



Introduction

@somkiat



Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info View Activity Log 10+ ...

Timeline About Friends 2,941 Photos More ▾

When did you attend Ubon Ratchathani University? X

... 14 Pending Items

Intro

Software Craftsmanship

- Software Practitioner at สยามช่างนาญกิจ พ.ศ. 2556
- Agile Practitioner and Technical at SPRINT3r
- Software analyst at TARAD.com
- Software Developer at True Corporation
- Former Software Engineer at Opendream

Status Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 1 hr · Twitter · ▼

พรุ่งนี้ไป share เรื่อง NoSQL ที่มหาวิทยาลัย

Like Comment Share

Nuttachot Dusitanont, Chitpong Few Wuttanan and 50 others



Page Messages Notifications Insights Publishing Tools Settings Help ▾



somkiat.cc

@somkiat.cc

Home

About

Events

Photos

Likes

Videos

Posts

Services



Write something...

...



Personal Blog

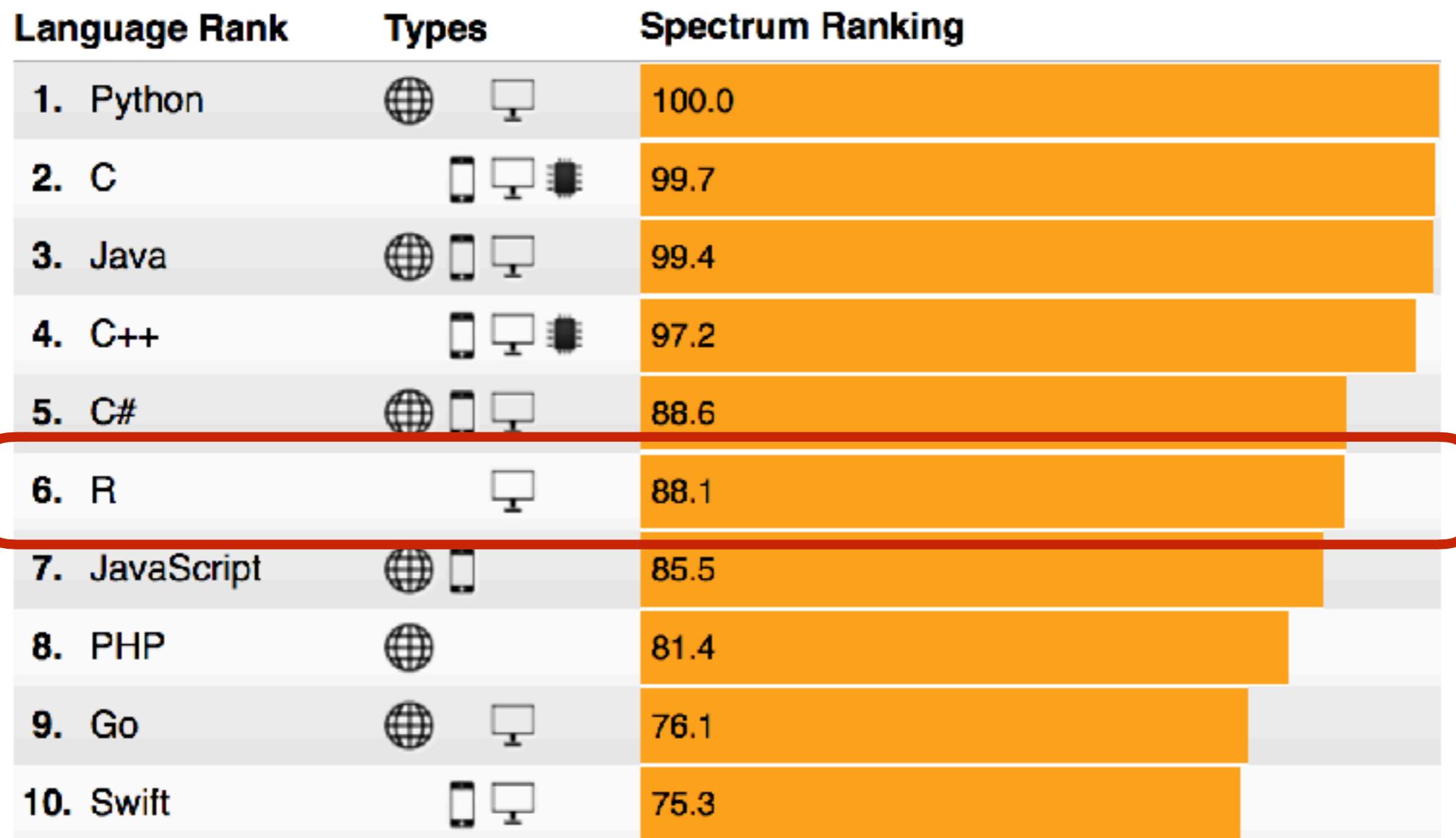
Page Tips

See All

What's a Boosted Post?

A boosted post is the easiest way to reach more people on Facebook.

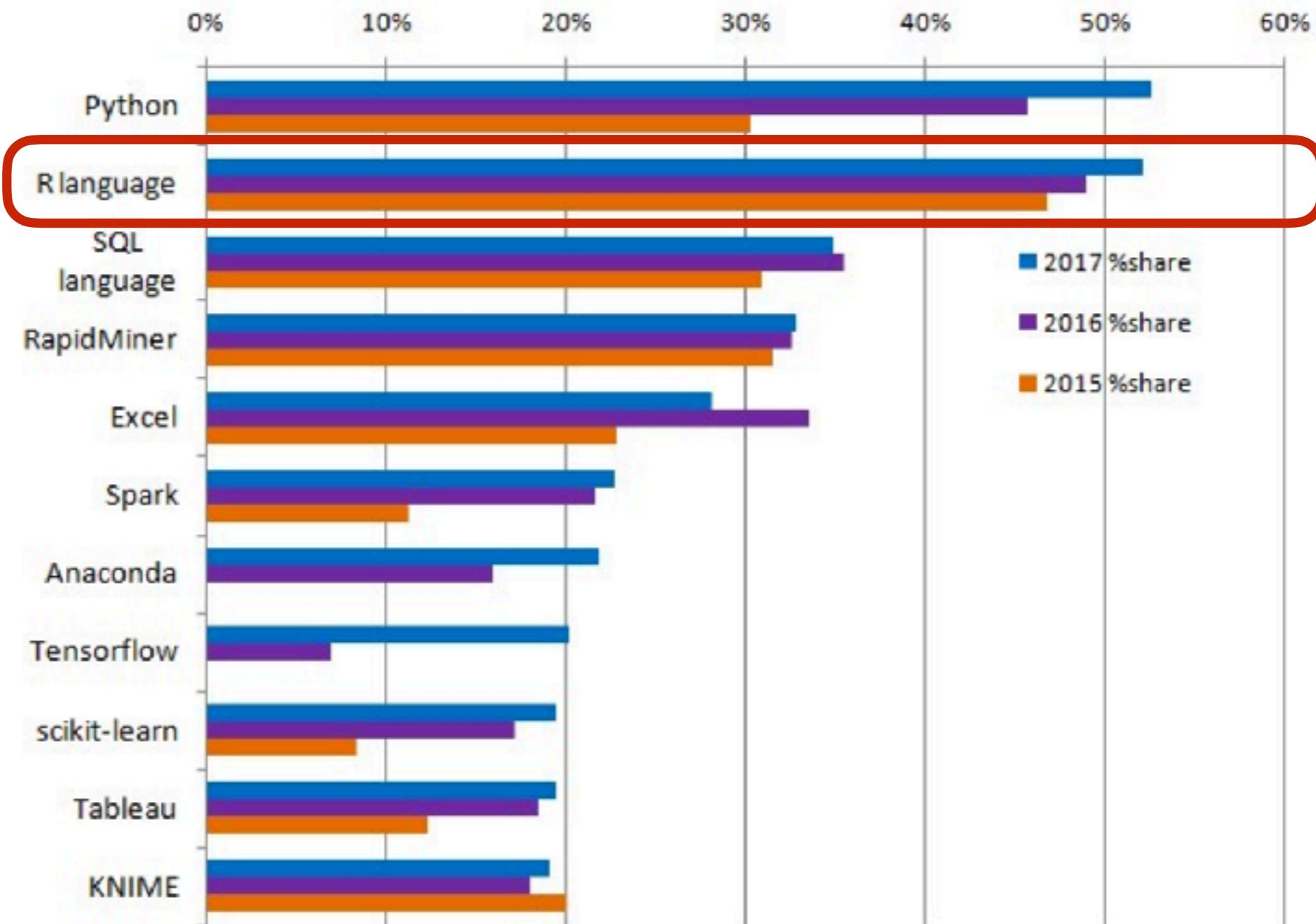
Top programming 2017



<http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>



KDnuggets Analytics, Data Science, Machine Learning Software Poll, top tools share, 2015-2017



<http://www.kdnuggets.com/2017/05/poll-analytics-data-science-machine-learning-software-leaders.html>



Try R

<http://tryr.codeschool.com/>



บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่



CHAPTER 1

Try R

In this first chapter, we'll cover basic R expressions. We'll start simple, with numbers, strings, and true/false values. Then we'll show you how to store those values in variables, and how to pass them to functions. We'll show you how to get help on functions when you're stuck. Finally we'll load an R script in from a file.

Let's get started!

<http://tryr.codeschool.com/>

Try R is Sponsored By:

O'REILLY®



Complete to
Unlock



บริษัท สยามชำนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

R programming

developed from S language
Public on 1993



Getting start with R



R programming tools

R runtime

R studio

R package



R runtime



[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 variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your mirror.

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

News

- [The R Journal Volume 9/1](#) is available.
- [R version 3.4.1 \(Single Candle\)](#) has been released on Friday 2017-06-30.
- [R version 3.3.3 \(Another Canoe\)](#) has been released on Monday 2017-03-06.

<https://www.r-project.org/>



R studio

The screenshot shows the official RStudio website. At the top is a blue navigation bar with the R Studio logo, a search icon, and links for rstudio::conf, Products, Resources, Pricing, About Us, and Blogs. Below the bar is a large banner featuring the RStudio logo and the text "Open source and enterprise-ready professional software for R". To the right of the banner are four call-to-action buttons: "Download RStudio", "Discover Shiny", "shinyapps.io Login", and "Discover RStudio Connect". Below the banner are three sections: "RStudio" showing the IDE interface, "Shiny" showing a dashboard with a map and chart, and "R Packages" showing icons for markdown, shiny, tidyverse, knitr, and ggplot2.

R Studio

rstudio::conf Products Resources Pricing About Us Blogs

RStudio

Open source and enterprise-ready professional software for R

Download RStudio

Discover Shiny

shinyapps.io Login

Discover RStudio Connect

RStudio

Shiny

R Packages

<https://www.rstudio.com/>



R package

Set of functions that perform analysis
on some kind of data



R package

CRAN (Comprehensive R Archive Network)

Available CRAN Packages By Name

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

[A3](#)

Accurate, Adaptable, and Accessible Error Metrics for Predictive Models

[abbyyR](#)

Access to Abbyy Optical Character Recognition (OCR) API

[abc](#)

Tools for Approximate Bayesian Computation (ABC)

[ABCanalysis](#)

Computed ABC Analysis

[abc.data](#)

Data Only: Tools for Approximate Bayesian Computation (ABC)

[abcdeFBA](#)

ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package

[ABCOptim](#)

Implementation of Artificial Bee Colony (ABC) Optimization

[ABCp2](#)

Approximate Bayesian Computational Model for Estimating P2

[ABC.RAP](#)

Array Based CpG Region Analysis Pipeline

[abcrf](#)

Approximate Bayesian Computation via Random Forests

[abctools](#)

Tools for ABC Analyses

[abd](#)

The Analysis of Biological Data

[abf2](#)

Load Gap-Free Axon ABF2 Files

[ABHgenotypeR](#)

Easy Visualization of ABH Genotypes

<https://cran.r-project.org>



R environment



R in command line

\$R --version

```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under the terms of the
GNU General Public License versions 2 or 3.
For more information about these matters see
http://www.gnu.org/licenses/.
```



R in command line

\$R

```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

```
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.
```

Natural language support but running in an English locale

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

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

> █



R Script

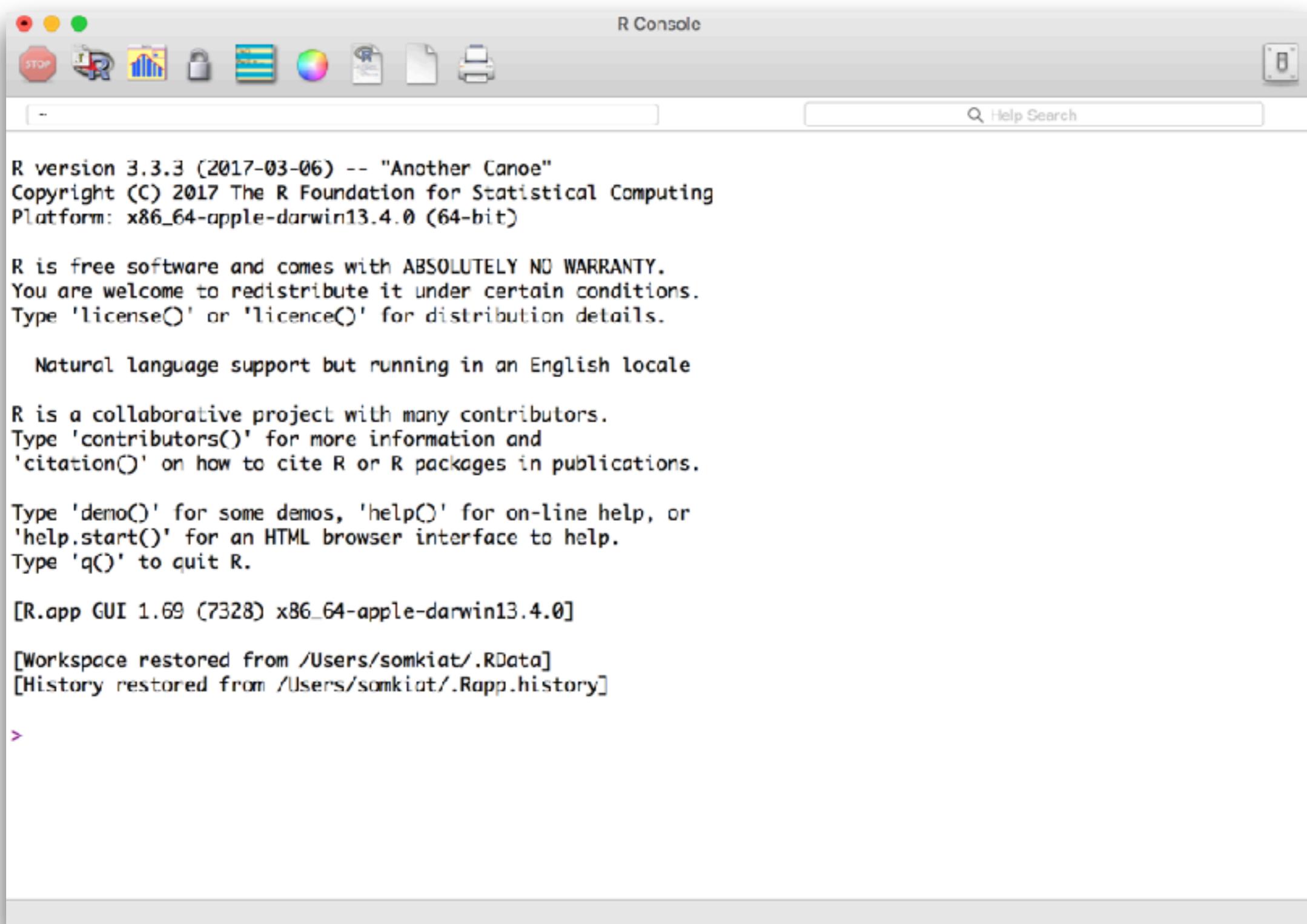
\$Rscript hello.R

