Problem Set 1

Jin Kweon

9/2/2017

Name: Jin Kweon

Lab: 11am - 1pm (section 102) (course number: 20980)

This is the problem 1.

Operation of matrices.

```
#Define matrix A, B, and C.
A <- matrix(c(1, 4, 2, 0, -3, 1), 2, 3)
B <- matrix(c(2, -1, 3, 2, 4, 0), 2, 3)
C <- matrix(c(0, 4, 1, -1, 0, -2), 2, 3)

#part a
A + B
```

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

```
#part b
(A + C) + B
```

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

```
#part c
A - (C + B)
```

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

```
#part d
-1 * (A + B)
```

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

```
#part e
(A - B) + C
```

```
##      [,1] [,2] [,3]
## [1,]   -1    0   -7
## [2,]    9   -3   -1
```

This is the problem 2.

Matrices operations without using R embeded functions.

part e) I used the formular following:

$$X_c = (I - \frac{1}{n}11^T)X$$

, where I is square identity matrix and 1 is column vector of ones.

part f) I used the formula following: Sample covaraince matrix =

$$\frac{1}{n-1}X^TX$$

, where X is mean-centered.

```
#Define a matrix X, by combinging vectors.
Y <- c(2, 4, 3, 7, 8, 9)
X1 <- c(1, 2, 5, 3, 7, 8)
X2 <- c(0, 3, 2, 4, 7, 7)
X3 <- c(9, 8, 4, 5, 2, 1)
X <- cbind(Y, X1, X2, X3)
rownames(X) <- c("a", "b", "c", "d", "e", "f") #Name the rows.
X
```

```
##   Y X1 X2 X3
## a 2  1  0  9
## b 4  2  3  8
## c 3  5  2  4
## d 7  3  4  5
## e 8  7  7  2
## f 9  8  7  1
```

```
#part a
one <- rep(1, nrow(X))
as.numeric(one %*% Y)
```

```
## [1] 33
```

```
#part b
as.numeric(one %*% X1) / nrow(X)
```

```
## [1] 4.333333
```

```
#part c
new <- Y %*% X2
new
```

```
##      [,1]
## [1,]  165
```

```
#part d
first <- (X3) %*% X3
second <- (one %*% X3)**2 / 6
first - second
```

```
##          [,1]
## [1,] 50.83333
```

```
#part e
#Use the formula in the slides.
centered <- (diag(nrow(X)) - ((one %*% t(one))/nrow(X))) %*% X
centered
```

```
##         Y         X1         X2         X3
## [1,] -3.5 -3.3333333 -3.8333333  4.1666667
## [2,] -1.5 -2.3333333 -0.8333333  3.1666667
## [3,] -2.5  0.6666667 -1.8333333 -0.8333333
## [4,]  1.5 -1.3333333  0.1666667  0.1666667
## [5,]  2.5  2.6666667  3.1666667 -2.8333333
## [6,]  3.5  3.6666667  3.1666667 -3.8333333
```

```
#part f
S <- (t(centered) %*% centered) / (nrow(centered) - 1)
S
```

```
##       Y        X1        X2        X3
## Y   8.3  6.200000  7.700000 -7.500000
## X1  6.2  7.866667  6.666667 -8.733333
## X2  7.7  6.666667  7.766667 -7.633333
## X3 -7.5 -8.733333 -7.633333 10.166667
```

This is the problem 3.

We try to define a scalar product (= dot product = Euclidean inner product) here. The Euclidean norms of two vectors and the angle between the two vectors are given. Here are the theorem and codes below:

$$\vec{a} \cdot \vec{b} = ||\vec{a}||\ ||\vec{b}||\ cos(\theta),\ where\ \theta\ is\ the\ angle\ between\ \vec{a}\ and\ \vec{b}.$$

```
#Function when you put two norms of vectors and between-angle, it computes the dot product.
dot_prod <- function(x, y, angle){
  x * y * angle
}

#part a
dot_prod(0.5, 4, cos(pi/4))
```

```
## [1] 1.414214
```

```r
#part b
dot_prod(4, 1, cos(pi/2))
```

```
## [1] 2.449294e-16
```

```r
#It is closed enough to be zero. ===> check with rounding!!!
round(dot_prod(4, 1, cos(pi/2)), 3)
```

```
## [1] 0
```

```r
#part c
dot_prod(1, 1, cos(pi * 2 / 3))
```

```
## [1] -0.5
```

This is the problem 4.

