Introduction to R

Lab Session: This session provides a set of beginners to intermediate level exercises. No prior experience to any programming language is required.

Estimated time for completion: 60 minutes

Contents

Introduction

With more than 2 000 000 users and 11 000 additional packages designed from experts, R must definitely not be underestimated. It allows scientists to apply intricate and complicated statistical analysis without the necessity of advanced computing skills and statistics. It provides packages for different fields such as Astrophysics and Astronomy, Chemistry, Energy Modelling, Finance, Genetics, High Performance Computing, Machine Learning, Medical Imaging, Social Sciences, Spatial Statistics, Spectroscopy, Thermodynamics and many more.

Packages Needed

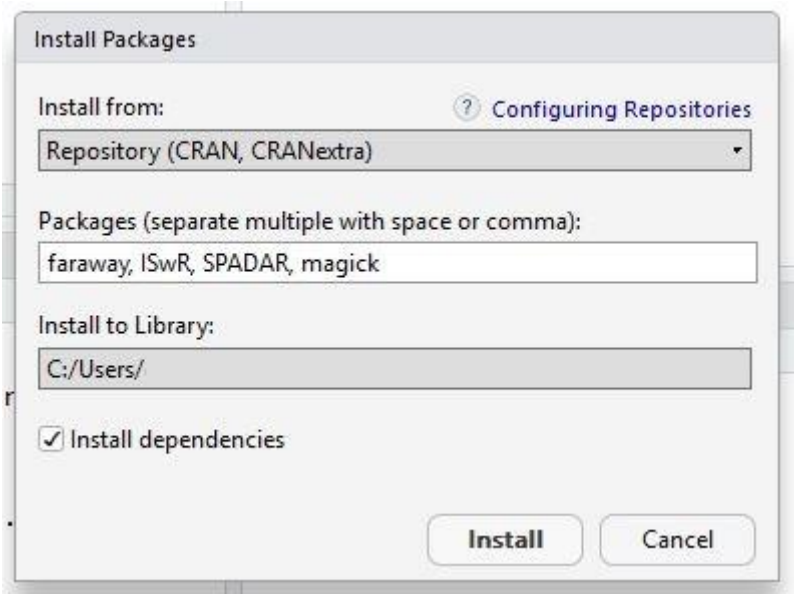
If you are using a Virtual Machine, please skip the following steps. You can go directly to page 3.

Before installing any package, a CRAN mirror needs to be used.

To do so, go to *Tools > Global Options > Packages > CRAN mirror* and choose a mirror. For this tutorial choose Global (CDN) - RStudio mirror (if it is not already selected).

For the purposes of this lab, three additional packages will be needed. SPADAR, magick, audio. It is recommended to download these packages at this step for saving time and possible function basis errors that might appear later on if a package has not been installed.

For installing a package in RStudio use the command *install.packages("name_of_package")*. Otherwise, go to *Tools > Install Packages*, type the package(s) needed and click ok.



The installation may take some time, depending on the size of the package(s).

Reading Data from Files

RStudio is capable of reading a huge amount of data, from text files to excel files and even SPSS, RDA and CDF files. For more information on reading different kinds of files you can click here and for CDF files click here.

This tutorial is focused on reading data from text files (.txt) and from generic data files (.dat).

Note: Before reading a file, it is essential to inform R where to find the file. In other words, the working directory has to be defined. To do so, go to *Session > Set Working Directory > Choose Directory* and choose the directory at which the files are listed. Otherwise using the command line:

Write *getwd()* to find out the current working directory that RStudio is currently using. Then, use the command *setwd("directory")* to change the directory.

Example:

```
setwd(C:/Users/Documents/University/ACM2)
```

Notice that forward slash "**/**" is used instead of backslash "****". Finally, check to make sure that the directory is changed using the command *getwd()*.

How to read .txt files

To read .txt files in RStudio use the function *read.table()*. This function comes with a number of required and optional arguments listed below:

- name of the file (required).
- header = TRUE (if the .txt file includes a first row with the name of each column). Otherwise, header = FALSE (optional).
- sep = "" or sep = "," or sep = "\t" or sep = "|" (if the data columns in the .txt file are separated by white spaces, by commas, by tabs or by bars respectively) (optional).
- skip = 10 (if the first 10 rows of the .txt file include comments and must be ignored by RStudio) (optional).
- as.is = TRUE (to prevent RStudio from automatically converting character variables, such as date variables or ID variables, into factors). Otherwise, as.is = FALSE (optional).
- na.strings = "NA" (if missing values are coded as NA in the .txt file) or
  na.strings = "" (if missing values are coded as blank cells in the .txt file) (optional).

