An Introduction to R Programming

Why Learn R?

R is a powerful **statistical programming language** widely used in data analysis and research. With its extensive library of packages, R allows you to **manipulate data**, perform **statistical tests**, create **stunning visualizations**, and much more. Learning R opens up a world of data exploration and analysis possibilities.

Why R?

- It's free!
- It runs on a variety of platforms including Windows, Unix and MacOS.
- It provides an unparalleled platform for programming new statistical methods in an easy and straightforward manner.
- It contains advanced statistical routines not yet available in other packages.
- It has state-of-the-art graphics capabilities.

What is R Programming?

- R programming is a ***powerful tool used in data analysis, statistical computing,*** and ***graphics.***
- It was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, in the mid-1990s.
- R programming has become increasingly popular in recent years due to its ***open-source nature, flexibility***, and ease of use.
- One of the key features of R programming is its ***ability to handle large datasets*** and perform complex statistical analyses.
- It also offers a wide range of built-in functions and packages for ***data visualization, machine learning, and predictive modeling***.
- These capabilities make R programming an essential tool for data scientists, statisticians, and researchers in various fields.

How to download?

- Google it using R or CRAN

(Comprehensive R Archive Network)

- http://www.r-project.org

Tutorials

Each of the following tutorials are in PDF format.

- P. Kuhnert & B. Venables, An Introduction to R: Software for Statistical Modeling & Computing
- J.H. Maindonald, Using R for Data Analysis and Graphics
- B. Muenchen, R for SAS and SPSS Users
- W.J. Owen, The R Guide
- D. Rossiter, Introduction to the R Project for Statistical Computing for Use at the ITC
- W.N. Venebles & D. M. Smith, An Introduction to R

Web links

- Paul Geissler's excellent R tutorial
- Dave Robert's Excellent Labs on Ecological Analysis
- Excellent Tutorials by David Rossitier
- Excellent tutorial an nearly every aspect of R **MOST of these notes follow this web page format**
- Introduction to R by Vincent Zoonekynd
- R Cookbook
- Data Manipulation Reference

Advantages of R Programming over other languages?

- It's ***open-source***. No fees or licenses are needed, so it's a low-risk venture if you're developing a new program.

- It's ***platform-independent***. R runs on all operating systems, so developers only need to create one program that can work on competing systems. This independence is yet another reason why R is cost-effective!

- It has ***lots of packages***. For example, the R language has more than 10,000 packages stored in the CRAN repository, and the number is continuously increasing.

- It's great for statistics. Statistics are a big thing today, and R shines in this regard. As a result, programmers prefer it over other languages for ***statistical tool development***.

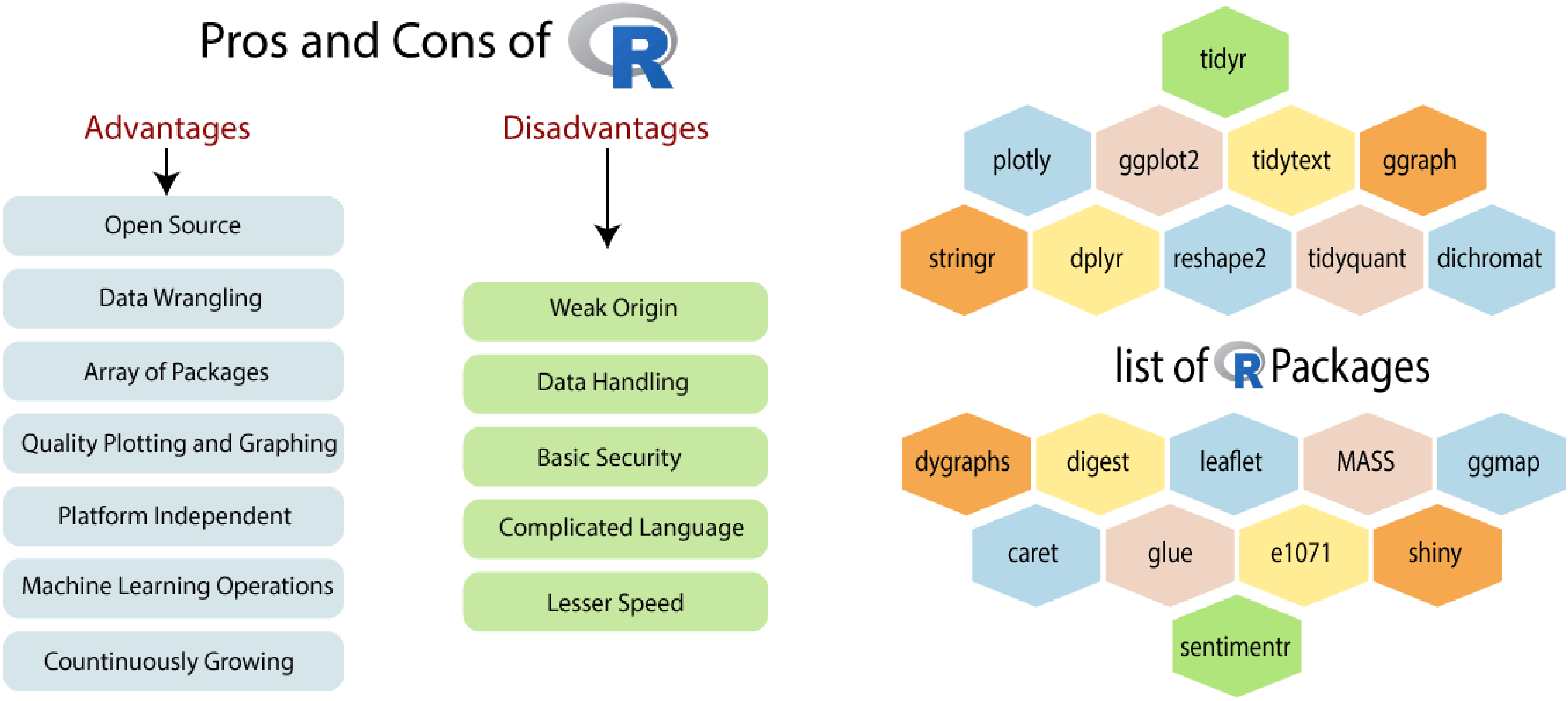
- It's well suited for ***Machine Learning***. R is ideal for machine learning operations such as regression and classification. It even offers many features and packages for artificial neural network development.

- R lets you perform ***data wrangling***. R offers a host of packages that help data analysts turn unstructured, messy data into a structured format.

Limitations of R Programming

While R Programming provides a wide range of statistical tools, it has some limitations:

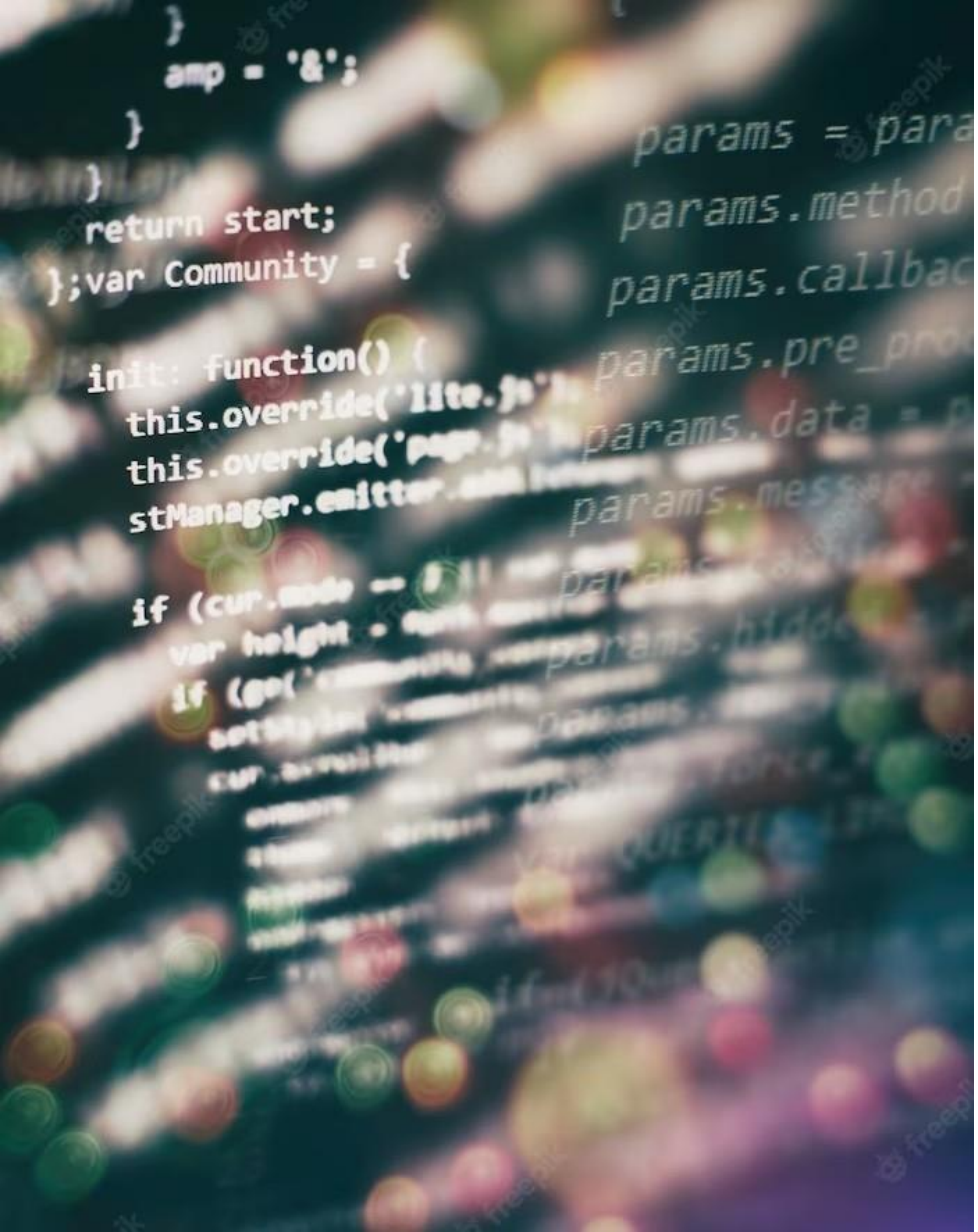
- One of the main limitations is its inability to handle very big data effectively. R requires loading all data into memory, which can be problematic for large datasets.
- Additionally, R's processing speed can be slow compared to other programming languages.
- Another limitation of R is its poor handling of non-numeric data. While R is excellent at analyzing numeric data, it struggles with handling text data. This can make it challenging to work with datasets that contain both numeric and non-numeric data.
- It’s not as secure. R doesn’t have basic security measures. Consequently, it’s not a good choice for making web-safe applications. Also, R can’t be embedded in web browsers.
- It takes up a lot of memory. Memory management isn’t one of R’s strong points. R’s data must be stored in physical memory. However, the increasing use of cloud-based memory may eventually make this drawback moot.


Pros and Cons of R
Advantages
Open Source
Data Wrangling
Array of Packages
Quality Plotting and Graphing
Platform Independent
Machine Learning Operations
Countinuously Growing
Disadvantages
Weak Origin
Data Handling
Basic Security
Complicated Language
Lesser Speed
tidyr
plotly
ggplot2
tidytext
ggraph
stringr
dplyr
reshape2
tidyquant
dichromat
list of R Packages
dygraphs
digest
leaflet
MASS
ggmap
caret
glue
e1071
shiny
sentimentr

Getting Started with R

To start using R, you need to install **R software** and an **integrated development environment (IDE)** like RStudio.

RStudio provides a user-friendly interface for coding in R and managing your projects. Once set up, you can begin writing and executing R code to unleash its power.



Basic R Syntax

R uses a concise and expressive syntax for data manipulation and analysis. Variables are assigned using the **assignment operator** '<-', and functions are called using parentheses.

R also supports **vectorized operations** that allow you to perform operations on entire data vectors at once, making it efficient and convenient for data analysis.

Data Manipulation in R

R provides powerful tools for data manipulation. With the help of **packages like dplyr and tidyr**, you can easily **filter**, **sort**, **group**, and **reshape data** to suit your analysis needs.

These packages make data wrangling a breeze, enabling you to efficiently clean and prepare your data for further analysis.

R Overview

- R is a comprehensive statistical and graphical programming language and is a dialect of the S language:
  - 1988 - S2: RA Becker, JM Chambers, A Wilks
  - 1992 - S3: JM Chambers, TJ Hastie
  - 1998 - S4: JM Chambers
- R: initially written by Ross Ihaka and Robert Gentleman at Dep. of Statistics of U of Auckland, New Zealand during 1990s.
- Since 1997: international “R-core” team of 15 people with access to common CVS archive.

R Overview

You can enter commands one at a time at the command prompt (>) or run a set of commands from a source file.

There is a wide variety of data types, including vectors (numerical, character, logical), matrices, data frames, and lists.

To quit R, use

```
>q()
```

R Overview

Most functionality is provided through built-in and user-created functions and all data objects are kept in memory during an interactive session.

Basic functions are available by default. Other functions are contained in packages that can be attached to a current session as needed

R Overview

A key skill to using **R** effectively is learning how to use the built-in help system. Other sections describe the working environment, inputting programs and outputting results, installing new functionality through packages and etc.

A fundamental design feature of **R** is that the output from most functions can be used as input to other functions. This is described in reusing results.

Your First R Session

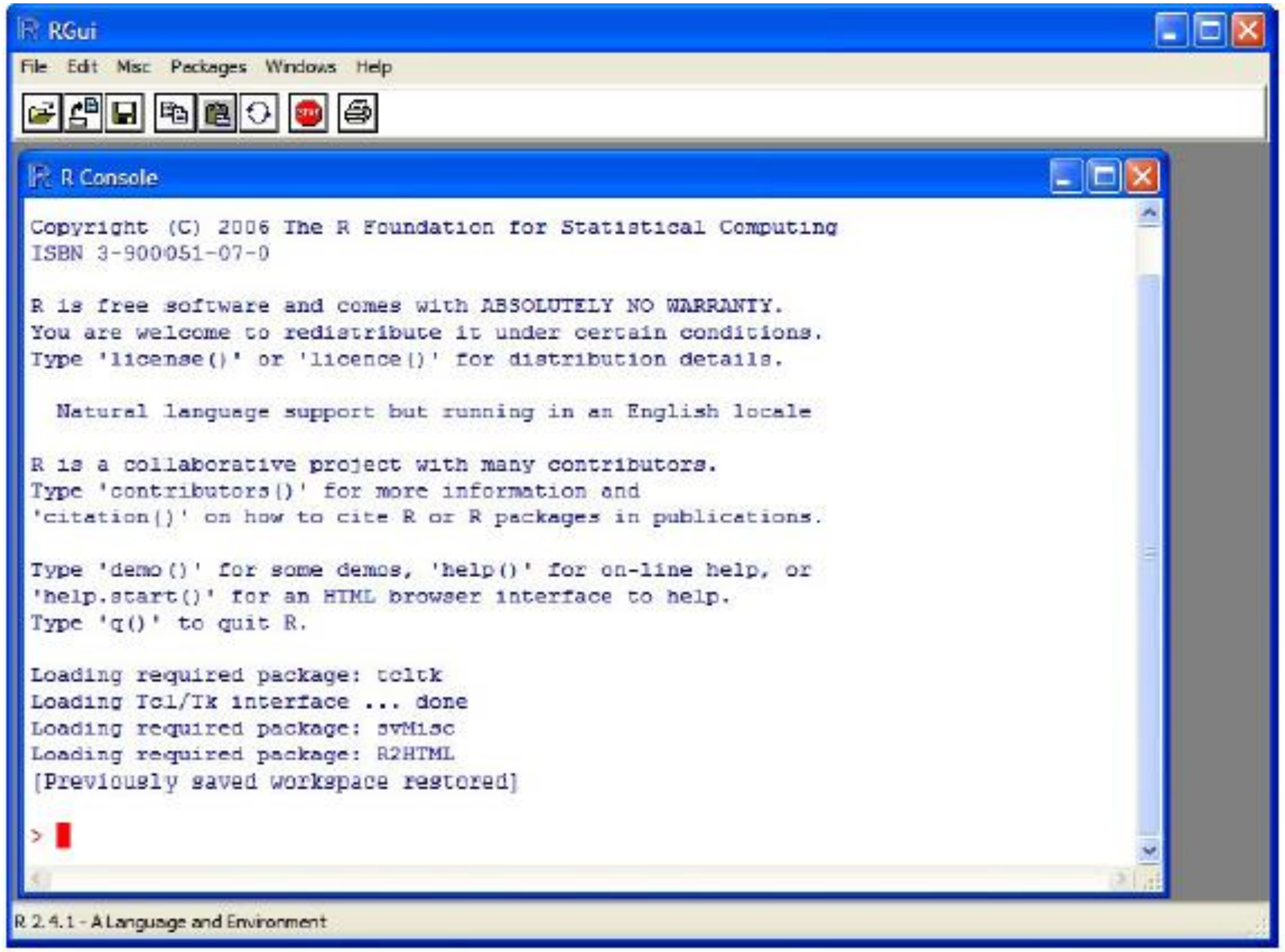


Figure 1.1: The R system on Windows

R Introduction

- These objects can then be used in other calculations. To print the object just enter the name of the object. There are some restrictions when giving an object a name:
  - Object names cannot contain `strange' symbols like !, +, -, #.
  - A dot (.) and an underscore ( ) are allowed, also a name starting with a dot.
  - Object names can contain a number but cannot start with a number.
  - R is case sensitive, X and x are two different objects, as well as temp and temP.

