DATA SCIENCE INDONESIA

Introduction to R for Data Science

Universitas Airlangga, 14 Agustus 2018

Achmad Wildan Al Aziz

Outline

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

Introduction to R

R and R Studio

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

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

Introduction to R

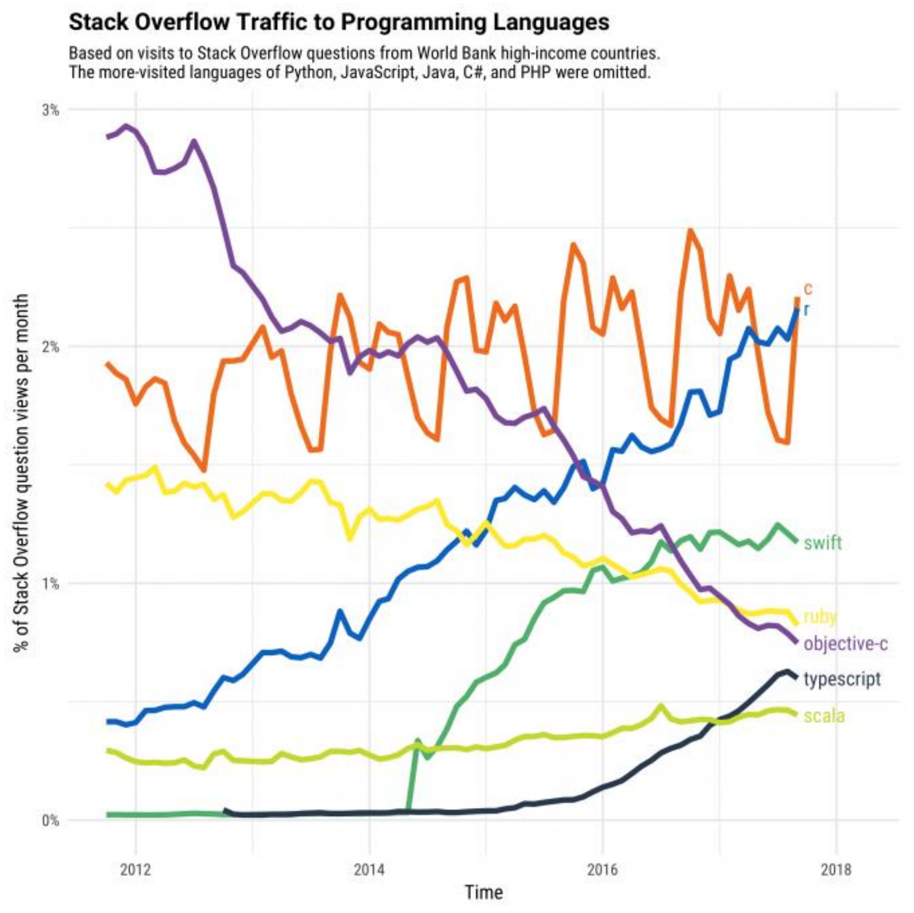
Why use R?

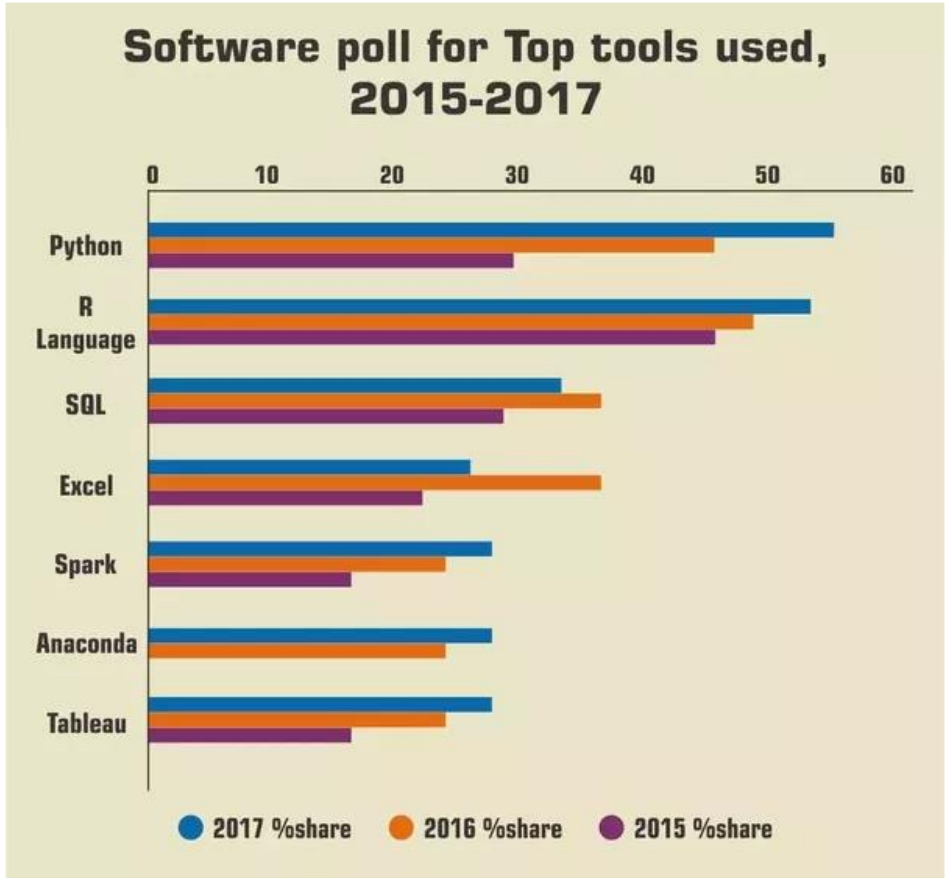
- **Data analysis software:** R is s data analysis software. It is used by data scientists for statistical analysis, predictive modeling and visualization.
- **Statistical analysis environment:** R provides a complete environment for statistical analysis. It is easy to implement statistical methods in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

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

Introduction to R

Why use R?





Basic Calculation

Aritmathic Operation

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

Assignment Variable

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

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

Basic Calculation

Mathematical Function

Function	**Meaning**
log(x)	log to base e of x
exp(x)	antilog of x (=2.7818x)
log(x,n)	log to base n of x
log10(x)	log to base 10 of x
sqrt(x)	square root of x
factorial(x)	x!
choose(n,x)	binomial coefficients n!/(x! (n – x)!)
gamma(x)	Γ.x.(x – 1)! for integer x
lgamma(x)	natural log of gamma(x)
floor(x)	greatest integer < x

Basic Calculation

Mathematical Function

Function	**Meaning**
ceiling(x)	smallest integer > x
trunc(x)	closest integer to x between x and 0: trunc(1.5) =1, trunc(-1.5) = -1
trunc	is like floor for positive values and like
ceiling	for negative values
round(x, digits=0)	round the value of x to an integer
signif(x, digits=6)	give x to six digits in scientific notation
runif(n)	generates n random numbers between 0 and 1 from a uniform distribution
cos(x)	cosine of x in radians
sin(x)	sine of x in radians
tan(x)	tangent of x in radians
acos(x), asin(x), atan(x)	inverse trigonometric transformations of real or complex numbers.
acosh(x), asinh(x), atanh(x)	inverse hyperbolic trigonometric transformations on real or complex numbers
abs(x)	the absolute value of x, ignoring the minus sign if there is one

