Introduction to RStudio

Unlocking R Functions with Pokémon Stats
with Professor Sohee Kang

Check-In Form

Data Science & Statistics Society

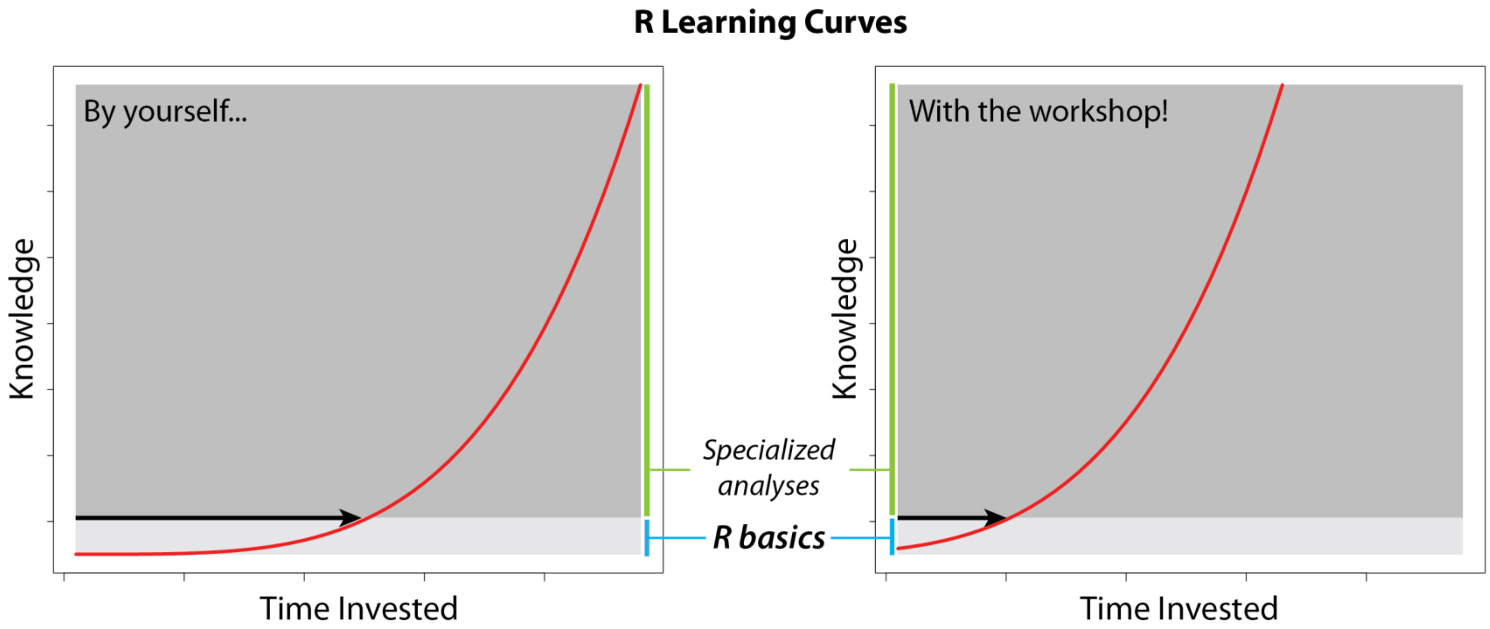
R basics workshop

Sohee Kang

Math and Stats Learning Centre
Department of Computer and Mathematical Sciences

Objective

- To teach the basic knowledge necessary to use R independently, thus helping participants initiate their own process of learning the specific tools needed for their research.



Logistics

- **Website**: https://github.com/theDS3/Intro-to-Data-Science

GitHub Repository:

- R and RStudio Installation Instructions
- Pokemon dataset
- Two R exercises

1. Introduction

What is R?

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

R can be used for: data manipulation, data analysis, creating graphs, designing and running computer simulations.

R is extended by thousands of packages described in CRAN Task Views at: http://cran.wustl.edu/

Why R?

Many tools □□A□A, SPSS, Matlab, Excel…

Advantages

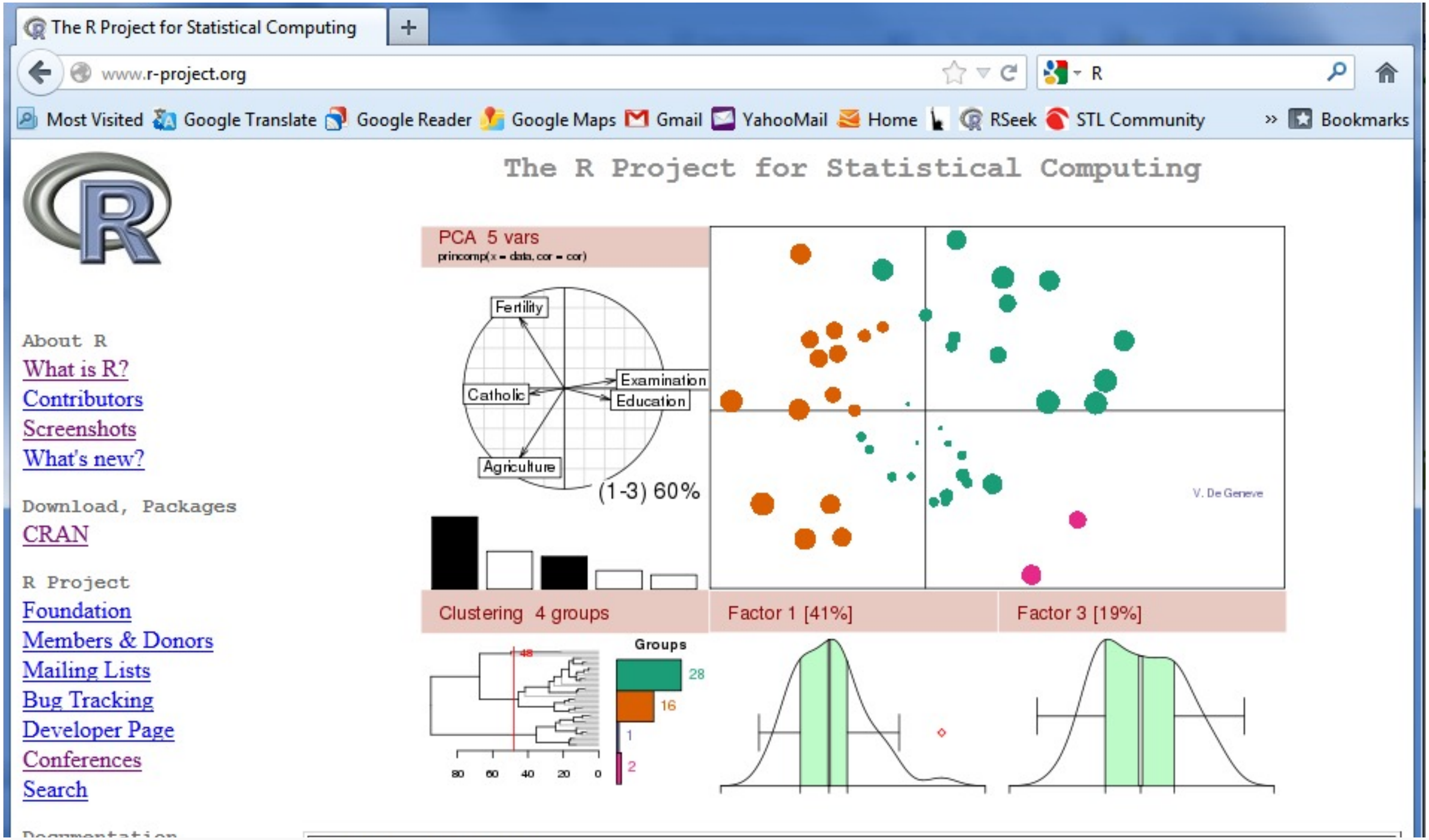
- powerful
- up to date with latest algorithms (packages, Bioconducto□)
- strong community of users
- Preferred by statistics community
- it’s free

Disadvantages

- steep learning curve, but can be useful quickly
- not pretty
- can be memory intensive

How to get R

- From www.r-project.org. R is a GNU project.