R Workspace

Objects that you create during an R session are hold in memory, the collection of objects that you currently have is called the workspace.

This workspace is not saved on disk unless you tell R to do so. This means that your objects are lost when you close R and not save the objects, or worse when R or your system crashes on you during a session.

R Workspace

When you close the RGui or the R console window, the system will ask if you want to save the workspace image.

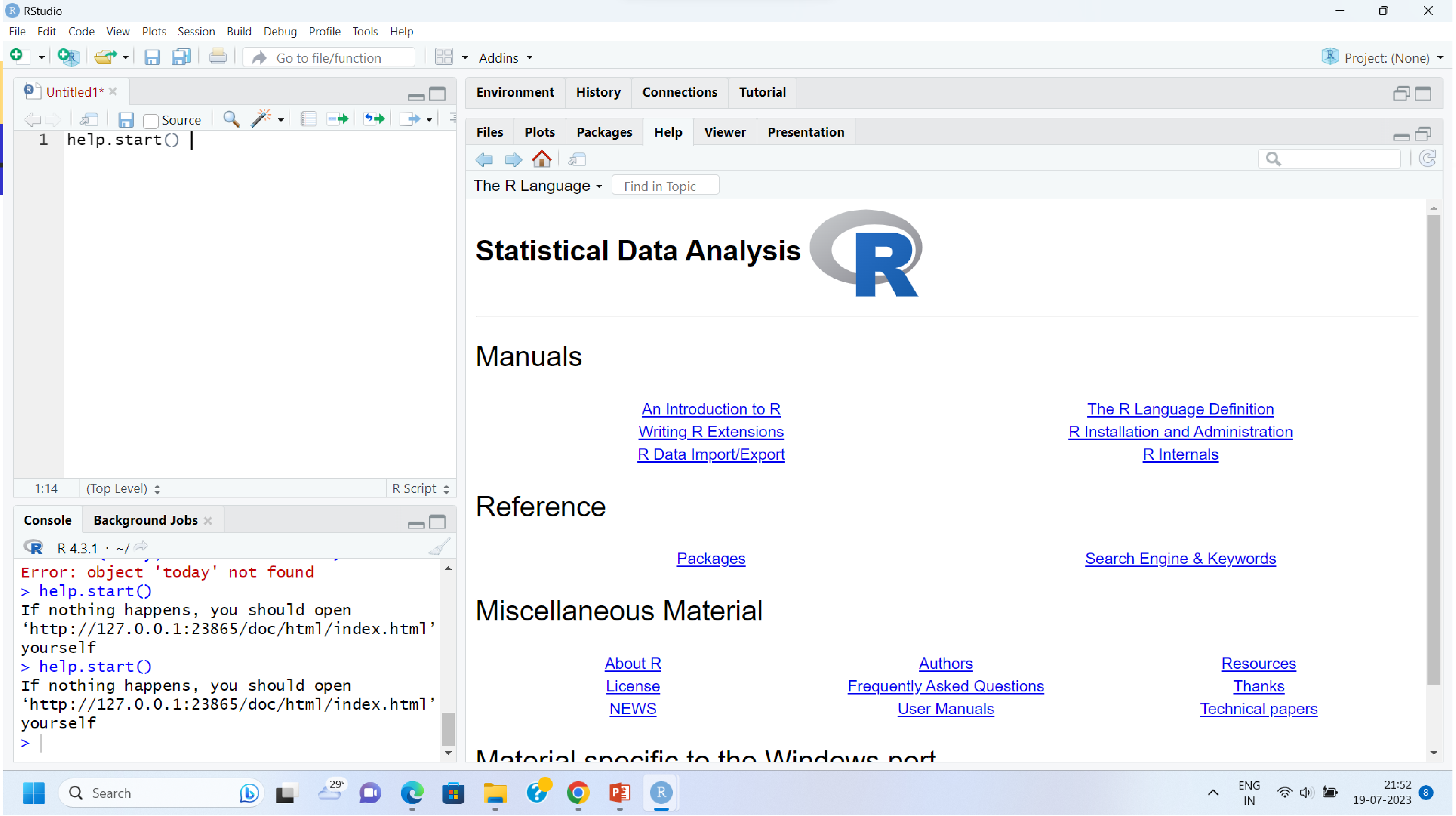
If you select to save the workspace image then all the objects in your current R session are saved in a file .RData.

This is a binary file located in the working directory of R, which is by default the installation directory of R.

R Help

Once **R** is installed, there is a comprehensive built-in help system. At the program's command prompt you can use any of the following:

help.start() # general help
help(*foo*) # help about function *foo*
?*foo* # same thing
apropos("*foo*") # list all function containing string foo
example(*foo*) # show an example of function *foo*


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source
1 help.start()
1:14 (Top Level) R Script
Console Background Jobs
R 4.3.1 · ~/
Error: object 'today' not found
> help.start()
If nothing happens, you should open
'http://127.0.0.1:23865/doc/html/index.html'
yourself
> help.start()
If nothing happens, you should open
'http://127.0.0.1:23865/doc/html/index.html'
yourself
>
Environment History Connections Tutorial
Files Plots Packages Help Viewer Presentation
The R Language
Find in Topic
Statistical Data Analysis
Manuals
An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals
Reference
Packages
Search Engine & Keywords
Miscellaneous Material
About R
License
NEWS
Authors
Frequently Asked Questions
User Manuals
Resources
Thanks
Technical papers
Material specific to the Windows port
Search
ENG IN
21:52
19-07-2023

R Help

```
# search for foo in help manuals and archived mailing lists
   RSiteSearch("foo")
# get vignettes on using installed packages
   vignette()      # show available vingettes
   vignette("foo") # show specific vignette
```

Unit-1_1.pdf
search.r-project.org: foo
https://search.r-project.org/?FMT=query&P=foo&HITSPERPAGE=20&SORT=&DB=cran-help&DB=cran-info&DB=cran-news&DB=r-help&DB=r-manuals&D...
foo
Search
Action center
Sort by: relevance Results per Page: 20
Search in: R Manuals Base Packages Help Pages | CRAN Task Views
CRAN Packages Description Help Pages News Readme Vignettes
Matching any words Matching all words
1-20 of about 2,511 matches
Term frequencies: foo: 2,511
Search took 0.005976 seconds
Modified: 2023-07-15 Size: 2.9K
R: Foo is Not a Real Name
...explanation of the placeholder name foo. Usage foo() ## S3 method for class 'foo' plot(x, ...) Arguments x Ignored. ... Ignored. Details The name foo is used by computer scientists as a place holder...
/CRAN/refmans/spatstat/html/foo.html
cran-help matching: foo, foo and foo
Modified: 2023-07-15 Size: 1.5K
R: Replication data for Foos, John, Muller, and Cunningham...
...Foos, John, Muller, and Cunningham (2021), Journal of Politics (derived from from Dataverse 10.7910/DVN/NDPXND) Description Replication data for Foos, John, Muller, and Cunningham (2021), Journal of...
/CRAN/refmans/rdss/html/foos_etal.html
cran-help matching: foo, foo and foo
Modified: 2023-07-15 Size: 1.6K
R: Simulated logistic regression data.
...predictor. x2 quantitative predictor. x3 quantitative predictor. y Bernoulli response. Examples library(mcmc) data(foo) out <- glm(y ~ x1 + x2 + x3, family = binomial, data = foo) summary(out)...
/CRAN/refmans/mcmc/html/foo.html
cran-help matching: foo and foo
Modified: 2023-07-15 Size: 1.7K
R: Useful function to estimate the parameter a
...Useful function to estimate the parameter a Usage foo_a(x, nb, left_bound, right_bound) Arguments x x. nb nb. left_bound left_bound. right_bound right_bound. Value a numeric. [Package cobiclust...
/CRAN/refmans/cobiclust/html/foo_a.html
cran-help matching: foo
Modified: 2023-07-15 Size: 4.5K
R: Math operations
...do(.data, ...) do_(.data, ..., .dots) lengthj(.data) sqrtj(.data) floorj(.data) minj(.data, ...) minj_(.data, ..., .dots) maxj(.data, ...) maxj_(.data, ..., .dots) ad(.data) map(.data, ...) map_(.data...
/CRAN/refmans/jqr/html/maths.html
cran-help matching: foo
Modified: 2023-07-15 Size: 4.8K
R: Manipulation operations
...Manipulation operations Usage join(.data, ...) join_(.data, ..., .dots) splitj(.data, ...) splitj_(.data, ..., .dots) ltrimstr(.data, ...) ltrimstr_(.data, ..., .dots) rtrimstr(.data, ...) rtrimstr_...
/CRAN/refmans/jqr/html/manip.html
cran-help matching: foo
Search
ENG IN
21:55 19-07-2023

