Introduction to R and R studio

An introduction to the R statistical framework

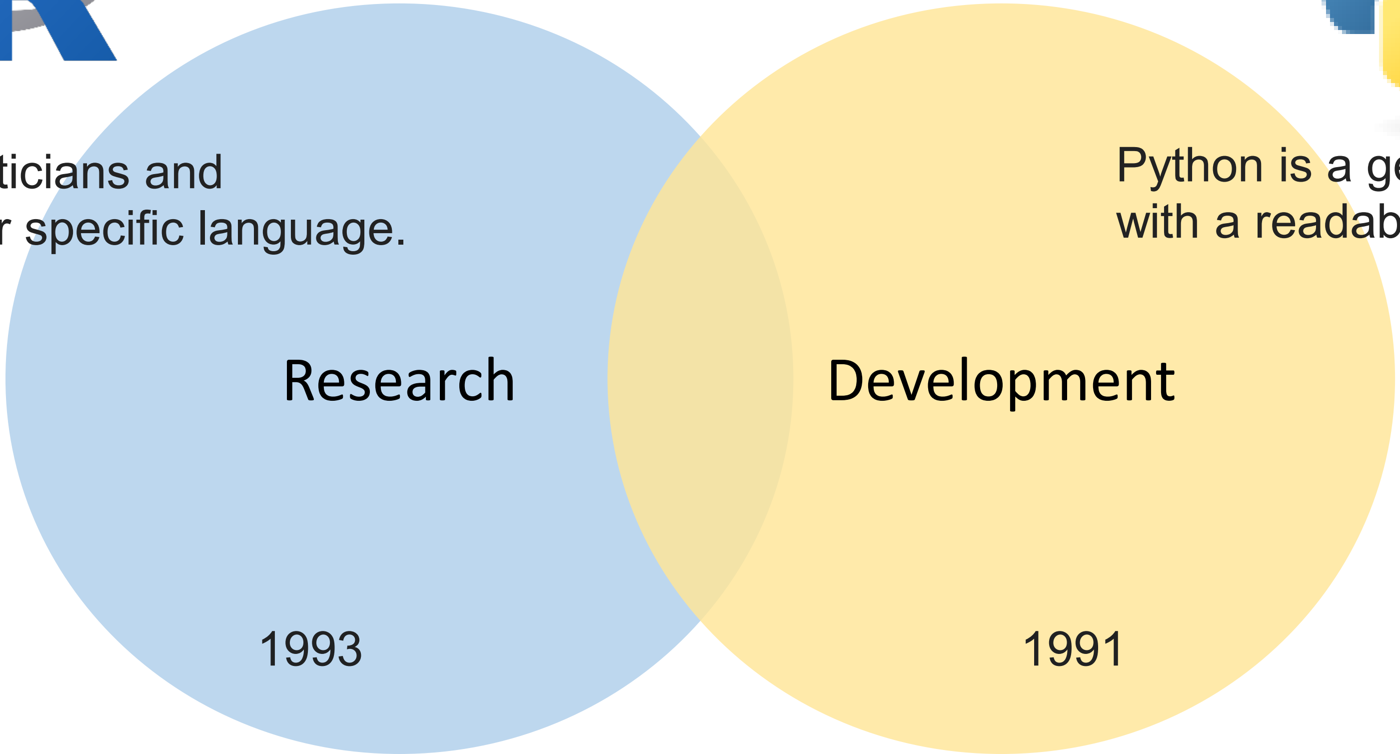
Data Visualization

Prof. Dr. Javier Valdes
Javier.valdes@th-deg.de
Technische Hochschule Deggendorf

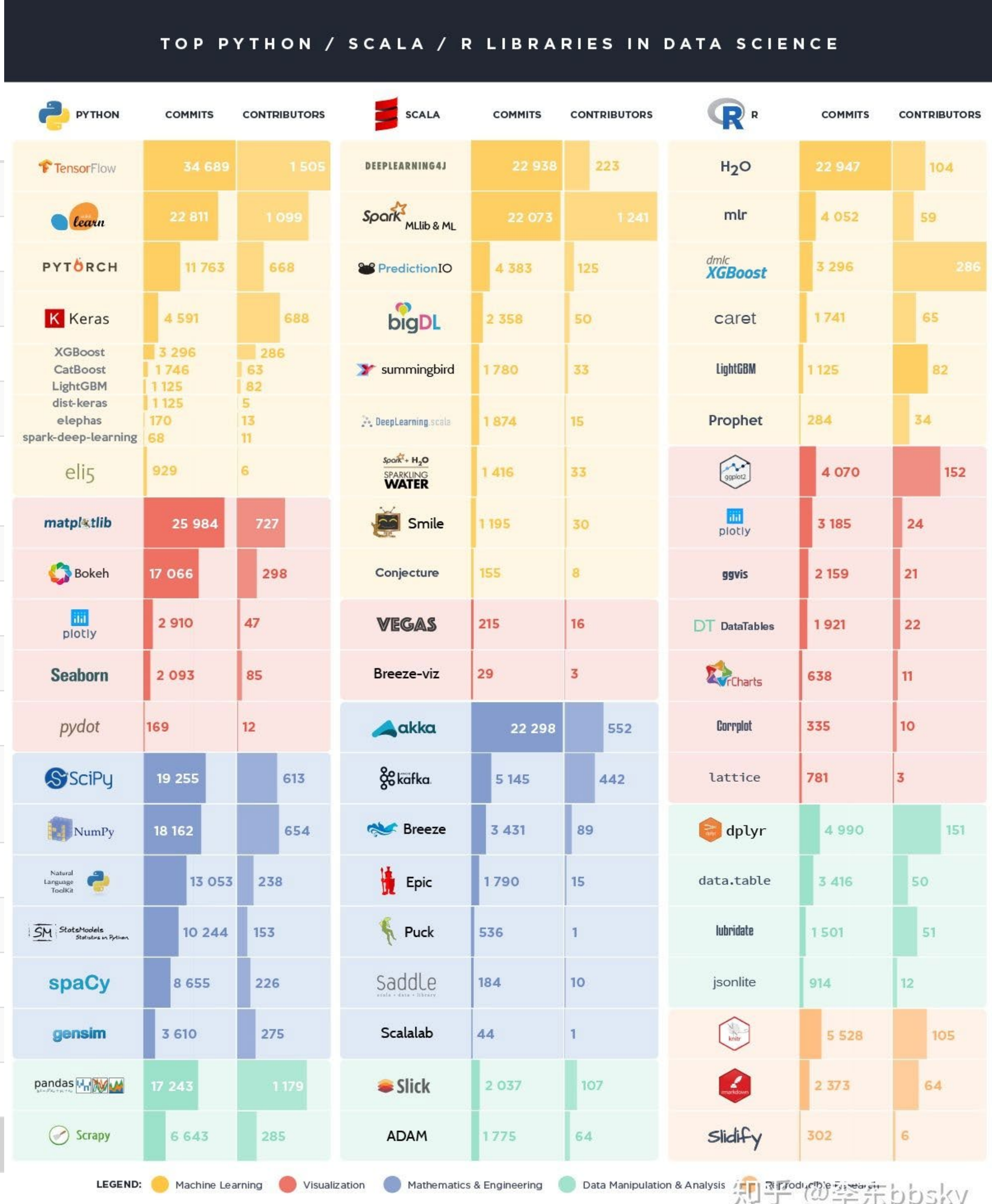
R as a programming language

R is built by statisticians and encompasses their specific language.

Python is a general-purpose language with a readable syntax.

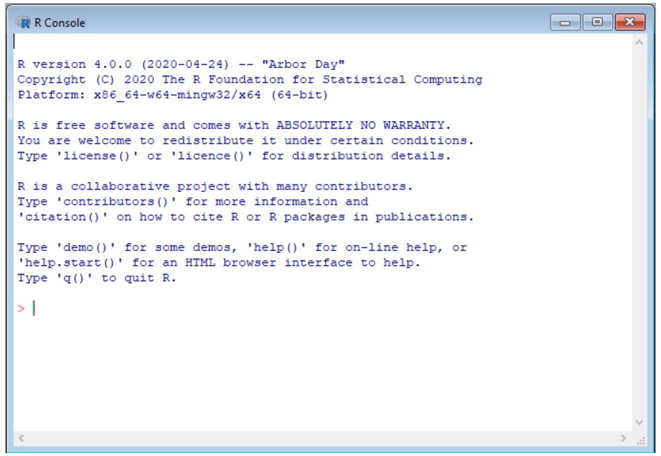


Jan 2021	Jan 2020	Programming Language	Ratings	Change
1	2	C	17.38%	+1.61%
2	1	Java	11.96%	-4.93%
3	3	Python	11.72%	+2.01%
4	4	C++	7.56%	+1.99%
5	5	C#	3.95%	-1.40%
6	6	Visual Basic	3.84%	-1.44%
7	7	JavaScript	2.20%	-0.25%
8	8	PHP	1.99%	-0.41%
9	18	R	1.90%	+1.10%
10	23	Groovy	1.84%	+1.23%
11	15	Assembly language	1.64%	+0.76%
12	10	SQL	1.61%	+0.10%
13	9	Swift	1.43%	-0.36%
14	14	Go	1.41%	+0.51%
15	11	Ruby	1.30%	+0.24%
16	20	MATLAB	1.15%	+0.41%
17	**19**	**Perl**	**1.02%**	**+0.27%**



What is R?

This is an easy question to answer. R is a dialect of S



What is S?

