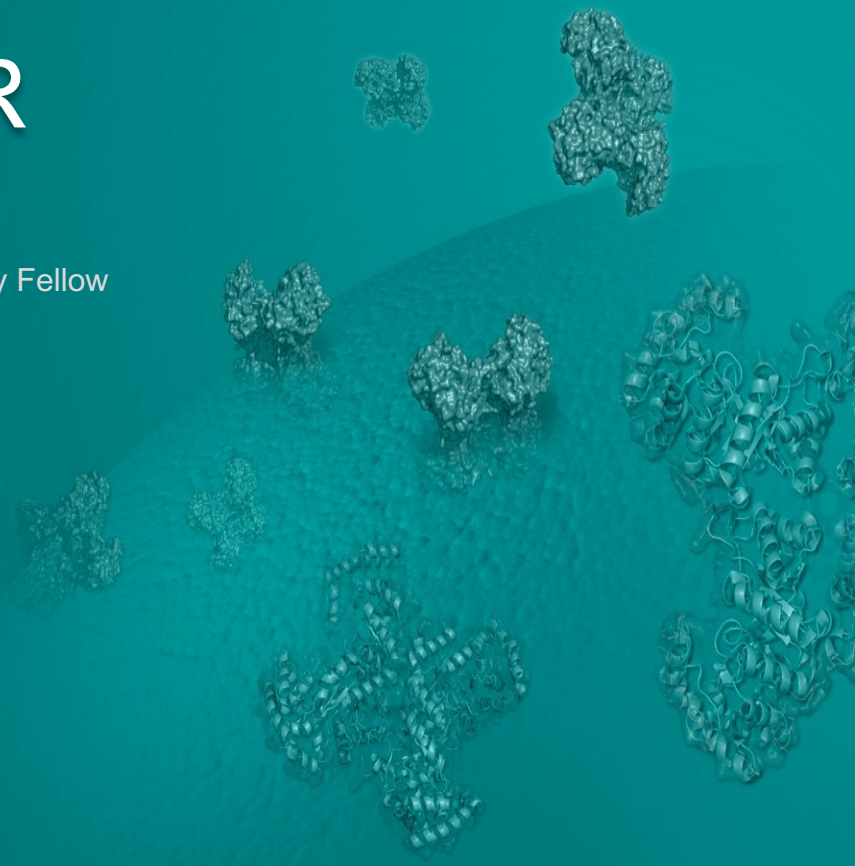
An introduction to R

Jiawei Wang, PhD
Marie Curie Postdoctoral Fellow & EMBO Non-Stipendiary Fellow
Finn & Marioni Groups
EMBL-EBI
jwang@ebi.ac.uk
2024/07/01



EMBL-EBI

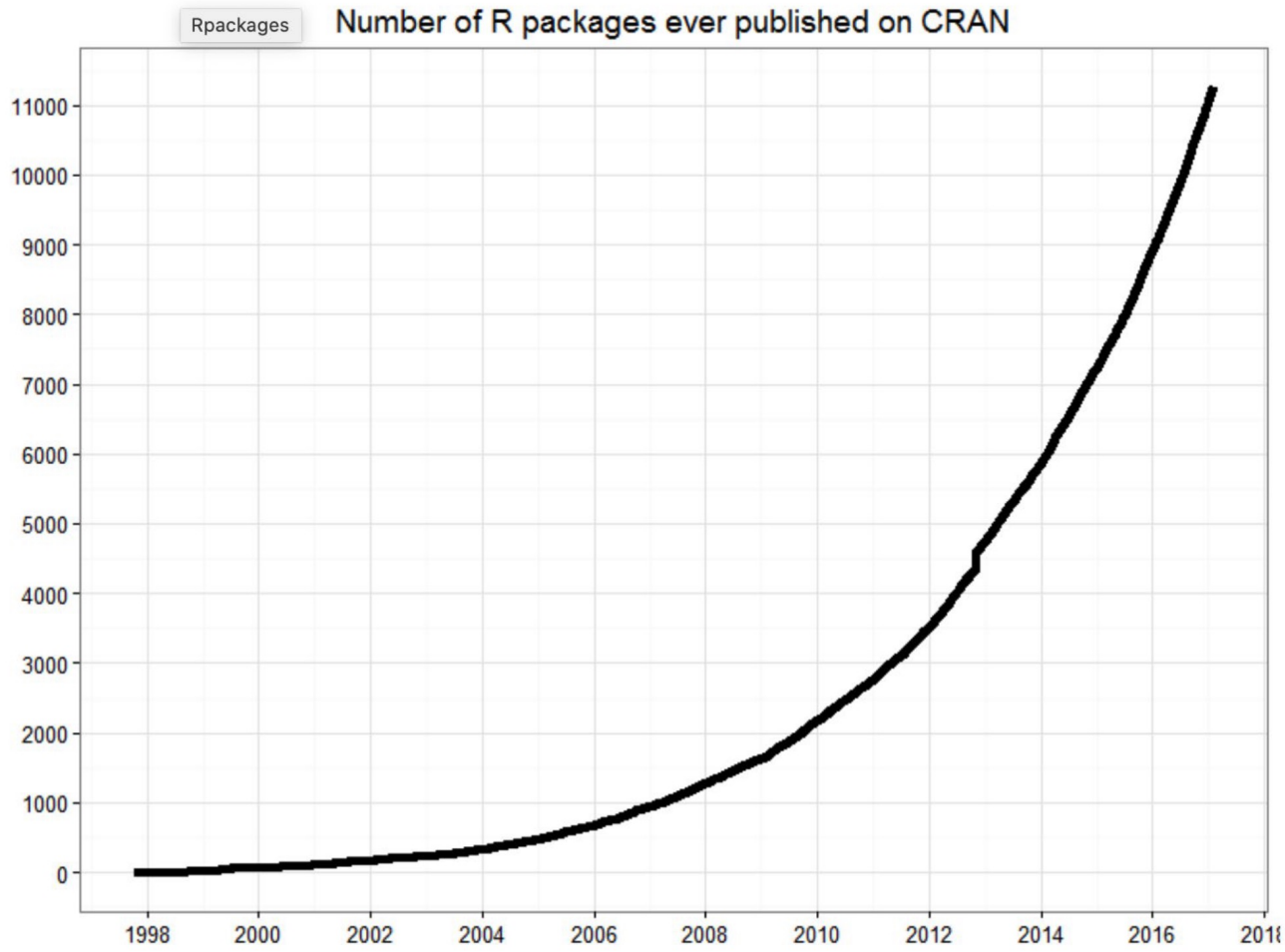
What is R?

