

Big Data Analysis Using Machine Learning for Social Scientists and Criminologists

Big Data Analysis Using Machine Learning for Social Scientists and Criminologists

By

Juyoung Song and Tae Min Song

Cambridge
Scholars
Publishing



Big Data Analysis Using Machine Learning for Social Scientists
and Criminologists

By Juyoung Song and Tae Min Song

This book first published 2019

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

Copyright © 2019 by Juyoung Song and Tae Min Song

All rights for this book reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

ISBN (10): 1-5275-3388-3

ISBN (13): 978-1-5275-3388-2

This work was supported by the Ministry of Education of the
Republic of Korea and the National Research Foundation of Korea
(NRF-2016S1A5A2A03925702)

TABLE OF CONTENTS

Installation and Use of R	1
Installation of R	1
Use of R	7
Scientific Research Design	35
Research Concepts	36
Variable Measurement	37
Unit of Analysis	39
Sampling and Hypothesis Testing	39
Statistical Analysis	44
Overview of Machine Learning	118
Introduction	118
Machine Learning Training Data	122
Development of a Cyber bullying Prediction Model Based on Machine Learning	124
Naïve Bayes Classification Model	124
Logistic Regression Model	130
Random Forest Model	134
Decision Tree Model	141
Neural Network Model	149
Support Vector Machine Model	162
Association Analysis	170
Cluster Analysis and Segmentation	179
Machine Learning Model Evaluation	186
Machine Learning Model Evaluation Using Misclassification Tables	189
Machine Learning Model Evaluation Using ROC Curves	208
Artificial Intelligence	215
Calculate the Effect of Input Variables on Output Variables (Prediction Probability)	215

Using Training Data with Input Variables to Create Dependent Variables.....	221
Creating Data with the Same Training-Data and Predicted-Data Classifications.....	225
Evaluating Existing Training Data and High Quality Training Data	228
Creating an Artificial Intelligence with Machine Learning.....	230
Visualization.....	236
Visualization of Text Data	236
Visualization of Time Series Data	239
Visualization of Geographical Data	250
Developing Machine Learning–Based Predictive Models of Adverse Drug Responses	258
Introduction.....	258
Research Subjects and Analysis Method	263
Result	269
Discussion and Conclusion	302
Index.....	307

INSTALLATION AND USE OF R

The R program (simply “R” hereafter) is an open-source (i.e., the source code is made public so that anyone can use, modify, or redistribute the code for free) program developed for statistical analysis and visualization. It is an object-oriented language based on objects that can take scripts used in one analysis and reuse them in another analysis. R is an open-source language derived from the S language developed at Bell Laboratories in 1976. In 1995, the source was made public by Robert Gentleman and Ross Ihaka at the University of Auckland in New Zealand. Since then, it has been continuously improved by the R core development team. It is executed in interactive mode so the execution results can be seen quickly. R is an object-oriented language that can reuse the instructions (i.e., scripts) used in analysis for other analyses. R is useful for developing functions and packages, which are collections of scripts that perform specific functions, and is widely used among statisticians for statistical software development and data analysis. Today, packages and functions developed by several experts are publicly available on CRAN (Comprehensive R Archive Network), and the program’s usefulness is continuously increasing.

Installation of R

Anyone can install and use R by downloading the program from the R project homepage (<http://www.r-project.org>). To use R for graphing or visualization, the Java program for modern Windows operating systems (32 bit or 64 bit) must be installed. The processes for installing R and Java follow.

- ① Download R-3.5.0-win.exe from the R project homepage and run the program.



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

[About R](#)

[R Homepage](#)

[The R Journal](#)

[Software](#)

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

[Documentation](#)

[Manuals](#)

[FAQs](#)

[Contributed](#)

R-3.5.0 for Windows (32/64 bit)

[Download R 3.5.0 for Windows \(62 megabytes, 32/64 bit\)](#)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe1 windows: both [graphical](#) and [command line](#) versions are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is
[<CRAN.MIRROR>/bin/windows/base/release.htm](#).

- ② Set the installation language to English and click [OK].

Select Setup Language



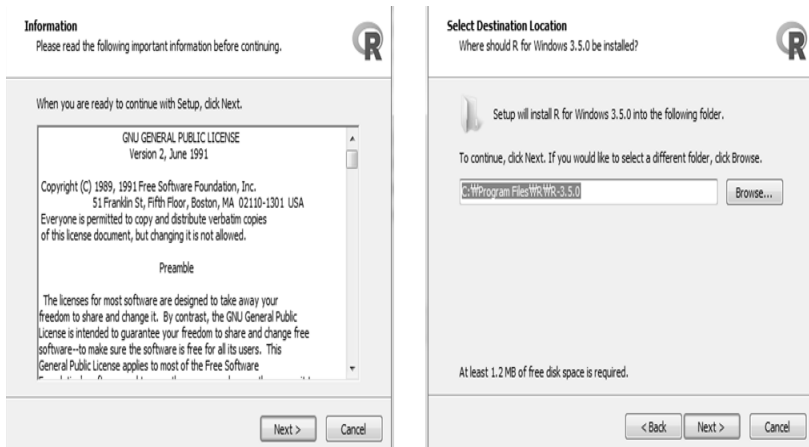
Select the language to use during the installation:

English

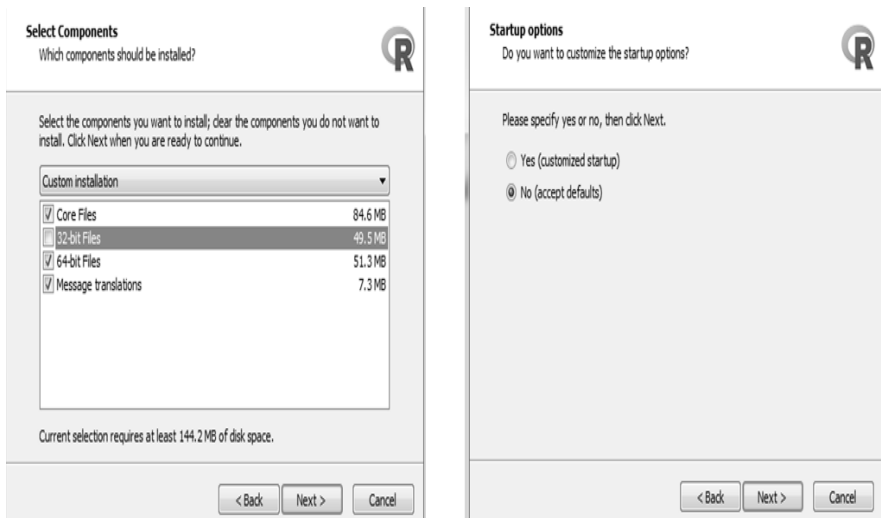
OK

Cancel

- ③ Click [Next] to begin the installation. As information about the installation appears, continue to click [Next].
- ④ Select the location where the R program will be installed. If using the default folder, click [Next].



