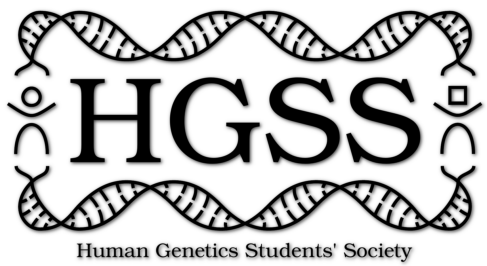
HGSS Workshop : Introduction to R

Jean Monlong

Human Genetics department

Sept 23rd, 2016



McGill

Why R ?

Why R ?

Simple

- Interpretative language (no compilation needed).
- No manual memory management.
- Vectorized.

Free

- Widely used, vast community of R users
- Good life expectancy.

Flexible

- Open-source: anyone can see/create/modify code.
- Multiplatform: Windows, Mac, Unix, it works everywhere.

Trendy

- More and more packages.
- More and more popular among (big) data scientist.

Easy installation

- Install R from
  `http://cran.r-project.org/`

- Additionally, you can get a nice interface through Rstudio Desktop from
  `http://www.rstudio.com/ide/download/desktop`

Workshop setup

Open Rstudio

- Click on the bottom-left corner
- Type *rstudio*, click on Rstudio icon.

In Rstudio

- On the bottom-right panel, go to *Documents* folder.
- Create one for your data and scripts. E.g. *Rworkshop*.
- Set this folder as working directory (*More* button).
- Create an empty script for today's session (*File*, *New File*, *R Script*).

Download today's slides and data

1. Go to `http://goo.gl/SYRrmy`.
2. Download everything.
3. Put it in your *Rworkshop* folder.

Console ? Script ?

Console

- Where R is running.
- You could write and run the commands directly there.

Script

- A text file with commands. *Extension: .R.*
- To keep a trace of your analysis.
- Recommended.
- Easy to send commands from a script to the console.

When you get an error

1. Read the command, look for typos.
2. Read the error message.
3. 1. and 2. again.
4. Raise your hand, someone will assist you.

Solving errors is an important skill to learn.

Data structure

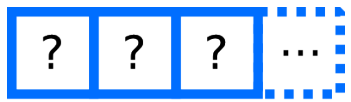
Data structure - Overview

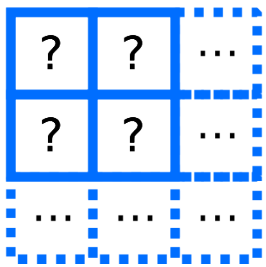
Unit type

numeric	Numbers, e.g. 0, 1, 42, −66.6.
character	Words, e.g. “male”, “ENSG0007”,“Allez les bleus”.
logical	Binary, i.e. two possible values: *TRUE* or *FALSE*.

Structure

vector	Ordered collection of elements of the same type.
matrix	Matrix of element of the same type.
list	Flexible container, mixed type possible. Recursive.





Assign a value to an object

Choose an object name

- **Starts with a letter** or the dot not followed by a number.
- **Letters**, **numbers**, **dot** or **underline** characters.
- Correct: “valid.name”, ”valid_name”, ”valid2name3”.
- Incorrect: ”valid name”, ”valid-name”, ”1valid2name3”.

Assign a value

The name of the object followed by the assignment symbol and the value.

```
valid.name_123 = 1
valid.name_123 <- 1

valid.name_123
```

Use a function

- **Parenthesis** are for **functions only**.
- The rest will be for data manipulation.
- Read help manual to know more about a function (help, ? or F1 in Rstudio).

```
print(1)
myFunction(valid.name_123)

help(print)
?print
```

Vectors

Vectors

vector construction

c	Concatenate function.
1:10	vector with numbers from 1 to 10.

Example

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

Extra

seq	Create a sequence of numbers.
rep	Repeat element several times.
runif	Simulate random numbers from Uniform distribution. Same for rnorm, rpois, ...

