

# Introduction to R - Part 2

## Data Science Skills Day 2022

Anjali Silva, PhD

Summer Undergraduate Data Science Research Program  
University of Toronto  
03 June 2022

 a.silva@utoronto.ca  anjalisilva.github.io  @Silva\_Anjali

# Course Description

- Introduction to R - Part 2  
Data Science Skills Day

- The vast amount of data produced by evolving information technology requires tools and skills. Among the many tools, R is a free, open-source language for data sciences. R is a programming language that can aid in the process of data analysis. This course is a beginner level, introductory course for R for data analysis. We will learn about R, RStudio (the environment used to work in R), including installation, and apply R for beginner-level data modeling and visualization. By the end of the course, you'll have an introduction to the flexibility of R, different functionalities, and understand how to apply it for basic data exploration.
- Friday 10:00 am – 4 pm EST; online - synchronous.

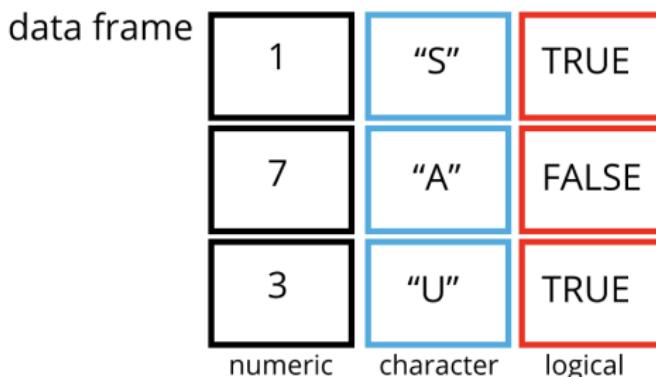
## Material

- Instructor Slides:
    - [https://github.com/anjalisilva/DSI\\_IntroductionToR](https://github.com/anjalisilva/DSI_IntroductionToR)
    - SlideIntroR2022.pdf
    - SlideIntroR2022\_part2.pdf

## Data Frames & Booleans

## Data Frames

- Data frames are used to store tabular data in R.



**Figure:** Data frames can store different classes of objects in each column. Figure from: <https://datacarpentry.org/r-socialsci/02-starting-with-data/index.html>

# Data Frames

- Data frames are used to store tabular data in R.
- Data frames can store different classes of objects in each column.

```
dataFrameExample <- data.frame(  
  numbers = 1:4,  
  sex = c("M", "M", "F", "F"))  
  
dataFrameExample  
class(dataFrameExample) # "data.frame"  
dim(dataFrameExample) # 4 2  
names(dataFrameExample) # "numbers" "sex"
```

## Data Frames

- Data frames can be converted to a matrix using `data.matrix()`.

```
dataMatrix <- data.matrix(dataFrameExample)
class(dataMatrix) # "matrix"
dim(dataMatrix) # 4 2
```

## Question 14:

Generate the following information into a data frame.

	numbers	sex	age	height
1				
2	1	M	30	72
3	2	M	31	70
4	3	F	40	65
5	4	F	35	62.4

## Question 15:

After generating the data frame, you realize the height is recorded in inches but should be changed to centimeters. How would you do this? Note, 1 inch = 2.54 cm.

	numbers	sex	age	height
1				
2	1	M	30	72
3	2	M	31	70
4	3	F	40	65
5	4	F	35	62.4

# Booleans

- TRUE/FALSE or T/F are called “boolean” values.

```
testValue <- FALSE  
typeof(testValue)  
is.logical(testValue)  
  
! testValue
```

- “!” is the NOT operator. It returns the opposite of the argument.

