Intro to R for Biologists
Session 1
R basics

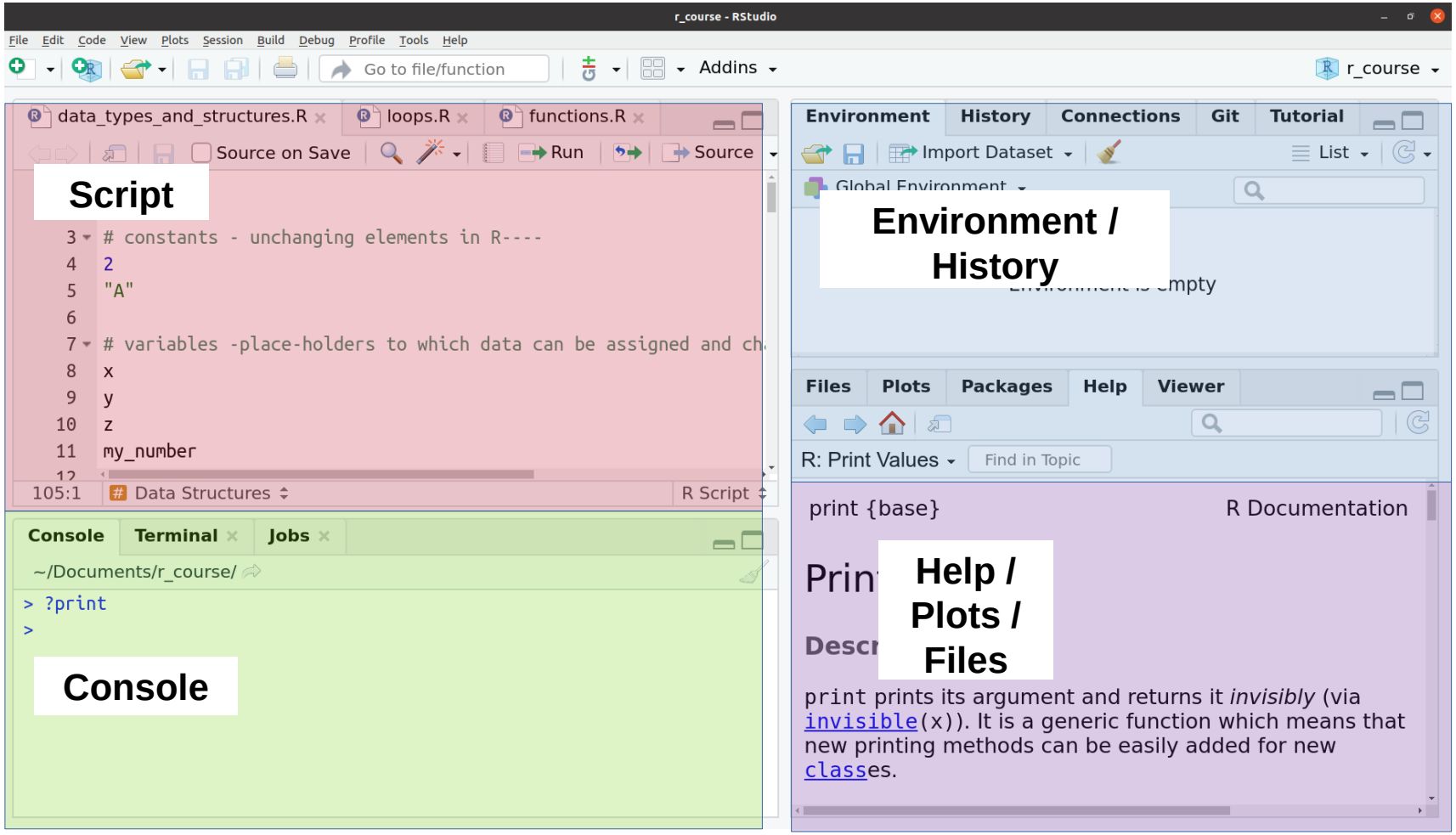
Irina & Rao

19/01/2021

Hilary 2021

INTRO TO R FOR BIOLOGISTS

- **Basic concepts**
  - Your R environment
  - Getting help
  - Assignment to variables
  - Data types
  - Data structures
  - Subsetting
  - Conditionals
  - Loops
  - Functions
  - Packages/libraries



SCRIPT CONSOLE HELP/PLOTS/FILES ENVIRONMENT/HISTORY

Let’s explore practically

Constants vs. variables

Constants are unchanging	Pi, Avogadro's number
Variables are placeholders	x = 10, y = 20; x + y = 30

How to get help

`?` and `??`

`help()`

Google!

