Introduction to **R** for **Data Mining**

Sahid Hotel, 19 Januari 2019

Outline

- Introduction to R
- Basic Calculation
- Data and Variable
- Read and Write Data
- Conditional Statement
- Looping
- Function

Introduction to R

R and R Studio

R is a language and environment for statistical computing and graphics. Available at https://cran.r-project.org/

RStudio allows the user to run R in a more user friendly environment. It is open source (i.e. free) and available at http://www.rstudio.com/

Introduction to R

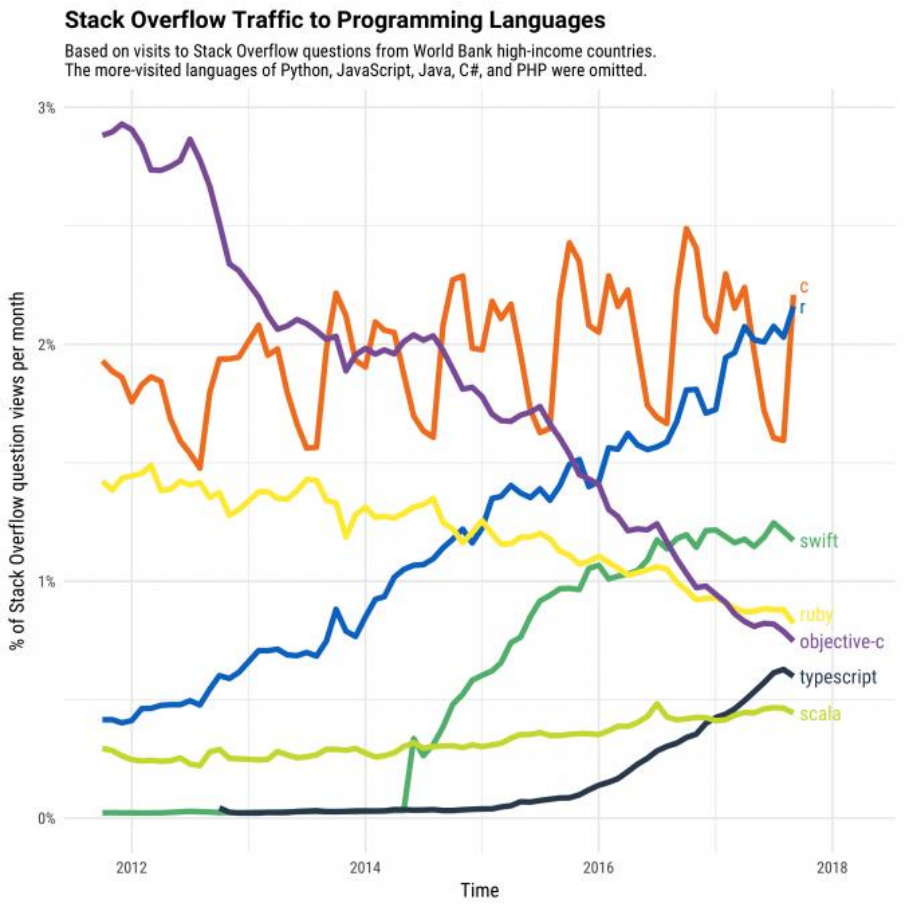
Why use R?

