



BIOC | BioInformatics Competence Center

Bioinformatic Analysis of RNA-sequencing

R Introduction: Variables

Allison Burns, Maxime Jan, Linda Mhalla, Christian Iseli, Nicolas Gueux

General R guidelines

Packages

To install a package available in [CRAN](#)

```
install.packages("ggplot2")
```

load the package using:

```
library(ggplot2) # Load  
detach(package:ggplot2, unload=T) # UnLoad
```

Other packages like DESeq2 are only available in [Bioconductor](#)

you can install them like this:

```
if (!require("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")
```

```
BiocManager::install("DESeq2")
```

If two package have the same function name use the format *Package::Function*, (e.g. above *BiocManager::install("DESeq2")*)

If you want to know where your packages are, run this command:

```
.libPaths()
```

Help

You can see the help of a function using “?”

?write.table

write.table {utils}

Data Output

Description

Usage

write.csv(...)

write.csv2(...)

Arguments

x

file

append

quote

sep

col

na

dec

row.names

the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce x to a data frame.

either a character string naming a file or a connection open for writing. "" indicates output to the console.

logical. Only relevant if file is a character string. If TRUE, the output is appended to the file. If FALSE, any existing file of the name is destroyed.

a logical value (TRUE or FALSE) or a numeric vector. If TRUE, any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. In both cases, row and column names are quoted if they are written. If FALSE, nothing is quoted.

the field separator string. Values within each row of x are separated by this string.

the character(s) to print at the end of each line (row). For example, col = "\r\n" will produce Windows' line endings on a Unix-alike OS, and col = "\r" will produce files as expected by Excelmac 2004.

the string to use for missing values in the data.

the string to use for decimal points in numeric or complex columns: must be a single character.

either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.

R Documentation

Working directory

Always think where is your “working directory (wd)” in R

In an Rmd, the working directory is located in the same directory as the Rmd file

```
# Get the working directory
getwd()
```

```
## [1] "C:/Users/Maxime-Laptop/OneDrive - Université de Lausanne/Bureau/Introduction_R/Introduction_R/lectures/2.Intro_And_Variables"
```

You can change the working directory, but it will be only valid for a chunk

```
setwd(dir = "image/")
list.files(path="." )
```

```
## [1] "help.PNG"
```

if you want to change the wd for all the document:

```
# knitr::opts_knit$set(root.dir = '/tmp')
```

Comments

You can use comment in your code or Rmarkdown chunk using the # character

```
# This is a comment  
1 + 1
```

```
## [1] 2
```

Variable name

Some variable name should not be used:

```
# Legal variable names:  
myvar <- "John"  
my_var <- "John"  
myVar <- "John"  
MYVAR <- "John"  
myvar2 <- "John"  
.myvar <- "John"
```

```
# Illegal variable names:  
2myvar <- "John"  
my-var <- "John"  
my var <- "John"  
_my_var <- "John"  
my_v@ar <- "John"  
TRUE <- "John"
```

Variables in R

Numeric variables

Attribute value to a variable (**R** is case sensitive !)

```
# Single variable
MyVariable<-15
# Vector
ManyVariables<-c(1,2,3,4,5)
```

Integer

```
# Add L to force number to be integer
SomeNumbers<-c(1L,22L,185L)
SomeNumbers
```

```
## [1] 1 22 185
```

```
class(SomeNumbers)
```

```
## [1] "integer"
```

Double

```
SomeDoubleNumbers<-c(1.21, 2/3 , pi)
SomeDoubleNumbers
```

```
## [1] 1.2100000 0.6666667 3.1415927
```

```
class(SomeDoubleNumbers)
```

```
## [1] "numeric"
```

Boolean variables

Variables are only TRUE or FALSE (**Do not use quotes !**)

```
MyBoolean<-c(TRUE,FALSE,TRUE,FALSE)
MyOtherBoolean<-c(TRUE,TRUE,FALSE,FALSE)
```

Some operation that return boolean:

AND

NOT

MyBoolean & MyOtherBoolean

!MyBoolean

[1] TRUE FALSE FALSE FALSE

[1] FALSE TRUE FALSE TRUE

OR

Compare values

MyBoolean | MyOtherBoolean

```
SomeDoubleNumbers<-c(1.21, 2/3 , pi)
SomeNumbers<-c(1L,1L,1L)
SomeDoubleNumbers > SomeNumbers
```