- R is a well developed and powerful programming language.
- R is used for data manipulation, statistics, and graphics.
- R is a highly extensible, open-source and free software.
- R compiles and runs on a wide variety of Linux platforms, Windows, and MacOS.

R is made up of

- variables (including integer, numeric, logic) and data structures (including vector, list, array, dataframe, matrix)
- operators (+ - <- * %*% ...) for calculations on variables, arrays & matrices
- large, coherent, integrated collection of functions
- facilities for making unlimited types of publication-quality graphics
- user written functions & sets of functions (packages); 16000+ contributed packages so far & growing, which are well-documented and easily accessible in The Comprehensive R Archive Network (CRAN)
- …

There are over 16K add-on packages



CRAN has over 10,000 R packages in 2018

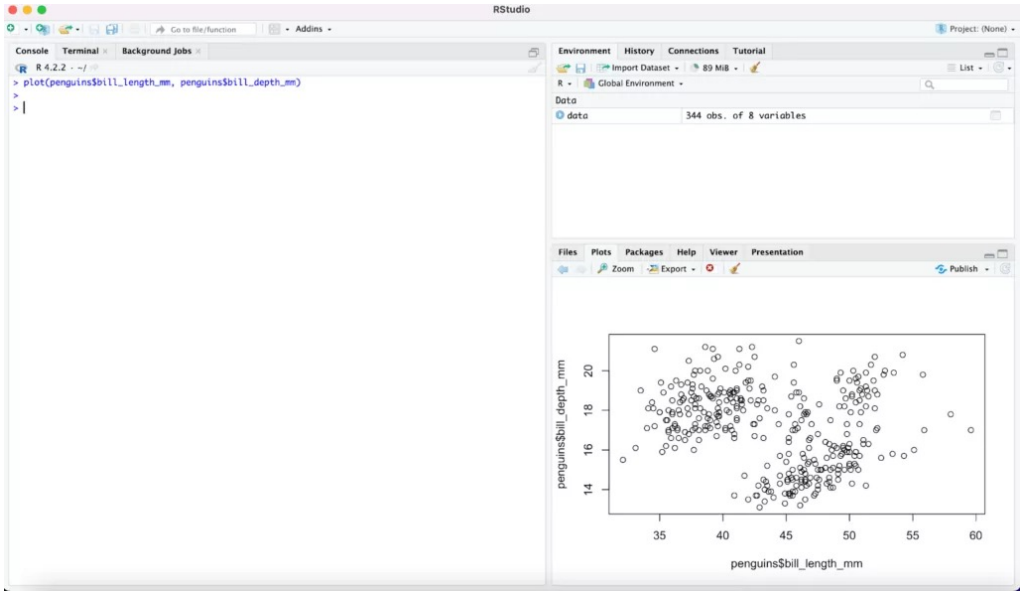
Credit: https://blog.revolutionanalytics.com/2017/01/cran-10000.html