Exercise - Create some vectors

Instructions

- Create a vector with 7 *numeric* values.
- Create a vector with 7 *character* values.
- Be creative !

Vectors

Manipulation

Using index/position between [ ].

Characterization

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

Example

```
luckyNumbers[3]
luckyNumbers[2:4]
luckyNumbers[2:4] = c(14,3,9)

length(luckyNumbers)

names(luckyNumbers)
names(luckyNumbers) = c("frank","henry","philip",
                        "steve","tom","francis")
luckyNumbers["philip"]
```

Vectors

Manipulation

sort	Sort a vector.
sample	Shuffle a vector.

Example

```
sort(luckyNumbers)

sort(c(luckyNumbers,1:10,tenOnes))

rev(1:10)

sample(1:10)
```

Extra

sort/sample	Explore extra parameters.
order	Get the index of the sorted elements.

Vectors

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)
min(luckyNumbers)
```

Extra

log/log2/log10 Logarithm functions.

sqrt Square-root function.

Vectors

Arithmetic operators

- Simple arithmetic operations over all the values of the vector.
- Or values by values when using vectors of same length.
- Arithmetic operation: +, -, *, /.
- Others exist but let's forget about them for now.

Example

```
luckyNumbers * 4
luckyNumbers - luckyNumbers
luckyNumbers / 1:length(luckyNumbers)
luckyNumbers + 2
```

Exercise - Guess my favorite number

Instructions

1. Create a vector with 5 *numeric* values
2. Multiply it by 6.
3. Add 21.
4. Divide it by 3
5. Subtract 1.
6. Halve it.
7. Subtract its original values.

Matrix

Matrix

Specific to matrices

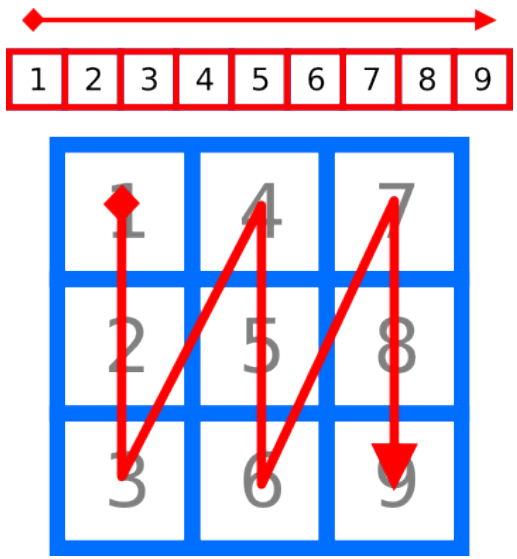
matrix Create a matrix from a vector.
2^{nd} and 3^{rd} parameters define the number of rows and columns.

mat[i,j] Element at row i and column j. If blank, the entire row/column is used.

Example

```
neo = matrix(1:12,3,4)
neo

neo[1,1] = 0
neo[1:2,1:3]
neo[1:2,1:3] = matrix(rep(1,6),2,3)
neo[1,]
```



Exercise

1. Create a matrix with 10 rows and 4 columns with numbers from 1 to 40.
2. Change the element in row 6 column 1 into the value 666.
3. Fill the 3rd row with ones.

Matrix

Specific to matrices

dim Dimension of the matrix: number of rows and columns.

rownames/colnames Get or set the names of the rows/columns.

Example

```
dim(neo)
dim(rbind(neo,neo))

colnames(neo) = c("gene1","gene2","gene3","gene4")
rownames(neo) = c("sample1","sample2","sample3")
neo
neo["sample2","gene3"]
```

Matrix

Same as vector

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

Example

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

mat * 2
mat + mat
```

Extra

log/log2/log10 Logarithm functions.

sqrt Square-root function.

Exercise

1. Create a matrix 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. Add 2 to the first column.
4. Multiply the second column by 4.
5. Find which column has the largest mean value.
6. Find which column has the largest value.