```
sayHello <- function(name){  
  cat("Hello ", name)  
}
```

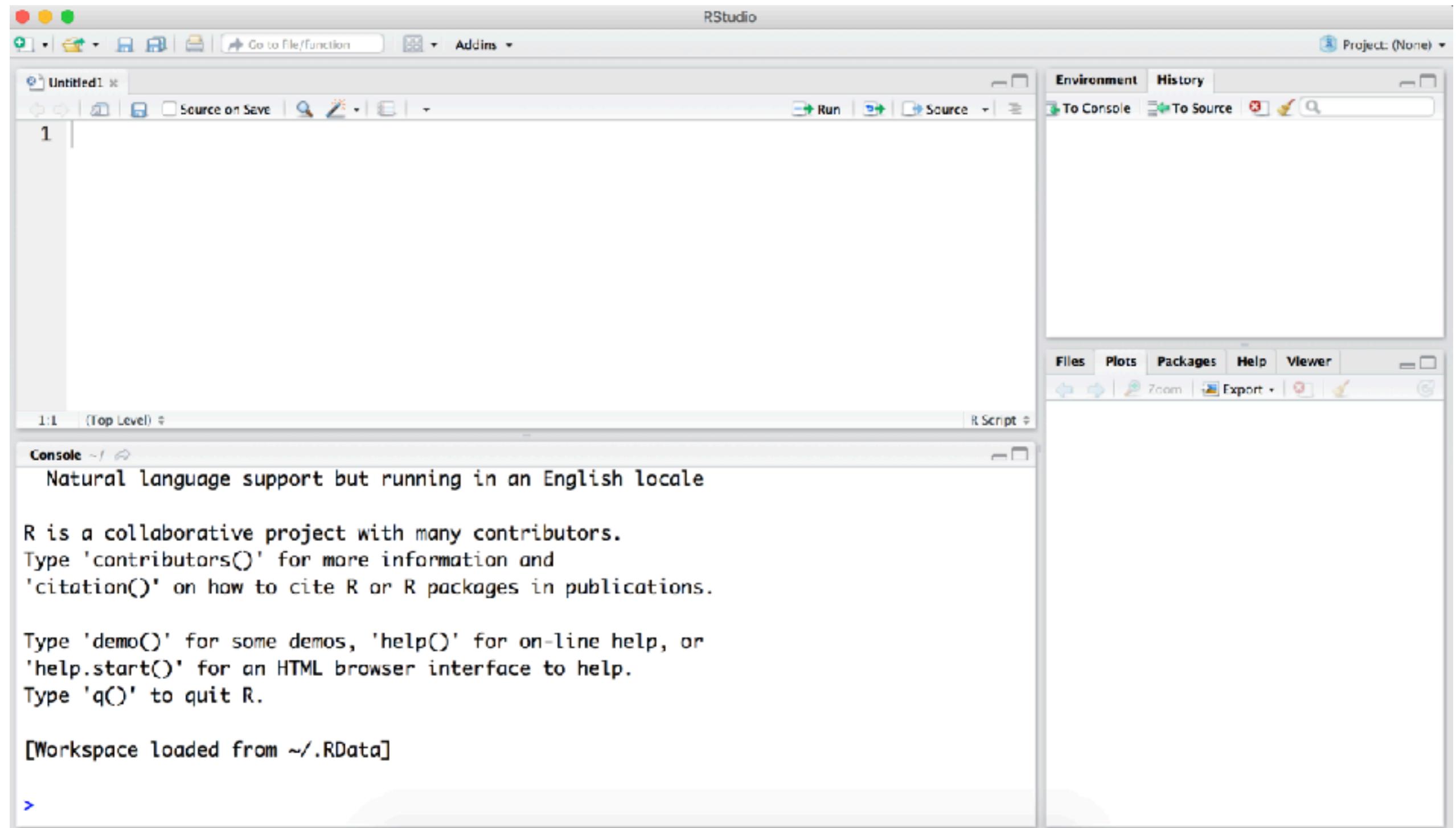
```
sayHello("R")
```



R console

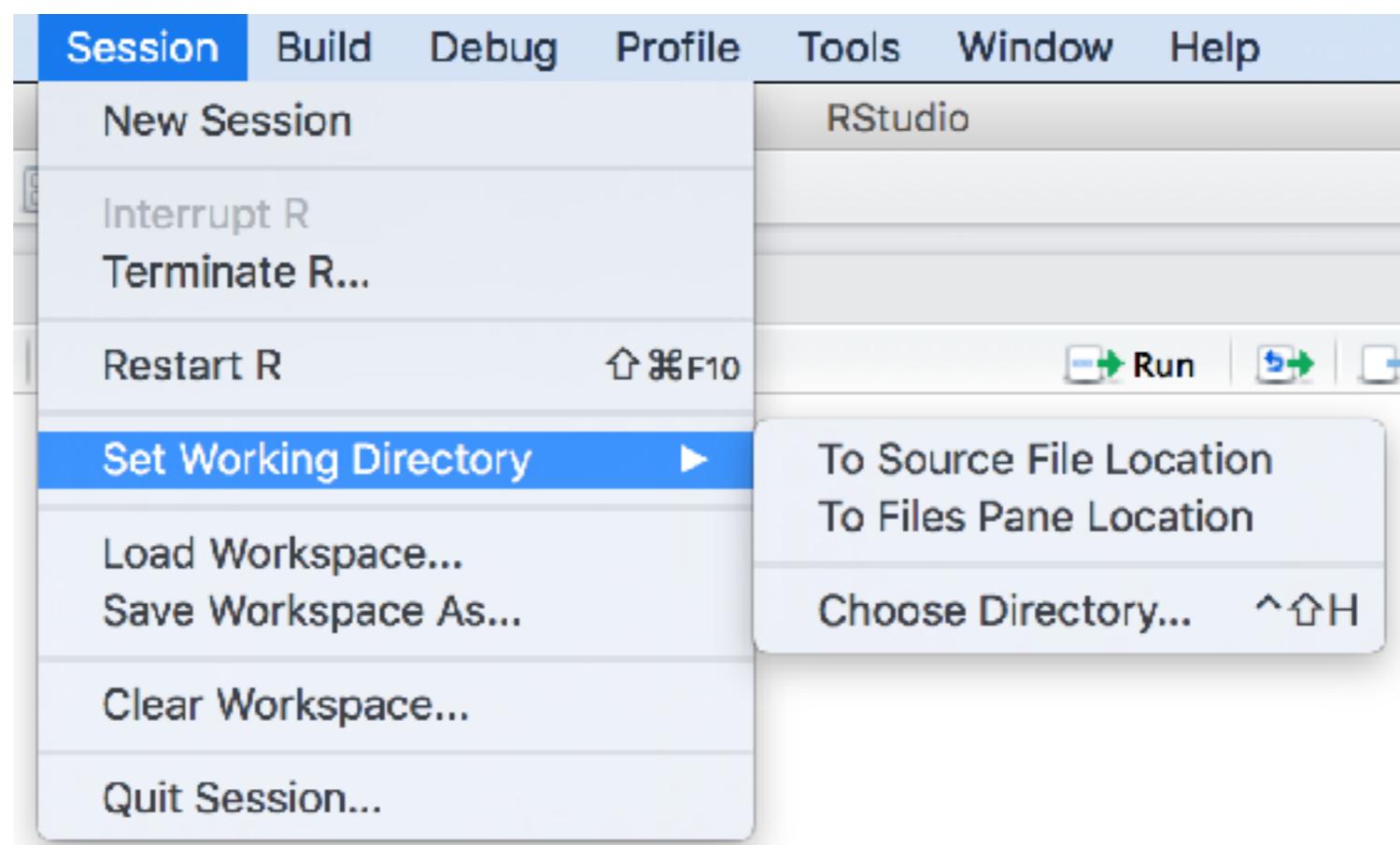


R studio



R studio

Setting workspace of project



Workspace

Current R working environment
is a directory

Function	Action
getwd()	List the current working directory
setwd("your directory")	Change the current working directory
ls()	List objects in the current workspace
rm(objectlist)	Remove/delete one or more objects
history(#)	Display your last command (default = 25)



Workspace

Function	Action
savehistory("filename")	Save the command history to file (default = .Rhistory)
loadhistory("filename")	Reload a command history (default = .Rhistory)
save.image("filename")	Save the workspace to file (default = .RData)
save(objectlist, file="filename")	Save objects to file
load("filename")	Load object into current session
q()	Quit R



Working with workspace