For more information use R's help command for this function *?read.table*.

Example:

Reading files from the Internet:

```
filename <- "http://www.101computing.net/wp/wp-content/uploads/US-States.txt"
data <- read.table(filename, header = FALSE, sep = ",", as.is = TRUE)
head(data)
```

Reading files from the working directory:

```
data <- read.table("ABELL_0085.txt", header = TRUE, sep = "|", skip = 16, as.is = TRUE)
View(data)
```

How to read .dat files

To read .dat files in RStudio you can use the function *read.delim()*. This function comes with a number of required and optional arguments listed below:

- name of the file (required).
- header = TRUE (if the .dat file includes a first row with the name of each column). Otherwise, header = FALSE (optional).
- sep = "" or sep = "," or sep = "\t" or sep = "|" (if the data columns in the .dat file are separated by white spaces, by commas, by tabs or by bars respectively) (optional).
- skip = 10 (if the first 10 rows of the .dat file include comments and must be ignored by RStudio) (optional).
- as.is = TRUE (to prevent R from automatically converting character variables, such as date variables or ID variables, into factors). Otherwise, as.is = FALSE (optional).
- na.strings = "NA" (if missing values are coded as NA in the .dat file) or
  na.strings = "" (if missing values are coded as blank cells in the .dat file (optional).

For more information use R's help command for this function *?read.delim.*

Example:

Reading files from the Internet:

```
filename <-
"https://www.nilu.no/projects/ccc/onlinedata/ozone/GB0006R_2015.dat"

data <- read.delim(filename, header = FALSE, sep = "", skip = 3, as.is =
TRUE)

head(data)
```

Reading files from the working directory:

```
data <- read.delim("GB0006R_2015.dat", header = FALSE, sep = "", skip = 3,
as.is = TRUE)

View(data)
```

Exercise 1

This exercise lies in the field of astrostatistics. Astrostatistics combine astrophysics and statistical analysis to process massive amounts of astronomical data collected from different research centres all over the world. In general, a statistical problem can be divided into six main sections.

1. understanding the scientific problem,
2. exploring the data,
3. converting the problem into mathematical equations,
4. choosing the appropriate statistical method to solve the problem,
5. calculating the statistical quantities and
6. evaluating the results and making observations.

When it comes to astrophysical problems, the amount of data to be processed is huge. As a result, the calculation of these statistical quantities and the evaluation of the results becomes easier using R.

For this exercise the package "SPADAR" is needed. This is a package that provides functions for **S**pherical **P**rojections of **A**stronomical **DA**ta in **R**. It uses equatorial, ecliptic and galactic coordinate systems and scatter plots of the data we provide. Ideally, this package is successfully installed by now. Otherwise, please advise the section packages needed. In order to use this package, R needs to be informed about it. To do so, use the command *require(SPADAR)* or *library(SPADAR)*. For more information click here.

The goal now is to use some data from a .dat file and plot the results on the screen. For using external data from .dat files see the section how to read .dat files. In order to create the chart needed, use the function *createAllSkyGridChart()*.

Arguments for this function:

- longitude A vector with the position of the first coordinate in degrees (Right Ascension for data in the Equatorial system, Ecliptic Longitude for the Ecliptic system and Galactic Longitude for the Galactic System) (optional).
- latitude A vector with the position of the second coordinate in degrees (Declination for data in the Equatorial system, Ecliptic Latitude for the Ecliptic system and Galactic Latitude for the Galactic System) (optional).
- mainGrid String. The main coordinate system of the plot. It can take the following values: "equatorial", "ecliptic" or "galactic". It defaults to "equatorial".
- eqCol String. The colour of the Equatorial coordinate system lines (optional).
- eclCol String. The colour of the Ecliptic coordinate system lines (optional).
- galCol String. The colour of the Galactic coordinate system lines (optional).
- eqLty Numeric. The line type of the Equatorial coordinate system lines (optional).
- eclLty Numeric. The line type of the Ecliptic coordinate system lines (optional).
- galLty Numeric. The line type of the Galactic coordinate system lines (optional).
- eqLwd Numeric. The line width of the Equatorial coordinate system lines (optional).
- eclLwd Numeric. The line width of the Ecliptic coordinate system lines (optional).
- galLwd Numeric. The line width of the Galactic coordinate system lines (optional).
- eqDraw Logical. A boolean to indicate if the Equatorial coordinate system lines should be drawn (optional).
- eclDraw Logical. A boolean to indicate if the Ecliptic coordinate system lines should be drawn (optional).

