Introduction to Computer Programming with R (FOR 6934)

Qing Zhao (School of Forest Resources & Conservation, qing.zhao@ufl.edu)
Daijiang Li (Department of Wildlife Ecology & Conservation, dli1@ufl.edu)
Denis Valle (School of Forest Resources & Conservation, drvalle@ufl.edu)

Class four

- Operators
- Math & other functions
- Use indexing in functions

Operators

- Arithmetic
- Relational
- Logical

Arithmetic operators

Operator	Description
+	addition
-	subtraction
	multiplication
/	division
ˆ	exponent
%/%	integer division
%%	remainder from division
%*%	matrix multiplication

Relational operators

Operator	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

Logical operators

Operator	Description
!	not
&	and
|	or
%in%	in
:	to

Sample code

Arithmetic operators

```
a <- 21
b <- 7
a + b
```

```
## [1] 28
```

```
a / b
```

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

```
b ^ 2
```

```
## [1] 49
```

Sample code

Relational operators

```
a <- 21
b <- 7
a > b
```

```
## [1] TRUE
```

```
a == b
```

```
## [1] FALSE
```

```
a != b
```

```
## [1] TRUE
```

Sample code

Logical operators

```
b <- 7
x <- 1:5
y <- 6:10
b %in% x
```

```
## [1] FALSE
```

```
b %in% y
```

```
## [1] TRUE
```

Sample code

Use “and” and “or” operators

```r
b <- 7
x <- 1:5
y <- 6:10
b %in% x & b %in% y
```

```
## [1] FALSE
```

```r
b %in% x | b %in% y
```

```
## [1] TRUE
```

Sample code

Combine multiple types of operators

```r
a <- 21
b <- 7
x <- 1:5
(a / b) > b
```

```
## [1] FALSE
```

```r
(a / b) %in% x
```

```
## [1] TRUE
```

Sample code

Use operators in index to find even numbers

```r
g <- 11:20
g
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
g %% 2
```

```
##  [1] 1 0 1 0 1 0 1 0 1 0
```

```r
g %% 2 == 0
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
```

```r
g
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
```

```r
which(g %% 2 == 0)
```

```
## [1]  2  4  6  8 10
```

```r
g[which(g %% 2 == 0)]
```

```
## [1] 12 14 16 18 20
```

Sample code

Arithmetic operators on vectors

```r
m <- 1:4
m
```

```
## [1] 1 2 3 4
```

```r
m + 10
```

```
## [1] 11 12 13 14
```

Sample code

Arithmetic operators on vectors, cont'd

```r
m <- 1:4
n <- c(10,100,1000,10000)
m
```

```
## [1] 1 2 3 4
```

```r
n
```

```
## [1]    10   100  1000 10000
```

```r
m + n
```

```
## [1]    11   102  1003 10004
```

Sample code

Arithmetic operators on vectors, cont'd

```r
m <- 1:8
n <- c(10,100)
m
```

```
## [1] 1 2 3 4 5 6 7 8
```

```r
n
```

```
## [1]  10 100
```

```r
m + n
```

```
## [1]  11 102  13 104  15 106  17 108
```

Sample code

Arithmetic operators on vectors, cont'd

```
m <- 1:8
n <- c(10,100,1000)
m + n
```

```
## Warning in m + n: longer object length is not a multiple of shorter object
## length
```

```
## [1]   11  102 1003   14  105 1006   17  108
```

Sample code

Arithmetic operators on matrices

```
mat <- matrix(1:4, nrow=2, ncol=2)
mat
```

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

```
mat + 10
```

```
##      [,1] [,2]
## [1,]   11   13
## [2,]   12   14
```

Sample code

Arithmetic operators on matrices, cont'd

```
mat
```

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

```
mat + c(10,100)
```

```
##      [,1] [,2]
## [1,]   11   13
## [2,]  102  104
```

Sample code

Arithmetic operators on matrices, cont'd

```
mat
```

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

```
mat * mat
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

Sample code

Arithmetic operators on matrices, cont'd

```
mat
```

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

```
mat %*% mat
```

```
##      [,1] [,2]
## [1,]    7   15
## [2,]   10   22
```

Some comments

- It is recommended to use arithmetic operators on two vectors with equal length or one vector and one scalar
- It is recommended to use arithmetic operators on two matrices with equal size or one matrix and one scalar
- Try to keep the structure of operations simple

This concludes Class 4, Section 1

Please continue on to the next video

Basic math functions

Function	Description
abs()	absolute value
round()	round
sqrt()	square root
log()	logarithm
exp()	exponential

More basic math functions

Function	Description
mean()	mean
sum()	sum
min()	minimum
max()	maximum
range()	range

Sample code

Absolute values