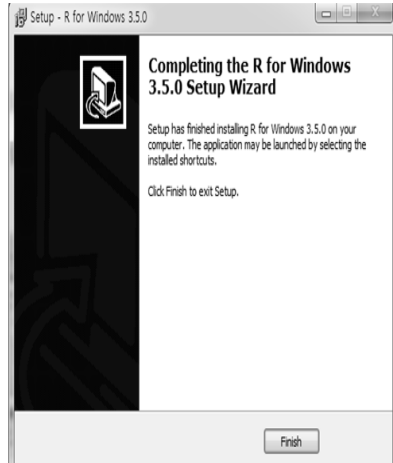
- ⑤ Install the components that are appropriate for operating on the PC where the program will be installed, and click [Next].
- ⑥ In the startup options, select “No (accept defaults)” and click [Next].



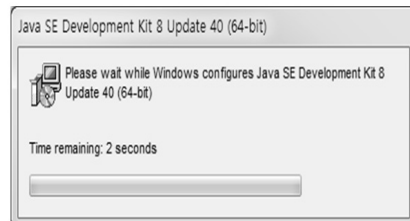
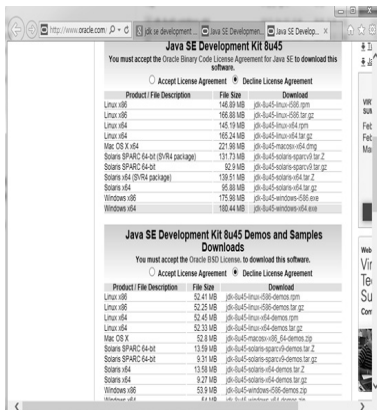
- ⑦ Select the R program's start menu folder and click [Next].
- ⑧ Select the additional installation items (use defaults) and click [Next].



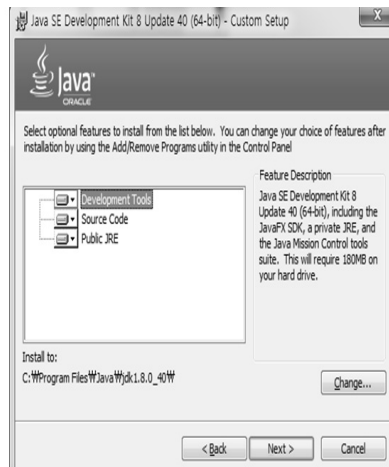
- ⑨ The installation-progress screen will appear. When the “installation complete” screen appears, click [Finish].



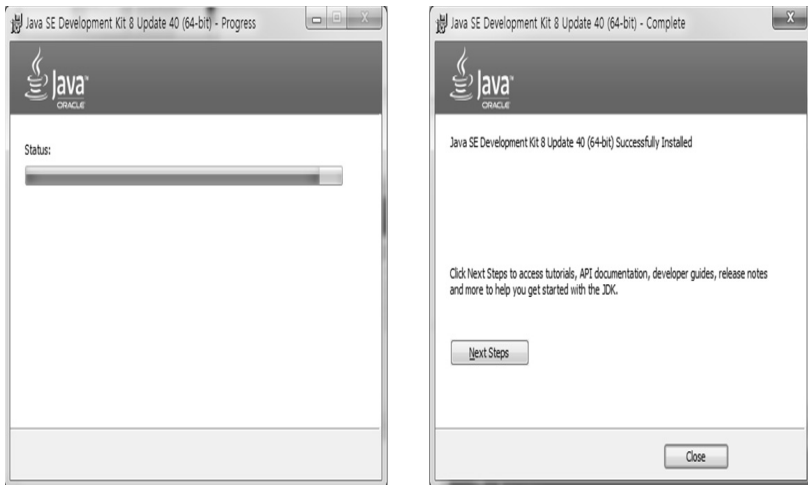
- ⑩ Search for the Java program (JDK SE development) on Google. On the download page, download the JDK file suitable for your PC and run `jdk-8u40-windows-x64`.



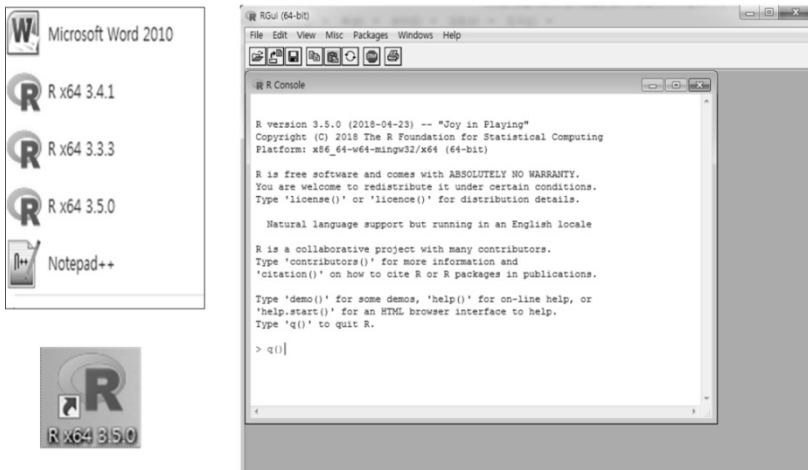
- ⑪ When the Java installation screen appears, click [Next]. Select the installation components (select the default items) and click [Next].



- ⑫ When the Java program's installation is complete, click [Close] to finish the Java installation.



- ⑬ After the installation is complete, go to the Windows start menu and click [All Programs] → [R] or click the R icon installed on the desktop. When closing the program, click the 'X' on the window or enter 'q()'.



★ Changing the R-Console to English

- ① Open a text editor, e.g., Wordpad, with administrator privileges and open the following file.

- C:\Program Files\R\R-3.5.0\etc\Rconsole

② Modify the content of the file as shown below.

- (text above)
- `## Language for messages`
- `language = en`
- (text below)

③ Run R-3.5.0 again and the English R-Console will appear.

Use of R

R is a script-command-based program. Packages needed for various analyses can be installed and used as libraries.

