Graduate Student Lunch & Learn

Rafael Della Coletta

02.17.2022

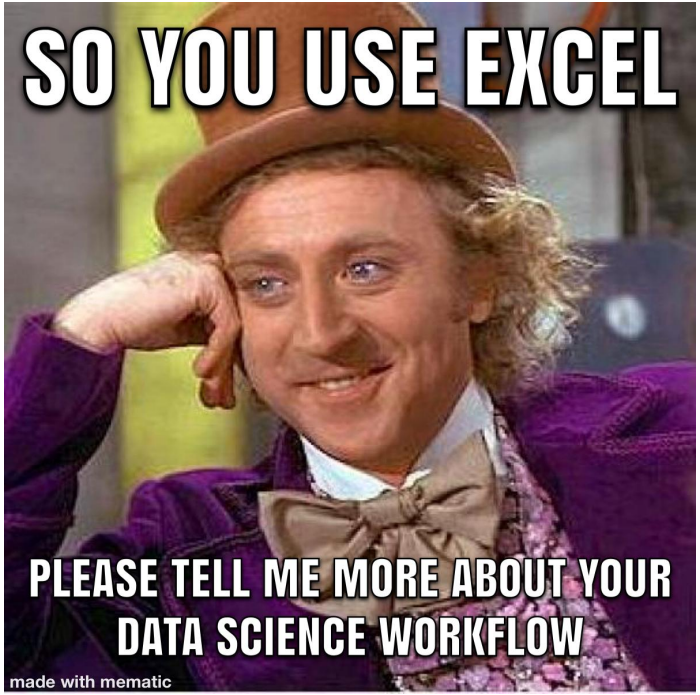
What I wish I knew when started grad school…

(with memes)

1 Excel sucks

If statistics programs/languages were cars...


EXCEL
INTERGER
IS THIS A DATE?
imgflip.com


SO YOU USE EXCEL
PLEASE TELL ME MORE ABOUT YOUR DATA SCIENCE WORKFLOW
made with mematic

2 Learning to code is hard, but worth it

Matt Dancho (Business Scie... · 9/27/21

Even seasoned useRs get smacked sometimes. Never give up. #rstats

3 Programming is 90% googling

Bonnie @The_GreatBonnie · 1/23/22

Are you a developer or you can't relate.

4 Best code is working code

Dorsa Amir
@DorsaAmir

When your code is a mess but it somehow still works.

R basics: syntax, data structures, loops, etc.

(no memes)

R ≠ RStudio®

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | R 4.1.2

basics.R

Run | Source

```r
# using your first function
hello <- "yet another hello world"
print(hello)

# modifying variables
genes_A <- 100
genes_B <- 50
total_genes <- genes_A + genes_B
print(total_genes)
```

10:1 (Top Level) R Script

Console Terminal Jobs

R 4.1.2 · /cloud/project/

```
> # using your first function
> hello <- "yet another hello world"
> print(hello)
[1] "yet another hello world"
>
> # modifying variables
> genes_A <- 100
> genes_B <- 50
> total_genes <- genes_A + genes_B
> print(total_genes)
[1] 150
>
```