- galDraq Logical. A boolean to indicate if the Galactic coordinate system lines should be drawn (optional).
- projname String. The spherical projection of the plot. It can take as argument any projection supported by the mapproj package, but only "aitoff", "mollweide" and "mercator" are tested automatically. It defaults to "aitoff".
- projparam Parameters to configure the projection. It defaults to NULL.
- projorient The orientation of the projection. It defaults to NULL.
- npoints Numeric. The number of points used to draw each grid line. Defaults to 50.
- overplot Logical. A boolean to indicate if the grid should be overploted or a new plot should be created (optional).
- addLab Logical. A boolean to indicate if the labels should be added (optional).
- label.cex Numeric. The size of the labels (optional).
- main String. The main title of the plot (optional).

Example:

```
createAllSkyGridChart(mainGrid = "equatorial", eqDraw = TRUE, eclDraw =
TRUE, galDraq = FALSE, eqCol = "green", eclCol = rgb(0,0,0), eqLty = 1,
eclLty = 2)
```

Now, let us plot the data in the chart created. The function used is *createAllSkyScatterPlotChart()*.

Arguments for this function:

- x A vector with the data in degrees for the first coordinate (Right Ascension for data in the Equatorial system, Ecliptic Longitude for the Ecliptic system and Galactic Longitude for the Galactic System) (required).
- y A vector with the data in degrees for the second coordinate (Declination for data in the Equatorial system, Ecliptic Latitude for the Ecliptic system and Galactic Latitude for the Galactic System) (required).
- pointcol A scalar or a vector, containing the colour of the points (optional).
- pointsize A scalar or a vector containing the sizes of the points (optional).
- dataCoordSys String. The name of the coordinate system of the x and y vector. It can take the following values: "equatorial", "ecliptic" or "galactic". It defaults to "equatorial".
- mainGrid String. The name of the main coordinate system of the plot. It can take the following values: "equatorial", "ecliptic" or "galactic". It defaults to "equatorial".
- eqCol String. The colour of the Equatorial coordinate system lines (optional).
- eclCol String. The colour of the Ecliptic coordinate system lines (optional).
- galCol String. The colour of the Galactic coordinate system lines (optional).
- eqLty Numeric. The line type of the Equatorial coordinate system lines (optional).
- eclLty Numeric. The line type of the Ecliptic coordinate system lines (optional).
- galLty Numeric. The line type of the Galactic coordinate system lines (optional).
- eqLwd Numeric. The line width of the Equatorial coordinate system lines (optional).
- eclLwd Numeric. The line width of the Ecliptic coordinate system lines (optional).
- galLwd Numeric. The line width of the Galactic coordinate system lines (optional).
- eqDraw Logical. A boolean to indicate if the Equatorial coordinate system lines should be drawn (optional).
- eclDraw Logical. A boolean to indicate if the Ecliptic coordinate system lines should be drawn (optional).

- galDraq Logical. A boolean to indicate if the Galactic coordinate system lines should be drawn (optional).
- projname String. The spherical projection of the plot. It can take as argument any projection supported by the mapproj package, but only "aitoff", "mollweide" and "mercator" are tested automatically. It defaults to "aitoff".
- projparam Parameters to configure the projection. It defaults to NULL.
- projorient The orientation of the projection. It defaults to NULL.
- nGridpoints Numeric. The number of points used to draw each grid line. Defaults to 50.
- addLab Logical. A boolean to indicate if the labels should be added (optional).
- label.cex Numeric. The size of the labels (optional).
- main String. The main title of the plot (optional).

Example:

```
data <- read.table("Nebulae_and_Clusters_of_Stars.txt", header = TRUE, sep = "\t", as.is = TRUE)[1]

createAllSkyScatterPlotChart(data[,2], data[,3], mainGrid = "galactic", dataCoordSys = "equatorial", pointcol = data[,7], pch = 19, pointsize = 0.5, eqDraw = TRUE, eclDraw = FALSE, galDraq = TRUE, galCol = "black", eqLty = 2, galLty = 1, eqCol = rgb(1,0,0.5))
```

Task:

Using the file “Local_Supercluster.dat” provided in the folder R_Lab, plot a scatter plot of the data. The data file was part of the 2 Micron All Sky Survey Redshift Survey (Crook et al., 2008), (Huchra et al., 2011). It stores all the galaxies within approximately 4500 km/s from the Milky Way. Below you can find a more detailed explanation of each column in the data file.

