Lecture 2: Introduction to R (Part I)

Pilsung Kang

School of Industrial Management Engineering

Korea University

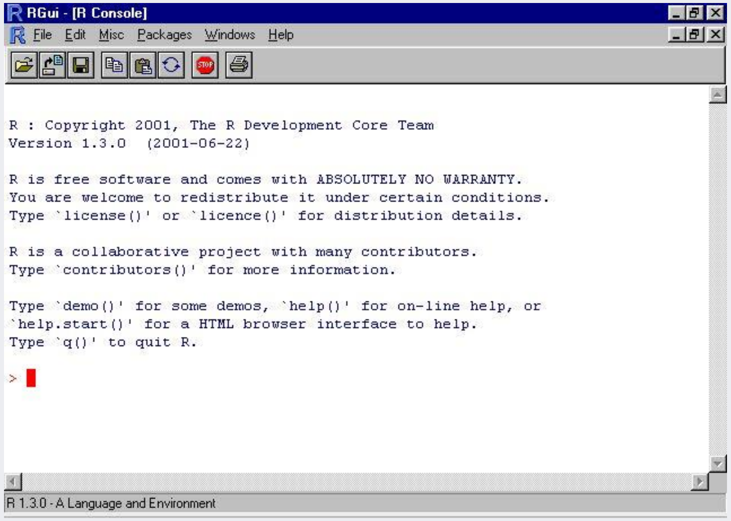
AGENDA

01 R Overview

02 Handling Different Data Types

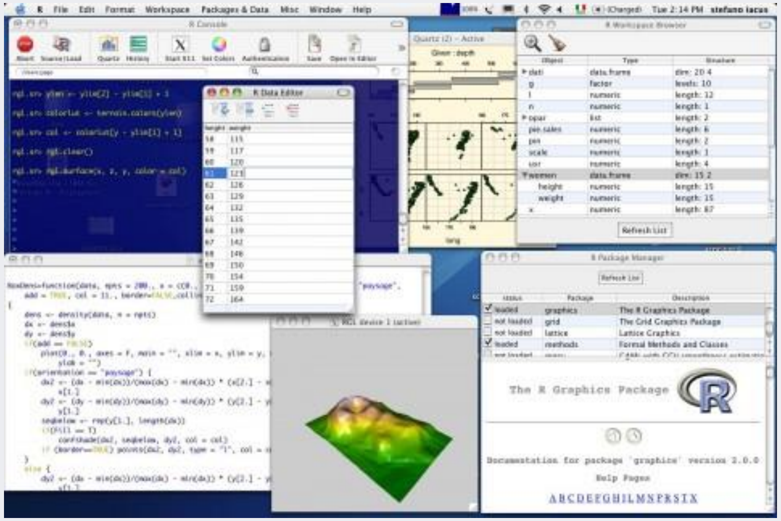
03 Text Processing

History of R



```
R : Copyright 2001, The R Development Core Team
Version 1.3.0  (2001-06-22)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type `license()' or `licence()' for distribution details.

R is a collaborative project with many contributors.
Type `contributors()' for more information.

Type `demo()' for some demos, `help()' for on-line help, or
`help.start()' for a HTML browser interface to help.
Type `q()' to quit R.

>
```

R 1.3.0 - A Language and Environment


R File Edit Format Workspace Packages & Data Misc Window Help
Tue 2:14 PM stefano iacus
R Console
R Data Editor
R Workspace Browser
Refresh List
R Package Manager
The R Graphics Package
Documentation for package 'graphics' version 2.0.0
Help Pages
ABCDEFGHILMNPRSTX

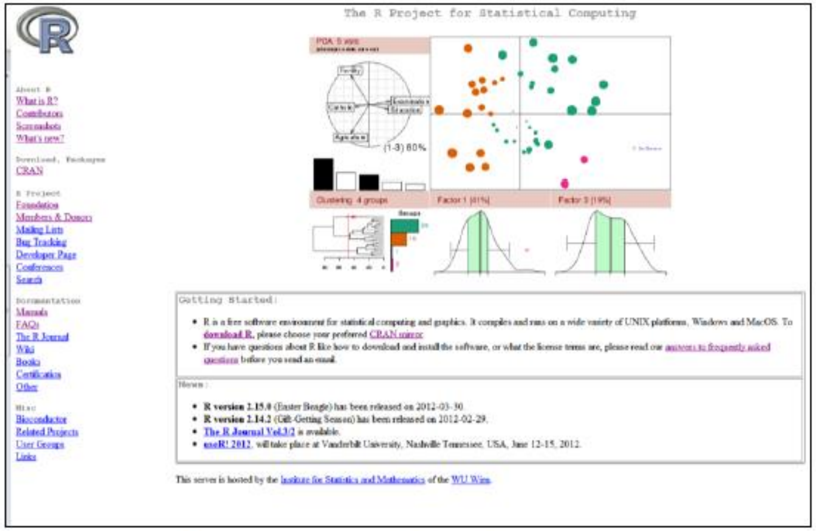
- R is an open source programming language and software environment for statistical computing and graphics
  - ✓ Created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team
  - ✓ Highly extensible through the use of user-submitted packages for specific functions or specific areas of study

The R Language

- R is an expression-based language
  - ✓ User type language expressions at the R prompt
  - ✓ These expressions are evaluated by the R interpreter
  - ✓ The computed values of the expressions are printed
- R is extensile
  - ✓ Users can implement new functionality in the form of functions
  - ✓ Developers can implement new packages of functionality that extends the base system

Why R?

- Provides a wide variety of statistical techniques
- Free software under GNU public license, highly extensible
- Independent to OS: Windows, Linux, MacOS X, etc.



The R Project for Statistical Computing

About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc
Bioconductor
Related Projects
User Groups
Links

Getting Started:

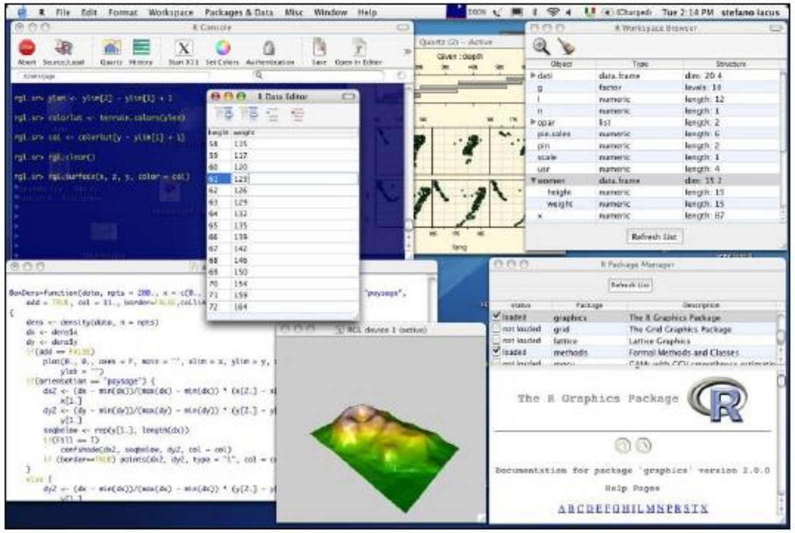
- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News:

- **R version 2.15.0** (Easter Beagle) has been released on 2012-03-30.
- **R version 2.14.2** (Gift-Getting Season) has been released on 2012-02-29.
- The R Journal Vol.3/2 is available.
- useR! 2012, will take place at Vanderbilt University, Nashville Tennessee, USA, June 12-15, 2012.

This server is hosted by the Institute for Statistics and Mathematics of the WU Wien.

〈 Homepage of R-project 〉



```
rgl.sr> ylen <- ylim[2] - ylim[1] + 1

rgl.sr> colorlut <- terrain.colors(ylen)

rgl.sr> col <- colorlut[y - ylim[1] + 1]

rgl.sr> rgl.clear()

rgl.sr> rgl.surface(x, z, y, color = col)
```

height	weight
58	115
59	117
60	120
61	123
62	126
63	129
64	132
65	135
66	139
67	142
68	146
69	150
70	154
71	159
72	164

```
BoxDens=function(data, npts = 200., x = c(0.,
    add = TRUE, col = 11., border=FALSE,col
{
    dens <- density(data, n = npts)
    dx <- dens$x
    dy <- dens$y
    if(add == FALSE)
        plot(0., 0., axes = F, main = "", xlim = x, ylim = y,
            ylab = "")
    if(orientation == "paysage") {
        dx2 <- (dx - min(dx))/(max(dx) - min(dx)) * (x[2.] -
            x[1.]
        dy2 <- (dy - min(dy))/(max(dy) - min(dy)) * (y[2.] -
            y[1.]
        seqbelow <- rep(y[1.], length(dx))
        if(fill == T)
            confshade(dx2, seqbelow, dy2, col = col)
        if (border==TRUE) points(dx2, dy2, type = "l", col =
    }
    else {
        dy2 <- (dx - min(dx))/(max(dx) - min(dx)) * (y[2.] -
            y[1.]
```

Object	Type	Structure
▶ dati	data.frame	dim: 20 4
g	factor	levels: 10
i	numeric	length: 12
n	numeric	length: 1
▶ opar	list	length: 2
pie.sales	numeric	length: 6
pin	numeric	length: 2
scale	numeric	length: 1
usr	numeric	length: 4
▼ women	data.frame	dim: 15 2
height	numeric	length: 15
weight	numeric	length: 15
x	numeric	length: 87

status	Package	Description
☑ loaded	graphics	The R Graphics Package
□ not loaded	grid	The Grid Graphics Package
□ not loaded	lattice	Lattice Graphics
☑ loaded	methods	Formal Methods and Classes

The R Graphics Package

Documentation for package 'graphics' version 2.0.0

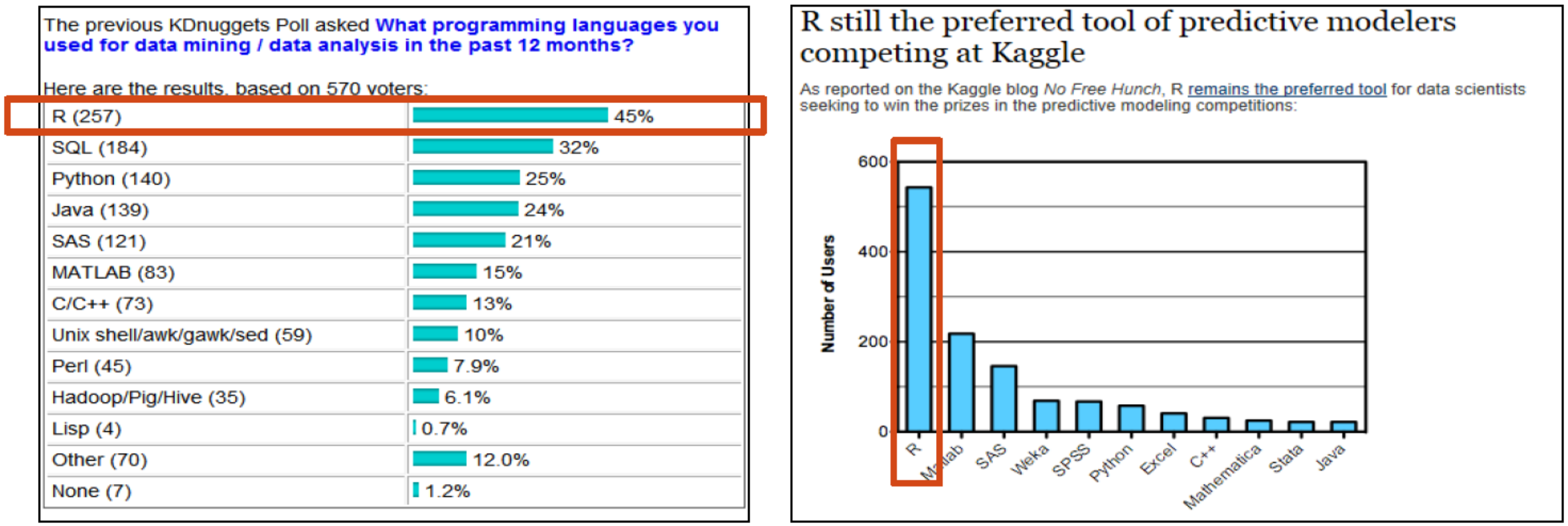
Help Pages

A B C D E F G H I L M N P R S T X

〈 R screenshot: Visualization 〉

Why R?

- Data analysis with R: new trend!
  - ✓ Many researchers now use R for their statistical analysis
  - ✓ Kdnuggets & Kaggle poll results



The previous KDnuggets Poll asked **What programming languages you used for data mining / data analysis in the past 12 months?**

Here are the results, based on 570 voters:

R (257)	45%
SQL (184)	32%
Python (140)	25%
Java (139)	24%
SAS (121)	21%
MATLAB (83)	15%
C/C++ (73)	13%
Unix shell/awk/gawk/sed (59)	10%
Perl (45)	7.9%
Hadoop/Pig/Hive (35)	6.1%
Lisp (4)	0.7%
Other (70)	12.0%
None (7)	1.2%

R still the preferred tool of predictive modelers competing at Kaggle

As reported on the Kaggle blog *No Free Hunch*, R remains the preferred tool for data scientists seeking to win the prizes in the predictive modeling competitions:

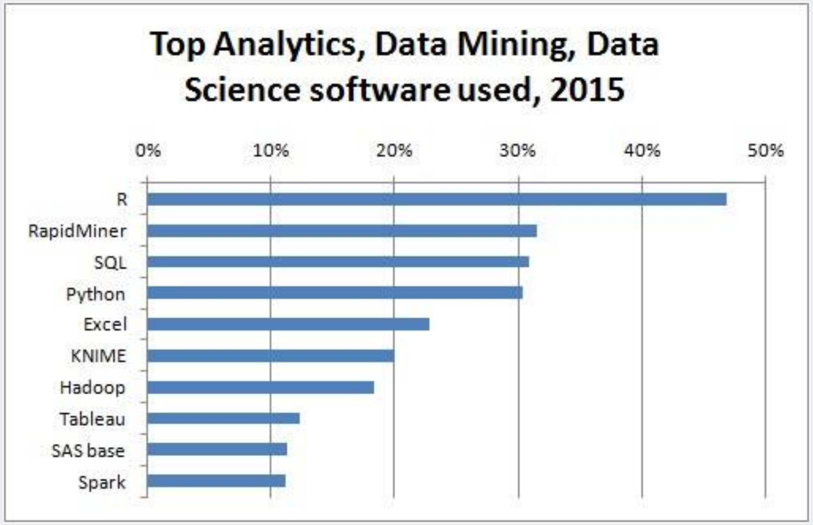
〈 R: the most widely used statistical tool 〉

- Collaboration among users: the reason why R is loved by a number of users

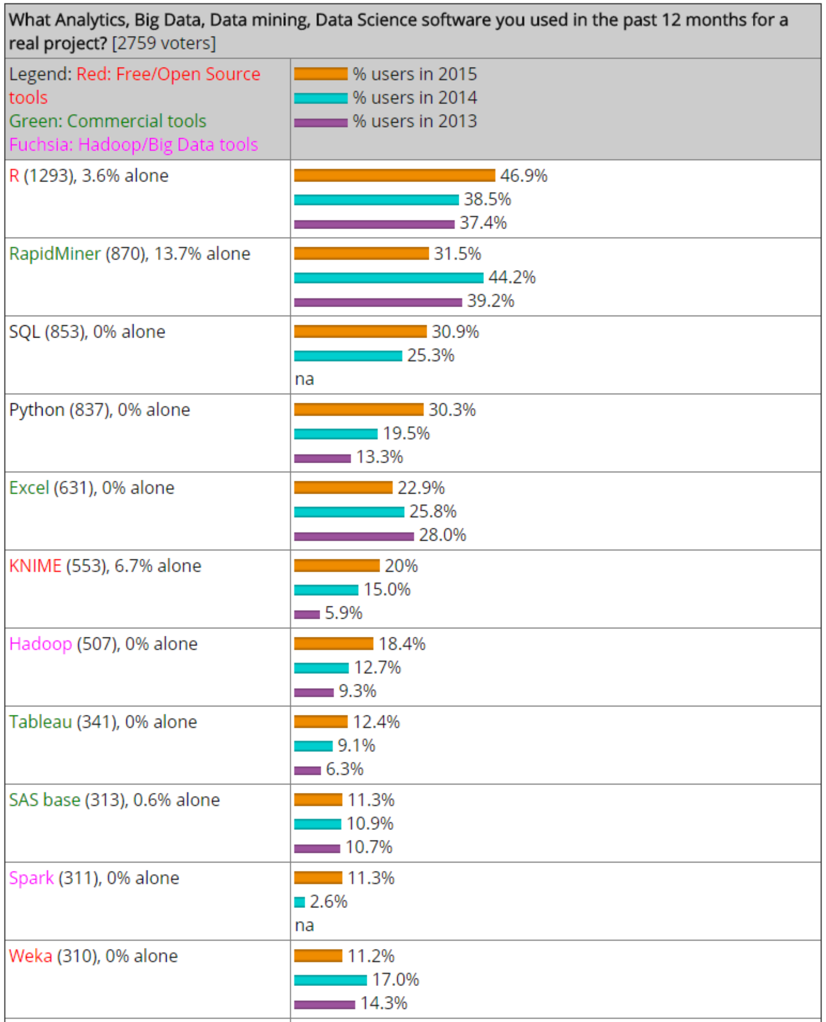
useR! The R Journal

Why R?

- Data analysis with R: new trend!
  - ✓ Many researchers now use R for their statistical analysis



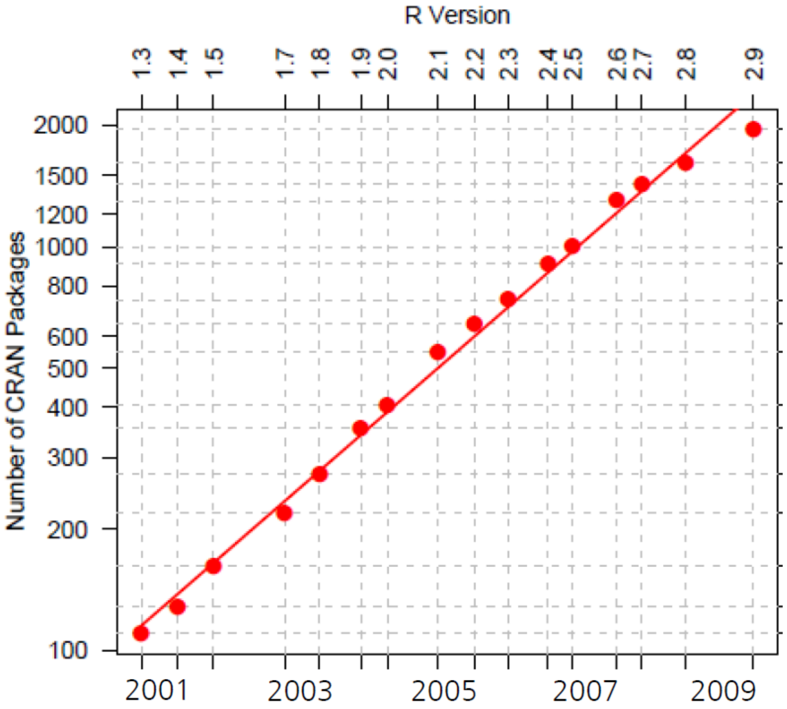
Top Analytics, Data Mining, Data Science software used, 2015



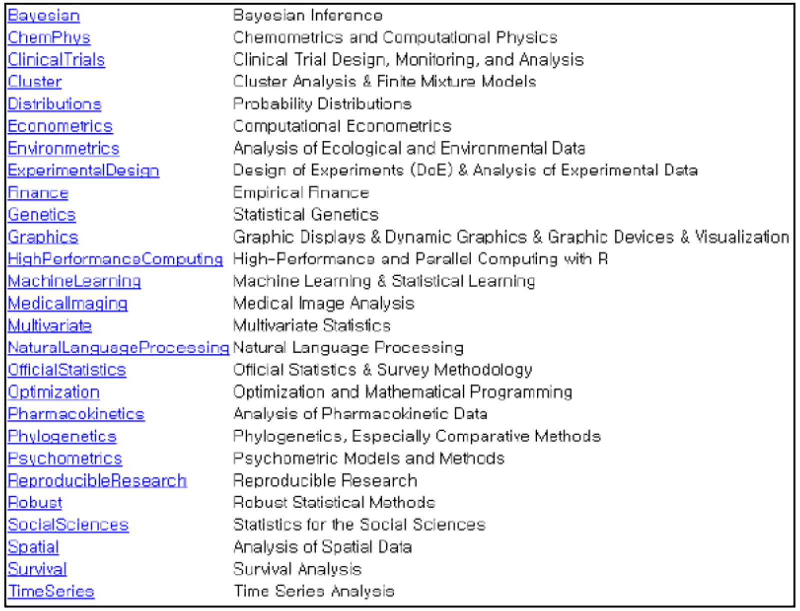
What Analytics, Big Data, Data mining, Data Science software you used in the past 12 months for a real project? [2759 voters]	
Legend: Red: Free/Open Source tools Green: Commercial tools Fuchsia: Hadoop/Big Data tools	🟧 % users in 2015 🟦 % users in 2014 🟪 % users in 2013
R (1293), 3.6% alone	46.9% 38.5% 37.4%
RapidMiner (870), 13.7% alone	31.5% 44.2% 39.2%
SQL (853), 0% alone	30.9% 25.3% na
Python (837), 0% alone	30.3% 19.5% 13.3%
Excel (631), 0% alone	22.9% 25.8% 28.0%
KNIME (553), 6.7% alone	20% 15.0% 5.9%
Hadoop (507), 0% alone	18.4% 12.7% 9.3%
Tableau (341), 0% alone	12.4% 9.1% 6.3%
SAS base (313), 0.6% alone	11.3% 10.9% 10.7%
Spark (311), 0% alone	11.3% 2.6% na
Weka (310), 0% alone	11.2% 17.0% 14.3%

Why R?