```
a <- c(1, -2, 3, -4, 5)
abs(a)
```

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

```
abs(a - 3)
```

```
## [1] 2 5 0 7 2
```

Sample code

Round

```
a <- c(3.597, 2.283, 5.184)
round(a)
```

```
## [1] 4 2 5
```

```
round(a, digits=2)
```

```
## [1] 3.60 2.28 5.18
```

Sample code

Square root, logarithm, and exponential

```
a <- c(1:6) ^ 2
sqrt(a)
```

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

```
log(a)
```

```
## [1] 0.000000 1.386294 2.197225 2.772589 3.218876 3.583519
```

```
exp(a)
```

```
## [1] 2.718282e+00 5.459815e+01 8.103084e+03 8.886111e+06 7.200490e+10
## [6] 4.311232e+15
```

Sample code

Use math functions on matrices

```
mat <- matrix(c(-1.1, 2.2, -3.3, 4.4), nrow=2, ncol=2)
mat
```

```
##      [,1] [,2]
## [1,] -1.1 -3.3
## [2,]  2.2  4.4
```

```
abs(mat)
```

```
##      [,1] [,2]
## [1,]  1.1  3.3
## [2,]  2.2  4.4
```

Sample code

Mean & sum

```
a <- seq(from=2, to=10, by=2)
mean(a)
```

```
## [1] 6
```

```
sum(a)
```

```
## [1] 30
```

Sample code

Mean and sum of matrices

```
mat <- matrix(1:6, nrow=2, ncol=3)
mean(mat)
```

```
## [1] 3.5
```

```
sum(mat)
```

```
## [1] 21
```

Sample code

Deal with missing values

```
a <- c(1, 2, NA, 4, 5)
mean(a)
```

```
## [1] NA
```

```
mean(a, na.rm=TRUE)
```

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

Sample code

Minimum, maximum and range

```
a <- 1:6
min(a)
```

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

```
max(a)
```

```
## [1] 6
```

```
range(a)
```

```
## [1] 1 6
```

Some other useful functions

Function	Description
seq()	create a sequence of numbers
rep()	repeat the same values

Sample code

Create a sequence of numbers

```r
seq(from=1, to=5, by=1)
```

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

```r
1:5
```

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

Sample code

seq() function is more flexible

```r
seq(from=1.2, to=2.8, by=0.4)
```

```
## [1] 1.2 1.6 2.0 2.4 2.8
```

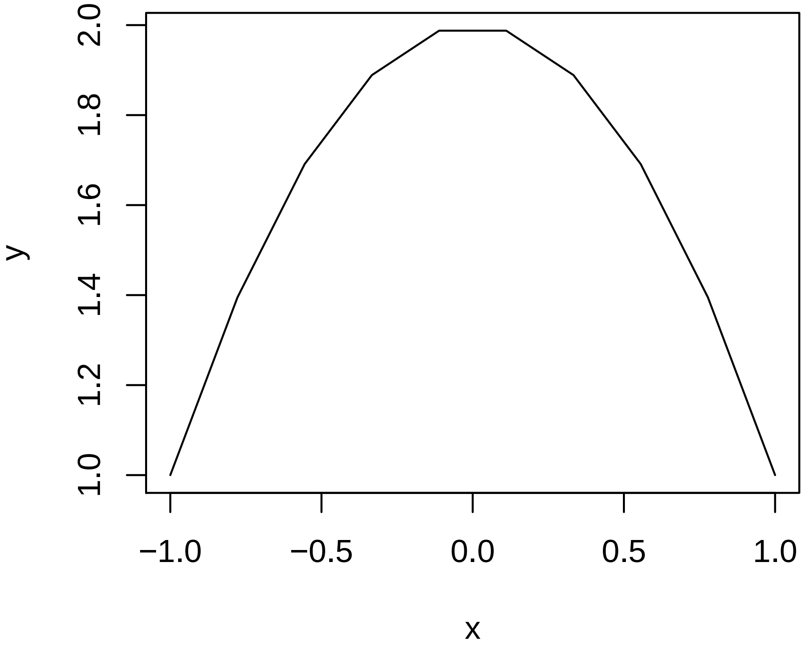
```r
seq(from=1, to=5, length.out=8)
```