1) Installing and Loading Packages

R is open source so it has no distribution limitations; i.e., R can be used to create and provide new solutions, even if they will be sold commercially. R can install and load a variety of packages, depending on the analysis method (statistical analysis, machine learning, visualization, etc.). Packages can be freely downloaded from the CRAN site (www.r-project.org) and installed. R comes with several basic packages, and 12,000 additional packages are available on CRAN (12,087 packages registered as of February 5, 2018). An internet connection is necessary when first installing additional packages. Packages can be installed from the homepage's CRAN mirrors by using the `install.packages()` function or the "Install Packages" command on the menu bar in R. The mirror site has copies of the same content at multiple locations to prevent a large amount of traffic congregating at a single site. As of May 10, 2018, 161 mirror sites were operating in 48 regions, including the '0-Cloud'. The United States has 15 mirror sites that can be used.

Script Example: Create a word cloud of keywords for strain and delinquency factors in cyber bullying.

```

> setwd("c:/cyberbullying_methodology"): Set the working directory
> install.packages('wordcloud')
  - Install the package that processes word clouds.
> library(wordcloud) : Load the package that processes word clouds.
> key=c('Domestic-violence','Child-abuse','Parental-divorce','Economic-
  problems','Friend-Violence','Break-ups','School-control','Academic-
  stress','School_records','School-violence-experience','Transfer',
  'Individualism','Materialism','Bullying-culture','Class-society','Hell-
  Korea','Female_dislike','Interested_soldier','Traffic-accidents','Games',
  'Internet-addiction','Celebrities','Movie','Adults','Gags','Chat-apps',
  'Youtube','Personal-broadcasting') : Assign keywords for cyber bullying
  strain and delinquency factors to the key vector.
> freq=c(2269,1338,3515,7269,5844,1101,32816,1503,32084,5849, 8949,
  2348, 858,539, 617,1085,6452,784,1852,1764,2496,29473,24413,488,
  799, 2253, 1497,1153)
  - Assign keyword frequencies for cyber bullying strain and
  delinquency factors to the freq vector.
> library(RColorBrewer) : Load the package for displaying color.
> palette=brewer.pal(9,"Set1")
  - Assign RColorBrewer's nine text colors to a palette variable.
> wordcloud(key,freq,scale=c(4,1),rot.per=.12,min.freq=100, random.
  order=F, random.color=T,colors=palette) : Display the word cloud.
> savePlot("cyber_bullying_strain_wordcloud",type="png")
  - Save the results as an image file.

```

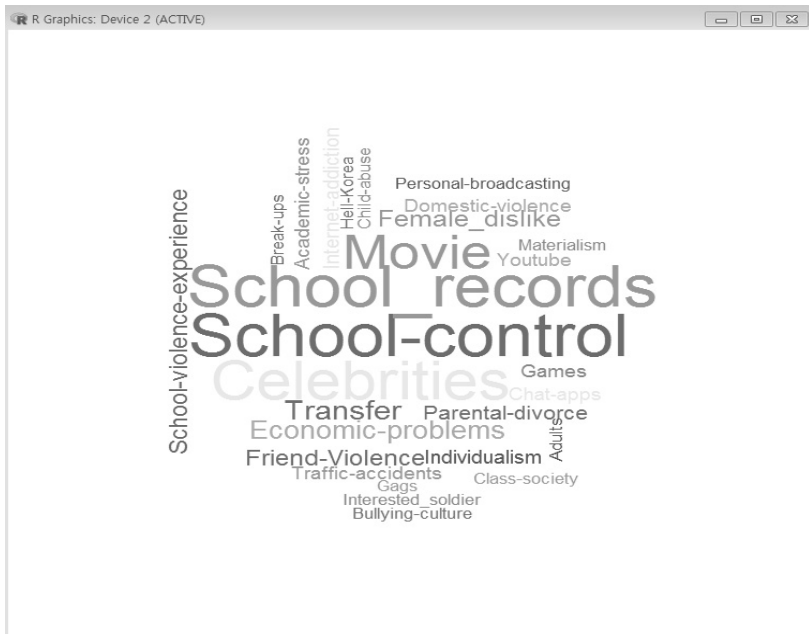


```

R Console

> ## machine learning wordcloud 2018. 5. 10.
>
> setwd("c:/cyberbullying_methodology")
> install.packages('wordcloud')
Warning: package 'wordcloud' is in use and will not be installed
> library(wordcloud)
>
> key=c('Domestic-violence', 'Child-abuse', 'Parental-divorce', 'Economic-problems',
+ 'Friend-Violence', 'Break-ups', 'School-control', 'Academic-stress', 'School_records',
+ 'School-violence-experience', 'Transfer', 'Individualism', 'Materialism', 'Bullying-culture',
+ 'Class-society', 'Hell-Korea', 'Female_dislike', 'Interested_soldier', 'Traffic-accidents',
+ 'Games', 'Internet-addiction', 'Celebrities', 'Movie', 'Adults', 'Gags', 'Chat-apps', 'Youtube',
+ 'Personal-broadcasting')
>
> freq=c(2269,1338,3515,7269,5844,1101,32816,1503,32084,5849,8949,2348,858,539,617,1085,
+ 6452,784,1852,1764,2496,29473,24413,488,799,2253,1497,1153)
> library(RColorBrewer)
> palette=brewer.pal(9,"Set1")
> wordcloud(key,freq,scale=c(4,1),rot.per=.12,min.freq=100,random.order=F,
+ random.color=T,colors=palette)
> savePlot("cyber_bullying_strain_wordcloud",type="png")
> |

```



2) Value Assignment and Calculation

- ① Run the R shortcut on the Windows desktop. In the initial screen, enter a script in the column after the prompt '>'. Press the 'ENTER' key to run it.
- ② In R, saving the execution results (values) to objects or variables is called assigning. Use "=" or "<-" to assign values in R. (This book uses '='.)
- ③ When an R script is long, use "+" to connect the next line.
- ④ Use ';' to connect several scripts.
- ⑤ R has the following rules for using variables.
 - Variable names are case sensitive.
 - Variable names can use English letters, numbers, periods (.), and underscores (_); however, the first character cannot be a number or underscore. When numbers are used as variables, "X" is automatically added to the first character.
 - The reserved words in the R system (if, else, NULL, NA, in, etc.) cannot be used as variable names.
- ⑥ Functions are collections of scripts that take an argument-type value as input and return the calculated result value. In R, functions can be used to make programs more concise
- ⑦ The following operators are available: Operators [+ , - , * , / , %% (modulus), ^ (exponent), etc.] and R's internal functions [sin(), exp(), log(), sqrt(), mean(), etc.].

