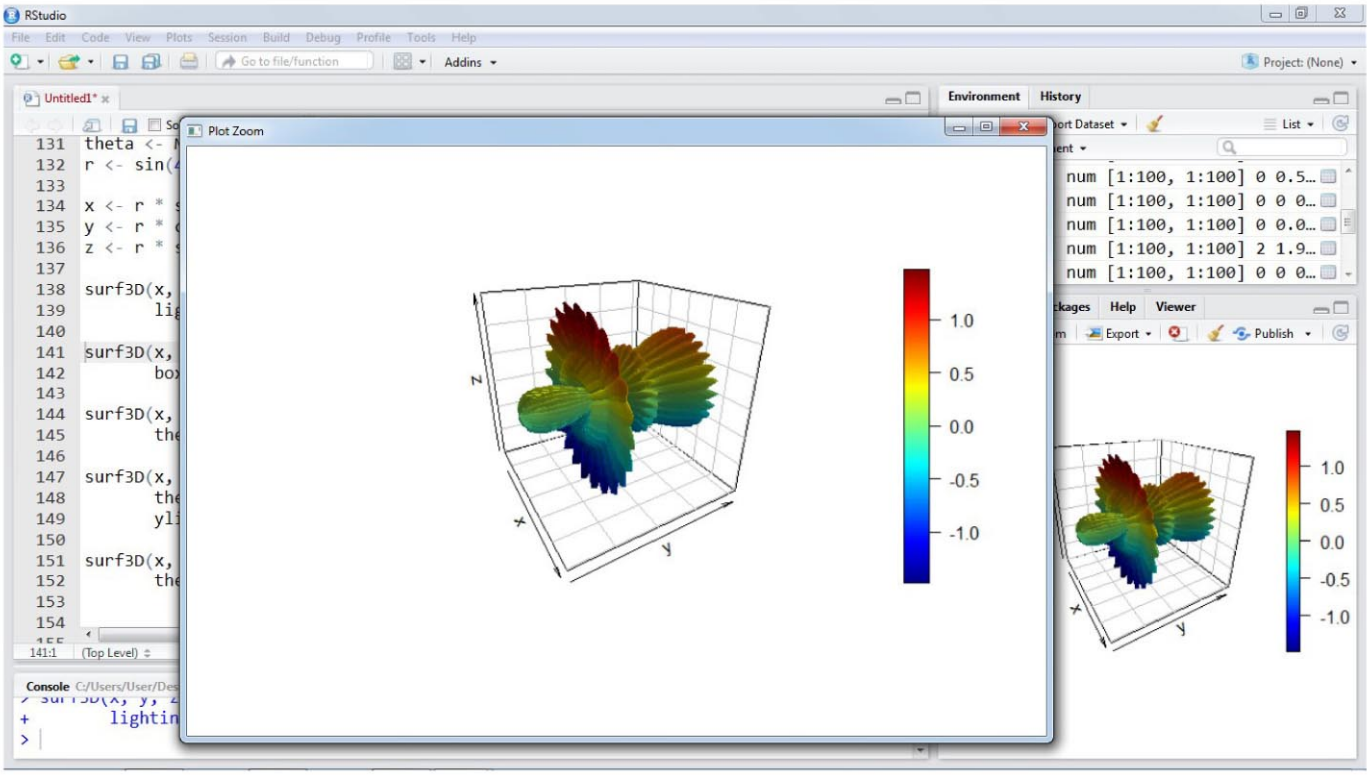
УЧЕБНИК

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Д.В. Берзин, С.А. Зададаев

# ЦИФРОВАЯ МАТЕМАТИКА В R




Издательство Прометей

**УДК 811.111(076.5)**
**ББК 22.1+81.2Англ**
**Б48**

**Авторы:**

***Берзин Д.В.***, кандидат физико-математических наук, доцент департамента математики Финансового университета при Правительстве Российской Федерации;

***Зададаев С.А.***, кандидат физико-математических наук, профессор, руководитель Департамента математики Финансового университета при Правительстве Российской Федерации.

**Рецензенты:**

***Шандра И.Г.***, канд. физ.-мат. наук, профессор департамента математики Финансового университета при Правительстве Российской Федерации;

***Булычев А.В.***, канд.тех.наук, ведущий научный сотрудник Федерального государственного учреждения «Федеральный исследовательский центр «Информатика и управление» Российской академии наук».

**Берзин Д.В.**

Б 48 **Цифровая математика в R:** Учебник (электронное издание) / Д.В. Берзин, С.А. Зададаев. — М.: Прометей, 2023. — 192 с.

ISBN 978-5-00172-464-3

Учебник состоит из 17 компьютерных практикумов, посвященных вычислительным возможностям программной среды R при решении задач математического анализа и линейной алгебры. Соответствует учебным планам дисциплин «Компьютерный практикум» и «Цифровая математика на языке R и Excel», преподаваемых в Финансовом университете при Правительстве Российской Федерации на первом курсе общеэкономических и управленческих специальностей (программы бакалавриата). В учебнике последовательно излагаются основы языка программирования R с постепенным углублением по мере продвижения по осваиваемым навыкам в применении к высшей математике. Для комфортного программирования на R практикумы ориентированы на популярную интегрированную среду разработки RStudio.

Учебник будет полезен студентам, изучающим математический анализ и линейную алгебру и желающим познакомиться с самыми современными вычислительными технологиями. Также он будет полезен тем, кто хочет научиться программировать на языке R и продолжить изучать его применение в статистическом анализе и анализе данных. Учебник может быть интересен студентам магистратуры, аспирантам, научным работникам, учителям, преподавателям университетов.

**ISBN 978-5-00172-464-3**

© Берзин Д.В., Зададаев С.А., 2023
© Издательство «Прометей», 2023

# CONTENTS

Preface . . . . . 6
Competences of the discipline . . . . . 7
1. Introduction to R . . . . . 8
2. User-defined functions . . . . . 22
3. Zeroes and extrema of functions . . . . . 37
4. Definite and improper integrals... . . . . 54
5. Surfaces and level curves . . . . . 61
6. Symbolic differentiation . . . . . 72
7. Data types . . . . . 85
8. Loops . . . . . 101
9. Differential equations . . . . . 114
10. Vectors and matrices . . . . . 126
11. Saving results in R and import/export of data from MS Excel (RStudio) . . . . . 132
12. Vector algebra . . . . . 143
13. Matrix algebra . . . . . 148
14. Matrix equations . . . . . 157
15. Extension of double-arithmetic . . . . . 165
16. Spectral and singular matrix decomposition . . . . . 171
17. Linear optimization problem . . . . . 181
Recommended literature . . . . . 190
Conclusion . . . . . 191

# COMPETENCES OF THE DISCIPLINE "DIGITAL MATHEMATICS IN MS EXCEL AND R"

In the modern paradigm of Russian university education, it is customary to indicate the professional competencies developed by a particular discipline. Following this tradition, we will briefly outline what exactly we will fight for when studying the material of this textbook. The discipline provides necessary tools to form competences listed below.

| | **Competence** | **Learning outcomes (skills and knowledge) and indicators that show competence development** |
|---|---|---|
| 1. | The ability to use application software to solve professional problems. | **Know** the basic methods of obtaining, presenting, storing, and processing data using R software environment and MS Excel;<br>**Be able to** use basic methods of obtaining, presenting, storing, and processing data by means of R and MS Excel;<br>**Possess** the skills in solving standard problems of Calculus and Linear Algebra using the tools of R and MS Excel. |
| 2. | The ability to apply mathematical methods to solve standard theoretical and applied problems, to interpret the obtained mathematical results. | **Know** the computational methods of the main problems of mathematical analysis and linear algebra;<br>**Be able to** use computer technology in the implementation of mathematical methods and models for the description and analysis of applied problems;<br>**Possess** the skills in computational work in R and communication with MS Excel. |
| 3. | The ability to visualize analytical and reporting materials based on the results of the work performed. | **Know** the basic tools of visualizing quantitative data in R;<br>**Be able to** use computer technologies for data presentation and graphical visualization of the results of applying mathematical methods and models to describe and analyze various applied problems;<br>**Possess** the skills to work in RStudio in terms of visualizing quantitative data. |

# 1. INTRODUCTION TO R

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used by statisticians and data miners for developing statistical software and data analysis. Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity; as of March 2022, R ranks 11th in the TIOBE index, which ranks programming languages in popularity among users.

A GNU package, source code for the R software environment is written primarily in C, FORTRAN, and R itself and is freely available under the GNU General Public License. Pre-compiled binary versions are provided for various operating systems. Although R has a command-line interface, there are several graphical user interfaces (GUI), such as RStudio, an integrated development environment.

