Introduction to Data Science

DSA1101

Semester 1, 2018/2019
Week 1, 14 August

Introduction to the R programming language

R is a programming language and software framework for statistical analysis and graphics distributed under the GNU general public license.

R can be downloaded from `http://cran.r-project.org/`

Linux/Mac OS X/Windows versions available

Basic commands in R

- R uses *functions* to perform operations

  To run a function called `funcname`, we type `funcname(input1,input2,...)` into the R console

- Example: the function `c()` concatenates the numbers inside the parentheses to form a vector

```
> x <- c(1,3,2,5)
> x
[1] 1 3 2 5
```

Basic commands in R

- Help file is available by typing `?funcname` into the console, for example

```
>?c
```

brings up the HTML page

c {base} R Documentation

Combine Values into a Vector or List

Description

This is a generic function which combines its arguments.

The default method combines its arguments to form a vector. All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.

Usage

```
## S3 Generic function
c(...)

## Default S3 method:
c(..., recursive = FALSE, use.names = TRUE)
```

Arguments

`...` objects to be concatenated.

`recursive`

Basic commands in R

- Addition, subtraction, multiplication and division are performed by +,-, * and / respectively
- Each element of a vector will be changed by operation with a constant, for example

```
> a <- c(1,2,3,4)
> a
[1] 1 2 3 4
> a + 5
[1] 6 7 8 9
> a - 10
[1] -9 -8 -7 -6
> a*4
[1]  4  8 12 16
> a/5
[1] 0.2 0.4 0.6 0.8
```

Basic commands in R

- Element-wise operations are possible for two vectors of the same length
- Length of a vector can be checked by the function `length()`

```
> x = c(1,6,2)
> x
[1] 1 6 2
> y = c(1,4,3)
> y
[1] 1 4 3
> length (x)
[1] 3
> length (y)
[1] 3
> x+y
[1] 2 10 5
```

Basic commands in R

- The function `sqrt()` returns the square root of each element of a vector
- The command `x^2` raises each element of x to the power 2; any powers are possible, including fractional or negative powers.

```
> x=c(2,3,9,16)
> sqrt(x)
[1] 1.414214 1.732051 3.000000 4.000000
> x^2
[1]   4   9  81 256
> x^3
[1]    8   27  729 4096
> x^-1
[1] 0.5000000 0.3333333 0.1111111 0.0625000
```

Basic commands in R

- We can quickly get a vector of numbers in sequence, $x = c(a_1, a_1 + 1, a_1 + 2, ..., b)$, using $x = a : b$
- We can quickly get a vector of numbers in sequence with a specified step, $x = c(a_1, a_1 + d, a_1 + 2d, ..., b)$, using $x = seq(a_1, b, d)$

```
> x = 1:5
> x
[1] 1 2 3 4 5
> x = 10:15
> x
[1] 10 11 12 13 14 15
> y = seq(1,10,2)
> y
[1] 1 3 5 7 9
```

Basic commands in R

- The function `ls()` provides a list of all the objects created so far
- The function `rm()` can delete any unwanted objects rm = remove

```
> ls()
[1] "a"   "x"  "y"
> rm(x,y)
> ls()
[1] "a"
> rm(a)
character (0)
```

- To remove all objects at once:

```
> rm(list=ls())
```

Basic statistics with R

Measures of central tendency

- Empirically speaking data shows a tendency to agglomerate around a central value. This is called central tendency. This central value can be used to summarize data. Suppose $x = c(x_1, x_2, ..., x_N)$ is a given vector of numbers.
- The mean is computed as

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- The function `mean()` computes this value in R:

```
> x=c(1,3,5,7,10)
> mean(x)
[1] 5.2
> sum(x)/length(x)
[1] 5.2
```

Measures of central tendency

- The sample median is computed as

$$median(x) = \begin{cases} x_{(N+1)/2}, & \text{if } N \text{ is odd} \\ \frac{x_{N/2}+x_{N/2+1}}{2}, & \text{if } N \text{ is even} \end{cases} \quad (1)$$