■ Saving formulas using operators

```
> pie=3.1415 : Assign 3.1415 to pie.  
> x=100 : Assign 100 to x.  
> y=2*pie+x : Assign 2 × pie + x to y.  
> y : Display the value of y on the screen.
```

■ Saving formulas using internal functions

```
> x=c(75, 80, 73, 65, 75, 83, 73, 82, 75, 72) : Assign 10 vector values  
(weights) to x.  
> mean(x) : Display the mean of x on the screen.  
> sd(x) : Display the standard deviation of x on the screen.
```



```

R Console
> ## value assignment
>
> setwd("c:/cyberbullying_methodology")
>
> pie=3.1415
> x=100
> y=2*pie+x
> y
[1] 106.283
>
> ## internal function
>
> x=c(75,80,73,65,75,83,73,82,75,72)
> mean(x)
[1] 75.3
> sd(x)
[1] 5.313505
>

```

- ⑧ To repeat a previously performed task, press the up-arrow key.
- ⑨ To end the R program, click the 'X' on the window or enter 'q()'.

3) Basic Data Types in R

- ① Set the directory where all of the objects (functions, data, etc.) that are used in R will be saved

```
> setwd("c:/cyberbullying_methodology")
```

- ② The basic data types in R are as follows.

- **Numeric type:** Uses arithmetic operators $+$, $-$, $*$, $/$, $%%$ (modulus), $^$ (exponent), etc.] to calculate results. □

```
> x = sqrt(50*(100^2))
```

- **Character type:** Groups together strings of text with single quotes (') or double quotes (").

```
> v_name = 'machine learning modeling'
```

- **NA type:** Use this when a value is not determined set.

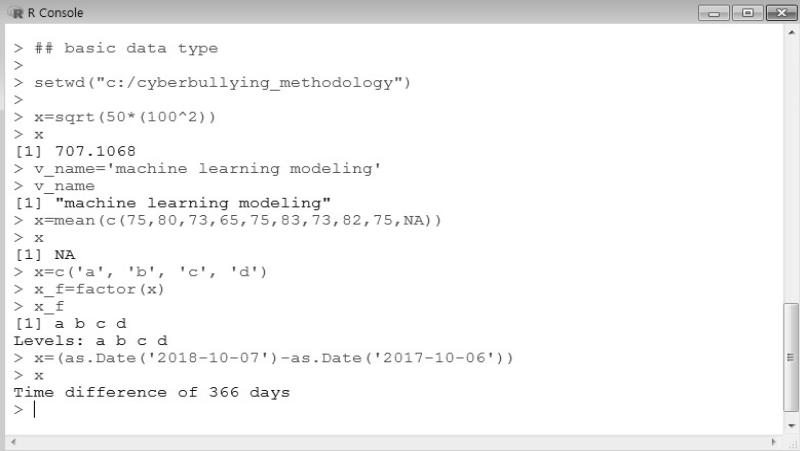
```
> x = mean(c(75, 80, 73, 65, 75, 83, 73, 82, 75, NA))
```

- **Factor type:** Use this to convert character-type data into numeric type.

```
> x = c('a', 'b', 'c', 'd'); x_f = factor(x)
```

- Date and time format: Use this when analyzing a certain time period or time.

```
> x = (as.Date('2018-10-07') - as.Date('2017-10-06'))
```



```
R Console
> ## basic data type
>
> setwd("c:/cyberbullying_methodology")
>
> x=sqrt(50*(100^2))
> x
[1] 707.1068
> v_name='machine learning modeling'
> v_name
[1] "machine learning modeling"
> x=mean(c(75,80,73,65,75,83,73,82,75,NA))
> x
[1] NA
> x=c('a', 'b', 'c', 'd')
> x_f=factor(x)
> x_f
[1] a b c d
Levels: a b c d
> x=(as.Date('2018-10-07')-as.Date('2017-10-06'))
> x
Time difference of 366 days
> |
```

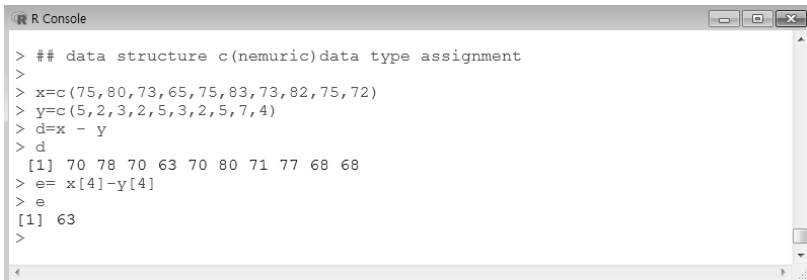
4) Data Structures in R

In R, data is managed through vector, matrix, array, and list-type data structures.

(1) Vector

A vector is the basic data structure in R. A vector is a data object that combines and stores several data items. Vector in R use the `c()` function to assign values.

- ```
> x=c(75, 80, 73, 65, 75, 83, 73, 82, 75, 72)
- Assign the weights of ten people to variable x as a vector.
> y=c(5, 2, 3, 2, 5, 3, 2, 5, 7, 4)
- Assign the weight loss of 10 people to variable y as a vector.
> d=x - y
- Subtract vector y from vector x and assign the result to vector d.
> d : Display the value of vector d on the screen.
> e= x[4] - y[4]
- Subtract the value of the 4th element in vector x (65) from the value
of the 4th element in vector y (2) and assign the result to variable e.
> e : Display the value of variable e on the screen.
```



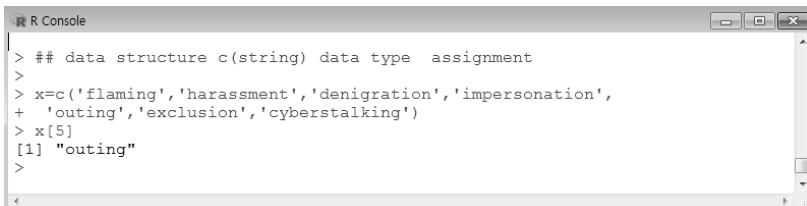
```

> ## data structure c(numeric) data type assignment
>
> x=c(75,80,73,65,75,83,73,82,75,72)
> y=c(5,2,3,2,5,3,2,5,7,4)
> d=x - y
> d
[1] 70 78 70 63 70 80 80 71 77 68 68
> e= x[4]-y[4]
> e
[1] 63
>

```

### ● Character-type data management

> x=c('flaming','harassment','denigration','impersonation','outing',  
'exclusion','cyberstalking') : Assign character data to vector x.  
> x[5] : Display the fifth element value of vector x on the screen.



