



Data manipulation and visualization with R

1.1 Introduction to RStudio



1.1 Introduction to RStudio

Programming with R requires two components: The programming language R and an Integrated Development Environment (IDE). There exists a few IDEs for R, but RStudio is the one most commonly used. RStudio is available for all operating systems.

First we will look at RStudio itself before we learn more about the programming language.

This includes:

- The Graphical User Interface (GUI) and its four main sections:
 - Scripts
 - Console
 - Environment and History
 - Miscellaneous
- Sessions
- Installation guide

Please note: On the following slides we will show RStudio with a dark theme applied. This helps to reduce eye strain, as many people tire easily when looking at mostly white computer screens. Darker screen displays can also reduce energy consumption. The theme can be changed under “Tools / Global Options / Appearance”. In this case, we applied the “Cobalt” scheme.

The Graphical User Interface (GUI)

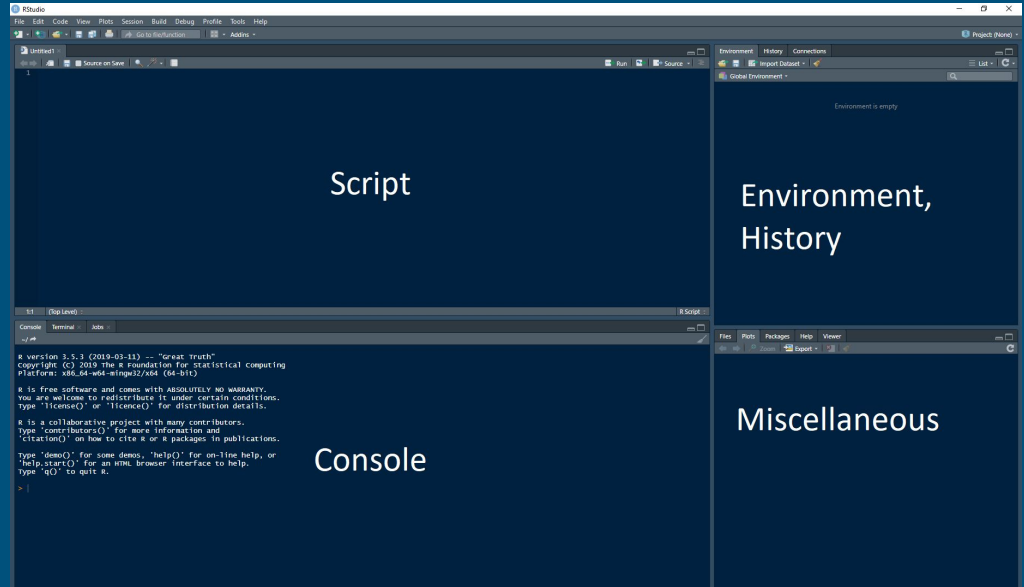
The Graphical User Interface (GUI)

RStudio is an **Integrated Development Environment (IDE)**: a software that helps programmers code, debug and compile their code. While some IDEs support multiple programming languages, RStudio only supports R.

The **GUI** supplies everything a user would need to write and execute R code. The main sections are:

- Script
- Console
- Environment and History
- Miscellaneous
- Menu bar

The following slides will go into detail about each individual section and its main functions. Please note that not every detail will be covered here.



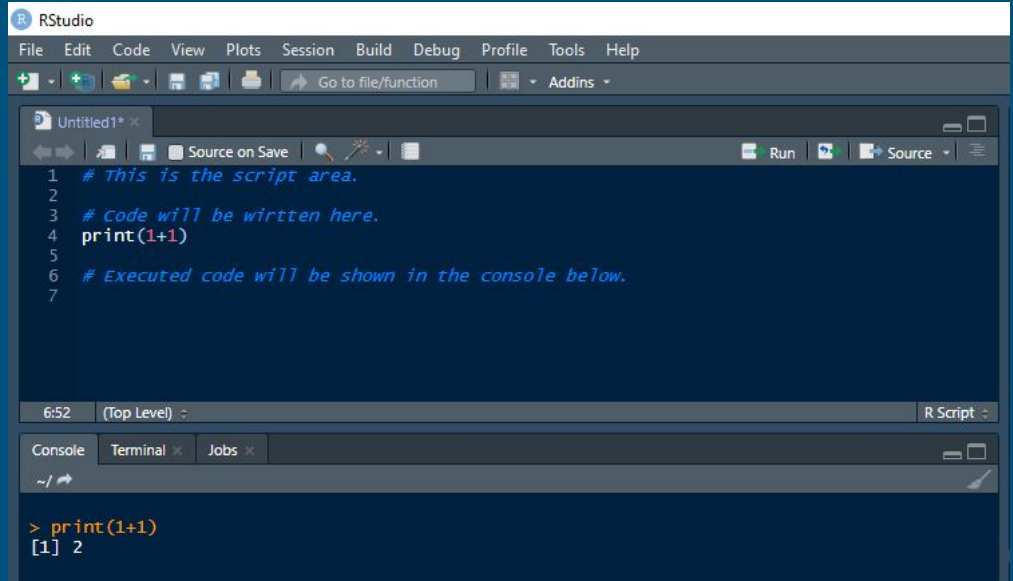
The Graphical User Interface (GUI)