Assignment of data to variables

The ‘arrow’: `x <- 2`

Equals sign: `x = 2`

Let’s explore practically

Data types in R

Character

- “a”, “apple”

Numeric

- 1, 23, 3.14

Logical

- TRUE, FALSE (also 1, 0 or T, F)

Factor

- For categorical variables, when data is classified into groups

Quotations

‘’ “” – denote character or string data

Conversion (and coercion)

- `as.character()`
  - **Numeric**, **logical** and **factor** types can be converted to character
- `as.numeric()`
  - **Logical**, and **character variables that are purely numbers** can be converted to numeric
- `as.logical()`
  - **"FALSE"** or **0** is converted to FALSE, **"TRUE"** or **any non-zero number** is converted to TRUE
- `as.factor()`
  - Any other data type can be converted to a factor

Warnings

```
Warning message:
NAs introduced by coercion
```

Silent effects

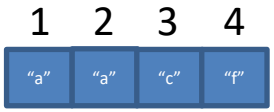
```
> as.numeric(as.character(14))
[1] 14
> as.numeric(as.factor(14))
[1] 1
```

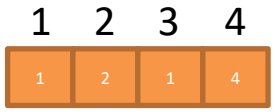
Let’s explore practically

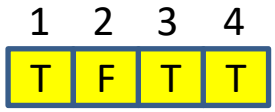
Data Structures: atomic vector

- atomic vector
  - All elements are of the same type

atomic vector



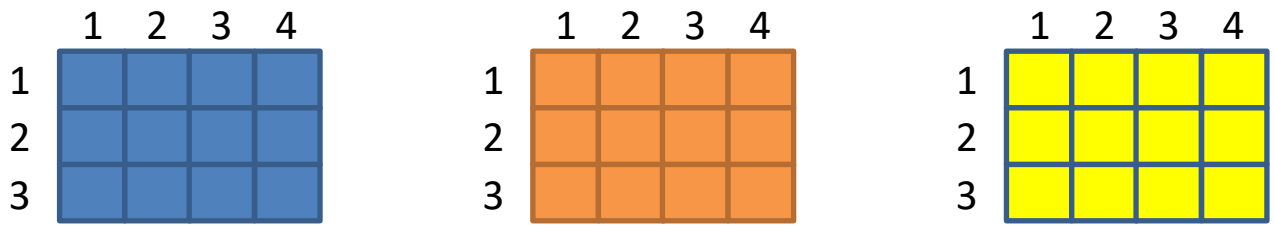




Data Structures: matrix

- matrix
  - All elements are of the same type, data organised in rows and columns