- R packages
  - ✓ Everyone can make, distribute, and use R packages
  - ✓ Almost every statistical analysis can be possible
  - ✓ 10,136 packages are available (2017. 02. 22)



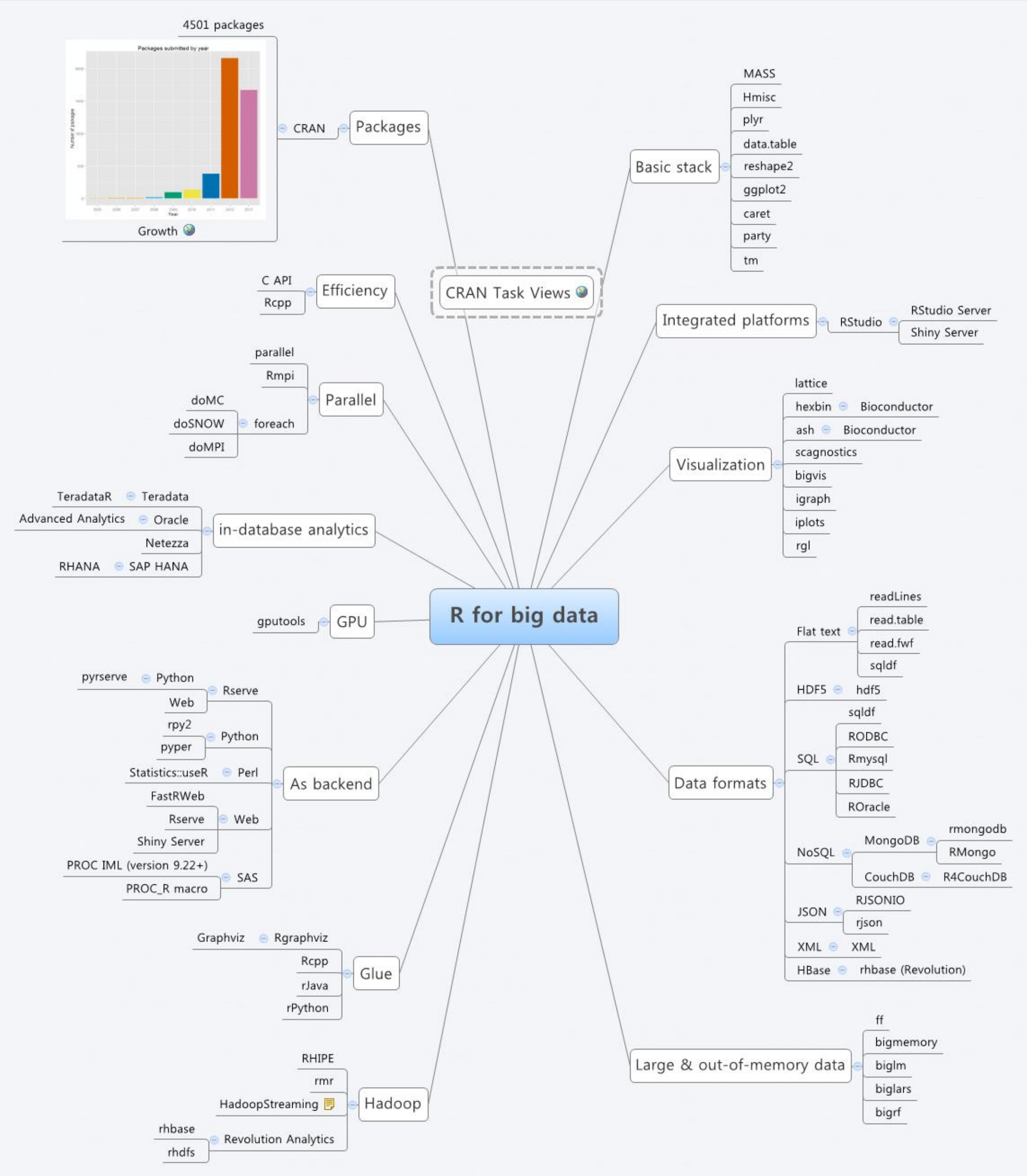
〈 The number of packages and R version on CRAN 〉



Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis

〈 Examle: packages available on CRAN 〉

R for Big Data

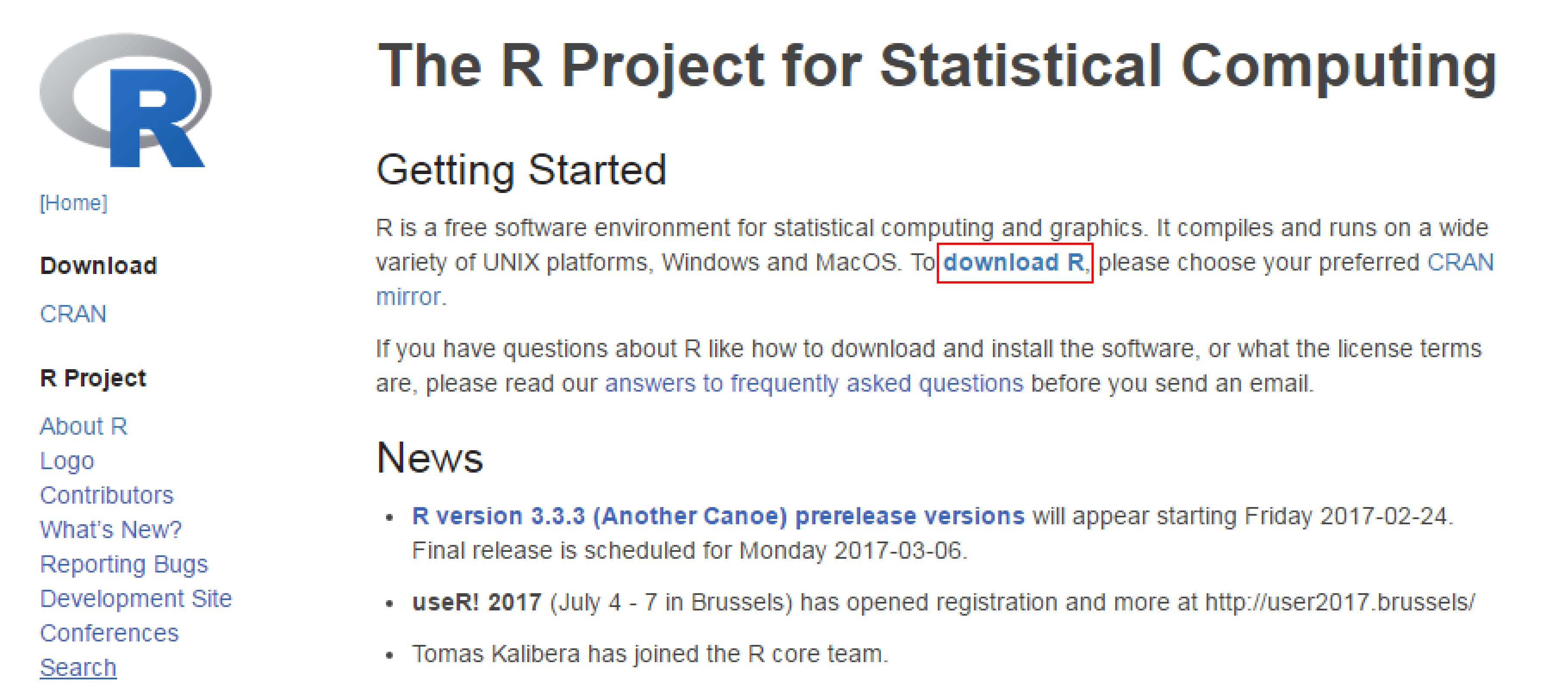


- R for big data
  - Packages
    - CRAN
      - 4501 packages
      - Growth
  - CRAN Task Views
  - Basic stack
    - MASS
    - Hmisc
    - plyr
    - data.table
    - reshape2
    - ggplot2
    - caret
    - party
    - tm
  - Integrated platforms
    - RStudio
      - RStudio Server
      - Shiny Server
  - Visualization
    - lattice
    - hexbin
      - Bioconductor
    - ash
      - Bioconductor
    - scagnostics
    - bigvis
    - igraph
    - iplots
    - rgl
  - Data formats
    - Flat text
      - readLines
      - read.table
      - read.fwf
      - sqldf
    - HDF5
      - hdf5
    - SQL
      - sqldf
      - RODBC
      - Rmysql
      - RJDBC
      - ROracle
    - NoSQL
      - MongoDB
        - rmongodb
        - RMongo
      - CouchDB
        - R4CouchDB
    - JSON
      - RJSONIO
      - rjson
    - XML
      - XML
    - HBase
      - rhbase (Revolution)
  - Large & out-of-memory data
    - ff
    - bigmemory
    - biglm
    - biglars
    - bigrf
  - Efficiency
    - C API
    - Rcpp
  - Parallel
    - parallel
    - Rmpi
    - foreach
      - doMC
      - doSNOW
      - doMPI
  - in-database analytics
    - Teradata
      - TeradataR
    - Oracle
      - Advanced Analytics
    - Netezza
    - SAP HANA
      - RHANA
  - GPU
    - gputools
  - As backend
    - Rserve
      - Python
        - pyrserve
      - Web
    - Python
      - rpy2
      - pyper
    - Perl
      - Statistics::useR
    - Web
      - FastRWeb
      - Rserve
      - Shiny Server
    - SAS
      - PROC IML (version 9.22+)
      - PROC_R macro
  - Glue
    - Rgraphviz
      - Graphviz
    - Rcpp
    - rJava
    - rPython
  - Hadoop
    - RHIPE
    - rmr
    - HadoopStreaming
    - Revolution Analytics
      - rhbase
      - rhdfs

http://www.r-bloggers.com/stepping-up-to-big-data-with-r-and-python-a-mind-map-of-all-the-packages-you-will-ever-need/

Install R

- For Windows
  - ✓ http://www.r-project.org
  - ✓ Download ‘R base’ file: the latest version is 3.3.3 for windows



[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Development Site
Conferences
Search

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

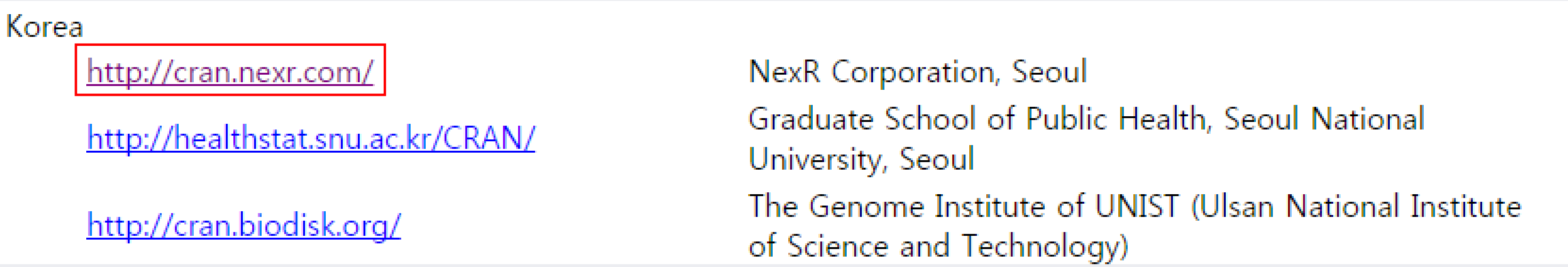
If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 3.3.3 (Another Canoe) prerelease versions** will appear starting Friday 2017-02-24. Final release is scheduled for Monday 2017-03-06.
- **useR! 2017** (July 4 - 7 in Brussels) has opened registration and more at http://user2017.brussels/
- Tomas Kalibera has joined the R core team.

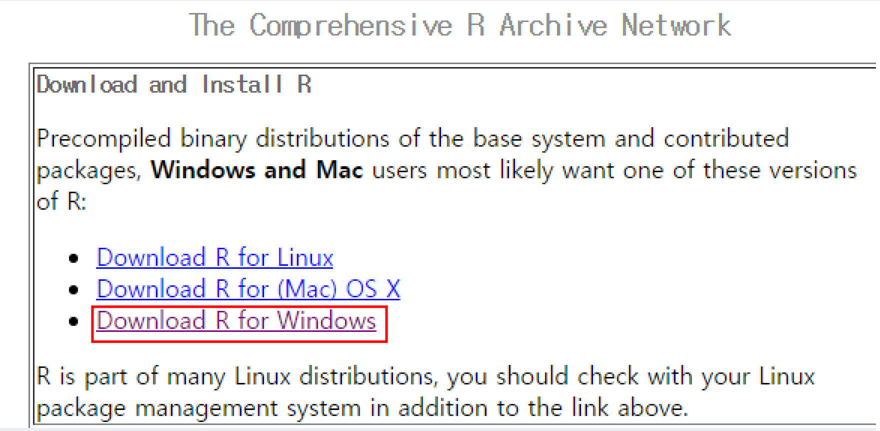
Install R

- For Windows
  - ✓ http://www.r-project.org
  - ✓ Download 'R base' file: the latest version is 3.3.3 for windows



Korea

http://cran.nexr.com/	NexR Corporation, Seoul
http://healthstat.snu.ac.kr/CRAN/	Graduate School of Public Health, Seoul National University, Seoul
http://cran.biodisk.org/	The Genome Institute of UNIST (Ulsan National Institute of Science and Technology)



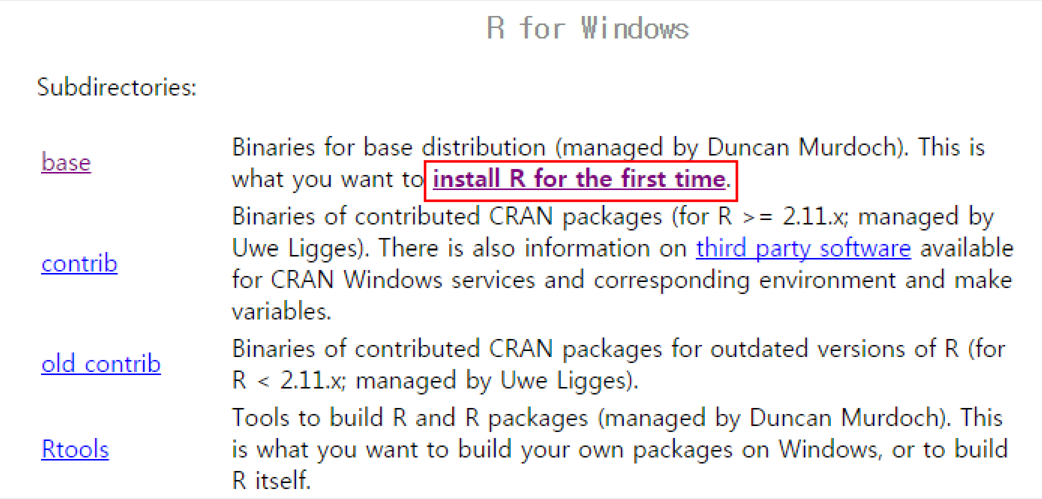
The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.



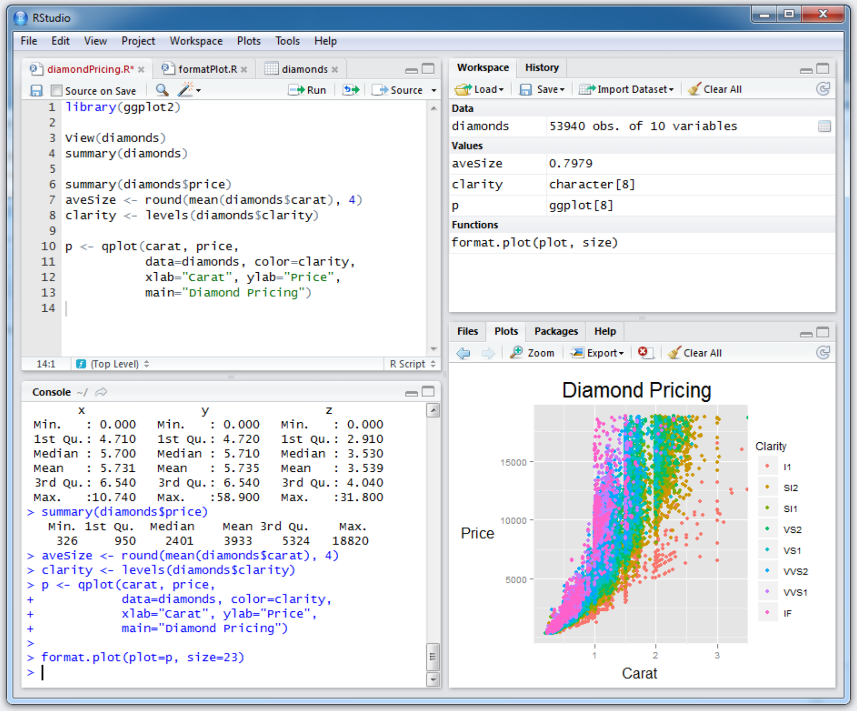
R for Windows

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.11.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.11.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Install R Studio

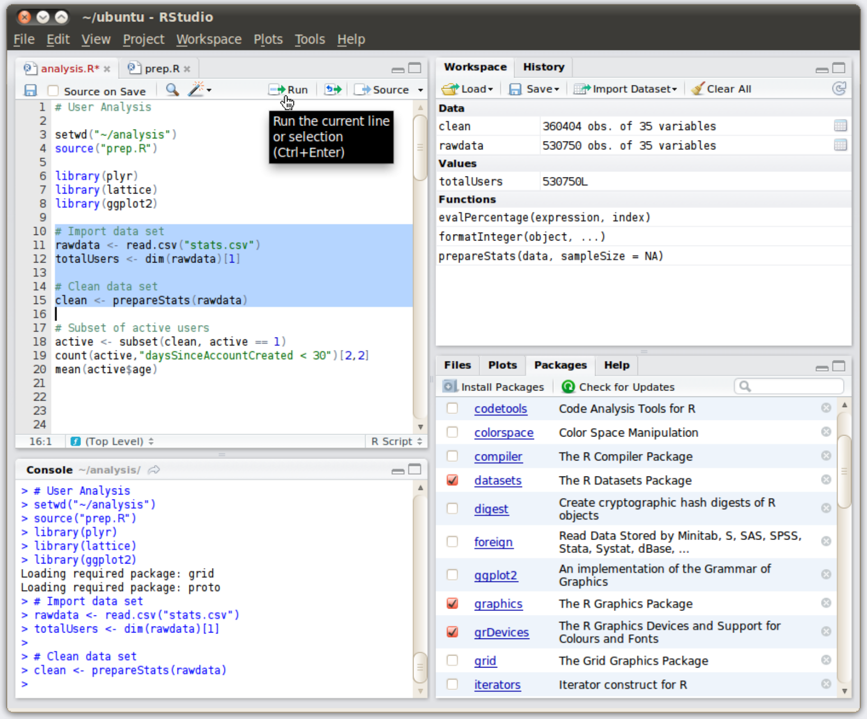
- RStudio
  - ✓ A free and open source integrated development environment (IDE) for R
    - Available OS: Windows, Mac, or Linux
    - Can run it over the web using RStudio Server



```r
library(ggplot2)

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")

```

```
       x               y               z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(plot=p, size=23)
> 
```

Data	
diamonds	53940 obs. of 10 variables
Values	
aveSize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	


~/ubuntu - RStudio
File Edit View Project Workspace Plots Tools Help
analysis.R*
prep.R
Source on Save
Run
Source
Run the current line or selection (Ctrl+Enter)
16:1
(Top Level)
R Script
Console ~/analysis/
Workspace History
Load Save Import Dataset Clear All
Files Plots Packages Help
Install Packages
Check for Updates

```r
# User Analysis

setwd("~/analysis")
source("prep.R")

library(plyr)
library(lattice)
library(ggplot2)

# Import data set
rawdata <- read.csv("stats.csv")
totalUsers <- dim(rawdata)[1]

# Clean data set
clean <- prepareStats(rawdata)

# Subset of active users
active <- subset(clean, active == 1)
count(active,"daysSinceAccountCreated < 30")[2,2]
mean(active$age)
```