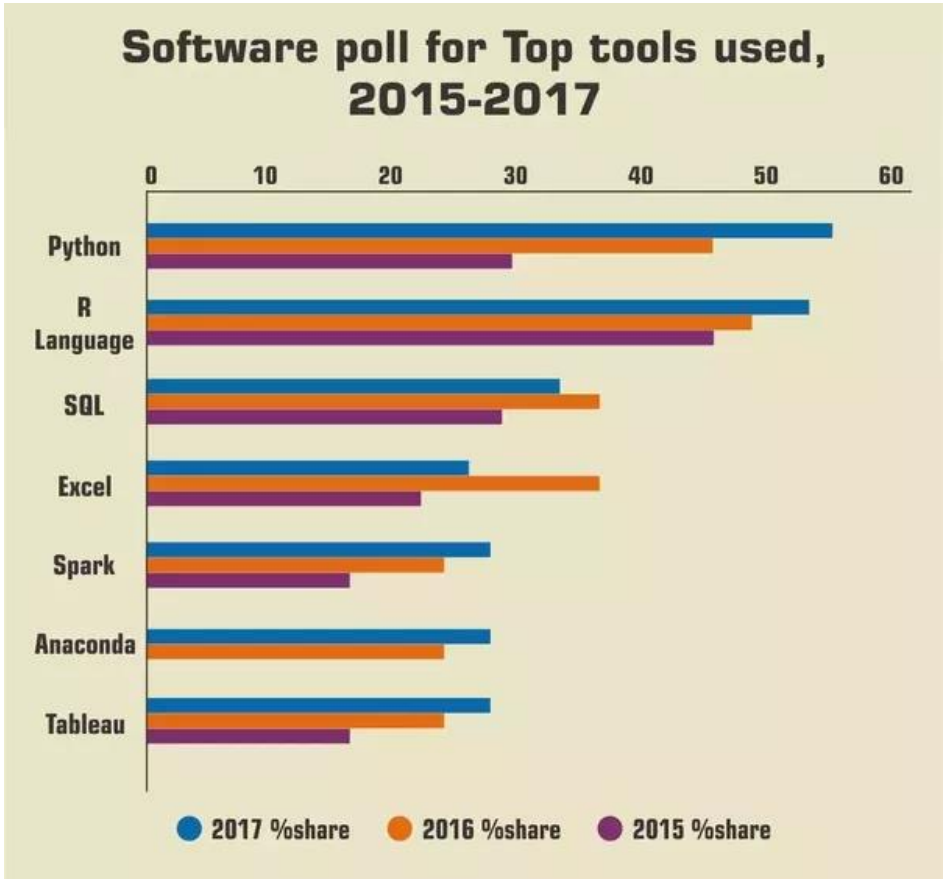
- **Data analysis software:** R is s data analysis software. It is used by data scientists for statistical analysis, predictive modeling and visualization.
- **Statistical analysis environment:** R provides a complete environment for statistical analysis. It is easy to implement statistical methods in R. Most of the new research in statistical analysis and modeling is done using R. So, the new techniques are first available only in R.
- **Open source:** R is open source technology, so it is very easy to integrate with other applications.
- **Community support:** R has the community support of leading statisticians, data scientists from different parts of the world and is growing rapidly.

Source : https://www.eduonix.com/blog/bigdata-and-hadoop/why-r-is-important-for-data-science-professionals/

Introduction to R

Why use R?





Introduction to R

Companies Using R Programming Language

R programming language is an integral part of Twitter's Data Science toolbox and is basically used to monitor user experience on Twitter.

The New York Times uses R programming language for interactive data analysis. The New York Times also uses R programming language to improve its traditional reporting.

The New York Times

Facebook processes more than 500TB of data and it uses R language for exploratory data analysis to understand how users interact with the service.

Basic Calculation

Aritmathic Operation

```
5+6+3
[1] 14
5+6-3
[1] 8
(7+15)/2
[1] 11
2^3
[1] 8
2^(2*3)
[1] 64
5 %/% 2  #integer division
[1] 2
5 %% 2  #modulo division
[1] 1
```

Assignment Variable

```
a <- 2
b = 2
2 -c
d = e = f = 3
```

- names are case sensitive.
- pi is a constant, but still can be used as variable name.
- print(x) prints content of x

Data dan Variable

Vector

```
a = 1:3 # 1 2 3

b = 2:4 # 2 3 4

c(a,b)
[1] 1 2 3 2 3 4

c(1 ,1:3)
[1] 1 1 2 3

array(1 ,4)
[1] 1 1 1 1
```

```
seq(1 ,3)
[1] 1 2 3
seq(1,3, by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0
seq(1,3, length.out = 4)
[1] 1.000000 1.666667 2.333333 3.000000
rep(1:4 ,2)
[1] 1 2 3 4 1 2 3 4
rep(1:4, each = 2)
[1] 1 1 2 2 3 3 4 4
rep(c(7 ,9 ,3), 1:3)
[1] 7 9 9 3 3 3
```

Data dan Variable

Matrix

```
matrix (1:12 , nrow =3)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
matrix (1:12 , nrow =3, byrow = TRUE)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

```
matrix (2, nrow =2, ncol =2)
     [,1] [,2]
[1,]    2    2
[2,]    2    2
matrix (1:12 , 3 ,4)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Data dan Variable