Functions - apply

New best friend

- Apply a function to each row (or column) of a matrix.
- No manual iteration, the loop is implicit.
- Second parameter: 1 means rows, 2 means columns.

Example

```
apply(mat,1,mean)
```

Apply - Exercise

1. Create a matrix with 100 rows and 100 columns with random numbers inside.
2. Compute the median value of each column.
3. What is the minimal median value ? Maximal ?

Import/export data

Import/export data - Text files

Easy but important

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

read.table

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, **recommended**. Default: *FALSE*.
sep=	the *character* that separate each column. Use ”^” for tabulation.
row.names=	the column number to use as row names.

Example

```
input.data = read.table("fileToRead.txt", as.is=TRUE,
                        header=TRUE, sep="\t", row.names=1)
```

Exercice

Instructions

Read dataForBasicPlots.tsv into an object called mat.ge.

dataForBasicPlots.tsv

- Columns separated by tabulation.
- First line represent the column names.
- First column is gene names, other columns are expression of these genes for different samples.

Questions

1. How many genes are there ?
2. How many samples ?
3. Print the first 5 row and columns.

Import/export data - Text files

write.table

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.

Example

```
write.table(resToWrite, file="fileToRead.txt", col.names=TRUE,
                          row.names=FALSE, quote=FALSE, sep="\t")
```

Import/export data

R objects

save Save R objects into a file. Usual extension: *.RData*. file= parameter 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")
```

Basic plotting

Basic plotting

hist

Plot the value distribution of a vector.

x The vector with the values to plot.

plot

Plot one vector against the other.

x The first vector to plot. *x-axis.*

y The second vector to plot. *y-axis.*

type How the points are plotted. "p" as points, "l" joined by lines.

Example

```
hist(mat.ge[,1])
plot(mat.ge[,1],mat.ge[,2])
```

Basic plotting

Common parameters

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

Example

```
hist(mat.ge[,1],main="A basic graph",
                xlab="first column values")

plot(mat.ge[,1],mat.ge[,2],main="Another basic graph",
  xlab="first column values",ylab="second column values")
```

Basic plotting

Extra parameters

col	the colour of the points/lines. 1:black, 2:red, ...
pch	Shape of the points. 1:circle, 2:triangle, ...
lty	Shape of the lines. 1:plain, 2:dotted, ...

Extra functions

lines	Same as plot but super-imposed to the existent one.
abline	Draw vertical/horizontal lines.

Example

```
plot(mat.ge[,1],mat.ge[,2],main="Another basic graph",
  xlab="first column values",ylab="second column values")
lines(mat.ge[,1],mat.ge[,3],type="p",col=2,pch=2)
abline(h=0,lty=2)
```

Basic plotting - Exercise

Plot:

1. the distribution of the median gene(row) expression. Add a vertical dotted line to mark their average value.
2. the distribution of the median sample(column) expression. If any visual outlier, remove it and check distribution again. Tips: which.min and which.max functions give the position of the minimum/maximum values.
3. the expression(row) of *gene333* against *gene666*. Superimpose in red triangles the expression(row) of *gene333* against *gene667*.

Conditions

Logical values

Logical type

TRUE / FALSE values

Example

```
hgssRules = TRUE
dwight = FALSE
male = c(TRUE, FALSE, TRUE)
```

Conditions

Logical tests

==	both values equal ?
> or >=	left value greater (greater or equal) than right value ?
< or <=	left value smaller (smaller or equal) than left value ?
!	NOT operator : negates the value.
|	OR operator : returns TRUE if either are TRUE.
&	AND operator : returns TRUE if both are TRUE.

Example

```
test <- 2 + 2 == 4    ## (TRUE)
!test                 ## (FALSE)
test & !test          ## (FALSE)
test | !test          ## (TRUE)
```

Conditions

Vectorized operations

Any logical tests can be vectorized (compare 2 vectors).

|	Is a OR operator for vectorized application.
&	Is an AND operator for vectorized application.
which	Returns the index of the vectors with *TRUE* values.

Example

