An introduction to R:
Basics of Algorithmics in R

Noémie Becker, Sonja Grath & Dirk Metzler

nbecker@bio.lmu.de - grath@bio.lmu.de

Winter semester 2017-18

1. Back to input files
2. Conditional execution
3. Loops
4. Executing a command from a script

Contents

1 Back to input files

2 Conditional execution

3 Loops

4 Executing a command from a script

Review on data frame

Generic functions:

`read.table()`
`write.table()`

Review on data frame

Generic functions:

`read.table()`
`write.table()`

Example 1:
wghtcls ”smoker” lifespan
”3” 0 50.3
3 0 52.8

Review on data frame

Generic functions:

`read.table()`
`write.table()`

Example 1:
wghtcls ”smoker” lifespan
”3” 0 50.3
3 0 52.8

```
riscfactor <- read.table("lifespan2.txt",header=TRUE)
```

Review on data frame

Example 2:
wghtcls,smoker,lifespan
3,0,50.3
3,0,52.8

Review on data frame

Example 2:
wghtcls,smoker,lifespan
3,0,50.3
3,0,52.8

```
riscfactor <- read.csv("lifespan.csv")
riscfactor <- read.table("lifespan.csv",header=TRUE,
sep=",", fill=TRUE)
```

Review on data frame

Example 2:
wghtcls,smoker,lifespan
3,0,50.3
3,0,52.8

```
riscfactor <- read.csv("lifespan.csv")
riscfactor <- read.table("lifespan.csv",header=TRUE,
sep=",", fill=TRUE)
```

Example 3:
weight class smoker lifespan
3 0 50.3
3 0 52.8

```
riscfactor <- read.table("lifespan3.txt",header=TRUE)
```

Review on data frame

Example 2:
wghtcls,smoker,lifespan
3,0,50.3
3,0,52.8

```
riscfactor <- read.csv("lifespan.csv")
riscfactor <- read.table("lifespan.csv",header=TRUE,
sep=",", fill=TRUE)
```

Example 3:
weight class smoker lifespan
3 0 50.3
3 0 52.8

```
riscfactor <- read.table("lifespan3.txt",header=TRUE)
```

You have to change the first line of the file because of the space between weight and class.

Factors

A variable (numeric or text) can be intended as a factor.

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
```

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
levels(x)
```

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
levels(x)
NULL
```

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
levels(x)
NULL
str(x)
```

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
levels(x)
NULL
str(x)
chr [1:5] "female" "male" "male" "female" "female"
```

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
levels(x)
NULL
str(x)
chr [1:5] "female" "male" "male" "female" "female"
x <-factor(x)
```

Factors

A variable (numeric or text) can be intended as a factor.

Example with text:

```
x <- c("female","male","male","female","female")
levels(x)
NULL
str(x)
chr [1:5] "female" "male" "male" "female" "female"
x <-factor(x)
levels(x)
[1] "female" "male"
str(x)
Factor w/ 2 levels "female","male":  1 2 2 1 1
```

Factors

Example with numbers:

```
y <- rep(c(17,17,18),4); str(y)
num [1:12] 17 17 18 17 17 18 17 17 18 17 ...
```

Factors

Example with numbers:

```
y <- rep(c(17,17,18),4); str(y)
num [1:12] 17 17 18 17 17 18 17 17 18 17 ...
summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  17.00   17.00   17.00   17.33   18.00   18.00
```

Factors

Example with numbers:

```
y <- rep(c(17,17,18),4); str(y)
num [1:12] 17 17 18 17 17 18 17 17 18 17 ...
summary(y)
Min.    1st Qu.     Median    Mean   3rd Qu.     Max.
17.00   17.00       17.00     17.33   18.00    18.00
y <- factor(y); str(y)
Factor w/ 2 levels "17","18":  1 1 2 1 1 2 1 1 2 1 ...
summary(y)
17   18
8     4
```

Back to input files

By default `read.table()` sets text variables as factors and not numerical variables.

Back to input files

By default `read.table()` sets text variables as factors and not numerical variables.
This can be changed by specifying the class of the columns.

```
riscfactor <- read.table("lifespan2.txt",header=TRUE,
colClasses=c("factor","numeric","numeric"))
```

Back to input files

By default `read.table()` sets text variables as factors and not numerical variables.
This can be changed by specifying the class of the columns.

```
riscfactor <- read.table("lifespan2.txt",header=TRUE,
colClasses=c("factor","numeric","numeric"))
```

Or by changing the variables afterwards.

```
riscfactor$wghtcls <- factor(riscfactor$wghtcls)
```

Contents

1 Back to input files

2 Conditional execution

3 Loops

4 Executing a command from a script

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
```

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
4 == 5 #Are both sides equal?
[1] FALSE #FALSE is a constant in R
```

