omit()`.

mean(df_insurance\$age)	
mean(df_insurance_NA\$age)	
mean(df_insurance_NA\$age, na.rm = TRUE)	
df_insurance_not_NA <- na.omit(df_insurance_NA)	

R Global Environment	
Data	
df_insurance	1338 obs. of 7 variables
df_insurance_NA	1338 obs. of 7 variables
df_insurance_not_NA	1337 obs. of 7 variables
Values	

Рис. 3.4. Исключение строк с пропусками из набора данных

Нам удалось избавиться от трех пропусков исключением всего одной строки. Но если таких пропусков много, то подобный подход может лишить нас значительной доли наблюдений. Особенно остро эта проблема стоит в данных, которые имеют пропуски, расположенные в разных строках для разных столбцов. Пусть, например, в наборе данных имеется 1000 наблюдений с 10 столбцами, в каждом столбце 30 пропущенных значений, но в каждой строке только один пропуск – это значит, что функция `na.omit()`, примененная к такому набору, исключит 300 строк, что составляет 30 % всех наблюдений.

Есть смысл заполнить пропуски подходящими значениями. Для этого удобно использовать пакет “Hmisc”:

```
install.packages('Hmisc', dependencies = TRUE)
library(Hmisc)
```

Используем для заполнения пропусков следующий код:

```
# среднее в исходных данных age
mean(df_insurance$age)
# добавить пропуски
df_insurance_NA[c(100, 345, 678, 1000),1] <- c(NA, NA, NA, NA)
# среднее с игнорированием пропусков
mean(df_insurance_NA$age, na.rm = TRUE) # среднее с игнорированием пропуска
# заменить пропуски средним
df_insurance_NA$age <- impute(df_insurance_NA$age, mean)
# среднее для заполненных пропусков
mean(df_insurance_NA$age)
# восстановить пропуски
df_insurance_NA[c(100, 345, 678, 1000),1] <- c(NA, NA, NA, NA)
# заполнить пропуски медианой
df_insurance_NA$age <- impute(df_insurance_NA$age, median)
# среднее для заполненных пропусков
mean(df_insurance_NA$age)
# восстановить пропуски
df_insurance_NA[c(100, 345, 678, 1000),1] <- c(NA, NA, NA, NA)
# заменить пропуски заданным числом
df_insurance_NA$age <- impute(df_insurance_NA$age, 26)
# среднее для заполненных пропусков
mean(df_insurance_NA$age)
```

На рис. 3.5 показано, как влияет на расчет среднего заполнение пропусков различными значениями.

```
> df_insurance_NA <- df_insurance
> # среднее в исходных данных age
> mean(df_insurance$age)
[1] 39.20703
> # добавить пропуски
> df_insurance_NA[c(100, 345, 678, 1000),1] <- c(NA, NA, NA, NA)
> # среднее с игнорированием пропусков
> mean(df_insurance_NA$age, na.rm = TRUE) # среднее с игнорированием пропуска
[1] 39.18741
> # заменить пропуски средним
> df_insurance_NA$age<-impute(df_insurance_NA$age, mean)
> # среднее для заполненных пропусков
> mean(df_insurance_NA$age)
[1] 39.18741
> # восстановить пропуски
> df_insurance_NA[c(100, 345, 678, 1000),1] <- c(NA, NA, NA, NA)
> # заполнить пропуски медианой
> df_insurance_NA$age<-impute(df_insurance_NA$age, median)
> # среднее для заполненных пропусков
> mean(df_insurance_NA$age)
[1] 39.18685
> # восстановить пропуски
> df_insurance_NA[c(100, 345, 678, 1000),1] <- c(NA, NA, NA, NA)
> # заменить пропуски заданным числом
> df_insurance_NA$age<-impute(df_insurance_NA$age, 26)
> # среднее для заполненных пропусков
> mean(df_insurance_NA$age)
[1] 39.14798
```

Рис. 3.5. Результат выполнения кода по заполнению пропусков

Функция `summary()` выводит сводку по параметрам описательной статистики для всех переменных набора. Представим в виде факторов переменные `sex`, `smoker` и `region` и применим эту функцию к набору данных `df_insurance` (рис. 3.6). Для количественных переменных будут рассчитаны максимальное и минимальное значения, среднее и три квартиля (второй квартиль – это медиана), для категориальных данных будет рассчитана абсолютная частота каждого уровня (сколько раз в наблюдениях встречается каждое значение признака).

```
> df_insurance$sex <- as.factor(df_insurance$sex)
> df_insurance$smoker <- as.factor(df_insurance$smoker)
> df_insurance$region <- as.factor(df_insurance$region)
> summary(df_insurance)
```

age	sex	bmi	children	smoker	region	charges
Min. :18.00	female:662	Min. :15.96	Min. :0.000	no :1064	northeast:324	Min. : 1122
1st Qu.:27.00	male :676	1st Qu.:26.30	1st Qu.:0.000	yes: 274	northwest:325	1st Qu.: 4740
Median :39.00		Median :30.40	Median :1.000		southeast:364	Median : 9382
Mean :39.21		Mean :30.66	Mean :1.095		southwest:325	Mean :13270
3rd Qu.:51.00		3rd Qu.:34.69	3rd Qu.:2.000			3rd Qu.:16640
Max. :64.00		Max. :53.13	Max. :5.000			Max. : 63770

```
> |
```