```
c(TRUE, TRUE) & c(TRUE, FALSE) -> TRUE, FALSE

which(5:10 == 6)
which(luckyNumbers > 2)

luckyNumbers[which(luckyNumbers>2 & luckyNumbers<10)]
```

Conditions - Exercise

1. Create a vector of random integer numbers between 0 and 10.

   Tips:

   - 2nd and 3rd parameters of sample function.
   - OR 2nd and 3rd parameters of runif function and round.
2. Remove values below 3.
3. Change to 8 any value higher than 8.

On mat.ge

Remove all genes with median expression lower than 1.

Testing conditions

if else

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

```
if( Condition ){
...   Instructions
}
```

Example

```
luck = "none"
if(length(luckyNumbers)>3){
  luck = "a lot"
} else if(length(luckyNumbers)==3){
  luck = "some"
} else {
  luck = "not enough"
}
```

Conditions - Exercise

Write a if block that automatically classify the expression of the first gene into :

- 'high' if its maximum value is higher than 4
- 'low' if not.

Functions

Functions

- Name of the function with parameters between parenthesis.
- Takes input(s) and return something. E.g. mean(luckyNumbers).

Do your own

- function To define functions.
- All the object created within the function are temporary.
- return Specify what will be returned by the function.

Structure

```
myFunctionName <- function(input.obj1,second.input.obj ) {
...
... Instructions on 'input.obj1' and 'second.input.obj'
...
return(my.output.obj)
}

myFunctionName(1,c(2,4,5))
```

Functions - Example

Function takes a vector as input and :

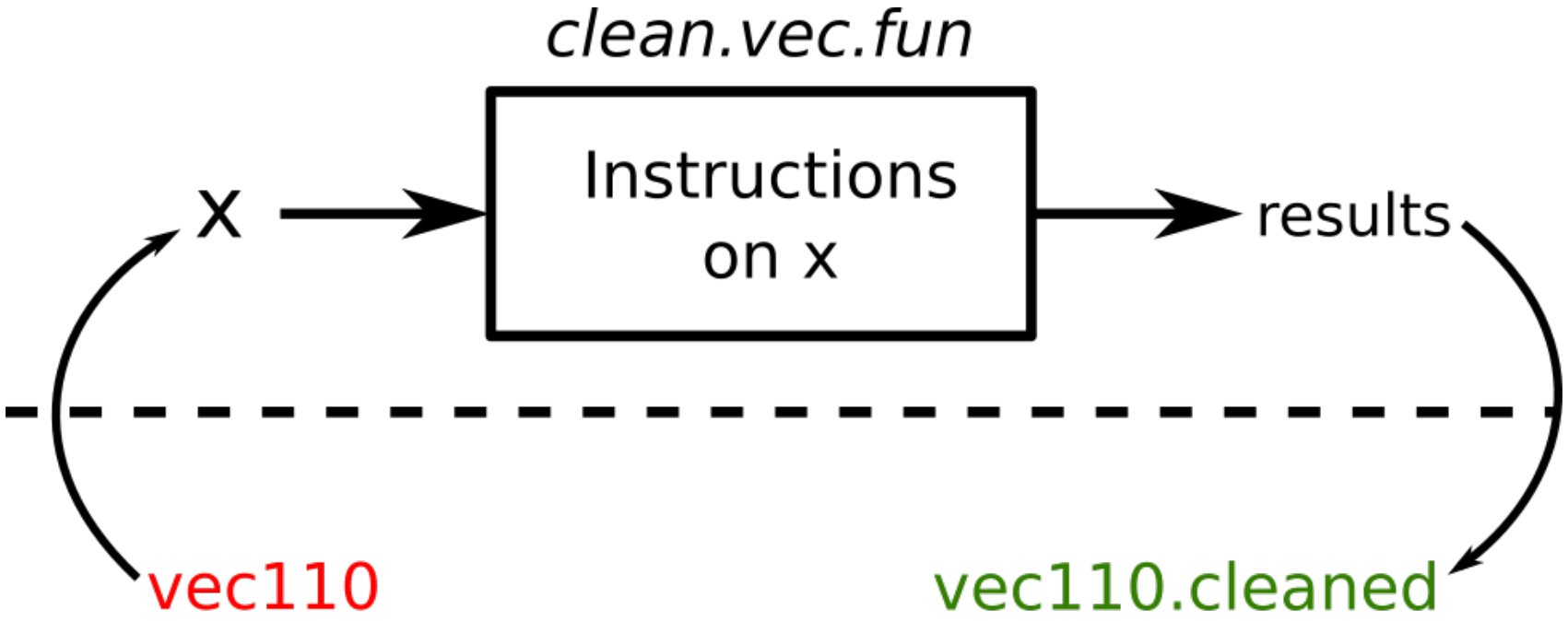
- removes values lower than 3.
- changes to 8 values higher than 8.

Example