[1] TRUE TRUE TRUE FALSE

[1] TRUE FALSE TRUE

Characters & Factors

Often these two data types are confunded

```
Condition<-c("Ctr", "Ctr", "Treat", "Treat")
Condition_AsFactor<-factor(Condition)
```

Character

```
# print character
Condition
```

```
## [1] "Ctr" "Ctr" "Treat" "Treat"
```

```
class(Condition)
```

```
## [1] "character"
```

```
class(Condition_AsFactor)
```

```
## [1] "factor"
```

Factor

```
# print factors
Condition_AsFactor
```

```
## [1] Ctr Ctr Treat Treat
## Levels: Ctr Treat
```

```
# At a lower level, factors are stored as integer
typeof(Condition_AsFactor)
```

```
## [1] "integer"
```

You can get the levels of a factor

```
levels(Condition_AsFactor)
```

```
## [1] "Ctr" "Treat"
```

Vectors

You can only have **one** type of variable in a vector.

```
# Here numeric values are converted into characters
MyVector<-c("A", 1.2, 2/4, 2L)
MyVector
```

```
## [1] "A"      "1.2" "0.5" "2"
```

To access a value in a vector use brackets “[]” (e.g. MyVector[1])

You can access multiple elements

```
MyVector[1:3]
```

```
## [1] "A"      "1.2" "0.5"
```

```
# To access the last element of a vector:
# MyVector[length(MyVector)] or tail(MyVector,1)
```

Vectors can be named (often required in packages)

```
MyGenes<-c("Arnt1"=1.5, "Per2"=1.8, "Cdk4"=8.5)
MyGenes
```

```
## Arnt1 Per2 Cdk4
## 1.5 1.8 8.5
```

You can change the names of the vectors

```
names(MyGenes)
```

```
## [1] "Arnt1" "Per2" "Cdk4"
```

```
# Change name 2
names(MyGenes)[2]<- "Cry1"
MyGenes
```

```
## Arnt1 Cry1 Cdk4
## 1.5 1.8 8.5
```

Lists

If you want multiple type of variable, you can use lists.

You don't need same length for each element in the list

```
MyList<-list("SomeNumber"=c(1.5,2.3,4.5),
            "SomeName"=c("Gene1", "Gene2", "Gene3", "Gene4", "Gene5"),
            "SomeFactors"=factor(c("C1", "C1", "C2", "C2", "C4")))
```

You can see the names of elements in the list

```
names(MyList)
```

```
## [1] "SomeNumber" "SomeName" "SomeFactors"
```

You can access the "Numbers" of the list using brackets

But, it will return a list of 1 element

```
MyList["SomeNumber"]
```

```
## $SomeNumber
```

```
## [1] 1.5 2.3 4.5
```

To access the vector use double brackets or \$

```
MyList$SomeNumber
```

```
## [1] 1.5 2.3 4.5
```

```
MyList[["SomeNumber"]]
```

```
## [1] 1.5 2.3 4.5
```

Matrix

Matrix contains only **one** type of variable

```
# Completion is done by column
MyMatrix<-matrix(c(1,2,3,4,5,6),ncol=2)

MyMatrix
```

```
##      [,1] [,2]
## [1,]  1   4
## [2,]  2   5
## [3,]  3   6
```

You can change rownames and colnames

```
colnames(MyMatrix)<-c("A", "B")
rownames(MyMatrix)<-c("C", "D", "E")
```

You can acces a specific column or row

If only 1 row/column is requested, R return a vector. Think to use ,drop=F !

```
MyMatrix[1,] # First row
```

```
## A B
## 1 4
```

```
MyMatrix[,1] # First column
```

```
## C D E
## 1 2 3
```

- MyMatrix[,1] will return a vector
- MyMatrix[,1,drop=F] will return a 2-dimension item
- MyMatrix[1:2] will return a 2-dimension item

Data.frame

Similar to matrix, but can contain multiple type of data

```
MyDataFrame<-data.frame("Read_counts"=rpois(4,lambda = 4),
                        "Genes"=c("Cry1" , "Per2" , "Arnt1" , "Cdk4"))
```

MyDataFrame

##	Read_counts	Genes
## 1	1	Cry1
## 2	6	Per2
## 3	7	Arnt1
## 4	3	Cdk4

Again, be carefull to variable type conversion when selecting a single item !