```
setwd("your directory")
options()
options(digits=3)
x <- runif(20)
summary(x)
hist(x)
q()
```



R package



12,583 packages

Available Packages

Currently, the CRAN package repository features 12583 available packages.

[Table of available packages, sorted by date of publication](#)

[Table of available packages, sorted by name](#)

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information. The R base sources) explains the process in detail.

<https://cran.r-project.org/web/packages/>



Installing package

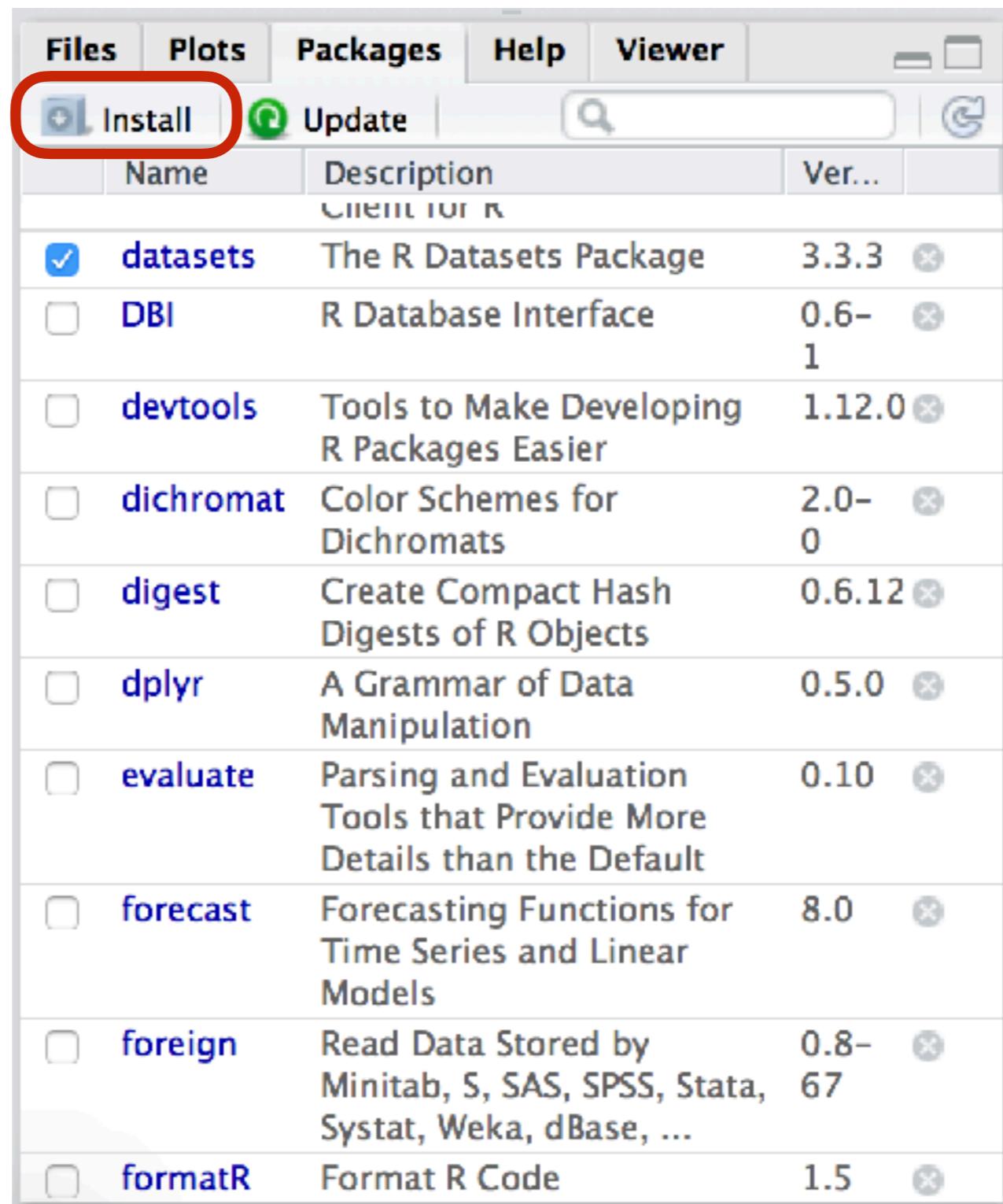
R studio

R console

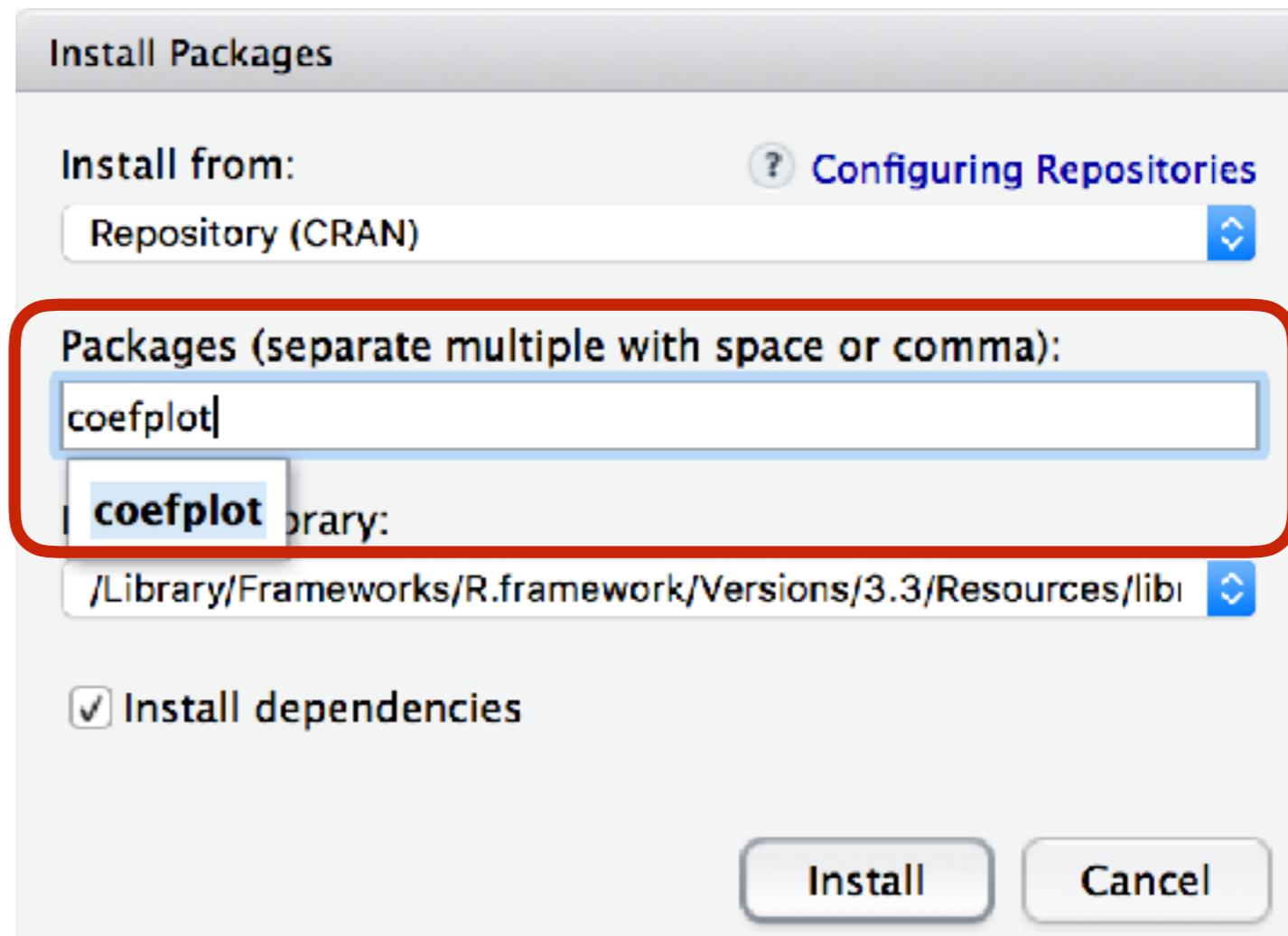
R command line



Install package on R studio



Install package on R studio



Install package on R console

```
>install.packages("coefplot")  
>install.packages("coefplot",  
                  dependencies = TRUE)
```



Uninstall package

```
>remove.packages("coefplot")
```



Loading package

```
>library("coefplot")
```



Unloading package

```
>detach("package:coefplot")
```



Clear the console in R Studio

```
>cat("\014")  
>cat("\f")
```

Press Control + K



see demo in R



List of all Demo

>demo()

Demos in package 'base':

error.catching	More examples on catching and handling errors
is.things	Explore some properties of R objects and is.FOO() functions. Not for newbies!
recursion	Using recursion for adaptive integration
scoping	An illustration of lexical scoping.

Demos in package 'graphics':

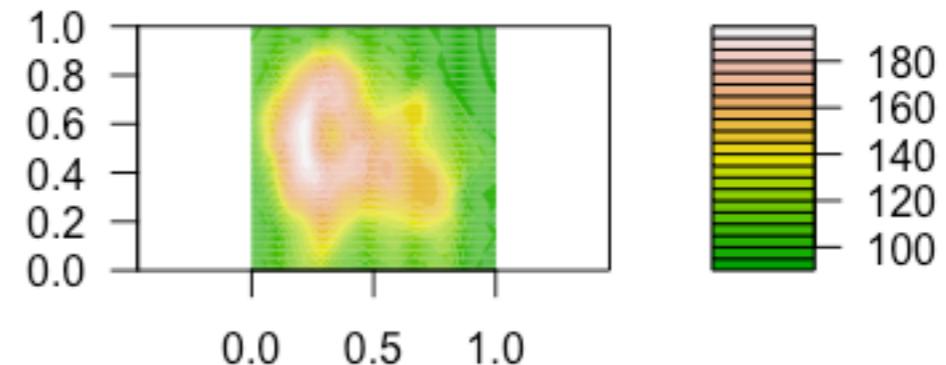
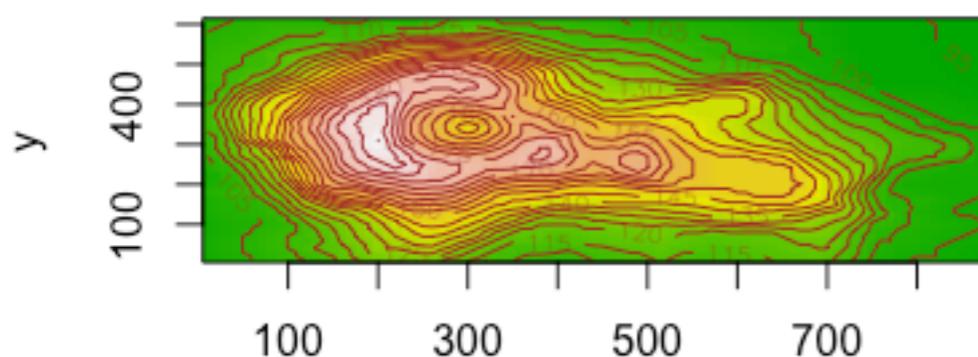
Hershey	Tables of the characters in the Hershey vector fonts
Japanese	Tables of the Japanese characters in the Hershey vector fonts
graphics	A show of some of R's graphics capabilities
image	The image-like graphics builtins of R
persp	Extended persp() examples
plotmath	Examples of the use of mathematics annotation



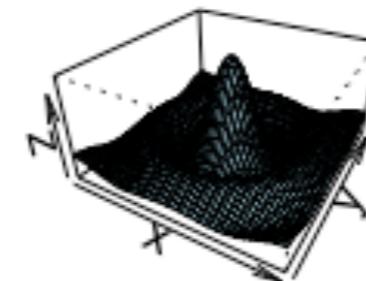
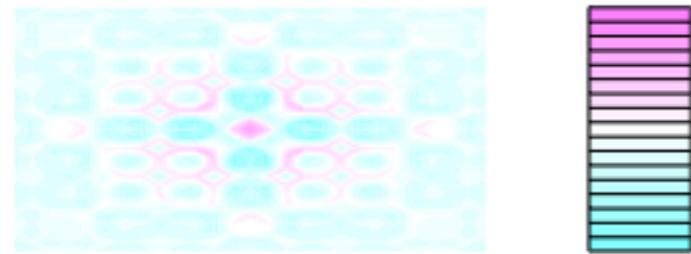
Image demo

>demo(image)

Maunga Whau Volcano



$$z = \text{Sinc}(\sqrt{x^2 + y^2})$$



Help function

>help.start()

The screenshot shows the R Help browser interface. The title bar includes 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. Below the title bar are standard operating system icons for file operations. A search bar is located at the top right. The main content area features the R logo and the text 'Statistical Data Analysis'. Below this, there are sections for 'Manuals' and 'Reference'. The 'Manuals' section contains links to 'An Introduction to R', 'Writing R Extensions', and 'R Data Import/Export'. The 'Reference' section contains links to 'The R Language Definition', 'R Installation and Administration', and 'R Internals'. At the bottom, there are sections for 'Miscellaneous Material' with links to 'About R', 'License', 'NEWS', 'Authors', 'Frequently Asked Questions', 'User Manuals', 'Resources', 'Thanks', and 'Technical papers'.

Files Plots Packages Help Viewer

The R Language Find in Topic

Statistical Data Analysis R

Manuals

[An Introduction to R](#)
[Writing R Extensions](#)
[R Data Import/Export](#)

[The R Language Definition](#)
[R Installation and Administration](#)
[R Internals](#)

Reference

[Packages](#) [Search Engine & Keywords](#)

Miscellaneous Material

[About R](#) [Authors](#) [Resources](#)
[License](#) [Frequently Asked Questions](#) [Thanks](#)
[NEWS](#) [User Manuals](#) [Technical papers](#)



Help function

Function	Action
help("foo") or ?foo	Help of function foo
help.search("foo") or ??foo	Search the help function for foo
example("foo")	See example usage of foo
data()	List all example datasets of loaded packages



Save to file

Function	Output
<code>bmp("filename")</code>	BMP file
<code>jpeg("filename")</code>	JPEG file
<code>pdf("filename")</code>	PDF file
<code>png("filename")</code>	PNG file
<code>svg("filename")</code>	SVG file
<code>postscript("filename")</code>	PostScript file

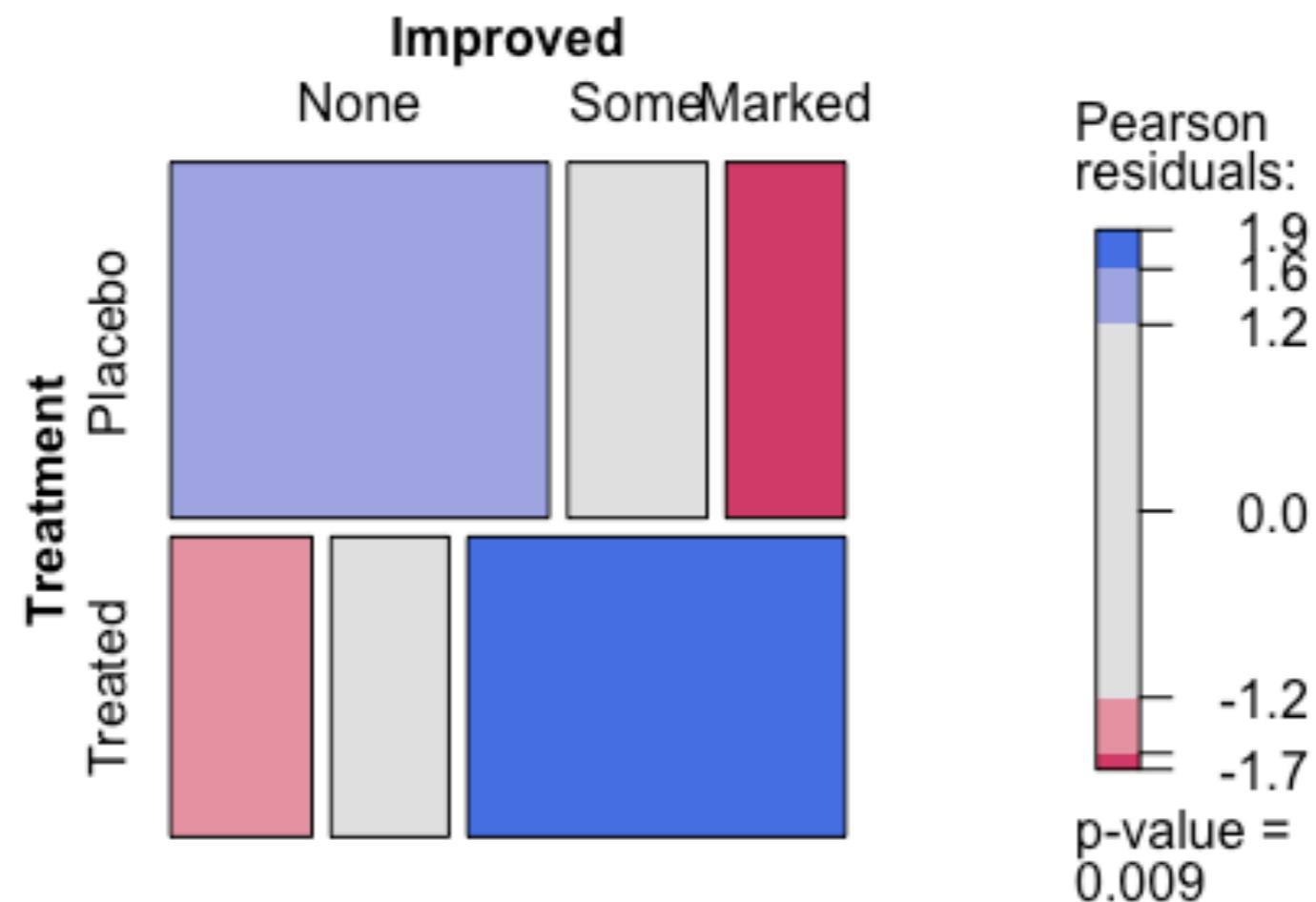


Try to write code