The R Project for Statistical Computing

About R

What is R?

Contributors

Screenshots

What's new?

Download, Packages

CRAN

R Project

Foundation

Members & Donors

Mailing Lists

Bug Tracking

Developer Page

Conferences

Search

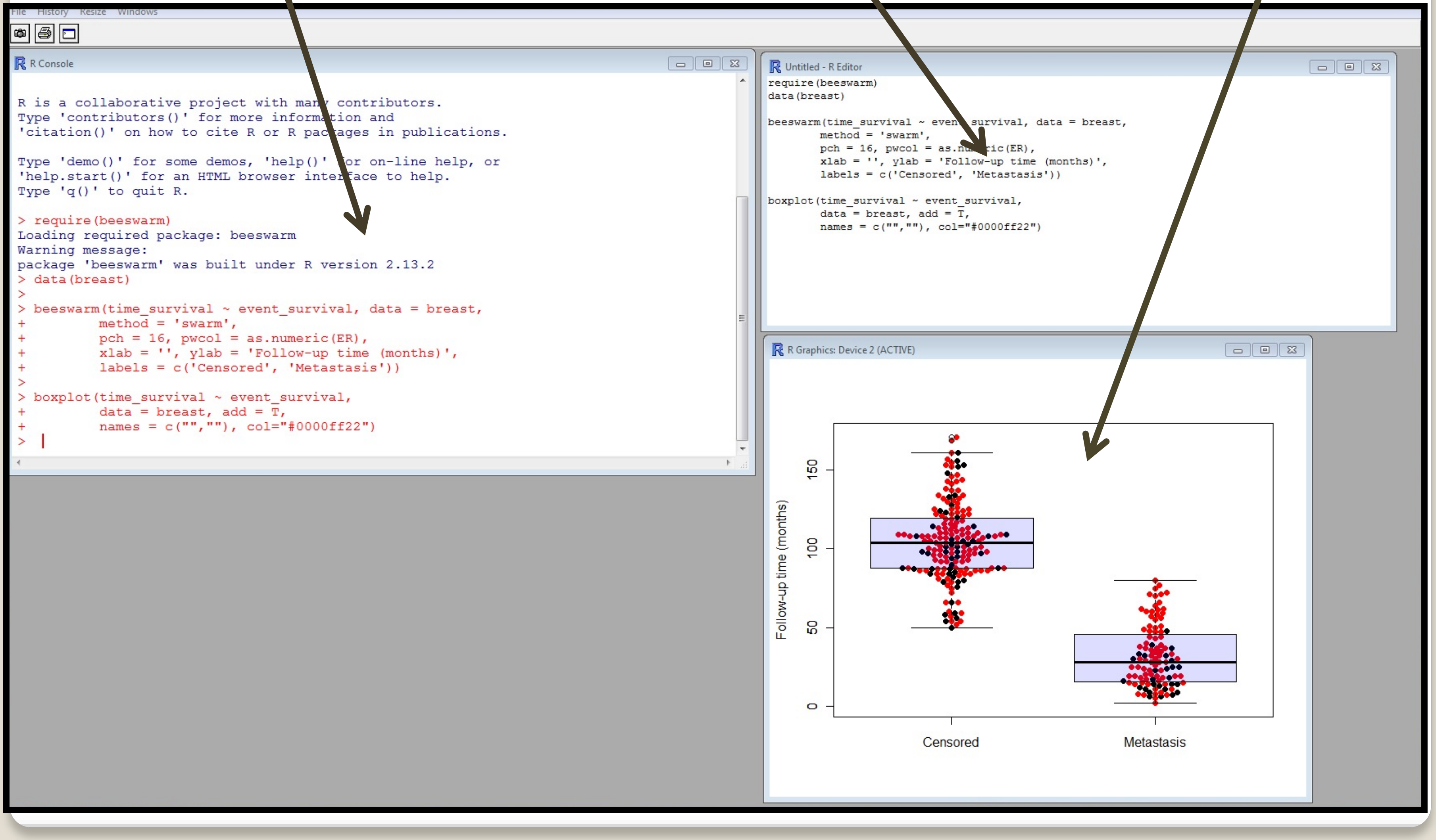
Documentation

Three windows in R

Console

Editor

Graphics



```
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.

> require(beeswarm)
Loading required package: beeswarm
Warning message:
package 'beeswarm' was built under R version 2.13.2
> data(breast)
>
> beeswarm(time_survival ~ event_survival, data = breast,
+         method = 'swarm',
+         pch = 16, pwcol = as.numeric(ER),
+         xlab = '', ylab = 'Follow-up time (months)',
+         labels = c('Censored', 'Metastasis'))
>
> boxplot(time_survival ~ event_survival,
+         data = breast, add = T,
+         names = c("",""), col="#0000ff22")
> |
```

```
require(beeswarm)
data(breast)

beeswarm(time_survival ~ event_survival, data = breast,
        method = 'swarm',
        pch = 16, pwcol = as.numeric(ER),
        xlab = '', ylab = 'Follow-up time (months)',
        labels = c('Censored', 'Metastasis'))

boxplot(time_survival ~ event_survival,
        data = breast, add = T,
        names = c("",""), col="#0000ff22")
```

Advice for learning R

There are no easy ways to learn R except by

1. Typing examples in yourself makes a difference (muscle memory, syntax)
2. Persisting
3. Iterating

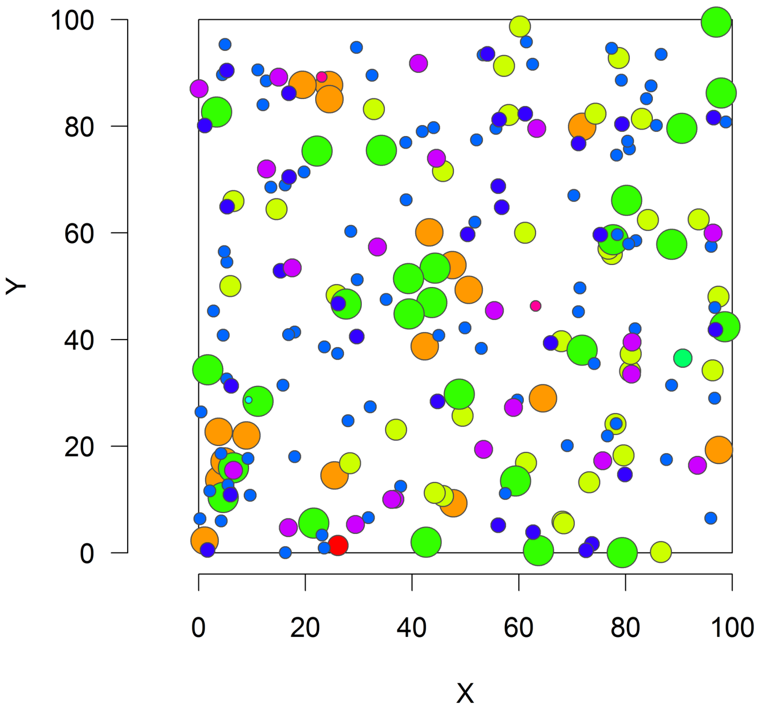
- Steep learning curve, but with a few commands it can be useful quickly
- Try things out – you will not break your data

2. Objects

Object are mainly used to hold data

- In R, an object is a pointer to a piece of memory that contains some information
- One of the main uses for objects is to hold data
- There are many different classes of objects each with its own properties and uses

Objects



- If we counted the number of individuals of tree species in a plot, we can concatenate the values with the function **c**:

```
c(1, 3, 5, 7, 9)
```

Objects are frequently created with the operator "<-"

- These values could be stored in an object with "any" name using the operator "<-":

```
abund <- c(1, 3, 5, 7, 9)

abund
```

