STA1373-Optimisasi Statistika: Pencarian Akar Persamaan/ Penyelesaian SPL

Sachnaz Desta Oktarina

2023-01-27

Contents

Disclaimer
Please do not cite this document. Data and syntax can be reused freely while keep attributing and refering to the original source (where applicable) in the reference. This document is used for educational purpose only.

Vector and Matrix

```
u <- seq(1,5)
v <- seq(6,10)
u
```

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

```
v
```

```
## [1]  6  7  8  9 10
```

##penjumlahan vektor

```
u + v
```

```
## [1]  7  9 11 13 15
```

##pengurangan vektor

```
u - v
```

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

Bagaimana jika panjang kedua vektor berbeda?

```
x <- seq(1,2)
x
```

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

```r
u + x
```

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

```
## [1] 2 4 4 6 6
```

Berdasarkan contoh tersebut, R akan mengeluarkan peringatan yang menunjukkan operasi dilakukan pada vektor dengan panjang berbeda. R akan tetap melakukan perhitungan dengan menjumlahkan kembali vektor u yang belum dijumlahkan dengan vektor x sampai seluruh elemen vektor u dilakukan operasi penjumlahan Menghitung Inner Product dan Panjang Vektor

##Inner product

```r
u%*%v
```

```
##      [,1]
## [1,]  130
```

##Panjang vektor

```r
sqrt(sum(u*u))
```

```
## [1] 7.416198
```

Operasi Matrix

```r
A <- matrix(1:9,3)
B <- matrix(10:18,3)
C <- matrix(1:6,3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
B
```

```
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
```

```r
C
```

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

##Penjumlahan A + B

```r
A+B
```

```
##      [,1] [,2] [,3]
## [1,]   11   17   23
## [2,]   13   19   25
## [3,]   15   21   27
```

```r
#A+C
```

##Perkalian

```
A%*%B
```

```
##      [,1] [,2] [,3]
## [1,]  138  174  210
## [2,]  171  216  261
## [3,]  204  258  312
```

Operasi Baris Elementer

##Row Scaling

```
scale_row <- function(m, row, k){
  m[row, ] <- m[row, ]*k
  return(m)
}

(A <- matrix(1:15, nrow=5))
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]    2    7   12
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    5   10   15
```

lakukan scaling pada row 2 dengan nilai 10

```
scale_row(m=A, row=2, 10)
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]   20   70  120
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    5   10   15
```

##Row Swapping

```
swap_row <- function(m, row1, row2){
  row_tmp <- m[row1, ]
  m[row1, ] <- m[row2, ]
  m[row2, ] <- row_tmp
  return(m)
}
```

Lakukan swapping baris 2 dengan baris 5

```
swap_row(m=A, row1 = 2, row2 = 5)
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]    5   10   15
## [3,]    3    8   13
## [4,]    4    9   14
## [5,]    2    7   12
```

##Row replacement

```
replace_row <- function(m, row1, row2, k){
  m[row2, ] <- m[row2, ] + m[row1, ]*k
  return(m)
}
```

Lakukan replacement

```
replace_row(m=A, row1=1, row2=3, k=-3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   11
## [2,]    2    7   12
## [3,]    0  -10  -20
## [4,]    4    9   14
## [5,]    5   10   15
```

Eliminasi Gauss

##Row Echelon Form

```
ref_matrix <- function(a){
  m <- nrow(a)
  n <- ncol(a)
  piv <- 1

  # cek elemen diagonal apakah bernilai nol
  for(row_curr in 1:m){
    if(piv <= n){
      i <- row_curr
      while(a[i, piv] == 0 && i < m){
        i <- i+1
        if(i > m){
          i <- row_curr
          piv <- piv+1
          if(piv > n)
            return(a)
        }
      }

      # jika diagonal bernilai nol, lakukan row swapping
      if(i != row_curr)
        a <- swap_row(a, i, row_curr)

      # proses triangulasi untuk membentuk matriks segitiga atas
      for(j in row_curr:m)
        if(j != row_curr){
          c <- a[j, piv]/a[row_curr, piv]
          a <- replace_row(a, row_curr, j, -c)
        }
      piv <- piv+1
    }
  }
  return(a)
}
```