Dataframe

```
Age <- c(10 ,20 ,15 ,43 ,76 ,41 ,25 ,46)
Sex <- factor (c("m","f","m","f","m","f","m","f"))
Sibblings <- c(2 ,5 ,8 ,3 ,6 ,1 ,5 ,6)
myframe <- data.frame(Age, Sex, Sibblings)
myframe
```

```
  Age Sex Sibblings
1  10   m         2
2  20   f         5
3  15   m         8
4  43   f         3
5  76   m         6
6  41   f         1
7  25   m         5
8  46   f         6
```

Conditional Statement

```
#simple if
x <- 1
if (x==2){ print ("x=2") }

# if - else
x <- 1
if (x==2) {print ("x = 2")} else {print ("x != 2")}
```

Logical Function

```
<    #smaller
<=   #smaller or equal
  #bigger
>=   #bigger or equal
!=   #unequal
```

```
==   #logical equal
!    #logical NOT ( unary )
&    #logical AND ( vector )
|    #logical OR ( vector )
&&   #logical AND (no vector )
||   #logical OR (no vector )
```

Looping

for

```
for (i in 1:4) {print(i)}
for (i in letters[1:4]) {print(i)}
```

while

```
i <- 0
while (i<4) {
  i <- i+1
  print(i)
}
```

repeat

```
i <- 0
repeat {
  i <- i+1
  print (i)
  if (i==4) break
}
```

Function

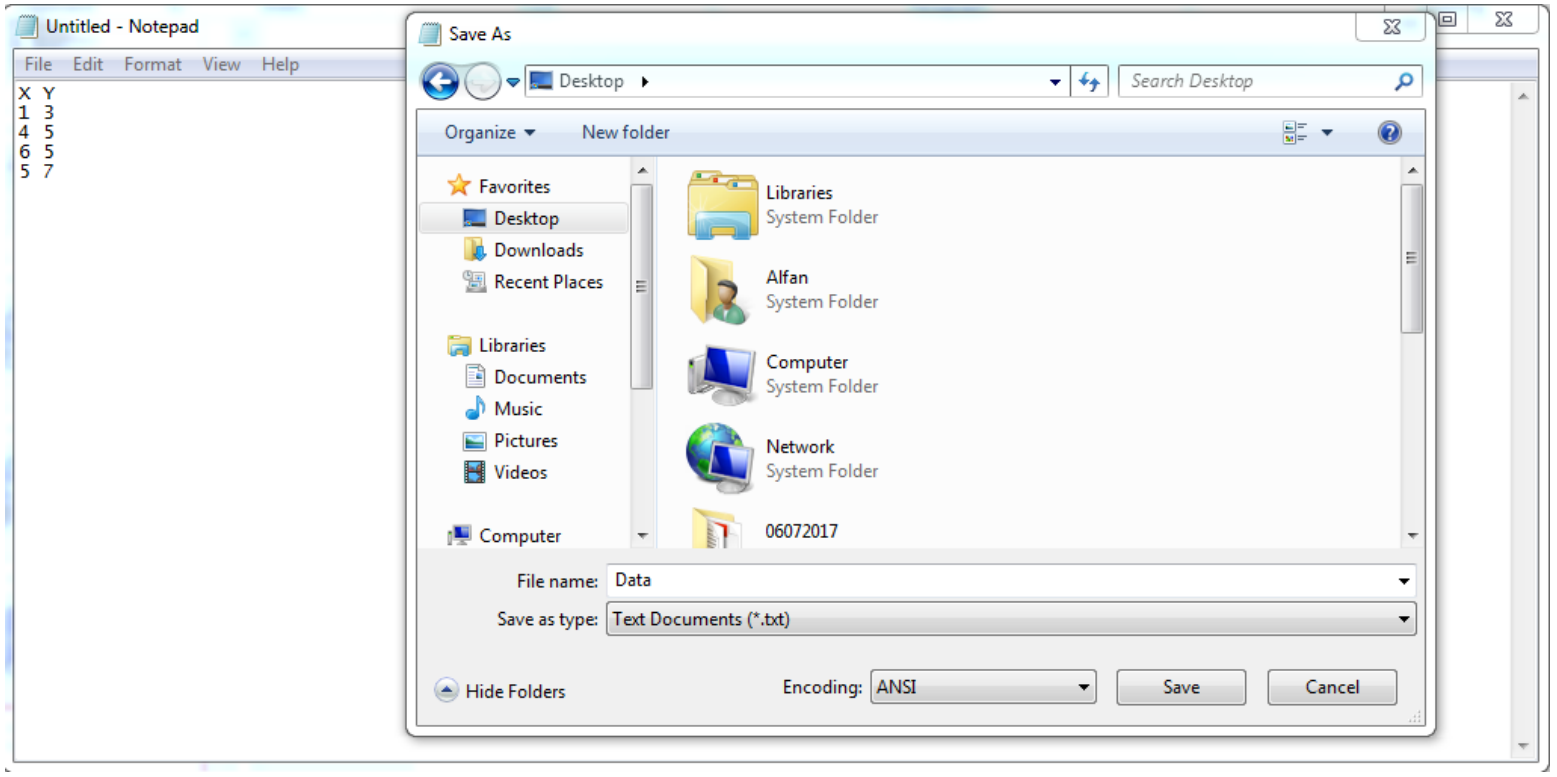
simple

```
myfun <- function(x){
  a=x^2/pi
  return(a)
  }
myfun(2)
```