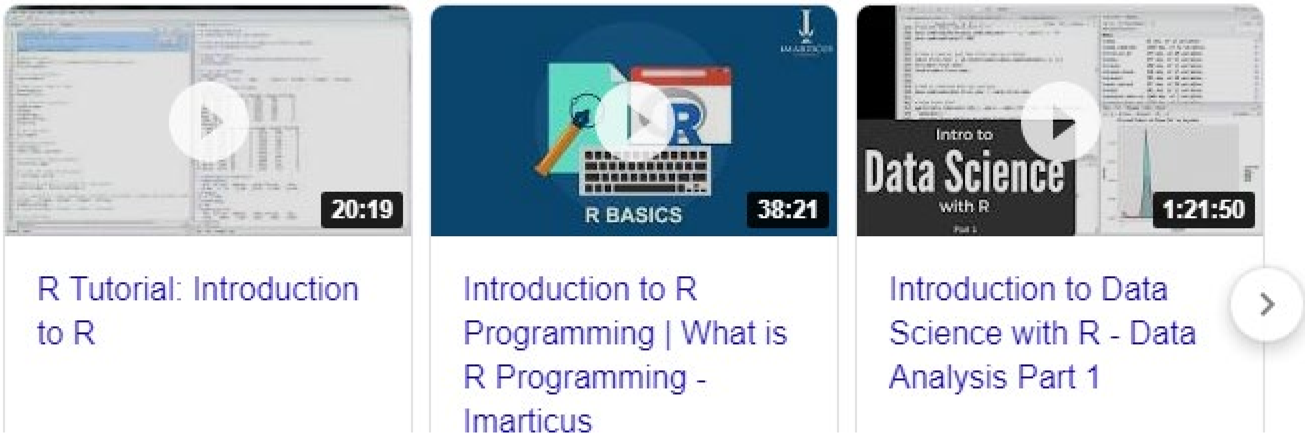
S is a language that was developed by John Chambers and others at the old Bell Telephone Laboratories, originally part of AT&T Corp. S was initiated in 1976 as an internal statistical analysis environment—originally implemented as Fortran libraries. Early versions of the language did not even contain functions for statistical modeling

S language had its roots in data analysis, and did not come from a traditional programming language background. Its inventors were focused on figuring out how to make data analysis easier, first for themselves, and then eventually for others.

OBTAINING R

- Comprehensive R Archive Network: http://cran.r-project.org
- Courses:

  https://www.datacamp.com/
- Videos:



Useful Standard Texts on S and R

Chambers (2008). Software for Data Analysis, Springer

Chambers (1998). Programming with Data, Springer: This book is not about R, but it describes the organization and philosophy of the current version of the S language, and is a useful reference.

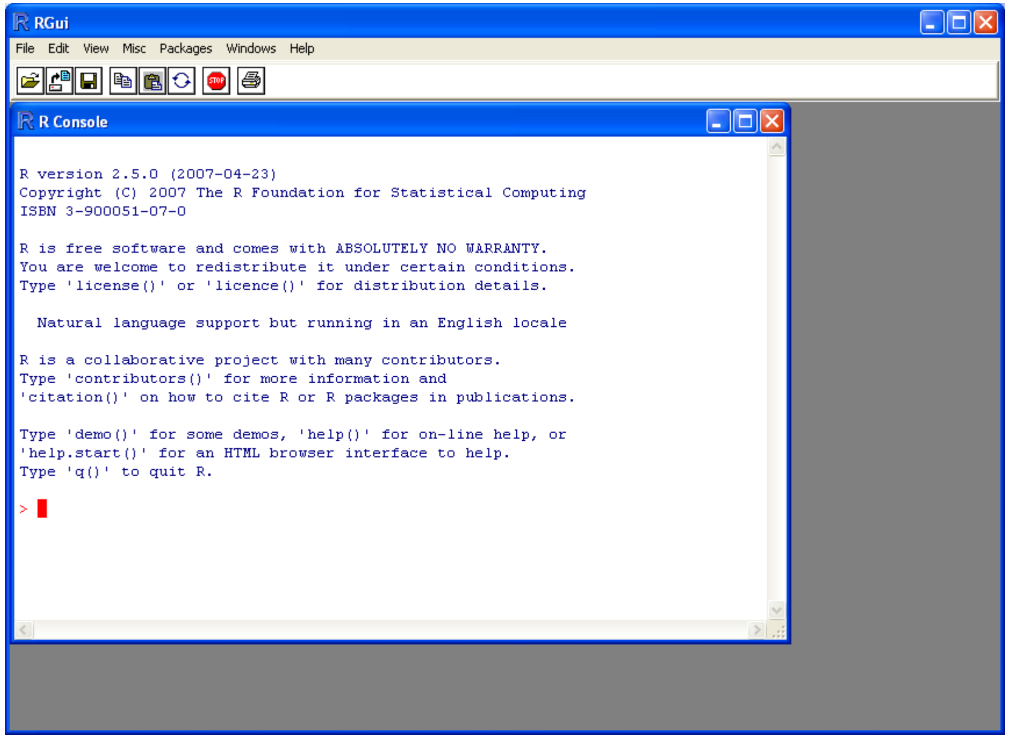
Venables & Ripley (2002). Modern Applied Statistics with S, Springer: This is a standard textbook in statistics and describes how to use many statistical methods in R. This book has an associated R package (the MASS package) that comes with every installation of R.