I define a projection operator function for this problem. It will be used later for Gram-Schmidt procedure in problem 5. I basically defined euclidean norm and used it for projection function.

```r
#Define a L2 (euclidean) norm of a vector
vnorm <- function(x){
  sqrt(t(x) %*% x)
}

#Define a projection x onto y.
proj <- function(x, y){
  (crossprod(y, x) / vnorm(y)^2) * y
}

#Test it.
u <- c(1, 3, 5)
v <- c(2, 4, 6)
proj(u, v)
```

```
## [1] 1.571429 3.142857 4.714286
```

This is the problem 5.

There are three vectors given for each part of the problems. I let u1 (the first vector of bases) be equal to x. Use proj() function from problem 4 and apply it to define the Gram-Schmidt procedure function in this problem. My goal is to transform given three vectors into orthonormal bases.

Here are the equations and codes, below:

$$u_1 \;=\; x$$

$$u_2 \;=\; y \;-\; proj_{u1}(y)$$

$$u_3 \;=\; z \;-\; proj_{u1}(z) \;-\; proj_{u2}(z)$$

$$e_1 \;=\; \frac{u_1}{||u_1||},\; e_2 \;=\; \frac{u_2}{||u_2||},\; and\; e_3 \;=\; \frac{u_3}{||u_3||}$$

Careful: By the definition, we could **ONLY** use Gram_schmidt procedure if u1, u2, and u3 (given vectors = original vectors) are lineary independent list of vectors in a vector space. Part a) already has linearly independent vectors; however, part b) does not have linear independent set of vectors. In part b), since none of the vectors are scalar multiple of each other, we should delete one of it even before we use the Gram_schmidt. By the convention, in my codes, I will delete the last one.

So, z2 (the third vector in part b) is almost a zero vector, and by the definition of the norm of zero vector, the norm of z2 should be closed to zero. So, e3 will be zero vector. Also, if we think the entire vector space is R^2, it does not make sense for us to have three orthogonal vectors in the space. However, R cannot tell me that trick, and they just gave me wrong answer. So, we only need to report the first two vectors for orthonormal basis from Gram_schmidt, since the last vector: e3 will be a zero vector and zero vector should not be in the basis. (And, also, span of any two vector in R^2 is a subspace of R^2, and since this is a subspace of R^2, the dimension of it should be smaller than dim(R^2) = 2. So, it makes sense for us to have only two vectors after Gram_schmidt procedure.)

```r
#part a
x1 <- c(1, 2, 3)

y <- c(3, 0, 2)
y1 <- y - proj(y, x1)

z <- c(3, 1, 1)
z1 <- z - proj(z, x1) - proj(z, y1)

e1 <- x1 / vnorm(x1)
e2 <- y1 / vnorm(y1)
e3 <- z1 / vnorm(z1)

#Set of vectors uk
list(x1, y1, z1)
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1]  2.35714286 -1.28571429  0.07142857
##
## [[3]]
## [1]  0.5148515  0.9009901 -0.7722772
```

```r
#Set of vectors ek
list(e1, e2, e3)
```

```
## [[1]]
## [1] 0.2672612 0.5345225 0.8017837
```

```
##
## [[2]]
## [1]  0.87758509 -0.47868278  0.02659349
##
## [[3]]
## [1]  0.3980149  0.6965260 -0.5970223
```

```
#Check whether this is orthonormal.
orthonormal_R3 <- function(x){
#round to three, so function sees orthogonal if it gets closed to 0.
  if(round(as.vector(crossprod(x[,1], x[,2])), 3) == 0){
    if(round(as.vector(crossprod(x[,1], x[,3])), 3) == 0){
      if(round(as.vector(crossprod(x[,2], x[,3])), 3) == 0){
#so, as long as all euclidean norm is equal to 1(= True), their norms are all 1.
        if(all(as.logical(sapply(list(x[,1], x[,2], x[,3]), vnorm))) == T){
          print("They are orthonormal.")
        }else F
      }else F
    }else F
  }else F
}


orthonormal_R3(cbind(e1, e2, e3))
```

```
## [1] "They are orthonormal."
```

```
#part b
x2 <- c(2, 1)

y2 <- c(1, 2)
y2 <- y2 - proj(y2, x2)

#Can ignore the third one, since we should have two vectors for the basis.
z2 <- c(1, 1)
z2 <- z2 - proj(z2, x2) - proj(z2, y2)


second_e1 <- x2 / vnorm(x2)
second_e2 <- y2 / vnorm(y2)

#ignore this e3.
second_e3 <- z2 / vnorm(z2)

#It is easy to see that z2 is a zero vector.
list(x2, y2, z2)
```

```
## [[1]]
## [1] 2 1
##
## [[2]]
## [1] -0.6  1.2
##
## [[3]]
## [1]  3.330669e-16 -2.775558e-16
```

```r
#Set of vectors ek
list(second_e1, second_e2)
```

```
## [[1]]
## [1] 0.8944272 0.4472136
##
## [[2]]
## [1] -0.4472136  0.8944272
```

```r
#Check whether this orthonormal.
orthonormal_R2 <- function(x){
#round to three, so function sees orthogonal if it gets closed to 0.
  if(round(as.vector(crossprod(x[,1], x[,2])), 3) == 0){
#so, as long as all euclidean norm is equal to 1(= True), their norms are all 1.
    if(all(as.logical(sapply(list(x[,1], x[,2]), vnorm))) == T){
      print("They are orthonormal.")
    }else F
  }else F
}

orthonormal_R2(cbind(second_e1, second_e2))
```

```
## [1] "They are orthonormal."
```