Environment History Connections Tutorial

Import Dataset | 162 MiB | List

R | Global Environment

Values

genes_A	100
genes_B	50
hello	"yet another hello world"
total_genes	150

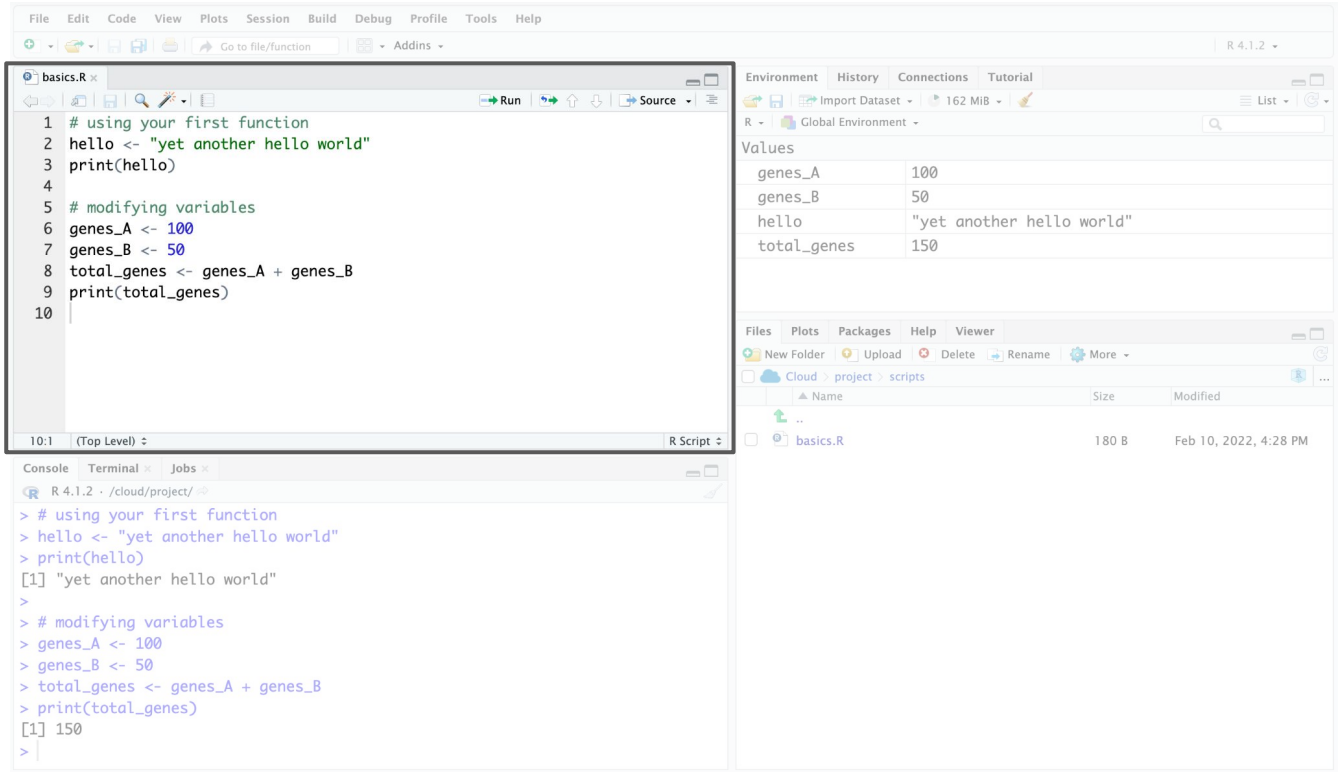
Files Plots Packages Help Viewer

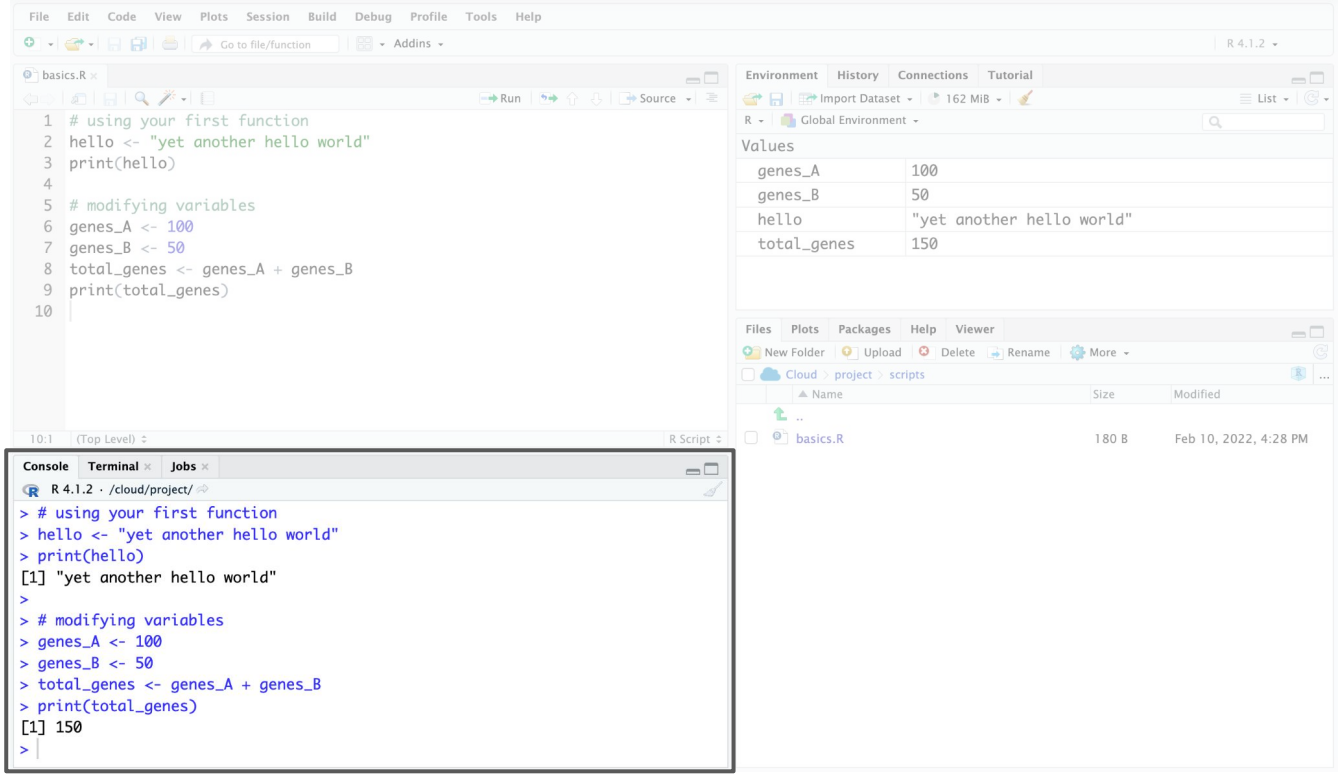
New Folder | Upload | Delete | Rename | More