- This would just re-write the object:

```
abund <- c(1, 17, 34, 26, 82)
```

Basic classes of R objects

Class	Type of data it holds	Various types of data possible?
numeric (vector)	**numeric**	No
character (vector)	**character**	No
logical (vector)	**logical**	No
matrix	numeric, character or logical	No
array	numeric, character or logical	No
factor	numeric or character	No
data.frame	numeric, character **and/or** logical	**Yes**
list	numeric, character **and/or** logical	**Yes**

Data in objects can be of various types

1. **Numeric**: E.g., number of individuals per species

```
abund <- c(1, 17, 34, 26, 82)

mode(x=abund)

mode(abund)
```

Data in objects can be of various types

1. **Numeric:** E.g., number of individuals per species

2. **Character**: E.g., names of species

```
spp <- c("I.ynga", "I.edulis",
     "I.macrophylla", "I.punctata",
     "I.alba")


spp


mode(spp)
```

Data in objects can be of various types

1. **Numeric:** E.g., number of individuals per species

2. **Character**: E.g., names of species

3. **Logical** (true/false): E.g., increment in abundance?

```
increm <- c(TRUE, FALSE, FALSE, TRUE,
     TRUE)
```

```
increm
```

```
mode(increm)
```

Special values

1. **NA**: missing value; not available; not applicable
2. **Inf** and **-Inf** : infinite

   ```
   100/0

   -100/0

   100 - Inf
   ```

3. **NaN**: not a number

   ```
   Inf - Inf
   ```

4. **NULL**: object missing

An object in R is similar to an object in the real world

- Objects have attributes which define their properties
- Class is one of the main attributes and it helps determine others

Class

Tree
(In the real world)

matrix
(In R)

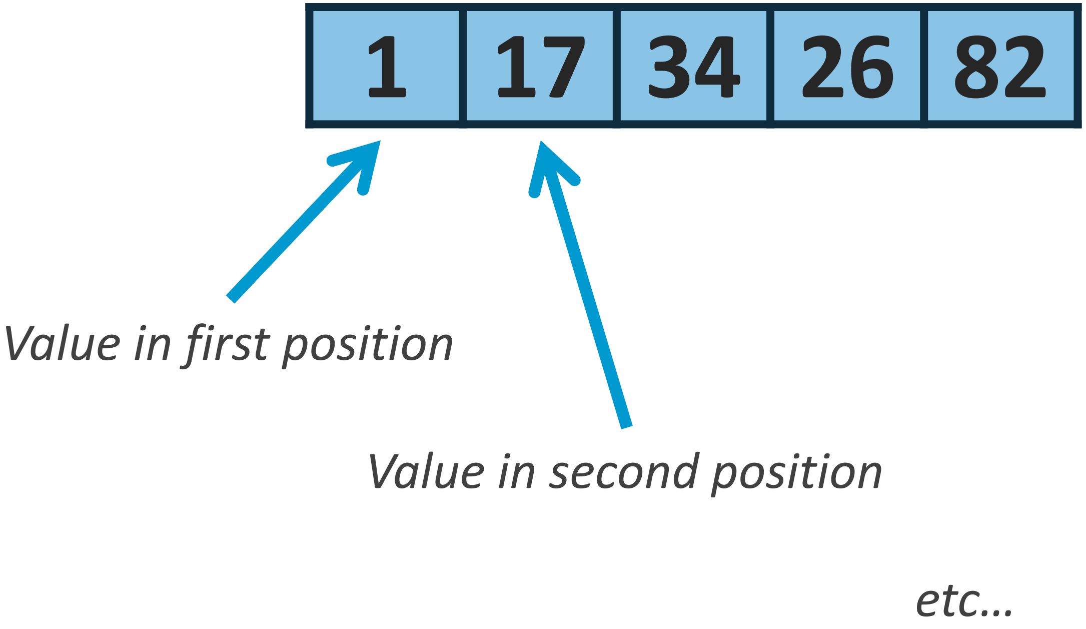
Other attributes

Height
DBH
Leaf shape

Number of rows
Number of columns

Basic classes of R objects – The **Vector**

- Vectors represent a linear sequence of values, e.g.:



1	17	34	26	82

Value in first position

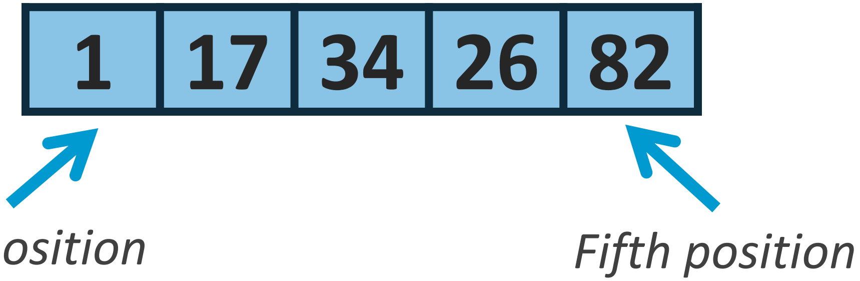
Value in second position

etc…

Basic classes of R objects – **Numeric vectors**

```
abund <- c(1, 17, 34, 26, 82)
class(abund)
```

- **Length** is an important attribute of vectors:



1	17	34	26	82

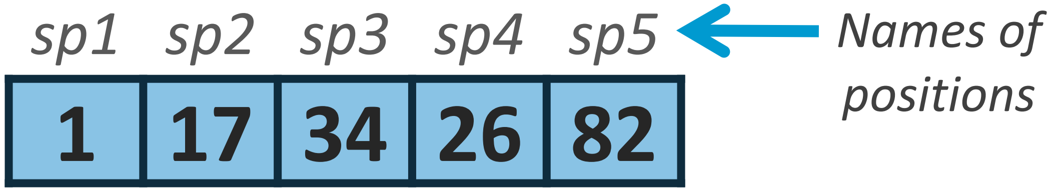
First position

Fifth position

```
length(x=abund)
```

Basic classes of R objects – **Numeric vectors**

- Another one is the **names** of positions in the vector:



sp1 sp2 sp3 sp4 sp5

1	17	34	26	82

Names of positions

```
names(x=abund)

names(abund) <- paste("sp",
     seq(1,length(abund)), sep="")

names(abund)

abund
```

Basic classes of R objects – **Character vectors**

```
spp <- c("I.ynga", "I.edulis",
     "I.macrophylla", "I.punctata",
     "I.alba")
```

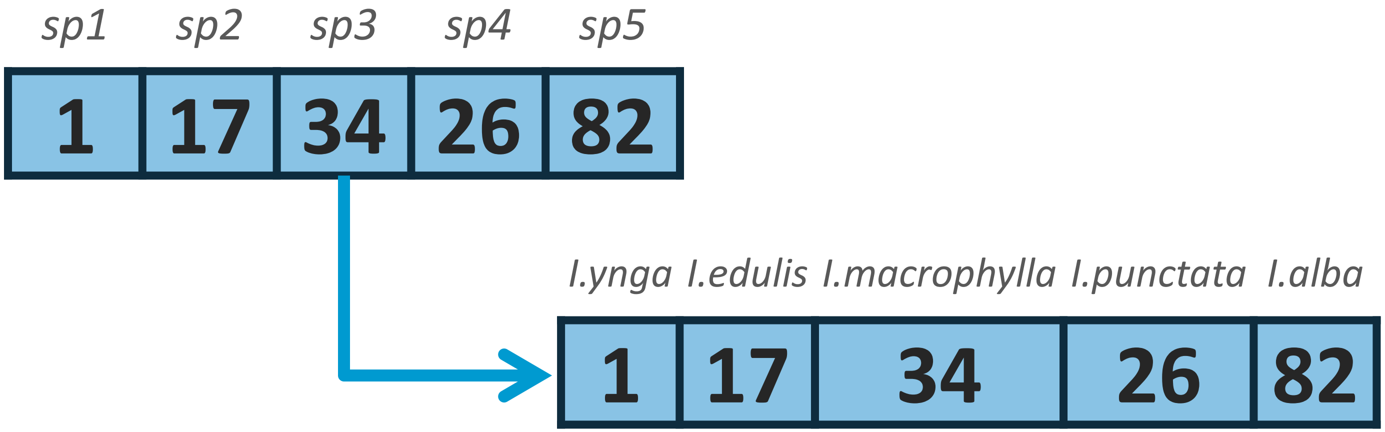
I.ynga	**I.edulis**	**I.macrophylla**	**I.punctata**	**I.alba**

```
class(spp)

length(spp)
```