```
help.start()  
install.packages("vcd")  
help(package="vcd")  
library(vcd)  
help(Arthritis)  
Arthritis  
example(Arthritis)  
q()
```



Try to write code



R Documentation

RDocumentation

R Enterprise Training

R package

Leaderboard

Sign in

Search all 15,180 CRAN, BioConductor and Github packages.

Search for packages, functions, etc

Search

Or explore packages in one of the [Task Views](#).

Top 5 packages

1. viridisLite
2. readxl
3. R6
4. ggplot2
5. dplyr

Top 5 authors

1. Hadley Wickham
2. Kirill Müller
3. Yihui Xie
4. Gabor Csardi
5. Dirk Eddelbuettel

Newest packages

1. BBI
2. scopr
3. mmabig
4. npsr
5. multDM

<https://www.rdocumentation.org/>



บริษัท สยามซัมนาณุกิจ จำกัด และเพื่อนพ้องน้องพี่

Working with Large Dataset



R holds objects in memory

Size of dataset ?

Statistical methods that will apply ?

Efficient Programming ?



R in BigData

Package	Description
bigmemory	Support the creation/storage/access/multiplication of massive metrics
ff	Provide data structures that store in disk
filehash	Simple key-value database
ROracle, RMySQL, RPostgreSQL	Access to RDBMS



R in BigData



<https://spark.apache.org/docs/latest/sparkr.html>



R in BigData

R Server for Azure HDInsight



Familiarity of R algorithms



Scalability of Hadoop + Spark



More Accurate Predictions

<https://azure.microsoft.com/en-us/services/hdinsight/r-server/>



บริษัท สยามชานาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Basic of R



Basic of R

Calculation
Data manipulation
Scientific computation



Variable assignment

```
first = 1
second <- 2
assign("third", 3)
4 -> fourth

cat(first, second, third, fourth)

#Remove variable
rm(second)
```



Math in R

2 + 2

2 - 2

2 * 2

2 %% 2

2 %/% 2

2 / 2

2 ** 2

2 ** .5

2 ** -1



Few more complicate !!

`abs(-2)`

`pi`

`round(pi,digits = 2)`

`sign(-2)`

`log(2)`

`log10(2)`

`cos(pi)`



Logical table and testing

TRUE & TRUE

TRUE | FALSE

xor(TRUE, FALSE)

! FALSE

1 & 1

1 & 0

! 0



Data types

Numeric
Character (string)
Date (time-based)
Logical (TRUE/FALSE)



Data types

```
class(FALSE)
```

```
class(pi)
```

```
class("Look at me")
```

```
class(as.Date("2018-08-09"))
```

```
class(factor(c('undergraduate', 'graduate')))
```



Numeric

By default, R store everything as double
64-bit floating point

```
#Force to integer  
number <- 5L  
  
class(number)  
is.integer(number)  
is.numeric(number)
```



Boolean

Logical values act like numeric

TRUE + TRUE

2 & 1

TRUE * TRUE

2 & -1



Boolean

Try to use in if-else statement !!

```
if (2017)
    print('This is evaluated as a boolean')
```



Character

Like character handling in UNIX !!

```
my.char <- paste("Hi", "guy", "I'm", "a", "string")
my.char <- paste0("Hi", "guy", "I'm", "a", "string")
```



Character

R is not a zero-indexed language !!

```
my.char <- paste("Hi", "guy")
substr(my.char, 1, 2)
substr(my.char, 1, 2) <- "OH"
strsplit(my.char, ' ')
gsub('.', 'X', my.char)
```



Date time

R stores datetime as a number of day
since the epoch (1 Jan 1970)

```
my.date <- as.Date("2017-08-09")
my.date + 7
weekdays(my.date + 7)
my.date - 365
weekdays(my.date - 365)
```



Testing and change data type

```
# Testing type  
is.character(my.character)  
is.numeric(my.character)
```

```
# Change type  
as.character(9)  
as.numeric(my.character)
```



Function in R

Try to break code in a small part which become easy to maintain and understand



Function in R

```
# Function  
pow <- function(x, y) {  
  result <- x^y  
}  
a = pow(10, 2)
```

```
# Named arguments  
a = pow(x=10, y=3)
```

```
# Default values for arguments  
pow <- function(x, y = 2) {  
  result <- x^y  
}  
a = pow(10)
```



Return value from function

```
check <- function(x) {  
  if(x>0) {  
    result <- "Positive"  
  } else if(x<0) {  
    result <- "Negative"  
  } else {  
    result <- "Zero"  
  }  
  return(result)  
}
```



Return value without return()

```
check <- function(x) {  
  if(x>0) {  
    result <- "Positive"  
  } else if(x<0) {  
    result <- "Negative"  
  } else {  
    result <- "Zero"  
  }  
  result  
}
```



Multiple return from function

```
multi <- function() {  
  my_list <- list("id" = 1, "name" = "Somkiat", "age"= 30)  
  return(my_list)  
}  
  
a = multi()  
a$id  
a$name  
a$age
```



Workshop :: Easy calculator



Requirement

2 Inputs from command line

Choose a operation from common line

Operations => Add, Minus, Multiply, Divide



Create calculator.R

Read Input from command line ?

```
print("Choose operation")
print("1. Add")
print("2. Minus")
print("3. Multiply")
print("4. Divide")

selected <- as.integer(readline(prompt = "Choose [1/2/3/4]: "))
first <- as.integer(readline(prompt = "Fill in first number: "))
second <- as.integer(readline(prompt = "Fill in second number: "))
```



Create calculator.R

Try to write functions by yourself

```
print("Choose operation")
print("1. Add")
print("2. Minus")
print("3. Multiply")
print("4. Divide")

selected <- as.integer(readline(prompt = "Choose [1/2/3/4]: "))
first <- as.integer(readline(prompt = "Fill in first number: "))
second <- as.integer(readline(prompt = "Fill in second number: "))
```



Create calculator.R

Introduce to switch() function

```
add <- function(a, b) {  
  return(a+b)  
}
```

```
operator <- switch(selected, "+", "-", "*", "/")  
result <- switch(selected, add(first, second))  
  
print(paste(first, operator, second, "=", result))
```



Test-Driven Development (TDD) with R



Real Situation

RStudio has a lot of debugging capabilities
Production code can get messy quickly
The biggest code == The bigger problem

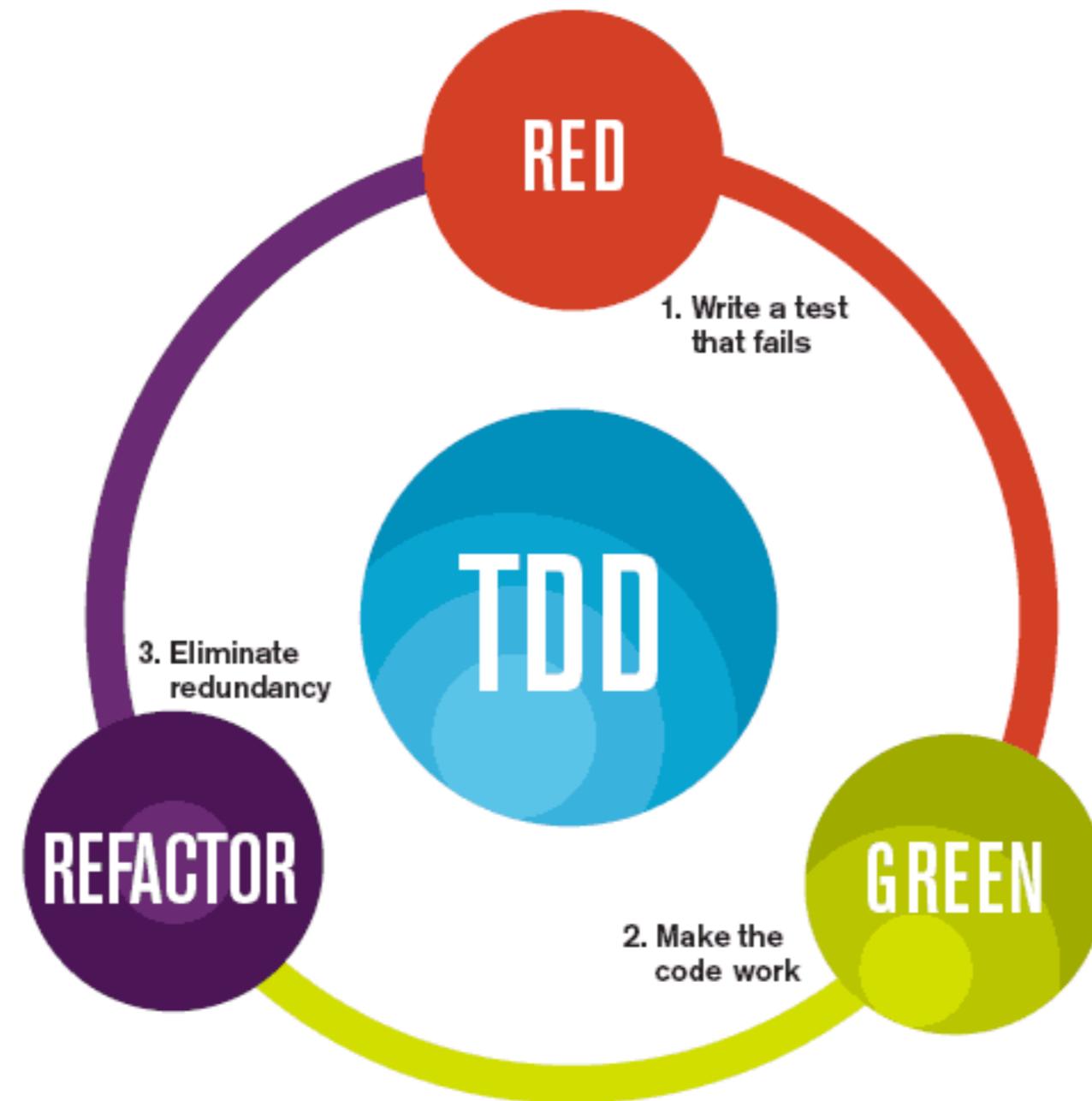


Unit testing

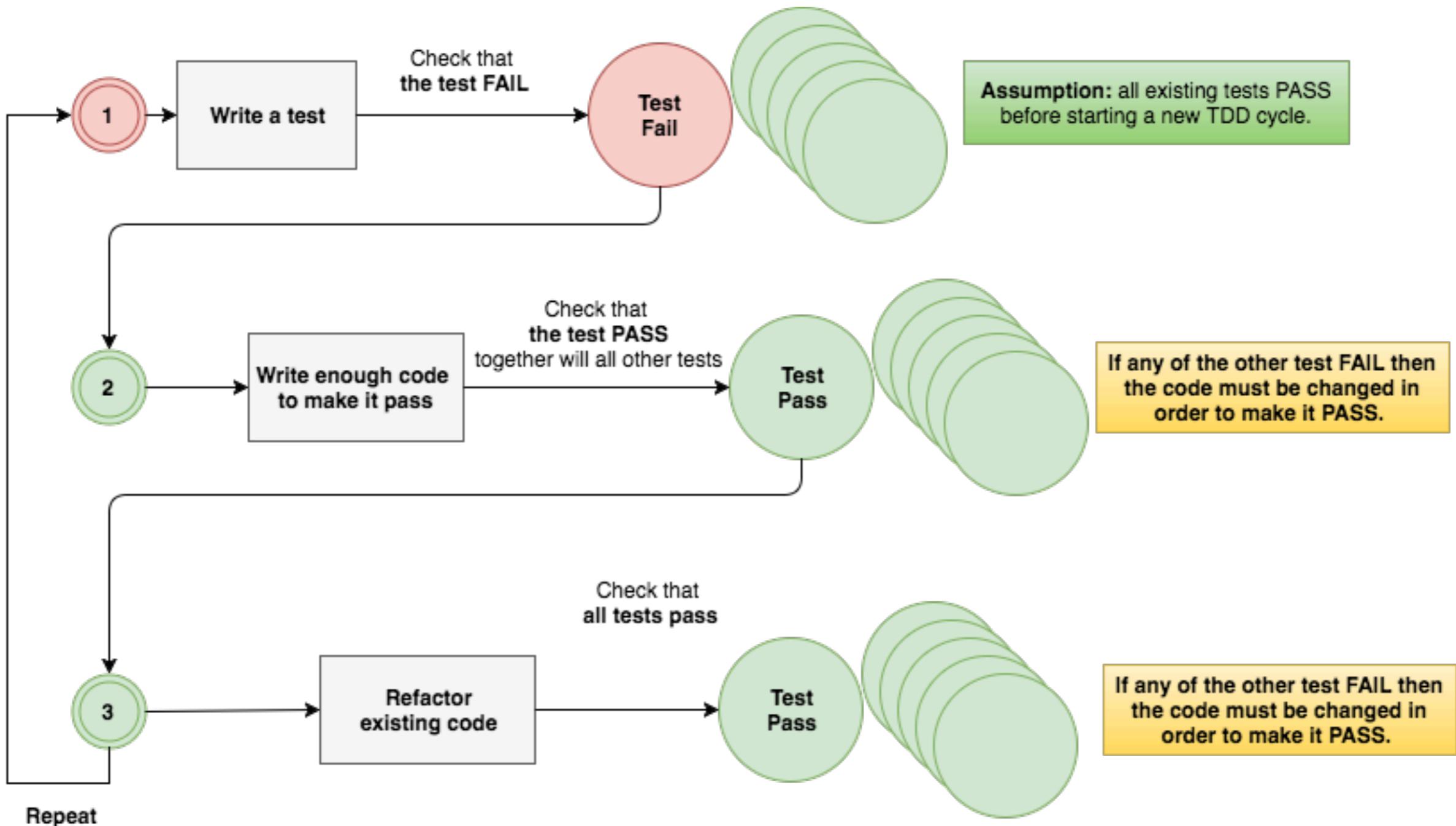
Low level tests of your code functionality
Written by developer
Defense against bugs in the code



Test-Driven Development (TDD)



TDD workflow



by Paracchini Pier Lorenzo, 12.05.2017

https://pparacch.github.io/2017/05/18/test_driven_development_in_r.html



Write Unit Test in R

Using **testthat** package

