STAT 302
Statistical Software and Its Applications
Other Data Objects

Yen-Chi Chen

Department of Statistics, University of Washington

Spring 2017

Matrices

- A matrix object is a rectangular $n \times m$ array of elements of same type: numerical, character, etc.
- n is the number of rows, m is the number of columns.
- Typically rows represent subjects, and columns represent different variables measured for each subject.
- The rectangular data structure ensures same number of measurements per subject.
- Having more than one variable per subject allows us to examine correlations between various measurements.
- We could also view such data as a collection of equal length variable vectors, stacked next to each other.

How to Create a Matrix

```
> A <- matrix(1:12,nrow=3,ncol=4,byrow=F)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

> B <- matrix(letters[1:12],nrow=3,byrow=T)
> B
     [,1] [,2] [,3] [,4]
[1,] "a"  "b"  "c"  "d"
[2,] "e"  "f"  "g"  "h"
[3,] "i"  "j"  "k"  "l"
```

Only `nrow` or `ncol` need to be specified.

Stacking Columns or Rows Using `cbind()` and `rbind()`

```
> A <- cbind(1:3,4:6,7:9,10:12)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

> B <- rbind(letters[1:4],letters[5:8],
+ letters[9:12])
> B
     [,1] [,2] [,3] [,4]
[1,] "a"  "b"  "c"  "d"
[2,] "e"  "f"  "g"  "h"
[3,] "i"  "j"  "k"  "l"
```

Naming Rows and Columns

```
> names(B)
NULL
> rownames(B) <- c("row1","row2","row3")
> B
     [,1] [,2] [,3] [,4]
row1 "a"  "b"  "c"  "d"
row2 "e"  "f"  "g"  "h"
row3 "i"  "j"  "k"  "l"
> colnames(B) <- c("col1","col2","col3","col4")
> B
     col1 col2 col3 col4
row1 "a"  "b"  "c"  "d"
row2 "e"  "f"  "g"  "h"
row3 "i"  "j"  "k"  "l"
```

Extracting Matrix Values by Index

```
> A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A[1:2,3:4]
     [,1] [,2]
[1,]    7   10
[2,]    8   11
```

Extracting Matrix Values by Name

```
> B
     col1 col2 col3 col4
row1 "a"  "b"  "c"  "d"
row2 "e"  "f"  "g"  "h"
row3 "i"  "j"  "k"  "l"

> B[c("row1","row3"),c("col2","col3")]
     col2 col3
row1 "b"  "c"
row3 "j"  "k"

> B[c("row1","row3"),2:3]
     col2 col3
row1 "b"  "c"
row3 "j"  "k"
```

Matrix Arithmetic

```
> Ar <- matrix(12:1,ncol=4)
> A+Ar
     [,1] [,2] [,3] [,4]
[1,]   13   13   13   13
[2,]   13   13   13   13
[3,]   13   13   13   13
```

Matrices are added by adding corresponding elements.
Same for `-` , `*` , `/` .
Matrices must have same dimension (columns and rows), otherwise the computer will cycle the smaller matrix.

Matrix/Vector Arithmetic

```
> A
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A+1:3
     [,1] [,2] [,3] [,4]
[1,]    2    5    8   11
[2,]    4    7   10   13
[3,]    6    9   12   15
> A+1:4
     [,1] [,2] [,3] [,4]
[1,]    2    8   10   12
[2,]    4    6   12   14
[3,]    6    8   10   16
```

Vectors are expanded by column to a conforming matrix
Same for `-` , `*` , `/` .

Matrix Multiply (Linear Algebra)

An $m \times n$ matrix C can be multiplied by an $n \times k$ matrix D using the command `C %*% D`

```
> C
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> D
     [,1] [,2] [,3]
[1,]    6    4    2
[2,]    5    3    1
> C%*%D
     [,1] [,2] [,3]
[1,]   21   13    5
[2,]   32   20    8
```

To partially verify: $1 \cdot 6 + 3 \cdot 5 = 21$, $1 \cdot 4 + 3 \cdot 3 = 13$

Matrix Vector Multiply (Linear Algebra)

An $m \times n$ matrix C can be multiplied by an $n \times 1$ vector d using the same command `C %*% d`

```
> C
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> d <- c(2,3)
> C%*%d
     [,1]
[1,]   11
[2,]   16
```

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 2 + 3 \cdot 3 \\ 2 \cdot 2 + 4 \cdot 3 \end{pmatrix} = \begin{pmatrix} 11 \\ 16 \end{pmatrix}$$

In-class Exercises - 1

Set `A <- matrix(1:9, nrow=3)`. Try the followings:

```
A
A[2,2]
A[1,]
A[,3]
```

Also try the followings

```
A[1,] = 0
A
A[,2] = c(-1,-2)
A
```

Think about what happened.