Basic classes of R objects – **Character vectors**

- We can use one vector to assign names to the other:

```
names(abund)

names(abund) <- spp

abund
```



Basic classes of R objects – **Logical vectors**

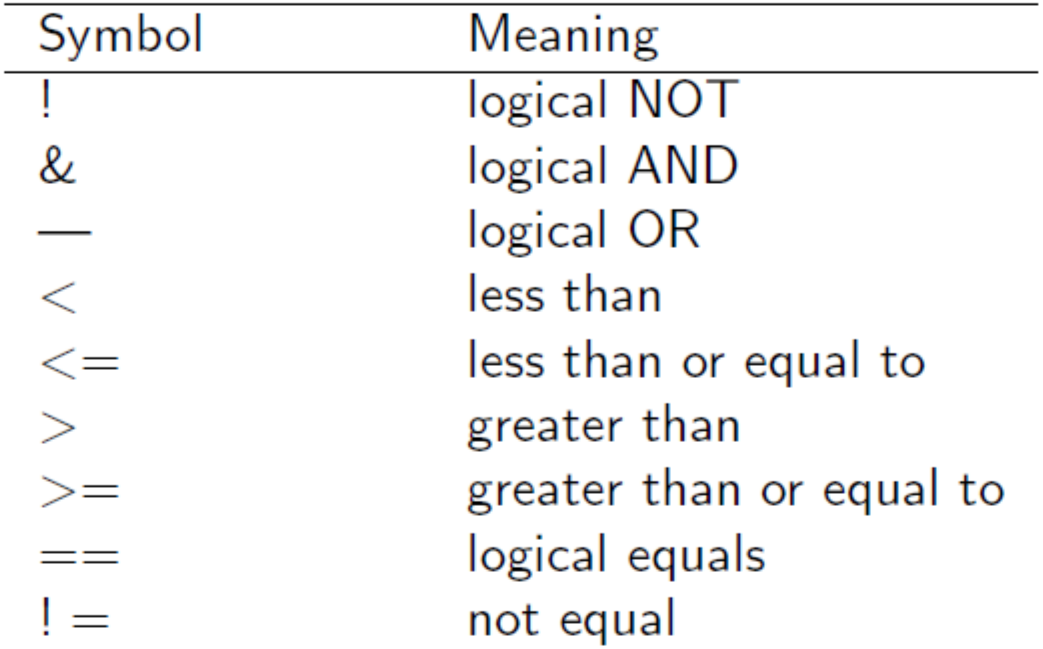
```
increm <- c(TRUE, FALSE, FALSE, TRUE,
     TRUE)
```

```
names(increm) <- spp
```

I.ynga	*I.edulis*	*I.macrophylla*	*I.punctata*	*I.alba*
TRUE	**FALSE**	**FALSE**	**TRUE**	**TRUE**

```
class(increm)

length(increm)

names(increm)
```

Comparisons in R



Symbol	Meaning
!	logical NOT
&	logical AND
—	logical OR
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	logical equals
! =	not equal

Subsetting with Vectors

```
d=c(3,4,7); d
[1] 3 4 7
```

To find out what is stored in a given element of the vector, use [ ]:

```
d[2]
[1] 4
```

To see if the elements of a vector equal a certain number, use ==:

```
d==3
[1] TRUE FALSE FALSE
```

To see if any of the elements of a vector do not equal a certain number, use `!=`:

```
d!=3
[1] FALSE TRUE TRUE
```

To obtain the element number of the vector when a condition is satisfied, use `which()`:

```
which(d==4)
[1] 2
```

To store the result, type: `a=which(d==4); a`

We can also tell R what we *do not* want when subsetting by using the minus - sign. To obtain everything but the 2nd element,

```
d <- seq(1,10,2)
d[-2]
 [1] 1 5 7 9
```

We can use subsetting to explicitly tell R what observations we want to use. To get all elements of d greater than or equal to 2,

```
d[d >= 2]
 [1] 3 5 7 9
```

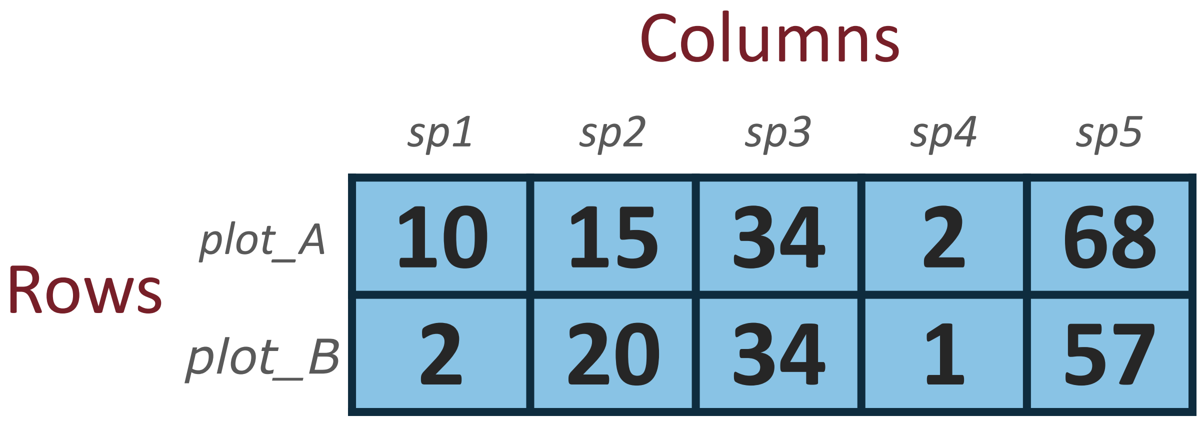
Exercise for Vectors

- Create a vector of the positive odd integers less than 100
- Remove the values greater than 60 and less than 80
- Find the variance of the remaining set of values

Basic classes of R objects – The **Matrix**

Basic classes of R objects – The **Matrix**

- Matrices are data organized in two dimensions: rows and columns



Columns

Rows

	sp1	*sp2*	*sp3*	*sp4*	*sp5*
plot_A	**10**	**15**	**34**	**2**	**68**
plot_B	**2**	**20**	**34**	**1**	**57**

- Matrices can hold **numeric, character** or **logical** data

Basic classes of R objects – The **Matrix**

- One way to create a matrix is the the function “`matrix`”

```
seq(1,10)

Abund <- matrix(seq(1,10), ncol=5)

abund
Abund

class(abund)
class(Abund)
```

Basic classes of R objects – The **Matrix**

- Matrices can be filled by columns (predetermined option) or by rows

```
Abund.2 <- matrix(seq(1,10), ncol=5,
     byrow=TRUE)

Abund
Abund.2

identical(Abund, Abund.2)

?matrix
```

Basic classes of R objects – The **Matrix**

- A matrix with abundance data:

```
v1 <- c(10,2,15,20,34,34,2,1,68,57)

Abund <- matrix(v1, ncol=5)
Abund
```

- Vectors have length, matrices also have dimensions

```
dim(Abund)

ncol(Abund)
nrow(Abund)

length(Abund)
```

Basic classes of R objects – The **Matrix**

- For matrices, we can put names to colums and rows

```
spp

colnames(Abund)

colnames(Abund) <- spp
rownames(Abund) <- c("plot_A", "plot_B")

Abund
```

