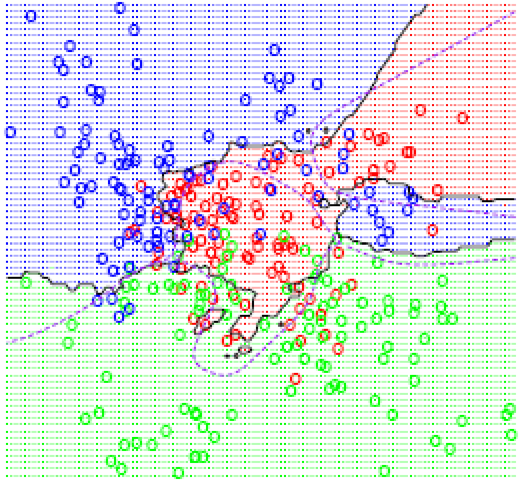
Machine Learning

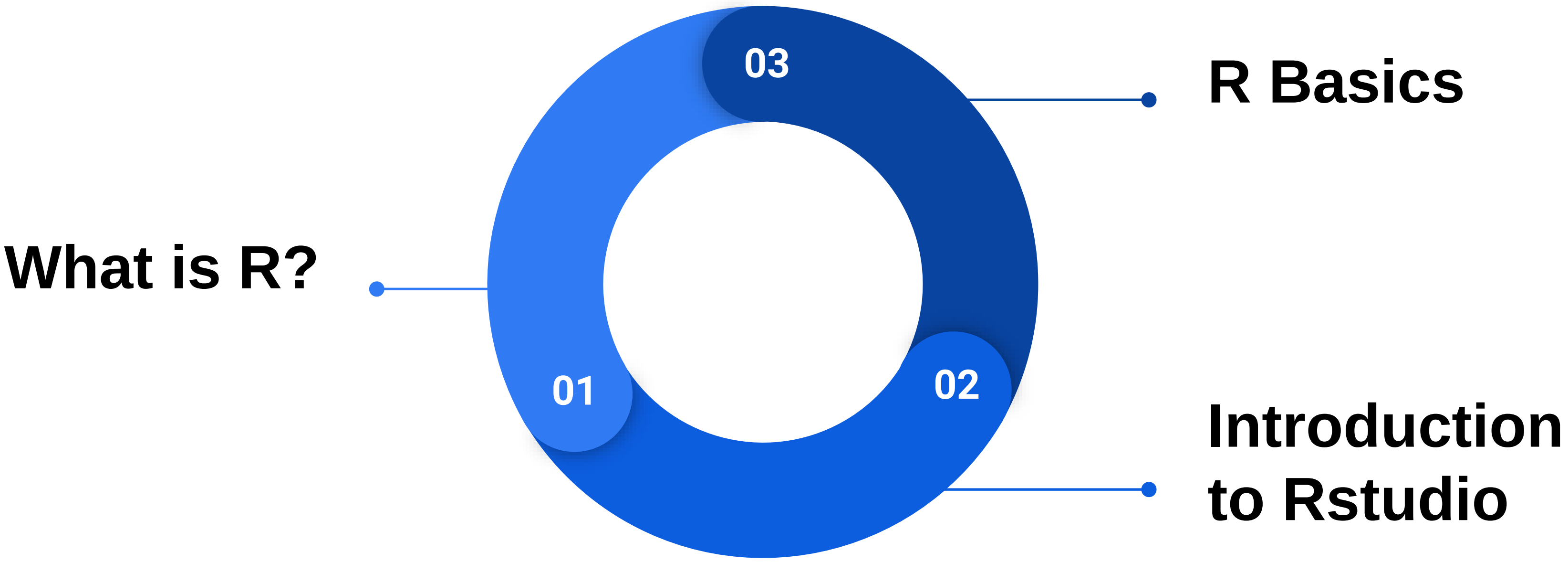
Tutorial 1: An introduction to R



Fahimeh Palizban
Department of Bioinformatics, IBB, University of Tehran

Mehr 2, 1398

