Statistics 1 Unit 2 Team 8

Nikolaos Kornilakis Rodrigo Viale Jakub Trnan Aleksandra Daneva
Luis Diego Pena Monge

2023-12-13

Exercise 15

(a) Suppose

$$B = \begin{bmatrix} \alpha & a' \\ a & A \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

is symmetric and positive definite.

Write

$$B = \begin{bmatrix} \alpha & a_1 & \cdots & a_n \\ a_1 & A_{11} & \cdots & A_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_n & A_{n1} & \cdots & A_{nn} \end{bmatrix}$$

First of all, notice that since $B^t = B \Rightarrow B_{ij} = B_{ji}$ for all $i, j \in \{1, \ldots, n+1$

Also, note that $A_{ij} := B_{(i+1)(j+1)} = B_{(j+1)(i+1)} = A_{ji}$, hence A is symmetric.

Since B is positive definite, for all $x \neq 0$, $x^t B x > 0$.

$$\begin{aligned} x^t \begin{bmatrix} \alpha & a_1 & \ldots & a_n \\ a_1 & A_{11} & \ldots & A_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_n & A_{n1} & \ldots & A_{nn} \end{bmatrix} x &= \begin{bmatrix} (x_1\alpha + x_2 a_1 + \cdots + x_{n+1} a_n) \\ (x_1 a_1 + x_2 A_{11} + \cdots + x_{n+1} A_{1n}) \\ \vdots \\ (x_1 a_1 + x_2 A_{n1} + \cdots + x_{n+1} A_{nn}) \end{bmatrix}^t x \\ &= x_1(x_1\alpha + x_2 a_1 + \cdots + x_{n+1} a_n) + \cdots + x_{n+1}(x_1 a_n + x_2 A_{n1} + \cdots + x_{n+1} A_{nn}) \quad (1) \end{aligned}$$

Take $x = (1, 0, \ldots, 0)^t \in \mathbb{R}^n \implies x^t B x = \alpha$. Because B is positive definite, $\alpha > 0$

Let $x \in \mathbb{R}^{n+1}$ s.t. $x \neq 0$ and $x_1 = 0$, i.e. $x = (0, x_2, \ldots, x_{n+1})^t$.

Denote $y = (x_2, \ldots, x_{n+1})^t \in \mathbb{R}^n$ and from (1) it's evident that:

$$x^t \begin{bmatrix} \alpha & a' \\ a & A \end{bmatrix} = y^t A y$$

Since $x \neq 0 \implies y \neq 0$, and since $x^t B x > 0 \implies y^t A y > 0 \implies y^t A y > 0 \; \forall y \neq 0$. Therefore, A is positive definite.

(b) Now

$$B = LL' \Leftrightarrow \begin{bmatrix} \alpha & a' \\ a & A \end{bmatrix} = LL' \begin{bmatrix} \alpha & a' \\ a & A \end{bmatrix} = \begin{bmatrix} \beta & 0 \\ V & C \end{bmatrix} \begin{bmatrix} \beta & V' \\ 0 & C' \end{bmatrix} = \begin{bmatrix} \beta^2 & \beta V' \\ \beta V & VV' + CC' \end{bmatrix}$$

C lower-triangular, $V \in \mathbb{R}^n, \beta \in \mathbb{R}$.

$$\beta_2 = \alpha \Leftrightarrow \beta = \sqrt{\alpha}$$
$$\beta V = a \Leftrightarrow \sqrt{\alpha} V = a \Leftrightarrow v = \frac{a}{\sqrt{\alpha}}$$

\

$$VV' + CC' = A \Leftrightarrow \frac{aa'}{\alpha} + CC' = A \Leftrightarrow CC' = A - \frac{aa'}{\alpha} \implies L = \begin{bmatrix} \sqrt{a} & 0 \\ \frac{a}{\sqrt{\alpha}} & C \end{bmatrix}$$

where $CC' = A - \frac{aa'}{\alpha} \Rightarrow C = \sqrt{A - \frac{aa'}{\alpha}}$

Exercise 16

Let B be symmetric and positive-definite, defined as:

$$B := \begin{bmatrix} A & a \\ a' & \alpha \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

Write

$$B = \begin{bmatrix} A_{11} & \cdots & A_{1n} & a_1 \\ \vdots & \ddots & \vdots & \vdots \\ A_{n1} & \cdots & A_{nn} & a_n \\ a_1 & \cdots & a_n & \alpha \end{bmatrix}$$

First of all, notice that since $B^t = B \implies B_{ij} = B_{ji} \forall i, j \in \{1, \ldots, n+1\}$.

