hw 3 solution

Statistical Computing, Jieun Shin

Autumn 2021

문제 1.

1. 먼저 행렬 A와 벡터 b를 정의한다.

```
#  정의
set.seed(123)
A = matrix(rnorm(25), 5, 5)
A = t(A) %*% A
b = matrix(rnorm(5), 5, 1)
```

2. 가우스 소거법과 삼각행렬을 풀기위한 후방대입법 함수를 정의한다 (강의안 참고).

```
gaussianelimination = function(Ab){
  n = nrow(Ab)
  for (k in (1:(n-1))){
    for (i in ((k+1):n)){
      mik = Ab[i,k]/Ab[k,k]
      Ab[i,k]=0

      for (j in ((k+1):(n+1))){
        Ab[i,j] = Ab[i,j] - mik*Ab[k,j]
      }
    }
  }
  return(Ab)
}

backwardsub = function(U,b){
  x = c(0)
  n = nrow(U)
  for (i in (n:1)){
    x[i] = b[i]
      if (i < n){
        for (j in ((i+1):n)){
          x[i] = x[i] - U[i,j]*x[j]
        }
      }

    x[i] = x[i]/U[i,i]
    }
  return(cbind(x))
}
```

3. 선형방정식 $Ax = b$의 해를 가우스 소거법으로 구한 해와 `solve`함수로 구한 해와 비교한다. 두 방법으로

구한 해가 같음을 확인할 수 있다.

```
# 가우스 소거법
Ab = cbind(A, b)
ge = gaussianelimination(Ab)
backwardsub(ge[, 1:5], ge[, 6])
```

```
##               x
## [1,] -303.89287
## [2,]  -99.34466
## [3,]  153.15147
## [4,] -128.24258
## [5,]  -33.69567
```

```
# solve
solve(A) %*% b
```

```
##            [,1]
## [1,] -303.89287
## [2,]  -99.34466
## [3,]  153.15147
## [4,] -128.24258
## [5,]  -33.69567
```

문제 2.

1. 문제에서 정의한 행렬 A를 정의한다.

```
set.seed(123)
A = matrix(rnorm(18), 6, 3)
```

2. svd함수를 이용하여 일반화 역행렬을 구하는 함수 ginvsvd를 정의한다.

```
ginvsvd = function(A){
  svd = svd(A)  # 특이값 분해
  u = svd$u
  s = svd$d
  v = svd$v

  dim = dim(A)
  d = min(dim)
  S = diag(1/s, d, d)   # S+는 S의 대각원소 중 0이 아닌 것을 역수로 취한 행렬

  invA = v %*% S %*% t(u)   # A+ = V S+ U^t

  out = list()
  out$u = u
  out$S = S
  out$v = v
  out$invA = invA
  return(out)

}
```

3. ginvsvd와 limSolve의 Solve함수와 비교하여 두 방법이 계산한 역행렬이 동일한 것을 확인하였다.

```
ginvsvd(A)
```

```
## $u
##             [,1]       [,2]        [,3]
## [1,] -0.19409634  0.2482957 -0.03901793
## [2,] -0.07844401 -0.5742156 -0.29715179
## [3,]  0.41339640 -0.4515290  0.44083368
## [4,] -0.40956052 -0.3413394  0.67977897
## [5,] -0.08327480  0.5233440  0.48523038
## [6,]  0.78141029  0.1197735  0.13526315
##
## $S
##           [,1]      [,2]     [,3]
## [1,] 0.2988448 0.0000000 0.000000
## [2,] 0.0000000 0.4850973 0.000000
## [3,] 0.0000000 0.0000000 0.671993
##
## $v
##             [,1]       [,2]      [,3]
## [1,]  0.61912651 -0.2240084 0.7526637
## [2,]  0.02617416  0.9638056 0.2653182
## [3,] -0.78485494 -0.1445652 0.6025808
##
## $invA
##             [,1]        [,2]       [,3]        [,4]      [,5]        [,6]
## [1,] -0.08262818 -0.10241113  0.3485205  0.30513573 0.1731447  0.19997744
## [2,]  0.10761323 -0.32206182 -0.1292769 -0.04159407 0.3305453  0.08622736
## [3,]  0.01231321 -0.06165796  0.1132097  0.39526282 0.1793153 -0.13690705
```

```
limSolve::Solve(A)
```

```
##             [,1]        [,2]       [,3]        [,4]      [,5]        [,6]
## [1,] -0.08262818 -0.10241113  0.3485205  0.30513573 0.1731447  0.19997744
## [2,]  0.10761323 -0.32206182 -0.1292769 -0.04159407 0.3305453  0.08622736
## [3,]  0.01231321 -0.06165796  0.1132097  0.39526282 0.1793153 -0.13690705
```

문제 3.

1. Gram-Schmidt 직교화 알고리즘을 구현한 함수를 정의한다.

```
norm = function(x) sqrt(sum(x^2))

GramSchmidt = function(A){
  dim = dim(A)
  n = dim[2]
  p = dim[1]

  # define the normal orthogonal basis q
  q = matrix(0, nrow = p, ncol = n)
  q[,1] = A[, 1] / norm(A[, 1])      # q_1 = x_1 /norm(x_1)

  for( i in 2:n ){

    val = 0
    for( j in 1:(i-1) ){
    v = t(A[, i]) %*% q[, j]
    v = c(v) * q[, j]
```

```
    val = val + v                # sum_{j=1}^{i-1} [ t(x_i) %*% q_j ] * q_j
    }

    qval = A[, i] - val
    q[, i] = qval / norm(qval)   # q_i = q_i / norm(q_i)
  }

  return(q)
}
```

3. 위에서 정의한 GramSchmidt와 qr.Q로부터 구한 직교행렬이 같음을 확인하였다.

```
GramSchmidt(A)
```

```
##             [,1]       [,2]        [,3]
## [1,] -0.23353515  0.2136135  0.02602221
## [2,] -0.09590878 -0.6308710 -0.13030532
## [3,]  0.64947190 -0.3000898  0.23925251
## [4,]  0.02937895 -0.2184255  0.83533169
## [5,]  0.05387073  0.6080801  0.37895734
## [6,]  0.71462152  0.2210114 -0.28933355
```

```
qr = qr(A)
qr.Q(qr)
```

```
##             [,1]       [,2]        [,3]
## [1,] -0.23353515  0.2136135 -0.02602221
## [2,] -0.09590878 -0.6308710  0.13030532
## [3,]  0.64947190 -0.3000898 -0.23925251
## [4,]  0.02937895 -0.2184255 -0.83533169
## [5,]  0.05387073  0.6080801 -0.37895734
## [6,]  0.71462152  0.2210114  0.28933355
```