Vector and Matrix Algebra

Lecture notes

Packages for this section

- This is (almost) all base R! We only need this for one thing later:

```
library(tidyverse)
```

Vector addition

Adds 2 to each element.

- Adding vectors:

```
u <- c(2, 3, 6, 5, 7)
v <- c(1, 8, 3, 2, 0)
u + v
```

```
## [1]  3 11  9  7  7
```

- Elementwise addition. (Linear algebra: vector addition.)

Adding a number to a vector

- Define a vector, then "add 2" to it:

```
u
```

```
## [1] 2 3 6 5 7
```

```
k <- 2
u + k
```

```
## [1] 4 5 8 7 9
```

- adds 2 to *each* element of `u`.

Scalar multiplication

As per linear algebra:

```
k
```

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

```
u
```

```
## [1] 2 3 6 5 7
```

```
k * u
```

```
## [1]  4  6 12 10 14
```

- Each element of vector multiplied by 2.

"Vector multiplication"

What about this?

```
u
```

```
## [1] 2 3 6 5 7
```

```
v
```

```
## [1] 1 8 3 2 0
```

```
u * v
```

```
## [1]  2 24 18 10  0
```

Each element of `u` multiplied by *corresponding* element of `v`. Could be called elementwise multiplication.

(Don't confuse with "outer" or "vector" product from linear algebra, or indeed "inner" or "scalar" multiplication, for which the answer is a number.)

Combining different-length vectors

- No error here (you get a warning). What happens?

```
u
```

```
## [1] 2 3 6 5 7
```

```
w <- c(1, 2)
u + w
```

```
## Warning in u + w: longer object length is not a
## multiple of shorter object length
```

```
## [1] 3 5 7 7 8
```

- Add 1 to first element of `u`, add 2 to second.
- Go back to beginning of `w` to find something to add: add 1 to 3rd element of `u`, 2 to 4th element, 1 to 5th.

How R does this

- Keep re-using shorter vector until reach length of longer one.
- "Recycling".
- If the longer vector's length not a multiple of the shorter vector's length, get a warning (probably not what you want).
- Same idea is used when multiplying a vector by a number: the number keeps getting recycled.

Matrices

- Create matrix like this:

```
(A <- matrix(1:4, nrow = 2, ncol = 2))
```

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

- First: stuff to make matrix from, then how many rows and columns.
- R goes down columns by default. To go along rows instead:

```
(B <- matrix(5:8, nrow = 2, ncol = 2, byrow = T))
```

```
##      [,1] [,2]
## [1,]    5    6
## [2,]    7    8
```

- One of `nrow` and `ncol` enough, since R knows how many things in the matrix.

Adding matrices

What happens if you add two matrices?

```
A
```

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

```
B
```

```
##      [,1] [,2]
## [1,]    5    6
## [2,]    7    8
```

```
A + B
```

```
##      [,1] [,2]
## [1,]    6    9
## [2,]    9   12
```

Adding matrices

- Nothing surprising here. This is matrix addition as we and linear algebra know it.

Multiplying matrices

- Now, what happens here?

```
A
```

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

```
B
```

```
##      [,1] [,2]
## [1,]    5    6
## [2,]    7    8
```

```
A * B
```

```
##      [,1] [,2]
## [1,]    5   18
## [2,]   14   32
```

Multiplying matrices?

- *Not* matrix multiplication (as per linear algebra).
- Elementwise multiplication. Also called *Hadamard product* of `A` and `B`.

Legit matrix multiplication

Like this:

```
A
```

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

```
B
```

```
##      [,1] [,2]
## [1,]    5    6
## [2,]    7    8
```

```
A %*% B
```

```
##      [,1] [,2]
## [1,]   26   30
## [2,]   38   44
```

Reading matrix from file

- The usual:

```
my_url <- "http://www.utsc.utoronto.ca/~butler/c32/m.txt"
M <- read_delim(my_url, " ", col_names = F)
```

```
##
## -- Column specification --------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
```

```
class(M)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"
## [4] "data.frame"
```

but...

- except that M is not an R matrix, and thus this doesn't work:

```
v <- c(1, 3)
M %*% v
```

```
## Error in M %*% v: requires numeric/complex matrix/vector ar
```

Making a genuine matrix

Do this first:

```
M <- as.matrix(M)
```

and then all is good:

```
M %*% v
```

```
##      [,1]
## [1,]   37
## [2,]   29
## [3,]   21
```

Linear algebra stuff

- To solve system of equations $Ax = w$ for x:

```
A
```

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

```
w
```

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

```
solve(A, w)
```

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

Matrix inverse

- To find the inverse of A:

```
A
```

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

```
solve(A)
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

- You can check that the matrix inverse and equation solution are correct.

Inner product

- Vectors in R are column vectors, so just do the matrix multiplication (`t()` is transpose):

```
a <- c(1, 2, 3)
b <- c(4, 5, 6)
t(a) %*% b
```

```
##      [,1]
## [1,]   32
```

- Note that the answer is actually a 1×1 matrix.
- Or as the sum of the elementwise multiplication:

```
sum(a * b)
```

```
## [1] 32
```