- The function `median()` computes this value in R:

```
> x=c(1,3,5,7,10)
> median(x)
[1] 5
> x=c(1,3,5,7,10,11)
> median(x)
[1] 6
```

Measures of central tendency

- The sample mode is the most frequent value in a data set
- Note that the function mode() in R returns the type of object, not the sample mode
- R allows users to write their own functions:

```
#function to get mode
> getmode <- function(v) {
+     uniqv <- unique(v)
+     uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
> x=c(1,3,5,7,10,10,11)
> getmode(x)
[1] 10
```

Measures of dispersion

- The tendency for data points to spread around a central value is called dispersion.
- The sample variance is computed as

$$var(x) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

- The function `var()` computes this value in R:

```
> x=c(1,3,5,7,10,10,11)
> var(x)
[1] 14.90476
```

Measures of dispersion

- The sample standard deviation computed by function `sd()` is the square root of the variance

$$sd(x) = \sqrt{var(x)}$$

```
> x=c(1,3,5,7,10,10,11)
> var(x)
[1] 14.90476
> sqrt(var(x))
[1] 3.860669
> sd(x)
[1] 3.860669
```

- The range is the difference between the largest and smallest values

```
> x=c(1,3,5,7,10,10,11)
> max(x)-min(x)
[1] 10
```

Measures of association

- For two data vectors $x = c(x_1, x_2, ..., x_N)$ and $y = c(y_1, y_2, ..., y_N)$, their sample covariance can be computed by the function `cov()` and is given by

$$cov(x, y) = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})$$

```
> x=c(2,4,7,1,10)
> y=c(1,6,3,10,9)
> cov(x,y)
[1] 1.95
#compute sample covariance from previous functions
> sum((x-mean(x))*(y-mean(y)))/(length(x)-1)
[1] 1.95
```

Measures of association

- The sample correlation coefficient for two data vectors can be computed by the function `cor()` and is given by

e.g height (cm) vs weight (kg)

$$r_{xy} = \frac{cov(x, y)}{sd(x)sd(y)}$$

- Unlike the sample covariance, the relation $-1 \leq r_{xy} \leq 1$ holds
- One can compare the degree of association using r_{xy}, which is invariant to scale and location changes

```
> x=c(2,4,7,1,10)
> y=c(1,6,3,10,9)
> cor(x,y)
[1] 0.1374092
> cov(x,y)/(sd(x)*sd(y))
[1] 0.1374092
#correlation invariant to scaling
> cor(100*x,100*y)
[1] 0.1374092
> cor(100*x,20*y)
[1] 0.1374092
```

Effects of scale and location changes

- Location changes: for two data vectors x and y, suppose we create two new data vectors $u = x + a$ and $v = y + b$, where a and b are constants
- Then the mean, median and mode for u will be different from those for x (similarly for v and y)
- Sample variance, standard deviation and covariance are invariant to changes in location

```
> x=c(2,2,7,1,10)
> y=c(1,6,3,10,10)
> u=x+10
> v=y-5
> stats_x=c(mean(x),median(x),getmode(x))
> stats_x
[1] 4.4 2.0 2.0
>  stats_u=c(mean(u),median(u),getmode(u))
> stats_u
[1] 14.4 12.0 12.0
```

Effects of scale and location changes

```
> x_stats= c(var(x),sd(x),cov(x,y),cor(x,y))
> x_stats
[1] 15.3000000  3.9115214  3.2500000  0.2045482
> u_stats= c(var(u),sd(u),cov(u,v),cor(u,v))
[1] 15.3000000  3.9115214  3.2500000  0.2045482
```