Also, note that $A_{ij} = B_{ij} = B_{ji} = A_{ji} \forall i, j \in \{1, \ldots, n\}$, hence A is symmetric.

$$x^t \begin{bmatrix} A_{11} & \cdots & A_{1n} & a_1 \\ \vdots & \ddots & \vdots & \vdots \\ A_{n1} & \cdots & A_{nn} & a_n \\ a_1 & \cdots & a_n & \alpha \end{bmatrix} x = \begin{bmatrix} x_1 A_{11} + \cdots + x_n A_{1n} + x_{n+1} a_1 \\ \vdots \\ x_1 A_{n1} + \cdots + x_n A_{nn} + x_{n+1} a_n \\ x_1 a_1 + \cdots + x_n a_n + x_{n+1} \alpha \end{bmatrix}$$
$$= x_1 \left(x_1 A_{11} + \cdots + x_n A_{1n} + x_{n+1} a_1 \right) + \cdots + x_{n+1} \left(x_1 a_1 + \cdots + x_n a_n + x_{n+1} \alpha \right) \alpha \Leftrightarrow \beta \quad (1)$$

Take $x = (0, \ldots, 0, 1) \neq 0$)

$\implies x^t B x = \alpha > 0$, since B is positive definite.

Now, let $x \in \mathbb{R}^{n+1}$ such that $x \neq 0$ and $x_{n+1} = 0$. Denote $y = (x_1, \ldots, x_n)^t \in \mathbb{R}^n$. From (1) it's evident that:

$x^t B x = y^t A y$, and since $x \neq 0 \implies y \neq 0$, and since $x^t B x > 0$, it follows that $y^t A y > 0 \forall y \neq 0$. A is positive definite.

Now,

$$B = LL' \Leftrightarrow \begin{bmatrix} A & a \\ a' & \alpha \end{bmatrix} = LL' \Leftrightarrow \begin{bmatrix} A & a \\ a' & \alpha \end{bmatrix} = \begin{bmatrix} C & 0 \\ v' & \beta \end{bmatrix} \begin{bmatrix} C' & v \\ 0 & \beta \end{bmatrix} = \begin{bmatrix} CC' & Cv \\ (Cv)' & v'v \end{bmatrix}$$

where C is lower triangular, $v \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$.

$$\implies CC' = A$$
$$Cv = A \Leftrightarrow v = C^{-1}a$$

$$(C^{-1}a)^{-1}C^{-1}a + \beta^2 = \alpha \Leftrightarrow a'(C^{-1})'C^{-1}a + \beta^2 = \alpha \Leftrightarrow a'A^{-1}a + \beta^2 = \alpha \Leftrightarrow \beta = \sqrt{\alpha - a'A^{-1}a}$$

$$\implies L = \begin{bmatrix} C & 0 \\ (C^{-1}a)' & \sqrt{\alpha - a'A^{-1}a} \end{bmatrix}$$

where $CC' = A \Rightarrow C = \sqrt{A}$

Exercise 17

```
gaussSolve <- function(eps){
  lapply(eps, function(e){
    A <- matrix( c(e,1,1,1), nrow = 2 )
    b <- c(1+e, 2)

    M <- matrix(c(1,-1/e,0,1), nrow = 2)

    backsolve(r = M%*%A, x = M%*%b)
  })
}

gaussSolve( 10^(-2*(1:10)) )
```

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

```
## [2,]    1
##
## [[6]]
##           [,1]
## [1,] 0.9998669
## [2,] 1.0000000
##
## [[7]]
##           [,1]
## [1,] 0.9992007
## [2,] 1.0000000
##
## [[8]]
##          [,1]
## [1,] 2.220446
## [2,] 1.000000
##
## [[9]]
##      [,1]
## [1,]    0
## [2,]    1
##
## [[10]]
##      [,1]
## [1,]    0
## [2,]    1
```

As the value of ϵ decreases, it is possible to see that the solution given by the backsolve strays away from the real solution. This has to do with the fact that we are introducing very small numbers in the computation, which due to floating point arithmetic and ill-conditioning make the problem not as stable.