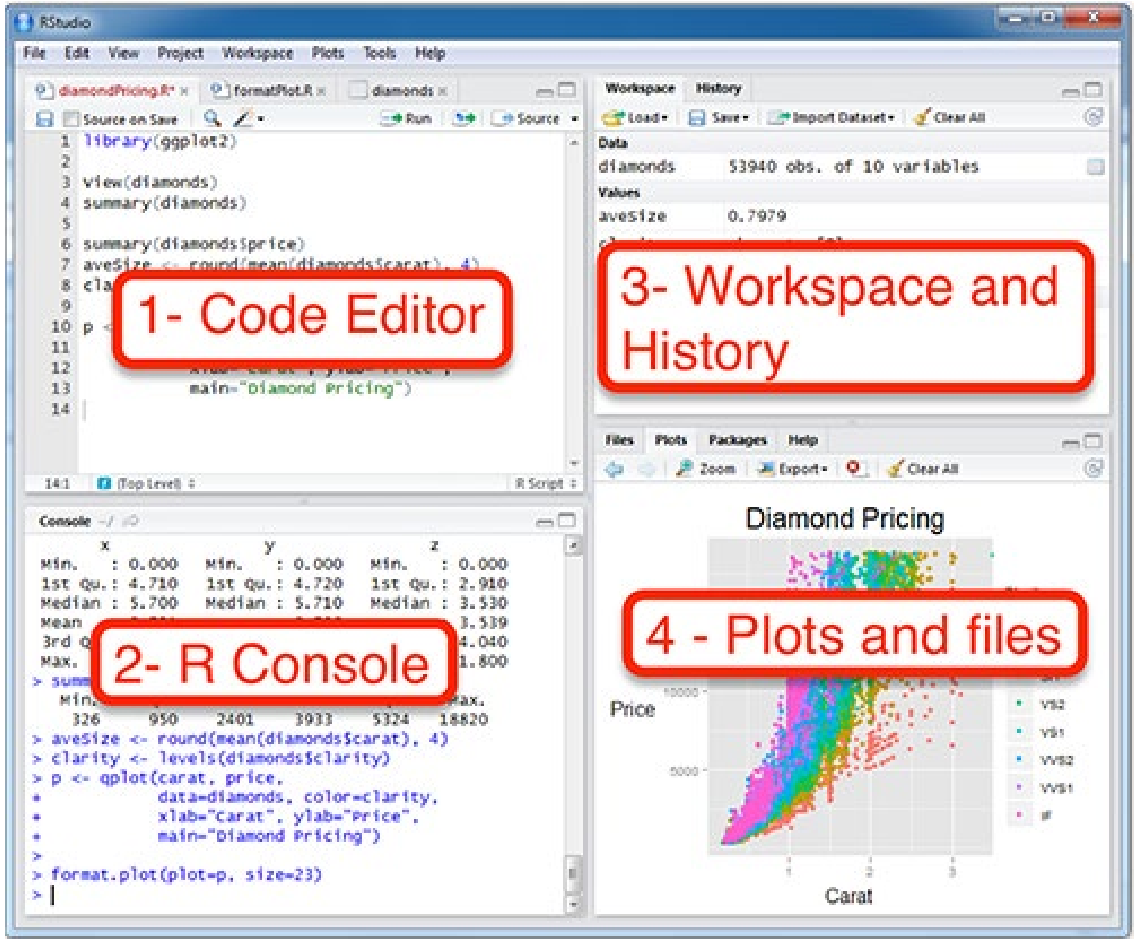
Venables & Ripley (2000). S Programming, Springer: This book is a little old but is still relevant and accurate. Despite its title, this book is useful for R also.

Murrell (2005). R Graphics, Chapman & Hall/CRC Press: Paul Murrell wrote and designed much of the graphics system in R and this book essentially documents the underlying details. This is not so much a "user-level" book as a developer-level book.

Wickham (2014). Advanced R, Chapman & Hall/CRC Press: This book by Hadley Wickham covers a number of areas including object-oriented programming, functional programming, profiling and other advanced topics.

R vs. Rstudio


RGui
File Edit View Misc Packages Windows Help
R Console
R version 2.5.0 (2007-04-23)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
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.
Natural language support but running in an English locale
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.
>


RStudio
File Edit View Project Workspace Plots Tools Help
library(ggplot2)
View(diamonds)
summary(diamonds)
summary(diamonds$price)
main="Diamond Pricing")
1- Code Editor
Workspace History
diamonds 53940 obs. of 10 variables
aveSize 0.7979
3- Workspace and History
Console
2- R Console
> aveSize <- round(mean(diamonds$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)
Files Plots Packages Help
Diamond Pricing
4 - Plots and files
Price
Carat

Exercices

1. Close R
2. Find a package in CRAN
3. Check the Vignettes
4. Check the documentation file
5. Check if there is a publication associated
6. Open R, generate a new project
7. Install the package, check if it has been installed loading the package
8. Check the help file in R

Basic operations

R is a calculator

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Base R

basic functions which let **R** function as a language

- R index starts from 1
- R and some packages come with data included *data()*
- ***NULL*** is not missing, it is nothingness. Null cannot exist within a vector.
- ***NaN*** means "not a number" and it means there is a result, but it cannot be represented
- ***NA*** *explains that the data is just missing for unknown reasons*

Variable Assignment

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

The Environment

`ls()`	List all variables in the environment.
`rm(x)`	Remove x from the environment.
`rm(list = ls())`	Remove all variables from the environment.

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

Exercices

180 sec

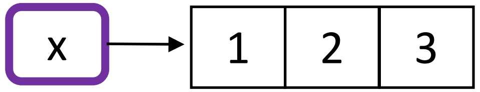
1. Create a vector containing the numbers 1, 2, 3, and 4. We then see how to add 5 to each of the numbers, subtract 10 from each of the numbers, multiply each number by 4, and divide each number by 5.
2. Bind each operation you have done to a new variable
3. take the square root, find e raised to each number, the logarithm and the absolute value
4. get a list of all of the variables that have been defined
5. Remove all the variables in the workspace

Base R

basic functions which let **R** function as a language

Binding basics:

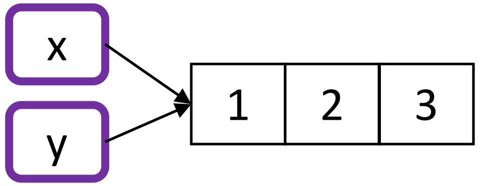
x <- c(1, 2, 3)



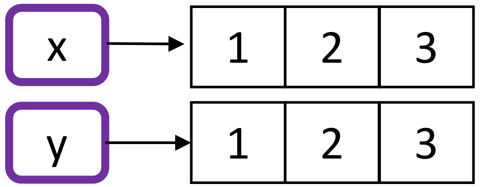
- Creating an object, a vector of values, c(1,2,3)

- And it is binding that object to a name x

y <- x



y[[3]] <- 4



Variable Assignment

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

The Environment

`ls()`	List all variables in the environment.
`rm(x)`	Remove x from the environment.
`rm(list = ls())`	Remove all variables from the environment.

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

Base R

basic functions which let **R** function as a language

R is case sensitive

Error: could not find function “meeen”

Error: object ‘dta’ not found

Error: could not find function “Mean”

R uses . for decimals

Error: unexpected numeric constant in “mean(c(1. 4.”

R will accept a **name** containing **spaces**, but the **spaces** then make it impossible to reference the object in a function

Error: unexpected symbol in “head(Chick Weight”

Maths Functions

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

Base R

basic functions which let **R** function as a language

R is case sensitive

Error: could not find function “meeen”

Error: object ‘dta’ not found

Error: could not find function “Mean”

R uses . for decimals

Error: unexpected numeric constant in “mean(c(1. 4.”

R will accept a **name** containing **spaces**, but the **spaces** then make it impossible to reference the object in a function

Error: unexpected symbol in “head(Chick Weight”

Solution

Escape

Scripts in R

Analyze. Share. Reproduce

What is R Markdown?

.Rmd files • An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.

Reproducible Research • At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.

Dynamic Documents • You can choose to export the finished report in a variety of formats, including html, pdf, MS Word, or RTF documents; html or pdf based slides, Notebooks, and more.

Formatting option	Symbols	Example
Headings	#	#Example Heading
Subheadings	##	##Subheading
Bold	**	**bold text**
Italic	*	*italic text*
Strike through	~~	~~crossed-out text~~
Superscript	^	x^2^
Subscript	~	CO~2~
Bulleted lists	*	* A list item * Another list item * Yet another list item
Numbered lists	1.	1. First list item 2. Second list item 3. Third list item
Horizontal rule	three or more -	----
Line break	two or more spaces plus return	

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Exercices

- Open a new R Markdown file with an output format of HTML. Give the document the title "My class notes".
- Save the file created in exercise 1 as "Notes" in a new project folder
- Remove all of the document text and commands after the metadata section.
- Add a level 2 header with the title of this article.
- Following the header created in the exercise above, write a note to remind yourself of at least one thing about formatting using Markdown
- In the text you wrote for the exercise above, use a text modifier (bold, italic, etc.) to highlight a key work or phrase from the text.
- Demonstrate the use of a chunk to calculate the results of ((43 - 17)*.1)^2
- Same problem as prior problem with the addition of using chunk option(s) to prevent the R source code from being displayed
- Generate a list of items
- Include a link to a website

Data Structures

This chapter summarizes the most important data structures in base R.

Outline:

Introduction to Data Structures
Data structures in R
- Vectors
- Attributes
- Matrices and arrays
- Data frames

Linear Algebra

A *scalar* is an ordinary number, such as 17.

A *matrix* is a rectangular array of numbers with *r* rows and *c* columns. For example, let **X** be the 4 × 3 matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 6 & 3 & 9 \\ 0 & -1 & 8 \\ 5 & 7 & 10 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \end{bmatrix} = [x_{ij}]$$

A row vector is a matrix with only one row
A column vector is a matrix with only one column.

$$\mathbf{y} = \begin{bmatrix} 17 \\ 23 \\ -9 \\ 38 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

Two matrices are equal if and only if

- they have the same dimension
- their corresponding elements are identical
  - i.e. the *ij* element of one matrix is equal to the *ij* element of the other

For example:

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 4 & 3 \\ 2 & 0 \end{bmatrix} \neq \begin{bmatrix} 2 & 0 \\ 4 & 3 \end{bmatrix}$$

How do we sum two matrices?

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix} =$$

How do we multiply two matrices?

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix} =$$

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix} = \begin{bmatrix} 8 & \\ & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ 6 & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ 6 & 0 \end{bmatrix}$$

Data Structures in R

str() # **str**ucture

Dimensions	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Note: scalars are vectors of length one

Commonly used data structure functions in R

vector()	as.vector()	is.vector()
data.frame()	as.data.frame()	is.data.frame()
numeric()	as.numeric()	is.numeric()
list()	as.list()	is.list()
character()	as.character()	is.character()
array()	as.array()	is.array()

Other commonly used data structure functions in R:

as.POSIX , as.table

Vectors

Basic data structure in R

Properties:

- typeof()
- length()
- attributes()

Atomic vectors:

Four common types of atomic vectors that I'll discuss in detail: logical, integer, double (often called numeric), and character.

List:

Elements can be of any type, including lists. You construct lists by using list() instead of c()

Note: R has no concept of row vectors or column vectors

Vectors

Basic data structure in R

Properties:

- typeof()
- length()
- attributes()

Types

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

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'1', '0', '1', levels: '1', '0'`	Character strings with preset levels. Needed for some statistical models.