## Boolean Operators

- Operators > or <:

```
a <- c(1:5)
a
a < 2 # TRUE FALSE FALSE FALSE FALSE
```

- Equivalence test:

```
a == 2 # FALSE  TRUE FALSE FALSE FALSE
```

## Boolean Operators

- & symbol is the "AND" operator:

```
a <- 45  
(a > 40) & (a < 50) # TRUE
```

- | symbol is the "OR" operator:

```
b <- 10  
(b < 20) | (b > 5) # TRUE
```

## Any questions?

## Question 16:

Given the below numeric vector 'numericVec', how can a user check if values are greater than 5?

```
numericVec <- c(1.1, 3, 5.3, 2)
```

## Question 17:

Given the below numeric vector 'numericVec', how can a user retrieve the value/s that is/are greater than 5, from the vector?

```
numericVec <- c(1.1, 3, 5.3, 2)
```

# Data Import/Export

# Data Import/Export

- To see the list of pre-loaded data in R:

```
data(package = "datasets")
```

```
AirPassengers # Example dataset  
head(AirPassengers) # see first few entries  
tail(AirPassengers) # see last few entries
```

# Data Slicing

- Let us look at another pre-loaded datasets:

```
women # another dataset (last)
?women
dim(women) # 15  2
class(women) # "data.frame"
head(women) # height and weight information

women$height > 60 # slicing
women[women$height > 60, ] # slicing
```

## Data Reading/Writing

- Files can be written using functions like `write.csv()`, `write.table()`:

```
getwd() # file will be saved here  
write.csv(x = women, file = "women.csv")  
# saving women dataset in current working directory
```

## Data Reading/Writing

- Txt files can be read using `read.table("location of the file")` or `read.csv()`

```
womenNew <- read.csv(file = "women.csv", row.names = 1)  
womenNew
```

```
head(womenNew) # to view first part of object  
tail(womenNew) # to view last part of object  
dim(womenNew) # 15  2  
womenNew[c(1:5), ] # to view first 5 rows  
womenNew[, 1] # to view first column
```

## Arithmetic

# Arithmetic

- “+” is used for addition:

```
x <- 2.5 + 2  
x # 4.5
```

```
y <- 2:15  
sum(y) # 119  
sum(y[1:3]) # 9
```

## Arithmetic

- “-” is used for subtraction:

```
x <- 2.5 - 2
```

x # 0.5

- “/” is used for division:

```
x <- 2 / 2
```

x # 1

## Arithmetic

- “\*” is used for multiplication:

```
x <- 2 * 2  
x # 4
```

- “%\*%” is used for matrix multiplication:

```
a <- matrix(1:6, nrow = 2, ncol = 3) # 2 x 3 matrix  
a  
b <- matrix(7:12, nrow = 3, ncol = 2) # 3 x 2 matrix  
b  
c <- a %*% b  
c # 2 x 2 matrix
```

Any questions?

# Functions

# Writing Functions

- Functions combine a sequence of expressions that are executed to achieve a goal.
- Can be reused without rewriting the sequence of expressions.
- Take input, *function arguments* and generate output, *return value*.
- When writing functions, ask yourself:
  - *What will the user want to modify in this function?*
  - This will help determine *function arguments*.

## Function Interface

- Functions are defined using 'function' assigned to a variable name.

```
firstFunction <- function(argumentOne, argumentTwo) {  
  cat("\n First argument is", argumentOne, "\n")  
  cat("\n Second argument is", argumentTwo, "\n")  
  argumentThree <- argumentOne + argumentTwo  
  cat("\n argumentOne + argumentTwo is",  
      argumentThree, "\n")  
  
  return(argumentThree)  
  # always end with return statement (best practice)  
}
```

# Function Interface

- Function names cannot begin with a number.

- To run the function:

```
firstFunction(argumentOne = 2, argumentTwo = 3)
```

- To “call” or “invoke” the function, type function name:

```
firstFunction
```

# Function Interface

- Default values play a vital role in R functions and influence user's behaviour.
- Default values should be assigned using “`=`”, and not “`←`”.

```
secondFunction <- function(argOne = 1, argTwo = 3) {  
  cat("\n First argument is", argOne, "\n")  
  cat("\n Second argument is", argTwo, "\n")  
  argThree <- argOne + argTwo  
  
  cat("\n argOne + argTwo is", argThree, "\n")  
  
  return(argThree)  
}
```

# Local and Global Variables

- Where variables are defined matters.
- Variables defined within functions are only accessible from within the function.
- Variables declared within a function are called “local”.
- Variables declared outside of functions are called “global”.