EMBL-EBI

R is interactive

- You can use R interactively within Console or Rstudio

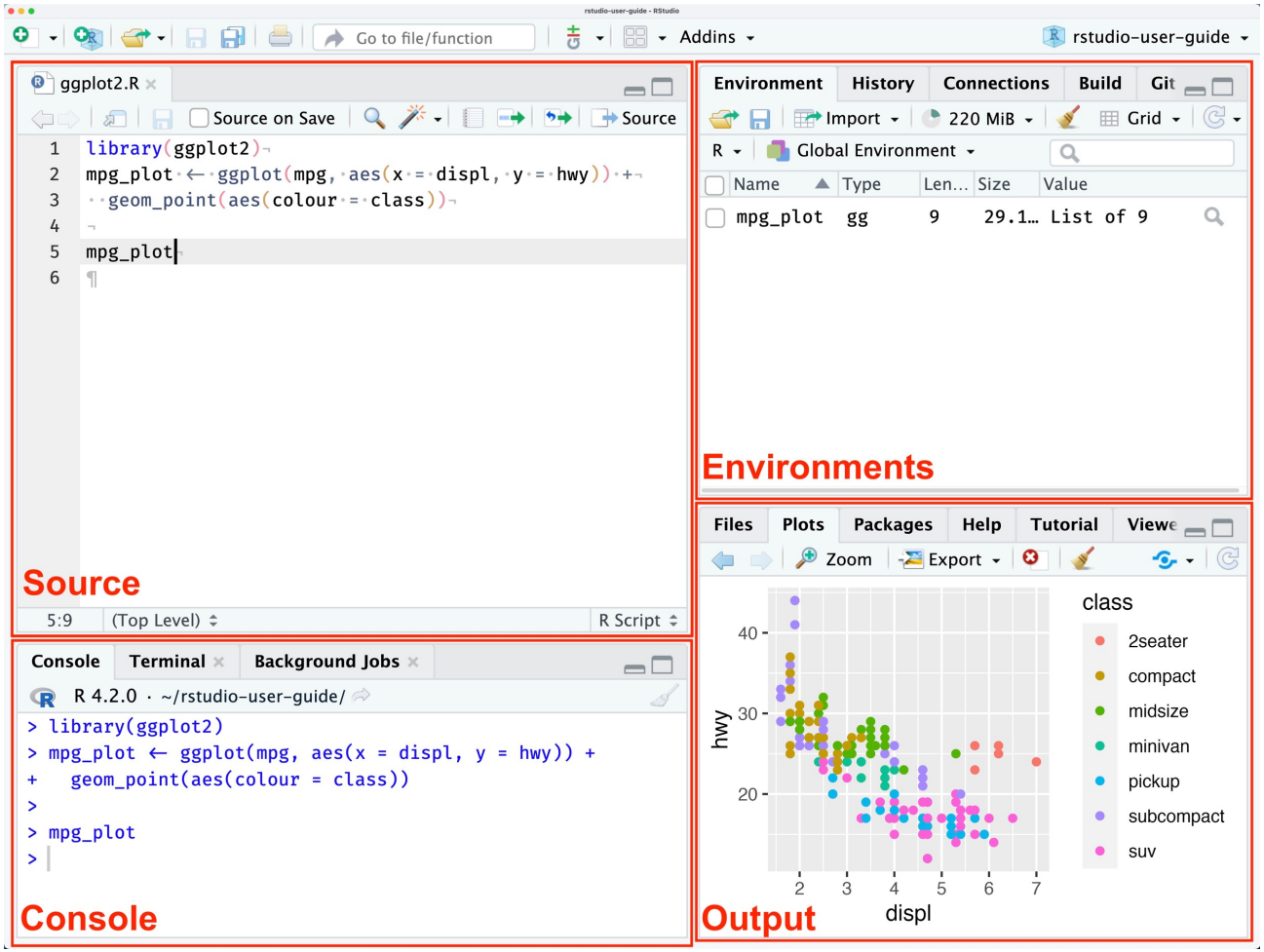
```
(base) ➜  ~ R

R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin20

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.

> a <- 1
> b <- 2
> a + b
[1] 3
>
```



Credit: https://careerfoundry.com/en/blog/data-analytics/what-is-rstudio/

EMBL-EBI

Typical Rstudio session



Credit: https://docs.posit.co/ide/user/ide/get-started/

EMBL-EBI

R variables, data structures and operators

Variables	Example
integer	100
numeric	0.05
character	“hello”
logical	TRUE
factor	“Green”

Arithmetic Operators

```
a + b #Sums two variables
a - b #Subtracts two variables
a * b #Multiply two variables
a / b #Divide two variables
a ^ b #Exponentiation of a variable
a%%b  #Remainder of a variable
a%/%b #Integer division of variables
```

Relational Operators

```
a == b #Tests for equality
a != b #Tests for inequality
a > b #Tests for greater than
a < b #Tests for lower than
a >= b #Tests for greater than or equal to
a <= b #Tests for less than or equal to
```

Logical Operators

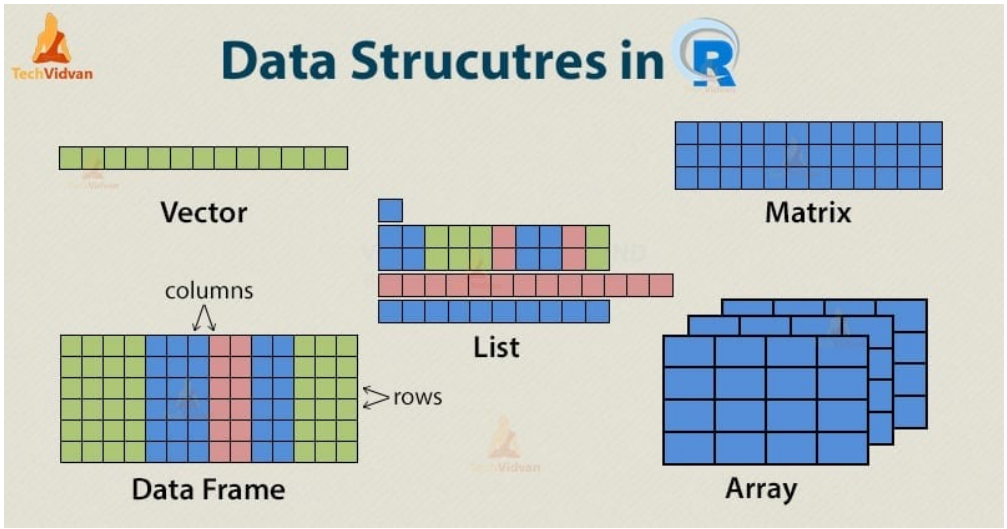
```
!  #Logical NOT
&  #Element-wise logical AND
&& #Logical AND
|  #Element-wise logical OR
|| #Logical OR
```

