# MSiA 400 Lab Introduction to R

Sep 29, 2014

Young Woong Park

# What is R?

- R is a language and environment for statistical computing and graphics

- Comparable tools: Matlab, SAS, SPSS, Stata

- Distinguished features from other tools: Open source
  - FREE to download and use
  - Source code is available
  - Packages

# For Your Laptop

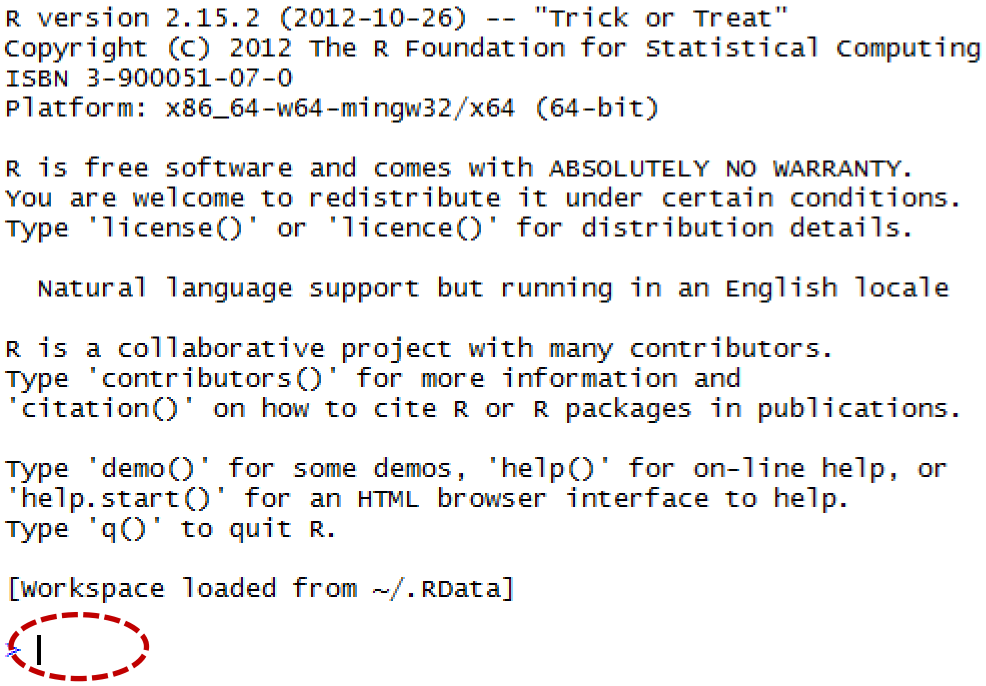
- R is available from: www.r-project.org/
- Eclipse is an Integrated Development Environment(IDE)
  - Available from http://www.eclipse.org/
- StatET is an Eclipse based IDE for R.
  - Available from http://www.walware.de/goto/statet
- R Studio is another popular IDE for R
  - Available from http://www.rstudio.com/ide/download/

# Simple Calculation Using R

- Try   > 5+3   or other basic operations

  > 5^3

  > 5/3



```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-w64-mingw32/x64 (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.

[Workspace loaded from ~/.RData]

> |
```

# Basic Syntax

- Syntax is very similar to other languages

- Basic unit is command
  - Commands are separated by a semi-colon (;) or a new line

    ```
    > 3+4;3^4;
    > 3+4
    > 3^4
    ```

  - Commands can be grouped together using braces ({ and })

- Assignment of values: two symbols can be used interchangeably

  ```
  > Result1 <- 3+5
  > Result2 = 3+5
  ```

# Data Type

- Numerical data: 1, 2, 3

  ```
  > MyNum <- 400
  ```

- Character (or String) data: “a”, “analytics”

  ```
  > MySchool <- “Northwestern”
  ```

  * Do not copy and past from the slides! It may give you an error message (double quotes)

