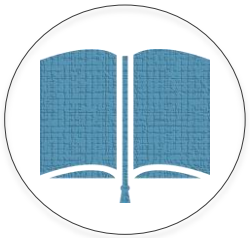
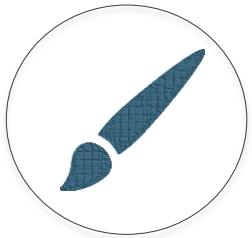




Data Science Using R

Lesson03–Using Functions and Loops in R

Objective

After completing this lesson you will be able to:

- Use the control structures in R
- Create a user defined function in R
- Identify the built in functions in R

Control Structures

Control structures in R allow you to control the flow of execution of the program, depending on runtime conditions. Common structures are:

- *if else — testing a condition*
- *for — execute a loop a fixed number of times*
- *while — execute a loop while a condition is true · repeat: execute an infinite loop*
- *break — break the execution of a loop*
- *next — skip an interaction of a loop*
- *return — exit a function*

Most control structures are not used in interactive sessions but when writing functions or longer expressions.

Control Structures—If Condition

General construct of "if" control structure is given below.

Construct 1:

```
if(<condition>) {
## do something
}
else{
## do something else
}

else clause is not necessary.
if(<condition1>) { }
if(<condition2>) { }
```

Construct 2:

```
if(<condition1>) {
## do something
}
else if(<condition2>) {
## do something different
}
else{
## do something different
}
```

Control Structures—For Loop

Example of "for" control structure is given below.

Example:

```
for(i in 1:10) { print(i)
}
```

The above loop takes the i variable and in each iteration of the loop gives it values 1, 2, 3, ..., 10, and then exits.

Example *with break statement*:

```
for(i in 1:10) {
  print(i)
  if (i==2){
    break
  }}
```

Print i and break as soon as i equals 2

For loops are most commonly used for iterating over the elements of an object (list, vector, etc.)

Control Structures—Nested For Loop

Example of "for" control structure is given below.

Example with nested for loops*:*

```
x <- matrix(1:6, 2, 3)
for(i in seq_len(nrow(x))) {
  for(j in seq_len(ncol(x))) {
    print(x[i, j])
    }
  }
```

*X is a 2*3 matrix. The for loop is being used to print the values of the matrix row wise.*

Be careful with nesting though. Nesting beyond 2–3 levels is often very difficult to read or understand the nested loops.

Control Structures—While Loops

Example of "while" control structure is given below.

Example *with one condition*:

```
count <- 0
while(count < 10) {
    print(count)
    count <- count + 1
    }
```

While loops begin by testing a condition. If it is true, then execute the loop body. Once the loop body is executed, the condition is tested again, and so forth.

Example with multiple condition:

```
z<-5
while(z>=3&&z<=10){
print(z)
coin <- rbinom(1, 1, 0.5)
#generate 0 or 1
if(coin == 1) {
#do not change z }
else{ z<-z-1  }}
```

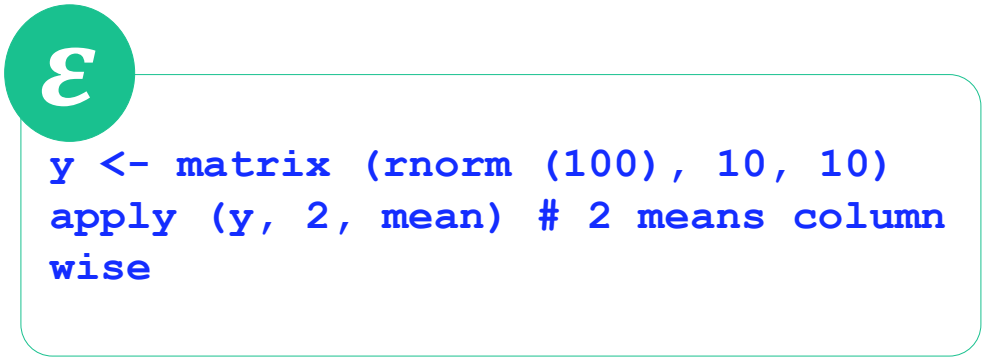
Loop, print z till the condition is satisfied. decrease the value of z when the coin toss results in 0.

While loops can potentially result in infinite loops if not written properly. Use with care. Conditions are always evaluated from left to right.