Assignment Operators

```
x <- 1 # Assigns a variable to x
x = 1 #Assigns a variable to x
```

Other Operators

```
%in% #Identifies whether an element belongs to a vector
$ #Allows you to access objects stored within an object
%>% #Part of magrittr package, it's used to pass objects to functions
```



Credit:
https://sydney-informatics-hub.github.io/lessonbmc/02-BMC_R_Day1_B/index.html
https://techvidvan.com/tutorials/r-data-structures/
https://images.datacamp.com/image/upload/v1697642178/Marketing/Blog/R_Cheat_Sheet_PNG_1.pdf

EMBL-EBI

R functions

```
log(x) #Returns the logarithm of a variable
exp(x) #Returns exponential of a variable
max(x) #Returns maximum value of a vector
min(x) #Returns minimum value of a vector
mean(x) #Returns mean of a vector
sum(x) #Returns sum of a vector
median(x) #Returns median of a vector
```

```
quantile(x) #Percentage quantiles of a vector
round(x, n) #Round to n decimal places
rank(x) #Rank of elements in a vector
signif(x, n) #Round off n significant figures
var(x) #Variance of a vector
cor(x, y) #Correlation between two vectors
sd(x) #Standard deviation of a vector
```

```
mySum <- function(firstNumber, secondNumber){
  result <- firstNumber + secondNumber
  return(result) # or just result
}

mySum(3,4)
```

Credit: https://images.datacamp.com/image/upload/v1697642178/Marketing/Blog/R_Cheat_Sheet_PNG_1.pdf

EMBL-EBI

Conditional execution and loops

- “if else” statement
- For loop
- While loop

```
tempChecker <- function(bodytemp){

  if(bodytemp<36){

      return("too cold")

  }else if(bodytemp>=38){

      return("too hot")

  }else{

      return("normal")
  }

}
```

```
result <- 0

for(count in 1:1000){

  result <- result + count

}

result
```

```
x <- 0
counter <- 0
while(x <= 1000){
  x <- x + 0.6
  counter <- counter + 1
}
counter
```

EMBL-EBI

Data manipulation

```
#This creates the data frame df, seen on the right
df <- data.frame(x = 1:3, y = c("h", "i", "j"), z = 12:14)
```

x	y	z
1	h	12
2	i	13
3	j	14

```
#This selects all columns of the third row
df[3, ]
```

x	y	z
1	h	12
2	i	13
3	j	14

```
#This selects the column z
df$z
```

x	y	z
1	h	12
2	i	13
3	j	14

```
#This selects all rows of the second column
df[ ,2]
```

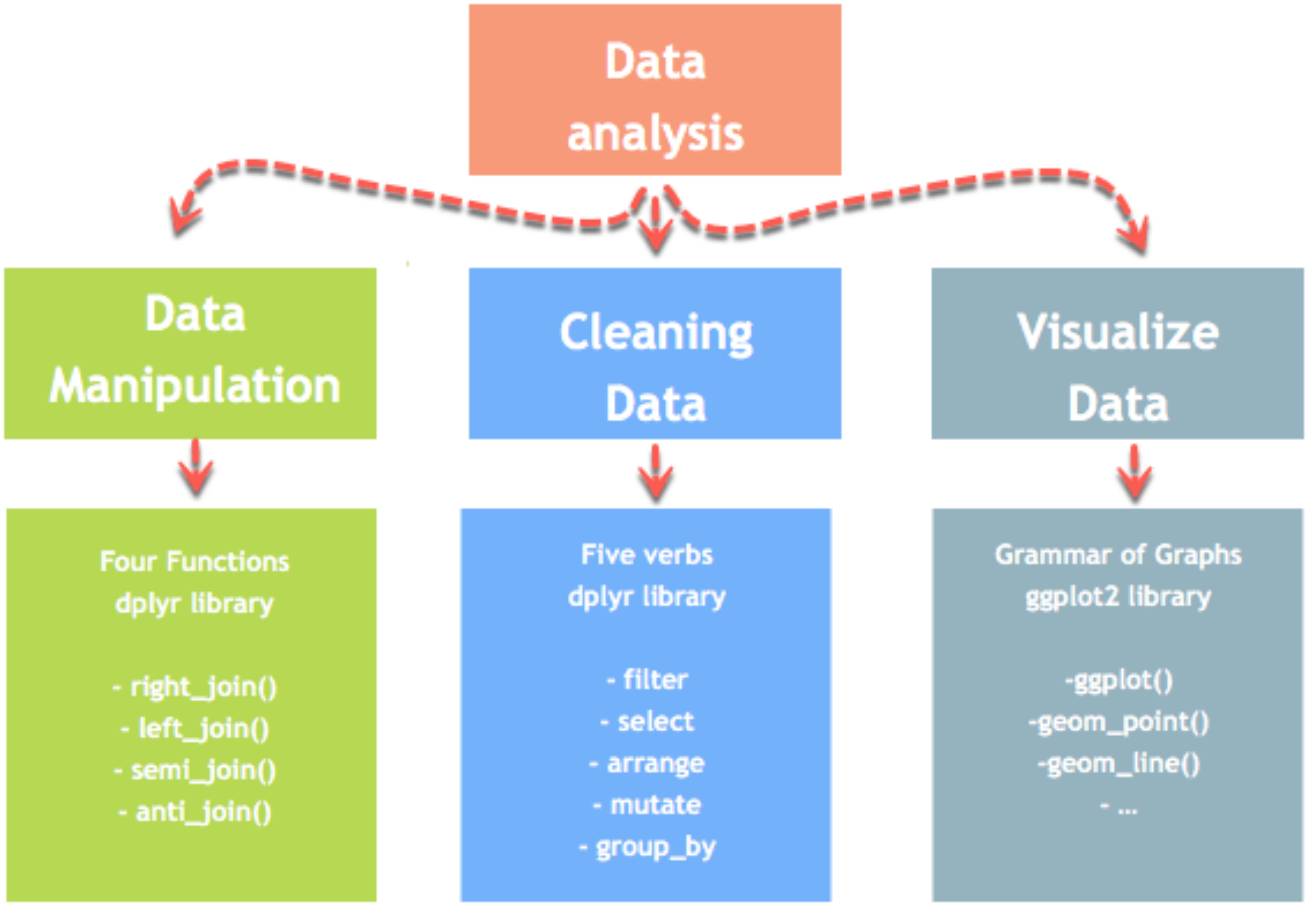
x	y	z
1	h	12
2	i	13
3	j	14

```
#This selects the third column of the second row
df[2,3]
```

