

Solutions to Selected Exercises  
for  
Braun and Murdoch's  
A First Course in Statistical Programming with R

Kristy Alexander, Yiwen Diao, Qiang Fu, and Yu Han  
W. John Braun and Duncan J. Murdoch

March 19, 2008

## Chapter 6

# Computational Linear Algebra

### 6.1 Vectors and Matrices in R

#### 6.1.1 Constructing matrix objects

```
1. > A <- matrix(rep(seq(1, 5), 5), nrow=5)+matrix(rep(seq(0, 4), 5), byrow=TRUE, nrow=5)
> A
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    2    3    4    5    6
[3,]    3    4    5    6    7
[4,]    4    5    6    7    8
[5,]    5    6    7    8    9

> Hankel <- function(x){
+   A <- matrix(rep(seq(1, x), x), nrow=x)+matrix(rep(seq(0, x-1), x), byrow=TRUE, nrow=x)
+   return(A)
+ }
> Hankel(10)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    2    3    4    5    6    7    8    9    10
[2,]    2    3    4    5    6    7    8    9    10    11
[3,]    3    4    5    6    7    8    9    10    11    12
[4,]    4    5    6    7    8    9    10    11    12    13
[5,]    5    6    7    8    9    10    11    12    13    14
[6,]    6    7    8    9    10    11    12    13    14    15
[7,]    7    8    9    10    11    12    13    14    15    16
[8,]    8    9    10    11    12    13    14    15    16    17
[9,]    9    10    11    12    13    14    15    16    17    18
[10,]   10    11    12    13    14    15    16    17    18    19

> Hankel(12)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    1    2    3    4    5    6    7    8    9    10    11    12
[2,]    2    3    4    5    6    7    8    9    10    11    12    13
[3,]    3    4    5    6    7    8    9    10    11    12    13    14
[4,]    4    5    6    7    8    9    10    11    12    13    14    15
[5,]    5    6    7    8    9    10    11    12    13    14    15    16
[6,]    6    7    8    9    10    11    12    13    14    15    16    17
[7,]    7    8    9    10    11    12    13    14    15    16    17    18
```

```

      [8,]  8  9 10 11 12 13 14 15 16 17 18 19
      [9,]  9 10 11 12 13 14 15 16 17 18 19 20
    [10,] 10 11 12 13 14 15 16 17 18 19 20 21
    [11,] 11 12 13 14 15 16 17 18 19 20 21 22
    [12,] 12 13 14 15 16 17 18 19 20 21 22 23

```

```

3. > W <- cbind(c(1, 1, 1, 1, 1, 1, 1), c(2, 3, 4, 5, 6, 7, 8), c(4, 7, 5, 6, 7, 5, 3))
> W

```

```

      [,1] [,2] [,3]
    [1,]  1   2   4
    [2,]  1   3   7
    [3,]  1   4   5
    [4,]  1   5   6
    [5,]  1   6   7
    [6,]  1   7   5
    [7,]  1   8   3

```

### 6.1.2 Accessing matrix elements; row and column names

```

1. > Stochastic_matrix <- matrix(c(0.2, 0.3, 0.8, 0.7), ncol=2)
> rownames(Stochastic_matrix) <- c("sunny", "rainy")
> colnames(Stochastic_matrix) <- c("sunny", "rainy")
> Stochastic_matrix

```

```

      sunny rainy
    sunny  0.2  0.8
    rainy  0.3  0.7

```

```

3. > matrix[3, 1] <- 162
> matrix[5, 1] <- 181
> matrix[5, 2] <- 68
> matrix

```

```

      height weight
    Neil     172    62
    Cindy     168    64
    Pardeep   162    51
    Deepak    175    71
    Hao       181    68

```

### 6.1.3 Matrix Properties

```

1. > A <- matrix(c(3, 5, 4, 8), ncol=2)
> det(A)

[1] 4

> det(t(A))

[1] 4

> A <- matrix(c(3, 20, 15, 7), ncol=2)
> det(A)

[1] -279

> det(t(A))

[1] -279

```

```

> A <- matrix(c(4, 2, 25, 23), ncol=2)
> det(A)

[1] 42
> det(t(A))

[1] 42

```

### 6.1.4 Triangular matrices

```

1. > H3 <- matrix(c(1, 1/2, 1/3, 1/2, 1/3, 1/4, 1/3, 1/4, 1/5), nrow=3)
> Hnew <- H3
> Hnew[lower.tri(H3, diag=TRUE)] <- 0
> Hnew

      [,1] [,2]      [,3]
[1,]    0 0.5 0.3333333
[2,]    0 0.0 0.2500000
[3,]    0 0.0 0.0000000

3. > X <- matrix(c(1, 2, 3, 1, 4, 9), ncol=2)
> X[3, 2]

[1] 9
> X[3, 2, drop=FALSE]

      [,1]
[1,]    9
> dim(X[3, 2])

NULL
> dim(X[3, 2, drop=FALSE])

[1] 1 1