Рис. 3.6. Сводка по параметрам описательной статистики для набора `df_insurance`

Иногда параметры описательной статистики нужно рассчитать по группам. Например, вычислить среднее значение медицинских расходов (`charges`) отдельно для курящих и для некурящих застрахованных. Можно применить функцию `mean()` к данным отобраным по условию (рис. 3.7).

```

> mean(df_insurance$charges[df_insurance$smoker == 'no'])
[1] 8434.268
> mean(df_insurance$charges[df_insurance$smoker == 'yes'])
[1] 32050.23

```

Рис. 3.7. Расчет среднего значения для срезов данных

Но для группировки данных по значениям категориальной переменной можно использовать пакет “dplyr”, в котором есть функция группировки `group_by()`:

```

install.packages('dplyr', dependencies = TRUE)
library(dplyr)

```

В сочетании с функцией `summarize()`, в которую передаются результаты расчета функций над сгруппированными данными, мы можем получить нужную таблицу статистических параметров.

Для набора данных `df_insurance` провели группировку по категориальной переменной `smoker` и вычислили для медицинских расходов (`charges`) среднее, медиану и дисперсию, а также средний возраст в каждой группе (рис. 3.8):

```

df_insurance %>%
  group_by(smoker) %>%
  summarize(
    avg_charges = mean(charges),
    median_charges = median(charges),
    var_charges = var(charges),
    avg_age = mean(age),
    count_values = n()
  )

```

```

> df_insurance %>%
+   group_by(smoker) %>%
+   summarize{
+     avg_charges = mean(charges),
+     median_charges = median(charges),
+     var_charges = var(charges),
+     avg_age = mean(age),
+     count_values = n()
+   }
# A tibble: 2 x 6
  smoker avg_charges median_charges var_charges avg_age count_values
  <chr>      <dbl>         <dbl>      <dbl>    <dbl>      <int>
1 no         8434.           7345.    35925420.    39.4         1064
2 yes        32050.          34456.  133207311.    38.5          274

```

Рис. 3.8. Вычисление параметров описательной статистики для сгруппированных данных

Как видим, средний возраст для курящих застрахованных и некурящих практически одинаков, а вот расходы отличаются значительно. Количество наблюдений соответствующих каждому уровню категориальной переменной подсчитано с помощью функции `n()`.

## 4. Разведочный анализ данных

Разведочный анализ данных основан на построении визуализаций. Рассмотрим высокоуровневые функции построения графиков (табл. 4.1), доступные без подключения специализированных пакетов для построения визуализаций.

Таблица 4.1

Функции для построения графиков

Функция	Описание
plot()	Общая функция для построения графиков. В зависимости от передаваемых данных и настройки параметров, определяемая ей визуализация может быть и точечным графиком, и столбчатой диаграммой
boxplot()	График «ящик с усами»
hist()	Гистограмма
barplot()	Столбчатая диаграмма
scatterplot()	Диаграмма рассеяния (точечная)
pie()	Круговая диаграмма

Базовые управляющие параметры для plot() представлены в табл. 4.2.

Таблица 4.2

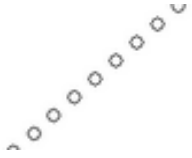



Базовые параметры для plot()

Опция	Описание
main	Заголовок
xlab, ylab	Подписи осей
xlim, ylim	Предельные значения для каждой из осей, передаются как вектор с двумя элементами, например c(2, 7). По умолчанию имеют значение NULL, и тогда пределы значений по осям определяются исходя из данных автоматически

Тип графического отображения, определяемый параметром type, представлен в табл. 4.3.

Таблица 4.3

**Возможные значения для type (список не исчерпывающий)**

Значение	Описание	Визуализация
p	Точки (по умолчанию)	
l	Линия	
b	Точки, соединенные линией	
o	Точки, по которым проходит линия	

В зависимости от передаваемых данных функция `plot()` определяет различные визуализации (табл. 4.4).

Таблица 4.4

**Визуализации на основе функции `plot()` в зависимости от данных**

Функция и аргументы	Построенный график
<code>plot(x, y)</code>	Диаграмма рассеяния числовых векторов <code>x</code> и <code>y</code> (рис. 4.1)
<code>plot(factor)</code>	График фактора (рис. 4.2)
<code>plot(factor, y)</code>	График «ящик с усами» (рис. 4.3)
<code>plot(data_frame)</code>	График корреляции всех столбцов фрейма данных (более двух столбцов)

Построим различные варианты визуализаций для набора данных `df_insurance` (важно, чтобы все символьные переменные были определены как факторы):

```
# Точечный график
plot(df_insurance$bmi, df_insurance$charges, main = "Медицинские расходы",
xlab = "Индекс массы тела", ylab = "Расходы")
# Столбчатая диаграмма для фактора
plot(df_insurance$region, main = "Где проживает застрахованный", xlab =
"Район", ylab = "Количество застрахованных")
```



```
# Медицинские расходы в зависимости от района проживания
plot(df_insurance$region, df_insurance$charges, main = "Медицинские
расходы", xlab = "Район", ylab = "Расходы")
```