```
clean.vec.fun <- function(x){
  x = x[which(x>=3)]
  x[which(x>8)] = 8
  return(x)
}

vec110 = 1:10
vec110.cleaned = clean.vec.fun(vec110)
```

Functions - Concept



vec110.cleaned = clean.vec.fun(vec110)

Functions - Exercise

Create a function that classify the average value of a vector. It returns:

- *low* if the average if below 3.
- *medium* if the average if between 3 and 7.
- *high* if the average if above 7.

Create a function that:

1. returns the average of the minimum and maximum value of a vector.
2. returns how many values are higher than 3 in a vector.

- Test your functions on vectors with random number from 0 to 10.
- How would you run them on all mat.gene genes ?

Final exercise

Differential gene expression

1. Load metadata.RData file. It has a groups vector with either case/control status for the mat.ge samples.
2. Write a function that would compute the difference between the gene expression of cases and controls.
3. Apply this function to each gene(row) in mat.ge.
4. Plot the distribution of the results.

Online resources

R basics

- `http://www.twotorials.com/` : small video-tutorials.
- `www.youtube.com/user/rdpeng/` : Coursera *Computing for Data Analysis* videos. Other interesting videos, e.g. *ggplot2*.

- `https://www.datacamp.com/` or `http://tryr.codeschool.com/` : Interactive tutorial of R basics.

- `http://www.r-tutor.com/` : R and statistics small web-tutorials.
- `http://www.computerworld.com/s/article/9239625/Beginner_s_guide_to_R_Introduction` : Beginner's guide with screenshots.
- `http://cran.r-project.org/manuals.html` : R manual.

Bioinformatics

- `http://stephenturner.us/p/edu` List of online resources for Bioinformatics.
- `http://bioinformatics.ca/workshops/2013/` : Bioinformatics workshop material.
- `http://manuals.bioinformatics.ucr.edu/home/R_BioCondManual` : Pieces of code for bioinformatics analysis, plots. Including Bioconductor.
- `http://bioconductor.org/help/course-materials/2013/` : Bioinformatics tutorials material: pdf and R scripts.

Extra

Loops

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
}
```

Example

```
facto = 1
for(n in 1:10){
   facto = facto * n
}
```

Loops - Exercise

Write a function that computes the mean values of the columns:

1. using the apply function.
2. using a for loop.
3. (using a while loop.)

Basic plotting

boxplot

Plot the distribution (quantiles/median/outliers) of variables.

x The matrix (or list) of distributions

Example

```
boxplot(mat.ge)
```

Save your plot into a *pdf/png*

Open a connection to a output file, plot as usual, close the connection.

pdf	Open the connection to a *pdf* output.
png	Open the connection to a *png* output.
dev.off()	Close the connection

Example

```
pdf("myNicePlot.pdf")
plot(...)
dev.off()
```

Type 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)
```

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)
```

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 column with the highest value.
3. From a matrix of *numeric*, print the rows with only positive values.