Multiple input and return

```
myfun5 <- function (x, a){
  r1 <- a* sin (x)
  r2 <- a* cos (x)
  return ( list (r1 ,r2))
}
myfun5 (2,4)
```

Read and Write Data



For example we create data in notepad

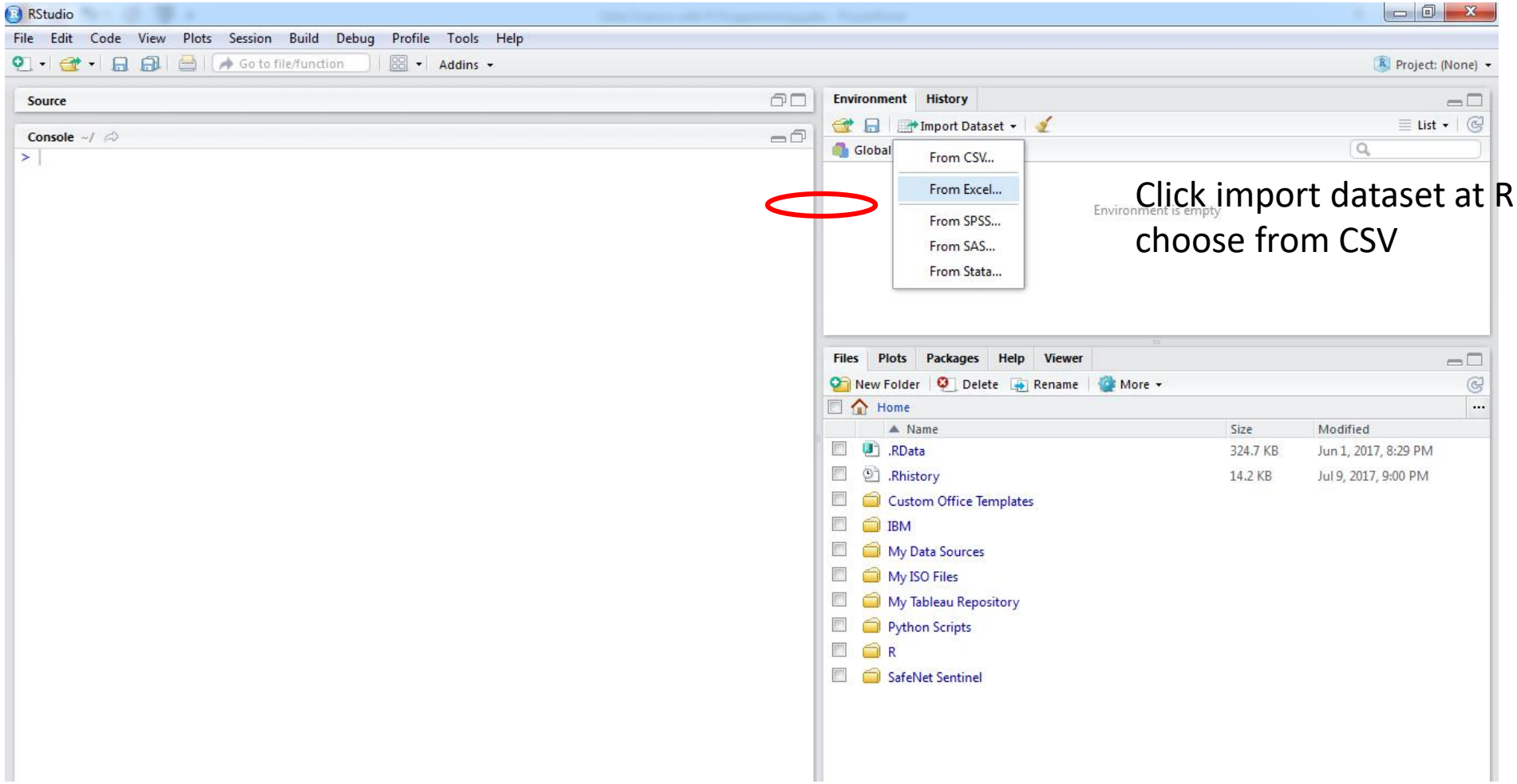
Read and Write Data

```
#Function read.table
read.table(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".“, ...)
```