Рис. 4.1. Точечный график

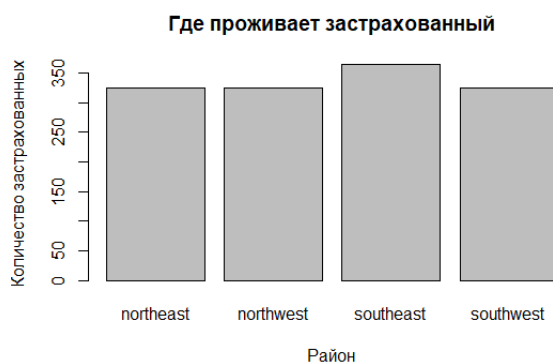


Рис. 4.2. Столбчатая диаграмма

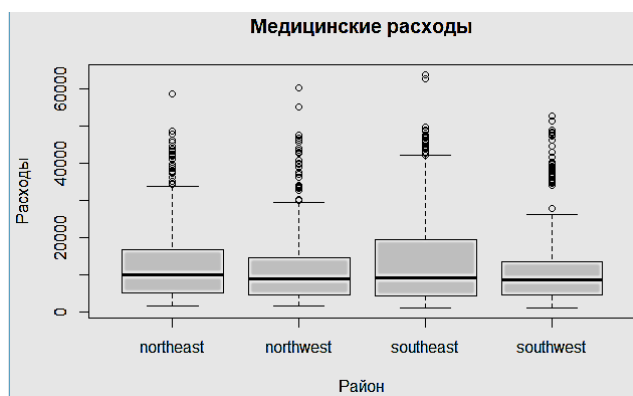


Рис. 4.3. «Ящик с усами»

Рассмотрим, какие параметры можно задать для графических функций (нужно уточнять в справке, есть ли у конкретной функции среди задаваемых параметров, приведенные ниже) [2].

Параметры, связанные с размером текста и графики, обозначаются `cex` (`character extension`, увеличение символов), приведены в табл. 4.5.

Размер текста и символов

Опция	Описание
<code>sex</code>	Число, указывающее величину, на которую следует масштабировать отображаемый текст и символы относительно значения по умолчанию: 1 = по умолчанию, 1,5 на 50 % больше, 0,5 на 50 % меньше и т.д.
<code>sex.axis</code>	Увеличение аннотации оси относительно <code>sex</code>
<code>sex.lab</code>	Увеличение меток x и y относительно <code>sex</code>
<code>sex.main</code>	Увеличение заголовков относительно <code>sex</code>

Тип маркера определяет параметр `pch` (**p**lotting **ch**aracter, символ изображения), соответствие числового значения параметра и вида маркера представлены на рис. 4.4.

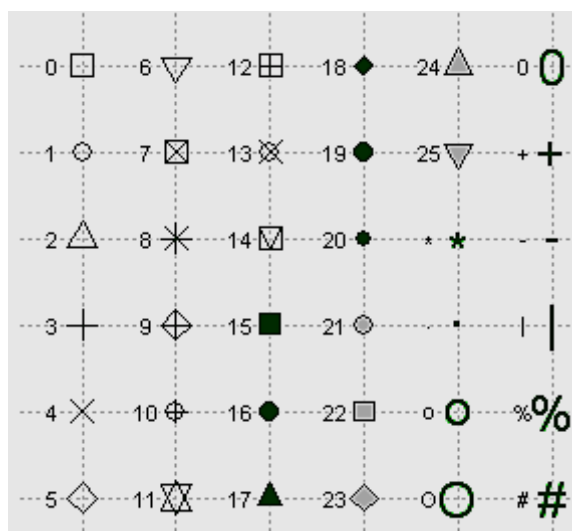


Рис 4.4. Типы маркеров для отображения точек на графике

Параметры, определяющие линии графика: `lty` – тип линии (рис. 4.5) и `lwd` – толщина линии.

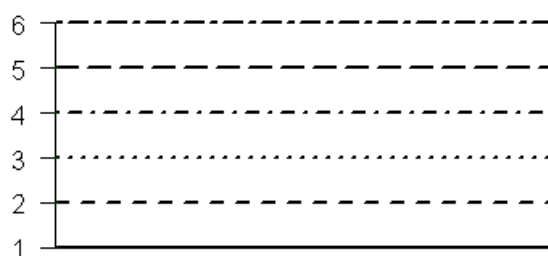


Рис. 4.5. Типы линий для графика

Параметры, определяющие поля и размер графика, приведены в табл. 4.6.

Поля и размер графика

Опция	Описание
mar	Числовой вектор, обозначающий размер поля с (нижний, левый, верхний, правый), в строках по умолчанию = с (5, 4, 4, 2) + 0,1
mai	Числовой вектор, обозначающий размер поля с (нижний, левый, верхний, правый) в дюймах
pin	Размеры участка (ширина, высота) в дюймах

Построим точечный график зависимости медицинских расходов от возраста застрахованного, точки графика изобразим красным цветом для женщин, синим – для мужчин, и добавим легенду на график (рис. 4.6).

```
plot(df_insurance$age, df_insurance$charges,
     col = ifelse(df_insurance$sex == "female", "red", "blue"),
     main = "Медицинские расходы",
     xlab = "Возраст застрахованного", ylab = "Расходы"
)
legend("topleft", cex = .7, pch = 1, col = c("red", "blue"), c('женщины', 'мужчины'))
```