The script

The user will spend most of their time working in the **script** area in the top left corner. Most of the code is written there. **The script works like a text document and can be saved and loaded** via “File” in the menu bar.

The “Run” button will execute the code currently selected in the script (shortcut: Ctrl+Enter). Results will be shown in the console below. If the whole script is selected (Ctrl+A), everything will be executed at once.

As R and RStudio are often used for statistical analysis and reporting, the processing of data should be as transparent as possible. While the console can also be used to execute code, only the script saves all processing steps and creates a “cooking recipe” documenting all of the user’s steps in the process.



The screenshot displays the RStudio interface. At the top, there is a menu bar with options: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main workspace is divided into two panes. The upper pane, titled 'Untitled1*', contains a script with the following code:

```
1 # This is the script area.
2
3 # Code will be written here.
4 print(1+1)
5
6 # Executed code will be shown in the console below.
7
```

The lower pane is the console, which shows the output of the executed code:

```
> print(1+1)
[1] 2
```

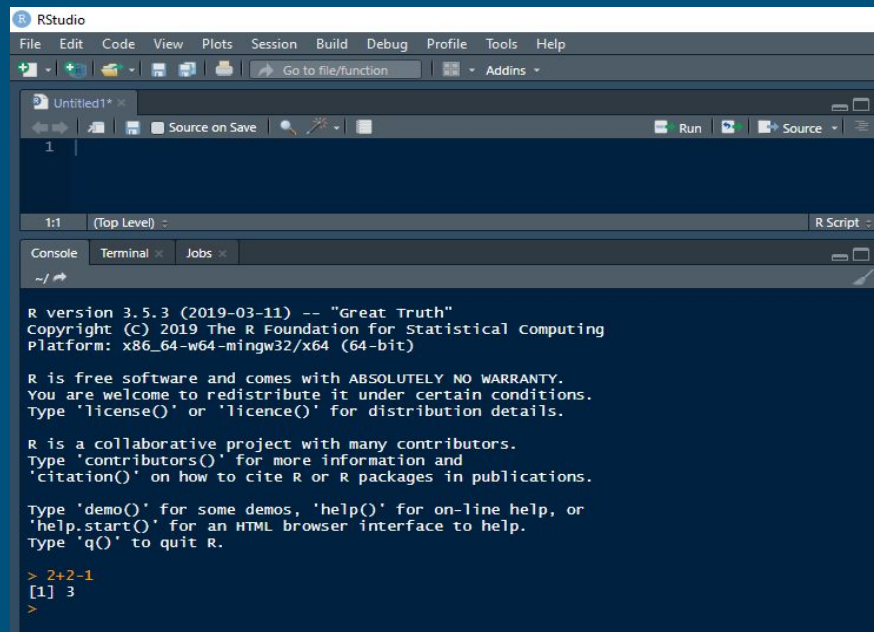
The Graphical User Interface (GUI)

The console

The **console** is right below the script area. **Every code that is executed in the script, as well as its results, will be shown here.** The console can also be used directly to quickly execute code. However, this is not recommended for code consisting of more than a few lines as the “cooking recipe” will not be saved as a document. The console can be used for intermediate steps that do not need to be documented in the script, e.g. checking whether a datafile was correctly imported.

The console also helps with debugging, i.e. findings errors in the code. Here, error messages will be shown, e.g. the use of an unsupported data type within a function.