RStudio is an integrated development environment (IDE) for R, a programming language for statistical computing and graphics. Loosely speaking, RStudio is a "shell" for R.

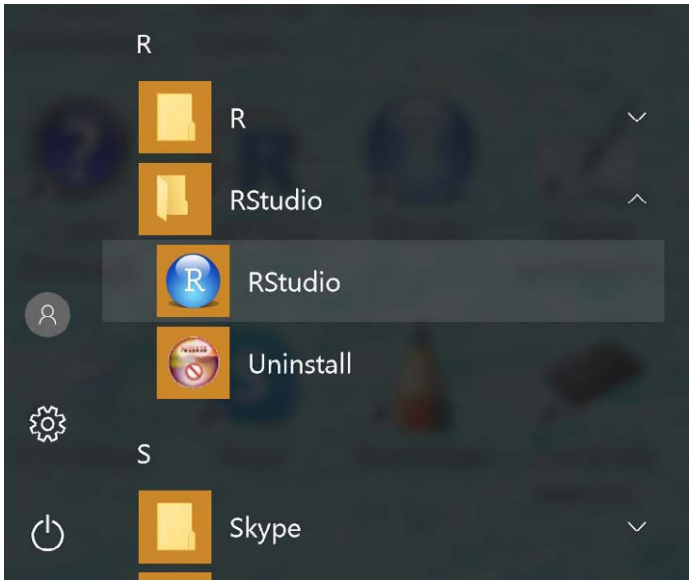
## Requirements

To work with this data (and the subsequent R training materials), you must have two programs installed on your computer: R programming language interpreter and the RStudio «shell». The procedure for the official free download and installation of these programs is described in the instructions (see the appendix at the end of this chapter).

We highly recommend installing these two programs on your personal computer (laptop, tablet, etc.), because together with tens of thousands of other specialists around the world, we consider this computing tool predominant now.

## Starting RStudio

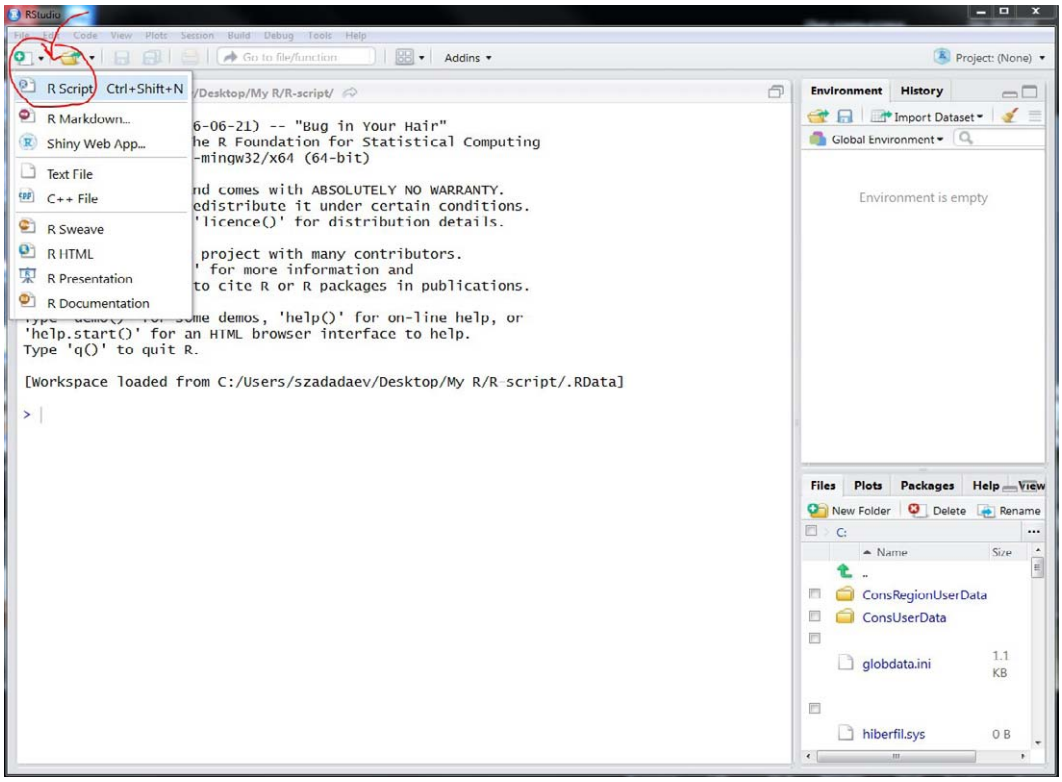
To work with R, we will use the user-friendly RStudio shell, which interacts with the R interpreter. You should start RStudio by clicking the corresponding shortcut.




*Note.* If text on the pictures is difficult to read, use the on-screen zoom by rotating

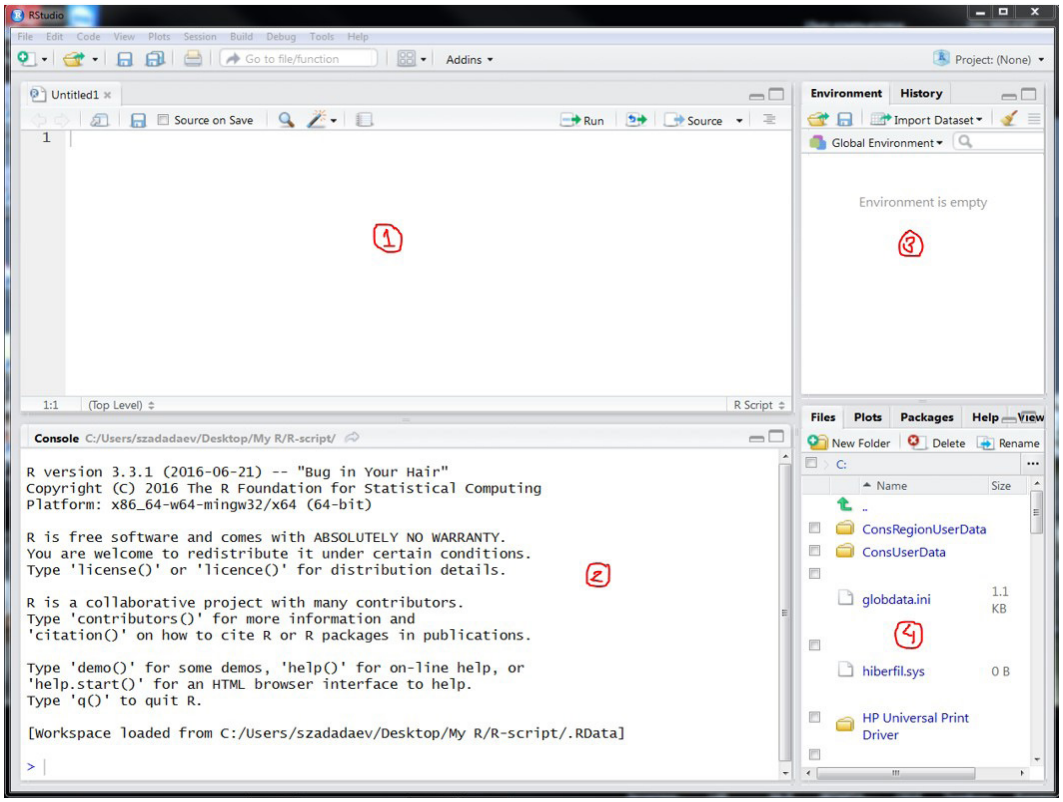
the mouse wheel while pressing the «Ctrl» button.

As a result, the program window opens up. Now, create a new document with the future program code (*script* or *source code*), by clicking the mouse on the upper-left menu item «R script» or pressing the key combination «Ctrl+Shift+N»:




After that, the RStudio working area will split into four windows ("panes") as follows

:




1. *Upper-left*: the window of the worksheet with the future program code in the

R language (source code window). Here we will enter the R commands and run them.

*2. Bottom-left:* «R console» — a window with the step-by-step compilation (sequence) of R commands. On this sheet, we will see the results of our programs.

*3. Upper-right:* the window of the history of R and the current values of objects.

*4. Bottom-right:* multi-function window is used for navigation / graphics / packages, etc.

## Comment operator

There is a programmer's joke that the comment operator is the most frequently used in any programming language. The comment statement is a transcript of each command used to make it clearer. It is the explanations we leave that make the program text interactive in some sense: these remarks refer to the commands we entered earlier.