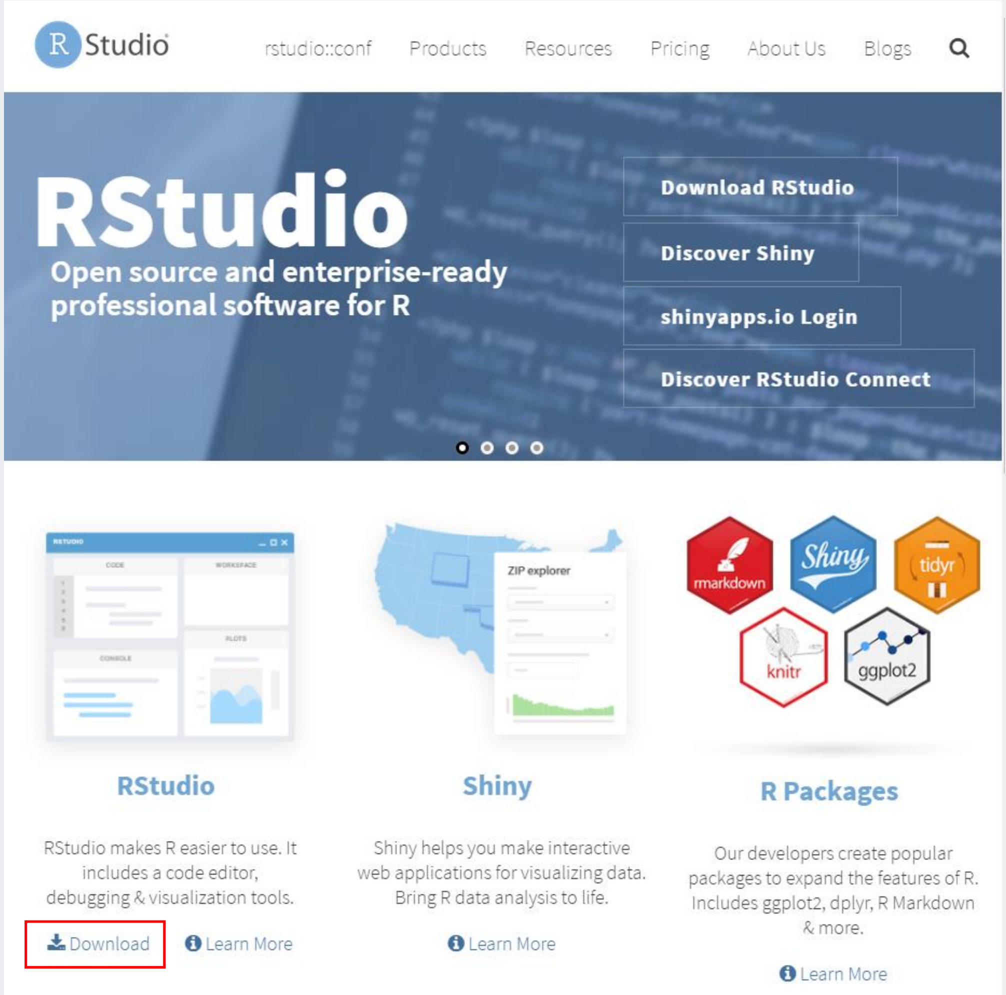
```
> # User Analysis
> setwd("~/analysis")
> source("prep.R")
> library(plyr)
> library(lattice)
> library(ggplot2)
Loading required package: grid
Loading required package: proto
> # Import data set
> rawdata <- read.csv("stats.csv")
> totalUsers <- dim(rawdata)[1]
>
> # Clean data set
> clean <- prepareStats(rawdata)
>
```

Data	
clean	360404 obs. of 35 variables
rawdata	530750 obs. of 35 variables
Values	
totalUsers	530750L
Functions	
evalPercentage(expression, index)	
formatInteger(object, ...)	
prepareStats(data, sampleSize = NA)	

☐	codetools	Code Analysis Tools for R
☐	colorspace	Color Space Manipulation
☐	compiler	The R Compiler Package
☑	datasets	The R Datasets Package
☐	digest	Create cryptographic hash digests of R objects
☐	foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
☐	ggplot2	An implementation of the Grammar of Graphics
☑	graphics	The R Graphics Package
☑	grDevices	The R Graphics Devices and Support for Colours and Fonts
☐	grid	The Grid Graphics Package
☐	iterators	Iterator construct for R

Install R Studio

- RStudio
  - ✓ https://www.rstudio.com/



RStudio

Open source and enterprise-ready professional software for R

RStudio

RStudio makes R easier to use. It includes a code editor, debugging & visualization tools.

Shiny

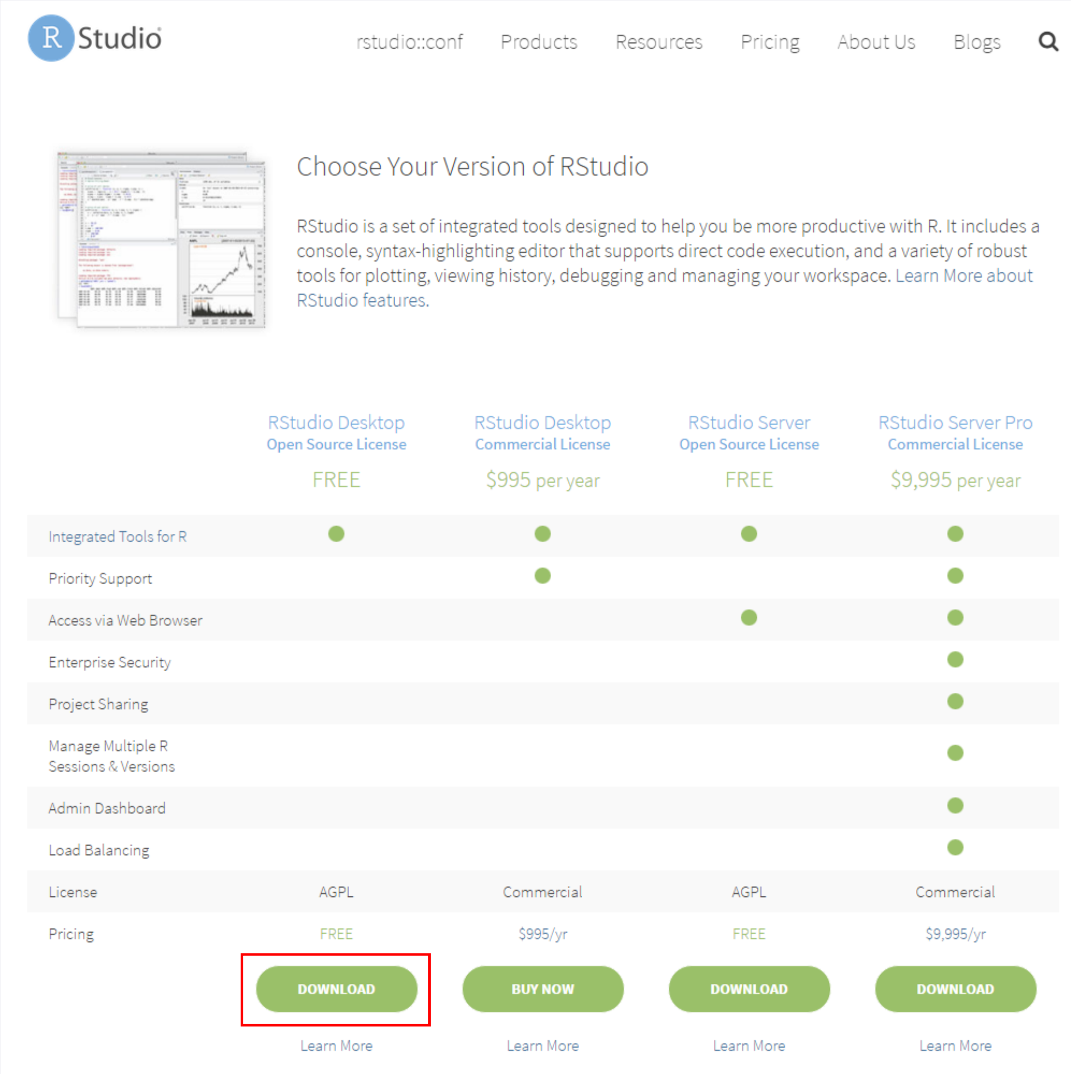
Shiny helps you make interactive web applications for visualizing data. Bring R data analysis to life.

R Packages

Our developers create popular packages to expand the features of R. Includes ggplot2, dplyr, R Markdown & more.

Install R Studio

- RStudio



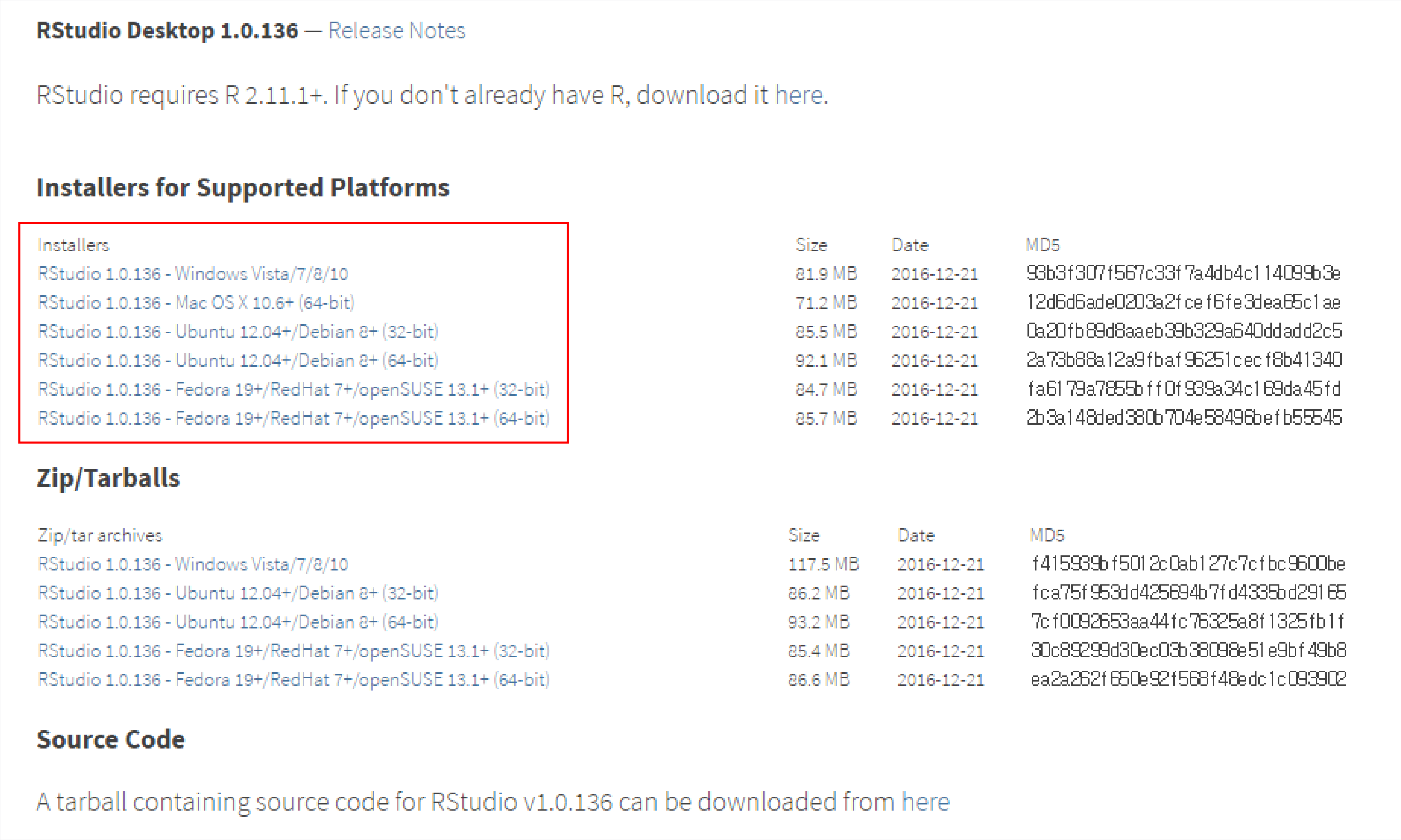
Choose Your Version of RStudio

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace. Learn More about RStudio features.

	RStudio Desktop Open Source License FREE	RStudio Desktop Commercial License $995 per year	RStudio Server Open Source License FREE	RStudio Server Pro Commercial License $9,995 per year
Integrated Tools for R	●	●	●	●
Priority Support		●		●
Access via Web Browser			●	●
Enterprise Security				●
Project Sharing				●
Manage Multiple R Sessions & Versions				●
Admin Dashboard				●
Load Balancing				●
License	AGPL	Commercial	AGPL	Commercial
Pricing	FREE	$995/yr	FREE	$9,995/yr
	DOWNLOAD	BUY NOW	DOWNLOAD	DOWNLOAD
	Learn More	Learn More	Learn More	Learn More

Install R Studio

- RStudio



RStudio Desktop 1.0.136 — Release Notes

RStudio requires R 2.11.1+. If you don't already have R, download it here.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.0.136 - Windows Vista/7/8/10	81.9 MB	2016-12-21	93b3f307f567c33f7a4db4c114099b3e
RStudio 1.0.136 - Mac OS X 10.6+ (64-bit)	71.2 MB	2016-12-21	12d6d6ade0203a2fcef6fe3dea65c1ae
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (32-bit)	85.5 MB	2016-12-21	0a20fb89d8aaeb39b329a640ddadd2c5
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (64-bit)	92.1 MB	2016-12-21	2a73b88a12a9fbaf96251cecf8b41340
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	84.7 MB	2016-12-21	fa6179a7855bff0f939a34c169da45fd
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	85.7 MB	2016-12-21	2b3a148ded380b704e58496befb55545

Zip/Tarballs

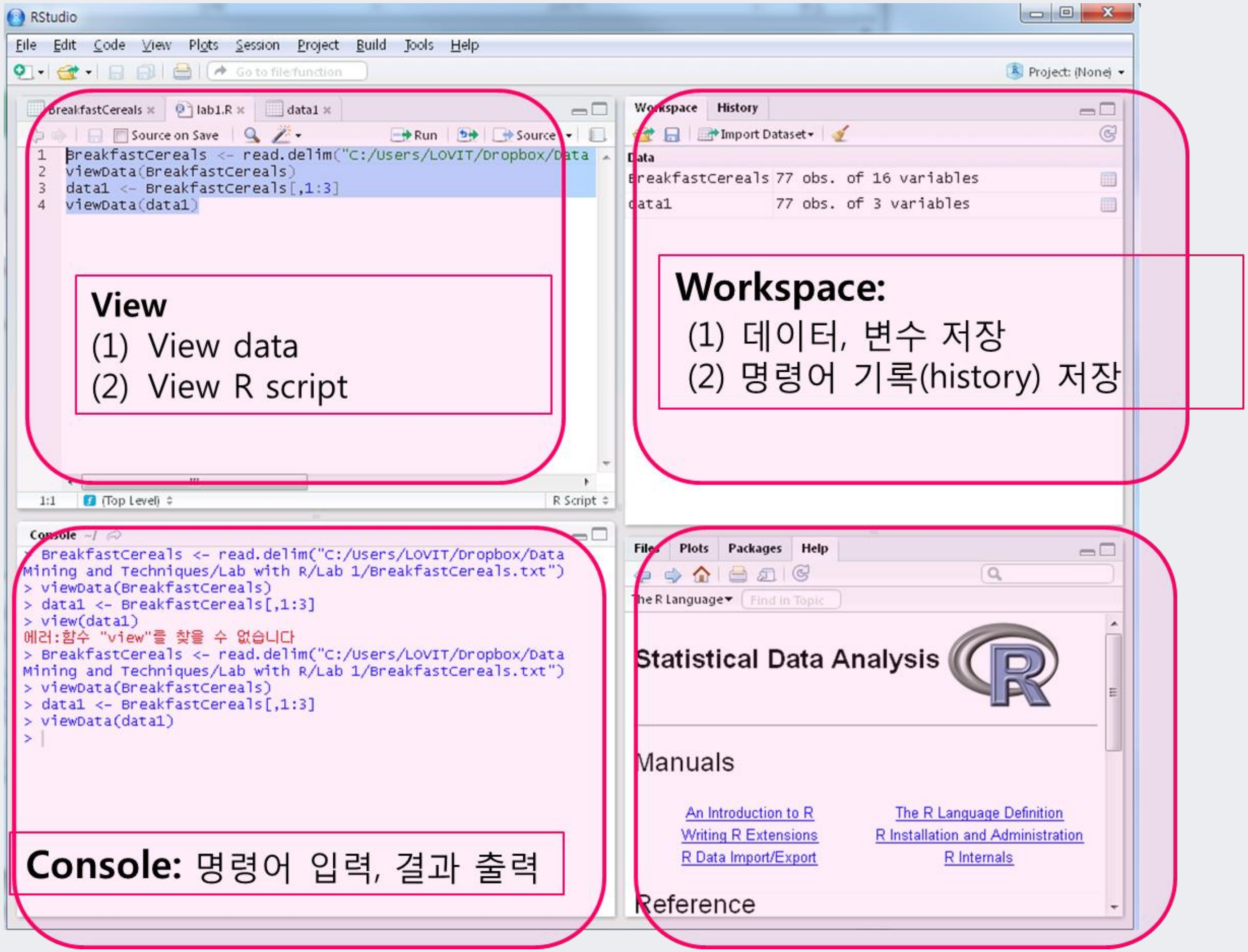
Zip/tar archives	Size	Date	MD5
RStudio 1.0.136 - Windows Vista/7/8/10	117.5 MB	2016-12-21	f415939bf5012c0ab127c7cfbc9600be
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (32-bit)	86.2 MB	2016-12-21	fca75f953dd425694b7fd4335bd29165
RStudio 1.0.136 - Ubuntu 12.04+/Debian 8+ (64-bit)	93.2 MB	2016-12-21	7cf0092653aa44fc76325a8f1325fb1f
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	85.4 MB	2016-12-21	30c89299d30ec03b38098e51e9bf49b8
RStudio 1.0.136 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	86.6 MB	2016-12-21	ea2a262f650e92f568f48edc1c093902

Source Code

A tarball containing source code for RStudio v1.0.136 can be downloaded from here

R Studio

- R Studio windows



```
BreakfastCereals <- read.delim("C:/Users/LOVIT/Dropbox/Data
viewData(BreakfastCereals)
data1 <- BreakfastCereals[,1:3]
viewData(data1)
```

View
(1) View data
(2) View R script

Data

BreakfastCereals	77 obs. of 16 variables
data1	77 obs. of 3 variables

Workspace:
(1) 데이터, 변수 저장
(2) 명령어 기록(history) 저장

```
> BreakfastCereals <- read.delim("C:/Users/LOVIT/Dropbox/Data
Mining and Techniques/Lab with R/Lab 1/BreakfastCereals.txt")
> viewData(BreakfastCereals)
> data1 <- BreakfastCereals[,1:3]
> view(data1)
에러:함수 "view"를 찾을 수 없습니다
> BreakfastCereals <- read.delim("C:/Users/LOVIT/Dropbox/Data
Mining and Techniques/Lab with R/Lab 1/BreakfastCereals.txt")
> viewData(BreakfastCereals)
> data1 <- BreakfastCereals[,1:3]
> viewData(data1)
> 
```

Console: 명령어 입력, 결과 출력

Statistical Data Analysis

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

Reference

The Three Sexy Skills of Data Geeks

- Statistics (Studying)
  - ✓ The most important skill and the hardest to learn
  - ✓ Deep and rigorous discipline and actively progressing
- Data Munging (Suffering)
  - ✓ The painful process of cleaning, parsing, and proofing one's data before it's suitable for analysis
  - ✓ Real world data is messy!!
- Visualization (Storytelling)
  - ✓ exploratory data visualizations: intended to facilitate a data analyst's understanding of the data
  - ✓ communicate to a wider audience, whose goal is to visually advocate for a hypothesis

Reading list 1: A very short history of data science

AGENDA

01 R Overview

02 Handling Different Data Types

03 Text Processing

Data Types in R

- Questions
  - ✓ Q1: Are all variables homogeneous?
  - ✓ Q2: Are there more than one record?