Data dan Variable

Main Structures

Vector	array 1 dimensi dengan ukuran m (1 tipe data)
Matrix	array 2 dimensi dengan ukuran m × n (1 tipe data)
Dataframe	seperti matrix, namun bisa menampung lebih dari 1 tipe data

Class

character	vector of strings
numeric	vector of real numbers
integer	vector of signed integer
logical	vector of boolean (TRUE or FALSE)
complex	vector of complex numbers
list	vector of R objects
factor	sets of labelled observations, pre-defined set of labels
NA	not available, missing value

Data dan Variable

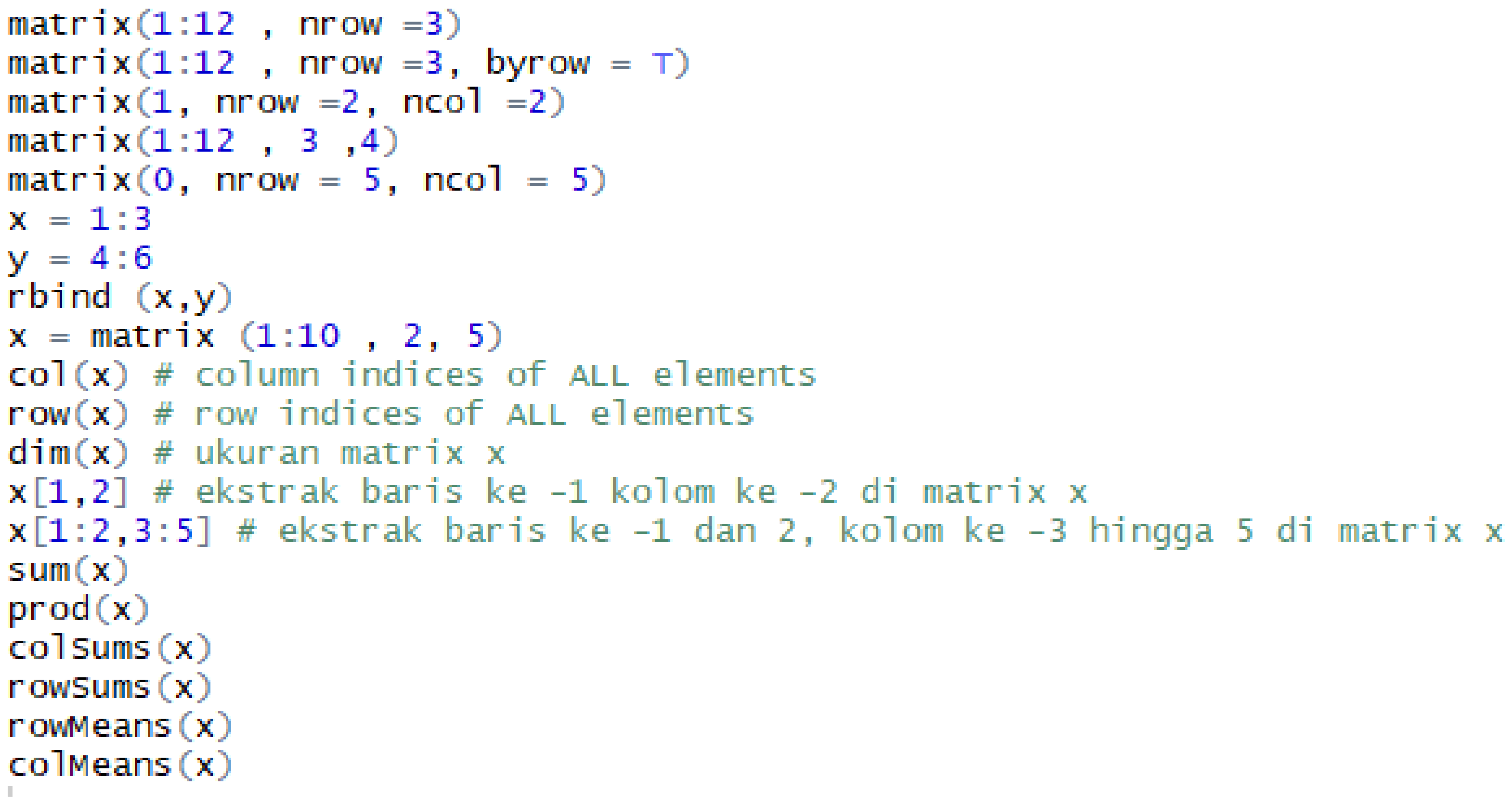
Vector

```
a = 1:3
b = 2:4
c(a,b) # [1] 1 2 3 2 3 4
c(1 ,1:3) # [1] 1 1 2 3
array(1 ,4) # [1] 1 1 1 1
seq(1 ,3) # [1] 1 2 3
seq(3) # [1] 1 2 3
seq(1 ,2 , by= 0.1) # [1] 1.1 1.2 1.3 1.4 1.5 ...
seq(1 ,3 ,0.5) # [1] 1.0 1.5 2.0 2.5 3
seq(1 ,3 , length.out = 4) # [1] 1.00 1.67 2.33 3.00
rep(1:4 ,2) # [1] 1 2 3 4 1 2 3 4
rep(1:4 , each = 2) # [1] 1 1 2 2 3 3 4 4
rep(c(7 ,9 ,3) , 1:3) # [1] 7 9 9 3 3 3
a <- c(2 ,3 ,1 ,4) # double vector
length(a) # [1] 4
rev(a) # [1] 4 1 3 2 reverse
a[2] # returns 2nd element of a
a[1:2] # [1] 2 3
a[ -1] # [1] 3 1 4
a[-c(1 ,2)] # [1] 1 4
a[a < 3] # [1] 2 1
which(a == 3) # [1] 2
a > 1 # [1] TRUE TRUE FALSE TRUE
```

```
a <- letters[1:3]
b <- LETTERS[1:3]
c <- month.abb[1:6]
d <- month.name[1:12]
```

Data dan Variable

Matrix



```r
matrix(1:12 , nrow =3)
matrix(1:12 , nrow =3, byrow = T)
matrix(1, nrow =2, ncol =2)
matrix(1:12 , 3 ,4)
matrix(0, nrow = 5, ncol = 5)
x = 1:3
y = 4:6
rbind (x,y)
x = matrix (1:10 , 2, 5)
col(x) # column indices of ALL elements
row(x) # row indices of ALL elements
dim(x) # ukuran matrix x
x[1,2] # ekstrak baris ke -1 kolom ke -2 di matrix x
x[1:2,3:5] # ekstrak baris ke -1 dan 2, kolom ke -3 hingga 5 di matrix x
sum(x)
prod(x)
colSums(x)
rowSums(x)
rowMeans(x)
colMeans(x)
```

Data dan Variable

Matrix

```
x1 = c(2,5)
x2 = c(4,7)
x=cbind (x1,x2)
t(x) #matrix transpose
solve(x) #inverse matrix
 [,1]       [,2]
x1 -1.1666667  0.6666667
x2  0.8333333 -0.3333333
det(x) #determinant matrix
[1] -6
diag(x) #diagonal matrix
[1] 2 7
```

```
y1 = c(3,6)
y2 = c(1,4)
y=cbind (y1,y2)
x*y
     x1 x2
[1,]  6  4
[2,] 30 28
x%*%y
     y1 y2
[1,] 30 18
[2,] 57 33
```