In the R language, the operator "#" ("hash" symbol) — is a vivid example: the R compiler will not recognize as code anything following the symbol «#» in the current line. For example, the string

```
# Hello, world! :)
```

will be a good page title, but the greeting itself will not be executed in the R code.

## Loading packages

We could have already written something more meaningful in the program text field (upper-left window), but when we first start RStudio, we need to download the current packages of the procedures and functions used in R from the Internet. There are two ways to do this.

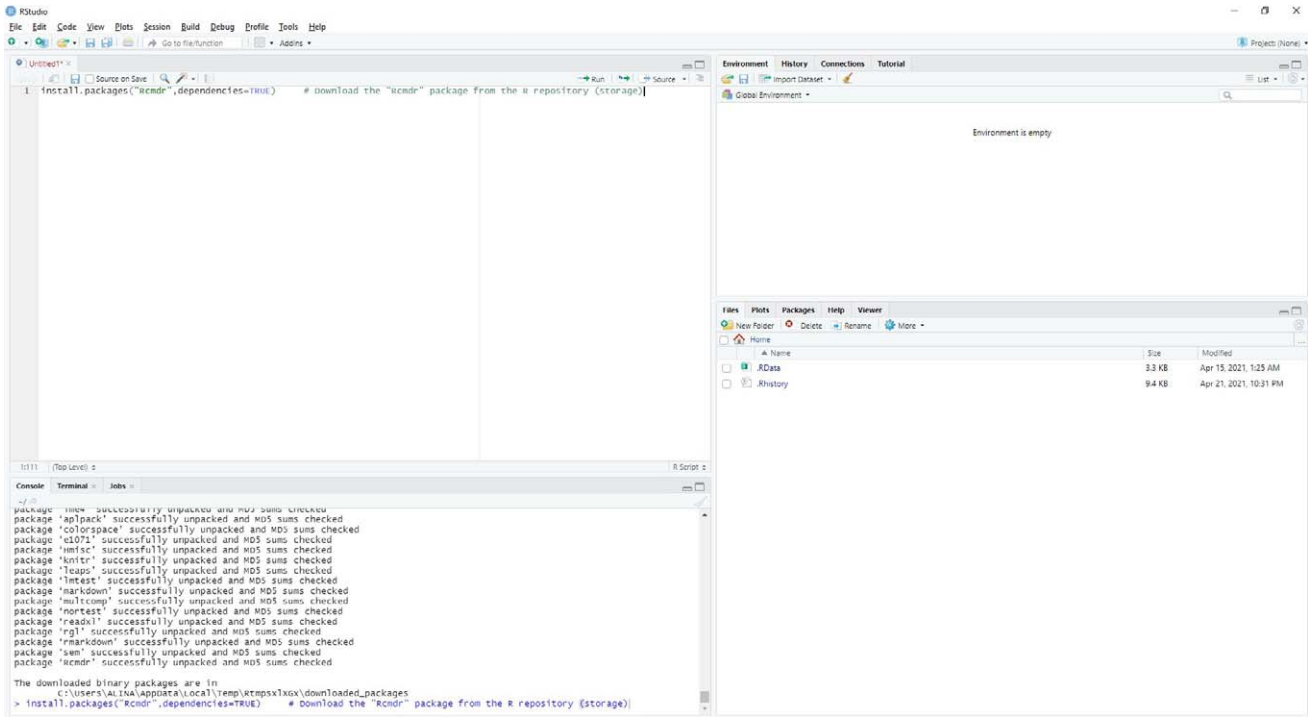
The first method is to execute a special command to load the required package into R. For example, the necessary initial download of the main database of software packages in R is the download of the package "*Rcmdr*". Enter the command in the first line of the worksheet (you can simply copy and paste the following text):

```
install.packages("Rcmdr",dependencies=TRUE)    # Download the "Rcmdr" package from the R repository (storage)"
```

*Note.* It is important to know that in the R language, lowercase and uppercase letters matter, that is, «a» and «A» are different! Here «Rcmdr» is not equal to «rcmdr»!

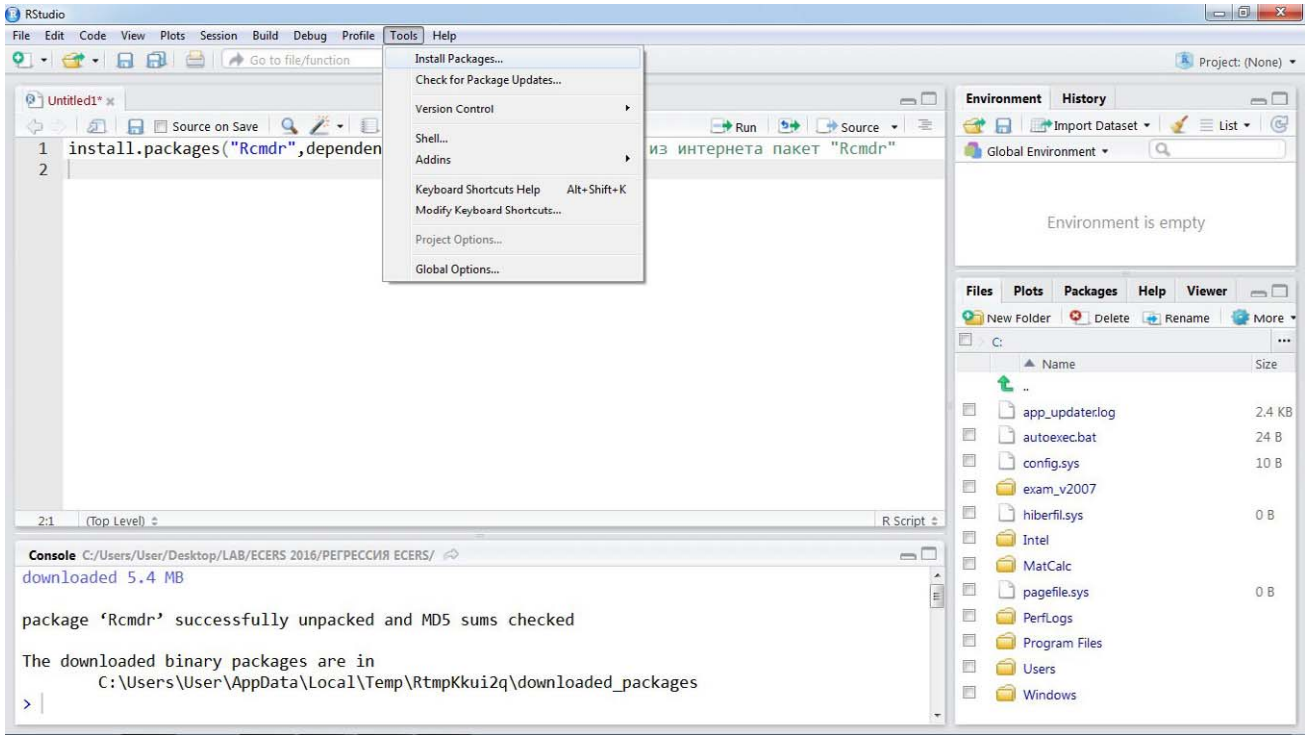
If you press "Enter" afterwards, the cursor will move to a new line and nothing else happens. But if you press the key combination «Ctrl+Enter», the cursor will also move to a new line, and the code of the previous line will be compiled, leading to execution of the specified command. You will notice how R loads the necessary modules from the Internet, it can take a few minutes.