Loop functions in R

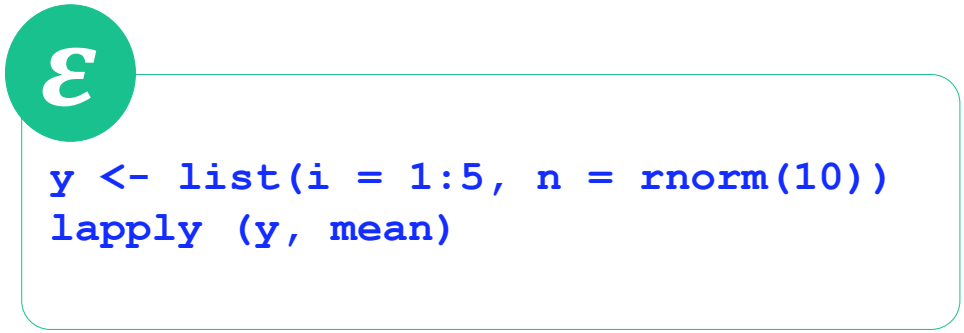
R has inbuilt functions to implement loops. These functions takes away the complexity of writing “for” or “while” loops.

- **Apply function**: Function over the margins of an array



```
y <- matrix (rnorm (100), 10, 10)
apply (y, 2, mean) # 2 means column
wise
```

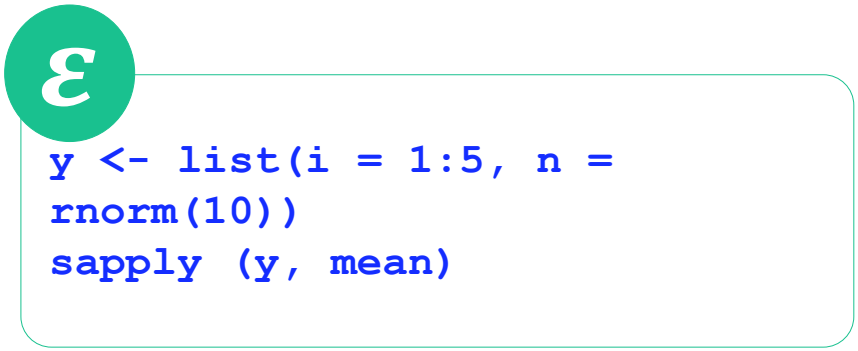
- **Lapply function**: Loop over a list and evaluate a function on each element



```
y <- list(i = 1:5, n = rnorm(10))
lapply (y, mean)
```

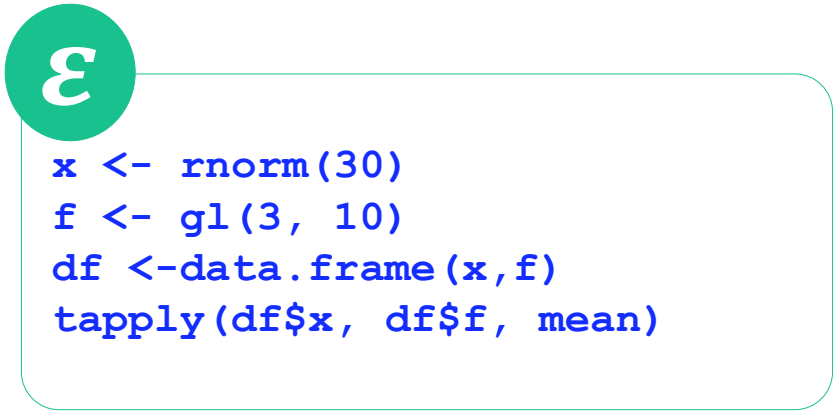
Loop functions in R

- **Sapply function**: Same as lapply but tries to simplify the result



```
y <- list(i = 1:5, n =
rnorm(10))
sapply (y, mean)
```

- **Tapply function**: Apply a function over subsets of a vector



```
x <- rnorm(30)
f <- gl(3, 10)
df <-data.frame(x,f)
tapply(df$x, df$f, mean)
```

- **Mapply function**: Multivariate version of lapply

User Defined Functions

R gives the flexibility of writing custom function.