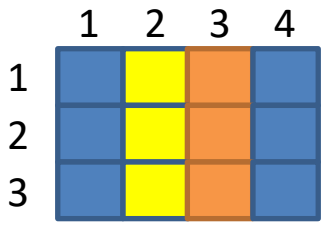
matrix



Data Structures: data.frame

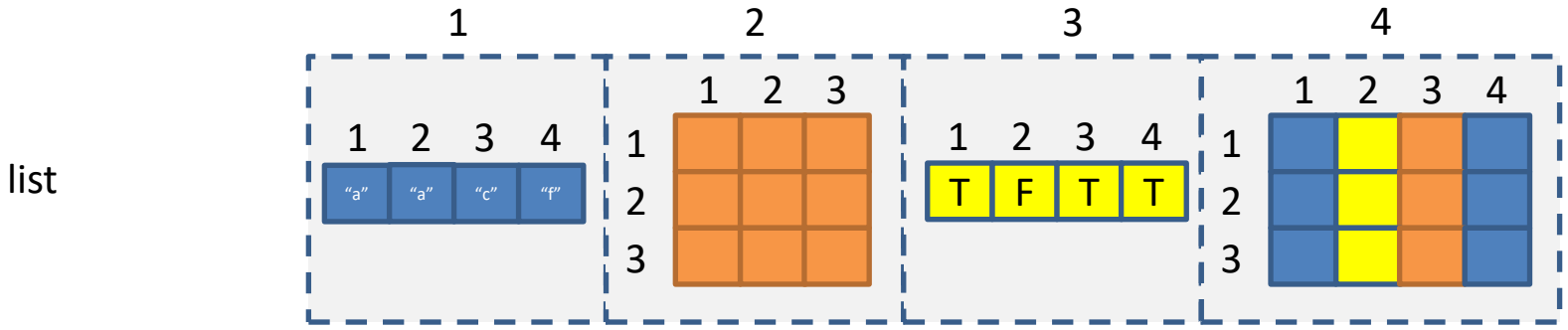
- data.frame
  - Data organised into rows and columns; elements of each column are of the same type, but different columns can be of different types