Basic classes of R objects – The **Matrix**

- Matrices can also hold character or logical values

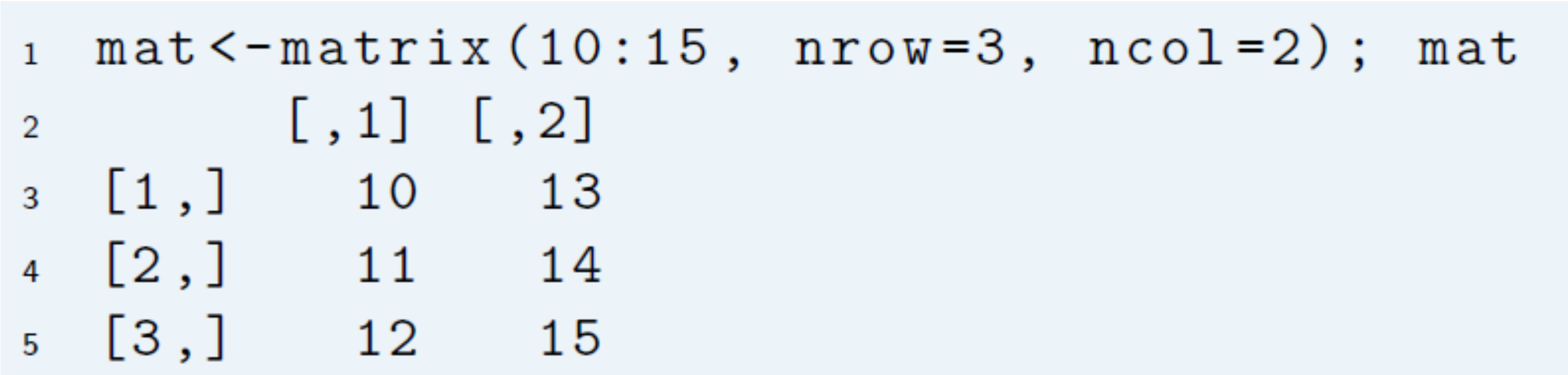
```
Spp <- matrix(esp, ncol=5, nrow=2,
      byrow=TRUE)
Spp

Increm <- matrix(increm, ncol=5, nrow=2,
      byrow=TRUE)
Increm

mode(Abund)
mode(Spp)
mode(Increm)
```

Basic classes of R objects – The **Matrix**

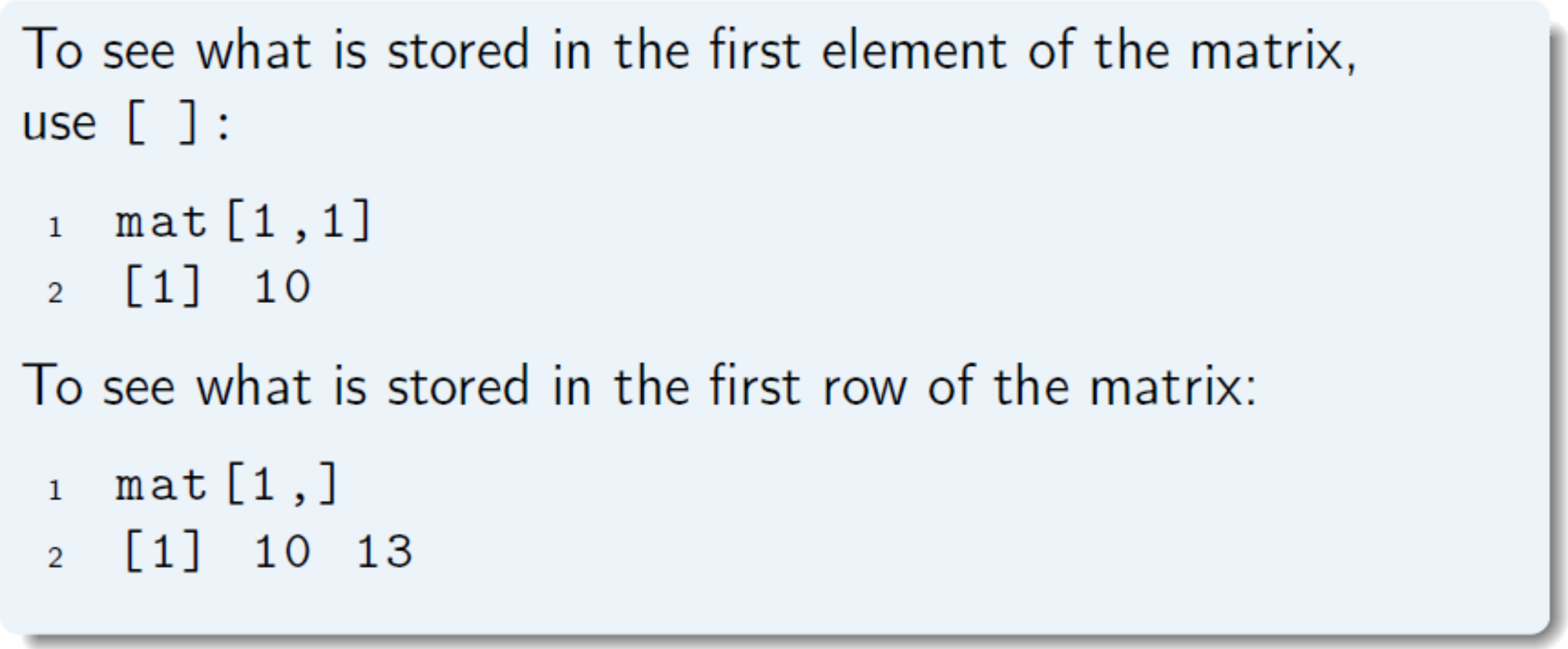
- What happens when we try to merge a character and numeric vectors into the same matrix?

```
Abund
mode(abund)

spp
mode(spp)

Mixed.matrix <- cbind(spp, abund)

Mixed.matrix

mode(Mixed.matrix)
```

Subsetting with Matrices



```
mat<-matrix(10:15, nrow=3, ncol=2); mat
      [,1] [,2]
[1,]    10   13
[2,]    11   14
[3,]    12   15
```



To see what is stored in the first element of the matrix, use [ ]:

```
mat[1,1]
[1] 10
```

To see what is stored in the first row of the matrix:

```
mat[1,]
[1] 10 13
```

To see what is stored in the second column of the matrix:

```
mat[, 2]
[1] 13 14 15
```

To extract elements 1 and 3 from the second column, use `c()` and `[ ]`:

```
mat[c(1,3), 2]
[1] 13 15
```

Basic classes of R objects – **Data Frame**

- Data frames organize data in two dimensions, each column is a variable; variables can be of different types

Variables

Observations

spp.code	Species	Abund_2003	Abund_2012	Increase
1	I.ynga	10	12	TRUE
2	I.edulis	15	9	FALSE
3	I.macrophylla	34	15	FALSE
4	I.punctata	2	3	TRUE
5	I.alba	68	75	TRUE

Basic classes of R objects – **Data Frame**

- Data frames organize data in two dimensions, each column is a variable; variables can be of different types

```
spp

Abund
t(Abund)

increm

spp.code<-1:length(spp)

Data<-data.frame(spp.code, spp, t(Abund),
     increm)
Data
```

Basic classes of R objects – **Data Frame**

```
class(Data)

Data.M <- as.matrix(Data)

class(Data.M)
Data.M

mode(Data.M)

dim(Data)
dim(Data.M)

length(Data)
length(Data.M)
```

Basic classes of R objects – **Data Frame**

```
x <- c("a", "b")
y <- 1:5
z <- 1:6

x.Y <- data.frame(x,y)
```