```

## 6.2 Matrix Multiplication and Inversion

```

1. > X <- matrix(c(1, 2, 3, 1, 4, 9), ncol=2)
> 1.5*X

      [,1] [,2]
[1,]  1.5  1.5
[2,]  3.0  6.0
[3,]  4.5 13.5

```

### 6.2.3 Matrix inversion in R

```

1. > XX <- t(X) %*% X
> XXinv <- solve(XX)
> XXinv

      [,1]      [,2]
[1,] 1.2894737 -0.4736842
[2,] -0.4736842  0.1842105

> # Verification:
> crossprod(XX, XXinv)

```

```

          [,1]      [,2]
[1,]  1.000000e+00 -1.332268e-15
[2,] -1.887379e-15  1.000000e+00

```

This is numerically close to the identity matrix.

```

3. (a) > Hilbert <- function(n) {
+   A <- matrix(rep(NA, n*n), nrow=n)
+   for(i in 1:n){
+     for(j in 1:n){
+       A[i, j] <- 1/(i+j-1)
+     }
+   }
+   return(A)
+ }
> # or
>
> Hilbert <- function(n) {
+   outer(1:n, 1:n, function(x, y) 1/(x+y-1))
+ }

```

(b) Yes. (Showing that all Hilbert matrices are invertible is not obvious, but see <http://planetmath.org/encyclopedia/HilbertMatrix.html> for the formula.)

```

(c) > qr.solve(Hilbert(1))
      [,1]
[1,]    1
> qr.solve(Hilbert(2))
      [,1] [,2]
[1,]    4  -6
[2,]   -6  12
> qr.solve(Hilbert(3))
      [,1] [,2] [,3]
[1,]    9 -36  30
[2,]   -36 192 -180
[3,]    30 -180 180
> qr.solve(Hilbert(4))
      [,1] [,2] [,3] [,4]
[1,]   16 -120  240 -140
[2,] -120 1200 -2700 1680
[3,]  240 -2700 6480 -4200
[4,] -140 1680 -4200 2800
> qr.solve(Hilbert(5))
      [,1] [,2] [,3] [,4] [,5]
[1,]   25 -300 1050 -1400  630
[2,] -300 4800 -18900 26880 -12600
[3,] 1050 -18900 79380 -117600 56700
[4,] -1400 26880 -117600 179200 -88200
[5,]  630 -12600 56700 -88200 44100
> qr.solve(Hilbert(6))
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   36 -630 3360 -7560 7560 -2772
[2,] -630 14700 -88200 211680 -220500 83160
[3,] 3360 -88200 564480 -1411200 1512000 -582120

```

```

[4,] -7560 211680 -1411200 3628800 -3969000 1552320
[5,] 7560 -220500 1512000 -3969000 4410000 -1746360
[6,] -2772 83160 -582120 1552320 -1746360 698544
> qr.solve(Hilbert(7))
> qr.solve(Hilbert(8))
> qr.solve(Hilbert(9))
> qr.solve(Hilbert(10))
(d) > eigen(Hilbert(10))$values
[1] 1.751920e+00 3.429295e-01 3.574182e-02 2.530891e-03 1.287496e-04
[6] 4.729689e-06 1.228968e-07 2.147439e-09 2.266746e-11 1.093249e-13
One of the eigenvalues is near  $10^{-13}$  which is very small, while the largest eigenvalue is over 1.
Therefore, the columns of the matrix are close to being linearly dependent.

```

## 6.2.4 Solving linear systems

```

1. > X1 <- c(10, 11, 12, 13, 14, 15)
> X2 <- X1^2
> X3 <- X1^3
> X4 <- X1^4
> X5 <- X1^5
> X0 <- c(1, 1, 1, 1, 1, 1)
> A <- cbind(X0, X1, X2, X3, X4, X5)
> f <- matrix(c(25, 16, 26, 19, 21, 20), nrow=6)
> a <- solve(A, f)
> a

      [,1]
X0 2.536100e+05
X1 -1.025510e+05
X2 1.650092e+04
X3 -1.320667e+03
X4 5.258333e+01
X5 -8.333333e-01
> A %*% a

      [,1]
[1,] 25
[2,] 16
[3,] 26
[4,] 19
[5,] 21
[6,] 20
> # or
>
> x <- c(10, 11, 12, 13, 14, 15)
> y <- c(25, 16, 26, 19, 21, 20)
> A <- outer(x, 0:5, function(x, y) x^y)
> A

      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1 10 100 1000 10000 100000
[2,] 1 11 121 1331 14641 161051
[3,] 1 12 144 1728 20736 248832
[4,] 1 13 169 2197 28561 371293