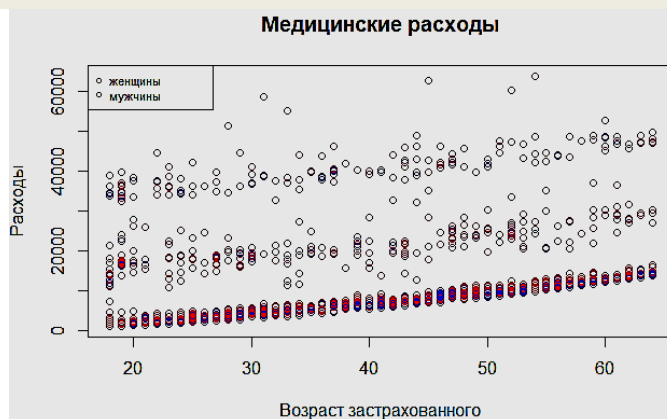


Рис. 4.6. Точечный график с легендой

Для построения круговой и столбчатой диаграммы по данным категориального признака `region` в наборе данных `df_insurance` составим таблицу частот (рис. 4.7):

```
# Создадим таблицу частот для region
frequency_region <- data.frame(table(df_insurance$region))

# Переименуем первый столбец таблицы
colnames(frequency_region)[1] <- "region"

# Вывод таблицы на экран
frequency_region
```

	region	Freq
1	northeast	324
2	northwest	325
3	southeast	364
4	southwest	325

Рис. 4.7. Таблица частот

Построим столбчатую (рис. 4.8) и круговую (рис. 4.9) диаграммы:

```
# Столбчатая диаграмма по данным frequency_region
par(mar = c(5, 10, 4, 2)) # изменим поля графика
barplot(height=frequency_region$Freq, names.arg=frequency_region$region,
        horiz = TRUE, las = 1,
        xlab = "Количество застрахованных",
        main = "Распределение застрахованных по районам")
# Круговая диаграмма по данным frequency_region
par(mar = c(2, 4, 4, 2)) # изменим поля графика
pie(frequency_region$Freq, labels=frequency_region$region,
    main = "Распределение застрахованных по районам")
```



Рис. 4.8. Столбчатая диаграмма

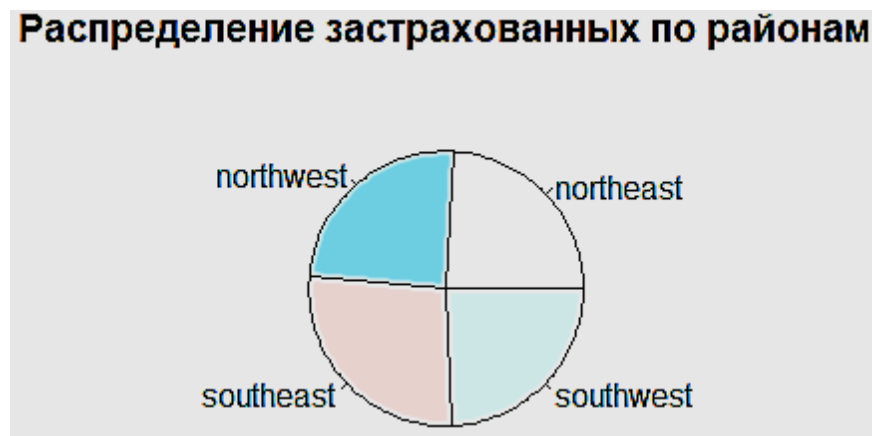


Рис. 4.9. Круговая диаграмма

Если необходимо, чтобы столбчатая диаграмма отображала уровни категорий по возрастанию или убыванию частоты, то данные необходимо упорядочить, прежде чем строить визуализацию:

```
frequency_region<-frequency_region[order(-frequency_region$Freq,  
decreasing = TRUE),]
```

Построим «ящик с усами» (рис. 4.10, 4.11) и гистограмму (рис. 4.12, 4.13):

```
# "Ящик с усами" по переменной bmi  
par(mar = c(5, 4, 4, 2))# изменим поля графика  
boxplot(df_insurance$bmi, main = "Индекс массы тела")
```

```
# "Ящик с усами" по переменной charges с группировкой по smoker  
boxplot(charges~smoker, df_insurance,  
xlab = "Наличие вредной привычки",  
ylab = "Медицинские расходы",  
main = "Медицинские расходы для курящих/некурящих")
```

```
# Гистограмма по переменной charges с 40 столбиками  
hist(df_insurance$charges, breaks=40,  
xlab = "Расходы",  
ylab = "Частота",  
main = "Медицинские расходы")
```

```
# Гистограмма по переменной charges с вероятностью по оси Y  
hist(df_insurance$charges, prob=TRUE,  
xlab = "Расходы",  
ylab = "Вероятность",  
main = "Медицинские расходы")
```