# Vectors and Matrices (1)

- Creating vectors
  - Numerical vector

    ```
    > MyNumVector <- c(1,2,3)
    ```
  - String vector

    ```
    > MyStrVector <- c(“MS”, “Analytics”, “Northwestern”)
    ```
  - Sequence of numbers

    ```
    > MyVec <- seq(1,10)
    > MyVec2 <- seq(5,1)
    > MyVec3 <- seq(25,30)
    ```
  - Vector of zeros or ones

    ```
    > MyZeroVec <- rep(0,10)
    > MyOneVec <- rep(1,20)
    ```

- Length of vector

  ```
  > length(MyVec)
  ```

# Vectors and Matrices (2)

- Creating matrix

```
> MyMat1  <- matrix(c(1,2,3,4,5,6), nrow=3)
> MyMat2  <- matrix(c(1,2,3,4,5,6), nrow=3, byrow=T)
> MyMat3  <- matrix(1:8, nrow=4)
```

- Dimension of matrix

```
> dim(MyMat1)
> dim(MyMat3)
```

# Vectors and Matrices (3)

- Referencing elements of vector

```
> MyVec[3]
> MyVec[5:7]
```

- Assigning values to vector

```
> MyVec[5] <- 50
> MyVec[9:10] <- 1000
```

- Referencing elements of matrix

```
> MyMat1[1,2]
> MyMat1[1:2,1:2]
```

- Assigning values to matrix

```
> MyMat1[1,2] = 50
> MyMat1[1:2,1:2] = 1000
```

# Clearing Workspace

- Clear data or matrix from the memory

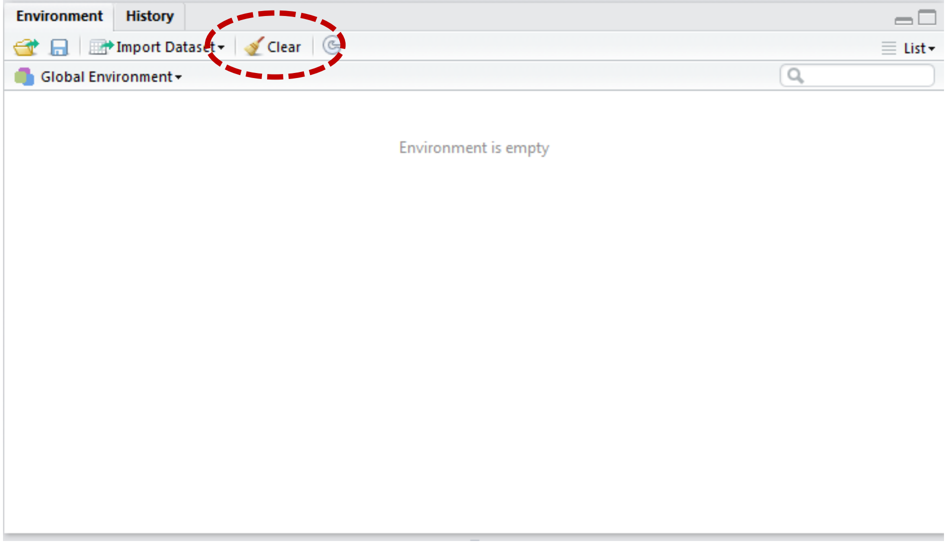
```
> rm(MyMat1)
> rm(MyMat3)
```

- List Object

```
> ls()
```

- Clear everything in the workspace

```
> rm(list = ls())
```



Environment is empty

# Simple Calculations

- Let’s define vectors and variables

```
> X <- c(1,3,5,7,9); Y <- c(10, 20,30,40,50); A <- 2; B <- 4;
```

- Simple calculations
  - Summation

    ```
    > C <- A + B
    > sum(X)
    > Z <- X + Y
    ```

  - Partial summation

    ```
    > sum(X[3:5])
    ```

  - Average and standard deviation

    ```
    > mean(X)
    > sd(X)
    ```

  - Squared root

    ```
    > sqrt(C)
    > sqrt(sum(Z))
    ```

# Exercise 1

- Step1 Create vectors of following sets

  Set1 = {1,2,3,4,…,10}

  Set2 = {101,102,…,110}
- Step2 Calculate the average of each set

  and save the two values into a vector
- Step3 Display the vector

# Exercise 1

- Step1 Create vectors of following sets

  Set1 = {1,2,3,4,…,10}
  Set2 = {101,102,…,110}

- Step2 Calculate the average of each set and save the two values into a vector
- Step3 Display the vector