```
>install_packages("testthat")
```

*“I wrote **testthat** because i discovered i spending to much time recreating bugs that i had previously fixed”*

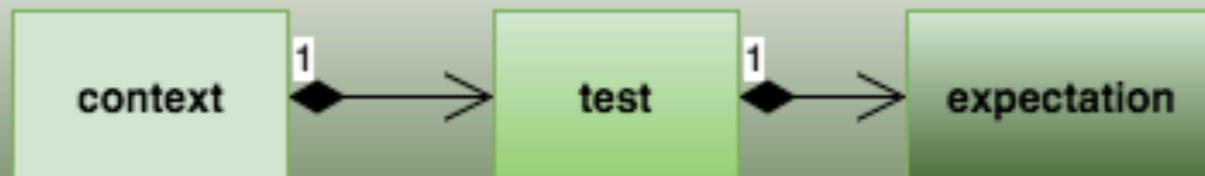
- Hadley Wickham -

<https://cran.r-project.org/web/packages/testthat/index.html>



Test Structure

'testthat' Test Structure



```
context("testing context")
```

```
test_that("passing .... then an error is thrown"), {  
  expect_error(object = functionUnderTest(arg1))  
  expect_error(object = functionUnderTest(arg2))  
}
```

```
test_that("passing .... then no error is thrown"), {  
  expect_match(object = functionUnderTest(arg3), regexp = "...")  
}
```



Project structure

```
. └── runTest.R  
    └── src  
        └── hello.R  
    └── tests  
        └── testHello.R
```



1. Create tests/testHello.R

```
context("TDD with R")
test_that("Hello World", {
  expect_equal("Hello somkiat", say_hi("somkiat"))
})
```



2. Create src/hello.R

```
say_hi <- function(name){  
  return("TODO")  
}
```



3. Create runTest.R

```
library('testthat')
source('./src/hello.R')
test_dir('./tests', reporter = 'Summary')
```



4. Run first test

```
$Rscript runTest.R
```

```
TDD with R: 1
```

```
== Failed ==  
— 1. Failure: Hello World (@testHello.R#3) —  
"Hello somkiat" not equal to say_hi("somkiat").  
1/1 mismatches  
x[1]: "Hello somkiat"  
y[1]: "TODO"  
  
== DONE ==
```



5. Modify src/hello.R

```
say_hi <- function(name){  
  return(sprintf("Hello %s", name))  
}
```



6. Run test again

```
$Rscript runTest.R
```

TDD with R: .

= DONE =====

You are a coding rockstar!



Try to practice
with Easy Calculator



Flow controls



บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

Flow controls

if ... else if ... else
ifelse(expression, true, false)
for loop
while loop
repeat loop

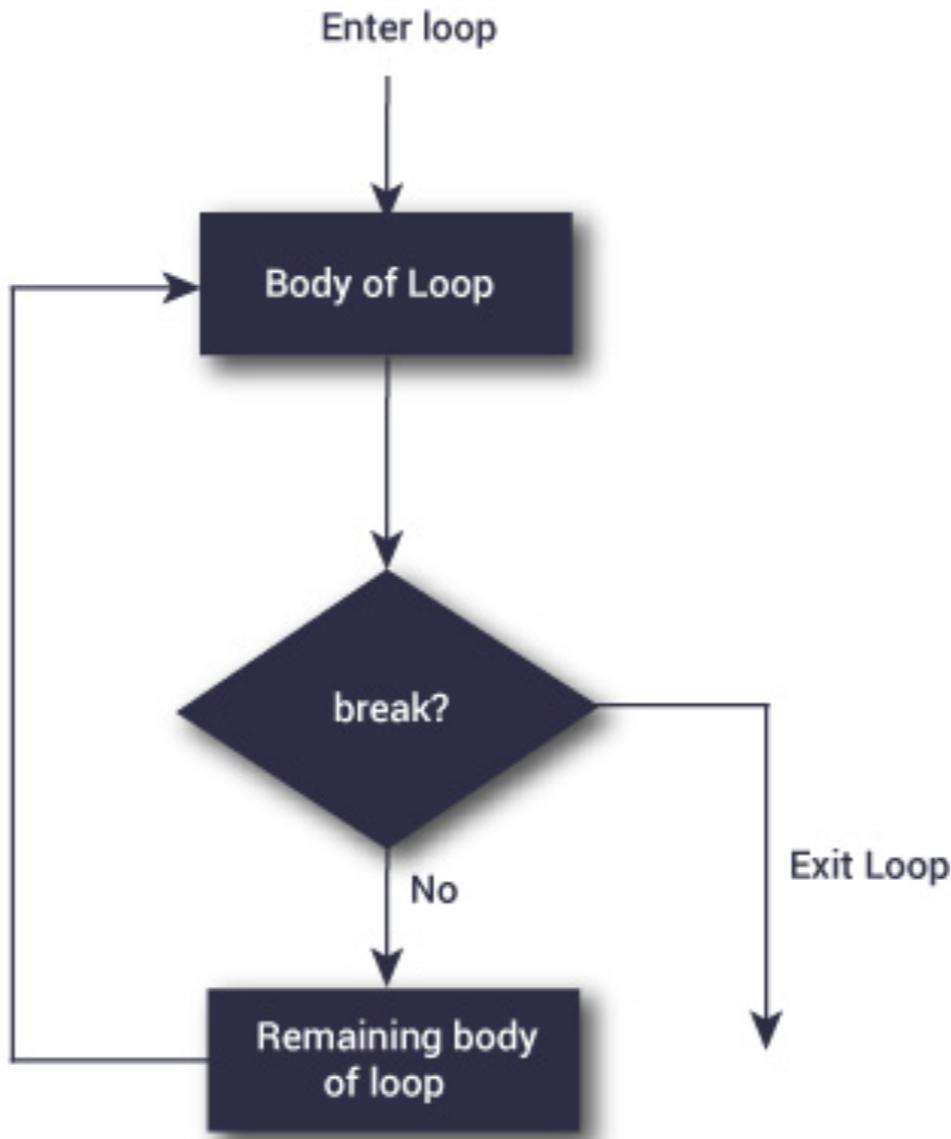


For loop

```
numbers <- c(1, 2, 3, 4, 5)
for(number in numbers) {
  print(number)
}
```



Repeat loop



Repeat loop

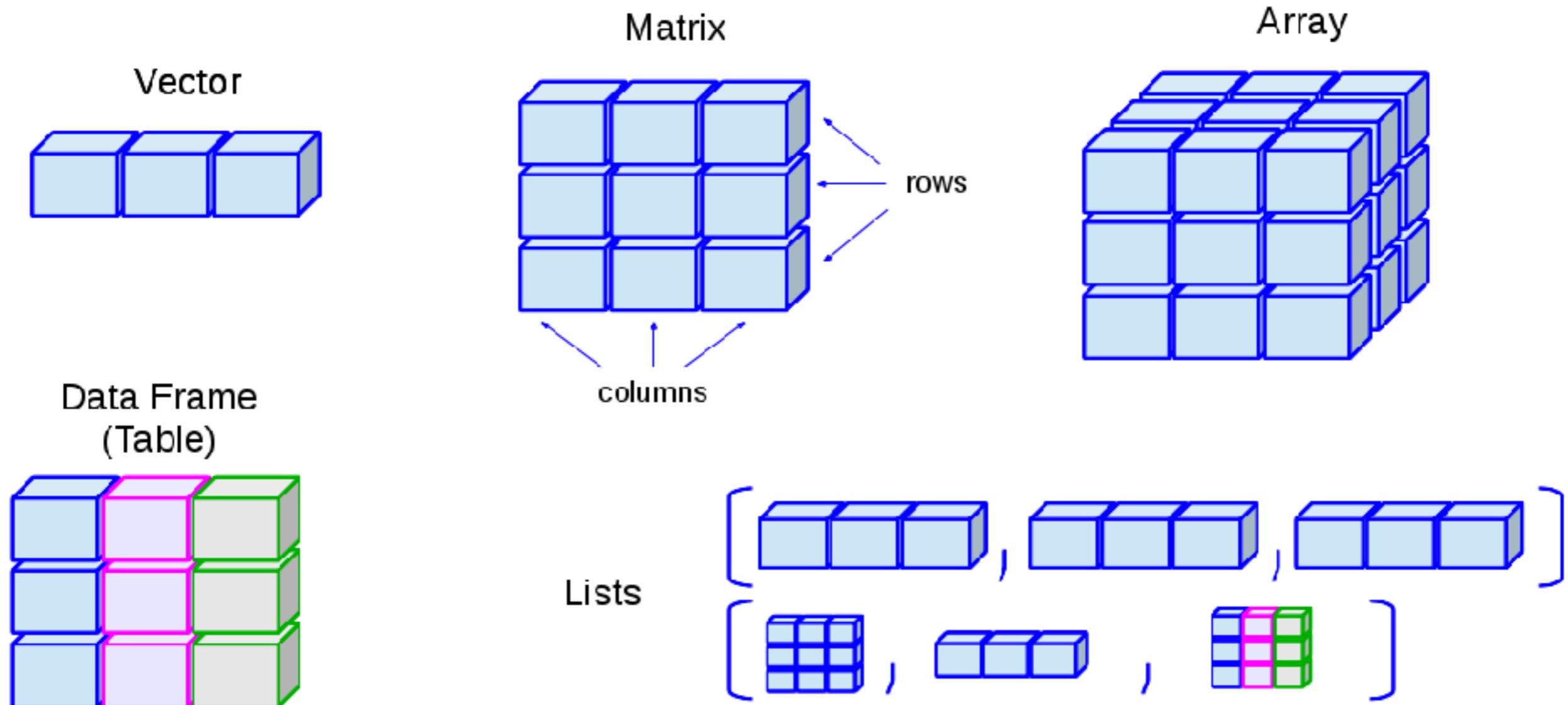
```
i <- 1
repeat {
  if(numbers[i] > 4) {
    break
  }
  print(numbers[i])
  i <- i+1
}
```



Data structure



Data structure



Vector

Collection of elements, all the same type

```
# Create vector  
datas <- c(1, 2, 3, 4, 5)  
datas
```



Vector

Create sequence of number

```
# Create vector  
datas <- 1:5  
datas
```



Matrices

Special case of Vector
Matrices have a dimension attribute

```
my.mat <- matrix(nrow = 2, ncol = 4)  
my.mat <- matrix(1:8, nrow = 2, ncol = 4)
```



Matrices

Special case of Vector
Matrices have a dimension attribute

```
my.vec <- 1:8  
dim(my.vec) <- c(2, 4)  
my.vec
```



Matrices

Special case of Vector

```
vec1 <- 1:4
vec2 <- sample(1:100, 4, replace = FALSE)
vec3 <- rnorm(4, mean = 0, sd = 1)

column.mat <- cbind(vec1, vec2, vec3)
```



Array

Similar matrices but can have more than
2 dimension

```
d1 <- c('a1', 'a2')  
d2 <- c('b1', 'b2', 'b3')  
d3 <- c('c1', 'c2', 'c3')  
  
e1 <- array(data=1:18, c(2, 3, 3))  
e2 <- array(data=1:18, c(2, 3, 3), dimnames = list(d1, d2, d3 ))
```



DataFrame

Different columns can contain different mode of data

```
my.data <- data.frame(  
  column1 = c(1,2,3),  
  column2 = c('one','two','three'),  
  column3 = c(TRUE,TRUE,FALSE))
```



Data

patientId	age	status	diabetes
1	25	Poor	Type1
2	35	Improved	Type2
3	28	Excellent	Type1
4	52	Poor	Type1
5	38	Poor	Type2



Use DataFrame

```
patientId <- c(1, 2, 3, 4, 5)
age <- c(25, 35, 28, 52, 38)
status <- c('Poor', 'Improved', 'Excellent', 'Poor', 'Poor')
diabetes <- c("Type1", "Type2", "Type1", "Type1", "Type2")

