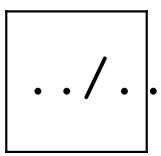
Workshop: Introduction to R

Jean Monlong & Simon Papillon

Human Genetics department

October 21, 2013



Workshop: Introduction to R

Jean Monlong & Simon Papillon

Human Genetics department

October 21, 2013

../../imgs/McGill-Logo1.png

Welcome blabla,
Who's a complete beginner ?
Give me a R, give me a ... well that's it
Potential addition: matrix fill up scheme, Rstudio use guide/slide, Tips box, Question box
Exercises/data to prepare:
nice/funny plots
useful function
debugging
one-liner quiz.

Why learning R ?

Useful for your research

- To explore your results. Curiosity and safety !
- To do/understand your analysis. Independence and control !
- To apply the latest Bioinformatics analyzes. Bioconductor !
- To keep track of your analysis. Reproducibility and automation !

- You do it, not some busy bioinformatician.

It's a good time investment

Simple: interpretative language(no compilation needed), no memory management, +++

Free: widely used, vast community of R users, good life expectancy.

Multiplatform: Windows, Mac, Unix, it works everywhere.

Why learning R ?

Useful for your research

- To explore your results. Curiosity and safety !
- To do/understand your analysis. Independence and control !
- To apply the latest Bioinformatics analyzes. Bioconductor !
- To keep track of your analysis. Reproducibility and automation !
- You do it, not some busy bioinformatician.

It's a good time investment

Simple: interpretative language(no compilation needed), no memory management, +++

Free: widely used, vast community of R users, good life expectancy.

Multiplatform: Windows, Mac, Unix, it works everywhere.

671 packages in Bioconductor. Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data.

to other languages Comparison with C ?

The shuffle array example is good

R and Rstudio

Easy installation

- Install R from
  `http://cran.r-project.org/`
- Install Rstudio Desktop from
  `http://www.rstudio.com/ide/download/desktop`

Easy installation

- Install R from `http://cran.r-project.org/`
- Install Rstudio Desktop from `http://www.rstudio.com/ide/download/desktop`

Emacs+ESS on Linux, R console on Mac

Data structure - Overview

Unit type

numeric	Numbers, e.g. 0, 1, 42, −66.6.
character	Words, e.g. “male”, “ENSG0007”,“Vive la France”.
logical	Boolean: *TRUE* or *FALSE*.

Container

vector	Concatenation of elements of the same type.
matrix	Matrix of element of the same type.
list	Flexible container, mixed type possible. Recursive.
data.frame	Table-like structure, same type within a column. Recursive.

Data structure - Overview

Unit type

numeric	Numbers, e.g. 0, 1, 42, −66.6.
character	Words, e.g. "male", "ENSG0007","Vive la France".
logical	Boolean: *TRUE* or *FALSE*.

Container

vector	Concatenation of elements of the same type.
matrix	Matrix of element of the same type.
list	Flexible container, mixed type possible. Recursive.
data.frame	Table-like structure, same type within a column. Recursive.

Other type but more complex and less useful, e.g. factors

Vector construction

c	Concatenate function.
1:10	Vector with numbers from 1 to 10.
rep	Repeat element several times.

Example

```
luckyNumbers = c(4,8,15,16,23,42)
luckyNumbers
oneToTen = 1:10
tenOnes = rep(1,10)
samples = c("sampA","sampB")
samples
```

Everything is a vector

is.vector(is.vector(1)) $->$ TRUE

Vector construction

- `c` Concatenate function.
- `1:10` Vector with numbers from 1 to 10.
- `rep` Repeat element several times.

Example

```
luckyNumbers = c(4,8,15,16,23,42)
luckyNumbers
oneToTen = 1:10
tenOnes = rep(1,10)
samples = c("sampA","sampB")
samples
```

Everything is a vector

is.vector(is.vector(1)) − > TRUE

Questions: Create your own numbers and favorite group of friends/hockey player/star/genes.

Characterization

length	Number of element in the vector.
names	Get or set the names of the vector.

Manipulation

vec[i:j]	Subset a vector from i^{th} to j^{th} values.
sort	Sort a vector.
order	Get the index of the sorted elements.
rev	Reverse a vector.
sample	Shuffle a vector.

Example

```
length(luckyNumbers)
luckyNumbers[2:4]
names(luckyNumbers) = c("frank","henry","philip",
                        "steve","tom","francis")
luckyNumbers
luckyNumbers["philip"]
rev(1:10)
order(c(luckyNumbers,1:10,tenOnes))
```

Characterization

length	Number of element in the vector.
names	Get or set the names of the vector.