Contents



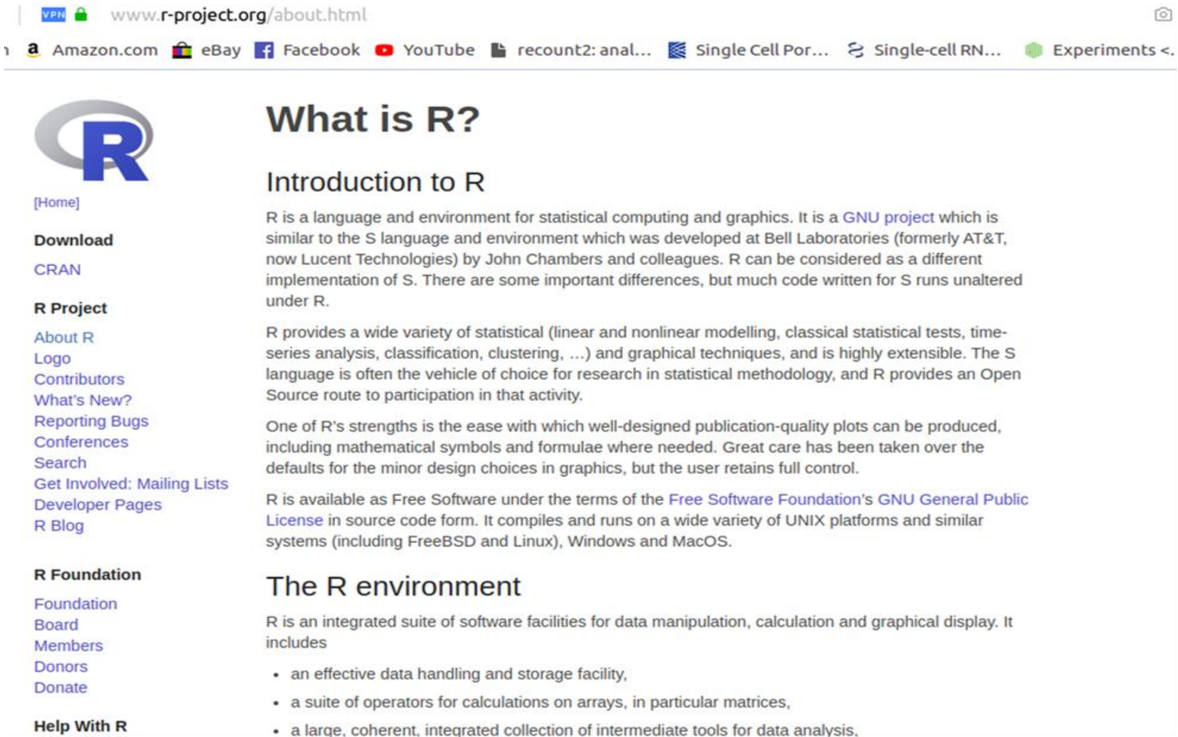
01 What is R?

02 Introduction to Rstudio

03 R Basics

Introducing R

R is a language and environment for statistical computing and graphics.



What is R?