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
4 == 5 #Are both sides equal?
[1] FALSE #FALSE is a constant in R
2 != 3 # !  is negation, != is 'not equal'
```

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
4 == 5 #Are both sides equal?
[1] FALSE #FALSE is a constant in R
2 != 3 # !  is negation, != is 'not equal'
3 != 3
3 <= 5
5 >= 2*2
```

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
4 == 5 #Are both sides equal?
[1] FALSE #FALSE is a constant in R
2 != 3 # !  is negation, != is 'not equal'
3 != 3
3 <= 5
5 >= 2*2
```

Caution:

```
cos(pi/2) == 0
```

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
4 == 5 #Are both sides equal?
[1] FALSE #FALSE is a constant in R
2 != 3 # !  is negation, != is 'not equal'
3 != 3
3 <= 5
5 >= 2*2
```

Caution:

```
cos(pi/2) == 0
[1] FALSE
```

Logic rules in R

```
4 == 4 #Are both sides equal?
[1] TRUE #TRUE is a constant in R
4 == 5 #Are both sides equal?
[1] FALSE #FALSE is a constant in R
2 != 3 # !  is negation, != is 'not equal'
3 != 3
3 <= 5
5 >= 2*2
```

Caution:

```
cos(pi/2) == 0
[1] FALSE
cos(pi/2)
[1] 6.123234e-17
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
[1] FALSE
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
[1] FALSE
TRUE | FALSE # | is the logical OR
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
[1] FALSE
TRUE | FALSE # | is the logical OR
[1] TRUE
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
[1] FALSE
TRUE | FALSE # | is the logical OR
[1] TRUE
5 > 3 & 0 != 1
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
[1] FALSE
TRUE | FALSE # | is the logical OR
[1] TRUE
5 > 3 & 0 != 1
5 > 3 & 0 != 0
```

Logic rules in R

```
TRUE & TRUE # & is the logical AND
[1] TRUE
TRUE & FALSE
[1] FALSE
TRUE | FALSE # | is the logical OR
[1] TRUE

5 > 3 & 0 != 1
5 > 3 & 0 != 0

as.integer(TRUE); as.integer(FALSE)
[1] 1 # the internal representation of TRUE is 1
[1] 0 # the internal representation of FALSE is 0
```

Conditional execution

if(), else() and ifelse()

Conditional execution

if(), else() and ifelse()
Syntax:

```
if ( condition ) { commands1 }
if ( condition ) { commands1 } else { commands2 }
ifelse ( conditions vector, yes vector, no vector )
```

Conditional execution

if(), else() and ifelse()
Syntax:

```
if ( condition ) { commands1 }
if ( condition ) { commands1 } else { commands2 }
ifelse ( conditions vector, yes vector, no vector )
```

Example:

```
x <- 4
if (x==5) {x <- x+1} else {x <- x*2}
```

Conditional execution

if(), else() and ifelse()
Syntax:

```
if ( condition ) { commands1 }
if ( condition ) { commands1 } else { commands2 }
ifelse ( conditions vector, yes vector, no vector )
```

Example:

```
x <- 4
if (x==5) {x <- x+1} else {x <- x*2}
x
[1] 8
```

Organization of the script and indentation

```
x <- 8
if ( x != 5 & x>3 ) {
    x <- x+1
    17+2
} else {
    x <- x*2
    21+5
}
```

Organization of the script and indentation

```
x <- 8
if ( x != 5 & x>3 ) {
    x <- x+1
    17+2
} else {
    x <- x*2
    21+5
}
[1] 19
x
[1] 9
```

Another example

```
T <- TRUE
F <- FALSE
if ( T & F ) {
    print("T & F is TRUE")
} else {
    print("T & F is FALSE")
}
```

Another example

```
T <- TRUE
F <- FALSE
if ( T & F ) {
    print("T & F is TRUE")
} else {
    print("T & F is FALSE")
}
[1] T & F is FALSE
```

Another example

```
T <- TRUE
F <- FALSE
if ( T & F ) {
    print("T & F is TRUE")
} else {
    print("T & F is FALSE")
}
[1] T & F is FALSE

T <- TRUE
F <- FALSE
if ( T | F ) {
    print("T | F is TRUE")
} else {
    print("T | F is FALSE")
}
```

Another example

```
T <- TRUE
F <- FALSE
if ( T & F ) {
    print("T & F is TRUE")
} else {
    print("T & F is FALSE")
}
```

```
[1] T & F is FALSE
```

```
T <- TRUE
F <- FALSE
if ( T | F ) {
    print("T | F is TRUE")
} else {
    print("T | F is FALSE")
}
```

```
[1] T | F is TRUE
```

Example from Day 1

```
Begin
Write "Enter water temperature:"
Read Temp
If Temp ≤ 0 then
    Write "This is ice"
Else then
    If Temp < 100 then
        Write "This is liquid"
    Else then
        Write "This is vapor"
    End of If
End of If
End
```