```

```
[5,] 1 14 196 2744 38416 537824
[6,] 1 15 225 3375 50625 759375

> A <- outer(x, 0:5, function(x, y) x^y)
> solve(A, y) # Solve Aa = y for a

[1] 2.536100e+05 -1.025510e+05 1.650092e+04 -1.320667e+03 5.258333e+01
[6] -8.333333e-01

f(x) = 253610 - 102551x + 16500.92x2 - 1320.667x3 + 52.58333x4 - .8333x5
```

### 6.3 Eigenvalues and Eigenvectors

```
1. > X <- matrix(c(1, 2, 3, 1, 4, 9), ncol=2)
> H <- X %*% solve(t(X) %*% X) %*% t(X)
> H
```

```
      [,1]      [,2]      [,3]
[1,] 0.5263158 0.4736842 -0.1578947
[2,] 0.4736842 0.5263158 0.1578947
[3,] -0.1578947 0.1578947 0.9473684
```

```
3. > sum(E$values)
```

```
[1] 2
```

```
> trace <- function(data)sum(diag(data))
> trace(H)
```

```
[1] 2
```

5. Hint: Note that  $HX = X$  and  $H(I - X) = 0$ .

### 6.5 Chapter Exercises

```
1. (a) > P <- matrix(c(0.1, 0.4, 0.3, 0.2,
+                   0.2, 0.1, 0.4, 0.3,
+                   0.3, 0.2, 0.1, 0.4,
+                   0.4, 0.3, 0.2, 0.1), nrow=4)
> apply(P, 1, sum)
```

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

```
(b) > P2 <- P %*% P
> P3 <- P %*% P %*% P
> P5 <- P2 %*% P3
> P10 <- P5 %*% P5
> P10
```

```
      [,1]      [,2]      [,3]      [,4]
[1,] 0.2500000 0.2500016 0.2500000 0.2499983
[2,] 0.2499983 0.2500000 0.2500016 0.2500000
[3,] 0.2500000 0.2499983 0.2500000 0.2500016
[4,] 0.2500016 0.2500000 0.2499983 0.2500000
```

```
(c) > solve(rbind(rep(1, 4), (diag(rep(1, 4)) - t(P))[-4, ]), c(1, 0, 0, 0))
[1] 0.25 0.25 0.25 0.25
```

The rows and columns of  $P^n$  appear to be converging to  $x$ .

```

(d) > set.seed(361)
> pseudo <- function(n) {
+   Y <- numeric(n)
+   Y[1] <- 1
+   for(j in 2:n) {
+     Y[j] <- sample(1:4, 1,
+                   prob=P[Y[j-1], ],
+                   replace=T)
+   }
+   return(Y)
+ }
> yresult <- pseudo(10000)
(e) > table(yresult)
yresult
  1    2    3    4
2502 2508 2523 2467
The relative frequency distribution is near  $x$ .
3. (a) > C <- matrix(c(245921, 7304620, 34390.48, 6864693), ncol=1)
> A <- matrix(c(1, 10, 0.1, 5,
+              1, 30, 0.15, 27,
+              1, 50, 0.03, 45,
+              1, 100, 0.5, 125), nrow=4)
> X <- solve(A, C)
> X
      [,1]
[1,] 113750
[2,] 75324
[3,] 35546
[4,] 21301
(b) > income <- c(X[1, 1]*10, X[2, 1]*30, X[3, 1]*50, X[4, 1]*100)
> income
[1] 1137500 2259720 1777300 2130100
> claimsize <- c(X[1, 1]*0.1*50, X[2, 1]*0.15*180, X[3, 1]*0.03*1500, X[4, 1]*0.5*250)
> claimsize
[1] 568750 2033748 1599570 2662625
(c) > claimamount <- numeric(1000)
> for(i in 1:1000){
+   A <- rgamma(rpois(1, 0.1*X[1, 1]), shape=2, scale=50/2)
+   B <- rgamma(rpois(1, 0.15*X[2, 1]), shape=2, scale=180/2)
+   C <- rgamma(rpois(1, 0.03*X[3, 1]), shape=2, scale=1500/2)
+   D <- rgamma(rpois(1, 0.5*X[4, 1]), shape=2, scale=250/2)
+   claimamount[i] <- sum(A, B, C, D)
+ }
> mean(claimamount)
[1] 6863177
> var(claimamount)
[1] 4846302489
> sd(claimamount)
[1] 69615.39
> sum(claimamount > 7304620)/length(claimamount)
[1] 0

```