This is the problem 6.

I write a function lp_norm(), which is taking two inputs x (vector) and p (p-norm value p), with a default value p = 1. For example, when p = 2, I define a euclidean norm and when p = "max," I define it as L-max/chebyshev norm.

Here are the equation and codes, below:

$$||\vec{x}||_p \;=\; (|x_1|^p \;+\; ... \;+\; |x_n|^p)^{(\frac{1}{p})}$$

```r
lp_norm <- function(x, p = 1){
  if(is.numeric(p)){
    (sum(abs(x)^p))^(1/p)
  }
  else if(p == "max"){
    max(abs(x))
  }
}

#part a
zero <- rep(0, 10)

lp_norm(zero, 1)
```

```
## [1] 0
```

```r
#part b
ones <- rep(1, 5)

lp_norm(ones, 3)
```

```
## [1] 1.709976
```

```r
#part c
u <- rep(0.4472136, 5)

lp_norm(u, 2)
```

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

```r
#part d
u <- -40:0

lp_norm(u, 100)
```

```
## [1] 40.03297
```

```r
#part 2
u <- 1:1000

lp_norm(u, "max")
```

```
## [1] 1000
```

```r
# Or, the easiest way is to change a given vector into matrix by using as.matrix() and use norm functio
```

This is the problem 7.

Test each vector are orthogonal and has norm equal to 1, using L2 norm (euclidean). Three vectors u1, u2, and u3 are given in the question. So, the way how I define the function orthonormal() is, as three vectors are given, I compare the dot product being zero (upto 3 decimal points) for each vector. And then, I test whether all three vectors have norm of 1. (If at least one of the categories does not work, it is not orthonormal, and returns False.)

The second option works for every size of a square matrix. (first option only works for 3 * 3 square matrix.) The second way of doing it is to use matrix multiplication of a matrix transpose and itself is equal to identity matrix if and only if the matrix is orthonormal matrix.

```r
#Define three vectors.
u1 <- c(3, 1, 1) / sqrt(11)
u2 <- c(-1, 2, 1) / sqrt(6)
u3 <- c(-1, -4, 7) / sqrt(66)

#Combine all three vectors, to transform it to matrix.
new <- cbind(u1, u2, u3)
```

```r
#Define euclidean norm
vnorm <- function(x){
  sqrt(t(x) %*% x)
}

orthonormal <- function(x){
#round to three, so function sees orthogonal if it gets closed to 0.
  if(round(as.vector(crossprod(x[,1], x[,2])), 3) == 0){
    if(round(as.vector(crossprod(x[,1], x[,3])), 3) == 0){
      if(round(as.vector(crossprod(x[,2], x[,3])), 3) == 0){
#so, as long as all euclidean norm is equal to 1(= True), their norms are all 1.
        if(all(as.logical(sapply(list(x[,1], x[,2], x[,3]), vnorm))) == T){
          print("They are orthonormal.")
        }else F
      }else F
    }else F
  }else F
}



#Second option
orthonormal2 <- function(x){
  ortho_matrix <- t(x) %*% x
  for (i in 1:nrow(ortho_matrix)){
    for (j in 1:ncol(ortho_matrix)){
      x<- all(sapply((round(ortho_matrix[i, j], 3)), identical, diag(1, nrow(ortho_matrix))[i, j]))
    }
  }
  print(paste("The matrix is orthnormal?:", x))
}


orthonormal(new)
```

```
## [1] "They are orthonormal."
```

```r
orthonormal2(new)
```

```
## [1] "The matrix is orthnormal?: TRUE"
```

This is the problem 8.