R Datasets

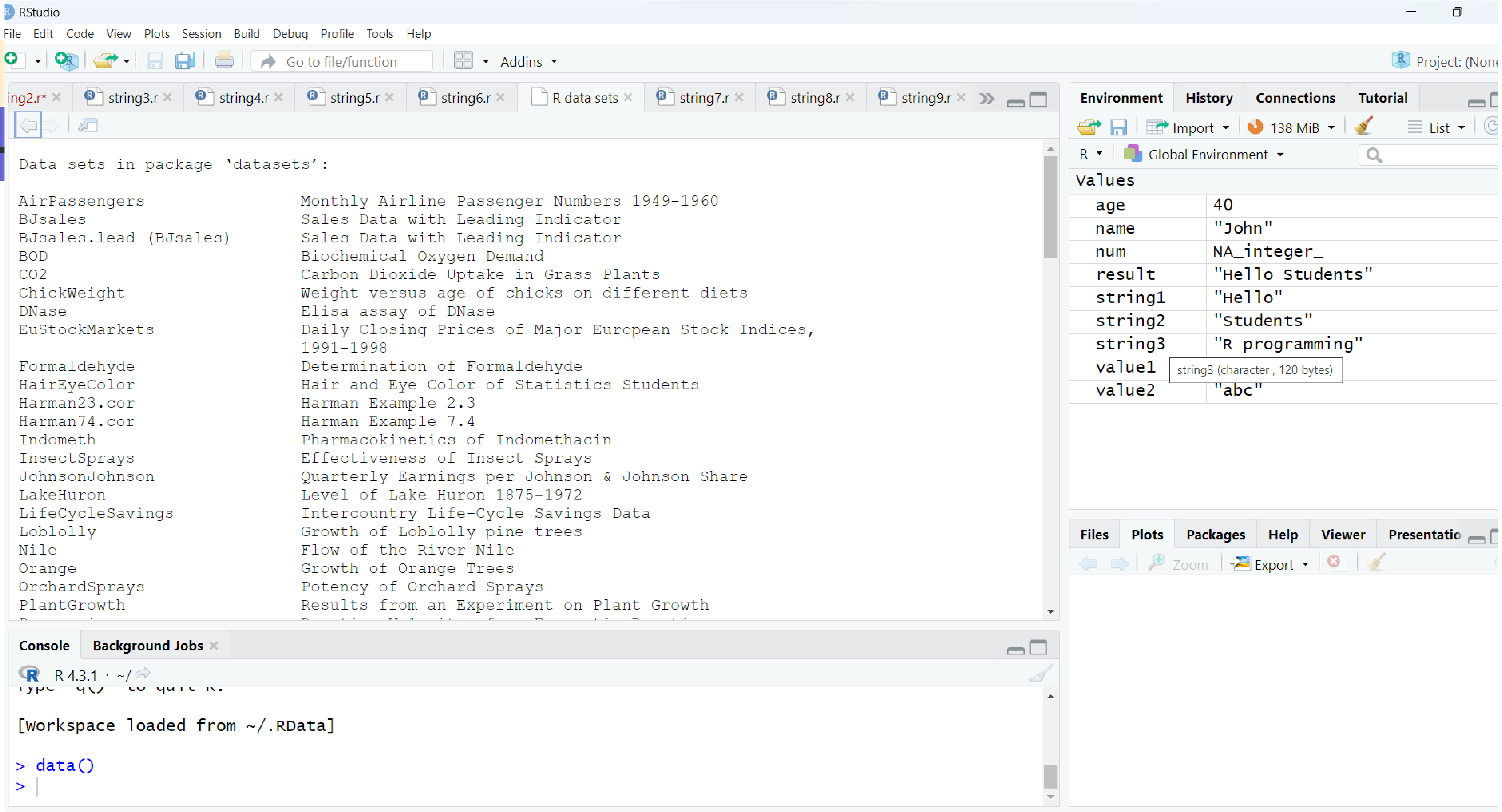
R comes with a number of sample datasets that you can experiment with. Type

> data( )

to see the available datasets. The results will depend on which packages you have loaded. Type

help(*datasetname***)**

for details on a sample dataset.


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None
ing2.r* string3.r string4.r string5.r string6.r R data sets string7.r string8.r string9.r
Data sets in package 'datasets':
AirPassengers Monthly Airline Passenger Numbers 1949-1960
BJsales Sales Data with Leading Indicator
BJsales.lead (BJsales) Sales Data with Leading Indicator
BOD Biochemical Oxygen Demand
CO2 Carbon Dioxide Uptake in Grass Plants
ChickWeight Weight versus age of chicks on different diets
DNase Elisa assay of DNase
EuStockMarkets Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde Determination of Formaldehyde
HairEyeColor Hair and Eye Color of Statistics Students
Harman23.cor Harman Example 2.3
Harman74.cor Harman Example 7.4
Indometh Pharmacokinetics of Indomethacin
InsectSprays Effectiveness of Insect Sprays
JohnsonJohnson Quarterly Earnings per Johnson & Johnson Share
LakeHuron Level of Lake Huron 1875-1972
LifeCycleSavings Intercountry Life-Cycle Savings Data
Loblolly Growth of Loblolly pine trees
Nile Flow of the River Nile
Orange Growth of Orange Trees
OrchardSprays Potency of Orchard Sprays
PlantGrowth Results from an Experiment on Plant Growth
Console Background Jobs
R 4.3.1 · ~/
[Workspace loaded from ~/.RData]
> data()
>
Environment History Connections Tutorial
Import 138 MiB List
R Global Environment
Values
age 40
name "John"
num NA_integer_
result "Hello Students"
string1 "Hello"
string2 "Students"
string3 "R programming"
value1 string3 (character , 120 bytes)
value2 "abc"
Files Plots Packages Help Viewer Presentatio
Zoom Export

R Packages

- One of the strengths of R is that the system can easily be extended.
- The system allows you to write new functions and package those functions in a so called `R package' (or `R library').
- The R package may also contain other R objects, for example data sets or documentation. There is a lively R user community and many R packages have been written and made available on CRAN for other users.
- Just a few examples, there are packages for portfolio optimization, drawing maps, exporting objects to html, time series analysis, spatial statistics and the list goes on and on.

R Packages

- When you download R, already a number (around 30) of packages are downloaded as well.
- To use a function in an R package, that package has to be attached to the system. When you start R not all of the downloaded packages are attached, only seven packages are attached to the system by default.
- You can use the function search to see a list of packages that are currently attached to the system, this list is also called the search path.