Diasumsikan terdapat sebuat matrix

```
am <- c(1,1,2,
        1,2,1,
        1,-1,2,
        6,2,10)
(m <- matrix(am, nrow=3))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    1    6
## [2,]    1    2   -1    2
## [3,]    2    1    2   10
```

Carilah solusi dari persamaan matrix di atas

```
ref_matrix(m)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    1    6
## [2,]    0    1   -2   -4
## [3,]    0    0   -2   -6
```

Eliminasi Gauss - Jordan

create a matrix

```
A <- matrix(c(-3,2,-1,6,-6,7,3,-4,4),byrow = T,nrow=3,ncol=3)
A
```

```
##      [,1] [,2] [,3]
## [1,]   -3    2   -1
## [2,]    6   -6    7
## [3,]    3   -4    4
```

```
b <- matrix(c(-1,-7,-6),nrow=3,ncol=1)
b
```

```
##      [,1]
## [1,]   -1
## [2,]   -7
## [3,]   -6
```

dimension of matrix A

```
nrow <- nrow(A)
nrow
```

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

concatenante matrix A and vector b to create Augmented Matrix Ugmt.mtx

```
Ugmt.mtx <- cbind(A,b)
Ugmt.mtx
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -3    2   -1   -1
## [2,]    6   -6    7   -7
## [3,]    3   -4    4   -6
```

```
Ugmt.mtx[1,] <- Ugmt.mtx[1,]/Ugmt.mtx[1,1]
Ugmt.mtx
```

```
##      [,1]       [,2]      [,3]       [,4]
## [1,]    1 -0.6666667 0.3333333  0.3333333
## [2,]    6 -6.0000000 7.0000000 -7.0000000
## [3,]    3 -4.0000000 4.0000000 -6.0000000
```

ILUSTRASI:

```
Ugmt.mtx[2, ] <- Ugmt.mtx[2, ] - Ugmt.mtx[2-1, ] * Ugmt.mtx[2, 2-1] #pembuat nol element matrix
Ugmt.mtx
```

```
##      [,1]       [,2]      [,3]       [,4]
## [1,]    1 -0.6666667 0.3333333  0.3333333
## [2,]    0 -2.0000000 5.0000000 -9.0000000
## [3,]    3 -4.0000000 4.0000000 -6.0000000
```

```
Ugmt.mtx[2,] <- Ugmt.mtx[2,]/Ugmt.mtx[2,2] #pembuat =1 diagonal matrix
Ugmt.mtx
```

```
##      [,1]       [,2]       [,3]       [,4]
## [1,]    1 -0.6666667  0.3333333  0.3333333
## [2,]    0  1.0000000 -2.5000000  4.5000000
## [3,]    3 -4.0000000  4.0000000 -6.0000000
```

dst, dalam bentuk loop:

```
A <- matrix(c(-3,2,-1,6,-6,7,3,-4,4),byrow = T,nrow=3,ncol=3)
A
```

```
##      [,1] [,2] [,3]
## [1,]   -3    2   -1
## [2,]    6   -6    7
## [3,]    3   -4    4
```

```
b <- matrix(c(-1,-7,-6),nrow=3,ncol=1)
b
```

```
##      [,1]
## [1,]   -1
## [2,]   -7
## [3,]   -6
```

```
nrow <- nrow(A)
Ugmt.mtx <- cbind(A,b)
Ugmt.mtx
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -3    2   -1   -1
## [2,]    6   -6    7   -7
## [3,]    3   -4    4   -6
```

```
Ugmt.mtx[1,] <- Ugmt.mtx[1,]/Ugmt.mtx[1,1]
Ugmt.mtx
```

```
##      [,1]       [,2]      [,3]       [,4]
## [1,]    1 -0.6666667 0.3333333  0.3333333
## [2,]    6 -6.0000000 7.0000000 -7.0000000
## [3,]    3 -4.0000000 4.0000000 -6.0000000
```

```
for (i in 2:nrow){ # loop over rows
  for (j in i:nrow) { # loop over columns
```

```
    Ugmt.mtx[j, ] <- Ugmt.mtx[j, ] - Ugmt.mtx[i-1, ] * Ugmt.mtx[j, i-1] # replace the row values at jth
  }
  Ugmt.mtx[i,] <- Ugmt.mtx[i,]/Ugmt.mtx[i,i]
}
# print output
Ugmt.mtx #Back Susbstitution needed
```

```
##      [,1]       [,2]       [,3]       [,4]
## [1,]    1 -0.6666667  0.3333333  0.3333333
## [2,]    0  1.0000000 -2.5000000  4.5000000
## [3,]    0  0.0000000  1.0000000 -1.0000000
```