- Solution

```
> Set1 <- seq(1,10);
> Set2 <- seq(101,110);
> SetAvg <- c(mean(Set1),mean(Set2));
> SetAvg;
```

Note: To print,

1. SetAvg;
2. print(SetAvg);

# Reading Data From Text File

- Change your working directory: Session > Set Working Directory

```
> setwd(“c:/mywork”);
> getwd();
```

- Text files ex_header.txt and ex_no_header.txt

  - Contain 6 observations of “age” and “gpa”
  - ex_header.txt contains header information

- Create four tables and compare the results

```
> tb1 <- read.table(“z:\\ msia400lab1 \\ex_header.txt”, header=T)
> tb2 <- read.table(“z:\\ msia400lab1 \\ex_no_header.txt”, header=F)
> tb3 <- read.table(“z:\\ msia400lab1 \\ex_header.txt”, header=F)
> tb4 <- read.table(“z:\\ msia400lab1 \\ex_no_header.txt”, header=T)
```

* default value for header is False

- Check the difference between the tables

# Reading Data From Text File (Cont.)

ex_header.txt

```
age     gpa
25      3.5
24      3.8
21      3.4
22      3.9
23      3.2
20      3.3
21      3.7
```

ex_no_header.txt

```
25      3.5
24      3.8
21      3.4
22      3.9
23      3.2
20      3.3
21      3.7
```

tb1

```
  age gpa
1  25 3.5
2  24 3.8
3  21 3.4
4  22 3.9
5  23 3.2
6  20 3.3
7  21 3.7
```

tb2

```
  v1  v2
1 25 3.5
2 24 3.8
3 21 3.4
4 22 3.9
5 23 3.2
6 20 3.3
7 21 3.7
```

tb3

```
   v1  v2
1 age gpa
2  25 3.5
3  24 3.8
4  21 3.4
5  22 3.9
6  23 3.2
7  20 3.3
8  21 3.7
```

tb4

```
  x25 x3.5
1  24  3.8
2  21  3.4
3  22  3.9
4  23  3.2
5  20  3.3
6  21  3.7
```

```
> tb1 <- read.table(“z:\\ msia400lab1 \\ex_header.txt”, header=T)
> tb2 <- read.table(“z:\\ msia400lab1 \\ex_no_header.txt”, header=F)
> tb3 <- read.table(“z:\\ msia400lab1 \\ex_header.txt”, header=F)
> tb4 <- read.table(“z:\\ msia400lab1 \\ex_no_header.txt”, header=T)
```

# Reading Data From DB

- We can access to databases (such as MS Access and SQL Server) using a package

- It will be covered in the future.

# Calculation from Table

- Similar syntax with vectors and matrices
  - Mean of the first column

    ```
    > mean(tb1[ ,1])
    ```

  - Sum of the second column from row 1 to 3

    ```
    > sum(tb1[1:3,2])
    ```

- Referencing using column name
  - Mean of the first column

    ```
    > mean(tb1$age)
    ```

  - Sum of the second column from row 1 to 3

    ```
    > sum(tb1$gpa[1:3]]
    ```

# Writing Data From Table

- Change your working directory: File > Change dir

- Create four tables and compare the results

```
> write.table(tb1,“z:\\ msia400lab1 \\ex_write1.txt”, sep=“\t”)
> write.table(tb1,“z:\\ msia400lab1 \\ex_write2.txt”, sep=“\t”, row.names=F)
> write.table(tb1,“z:\\ msia400lab1 \\ex_write3.txt”, sep=“\t”, col.names=F)
```