As a result, we will see a message about the successful installation:

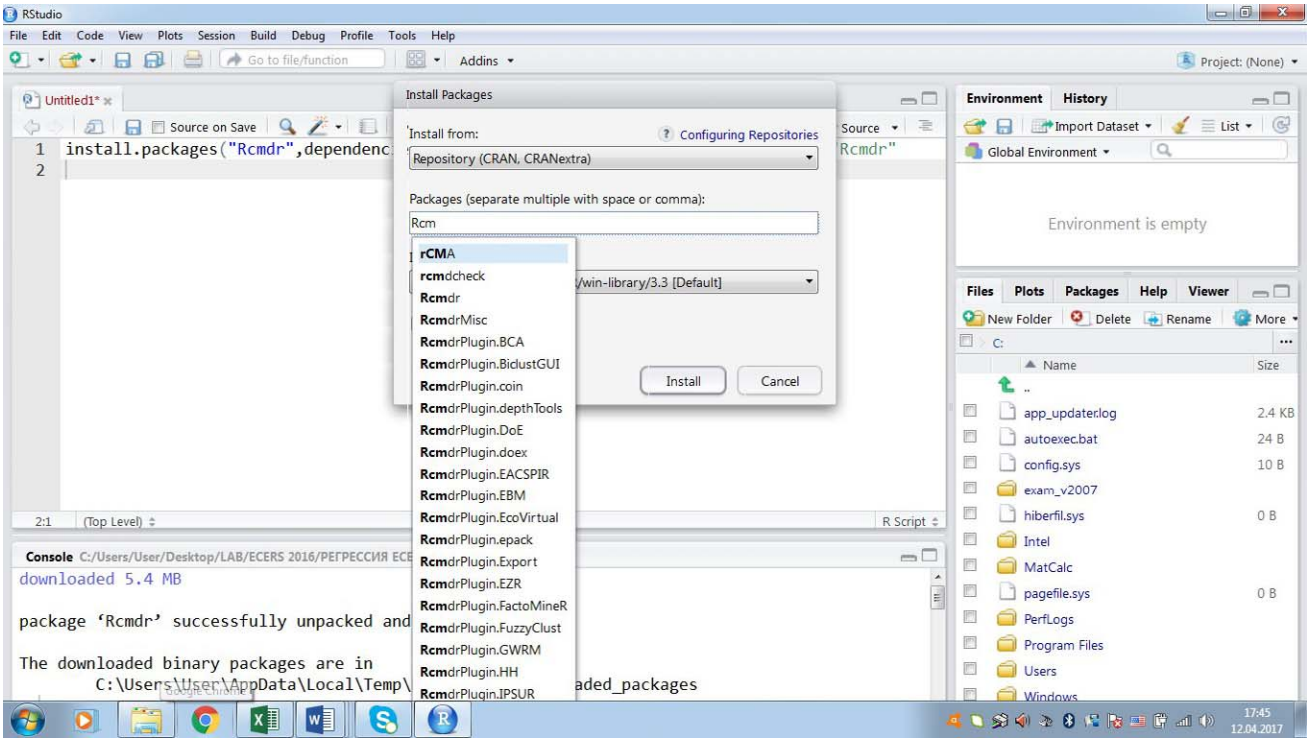



This installation must be done once for each computer. If additional packages are used, then the same procedure should be done.

The second way is to open the interactive packages loading window by choosing Tools –> Install Packages...



In the window that appears, you need to enter the necessary package names (if several at once, then separate them with space) and click *Install*:




However, even when the package is downloaded either way, the access to its procedures and functions during the current session (in the current RStudio session) remains unavailable until we connect (*activate*) this package in the current session using the *library*() command. This is done in R to save RAM (random-access memory of the computer).

For example, we want to use the advanced features of matrix algebra using the *«Matrix»* package. In this case we must download this package once using the following command:

```
install.packages("Matrix", dependencies=TRUE)    # Download the "Matrix" package from the R repository
```

Whenever we are going to use this package, we need to run the activation command at the beginning of the session in this way:

```
library(Matrix)    # Activate the "Matrix" package loaded in R
```

**Example 1.1.** RStudio has built-in data concerning the dependence of the cars speed on a braking distance (Ford study, 1920). This data is stored in the "cars" variable. To understand how the data about the velocities and distances of the braking paths are stored in *cars*, you need to run the *glimpse(cars)* command, which is included in the *«dplyr»* package.

**Solution.** Technically, we need to do three small operations:

1. Download the *«dplyr»* package to your computer (this is a very convenient and popular package for tampering with various data):

```
install.packages("dplyr",dependencies=TRUE)    # Download the "dplyr" package from the R repository
```

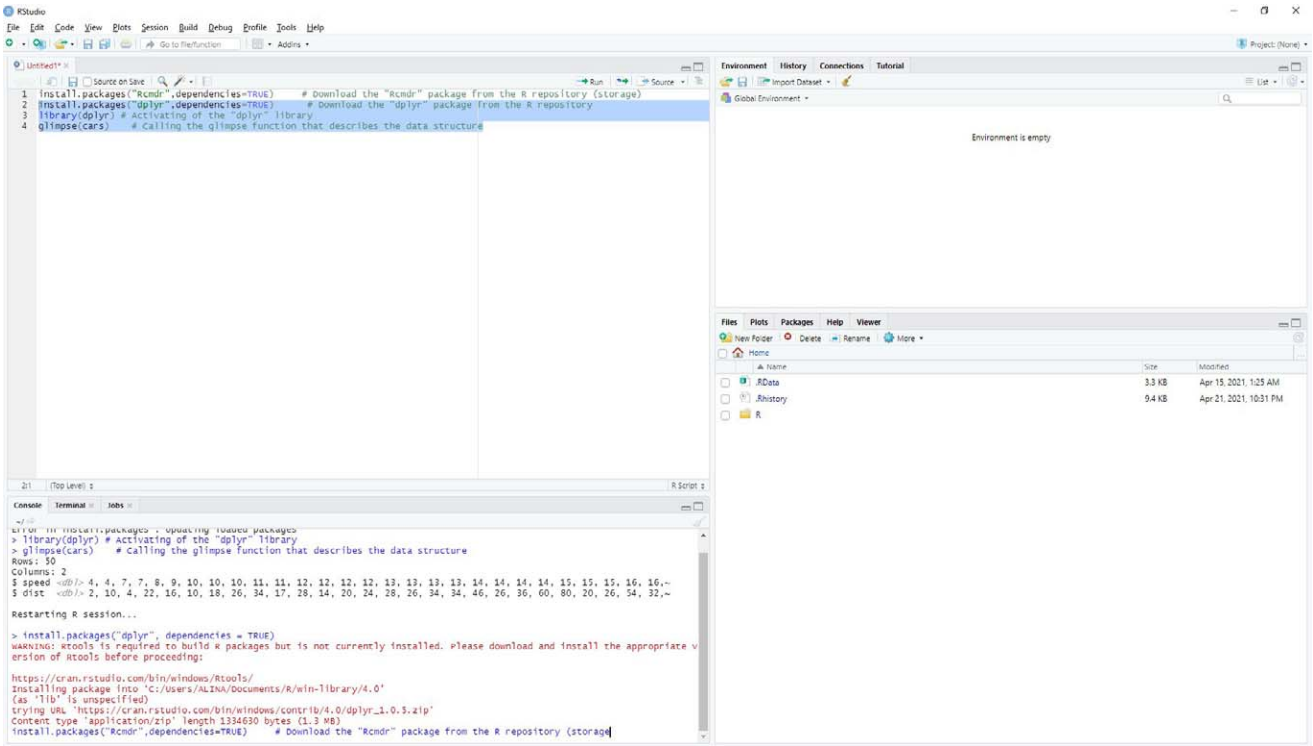
2. Activate this library during the current session:

```
library(dplyr)      # Activating of the “dplyr” library
```

3. Execute the appropriate function from the package:

```
glimpse(cars)    # Calling the glimpse function that describes the data structure
```

After that, we select these three typed lines with the left mouse button and run the code for compilation by clicking the *Run* (or pressing *«Ctrl+Enter»* combination):




As a result, we get full information about the cars variable in the console window. We are going to return to this report later.

## R as calculator

Here are the main mathematical functions that, having been typed as a code, result in calculated values:

```
sin(2); cos(2); tan(2); asin(0.5); acos(0.5); atan(2); log(0.3); log(1024, 2); exp(2); log10(4);
sinh(2); cosh(2); tanh(2); asinh(2); acosh(2); atanh(1/2);
```

*Note:* here and further, arbitrary numbers are conditionally taken as arguments for functions.

Mind that in R a dot generally serves as a decimal separator (in MS Excel, by the way, it depends: in cells, a comma is usually used, while dots are used in macros commands). Sometimes, the dot symbol is also called “period”.