`File`	the name of the file which the data are to be read from
`header`	a logical value indicating whether the file contains the names of the variables as its first line
`sep`	the field separator string. Values within each row of `x` are separated by this string.
`quote`	the set of quoting characters
`dec`	the string to use for decimal points in numeric or complex columns: must be a single character.

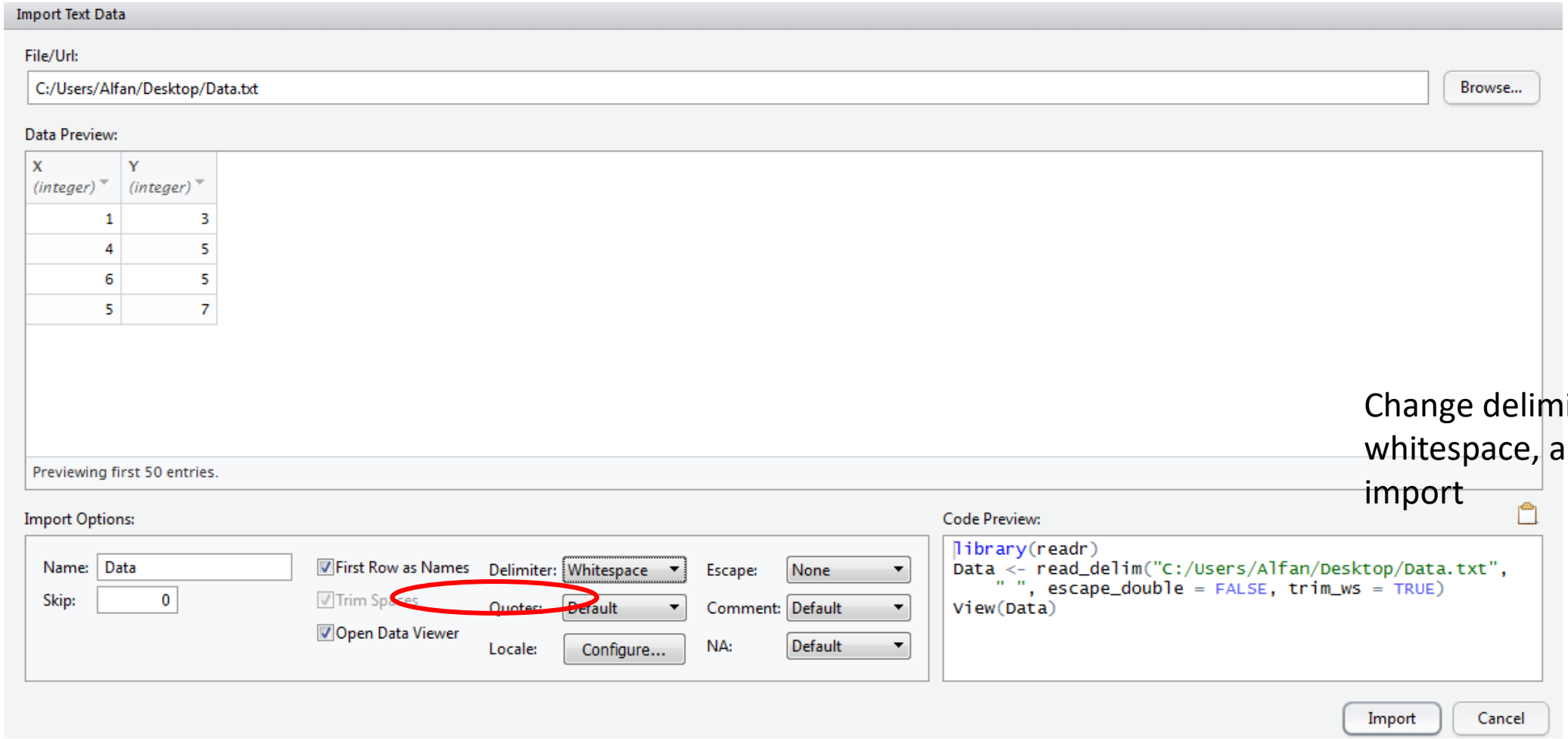
```
read.table("E:/Data.txt",header = T)
```

