Advanced Data Analysis 1
Stat 427/527

Chapter 00-2
R building blocks

Erik B. Erhardt

Department of Mathematics and Statistics
MSC01 1115
1 University of New Mexico
Albuquerque, New Mexico, 87131-0001
Office: MSLC 312
erike@stat.unm.edu

Fall 2012

R building blocks

Learning Objectives

R building blocks

After completing this topic, you should be able to:

- **identify** a function or operation and describe its use
- **apply** functions and operations to achieve a specific result
- **predict** answers of calculations written in R

Achieving these goals contributes to mastery in these course learning outcomes:

8. use statistical software

Your turn

R as calculator

What values will R return?

```
> # Arithmetic
> 2 * 10
> 1 + 2
> # Order of operations is preserved
> 1 + 5 * 10
> (1 + 5) * 10
> # Exponents use the ^ symbol
> 2 ^ 5
> 4 ^ (1/2)
```

R as calculator

```
> # Arithmetic
> 2 * 10
[1] 20
> 1 + 2
[1] 3
> # Order of operations is preserved
> 1 + 5 * 10
[1] 51
> (1 + 5) * 10
[1] 60
> # Exponents use the ^ symbol
> 2 ^ 5
[1] 32
> 4 ^ (1/2)
[1] 2
```

Vectors

```
> # Create a vector with the c (short for combine) function
> c(1, 4, 6, 7)
[1] 1 4 6 7
> c(1:5, 10)
[1]  1  2  3  4  5 10
> # or use a function
> # (seq is short for sequence)
> seq(1, 10, by = 2)
[1] 1 3 5 7 9
> seq(0, 50, length = 11)
 [1]  0  5 10 15 20 25 30 35 40 45 50
> seq(1, 50, length = 11)
 [1]  1.0  5.9 10.8 15.7 20.6 25.5 30.4 35.3 40.2 45.1 50.0
> 1:10 # short hand for seq(1, 10, by = 1), or just
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(1, 10)
 [1]  1  2  3  4  5  6  7  8  9 10
> 5:1
[1] 5 4 3 2 1
```

Your turn

Assign variables

What values will R return?

```
> # Assign a vector to a variable with <-
> a <- 1:5
> a
> b <- seq(15, 3, length = 5)
> b
> c <- a*b
> c
```

Assign variables

```
> # Assign a vector to a variable with <-
> a <- 1:5
> a
[1] 1 2 3 4 5
> b <- seq(15, 3, length = 5)
> b
[1] 15 12  9  6  3
> c <- a*b
> c
[1] 15 24 27 24 15
```

Your turn

Basic functions

What values will R return?

```
> a
> sum(a)
> prod(a)
> mean(a)
> sd(a)
> var(a)
> min(a)
> median(a)
> max(a)
> range(a)
```

Basic functions

```
> a
[1] 1 2 3 4 5
> sum(a)
[1] 15
> prod(a)
[1] 120
> mean(a)
[1] 3
> sd(a)
[1] 1.581139
> var(a)
[1] 2.5
> min(a)
[1] 1
> median(a)
[1] 3
> max(a)
[1] 5
> range(a)
[1] 1 5
```

Your turn

Extracting subsets

What values will R return?

```
> # Specify the indices you want in the square brackets []
> a <- seq(0, 100, by = 10)
> a
> a[]
> # integer +=include, 0=include none, -=exclude
> a[5]
> a[c(2, 4, 6, 8)]
> a[0]
> a[-c(2, 4, 6, 8)]
> a[c(1, 1, 1, 6, 6, 9)]
> # subsets can be bigger
> a[c(1,2)] <- c(333, 555) # update a subset
> a
```

Extracting subsets

```
> # Specify the indices you want in the square brackets []
> a <- seq(0, 100, by = 10)
> a
 [1]   0  10  20  30  40  50  60  70  80  90 100
> a[]
 [1]   0  10  20  30  40  50  60  70  80  90 100
> # integer +=include, 0=include none, -=exclude
> a[5]
[1] 40
> a[c(2, 4, 6, 8)]
[1] 10 30 50 70
> a[0]
numeric(0)
> a[-c(2, 4, 6, 8)]
[1]   0  20  40  60  80  90 100
> a[c(1, 1, 1, 6, 6, 9)]
[1]  0  0  0 50 50 80
> # subsets can be bigger
> a[c(1,2)] <- c(333, 555) # update a subset
> a
 [1] 333 555  20  30  40  50  60  70  80  90 100
```

True/False

```
> a
 [1] 333 555  20  30  40  50  60  70  80  90 100
> (a > 50)
 [1]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> a[(a > 50)]
[1] 333 555  60  70  80  90 100
> !(a > 50)        # ! negates (flips) TRUE/FALSE values
 [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
> a[!(a > 50)]
[1] 20 30 40 50
```

Your turn

Comparison functions

What values will R return?

```
> a
> # equal to
> a[(a == 50)]
> # equal to
> a[(a == 55)]
> # not equal to
> a[(a != 50)]
> # greater than
> a[(a > 50)]
> # less than
> a[(a < 50)]
> # less than or equal to
> a[(a <= 50)]
> # which values on left are in the vector on right
> (c(10, 14, 40, 60, 99) %in% a)
```