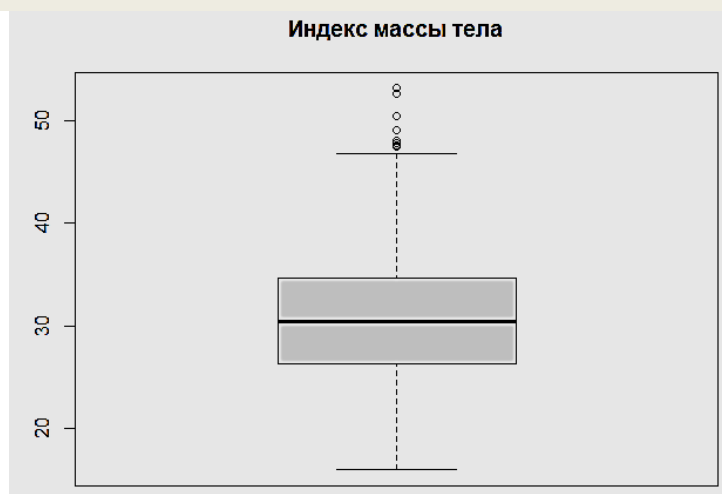


Рис. 4.10. «Ящик с усами» для количественной переменной

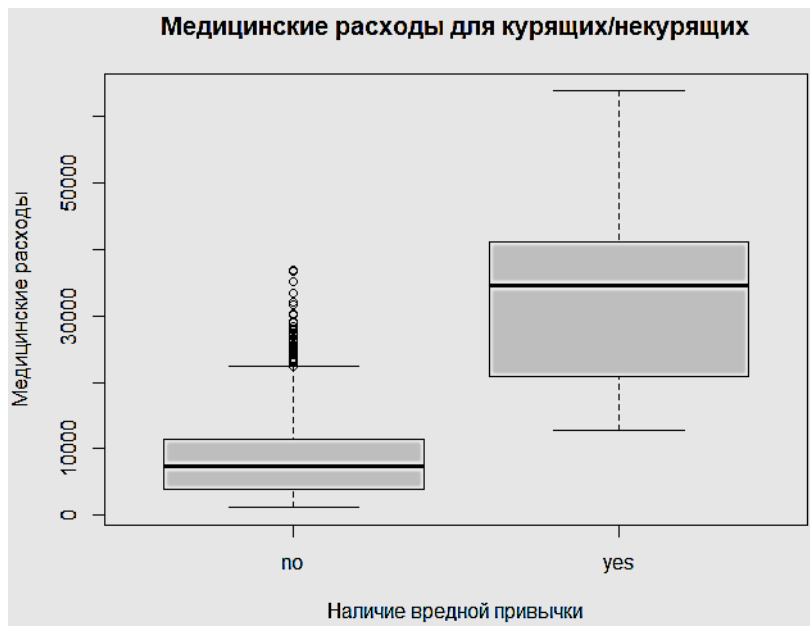


Рис. 4.11. «Ящик с усами» для количественной переменной с группировкой по категориальной переменной

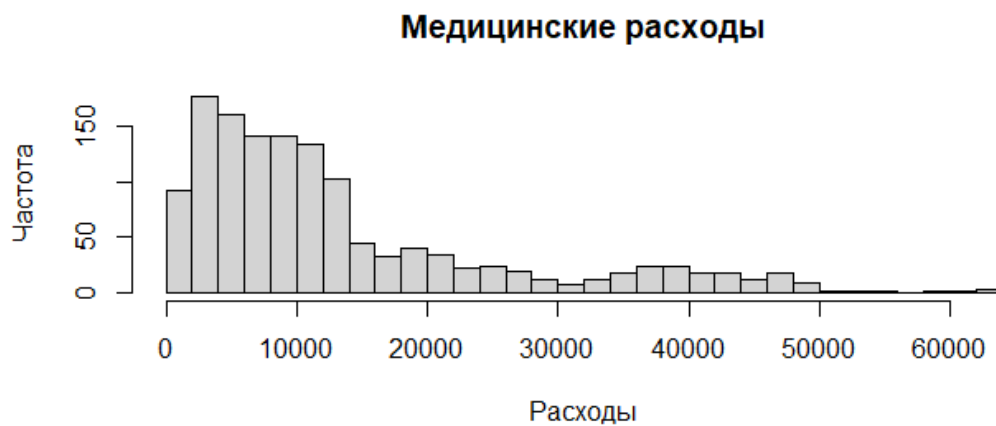


Рис. 4.12. Гистограмма для количественной переменной (частота)

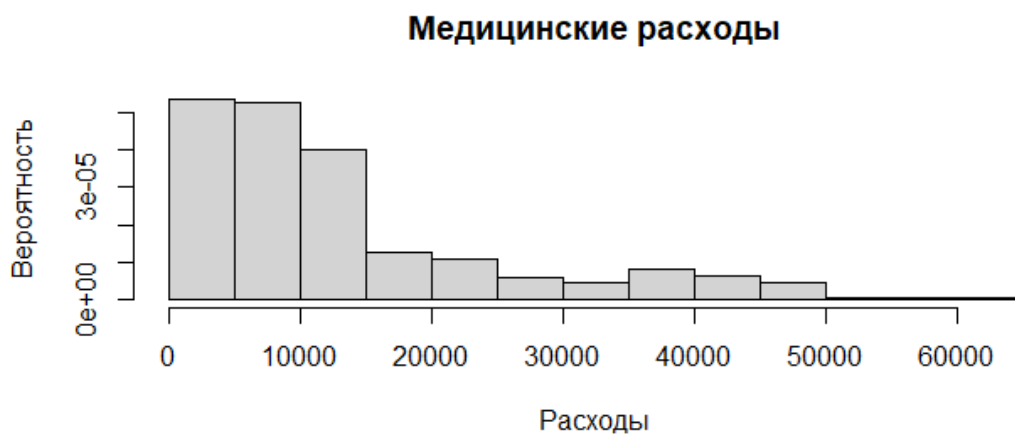


Рис. 4.13. Гистограмма для количественной переменной (вероятность)

## 5. Проверка статистических гипотез

На основе имеющихся данных можно сформулировать гипотезы, подлежащие проверке. Формулируются две гипотезы: нулевая  $H_0$  и альтернативная  $H_1$ .