Introduction to R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

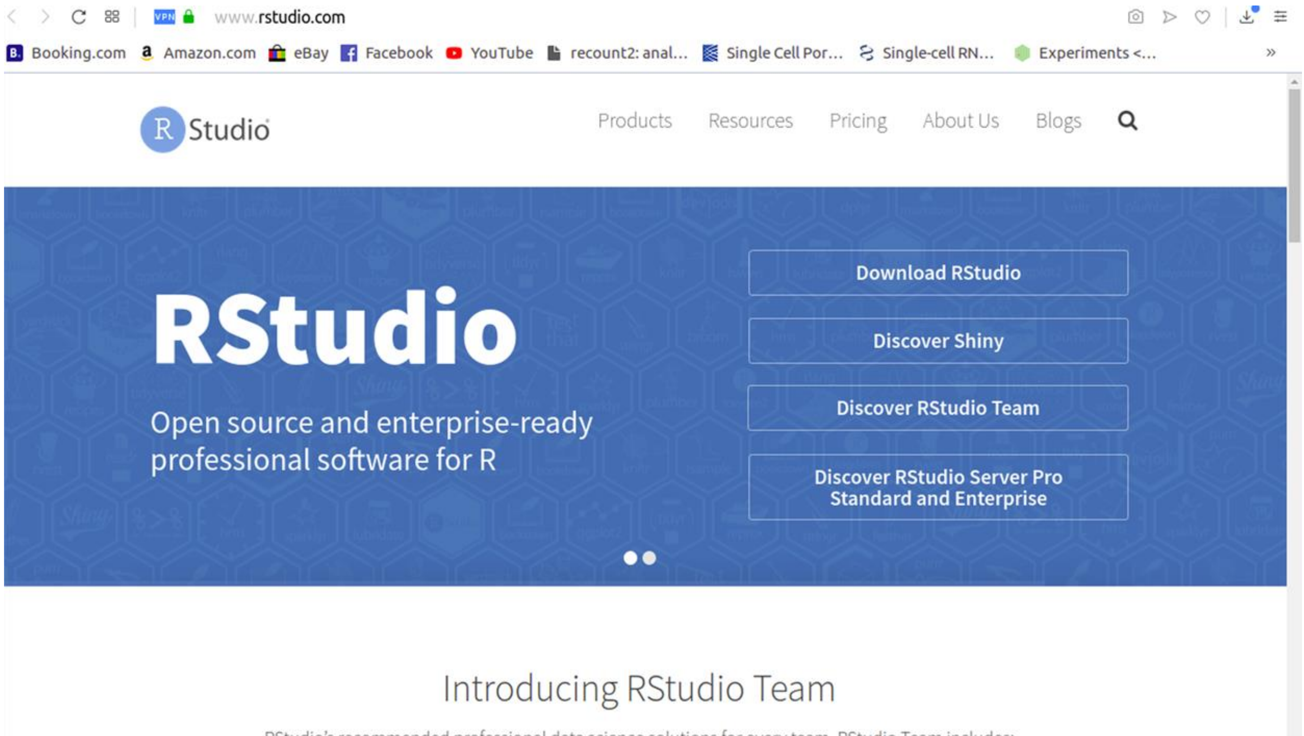
The R environment

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,

Introducing Rstudio

RStudio is an integrated development environment for R



RStudio

Open source and enterprise-ready professional software for R

Introducing RStudio Team

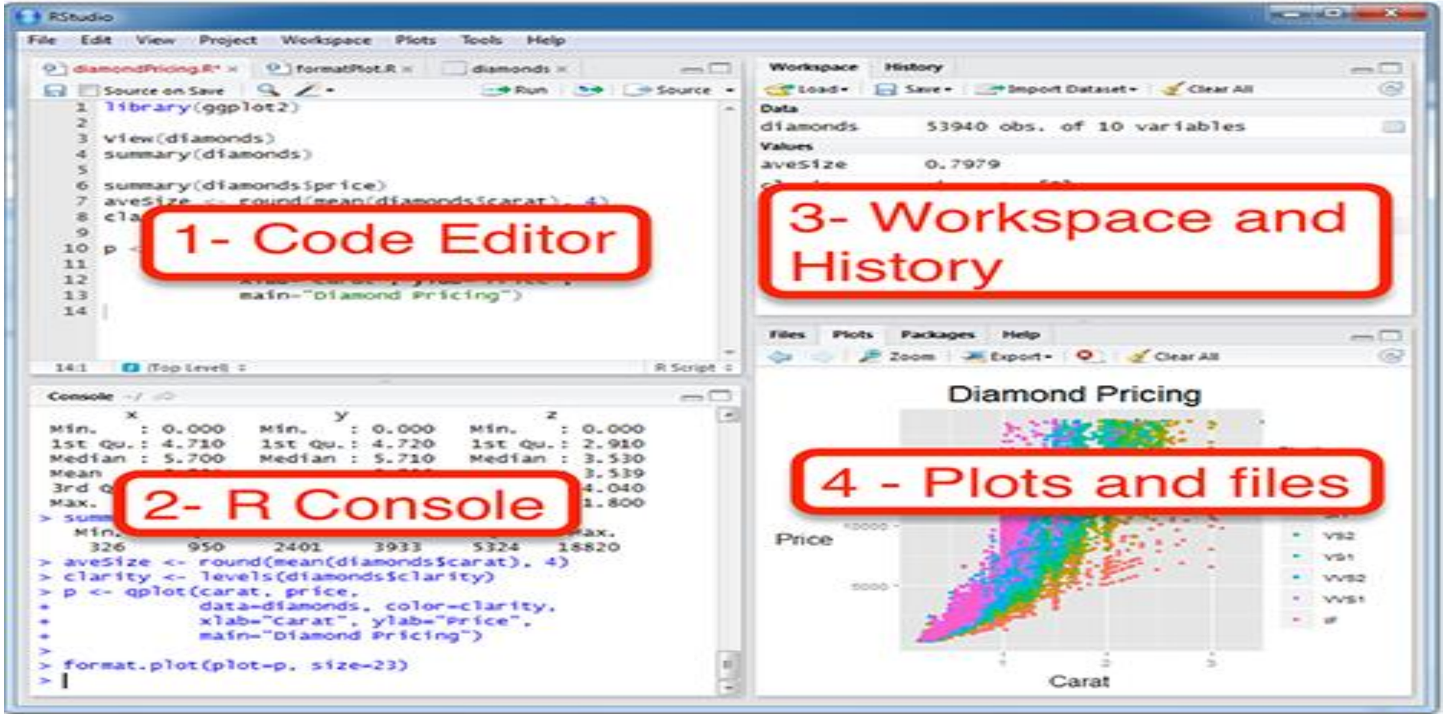
Installing R & Rstudio

Download R from http://cran.us.r-project.org/

Install R. Leave all default settings in the installation options

Download RStudio from http://rstudio.org/download/desktop and install it. Leave all default settings in the installation options

RStudio



1- Code Editor

2- R Console

3- Workspace and History

4 - Plots and files

```
library(ggplot2)