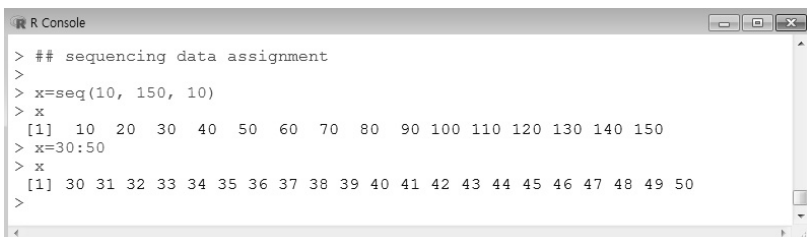
```

> ## data structure c(string) data type assignment
>
> x=c('flaming','harassment','denigration','impersonation',
+ 'outing','exclusion','cyberstalking')
> x[5]
[1] "outing"
>

```

### ● Use the seq() function or “:” to assign sequential data to a vector.

> x=seq(10, 150, 10)  
- Create a sequence of numbers from 10 to 150 with an increment of 10,  
and assign the results to vector x.  
> x=30:50  
- Create a sequence of numbers from 30 to 50 with an increment of 1  
and assign the results to vector x



```

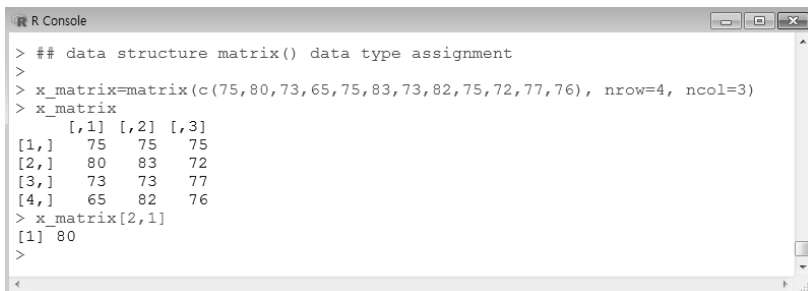
> ## sequencing data assignment
>
> x=seq(10, 150, 10)
> x
[1] 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150
> x=30:50
> x
[1] 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
>

```

## (2) Matrix

A matrix is two-dimensional vector data structures that additionally have rows and columns. The `matrix()` function is used for data management.

- ```
> x_matrix=matrix(c(75, 80, 73, 65, 75, 83, 73, 82, 75, 72, 77, 76),  
nrow=4, ncol=3)
```
- Create a matrix with 4 rows and 3 columns containing 12 people's weights, and assign the results to `x_matrix`.
- ```
> x_matrix : Display the value of x_matrix on the screen.
> x_matrix[2,1]
```
- Display the value of the element at row 2, column 1 of `x_matrix` on the screen.

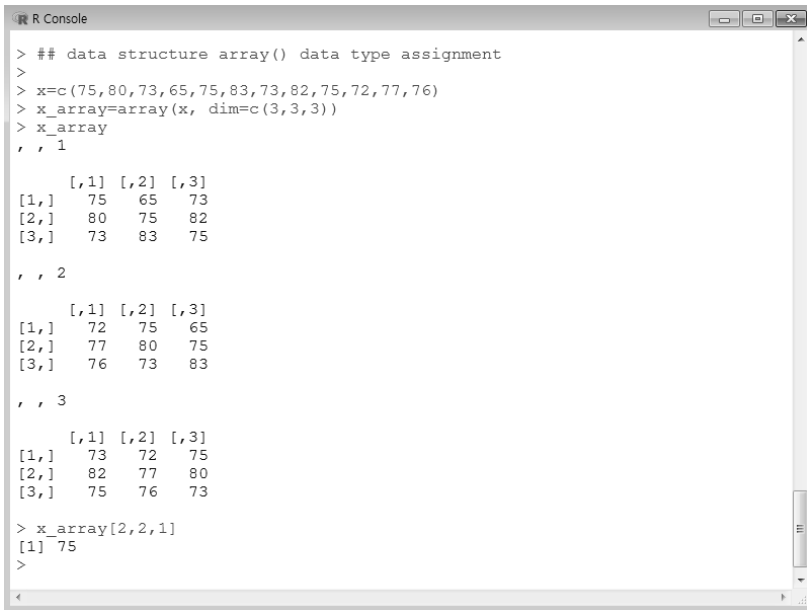


```
R Console
> ## data structure matrix() data type assignment
>
> x_matrix=matrix(c(75,80,73,65,75,83,73,82,75,72,77,76), nrow=4, ncol=3)
> x_matrix
 [,1] [,2] [,3]
[1,] 75 75 75
[2,] 80 83 72
[3,] 73 73 77
[4,] 65 82 76
> x_matrix[2,1]
[1] 80
>
```

## (3) Array

An array is three or more dimensions and can extend matrices multidimensionally. The `array()` function is used for data management.

- ```
> x=c(75, 80, 73, 65, 75, 83, 73, 82, 75, 72, 77, 76)
```
- Assign the weights of 12 people to vector `x`.
- ```
> x_array=array(x, dim=c(3, 3, 3))
```
- Assign vector `x` to the `x_array` variable as a 3D structure.
- ```
> x_array : Display the value of the array variable x_array on the screen.  
> x_array[2,2,1]
```
- Display the value of the element at [2, 2, 1] in `x_array` on the screen.



```

> ## data structure array() data type assignment
>
> x=c(75,80,73,65,75,83,73,82,75,72,77,76)
> x_array=array(x, dim=c(3,3,3))
> x_array
, , 1
      [,1] [,2] [,3]
[1,]   75   65   73
[2,]   80   75   82
[3,]   73   83   75

, , 2
      [,1] [,2] [,3]
[1,]   72   75   65
[2,]   77   80   75
[3,]   76   73   83

, , 3
      [,1] [,2] [,3]
[1,]   73   72   75
[2,]   82   77   80
[3,]   75   76   73

> x_array[2,2,1]
[1] 75
>

```

(4) List

A list is a type of matrix or array that can specify the data type in the format of (address, value).

```

> x_address=list(name='Pennsylvania State University Schuylkill,
Criminal Justice',address='200 University Drive, Schuylkill Haven, PA
17972', homepage='http://www.sl.psu.edu/')
  - Assign the address to the list format variable x_address.
> x_address : Display the value of the x_address variable on the screen.
> x_address=list(name="Sahmyook university, department of health
management",address='815, Hwarang-ro, Nowon-gu, Seoul, 01795,
KOREA', homepage='https://www.syu.ac.kr/')
> x_address