Comparison functions

```
> a
 [1] 333 555  20  30  40  50  60  70  80  90 100
> # equal to
> a[(a == 50)]
[1] 50
> # equal to
> a[(a == 55)]
numeric(0)
> # not equal to
> a[(a != 50)]
 [1] 333 555  20  30  40  60  70  80  90 100
> # greater than
> a[(a > 50)]
[1] 333 555  60  70  80  90 100
> # less than
> a[(a < 50)]
[1] 20 30 40
> # less than or equal to
> a[(a <= 50)]
[1] 20 30 40 50
> # which values on left are in the vector on right
> (c(10, 14, 40, 60, 99) %in% a)
[1] FALSE FALSE  TRUE  TRUE FALSE
```

Your turn

Boolean operators

What values will R return?

```
> a
> a[(a >= 50) & (a <= 90)]
> a[(a < 50) | (a > 100)]
> a[(a < 50) | !(a > 100)]
> a[(a >= 50) & !(a <= 90)]
```

Boolean operators

```
> a
 [1] 333 555  20  30  40  50  60  70  80  90 100
> a[(a >= 50) & (a <= 90)]
[1] 50 60 70 80 90
> a[(a < 50) | (a > 100)]
[1] 333 555  20  30  40
> a[(a < 50) | !(a > 100)]
[1]  20  30  40  50  60  70  80  90 100
> a[(a >= 50) & !(a <= 90)]
[1] 333 555 100
```

Missing values

```
> # NA (not available) means the value is missing.
> #   Any calculation involving NA will return an NA by
default
> NA + 8
[1] NA
> 3 * NA
[1] NA
> mean(c(1, 2, NA))
[1] NA
> # Many functions have an na.rm argument  (NA remove)
> mean(c(NA, 1, 2), na.rm = TRUE)
[1] 1.5
> sum(c(NA, 1, 2))
[1] NA
> sum(c(NA, 1, 2), na.rm = TRUE)
[1] 3
```

Missing values

```
> # Or you can remove them yourself
> a <- c(NA, 1:5, NA)
> a
[1] NA  1  2  3  4  5 NA
> a[!is.na(a)]
[1] 1 2 3 4 5
> a
[1] NA  1  2  3  4  5 NA
> # To save the results of removing the NAs, reassign
> #   write over variable a and the
> #   previous version is gone forever!
> a <- a[!is.na(a)]
> a
[1] 1 2 3 4 5
```

iClicker

R building blocks 1

CLICKER*Q* 1. What value will R return for z?

```
> x <- 3:7
> y <- x[c(1, 2)] + x[-c(1:3)]
> z <- prod(y)
> z
```

A 99

B 20

C 91

D 54

E NA

R building blocks 1

Answer

```
> x <- 3:7
> x
[1] 3 4 5 6 7
> x[c(1, 2)]
[1] 3 4
> x[-c(1:3)]
[1] 6 7
> y <- x[c(1, 2)] + x[-c(1:3)]
> y
[1]  9 11
> z <- prod(y)
> z
[1] 99
```

iClicker

R building blocks 2

CLICKER*Q* 2. What value will R return for z?

```
> x <- seq(-3, 3, by = 2)
> a <- x[(x > 0)]
> b <- x[(x < 0)]
> z <- a[1] - b[2]
> z
```

A -2

B 0

C 1

D 2

E 6

R building blocks 2

Answer

```
> x <- seq(-3, 3, by = 2)
> x
[1] -3 -1  1  3
> a <- x[(x > 0)]
> a
[1] 1 3
> b <- x[(x < 0)]
> b
[1] -3 -1
> z <- a[1] - b[2]
> z
[1] 2
```

iClicker

R building blocks 3

CLICKER*Q* 3. What value will R return for z?

```
> a <- 2:-3
> b <- a[(a > 0) & (a <= 0)]
> d <- a[!(a > 1) & (a <= -1)]
> z <- sum(c(b,d))
> z
```

E -6

A -3

D 0

B 3

C 6

R building blocks 3

Answer

```
> a <- 2:-3
> a
[1]  2  1  0 -1 -2 -3
> a[(a > 0)]
[1] 2 1
> a[(a <= 0)]
[1]  0 -1 -2 -3
> b <- a[(a > 0) & (a <= 0)]
> b
integer(0)
> a[!(a > 1)]
[1]  1  0 -1 -2 -3
> a[(a <= -1)]
[1] -1 -2 -3
> d <- a[!(a > 1) & (a <= -1)]
> d
[1] -1 -2 -3
> z <- sum(c(b,d))
> z
[1] -6
```

How'd you do?

Outstanding Understanding the operations and how to put them together, without skipping steps.

Good Understanding most of the small steps, missed a couple details.

Hang in there Understanding some of the concepts but all the symbols make my eyes spin.

Reading and writing a new language takes work.
You'll get better as you practice.
Having a buddy to work with will help.

Summary

R commands

```
> 1
> # <-
> # + - * / ^
> # c()
> # seq() # by=, length=
> # sum(), prod(), mean(), sd(), var(),
> # min(), median(), max(), range()
> # a[]
> # (a > 1), ==, !=, >, <, >=, <=, %in%
> # &, |, !
> # NA, mean(a, na.rm = TRUE), !is.na()
> 9
```

Your turn

One-minute paper

Muddy Any “muddy” points — anything that doesn’t make sense yet?

Thumbs up Anything you really enjoyed or feel excited about?