A Brief Introduction to R Data Structures

Jared Harpole

Kansas City R Users Group December, 2014

Outline

Outline

Before We Begin

- If you are new to R take Hadley Wickham's advice

Whenever you're learning a new tool, for a long time you're going to suck ... But the good news is that is typical, that's something that happens to everyone, and it's only temporary.

Overview

Table : Five Common Data Structures in R[1]

	Homogeneous	**Heterogeneous**
1-d	Atomic Vector	List
2-d	Matrix	Data Frame
n-d	Array	

[1] Table from Advanced R by Hadley Wickham

Types of Objects in R

- Elements in your data structures will be of a certain type.
- Types from the least flexible to most flexible are given:
  - logical (e.g. TRUE/FALSE)
  - integer (e.g. 1L)
  - double (e.g. 1.0)
  - character (e.g. "jared")
- If you attempt to combine elements of different types R will coerce

Helper Functions

- typeof() What is this?
- str() What is the structure (SUPER USEFUL!!!)
- length() What is the length
- dim() What are the dimensions
- class() What class is it? (e.g. list, data.frame, etc.)
- If you get stuck remember ?<function> (e.g. ?typeof)

Examples of Some Helper Functions

```
> x <- c(TRUE, TRUE)
> typeof(x)
```

```
[1] "logical"
```

```
> str(x)
```

```
 logi [1:2] TRUE TRUE
```

```
> class(x)
```

```
[1] "logical"
```

```
> length(x)
```

```
[1] 2
```

Outline

The Almighty Vector

- Why are vectors important?
  - Form the foundation of other objects (e.g. matricies and arrays)
  - If you have heard of vectorizing your code this involves using vectors
  - Useful in subsetting with logicals (probably a future topic)
  - Useful in apply functions (probably a future topic)

Vectors Continued

```
> vec1 <- rep(NA, times = 10)
> vec1
```

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

```
> typeof(vec1)
```

```
[1] "logical"
```

```
> vec1[1] <- 1.0
> vec1
```

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

```
> typeof(vec1)
```

```
[1] "double"
```

Vectors Contin...

```
> vec1 <- c(1.0, 2.0, 3.0)
> vec1
```

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

```
> typeof(vec1)
```

```
[1] "double"
```

```
> vec2 <- c(vec1, "3.7")
> vec2
```

```
[1] "1"   "2"   "3"   "3.7"
```

```
> typeof(vec2)
```

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

Outline

The Mammoth Matrix

```
> X <- matrix(data = 1:4, ncol = 2)
> str(X)
```

```
 int [1:2, 1:2] 1 2 3 4
```

```
> class(X)
```

```
[1] "matrix"
```

```
> dim(X)
```

```
[1] 2 2
```

```
> length(X)
```

```
[1] 4
```

Matricies Contin..

```
> X <- matrix(data = 1:4, ncol = 2, byrow =
    TRUE)
> X
```

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

```
> X <- cbind(X, c("100", "200"))
> X
```

```
     [,1] [,2] [,3]
[1,] "1"  "2"  "100"
[2,] "3"  "4"  "200"
```

Matricies Contin..

```
> X <- matrix(data = 1:4, ncol = 2, byrow =
    TRUE)
> X
```

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

```
> X[1,2] <- 100
> X[1,]
```

```
[1]   1 100
```

Arrays

```
> Z <- array(data = 1:27, dim = c(3, 3, 3))
> str(Z)
```

```
 int [1:3, 1:3, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
```

```
> dim(Z)
```

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

```
> class(Z)
```

```
[1] "array"
```

Arrays Contin..

```
> Z <- array(data = 1:27, dim = c(3, 3, 3))
> Z
```

```
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
```

Outline

Heterogeneous Structures: Lists

- Lists are a 1-d object that can store heterogenous data types (like a python list)
- Lists are extremely flexible and many things in R are lists under the hood
- Often very convenient if you need to store multiple objects of different lengths and/or types

Lists Contin...

```
> test_list <- list(a = 1:3, b = letters[1:5],
    c = FALSE)
> str(test_list)
```

```
List of 3
 $ a: int [1:3] 1 2 3
 $ b: chr [1:5] "a" "b" "c" "d" ...
 $ c: logi FALSE
```

```
> sapply(test_list, class)
```

```
        a         b         c
"integer" "character"   "logical"
```

Lists Contin...

```
> new_list <- list(a = list(b = list(c = 'hi')
   ))
> str(new_list)
```

```
List of 1
 $ a:List of 1
  ..$ b:List of 1
  .. ..$ c: chr "hi"
```

```
> new_list
```

```
$a
$a$b
$a$b$c
[1] "hi"
```

Lists Contin...

```
> library(car)
> regmod <- lm(mpg ~ disp + hp, data = mtcars)
> str(regmod)
```

```
List of 12
 $ coefficients : Named num [1:3] 30.7359
     -0.0303 -0.0248
  ..- attr(*, "names")= chr [1:3] "(Intercept)
     " "disp" "hp"
 $ residuals    : Named num [1:32] -2.15 -2.15
     -2.35 1.23 3.24 ...
  ..- attr(*, "names")= chr [1:32] "Mazda RX4"
     "Mazda RX4 Wag" "Datsun 710" "Hornet 4
     Drive" ...
 $ effects      : Named num [1:32] -113.65
     -28.44 5.8 1.1 3.01 ...
```

Outline

The All Important Data Frame

- A data frame is the most common way of storing data in R and is critical for statistics, data mining, and machine learning
- Under the hood a data frame is a list of equal length vectors making it a 2-d structure
- Has the flavor of a matrix with the flexibility of a list
- When you read in data to R you are probably storing this as a data frame initially

Data Frames Contin...

```
> dat <- data.frame(V1 = 1:4, V2 = letters
    [3:6], V3 = gl(n = 2, k = 2, labels = c("M"
    , "F")), stringsAsFactors = FALSE)
> str(dat)
```

```
'data.frame': 4 obs. of  3 variables:
 $ V1: int  1 2 3 4
 $ V2: chr  "c" "d" "e" "f"
 $ V3: Factor w/ 2 levels "M","F": 1 1 2 2
```

```
> dim(dat)
```

```
[1] 4 3
```

Data Frames Contin...

```
> dat <- data.frame(V1 = 1:4, V2 = letters
    [3:6], V3 = gl(n = 2, k = 2, labels = c("M"
    , "F")), stringsAsFactors = FALSE)
> dat
```

```
  V1 V2 V3
1  1  c  M
2  2  d  M
3  3  e  F
4  4  f  F
```

```
> sapply(dat, class)
```

```
        V1         V2         V3
"integer" "character"   "factor"
```

Data Frames Contin...

```
> dat <- data.frame(V1 = 1:4, V2 = letters
    [3:6], V3 = gl(n = 2, k = 2, labels = c("M"
    , "F")), stringsAsFactors = FALSE)
> dat
```

```
  V1 V2 V3
1  1  c  M
2  2  d  M
3  3  e  F
4  4  f  F
```

```
> sapply(dat, mean)
```

```
 V1  V2  V3
2.5  NA  NA
```

Questions