NAME:	the coordinate name of the object from the 2MASS catalogue
RA:	the right ascension in hours, minutes and seconds and then in degrees (0^o - 360^o)
DEC:	the declination in degrees, minutes and seconds and then in degrees (-90^o - $+90^o$)
Mag:	the K band (2.2 micron) magnitude
V_h	the heliocentric radial velocity
err	the error in the velocity
TYPE	the morphological type of the galaxy as coded by the RC
Dist	the radial distance of the galaxy in Mpc
TL	the galactic longitude of the galaxy in decimal degrees
TB	the galactic latitude of the galaxy in decimal degrees
SGL	the Supergalactic longitude in decimal degrees (0^o - 360^o)
SGB	the Supergalactic latitude in decimal degrees (-90^o - $+90^o$)
SX, SY, SZ	the cartesian supergalactic coordinates in Mpc

[1] The data of the Nebulae_and_Clusters_of_Stars.dat file are provided by NASA/IPAC EXTRAGALACTIC DATABASE

Most of the columns will not be needed for this exercise but they are included for complexity.

You are required to make a plot of these data in galactic coordinates. In the graph, you need to include the equatorial and galactic coordinates but not the ecliptic. The size of each point needs to be relative to its radial distance “Dist”. (Hint: use it as a ratio of the radial distance of each point times two, over the maximum radial distance of all the points). For the colour of each point you need to use the heliocentric radial velocity. The colours should be as follows:

Red $V_h < 500$ km/s,

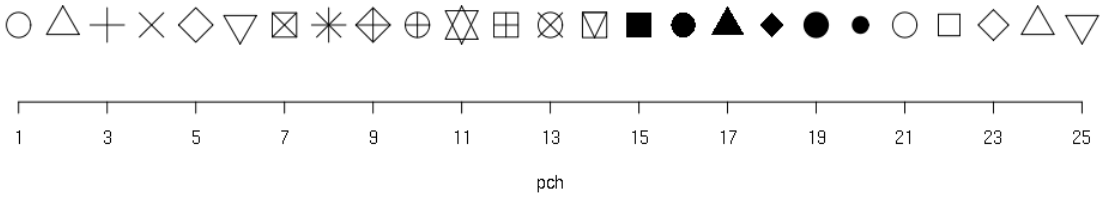
Blue $500 \leq V_h < 1500$ km/s,

Magenta $1500 \leq V_h < 2500$ km/s,

Cyan $2500 \leq V_h < 3500$ km/s,

Green $3500 \leq V_h \leq 4500$ km/s

The easiest way to define the colours is by using a combination of elseif functions. Use pch =19. (If you cannot create the colour distribution using RStudio you can use the last column of the file Local_Supercluster.dat which provides the colours).



After plotting the graph save it as a .png file with transparent background. You will need it later for exercise 2. To do so use the command:

```
png("Local_Supercluster.png", bg = "transparent", width = 1840, height = 1036)
```

The function png() will copy the plot only AFTER the function is applied. So, you need to rerun the command for creating the scatter plot. After creating the plot run the following command.

```
dev.off() # this command is necessary to save the plot in the file
```

Exercise 2

Image Analysis and Processing using RStudio

In this exercise you will need a new package called magick. This package is designed for processing high quality images. It can read and write picture files, convert formats, produce transformations (cut, edit, rotate, filter), combine images, produce animation from image frames, produce intensity histograms. Install the package if you have not done it already and let R know that you will use this package. You can use the command str(magick::magick_config()) to find out which features and formats are supported by the version you are using.

Read the image ESA listed in the folder R_Lab. Give it the name B for background. You can use the command: `B <- image_read("name_of_the_image")`

You can also find more information about the image using the command: *image_info(name_of_the_image)*

After inserting B, insert the png image of the plot you created in exercise 1. Name it F for foreground.

Making Transformations: The best way to find out what is available in terms of transformations is to use help (*?transformations*).

In terms of this exercise, edit the size of the two images to be centred.

```
Result <- B %>% image_composite(F, offset = "-57-43") # places the ellipse
in the centre by changing the offset.
```

If you want to export an image you need to use the *image_write()* command.

```
image_write(name, path = "filename.format", format = "format")
```

Example:

```
> image_write(Result, path = "Supercluster.jpg", format = "jpg")
```

Exercise 3

Open NO3.r script file and follow the instructions provided.

Exercise 4

Open Exercise_4 script in RStudio select it and run it. Try to understand what it is doing.