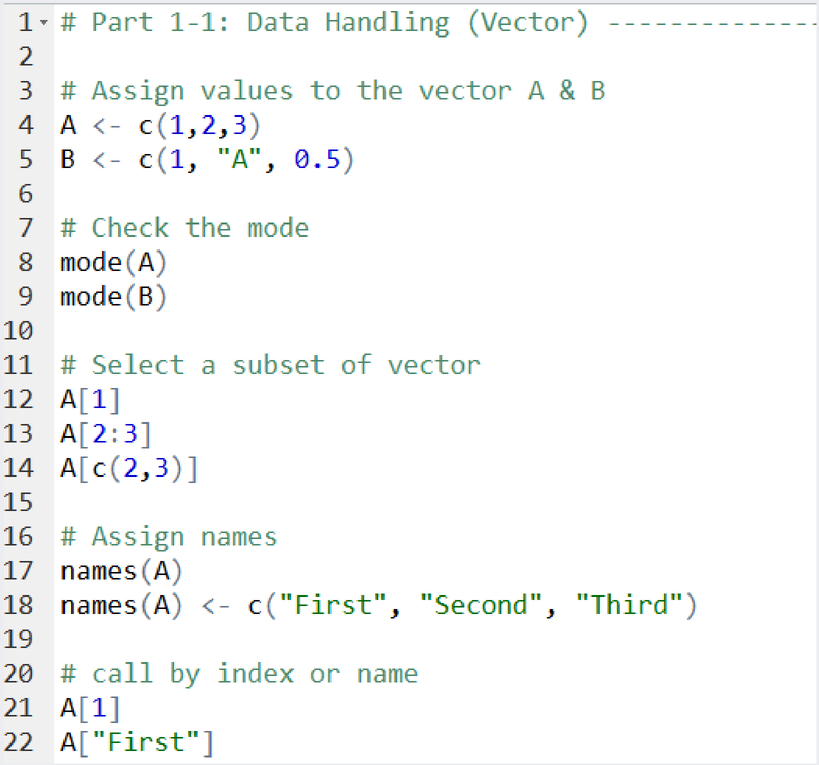
Attribute\No. Records	1	>= 2
Homogeneous	Vector	Matrix or Array
Heterogeneous	List	Dataframe

- Dataframe makes R powerful to analyze heterogeneous multivariate data

Data Types in R

Scalar | Vector | List | Matrix | Array | Factor | Data.frame

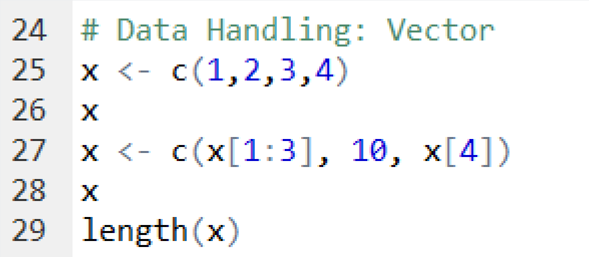
- Vector
  - ✓ Vectors are homogeneous
    - All elements in a vector should be the same mode
  - ✓ Vector has an index for each element
    - A set of indices returns the corresponding sub-vector
    - Index starts from 1 (python: 0)
  - ✓ The elements of a vector can have its own name
  - ✓ Vectors in R is a column-wise vectors



```
# Part 1-1: Data Handling (Vector) -------------

# Assign values to the vector A & B
A <- c(1,2,3)
B <- c(1, "A", 0.5)

# Check the mode
mode(A)
mode(B)

# Select a subset of vector
A[1]
A[2:3]
A[c(2,3)]

# Assign names
names(A)
names(A) <- c("First", "Second", "Third")

# call by index or name
A[1]
A["First"]
```

Handling Vectors

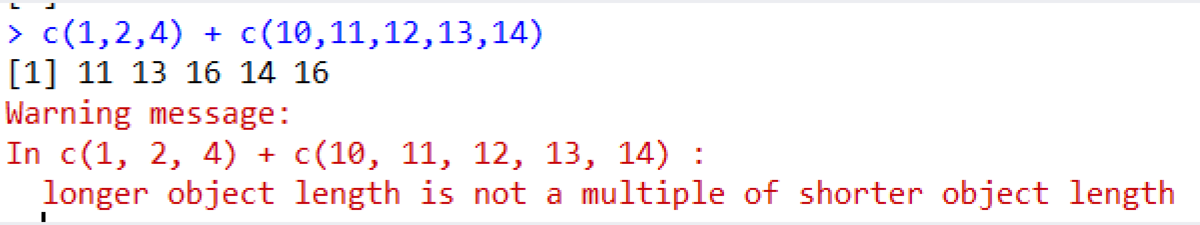
- Vector initiation
  - ✓ Do not have to initiate → creation and value assignment are done at the same time
    - a <- 3: create a vector named ‘a’ and assign the value 3 to it
- Add elements to an existing vector
  - ✓ The size of a vector is fixed when it is created
  - ✓ We have to recreate the vector if we want to add or remove some elements



```r
# Data Handling: Vector
x <- c(1,2,3,4)
x
x <- c(x[1:3], 10, x[4])
x
length(x)
```

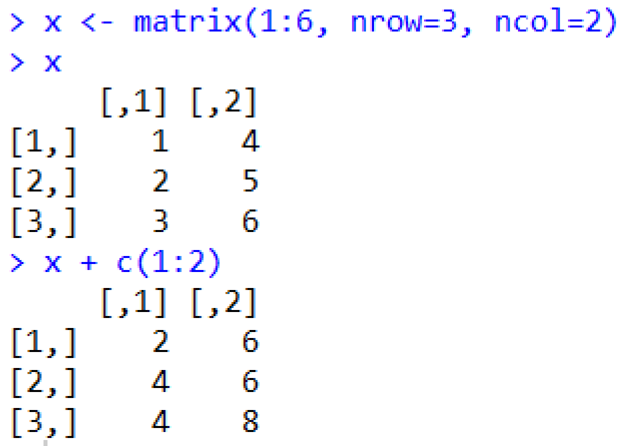
Handling Vectors

- Vector reuse
  - ✓ When R conduct an operation with two vectors, the shorter vector is reused to avoid an error

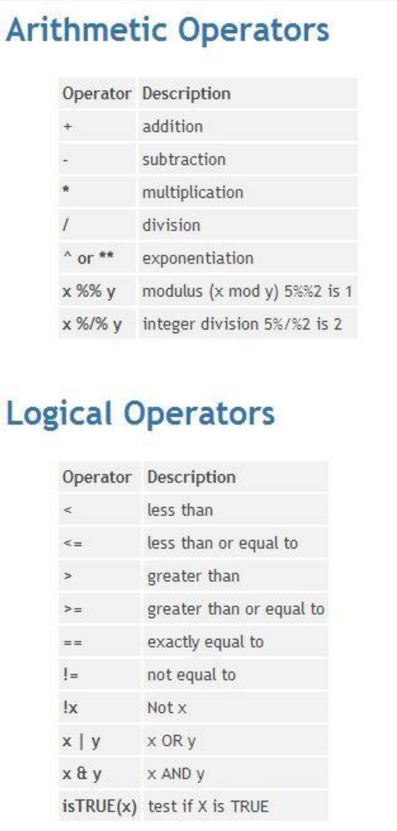


```
> c(1,2,4) + c(10,11,12,13,14)
[1] 11 13 16 14 16
Warning message:
In c(1, 2, 4) + c(10, 11, 12, 13, 14) :
  longer object length is not a multiple of shorter object length
```

  - ✓ Column-first



```
> x <- matrix(1:6, nrow=3, ncol=2)
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> x + c(1:2)
     [,1] [,2]
[1,]    2    6
[2,]    4    6
[3,]    4    8
```



Arithmetic Operators

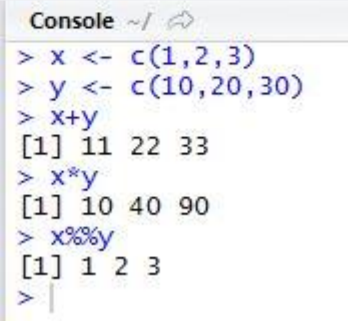
Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Logical Operators

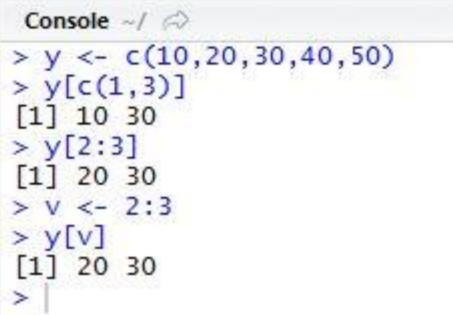
Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Handling Vectors

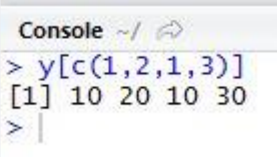
- Vector operations are element-wise
- Vector indexing
  - ✓ Extract a subset of vectors
  - ✓ Index can be used redundantly
  - ✓ A negative index is used to remove the corresponding element



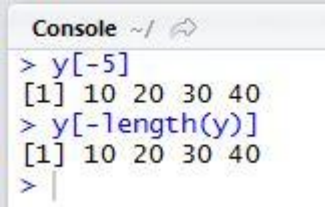
```
> x <- c(1,2,3)
> y <- c(10,20,30)
> x+y
[1] 11 22 33
> x*y
[1] 10 40 90
> x%%y
[1] 1 2 3
>
```



```
> y <- c(10,20,30,40,50)
> y[c(1,3)]
[1] 10 30
> y[2:3]
[1] 20 30
> v <- 2:3
> y[v]
[1] 20 30
>
```



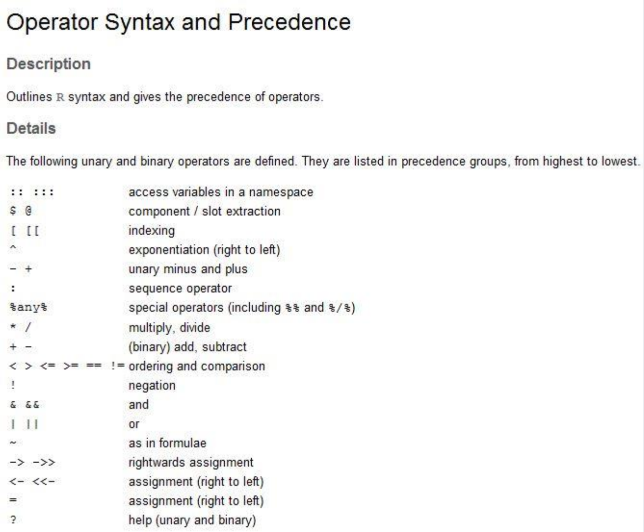
```
> y[c(1,2,1,3)]
[1] 10 20 10 30
>
```



```
> y[-5]
[1] 10 20 30 40
> y[-length(y)]
[1] 10 20 30 40
>
```

Handling Vectors

- Creating vectors with operators
  - ✓ : operator: create vectors with certain range
  - ✓ seq: a generalized version of “:” operator
  - ✓ rep: repeat values



Operator Syntax and Precedence

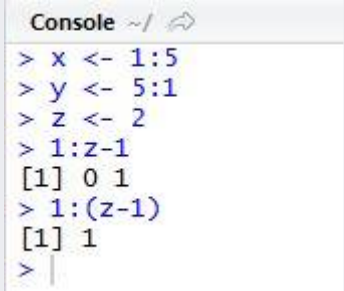
Description

Outlines R syntax and gives the precedence of operators.

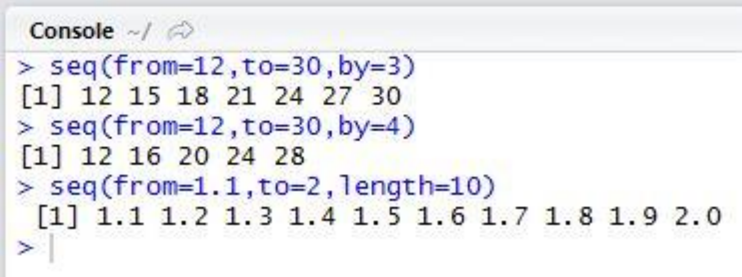
Details

The following unary and binary operators are defined. They are listed in precedence groups, from highest to lowest.

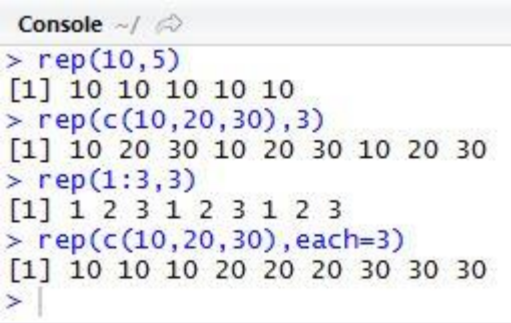
`:: :::`	access variables in a namespace
`$ @`	component / slot extraction
`[ [[`	indexing
`^`	exponentiation (right to left)
`- +`	unary minus and plus
`:`	sequence operator
`%any%`	special operators (including `%%` and `%/%`)
`* /`	multiply, divide
`+ -`	(binary) add, subtract
`< > <= >= == !=`	ordering and comparison
`!`	negation
`& &&`	and
`| ||`	or
`~`	as in formulae
`-> ->>`	rightwards assignment
`<- <<-`	assignment (right to left)
`=`	assignment (right to left)
`?`	help (unary and binary)



```
> x <- 1:5
> y <- 5:1
> z <- 2
> 1:z-1
[1] 0 1
> 1:(z-1)
[1] 1
> 
```



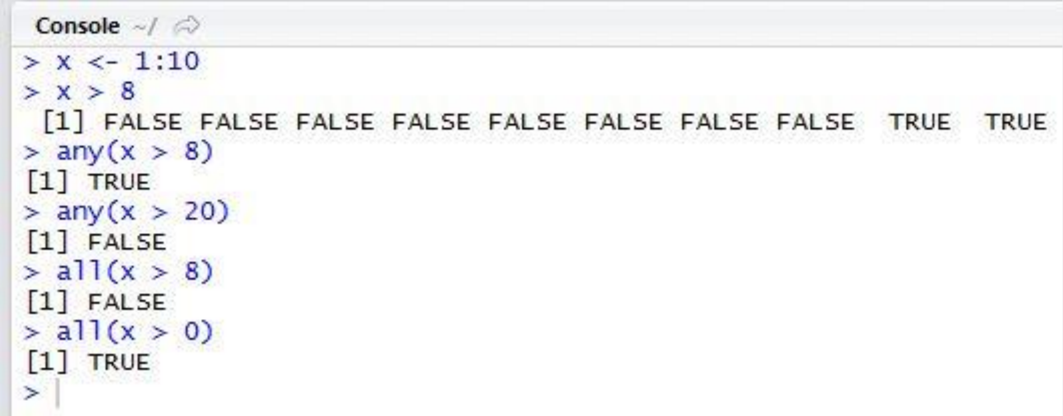
```
> seq(from=12,to=30,by=3)
[1] 12 15 18 21 24 27 30
> seq(from=12,to=30,by=4)
[1] 12 16 20 24 28
> seq(from=1.1,to=2,length=10)
 [1] 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
> 
```



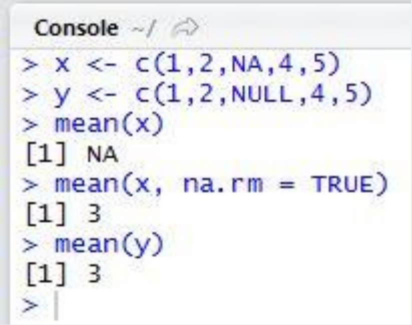
```
> rep(10,5)
[1] 10 10 10 10 10
> rep(c(10,20,30),3)
[1] 10 20 30 10 20 30 10 20 30
> rep(1:3,3)
[1] 1 2 3 1 2 3 1 2 3
> rep(c(10,20,30),each=3)
[1] 10 10 10 20 20 20 30 30 30
> 
```

Handling Vectors

- Apply conditions for each element in a vector
  - ✓ any() function: return TRUE if at least one of the elements satisfies the condiditon
  - ✓ all() function: return TRUE only when all elements satisfy the condition
- NA vs NULL
  - ✓ NA (Not Available): Some value exists but we cannot exactly know the value
  - ✓ NULL: Physically not exist



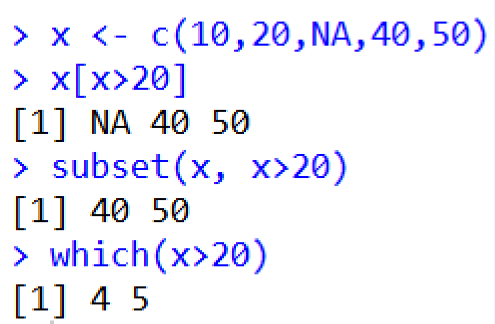
```
> x <- 1:10
> x > 8
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
> any(x > 8)
[1] TRUE
> any(x > 20)
[1] FALSE
> all(x > 8)
[1] FALSE
> all(x > 0)
[1] TRUE
> |
```



```
> x <- c(1,2,NA,4,5)
> y <- c(1,2,NULL,4,5)
> mean(x)
[1] NA
> mean(x, na.rm = TRUE)
[1] 3
> mean(y)
[1] 3
> |
```

Handling Vectors

- Filtering: Extract the element that satisfy a given condition
  - ✓ Directly extract from index
  - ✓ subset(): return the values that satisfy the condition
  - ✓ which(): return the indices that satisfy the condition

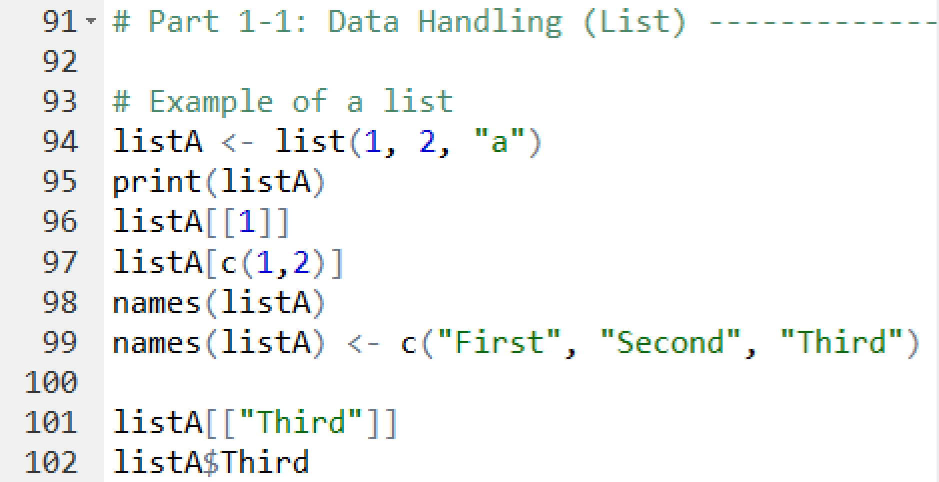


```
> x <- c(10,20,NA,40,50)
> x[x>20]
[1] NA 40 50
> subset(x, x>20)
[1] 40 50
> which(x>20)
[1] 4 5
```

Handling List

Scalar | Vector | List | Matrix | Array | Factor | Data.frame

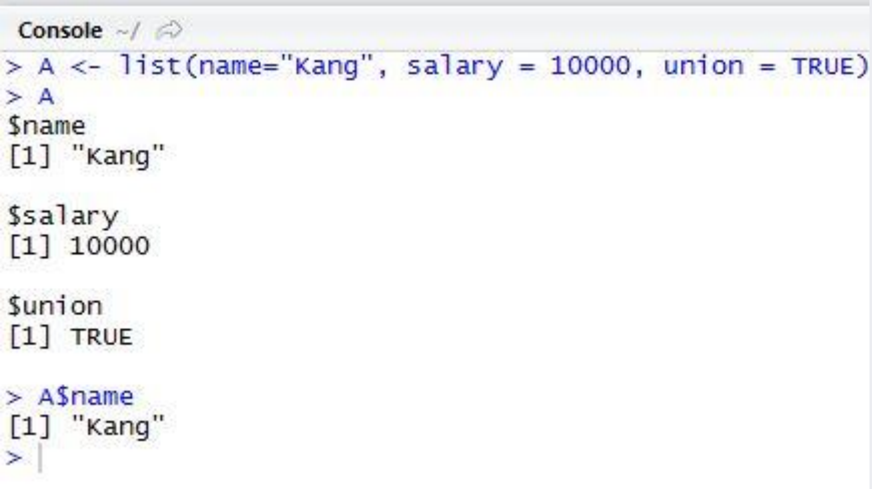
- Lists are heterogeneous
  - ✓ Element in a list can have different modes
  - ✓ List can have other structured object such as dataframe as its element
- Elements in a list are referred by their index
- Elements in a list can have their names



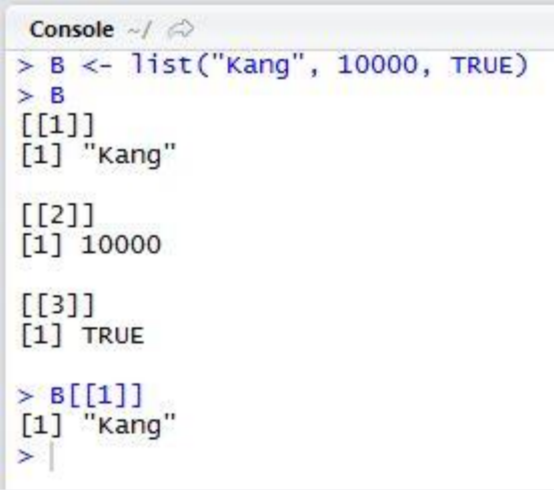
```r
# Part 1-1: Data Handling (List) -------------

# Example of a list
listA <- list(1, 2, "a")
print(listA)
listA[[1]]
listA[c(1,2)]
names(listA)
names(listA) <- c("First", "Second", "Third")

listA[["Third"]]
listA$Third
```

Handling List

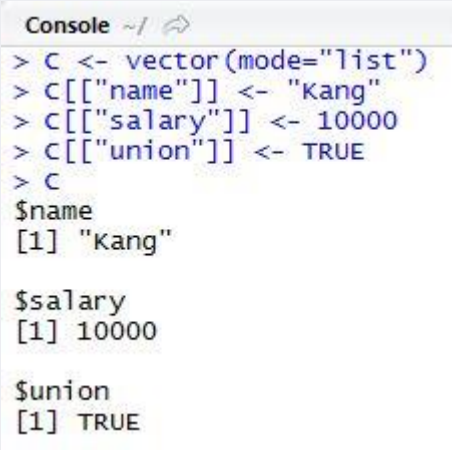
- Creating a list
  - ✓ Use list() or vector() function
    - ▪ Element names can be assigned using tags



```
> A <- list(name="Kang", salary = 10000, union = TRUE)
> A
$name
[1] "Kang"

$salary
[1] 10000

$union
[1] TRUE

> A$name
[1] "Kang"
>
```



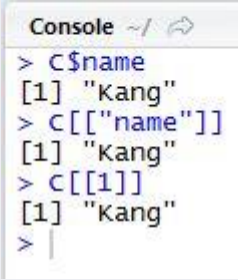
```
> B <- list("Kang", 10000, TRUE)
> B
[[1]]
[1] "Kang"

[[2]]
[1] 10000

[[3]]
[1] TRUE

> B[[1]]
[1] "Kang"
>
```



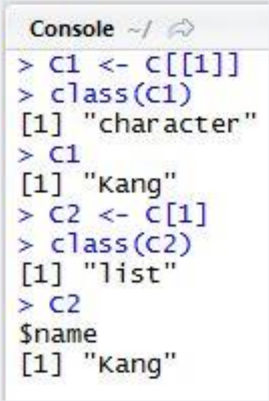
```
> C <- vector(mode="list")
> C[["name"]] <- "Kang"
> C[["salary"]] <- 10000
> C[["union"]] <- TRUE
> C
$name
[1] "Kang"

$salary
[1] 10000

$union
[1] TRUE
```