Нулевая гипотеза – это некоторое утверждение о параметре(ах) генеральной совокупности или распределении. Обычно нулевая гипотеза описывает ситуацию отсутствия различий, равенстве какого-то параметра нулю и т.п. Альтернативная гипотеза – это утверждение, опровергающее нулевую гипотезу, именно альтернативная гипотеза содержит исследовательский вопрос. Альтернативная гипотеза формулируется, но не проверяется.

Важно понимать, что никакие экспериментальные данные не могут подтвердить ни одну из гипотез. Данные могут лишь не противоречить какой-либо гипотезе или показывать крайне маловероятные результаты в предположении, что гипотеза верна и тогда ее обычно отклоняют.

Проверке подлежит именно нулевая гипотеза. Исходим из начального предположения, что  $H_0$  истинна, и пытаемся это подтвердить, а в случае неудачи придерживаемся альтернативного варианта  $H_1$ .

Очень важно понимать, можно ли на основе данных проверить исследовательский вопрос.

Например, на основе данных набора `df_insurance` можно сформулировать несколько исследовательских вопросов:

- проверить нулевую гипотезу о равенстве средних медицинских расходов для всех застрахованных независимо от наличия вредной привычки какому-то конкретному значению (альтернативная гипотеза: средние расходы на медицинское обслуживание больше/меньше этого значения);

- проверить нулевую гипотезу о равенстве средних медицинских расходов для курящих/некурящих какому-то конкретному значению (альтернативная гипотеза: средние расходы на медицинское обслуживание для курящих/некурящих больше/меньше этого значения);

- проверить нулевую гипотезу об отсутствии влияния вредной привычки на величину медицинских расходов, т.е. средние значения медицинских расходов для курящих такие же, как и для некурящих (альтернативная гипотеза: средние расходы на медицинское обслуживание для курящих отличны от средних расходов для некурящих).

Но проверить гипотезу об отсутствии различий в средних медицинских расходах тех, кто был застрахован, и тех, кто обходится без ме-

дицинской страховки, по имеющимся данным невозможно, так как известны медицинские расходы только для застрахованных.

Проверить гипотезу об отсутствии различий в средних медицинских расходах тех, кто никогда не имел вредной привычки, и тех, кто бросил курить, по имеющимся данным невозможно, так как известны данные о некурящих в текущий момент времени и нет данных об отсутствии этой привычки или ее наличии в прошлом.

Всегда нужно определять ограничения, которые накладывают на исследования имеющиеся данные. Но это не значит, что такие исследовательские вопросы бесполезны. То, что они выходят за рамки имеющихся данных, не означает отсутствие практической ценности в таких вопросах. Они позволяют понять, какие данные необходимы для проверки таких исследовательских вопросов и обогатить данные из доступных источников или сформировать проект по сбору данных, определив какие показатели необходимо регистрировать, чтобы решить подобную исследовательскую задачу в будущем.

В R для тестирования гипотез о выборках будем использовать функцию `t.test()`, аргументы которой представлены в табл. 5.1.

Таблица 5.1

#### Аргументы `t.test()`

Аргумент	Описание
<code>x</code>	Числовой вектор
<code>y</code>	Числовой вектор (необязательный аргумент, используется при двухвыборочном тесте)
<code>alternative</code>	Параметр определяющий, какая альтернативная гипотеза тестируется: "two.sided" – двусторонняя, "greater" (больше, т.е. правосторонняя) или "less" (меньше, т.е. левосторонняя). По умолчанию значение "two.sided"
<code>mu</code>	Число, показывающее истинное значение среднего (или разницу в средних, если выполняется двухвыборочный тест)
<code>paired</code>	Логическое условие выбора двухвыборочного теста ( <code>paired = TRUE</code> ) или одновыборочного теста ( <code>paired = FALSE</code> ). Значение <code>FALSE</code> по умолчанию
<code>var.equal</code>	Логическая переменная, указывающая, следует ли рассматривать две дисперсии как равные. Если <code>var.equal = TRUE</code> для оценки дисперсии используется объединенная дисперсия (тест Стьюдента), <code>var.equal = FALSE</code> используется приближение Уэлча (или Саттертуэйта) для степеней свободы



## 5.1. Тестирование гипотез для одной выборки

Пусть компания, которая занимается оформлением полисов добровольного медицинского страхования, рассчитала годовой взнос по страховке из расчета, что средние расходы на медицинское обслуживание составляют 12 000 д.е. Исследовательский вопрос, который можно проверить на основе имеющихся данных о реальных расходах на медицинское обслуживание застрахованных: соответствует ли средний расход ожидаемому.

Альтернативная гипотеза двусторонняя: средние расходы отличаются от значения, использованного для расчета стоимости страховки (те самые 12 000 д.е.).

Альтернативная гипотеза односторонняя: средние расходы больше, чем значение, использованное для расчета стоимости страховки.

Если мы не учитываем никакие входные данные о клиентах при расчете страхового взноса (т.е. стоимости полиса для клиента), т.е. для всех клиентов она стоит одинаково, то ошибка в значении, выбранном для среднего значения расходов на медицинское обслуживание в генеральной совокупности, может разорить страховую компанию.

Если записать гипотезы с использованием математических символов, то для двусторонней альтернативы

$$H_0 : \mu = \mu_0,$$

$$H_1 : \mu \neq \mu_0.$$

А в случае односторонней альтернативы

$$H_0 : \mu = \mu_0,$$

$$H_1 : \mu > \mu_0,$$

где  $\mu_0$  – это выбранное значение 12 000 д.е.