```

```

R Console
> ## data structure list() data type assignment
>
> x_address=list(name='Pennsylvania State University Schuylkill, Criminal Justice',
+ address='200 University Drive, Schuylkill Haven, PA 17972',
+ homepage='http://www.sl.psu.edu/')
> x_address
$name
[1] "Pennsylvania State University Schuylkill, Criminal Justice"
$address
[1] "200 University Drive, Schuylkill Haven, PA 17972"
$homepage
[1] "http://www.sl.psu.edu/"
> x_address=list(name='Sahmyook university, department of health management',
+ address='815, Hwarang-ro, Nowon-gu, Seoul, 01795, KOREA',
+ homepage='https://www.syu.ac.kr/')
> x_address
$name
[1] "Sahmyook university, department of health management"
$address
[1] "815, Hwarang-ro, Nowon-gu, Seoul, 01795, KOREA"
$homepage
[1] "https://www.syu.ac.kr/"
> |

```

5) Using Functions in R

Users can use R's built-in functions or they can use function() to create their own functions. User-defined functions should use the following basic format.

```

Function name = function(argument1, argument2, ...) {
  Calculation formula or executable program
  return(calculation results or return value) }

```

- Exercise 1: Find the sample size using the confidence level and sampling error
 - Formula: $n = (\pm Z)^2 \times P(1 - P) / (SE)^2$
 - Create a function (SZ) to find the sample size when a phone survey is conducted to analyze the state of school bullying with a sample error of 3% at a confidence level of 95% ($Z = 1.96$) at $p = .5$.

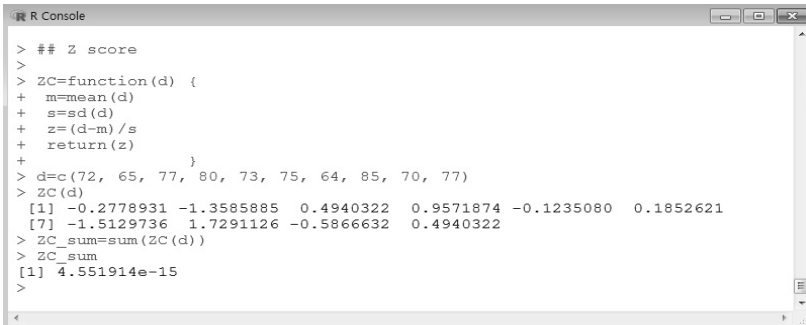
```

R Console
> ## Function usage
> ## sample size
>
> SZ=function(p, z, s) {
+   n=z^2*p*(1-p)/s^2
+   return(n)
+ }
> SZ(0.5, 1.96, 0.03)
[1] 1067.111
>

```

● Exercise 2: Find the standard score

- The standard score is a measure of the extent to which an observed value differs from the mean. It can be used to find the data's relative position. The sum of the observed values' standard scores is 0.
- Formula: $Z_i = (X_i - \bar{x})/s_X$
- Create a function (ZC) to find the standard score after measuring the weights of 10 people.



```

> ## Z score
>
> ZC=function(d) {
+   m=mean(d)
+   s=sd(d)
+   z=(d-m)/s
+   return(z)
+ }
> d=c(72, 65, 77, 80, 73, 75, 64, 85, 70, 77)
> ZC(d)
[1] -0.2778931 -1.3585885  0.4940322  0.9571874 -0.1235080  0.1852621
[7] -1.5129736  1.7291126 -0.5866632  0.4940322
> ZC_sum=sum(ZC(d))
> ZC_sum
[1] 4.551914e-15

```

● Exercise 3: Find the population variance

- ```

> setwd("setwd("c:/cyberbullying_methodology"))

```
- Set the working directory.

```

> cyber_bullying=read.table(file="cyber_bullying_descriptive_analysis.txt",header=T)

```

  - : Import the text data and assign it to cyber\_bullying.

```

> attach(cyber_bullying)

```

  - : Attach cyber\_bullying as execution data.

```

> VAR=function(x) var(x)*(length(x)-1)/length(x)

```

  - Create a new function with the format of “function (argument or input value) formula”.
  - length(x): Calculate the sample size of the x variable that is passed to the VAR function as an argument.
  - Create the function (VAR) to find the population variance for the x argument.

```

> VAR(Onespread)

```

  - : Call the VAR function and calculate the population variance for Onespread.

```

> sqrt(VAR(Onespread))

```

  - : Call the VAR function and calculate the population standard deviation for Onespread.

```

> VAR(Twospread)

```

  - : Call the VAR function and calculate the population variance for Twospread.

```

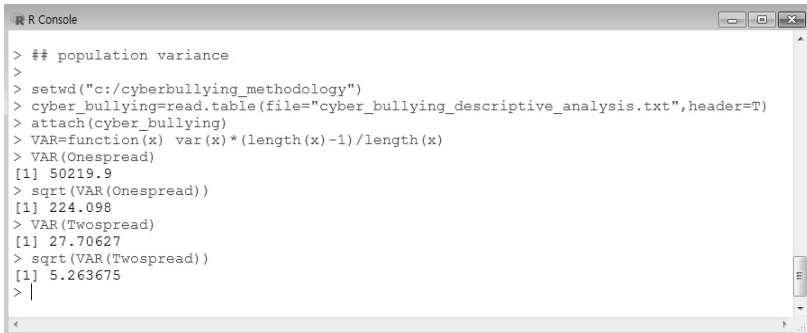
> sqrt(VAR(Twospread))

```

  - : Call the VAR function and calculate the



population standard deviation for Twospread.



```
> ## population variance
>
> setwd("c:/cyberbullying_methodology")
> cyber_bullying=read.table(file="cyber_bullying_descriptive_analysis.txt",header=T)
> attach(cyber_bullying)
> VAR=function(x) var(x) * (length(x)-1) / length(x)
> VAR(Onespread)
[1] 50219.9
> sqrt(VAR(Onespread))
[1] 224.098
> VAR(Twospread)
[1] 27.70627
> sqrt(VAR(Twospread))
[1] 5.263675
> |
```