Determine some of the functions listed below:

```
exp(2)              # Exponential of 2, e^2
log(1024, 2)        # Logarithm 1024 with the basis 2, log_2 1024
log(0.3)            # Natural logarithm of 0.3, ln 0.3
abs(-5)             # Modulus of -5, |−5|
```

```
atan2(0,-3)    # The angle between the OX-axis and the vector (-3,0) / here x, y are the opposite!
2**3; 2^3              # Exponentiation of 2 in 3 degree, 2^3  / both options are possible
sqrt(4)                # Root of 4, √4
factorial(5)           # Factorial of 5, 5!=1*2*3*4*5
choose(5,3)            # Number of combinations 3 out of 5:  C_5^3
pi                     # π number
exp(1)                 # e number
```

Rounding functions are also useful:

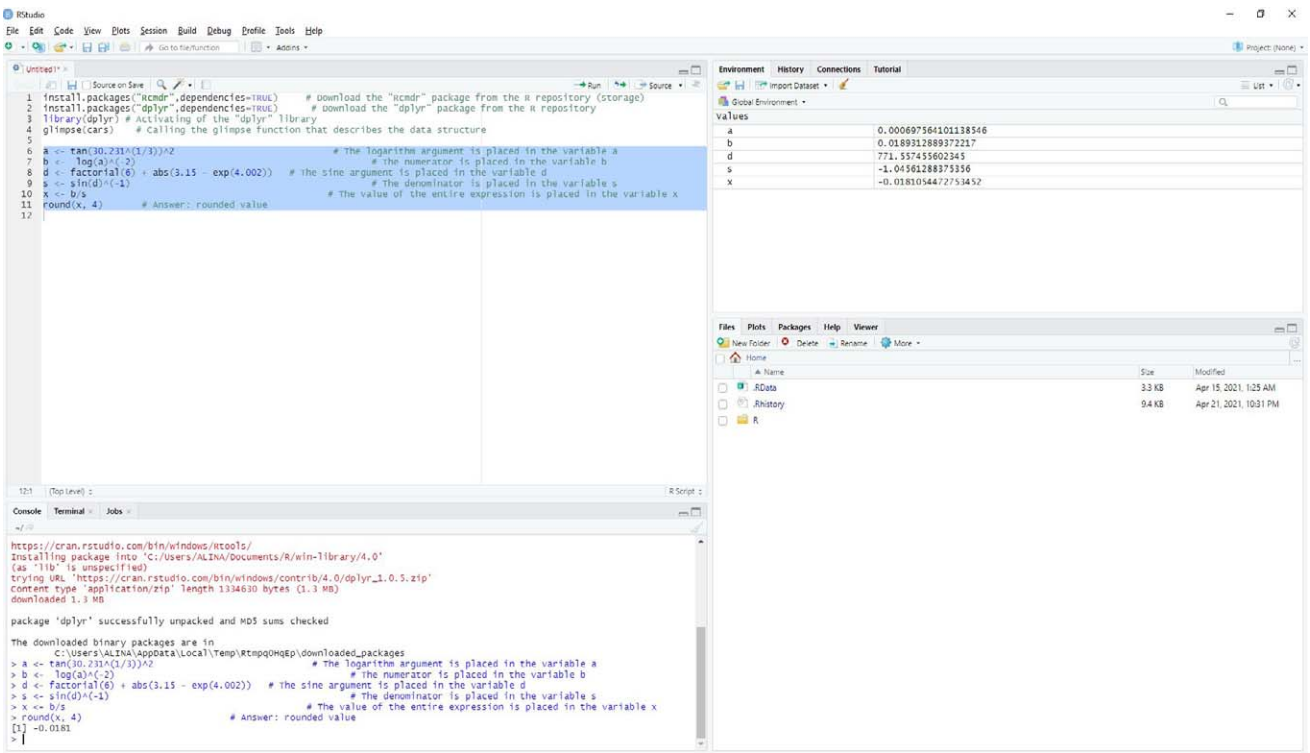
```
round(1.0023857, 6)  # Rounds the number 1.0023857 to 6 decimal places.
signif(1.0023857, 6) # Rounds the number 1.0023857 to 6 significant digits.
trunc(-3.999)        # Cutting off the fractional part -3.999.
floor(-4.2)          # The largest integer number not exceeding -4.2.
```

**Example 1.2.** Calculate the expression with accuracy of four digits after the decimal point.

$$\frac{\ln^{-2}(\mathrm{tg}^2\left(\sqrt[3]{30.231}\right))}{\sin^{-1}(6!+\left|3.15-e^{4.002}\right|)}.$$

**Solution.** Of course, we should not immediately rush to type this complex expression in a single line. The best way is to divide this expression into small parts, put intermediate calculations in new variables, and then come to answer:

```
a <- tan(30.231^(1/3))^2              # The logarithm argument is placed in the variable a
b <-  log(a)^(-2)                     # The numerator is placed in the variable b
d <- factorial(6) + abs(3.15 — exp(4.002))   # The sine argument is placed in the variable d
s <- sin(d)^(-1)                      # The denominator is placed in the variable s
x <- b/s                              # The value of the entire expression is placed in the variable x
round(x, 4)                           # Answer: rounded value — leave 4 digits after period
```



Here we introduced new variables $a, b, d, s$, and $x$, and used an assignment symbol «<-». It would not be a mistake to use the usual equal sign, but those who want to demonstrate the top class should use in this context the combination «<-» or «- >» when the variable is placed to the right of the assigned expression.

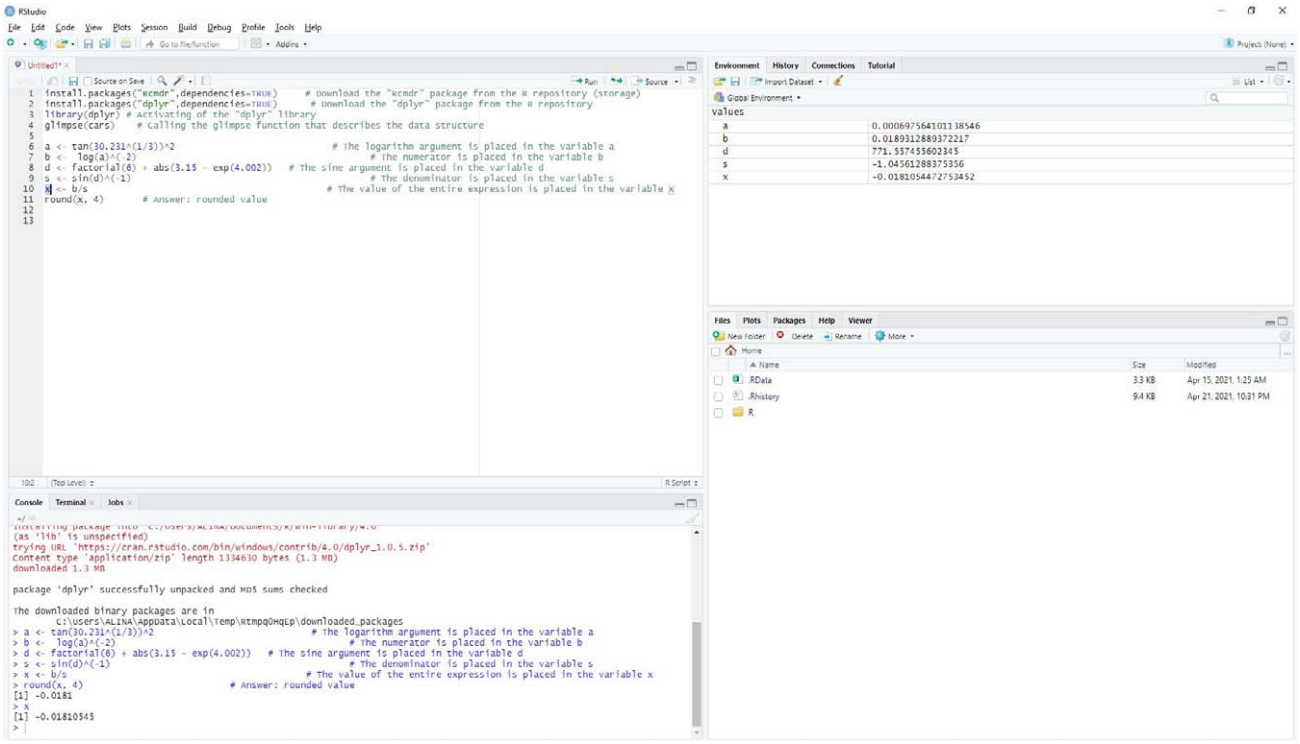
Note that the values of our intermediate variables are displayed in the upper-right window of the environment. If there are so many values that they do not fit in this window, then you can find out the required value of the variable in another way. Alternatively, just type it on a free line and press the combination «Ctrl+Enter»:

```
x
```

or separate it with a semicolon («;») from the main expression:

```
x <- b/s; x                    # The value of the entire expression is placed in the variable x
```

or select this variable with the mouse in the code text and send it for compilation by clicking the combination «Ctrl+Enter» (or click on the "Run" button in the RStudio interface):




## Accuracy of calculations

In addition to rounding numerical results, you might need to get more accurate calculations. The following command allows you to increase the output precision of *double*-arithmetic to 22 digits used in the number record (the parameter is called *digits*):

```
options(digits=12)                    # Setting the maximum number of digits used at 12
```

To get an even more accurate result, use specialized packages, for example, the «Rmpfr» package (the information about it will be available in RStudio upon standard installation of the package from the repository).