Read and Write Data



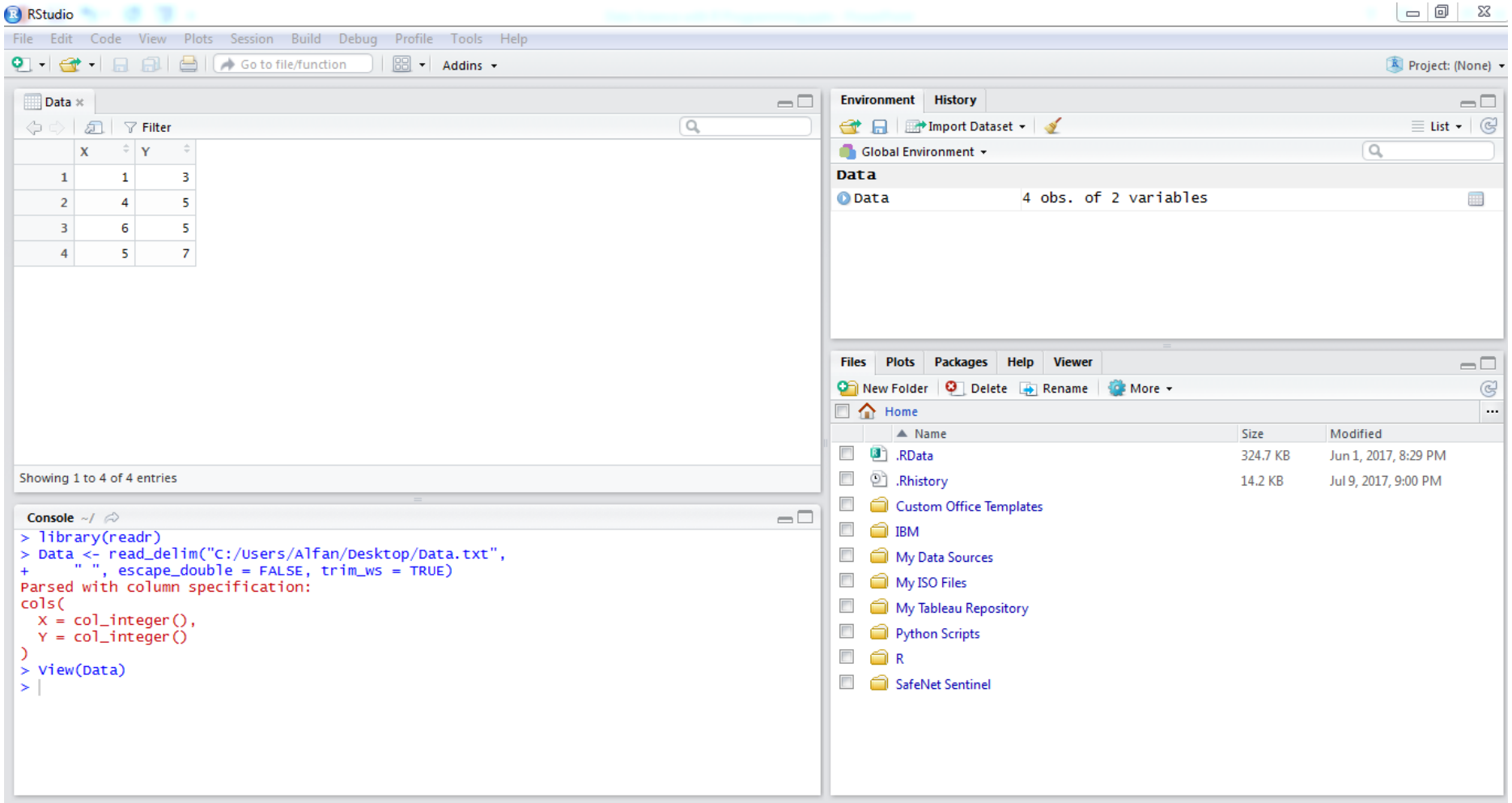
Click import dataset at R Studio, choose from CSV