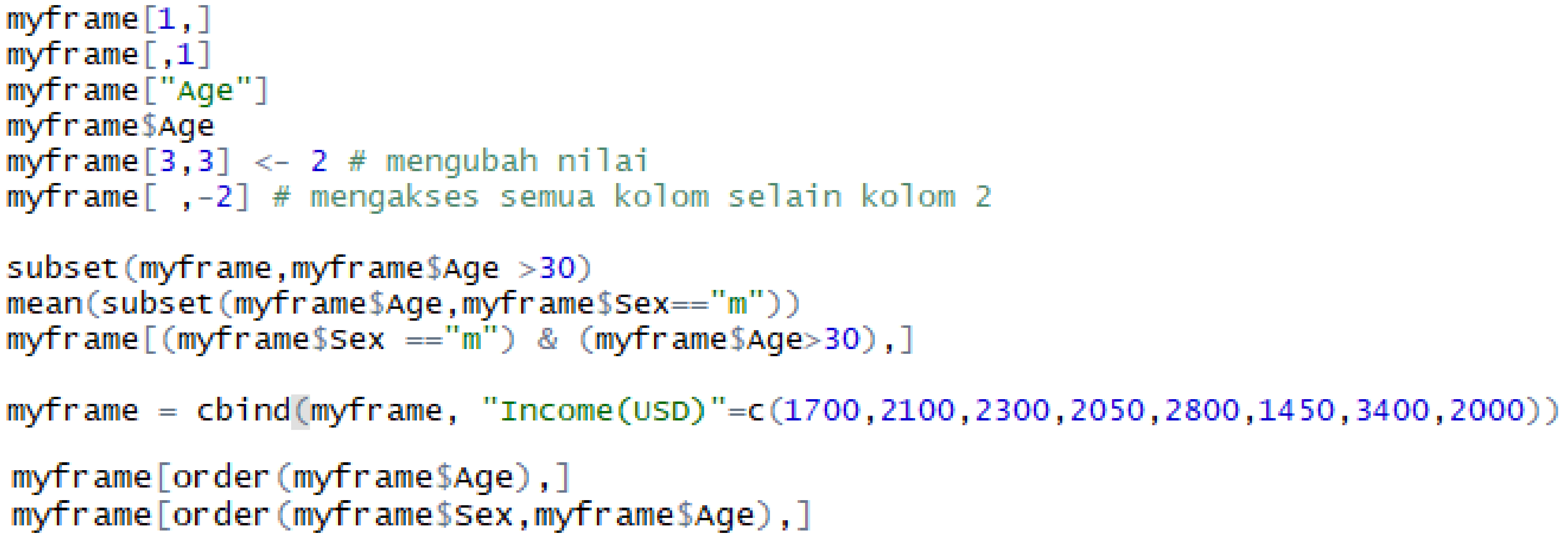
Data dan Variable

Dataframe

```
Age <- c(10 ,20 ,15 ,43 ,76 ,41 ,25 ,46)

Sex <- factor (c("m","f","m","f","m","f","m","f"))

Sibblings <- c(2 ,5 ,8 ,3 ,6 ,1 ,5 ,6)

myframe <- data.frame(Age, Sex, Sibblings)

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

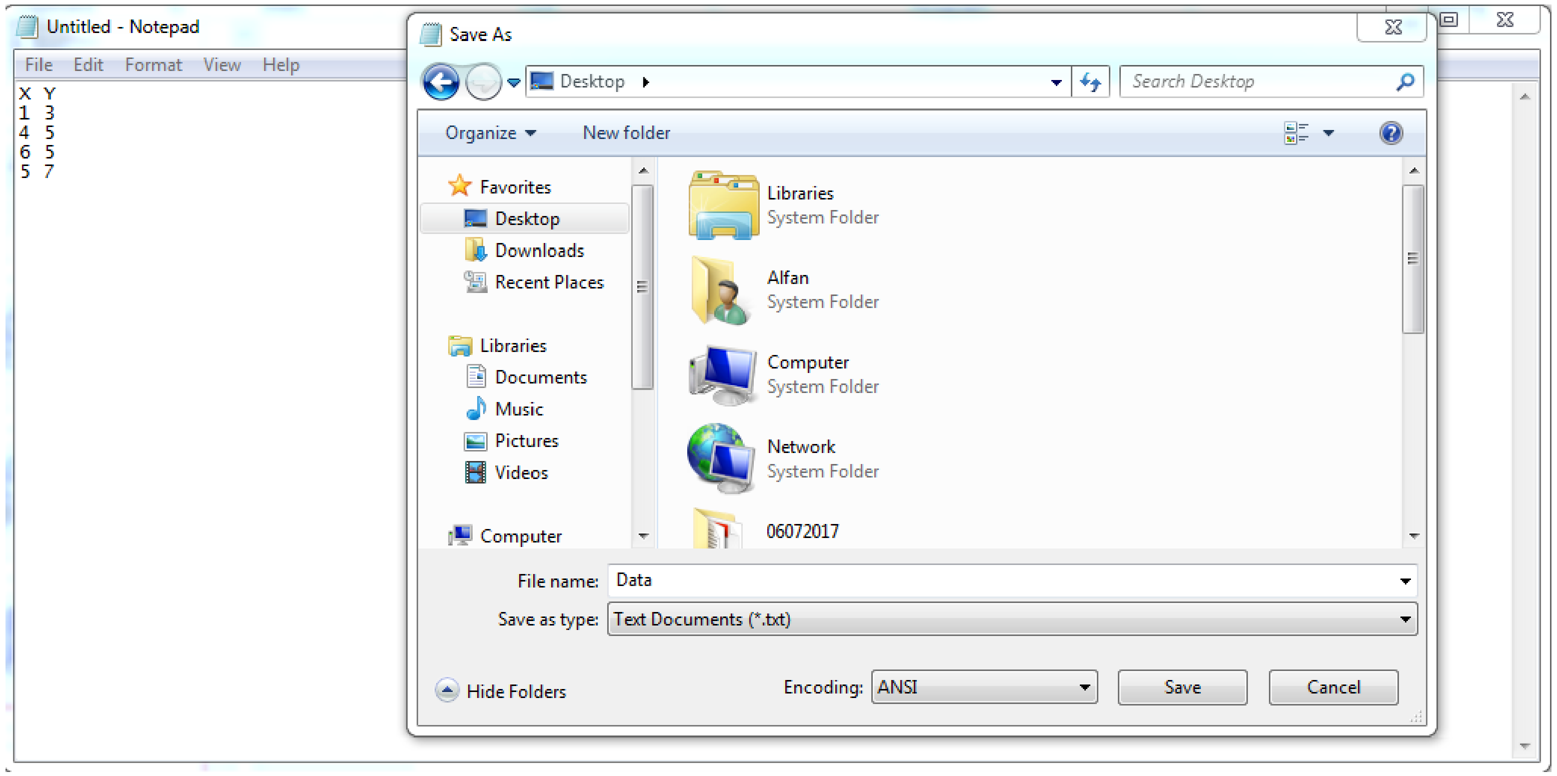
Data dan Variable

Dataframe



```
myframe[1,]
myframe[,1]
myframe["Age"]
myframe$Age
myframe[3,3] <- 2 # mengubah nilai
myframe[ ,-2] # mengakses semua kolom selain kolom 2

subset(myframe,myframe$Age >30)
mean(subset(myframe$Age,myframe$Sex=="m"))
myframe[(myframe$Sex =="m") & (myframe$Age>30),]

myframe = cbind(myframe, "Income(USD)"=c(1700,2100,2300,2050,2800,1450,3400,2000))

 myframe[order(myframe$Age),]
 myframe[order(myframe$Sex,myframe$Age),]
```

Read and Write Data