Cloud > project > scripts

Name	Size	Modified
..		
basics.R	180 B	Feb 10, 2022, 4:28 PM

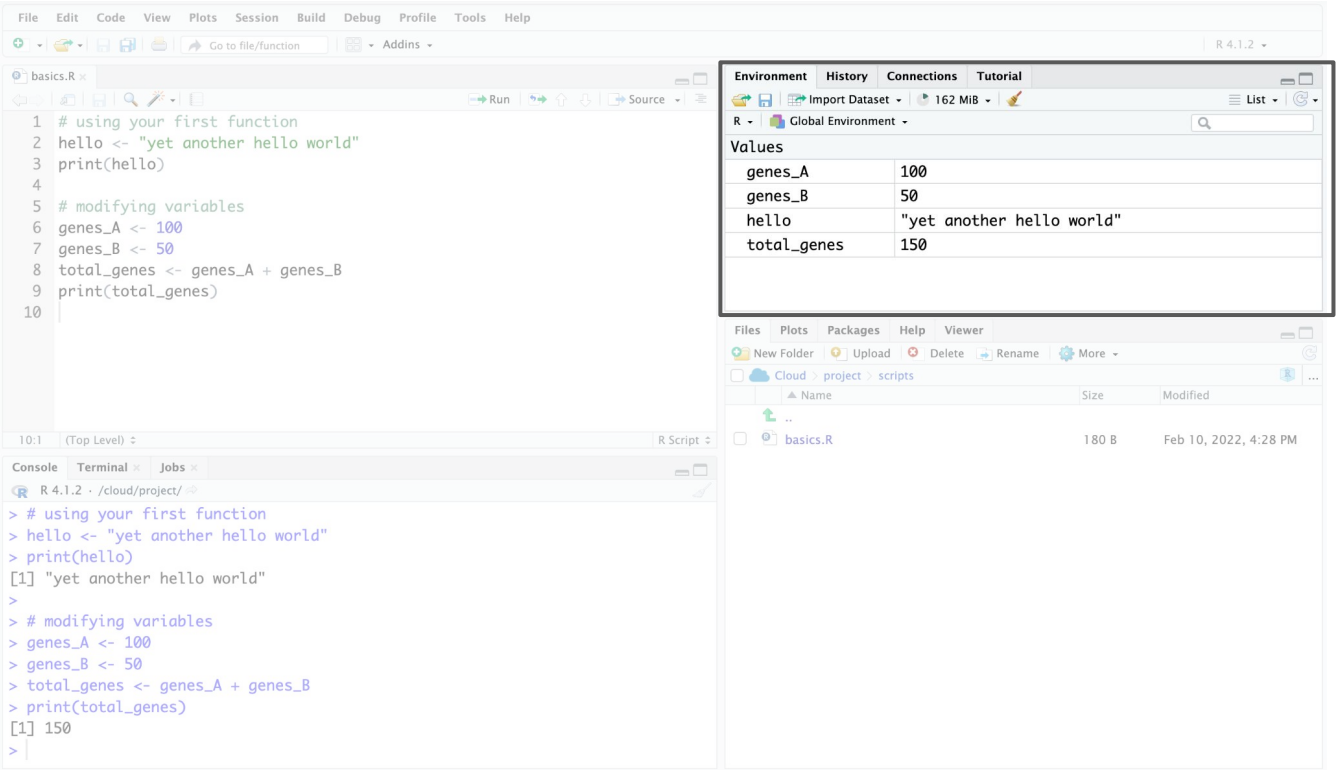
Write your code here





R does its things here

Keep track of all your objects here



File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | R 4.1.2

basics.R

Run | Source

```
# using your first function
hello <- "yet another hello world"
print(hello)

# modifying variables
genes_A <- 100
genes_B <- 50
total_genes <- genes_A + genes_B
print(total_genes)
```

10:1 (Top Level) R Script

Console Terminal Jobs

R 4.1.2 · /cloud/project/

```
> # using your first function
> hello <- "yet another hello world"
> print(hello)
[1] "yet another hello world"
>
> # modifying variables
> genes_A <- 100
> genes_B <- 50
> total_genes <- genes_A + genes_B
> print(total_genes)
[1] 150
>
```

Environment History Connections Tutorial

Import Dataset | 162 MiB | List

R | Global Environment

Values

genes_A	100
genes_B	50
hello	"yet another hello world"
total_genes	150

Files Plots Packages Help Viewer

New Folder | Upload | Delete | Rename | More

Cloud > project > scripts

Name	Size	Modified
..		
basics.R	180 B	Feb 10, 2022, 4:28 PM

See where your files are

Basic syntax

```
# using your first function
hello <- "yet another hello world"
print(hello)
```

Basic syntax

Comment (aka your best friend)

```
# using your first function
hello <- "yet another hello world"
print(hello)
```

Basic syntax

Assignment operator

```
# using your first function
hello <- "yet another hello world"
print(hello)
```

Object / Variable

Some data:

- **Character / string** → "hello world!"
- **Integer** → 30
- **Numeric** → 2.5
- **Logical** → TRUE or FALSE

Basic syntax

```
# using your first function
hello <- "yet another hello world"
print(hello)
```

Function

Style guide

https://style.tidyverse.org/

Best practices on how to name variables and write clean code!