view(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
cla

p <

           main="Diamond Pricing")

```

```
      x                y                z
Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
Median : 5.700   Median : 5.710   Median : 3.530
Mean                                        3.539
3rd Q                                       4.040
Max.                                        1.800
> summ
  Min.                                      Max.
   326      950     2401     3933     5324  18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(plot=p, size=23)
> 
```

```
Data
diamonds        53940 obs. of 10 variables
Values
aveSize         0.7979
```

Using Projects

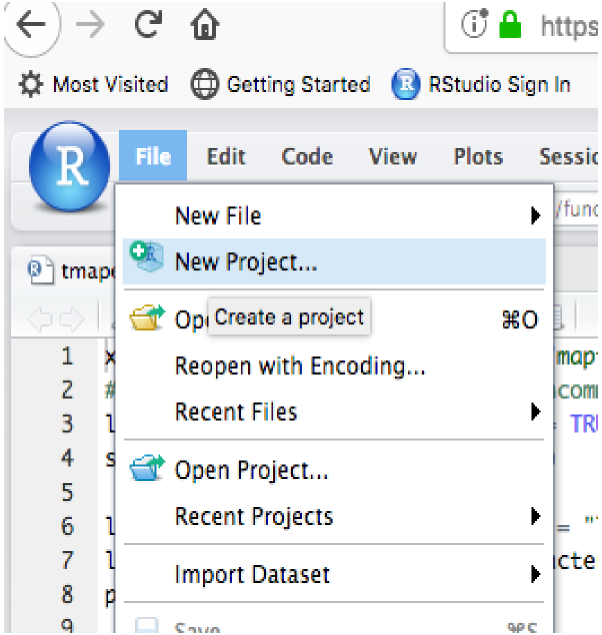
RStudio projects make it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents.