Read and Write Data



Change delimiter with whitespace, and click import

Read and Write Data



Read and Write Data

```
#Function write.table
write.table(x, file = "", , quote = TRUE, sep = " ", na = "NA", dec = ".",
row.names = TRUE, col.names = TRUE)
```

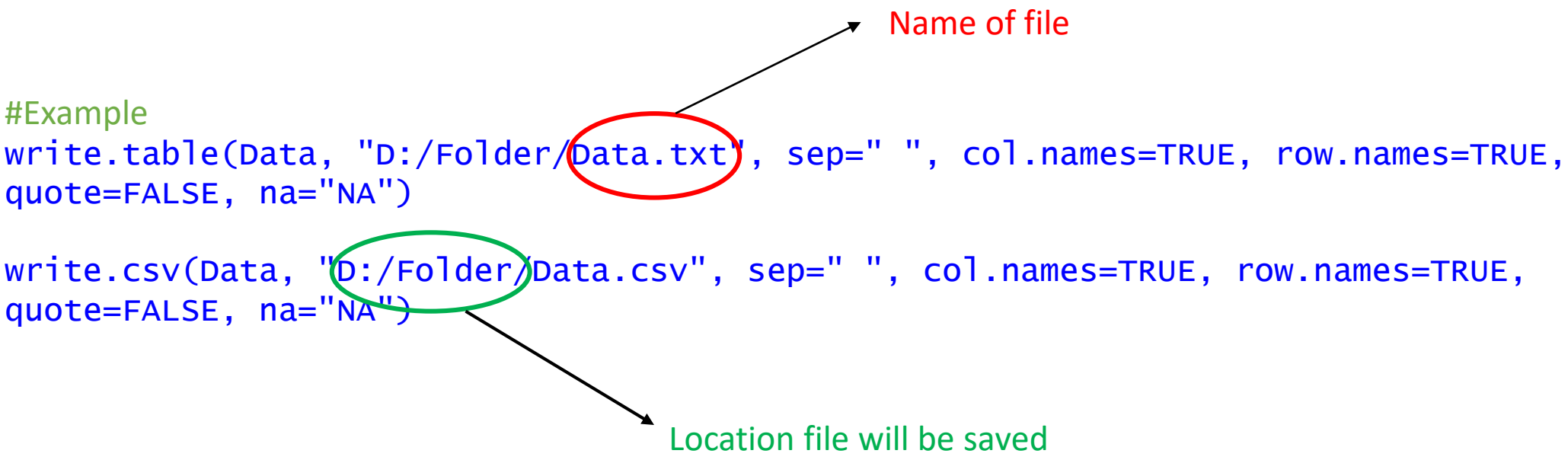
x	the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce x to a data frame.
file	either a character string naming a file or a connection open for writing. ""indicates output to the console.
quote	a logical value (TRUE or FALSE) or a numeric vector. If TRUE, any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. In both cases, row and column names are quoted if they are written. If FALSE, nothing is quoted.
sep	the field separator string. Values within each row of x are separated by this string.
na	the string to use for missing values in the data.
dec	the string to use for decimal points in numeric or complex columns: must be a single character.
row.names	either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
col.names	either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written. See the section on 'CSV files' for the meaning of col.names = NA.

Read and Write Data

```
#Function write.csv
write.csv(x, file = "", , quote = TRUE, sep = " ", na = "NA", dec = ".",
row.names = TRUE, col.names = TRUE)
```

`x`	the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce `x` to a data frame.
`file`	either a character string naming a file or a connection open for writing. `""`indicates output to the console.
`quote`	a logical value (`TRUE` or `FALSE`) or a numeric vector. If `TRUE`, any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. In both cases, row and column names are quoted if they are written. If `FALSE`, nothing is quoted.
`sep`	the field separator string. Values within each row of `x` are separated by this string.
`na`	the string to use for missing values in the data.
`dec`	the string to use for decimal points in numeric or complex columns: must be a single character.
`row.names`	either a logical value indicating whether the row names of `x` are to be written along with `x`, or a character vector of row names to be written.
`col.names`	either a logical value indicating whether the column names of `x` are to be written along with `x`, or a character vector of column names to be written. See the section on 'CSV files' for the meaning of `col.names = NA`.

Read and Write Data



Manipulating Dataframe

Thank You