Manipulation

vec[i:j]	Subset a vector from i^{th} to j^{th} values.
sort	Sort a vector.
order	Get the index of the sorted elements.
rev	Reverse a vector.
sample	Shuffle a vector.

Example

```
length(luckyNumbers)
luckyNumbers[2:4]
names(luckyNumbers) = c("frank","henry","philip",
                        "steve","tom","francis")
luckyNumbers
luckyNumbers["philip"]
rev(1:10)
order(c(luckyNumbers,1:10,tenOnes))
```

Square-brackets
Questions:
change the third number,
print a shuffle version of the vector
add “Jean” at the end of the character vector,
reverse it,
make the reverse the new value.

Exploration

head/tail Print the first/last values.

On numeric vectors:

summary Summary statistics: minimum, mean, maximum, ...

min/max/mean/var Minimum, maximum, average, variance.

sum Sum of the vector's values.

Example

```
head(samples)
summary(luckyNumbers)
mean(luckyNumbers)
```

Exploration

head/tail Print the first/last values.

On numeric vectors:

summary Summary statistics: minimum, mean, maximum, ...

min/max/mean/var Minimum, maximum, average, variance.

sum Sum of the vector's values.

Example

```
head(samples)
summary(luckyNumbers)
mean(luckyNumbers)
```

Tips: na.rm

Questions:

Show me the beginning of your numbers

the names of your numbers

change the name of the second value to something

average value of this beginning

the sum of the minimum and maximum value.

Operations

- Simple arithmetic operations over all the values of the vector.
- Or values by values when using vectors of same length.
- Arithmetic operation: +, -, *, /.

Example

```
luckyNumbers * 4 - 2
luckyNumber s* 1:length(luckyNumbers) -
                              rev(1:length(luckyNumbers))
```

Operations

- Simple arithmetic operations over all the values of the vector.
- Or values by values when using vectors of same length.
- Arithmetic operation: +, -, *, /.

Example

```
luckyNumbers * 4 - 2
luckyNumber s* 1:length(luckyNumbers) -
                          rev(1:length(luckyNumbers))
```

Let's apply it to the Exercise

Exercise - Guess my favorite number

Instructions

1. Create a vector of *numeric* values. At least two values.
2. Multiply it by 6.
3. Add 21.
4. Divide it by 3
5. Remove 1.
6. Halve it.
7. Remove its original values.

Instructions

1. Create a vector of *numeric* values. At least two values.
2. Multiply it by 6.
3. Add 21.
4. Divide it by 3
5. Remove 1.
6. Halve it.
7. Remove its original values.

Tips: save the original values somewhere or change the values of a new vector.

Specific to matrices

matrix	Create a matrix.
rbind/cbind	Concatenate vectors or matrix by row or column.
mat[i:j,k:l]	Subset from the i to j row and k to l column.
dim	Dimension of the matrix: number of rows and columns.
rownames/colnames	Get or set the names of the rows/columns.

Example

```
mat = matrix(runif(12),3,4)
colnames(mat) = c("col1","col2","col3","col4")
rownames(mat) = c("row1","row2","row3")
```

Specific to matrices

matrix	Create a matrix.
rbind/cbind	Concatenate vectors or matrix by row or column.
mat[i:j,k:l]	Subset from the i to j row and k to l column.
dim	Dimension of the matrix: number of rows and columns.
rownames/colnames	Get or set the names of the rows/columns.

Example

```
mat = matrix(runif(12),3,4)
colnames(mat) = c("col1","col2","col3","col4")
rownames(mat) = c("row1","row2","row3")
```

Questions:

create 4x4 matrix with number from 1 to 16

the same but shuffled

print the first column

the three first columns

Add an extra line to the matrix

Print the new dimension

Same as vector

- length, head, tail.
- For numeric matrix: min, max, sum, mean.
- Arithmetic operations: +, -, *, /.

Example

```
mean(mat)
sum(mat) / length(mat)

mat * 2
mat + mat
```

Same as vector

- length, head, tail.
- For numeric matrix: min, max, sum, mean.
- Arithmetic operations: +, -, *, /.

Example

```
mean(mat)
sum(mat) / length(mat)

mat * 2
mat + mat
```

Questions:
Average of the matrix
Average of the first two columns
multiply by 2 and remove the matrix

Flexible container

E.g. can concatenate a vector of *numeric* with a matrix of *numeric* and a matrix of *character*.

list	Create a list.
l[[i]]	Get or set the i^{th} object of the list.
l$toto	Get or set the element labeled as *toto*.
names	Get or set the names of the list elements.
length	Get the number of element in the list.
str	Output the structure of a R object.