Upon opening RStudio, the console also shows the current version of R installed on the user’s computer.



```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Untitled1* x
1
1:1 (Top Level) R Script
Console Terminal Jobs
~/
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 2+2-1
[1] 3
>
```

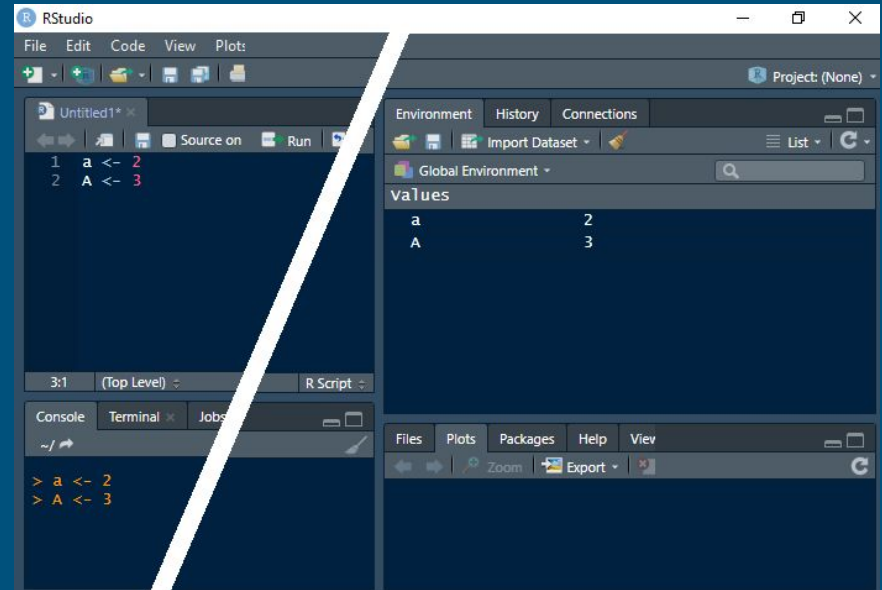
The Graphical User Interface (GUI)

Environment and History

The “Environment” tab is located on the right side and displays all variables that are currently in use.

The picture to the right shows a script where the variables “a” and “A” are declared as 2 and 3. The results are also shown in the console indicating that the code has already been executed. The environment then displays the variables and their values.

The environment has different layers. The “Global Environment” contains variables that are accessible to all functions. Variables stored in local environments are only accessible within certain functions or loops. As every variable name can only exist once in each environment, this layered structure helps to avoid the accidental overwriting of certain variable names by variables automatically defined within functions.



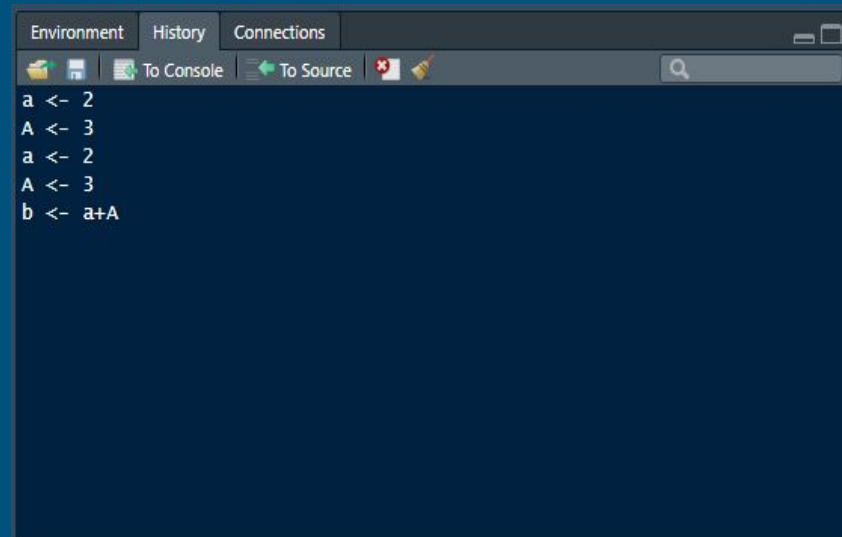
The Graphical User Interface (GUI)

Environment and History

The “**History**” tab is located next to the Environment tab. It contains a **chronological collection of all lines of code that were executed** in the console. Unlike the code shown in the console, the history persists over all sessions.