Handling List

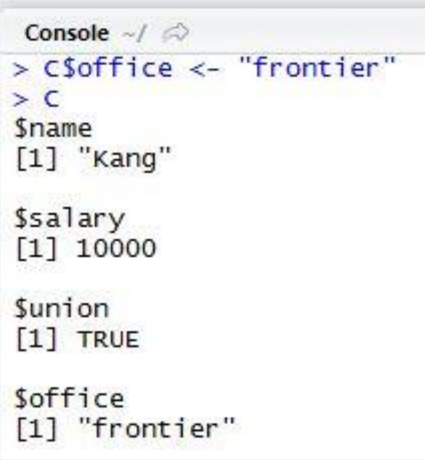
- List operations
  - ✓ List indexing
    - Three ways of accessing the elements in a list
    - list$element_name, list[[“element_name”]], list[[element’s index]]
    - A list is returned if [ ] is used
  - ✓ Add/remove element in a list
    - Add: use a new name
    - Remove: use NULL



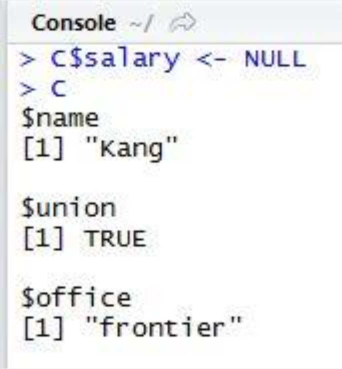
```
> C$name
[1] "Kang"
> C[["name"]]
[1] "Kang"
> C[[1]]
[1] "Kang"
>
```



```
> C1 <- C[[1]]
> class(C1)
[1] "character"
> C1
[1] "Kang"
> C2 <- C[1]
> class(C2)
[1] "list"
> C2
$name
[1] "Kang"
```



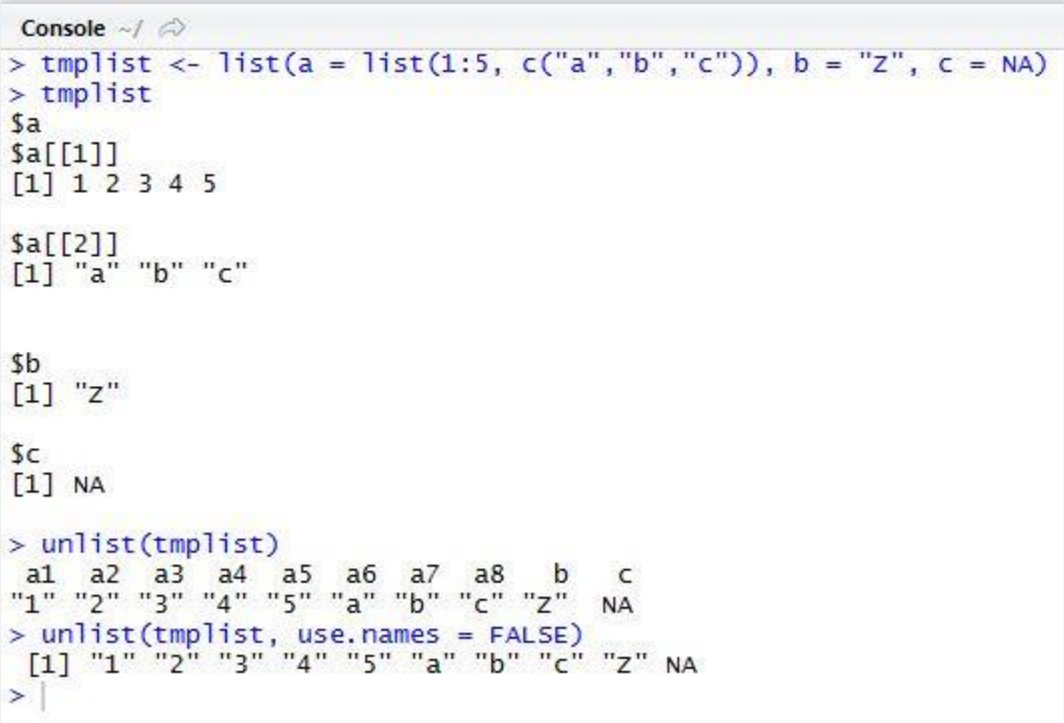
```
> C$office <- "frontier"
> C
$name
[1] "Kang"

$salary
[1] 10000

$union
[1] TRUE

$office
[1] "frontier"
```



```
> C$salary <- NULL
> C
$name
[1] "Kang"

$union
[1] TRUE

$office
[1] "frontier"
```

Handling List

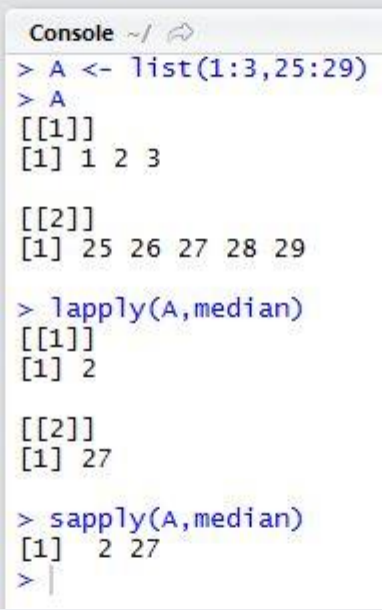
- List operations (cont’)
  - ✓ Unlist returns a vector with a single mode values



```
> tmplist <- list(a = list(1:5, c("a","b","c")), b = "Z", c = NA)
> tmplist
$a
$a[[1]]
[1] 1 2 3 4 5

$a[[2]]
[1] "a" "b" "c"


$b
[1] "Z"

$c
[1] NA

> unlist(tmplist)
 a1  a2  a3  a4  a5  a6  a7  a8   b   c 
"1" "2" "3" "4" "5" "a" "b" "c" "Z"  NA 
> unlist(tmplist, use.names = FALSE)
 [1] "1" "2" "3" "4" "5" "a" "b" "c" "Z" NA
> 
```

Handling List

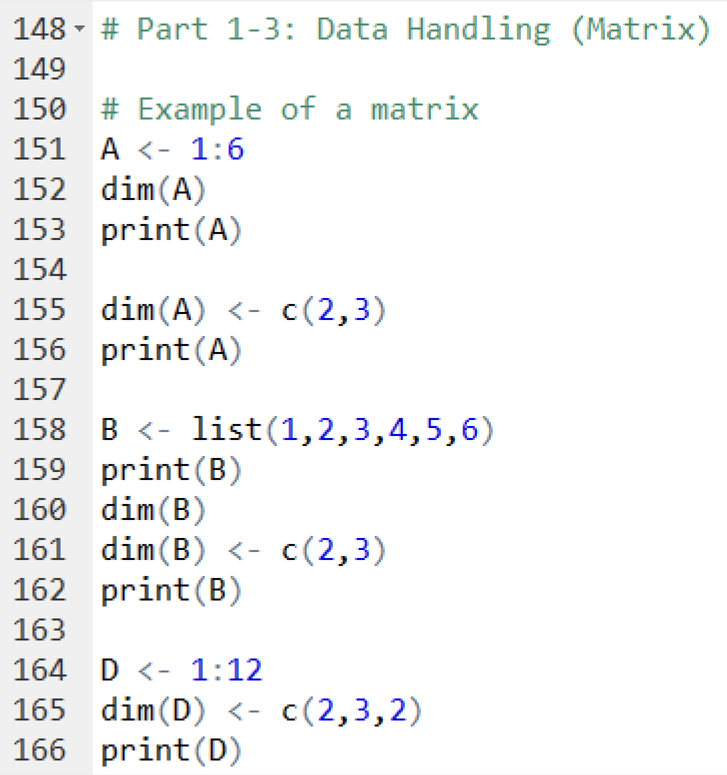
- Applying functions to list
  - ✓ lapply( ) returns a list while sapply( ) returns a vector



```
> A <- list(1:3,25:29)
> A
[[1]]
[1] 1 2 3

[[2]]
[1] 25 26 27 28 29

> lapply(A,median)
[[1]]
[1] 2

[[2]]
[1] 27

> sapply(A,median)
[1]  2 27
> 
```

Handling Matrix and Array

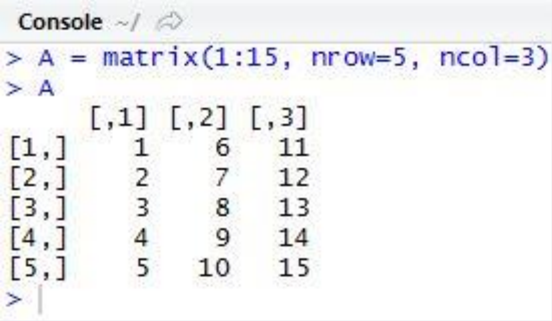
- Matrix
  - ✓ Matrix is a vector with dimensions
    - ▪ Vectors and lists can be transformed into a matrix
- Array
  - ✓ Matrix can be extended to n-dimensions
    - ▪ Indexed by multiple locations and returns subvectors



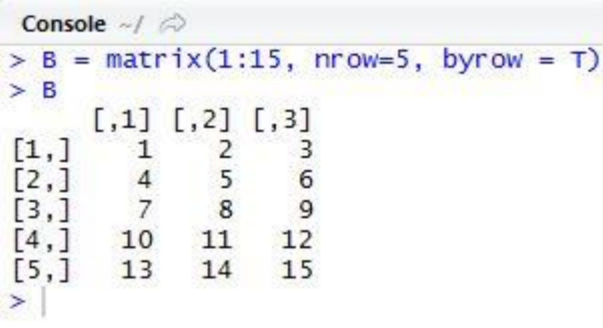
```r
# Part 1-3: Data Handling (Matrix)

# Example of a matrix
A <- 1:6
dim(A)
print(A)

dim(A) <- c(2,3)
print(A)

B <- list(1,2,3,4,5,6)
print(B)
dim(B)
dim(B) <- c(2,3)
print(B)

D <- 1:12
dim(D) <- c(2,3,2)
print(D)
```

Handling Matrix and Array

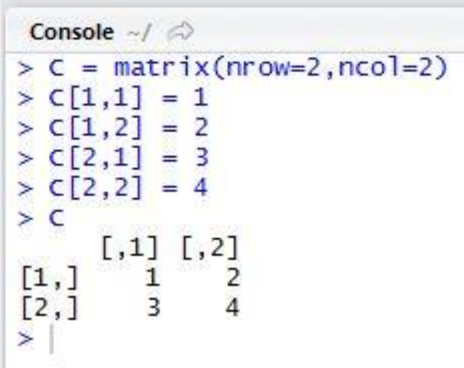
- Features of matrix in R
  - ✓ Index begins with 1 (0 for python)
  - ✓ Column-major order
- Create a matrix: matrix( )
  - ✓ Method 1: provide all elements and assign the number of columns and rows (column first)
  - ✓ Method 2: provide all elements and assign the number of columns and rows (use row first option)
  - ✓ Method 3: Create an empty matrix and fill each element in



```
> A = matrix(1:15, nrow=5, ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> 
```



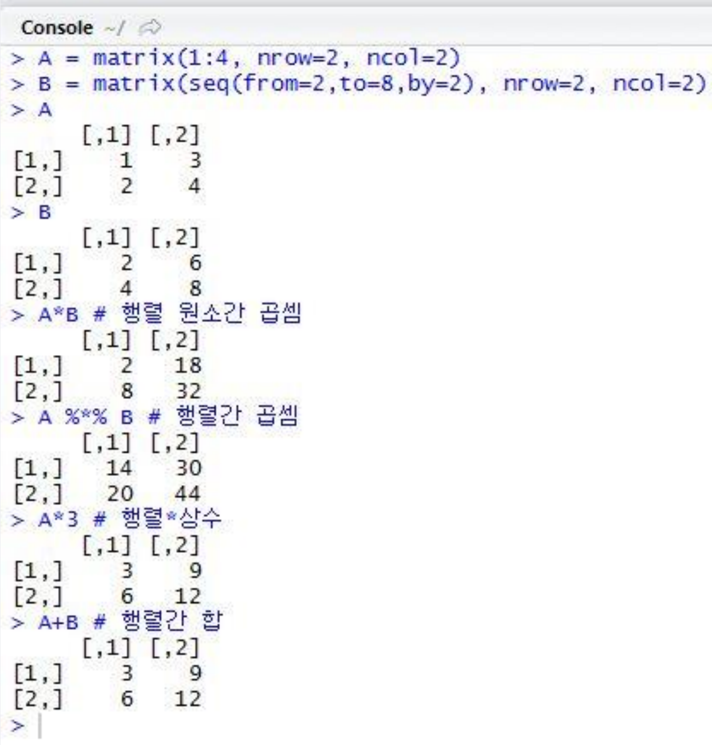
```
> B = matrix(1:15, nrow=5, byrow = T)
> B
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15
> 
```



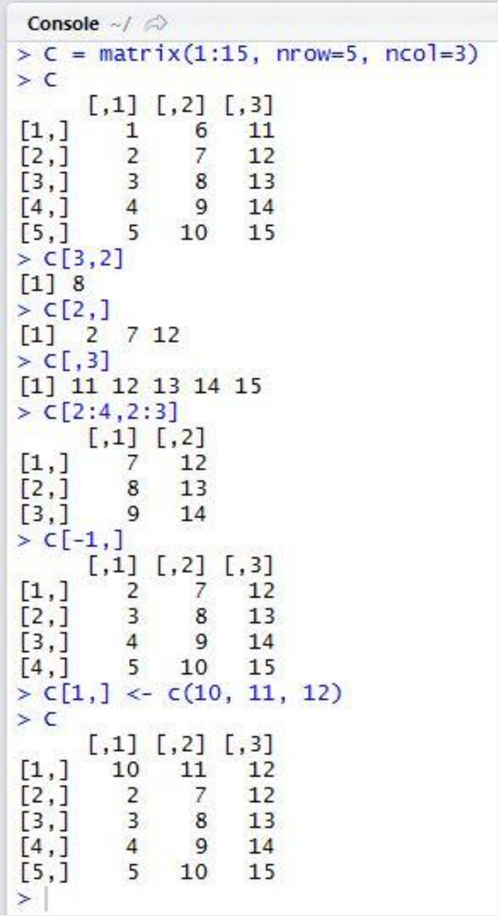
```
> C = matrix(nrow=2,ncol=2)
> C[1,1] = 1
> C[1,2] = 2
> C[2,1] = 3
> C[2,2] = 4
> C
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> 
```

Handling Matrix and Array

- Matrix operations
  - ✓ Linear algebra of matrix: matrix multiplication, matrix-constant multiplication, etc.
  - ✓ Indexing and filtering



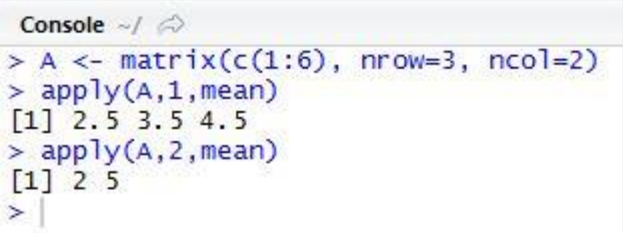
```
> A = matrix(1:4, nrow=2, ncol=2)
> B = matrix(seq(from=2,to=8,by=2), nrow=2, ncol=2)
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> B
     [,1] [,2]
[1,]    2    6
[2,]    4    8
> A*B # 행렬 원소간 곱셈
     [,1] [,2]
[1,]    2   18
[2,]    8   32
> A %*% B # 행렬간 곱셈
     [,1] [,2]
[1,]   14   30
[2,]   20   44
> A*3 # 행렬*상수
     [,1] [,2]
[1,]    3    9
[2,]    6   12
> A+B # 행렬간 합
     [,1] [,2]
[1,]    3    9
[2,]    6   12
> 
```



```
> C = matrix(1:15, nrow=5, ncol=3)
> C
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> C[3,2]
[1] 8
> C[2,]
[1]  2  7 12
> C[,3]
[1] 11 12 13 14 15
> C[2:4,2:3]
     [,1] [,2]
[1,]    7   12
[2,]    8   13
[3,]    9   14
> C[-1,]
     [,1] [,2] [,3]
[1,]    2    7   12
[2,]    3    8   13
[3,]    4    9   14
[4,]    5   10   15
> C[1,] <- c(10, 11, 12)
> C
     [,1] [,2] [,3]
[1,]   10   11   12
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> 
```

Handling Matrix and Array

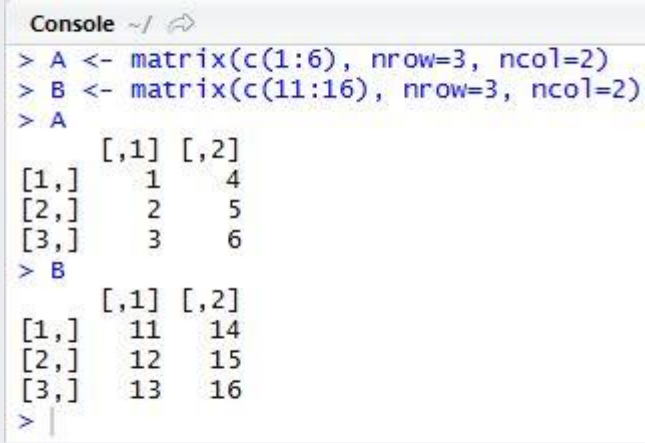
- Applying functions to the rows/columns of matrix
  - ✓ Use apply( ) function family: apply( ), sapply( ), tapply( ), lapply( ), etc.
    - apply(m, dimcode, f, fargs)
      - m: matrix
      - dimcode: dimension to apply (1: row, 2: column)
      - f: function
      - fargs: arguments needed to execute f



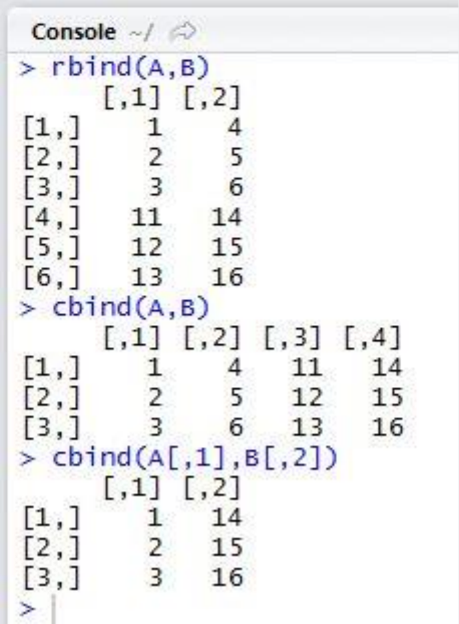
```
> A <- matrix(c(1:6), nrow=3, ncol=2)
> apply(A,1,mean)
[1] 2.5 3.5 4.5
> apply(A,2,mean)
[1] 2 5
> 
```

Handling Matrix and Array

- Modifying matrix
  - ✓ rbind( ) & cbind( ): combine two matrices
  - ✓ rbind( ): combine two matrices with the same column names (top and bottom)
  - ✓ cbind( ): combine two matrices with the same row names (left and right)



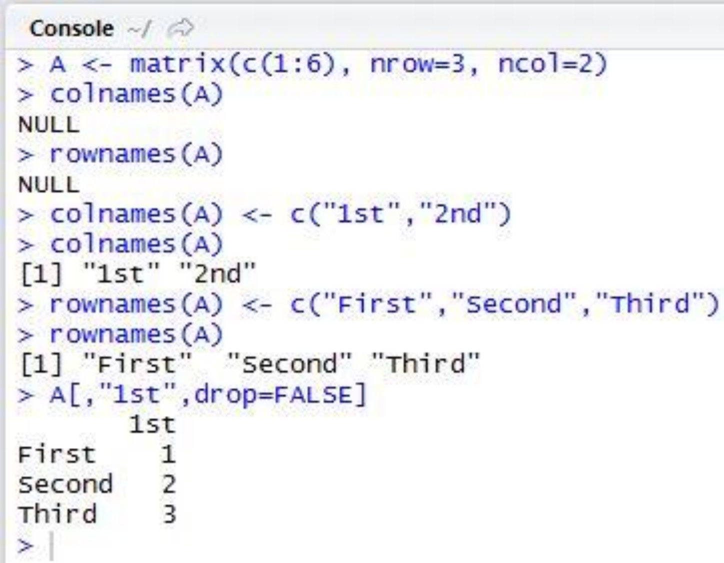
```
> A <- matrix(c(1:6), nrow=3, ncol=2)
> B <- matrix(c(11:16), nrow=3, ncol=2)
> A
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> B
     [,1] [,2]
[1,]   11   14
[2,]   12   15
[3,]   13   16
> 
```



```
> rbind(A,B)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
[4,]   11   14
[5,]   12   15
[6,]   13   16
> cbind(A,B)
     [,1] [,2] [,3] [,4]
[1,]    1    4   11   14
[2,]    2    5   12   15
[3,]    3    6   13   16
> cbind(A[,1],B[,2])
     [,1] [,2]
[1,]    1   14
[2,]    2   15
[3,]    3   16
> 
```