x	y	z
1	h	12
2	i	13
3	j	14

Credit: https://images.datacamp.com/image/upload/v1697642178/Marketing/Blog/R_Cheat_Sheet_PNG_1.pdf

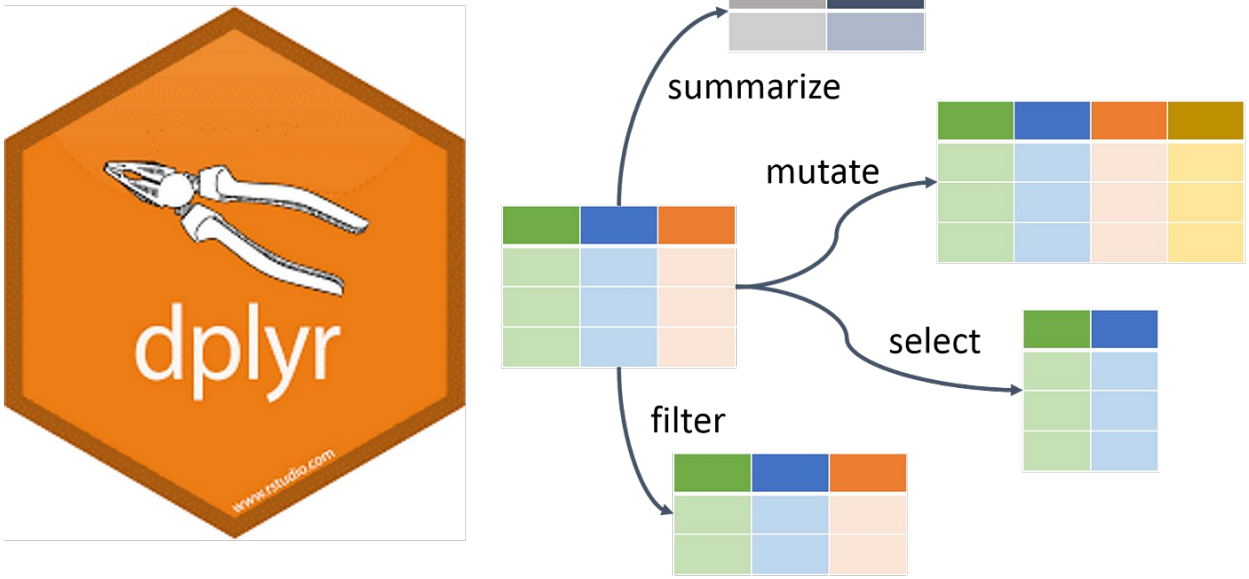
Data manipulation/analysis



Credit: https://www.guru99.com/r-dplyr-tutorial.html