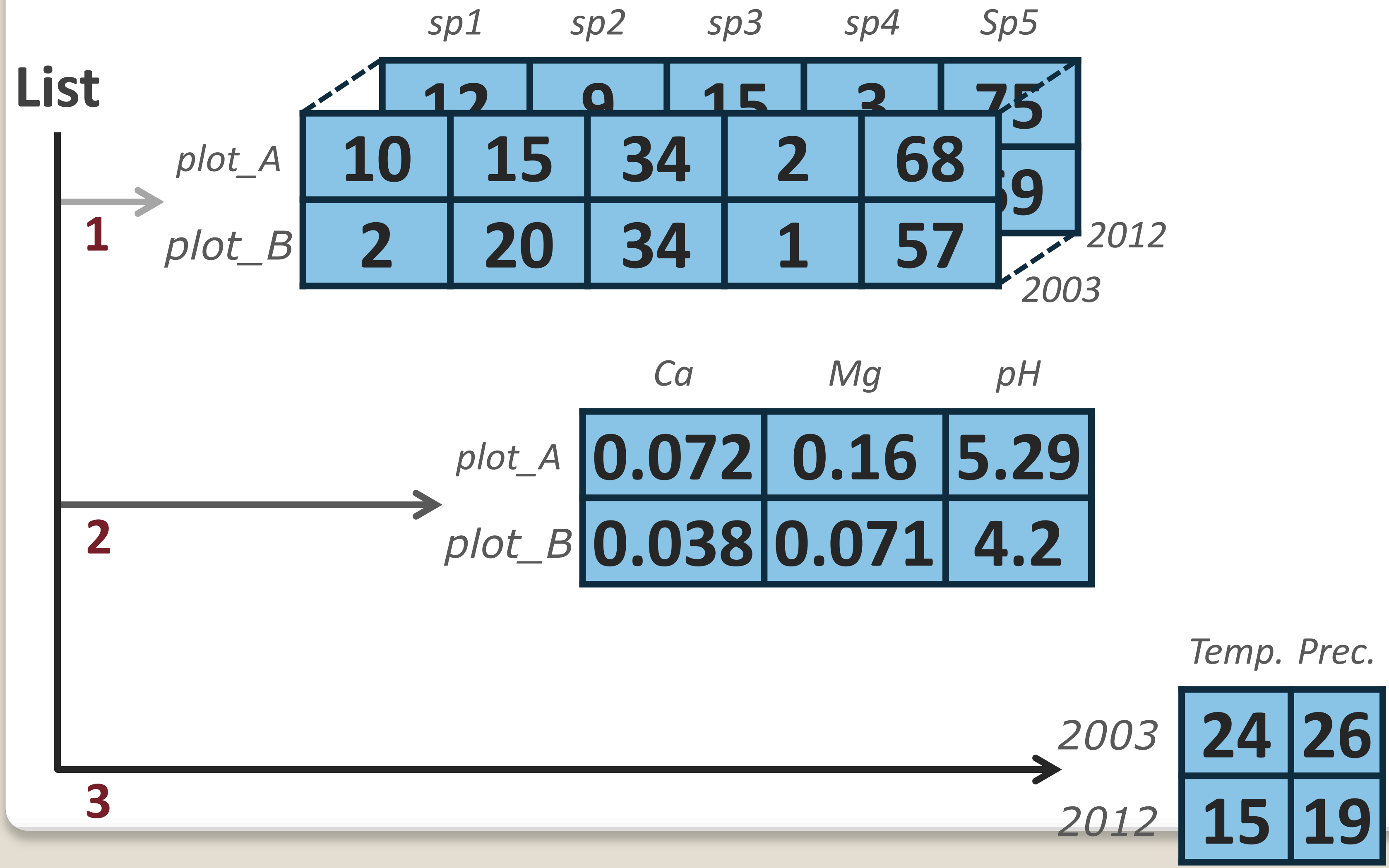
- Elements (columns) in a data frame must have the same length

```
length(x)
length(y)
length(z)

x.z <- data.frame(x,z)
x.z
```

Basic classes of R objects – R's **List**

Basic classes of R objects – **List**

- Lists can contain data of different types, dimensions and even classes



2012

	sp1	sp2	sp3	sp4	Sp5
plot_A	12	9	15	3	75
plot_B					9

2003

	sp1	sp2	sp3	sp4	Sp5
plot_A	10	15	34	2	68
plot_B	2	20	34	1	57

	Ca	Mg	pH
plot_A	0.072	0.16	5.29
plot_B	0.038	0.071	4.2

	Temp.	Prec.
2003	24	26
2012	15	19

Basic classes of R objects – **List**

```
Soils.plot <- matrix(c(0.072, 0.16, 5.29,
     0.038, 0.071, 4.2), byrow=TRUE,
     nrow=2)


Climate.year <- matrix(c(24, 26, 15, 19),
     byrow=TRUE, nrow=2)


ListData <- list(Abund, Soils.plot,
     Climate.year)

ListData
```

Basic classes of R objects – **List**

```
class(ListData)

dim(ListData)

length(ListData)

names(ListData)

names(ListData)<-c("Abund.", "Soils",
     "Climate")

ListData

str(ListData)
```

Other classes of R objects

- There is a large number of other R objects,

- Most rely on the same structure as vectors, matrices, data frames and lists. E.g.:

```
v1 <- rnorm(100, 10, 5)
v2 <- v1 + rnorm(100, 0, 2)

plot(v2~v1)

LM.v2v1 <- lm(v2~v1)

summary(LM.v2v1)

class(LM.v2v1)

str(LM.v2v1)
```

