Chapter 16: Numeric classes and storage modes

Section 16.1: Numeric

Numeric represents integers and doubles and is the default mode assigned to vectors of numbers. The function `is.numeric()` will evaluate whether a vector is numeric. It is important to note that although integers and doubles will pass `is.numeric()`, the function `as.numeric()` will always attempt to convert to type double.

```
x <- 12.3
y <- 12L

#confirm types
typeof(x)
[1] "double"
typeof(y)
[1] "integer"

# confirm both numeric
is.numeric(x)
[1] TRUE
is.numeric(y)
[1] TRUE

# logical to numeric
as.numeric(TRUE)
[1] 1

# While TRUE == 1, it is a double and not an integer
is.integer(as.numeric(TRUE))
[1] FALSE
```

Doubles are R's default numeric value. They are double precision vectors, meaning that they take up 8 bytes of memory for each value in the vector. R has no single precision data type and so all real numbers are stored in the double precision format.

```
is.double(1)
TRUE
is.double(1.0)
TRUE
is.double(1L)
FALSE
```

Integers are whole numbers that can be written without a fractional component. Integers are represented by a number with an L after it. Any number without an L after it will be considered a double.

```
typeof(1)
[1] "double"
class(1)
[1] "numeric"
typeof(1L)
[1] "integer"
class(1L)
[1] "integer"
```

Though in most cases using an integer or double will not matter, sometimes replacing doubles with integers will

consume less memory and operational time. A double vector uses 8 bytes per element while an integer vector uses only 4 bytes per element. As the size of vectors increases, using proper types can dramatically speed up processes.

```
#  test speed on lots of arithmetic
microbenchmark(
  for( i in 1:100000){
  2L * i
  10L + i
},

for( i in 1:100000){
  2.0 * i
  10.0 + i
}
)
Unit: milliseconds
                                          expr      min       lq     mean   median       uq
 max neval
 for (i in 1:1e+05) {     2L * i     10L + i } 40.74775 42.34747 50.70543 42.99120 65.46864
94.11804   100
   for (i in 1:1e+05) {     2 * i     10 + i } 41.07807 42.38358 53.52588 44.26364 65.84971
83.00456   100
```