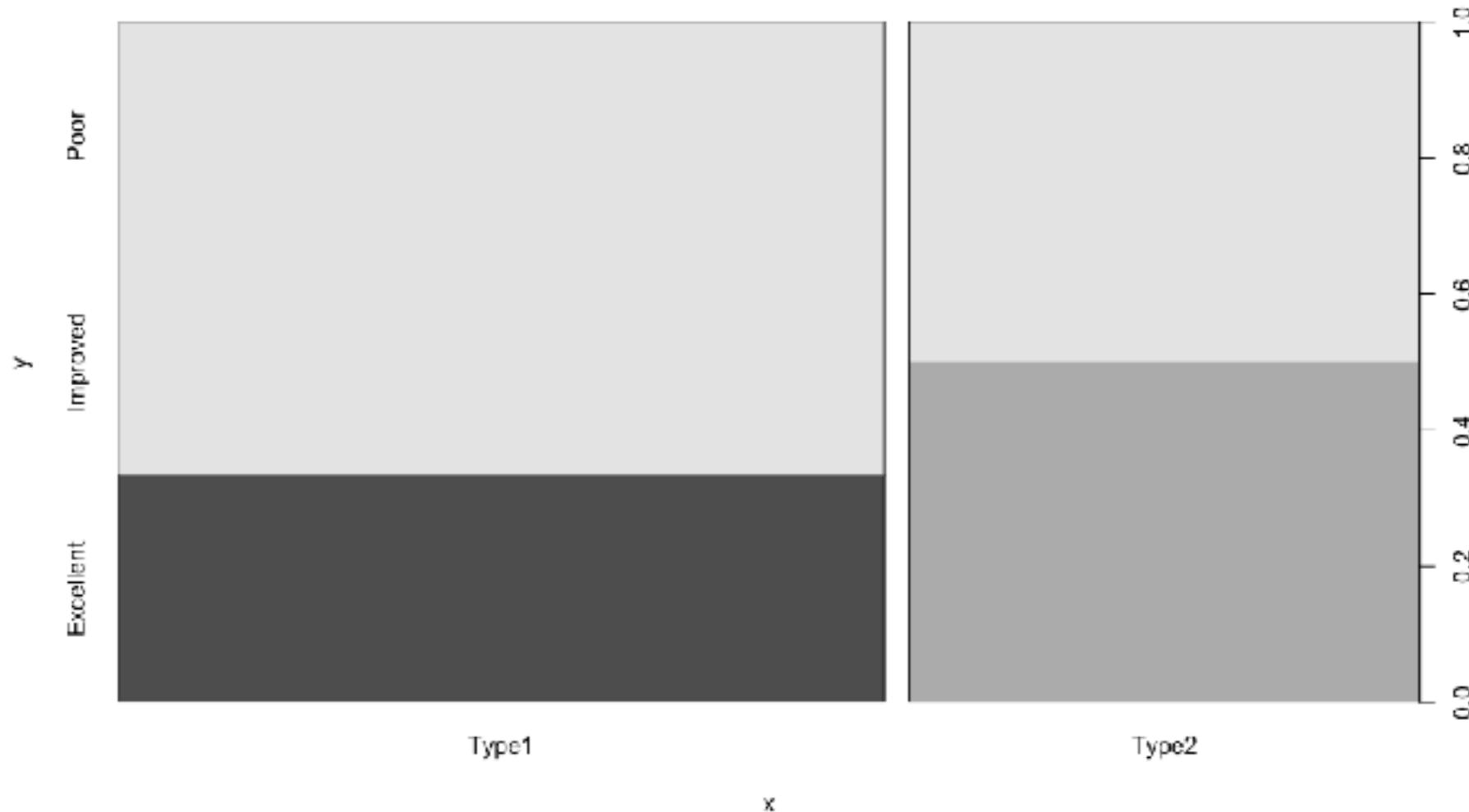
data <- data.frame(patientId, age, status, diabetes)

data[1:1]
data[1:2]
data['age']
data[c('patientId', 'age')]
data$age
data$patient
```



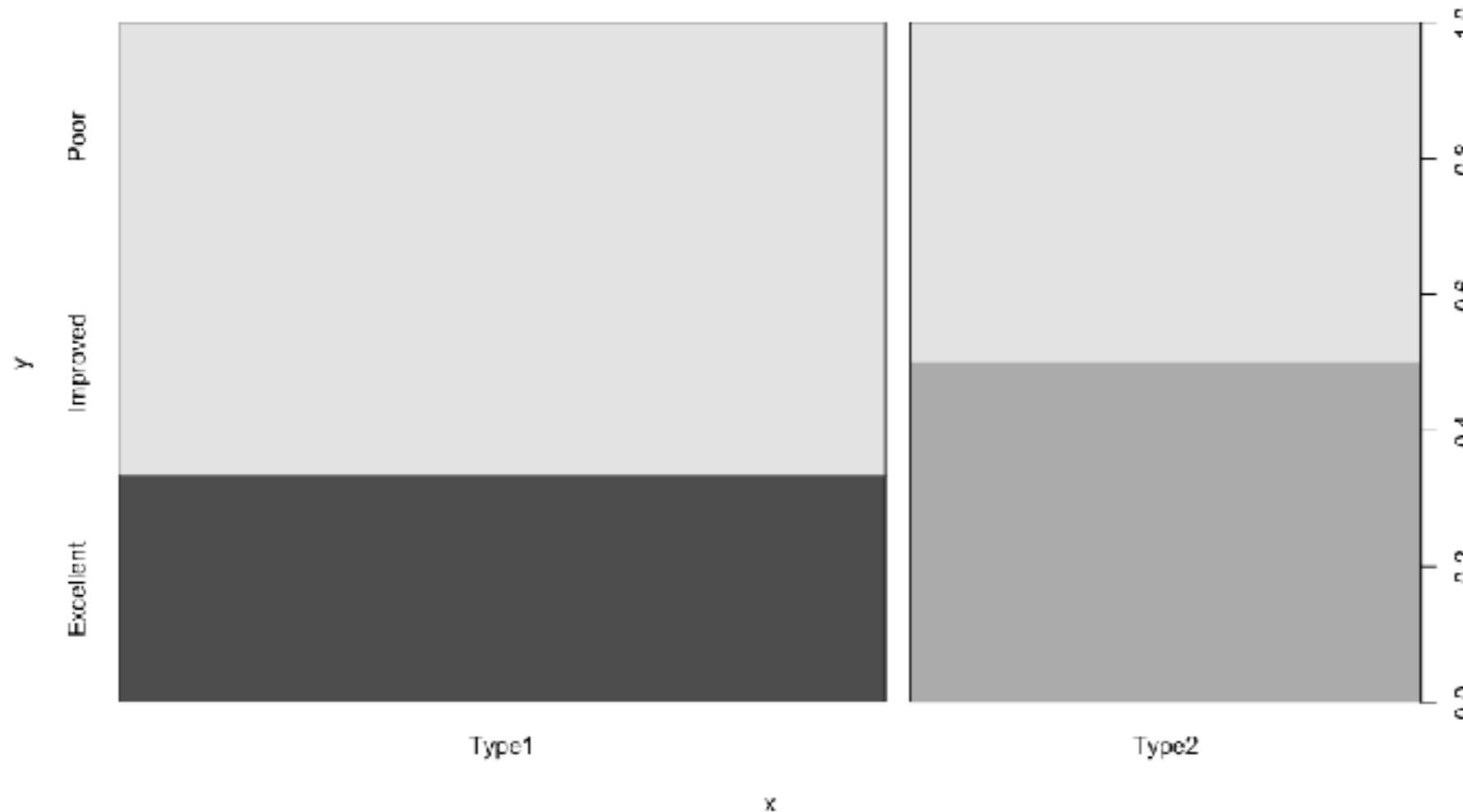
Use DataFrame

```
table(data$diabetes, data$status)  
plot(data$diabetes, data$status)  
str(data)  
summary(data)
```



Use with() function

```
with(data, {  
  plot(diabetes, status)  
})
```



Factor

R stores factors internally as integer
Use the character strings as labels

```
diabetes <- c("Type1", "Type2", "Type1", "Type1")
diabetes <- factor(diabetes)

status <- c("Poor", "Improved", "Excellent", "Poor")
status <- factor(status, order=TRUE,
                  levels=c("Poor", "Improved", "Excellent"))
```



Factor

Use to create value labels
for categorical variables

```
gender <- c(1, 1, 2, 2)
```

```
patientdata$gender <- factor(patientdata$gender,  
                                levels = c(1,2),  
                                labels = c("male", "female"))
```



List

Collection of elements, are not same type

```
my.list = list(1, 'two', 3)
```

```
my.list
```

```
class(my.list)
```

```
my.list[1]
```

```
my.list[[1]]
```

```
str(my.list)
```



List

Most complex of the R data types

```
x <- "Somkiat"  
y <- c(1, 2, 3, 4)  
z <- matrix(1:10, nrow = 2)
```

```
my_list <- list(title = x, id = y, z)
```

```
my_list[[1]]  
my_list[[2]]  
my_list[[3]]
```

```
my_list[["title"]]  
my_list[["id"]]
```

```
my_list$title  
my_list$id
```



Note for programmer

The period(.) has no special significance
Not provide multiline or block comment(#)

Indices in R start at 1

Variables can't be declared



Note for programmer

Assign value to nonexistent element of vector/matrix/list expands that structure to the new value

```
x <- c(1, 2, 3, 4)
```

```
x[10] <- 10
```

```
x
```

```
[1] 1 2 3 4 NA NA NA NA NA 10
```

```
x <- x[1:3] # Back to three elements
```



Write some code with R



Data

Age (month)	Weight (kg.)
1	4.4
3	5.3
5	7.2
2	5.2
11	8.5
9	7.1
12	10.5
3	6.2
6	6.9



Write R Code

Create 2 lists => keep age and weight

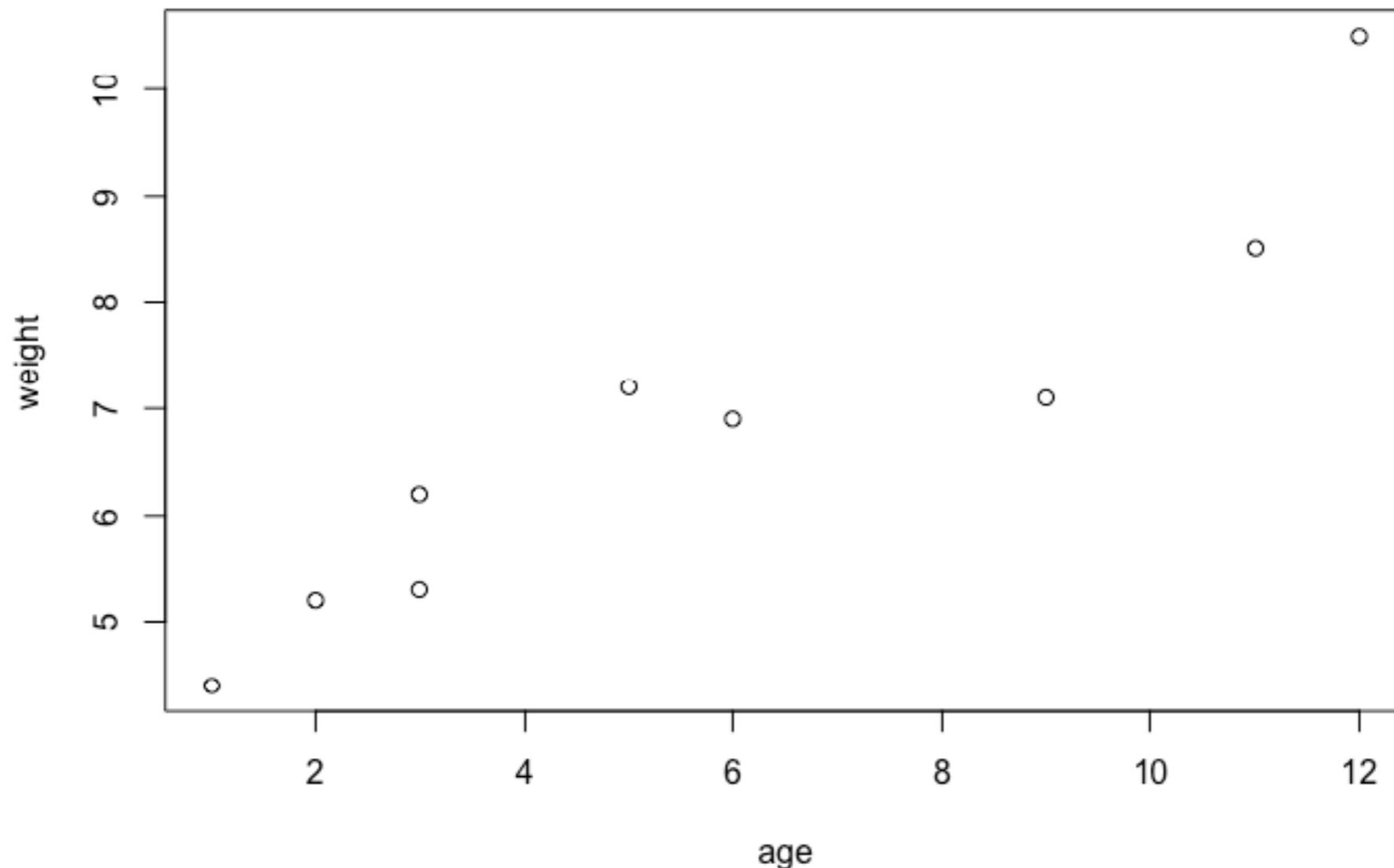
```
age <- c(1,3,5,2,11,9,12,3,6)  
weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1)
```