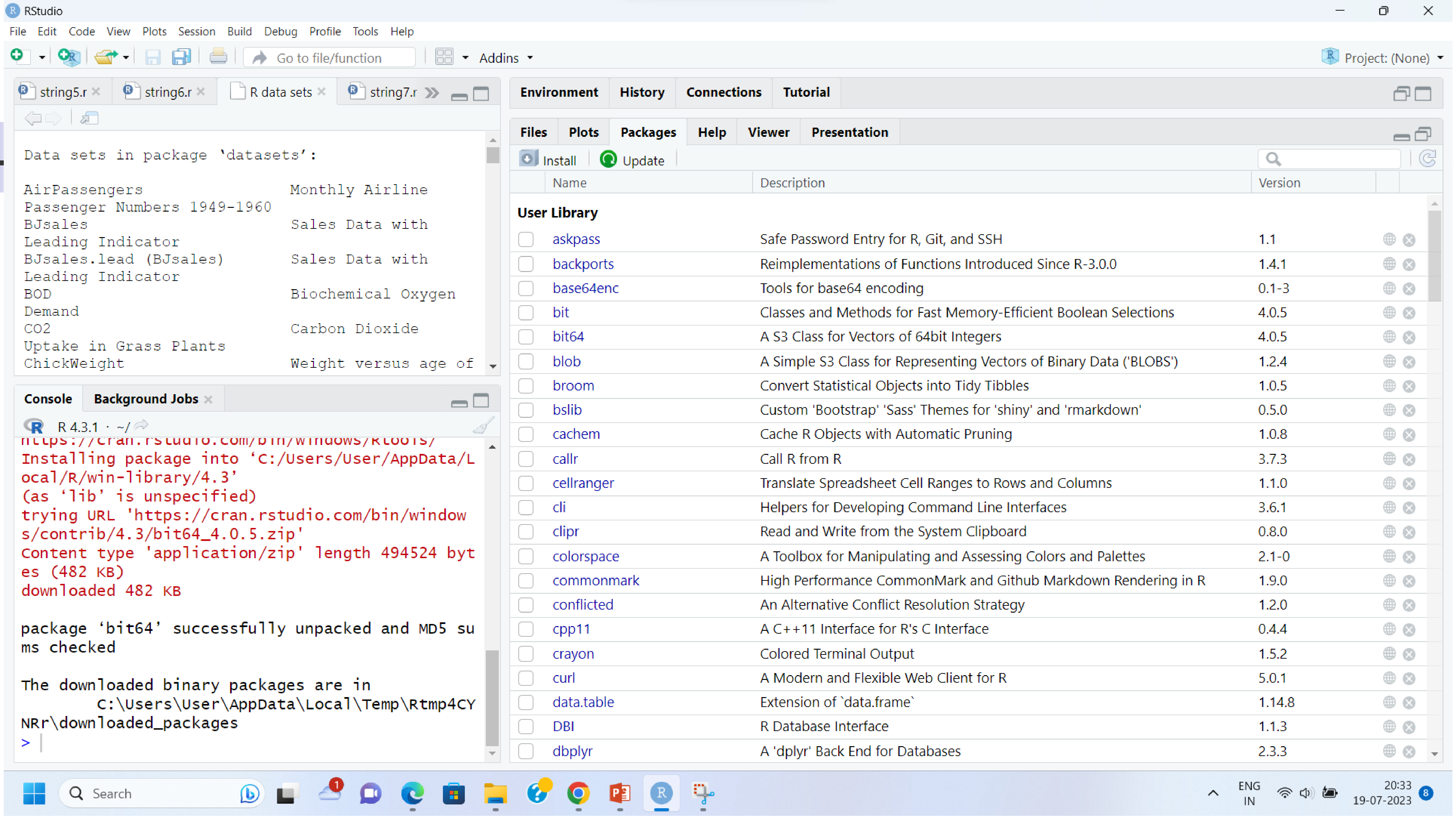
```
> search()
[1] ".GlobalEnv" "package:stats" "package:graphics"
[4] "package:grDevices" "package:datasets" "package:utils"
[7] "package:methods" "Autoloads" "package:base"
```

R Packages

To attach another package to the system you can use the menu or the library function. Via the menu:

Select the `Packages' menu and select `Load package...', a list of available packages on your system will be displayed. Select one and click `OK', the package is now attached to your current R session. Via the library function:

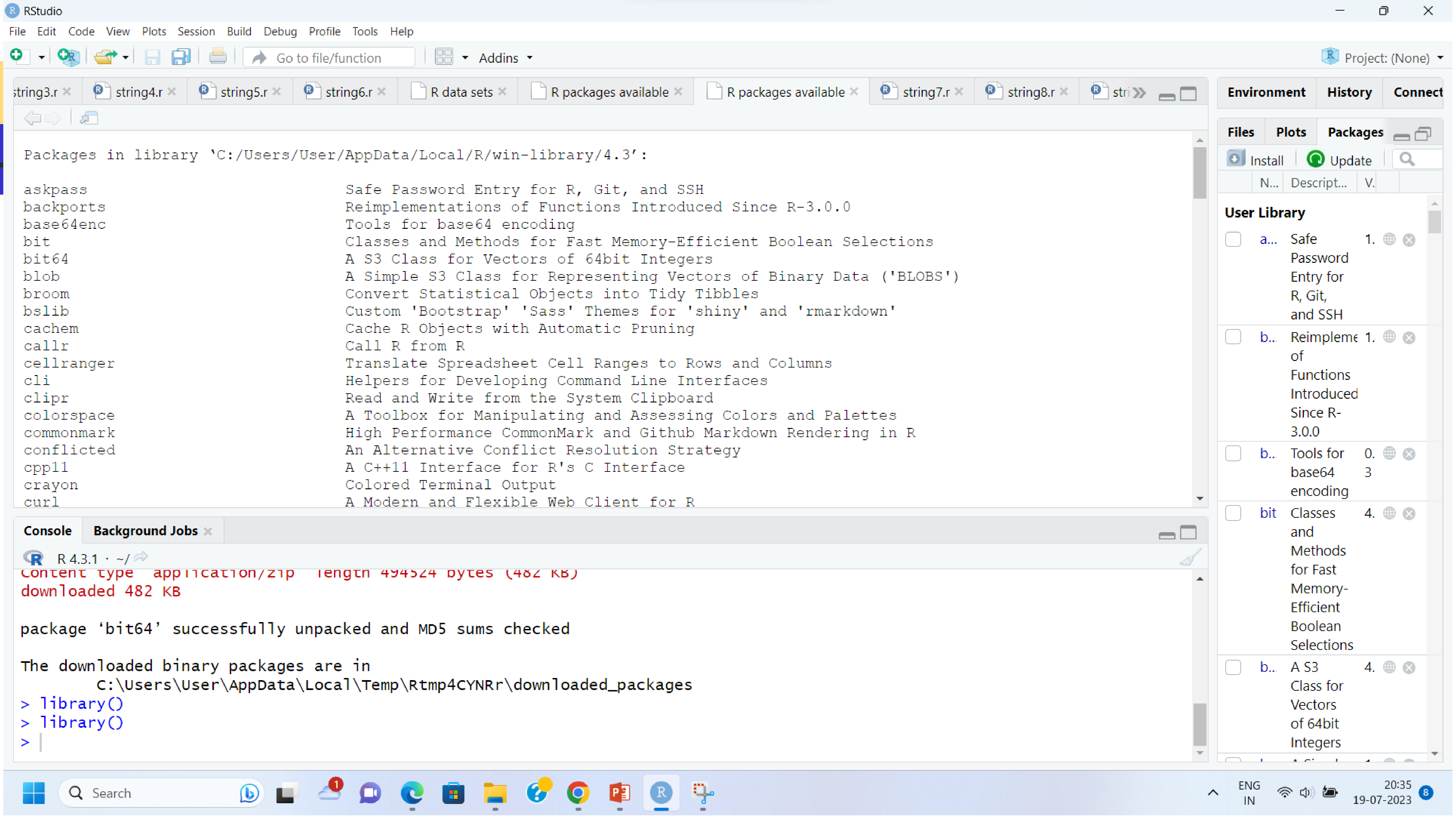
```
> library(MASS)
> shoes
$A
[1] 13.2 8.2 10.9 14.3 10.7 6.6 9.5 10.8 8.8 13.3
$B
[1] 14.0 8.8 11.2 14.2 11.8 6.4 9.8 11.3 9.3 13.6
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
string5.r
string6.r
R data sets
string7.r
Data sets in package 'datasets':
AirPassengers Monthly Airline Passenger Numbers 1949-1960
BJsales Sales Data with Leading Indicator
BJsales.lead (BJsales) Sales Data with Leading Indicator
BOD Biochemical Oxygen Demand
CO2 Carbon Dioxide Uptake in Grass Plants
ChickWeight Weight versus age of
Console
Background Jobs
R 4.3.1 · ~/
Installing package into 'C:/Users/User/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/bit64_4.0.5.zip'
Content type 'application/zip' length 494524 bytes (482 KB)
downloaded 482 KB
package 'bit64' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\User\AppData\Local\Temp\Rtmp4CYNRr\downloaded_packages
Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Install
Update
Name
Description
Version
User Library
askpass Safe Password Entry for R, Git, and SSH 1.1
backports Reimplementations of Functions Introduced Since R-3.0.0 1.4.1
base64enc Tools for base64 encoding 0.1-3
bit Classes and Methods for Fast Memory-Efficient Boolean Selections 4.0.5
bit64 A S3 Class for Vectors of 64bit Integers 4.0.5
blob A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS') 1.2.4
broom Convert Statistical Objects into Tidy Tibbles 1.0.5
bslib Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown' 0.5.0
cachem Cache R Objects with Automatic Pruning 1.0.8
callr Call R from R 3.7.3
cellranger Translate Spreadsheet Cell Ranges to Rows and Columns 1.1.0
cli Helpers for Developing Command Line Interfaces 3.6.1
clipr Read and Write from the System Clipboard 0.8.0
colorspace A Toolbox for Manipulating and Assessing Colors and Palettes 2.1-0
commonmark High Performance CommonMark and Github Markdown Rendering in R 1.9.0
conflicted An Alternative Conflict Resolution Strategy 1.2.0
cpp11 A C++11 Interface for R's C Interface 0.4.4
crayon Colored Terminal Output 1.5.2
curl A Modern and Flexible Web Client for R 5.0.1
data.table Extension of `data.frame` 1.14.8
DBI R Database Interface 1.1.3
dbplyr A 'dplyr' Back End for Databases 2.3.3
Search
ENG
IN
20:33
19-07-2023

R Packages

- The function library can also be used to list all the available libraries on your system with a short description. Run the function without any arguments