```
## [1] 1.000000 1.571429 2.142857 2.714286 3.285714 3.857143 4.428571 5.000000
```

Sample code

Use seq() in graphing

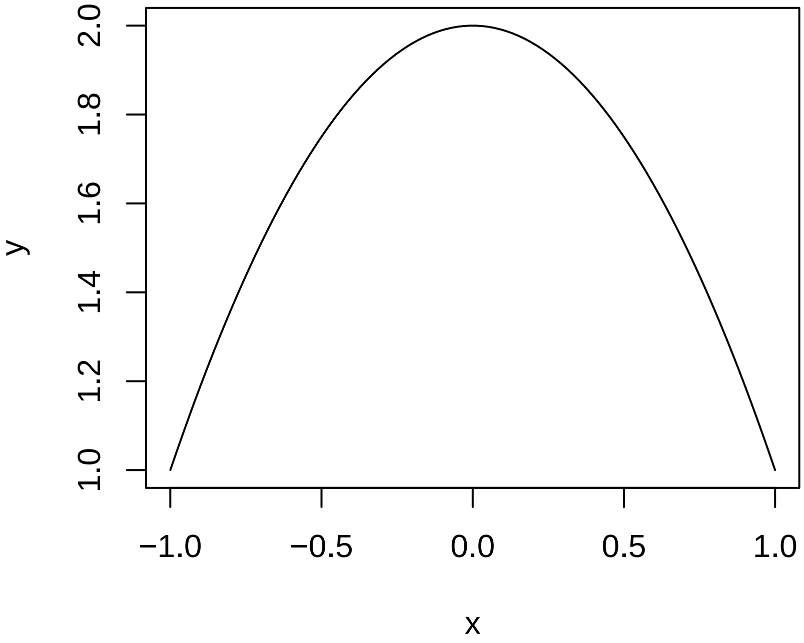
```r
x <- seq(-1, 1, length.out=10)
y <- 2 - x^2
par(mar=c(9,5,1,3))
plot(y ~ x, type='l')
```



Sample code

Make the curve smoother

```
x <- seq(-1, 1, length.out=100)
y <- 2 - x^2
par(mar=c(9,5,1,3))
plot(y ~ x, type='l')
```



Sample code

Create repeated values

```r
rep('a', times=5)
```

```
## [1] "a" "a" "a" "a" "a"
```

Sample code

Repeat a vector

```r
rep(c('a','b','c'), times=3)
```

```
## [1] "a" "b" "c" "a" "b" "c" "a" "b" "c"
```

```r
rep(c('a','b','c'), each=3)
```

```
## [1] "a" "a" "a" "b" "b" "b" "c" "c" "c"
```

This concludes Class 4, Section 2

Please continue on to the next video

Indexing can be used in math functions

- Use indices to select a subset of data
- Use math functions on selected data
- Indexing and math can be done together

Sample code

Read in NBA data

```
dat <- read.csv("c:/data/nba.csv")
head(dat, n=3)
```

```
##                   Team Win Lose Rank Conference
## 1      Boston Celtics  53   29    1    Eastern
## 2 Cleveland Cavaliers  51   31    2    Eastern
## 3     Toronto Raptors  51   31    3    Eastern
```

```
tail(dat, n=3)
```

```
##                        Team Win Lose Rank Conference
## 28 Minnesota Timberwolves   31   51   13    Western
## 29      Los Angeles Lakers  26   56   14    Western
## 30            Phoenix Suns  24   58   15    Western
```

Sample code

Calculate mean without indexing

```
mean(dat$Win)
```

```
## [1] 41
```

Sample code

Select data first, then calculate mean

```
dat_east <- dat[which(dat$Conference == 'Eastern'),]
dat_west <- dat[which(dat$Conference == 'Western'),]
mean(dat_east$Win)
```

```
## [1] 39.6
```

```
mean(dat_west$Win)
```

```
## [1] 42.4
```

Sample code

Select data and calculate mean together

```
mean(dat$Win[which(dat$Conference=='Eastern')])
```

```
## [1] 39.6
```

```
mean(dat$Win[which(dat$Conference=='Western')])
```

```
## [1] 42.4
```

Sample code

Select data and calculate variance together

```
var(dat$Win[which(dat$Conference=='Eastern')])
```

```
## [1] 91.4
```

```
var(dat$Win[which(dat$Conference=='Western')])
```

```
## [1] 163.6857
```

Summary

- Arithmetic, relational and logical operators
- Math functions (e.g. abs(), mean()) and other functions (seq(), rep())
- Math and indexing can be done together

Thank you and see you next class