```
sd(weight)  
mean(weight)  
cor(age, weight)  
plot(age, weight)  
q()
```



Plot data

plot(age, weight)



Reading data



- Flat Files



- Excel Files



- Statistical Software

SPSS
Sas
STATA

- Databases

PostgreSQL
MySQL

- Data from the Web



<https://www.r-bloggers.com/how-to-learn-r-2/>

Reading data

1. From keyboard
2. From text file (delimited text)
3. From connections
4. From Excel (XLConnect, Readxl)
5. From XML/JSON (XML, rjson)
6. From Web

<https://www.datacamp.com/community/tutorials/r-data-import-tutorial>



Reading data (more)

SPSS, SAS, Stata, NetCDF, HDF5

RDBMS => Oracle, MSSQL, MySQL etc.



Read data from text file



Read data from text file

Using `read.table()` function

Read a file in table format and
save to data frame

dataframe <- read.table(file, options)

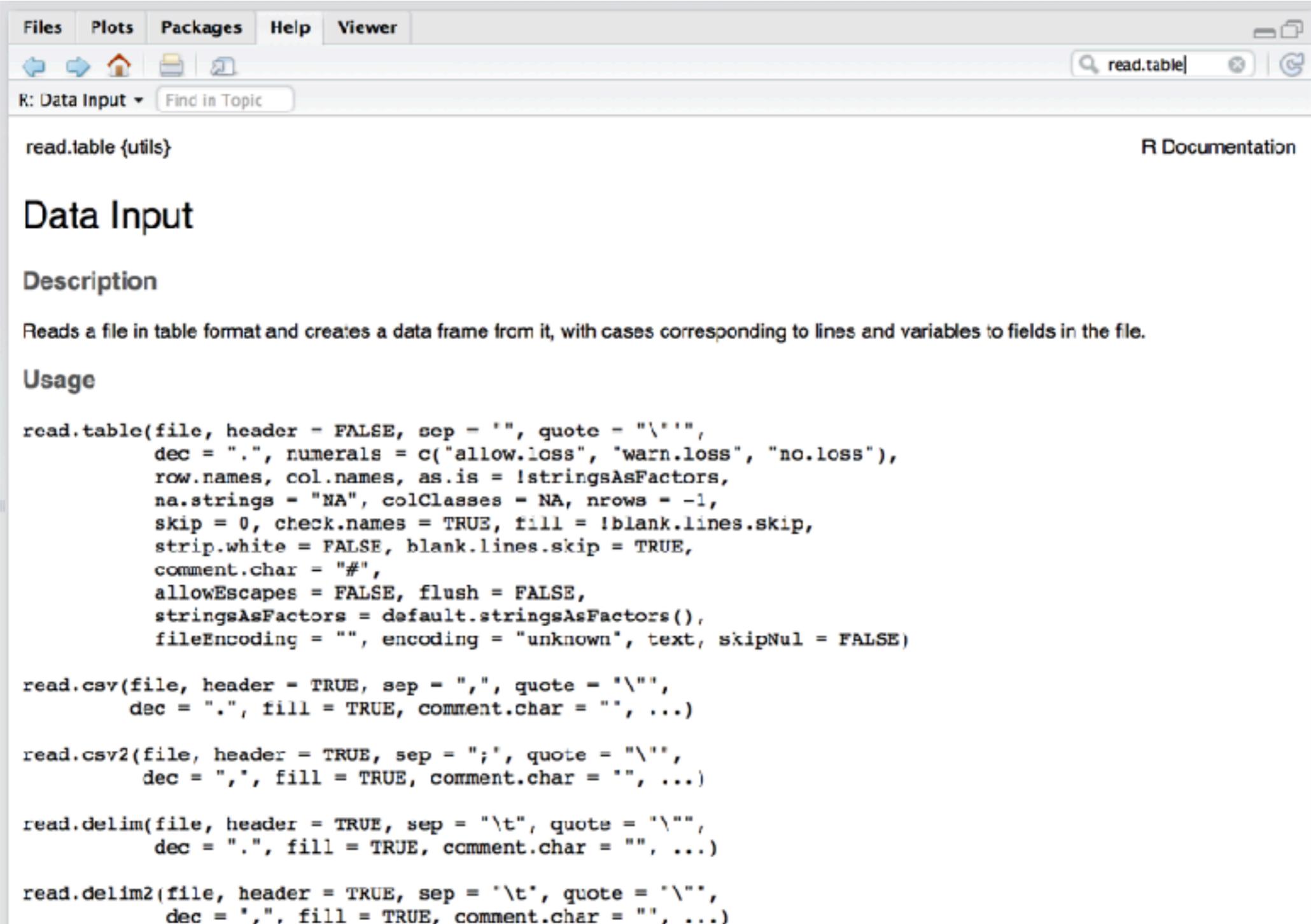


read.table() options

Options	Description
header	Variable names in first line or not
sep	Delimiter separating data values
row.names	Variables to represent row identifiers
col.names	Use when header=FALSE
na.strings	Vector indicating missing value codes



See more in help()



The screenshot shows the RStudio interface with the 'Help' tab selected in the top menu bar. A search bar at the top right contains the text 'read.table'. Below the search bar, the title 'R: Data Input' is followed by a 'Find in Topic' button. The main content area displays the documentation for the 'read.table' function, which is part of the 'utils' package. The title 'Data Input' is bolded. The 'Description' section states that the function reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file. The 'Usage' section provides the R code for various functions related to reading tables:

```
read.table(file, header = FALSE, sep = "", quote = "\"\"",  
          dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),  
          row.names, col.names, as.is = !stringsAsFactors,  
          na.strings = "NA", colClasses = NA, nrows = -1,  
          skip = 0, check.names = TRUE, fill = !blank.lines.skip,  
          strip.white = FALSE, blank.lines.skip = TRUE,  
          comment.char = "#",  
          allowEscapes = FALSE, flush = FALSE,  
          stringsAsFactors = default.stringsAsFactors(),  
          fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)  
  
read.csv(file, header = TRUE, sep = ",", quote = "\"\"",  
         dec = ".", fill = TRUE, comment.char = "", ...)  
  
read.csv2(file, header = TRUE, sep = ";", quote = "\"\"",  
          dec = ",", fill = TRUE, comment.char = "", ...)  
  
read.delim(file, header = TRUE, sep = "\t", quote = "\"\"",  
           dec = ".", fill = TRUE, comment.char = "", ...)  
  
read.delim2(file, header = TRUE, sep = "\t", quote = "\"\"",  
            dec = ",", fill = TRUE, comment.char = "", ...)
```



Read data

```
StudentID,First name,Last name,Math,Science  
011,Bob,Smith,90,80  
012,Jane,Weary,75,  
010,Dan,"Thornton, III",65,75  
040,Mary,"O'Leary",90,95
```

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",")
```



Read data

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",")
```

```
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: Factor w/ 4 levels "Bob","Dan","Jane",...: 1 3 2 4  
 $ Last.name : Factor w/ 4 levels "O'Leary","Smith",...: 2 4 3 1  
 $ Math      : int  90 75 65 90  
 $ Science   : int  80 NA 75 95
```



See in result

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",")  
  
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: Factor w/ 4 levels "Bob","Dan","Jane",...: 1 3 2 4  
 $ Last.name : Factor w/ 4 levels "O'Leary","Smith",...: 2 4 3 1  
 $ Math       : int  90 75 65 90  
 $ Science    : int  80 NA 75 95
```

StudentID column is the row name, not has a label



See in result

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",")  
  
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: Factor w/ 4 levels "Bob","Dan","Jane",...: 1 3 2 4  
 $ Last.name : Factor w/ 4 levels "O'Leary","Smith",...: 2 4 3 1  
 $ Math       : int  90 75 65 90  
 $ Science    : int  80 NA 75 95
```

Renamed of variables to R convention



See in result

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",")
```

```
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: Factor w/ 4 levels "Bob","Dan","Jane",...: 1 3 2 4  
 $ Last.name : Factor w/ 4 levels "O'Leary","Smith",...: 2 4 3 1  
 $ Math       : int 90 75 65 90  
 $ Science    : int 80 NA 75 95
```

See missing value



See in result

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",")  
  
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: Factor w/ 4 levels "Bob","Dan","Jane",...: 1 3 2 4  
 $ Last.name : Factor w/ 4 levels "O'Leary","Smith",...: 2 4 3 1  
 $ Math       : int  90 75 65 90  
 $ Science    : int  80 NA 75 95
```

First and Last name are converted to factors

By default, `read.table()` convert character variable to factors



Change option :: disable factor

```
datas <- read.table("data01.csv", header=TRUE,  
                     row.names="StudentID", sep=",",  
                     stringsAsFactors = FALSE)
```

```
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: chr "Bob" "Jane" "Dan" "Mary"  
 $ Last.name : chr "Smith" "Weary" "Thornton, III" "O'Leary"  
 $ Math       : int 90 75 65 90  
 $ Science    : int 80 NA 75 95
```



Change option :: change class