## Numeric arrays

Variables in R can be not only single numbers but also arrays of numbers (we'd like to remind that *array* is a systematic arrangement of similar objects, usually in rows and

columns). The simplest example is an ordered set of integers from 0 to 10. We can save this entire set in an array using the following command:

```
m <- 0:10                    # Array of numbers 0,1,2,..., 10
m                            # Display the contents of m
```

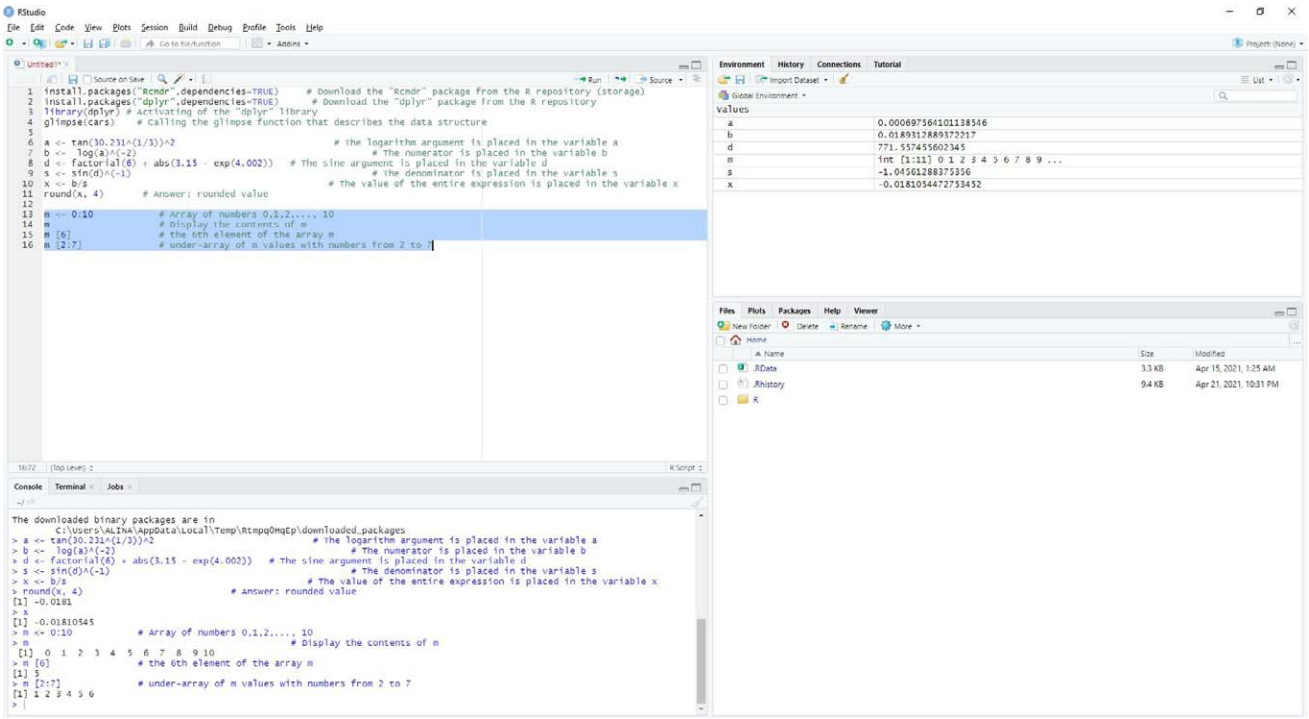
If you now run this code, you will get a whole set of numbers under the name "*m*". In case you want to refer to a specific element of this array, you need to use the square brackets operator [..]. For example, to find out the 6th element (entry) in the array "*m*", you need to type:

```
m [6]                        # the 6th element of the array m and the expression
```

as follows:

```
m [2:7]                      # subarray of m values with numbers from 2 to 7
```

It forms a subset of the values of the array *m* with numbers 2, 3,...,7, i.e. there is a new array m[2], m[3],..., m[7]:




Another example of an array of numbers is the *cars* object considered in Example №1 with data on velocities and braking distances. Mind that the report in the console window on the structure of the cars object by the *glimpse(cars)* function had the form:

```
Observations: 50

Variables: 2

$ speed <dbl> 4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, 13, 13,
13, 14,...

$ dist  <dbl> 2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24, 28, 26, 34,
34, 46,...
```

It follows that the cars variable is an array of data (more precisely, a data table — the type *data.frame*, but this does not make any difference for us):

*«Observations: 50»* literally means «there are observations of 50 pieces» and, if you think of this data as a table, then we are talking about 50 rows;

*«Variables: 2»* literally means «there are 2 variable pieces» and in fact, it is the number of columns;

*«$ speed <dbl> 4,4,...»* means that the first column in the cars variable is called «speed» (speed data), it is of the *«dbl»* type (*double* — double precision — real numbers whose modulus does not exceed 1.7*10^308), everything that exceeds this limit is denoted in R by the infinity symbol «Inf» (isn't it hard to imagine the real amount of something expressed by one with 308 zeros?);

*«$ dist <dbl> 2, 10,...»* is the same, but with the name of the second column as «dist» (distance). In other words, the data on the speed and the corresponding distance are grouped in two columns, and the rows correspond to the same observation (a specific test car).

In R, this means that you can access the various observations collected in the cars object in the following ways:

```
cars                    # The object itself is an array, display on the console screen
cars$speed              # Array of velocities (vector)
cars[ ,1]               # The same (values of all rows from the first column)
cars$dist               # Array of braking paths (vector)
cars[ ,2]               # The same (values of all rows from the second column)
cars[1, ]               # First observation (values of all columns from the 1st row)
cars[23,]               # 23rd observation (values of all columns from the 23rd row)
cars[6,1]               # Speed at the 6th observation
cars$speed[6]           # The same
cars[5,2]               # Braking distance at the 5th observation
cars$dist[5]            # The same
```

Then one should compare the line-by-line displays on the R console by running each line separately by pressing «Ctrl+Enter».

**Example 1.3.** Calculate the average speed of the first five observations in the *cars* object.

**Solution.** Type a code that calculates the arithmetic mean of the first five velocities:

```
(cars$speed[1] + cars$speed[2] + cars$speed[3] + cars$speed[4] + cars$speed[5])/5
```

or shorter:

```
(cars[1,1] + cars[2,1] + cars[3,1] + cars[4, 1] + cars[5,1])/5          # The same
```

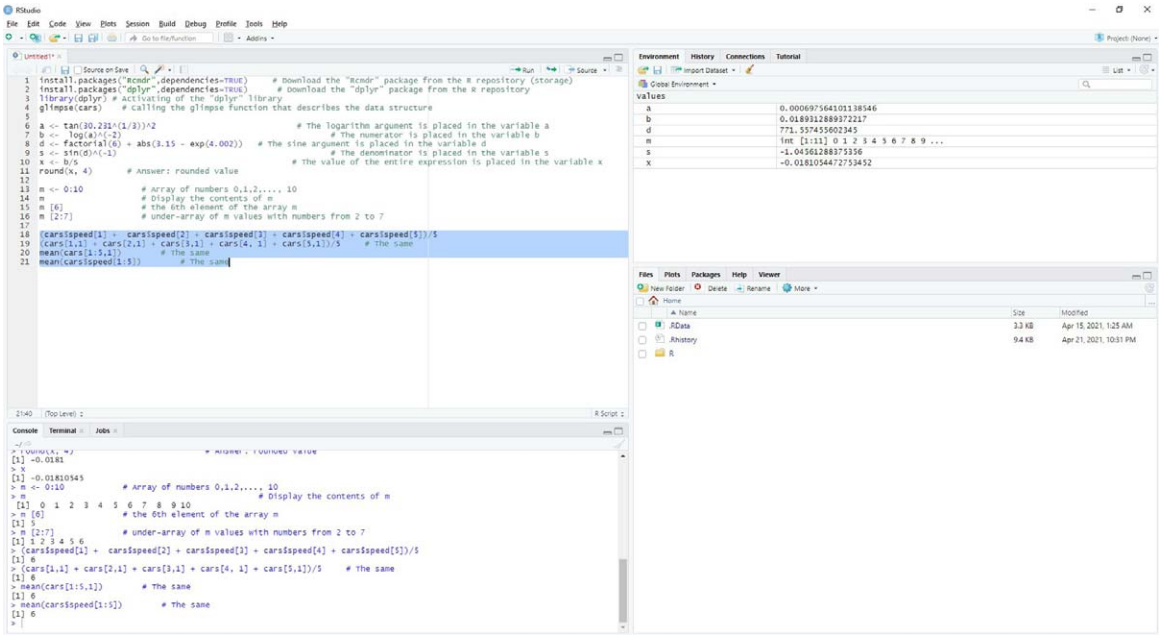
or even shorter:

```
mean(cars[1:5,1])                    # The same
```

or like this:

```
mean(cars$speed[1:5])                # The same
```

The last two commands return the built-in function for calculating the arithmetic mean for the elements of the first column (speed) of the cars array, chosen from rows 1 through 5.