Exercise 18

Let $A = UDV'$ be the svd of A. Then:

$$\begin{aligned}\frac{||Ax||_2}{||x||_2} &= \frac{||UDV'x||_2}{||x||_2} \\ &= \frac{\sqrt{(UDV'x)'(UDV'x)}}{||x||_2} \\ &= \frac{||DV'x||_2}{||x||_2}\end{aligned}$$

Put $y = V'x$, then $Vy = x$, so $||y||_2^2 = x'VV'x = x'x = ||x||_2^2$. Now:

$$\begin{aligned}\frac{||Ax||_2}{||x||_2} &= \frac{||Dy||_2}{||y||_2} \\ &= \frac{\sqrt{(Dy)'Dy}}{\sqrt{y'y}} \\ &= \frac{\sqrt{y'D^2y}}{\sqrt{y'y}} \\ &= \sqrt{\frac{\sum \sigma_i^2 y_i^2}{\sum y_i^2}}\end{aligned}$$

Note maximizing $\sqrt{\cdot}$ is equivalent to maximizing $\cdot$. Denote $c = \sum y_i^2$ and note that $\sum \frac{y_i}{c} = 1$, and $0 \leq \frac{y_i}{c} \leq 1$ for all i.

Then, we want to maximize $\sum \sigma_i^2 \frac{y_i^2}{c}$. Note that this is essentially a weighted average, with values σ_i and weights $\frac{y_i^2}{c}$. Note that:

$$0 \leq \sum \sigma_i^2 \frac{y_i^2}{c} \leq max\{\sigma_i^2\}$$

Also, we can attain $max\{\sigma_i^2\} = \sigma_1$ (assuming the singular values are ordered in a decreasing fashion) by taking $y = (1, 0, \ldots, 0)^t$. Therefore:

$$||A||_2 = \max_{x \neq 0} \frac{||Ax||_2}{||x_2||} = \max_{y \neq 0} \sqrt{\frac{\sum \sigma_i^2 y_i^2}{\sum y_i^2}} = \sqrt{max\{\sigma_i^2\}} = \sigma_1$$

```
A2norm <- function(A){
  max(svd(A)$d, na.rm = T)
}
```

Exercise 19

Let $A = UDV'$ be the svd of A. Then the svd of A^{-1} is $A^{-1} = VD^{-1}U'$.

Let $\sigma_1, \ldots, \sigma_n$ be the non-zero singular values of A, and w.l.o.g assume $\sigma_1 \geq \ldots \geq \sigma_n$. Then: $\frac{1}{\sigma_1} \leq \ldots \leq \frac{1}{\sigma_n}$ are the singular values of A^{-1}.

Then, $||A||_2||A^{-1}||_2 = \sigma_1 \cdot \frac{1}{\sigma_n}$

```
cond <- function(A){
  sv <- svd(A)$d
  max(sv, na.rm = T)/min(sv, na.rm = T)
}
```

Exercise 20

Note that:

$$\frac{||\Delta x||}{||x||} = \frac{||A^{-1}\Delta b||}{||x||} \leq \frac{||A^{-1}|| \cdot ||\Delta b||}{||x||}$$

Now, note that $b = Ax \Rightarrow ||b|| = ||Ax|| \leq ||A|| \cdot ||x||$, then:

$$\frac{||b||}{||A||} \leq ||x|| \Rightarrow \frac{||A||}{||b||} \geq \frac{1}{||x||}$$

Thus:

$$\frac{||A^{-1}|| \cdot ||\Delta b||}{||x||} \leq \frac{||A|| \cdot ||A^{-1}|| \cdot ||\Delta b||}{||b||} = \kappa(A)\frac{||\Delta b||}{||b||}$$