Variable and function names should use only lowercase letters, numbers, and `_`. Use underscores (`_`) (so called snake case) to separate words within a name.

```
# Good
day_one
day_1

# Bad
DayOne
dayone
```

Generally, variable names should be nouns and function names should be verbs. Strive for names that are concise and meaningful (this is not easy!).

```
# Good
day_one

# Bad
first_day_of_the_month
djm1
```

Operators

Arithmetic

+	addition
-	subtraction
*****	multiplication
/	division
^ or ******	exponentiation
x **%%** y	integer-divide x by y and return the remainder
x **%/%** y	integer division

Operators

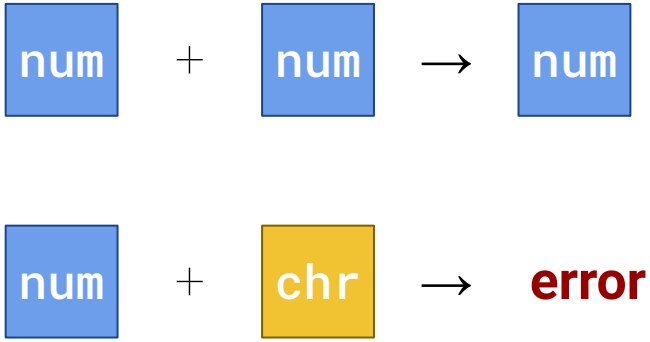
Logical

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	**exactly equal to**
!=	not equal to
! x	not x
x **|** y	x or y
x **&** y	x and y

Operations

Operations

Be careful! Certain operations are **specific** to certain data types



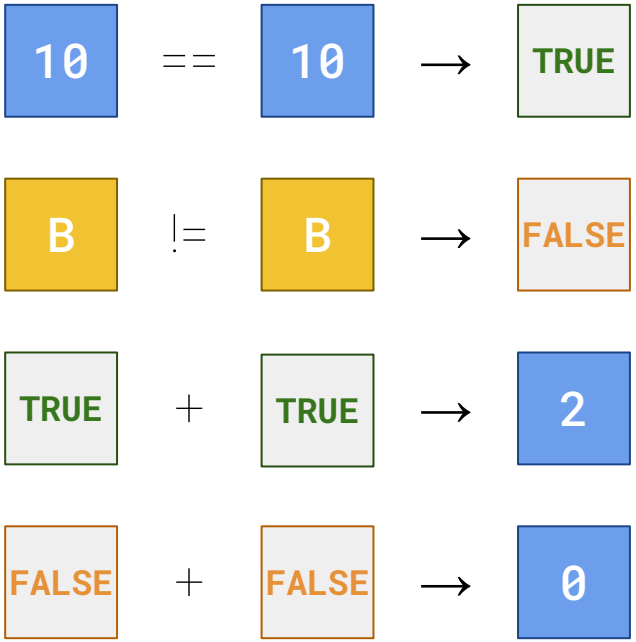
Operations

```
class( 10 ) → numeric

class( 10 ) → character

as.numeric( 10 ) → 10
```

Operations

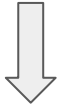


Functions

Functions

```
print(my_variable)
```


```
?round
```



Rounds the values in its first argument to the specified number of decimal places (default 0)

```
round(my_variable)
```

Functions

```
my_data <- read.file(file = "filename.txt", sep = "\t")
```

Function name

Arguments

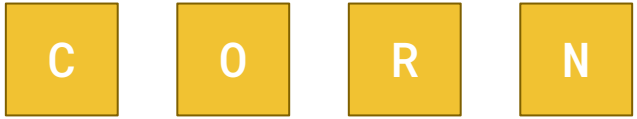
*type **?read.file** for more arguments and their default values

Data structures

Data structures

```
my_vector <- c(1, 55, 2.5, 333)
```


```
my_vector <- c("C", "O", "R", "N")
```



Data structures

```
my_matrix <- matrix(
```

1	55	2.5	333
0.2	0	11	2
0	9.9	2.5	758
1	5	2	90

```
)
```

Data structures

```
my_dataframe <- data.frame(
```

1	C	TRUE	333
0.2	O	TRUE	2
0	R	TRUE	758
1	N	FALSE	90

```
)
```

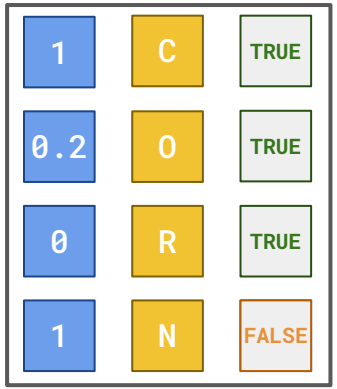
Data structures

```
my_list <- list(
```