Уровень значимости определим как  $\alpha = 0,05$ . Для тестирования гипотез будем использовать *одновыборочный t-тест*.

В случае двусторонней альтернативы

```
t.test(df_insurance$charges, mu=12000)
```

Результаты тестирования представлены на рис. 5.1.

Статистический вывод: предположение о том, что средние ожидаемые медицинские расходы составляют 12 000 д.е., не подтвердилось на уровне значимости 0,05 (для двусторонней альтернативной гипотезы). Нулевую гипотезу о равенстве средних расходов 12 000 отклоняем ( $p - value < 0,0001302$ ).

```
> t.test(df_insurance$charges, mu=12000)

One Sample t-test

data: df_insurance$charges
t = 3.8374, df = 1337, p-value = 0.0001302
alternative hypothesis: true mean is not equal to 12000
95 percent confidence interval:
 12620.95 13919.89
sample estimates:
mean of x
 13270.42
```

Рис. 5.1. Результаты тестирования (двусторонняя альтернатива)

В случае односторонней альтернативы

```
t.test(df_insurance$charges, mu=12000, alternative = "greater")
```

Результаты тестирования отображены на рис. 5.2.

```
> t.test(df_insurance$charges, mu=12000, alternative = "greater")

One Sample t-test

data: df_insurance$charges
t = 3.8374, df = 1337, p-value = 6.509e-05
alternative hypothesis: true mean is greater than 12000
95 percent confidence interval:
 12725.49      Inf
sample estimates:
mean of x
 13270.42
```

Рис. 5.2. Результаты тестирования (односторонняя альтернатива)

Статистический вывод: средние ожидаемые медицинские расходы превышают 12 000 д.е. с вероятностью 0,95 (для односторонней альтернативной гипотезы). Нулевую гипотезу о равенстве средних расходов 12 000 отклоняем ( $p - value < 6,509 \cdot 10^{-5}$ ).

## 5.2. Тестирование гипотез для двух выборок

Исследовательский вопрос, который можно сформулировать для компании, занимающейся медицинским страхованием: одинаковы ли средние расходы на медицинское обслуживание для курящих и некурящих застрахованных.

Однако хотя выборки отдельно по курящим застрахованным и некурящим независимые и количество наблюдений достаточно большое, чтобы приблизить выборки к нормально распределенной случайной величине, нет оснований полагать, что обе выборки происходят из генеральной совокупности с одинаковыми дисперсиями. В этом случае используют модификацию Уэлча для исходного теста Стьюдента.

Альтернативная гипотеза двусторонняя: средние расходы для двух выборок различны (не одинаковы).

Если записать гипотезы с использованием математических символов, то для двусторонней альтернативы

$$H_0 : \mu_1 = \mu_2 \text{ или } \mu_1 - \mu_2 = 0,$$

$$H_1 : \mu_1 \neq \mu_2 \text{ или } \mu_1 - \mu_2 \neq 0.$$

Уровень значимости определим как  $\alpha = 0,05$ . Используем *двухвыборочный t-тест для независимых выборок* (модификация Уэлча, так как нет оснований предполагать, что распределение у рассматриваемых выборок одинаковое). Результаты тестирования представлены на рис. 5.3.

```
> t.test(charges~smoker,df_insurance)

Welch Two Sample t-test

data:  charges by smoker
t = -32.752, df = 311.85, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -25034.71 -22197.21
sample estimates:
mean in group no mean in group yes
 8434.268          32050.232
```

Рис. 5.3. Результаты тестирования

Можно применять функцию `t.test()` для двух выборок в следующих форматах: `t.test(x, y)`, где `x` и `y` – числовые векторы и в формате `t.test(y~x, data)`, где `y` – количественная переменная, `x` – дихотомическая категориальная переменная, а `data` – набор данных, для которого `x` и `y` – столбцы. Так как `smoker` – дихотомическая категориальная переменная, а `charges` – количественная, то воспользуемся вторым вариантом для обращения к функции `t.test()`:

```
t.test(charges~smoker, df_insurance)
```

Статистический вывод: с вероятностью 95 % средние медицинские расходы для курящих отличаются от средних медицинских расходов для некурящих. Гипотезу о равенстве средних расходов у этих двух групп отклонили ( $p - value < 2,2 \cdot 10^{-16}$ ).

## **6. Лабораторные работы**

### **6.1. Лабораторная работа «Импорт и преобразование данных»**

**Цель работы:** научиться загружать наборы данных, изменять тип данных, делать срезы данных по условию.

**Формируемые знания, умения и навыки:** знать основные типы и структуры данных. Уметь загружать данные, преобразовывать данные. Владеть навыками создания срезов данных.

**Необходимо:**

1. Используя ресурс kaggle: <https://www.kaggle.com/>, выбрать один из наборов данных. Загрузить этот набор в рабочую директорию. Считать данные, определить тип данных.

2. Описать данные набора: какие переменные в нем присутствуют, какой тип данных у этих переменных.

#### **Контрольные вопросы и задания**

1. Как выполняется считывание данных?
2. Какие проблемы при импорте данных могут возникнуть, какие способы их решения могут быть использованы?
3. Есть ли в наборе данных количественная дискретная переменная?
4. Есть ли в данных категориальные переменные? Как преобразовать их в факторы?
5. Есть ли в данных количественные переменные? Какой у них тип?
6. Как построить срез данных по условию? Как провести сортировку набора данных по одному из столбцов?

