Basic Operators

```
library(tidyverse)
library(mosaic)

# Arithmetic Operators:
# addition: +
# negation/subtraction: -
# multiplication: *
# division: /
# exponentiation: ^
#
# These work on vectors.  Here are some examples:
3 + 7

## [1] 10

-c(2,3,5)

## [1] -2 -3 -5

c(0,1,10)*c(2,3,5)

## [1]  0  3 50

c(0,1,10)/c(2,3,5)

## [1] 0.0000000 0.3333333 2.0000000

c(2,3,4)^c(3,2,1/2)

## [1] 8 9 2

# Addition and negation/subtraction work on two matrices of the same
# size.  Here's an example:
a <- matrix(c(1,2,3,4,5,6),2,3,byrow = TRUE)
a

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

b <- matrix(c(1,-2,4,-5,2,3),2,3,byrow = TRUE)
b

##      [,1] [,2] [,3]
## [1,]    1   -2    4
## [2,]   -5    2    3

a + b
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    7
## [2,]   -1    7    9
```

```
a - b
```

```
##      [,1] [,2] [,3]
## [1,]    0    4   -1
## [2,]    9    3    3
```

```
-b
```

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

```
# We wonder, can we multiply a matrix by a scalar (number)?
# Let's try it.
3*b
```

```
##      [,1] [,2] [,3]
## [1,]    3   -6   12
## [2,]  -15    6    9
```

```
# Yes, that works!  In fact, we can always combine any vector with
# a vector of length 1.  For example:
c(2,3,5)^2
```

```
## [1]  4  9 25
```

```
2^c(2,3,5)
```

```
## [1]  4  8 32
```

```
# Two more arithmetic operators:
# integer division: %/% - returns the quotient
#                   %%  - returns the remainder
# Example:
dividend <- 31
divisor <- 7
quotient <- dividend %/% divisor
remainder <- dividend %% divisor
c(quotient,remainder)
```

```
## [1] 4 3
```

```
# Let's check the type of quotient and remainder.
class(c(quotient,remainder))
```

```
## [1] "numeric"
```

```r
# Exercise: Explore what happens if the dividend or the divisor are
# not whole numbers.  For example:
c(7 %/% 2.2, 7 %% 2.2)
```

```
## [1] 3.0 0.4
```

```r
# In some programming languages, the quotient and remainder operators
# do not work correctly in all cases.  Techically, for integers
# m and n with n not 0, if q = m %/% n and r = m %% n, then it should
# be the case that m = n*q + r and 0 <= r < |n| (in words,
# dividend = divisor times quotient plus remainder, with the remainder
# being nonnegative and less than the absolute value of the divisor).
# For example, if we divide -31 by 7, we should get a quotient of -5
# and a remainder of 4,  Let's see:
c(-31 %/% 7, -31 %% 7)
```

```
## [1] -5  4
```

```r
# Great, that works!
# What if we divide 31 by -7?  We should get a quotient of -4 and a
# remainder of 3.  Let's see:
c(31 %/% -7, 31 %% -7)
```

```
## [1] -5 -4
```

```r
# Aha!  Just as I feared, R computes a negative remainder.
# Let's try dividing -31 by -7.  We should get a quotient of 5 and
# a remainder of 4.
c(-31 %/% -7, -31 %% -7)
```

```
## [1]  4 -3
```

```r
# Again, R computes a negative remainder.
#
# Exercise: Write a function, integer_divide, so that
# integer_divide(m,n) returns (as a vector) the quotient and
# remainder when the integer m is divided by the nonzero integer n.
#
# Recall that, if A in an m by p matrix and B is a p by n matrix,
# then AB (the product of A with B) is the m by n matrix defined by
# (AB)[i,j] = the dot product of the ith row of A and the jth column
# of B.  The operator %*% is used for matrix multiplication.
# Example: A is a 2 by 3 matrix:
a <- matrix(c(2,0,3,-1,4,2),2,3,byrow = TRUE)
a
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    3
## [2,]   -1    4    2
```

```
# B is a 3 by 2 matrix:
b <- matrix(c(3,8,5,1,-1,2),3,2,byrow = TRUE)
b

##      [,1] [,2]
## [1,]    3    8
## [2,]    5    1
## [3,]   -1    2

# AB is a 2 by 2 matrix:
a %*% b

##      [,1] [,2]
## [1,]    3   22
## [2,]   15    0

# BA is a 3 by 3 matrix:
b %*% a

##      [,1] [,2] [,3]
## [1,]   -2   32   25
## [2,]    9    4   17
## [3,]   -4    8    1

# Relational Operators:
# less than: <
# less than or equal to: <=
# equal to: ==
# greater than or equal to: >=
# greater than: >
# not equal to: !=
# Each of these operators may be placed between two vectors with
# the same length.  The result is a logical vector with that length.
# Examples:
5 < 7

## [1] TRUE

c(13,17,19) == c(19,17,13)

## [1] FALSE  TRUE FALSE

# Strings are ordered as in a dictionary.
c("BANANA","PEACH") >= c("APPLE","PEAR")

## [1]  TRUE FALSE

c(13,17,19) != c(19,17,13)

## [1]  TRUE FALSE  TRUE

# Logical Operators:
# and: &
```

```r
# or: |
# not: !
# Examples:
(2 < 3) & (7 > 5)

## [1] TRUE

(2 > 3) | (7 > 5)

## [1] TRUE

!(2 < 3)

## [1] FALSE

# Assignment Operators: As seen, we prefer to use <- for assignment.
# However, = may also be used.
# Example:
some_primes = c(11L, 13L, 17L, 19L)
some_primes

## [1] 11 13 17 19

# Miscellaneous Operators:
# The operator : is used to produce a sequence.
# The operator %in% tests for membership.
# Examples:
octal_digits <- 0:7
octal_digits

## [1] 0 1 2 3 4 5 6 7

4 %in% octal_digits

## [1] TRUE

9 %in% octal_digits

## [1] FALSE

# We wonder what this will do.
try_it <- 3.2:7.2
try_it

## [1] 3.2 4.2 5.2 6.2 7.2

another_try <- 7.2:3.2
another_try

## [1] 7.2 6.2 5.2 4.2 3.2

# Interesting!
flag_colors <- c("red","white","blue")
flag_colors
```

```
## [1] "red"   "white" "blue"
```

```
"green" %in% flag_colors
```

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

```
# Exercises:
# Is there an operator for raising a square matrix to a power?
# Is there an operator for computing the inverse of a square matrix?
# Is there an operator for computing the transpose of a matrix?
```