numbers =

best_plant =

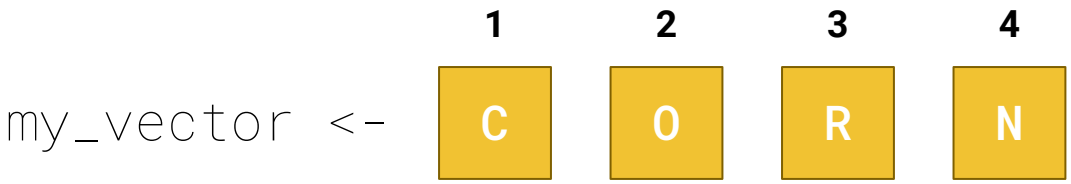
data =



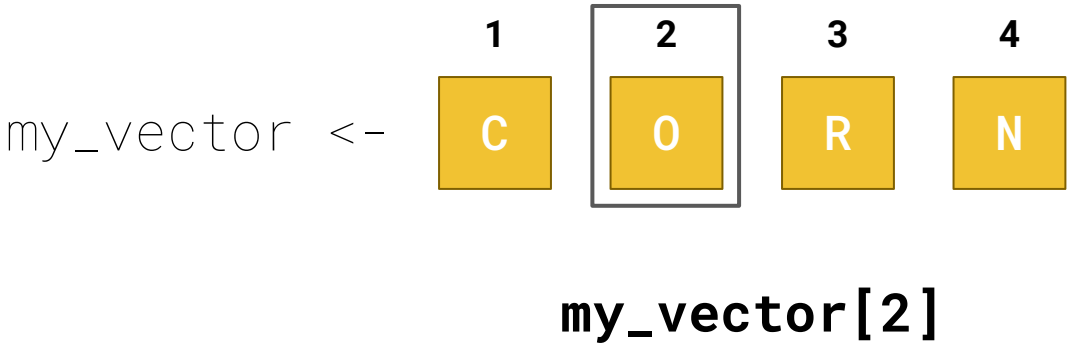

```
)
```

Subsetting

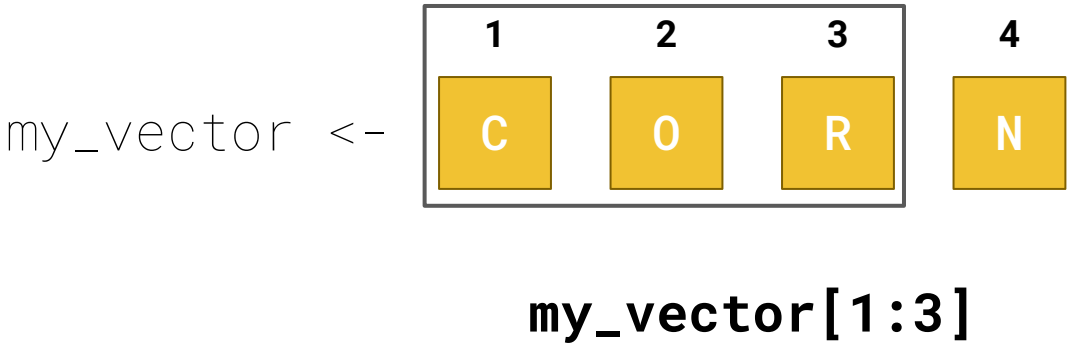
Subsetting



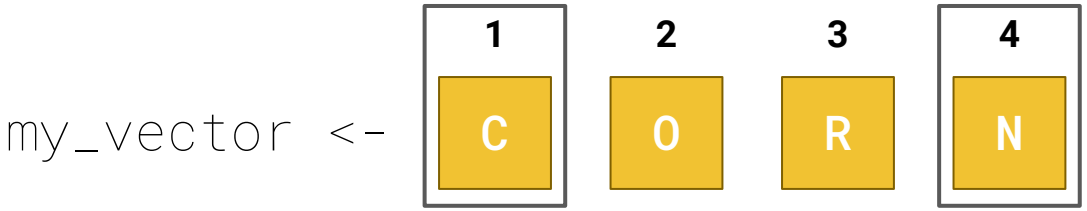
Subsetting



Subsetting



Subsetting




```
my_vector[c(1,4)]
```

Subsetting

```
my_df <-
```

	1	2	3	4
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

Subsetting

my_df <-

	1	2	3	4
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
my_df[ , ]
```

Subsetting

my_df <-

	1	2	3	4
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

my_df[2,3]

Subsetting

my_df <-

	1	2	3	4
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
my_df[4, ]
```

Subsetting

my_df <-

	1	2	3	4
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
my_df[ ,c(1,3)]
```

Subsetting

my_df <-

	1	2	3	4
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
my_df[my_df[, 1] < 0.5, c(1,3)]
```

*also check **subset()** and **grep()** functions

Subsetting

```
df_with_header <-
```

	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

Subsetting

```
df_with_header <-
```

	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
df_with_header$plant
```

Subsetting

```
df_with_header <-
```

	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
df_with_header[ ,2]
```

Subsetting

```
df_with_header <-
```

	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```
df_with_header[3:4, c("maf","n")]
```

if / else statements

```
if (condition) {
    …do something…
} else {
    …do other thing…
}
```

if / else statements

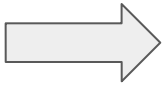
```
big_number <- 1.000

if (big_number > 100) {
  print("this is a big number")
} else {
  print("not that big")
}
```

if / else statements

```
big_number <- 1.000

if (big_number > 100) {
  print("this is a big number")
} else {
  print("not that big")
}
```



“not that big”

if / else statements

```r
plant <- "corn"

if (plant == "corn") {
  plant <- paste(plant, "is the best")
} else {
  plant <- "not corn"
}

print(plant)
```

if / else statements

```r
plant <- "corn"

if (plant == "corn") {
  plant <- paste(plant, "is the best")
} else {
  plant <- "not corn"
}

print(plant)
```

→ **“corn is the best”**

for loop

```
for (variable in vector) {
    …do something…
}
```

I want to create a vector containing **only vowels** from the plant column

	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

df$plant

C O R N

new_vector

O

Need to tell R some important things first...