Exercise 21

```
cmult <- function(A, v) sweep(A, 2, v, `*`)

solve21 <- function(eps){

  lapply(eps, function(e){
    A <- matrix(c(1,1-e,1+e,1), nrow = 2)
    b <- c(1+e+e^2, 1)

    cNum <- cond(A)
    svd_A <- svd(A)



    Inverse <-  tryCatch(expr = solve(A, b),
                         error = function(e) "Matrix is numerically singular")
    QR <- tryCatch(expr = qr.solve(A, b),
                   error = function(e) "Matrix is numerically singular")

    SVD <- tryCatch(expr = cmult(svd_A$v, 1/svd_A$d) %*% crossprod(svd_A$u, b),
                    error = function(e) "Matrix is numerically singular")

    list(Epsilon = e,
         Condition = cNum,
         Inverse = Inverse,
         QR = QR,
         SVD = SVD)
  })
}

test <- solve21(c(1e-1, 1e-3, 1e-6, 1e-9 ,sqrt(.Machine$double.eps)))
test
```

```
## [[1]]
## [[1]]$Epsilon
## [1] 0.1
##
## [[1]]$Condition
## [1] 401.9975
##
## [[1]]$Inverse
## [1] 1.0 0.1
##
## [[1]]$QR
```

```
## [1] 1.0 0.1
##
## [[1]]$SVD
##      [,1]
## [1,]  1.0
## [2,]  0.1
##
##
## [[2]]
## [[2]]$Epsilon
## [1] 0.001
##
## [[2]]$Condition
## [1] 4000002
##
## [[2]]$Inverse
## [1] 1.000 0.001
##
## [[2]]$QR
## [1] 1.000 0.001
##
## [[2]]$SVD
##              [,1]
## [1,] 1.0000000001
## [2,] 0.0009999999
##
##
## [[3]]
## [[3]]$Epsilon
## [1] 1e-06
##
## [[3]]$Condition
## [1] 4.000663e+12
##
## [[3]]$Inverse
## [1] 1.000001 0.000000
##
## [[3]]$QR
## [1] "Matrix is numerically singular"
##
## [[3]]$SVD
##                [,1]
## [1,]  1.0001575353
## [2,] -0.0001565352
##
##
## [[4]]
## [[4]]$Epsilon
## [1] 1e-09
##
## [[4]]$Condition
## [1] 2.547621e+16
##
## [[4]]$Inverse
```

```
## [1] "Matrix is numerically singular"
##
## [[4]]$QR
## [1] "Matrix is numerically singular"
##
## [[4]]$SVD
##      [,1]
## [1,]  1.5
## [2,] -0.5
##
##
## [[5]]
## [[5]]$Epsilon
## [1] 1.490116e-08
##
## [[5]]$Condition
## [1] 2.547621e+16
##
## [[5]]$Inverse
## [1] "Matrix is numerically singular"
##
## [[5]]$QR
## [1] "Matrix is numerically singular"
##
## [[5]]$SVD
##      [,1]
## [1,]  1.5
## [2,] -0.5
```

Notice that as ϵ increases so does the condition number of the matrix. As such, the accuracy of the solutions decreases significantly. Notice that standard solve and QR solve both give singular errors, while the SVD does not. However, the solutions in these cases are not great.

Exercise 23

A is a $n \times n$ matrix of rank 1 so $A = uv'$, $u, v \neq 0$

a)

Note that $Au = uv'u = (v'u)u = (u'v)u \Rightarrow u'v$ is an eigenvalue and u is an eigenvector of A.

b)

Let $\lambda \in \mathbb{R}$ s.t. $v \perp x \Rightarrow Ax = uv'x = 0x$

Recall $rank(A) + dim(ker(A)) = n \Rightarrow 1 + dim(ker(A)) = n$, $\Rightarrow dim(ker(A)) = n - 1 \Rightarrow 0$ is the only other eigenvalue of A.

c)

We want to find U, V, D s.t. $A = UDV'$, U, V orthogonal, D diagonal and non-negative entries.

Consider $U = \left(\frac{u}{\|u\|}, u_2, \ldots, u_n\right), V = \left(\frac{v}{\|v\|}, v_2, \ldots, v_n\right)$

$$D = \begin{pmatrix} \|u\|\|v\| & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

where $u_2, ..., u_n$ and $v_2, ..., v_n$ are vectors s.t. U, V respectively are orthogonal matrices, found through for example Gram Schmidt.

$$\Rightarrow UDV' = uv' = A$$

d)

Given a starting non-zero vector x_0:

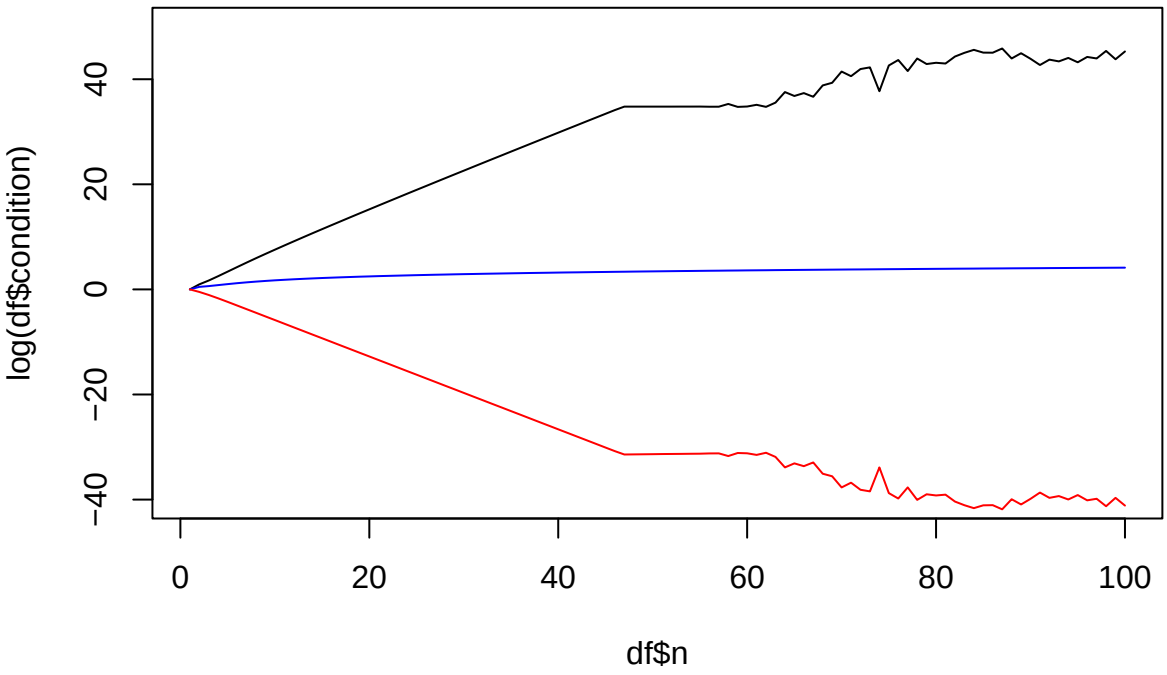
Case 1: $x0 \perp v \Rightarrow Ax_0 = uv'x_0 = 0 \Rightarrow x_k = 0 \quad \forall k > 0 \Rightarrow$ The iteration never converges. *Case 2:* $x_0 = \alpha u \Rightarrow Ax_0 = uv'\alpha u = (v'u)\alpha u \Rightarrow 0$ iterations. *Case 3:* None of the above $\Rightarrow x_1 = Ax_0 = uv'x_0 \Rightarrow x_2 Ax_1 = uv'x_1 = uv'uv'x_0 = (v'u)uv'x_0 = (v'u)x_1 \Rightarrow 1$ iteration

Exercise 26

```
ex26 <- function(n){
  lapply(n, function(i){
    A  <- diag(1, nrow = i)
    A[upper.tri(A)] <- -1

    svdA <- svd(A)$d

    list(n = i,
         maxSV = max(svdA, na.rm = T),
         minSV = min(svdA, na.rm = T),
         condition = max(svdA, na.rm = T)/min(svdA, na.rm = T))
  })

}

tmp <- ex26(c(1:100))
```

We verify that the condition number of the matrices grows exponentially, due mostly to the decay of the last (smallest) singular value, as can be seen in the following plot:

```
df <- do.call(rbind.data.frame, tmp)

plot(df$n, log(df$condition), type = "l", ylim = c(-40,50))
lines(df$n, log(df$maxSV), col = "blue")
lines(df$n, log(df$minSV), col = "red")
```



Exercise 28

Let $A = UDV'$, and assume $rank(A) = k \leq min(n, m)$. Write the following block matrices such that the dimensions match:

$$D = \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \qquad U = (U_1, U_2), \qquad V = (V_1, V_2)$$

Where Σ is a diagonal matrix whose elements are the non-zero singular values of A. Then:

$$UDV'x = (U_1, U_2) \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} U_1 \Sigma V_1' x_1 \\ 0 \end{pmatrix}$$

Note: Here there is a slight abuse of notation: When stating

Let's see $range(A) = range(U)$: Given $y \in range(A)$, there exists x such that $Ax = y$, then $UDV'x = y$. Take $x* = DV'x$, then $Ux* = y$, so $y \in range(U)$.