in case we want to do it. and want to produce the solution instantly: ILUSTRASI:

```
A <- matrix(c(-3,2,-1,6,-6,7,3,-4,4),byrow = T,nrow=3,ncol=3)
A
```

```
##      [,1] [,2] [,3]
## [1,]   -3    2   -1
## [2,]    6   -6    7
## [3,]    3   -4    4
```

```
b <- matrix(c(-1,-7,-6),nrow=3,ncol=1)
b
```

```
##      [,1]
## [1,]   -1
## [2,]   -7
## [3,]   -6
```

```
# dimension of matrix A
nrow <- nrow(A)
nrow
```

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

```
# concatenante matrix A and vector b
Ugmt.mtx <- cbind(A,b)
Ugmt.mtx
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -3    2   -1   -1
## [2,]    6   -6    7   -7
## [3,]    3   -4    4   -6
```

```
Ugmt.mtx[1,] <- Ugmt.mtx[1,]/Ugmt.mtx[1,1]
for (i in 2:nrow){ # loop over rows
  for (j in i:nrow) { # loop over columns
    Ugmt.mtx[j, ] <- Ugmt.mtx[j, ] - Ugmt.mtx[i-1, ] * Ugmt.mtx[j, i-1] # replace the row values at jth
  }
  Ugmt.mtx[i,] <- Ugmt.mtx[i,]/Ugmt.mtx[i,i]
}

Ugmt.mtx[1, ] <- Ugmt.mtx[1, ] - Ugmt.mtx[2, ] * Ugmt.mtx[1, 2]
Ugmt.mtx
```

```
##      [,1] [,2]      [,3]      [,4]
## [1,]    1    0 -1.333333  3.333333
## [2,]    0    1 -2.500000  4.500000
```

```
## [3,]    0    0  1.000000 -1.000000
```

Dengan menggunakan loop:

```
A <- matrix(c(-3,2,-1,6,-6,7,3,-4,4),byrow = T,nrow=3,ncol=3)
A
```

```
##      [,1] [,2] [,3]
## [1,]   -3    2   -1
## [2,]    6   -6    7
## [3,]    3   -4    4
```

```
b <- matrix(c(-1,-7,-6),nrow=3,ncol=1)
b
```

```
##      [,1]
## [1,]   -1
## [2,]   -7
## [3,]   -6
```

```
# dimension of matrix A
nrow <- nrow(A)
nrow
```

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

```
# concatenante matrix A and vector b
Ugmt.mtx <- cbind(A,b)
Ugmt.mtx
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -3    2   -1   -1
## [2,]    6   -6    7   -7
## [3,]    3   -4    4   -6
```

```
Ugmt.mtx[1,] <- Ugmt.mtx[1,]/Ugmt.mtx[1,1]
for (i in 2:nrow){
  for (j in i:nrow) {
    Ugmt.mtx[j, ] <- Ugmt.mtx[j, ] - Ugmt.mtx[i-1, ] * Ugmt.mtx[j, i-1]
  }
  Ugmt.mtx[i,] <- Ugmt.mtx[i,]/Ugmt.mtx[i,i]
}
for (i in j:2){
  for (j in i:2-1) {
    Ugmt.mtx[j, ] <- Ugmt.mtx[j, ] - Ugmt.mtx[i, ] * Ugmt.mtx[j, i]
  }
}
Ugmt.mtx
```