```
X  Y
1  3
4  5
6  5
5  7
```

File name: Data

Save as type: Text Documents (*.txt)

Encoding: ANSI

For example we create data in notepad

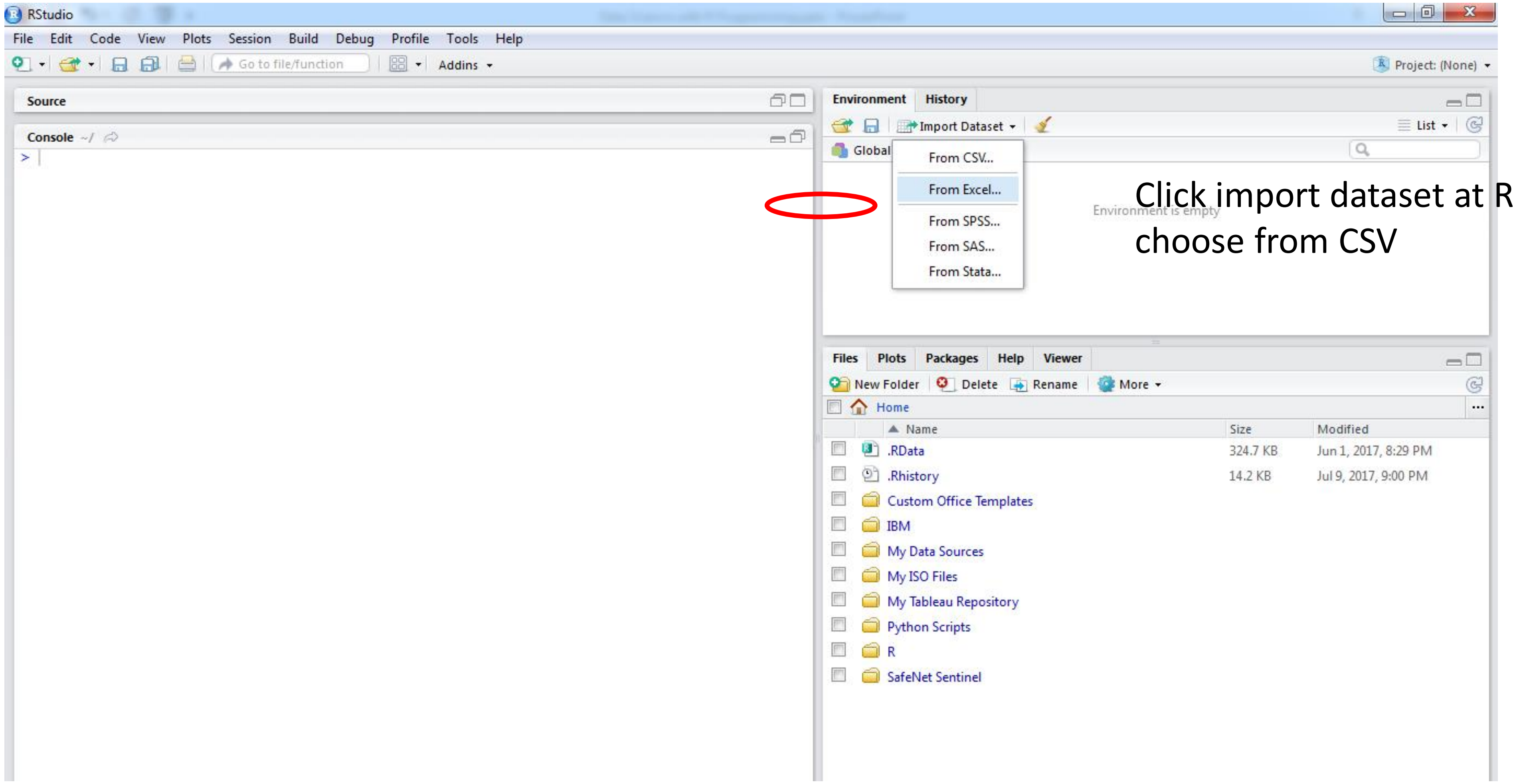
Read and Write Data

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

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

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

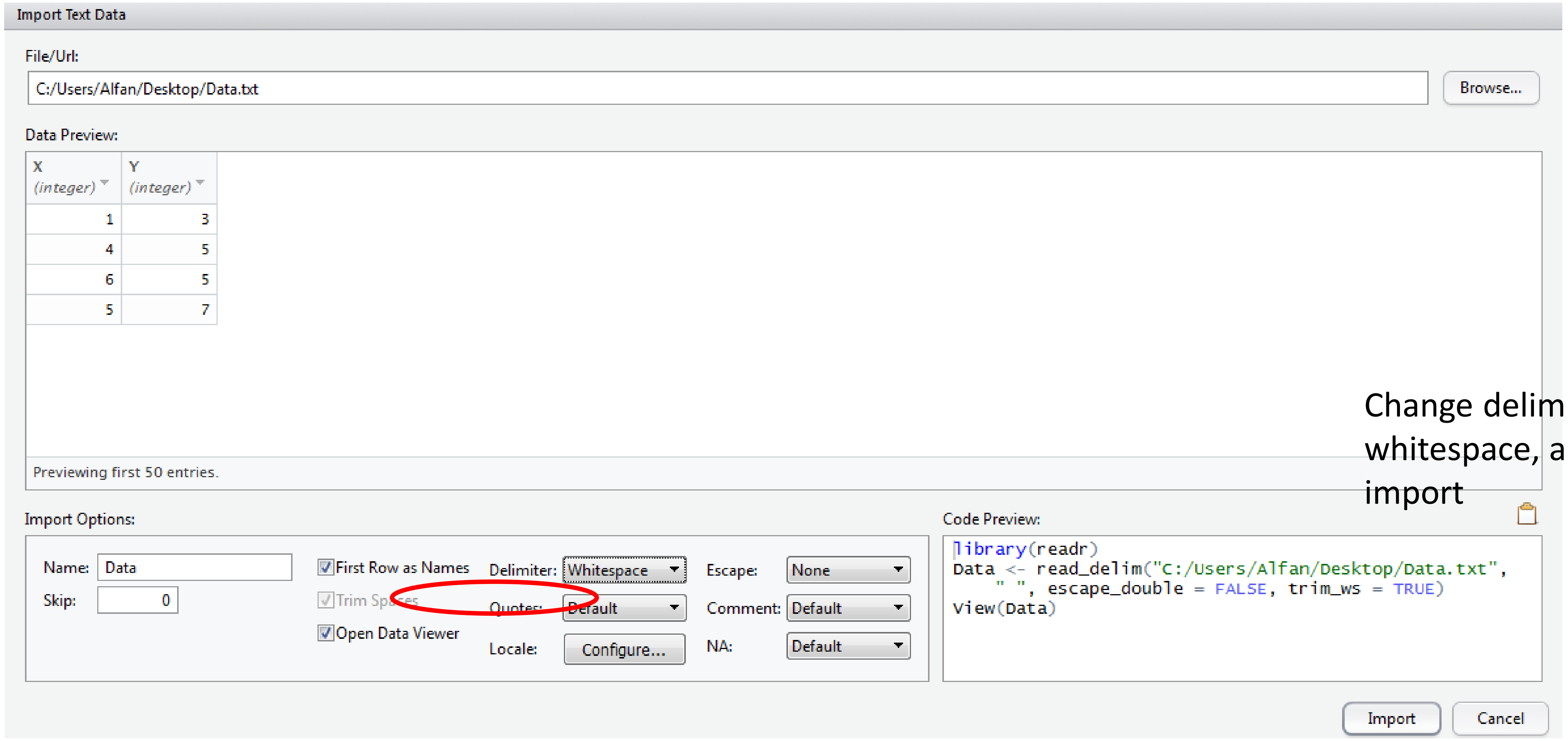
Read and Write Data



Name	Size	Modified
.RData	324.7 KB	Jun 1, 2017, 8:29 PM
.Rhistory	14.2 KB	Jul 9, 2017, 9:00 PM
Custom Office Templates		
IBM		
My Data Sources		
My ISO Files		
My Tableau Repository		
Python Scripts		
R		
SafeNet Sentinel		

Click import dataset at R Studio, choose from CSV

Read and Write Data



File/Url: C:/Users/Alfan/Desktop/Data.txt

X (integer)	Y (integer)
1	3
4	5
6	5
5	7

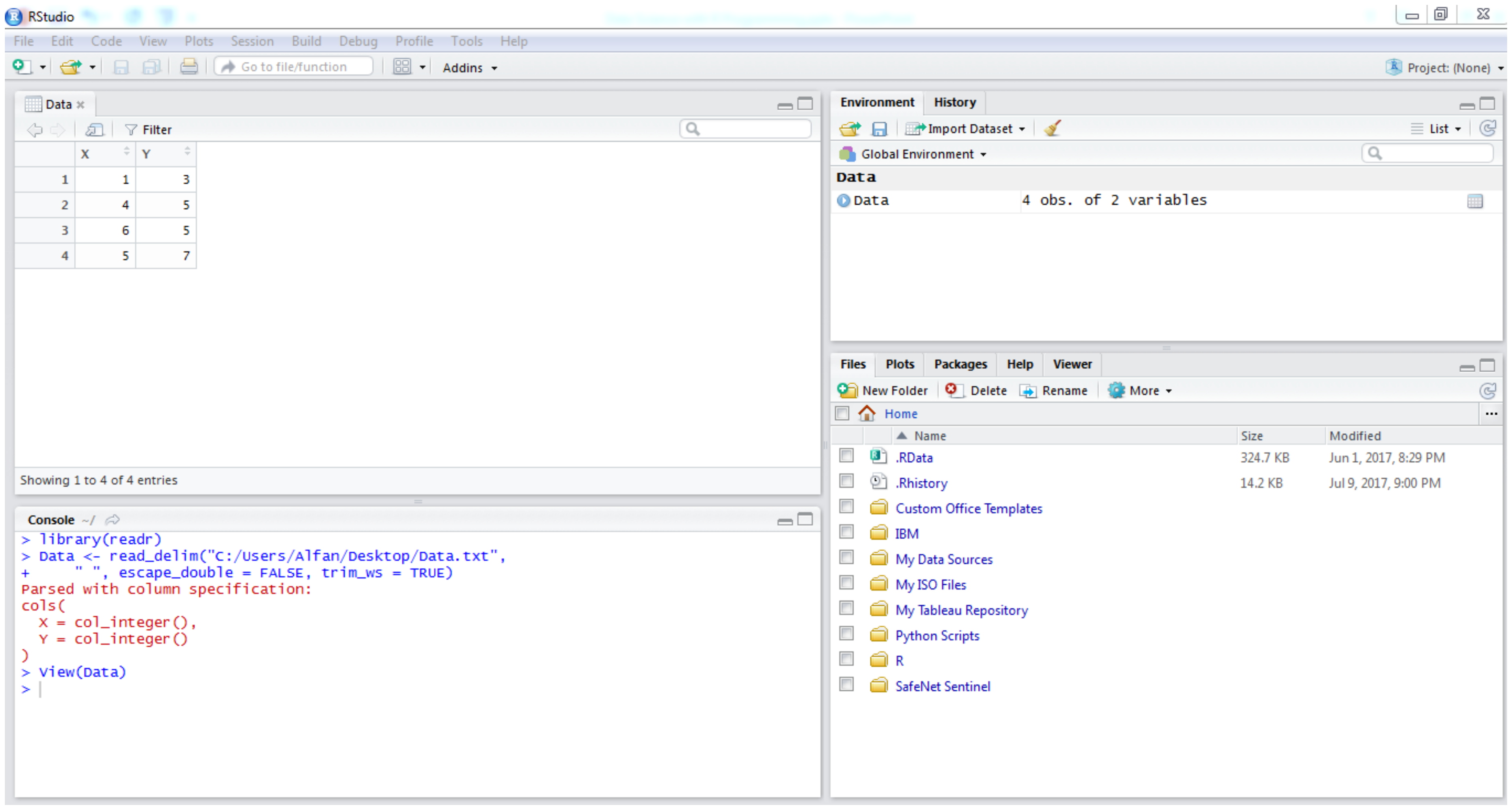
Previewing first 50 entries.

Name: Data
Skip: 0
[✓] First Row as Names
[✓] Trim Spaces
[✓] Open Data Viewer
Delimiter: Whitespace
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default