## Variables

- If variables are defined outside of functions, globally, they hold that value.

```
argOne <- "Hello"
```

```
anotherFunction <- function() {  
  argOne <- 10  
  return(argOne)  
}
```

anotherFunction()

```
argOne # What would this return?
```

# Variables

- Need to pass data into functions:

```
women
```

```
# Incorrect
dataAnalysis <- function() {
    heightData <- women$height
    return(mean(heightData))
}
dataAnalysis() # will work only for women$height
```

```
# Recommended
dataAnalysis <- function(inputData) {
    meanOfHeight <- mean(inputData)
    return(meanOfHeight)
}
dataAnalysis(inputData = women$height)
# will work for any dataset with height information
```

## Any questions?

# Non-base Functions

- Functions have been written by other authors.
- Non-base functions in R can be utilized by downloading R packages.
- To see all the currently loaded packages:

```
search()
```

## Non-base Functions

- Popular repositories for Packages:
    - The Comprehensive R Archive Network (CRAN)
      - Link: <https://cran.r-project.org/web/packages/>
    - Bioconductor
      - Link: <https://www.bioconductor.org/packages/release/bioc/>
    - GitHub
  - Depending on the source of Package, downloading instructions may differ.

# Example

- User wants to do a cluster analysis of the women built-in dataset.
- Assume the user is interested in performing model-based clustering using Gaussian finite mixture models.
  - Do an R search:  
`??clustering`
  - Google R packages for model-based clustering.
  - Read blogs on clustering.

## Example, continue...

- Get to know about R package 'mclust'.
- Visit <https://cran.r-project.org/web/packages/mclust/index.html>.
- Click on Reference manual to see all the functions within the package.
- Download package to R session:

```
# Install package from CRAN, case matters!
install.packages("mclust")
library("mclust") # to load and attach
```

## Example, continue...

- More details on 'mclust' package (only after downloading):

```
vignette("mclust") # vignette for 'mclust'  
  
?`mclust-package` # get information on package  
  
?mclust # get information on package  
  
ls("package:mclust") # list all functions in package
```

## Example, continue...

- Work with 'mclust' package:

```
# Running mclust
MclustResults <- mclust::Mclust(data = women)
str(MclustResults) # provide the structure; list of 15
names(MclustResults)
MclustResults$G # There are four clusters in the dataset

# Citing the package
citation("mclust")
```

# Bioconductor

- Bioconductor packages are available from [bioconductor.org](http://bioconductor.org).
- URL: <https://www.bioconductor.org/install/>.

```
if (! requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
```

- Bioconductor packages are listed here:  
<https://www.bioconductor.org/packages/release/bioc/>.

Bioconductor



[Home](#) » [Bioconductor 3.11](#) » [3.11 Software Packages](#)

## Bioconductor Software Packages

Bioconductor version: Release (3.11)

Package	Maintainer	Title
a4	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Umbrella Package
a4Base	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Base Package
a4Classif	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Classification Package
a4Core	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Core Package
a4Preproc	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Preprocessing Package
a4Reporting	Tobias Verbeke, Laure Cougnaud	Automated Affymetrix Array Analysis Reporting Package
ABAEnrichment	Steffi Grote	Gene expression enrichment in human brain regions
ABarray	Yongming Andrew Sun	Microarray QA and statistical data analysis for Applied Biosystems Genome Survey Microrarray (AB1700) gene expression data.
abseqR	JiaHong Fong	Reporting and data analysis functionalities for Rep-Seq datasets of antibody libraries

Packages \*

### *Bioconductor's stable, semi-annual releases:*

- Analysis [software](#) packages.
  - [Annotation](#) packages.
  - Illustrative [experiment data](#) packages.
  - [Workflow](#) packages.
  - Latest [release announcement](#).

*Bioconductor* is also available via [Docker](#) and [Amazon Machine Images](#).

Development Version ➤

## Bioconductor packages under development:

- Analysis [software](#) packages.
  - [Annotation](#) packages
  - Illustrative [experiment data](#) packages

## Developer Resources:

- GIT Log

# Example

- User is interested in Bioconductor package GenomicFeatures.
- To load package:

```
BiocManager::install("GenomicFeatures")
library("GenomicFeatures")

# list all functions in the package
ls("package:GenomicFeatures")
```

- More about installation:

<https://www.bioconductor.org/install/>.