Data manipulation

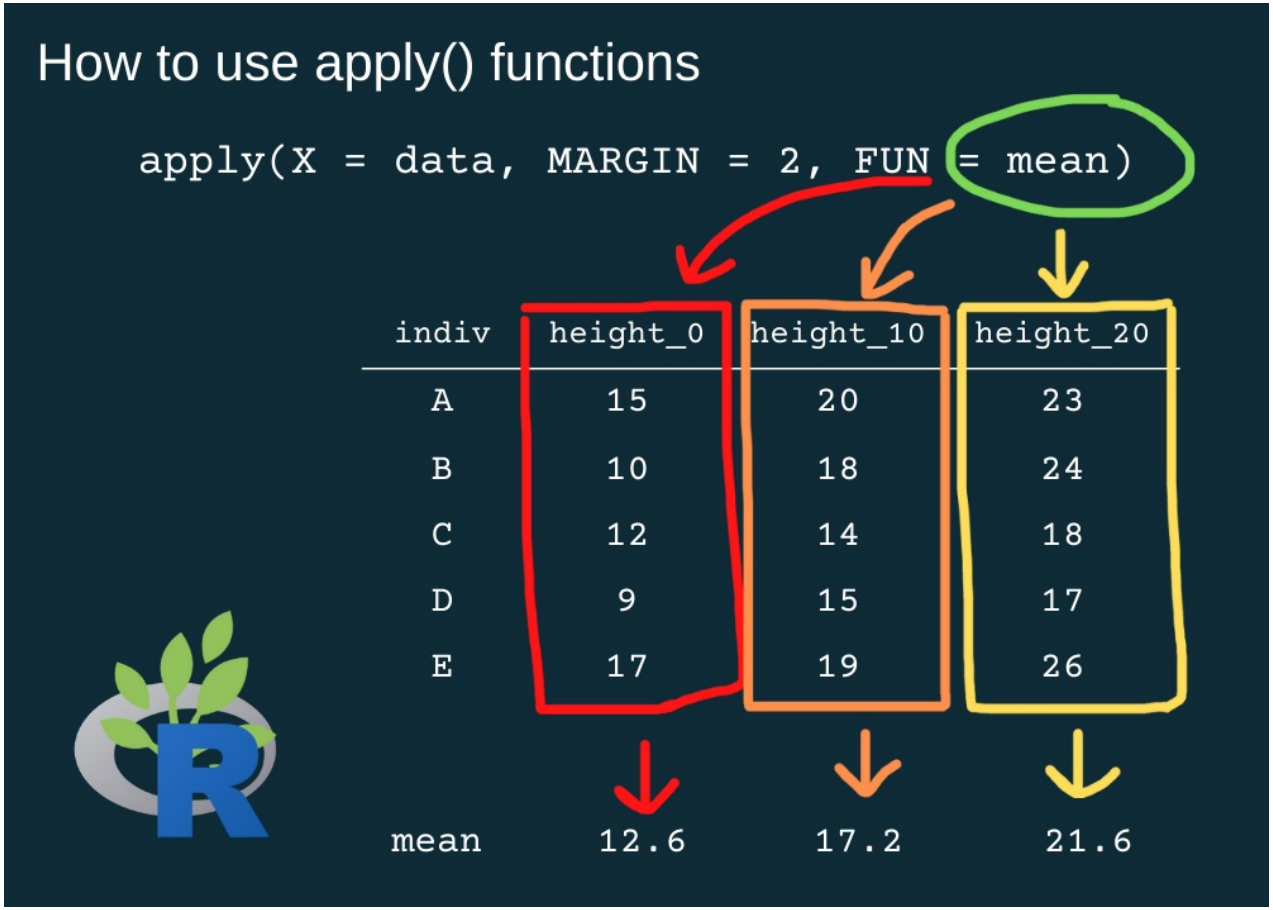
- select()
- filter()
- mutatle()
- group_by()



Credit: https://towardsdatascience.com/data-manipulation-in-r-with-dplyr-3095e0867f75

Data manipulation

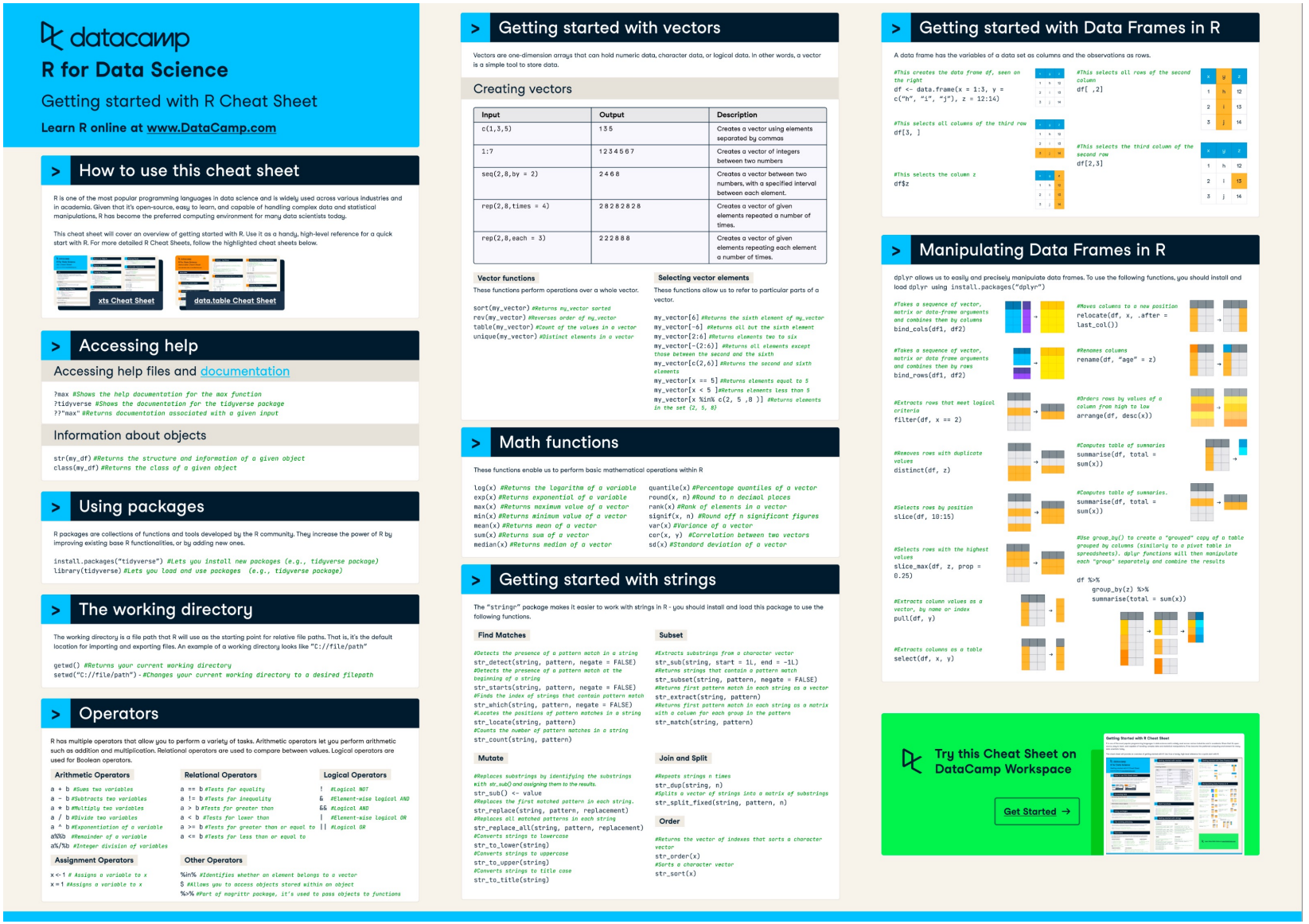
- apply()