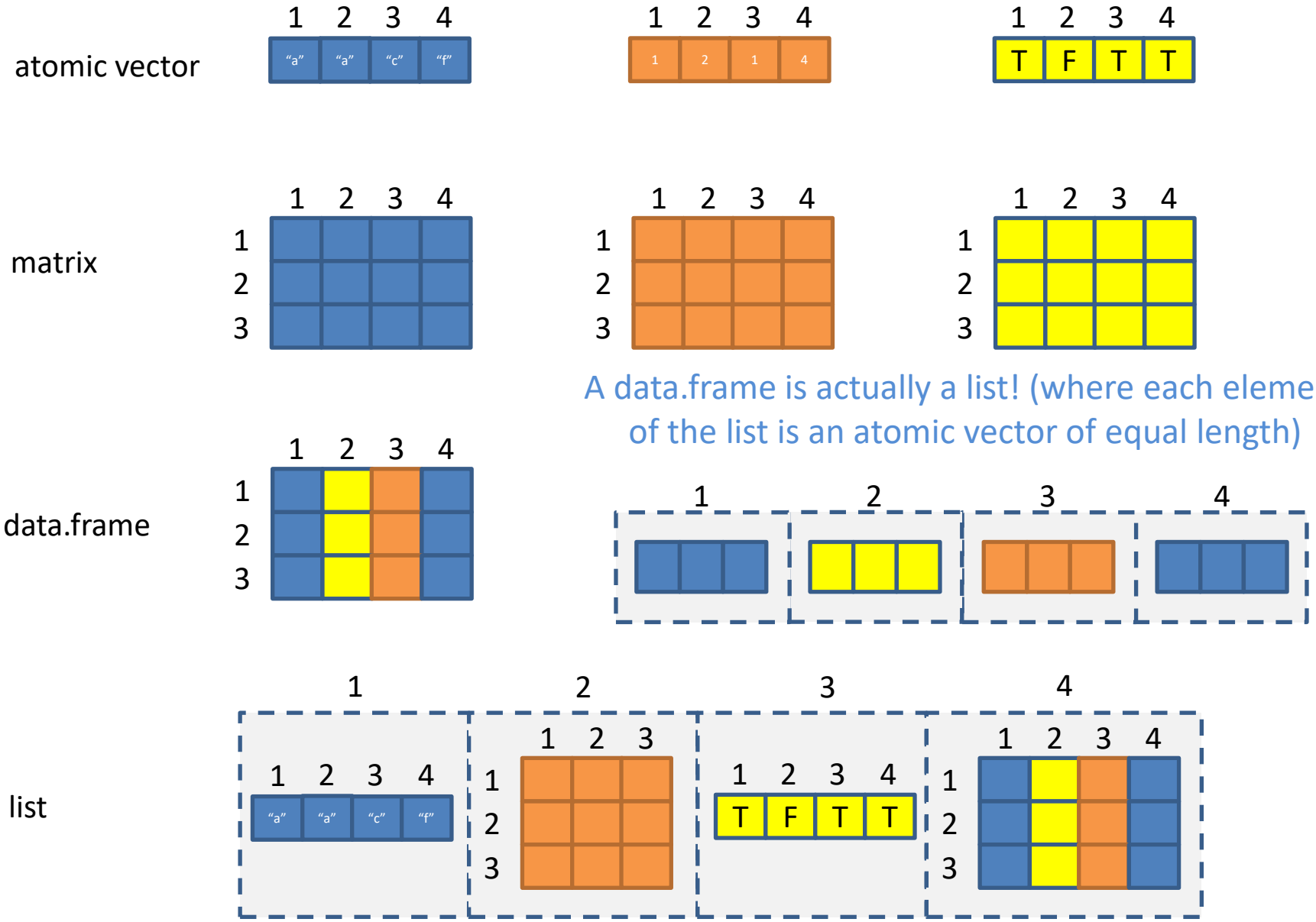
data.frame



Data Structures: list

- list
  - Elements can be different types, elements can also be other data structures



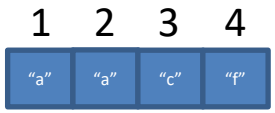

atomic vector
1 2 3 4
"a" "a" "c" "f"
1 2 3 4
1 2 1 4
1 2 3 4
T F T T
matrix
1 2 3 4
1
2
3
A data.frame is actually a list! (where each element of the list is an atomic vector of equal length)
data.frame
list

Subsetting R objects

- [ ] – To get a subset of elements
- [[ ]] – To get a specific element from a list (or a column of a data.frame)
- $ - To subset a specific column in a data.frame