- Check the difference between the text files

# Functions and Looping

- Create functions to compute the same thing for several data sets when no implementation or package is available
- We have seen several functions: mean, sd, sum
- We are going to implement our own function
- Format of function

| Basic Format | Example |
|---|---|
| function_name <- function(arg1,arg2,…){<br>command1;<br>command2;<br>…<br>output<br>} | MyFunc <- function(x){<br>variance <- sd(x)^2;<br>answer <- variance + sum(x);<br>answer<br>} |

- Use of function: > MyFunc(x)

# Functions and Looping (Cont.)

| Type | If statement | For | While |
|---|---|---|---|
| example | if (number != 1){<br>    result = F;<br>}else{<br>    result = T;<br>} | For(i in 1:length(x)){<br>    sum = sum + x[i];<br>} | i <- 1<br>while(i <= length(x)){<br>    sum <- sum + x[i];<br>    i <- i+1;<br>} |

- Exercise 2 From the table tb1, create a function that calculates average gpa of people older than 21

# Functions and Looping (Cont.)

| Type | If statement | For | While |
|---|---|---|---|
| example | if (number != 1){<br>result = F;<br>}else{<br>result = T;<br>} | For(i in 1:length(x)){<br>sum = sum + x[i]<br>} | i <- 1<br>while(i <= length(x)){<br>sum <- sum + x[i]<br>i <- i+1<br>} |

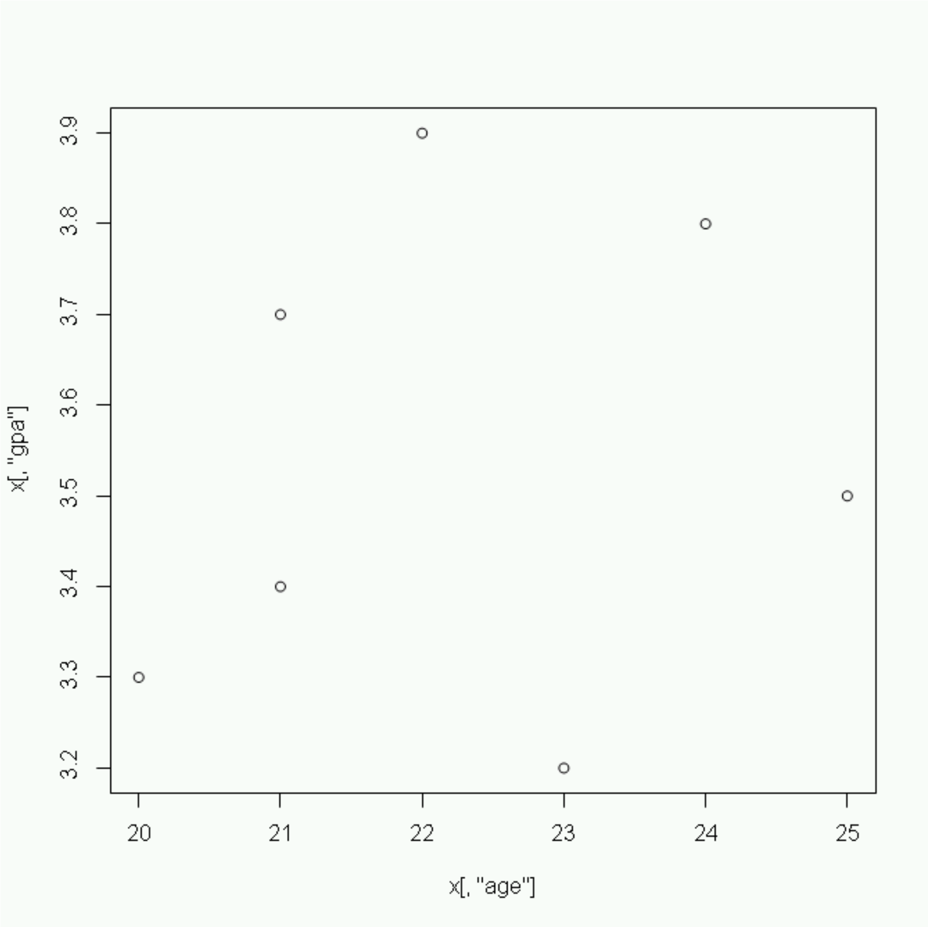
- Exercise 2 From the table tb1, create a function that calculates average gpa of people older than 21

```
myAvg <- function(mytb){
           sum = 0;  cnt = 0;
           for(i in 1:length(mytb[,1])){
                      if(mytb[i,1]>21){
                                 sum = sum + mytb[i,2];
                                 cnt = cnt + 1;
                      }
           }
           avg = sum / cnt;
           print(avg);
}
```