Via the icons, the history can be saved as files and loaded again. However, the history will also be saved automatically after each execution. Selected lines can be executed again (“To Console”) or even copied to the active script (“To Source”). The last two icons in the menu bar either delete selected lines or the entire history.

The history can be useful to look at previously executed code -- especially after a crash when the script and the console were not saved. However, code should always be saved in a script, where it can be formatted and maintained in a reader-friendly manner.

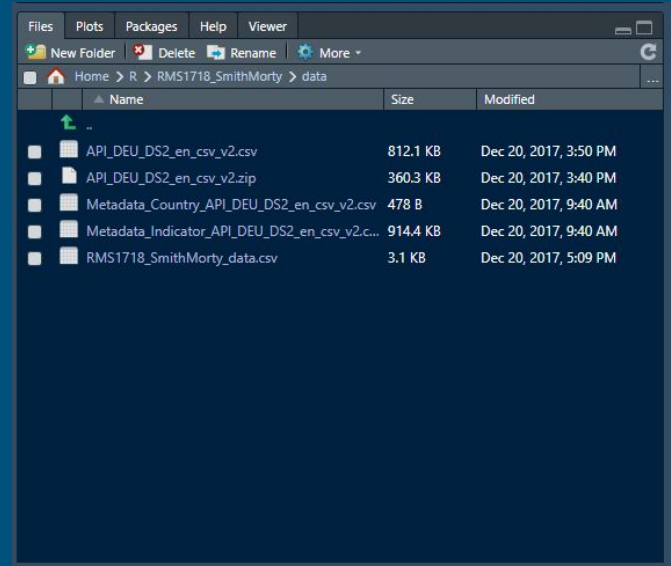


The Graphical User Interface (GUI)

Miscellaneous

The bottom right side of the window shows a collection of several tabs, including a file manager, a window for graphics, a summary of installed packages and a help section.

The “Files” tab shows a file tree that can be used to **organize data within folders and sub-folders**. Within the tab “More” there are also options to change the working directory, i.e. the folder and its contents currently accessible to the program and code. However, there are better ways to set working directories (-> see chapter 2.1). The button “...” opens the explorer.



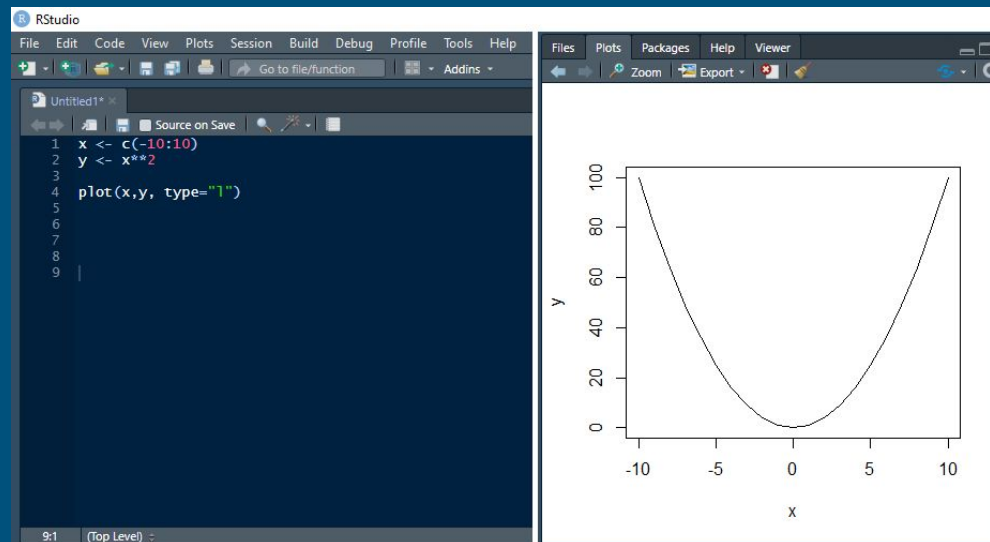
The Graphical User Interface (GUI)

Miscellaneous

The “Plots” tab displays graphs created by code executed in the console. As the window can only show one plot at a time, left and right arrows in the tab’s menu bar can be used to switch between all plots created within the current session.

