STA 141A
Fundamentals of Statistical Data Science

Fall 2016

Instructor: Debashis Paul

Lecture 4

Matrix operations

M %*% c(4,-3,1) # multiply matrix M to vector c(4,-3,1)

M + c(1,-2) # add the column vector c(1,-2) to each column of M

M2 = cbind(M,c(4,5)) # create matrix M2 by appending column c(4,5) to M

M3 = rbind(M2,c(5,6,-4)) # create matrix M3 by appending row c(5,6,-4) to M2

onealt = rep(c(1,-1),4) # creates the vector c(1,-1,1,-1)

M4 = cbind(onealt,c(-2,4,1,7)) # 4 x 2 matrix 1st column c(1,-1,1,-1) and 2nd column c(-2,4,1,7)

- For easier access of a data set, expressed as a matrix, it is useful to name its rows and/or columns

rownames(M) = c("first", "second") # name the rows of M

colnames(M) = c("a", "b","c") # name the columns of M

identical(M[,"a"],M[,1]) # returns TRUE

apply( ) function

- Use of apply( ) function; Syntax: apply(m,dimcode,f,args)

apply(M,1,sum) # computes sum of rows of M (returns a 2 x 1 vector)

apply(M,2,sd) # computes standard deviations of columns of M (a 3 x 1 vector)

- Also useful while creating arrays and accessing its elements

namelist=list(c("x", "y","z"), c("a", "b"), c("P", "Q"))

A = array(c(4:15), dim=c(3,2,2), dimnames=namelist) # creates a 3 x 2 x 2 array

A[, , 1] # same as A[, , "P"] , i.e., slice "P" (or first slice) of array A (a 3 x 2 matrix)

A[1:2, 2,] # submatrix of A corresponding to

indices 1:2 of first dimension and index 2 of second dimension

Handling categorical data : Titanic

- R has many `built-in' data sets as part of package "datasets". Look at the full list by typing data( ).
- Work with dataset **Titanic**, which casualty figures from sinking of the ship *Titanic*, categorized by class (1^{st}, 2^{nd}, 3^{rd}, Crew), gender (Male, Female), age (Child, Adult) and survival indicator (yes, no)

```
class(Titanic)  # returns table (we will learn more about it, for now think of it as an array)
dim(Titanic) # returns c(4,2,2,2)
dimnames(Titanic) # returns a list with four fields containing the names of the dimensions
apply(Titanic,4,sum) # how many survived and how many died ?
apply(Titanic,1,sum) # how many passengers in each class ?
apply(Titanic,c(2,4),sum) # distribution of survivors among male and female passengers
apply(Titanic,c(3,4),sum) # distribution of survivors across age groups
```

List : Second example

- Lists in R are structures that can be used to combine data that are of different types

```
bob = list(name="Bob",age=19,school="UC Davis",GPA=3.2,resident=T)
bob$name # "Bob"
bob[[3]] # "UC Davis"
bob[["age"]] # 19
bob[1:2] # returns first two fields of bob as a list; however bob[[1:2]] will not work
bob[[4]] = 3.35  # changes value of the field GPA to 3.35
names(bob)  # returns a vector consisting of names of the fields
length(bob) # returns 5
```

List : Third example

- We can append elements to an existing list

z = list(a="abc",b=10) # creates a list with two fields, named "a" and "b"

z$c = "new element" # adds a third field "c" with value "new element"

z[[4]] = 28 # adds a fourth field (with no name) with value 28

z[5:7] = c(T,F,T) # adds 5th, 6th and 7th fields (without name) with logical values

z$matrix = matrix(rnorm(6),nrow=2,ncol=3) # adds the 8th field which is a matrix

z$list = list(p="P",q=c(2,7,-9),r=c(T,F)) # adds the 9th field which is a list

names(z)[4:7] = letters[4:7] # assigns values "d", "e", "f" "g" as names of 4th to 7th fields

z[c(4,6)] = NULL # eliminate 4th and 6th fields of the list z

lapply( ) and sapply( )

- We can use the functions lapply( ) and sapply( ) to apply a function to the elements of a list

```
lapply(list(1:7,25:29),median) # returns a list with two fields with values 4 and 27

sapply(list(1:7,25:29),median) # returns the vector c(4,27)

lapply(1:5,seq) # returns a list with 5 fields with j-th field being the vector 1:j for j = 1,…,5

u = NULL # creates a NULL object

u[[1]] = rnorm(50) # creates the list u with first field being a vector of 50 i.i.d. N(0,1) r.v.

u[[2]] = rchisq(100,3) # adds a second field to u, which is a vector 100 i.i.d. chi-square(3) r.v.

sapply(u,quantile,seq(0.1,0.9,0.1)) # calculates quantiles for each vector corresponding to prob. 0.1,…,0.9

lapply(u,summary) # gives 5 point summary for each vector
```

for loop

- We can use for loop to repeat a task by cycling over the elements of a vector

```
x = c(4.1,2.3,-2.2,3.1,4.0,3.3); y = numeric(0)
for(i in 1:length(x)){ y[i] = sum(x[1:i]^2) } # sum of x[1:i]^2
```

- We can often use sapply( ) to avoid using a for loop

```
z = list(a=1:3,b=5:10,c=10:50,d=seq(-2,2,1))
zm = numeric(0)
for(i in 1:length(z)){ zm[i] = median(z[[i]])}  # compute the median of each field (numeric vector) of z
sapply(z,median) # returns the same value as zm
```