Handling Matrix and Array

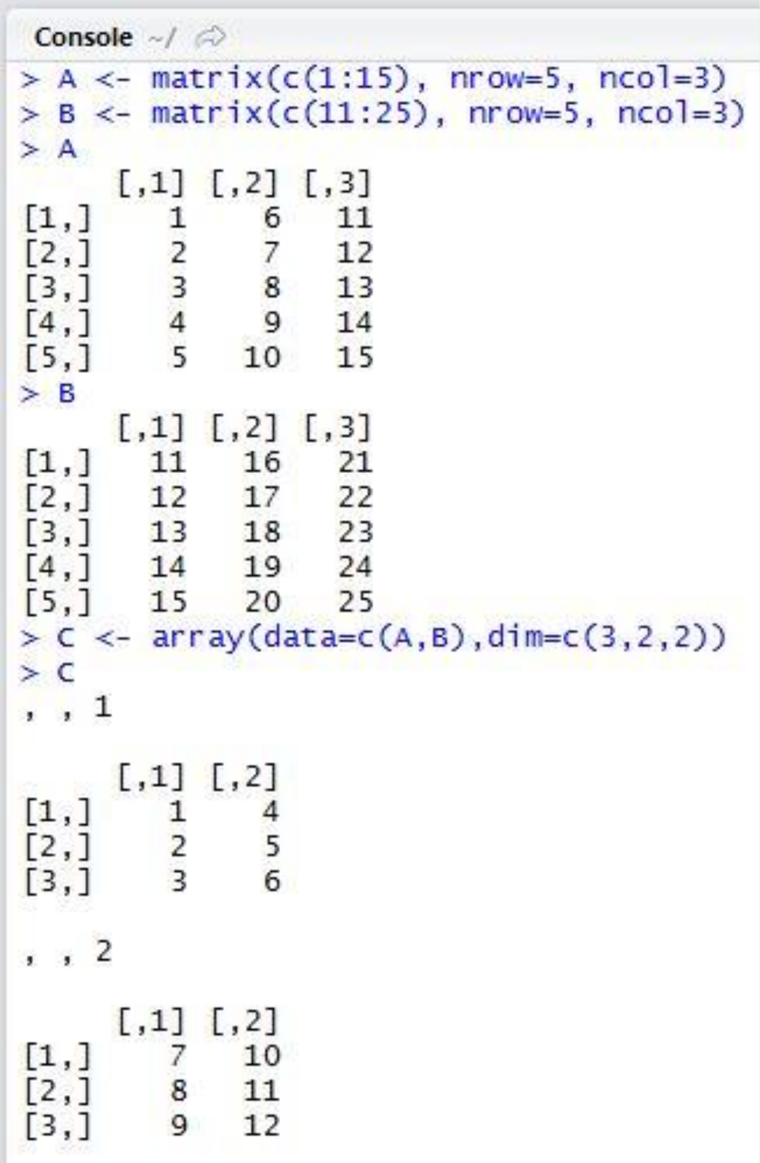
- Assign names for matrix columns/rows
  - ✓Use colnames( ) and rownames( )



```
> A <- matrix(c(1:6), nrow=3, ncol=2)
> colnames(A)
NULL
> rownames(A)
NULL
> colnames(A) <- c("1st","2nd")
> colnames(A)
[1] "1st" "2nd"
> rownames(A) <- c("First","Second","Third")
> rownames(A)
[1] "First"  "Second" "Third"
> A[,"1st",drop=FALSE]
       1st
First    1
Second   2
Third    3
> 
```

Handling Matrix and Array

- High dimensional array
  - ✓ Use array( ) function

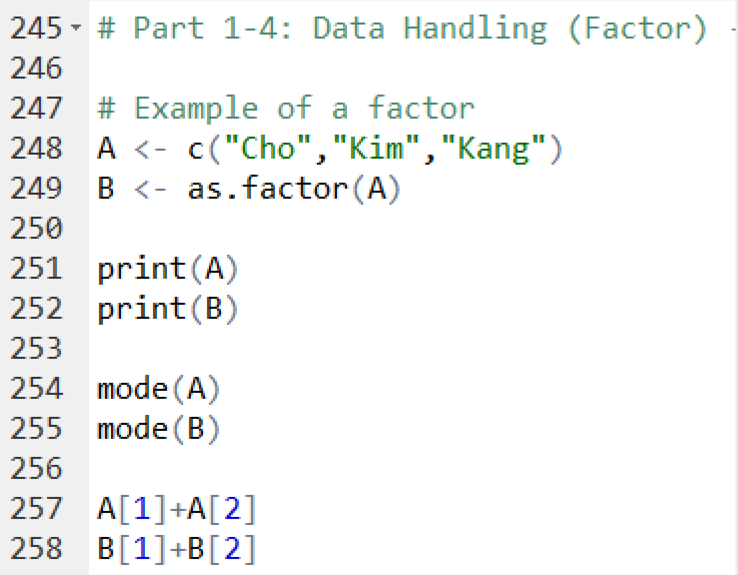


```
> A <- matrix(c(1:15), nrow=5, ncol=3)
> B <- matrix(c(11:25), nrow=5, ncol=3)
> A
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> B
     [,1] [,2] [,3]
[1,]   11   16   21
[2,]   12   17   22
[3,]   13   18   23
[4,]   14   19   24
[5,]   15   20   25
> C <- array(data=c(A,B),dim=c(3,2,2))
> C
, , 1

     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

, , 2

     [,1] [,2]
[1,]    7   10
[2,]    8   11
[3,]    9   12
```

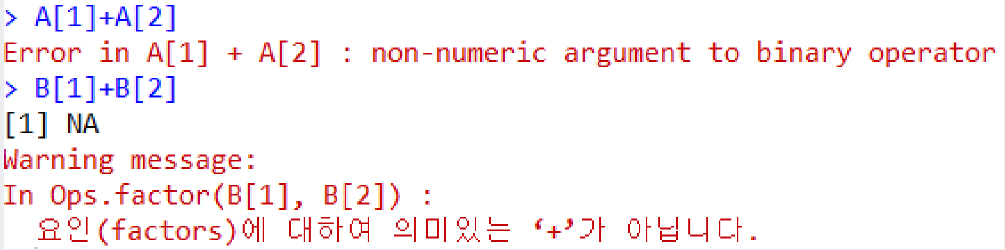
Handling Factor

Scalar | Vector | List | Matrix | Array | Factor | Data.frame

- Factor
  - ✓ Vector representation for nominal/categorical variables
    - ▪ Factor has its levels that are equivalent the number of possible values
- Usage
  - ✓ Categorical variable representation: 1 level for 1 category
  - ✓ Grouping and tagging
- Factor levels must be consistent!



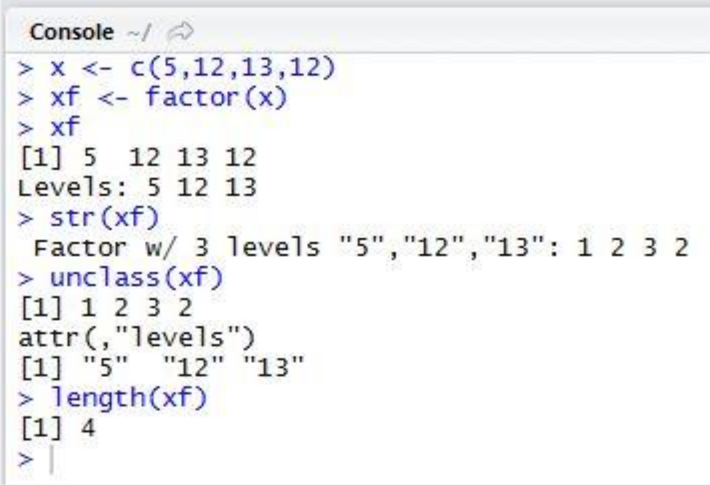
```
# Part 1-4: Data Handling (Factor) -

# Example of a factor
A <- c("Cho","Kim","Kang")
B <- as.factor(A)

print(A)
print(B)

mode(A)
mode(B)

A[1]+A[2]
B[1]+B[2]
```



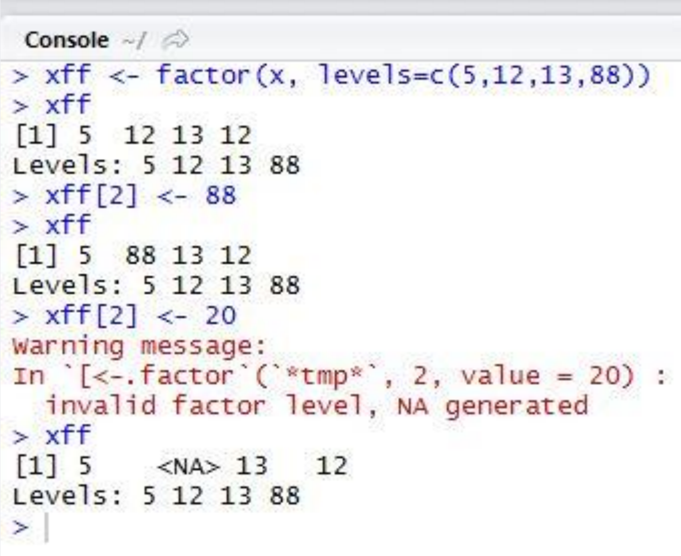
```
> A[1]+A[2]
Error in A[1] + A[2] : non-numeric argument to binary operator
> B[1]+B[2]
[1] NA
Warning message:
In Ops.factor(B[1], B[2]) :
  요인(factors)에 대하여 의미있는 ‘+’가 아닙니다.
```

Handling Factor

- Factor
  - ✓ Factor in R is a vector with additional information
    - Additional information is a set of non-redundant values called level
    - The length of a factor is the number of elements, not levels
    - Possible to add a new level
    - A new value with non-existing level is considered as NA



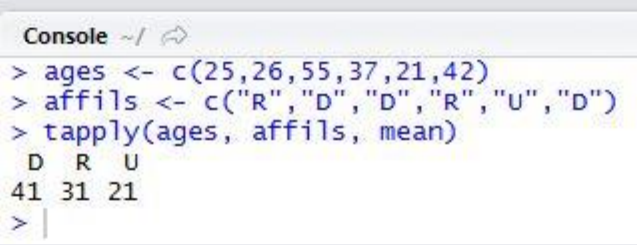
```
> x <- c(5,12,13,12)
> xf <- factor(x)
> xf
[1] 5  12 13 12
Levels: 5 12 13
> str(xf)
 Factor w/ 3 levels "5","12","13": 1 2 3 2
> unclass(xf)
[1] 1 2 3 2
attr(,"levels")
[1] "5"  "12" "13"
> length(xf)
[1] 4
> 
```



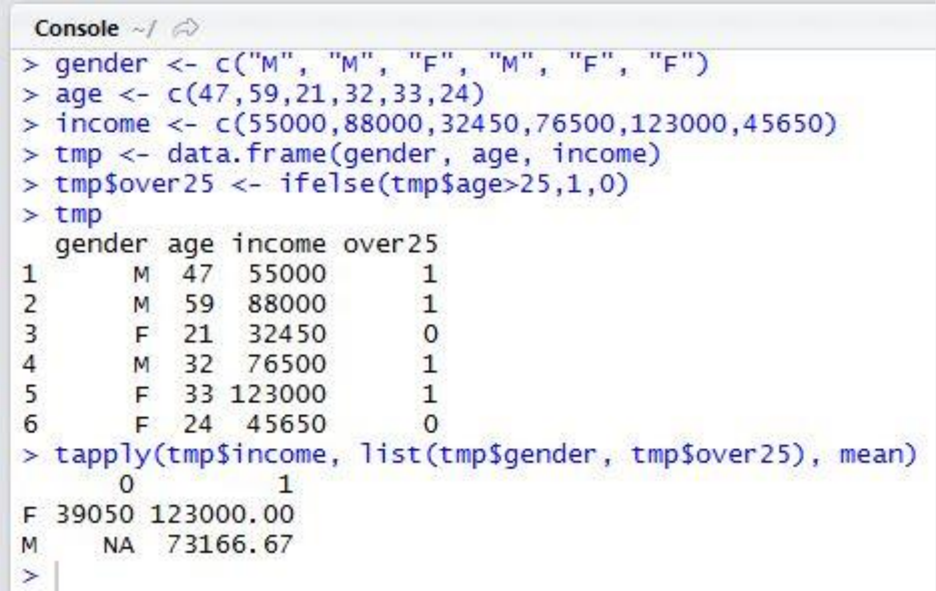
```
> xff <- factor(x, levels=c(5,12,13,88))
> xff
[1] 5  12 13 12
Levels: 5 12 13 88
> xff[2] <- 88
> xff
[1] 5  88 13 12
Levels: 5 12 13 88
> xff[2] <- 20
Warning message:
In `[<-.factor`(`*tmp*`, 2, value = 20) :
  invalid factor level, NA generated
> xff
[1] 5    <NA> 13   12
Levels: 5 12 13 88
> 
```

Handling Factor

- Applying function to a factor
  - ✓ tapply( )
    - Useful to make frequency table with different categories
    - Can be used to more than two factors



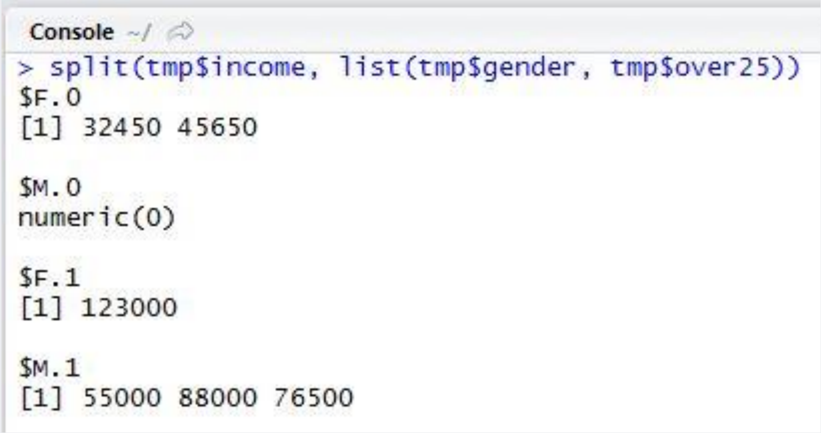
```
> ages <- c(25,26,55,37,21,42)
> affils <- c("R","D","D","R","U","D")
> tapply(ages, affils, mean)
 D  R  U
41 31 21
>
```



```
> gender <- c("M", "M", "F", "M", "F", "F")
> age <- c(47,59,21,32,33,24)
> income <- c(55000,88000,32450,76500,123000,45650)
> tmp <- data.frame(gender, age, income)
> tmp$over25 <- ifelse(tmp$age>25,1,0)
> tmp
  gender age income over25
1      M  47  55000      1
2      M  59  88000      1
3      F  21  32450      0
4      M  32  76500      1
5      F  33 123000      1
6      F  24  45650      0
> tapply(tmp$income, list(tmp$gender, tmp$over25), mean)
      0         1
F 39050 123000.00
M    NA  73166.67
>
```

Handling Factor

- Applying function to a factor
  - ✓ split( ) function
    - Used to make groups
    - can be used to more than two factors

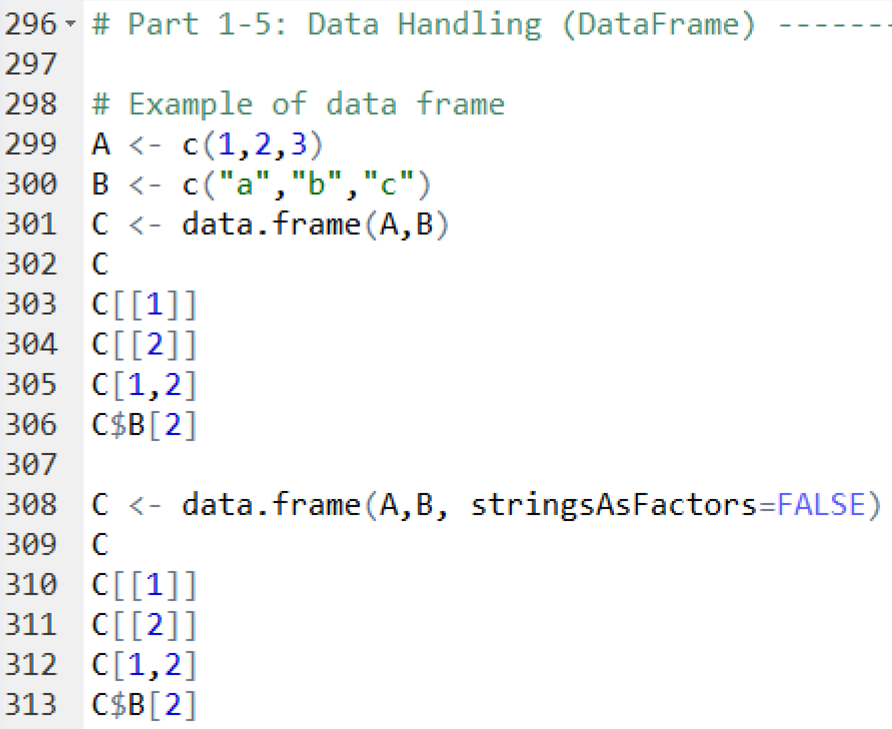


```
> split(tmp$income, list(tmp$gender, tmp$over25))
$F.0
[1] 32450 45650

$M.0
numeric(0)

$F.1
[1] 123000

$M.1
[1] 55000 88000 76500
```

Handling Dataframe

Scalar | Vector | List | Matrix | Array | Factor | Data.frame

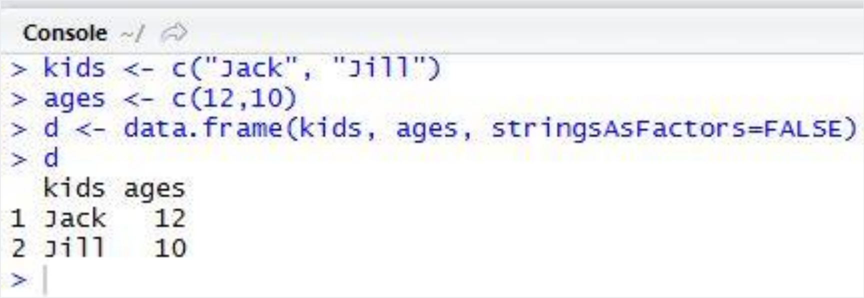
- Dataframe
  - ✓ A table with rows and columns
  - ✓ Regarded as a special case of list
  - ✓ Can have different modes for different columns
  - ✓ Elements in a column must have the same modes
  - ✓ Columns can have names



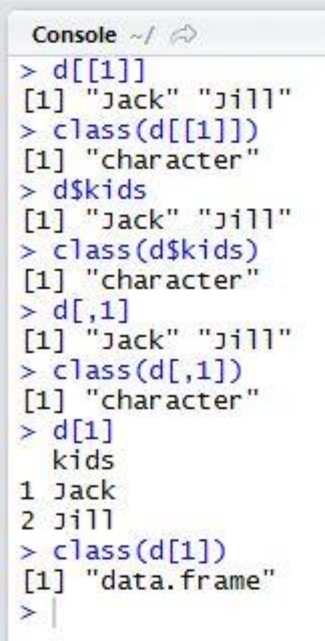
```r
# Part 1-5: Data Handling (DataFrame) -------

# Example of data frame
A <- c(1,2,3)
B <- c("a","b","c")
C <- data.frame(A,B)
C
C[[1]]
C[[2]]
C[1,2]
C$B[2]

C <- data.frame(A,B, stringsAsFactors=FALSE)
C
C[[1]]
C[[2]]
C[1,2]
C$B[2]
```

Handling Dataframe

- Creating and accessing Dataframe
  - ✓ Use data.frame( ) function to create a dataframe
  - ✓ Three ways to access a certain element



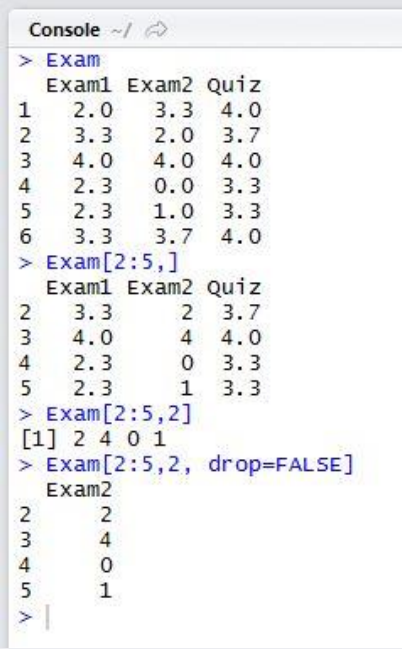
```
> kids <- c("Jack", "Jill")
> ages <- c(12,10)
> d <- data.frame(kids, ages, stringsAsFactors=FALSE)
> d
  kids ages
1 Jack   12
2 Jill   10
>
```



```
> d[[1]]
[1] "Jack" "Jill"
> class(d[[1]])
[1] "character"
> d$kids
[1] "Jack" "Jill"
> class(d$kids)
[1] "character"
> d[,1]
[1] "Jack" "Jill"
> class(d[,1])
[1] "character"
> d[1]
  kids
1 Jack
2 Jill
> class(d[1])
[1] "data.frame"
>
```

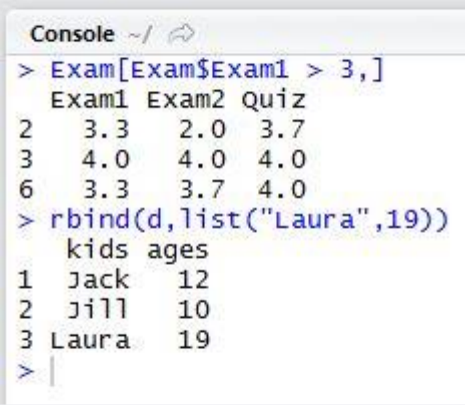
Handling Dataframe

- Extracting and filtering a subset of dataframe
  - ✓Same as matrix
- Combine dataframe
  - ✓Same as matrix