Exercise 1

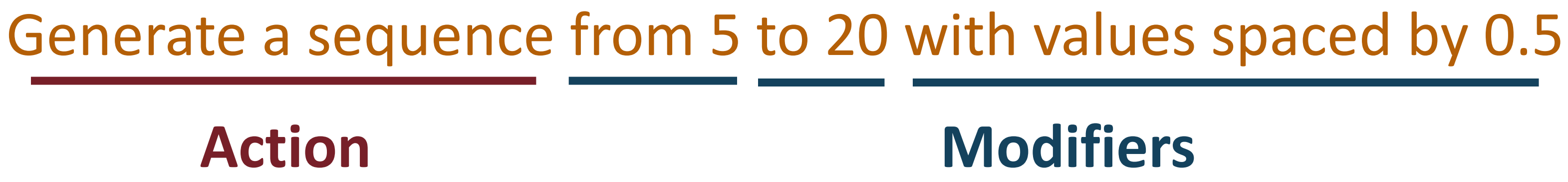
3. Functions and Arguments

Writing in R is like writing in English

Jump three times forward

Action **Modifiers**

Writing in R is like writing in English

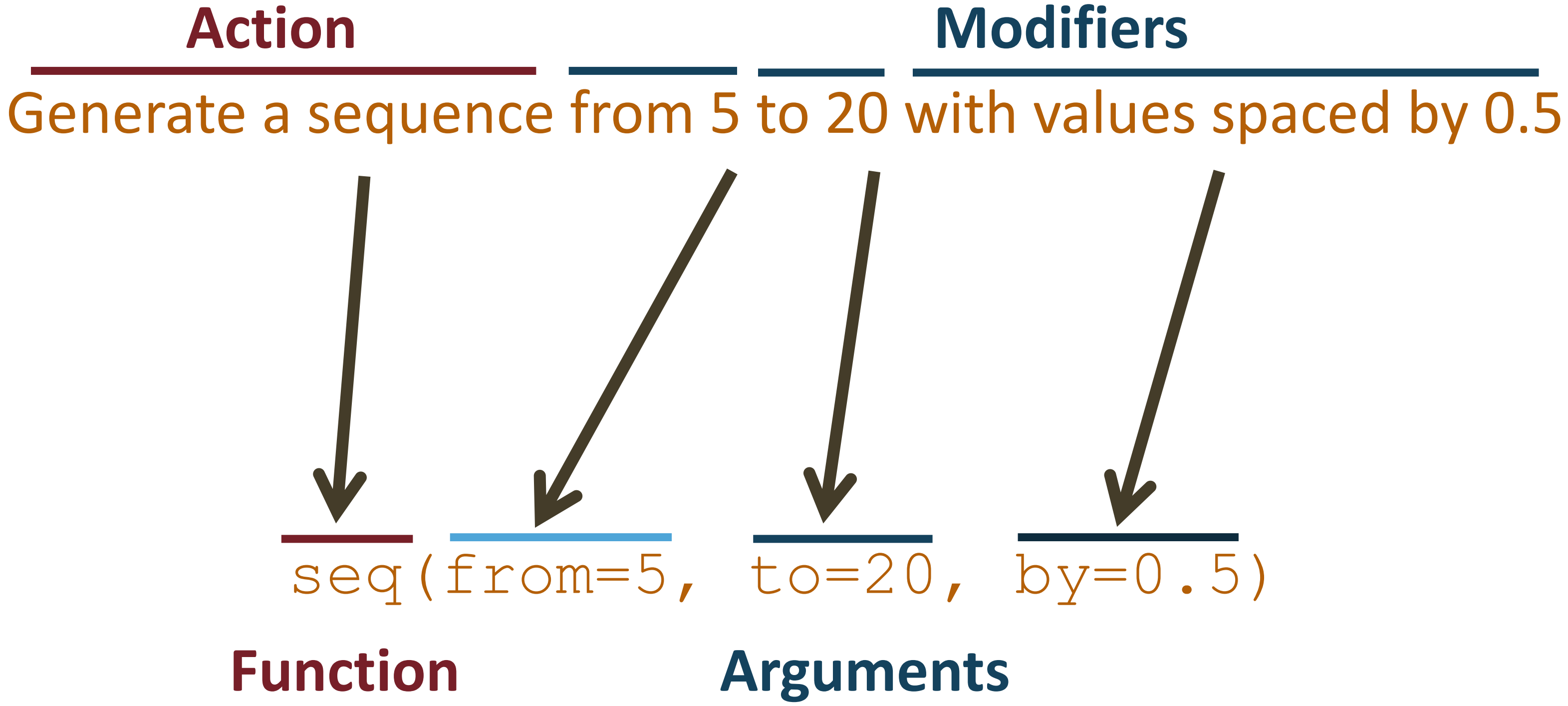


Generate a sequence from 5 to 20 with values spaced by 0.5

Action

Modifiers

Writing in R is like writing in English



Action

Modifiers

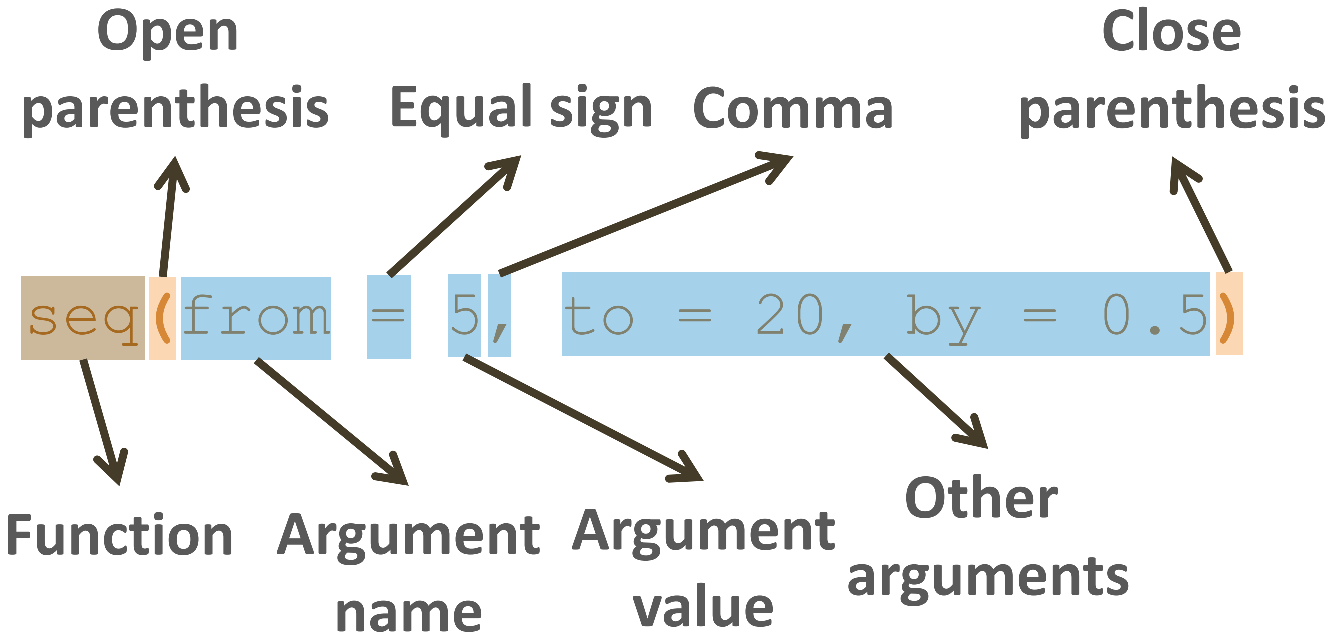
Generate a sequence from 5 to 20 with values spaced by 0.5

```
seq(from=5, to=20, by=0.5)
```

Function

Arguments

Basic anatomy of an R command



Open parenthesis

Equal sign

Comma

Close parenthesis

```
seq(from = 5, to = 20, by = 0.5)
```

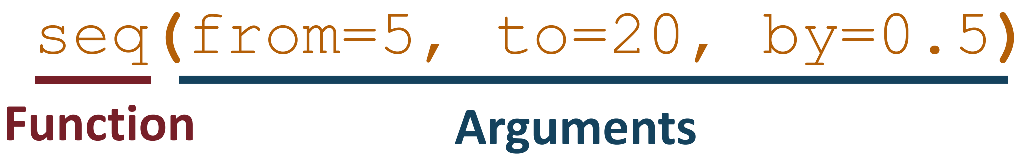
Function

Argument name

Argument value

Other arguments

Basic anatomy of an R command



```
seq(from=5, to=20, by=0.5)
```

Function

Arguments

- A function in R defines an action to take, and is similar to a verb in English

- Functions apply to arguments, which define on what and how a function will work

- Arguments are ***usually*** given within parenthesis after the function

Basic anatomy of an R command

1. Arguments almost always have names (e.g., "from ", "to", etc.)

```
    seq(from=5, to=20, by=0.5)
```

2. Names can be eliminated if arguments are given in predetermined order

```
    seq(5, 20, 0.5)

    seq(0.5, 5, 20)
```

3. Arguments can be reordered if you use names

```
    seq(by=0.5, to=20, from=5)
```

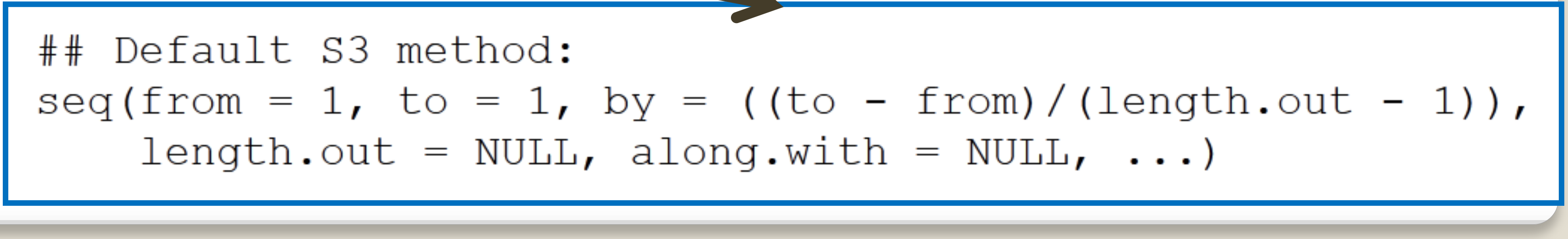
Basic anatomy of an R command

```
seq(from=5, to=20, by=0.5)

seq(to=10)
```