Inverting a Square Matrix

For some square matrices G we can find a matrix G^{-1} such that by matrix multiply we get $GG^{-1} = G^{-1}G = I$. G^{-1} = `solve(G)`. Here I is the identity matrix, 1's on diagonal, 0's off diagonal.

```
> G <- matrix(1:4,ncol=2)
> G
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> solve(G)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
> solve(G)%*%G
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Solving an $n \times n$ System of Equations

For a given $n \times n$ matrix $A = (a_{ij})$ and given vector $b = (b_1, \ldots, b_n)$ solve the following equations for the unknown vector $x = (x_1, \ldots, x_n)$

$$\begin{aligned} a_{11}x_1 + \ldots + a_{1n}x_n &= b_1 \\ \ldots &= \ldots \\ a_{n1}x_1 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

in matrix multiply form this is just $Ax = b$ for vectors $x = (x_1, \ldots, x_n)$ and $b = (b_1, \ldots, b_n)$. $x = A^{-1}Ax = A^{-1}b$. x can be obtained by the `solve` command via `solve(A,b)` $= x$. For some A (singular) the equations cannot be solved, and A^{-1} does not exist.

Lists

Lists are objects which are collections of other objects, such as data or function objects, lists, and lists of lists,...

```
> L <- list(M=1:4,A=letters[1:6],
+ F = function(x){x^2})
> L
$M
[1] 1 2 3 4

$A
[1] "a" "b" "c" "d" "e" "f"

$F
function (x)
{
    x^2
}
```

Indexing of Lists via [ ]

Within [ ] use an index vector or vector of component names

```
> L[1:2]
$M
[1] 1 2 3 4

$A
[1] "a" "b" "c" "d" "e" "f"

> L[c("M","A")]
$M
[1] 1 2 3 4

$A
[1] "a" "b" "c" "d" "e" "f"

# sublist of first 2 elements of the source list
```

Indexing of Lists via [[ ]] and $

Within [[ ]] use a single index or component name

```
> L[["A"]] # same as L$A
[1] "a" "b" "c" "d" "e" "f"
> L[[2]]
[1] "a" "b" "c" "d" "e" "f"
# You get the indicated list object,
# not a sublist

> L[[2]][3] # same as L$A[3]
[1] "c"

> L[[3]](6) # same as L$F(6)
[1] 36
```

The `$` referencing works only when list component is named.

List within a List

```
> LL <- list(num = 1:3,list(letters[3:1],
+ LETTERS[1:2]))
> LL
$num # first component has name num
[1] 1 2 3

[[2]] # 2nd list component does not have a name
[[2]][[1]] # 1st subcomponent of 2nd component
[1] "c" "b" "a"

[[2]][[2]]  # 2nd subcomponent of 2nd component
[1] "A" "B"

> LL[[2]][[1]] # 1st subcomp. of 2nd comp.
[1] "c" "b" "a"
> LL[[2]][[1]][2] # 2nd element of previous
[1] "b"
```

Data Frames

Data of different types can be captured in data frame objects.

```
> X <- data.frame(num=1:6,let=letters[6:1],
+ Date=as.Date("1965/5/15")+0:5)
> X
  num let       Date
1   1   f 1965-05-15
2   2   e 1965-05-16
3   3   d 1965-05-17
4   4   c 1965-05-18
5   5   b 1965-05-19
6   6   a 1965-05-20
> str(X)
'data.frame':   6 obs. of  3 variables:
 $ num : int  1 2 3 4 5 6
 $ let : Factor w/ 6 levels "a","b","c","d",..: 6 5
 $ Date: Date, format: "1965-05-15" "1965-05-16" ..
```

The Nature of Data Frames

A data frame is really a special list, with the restriction that all its components are vectors of various types, all of the same length.

Referencing is the same as with lists

```
> X[[1]] # same as X$num
[1] 1 2 3 4 5 6
```

Note that `X$let` is automatically a factor.

To keep strings as character, use `stringsAsFactors=F` in `data.frame()`.

stringsAsFactors=F in data.frame()

```
> X <-data.frame(num=1:6,let=letters[6:1],
+ Date=as.Date("1965/5/15")+0:5,
+ stringsAsFactors=F)
> X[1:3,2:3] # extract from data frames ~ matrices
  let       Date
1   f 1965-05-15
2   e 1965-05-16
3   d 1965-05-17
> str(X[1:3,2:3])
'data.frame':   3 obs. of  2 variables:
 $ let : chr  "f" "e" "d"
 $ Date: Date, format: "1965-05-15" "1965-05-16" ..
```

Why do we want to use data.frame?

Many datasets have different types of attributes. Here is an example from the *CO2* dataset in R.

```
> head(CO2)
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
> is.data.frame(CO2)
[1] TRUE
```

Try `str(CO2)`.

In-class Exercises - 2

What would happen if we `cbind` vectors with different structures? Try the following:

```
cbind(c(1:6), letters[1:6])
str(cbind(c(1:6), letters[1:6]))
```

Also try the following:

```
X <-data.frame(num=1:6, let=letters[6:1],
stringsAsFactors=F)
as.matrix(X)
is.character(X)
is.character(as.matrix(X))
is.character(X$let)
```

Think about what happened.