The “Zoom” option opens a new and bigger window with a higher resolution. The “Export” tab enables the user to save plots as raster or vector data. Please note that plots not saved manually (or explicitly by the code) will be lost upon ending the current session.

If the “Plots” window is resized too small to actually show a plot, an error message (“figure margins too large”) can occur when trying to create a plot. In this case, the window area has to be increased.



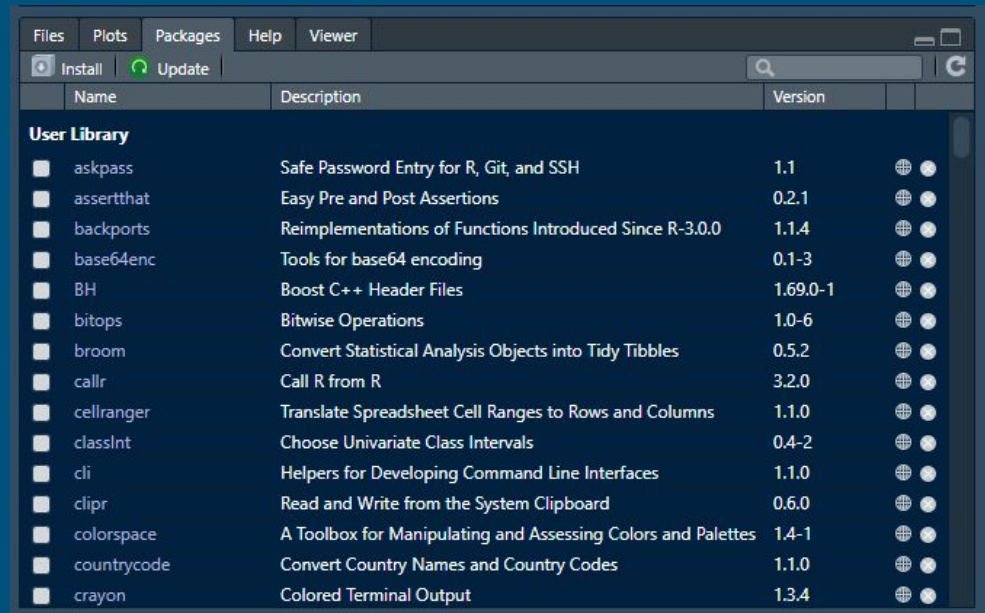
The Graphical User Interface (GUI)

Miscellaneous

The “**Packages**” tab lists all installed packages as well as their installed versions. **Packages are downloadable extensions** to the standard (“Vanilla”) code providing additional functions. R is an open source programming language which means that every user can create new functions and may share them with others by creating a package. The chapter 1.2 will focus more on packages.

The “**Install**” button provides the user with a list of downloadable packages. The “**Update**” button installs the latest version of each installed package. (This is not always recommended when a specific older version of package is required.)

Packages have to be activated (indicated by a tick) each session they are needed.



The Graphical User Interface (GUI)

Miscellaneous

The “[Help](#)” tab provides a concise overview of functions, including a description, its arguments and a few examples.

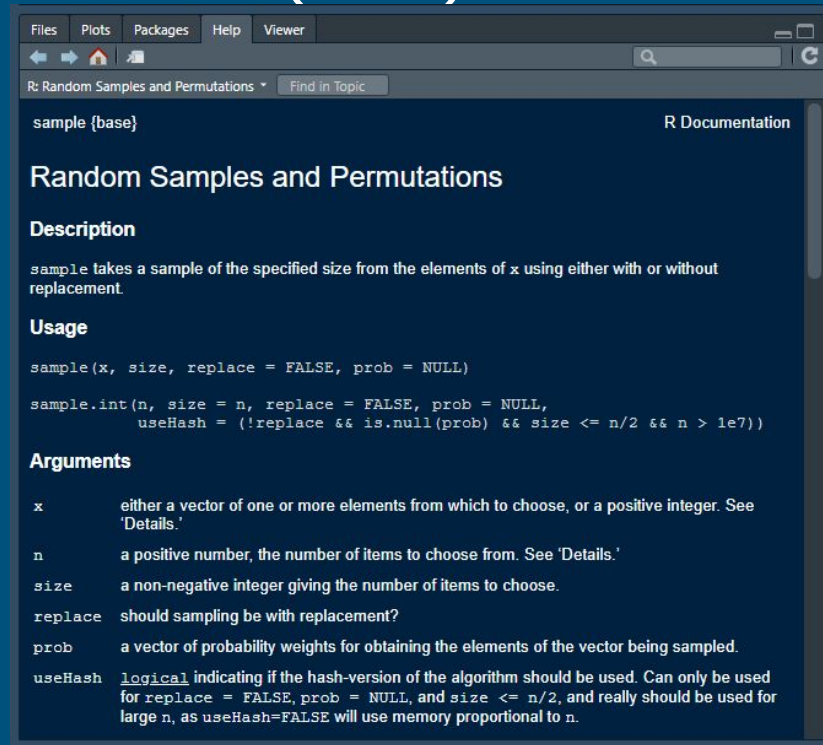
To open the help section of a specific function, the user can enter “?” followed by the function’s name (without parentheses):

```
# get help section of sample() function
?sample
```