Credit: https://www.rforecology.com/post/how-to-use-apply-functions/

EMBL-EBI

You don’t need to remember everything…



datacamp

R for Data Science

Getting started with R Cheat Sheet

Learn R online at www.DataCamp.com

How to use this cheat sheet

R is one of the most popular programming languages in data science and is widely used across various industries and in academia. Given that it’s open-source, easy to learn, and capable of handling complex data and statistical manipulations, R has become the preferred computing environment for many data scientists today.

This cheat sheet will cover an overview of getting started with R. Use it as a handy, high-level reference for a quick start with R. For more detailed R Cheat Sheets, follow the highlighted cheat sheets below.

xts Cheat Sheet

data.table Cheat Sheet

Accessing help

Accessing help files and documentation

```
?max #Shows the help documentation for the max function
?tidyverse #Shows the documentation for the tidyverse package
??"max" #Returns documentation associated with a given input
```

Information about objects

```
str(my_df) #Returns the structure and information of a given object
class(my_df) #Returns the class of a given object
```

Using packages

R packages are collections of functions and tools developed by the R community. They increase the power of R by improving existing base R functionalities, or by adding new ones.

```
install.packages("tidyverse") #Lets you install new packages (e.g., tidyverse package)
library(tidyverse) #Lets you load and use packages  (e.g., tidyverse package)
```

The working directory

The working directory is a file path that R will use as the starting point for relative file paths. That is, it’s the default location for importing and exporting files. An example of a working directory looks like “C://file/path”

```
getwd() #Returns your current working directory
setwd("C://file/path") - #Changes your current working directory to a desired filepath
```

Operators

R has multiple operators that allow you to perform a variety of tasks. Arithmetic operators let you perform arithmetic such as addition and multiplication. Relational operators are used to compare between values. Logical operators are used for Boolean operators.

Arithmetic Operators

```
a + b #Sums two variables
a - b #Subtracts two variables
a * b #Multiply two variables
a / b #Divide two variables
a ^ b #Exponentiation of a variable
a%%b  #Remainder of a variable
a%/%b #Integer division of variables
```

Relational Operators

```
a == b #Tests for equality
a != b #Tests for inequality
a > b #Tests for greater than
a < b #Tests for lower than
a >= b #Tests for greater than or equal to
a <= b #Tests for less than or equal to
```

Logical Operators

```
!  #Logical NOT
&  #Element-wise logical AND
&& #Logical AND
|  #Element-wise logical OR
|| #Logical OR
```

Assignment Operators

```
x <- 1 # Assigns a variable to x
x = 1 #Assigns a variable to x
```

Other Operators

```
%in% #Identifies whether an element belongs to a vector
$ #Allows you to access objects stored within an object
%>% #Part of magrittr package, it’s used to pass objects to functions
```

Getting started with vectors

Vectors are one-dimension arrays that can hold numeric data, character data, or logical data. In other words, a vector is a simple tool to store data.

Creating vectors

Input	Output	Description
c(1,3,5)	1 3 5	Creates a vector using elements separated by commas
1:7	1 2 3 4 5 6 7	Creates a vector of integers between two numbers
seq(2,8,by = 2)	2 4 6 8	Creates a vector between two numbers, with a specified interval between each element.
rep(2,8,times = 4)	2 8 2 8 2 8 2 8	Creates a vector of given elements repeated a number of times.
rep(2,8,each = 3)	2 2 2 8 8 8	Creates a vector of given elements repeating each element a number of times.