Given $y \in range(U)$, there exists x such that $Ux = y$. Take $x* = (\Sigma^{-1} V_1' x_1, 0)^t$, which exists by construction of Σ and V_1. Therefore $Ax* = UDV'x* = y$. Therefore $y \in range(A)$.

Since U is orthogonal, its linearly independent columns (those associated to the non-zero singular values of A) create an orthonormal basis for its range, and therefore also for the range of A.

Similarly, it is possible to prove $ker(A) = ker(V')$, therefore, the columns of V associated to the zero singular values of A create an orthonormal basis for $ker(A)$.

```
rangeBasis <- function(A, eps = .Machine$double.eps){
  n <- nrow(A)
  m <- ncol(A)

  svd_A <- svd(A, nu = n, nv = m)
```

```
  indx <- which(svd_A$d >= svd_A$d[1]*max(n,m)*eps)

  as.matrix(svd_A$u[, indx ])
}

kernelBasis <- function(A, eps = .Machine$double.eps){
  n <- nrow(A)
  m <- ncol(A)

  svd_A <- svd(A, nu = n, nv = m)

  sigma <- svd_A$d
  if(length(sigma) < m) sigma <- c(sigma, rep(0, m-length(sigma)))

  indx <- which(sigma < sigma[1]*max(n,m)*eps )

  as.matrix(svd_A$v[,indx])
}

set.seed(123)
A<-matrix(runif(24, min=5, max=20),6,4)
A[,3]<-0.5*A[,1]

rangeBasis(A)
```

```
##            [,1]        [,2]        [,3]
## [1,] -0.2897937  0.06516161  0.51960535
## [2,] -0.4949567  0.01832276  0.09489833
## [3,] -0.3875786 -0.31356331 -0.13683773
## [4,] -0.4184871  0.32226288 -0.75675015
## [5,] -0.4823505  0.45008258  0.35998653
## [6,] -0.3361948 -0.76854967 -0.00435157
```

```
kernelBasis(A)
```

```
##               [,1]
## [1,]  4.472136e-01
## [2,]  7.771561e-16
## [3,] -8.944272e-01
## [4,] -2.220446e-16
```

Exercise 31

```
vec <- function(A){
  c(A)
}

vech <- function(A){
  vec(A[lower.tri(A, diag = T)])
}

duplication <- function(n){
  S_mat <- diag(0, nrow = n)
```

```
  S_mat[lower.tri(S_mat, diag = T)] <- 1:(n*(n+1)/2)
  S_mat[upper.tri(S_mat, diag = F)] <- t(S_mat)[upper.tri(S_mat)]

  outer(c(S_mat), 1:(n*(n+1)/2), function(a,b) as.numeric(a == b) )
}

duplication(3)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    0    0    0    0    0
##  [2,]    0    1    0    0    0    0
##  [3,]    0    0    1    0    0    0
##  [4,]    0    1    0    0    0    0
##  [5,]    0    0    0    1    0    0
##  [6,]    0    0    0    0    1    0
##  [7,]    0    0    1    0    0    0
##  [8,]    0    0    0    0    1    0
##  [9,]    0    0    0    0    0    1
```

```
hilb <- function(n){
  i <- 1:n
  outer(i, i, function(i,j) 1/(i+j-1))
}


c(duplication(3) %*% vech(hilb(3)))
```

```
## [1] 1.0000000 0.5000000 0.3333333 0.5000000 0.3333333 0.2500000 0.3333333
## [8] 0.2500000 0.2000000
```

```
vec(hilb(3))
```

```
## [1] 1.0000000 0.5000000 0.3333333 0.5000000 0.3333333 0.2500000 0.3333333
## [8] 0.2500000 0.2000000
```