```
datas <- read.table("data01.csv", header=TRUE,  
                      row.names="StudentID", sep=",",  
                      stringsAsFactors = FALSE,  
                      colClasses = c("character", "character",  
"character", "numeric", "numeric"))
```

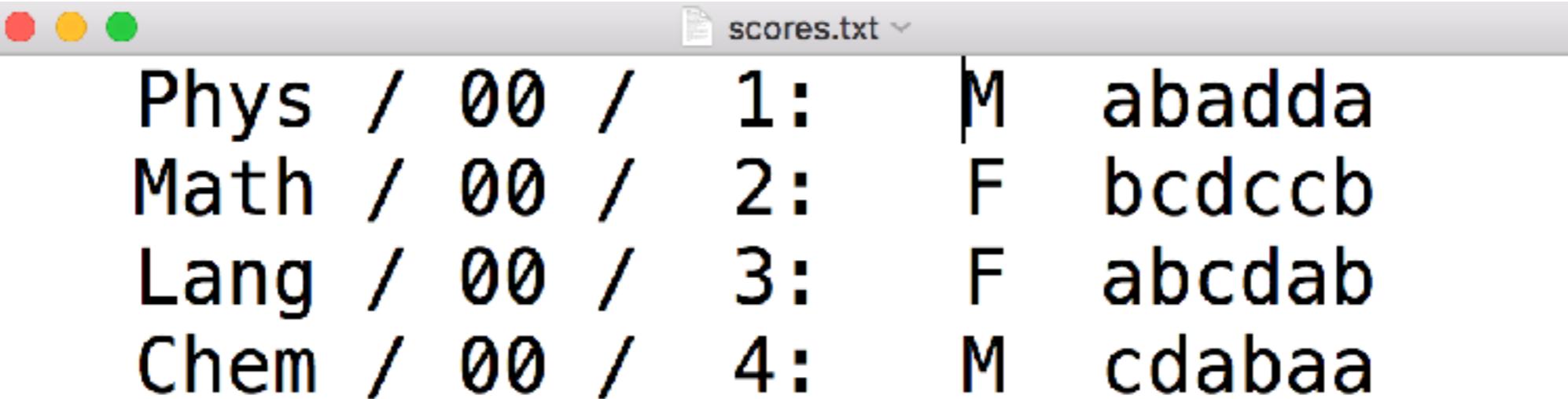
```
> str(datas)  
'data.frame': 4 obs. of 4 variables:  
 $ First.name: chr  "Bob" "Jane" "Dan" "Mary"  
 $ Last.name : chr  "Smith" "Weary" "Thornton, III" "O'Leary"  
 $ Math       : num  90 75 65 90  
 $ Science    : num  80 NA 75 95
```



Read data from Fixed column data



Read data



The screenshot shows a Mac OS X window titled "scores.txt". The window contains the following data:

Subject	Score	Grade	Letter Grade	Initials	Name
Phys	/ 00	/ 1:	M	abadda	
Math	/ 00	/ 2:	F	bcdccb	
Lang	/ 00	/ 3:	F	abcdab	
Chem	/ 00	/ 4:	M	cdabaa	



Read data from fixed column file

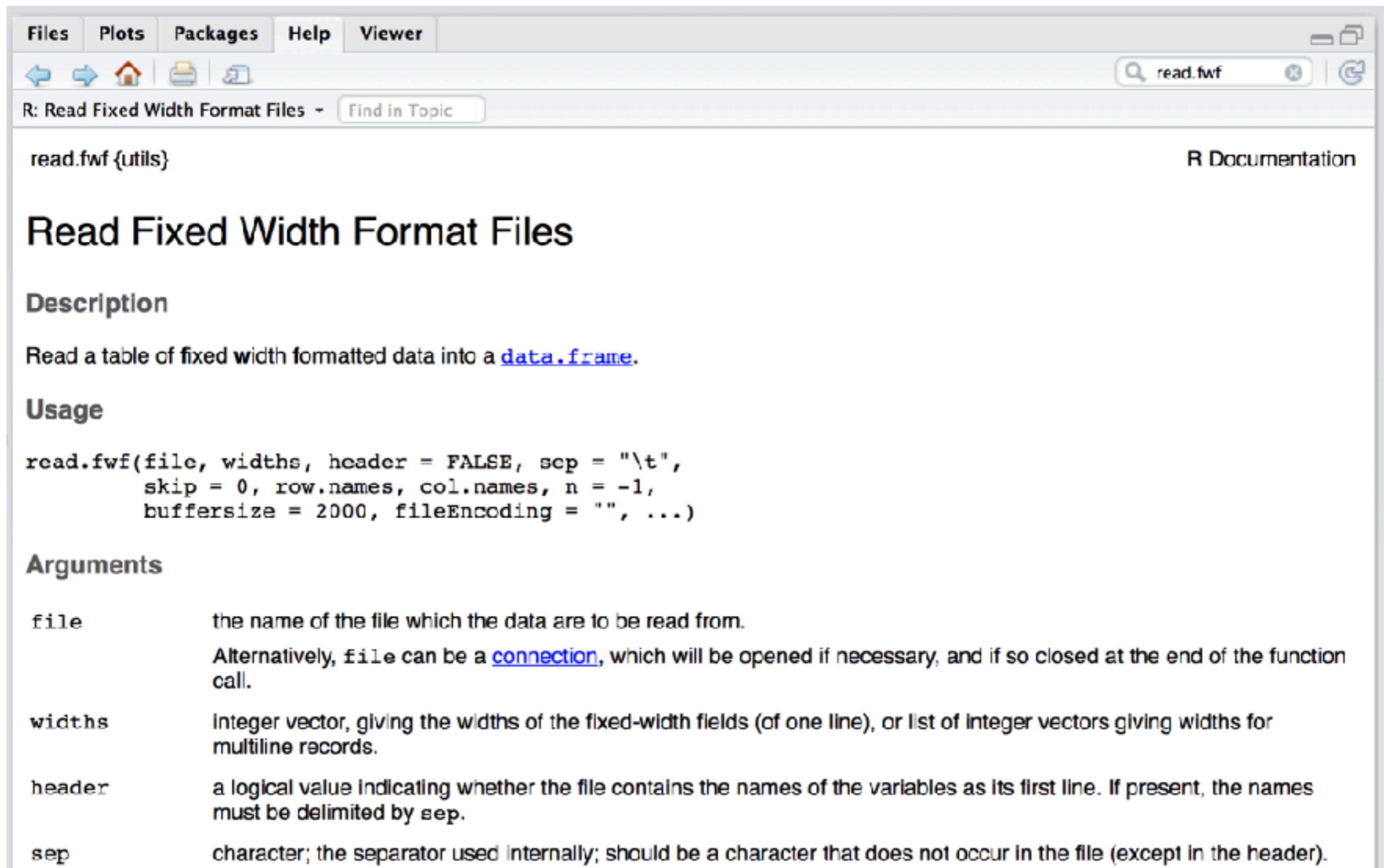
Using `read.fwf()` function

Fixed With Formatted data

dataframe <- read.fwf(file, options)



See more in `help()`



The screenshot shows the RStudio interface with the 'Viewer' tab selected. The title bar says 'read.fwf'. The main content area displays the documentation for the `read.fwf` function.

read.fwf {utils} R Documentation

Read Fixed Width Format Files

Description

Read a table of fixed width formatted data into a [data.frame](#).

Usage

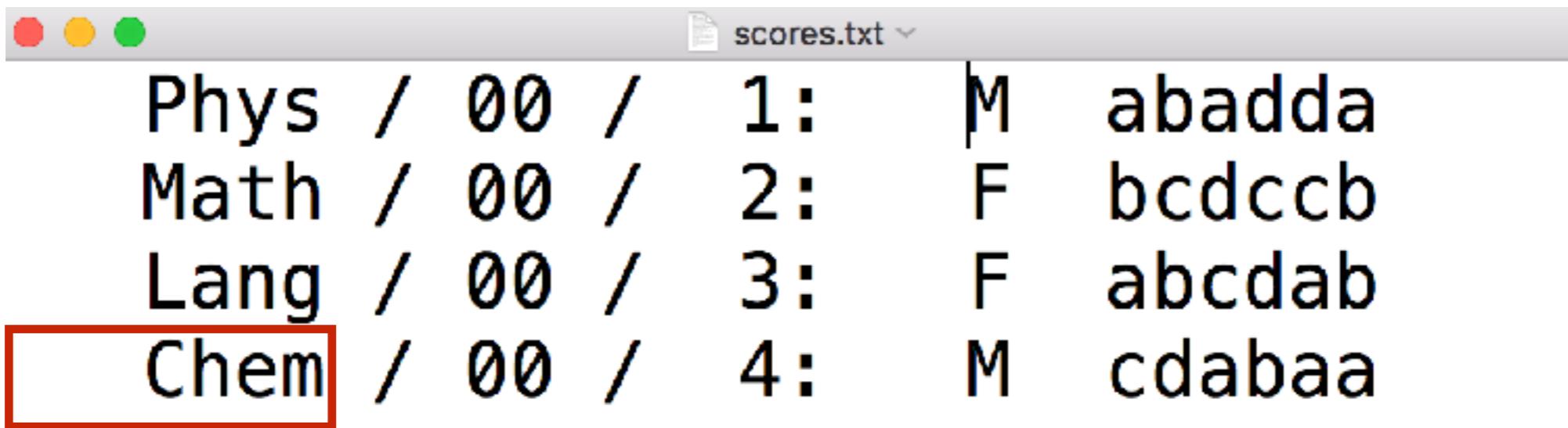
```
read.fwf(file, widths, header = FALSE, sep = "\t",
         skip = 0, row.names, col.names, n = -1,
         bufsize = 2000, fileEncoding = "", ...)
```

Arguments

file	the name of the file which the data are to be read from. Alternatively, <code>file</code> can be a connection , which will be opened if necessary, and if so closed at the end of the function call.
widths	integer vector, giving the widths of the fixed-width fields (of one line), or list of integer vectors giving widths for multiline records.
header	a logical value indicating whether the file contains the names of the variables as its first line. If present, the names must be delimited by <code>sep</code> .
sep	character; the separator used internally; should be a character that does not occur in the file (except in the header).



Read data



Phys	/	00	/	1:	M	abadda
Math	/	00	/	2:	F	bcdccb
Lang	/	00	/	3:	F	abcdab
Chem	/	00	/	4:	M	cdabaa

7



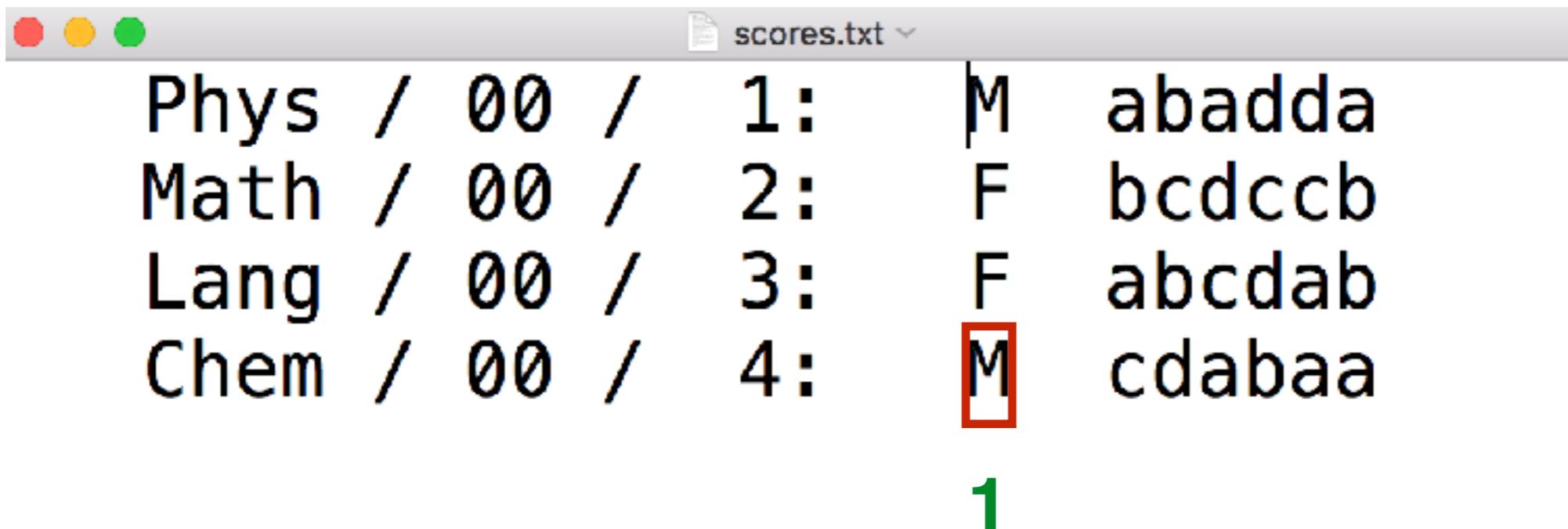
Read data

scores.txt					
Phys	/	00	/	1:	M abadda
Math	/	00	/	2:	F bcdccb
Lang	/	00	/	3:	F abcdab
Chem	/	00	/	4:	M cdabaa

14



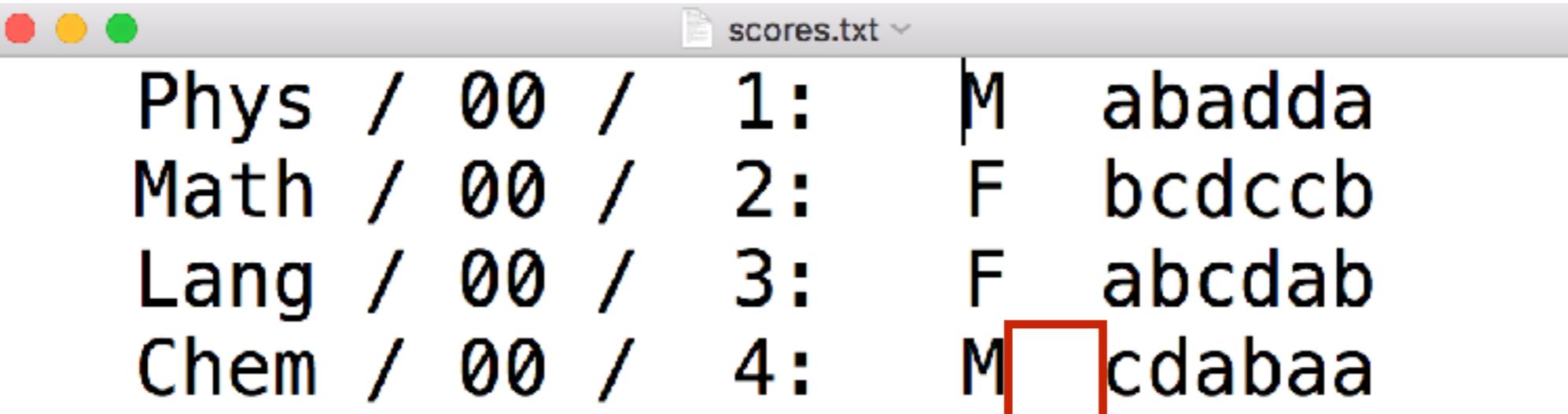
Read data



```
scores.txt
Phys / 00 / 1: M abadda
Math / 00 / 2: F bcdccb
Lang / 00 / 3: F abcdab
Chem / 00 / 4: M cdabaa
1
```



Read data



The screenshot shows a terminal window with the title bar "scores.txt". The file contains the following data:

Subject	Score	Grade	Name
Phys	00	1:	M abadda
Math	00	2:	F bcdccb
Lang	00	3:	F abcdab
Chem	00	4:	M <input type="text"/> cdabaaa

2



Read data

scores.txt						
Phys	/	00	/	1:	M	abadda
Math	/	00	/	2:	F	bcdccb
Lang	/	00	/	3:	F	abcdab
Chem	/	00	/	4:	M	cdabaa

111111



Read data

```
datas <- read.fwf("scores.txt",
  widths= c(7,-14, 1, -2, 1, 1, 1, 1, 1, 1),
col.names=c("subject","sex","s1","s2","s3","s4","s5","s6"),
  strip.white=TRUE)
```

> datas

	subject	sex	s1	s2	s3	s4	s5	s6
1	Phys	M	a	b	a	d	d	a
2	Math	F	b	c	d	c	c	b
3	Lang	F	a	b	c	d	a	b
4	Chem	M	c	d	a	b	a	a



Useful function with dataset

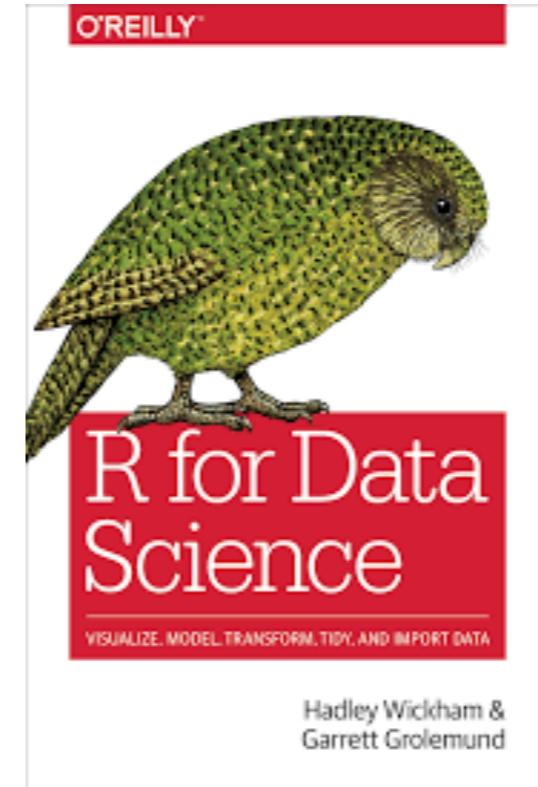
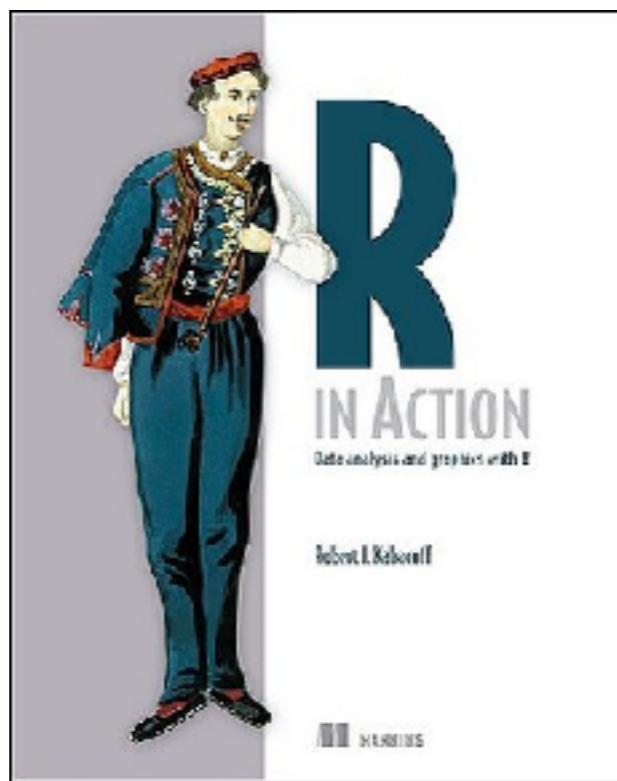
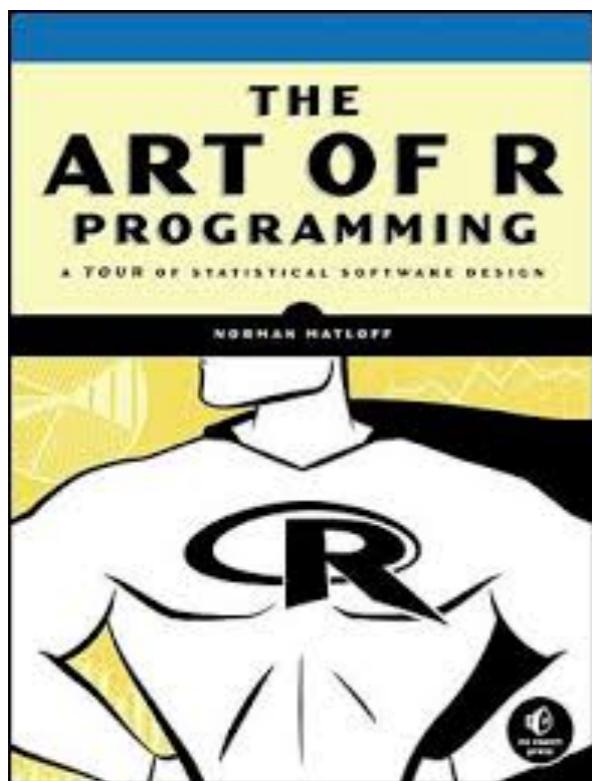
Function	Description
length(object)	Number of elements in an object
dim(object)	Dimension of an object
str(object)	Structure of an object
class(object)	Class of an object
mode(object)	How an object is stored
names(object)	Given the names of elements in an
c(object, object, ...)	Combines objects into a vector
head(object)	List of first part of an object
tail(object)	List of last part of an object
rm(object, object, ...)	Delete one or more objects



Statistical graphics



Resources



Data manipulation



Workshop



บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่