```
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

Need to tell R some important things first...

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

... and then do what you gotta do

```r
# get letter from first row of column "plant"
letter <- df[1, "plant"]
```

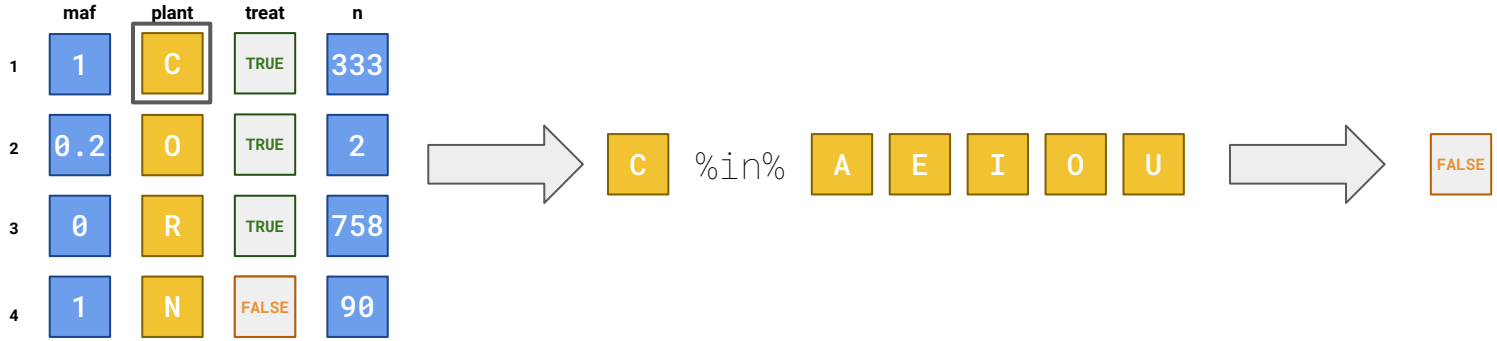
	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

Need to tell R some important things first...

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

... and then do what you gotta do

```r
# get letter from first row of column "plant"
letter <- df[1, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```



Need to tell R some important things first...

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

... and then do what you gotta do

```r
# get letter from first row of column "plant"
letter <- df[2, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```

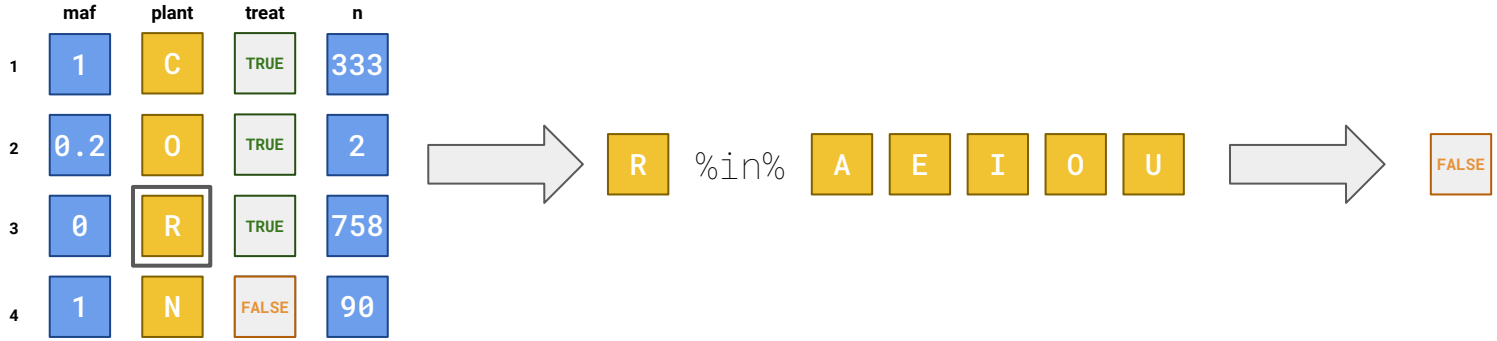
	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

Need to tell R some important things first...

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

... and then do what you gotta do

```r
# get letter from first row of column "plant"
letter <- df[3, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```



Need to tell R some important things first...

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

... and then do what you gotta do

```r
# get letter from first row of column "plant"
letter <- df[4, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```

	maf	plant	treat	n
1	1	C	TRUE	333
2	0.2	O	TRUE	2
3	0	R	TRUE	758
4	1	N	FALSE	90

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()

# get letter from first row of column "plant"
letter <- df[1, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[2, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[3, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[4, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

```r
# get letter from first row of column "plant"
letter <- df[1, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[2, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[3, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[4, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()
```

```r
for (row in 1:4) {
  # get letter from column "plant"
  letter <- df[row, "plant"]
  # if letter is a vowel
  if (letter %in% vowels) {
    # append letter to vector
    vowels_in_plant <- append(vowels_in_plant, letter)
  }
}
```

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()

# get letter from first row of column "plant"
letter <- df[1, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[2, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[3, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}