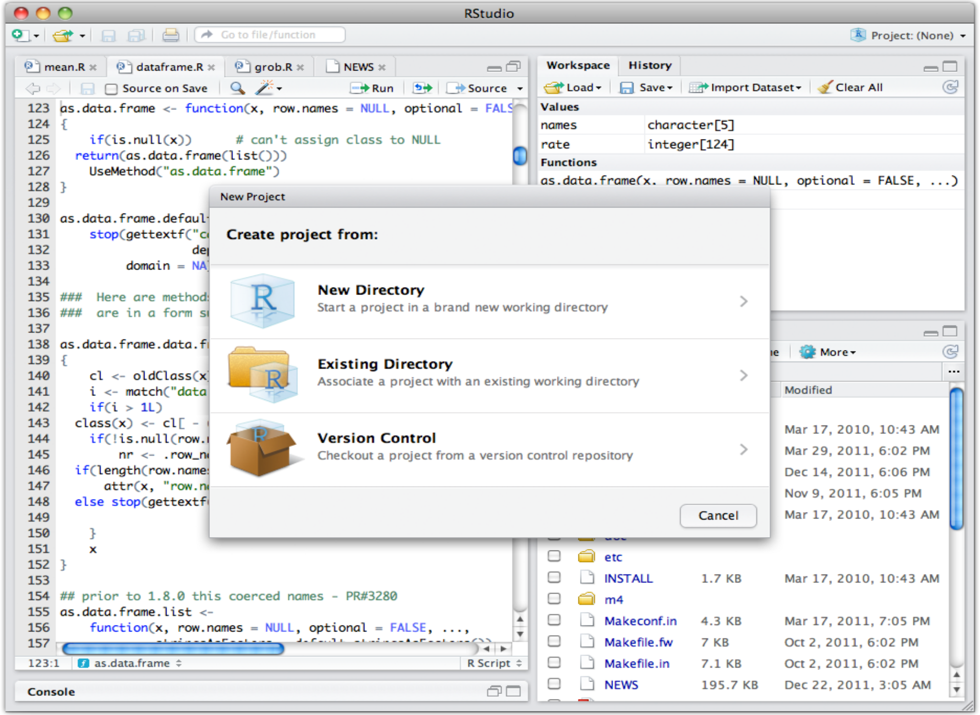
Creating Projects

RStudio projects are associated with R working directories. You can create an RStudio project:

- In a brand new directory
- In an existing directory where you already have R code and data
- By cloning a version control (Git or Subversion) repository

To create a new project use the Create Project command (available on the Projects menu and on the global toolbar)





New Project

Create project from:

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

Cancel

Packages

In R, the fundamental unit of shareable code is the package. A package bundles together code, data, documentation, and tests, and is easy to share with others

What Are Repositories?

A repository is a place where packages are located so you can install them from it.

Three of the most popular repositories for R packages are:

- CRAN
- Bioconductor
- Github

University of TEHRAN

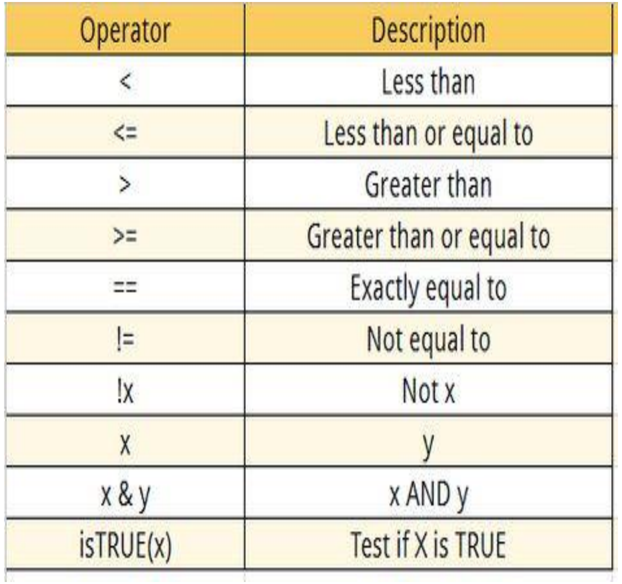
Installing Packages

```
install.packages("package")    #Installing Packages From CRAN

if (!requireNamespace("BiocManager", quietly = TRUE))

    install.packages("BiocManager")

BiocManager::install()                                        #Installing Bioconductor Packages
```