- Below is a structure of a function followed with example:

Structure of a user defined function:

```
newfunction <- function(arg1, arg2, ... ){
statements
return(object)
}
```

Example of a user defined function:

```
 summarize <- function(x) {
    center <- mean(x); spread <- sd(x)
    cat("Mean is", center, "\n", "Std dev is", spread, "\n")
    result <- list(center=center, spread=spread)
    return(result)
  }
  set.seed(12345)
  x <- rnorm(500)
  y <- summarize(x)
```

R Built-in Functions

- Numeric functions:

Function	Description
abs(x)	absolute value
sqrt(x)	square root
ceiling(x)	ceiling(3.475) is 4
floor(x)	floor(3.475) is 3
trunc(x)	trunc(5.99) is 5
round(x, digits=n)	round(3.475, digits=2) is 3.48
signif(x, digits=n)	signif(3.475, digits=2) is 3.5
cos(x), sin(x), tan(x)	also acos(x), cosh(x), acosh(x), etc.
log(x)	natural logarithm
log10(x)	common logarithm
exp(x)	e^x

R Built-in Functions

- Character functions:

Function	Description
substr(x, start=n1, stop=n2)	Extract or replace substrings in a character vector. x <- "abcdef" substr(x, 2, 4) is "bcd" substr(x, 2, 4) <- "22222" is "a222ef"
grep(pattern, x , ignore.case=FALSE, fixed=FALSE)	Search for pattern in x. If fixed =FALSE then pattern is a regular expression. If fixed=TRUE then pattern is a text string. Returns matching indices. grep("A", c("b","A","c"), fixed=TRUE) returns 2
sub(pattern, replacement, x, ignore.case =FALSE, fixed=FALSE)	Find pattern in x and replace with replacement text. If fixed=FALSE then pattern is a regular expression. If fixed = T then pattern is a text string. sub("\\s",".","Hello There") returns "Hello.There"

R Built-in Functions

- Character functions:

Function	Description
strsplit(x, split)	Split the elements of character vector x at split. strsplit("abc", "") returns 3 element vector "a","b","c"
paste(..., sep="")	Concatenate strings after using sep string to seperate them. paste("x",1:3,sep="") returns c("x1","x2" "x3") paste("x",1:3,sep="M") returns c("xM1","xM2" "xM3") paste("Today is", date())
toupper(x)	Uppercase
tolower(x)	Lowercase

R Built-in Functions

- Stat functions:

Function	Description

dnorm(x)	normal density function (by default m=0 sd=1)
	# plot standard normal curve
	x <- pretty(c(-3,3), 30)
	y <- dnorm(x)
	plot(x, y, type='l', xlab="Normal Deviate",
	ylab="Density", yaxs="i")

pnorm(q)	cumulative normal probability for q (area under the normal curve to the right of q) pnorm(1.96) is 0.975
qnorm(p)	normal quantile. value at the p percentile of normal distribution qnorm(.9) is 1.28 # 90th percentile
rnorm(n, m=0,sd=1)	n random normal deviates with mean m and standard deviation sd. #50 random normal variates with mean=50, sd=10 x <- rnorm(50, m=50, sd=10)

R Built-in Functions

- Stat functions:

Function	Description

dbinom(x, size, prob)
pbinom(q, size, prob)
qbinom(p, size, prob)
rbinom(n, size, prob)
	binomial distribution where size is the sample size and prob is the probability of a heads (pi)
	# prob of 0 to 5 heads of fair coin out of 10 flips
	dbinom(0:5, 10, .5)
	# prob of 5 or less heads of fair coin out of 10 flips
	pbinom(5, 10, .5)

dpois(x, lamda)
ppois(q, lamda)
qpois(p, lamda)
rpois(n, lamda)
	poisson distribution with m=std=lamda
	#probability of 0,1, or 2 events with lamda=4
	dpois(0:2, 4)
	# probability of at least 3 events with lamda=4
	1- ppois(2,4)

dunif(x, min=0, max=1) punif(q, min=0, max=1) qunif(p, min=0, max=1) runif(n, min=0, max=1)	uniform distribution, follows the same pattern as the normal distribution above. #10 uniform random variates x <- runif(10)

R Built-in Functions

- Stat functions:

Function	Description
mean(x, trim=0, na.rm=FALSE)	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x,trim=.05,na.rm=TRUE)
sd(x)	standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(x)	median
mean(x, trim=0, na.rm=FALSE)	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x,trim=.05,na.rm=TRUE)

R Built-in Functions

- Stat functions:

Function	Description
quantile(x, probs)	quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in [0,1]. # 30th and 84th percentiles of x y <- quantile(x, c(.3,.84))
range(x)	range
sum(x)	sum
diff(x, lag=1)	lagged differences, with lag indicating which lag to use
min(x)	minimum
max(x)	maximum

R Built-in Functions

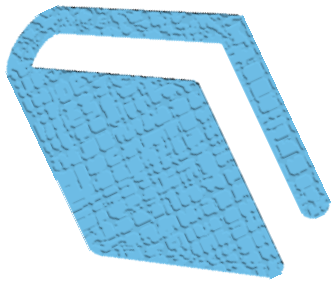
- Stat functions:

Function	Description
scale(x, center=TRUE, scale=TRUE)	column center or standardize a matrix.
seq(from , to, by)	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)
rep(x, ntimes)	repeat x n times y <- rep(1:3, 2) # y is c(1, 2, 3, 1, 2, 3)
cut(x, n)	divide continuous variable in factor with n levels y <- cut(x, 5)

This demo will show the use of concepts covered in this lesson using Rstudio.