# get letter from first row of column "plant"
letter <- df[4, "plant"]
# if letter is a vowel
if (letter %in% vowels) {
  # append letter to vector
  vowels_in_plant <- append(vowels_in_plant, letter)
}
```

```r
# vowels I know
vowels <- c("A", "E", "I", "O", "U")
# create an empty vector to store values
vowels_in_plant <- c()

for (row in 1:4) {
  # get letter from column "plant"
  letter <- df[row, "plant"]
  # if letter is a vowel
  if (letter %in% vowels) {
    # append letter to vector
    vowels_in_plant <- append(vowels_in_plant, letter)
  }
}
```

while loop

```
while (condition) {
    …do something…
}
```

```r
# initial number of seeds
total_seeds <- 0
# initial number of surprise packages opened
surprise_packages <- 0

while(total_seeds < 1000) {

  # open surprise packages and add to total number of seeds
  total_seeds <- total_seeds + sample(x = 1:100, size = 1)
  # increase number of surprise packages opened
  surprise_packages <- surprise_packages + 1

}

# let me know how many packages I opened
print(surprise_packages)
```

```r
# initial number of seeds
total_seeds <- 0
# initial number of surprise packages opened
surprise_packages <- 0

while(total_seeds < 1000) {

  # open surprise packages and add to total number of seeds
  total_seeds <- total_seeds + sample(x = 1:100, size = 1)
  # increase number of surprise packages opened
  surprise_packages <- surprise_packages + 1

}

# let me know how many packages I opened
print(surprise_packages)
```

```r
# initial number of seeds
total_seeds <- 0
# initial number of surprise packages opened
surprise_packages <- 0

while(total_seeds < 1000) {

  # open surprise packages and add to total number of seeds
  total_seeds <- total_seeds + sample(x = 1:100, size = 1)
  # increase number of surprise packages opened
  surprise_packages <- surprise_packages + 1

}

# let me know how many packages I opened
print(surprise_packages)
```

First time →**19**

Second time → **17**

Third time → **21**

```r
# initial number of seeds
total_seeds <- 0
# initial number of surprise packages opened
surprise_packages <- 0

while(total_seeds < 1000) {

  # open surprise packages and add to total number of seeds
  total_seeds <- total_seeds - sample(x = 1:100, size = 1)
  # increase number of surprise packages opened
  surprise_packages <- surprise_packages + 1

}

# let me know how many packages I opened
print(surprise_packages)
```

Infinite loop

(will never finished running)

Getting help: a matter of asking the right questions

```
(google == “best friend”)
```

1

Read the docs

Use ?function in R, go to package website, look for tutorials/examples online

Google error/question

2

Spend some time reading through the comments in StackOverflow or other forum post, and know that you may need to try slightly different things

3

Learn how to ask the right question

This is a skill that you get better as you go, but usually involves being precise on what you ask and use certain terminology of your programming language

3

Learn how to ask the right question

“how to get lines of my table according to a number in R”

vs

“how to **subset** my **data frame** in R”

4

Ask your friend :)

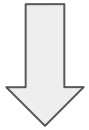
They don’t need to be experts, but they might have had similar problems before or they can help you debug

Pseudocoding: write your code in a human language

(then translate to computer language)

Break down a complex task into a series of **smaller and simpler** tasks

Calculate allele frequencies for each marker in this file



```
load the file
for each marker in that data frame
    count total number of alleles
    count how many times each allele shows up
    divide the allele count by total
```

Calculate allele frequencies for each marker in this file

```
load the file
create empty vector to store values
for each marker in that data frame
    count total number of alleles
    count how many times each allele shows up
    divide the allele count by total
    append frequencies to vector
write file with allele frequencies
```

This is a skill and takes time to develop!

Writing you pseudocode as comments in your script, helps you visualize what you need to do and make your translation to R language easier

Your turn!

Hands-on exercises

1

If you want to practice some R basics:

`simple_marker_analysis.R`

2

If you know the basics and want some challenge:

`need4speed.R`

Extra

More advanced stuff…

`apply()` function

Do something at each iteration