# Bioconductor



Home

# Explore

Notifications

Messages

Bookmarks

Lists

Profile

[Bioconductor](#)  
2,020 Tweets



[Bioconductor](#)  
@Bioconductor

Roswell Park Cancer Institute bioconductor.org  
Joined November 2011

5 Following 7,028 Followers

GitHub

- To download from GitHub, need to use 'devtools' package.

```
library("devtools")
devtools::install_github("<GitHubUsername>/<PackageName>",
  build_vignettes = TRUE)
```

- Example: <https://github.com/anjalisilva/mixGaussian>

- To download this GitHub R package:

```
library("devtools")
devtools::install_github("anjalisilva/mixGaussian",
build_vignettes = TRUE)
library("mixGaussian")
ls("package:mixGaussian") # to see all functions
```

## Any questions?

## Question 18:

You are interested in R packages for analyzing Covid-19 data for a project you are doing. So you search Google for such R packages.

You come across a package called 'covid19.analytics' which has a GitHub R package

(<https://github.com/mponce0/covid19.analytics>) that was later developed into a CRAN package (<https://cran.r-project.org/web/packages/covid19.analytics/index.html>). How would you download both the GitHub and CRAN package?

# R: Graphics

# Applying Base Functions

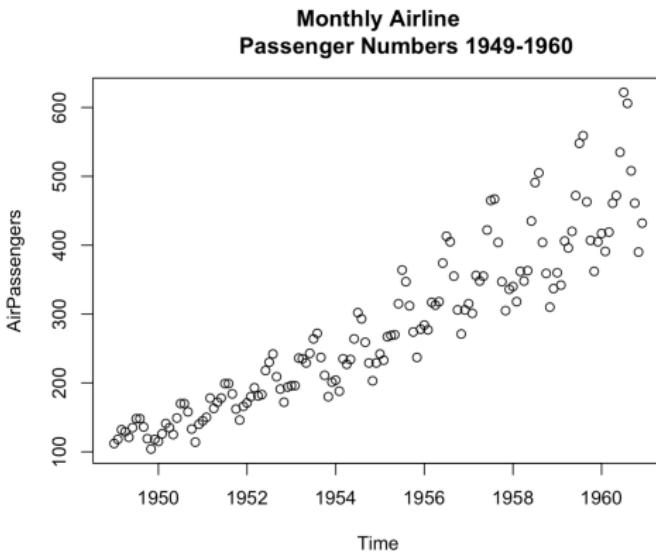
- Package 'graphics' is a standard or base package.

```
library(graphics) # to load and attach

?AirPassengers
# Monthly Airline Passenger Numbers 1949-1960

# plot
plot(AirPassengers)
plot(AirPassengers, type = "p")
plot(AirPassengers, type = "p", main = "Monthly Airline
    Passenger Numbers 1949-1960") # zoom
```

# Applying Base Functions



**Figure:** Plotting AirPassengers dataset as a scatter plot.

# Applying Base Functions

- Package 'graphics' is a base package.

```
library(graphics) # to load and attach  
  
# plot  
plot(AirPassengers)  
plot(AirPassengers, type = "l", main = "Monthly Airline  
Passenger Numbers 1949-1960") # zoom
```

# Applying Base Functions

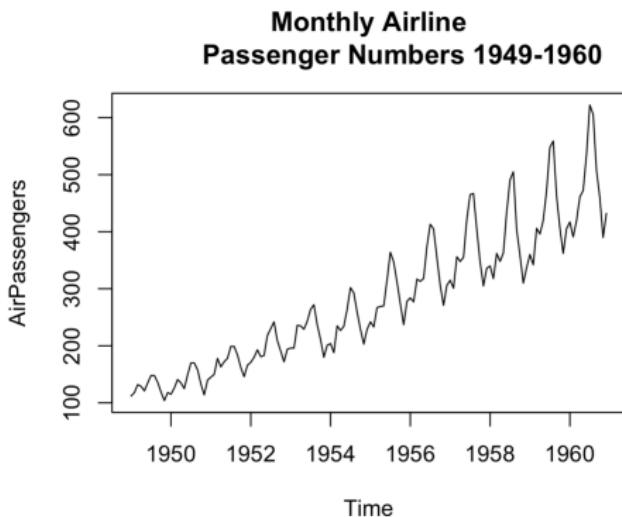


Figure: Plotting AirPassengers dataset as a line plot.