Using the embedded data "USArrests," I define different types of matrices: diagonal, mean-centered, central, covariance, standardized, correlation matrics, etc.

```r
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```r
#part a
X <- as.matrix(USArrests)
class(X)
```

```
## [1] "matrix"
```

n and p are the number of rows and columns for X, respectively.

```r
#part b
n <- nrow(X)
p <- ncol(X)
n
```

```
## [1] 50
```

```r
p
```

```
## [1] 4
```

D is a diagonal matrix, and I am going to output sum of diagonal elements.

```r
#part c
D <- diag(n) / n
print(sum(diag(D)))
```

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

g is a column means vector matrix. (Same as apply(X, 2, mean))

```r
#part d
one <- rep(1, n)
g <- t(X) %*% D %*% one
print(g)
```

```
##             [,1]
## Murder     7.788
## Assault  170.760
## UrbanPop  65.540
## Rape      21.232
```

Xc is a mean-centered matrix.

```r
#part e
Xc <- X - one %*% t(g)
colMeans(Xc) #Closed to zero!
```

```
##        Murder       Assault      UrbanPop          Rape
##  2.469136e-15 -7.617018e-14 -6.252776e-15 -2.700062e-15
```

V is a covariance-variance matrix.

```r
#part f
#Q. why this is covaraince matrix? EXY - EX EY
V <- t(X) %*% D %*% X - g %*% t(g)
print(V)
```

```
##             Murder   Assault  UrbanPop      Rape
## Murder    18.59106  285.2411   4.29848  22.53158
## Assault  285.24112 6806.2624 306.02960 508.88368
## UrbanPop   4.29848  306.0296 205.32840  54.65272
## Rape      22.53158  508.8837  54.65272  85.97458
```

new_D ($= D_{\frac{1}{s}}$) is a p * p diagonal matrix with reciprocal of standard deviation of each column.

```r
#part g
new_D <- diag(1 / sqrt(diag(V)))
print(diag(new_D))
```

```
## [1] 0.23192522 0.01212120 0.06978715 0.10784872
```

Column mean should be approximately zero, and standard deviation of each column should be around 1, as Z is the standardized matrix.

```r
#part h
Z <- Xc %*% new_D

colMeans(Z) #mean ~ 0
```

```
## [1]  5.630219e-16 -9.207912e-16 -4.440892e-16 -2.925438e-16
```

```r
apply(Z, 2, sd) #sd ~ 1. R used sample mean/variance, and there is small computing difference.
```

```
## [1] 1.010153 1.010153 1.010153 1.010153
```

Both R and R2 are correlation matrices. Thus, it shoudl be equal to cor(X)

```r
#part i
#Q. why this is correlation matrix? SD(x)^-1 cov(x,y) SD(y)^-1
R <- new_D %*% V %*% new_D
R
```

```
##             [,1]      [,2]       [,3]      [,4]
## [1,] 1.00000000 0.8018733 0.06957262 0.5635788
## [2,] 0.80187331 1.0000000 0.25887170 0.6652412
## [3,] 0.06957262 0.2588717 1.00000000 0.4113412
## [4,] 0.56357883 0.6652412 0.41134124 1.0000000
```

```
#part j
R2 <- t(Z) %*% D %*% Z
R2
```

```
##             [,1]      [,2]       [,3]      [,4]
## [1,] 1.00000000 0.8018733 0.06957262 0.5635788
## [2,] 0.80187331 1.0000000 0.25887170 0.6652412
## [3,] 0.06957262 0.2588717 1.00000000 0.4113412
## [4,] 0.56357883 0.6652412 0.41134124 1.0000000
```

```
cor(X)
```

```
##              Murder   Assault   UrbanPop      Rape
## Murder   1.00000000 0.8018733 0.06957262 0.5635788
## Assault  0.80187331 1.0000000 0.25887170 0.6652412
## UrbanPop 0.06957262 0.2588717 1.00000000 0.4113412
## Rape     0.56357883 0.6652412 0.41134124 1.0000000
```