Vectors - Exercises

180 sec

1. Test your knowledge of vector coercion rules by predicting the output of the following uses of c():

```
c(1, FALSE)
c(TRUE, 1L)
c(list(1), "a")
c("a", 1)
```

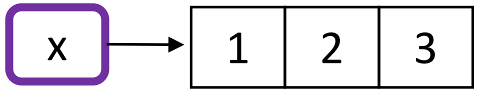
2. Why do you need to use unlist() to convert a list to an atomic vector? Why doesn't as.vector() work?

3. Why is 1 == "1" true? Why is -1 < FALSE true? Why is "one" < 2 false?

Differences between vectors

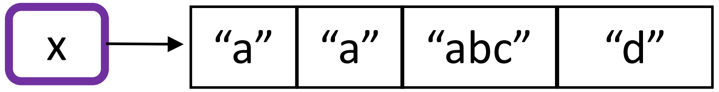
- Vectors:

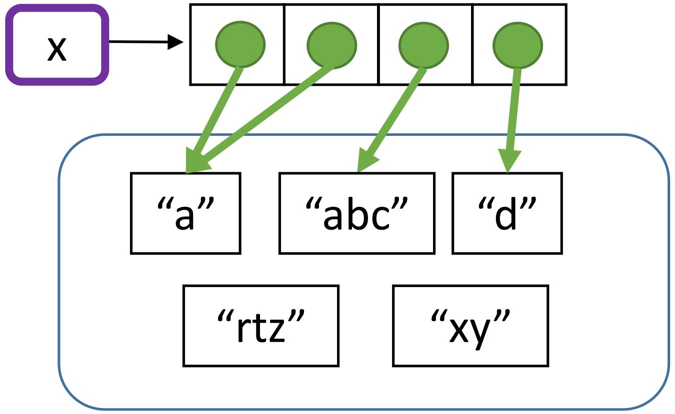
```
x <- c(1, 2, 3)
```



- Character vectors:

```
x <- c("a","a","abc","d")
```



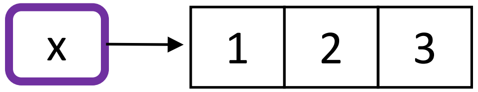


The global string pool

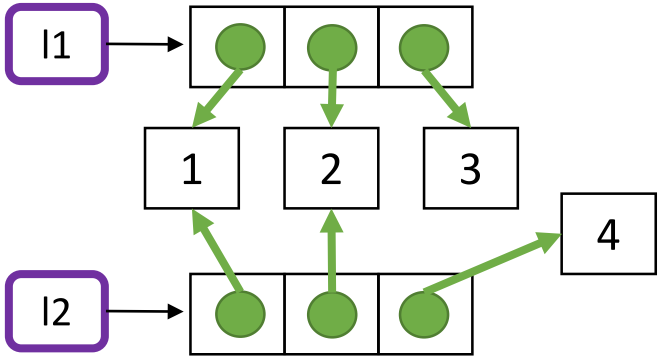
Differences between vectors and lists

- Vectors:

```
x <- c(1, 2, 3)
```

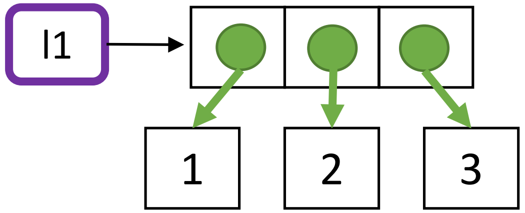



```
l2[[3]] <- 4
```

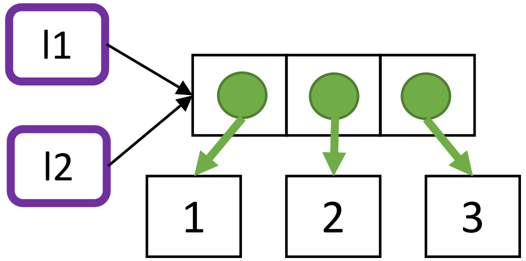


- Lists:

```
l1 <- list(1, 2, 3)
```




```
l2 <- l1
```



Generating data

basic functions which let **R** function as a language

Vectors

Creating Vectors

`c(2, 4, 6)`	2 4 6	Join elements into a vector
`2:6`	2 3 4 5 6	An integer sequence
`seq(2, 3, by=0.5)`	2.0 2.5 3.0	A complex sequence
`rep(1:2, times=3)`	1 2 1 2 1 2	Repeat a vector
`rep(1:2, each=3)`	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

`sort(x)`
Return x sorted.

`rev(x)`
Return x reversed.

`table(x)`
See counts of values.

`unique(x)`
See unique values.

Attributes

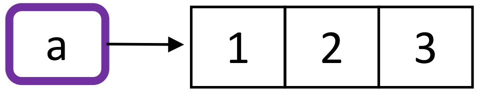
attr() # **Attr**ibutes

All objects can have arbitrary additional attributes, used to store metadata about the object.

Metadata is data that describes other data.