In general, it is interesting that if we type the previous expression without specifying a certain range of observations, we will calculate the average speed of all the tested cars (1920):

```
mean(cars$speed)                    # Average speed of all tested cars
```

which will be 15.4 mph, which equals to 24.78 km/h (please compare with average velocities today).

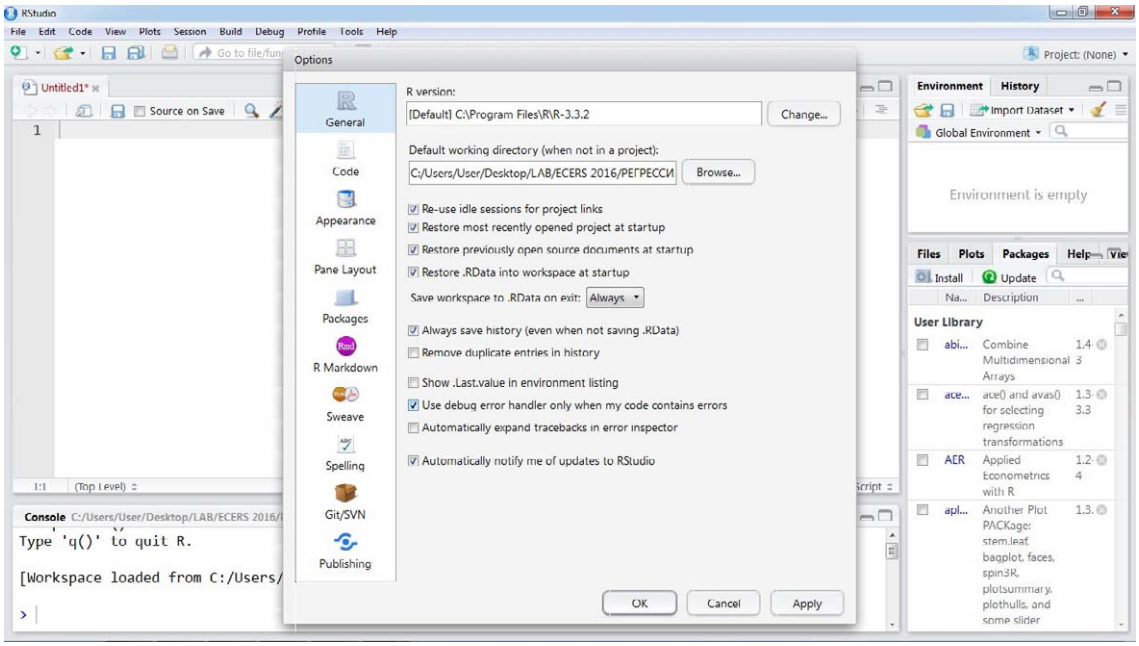
In the subsequent chapters, we will look closely at the data types in R and learn how to work with them.

## Some additional RStudio settings

Go to the main menu items Tools –> Global Options.

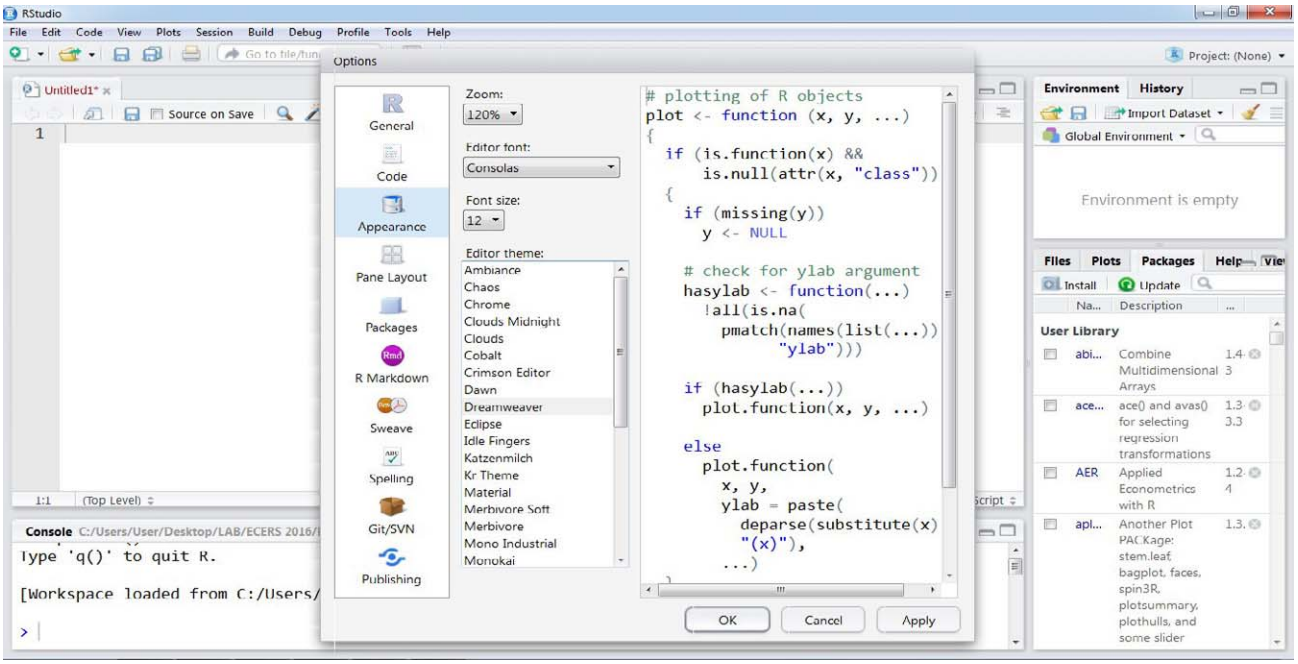
*Submenu «General»*. In the dialog box that appears, specify the path to the working directory R (default working directory) in the second line. This is where all your data will be stored by default, and this is default location for opening existing files.

It is useful to check the appropriate locations that determine *Autosave*, etc.



*Submenu «Appearance»*. These settings allow you to choose preferable colour palette, font combinations, and show scale on the screen. Feel free to choose different themes and experiment with Zoom and Editor Font. Do not forget to press the *Apply* key in the end.

The figure below shows the settings used by the authors of this textbook:




*Submenu «Pane Layout»*. This section allows you to arrange the information within the four windows of RStudio, adjusting everything you wish. The description of the other sections of the menu is beyond the scope of this lesson, but it is no less useful when using pro features of the package.

## Exercises for independent work

1. Calculate the expression in R with the accuracy of 6 significant digits:

$$\log_{48.23}\left(2^{-3}+\frac{\sin^3\left(7!+C_{32}^{11}\right)}{\sqrt{1+\operatorname{arctg}\left(\dfrac{1}{1+0.2435}\right)}}\right);$$

2. Calculate the expression in R with an accuracy of 3 digits after the decimal point:

$$\cos^{-1}\left(\frac{1}{\sqrt[3]{0.3532}}-\frac{\coth^3(12)*e^{-1/4.8}}{\sqrt{\left|\log_{13.76}\left(\dfrac{256}{1809.43}\right)\right|+\operatorname{arcctg}\left(7^{-3}\right)}}\right);$$

3. Calculate the arithmetic mean of the braking length for the cars data, expressed in meters. Assume that 1 foot is 0.3048 meters. (Answer: 13.1 m.)

4. After loading the package «*ggplot2*» in R, a data table named diamonds becomes available, which contains statistical studies of diamonds. Determine how many diamonds were examined and find the average weight of the diamonds (in carats).

*Note*. Download the package «ggplot2» and use the «glimpse» function from the package «dplyr» to understand which variable from the table to select for further actions.

5. Check whether for very small values of x it is assumed that $\sin x \approx x$. In this case, what function will $\cos x$ look like?