Exercise 32

```
Dn_sv <- function(n){
  tmp <- lapply(n, function(i){
    svd(duplication(i))$d
  })

  names(tmp) <- as.character(n)
  tmp
}

Dn_sv(1:10)
```

```
## $`1`
## [1] 1
##
## $`2`
## [1] 1.414214 1.000000 1.000000
##
```

```
## $`3`
## [1] 1.414214 1.414214 1.414214 1.000000 1.000000 1.000000
##
## $`4`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.000000 1.000000
##  [9] 1.000000 1.000000
##
## $`5`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
##  [9] 1.414214 1.414214 1.000000 1.000000 1.000000 1.000000 1.000000
##
## $`6`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
##  [9] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.000000
## [17] 1.000000 1.000000 1.000000 1.000000 1.000000
##
## $`7`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
##  [9] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [17] 1.414214 1.414214 1.414214 1.414214 1.414214 1.000000 1.000000 1.000000
## [25] 1.000000 1.000000 1.000000 1.000000
##
## $`8`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
##  [9] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [17] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [25] 1.414214 1.414214 1.414214 1.414214 1.000000 1.000000 1.000000 1.000000
## [33] 1.000000 1.000000 1.000000 1.000000
##
## $`9`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
##  [9] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [17] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [25] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [33] 1.414214 1.414214 1.414214 1.414214 1.000000 1.000000 1.000000 1.000000
## [41] 1.000000 1.000000 1.000000 1.000000 1.000000
##
## $`10`
##  [1] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
##  [9] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [17] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [25] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [33] 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214 1.414214
## [41] 1.414214 1.414214 1.414214 1.414214 1.414214 1.000000 1.000000 1.000000
## [49] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
```

It is possible to see that all singular values of D_n are 1 exactly n times and $\sqrt{2}$ all remaining times.

Exercise 33

```
elimination <- function(n){
  elim <- diag(1, nrow = n*(n+1)/2)
```

```
  S_mat <- diag(0, nrow = n)
  S_mat[lower.tri(S_mat, diag = T)] <- 1:(n*(n+1)/2)
  zero_cols <- which(c(S_mat) == 0)

  for(i in zero_cols){
    elim <- cbind( elim[,1:(i-1)], 0, elim[,i:ncol(elim)])
  }

  elim
}

elimination(3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    0    0    0    0    0    0    0    0
## [2,]    0    1    0    0    0    0    0    0    0
## [3,]    0    0    1    0    0    0    0    0    0
## [4,]    0    0    0    0    1    0    0    0    0
## [5,]    0    0    0    0    0    1    0    0    0
## [6,]    0    0    0    0    0    0    0    0    1
```

Exercise 34

```
Ln_svd <- function(n){
  tmp <- lapply(n, function(i){
    svd(elimination(i))
  })

  names(tmp) <- as.character(n)
  tmp
}

Ln_svd(c(1:3))
```

```
## $`1`
## $`1`$d
## [1] 1
##
## $`1`$u
##      [,1]
## [1,]    1
##
## $`1`$v
##      [,1]
## [1,]    1
##
##
## $`2`
## $`2`$d
## [1] 1 1 1
##
## $`2`$u
##      [,1] [,2] [,3]
```

```
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
##
## $`2`$v
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    0
## [4,]    0    0    1
##
##
## $`3`
## $`3`$d
## [1] 1 1 1 1 1 1
##
## $`3`$u
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    0    0    0
## [2,]    0    1    0    0    0    0
## [3,]    0    0    1    0    0    0
## [4,]    0    0    0    1    0    0
## [5,]    0    0    0    0    1    0
## [6,]    0    0    0    0    0    1
##
## $`3`$v
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]    1    0    0    0    0    0
##  [2,]    0    1    0    0    0    0
##  [3,]    0    0    1    0    0    0
##  [4,]    0    0    0    0    0    0
##  [5,]    0    0    0    1    0    0
##  [6,]    0    0    0    0    1    0
##  [7,]    0    0    0    0    0    0
##  [8,]    0    0    0    0    0    0
##  [9,]    0    0    0    0    0    1
```

It is possible to see that the singular value decomposition of L_n is given by $I \cdot I \cdot (L_n)'$.