Example

```
l = list(vec=1:10,mat=matrix(runif(25),5))
str(l)
l
l$vec = 1
l
```

Flexible container

E.g. can concatenate a vector of *numeric* with a matrix of *numeric* and a matrix of *character*.

list	Create a list.
l[[i]]	Get or set the i^{th} object of the list.
l$toto	Get or set the element labeled as *toto*.
names	Get or set the names of the list elements.
length	Get the number of element in the list.
str	Output the structure of a R object.

Example

```
l = list(vec=1:10,mat=matrix(runif(25),5))
str(l)
l
l$vec = 1
l
```

Questions:
Make a 3-dimensional (4x4x4) data type using a list

Exercise

1. Create a matrix of with 100 rows and 4 columns with random numbers inside. *Tip:* runif *function for random numbers.*
2. Name the columns. E.g. *sampleA*, *sampleB*, ...
3. Print the name of the column with the largest mean value.
4. Print the name of the column with the largest value.

Exercise

1. Create a matrix of with 100 rows and 4 columns with random numbers inside. *Tip:* `runif` *function for random numbers.*
2. Name the columns. E.g. *sampleA*, *sampleB*, ...
3. Print the name of the column with the largest mean value.
4. Print the name of the column with the largest value.

What if it had 100 rows...

- Name of the function with arguments between parenthesis.
- E.g. mean(x).

Do your own

function To define functions.

- All the object created within the function are temporary.

return Define what will be returned by the function.

Example

```
almostMean = function(x){
  x.mean = mean(x)
  return(x.mean+1)
}
almostMean(0:10)
x.mean
```

- Name of the function with arguments between parenthesis.
- E.g. mean(x).

Do your own

function To define functions.
- All the object created within the function are temporary.

return Define what will be returned by the function.

Example

```
almostMean = function(x){
  x.mean = mean(x)
  return(x.mean+1)
}
almostMean(0:10)
x.mean
```

Question:create a function that returns the power: pow ¡- function(base, exp) ...

New best friend

- Apply a function to row or columns of a 2 dimension data structure (matrix or data frame).
- No manual iteration, the loop is implicit.
- Second argument: 1 means rows, 2 means columns.

Example

```
apply(mat,1,mean)
apply(mat,2,function(x){
  x.mean = mean(x)
  return(x.mean+1)
})
```

Same for list, etc
output

New best friend

- Apply a function to row or columns of a 2 dimension data structure (matrix or data frame).
- No manual iteration, the loop is implicit.
- Second argument: 1 means rows, 2 means columns.

Example

```
apply(mat,1,mean)
apply(mat,2,function(x){
  x.mean = mean(x)
  return(x.mean+1)
})
```

apply for lists

- Useful way to iterate through lists.

Example

```
file_list <- read.files('.')
files_content <- lapply(file_list, function(file) \{
data <- read.csv(file)
#Do something with the data
return(data)
\})
```

Boolean

logical Binary data: *TRUE* or *FALSE*.

Numeric comparison ==, !=, >, <, >=, <=.

Boolean operation AND: &, OR: |, NOT: !

which Returns the index of the vectors with *TRUE* values.

any Take a vector of *logical* and return *TRUE* if at least one value is *TRUE*.

%in% Vectorized any. See example/supp material.

Example

```
2 + 2 == 4
(2 < 3) & (3 != 1+2)
which(5:10 == 6)
any(9>1:10)
any(9>1:10 & 8<=1:10)
luckyNumbers[which(luckyNumbers %in% c(16,42,-66.6))]
```

Boolean

logical Binary data: *TRUE* or *FALSE*.

Numeric comparison ==, !=, >, <, >=, <=.

Boolean operation AND: &, OR: |, NOT: !

which Returns the index of the vectors with *TRUE* values.

any Take a vector of *logical* and return *TRUE* if at least one value is *TRUE*.

%in% Vectorized any. See example/supp material.

Example

```
2 + 2 == 4
(2 < 3) & (3 != 1+2)
which(5:10 == 6)
any(9>1:10)
any(9>1:10 & 8<=1:10)
luckyNumbers[which(luckyNumbers %in% c(16,42,-66.6))]
```

Is more details on logical rules necessary ?