```
library(readr)
Data <- read_delim("C:/Users/Alfan/Desktop/Data.txt",
    " ", escape_double = FALSE, trim_ws = TRUE)
View(Data)
```

Change delimiter with whitespace, and click import

Read and Write Data


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Environment History
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home

	X	Y
1	1	3
2	4	5
3	6	5
4	5	7

Showing 1 to 4 of 4 entries

```
> library(readr)
> Data <- read_delim("C:/Users/Alfan/Desktop/Data.txt",
+     " ", escape_double = FALSE, trim_ws = TRUE)
Parsed with column specification:
cols(
  X = col_integer(),
  Y = col_integer()
)
> View(Data)
>
```

Data
Data 4 obs. of 2 variables

Name	Size	Modified
.RData	324.7 KB	Jun 1, 2017, 8:29 PM
.Rhistory	14.2 KB	Jul 9, 2017, 9:00 PM
Custom Office Templates		
IBM		
My Data Sources		
My ISO Files		
My Tableau Repository		
Python Scripts		
R		
SafeNet Sentinel		

Read and Write Data

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

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

Read and Write Data

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

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

Read and Write Data

Name of file

```
#Example
write.table(Data, "D:/Folder/Data.txt", sep=" ", col.names=TRUE, row.names=TRUE,
quote=FALSE, na="NA")