# Applying Base Functions

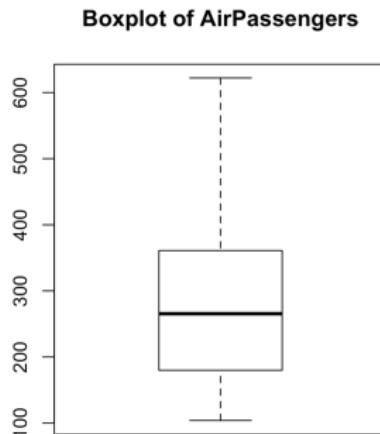
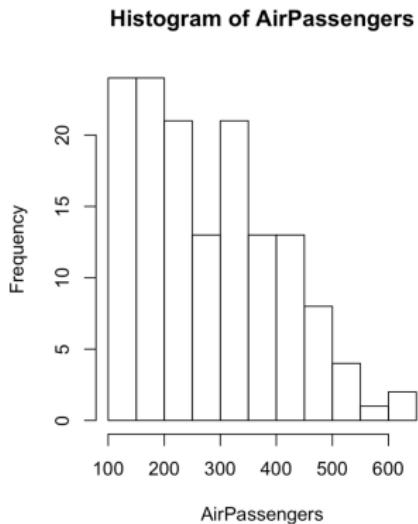
- Package 'graphics' is a base package.

```
library(graphics) # to load and attach

?AirPassengers
# Monthly Airline Passenger Numbers 1949-1960

# To have multiple plots in one overall plot using
par(mfrow = c(1, 2))
hist(AirPassengers) # histogram
boxplot(AirPassengers) # boxplot
```

# Applying Base Functions



**Figure:** Output from plotting AirPassengers dataset.

# Applying Base Functions

- Package 'graphics' is a base package.

```
library(graphics) # to load and attach

?iris
?pairs

pairs(iris[, c(1:4)], col = iris$Species,
      main = "Scatter plot of iris dataset")
```

# Applying Base Functions

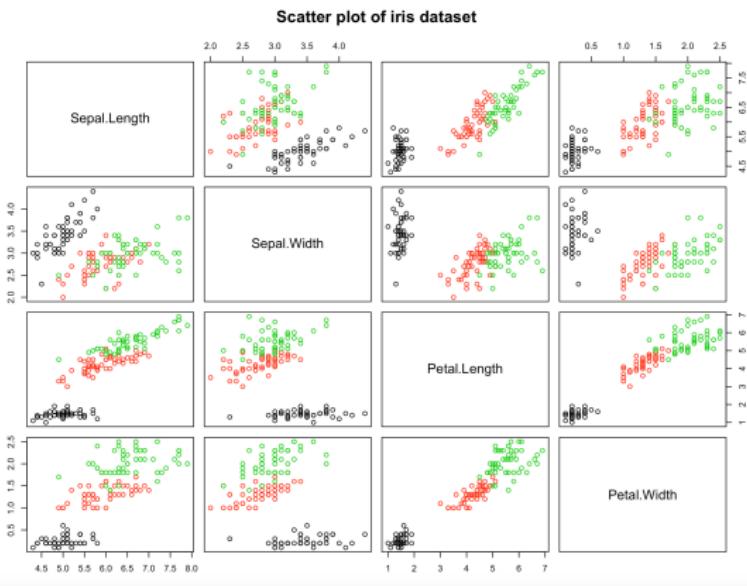


Figure: Output from plotting iris dataset as a pairs plot.

# Applying Base Functions

- Package 'graphics' is a base package.

```
library(graphics) # to load and attach

barplot(as.matrix(iris[, c(1:4)]),
        legend.text = TRUE,
        main = "Bar plot of iris dataset")

# Calculate column sums of iris dataset
colSums(as.matrix(iris[, c(1:4)]))

# Sepal.Length  Sepal.Width Petal.Length  Petal.Width
# 876.5         458.6       563.7       179.9
```

# Applying Base Functions

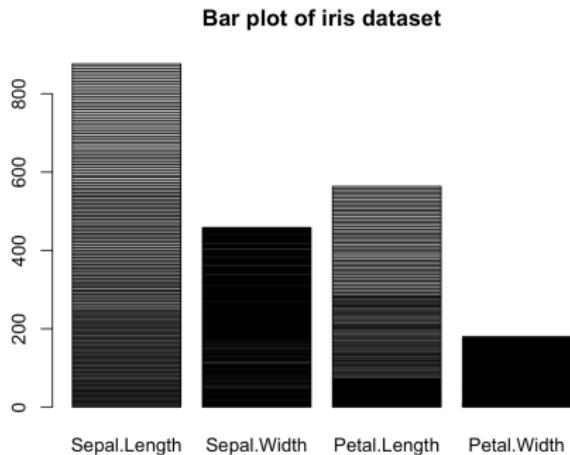


Figure: Output from plotting iris dataset as a bar plot.