Exercise 35

```
commutation <- function(m, n){

  A <- matrix(1:(m*n), nrow = n, ncol = m)

  outer(c(A), c(t(A)), function(a,b) as.numeric(a == b) )
}

commutation(3,3)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
##  [1,]    1    0    0    0    0    0    0    0    0
##  [2,]    0    0    0    1    0    0    0    0    0
##  [3,]    0    0    0    0    0    0    1    0    0
```

```
##  [4,]    0    1    0    0    0    0    0    0    0
##  [5,]    0    0    0    0    1    0    0    0    0
##  [6,]    0    0    0    0    0    0    0    1    0
##  [7,]    0    0    1    0    0    0    0    0    0
##  [8,]    0    0    0    0    0    1    0    0    0
##  [9,]    0    0    0    0    0    0    0    0    1
```

Exercise 36

```
Kmn_sv <- function(m,n){
  tmp <- list()
  for(i in m){
    for(j in n){
      tmp[[paste(i,j, sep = ',')]] <- svd(commutation(i,j))$d
    }
  }
  tmp
}

Kmn_sv(1:3, 1:3)
```

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

It is possible to see that all singular values are 1, regardless of the number of rows or columns.

Exercise 37

The Moore Penrose matrix A^+ satisfies $AA^+A = A$. Consider the svd decomposition of A, $A = UDV'$.

$$AA^+A = A \Leftrightarrow UDV'A^+UDV' = UDV'$$
$$\Leftrightarrow A^+ = VD^{-1}U'$$

```
moorePenrose <- function(A, tol = 1e-8){
  svd_A <- svd(A)
  sigma <- svd_A$d
  sigma_inv <- ifelse(sigma < tol, 0, 1/sigma)

  tcrossprod(cmult(svd_A$v, sigma_inv), svd_A$u)
}

A <- matrix(1:9,3,3)
A
```

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

```
moorePenrose(A)
```

```
##            [,1]          [,2]       [,3]
## [1,] -0.6388889 -5.555556e-02  0.5277778
## [2,] -0.1666667 -2.428613e-17  0.1666667
## [3,]  0.3055556  5.555556e-02 -0.1944444
```

```
A %*% moorePenrose(A) %*% A
```

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

Exercise 38

Notice that since $tr(AB) = tr(BA)$, then $tr(A'diag(w)A) = tr(diag(w)AA')$. Now:

$$\begin{aligned} tr(diag(w)AA') &= tr\left(\begin{pmatrix} w_1A_{11} & \cdots & w_nA_{1n} \\ \vdots & \ddots & \vdots \\ w_nA_{n1} & \cdots & w_nA_{nn} \end{pmatrix}\begin{pmatrix} A_{11} & \cdots & A_{n1} \\ \vdots & \ddots & \vdots \\ A_{1n} & \cdots & A_{nn} \end{pmatrix}\right) \\ &= w_1(A_{11}^2 + \ldots + A_{1n}^2) + \ldots + w_n(A_{n1}^2 + \ldots A_{nn}^2) \\ &= \sum(w_1 \ldots w_n)\begin{pmatrix} A_{11}^2 & \cdots & A_{n1}^2 \\ \vdots & \ddots & \vdots \\ A_{1n}^2 & \cdots & A_{nn}^2 \end{pmatrix} \end{aligned}$$

```
wcptrace <- function(A, w){
  sum(w %*% A^2)
}

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

```r
w <- c(2,4)

wcptrace(A, w)
```

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

```r
sum(diag(t(A) %*% diag(w) %*% A))
```

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

Exercise 39

Let

$$\begin{aligned} V = UDU' \Rightarrow V^{-1} &= UD^{-1}U' \\ \Rightarrow \det(2\pi V)^{-\frac{1}{2}} &= \det(2\pi UDU')^{-\frac{1}{2}} \\ &= (2\pi)^{-\frac{1}{2}} \det(D)^{-\frac{1}{2}} \\ &= (2\pi)^{-\frac{1}{2}} \left(\prod d_{ii}\right)^{-\frac{1}{2}} \end{aligned}$$

Also:

$$-(x-\mu)'V^{-1}(x-\mu) = -(x-\mu)'UD^{-1}U'(x-\mu)$$

```r
dmvnorm <- function(X, m, V){
  lambda <- eigen(V)$values
  U <- eigen(V)$vectors

  lambda_inv <- 1/lambda

  apply(X, 1, function(x){

    1/sqrt(prod(2*pi*lambda)) * exp((-t(x-m) %*%
                                     tcrossprod(cmult(U, lambda_inv), U) %*%
                                     (x-m))/2)

  })
}

X <- matrix(c(1:4), 2, 2)
m <- c(1:2)
V <- matrix(c(3,2,2,3), 2, 2)

dmvnorm(X, m, V)
```

```
## [1] 0.05272867 0.03534508
```