Introduction: Statistical Computing with R

Part 2: matrices and data frames

Complied by: Dr Birgit Erni (amended by Allan Clark)

Department of Statistical Sciences, University of Cape Town

28 February 2021

Outline

- matrices and matrix algebra
- data frames

Matrices

$$X = \begin{bmatrix} 1 & 1.2 & 3 \\ 1 & 5.6 & 10 \\ 1 & 3.5 & 7 \end{bmatrix}$$

e.g. regression (design) matrix

Creating Matrices

1. Convert a vector into an n x m matrix. The following code will create a 4×4 matrix. Note that the matrix is created column wise, i.e. column one is filled up first, etc.. You can change this by specifying `byrow = T`.

```
numbers <- 1:16
m1 <- matrix(numbers, nrow = 4)
m1
```

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

Creating Matrices

2. With cbind() or rbind(): binds vectors column- or row-wise

```
numbers1 <- 1:4
numbers2 <- 11:14
m2 <- cbind(numbers1, numbers2)  # column-bind
m2
```

```
##      numbers1 numbers2
## [1,]        1       11
## [2,]        2       12
## [3,]        3       13
## [4,]        4       14
```

Creating Matrices

3. diagonal matrix

```
diag(4)           # 4 by 4 identity matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
diag(c(0.1, 0.2, 0.7))
```

```
##      [,1] [,2] [,3]
## [1,]  0.1  0.0  0.0
## [2,]  0.0  0.2  0.0
## [3,]  0.0  0.0  0.7
```

Calculations with matrices

```r
t(m1)                    # transpose of m1
dim(m1)                  # 4 by 4 matrix

m1 * m1                # element-wise multiplication

v1 <- rep(1, times = 4)
v1 %*% m1              # matrix multiplication
m1 %*% m1

m1 <- matrix(1:4,nrow = 2)
solve(m1)              # inverse of m1
solve(m1) %*% m1

rowSums(m1)
colMeans(m1)
```

Subsetting/indexing matrices

```
m3 <- matrix(1:12, nrow = 3)
m3
```

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

```
m3[2, ]      # row 2
```

```
## [1]  2  5  8 11
```

```
m3[, 3]      # column 3
```

```
## [1] 7 8 9
```

Subsetting/indexing matrices

```
m3 <- matrix(1:12, nrow = 3)
m3[2, 3]    # element at row 2, column 3
```

```
## [1] 8
```

```
m3[-1, ]    # all except row 1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    5    8   11
## [2,]    3    6    9   12
```

```
m3[c(1,3), ]    # row 1 and row 3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    3    6    9   12
```

Subsetting/indexing matrices

```
m3 <- matrix(1:12, nrow = 3)
m3[, c(1,3) ]     # col 1 and col 3
```

```
##      [,1] [,2]
## [1,]    1    7
## [2,]    2    8
## [3,]    3    9
```

```
m3[c(2,3), c(1,3) ]     # (col 1 and col 3) intersect (row2
```

```
##      [,1] [,2]
## [1,]    2    8
## [2,]    3    9
```

Prac 2

There are 2 trees in the middle of the Kalahari. On each tree birds of unknown species are sitting and feeling very hot. A bird from the first tree says to those on the second tree: "Hi – if one of you come to our tree then there will be the same number of us on each tree". "Yeah, right", says a bird from the second tree, "but if one of you comes to our tree, then we will be twice as many on our tree as on yours".[1]

Question: How many birds are on each tree? More specifically:

- Write up two equations with two unknowns.
- Solve these equations using the methods you have learned above.
- Don't use trial–and–error!

[1] adapted from https://www.math.uh.edu/~jmorgan/Math6397/day13/LinearAlgebraR-Handout.pdf

Logical operators (!, &, |, ==)

```r
y <- c(1, 2, 3, NA, 4, 5)

is.na(y) #test if each element of y is an NA
!is.na(y) #test if each element of y is not NA

y > 1 & y < 2 #and operator

y > 1 | y < 2 #or operator

y[!is.na(y)] #subset y
```

Logical operators (!, &, |, ==)

```r
y <- c(1, 2, 3, NA, 4, 5)

is.na(y)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE
```

```r
!is.na(y)
```

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

```r
y > 1 & y < 2
```

```
## [1] FALSE FALSE FALSE    NA FALSE FALSE
```

```r
y > 1 | y < 2
```

```
## [1] TRUE TRUE TRUE   NA TRUE TRUE
```

```r
y[!is.na(y)]
```

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

Character vectors

```
site <- c("CT", "JHB", "JHB", "DB")
filename <- "Rintro.R"     # vector of length 1
lett <- letters; letters[1:5]
treat <- rep(LETTERS[1:3], each = 3); treat
```

- 'letters' is a built in object that contains the lower case letters of the alphabet.
- 'LETTERS' is a built in object that contains the upper case letters of the alphabet.
- Notice you can have multiple lines of code on the same line. They should be separated by ';'
- 'rep' repeats something!