```
> library()
Packages in library 'C:/PROGRA~1/R/R-25~1.0/library':
base            The R Base Package
Boot             Bootstrap R (S-Plus) Functions  (Canty)
class           Functions for Classification
cluster         Cluster Analysis Extended Rousseeuw  et al.
codetools       Code Analysis Tools for R
datasets        The R Datasets Package
DBI             R Database Interface
foreign         Read Data Stored by Minitab, S, SAS,
      SPSS, Stata, Systat, dBase, ...
graphics        The R Graphics Package
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
string3.r
string4.r
string5.r
string6.r
R data sets
R packages available
R packages available
string7.r
string8.r
Packages in library ‘C:/Users/User/AppData/Local/R/win-library/4.3’:
askpass Safe Password Entry for R, Git, and SSH
backports Reimplementations of Functions Introduced Since R-3.0.0
base64enc Tools for base64 encoding
bit Classes and Methods for Fast Memory-Efficient Boolean Selections
bit64 A S3 Class for Vectors of 64bit Integers
blob A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')
broom Convert Statistical Objects into Tidy Tibbles
bslib Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'
cachem Cache R Objects with Automatic Pruning
callr Call R from R
cellranger Translate Spreadsheet Cell Ranges to Rows and Columns
cli Helpers for Developing Command Line Interfaces
clipr Read and Write from the System Clipboard
colorspace A Toolbox for Manipulating and Assessing Colors and Palettes
commonmark High Performance CommonMark and Github Markdown Rendering in R
conflicted An Alternative Conflict Resolution Strategy
cpp11 A C++11 Interface for R's C Interface
crayon Colored Terminal Output
curl A Modern and Flexible Web Client for R
Console
Background Jobs
R 4.3.1 · ~/
downloaded 482 KB
package ‘bit64’ successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\User\AppData\Local\Temp\Rtmp4CYNRr\downloaded_packages
> library()
> library()
>
Environment
History
Connect
Files
Plots
Packages
Install
Update
User Library
Search
ENG
IN
20:35
19-07-2023

The Popularity of R by Industry

Thanks to its versatility, many different industries use the R programming language. Here is a list of industries/disciplines that use the R programming language:

- Fintech Companies (financial services)
- Academic Research
- Government (FDA, National Weather Service)
- Retail
- Social Media
- Data Journalism
- Manufacturing
- Healthcare

Some Popular R Books

- A First Course in Statistical Programming with R. Braun, W. and Murdoch, D. (2007). Cambridge, MA: Cambridge University Press.
- R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. Wickham, H. (Author), Grolemund, G. (2017). O’Reilly Media.
- Programming with Data: A Guide to the S Language. Chambers, J. M. (1998). Murray Hill, NJ: Bell Laboratories.
- Introductory Statistics with R (2nd edition). Dalgaard, P. (2008). New York: Springer.
- A Handbook of Statistical Analyses Using R. Everitt, B., and Hothorn, T. (2006). Boca Raton, FL: Chapman & Hall/CRC.
- Learning R: A Step-by-Step Function Guide to Data Analysis. Cotton, R. (2013). O’Reilly Media.
- R for Everyone: Advanced Analytics and Graphics. Lander, J. (2017). Addison-Wesley Professional; 2nd edition.
- Linear Models with R. Faraway, J. J. (2005). Boca Raton, FL: Chapman & Hall/CRC.
- Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models. Faraway, J. J. (2006). Boca Raton, FL: Chapman & Hall/CRC.
- An R and S-Plus Companion to Applied Regression. Fox, J. (2002). Thousand Oaks, CA: Sage Publications.
- R for SAS and SPSS Users. Springer Series in Statistics and Computing. Muenchen, R. A. (2009). New York: Springer.
- R Cookbook: Proven Recipes for Data Analysis, Statistics, and Graphics. Long, J.D. and Teetor, P. (2019). O’Reilly Media; 2nd edition.

Outline

- Data Types
- Importing Data
- Keyboard Input
- Database Input
- Exporting Data
- Viewing Data
- Variable Labels
- Value Labels
- Missing Data
- Date Values

Data Types

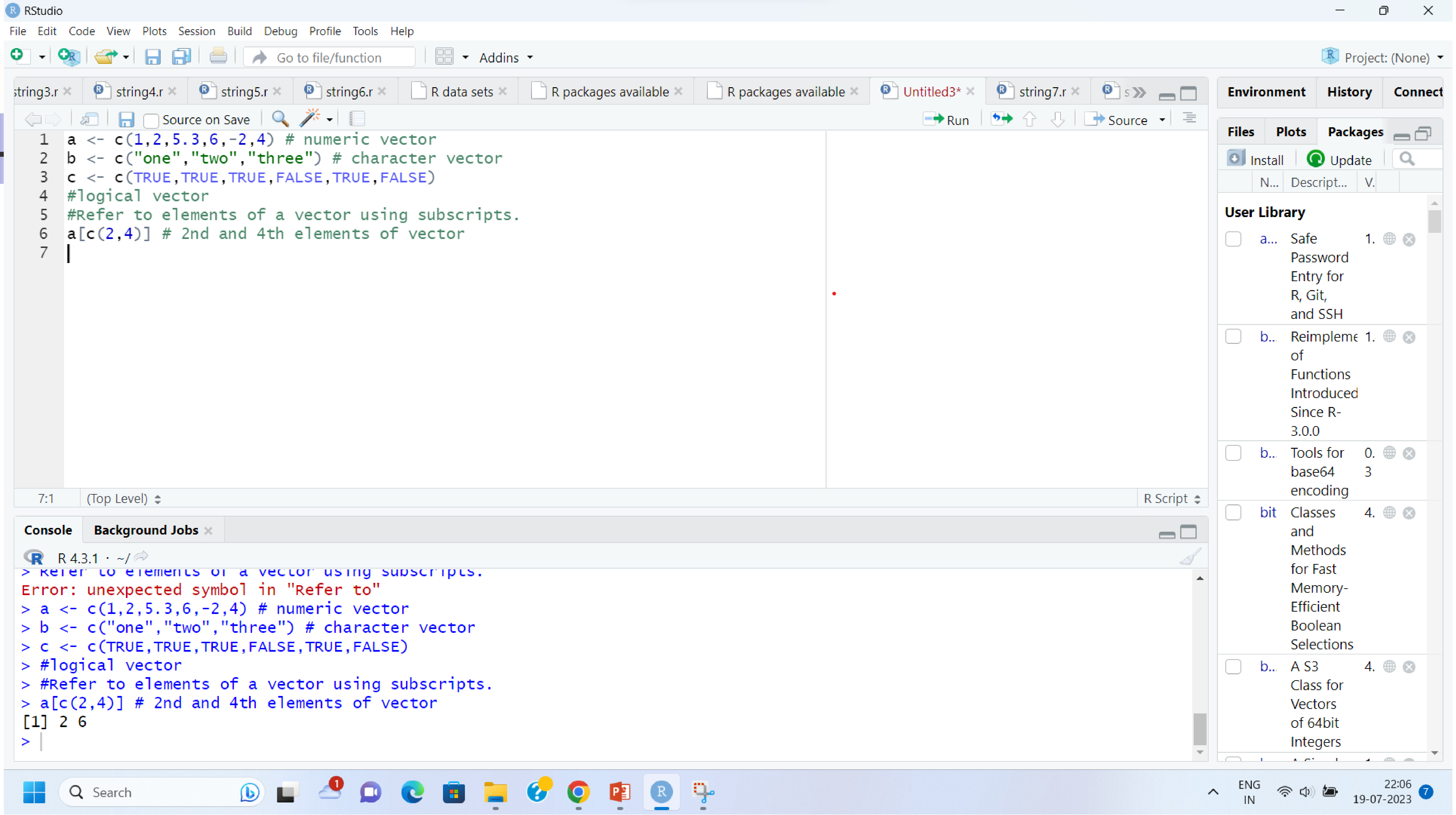
R has a wide variety of data types including scalars, vectors (numerical, character, logical), matrices, dataframes, and lists.

