hw04: Eigenvalues

Kenji Sato[*]

30 March, 2017

1 Overview

Purpose

- Study lists in R
- Compute eigenvalues and eigenvectors in R.

Instructions

In this assignment, you will

- clone the assignment repository and make a working branch (eg. `solution` branch);
- solve the problems in Section 5;
- write the solutions in `solution.Rmd`;
- commit `solution.Rmd` and `solution.pdf`; and
- open a Pull Request.

2 Lists in R

2.1 List basics

A list in R is a vector that can contain another vector. It is also called as a recursive vector. Unlike atomic vectors, a list can have any type of objects. `list()` function creates a list.

```
(x <- list(3.1, -2.2, 0.3))
```

[*]Kobe University. Email: mail@kenjisato.jp

```
## [[1]]
## [1] 3.1
##
## [[2]]
## [1] -2.2
##
## [[3]]
## [1] 0.3
```

Subscription `[` works differently for lists.

```r
x[1]
```

```
## [[1]]
## [1] 3.1
```

`x[1]` is again a list. To get `3.1` you need to use `[[`.

```r
x[[1]]
```

```
## [1] 3.1
```

Here is a more practical example.

```r
y <- list(1.1, c(2.1, 2.2), c(3.1, 3.2, 3.2))
y
```

```
## [[1]]
## [1] 1.1
##
## [[2]]
## [1] 2.1 2.2
##
## [[3]]
## [1] 3.1 3.2 3.2
```

Now you must be able to get `3.2`.

```r
y[[3]][1]
```

```
## [1] 3.1
```

How about this? Each element of `z` is a list.

```r
z <- list()
for (i in seq_along(y)) {
  z[i] <- list(y[[i]])
}
z
```

```
## [[1]]
## [1] 1.1
##
## [[2]]
## [1] 2.1 2.2
##
## [[3]]
## [1] 3.1 3.2 3.2
```

To get `3.2`, do this.

```r
z[[3]][[2]]
```

```
## [1] 3.2
```

2.2 Named elements

Elements in list can have names.

```r
w <- list(a = 1, b = 2)
w
```

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

There are several ways to get elements from `w`. To get sublists:

```r
w[1]
```

```
## $a
## [1] 1
```

```r
w["a"]
```

```
## $a
## [1] 1
```

To get an element:

```r
w[[1]]
```

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

```r
w$a
```

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

or

```r
w[["a"]]
```

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

2.3 List as information store

Since list can contain any vector as elements, you can use lists as an information store.

```r
alice <- list(
  name = "Alice",
  age = 18,
  grades = c(100, 100, 95)
)

bob <- list(
  name = "Bob",
  age = 19,
  grades = c(90, 100, 100)
)
```

```r
students <- list(alice, bob)
students
```

```
## [[1]]
## [[1]]$name
## [1] "Alice"
##
## [[1]]$age
## [1] 18
##
## [[1]]$grades
## [1] 100 100  95
##
##
## [[2]]
## [[2]]$name
## [1] "Bob"
##
## [[2]]$age
## [1] 19
##
## [[2]]$grades
## [1]  90 100 100
```

To get Alice's grade for the second course, do something like this.

```
students[[1]]$grades[2]
```

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

3 Eigenvalues and eigenvectors

Now we are ready to compute eigenvalues.

```
set.seed(10903)
A <- matrix(rnorm(9), nrow = 3)
A
```

```
##            [,1]       [,2]       [,3]
## [1,] 0.3394338  0.8953551 -0.6204871
## [2,] 0.7093760  0.1422706 -1.7606138
## [3,] 1.2608842 -0.2606964 -0.9170967
```

```
eig_A <- eigen(A)
eig_A
```

```
## $values
## [1] -1.5087275+0.0000000i  0.5366676+0.7912732i  0.5366676-0.7912732i
##
## $vectors
##                 [,1]                  [,2]                  [,3]
## [1,]  0.1472020+0i 0.6780378+0.0000000i 0.6780378+0.0000000i
## [2,] -0.7504067+0i 0.4036736+0.4098632i 0.4036736-0.4098632i
## [3,] -0.6443767+0i 0.3669686-0.2732371i 0.3669686+0.2732371i
```

`eigen(A)` returns a list of eigenvalues and eigenvectors. `eigen(A)$values` is an atomic vector that consists of eigenvalues of `A`. `eigen(A)$vectors` is a matrix of eigenvectors. The n-th column of `eigen(A)$vectors` is an eigenvecotr that corresponds to the n-th element of `eigen(A)$values`.

```
lambda = eig_A$values
V = eig_A$vectors

all.equal(A %*% V[, 2], matrix(lambda[2] * V[, 2]))
```

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

`V[, 2]` is the atomic vector consisting of the second columns of `V`.

The above computation corresponds to the following identity.

$$Av_2 = \lambda_2 v_2$$

with the matrix of eigenvectors

$$V = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

The following identidy is true.

$$A \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix}$$

In R, this can be written as

```
all.equal(A %*% V, V %*% diag(lambda))
```

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

4 Exercise

Let A be a square matrix, Λ a diagonal matrix consisting of A's eigenvalues and V the matrix of eigenvalues for which $AV = V\Lambda$ holds. If A is diagonalizable,

$$A = V\Lambda V^{-1}$$

holds.

Let `A` be a 1000×1000 matrix.

```
set.seed(1093)
n <- 100
A <- matrix(rnorm(n * n), nrow = n)
```

Verify that $A = V\Lambda V^{-1}$.

Solution

Here is a sample code.

```
r <- eigen(A)
V <- r$vectors
Lambda <- diag(r$values)

all.equal(A + 0i, V %*% Lambda %*% solve(V))
```

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

5 Problems

Let

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix}.$$

```
B = matrix(c(
    2, 0, 0,
    0, 2, 0,
    1, 0, 2
  ), nrow = 3, byrow = TRUE)
B
```

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

Report on what happes if you try to run the following code to check if $B = V \Lambda V^{-1}$ is true? Why does it happen?

```
r <- eigen(B)
V <- r$vectors
Lambda <- diag(r$values)

all.equal(B + 0i, V %*% Lambda %*% solve(V))
```