Console ~/

```
> Exam
  Exam1 Exam2 Quiz
1   2.0   3.3  4.0
2   3.3   2.0  3.7
3   4.0   4.0  4.0
4   2.3   0.0  3.3
5   2.3   1.0  3.3
6   3.3   3.7  4.0
> Exam[2:5,]
  Exam1 Exam2 Quiz
2   3.3     2  3.7
3   4.0     4  4.0
4   2.3     0  3.3
5   2.3     1  3.3
> Exam[2:5,2]
[1] 2 4 0 1
> Exam[2:5,2, drop=FALSE]
  Exam2
2     2
3     4
4     0
5     1
> |
```

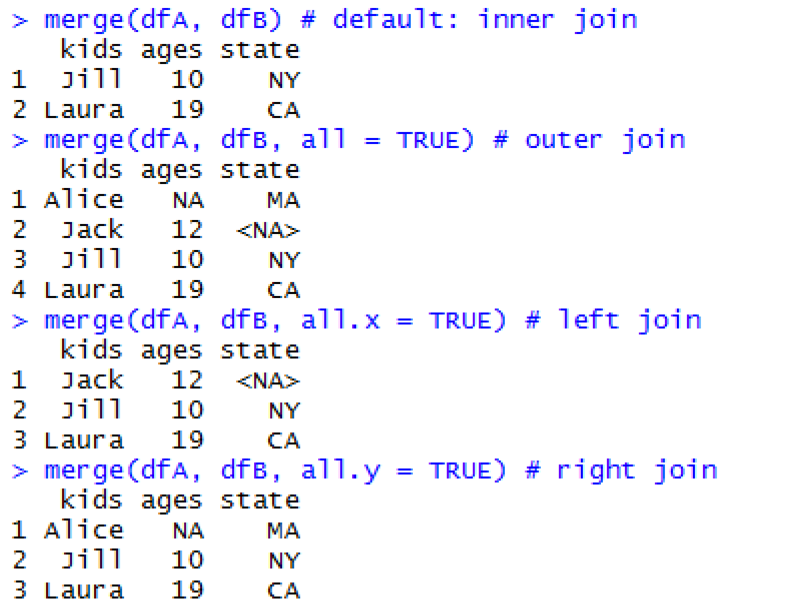


Console ~/

```
> Exam[Exam$Exam1 > 3,]
  Exam1 Exam2 Quiz
2   3.3   2.0  3.7
3   4.0   4.0  4.0
6   3.3   3.7  4.0
> rbind(d,list("Laura",19))
   kids ages
1  Jack   12
2  Jill   10
3 Laura   19
> |
```

Handling Dataframe

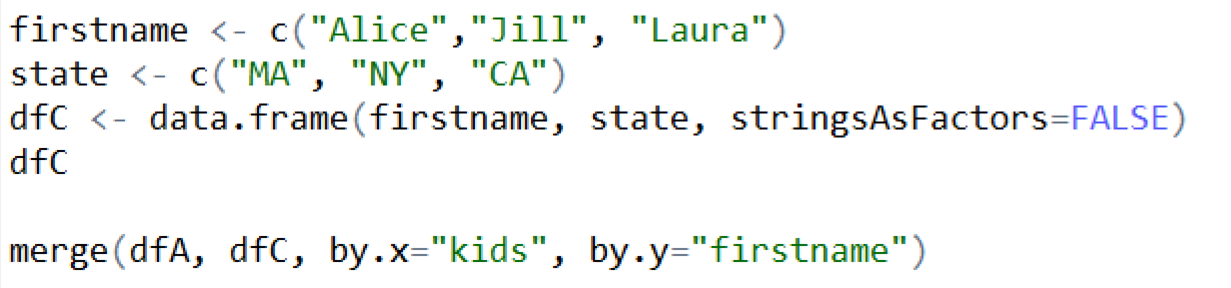
- Merge dataframes
  - ✓ If there are more than on sources of data tables in a database
  - ✓ Inner/outer/left/right joins are possible



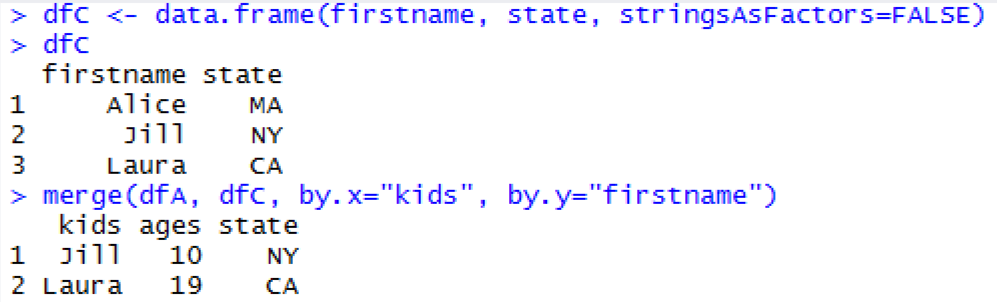
```
> merge(dfA, dfB) # default: inner join
   kids ages state
1  Jill   10    NY
2 Laura   19    CA
> merge(dfA, dfB, all = TRUE) # outer join
   kids ages state
1 Alice   NA    MA
2  Jack   12  <NA>
3  Jill   10    NY
4 Laura   19    CA
> merge(dfA, dfB, all.x = TRUE) # left join
   kids ages state
1  Jack   12  <NA>
2  Jill   10    NY
3 Laura   19    CA
> merge(dfA, dfB, all.y = TRUE) # right join
   kids ages state
1 Alice   NA    MA
2  Jill   10    NY
3 Laura   19    CA
```

Handling Dataframe

- Merge dataframes
  - ✓ If different data frames have different column name strategy, explicitly state the column names to use



```
firstname <- c("Alice","Jill", "Laura")
state <- c("MA", "NY", "CA")
dfC <- data.frame(firstname, state, stringsAsFactors=FALSE)
dfC

merge(dfA, dfC, by.x="kids", by.y="firstname")
```



```
> dfC <- data.frame(firstname, state, stringsAsFactors=FALSE)
> dfC
  firstname state
1     Alice    MA
2      Jill    NY
3     Laura    CA
> merge(dfA, dfC, by.x="kids", by.y="firstname")
   kids ages state
1  Jill   10    NY
2 Laura   19    CA
```

AGENDA

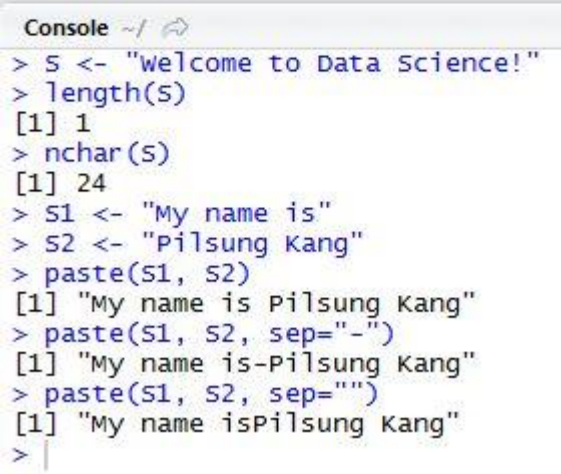
01 R Overview

02 Handling Different Data Types

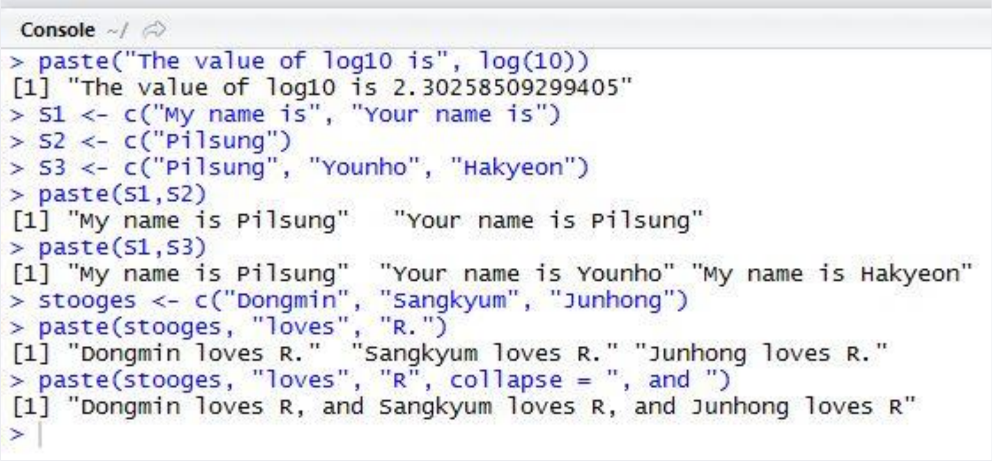
03 Text Processing

Text Processing

- The length of a string
  - ✓ Use nchar( ) function instead of length( )
    - Space and special characters can be counted as well
- Concatenate strings
  - ✓ Use paste( ) function
    - Various spacing strategies can be used
    - Non-character values are also possible



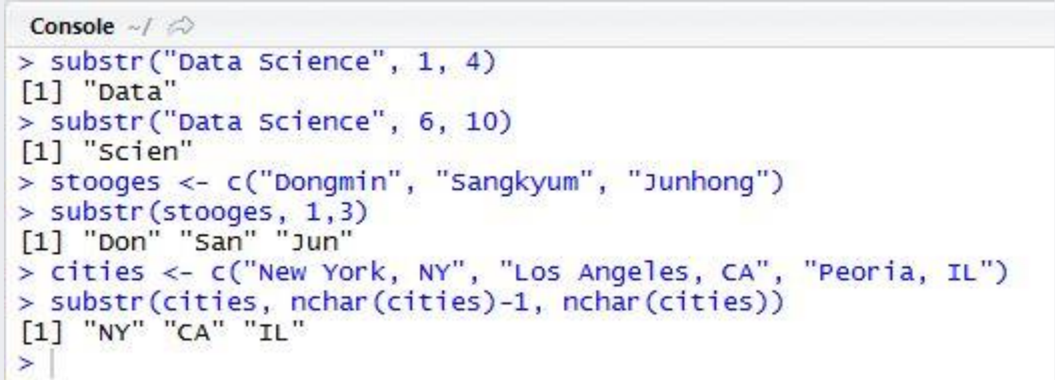
```
> S <- "Welcome to Data Science!"
> length(S)
[1] 1
> nchar(S)
[1] 24
> S1 <- "My name is"
> S2 <- "Pilsung Kang"
> paste(S1, S2)
[1] "My name is Pilsung Kang"
> paste(S1, S2, sep="-")
[1] "My name is-Pilsung Kang"
> paste(S1, S2, sep="")
[1] "My name isPilsung Kang"
>
```



```
> paste("The value of log10 is", log(10))
[1] "The value of log10 is 2.30258509299405"
> S1 <- c("My name is", "Your name is")
> S2 <- c("Pilsung")
> S3 <- c("Pilsung", "Younho", "Hakyeon")
> paste(S1,S2)
[1] "My name is Pilsung"   "Your name is Pilsung"
> paste(S1,S3)
[1] "My name is Pilsung"  "Your name is Younho" "My name is Hakyeon"
> stooges <- c("Dongmin", "Sangkyum", "Junhong")
> paste(stooges, "loves", "R.")
[1] "Dongmin loves R."  "Sangkyum loves R." "Junhong loves R."
> paste(stooges, "loves", "R", collapse = ", and ")
[1] "Dongmin loves R, and Sangkyum loves R, and Junhong loves R"
>
```

Text Processing

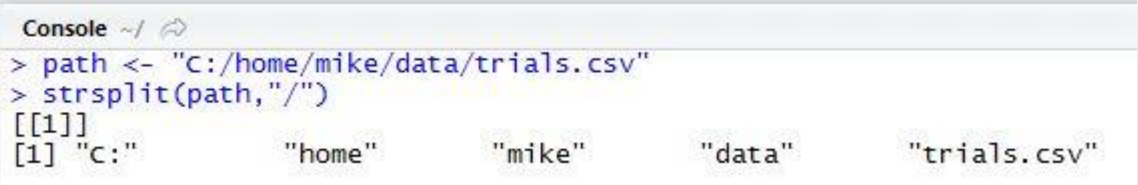
- Extract sub-strings
  - ✓ Use substring(string, start, end) function
    - Extract the substring that begins with “start” and ends with “end”
    - If the string argument is a vector, the other options are applied to all elements



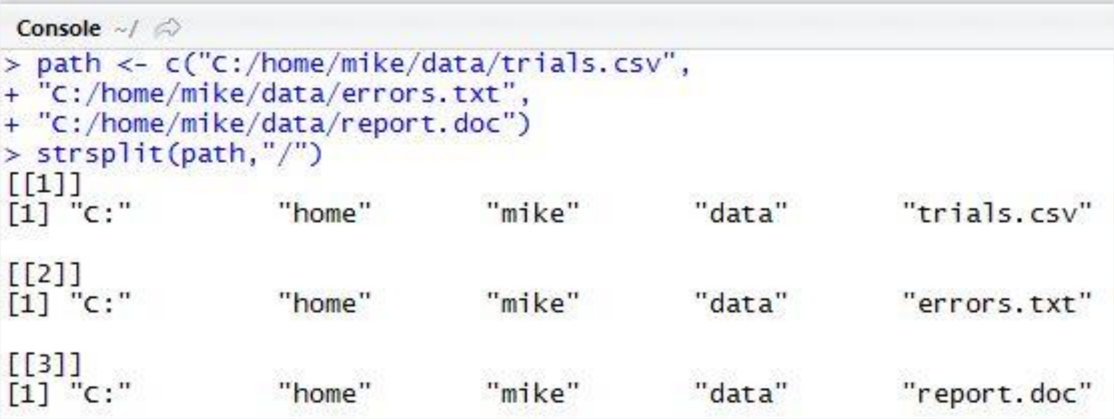
```
> substr("Data Science", 1, 4)
[1] "Data"
> substr("Data Science", 6, 10)
[1] "Scien"
> stooges <- c("Dongmin", "Sangkyum", "Junhong")
> substr(stooges, 1,3)
[1] "Don" "San" "Jun"
> cities <- c("New York, NY", "Los Angeles, CA", "Peoria, IL")
> substr(cities, nchar(cities)-1, nchar(cities))
[1] "NY" "CA" "IL"
> |
```

Text Processing

- Split text
  - ✓ Use strsplit(string, separator) function
    - A simple string or regular expression can be used as a separator
    - Ex: split the file path using “/” as a separator



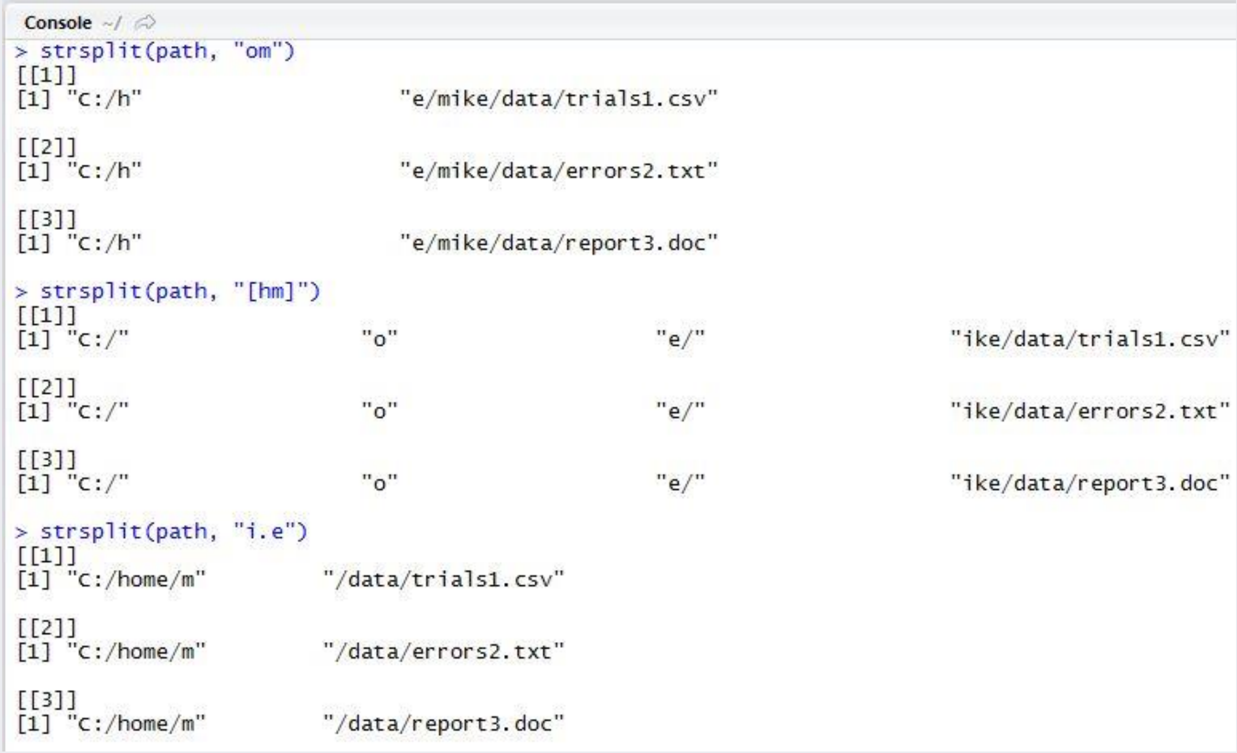
```
> path <- "C:/home/mike/data/trials.csv"
> strsplit(path,"/")
[[1]]
[1] "C:"          "home"        "mike"        "data"        "trials.csv"
```



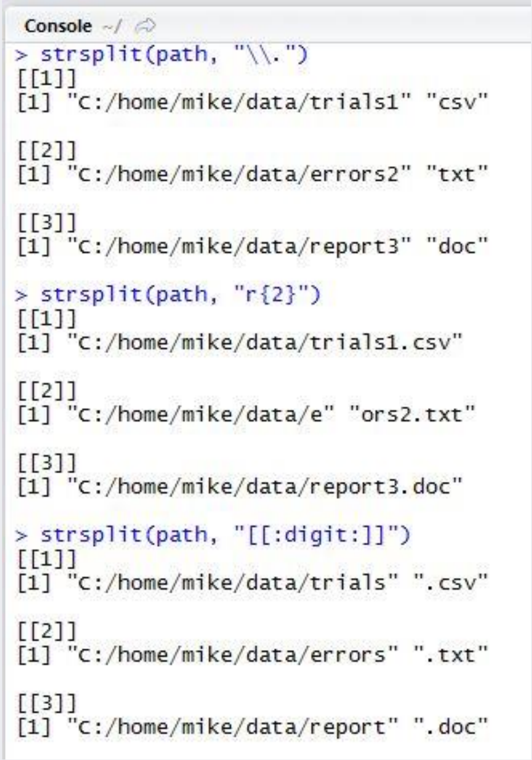
```
> path <- c("C:/home/mike/data/trials.csv",
+ "C:/home/mike/data/errors.txt",
+ "C:/home/mike/data/report.doc")
> strsplit(path,"/")
[[1]]
[1] "C:"          "home"        "mike"        "data"        "trials.csv"

[[2]]
[1] "C:"          "home"        "mike"        "data"        "errors.txt"

[[3]]
[1] "C:"          "home"        "mike"        "data"        "report.doc"
```

Text Processing

- Regular expression
  - ✓ a sequence of characters that define a search pattern
  - ✓ this pattern is then used by string searching algorithms



```
> strsplit(path, "om")
[[1]]
[1] "C:/h"                      "e/mike/data/trials1.csv"

[[2]]
[1] "C:/h"                      "e/mike/data/errors2.txt"

[[3]]
[1] "C:/h"                      "e/mike/data/report3.doc"

> strsplit(path, "[hm]")
[[1]]
[1] "C:/"                 "o"                   "e/"                  "ike/data/trials1.csv"

[[2]]
[1] "C:/"                 "o"                   "e/"                  "ike/data/errors2.txt"

[[3]]
[1] "C:/"                 "o"                   "e/"                  "ike/data/report3.doc"

> strsplit(path, "i.e")
[[1]]
[1] "C:/home/m"        "/data/trials1.csv"

[[2]]
[1] "C:/home/m"        "/data/errors2.txt"

[[3]]
[1] "C:/home/m"        "/data/report3.doc"
```