### **6.2. Лабораторная работа «Описательный анализ данных»**

**Цель работы:** научиться на основе параметров описательных статистик делать выводы о распределении данных.

**Формируемые знания, умения и навыки:** знать основные меры: центральной тенденции, положения, рассеяния. Уметь определять тип данных, вычислять статистические параметры для данных выборки с использованием функций на языке R. Владеть навыками использования описательной статистики для определения характера распределения данных.

**Необходимо:** рассчитать параметры описательной статистики для переменных набора с использованием языка R. Сделать содержательные выводы.

### **Контрольные вопросы и задания**

1. С помощью параметров описательной статистики опишите одну из количественных переменных и одну из категориальных переменных.
2. Есть ли в количественной переменной выбросы? Какое значение является выбросом и почему?
3. По какой категориальной переменной можно группировать данные? Какие группировки представляют интерес и почему?

## **6.3. Лабораторная работа**

### **«Визуализация данных. Разведочный анализ данных»**

**Цель работы:** научиться использовать визуализации для формулирования исследовательских гипотез, проверки предположений относительно распределения данных и зависимостей между показателями.

**Формируемые знания, умения и навыки:** знать основные визуализации для количественных и категориальных данных, уметь использовать для построения визуализаций функции языка R. Уметь анализировать построенные визуализации для описания распределения данных, которое они графически представляют.

**Необходимо:** на том же наборе данных провести построение графиков на языке R.

### **Контрольные вопросы и задания**

1. Какие визуализации построены для количественных переменных? Какие функции на языке R и с какими входными параметрами при этом использовались? Какие особенности распределения данных они позволяют наблюдать?
2. Какие визуализации построены для категориальных переменных? Какие функции на языке R и с какими входными параметрами при этом использовались? Какие особенности распределения данных они позволяют наблюдать?
3. Какие визуализации построены для пар переменных? Какие функции на языке R и с какими входными параметрами при этом использовались? Какие особенности распределения данных они позволяют наблюдать?

## **6.4. Лабораторная работа**

### **«Исследовательский анализ данных»**

**Цель работы:** научиться формулировать исследовательские вопросы для данных об объектах.

**Формируемые знания, умения и навыки:** знать, как на основе исследовательского вопроса формулируется нулевая гипотеза и альтернативная, уметь для выборки данных сформулировать гипотезы для тестирования и определить критерий для тестирования, владеть навыками тестирования гипотез на языке R.

**Необходимо:**

1. На том же наборе данных поставить исследовательские вопросы, сформулировать нулевую и альтернативную гипотезы, выбрать статистический критерий для проверки гипотез.

2. Провести тестирование гипотез с использованием методов на языке R. Сделать вывод по результатам тестирования.

### **Контрольные вопросы и задания**

1. Какие исследовательские задачи можно сформулировать для вашего набора данных?

2. Какие гипотезы нужно протестировать для ответа на исследовательский вопрос? Альтернативная гипотеза, сформулированная вами, является односторонней или двусторонней?

3. Какой статистический критерий необходимо использовать при тестировании гипотез?

4. Какой уровень значимости был выбран? Что позволяет определить уровень значимости при тестировании гипотез? Как уровень значимости влияет на вывод?

5. Какой метод на языке R использовался для тестирования гипотез? Какой содержательный вывод можно сделать по результатам тестирования?

## **Список литературы**

1. Груздев А. В. Прогнозное моделирование в IBM SPSS Statistics, R и Python: метод деревьев решений и случайный лес : руководство. М. : ДМК Пресс, 2018. 642 с.
2. Graphical Parameters. URL: <https://www.statmethods.net> (дата обращения: 11.06.2021).
3. Мастицкий С. Э., Шитиков В. К. Статистический анализ и визуализация данных с помощью R. М. : ДМК Пресс, 2015. 496 с.
4. Роберт И. R в действии. Анализ и визуализация данных в программе R : руководство : пер. с англ. П. А. Волковой. М. : ДМК Пресс, 2014. 588 с.
5. Буховец А. Г., Москалев П. В. Алгоритмы вычислительной статистики в системе R : учеб. пособие. 2-е изд., перераб. и доп. СПб. : Лань, 2015. 160 с.

*Учебное издание*

**Рындина** Светлана Валентиновна

## БАЗОВЫЕ ВОЗМОЖНОСТИ ЯЗЫКА R ДЛЯ АНАЛИЗА ДАННЫХ

Редактор *Е. В. Шмелева*  
Технический редактор *С. В. Денисова*  
Компьютерная верстка *С. В. Денисовой*

Подписано в печать 27.10.2021.  
Формат 60×84<sup>1</sup>/<sub>16</sub>. Усл. печ. л. 3,26.  
Тираж 5. Заказ № 564.

---

Издательство ПГУ  
440026, Пенза, Красная, 40  
Тел.: (8412) 66-60-49, 66-67-77; e-mail: iic@pnzgu.ru





**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

---

**С. В. Рындина**

**БАЗОВЫЕ ВОЗМОЖНОСТИ  
ЯЗЫКА R  
ДЛЯ АНАЛИЗА ДАННЫХ**

**Учебно-методическое пособие**

**ПЕНЗА 2021**