Question: write a function that filters out numbers: largerThan ¡- function(data, threshold) {...}

conditions

if else

Test if a condition, if *TRUE* run some instruction, if *FALSE* something else (or nothing).

Example

```
if(length(luckyNumbers)>3){
  cat("Too many lucky numbers.\n")
  luckyNumbers = luckyNumbers[1:3]
} else if(length(luckyNumbers)==3){
  cat("Just enough lucky numbers.\n")
} else {
  cat("You need more lucky numbers.\n")
}
```

conditions

if else

Test if a condition, if *TRUE* run some instruction, if *FALSE* something else (or nothing).

Example

```
if(length(luckyNumbers)>3){
  cat("Too many lucky numbers.\n")
  luckyNumbers = luckyNumbers[1:3]
} else if(length(luckyNumbers)==3){
  cat("Just enough lucky numbers.\n")
} else {
  cat("You need more lucky numbers.\n")
}
```

Maybe more theoretical structure

Question: write a function that filter number higher than 10

for loops

Iterate over the element of a container and run instructions.

```
for(v in vec){
...  Instruction
}
```

while loops

Run instructions as long as a condition is *TRUE*.

```
while( CONDITION ){
...  Instruction
}
```

for loops

Iterate over the element of a container and run instructions.

```
for(v in vec){
...  Instruction
}
```

while loops

Run instructions as long as a condition is *TRUE*.

```
while( CONDITION ){
...  Instruction
}
```

Question:

Import/export data

Easy but important

- What data structure is the more appropriate ? vector, matrix ?
- Does R read/write the file the way you want ?
- The extra arguments of the functions are your allies.

scan

To read a vector from a file with, for example, one value per line.

file=	the file name.
what=	the type of the argument gives the type of the values, e.g 1, $''a''$.
sep=	the character that separate each value. By default, a white-space or end of line.

write

To write a vector from a file with one value per line.

vec	the vector to write.
file=	the file name.
sep=	the character that separate each value.

Questions: try to write on vector
Then re-read it.

Easy but important

- What data structure is the more appropriate ? vector, matrix ?
- Does R read/write the file the way you want ?
- The extra arguments of the functions are your allies.

scan

To read a vector from a file with, for example, one value per line.

- file= the file name.
- what= the type of the argument gives the type of the values, e.g 1, "a".
- sep= the character that separate each value. By default, a white-space or end of line.

write

To write a vector from a file with one value per line.

- vec the vector to write.
- file= the file name.
- sep= the character that separate each value.

Import/export data

read.data

To read a data.frame from a multi-column file.

file=	the file name.
header=	*TRUE* use the first line for the column names. Default: *FALSE*.
as.is=	*TRUE* read the values as simple type, no complex type inference, **recommended**. Default: *FALSE*.
sep=	the character that separate each column. By default, a white-space or end of line.

write.data

To write a data.frame in a multi-column file.

df	the matrix or data.frame to write.
file=	the file name.
col.names=	*TRUE* print the column names in the first line. Default: *TRUE*.
row.names=	*TRUE* print the rows names in the first columns. Default: *TRUE*.
quote=	*TRUE* surround character by quotes(″). Default: *TRUE* → messy.
sep=	the character that separate each column. By default, a white-space.

Import/export data

read.data

To read a data.frame from a multi-column file.

- file= the file name.
- header= *TRUE* use the first line for the column names. Default: *FALSE*.
- as.is= *TRUE* read the values as simple type, no complex type inference, **recommended**. Default: *FALSE*.
- sep= the character that separate each column. By default, a white-space or end of line.

write.data

To write a data.frame in a multi-column file.

- df the matrix or data.frame to write.
- file= the file name.
- col.names= *TRUE* print the column names in the first line. Default: *TRUE*.
- row.names= *TRUE* print the rows names in the first columns. Default: *TRUE*.
- quote= *TRUE* surround character by quotes(″). Default: *TRUE* → messy.
- sep= the character that separate each column. By default, a white-space.

Questions: try to write a matrix with the different arguments
Then re-read it.

/export data

R objects

save Save R objects into a file. Usual extension: *.RData*. file= argument to specify file name.

save.image Save the entire R environment.

load Load R objects from a (*.RData*) file. verbose to print the names of the objects loaded.

Example

```
save(luckyNumbers, tenOnes, mat, file="uselessData.RData")
load(file="uselessData.RData")
load(file="dataForBasicPlots.RData",verbose=TRUE)
```

/export data

R objects

save	Save R objects into a file. Usual extension: *.RData*. file= argument to specify file name.
save.image	Save the entire R environment.
load	Load R objects from a (*.RData*) file. verbose to print the names of the objects loaded.