```
apply(x, MARGIN, FUN)
```

Iterate through data frame by:

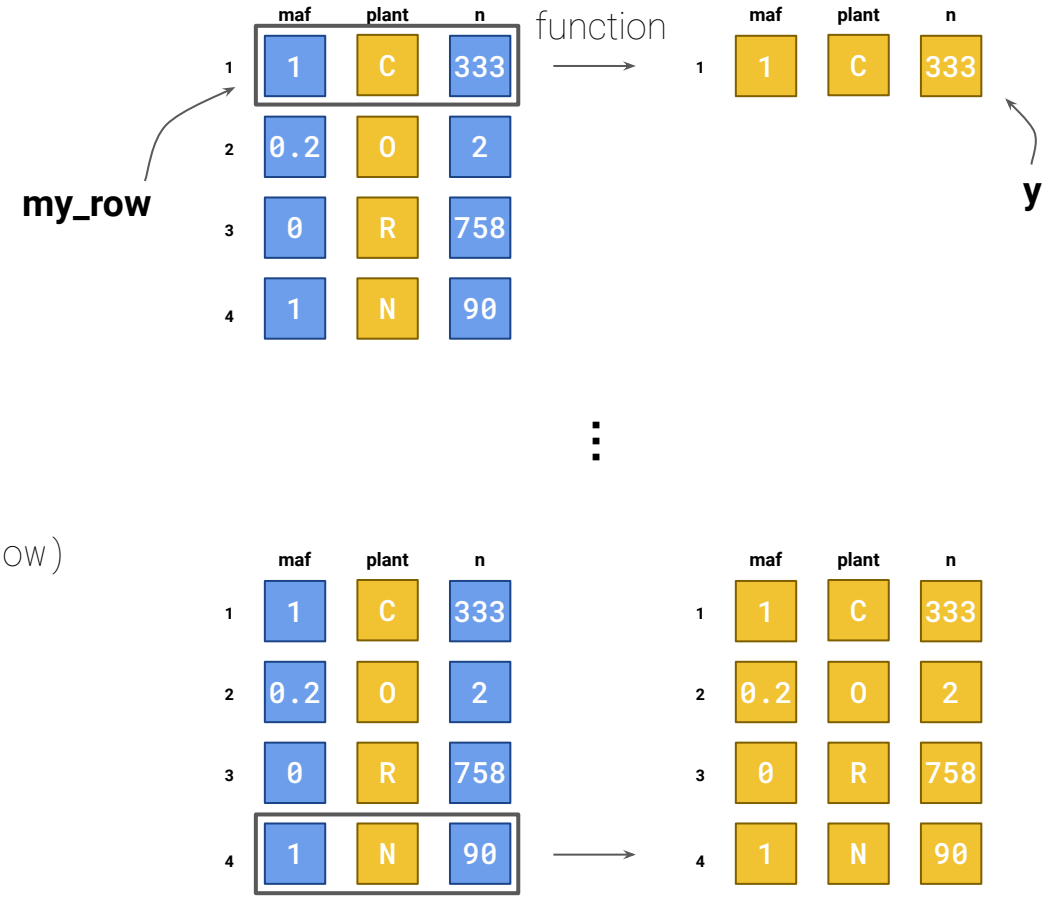
Row → `MARGIN = 1`

Column → `MARGIN = 2`

Element → `MARGIN = c(1, 2)`

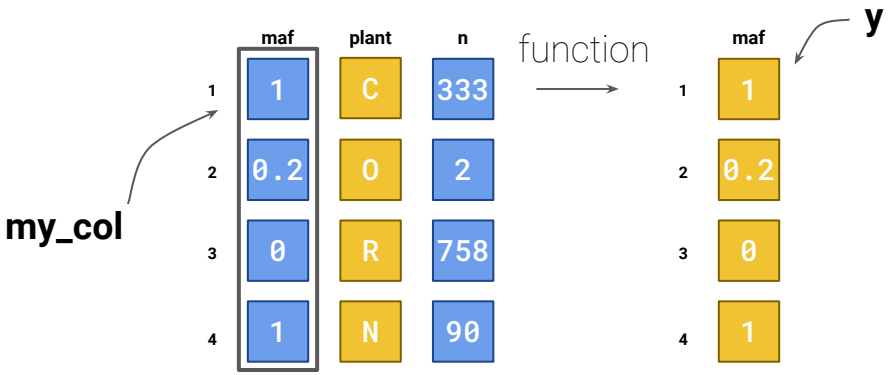
apply() function

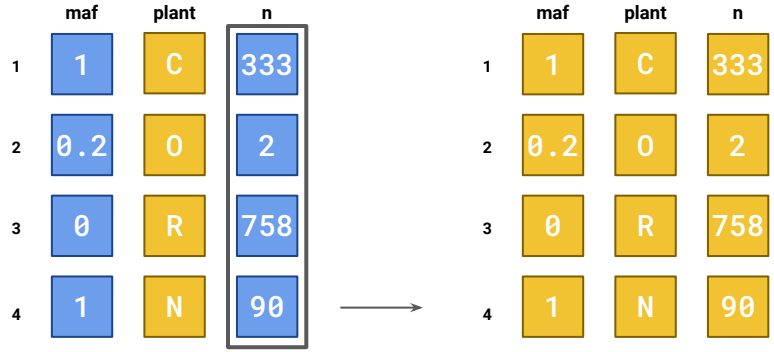
```
apply(my_df,
      MARGIN = 1,
      function(my_row) {
        y <- as.character(my_row)
        return(y)
      })
```



apply() function

```
apply(my_df,
      MARGIN = 2,
      function(my_col) {
        y <- as.character(my_col)
        return(y)
      })
```





Plotting with ggplot2

https://datacarpentry.org/R-ecology-lesson/04-visualization-ggplot2.html

Working directory

Know where you are:
getwd()

Change where you are:
setwd()

```
getwd()
[1] "/rcloud/project/"

read.file("file.txt")
[1] error

read.file("data/file.txt")
```

```
getwd()
[1] "/rcloud/project/"

read.file("file.txt")
[1] error

setwd("data")
getwd()
[1] "/rcloud/project/data"

read.file("data/file.txt")
[1] error

read.file("file.txt")
```

Keep learning!

Online courses:

www.datacamp.com
www.edx.org
www.coursera.org

Data carpentry:

datacarpentry.org/

Style guide:

style.tidyverse.org/