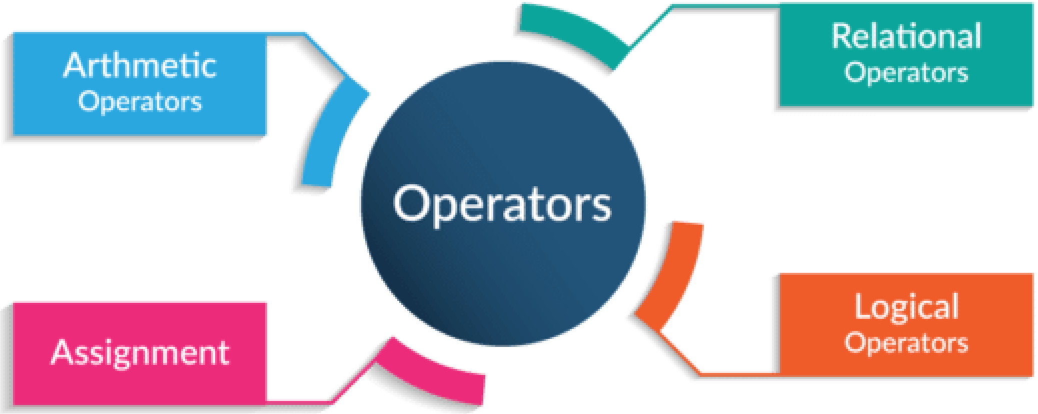
How To Update, Remove , load And Check Installed Packages

```
installed.packages()

remove.packages()

old.packages()

update.packages()

library() != detach()
```

Data Operators in R

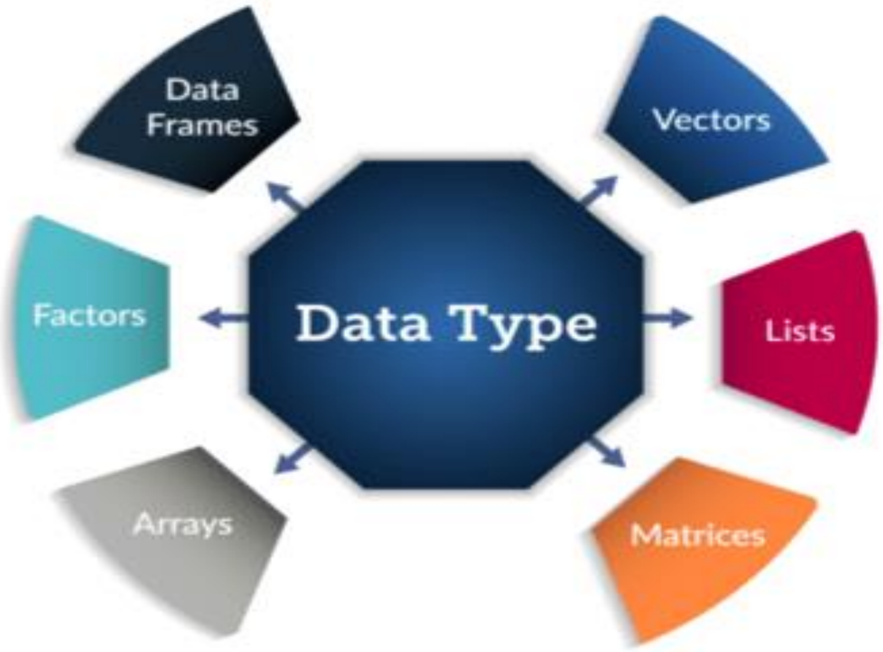


Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Exactly equal to
!=	Not equal to
!x	Not x
x	y
x & y	x AND y
isTRUE(x)	Test if X is TRUE



https://www.edureka.co/blog/r-tutorial/

Data Types



https://www.edureka.co/blog/r-tutorial/

University of TEHRAN

Data Types

Vectors

```
a <- c(1,2,5.3,6,-2,4)
```

Matrices

```
y<-matrix(1:20, nrow=5,ncol=4)
```

Data Frames

```
mydata <- data.frame(d,e,f)
```

Lists

```
w <- list(name="Fred", mynumbers=a, mymatrix=y, age=5.3)
```

Factors

```
gender <- c(rep("male",20), rep("female", 30))

gender <- factor(gender)
```

Objects Attributes

Objects can have **attributes**. Attributes are part of the object. These include:

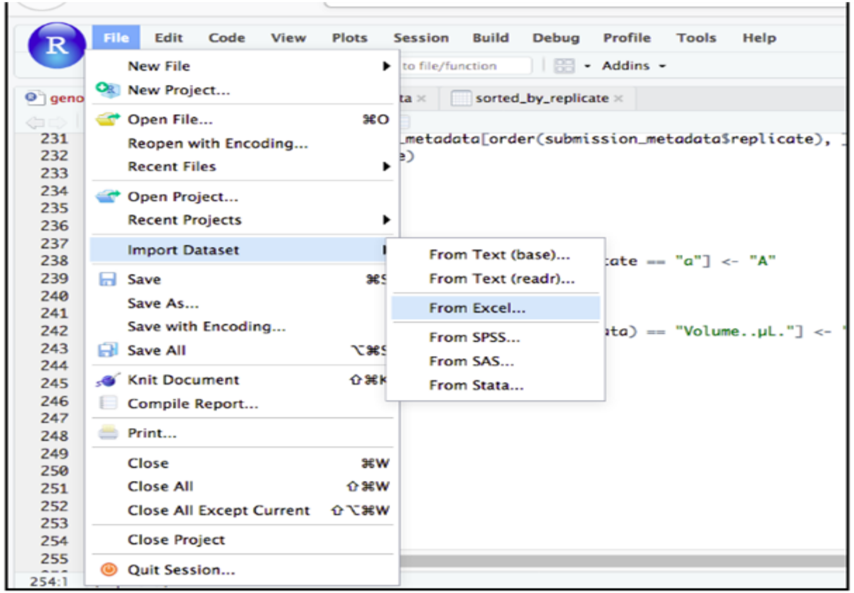
- names
- dimnames
- dim
- class
- attributes (contain metadata)
- Length
- Number of characters

Importing Data in R Studio

read.table()

read.csv()

Importing data from Excel



```
_metadata[order(submission_metadata$replicate), ]
e)
:ate == "a"] <- "A"
ata) == "Volume..µL."] <- '
```

save the data frame

```
save(df, file = "df.RData")

write.csv(df, file = "df.csv")

write.table()

fwrite()
```

Save into Rdata

```
saveRDS(object = final_model, file = "final_model.rds")

save()
```

Exporting to an Excel file

```
WriteXLS()
```

Defining a Function

Functions are defined by code with a specific format:

```
myfunction <- function(arg1, arg2, ... ){
          Statements
          return(object)
          }
```

Example of a Function:

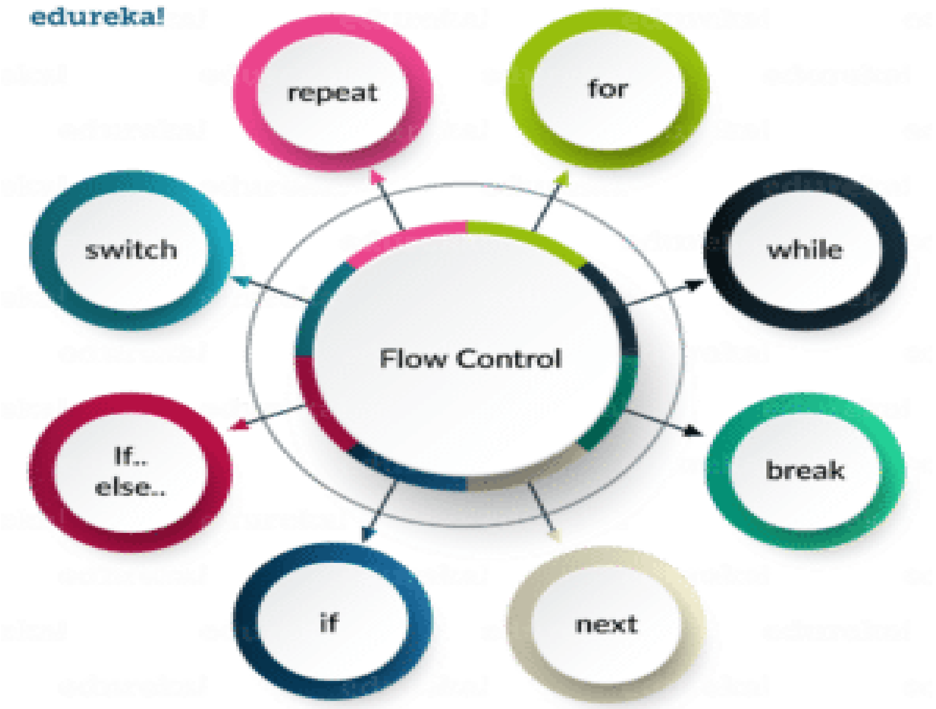
```
pow <- function(x, y) {
# function to print x raised to the power y
result <- x^y
print(paste(x,"raised to the power", y, "is", result))
}
pow(2,3)
```