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

#Metode Iterasi

Jacobi iteration

```
jacobi <- function(A, b, x0, tol=1e-6, maxiter=1000) {
  n <- length(b)
  x <- x0
  for (iter in 1:maxiter) {
    x_new <- numeric(n)
    for (i in 1:n) {
      s <- 0
      for (j in 1:n) {
        if (j != i) {
          s <- s + A[i,j] * x[j]
        }
      }
      x_new[i] <- (b[i] - s) / A[i,i]
    }
    if (all(abs(x - x_new) < tol)) {
      break
    }
    x <- x_new
  }
  return(list(x=x, iter=iter))
}
```

Contoh penggunaan:

```
A <- matrix(c(3,1,-1,4,7,-3,2,-2,5), nrow=3, byrow=TRUE)
b <- c(5,20,10)
x0 <- c(0,0,0)
result <- jacobi(A, b, x0)
x <- result$x
iter <- result$iter
cat("The solution is x =", x, " after ", iter, " iterations\n")
```

```
## The solution is x = 1.506025 3.132531 2.650602  after  22  iterations
```

Gauss Seidel iteration

```
gaussSeidel <- function(A, b, epsilon, maxIterations) {
  # Extract the size of the system
  n <- length(b)

  # Initialize the solution vector x
  x <- numeric(n)

  # Iterate until either epsilon is reached or maxIterations is reached
  for (iteration in 1:maxIterations) {
    x_old <- x

    # Update each variable using the Gauss-Seidel formula
    for (i in 1:n) {
      x[i] <- (b[i] - sum(A[i, -i] * x[-i])) / A[i, i]
    }

    # Check if epsilon is reached
```

```
    if (max(abs(x - x_old)) < epsilon) {
      break
    }
  }

  # Return the solution and the number of iterations
  list(x = x, iterations = iteration)
}
```

Contoh aplikasi penggunaan:

```
#A <- matrix(c(4, 1, 1, 4, 1, 1, 1, 1, 4), nrow = 3)
#b <- c(6, 6, 6)
result <- gaussSeidel(A, b, 1e-6, 100)
x <- result$x
iterations <- result$iterations
cat("Number of iterations:", iterations, "\n")
```

```
## Number of iterations: 15
```

```
cat("Solution:", x, "\n")
```

```
## Solution: 1.506024 3.13253 2.650602
```

STUDI KASUS

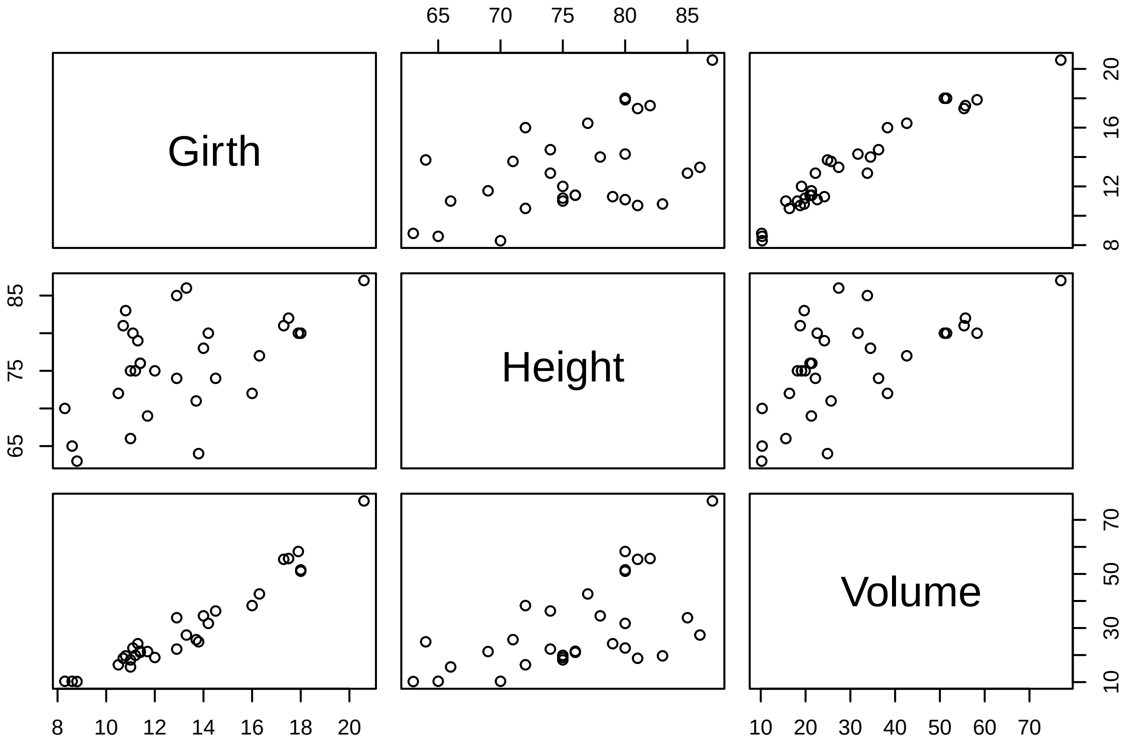
```
head(trees)
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
```

```
str(trees)
```

```
## 'data.frame':    31 obs. of  3 variables:
##  $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
##  $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
##  $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