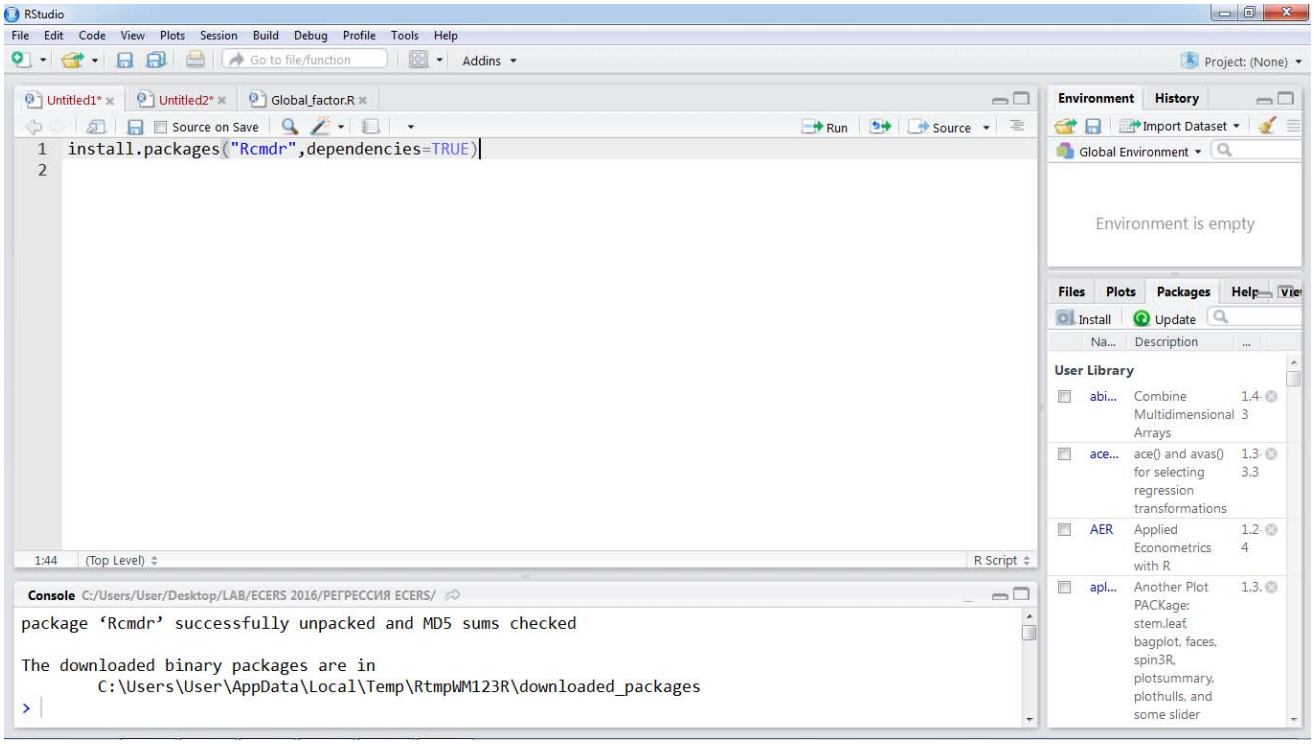
## Appendix: instructions for installing the R and RStudio packages

To install R and RStudio under the Windows operating system, follow the steps below.

1. Make sure you have access to the Internet during the entire installation process.
2. Follow the link **https://cran.r-project.org/** and download the R loader "R-4.2.2-win" (or a newer version).
3. Run the saved R loader, accepting all its recommendations and default settings, and wait for the installation of the R language to finish.

For Windows 10 (or higher), you do not need to specify the path to the R folder.

4. Using the link **https://www.rstudio.com/products/rstudio/download/** to download the appropriate R Studio loader for your operating system and run it.
5. After installation, run RStudio, and load the «Rcmdr» package running the following command: install.packages("Rcmdr", dependencies=TRUE) — then press the key combination «Ctrl+Enter»:




6. Installation will take a few minutes, during which the Internet access is required.
7. Set the working directory in RStudio as follows: Tools –> Global Options ... Specify the route to the working folder where your files and projects will be stored.
8. Restart RStudio.

For installation under operating systems other than Windows, proceed the same way. In order to install several packages, such as the crucial package "*xlsx*" for R interaction with MS Excel, you generally need to install Java on your computer. You can do this by following the links below consistently and accepting all the default installation settings:

https://java.com/ru/download/

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Do not forget to choose the appropriate operating system. Having downloaded Java, you can then download the "*xlsx*" package, which is described in detail in other parts of the textbook.

# 2. USER-DEFINED FUNCTIONS

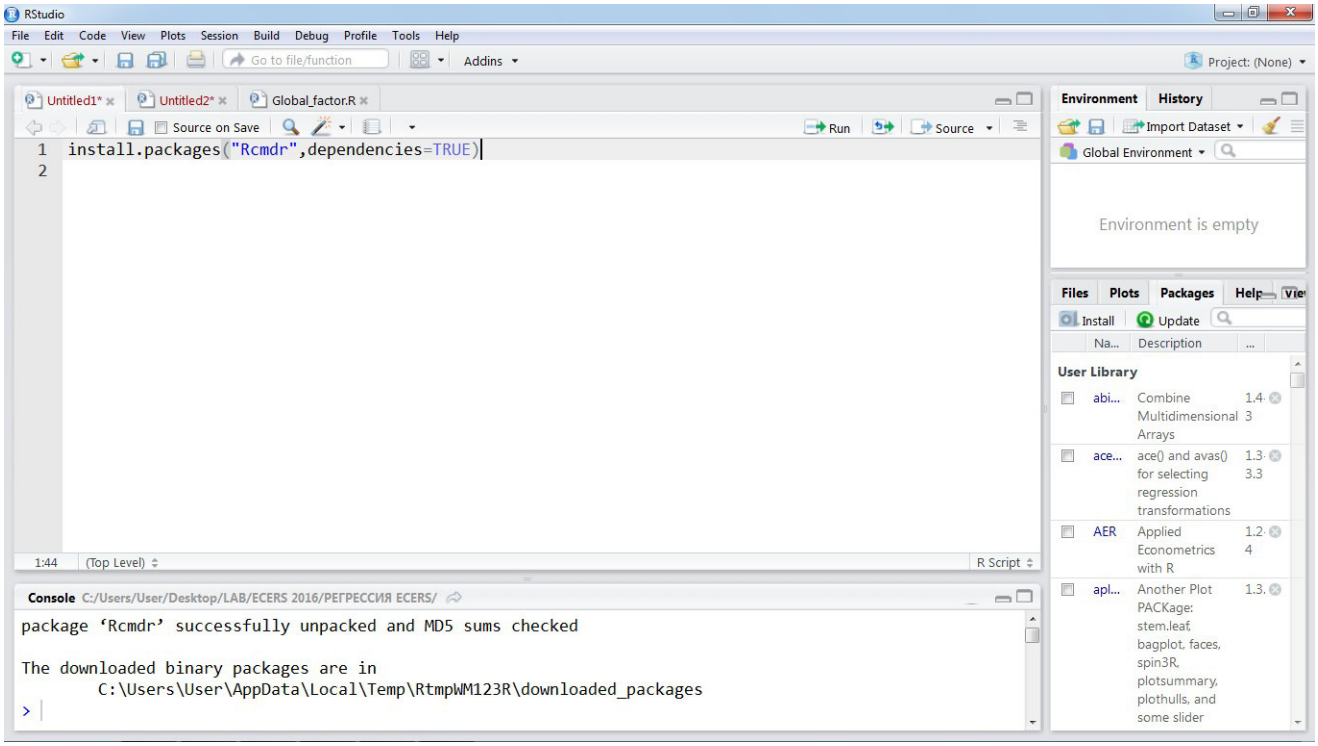
## Defining mathematical functions

In the R language, as in any other programming language, it is possible to create different variables and functions. If we consider mathematical functions, here the object will not be defined by a specific number or array, but by the formula itself or a special algorithm determining, further calculations. This turns out to be rather convenient when you perform the same actions repeatedly.

Launch RStudio, create a new code window (for example, using the key combination *Ctrl+Shift+N*), and enter or copy the following code:

```
f <- function(x) {(x-3)^2}                    # Setting of a function f(x)=(x-3)^2
```

What will happen if we pass this code for compilation by using *Ctrl+Enter* combination?




At first glance, the R environment hasn't reacted to the entered text in any way, but in fact, a function named $f$ was created in the memory of the *R* software environment, and its values, according to our code in curly brackets {(x-3)^2}, depend on one variable:

$$f(x)=(x-3)^2.$$

Mind the upper-right window of RStudio: a newly created function has emerged.

This means that R now understands the reference to this function for specific numbers. For example, we calculate the value of this function at the point $x=5$ by entering:

```
f(5)                                          # The value of the function f at 5
```