write.csv(Data, "D:/Folder/Data.csv", sep=" ", col.names=TRUE, row.names=TRUE,
quote=FALSE, na="NA")
```

Location file will be saved

Conditional Statement

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

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

Logical Function

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

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

Looping

for

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

while

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

repeat

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

Function

simple

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

Multiple input and return

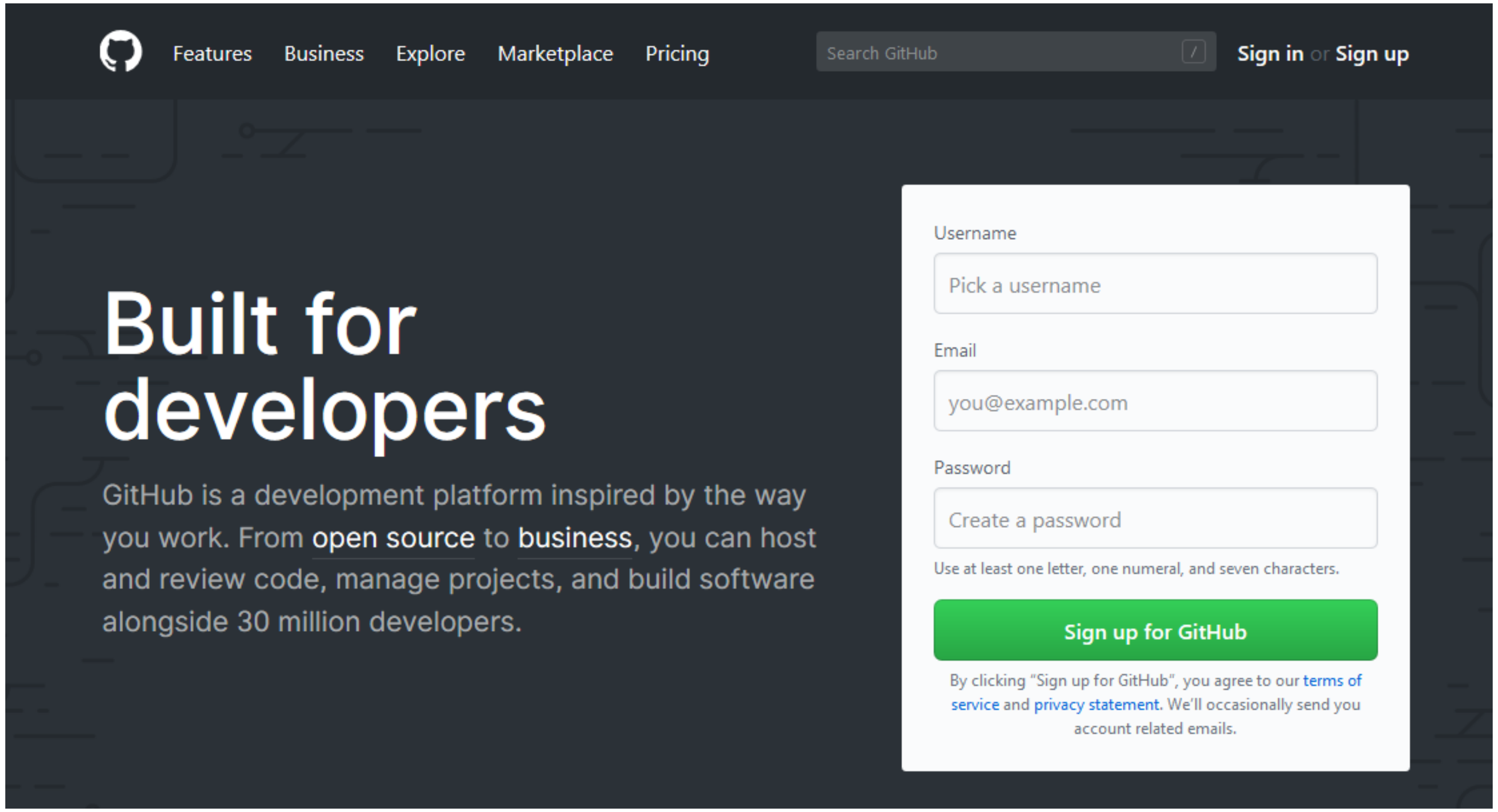
```
myfun5 <- function (x, a){
  r1 <- a* sin (x)
  r2 <- a* cos (x)
  return ( list (r1 ,r2))
}
myfun5 (2,4)
```

GitHub

DATA SCIENCE INDONESIA

What is GitHub?

code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.



Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 30 million developers.

Username: Pick a username

Email: you@example.com

Password: Create a password

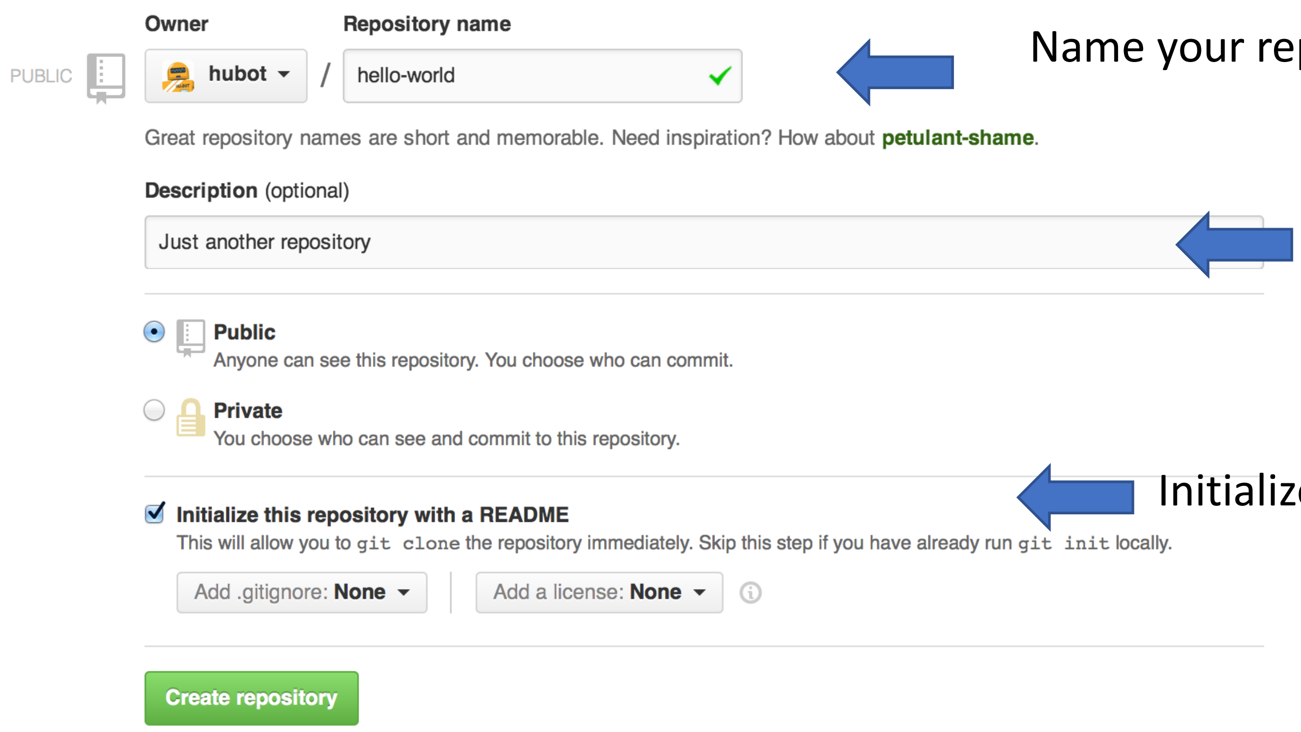
Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our terms of service and privacy statement. We'll occasionally send you account related emails.

Repository

A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.



PUBLIC

Owner: hubot / Repository name: hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional)

Just another repository

(●) **Public**
Anyone can see this repository. You choose who can commit.

( ) **Private**
You choose who can see and commit to this repository.

[✓] **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None**

Name your repository

Write a short description

Initialize this repository with a README

Branch

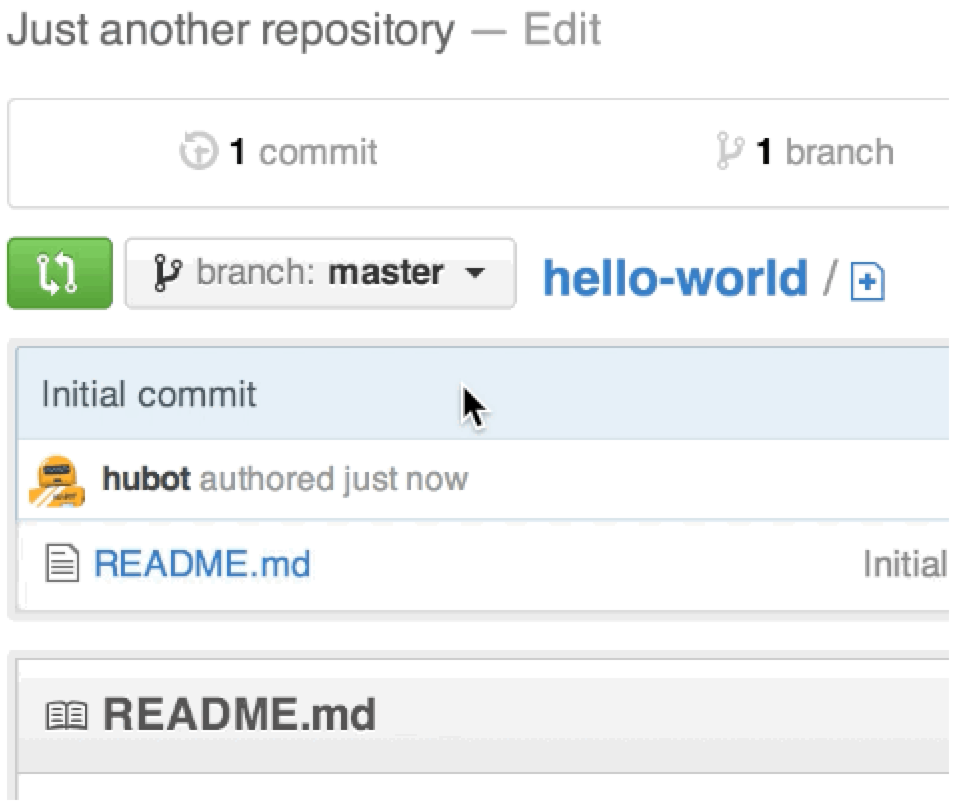
Branching is the way to work on different versions of a repository at one time.

Have you ever saved different versions of a file? Something like:

- `Laporan.docx`
- `Laporan-revisi-1.docx`
- `Laporan-final.docx`

Branches accomplish similar goals in GitHub repositories.

Branch



Just another repository — Edit

1 commit

1 branch

branch: master

hello-world /

Initial commit

hubot authored just now

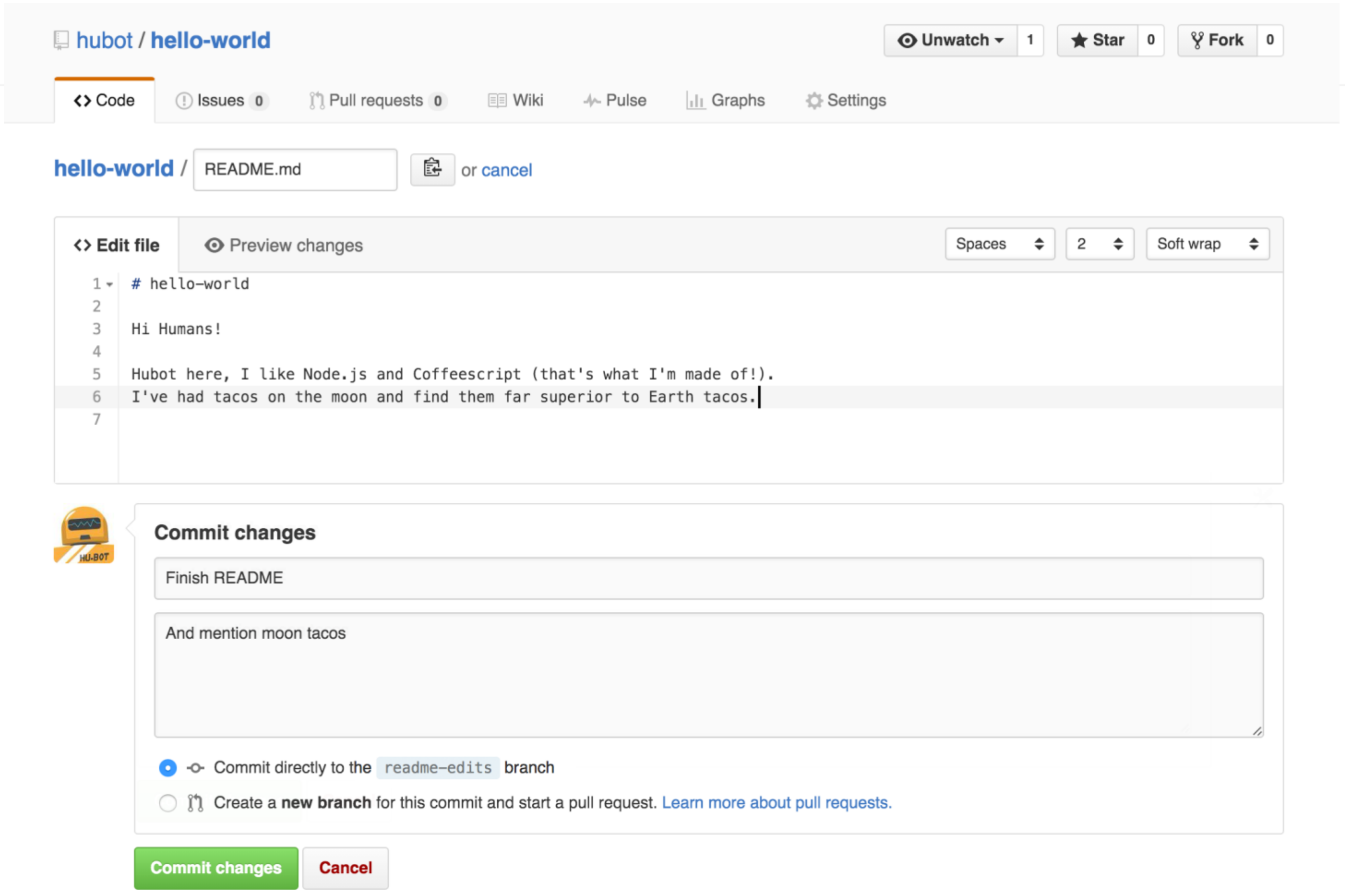
README.md

Initial

README.md

Make and commit changes

On GitHub, saved changes are called *commits*.