Summary

Summary of the topics covered in this lesson:



- If/else, for, while are typical control structures available in R. R also has specific loop functions like apply, tapply, mapply etc. which works exactly like the control structures.
- User defined functions give the flexibility of writing generic functions which can be used to structure a complex code.
- The in-built functions in R gives flexibility to perform complex data manipulations while analyzing datasets.

QUIZ TIME

© Copyright 2015 All rights reserved.

Quiz Question 1

Quiz 1

Which of the following is a loop function in R?

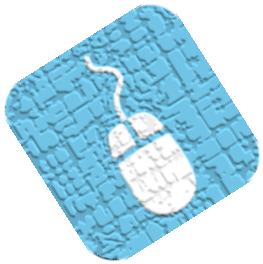
Select all that apply.

a. *apply*

b. *gapply*

c. *tapply*

d. *mapply*



Quiz Question 1

Quiz 1

Which of the following is a loop function in R?

Select all that apply.

a. *apply*

b. *gapply*

c. *tapply*

d. *mapply*

Correct answer is: ***a, c & d***

All the options are correct except b. gapply is not a function in R.

Quiz Question 2

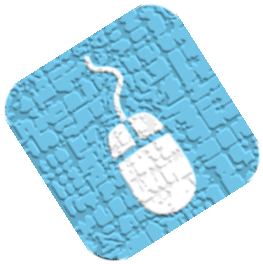
Quiz 2	What is the output of following code? count <- 0 while(count < 10) { count <- count + 1 } print(count)

a. *10*

b. *11*

c. *9*

d. *8*



Quiz Question 2

Quiz 2	What is the output of following code? count <- 0 while(count < 10) { count <- count + 1 } print(count)

a. *10*

b. *11*

c. *9*

d. *8*

Correct answer is: The code when run in R will give 10 as an output.

a

Quiz Question 3

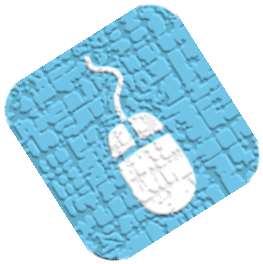
Quiz 3 What will the following code give as output: rnorm(40, 50, 10)

a. *Generate 40 random numbers with a mean of 50 and std. dev of 10*

b. *Generate 50 random numbers with a mean of 40 and std. dev of 10*

c. *Generate 10 random numbers with a mean of 40 and std. dev of 50*

d. *Generate 40 random numbers with a mean of 10 and std. dev of 50*



Quiz Question 3

Quiz 3 What will the following code give as output: rnorm(40, 50, 10)

a. *Generate 40 random numbers with a mean of 50 and std. dev of 10*

b. *Generate 50 random numbers with a mean of 40 and std. dev of 10*

c. *Generate 10 random numbers with a mean of 40 and std. dev of 50*

d. *Generate 40 random numbers with a mean of 10 and std. dev of 50*

Correct answer is: ***a***

First variable is the number of observation, second is the mean and third is std dev.

Quiz Question 4

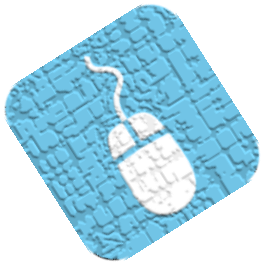
Quiz 4 Which of the following is a function in R?

a. *abs(), sqrt(), sub(), dnorm()*

b. *abs(), sqrt(), subtract(), dnorm()*

c. *abs(), sqrted(), sub(), dnorm()*

d. *abs(), sqrt(), sub(), wnorm()*



Quiz Question 4

Quiz 4 What will the following code give as output: rnorm(40, 50, 10)

a. *abs(), sqrt(), sub(), dnorm()*

b. *abs(), sqrt(), subtract(), dnorm()*

c. *abs(), sqrted(), sub(), dnorm()*

d. *abs(), sqrt(), sub(), wnorm()*

Correct answer is: *a*

Other options have one or more options which are not functions in R

End of Lesson03–Using Loop and Functions in R



R is an official part of the Free Software Foundation's GNU project.
RStudio and Shiny are affiliated projects of the Foundation for Open Access Statistics.