R Language

LRS

University of Guelph

July-Aug 2012

Every command is an object and every object has some parameters that need to be given to it. Thus, the basic structure is

```
command( arg1, arg1, ... )
```

Data

How to get data into R?

1. Enter the data manually.
2. Read data from a file.
3. Have R create your data.

Enter Data

```
A = matrix(data=c(3,-1,-2,4),byrow=TRUE,ncol=2)
```

$$\mathbf{A} = \begin{pmatrix} 3 & -1 \\ -2 & 4 \end{pmatrix}$$

```
ages = c(23, 14, 38, 54, 17)
mean(ages)
var(ages)

N = length(ages)
average = sum(ages)/N
yy = sum(ages*ages)
vage = (yy - average*sum(ages))/(N-1)
```

Matrix multiplication

```
A = matrix(data=c(3,-1,-2,4),byrow=TRUE,ncol=2)
B = matrix(data=c(1,-1,3,1,1,-1),byrow=TRUE,ncol=3)
M = A %*% B
M
```

Enter matrices **A** and **B**. **A** times **B** is conformable, but not **B** times **A**.
Matrix multiplication in R is given by %*%.
What happens if you use *

Direct Product

```
A = matrix(data=c(3,-1,-2,4),byrow=TRUE,ncol=2)
B = matrix(data=c(1,-1,3,1,1,-1),byrow=TRUE,ncol=3)
M = A %x% B   # Note the difference, small x
M
```

$$\mathbf{M} = \begin{pmatrix} 3 & -3 & 9 & -1 & 1 & -3 \\ 3 & 3 & -3 & -1 & -1 & 1 \\ -2 & 2 & -6 & 4 & -4 & 12 \\ -2 & -2 & 2 & 4 & 4 & -4 \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} \end{pmatrix}$$

Direct Sum

```
block= function( ... ) {
    argv = list( ... )
    i = 0
    for( a in argv ){
       m = as.matrix(a)
       if(i == 0)
       rmat = m
    else
      {
       nr = dim(m)[1]
       nc = dim(m)[2]
       aa = cbind(matrix(0,nr,dim(rmat)[2]),m)
       rmat = cbind(rmat,matrix(0,dim(rmat)[1],nc))
       rmat = rbind(rmat,aa)
     }
  i = i+1
   }
  rmat
  }
```

Direct Sum

```
A = matrix(data=c(3,-1,-2,4),byrow=TRUE,ncol=2)
B = matrix(data=c(1,-1,3,1,1,-1),byrow=TRUE,ncol=3)
M = block(A,B)
```

$$\mathbf{M} = \begin{pmatrix} 3 & -1 & 0 & 0 & 0 \\ -2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 3 \\ 0 & 0 & 1 & 1 & -1 \end{pmatrix}$$

Joining Strings

```
SA = c(23, 14, 38, 54, 17)
SB = c(1,-1,1,-1,1)
M1 = cbind(SA,SB)  # order 5 x 2
M2 = rbind(SA,SB)  # order 2 x 5
```

Arguments going into rbind and cbind must have the same length.

Subsets, Partitions

```
A   # matrix of order 200 x 53
# keep only rows where first element
#  is greater than 10
B = A[A[ ,1]>10, ]

# keep rows 4,5, and 9, and columns
#   21 to 30
kr = c(4, 5, 9)
kc = c(21:30)
C = A[kr,kc]
```

Data Files

```
zdat = file.choose() # bodytrt.d
bods = read.table(file=zdat,header=FALSE,
  col.names=c("height","fore","foot","gender",
  "waist","head","GPA") )
# bods is a data frame, matrix
summary(bods)
mean( bods$GPA )
N = nrow(bods) # number of records in bods
```

Matrix Inversion

```
A               # square matrix
det(A)          # check determinant, not zero
AI = solve(A)   # inversion routine
help("solve")
C = AI %*% A    # identity?
```

Generalized Inverse

```
library(MASS)    # needed
det(A)           # is zero
G = ginv(A)      # Moore-Penrose inverse
AG = A %*% G     # not an identity
AGA = AG %*% A   # should equal A
```

"ginv" sometimes gives rounding errors, and thus, problems with solutions to equations, always check the results.

Rank

```
A                   # singular matrix
G = ginv(A)
AG = A %*% G
H = AG %*% AG       # idempotent H = AG
rnk = sum(diag(H))  # equals rank of A
         # sum(diag( )) is trace of matrix

# diag  - extracts diagonals of matrix into a string
#       OR creates a diagonal matrix from a string
```

User Functions

Users can make their own functions, "lrscrips.R"

```
M   # matrix to be reduced
ELMO = function(M,mr,mc,cons){
  k = nrow(M)
  OM = diag(rep(1,k))  # create identity
  OM[mr,mc] = cons
  X = OM %*% M
  return(X) }
```

What happens if "mr" is greater than "k"

Ordering a string of elements

```
S = c( 3, 6, -1, 2, 11, 4,  5)
ka = order(S)     # ascending
kd = order(-S)    # descending
ka
kd
S[ka]
S[kd]
```