# R Packages for Plotting

- More resources:
  - R Graphics Cookbook: <https://r-graphics.org/>.
  - sthda:  
<http://www.sthda.com/english/wiki/ggplot2-essentials>.

## Any questions?

## Question 19:

Anscombes quartet is a set of 4 x,y data sets that were published by Francis Anscombe in a 1973 paper Graphs in statistical analysis. This is available as a standard or base dataset in R.

- Obtain the dataset. What is the size of the dataset? Check the column names of the dataset. Summarise the data by calculating the mean, variance, for each column and the correlation between each pair, eg.,  $(x_1, y_1)$ ,  $(x_2, y_2)$ , etc. Then create a scatter plot for each x,y pair of data.

## Question 19: continue...

- Anscombe's quartet has four data sets that have nearly identical descriptive statistics, but very different distributions.

<u>Property</u>	<u>Value</u>
Mean of X (average)	9 in all 4 XY plots
Sample variance of X	11 in all four XY plots
Mean of Y	7.50 in all 4 XY plots
Sample variance of Y	4.122 or 4.127 in all 4 XY plots
Correlation (r)	0.816 in all 4 XY plots

- Original paper:

<https://www.jstor.org/stable/2682899?seq=1>.

More Problems (if time permits)...

## Question 20:

Write a function that take user name and print the following to the user: user name, current R version and list of objects currently in memory.

## Question 21:

Make a matrix as follows. Replace all numbers greater than 50 with the value of 0. How can you double check your answer? Note: this will be done using random number generator, hence the solution may differ from that of instructor.

```
randMatrix <- matrix(sample(1:100, 100, replace = TRUE),  
                      nrow = 10,  
                      ncol = 10)
```

## Question 22:

Plot the ‘ToothGrowth’ dataset available in R as a scatter plot. Show dose vs tooth length. Color by supplemental type. Label the axis with correct unit (if applicable) and put a title.

- ToothGrowth

## Question 23:

Dataset ‘trees’ is available on R. Use this dataset and calculate summary statistics for diameter, height and volume for Black Cherry trees. What maybe the best method of plotting all 3 variables (diameter, height and volume) using methods learned today, so the user is able to identify potential trends?

- trees

## Summary

## Summary

Base R Cheat Sheet		Vectors	Programming
Getting Help		Creating Vectors	While Loop
Accessing the help files		c(2, 4, 6) 2:6 seq(2, 3, by=0.5) rep(1:2, times=3) rep(1:2, each=3)	Join elements into a vector An integer sequence A complex sequence Repeat a vector Repeat elements of a vector
Tweak		sort(x) table(x)	for (variable in sequence){ Do something }  for (i in 1:10{ j <- i + 10 print(j) })
Get help of a particular function. help('search'='weighted mean')		rev(x) unique(x)	Example Example
Search the help files for a word or phrase. help(package = 'dplyr')		See counts of values.	while (condition){ Do something }  while (i < 5){ print(i) i <- i + 1 }
Find help for a package.		More about an object	Example
Using Packages		Vector Functions	Functions
install.packages('dplyr')		sort(x) table(x)	function_name <- function(var){ Do something }  square <- function(x){ squared <- x*x return(squared) }
Download and install a package from CRAN.		rev(x) unique(x)	Example
library(dplyr)		See unique values.	if (condition){ Do something } else { Do something different }
Load the package into the session, making all its functions available to use.		By Position	Example
dplyr::select		x[4]	if (i > 3){ print('Yes') } else { print('No') }
Use a particular function from a package.		x[-4]	Reading and Writing Data
data(iris)		x[2:4]	Also see the <code>readr</code> package.
Load a built-in dataset into the environment.		x[-(2:4)]	Input
		x[c(3, 5)]	Output
		Elements two to four.	Description
		All elements except two to four.	df <- read.table('file.txt')
		Elements one and five.	write.table(df, 'file.txt')
		By Value	Read and write a delimited text file
		x == 10	df <- read.csv('file.csv')
		Elements which are equal to 10.	write.csv(df, 'file.csv')
		x < 0	Read and write a comma separated value file. This is a special case of <code>read.table</code> / <code>write.table</code> .
		All elements less than zero.	load('file.RData')
		x %in% c(1, 2, 5)	save(df, file = 'file.RData')
		Elements in the set 1, 2, 5.	Read and write an R data file, a file type specific for R.
Working Directory		Named Vectors	Conditions
getwd()		Named Vectors	a == b a != b a >= b a <= b a > b a < b a == b a != b a >= b a <= b a > b a < b is.na(a) is.null(a) is missing
Find the current working directory (where inputs and outputs are sent).		x['apple']	Element with name 'apple'
setwd('C://file/path')			
Change the current working directory.			
Use projects in RStudio to set the working directory to the folder you are working in.			