Vector functions

These functions perform operations over a whole vector.

```
sort(my_vector) #Returns my_vector sorted
rev(my_vector) #Reverses order of my_vector
table(my_vector) #Count of the values in a vector
unique(my_vector) #Distinct elements in a vector
```

Selecting vector elements

These functions allow us to refer to particular parts of a vector.

```
my_vector[6] #Returns the sixth element of my_vector
my_vector[-6] #Returns all but the sixth element
my_vector[2:6] #Returns elements two to six
my_vector[-(2:6)] #Returns all elements except those between the second and the sixth
my_vector[c(2,6)] #Returns the second and sixth elements
my_vector[x == 5] #Returns elements equal to 5
my_vector[x < 5 ]#Returns elements less than 5
my_vector[x %in% c(2, 5 ,8 )] #Returns elements in the set (2, 5, 8)
```

Math functions

These functions enable us to perform basic mathematical operations within R

```
log(x) #Returns the logarithm of a variable
exp(x) #Returns exponential of a variable
max(x) #Returns maximum value of a vector
min(x) #Returns minimum value of a vector
mean(x) #Returns mean of a vector
sum(x) #Returns sum of a vector
median(x) #Returns median of a vector
quantile(x) #Percentage quantiles of a vector
round(x, n) #Round to n decimal places
rank(x) #Rank of elements in a vector
signif(x, n) #Round off n significant figures
var(x) #Variance of a vector
cor(x, y) #Correlation between two vectors
sd(x) #Standard deviation of a vector
```

Getting started with strings

The “stringr” package makes it easier to work with strings in R - you should install and load this package to use the following functions.

Find Matches

```
#Detects the presence of a pattern match in a string
str_detect(string, pattern, negate = FALSE)
#Detects the presence of a pattern match at the beginning of a string
str_starts(string, pattern, negate = FALSE)
#Finds the index of strings that contain pattern match
str_which(string, pattern, negate = FALSE)
#Locates the positions of pattern matches in a string
str_locate(string, pattern)
#Counts the number of pattern matches in a string
str_count(string, pattern)
```

Subset

```
#Extracts substrings from a character vector
str_sub(string, start = 1L, end = -1L)
#Returns strings that contain a pattern match
str_subset(string, pattern, negate = FALSE)
#Returns first pattern match in each string as a vector
str_extract(string, pattern)
#Returns first pattern match in each string as a matrix with a column for each group in the pattern
str_match(string, pattern)
```

Mutate

```
#Replaces substrings by identifying the substrings with str_sub() and assigning them to the results.
str_sub() <- value
#Replaces the first matched pattern in each string.
str_replace(string, pattern, replacement)
#Replaces all matched patterns in each string
str_replace_all(string, pattern, replacement)
#Converts strings to lowercase
str_to_lower(string)
#Converts strings to uppercase
str_to_upper(string)
#Converts strings to title case
str_to_title(string)
```

Join and Split

```
#Repeats strings n times
str_dup(string, n)
#Splits a vector of strings into a matrix of substrings
str_split_fixed(string, pattern, n)
```

Order

```
#Returns the vector of indexes that sorts a character vector
str_order(x)
#Sorts a character vector
str_sort(x)
```

Getting started with Data Frames in R

A data frame has the variables of a data set as columns and the observations as rows.

```
#This creates the data frame df, seen on the right
df <- data.frame(x = 1:3, y =
c("h", "i", "j"), z = 12:14)

#This selects all columns of the third row
df[3, ]

#This selects the column z
df$z

#This selects all rows of the second column
df[ ,2]

#This selects the third column of the second row
df[2,3]
```

Manipulating Data Frames in R

dplyr allows us to easily and precisely manipulate data frames. To use the following functions, you should install and load dplyr using `install.packages("dplyr")`

```
#Takes a sequence of vector, matrix or data-frame arguments and combines them by columns
bind_cols(df1, df2)

#Takes a sequence of vector, matrix or data frame arguments and combines them by rows
bind_rows(df1, df2)

#Extracts rows that meet logical criteria
filter(df, x == 2)

#Removes rows with duplicate values
distinct(df, z)

#Selects rows by position
slice(df, 10:15)

#Selects rows with the highest values
slice_max(df, z, prop =
0.25)

#Extracts column values as a vector, by name or index
pull(df, y)

#Extracts columns as a table
select(df, x, y)

#Moves columns to a new position
relocate(df, x, .after =
last_col())

#Renames columns
rename(df, “age” = z)

#Orders rows by values of a column from high to low
arrange(df, desc(x))

#Computes table of summaries
summarise(df, total =
sum(x))

#Computes table of summaries.
summarise(df, total =
sum(x))

#Use group_by() to create a “grouped” copy of a table grouped by columns (similarly to a pivot table in spreadsheets). dplyr functions will then manipulate each “group” separately and combine the results

df %>%
    group_by(z) %>%
    summarise(total = sum(x))
```

Try this Cheat Sheet on DataCamp Workspace

Get Started →

Credit: https://images.datacamp.com/image/upload/v1697642178/Marketing/Blog/R_Cheat_Sheet_PNG_1.pdf

EMBL-EBI

Reference

- The content and figures have been credited with links on the relevant slides. These sources are also excellent material for further reading.
- Most of the content presented in the accompanying demonstration material is based on the following material (by Julia Palm from Jena University Hospital, Institute of Medical Statistics, Computer and Data Sciences) with a few modifications: https://bookdown.org/palmjulia/r_intro_script/