Getting Started and R Nuts and Bolts

Nathan Young

Last updated: April 11, 2021

Contents

Vectors and Lists

```r
x <- c(0.5, 0.6) ## numeric
x <- c(TRUE, FALSE) ## logical
x <- c(T, F) ## logical
x <- c("a", "b", "c") ## character
x <- 9:29 ## integer
x <- c(1 + 0i, 2 + 4i) ## complex
```

Creating Vectors Using the `vector()` function:

```r
x <- vector("numeric", length = 10)
x
```

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

```r
y <- c(1.7, "a") ## character
y <- c(TRUE, 2) ## numeric
y <- c("a", TRUE) ## character
```

Mixing Objects

Explicit Coercion Objects can be explicity coerced from one class to another using the `as.*` functions, if available.

```
x <- 0:6
class(x)
```

```
## [1] "integer"
```

```
as.numeric(x)
```

```
## [1] 0 1 2 3 4 5 6
```

```
as.logical(x)
```

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

```
as.character(x)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6"
```

Nonsensical coercion results in NAs.

```
x <- c("a", "b", "c")
as.numeric(x)
```

```
## Warning: NAs introduced by coercion
```

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

```
as.logical(x)
```

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

```
as.complex(x)
```

```
## Warning: NAs introduced by coercion
```

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

Lists Lists are a special type of vector that can contain elements of different classes.

```
x <- list(1, "a", TRUE, 1 + 4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
```

Matrices

Vectors with a *dimension* attribute.

Matrices are constructed *column-wise*, so entries can be thought of starting in the top left corner and running down columns.

```
m <- matrix(nrow = 2, ncol = 3)
m
```

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

```
dim(m)
```

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

```
attributes(m)
```

```
## $dim
## [1] 2 3
```

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

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

```
m <- 1:10
m
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
dim(m) <- c(2, 5)
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

Matrices can be created by column-binding (`cbind()`) or row-binding (`rbind()`).

```
x <- 1:3
y <- 10:12
cbind(x, y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```
rbind(x, y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

Factors

Used to represent categorical data. Can by unordered or ordered. An integer vector where each integer has a label.

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
```

```
## [1] yes yes no  yes no 
## Levels: no yes
```

```
table(x)
```

```
## x
##  no yes 
##   2   3
```

```
unclass(x)
```

```
## [1] 2 2 1 2 1
## attr(,"levels")
## [1] "no"  "yes"
```

```
x <- factor(c("yes", "yes", "no", "yes", "no"),
            levels = c("yes", "no"))
x
```

```
## [1] yes yes no  yes no 
## Levels: yes no
```

Missing Values

Missing values are denoted by `NA` or `NaN` for undefined math operations.

```
x <- c(1, 2, NA, 10, 3)
is.na(x)
```

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

```
is.nan(x)
```

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

```r
x <- c(1, 2, NaN, NA, 4)
is.na(x)
```

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

```r
is.nan(x)
```

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

Data Frames

Used to store tabular data.

```r
x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
x
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```r
nrow(x)
```

```
## [1] 4
```

```r
ncol(x)
```

```
## [1] 2
```

Names Attribute

Objects can have names.

```r
x <- 1:3
names(x)
```

```
## NULL
```

```r
names(x) <- c("foo", "bar", "norf")
x
```

```
##  foo  bar norf
##    1    2    3
```

```
names(x)
```

```
## [1] "foo"  "bar"  "norf"
```

Lists can also have names.

```
x <- list(a = 1, b = 2, c = 3)
x
```

```
## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## [1] 3
```

Matrix names are called `dim names`.

```
m <- matrix(1:4, nrow = 2, ncol = 2)
dimnames(m) <- list(c("a", "b"), c("c", "d"))
m
```

```
##   c d
## a 1 3
## b 2 4
```

read.table

Reading Data One of the most commonly used functions for reading data is `read.table`.

Reading Large Data To make reading large data with `read.table` easier:

- Read the docs
- If data set is larger than RAM, stop.
- Set `comment.char = ""` if there are no commented lines.
- Use `colClasses` command instead of letting R figure data type of column.
- Set `nrows`