

R programming for beginners

Ni Shuai

Computational Genome Biology
German Cancer Research Center (DKFZ)

November, 2016



Data frames

In one word, **data frame** is a matrix that doesn't require its columns to be of the same data type

Like a matrix,

- Data frame is a rectangular grids of elements
- Most functions in matrix also apply in data frames

Unlike a matrix,

- Empty or duplicated column/row names are not allowed in a data frame
- Data frame is a column-based structure



Create a data frame

Since data frame requires its columns to be equal-length vectors of different types, lets make some vectors first:

```
Name <- c('John', 'Wang', 'Michael', 'Mary')
Date.Of.Birth<- c("1989-03", "1987-10", "1990-02","1993-09")
Score=c(86, 99, 92, 87)
```

Then a data frame can be constructed with the `data.frame()` function:

```
Mydf<- data.frame(FirstName=Name, Birthday=Date.Of.Birth, Score=Score,
                  stringsAsFactors = TRUE)
# input vectors must be of same length
str(Mydf) # Check the structure of my data frame

## 'data.frame': 4 obs. of 3 variables:
## $ FirstName: Factor w/ 4 levels "John","Mary",...: 1 4 3 2
## $ Birthday : Factor w/ 4 levels "1987-10","1989-03",...: 2 1 3 4
## $ Score : num 86 99 92 87
```



Factors in R

Factor is a character vector in R which takes on a limited number of different values, these values are often referred to as **categorical values**

Using of factors helps to store character values efficiently and to avoid incorrect operational treatments

For example:

```
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data, levels=c(1,2,3))
fdata

## [1] 1 2 2 3 1 2 3 3 1 2 3 3 1
## Levels: 1 2 3

rdata = factor(data, labels=c("I", "II", "III"))
rdata

## [1] I  II  II  III I  II  III III I  II  III III I
## Levels: I II III
```



Manipulating a data frame

Indexing in data frame is the same as it is in a matrix:

- The `[]` operator is also used to index a data frame

```
Mydf[, 'Score']
## [1] 86 99 92 87

Mydf[nrow(Mydf),]
##   FirstName Birthday Score
## 4      Mary  1993-09     87
```

- Accessing columns in a data frame is more convenient by using `$`

```
Mydf$FirstName
## [1] John    Wang    Michael Mary
## Levels: John Mary Michael Wang

Mydf$Birthday
## [1] 1989-03 1987-10 1990-02 1993-09
## Levels: 1987-10 1989-03 1990-02 1993-09
```



Manipulating a data frame

Rownames and column names can also be changed as it is in a matrix:

```
rownames(Mydf)

## [1] "1" "2" "3" "4"

rownames(Mydf)[4]='5'
rownames(Mydf)

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

One can also retrieve column names by simply using `names()` function

```
names(Mydf)

## [1] "FirstName" "Birthday" "Score"

names(Mydf)[3]='Math.Score'; names(Mydf)

## [1] "FirstName" "Birthday" "Math.Score"
```



Merging two dataframes

Merge allows joining two data frames by one or more common key variables:

```
Hisdf<- data.frame(FirstName = c('John', 'Wang', 'Mary', 'Michael'),
PE.Score=c(77, 66, 82, 61), GM.Score=c(60, 62, 75, 72))
```

```
Mydf
```

```
##   FirstName Birthday Math.Score
## 1      John  1989-03         86
## 2      Wang  1987-10         99
## 3   Michael  1990-02         92
## 5      Mary  1993-09         87
```