```
plot(trees)
```



bentuk dalam sebuah matrix

```
pred <- cbind(intercept=1, Girth=trees$Girth, Height=trees$Height)
head(pred)
```

```
##      intercept Girth Height
## [1,]         1   8.3     70
## [2,]         1   8.6     65
## [3,]         1   8.8     63
## [4,]         1  10.5     72
## [5,]         1  10.7     81
## [6,]         1  10.8     83
```

Langkah selanjutnya adalah membentuk vektor b

```
resp<- trees$Volume
head(resp)
```

```
## [1] 10.3 10.3 10.2 16.4 18.8 19.7
```

lakukan transformasi:

```
A <- t(pred) %*% pred #(X'X)
b <- t(pred) %*% resp #(X'y)
Ab <- cbind(A,b)
```

Dengan menggunakan metode eliminasi Gauss/ Row echelon form:

```
ref_matrix(Ab)
```

```
##           intercept    Girth    Height
## intercept        31 410.7000 2356.0000  935.3000
## Girth             0 295.4374  311.5000 1496.6435
## Height            0   0.0000  889.5641  301.7857
```

Menggunakan substitusi balik didapatkan koefficient variable height: 301.7857/889.5641 = 0.3392512

```
301.7857/889.5641
```

```
## [1] 0.3392512
```

Dengan menggunakan metode eliminasi Gauss Jordan:

```
Ugmt.mtx <- cbind(A,b)
Ugmt.mtx[1,] <- Ugmt.mtx[1,]/Ugmt.mtx[1,1]
for (i in 2:nrow){
  for (j in i:nrow) {
    Ugmt.mtx[j, ] <- Ugmt.mtx[j, ] - Ugmt.mtx[i-1, ] * Ugmt.mtx[j, i-1]
  }
  Ugmt.mtx[i,] <- Ugmt.mtx[i,]/Ugmt.mtx[i,i]
}
for (i in j:2){
  for (j in i:2-1) {
    Ugmt.mtx[j, ] <- Ugmt.mtx[j, ] - Ugmt.mtx[i, ] * Ugmt.mtx[j, i]
  }
}
Ugmt.mtx
```

```
##           intercept Girth Height
## intercept         1     0      0 -57.9876589
## Girth             0     1      0   4.7081605
## Height            0     0      1   0.3392512
```

Dengan menggunakan metode iterative Jacobi ?

Dengan menggunakan metode iterative Gauss Siedel:

```
result <- gaussSeidel(A, b, 1e-6, 1000)
x <- result$x
iterations <- result$iterations
cat("Number of iterations:", iterations, "\n")
```

```
## Number of iterations: 1000
```

```
cat("Solution:", x, "\n")
```

```
## Solution: -57.77524 4.71234 0.3357443
```

Dengan menggunakan fungsi base R

```
## fungsi solve
solve(A,b)
```

```
##                  [,1]
## intercept -57.9876589
## Girth       4.7081605
## Height      0.3392512
```