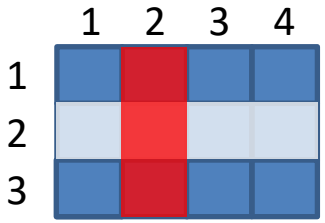
Subsetting R objects

My_av



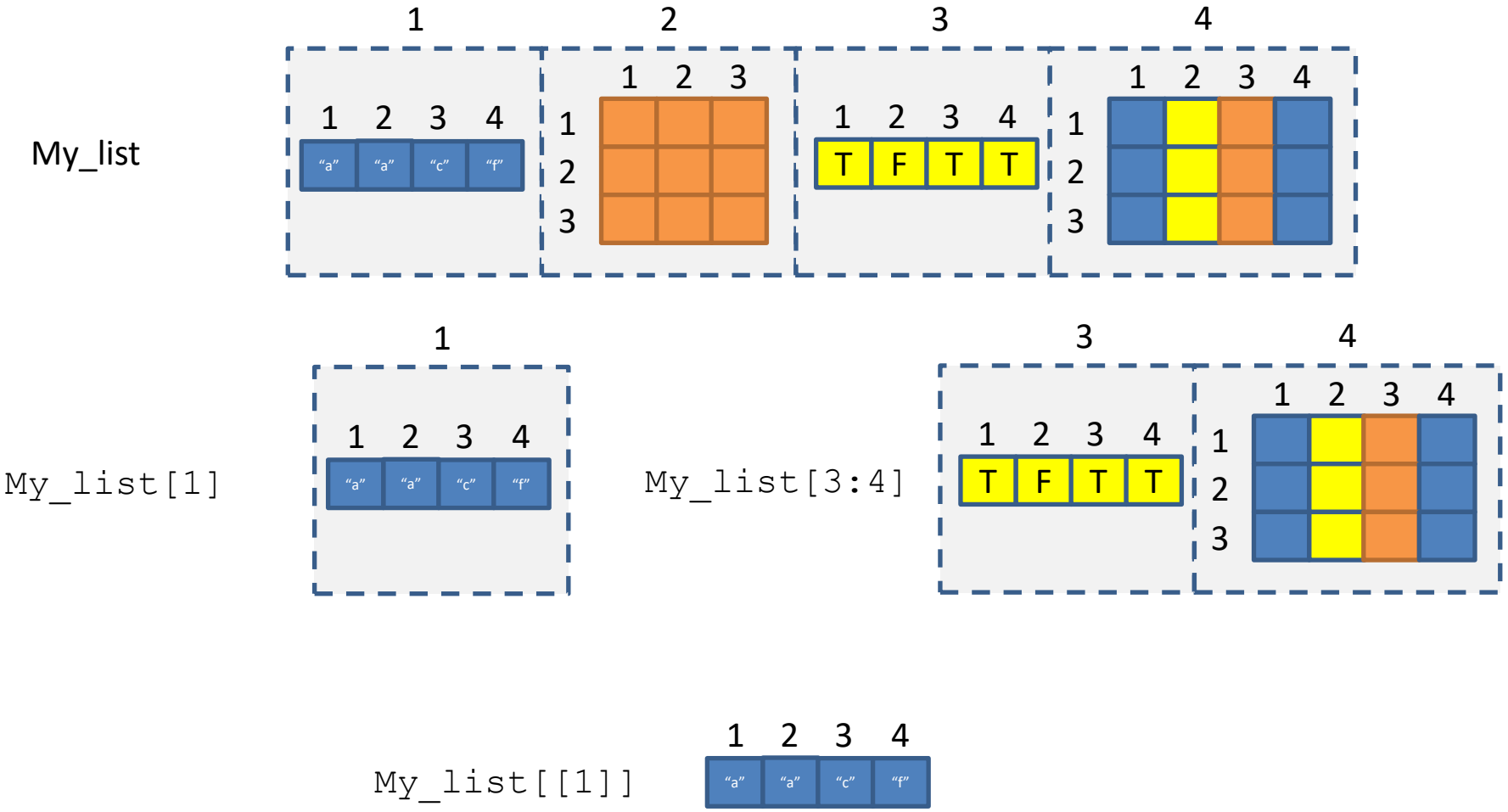

```
My_av[2]
My_av[c(F, T, F, F)]
```

My_mat

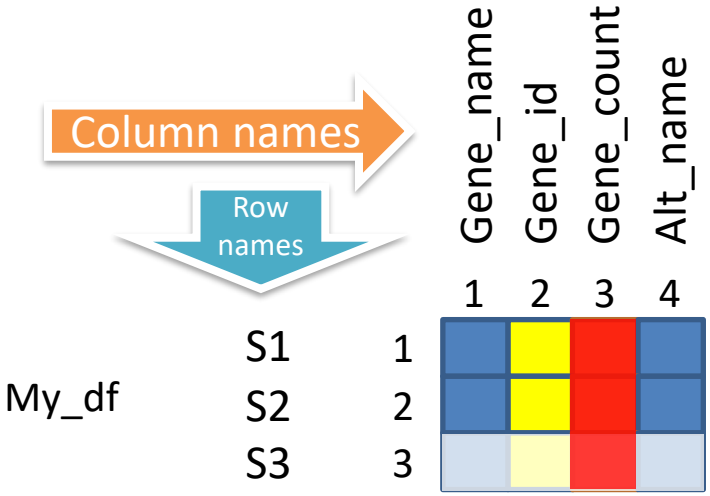



```
My_mat[2, ]
My_mat[ , 2]
```

Subsetting R objects: list



Subsetting R objects: data.frame




```
My_df[3, ]
My_df[“S3”, ]

My_df[ , 3]
My_df[ , “Gene_count”]
My_df$S3
My_df[[“Gene_count”]]
```