```
# hello-world

Hi Humans!

Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
I've had tacos on the moon and find them far superior to Earth tacos.
```

Commit changes

Finish README

And mention moon tacos

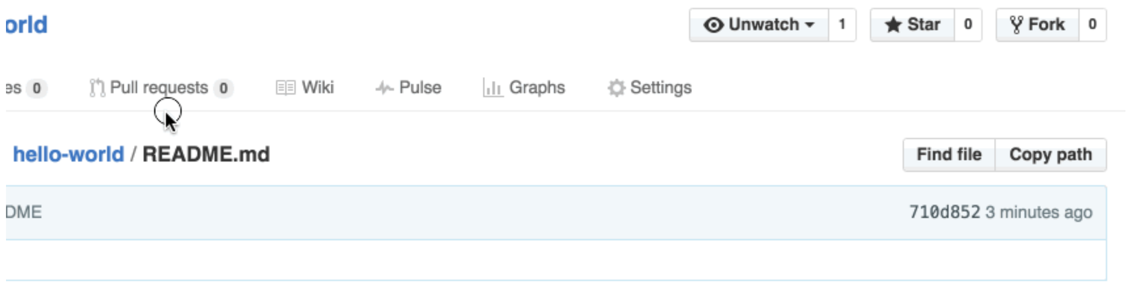
[✓ Commit directly to the `readme-edits` branch]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

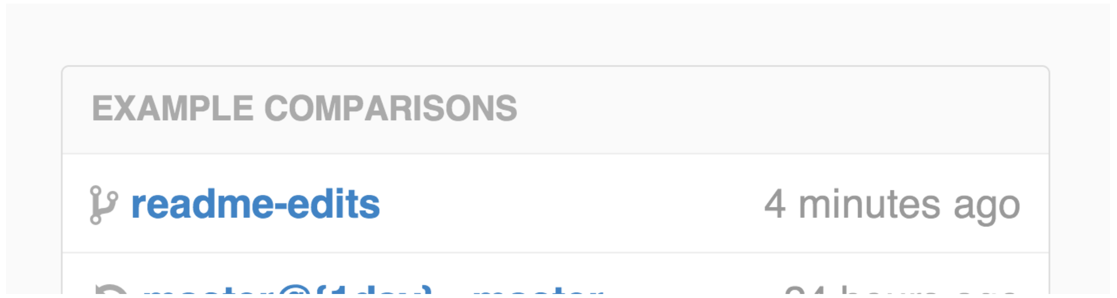
These changes will be made to just the README file on your branch, so now this branch contains content that's different from master.

Pull Request

When you open a ***pull request***, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch.

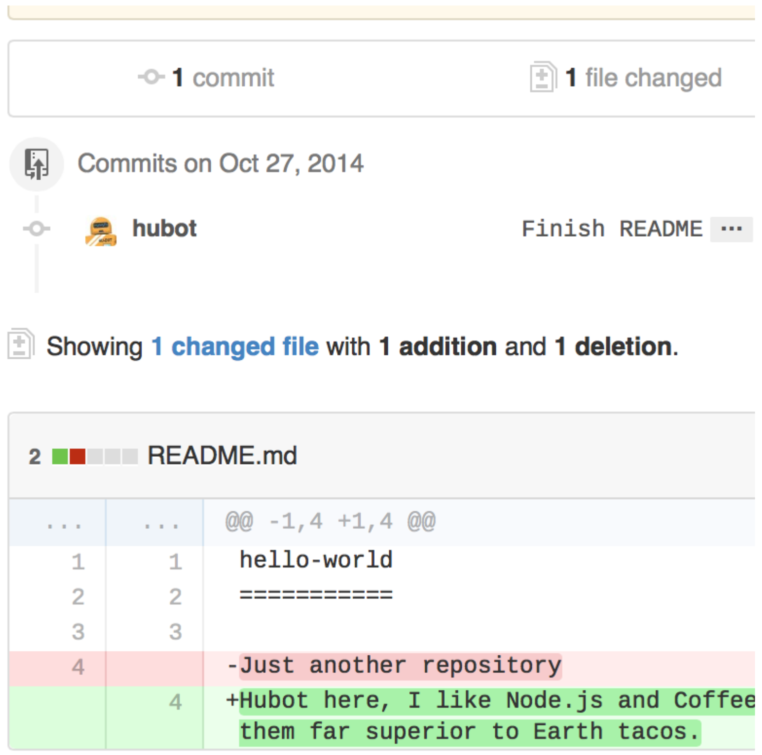


click the green **New pull request**



in the **Example Comparisons** box, select the branch you made, to compare with master (the original).

Pull Request


1 commit   1 file changed

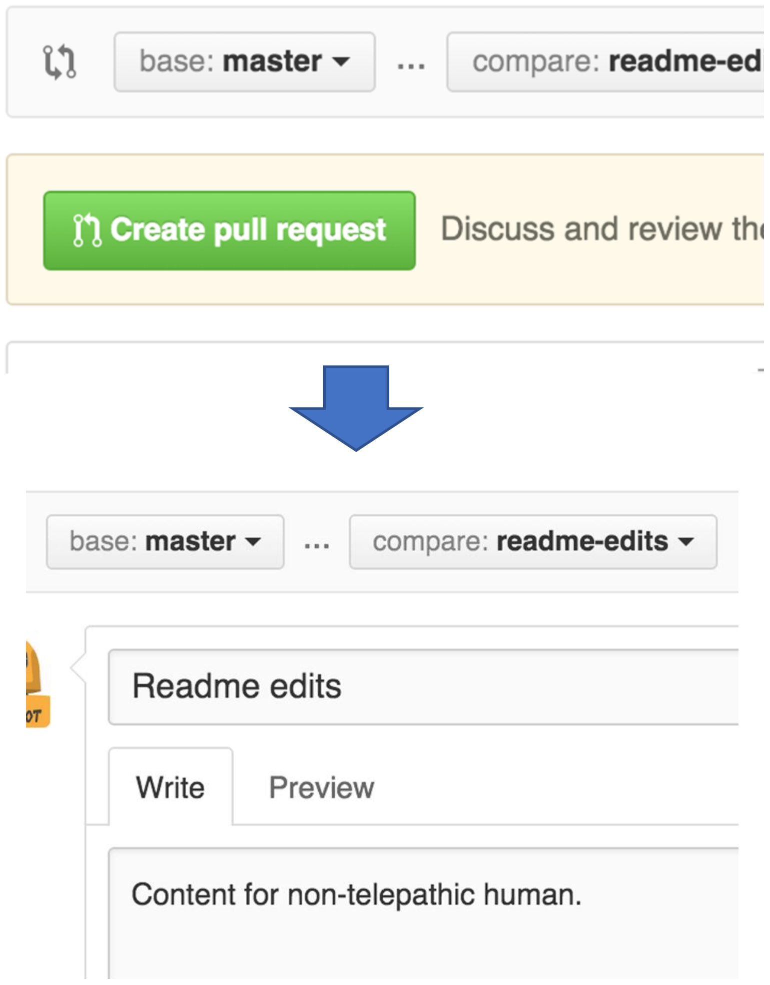
Commits on Oct 27, 2014

hubot   Finish README

Showing 1 changed file with 1 addition and 1 deletion.

2 README.md

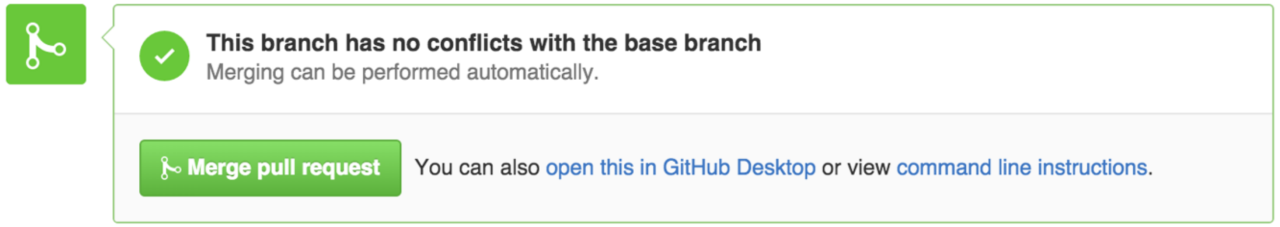
```
@@ -1,4 +1,4 @@
 hello-world
 ===========

-Just another repository
+Hubot here, I like Node.js and Coffee
them far superior to Earth tacos.
```



Look over your changes in the diffs on the Compare page

Merge your Pull Request

In this final step, it's time to bring your changes together – merging your **branch** into the **master branch**.


This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

DATA
SCIENCE
INDONESIA

Thank You