Functions included in packages need their package name followed by “::” to access their help section:

```
# get help section of gather() function from “tidyr” package
?tidyr::gather
```

Please note: Code starting with “#” contains annotations made by the user; they are not necessary to open the help section.



The screenshot shows the RStudio interface with the 'Help' tab selected. The main pane displays the help page for the `sample` function. The page title is 'Random Samples and Permutations'. The content includes a description, usage examples, and a list of arguments.

sample {base} R Documentation

Random Samples and Permutations

Description

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

```
sample(x, size, replace = FALSE, prob = NULL)
sample.int(n, size = n, replace = FALSE, prob = NULL,
           useHash = (!replace && is.null(prob) && size <= n/2 && n > 1e7))
```

Arguments

<code>x</code>	either a vector of one or more elements from which to choose, or a positive integer. See 'Details.'
<code>n</code>	a positive number, the number of items to choose from. See 'Details.'
<code>size</code>	a non-negative integer giving the number of items to choose.
<code>replace</code>	should sampling be with replacement?
<code>prob</code>	a vector of probability weights for obtaining the elements of the vector being sampled.
<code>useHash</code>	<code>logical</code> indicating if the hash-version of the algorithm should be used. Can only be used for <code>replace = FALSE</code> , <code>prob = NULL</code> , and <code>size <= n/2</code> , and really should be used for large <code>n</code> , as <code>useHash=FALSE</code> will use memory proportional to <code>n</code> .

Sessions

Sessions

Everytime the user opens the program, a new session starts. Unlike programs like Word or Excel, RStudio may require a few preparatory steps before the user can repeat working on a script or an analysis after closing the program (or an unwanted termination). This may include the following procedures:

Set working directories (see chapter 2.1):

RStudio needs to know where external files are stored, e.g. a datafile that should be accessible or a folder where graphs shall be saved as files.

Reload packages :

Packages are downloadable plugins that provide further functions (see chapter 1.2). While packages only have to be downloaded once, the user has to decide which packages should be activated for the current session. This can be done manually or via the script. Not every package has to be activated for every analysis or every session.

Load data into environment:

Data loaded from external sources or created in a prior session, will be lost upon reopening the program. It is recommended to load data that should be available across different sessions assigned via the script to have an easy way to reload or recreate the data.

Installation guide

Installation

The installation of R and RStudio is easy in comparison to other programming languages. Please note: **Both R and RStudio must be installed.**

R can be downloaded from <https://cran.r-project.org/> (CRAN: Comprehensive R Archive Network) for Windows, Mac OS X or Linux. The following link contains installation guides for all three operating systems: https://cran.r-project.org/doc/FAQ/R-FAQ.html#How-can-R-be-installed_003f

Once R is installed, the program “R GUI” will appear on the desktop and the software menus. This IDE can also be used to code with R, but lacks many functions present in RStudio. We therefore highly recommend to install RStudio and use this IDE instead. The software can be downloaded here: <https://rstudio.com/products/rstudio/download/>. For the purpose of this tutorial, the free Desktop version suffices. For commercial uses, a licence might be required. Please investigate which licence model you will need before using the software commercially.

The latest download versions for R and RStudio are usually the preferable options. This course will not discuss explicit version control when used in e.g. virtual environments, as the lessons only provide a beginner’s introduction to the language R and the software RStudio.

The next lesson will focus on fundamental concepts of the R language.