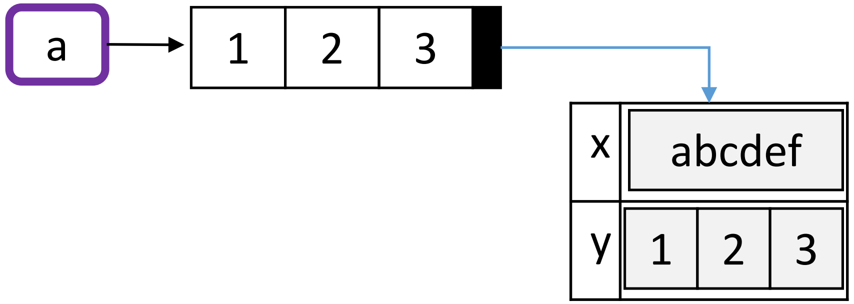
Attributes can be thought of as a named list (with unique names). Attributes can be accessed individually with attr() or all at once (as a list) with attributes().

```
a <- 1:3
```




```
attr (a, “x”) <-”abcdef”
attributes (a)
```

```
attr (a, “y”) <- 4:6
attributes (a)
```

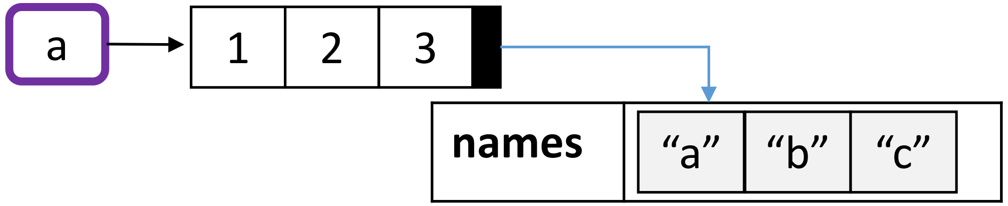


Attributes

names()

Names are special attributes, they are used to label the vector directly and together with the dimensions they are not erased after transforming

```
names(a) <- c("a", "b", "c")
```



dim() #**dim**ensions

The dimension of a vector is not 1-dimensional, but has NULL dimensions. Adding a dimension attribute to a vector allows is to behave like a 2-dimensional **matrix**

Data Structures in R

str() # **str**ucture

Dimensions	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Note: scalars are vectors of length one

Commonly used data structure functions in R

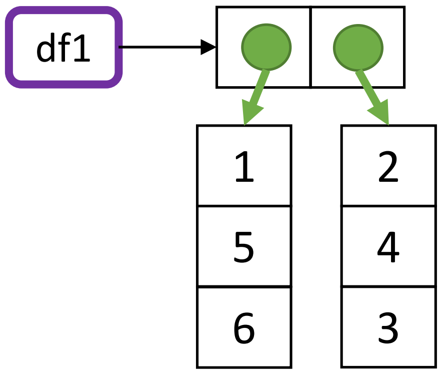
vector()	as.vector()	is.vector()
data.frame()	as.data.frame()	is.data.frame()
numeric()	as.numeric()	is.numeric()
list()	as.list()	is.list()
character()	as.character()	is.character()
array()	as.array()	is.array()

Other commonly used data structure functions in R:

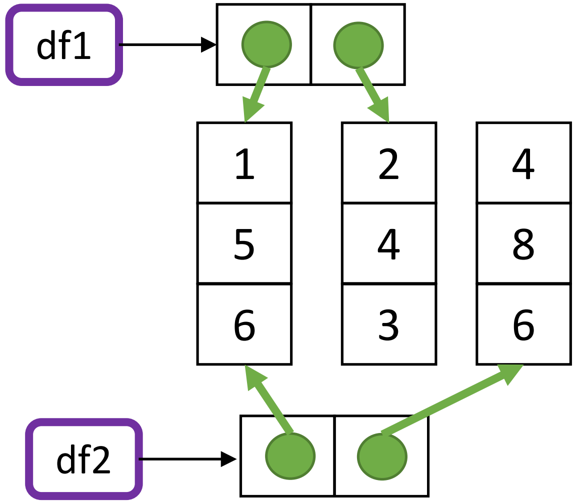
as.POSIX , as.table

Matrices and data frames

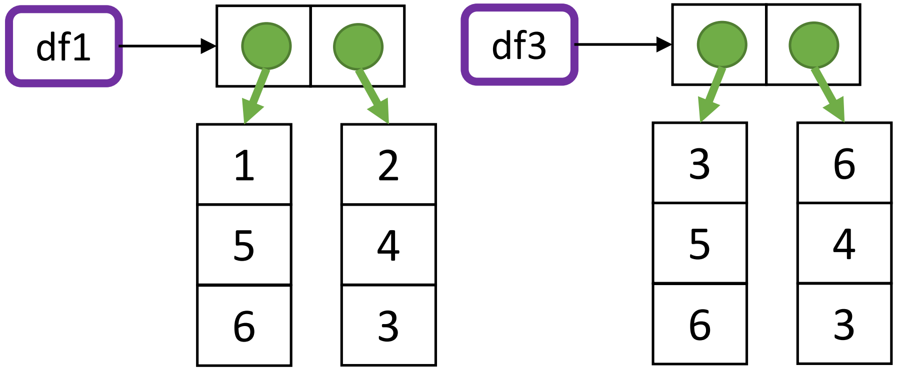
```
df1 <- data.frame(x = c(1, 5, 6), y = c(2, 4, 3))
```




```
df2 <- df1
df2[, 2] <- d2[, 2] * 2
```




```
df3 <- df1
df2[1, ]  <- d3[1, ] * 3
```



Exercicies Matrix: exercicies_matrix.R

1: Create three vectors x,y,z with integers and each vector has 3 elements. Combine the three vectors to become a 3×3 matrix A where each column represents a vector. Change the row names to a,b,c. Think: How about each row represents a vector, can you modify your code to implement it?