Vectors

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
#logical vector
```

Refer to elements of a vector using subscripts.

```
a[c(2,4)] # 2nd and 4th elements of vector
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
string3.r string4.r string5.r string6.r R data sets R packages available R packages available Untitled3* string7.r
Source on Save
Run
Source
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
#logical vector
#Refer to elements of a vector using subscripts.
a[c(2,4)] # 2nd and 4th elements of vector
7:1
(Top Level)
R Script
Console
Background Jobs
R 4.3.1 · ~/
Error: unexpected symbol in "Refer to"
> a <- c(1,2,5.3,6,-2,4) # numeric vector
> b <- c("one","two","three") # character vector
> c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)
> #logical vector
> #Refer to elements of a vector using subscripts.
> a[c(2,4)] # 2nd and 4th elements of vector
[1] 2 6
>
Environment
History
Connect
Files
Plots
Packages
Install
Update
User Library
Safe Password Entry for R, Git, and SSH
Reimpleme of Functions Introduced Since R-3.0.0
Tools for base64 encoding
bit
Classes and Methods for Fast Memory-Efficient Boolean Selections
A S3 Class for Vectors of 64bit Integers
Search
ENG IN
22:06
19-07-2023

Matrices

All columns in a matrix must have the same mode(numeric, character, etc.) and the same length.

The general format is

mymatrix <- **matrix(***vector*, **nrow=***r*, **ncol=***c*, **byrow=***FALSE*,**dimnames=list(***char_vector_rownames*, *char_vector_colnames***))**

byrow=TRUE indicates that the matrix should be filled by rows. **byrow=FALSE** indicates that the matrix should be filled by columns (the default). **dimnames** provides optional labels for the columns and rows.

Matrices

```
# generates 5 x 4 numeric matrix
    y<-matrix(1:20, nrow=5,ncol=4)
# another example
    cells <- c(1,26,24,68)
    rnames <- c("R1", "R2")
    cnames <- c("C1", "C2")
    mymatrix <- matrix(cells, nrow=2, ncol=2,
    byrow=TRUE, dimnames=list(rnames, cnames))
#Identify rows, columns or elements using subscripts.
x[,4] # 4th column of matrix
    x[3,] # 3rd row of matrix
    x[2:4,1:3] # rows 2,3,4 of columns 1,2,3
```

Arrays

Arrays are similar to matrices but can have more than two dimensions. See **help(array)** for details.

Data frames

A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.).

```
d <- c(1,2,3,4)
e <- c("red", "white", "red", NA)
f <- c(TRUE,TRUE,TRUE,FALSE)
mydata <- data.frame(d,e,f)
names(mydata) <- c("ID","Color","Passed")
  #variable names
```

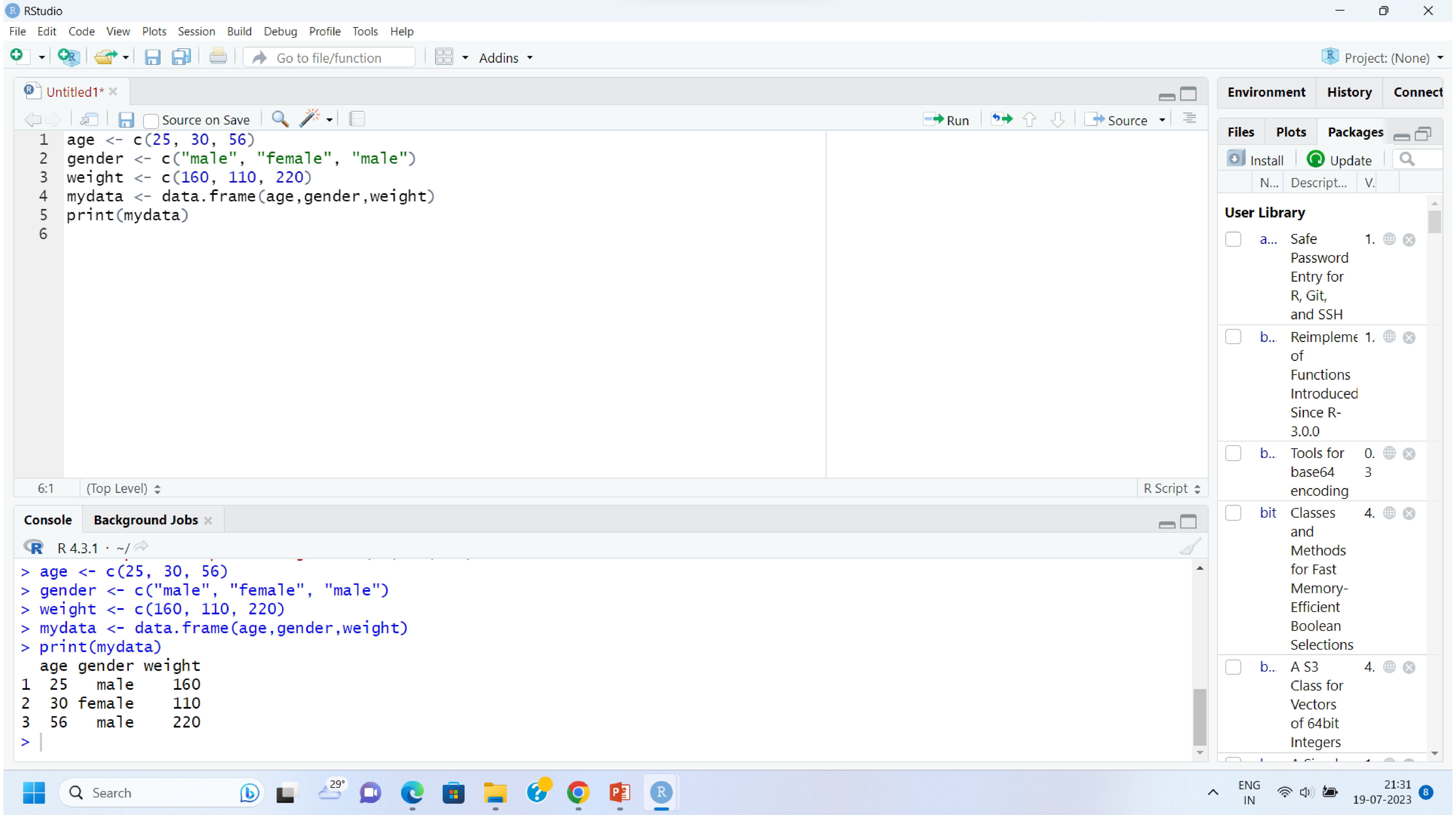
Data frames

There are a variety of ways to identify the elements of a dataframe .

myframe[3:5] # columns 3,4,5 of dataframe

myframe[c("ID","Age")] # columns ID and Age from dataframe

myframe$X1 # variable x1 in the dataframe


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 age <- c(25, 30, 56)
2 gender <- c("male", "female", "male")
3 weight <- c(160, 110, 220)
4 mydata <- data.frame(age,gender,weight)
5 print(mydata)
6
6:1
(Top Level)
R Script
Console
Background Jobs
R 4.3.1 · ~/
> age <- c(25, 30, 56)
> gender <- c("male", "female", "male")
> weight <- c(160, 110, 220)
> mydata <- data.frame(age,gender,weight)
> print(mydata)
age gender weight
1 25 male 160
2 30 female 110
3 56 male 220
>
Environment
History
Connect
Files
Plots
Packages
Install
Update
N...
Descript...
V.
User Library
a...
Safe Password Entry for R, Git, and SSH
1.
b..
Reimpleme of Functions Introduced Since R-3.0.0
1.
b..
Tools for base64 encoding
0.3
bit
Classes and Methods for Fast Memory-Efficient Boolean Selections
4.
b..
A S3 Class for Vectors of 64bit Integers
4.
Search
29°
ENG
IN
21:31
19-07-2023

Lists

An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```
# example of a list with 4 components -
# a string, a numeric vector, a matrix, and a scaler
w <- list(name="Fred", mynumbers=a, mymatrix=y,
age=5.3)

# example of a list containing two lists
v <- c(list1,list2)
```

Lists

Identify elements of a list using the [[]] convention.

mylist[[2]] # 2nd component of the list

Factors

Tell **R** that a variable is **nominal** by making it a factor. The factor stores the nominal values as a vector of integers in the range [1... k] (where k is the number of unique values in the nominal variable), and an internal vector of character strings (the original values) mapped to these integers.