4. Frequently, functions have **arguments with predetermined values**

- Predetermined arguments do not need to be specified
- You can find predetermined values in the **help** page

```
?seq
```



```
## Default S3 method:
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)
```

Basic anatomy of an R command

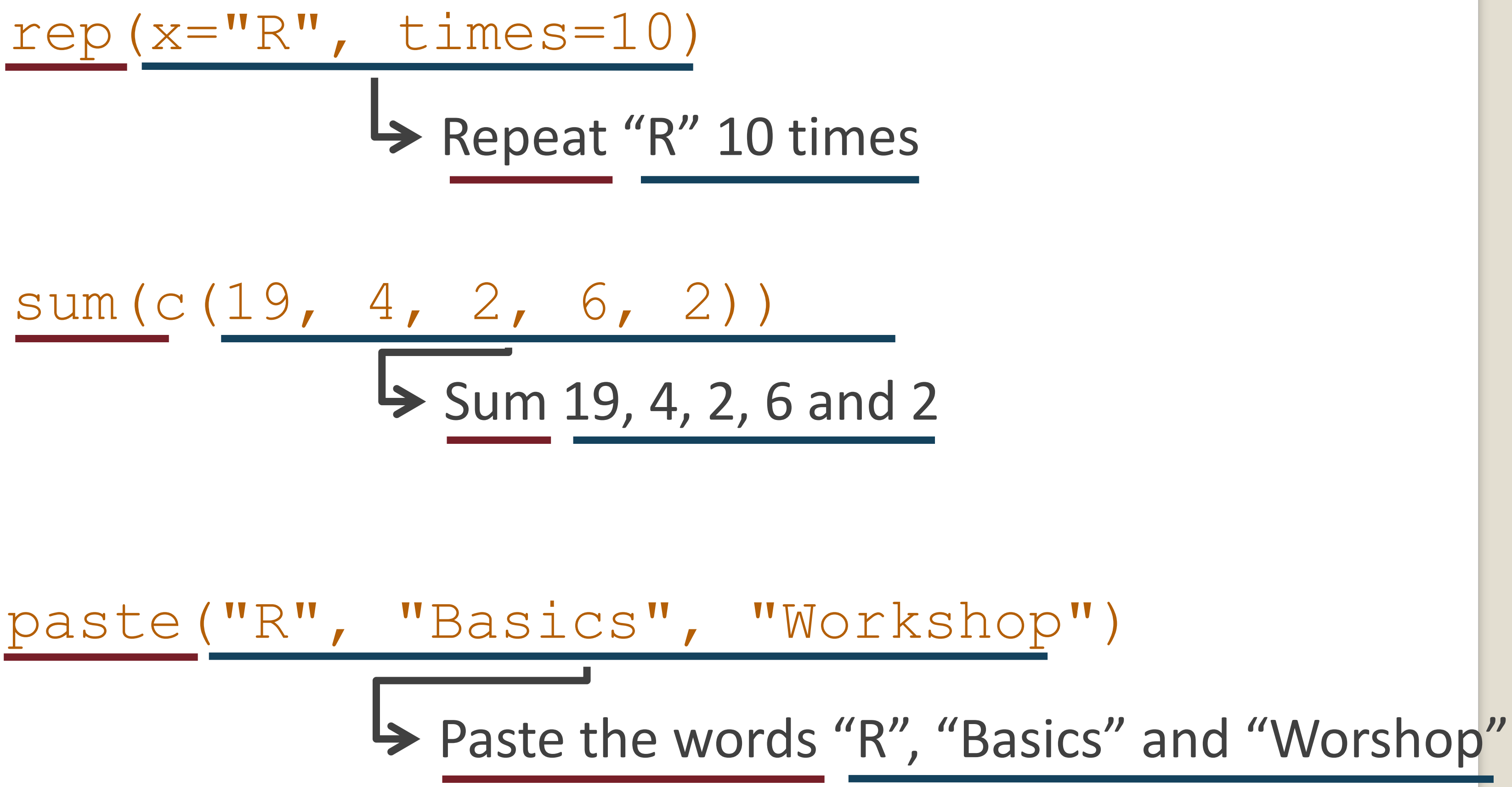
5. You can use functions to give values to an argument (**functions within functions**)

```
c(19, 4, 2, 6, 2)

mean(x=c(19, 4, 2, 6, 2))
```

```
rnorm(n=50, mean=0, sd=1)

rnorm(n=50, mean=3, sd=1)

boxplot(x=list(rnorm(n=50, mean=0,
sd=1), rnorm(n=50, mean=3, sd=1)))
```

Basic anatomy of an R command

- **Writing an R command is like writing a command in English**



```
rep(x="R", times=10)
```

Repeat “R” 10 times

```
sum(c(19, 4, 2, 6, 2))
```

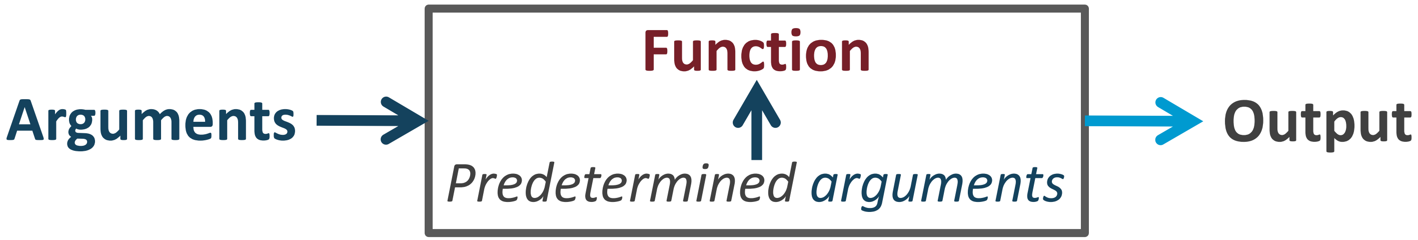
Sum 19, 4, 2, 6 and 2

```
paste("R", "Basics", "Workshop")
```

Paste the words “R”, “Basics” and “Worshop”

Summary: functions and arguments

```
seq(to=20, by=0.5)
```



Arguments

Function

Predetermined arguments

Output

Exercise 2

Functions and arguments

4. Opening/Saving Files

The Working Directory

- The working directory **is a folder in your computer** where R will search for files to open and where it will save file

- To know what the working directory is:

```
getwd()
```

- To modify the working directory:

```
setwd("C:/MyFiles/Are/InThisFolder")
```

- Also you can go to *File* and use the *Change dir…*option

Save and Open Data

- To save data:

  `write.table`

  `write.csv`

  `save`

- To read data:

  `read.table`

  `read.csv`

  `load`

Save Data Tables

- To save data tables :

  ```
  ?write.table
  ```

- Main arguments:
  - ***x***: is the R object that you want to save – usually a vector, matrix or data frame
  - ***file***: is the name and location of the file you want to create
  - ***sep***: defines the character that separates columns; frequently "**,**" or "**\t**"

Save Data Tables

```
M <- matrix(rnorm(100), ncol=5)

colnames(M) <- 1:ncol(M)

M

save.as <- "matrix_M.txt"

save.as <- "folder_test/matrix_M.txt"


write.table(x=M, file=save.as, sep="\t")
```

Open Data Tables

- To read data tables :

  `read.table`

- Main arguments:
  - ***file***: where the file is located and what its name is
  - ***header***: TRUE or FALSE, whether the first row are the names of the variables
  - ***sep***: defines the character that separates columns; frequently “**,**” o “**\t**”

Open Data Tables

```
Data <- read.table(file = file.choose(),
    header=TRUE, sep="\t")

Data <- read.table(file = "matrix_M.txt",
    header=TRUE, sep="\t")


class(Data)

names(Data)
```

Exercise 3

Opening/saving files

Thank you for attending!

linktr.ee/datasciencecube

✉ thedatasciencecube@gmail.com

@ds3.utsc

ds3utsc.com

Feedback Form

Data Science & Statistics Society