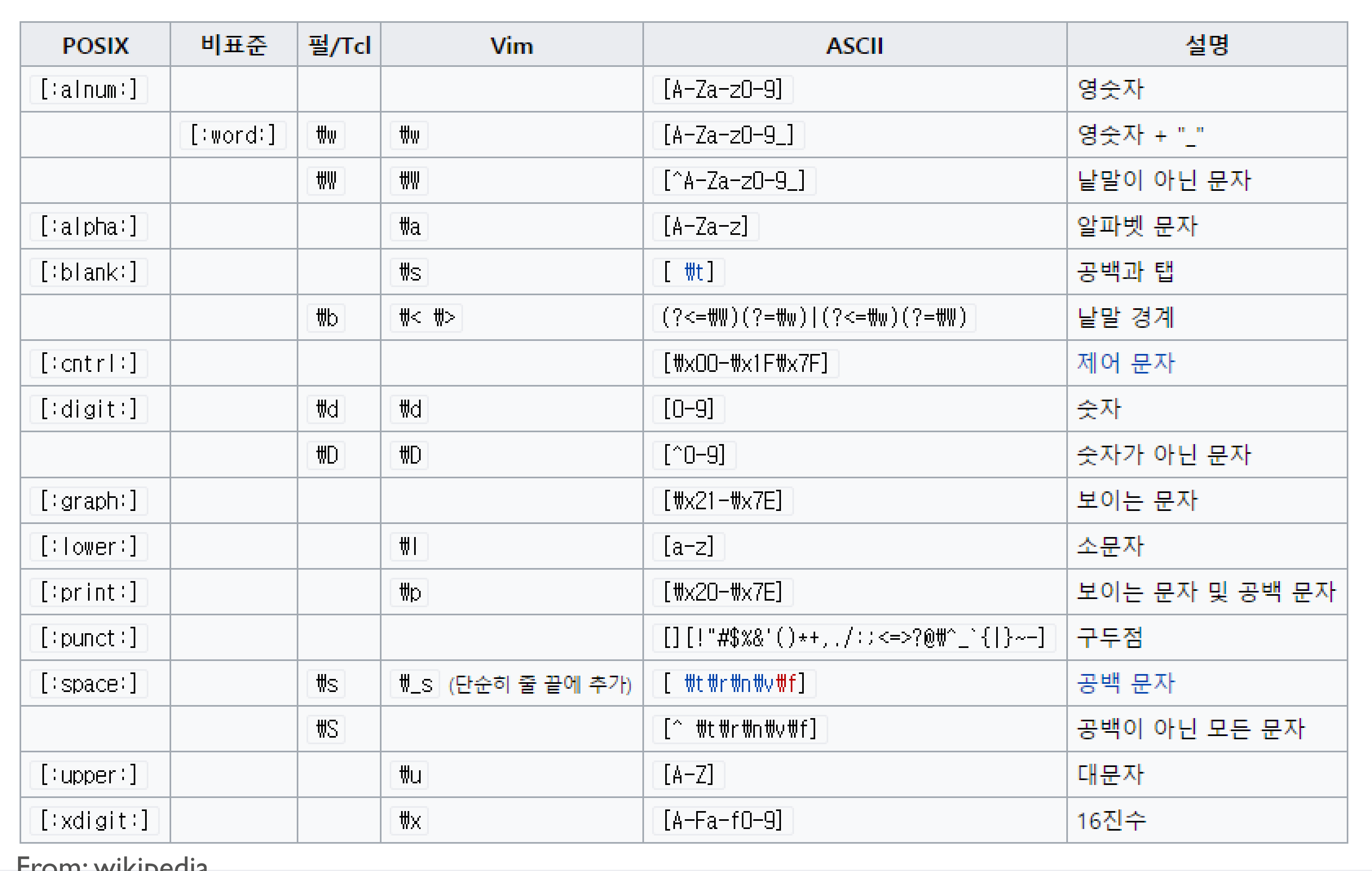


```
> strsplit(path, "\\.")
[[1]]
[1] "C:/home/mike/data/trials1" "csv"

[[2]]
[1] "C:/home/mike/data/errors2" "txt"

[[3]]
[1] "C:/home/mike/data/report3" "doc"

> strsplit(path, "r{2}")
[[1]]
[1] "C:/home/mike/data/trials1.csv"

[[2]]
[1] "C:/home/mike/data/e" "ors2.txt"

[[3]]
[1] "C:/home/mike/data/report3.doc"

> strsplit(path, "[[:digit:]]")
[[1]]
[1] "C:/home/mike/data/trials" ".csv"

[[2]]
[1] "C:/home/mike/data/errors" ".txt"

[[3]]
[1] "C:/home/mike/data/report" ".doc"
```

Regular expression in R

Text Processing

- Regular expression

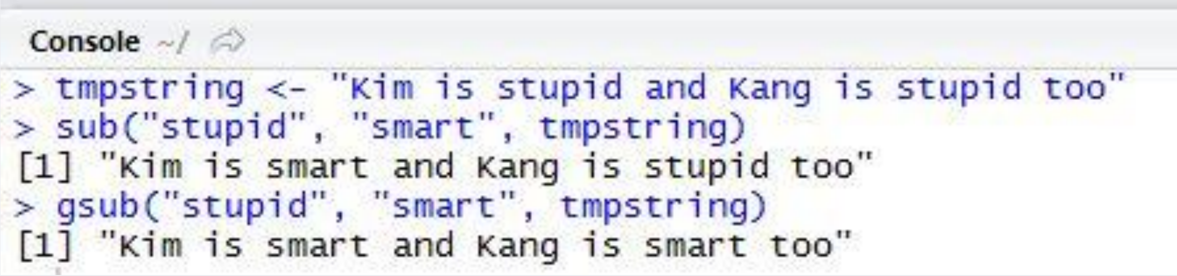


POSIX	비표준	펄/Tcl	Vim	ASCII	설명
[:alnum:]				[A-Za-z0-9]	영숫자
	[:word:]	₩w	₩w	[A-Za-z0-9_]	영숫자 + "_"
		₩W	₩W	[^A-Za-z0-9_]	낱말이 아닌 문자
[:alpha:]			₩a	[A-Za-z]	알파벳 문자
[:blank:]			₩s	[₩t]	공백과 탭
		₩b	₩< ₩>	(?<=₩W)(?=₩w)|(?<=₩w)(?=₩W)	낱말 경계
[:cntrl:]				[₩x00-₩x1F₩x7F]	제어 문자
[:digit:]		₩d	₩d	[0-9]	숫자
		₩D	₩D	[^0-9]	숫자가 아닌 문자
[:graph:]				[₩x21-₩x7E]	보이는 문자
[:lower:]			₩l	[a-z]	소문자
[:print:]			₩p	[₩x20-₩x7E]	보이는 문자 및 공백 문자
[:punct:]				[][!"#$%&'()*+,./:;<=>?@₩^_`{|}~-]	구두점
[:space:]		₩s	₩_s (단순히 줄 끝에 추가)	[₩t₩r₩n₩v₩f]	공백 문자
		₩S		[^ ₩t₩r₩n₩v₩f]	공백이 아닌 모든 문자
[:upper:]			₩u	[A-Z]	대문자
[:xdigit:]			₩x	[A-Fa-f0-9]	16진수

From: wikipedia

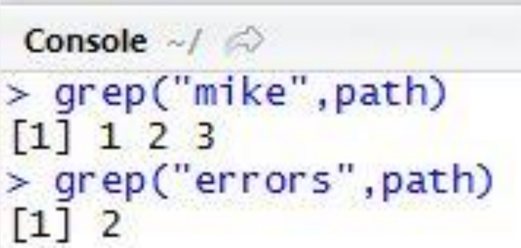
Text Processing

- Substitution
  - ✓ Use sub(old, new, string) or gsub(old, new, string) functions
  - ✓ sub( ) replaces the first substring whereas gsub( ) replaces all substrings



```
> tmpstring <- "Kim is stupid and Kang is stupid too"
> sub("stupid", "smart", tmpstring)
[1] "Kim is smart and Kang is stupid too"
> gsub("stupid", "smart", tmpstring)
[1] "Kim is smart and Kang is smart too"
```

- String pattern matching
  - ✓ Use grep(pattern, x) function
    - Return the index that matches pattern

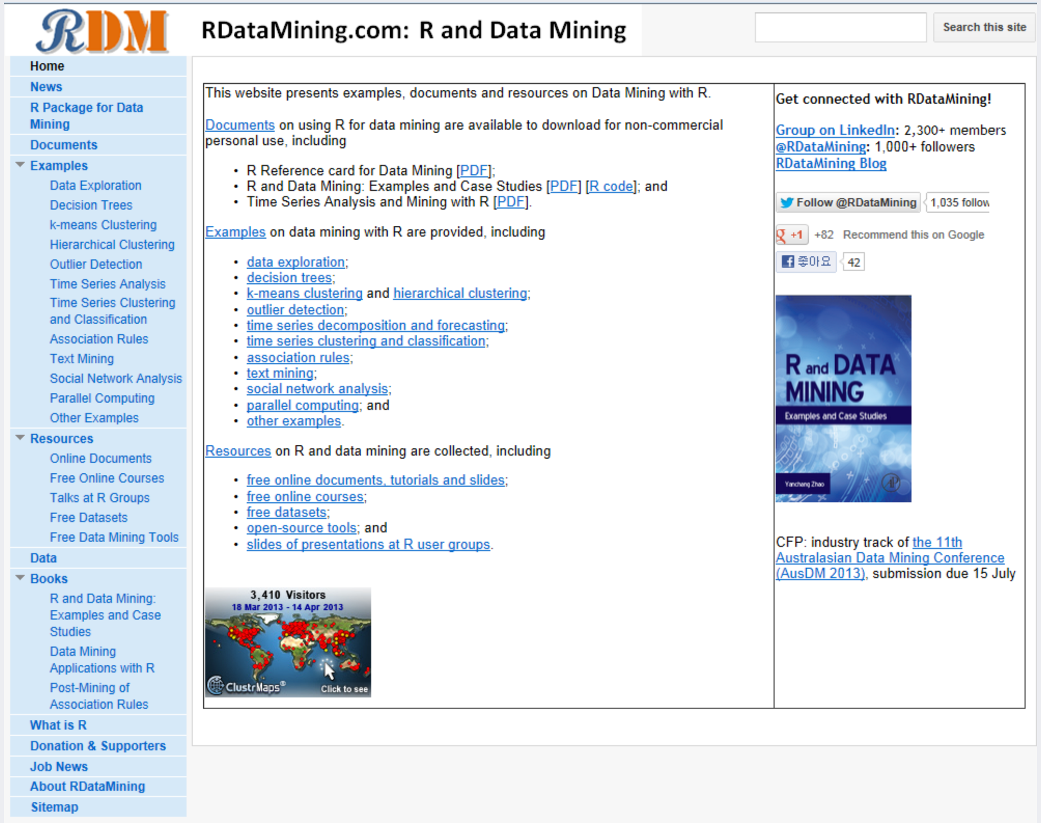


```
> grep("mike",path)
[1] 1 2 3
> grep("errors",path)
[1] 2
```

ANY
questions?

References

- R Datamining
  - ✓ http://www.rdatamining.com



RDataMining.com: R and Data Mining

- Home
- News
- R Package for Data Mining
- Documents
- Examples
  - Data Exploration
  - Decision Trees
  - k-means Clustering
  - Hierarchical Clustering
  - Outlier Detection
  - Time Series Analysis
  - Time Series Clustering and Classification
  - Association Rules
  - Text Mining
  - Social Network Analysis
  - Parallel Computing
  - Other Examples
- Resources
  - Online Documents
  - Free Online Courses
  - Talks at R Groups
  - Free Datasets
  - Free Data Mining Tools
- Data
- Books
  - R and Data Mining: Examples and Case Studies
  - Data Mining Applications with R
  - Post-Mining of Association Rules
- What is R
- Donation & Supporters
- Job News
- About RDataMining
- Sitemap

This website presents examples, documents and resources on Data Mining with R.

Documents on using R for data mining are available to download for non-commercial personal use, including

- R Reference card for Data Mining [PDF];
- R and Data Mining: Examples and Case Studies [PDF] [R code]; and
- Time Series Analysis and Mining with R [PDF].

Examples on data mining with R are provided, including

- data exploration;
- decision trees;
- k-means clustering and hierarchical clustering;
- outlier detection;
- time series decomposition and forecasting;
- time series clustering and classification;
- association rules;
- text mining;
- social network analysis;
- parallel computing; and
- other examples.

Resources on R and data mining are collected, including

- free online documents, tutorials and slides;
- free online courses;
- free datasets;
- open-source tools; and
- slides of presentations at R user groups.

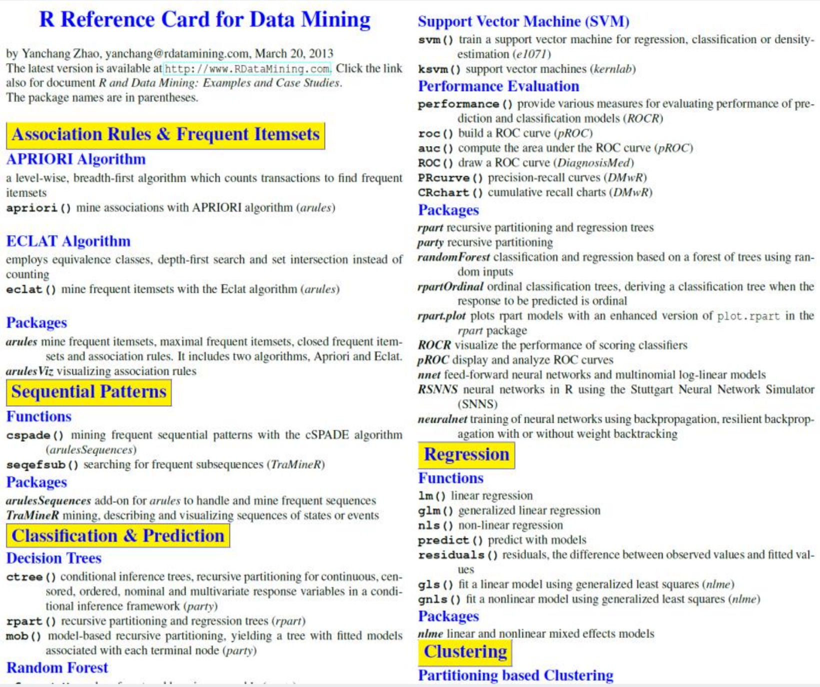
3,410 Visitors
18 Mar 2013 - 14 Apr 2013

Get connected with RDataMining!

Group on LinkedIn: 2,300+ members
@RDataMining: 1,000+ followers
RDataMining Blog

+82

CFP: industry track of the 11th Australasian Data Mining Conference (AusDM 2013), submission due 15 July



R Reference Card for Data Mining

by Yanchang Zhao, yanchang@rdatamining.com, March 20, 2013
The latest version is available at http://www.RDataMining.com. Click the link also for document *R and Data Mining: Examples and Case Studies*.
The package names are in parentheses.

Association Rules & Frequent Itemsets

APRIORI Algorithm

a level-wise, breadth-first algorithm which counts transactions to find frequent itemsets

apriori() mine associations with APRIORI algorithm (*arules*)

ECLAT Algorithm

employs equivalence classes, depth-first search and set intersection instead of counting

eclat() mine frequent itemsets with the Eclat algorithm (*arules*)

Packages

arules mine frequent itemsets, maximal frequent itemsets, closed frequent itemsets and association rules. It includes two algorithms, Apriori and Eclat.

arulesViz visualizing association rules

Sequential Patterns

Functions

cspade() mining frequent sequential patterns with the cSPADE algorithm (*arulesSequences*)

seqefsub() searching for frequent subsequences (*TraMineR*)

Packages

arulesSequences add-on for *arules* to handle and mine frequent sequences

TraMineR mining, describing and visualizing sequences of states or events

Classification & Prediction

Decision Trees

ctree() conditional inference trees, recursive partitioning for continuous, censored, ordered, nominal and multivariate response variables in a conditional inference framework (*party*)

rpart() recursive partitioning and regression trees (*rpart*)

mob() model-based recursive partitioning, yielding a tree with fitted models associated with each terminal node (*party*)

Random Forest

Support Vector Machine (SVM)

svm() train a support vector machine for regression, classification or density-estimation (*e1071*)

ksvm() support vector machines (*kernlab*)

Performance Evaluation

performance() provide various measures for evaluating performance of prediction and classification models (*ROCR*)

roc() build a ROC curve (*pROC*)

auc() compute the area under the ROC curve (*pROC*)

ROC() draw a ROC curve (*DiagnosisMed*)

PRcurve() precision-recall curves (*DMwR*)

CRchart() cumulative recall charts (*DMwR*)

Packages

rpart recursive partitioning and regression trees

party recursive partitioning

randomForest classification and regression based on a forest of trees using random inputs

rpartOrdinal ordinal classification trees, deriving a classification tree when the response to be predicted is ordinal

rpart.plot plots rpart models with an enhanced version of `plot.rpart` in the *rpart* package

ROCR visualize the performance of scoring classifiers

pROC display and analyze ROC curves

nnet feed-forward neural networks and multinomial log-linear models

RSNNS neural networks in R using the Stuttgart Neural Network Simulator (SNNS)

neuralnet training of neural networks using backpropagation, resilient backpropagation with or without weight backtracking

Regression

Functions

lm() linear regression

glm() generalized linear regression

nls() non-linear regression

predict() predict with models

residuals() residuals, the difference between observed values and fitted values

gls() fit a linear model using generalized least squares (*nlme*)

gnls() fit a nonlinear model using generalized least squares (*nlme*)

Packages

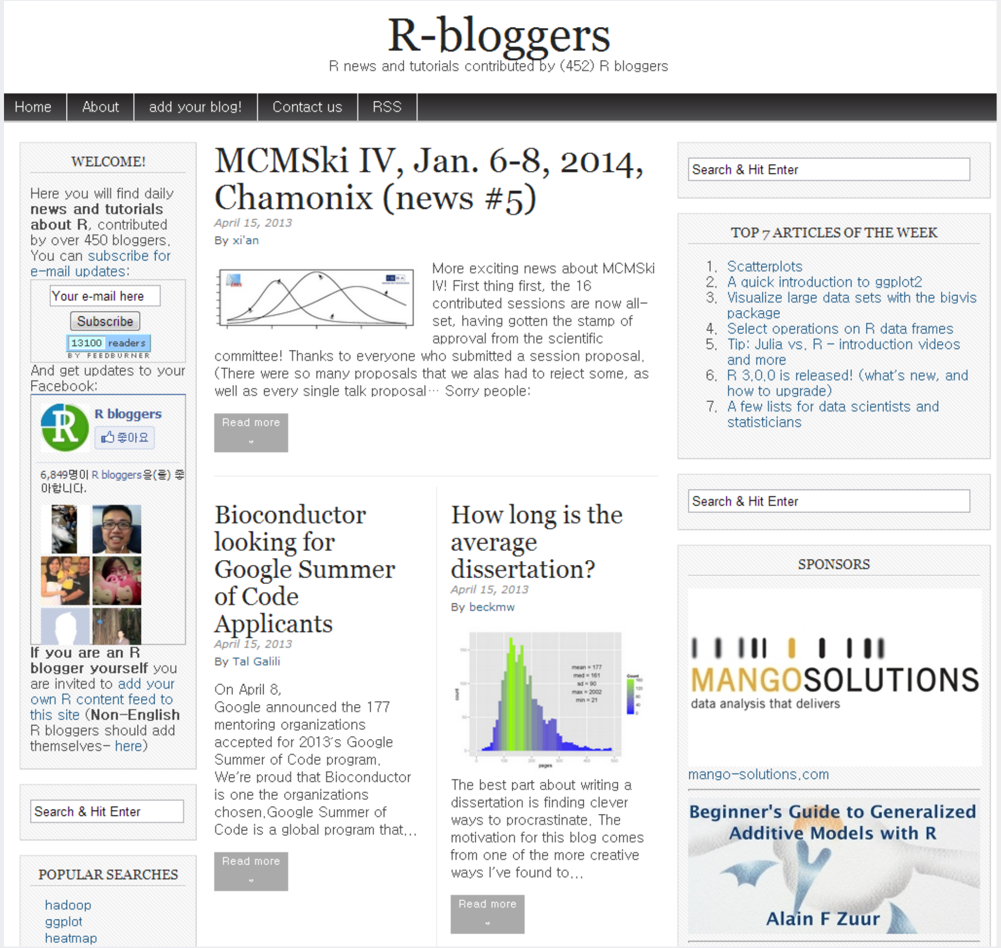
nlme linear and nonlinear mixed effects models

Clustering

Partitioning based Clustering

References

- R Bloggers
  - ✓ http://r-bloggers.com



R-bloggers

R news and tutorials contributed by (452) R bloggers

WELCOME!

Here you will find daily **news and tutorials about R**, contributed by over 450 bloggers. You can subscribe for e-mail updates:

And get updates to your Facebook:

6,849명이 R bloggers을(를) 좋아합니다.

If you are an R blogger yourself you are invited to add your own R content feed to this site (**Non-English** R bloggers should add themselves- here)

POPULAR SEARCHES

hadoop
ggplot
heatmap

MCMSki IV, Jan. 6-8, 2014, Chamonix (news #5)

April 15, 2013

By xi'an

More exciting news about MCMSki IV! First thing first, the 16 contributed sessions are now all-set, having gotten the stamp of approval from the scientific committee! Thanks to everyone who submitted a session proposal. (There were so many proposals that we alas had to reject some, as well as every single talk proposal… Sorry people:

Bioconductor looking for Google Summer of Code Applicants

April 15, 2013

By Tal Galili

On April 8,
Google announced the 177 mentoring organizations accepted for 2013's Google Summer of Code program. We're proud that Bioconductor is one the organizations chosen.Google Summer of Code is a global program that...

How long is the average dissertation?

April 15, 2013

By beckmw

The best part about writing a dissertation is finding clever ways to procrastinate. The motivation for this blog comes from one of the more creative ways I've found to...

TOP 7 ARTICLES OF THE WEEK

1. Scatterplots
2. A quick introduction to ggplot2
3. Visualize large data sets with the bigvis package
4. Select operations on R data frames
5. Tip: Julia vs. R – introduction videos and more
6. R 3.0.0 is released! (what's new, and how to upgrade)
7. A few lists for data scientists and statisticians

SPONSORS

MANGOSOLUTIONS
data analysis that delivers
mango-solutions.com
Beginner's Guide to Generalized Additive Models with R
Alain F Zuur