RStudio® is a trademark of RStudio, Inc. • [CC BY](#) Mhairi McNeill • mhairimcnell@gmail.com

Learn more at [web page](#) or [vignettes](#) • package version • Updated: 3/15

Figure: Figure from <https://www.rstudio.com/resources/cheatsheets/>.

## Summary

### Types

Converting between common data types in R can always go from a higher value in the table to a lower value.

<code>as.logical</code>	<code>TRUE, FALSE, TRUE</code>	Boolean values (TRUE or FALSE).
<code>as.numeric</code>	<code>1, 0, 1</code>	Integers or floating point numbers.
<code>as.character</code>	<code>'1', 'B', '1'</code>	Character strings. Generally preferred to factors.
<code>as.factor</code>	<code>'1', 'B', '1'</code> <code>levels(c('1', 'B'))</code>	Character strings with preset levels. Preferred for some statistical models.

### Matrices

`m <- matrix(x, nrow = 3, ncol = 3)`  
Create a matrix from x.

<code>m[2, ]</code>	— Select a row
<code>m[, 1]</code>	— Select a column
<code>m[2, 3]</code>	— Select an element

### Strings

`paste(x, y, sep = '')`  
Join multiple vectors together.

`paste(x, collapse = ',')`  
Join elements of a vector together.

`grep(pattern, x)`  
Find regular expression matches in x.

`gsub(pattern, replace, x)`  
Replace matches in x with a string.

`toupper(x)`  
Convert to uppercase.

`tolower(x)`  
Convert to lowercase.

`nchar(x)`  
Number of characters in a string.

### Maths Functions

<code>log(x)</code>	Natural log.	<code>sum(x)</code>	Sum.
<code>exp(x)</code>	Exponential.	<code>mean(x)</code>	Mean.
<code>max(x)</code>	Largest element.	<code>median(x)</code>	Median.
<code>min(x)</code>	Smallest element.	<code>quantile(x)</code>	Percentage quantiles.
<code>round(x, n)</code>	Round to n decimal places.	<code>rank(x)</code>	Rank of elements.
<code>signif(x, n)</code>	Round to n significant figures.	<code>var(x)</code>	The variance.
<code>cor(x, y)</code>	Correlation.	<code>sd(x)</code>	The standard deviation.

### Lists

`l <- list(x = 115, y = c('a', 'b'))`  
A list is a collection of elements which can be of different types.

<code>l[[1]]</code>	<code>l[1]</code>	<code>l\$x</code>	<code>l['y']</code>
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

### Factors

`factor(x)`  
Turn a vector into a factor. Can set the levels of the factor and the order.

`cut(x, breaks = 4)`  
Cut a numeric vector into a factor by 'cutting' into four sections.

### Statistics

<code>lm(y ~ x, data=df)</code>	Linear model.
<code>glm(y ~ x, data=df)</code>	Generalised linear model.
<code>summary</code>	Get more detailed information out a model.
<code>pairwise.t.test</code>	Perform a test for paired data.
<code>adv</code>	Analysis of variance.