Example from Day 1

```
Begin
Write "Enter water temperature:"
Read Temp
If Temp ≤ 0 then
    Write "This is ice"
Else then
    If Temp < 100 then
        Write "This is liquid"
    Else then
        Write "This is vapor"
    End of If
End of If
End
```

```
Temp <- readline(prompt="Enter water +
temperature:  ")
if (Temp <= 0) {
    print("This is ice")
} else {
    if (Temp < 100) {
        print("This is liquid")
    } else {
        print("This is vapor")
    }
}
```

Mind the sign for <=

ifelse()

```
y <- 1:10
z <- ifelse( y<6, y^2, y-1 )
```

ifelse()

```
y <- 1:10
z <- ifelse( y<6, y^2, y-1 )
z
[1] 1 4 9 16 25 5 6 7 8 9
```

Contents

1 Back to input files

2 Conditional execution

3 Loops

4 Executing a command from a script

Loops

for(), while() and repeat()

Loops

for(), while() and repeat()

Syntax:

```
for ( var in set ) { commands }
while ( condition ) { commands }
repeat { commands }
```

Loops

for(), while() and repeat()

Syntax:

```
for ( var in set ) { commands }
while ( condition ) { commands }
repeat { commands }
```

`break` stops all loops
`next` goes directly to the next iteration of the loop

Examples

```
x <- 0
for ( i in 1:5 ) { if (i==3) { next } ; x <- x + i }
```

Examples

```
x <- 0
for ( i in 1:5 ) { if (i==3) { next } ; x <- x + i }
 # i=3 is skipped, so x <- 1+2+4+5
x
[1] 12
```

Examples

```
x <- 0
for ( i in 1:5 ) { if (i==3) { next } ; x <- x + i }
 # i=3 is skipped, so x <- 1+2+4+5
x
[1] 12

y <- 1; j <- 1
while ( y < 12 & j < 8 ) { y <- y*2 ; j <- j + 1}
```

Examples

```
x <- 0
for ( i in 1:5 ) { if (i==3) { next } ; x <- x + i }
 # i=3 is skipped, so x <- 1+2+4+5
x
[1] 12

y <- 1; j <- 1
while ( y < 12 & j < 8 ) { y <- y*2 ; j <- j + 1}
y ; j
[1] 16
[1] 5
```

Examples

```
x <- 0
for ( i in 1:5 ) { if (i==3) { next } ; x <- x + i }
 # i=3 is skipped, so x <- 1+2+4+5
x
[1] 12

y <- 1; j <- 1
while ( y < 12 & j < 8 ) { y <- y*2 ; j <- j + 1}
y ; j
[1] 16
[1] 5

z <- 3
repeat { z<- z^2; if ( z>100 ) { break }; print(z)}
```

Examples

```
x <- 0
for ( i in 1:5 ) { if (i==3) { next } ; x <- x + i }
 # i=3 is skipped, so x <- 1+2+4+5
x
[1] 12

y <- 1; j <- 1
while ( y < 12 & j < 8 ) { y <- y*2 ; j <- j + 1}
y ; j
[1] 16
[1] 5

z <- 3
repeat { z<- z^2; if ( z>100 ) { break }; print(z)}
[1] 9
[1] 81
```

The loop stopped after 81^2 so z is 6561.

Contents

1 Back to input files

2 Conditional execution

3 Loops

4 Executing a command from a script

Executing a command from a script

R scripts and stored in .R or .r files and are executed with the command `source()`

Executing a command from a script

R scripts and stored in .R or .r files and are executed with the command `source()`

```
source('C:/Documents/R/myscript.R')
```

Executing a command from a script

R scripts and stored in .R or .r files and are executed with the command `source()`

```
source('C:/Documents/R/myscript.R')
```

You can specify the current working directory using the command `setwd()`

Executing a command from a script

R scripts and stored in .R or .r files and are executed with the command `source()`

```
source('C:/Documents/R/myscript.R')
```

You can specify the current working directory using the command `setwd()`

```
setwd('C:/Documents/R')
getwd()
```

Executing a command from a script

R scripts and stored in .R or .r files and are executed with the command `source()`

```
source('C:/Documents/R/myscript.R')
```

You can specify the current working directory using the command `setwd()`

```
setwd('C:/Documents/R')
getwd()
```

From a command line terminal, you can execute your script directly without opening an R session with

```
Rscript myscript.R
```

Executing a command from a script

R scripts and stored in .R or .r files and are executed with the command `source()`

```
source('C:/Documents/R/myscript.R')
```

You can specify the current working directory using the command `setwd()`

```
setwd('C:/Documents/R')
getwd()
```

From a command line terminal, you can execute your script directly without opening an R session with

```
Rscript myscript.R
```

Tomorrow

To be continued ...