Selecting 1 row kept the data.frame format

```
class(MyDataFrame[1,])
```

```
## [1] "data.frame"
```

but not when selecting 1 column

```
# Same if you use class( MyDataFrame$Read_counts )
class(MyDataFrame[,1])
```

```
## [1] "integer"
```

cbind and rbind

You may want to add new data to your data.frame or matrix, you can use rbind to add new line or cbind to add new row.

```
MyDataFrame
```

##	Read_counts	Genes
## 1	1	Cry1
## 2	6	Per2
## 3	7	Arnt1
## 4	3	Cdk4

```
MyDataFrame<-cbind(MyDataFrame,PassQC=c(T,F,F,T))
MyDataFrame
```

##	Read_counts	Genes	PassQC
## 1	1	Cry1	TRUE
## 2	6	Per2	FALSE
## 3	7	Arnt1	FALSE
## 4	3	Cdk4	TRUE

##	Read_counts	Genes	PassQC
## 1	1	Cry1	TRUE
## 2	6	Per2	FALSE
## 3	7	Arnt1	FALSE
## 4	3	Cdk4	TRUE
## 5	3	P53	TRUE

```
MyDataFrame<-rbind(MyDataFrame,list(3,"P53",T))
MyDataFrame
```


Indexing

To acces certain information, you can in use logical indexing.
e.g. filter low count reads:

MyDataFrame

##	Read_counts	Genes	PassQC
## 1	1	Cry1	TRUE
## 2	6	Per2	FALSE
## 3	7	Arnt1	FALSE
## 4	3	Cdk4	TRUE
## 5	3	P53	TRUE

Here we keep only gene with a minimum of 3 counts

MyDataFrame[MyDataFrame\$Read_counts > 3,]

##	Read_counts	Genes	PassQC
## 2	6	Per2	FALSE
## 3	7	Arnt1	FALSE

You can see the logical index that was given

MyDataFrame\$Read_counts > 3

[1] FALSE TRUE TRUE FALSE FALSE

This is equivalent to give:

MyDataFrame[c(FALSE,TRUE,TRUE,FALSE),]

##	Read_counts	Genes	PassQC
## 2	6	Per2	FALSE
## 3	7	Arnt1	FALSE

Some operator

- ">" more than
- "<" less than
- "==" equal (but see which)
- "!=" not equal

Which functions

The standard operator can be sometime problematic with NA

```
MyGenes<-c("Arnt1"=1.5, "Per2"=1.8, "Cdk4"=8.5, "Cry1"=NA)
MyGenes
```

```
## Arnt1 Per2 Cdk4 Cry1
## 1.5 1.8 8.5 NA
```

it will return NA value

```
MyGenes[MyGenes>2]
```

```
## Cdk4 <NA>
## 8.5 NA
```

you can easily use which

```
MyGenes[which(MyGenes>2)]
```

```
## Cdk4
## 8.5
```

it return the an index which are TRUE

```
which(MyGenes>2)
```

```
## Cdk4
## 3
```

you can also use it for min or max value

- which.min(MyGenes)
- which.max(MyGenes)

The %in% operator

You can return index for item that match multiple requirement

```
MyGenes<-c("Arnt1"=1.5, "Per2"=1.8, "Cdk4"=8.5, "Cry1"=NA)  
MyGenesName<-names(MyGenes)
```

Find the names equal to Arnt1 and Per2

```
idx<-MyGenesName %in% c("Arnt1", "Per2")  
MyGenesName[idx]
```

```
## [1] "Arnt1" "Per2"
```

or genes with specific values

```
idx<-MyGenes %in% c(1.5,8.5)  
MyGenesName[idx]
```

```
## [1] "Arnt1" "Cdk4"
```

Remove item in list

You can remove some item by using the ‘-’ sign

```
MyGenes<-c("Arnt1"=1.5, "Per2"=1.8, "Cdk4"=8.5, "Cry1"=NA)
```

Remove genes with value < 1.8

```
MyGenes[-which(MyGenes<1.8)]
```

Per2 Cdk4 Cry1

1.8 8.5 NA

also remove NA

```
# Less than 1.8 OR value equal NA  
MyGenes[-which(MyGenes<1.8 | is.na(MyGenes))]
```

Per2 Cdk4

1.8 8.5

Transform data

```
as.numeric(c("1", "2", "3"))
```

```
## [1] 1 2 3
```

```
as.character(22:25)
```

```
## [1] "22" "23" "24" "25"
```

Exercise 2