```
# variable gender with 20 "male" entries and
# 30 "female" entries
gender <- c(rep("male",20), rep("female", 30))
gender <- factor(gender)
# stores gender as 20 1s and 30 2s and associates
# 1=female, 2=male internally (alphabetically)
# R now treats gender as a nominal variable
summary(gender)
```

Useful Functions

length(object) # number of elements or components
str(object) # structure of an object
class(object) # class or type of an object
names(object) # names
c(object,object,...) # combine objects into a vector
cbind(object, object, ...) # combine objects as columns
rbind(object, object, ...) # combine objects as rows
ls() # list current objects
rm(object) # delete an object
newobject <- edit(object) # edit copy and save a newobject
fix(object) # edit in place

Importing Data

Importing data into **R** is fairly simple.

For Stata and Systat, use the **foreign** package.

For SPSS and SAS I would recommend the **Hmisc** package for ease and functionality.

See the **Quick-R** section on **packages**, for information on obtaining and installing the these packages.

Example of importing data are provided below.

From A Comma Delimited Text File

first row contains variable names, comma is separator
assign the variable *id* to row names
note the / instead of \ on mswindows systems

```
mydata <- read.table("c:/mydata.csv",
header=TRUE, sep=",", row.names="id")
```

From Excel

The best way to read an Excel file is to export it to a comma delimited file and import it using the method above.

On windows systems you can use the **RODBC** package to access Excel files. The first row should contain variable/column names.

```
# first row contains variable names
# we will read in workSheet mysheet
library(RODBC)
channel <- odbcConnectExcel("c:/myexel.xls")
mydata <- sqlFetch(channel, "mysheet")
odbcClose(channel)
```

From SAS

- # save SAS dataset in trasport format
```
libname out xport 'c:/mydata.xpt';
data out.mydata;
set sasuser.mydata;
run;
```
- library(foreign)

```
#bsl=read.xport("mydata.xpt")
```

Keyboard Input

Usually you will obtain a dataframe by importing it from **SAS**, **SPSS**, **Excel**, **Stata**, a database, or an ASCII file. To create it interactively, you can do something like the following.

```
# create a dataframe from scratch
age <- c(25, 30, 56)
gender <- c("male", "female", "male")
weight <- c(160, 110, 220)
mydata <- data.frame(age,gender,weight)
```

Keyboard Input

You can also use **R**'s built in spreadsheet to enter the data interactively, as in the following example.

```
# enter data using editor
mydata <- data.frame(age=numeric(0),
gender=character(0), weight=numeric(0))
mydata <- edit(mydata)
# note that without the assignment in the line
above,
# the edits are not saved!
```

Exporting Data

There are numerous methods for exporting **R** objects into other formats . For SPSS, SAS and Stata. you will need to load the **foreign** packages. For Excel, you will need the **xlsReadWrite** package.

Exporting Data

To A Tab Delimited Text File

```
write.table(mydata, "c:/mydata.txt", sep="\t")
```

To an Excel Spreadsheet

```
library(xlsReadWrite)
  write.xls(mydata, "c:/mydata.xls")
```

To SAS

```
library(foreign)
  write.foreign(mydata, "c:/mydata.txt",
  "c:/mydata.sas",   package="SAS")
```

Viewing Data

There are a number of functions for listing the contents of an object or dataset.

```
# list objects in the working environment
ls()

# list the variables in mydata
names(mydata)

# list the structure of mydata
str(mydata)

# list levels of factor v1 in mydata
levels(mydata$v1)

# dimensions of an object
dim(object)
```

Viewing Data

There are a number of functions for listing the contents of an object or dataset.

```
# class of an object (numeric, matrix, dataframe, etc)
class(object)

# print mydata
mydata

# print first 10 rows of mydata
head(mydata, n=10)

# print last 5 rows of mydata
tail(mydata, n=5)
```

Variable Labels

R's ability to handle variable labels is somewhat unsatisfying.

If you use the **Hmisc** package, you can take advantage of some labeling features.

library(Hmisc)
label(mydata$myvar) <- "Variable label for variable *myvar*"
describe(mydata)

Variable Labels

Unfortunately the label is only in effect for functions provided by the **Hmisc** package, such as **describe()**. Your other option is to use the variable label as the variable name and then refer to the variable by position index.

names(mydata)[3] <- "This is the label for variable 3"
mydata[3] # list the variable

Value Labels

To understand value labels in **R**, you need to understand the data structure factor.

You can use the factor function to create your own value lables.

variable v1 is coded 1, 2 or 3

we want to attach value labels 1=red, 2=blue,3=green

mydata$v1 <- factor(mydata$v1,
levels = c(1,2,3),
labels = c("red", "blue", "green"))

variable y is coded 1, 3 or 5

we want to attach value labels 1=Low, 3=Medium, 5=High

Value Labels

```
mydata$v1 <- ordered(mydata$y,
    levels = c(1,3, 5),
    labels = c("Low", "Medium", "High"))
```

Use the **factor()** function for **nominal data** and the **ordered()** function for **ordinal data**. **R** statistical and graphic functions will then treat the data appropriately.

Note: factor and ordered are used the same way, with the same arguments. The former creates factors and the later creates ordered factors.

Missing Data

In **R**, missing values are represented by the symbol **NA** (not available) . Impossible values (e.g., dividing by zero) are represented by the symbol **NaN** (not a number). Unlike SAS, **R** uses the same symbol for character and numeric data.

Testing for Missing Values

is.na(x) # returns TRUE of x is missing

y <- c(1,2,3,NA)

is.na(y) # returns a vector (F F F T)

Missing Data

Recoding Values to Missing

```
# recode 99 to missing for variable v1
# select rows where v1 is 99 and recode column v1
mydata[mydata$v1==99,"v1"] <- NA
```

Excluding Missing Values from Analyses

Arithmetic functions on missing values yield missing values.

```
x <- c(1,2,NA,3)
mean(x)          # returns NA
mean(x, na.rm=TRUE) # returns 2
```

Missing Data

The function **complete.cases()** returns a logical vector indicating which cases are complete.

```
# list rows of data that have missing values
mydata[!complete.cases(mydata),]
```

The function **na.omit()** returns the object with listwise deletion of missing values.

```
# create new dataset without missing data
newdata <- na.omit(mydata)
```

Missing Data

Advanced Handling of Missing Data

Most modeling functions in **R** offer options for dealing with missing values. You can go beyond pairwise of listwise deletion of missing values through methods such as multiple imputation. Good implementations that can be accessed through **R** include **Amelia II**, **Mice**, and **mitools**.

Date Values

Dates are represented as the number of days since 1970-01-01, with negative values for earlier dates.

```
# use as.Date( ) to convert strings to dates
mydates <- as.Date(c("2007-06-22", "2004-02-13"))
# number of days between 6/22/07 and 2/13/04
days <- mydates[1] - mydates[2]
```

Sys.Date( ) returns today's date.

Date() returns the current date and time.

Date Values

The following symbols can be used with the format( ) function to print dates.

Symbol	Meaning	Example
%d	day as a number (0-31)	01-31
%a **%A**	abbreviated weekday unabbreviated weekday	Mon Monday
%m	month (00-12)	00-12
%b **%B**	abbreviated month unabbreviated month	Jan January
%y **%Y**	2-digit year 4-digit year	07 2007

Date Values

 # print today's date

```
today <- Sys.Date()
format(today, format="%B %d %Y")
    "June 20 2007"
```