Introducing vectors

Mark Andrews

April 5, 2017

Introduction

Vectors are simply arrays or lists of numbers. Most of time, when we are working with data, we work with *data frames*. Data frames can be seen as similar to spreadsheets, i.e. with multiple rows and multiple columns, and each column representing a variable. Here, each column is a vector and often we need to work directly with it.

We'll start by opening up a data set. This data set has one column, called x and we'll save this as a stand alone vector, also called x.

```
Df <- read.csv('../data/foo.csv', header=TRUE)
(x <- Df$x) # The enclosing parentheses allow us to view the vector, and assign it to x
```

```
##   [1] 18 11 19 12 11 16 14 11 19 18 16 19 13 12 20 20 19 13 14 17 17 14 13
##  [24] 15 18 19 13 18 13 18 14 12 19 12 13 19 13 14 16 12 17 11 11 20 15 19
##  [47] 19 12 13 19 13 13 19 15 16 19 19 19 15 17 17 13 18 16 13 18 19 13 18
##  [70] 11 19 17 13 13 15 14 20 11 14 19 11 14 12 13 18 11 18 18 14 11 18 18
##  [93] 14 19 11 18 17 18 19 11
```

Let's examine the vector

```
class(x) # What kind of object is it?
```

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

```
length(x)
```

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

```
str(x)   # compactly display internal structure of x
```

```
##  int [1:100] 18 11 19 12 11 16 14 11 19 18 ...
```

```
summary(x) # summarize the info in x
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   13.00   15.50   15.49   18.00   20.00
```

Indexing

- What is the values of elements 68?

```
x[68]
```

```
## [1] 13
```

Slicing

Slicing will give us a contiguous subset of the vector. For example, what are the values of elements 12 to 18 inclusive?

```
x[12:18]
```

```
## [1] 19 13 12 20 20 19 13
```

Multiple indices and subsets

What are the values of elements 17, 89, 39, 42? To do this, first create a vector of those elements and call it *indx* and then use this to slice x

```
indx <- c(17, 89, 39, 42)
x[indx]
```

```
## [1] 19 14 16 11
```

This is the same thing as doing

```
x[c(17, 89, 39, 42)]
```

```
## [1] 19 14 16 11
```

What if we needed to find all elements that are equal to or less than the value of 12? Here, we can create an indexing vector called *indx* and then use this to extract the elements:

```
(indx <- which(x <= 12)) # Get indices of elements whose values are = or < 12.
```

```
##  [1]   2   4   5   8  14  32  34  40  42  43  48  70  78  81  83  86  90
## [18]  95 100
```

```
x[indx]
```

```
##  [1] 11 12 11 11 12 12 12 12 11 11 12 11 11 11 12 11 11 11 11
```

We also use Boolean indices here:

```
(indx <- x <= 12)
```

```
##   [1] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [12] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [34]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
##  [45] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78]  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
##  [89] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [100]  TRUE
```

```
x[indx]
```

```
##  [1] 11 12 11 11 12 12 12 12 11 11 12 11 11 11 12 11 11 11 11
```

Note that to find how many elements are equal to e.g. 12, we can do

```
indx <- x == 12
sum(indx) # Sums up the Boolean vector
```

```
## [1] 7
```

which gives the same result as

```
indx <- which(x == 12)
length(indx)
```

```
## [1] 7
```

Warning: Remember that <= is an inequality test, and <- is an assignment operator, and == is an equality test and = is an assignment operator.

We can use Boolean operators to do more interesting subsetting operations:

```
indx <- x == 11 | x > 17 # elements that equal to 11 OR greater than 17
x[indx]
```

```
##  [1] 18 11 19 11 11 19 18 19 20 20 19 18 19 18 18 19 19 11 11 20 19 19 19
## [24] 19 19 19 19 18 18 19 18 11 19 20 11 19 11 18 11 18 18 11 18 18 19 11
## [47] 18 18 19 11
```

```
indx <- x == 11 | x == 13 | x == 17 #  elements that are equal to 11 or 13 or 17
x[indx]
```

```
##  [1] 11 11 11 13 13 17 17 13 13 13 13 13 17 11 11 13 13 13 17 17 13 13 13
## [24] 11 17 13 13 11 11 13 11 11 11 17 11
```

```
indx <- x %in% c(11, 13, 17) # Same as above, i.e. values that are elements of set 11, 13, 17
x[indx]
```

```
##  [1] 11 11 11 13 13 17 17 13 13 13 13 13 17 11 11 13 13 13 17 17 13 13 13
## [24] 11 17 13 13 11 11 13 11 11 11 17 11
```

Descriptive statistics

We can easily get things like *mean*, *median*, *sd*, etc, etc.

```
mean(x)
```

```
## [1] 15.49
```

```
median(x)
```

```
## [1] 15.5
```

```
sd(x)
```

```
## [1] 2.976321
```

```
var(x)
```

```
## [1] 8.858485
```

```
min(x)
```

```
## [1] 11
```

```
max(x)
```

```
## [1] 20
```

```
range(x)
```

```
## [1] 11 20
```

```
IQR(x) # inter quartile range
```

```
## [1] 5
```

This will give us some standard percentiles,

```
quantile(x)
```

```
##   0%  25%  50%  75% 100%
## 11.0 13.0 15.5 18.0 20.0
```

and we can ask for specific percentiles too:

```
quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975))
```

```
##  2.5%   25%   50%   75% 97.5%
##  11.0  13.0  15.5  18.0  20.0
```

This will do a frequency tabulation of the values of `x`.

```
table(x)
```

```
## x
## 11 12 13 14 15 16 17 18 19 20
## 12  7 16 10  5  5  7 15 19  4
```

Concatenating vectors

We can join up vectors using the generic "combine" function *c()*:

```
z <- c(1, 2, 3, 27, 42)
(y <- c(x, z))
```

```
##   [1] 18 11 19 12 11 16 14 11 19 18 16 19 13 12 20 20 19 13 14 17 17 14 13
##  [24] 15 18 19 13 18 13 18 14 12 19 12 13 19 13 14 16 12 17 11 11 20 15 19
##  [47] 19 12 13 19 13 13 19 15 16 19 19 19 15 17 17 13 18 16 13 18 19 13 18
##  [70] 11 19 17 13 13 15 14 20 11 14 19 11 14 12 13 18 11 18 18 14 11 18 18
##  [93] 14 19 11 18 17 18 19 11  1  2  3 27 42
```