Let’s explore practically

Math operations

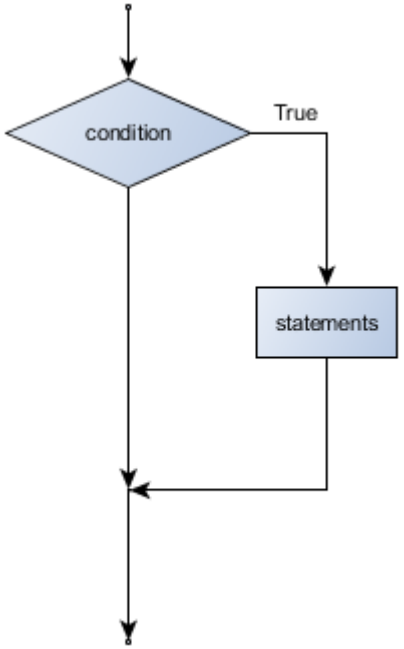
- Add, subtract, divide, multiply
  - `+ - / *`
- Modulo (remainder of division)
  - `%%`
- Exponent
  - `^`
- Rounding
  - `round(),floor(),ceiling()`

Comparisons

- Comparison between two objects returns TRUE or FALSE or NA
  - == (Note the difference between this and = for assignment)
  - <
  - >
  - <=
  - >=
  - != (Negation of anything in R is done with !)

Conditionals
if statement

```
if(TRUE or FALSE) {
   Do something
}
```

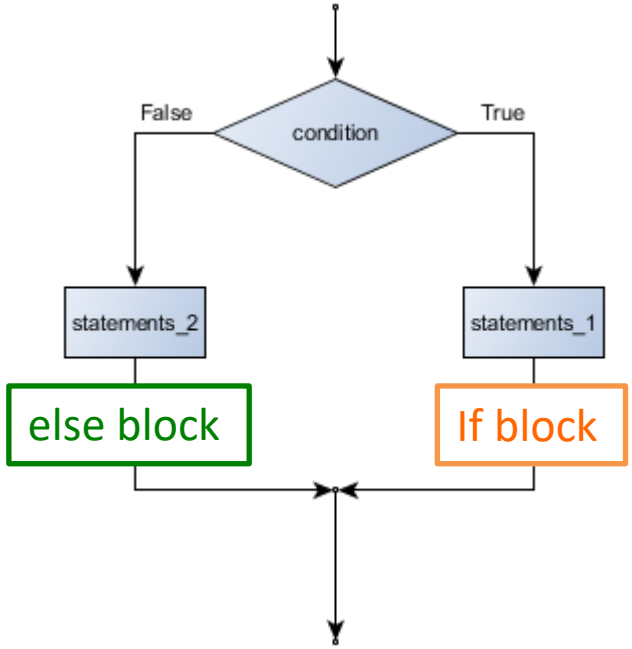



```
x <- 4
if(x > 0) {
print("Positive
number")
}

x <- -4
```

Conditionals
if … else statement

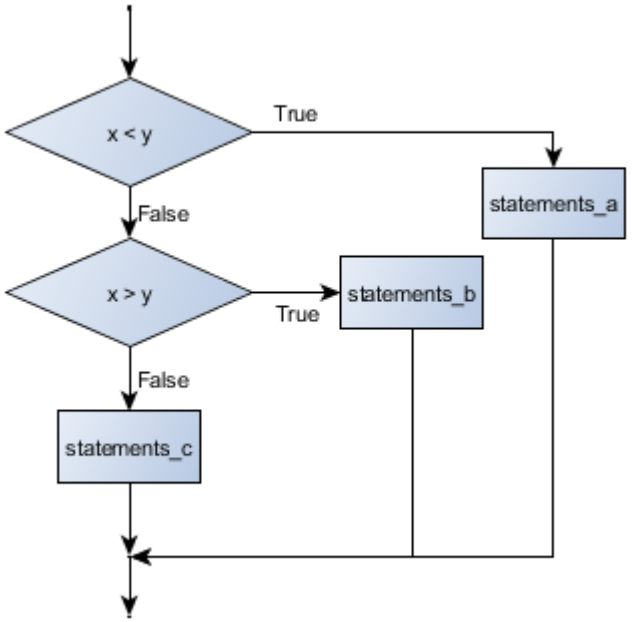
```
if(TRUE or FALSE) {
   Do something
} else {
   Do some other thing
}
```




```
x <- 4
if(x > 0) {
print("Positive number")
} else {
print(”Not positive number")
}
x <- -4
```