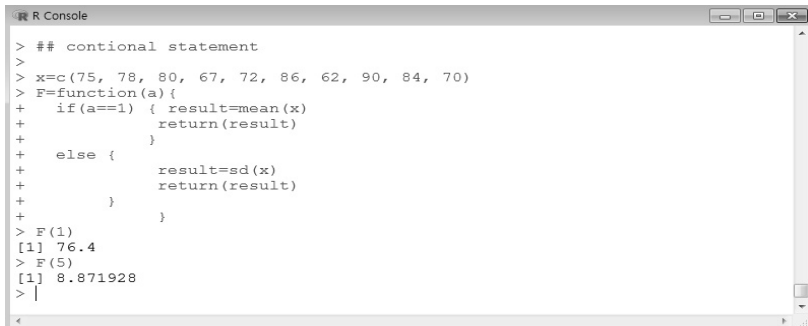
## 6) Basic Program in R (Conditional Statements and Loop Statements)

- R provides conditional statements, which determine the execution flow, as well as loop statements, which repeat the same statement multiple times.
- Conditional statements use comparative operators [equal (==), not equal (!=), greater than or equal (>=), greater than (>), less than or equal (<=), and less than (<)].
- Conditional statements are formatted as follows.

```
if(condition formula) {
<Calculation to be performed if condition is true>
}
else {
<Calculation to be performed if condition is false>
}
```

### ● Exercise 4: Using conditional statements

- Create a function (F) that returns the mean of vector x, which holds the weights of 10 people if argument is '1', and which returns the standard deviation if argument is not '1'.



```

> ## contional statement
>
> x=c(75, 78, 80, 67, 72, 86, 62, 90, 84, 70)
> F=function(a){
+ if(a==1){ result=mean(x)
+ return(result)
+ }
+ else {
+ result=sd(x)
+ return(result)
+ }
+ }
> F(1)
[1] 76.4
> F(5)
[1] 8.871928
> |

```

- The format for loop statements is as follows.
- The “number of times” used in a for loop is either the “vector data” or “n: number of times.”

|                                                                             |
|-----------------------------------------------------------------------------|
| <pre> for(loop variable in number of times) {   Executed statement } </pre> |
|-----------------------------------------------------------------------------|

● Exercise 5: Using loop statements

- Create a function (F) that finds the sum of numbers from 1 to a given number.



```

> ## iteration structure
>
> F=function(a){
+ result=0
+ for(i in 1:a){
+ result=result+i
+ }
+ return(result)
+ }
> F(100)
[1] 5050
> F(50000)
[1] 1250025000
> F(2018)
[1] 2037171
>

```

## 7) Using Variables in R Data Frames

Variables can be used for statistical analysis in R as follows.

### (1) Using “data\$variables”

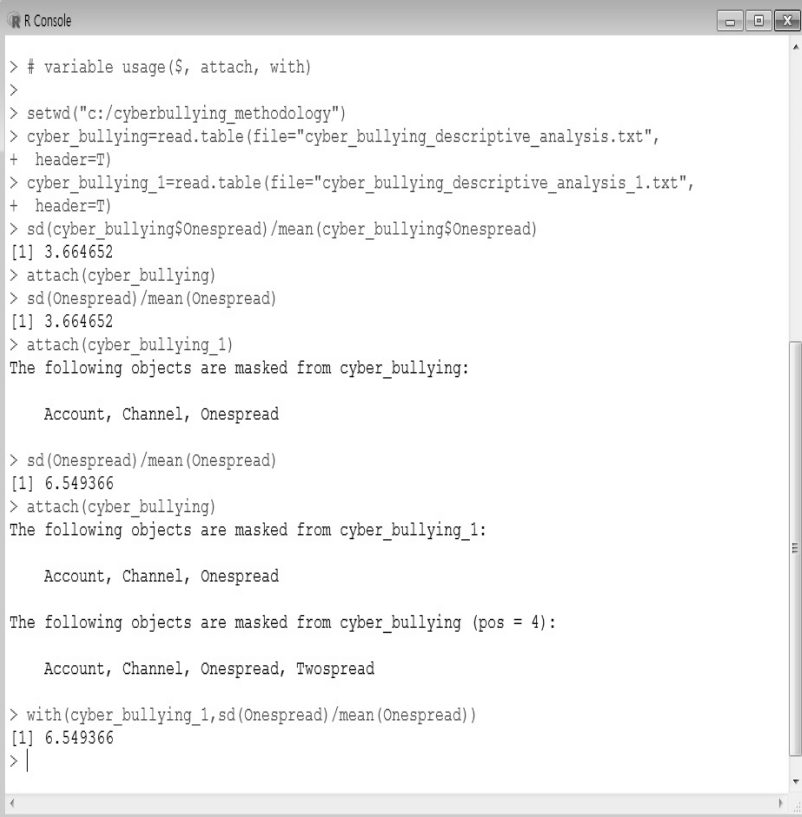
```
> setwd("c:/cyberbullying_methodology")
> cyber_bullying=read.table(file="cyber_bullying_descriptive_analysis.
txt",header=T)
- Assign 'cyber_bullying_descriptive_analysis.txt' to cyber_bullying.
> cyber_bullying_1=read.table(file="cyber_bullying_descriptive_analysis
_1.txt", header=T)
> sd(cyber_bullying$Onespread)/mean(cyber_bullying$Onespread)
- Use the Onespread variable of the cyber_bullying data frame to find
the variation coefficient.
```

### (2) Using the attach(data) function

```
> attach(cyber_bullying)
- The attach function attaches execution data as a “data” argument.
> sd(Onespread)/mean(Onespread)
- Unlike “data$variable”, after attach is executed, the variable alone can
be used to find the variation coefficient.
```

### (3) Using the with(data, script) function

```
> with(cyber_bullying_1,sd(Onespread)/mean(Onespread))
- A script can be executed using the data frame variable via the with()
function without using the attach function .
```



```

> # variable usage($, attach, with)
>
> setwd("c:/cyberbullying_methodology")
> cyber_bullying=read.table(file="cyber_bullying_descriptive_analysis.txt",
+ header=T)
> cyber_bullying_1=read.table(file="cyber_bullying_descriptive_analysis_1.txt",
+ header=T)
> sd(cyber_bullying$Onespread)/mean(cyber_bullying$Onespread)
[1] 3.664652
> attach(cyber_bullying)
> sd(Onespread)/mean(Onespread)
[1] 3.664652
> attach(cyber_bullying_1)
The following objects are masked from cyber_bullying:

 Account, Channel, Onespread

> sd(Onespread)/mean(Onespread)
[1] 6.549366
> attach(cyber_bullying)
The following objects are masked from cyber_bullying_1:

 Account, Channel, Onespread

The following objects are masked from cyber_bullying (pos = 4):

 Account, Channel, Onespread, Twospread

> with(cyber_bullying_1,sd(Onespread)/mean(Onespread))
[1] 6.549366
> |

```

## 8) Creating R Data Frames

R can create data frames with a variety of formats. The most often used data frame is the two-dimensional matrix, which has rows and columns. Data frames are also called data sets. Their columns are called variables, and their rows are called records.