Character vectors

```
site <- c("CT", "JHB", "JHB", "DB")
filename <- "Rintro.R"     # vector of length 1
lett <- letters; letters[1:5]
```

```
## [1] "a" "b" "c" "d" "e"
```

```
treat <- rep(LETTERS[1:3], each = 3); treat
```

```
## [1] "A" "A" "A" "B" "B" "B" "C" "C" "C"
```

Types of R objects

- **data objects/structures:** vectors, matrices, data frames, lists, arrays, factors
- **functions**
- output from regression model

Data frames

- This is the way statisticians, and R, like to store data.
- Like a spreadsheet (rows and columns).
- Ideally: each row is an observation (case), each column is a variable (*tidy data*).
- Similar to matrices, but columns can be of different data types.

Example (one data set from R, one from Gapminder):

```
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

Data frames

```
library(gapminder)
#str(gapminder)  #uncomment and see the output
head(gapminder)  #the top six elements of gapminder
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

Extracting values from data frames

Similar to matrices

```
names(gapminder)[1:5] #first 5 variables of gapminder
```

```
## [1] "country"   "continent" "year"      "lifeExp"   "pop"
```

```
c(gapminder$country)[1:20]  #country column, as vector
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
```

```
head(gapminder["country"])
```

```
## # A tibble: 6 x 1
##   country
##   <fct>
## 1 Afghanistan
## 2 Afghanistan
## 3 Afghanistan
## 4 Afghanistan
## 5 Afghanistan
## 6 Afghanistan
```

Extracting values from data frames

Similar to matrices

```
## only SA rows
gapminder[gapminder$country == "South Africa", ]
```

```
## # A tibble: 12 x 6
##    country      continent  year lifeExp      pop gdpPercap
##    <fct>        <fct>     <int>   <dbl>    <int>     <dbl>
##  1 South Africa Africa     1952    45.0 14264935     4725.
##  2 South Africa Africa     1957    48.0 16151549     5487.
##  3 South Africa Africa     1962    50.0 18356657     5769.
##  4 South Africa Africa     1967    51.9 20997321     7114.
##  5 South Africa Africa     1972    53.7 23935810     7766.
##  6 South Africa Africa     1977    55.5 27129932     8029.
##  7 South Africa Africa     1982    58.2 31140029     8568.
##  8 South Africa Africa     1987    60.8 35933379     7826.
##  9 South Africa Africa     1992    61.9 39964159     7225.
## 10 South Africa Africa     1997    60.2 42835005     7479.
## 11 South Africa Africa     2002    53.4 44433622     7711.
## 12 South Africa Africa     2007    49.3 43997828     9270.
```

Some basic statistics

Frequency Tables

```
#table(continent) # will not work
```

- 'continent' is a variable that sits inside the 'gapminder' object.
- It has to be extracted using the '$' sign.
- ie gapminder$continent

Some basic statistics

Frequency Tables

```
#table(continent) # will not work
table(gapminder$continent)
```

```
##
##   Africa Americas     Asia   Europe  Oceania
##      624      300      396      360       24
```

```
with(gapminder, table(continent))
```

```
## continent
##   Africa Americas     Asia   Europe  Oceania
##      624      300      396      360       24
```

Some basic statistics

Frequency Tables

```
## avoid attach!
attach(gapminder)
table(continent)
```

```
## continent
##   Africa Americas     Asia   Europe  Oceania
##      624      300      396      360       24
```

```
detach(gapminder)
```

Summary Statistics

```
mean(gapminder$lifeExp)
```

```
## [1] 59.47444
```

```
summary(gapminder$gdpPercap)  # 5-number summary
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    241.2   1202.1   3531.8   7215.3   9325.5 113523.1
```

- Play around with the following functions: sd, var, min, range, quantile

Prac 3

1. Find mean, sd, min, max life expectancy for each country in the gapminder object.
2. For **linear regression**, parameter estimates can be found as follows.

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Here, Y is the response variable, and X is the design matrix.

The `cars` data (an R data set) contains two variables: speed and distance to stop. Fit a simple linear regression model to these data, i.e. find the $\hat{\beta}$ estimates, using the equation above, and matrix calculations in R.

Prac 3

3. Check that you get the same $\hat{\beta}$ estimates as when fitting the linear regression model using `lm()` in R.

```
m1 <- lm(dist ~ speed, data = cars)
summary(m1)
```

4. Check that the mean life expectancy in South Africa is 53.99317.