Chained conditionals
if … else if … else

```
if (x < y) {
    STATEMENTS_A
} else if (x > y) {
    STATEMENTS_B
} else {
    STATEMENTS_C
}
```



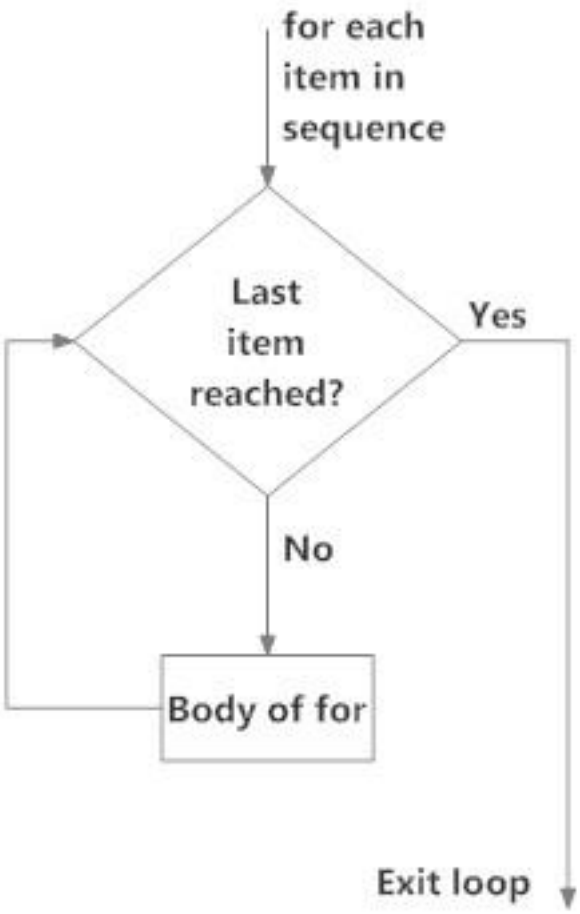

```
x <- 0
if (x < 0) {
print("Negative number")
} else if (x > 0) {
print("Positive number")
} else {
print("Zero")
}
```

Let’s explore practically

Loops
for

```
for (val in sequence)
{
statement
}
```

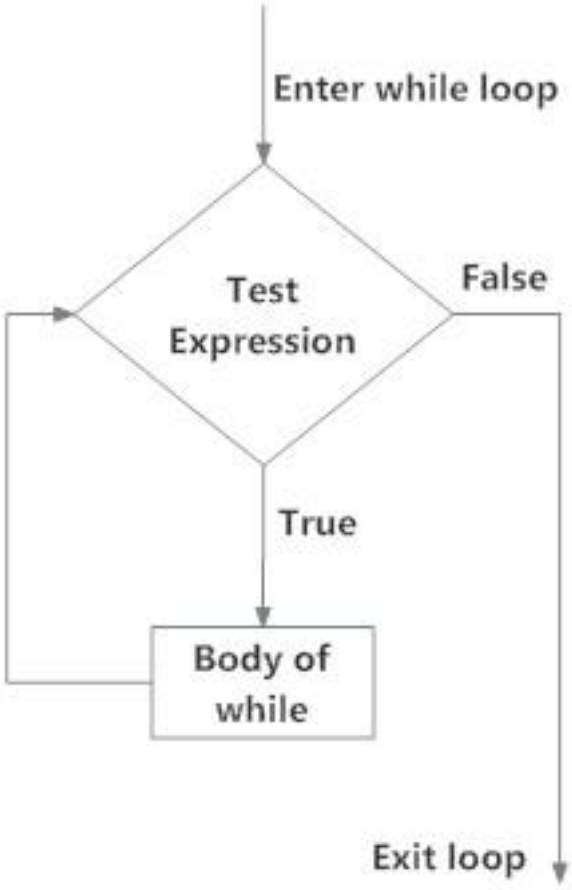
```
for (year in
c(2011,2012,2013,2014,2015,2016))
{
  print(paste("The year is", year))
}
```