```
merge(Hisdf, Mydf, by='FirstName')
```

```
##   FirstName PE.Score GM.Score Birthday Math.Score
## 1      John      77      60  1989-03         86
## 2      Mary      82      75  1993-09         87
## 3   Michael      61      72  1990-02         92
## 4      Wang      66      62  1987-10         99
```



Managing your work space

The workspace is your current R working environment and includes any user-defined objects (Variables, matrices, functions...), below are some useful functions to navigate through directories.

<code>getwd()</code>	Get the current location of current working directory
<code>setwd()</code>	Set the current location to a spicified directory
<code>list.files()</code>	Return the names of directories and files in the named directory
<code>ls()</code>	List the names of the objects in the current environment
<code>rm()</code>	Remove the specified object from the current environment

Note that (`\`) is a special character, hence the path pointing to some file cannot be specified as usual under Windows. Instead double it (`\\`) or use the `/` character.



Managing your work space

Exercises:

- Check what is your current working directory
- Check how many directories and files are in your current working directory, navigate further to one of the subfolders.
- Get back to the beginning directory using `setwd("../")`
- Merge the defined data.frames 'Mydf' and 'Hisdf' to 'Alldf', keep the final table 'Alldf' and remove the rest two from the current working environment

Tips

```
Name <- c('John', 'Wang', 'Michael', 'Mary')
Date.Of.Birth<- c("1989-03", "1987-10", "1990-02", "1993-09")
Score=c(86, 99, 92, 87)
Mydf<- data.frame(FirstName=Name, Birthday=Date.Of.Birth, Score=Score,
                  stringsAsFactors = TRUE)
```



Importing and exporting data

If you want to analyze data from some where else, you have to import the data into R, most commonly it will be a text file. To import a text file, one can easily do with the function `read.table()`
`read.table()` will always put your data into a `data.frame` object.

For example:

```
Auto<- read.table("http://www-bcf.usc.edu/~gareth/ISL/Auto.data",
                  header = TRUE)
# We can read from a URL instead of a local file.
```

Some useful arguments in `read.table()` function:

<code>file()</code>	Name of the file which the data are to be read from
<code>header=TRUE</code>	Does the data contain column names?
<code>sep=','</code>	How the elements are separated in the file?
<code>dec='.'</code>	The expected decimal character
<code>stringsAsFactors=TRUE</code>	Should character vectors be converted to factors?



Importing and exporting data

To export(save) a dataset from your R work space to a local file, one can use the function `write.table()`

x	The object to be written to file
file	The file name to be open for writing
quote()=TRUE	All characters and factors will be put inside double quotes
sep=','	How the elements will be separated in the file?
dec='.'	The expected decimal character
row.names=TRUE	Whether the row names of x are to be written along with x

Lets's try the following:

```
Auto<- read.table("http://www-bcf.usc.edu/~gareth/ISL/Auto.data",
                  header = TRUE, stringsAsFactors = FALSE)
list.files()
write.table(Auto, 'Auto.csv', quote=FALSE, sep=',')
Auto2=read.table('Auto.csv', header=TRUE, sep=',',
                 quote = "", dec='.', stringsAsFactors = FALSE)
```



Importing and exporting data

Reading a CSV file is a more common task because data are always stored in the format called 'comma separated values' (csv). That is, each line contains a row of values which can be numbers or letters, and each value is separated by a comma. The command to read the data file is

`read.csv()`

For example:

```
Auto=read.csv('http://www-bcf.usc.edu/~gareth/ISL/Auto.csv')
head(Auto, 2)
```

```
##      mpg cylinders displacement horsepower weight acceleration year origin
## 1   18          8           307         130   3504          12.0    70      1
## 2   15          8           350         165   3693          11.5    70      1
##
##                               name
## 1 chevrolet chevelle malibu
## 2          buick skylark 320
```

The alike function `write.csv()` writes your dataset into a specified local file.

Exercises

- Download the 'Auto.data' from the website and save it somewhere in your local space with the name 'Auto.data'
- Read the file into R using `read.table()`
- Delete the column specifying the of the car
- Figure out which year in 1970 and 1971 produces cars with a higher average mpg
- Figure out which year in 1977 and 1978 produces cars with more 8-cylinder engine
- Save the 'Auto' data into a csv file, call it 'Auto2.csv'