### (1) Creating data frames from vectors

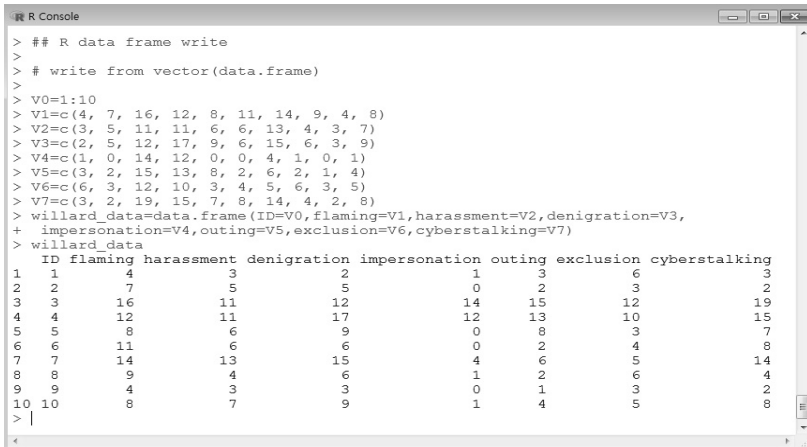
- Use the `data.frame()` function.

> `V0=1:10` : Assign numbers 1–10 to the V0 vector.

> `V1=c(4, 7, 16, 12, 8, 11, 14, 9, 4, 8)`

- Assign 10 numbers to the V1 vector, and the next six vectors.

- ```
> V2=c(3, 5, 11, 11, 6, 6, 13, 4, 3, 7)
> V3=c(2, 5, 12, 17, 9, 6, 15, 6, 3, 9)
> V4=c(1, 0, 14, 12, 0, 0, 4, 1, 0, 1)
> V5=c(3, 2, 15, 13, 8, 2, 6, 2, 1, 4)
> V6=c(6, 3, 12, 10, 3, 4, 5, 6, 3, 5)
> V7=c(3, 2, 19, 15, 7, 8, 14, 4, 2, 8)
> willard_data=data.frame(ID=V0,flaming=V1,harassment=V2,
  denigration=V3,impersonation=V4,outing=V5,exclusion=V6,
  cyberstalking=V7)
  - Assign the 8 vectors (V0–V7) to the willard_data data frame object.
> willard_data
  - Display the value of the willard_data data frame on the screen.
```



```
## R Console

> ## R data frame write
>
> # write from vector(data.frame)
>
> V0=1:10
> V1=c(4, 7, 16, 12, 8, 11, 14, 9, 4, 8)
> V2=c(3, 5, 11, 11, 6, 6, 13, 4, 3, 7)
> V3=c(2, 5, 12, 17, 9, 6, 15, 6, 3, 9)
> V4=c(1, 0, 14, 12, 0, 0, 4, 1, 0, 1)
> V5=c(3, 2, 15, 13, 8, 2, 6, 2, 1, 4)
> V6=c(6, 3, 12, 10, 3, 4, 5, 6, 3, 5)
> V7=c(3, 2, 19, 15, 7, 8, 14, 4, 2, 8)
> willard_data=data.frame(ID=V0,flaming=V1,harassment=V2,
+   impersonation=V4,outing=V5,exclusion=V6,cyberstalking=V7)
> willard_data
  ID flaming harassment denigration impersonation outing exclusion cyberstalking
1  1      4           3           2           1           3           6           3
2  2      7           5           5           0           2           3           2
3  3     16          11          12          14          15          12          19
4  4     12          11          17          12          13          10          15
5  5      8           6           9           0           8           3           7
6  6     11           6           6           0           2           4           8
7  7     14          13          15           4           6           5          14
8  8      9           4           6           1           2           6           4
9  9      4           3           3           0           1           3           2
10 10      8           7           9           1           4           5           8
> |
```

(2) Creating data frames from text files

- Use the read.table() function.
- ```
> setwd("c:/cyberbullying_methodology") : Set the working directory.
> willard_data=read.table(file="willard_data.txt",header=T)
 - Assign the "willard_data.txt" file to the willard_data object via the
 data frame.
> willard_data : Display the value of the willard_data object on the screen.
```

```

> # write from text(read.table)
>
> setwd("c:/cyberbullying_methodology")
> willard_data=read.table(file="willard_data.txt",header=T)
> willard_data
 ID flaming harassment denigration impersonation outing exclusion cyberstalking
1 1 1 4 3 2 1 3 6 3
2 2 2 7 5 5 0 2 3 2
3 3 3 16 11 12 14 15 12 19
4 4 4 12 11 17 12 13 10 15
5 5 5 8 6 9 0 8 3 7
6 6 6 11 6 6 0 2 4 8
7 7 7 14 13 15 4 6 5 14
8 8 8 9 4 6 1 2 6 4
9 9 9 4 3 3 0 1 3 2
10 10 10 8 7 9 1 4 5 8

```

### (3) Creating data frames from CSV files

- Use the read.table() function.

> setwd("c:/cyberbullying\_methodology") : Set the working directory.  
 > willard\_data=read.table(file="willard\_data.csv",header=T)  
 - Assign "willard\_data.csv" to the willard\_data object via the data frame.  
 > willard\_data : Display the value of the willard\_data object on the screen.

```

> # write from CSV file(read.table)
>
> setwd("c:/cyberbullying_methodology")
> willard_data=read.table(file="willard_data.csv",header=T)
> willard_data
 ID flaming harassment denigration impersonation outing exclusion cyberstalking
1 1 1 4 3 2 1 3 6 3
2 2 2 7 5 5 0 2 3 2
3 3 3 16 11 12 14 15 12 19
4 4 4 12 11 17 12 13 10 15
5 5 5 8 6 9 0 8 3 7
6 6 6 11 6 6 0 2 4 8
7 7 7 14 13 15 4 6 5 14
8 8 8 9 4 6 1 2 6 4
9 9 9 4 3 3 0 1 3 2
10 10 10 8 7 9 1 4 5 8

```

### (4) Creating data frames from SPSS files

- Use the read.spss() function.

> install.packages("foreign")  
 - Install a package for reading external data created by statistics software other than R; e.g., SPSS or SAS.