### Data Frames

`df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))`  
A special case of a list where all elements are the same length.

x	y
1	a
2	b
3	c

`df$cx`      `df[[2]]`

`df$y`      `df[2]`

`df`

**List subsetting**

`df[1]`

`df[1:2]`

`df[1:3]`

`df[-1]`

`df[-1:3]`

`df[-1:-3]`

`df[-1:2]`

`df[-1:2]`

**Matrix subsetting**

`df[, 1:2]`

`df[, -1]`

`df[1, ]`

`df[-1, ]`

`df[1:2, ]`

`df[1:2, -1]`

`df[1, 1:2]`

`df[1, -1]`

`df[1:2, 1:2]`

`df[1:2, -1]`

`df[1, 1:2]`

`df[1, -1]`

**Understanding a data frame:**

`View(df)`

`head(df)`

`tail(df)`

`nrow(df)`

`ncol(df)`

`dim(df)`

`cbind` — Bind columns.

`rbind` — Bind rows.

`Random Variables`

`Normal`

`Poisson`

`Binomial`

`Uniform`

`density`

`dpois`

`dbinom`

`runif`

`pnorm`

`ppois`

`pbinom`

`qunif`

`dnorm`

`dpois`

`dbinom`

`unif`

`pnorm`

`ppois`

`qbinom`

`qunif`

**Distributions**

`plot(x)`

`plot(x, y)`

`hist(x)`

`plot(x, y, xlab = 'X', ylab = 'Y')`

`ggplot2`

`See the lubridate package`

### Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

### The Environment

<code>ls()</code>	List all variables in the environment.
<code>rm(x)</code>	Remove x from the environment.
<code>rm(list = ls())</code>	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

### Plotting

`plot(x)`  
Values of x in order.

`plot(x, y)`  
Values of x and y.

`hist(x)`  
Histogram of x.

### Dates

`See the lubridate package`

Figure: Figure from <https://www.rstudio.com/resources/cheatsheets/>.

## Next Steps

# Resources

- The R Journal: <https://journal.r-project.org/>
- R for Data Science textbook (free): <https://r4ds.had.co.nz/>.
- RStudio Education Series (free):  
<https://education.rstudio.com/learn/>
- Bioconductor Workshops:  
<https://www.bioconductor.org/help/events/>
- Coursera offers courses via both paid + free options
  - <https://www.coursera.org/learn/r-programming>

# Resources



## R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

**Figure:** Tidyverse R Packages designed for data science. Figure from <https://www.tidyverse.org/>.

# Conferences

## useR! — International R User Conference



useR! 2022 will be a hybrid conference from June 20 to June 23, with opportunities to participate as an online attendee or to attend in person at Vanderbilt University Medical Center, Nashville, TN, USA. For the latest updates, follow '[@\\_useRconf](#)' on Twitter, or '[user-conf](#)' on LinkedIn.

# Further Readings

- Hackenberger, B. K. (2020) R software: unfriendly but probably the best. *Croatian Medical Journal* 61. URL:  
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7063554/>.
- Gentleman, R. C., Carey, V. J., Bates, D. M. et al. (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol* 5(R80). URL:  
<https://doi.org/10.1186/gb-2004-5-10-r80>.
- Huber W., Carey V. J., Gentleman R. et al. (2015) Orchestrating high-throughput genomic analysis with Bioconductor. *Nat Methods*. 12(2). URL:  
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4509590/>.

# Conferences

 Bioconductor  
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

HOME NEWS REGISTRATION SPONSORS SCHEDULE ABOUT ▾

## BIOCONDUCTOR CONFERENCE, JULY 27-29, 2022

WHERE SOFTWARE AND BIOLOGY CONNECT



• • •

Registration for the BioC2022 conference is now open!

BioC2022 is organized as a hybrid event. This includes in-person gathering in Seattle (Seattle Children's Hospital Cure Building, 1920 Terry Ave, Seattle, WA 98101) and online participation.

Seattle's [conference hotels, restaurants, activities, transportation](#).

# Data Revolution



Subscribe



Regulating the internet giants

The world's most valuable  
resource is no longer oil, but  
data



Figure: *Economist*, 2017.

## Feedback

- Take the time to provide feedback (anonymous):  
<https://forms.office.com/r/Dne92Br8US>
  - Thank you.