Loops
while

```
while (test_expression)
{
statement
}
```

```
i <- 1
while (i < 6) {
print(i)
i = i+1
}
```

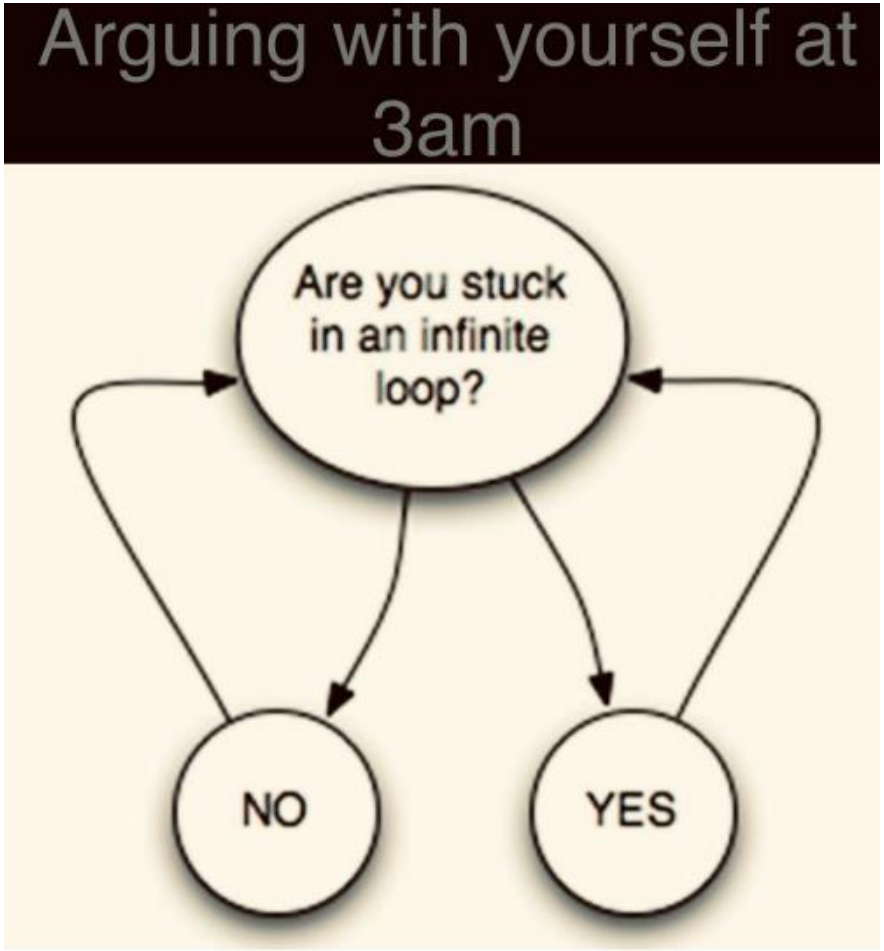


Infinite loops

```
x <- 1
while (x=1)
{
Print("Let's move on!")
}
```

Something to keep in mind!

*`break` and `next` statements might help – advanced level



Let’s explore practically

Functions

Functions are created using the function() directive and are stored as R objects just like anything else. In particular, they are R objects of class "function".

```
f <- function(<arguments>) {
        ## Do something interesting
}
```

Note: majority of the functions you would try to create exist in the libraries!

Packages/libraries

A **package** is a collection of R functions, data and compiled code. The location where the packages are stored is called the **library**.

Package extends basic R functionality and standardizes the distribution of code. For example, a package can contain a set of functions relating to a specific topic or tasks.

Where to get the libraries/packages?

- CRAN (https://cran.r-project.org)

Install:

```
install.packages("dplyr")
```

Activate:

```
Library("dplyr")
```

- Bioconductor (https://bioconductor.org)

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

- Github (https://github.com)

```
install.packages("devtools")
library(devtools)
install_github("hadley/dplyr”)
```

Our Github page

https://github.com/sraorao/MSD_R_course

Will become public soon for all of you!

Let’s explore practically