- r_{xy} is invariant to location changes

Effects of scale and location changes

- Scale changes: for two data vectors x and y, suppose we create two new data vectors $u = a * x$ and $v = b * y$, where a and b are constants
- Then the mean, median and mode for u will be different from those for x (similarly for v and y)
- Sample variance, standard deviation and covariance between the two data vectors will also be different

```
> x=c(2,2,7,1,10)
> y=c(1,6,3,10,10)
> u=-5*x
> v=-10*y
> x_stats= c(mean(x),median(x),getmode(x))
> x_stats
[1] 4.4 2.0 2.0
> u_stats= c(mean(u),median(u),getmode(u))
> u_stats
[1] -22 -10 -10
```

Effects of scale and location changes

```
> x_stats= c(var(x),sd(x),cov(x,y),cor(x,y))
> x_stats
[1] 15.3000000  3.9115214  3.2500000  0.2045482
> u_stats= c(var(u),sd(u),cov(u,v),cor(u,v))
> u_stats
[1] 382.5000000  19.5576072 162.5000000
    0.2045482
```

- Notice that sample variance changes multiplicatively by the square of the scale, $var(u) = (-5)^2 \times var(x)$
- While sample standard deviation changes according to the absolute value of the scale, $sd(u) = |-5| \times sd(x)$
- r_{xy} is invariant to scale changes (provided the constants a and b are of the same sign)

Data manipulation with `R`

Data manipulation in R

- Besides the console, data from external sources can be read directly into R
- Example: a comma-separated-value (CSV) file containing the annual sales in U.S. dollars for 10,000 retail customers

  Code comments are preceded by '#'

  You can either specify the complete path to file in function `read.csv()`:

```
# import a CSV file of the total annual sales
sales <- read.csv("c:/data/yearly_sales.csv")
```

Or set the working directory to the one containing the file `yearly_sales.csv`

```
setwd("c:/data")
sales <- read.csv("yearly_sales.csv")
```

Data manipulation in R

- R offers other functions to read in data with different default values for column separator and decimal symbol

Function	Headers	Separator	Decimal Point
`read.table()`	FALSE	""	"."
`read.csv()`	TRUE	","	"."
`read.csv2()`	TRUE	";"	","
`read.delim()`	TRUE	"\t"	"."
`read.delim2()`	TRUE	"\t"	","

True: skip first line intro; number separator, ./,

Data manipulation in R

- Examine the imported data

  The function head() displays the first few records in the dataset

```
# examine the imported dataset
head(sales)
  cust_id sales_total num_of_orders gender
1  100001      800.64             3      F
2  100002      217.53             3      F
3  100003       74.58             2      M
4  100004      498.60             3      M
5  100005      723.11             4      F
6  100006       69.43             2      F
```

Data manipulation in R

- Descriptive statistics

  The function summary() provides summary statistics such as mean, median and quantiles

```
# examine the imported dataset
summary(sales)
   cust_id          sales_total       num_of_orders     gender
 Min.   :100001   Min.   :  30.02   Min.   : 1.000   F:5035
 1st Qu.:102501   1st Qu.:  80.29   1st Qu.: 2.000   M:4965
 Median :105001   Median : 151.65   Median : 2.000
 Mean   :105001   Mean   : 249.46   Mean   : 2.428
 3rd Qu.:107500   3rd Qu.: 295.50   3rd Qu.: 3.000
 Max.   :110000   Max.   :7606.09   Max.   :22.000
```

Data manipulation in R

- Preliminary observations on data characteristics

  The function dim() tells us that the data in sales has 10000 observations or rows, and 4 variables or columns

```
> dim(sales)
[1] 10000     4
```

  The function names() tells us the variable names in the data set

```
> names(sales)
[1] "cust_id"       "sales_total"   "num_of_orders"
    "gender"
```

Data manipulation in R

- Preliminary observations on data characteristics

  The function `dim()` tells us that the data in `sales` has 10000 observations or rows, and 4 variables or columns

```
> dim(sales)
[1] 10000     4
```

  The function `names()` tells us the variable names in the data set

```
> names(sales)
[1] "cust_id"       "sales_total"   "num_of_orders"
     "gender"
```