Control Structures in R Programming

In order to control the execution of the expressions flow in R, we make use of the control structures. These control structures are also called as loops in R. There are eight types of control structures in R:

- If
- If-else
- for
- nested loops
- while
- repeat and break
- next
- return

Control Structures in R Programming



https://www.edureka.co/blog/r-tutorial/

Flow Control

If:

```
        values <- 1:10

    if (sample(values,1) <= 10)

    print(paste(values, "is less than or equal to 10"))
```

For:

```
        values <- c(1,2,3,4,5)

    for(id in 1:5){

                print(values[id])

                }
```

While:

```
        x=2

    while(x<1000)
    {
    x=x^2
    print(x)
    }
```

References

https://www.edureka.co/blog/r-tutorial/

https://datacarpentry.org/genomics-r-intro/

http://www.sr.bham.ac.uk/~ajrs/R/r-function_list.html

Thanks!

Useful Functions

```
paste(x)     # Concatenate vectors after converting to character

range(x)     # Returns the minimum and maximum of x

rep(1,5)     # Repeat the number 1 five times

rev(x)       # List the elements of "x" in reverse order

seq(1,10,0.4)  # Generate a sequence (1 -> 10, spaced by 0.4)

sequence()     # Create a vector of sequences

sign(x)        # Returns the signs of the elements of x

sort(x)        # Sort the vector x

order(x)       # list sorted element numbers of x

tolower(),toupper()  # Convert string to lower/upper case letters
```

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

object     # prints the object

ls()       # list current objects

rm(object) # delete an object
```

Useful Functions

```
newobject <- edit(object) # edit copy and save as newobject

fix(object)          # edit in place

head()          # shows first 6 rows

tail()          # shows last 6 rows

dim()           # returns the dimensions of data frame (i.e. number of rows and number of columns)

nrow()          # number of rows

ncol()          # number of columns

str()           # structure of data frame - name, type and preview of data in each column

names() or colnames()     # both show the names attribute for a data frame

sapply(dataframe, class)    # shows the class of each column in the data frame
```

Useful Functions

```
unique(x)      # Remove duplicate entries from vector

system("cmd")  # Execute "cmd" in operating system (outside of R)

vector()       # Produces a vector of given length and mode

log(x),logb(),log10(),log2(),exp(),expm1(),log1p(),sqrt()   # Fairly obvious

cos(),sin(),tan(),acos(),asin(),atan(),atan2()       # Usual stuff

cosh(),sinh(),tanh(),acosh(),asinh(),atanh()         # Hyperbolic functions

union(),intersect(),setdiff(),setequal()             # Set operations

eigen()      # Computes eigenvalues and eigenvectors

deriv()      # Symbolic and algorithmic derivatives of simple expressions

integrate()  # Adaptive quadrature over a finite or infinite interval.
```

Useful Functions

```
getwd()        # Return working directory

setwd()        # Set working directory

help(package=graphics)     # List all graphics functions

plot()                  # Generic function for plotting of R objects

par()                    # Set or query graphical parameters

arrows()                  # Draw arrows [see errorbar script]

hist(x)                  # Plot a histogram of x

pairs()                  # Plot matrix of scatter plots

matplot()                 # Plot columns of matrices
```

Useful Functions

```
lm                                    # Fit linear model

glm                                   # Fit generalised linear model

nls                                   # non-linear (weighted) least-squares fitting

lqs                                   # "library(MASS)" resistant regression

density(x)             # Compute kernel density estimates

mean(x), weighted.mean(x), median(x), min(x), max(x), quantile(x)

rnorm(), runif()  # Generate random data with Gaussian/uniform distribution

sd()                   # Calculate standard deviation

summary(x)             # Returns a summary of x: mean, min, max etc.

t.test()               # Student's t-test
```

Useful Functions

```
var()                       # Calculate variance

sample()                    # Random samples & permutations

getwd()                     #shows the current working directory

setwd()                     #sets the working directory

which()                     #return the indices of any item that evaluates as TRUE in our comparison
```