Example

```
save(luckyNumbers, tenOnes, mat, file="uselessData.RData")
load(file="uselessData.RData")
load(file="dataForBasicPlots.RData",verbose=TRUE)
```

Rstudio tips
Questions: load data for next exercise.
Save your objects if you want to...

Basic plotting

hist	Plot the value distribution of a vector.
plot	Plot one vector against the other.
line	Same as plot but super-imposed to the existent one.
abline	Draw vertical/horizontal lines.

Common arguments

main=	A title for the plot.
xlim=/ylim	A vector of size two defining the desired limit on the x/y axis.
xlab=/ylab=	A name for the x/y axis.

hist	Plot the value distribution of a vector.
plot	Plot one vector against the other.
line	Same as plot but super-imposed to the existent one.
abline	Draw vertical/horizontal lines.

Common arguments

main=	A title for the plot.
xlim=/ylim	A vector of size two defining the desired limit on the x/y axis.
xlab=/ylab=	A name for the x/y axis.

Questions: plot the prepared data(some funny shaped plots ?)
Histogram with vertical line on the mean

Debugging

Instructions

1. Open **scriptToDebug.R** document.
2. Run and debug it !

Instructions

1. Open **scriptToDebug.R** document.
2. Run and debug it !

Bugs: header load table, type read.table, parenthesis/brackets, infinite loop, NA in mean etc, operation different length, type coercion numeric character, non-unique (col)names, (global variable within function), apply rows returning matrix

One-liner quiz

Instructions

Write R command to address each question. Only one-line command allowed. The shorter the better.

Questions

1. From a matrix of numeric, compute the proportion of columns with average value higher than 0.
2. From a matrix of numeric, print the name of the columns with the highest value.
3. From a matrix of numeric, print the rows with only positive values.
4.

Find more questions.

One-liner quiz

Instructions

Write R command to address each question. Only one-line command allowed. The shorter the better.

Questions

1. From a matrix of numeric, compute the proportion of columns with average value higher than 0.
2. From a matrix of numeric, print the name of the columns with the highest value.
3. From a matrix of numeric, print the rows with only positive values.
4.

coercion.

- Automatic conversion of an object to another type, e.g numeric→character, logical→numeric.
- Awareness for debugging.
- Useful sometimes.

Example

```
is.numeric( c(1:10,"eleven") )

logical.vector = c(TRUE,TRUE,FALSE,TRUE,FALSE)
sum(logical.vector)
mean(logical.vector)
```

coercion.

- Automatic conversion of an object to another type, e.g numeric→character, logical→numeric.
- Awareness for debugging.
- Useful sometimes.

Example

```
is.numeric( c(1:10,"eleven") )

logical.vector = c(TRUE,TRUE,FALSE,TRUE,FALSE)
sum(logical.vector)
mean(logical.vector)
```

Questions: How would you do it

character

operations

paste	Paste several character into one.
grep	Search a pattern in a vector and return the index when matched.
grepl	Search a pattern in a vector and return *TRUE* if found.
strsplit	Split character into several.

Example

```
sample.name = "Ob5cU8eN4mE"
file.name = paste("pathToYourDirectory/greatAnalysis-",
                                    sample.name,".txt",sep="")

which(sample.names=="controlA" & sample.names=="controlB")
grep("control",sample.names)
```

More details

character

operations

paste	Paste several character into one.
grep	Search a pattern in a vector and return the index when matched.
grepl	Search a pattern in a vector and return *TRUE* if found.
strsplit	Split character into several.

Example

```
sample.name = "Ob5cU8eN4mE"
file.name = paste("pathToYourDirectory/greatAnalysis-",
                                  sample.name,".txt",sep="")

which(sample.names=="controlA" & sample.names=="controlB")
grep("control",sample.names)
```

object name

- **Letters**, **numbers**, **dot** or **underline** characters.
- **Starts with a letter** or the dot not followed by a number.
- make.names convert character into valid object names.

Example

```
make.names(c("valid name","valid_name","valid.name",
             "valid-name","2.valid.name","x2.valid-name"))
```

object name

- **Letters**, **numbers**, **dot** or **underline** characters.
- **Starts with a letter** or the dot not followed by a number.
- make.names convert character into valid object names.

Example

```
make.names(c("valid name","valid_name","valid.name",
             "valid-name","2.valid.name","x2.valid-name"))
```

Should it be present in the beginning ?