2: Please check your result from Exercise 1, using is.matrix(A). It should return TRUE, if your answer is correct. Otherwise, please correct your answer. Hint: Note that is.matrix() will return FALSE on a non-matrix type of input. Eg: a vector and so on.

3: Create a vector with 12 integers. Convert the vector to a 4*3 matrix B using matrix(). Please change the column names to x, y, z and row names to a, b, c, d.

4: Please obtain the transpose matrix of B named tB .

5: Now tB is a 3×4 matrix. By the rule of matrix multiplication in algebra, can we perform tB*tB in R language? (Is a 3×4 matrix multiplied by a 3×4 allowed?) What result would we get?

Exercicies Matrix: exercicies_matrix.R

6: As we can see from Exercise 5, we were expecting that tB*tB would not be allowed because it disobeys the algebra rules. But it actually went through the computation in R. However, as we check the output result , we notice the multiplication with a single * operator is performing the componentwise multiplication. It is not the conventional matrix multiplication. How to perform the conventional matrix multiplication in R? Can you compute matrix A multiplies tB ?

7: If we convert A to a data.frame type instead of a matrix , can we still compute a conventional matrix multiplication for matrix A multiplies matrix A ? Is there any way we could still perform the matrix multiplication for two data.frame type variables? (Assuming proper dimension)

8: Extract a sub-matrix from B named subB . It should be a 3×3 matrix which includes the last three rows of matrix B and their corresponding columns.

9: Compute 3*A , A+subB , A-subB . Can we compute A+B? Why?

10: Generate a n * n matrix (square matrix) A1 with proper number of random numbers, then generate another n * m matrix A2. If we have A1*M=A2 (Here * represents the conventional multiplication), please solve for M.

Subsetting

Selecting Vector Elements

By Position

`x[4]` The fourth element.

`x[-4]` All but the fourth.

`x[2:4]` Elements two to four.

`x[-(2:4)]` All elements except two to four.

`x[c(1, 5)]` Elements one and five.

By Value

`x[x == 10]` Elements which are equal to 10.

`x[x < 0]` All elements less than zero.

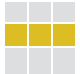
`x[x %in% c(1, 2, 5)]` Elements in the set 1, 2, 5.

Named Vectors

`x['apple']` Element with name 'apple'.

Matrices

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


`m[2, ]` - Select a row


`m[ , 1]` - Select a column


`m[2, 3]` - Select an element

`t(m)`
Transpose

`m %*% n`
Matrix Multiplication

`solve(m, n)`
Find x in: m * x = n

Lists

`l <- list(x = 1:5, y = c('a', 'b'))`
A list is a collection of elements which can be of different types.

`l[[2]]` Second element of l.

`l[1]` New list with only the first element.

`l$x` Element named x.

`l['y']` New list with only element named y.

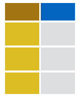
Also see the **dplyr** package.

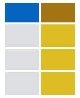
Data Frames

`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

List subsetting

`df$x`


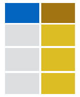
`df[[2]]`


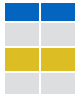
Understanding a data frame

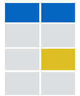
`View(df)` See the full data frame.

`head(df)` See the first 6 rows.

Matrix subsetting

`df[ , 2]`


`df[2, ]`


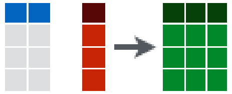
`df[2, 2]`


`nrow(df)`
Number of rows.

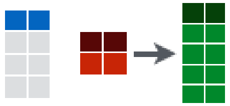
`ncol(df)`
Number of columns.

`dim(df)`
Number of columns and rows.

cbind - Bind columns.



rbind - Bind rows.



Objects in the workspace

basic functions which let **R** function as a language

Working Directory

`getwd()`

Find the current working directory (where inputs are found and outputs are sent).

`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.

Error: '\U' used without hex digits in character string starting ""C:\U"

Getting Help

Accessing the help files

`?mean`

Get help of a particular function.

`help.search('weighted mean')`

Search the help files for a word or phrase.

`help(package = 'dplyr')`

Find help for a package.

More about an object

`str(iris)`

Get a summary of an object's structure.

`class(iris)`

Find the class an object belongs to.

Readinng and writing data in R

Reading and Writing Data

Also see the **readr** package.

Input	Ouput	Description
`df <- read.table('file.txt')`	`write.table(df, 'file.txt')`	Read and write a delimited text file.
`df <- read.csv('file.csv')`	`write.csv(df, 'file.csv')`	Read and write a comma separated value file. This is a special case of read.table/ write.table.
`load('file.RData')`	`save(df, file = 'file.Rdata')`	Read and write an R data file, a file type special for R.

Downloading our data: https://www.stat.berkeley.edu/users/statlabs/labs.html

Exercice

- Read the datacluster file in R
- Check for problems (data cleaning): Are there NA? does all the values make sense?
- Get an overview of the data
- Generate new variables for days and hours with rep
- Generate a variable for dates as a sequence of hours