# Simple Plotting

- Scatter Plot

```
> plot(tb1[,“age”],tb1[,“gpa”])
```

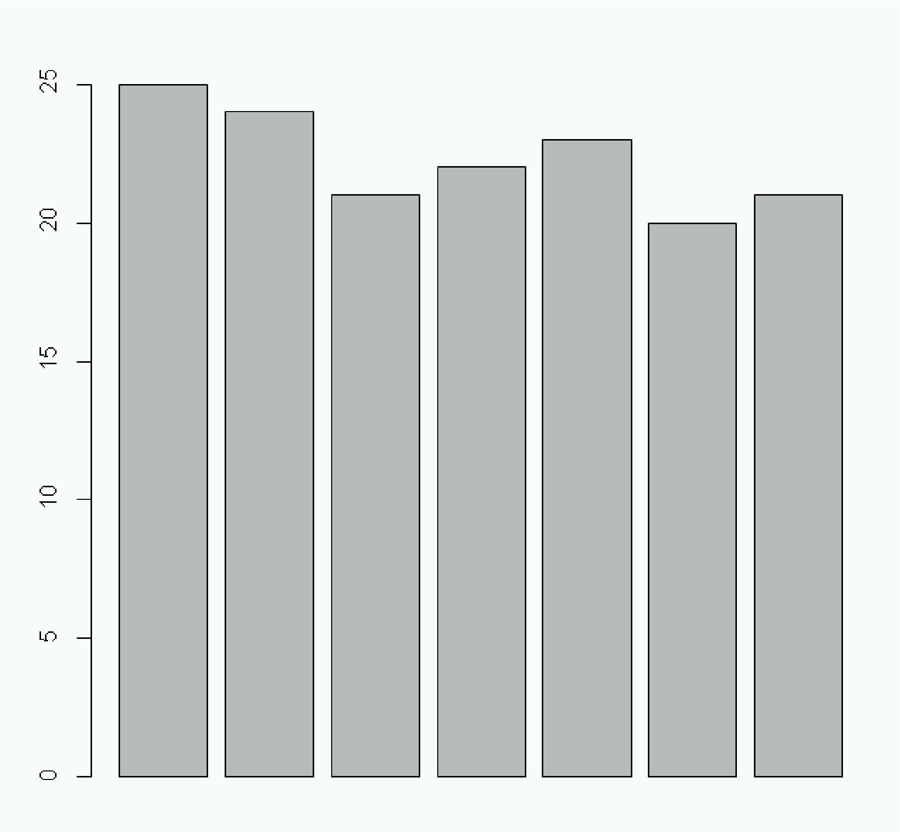


```
> plot(tb1[,1],tb1[,2])
> plot(tb1$age,tb1$gpa)
> plot(tb1)
```

# Simple Plotting

- Bar Plot

  ```
  > barplot(tb1[,“age”])
  ```



# Exercise 3

- Step1 Import data set from height.txt and make a table

  The data contains the average heights of men and women of 71 countries

  * data is from Wikipedia

- Step2 Create a function
  - ○ Calculating the  world averages of (1) men (2) women
  - ○ Generating scatter plot of men’s height over women’s height

# Exercise 3

- Step1 Import data set from height.txt and make a table

  The data contains the average heights of men and women of 71 countries

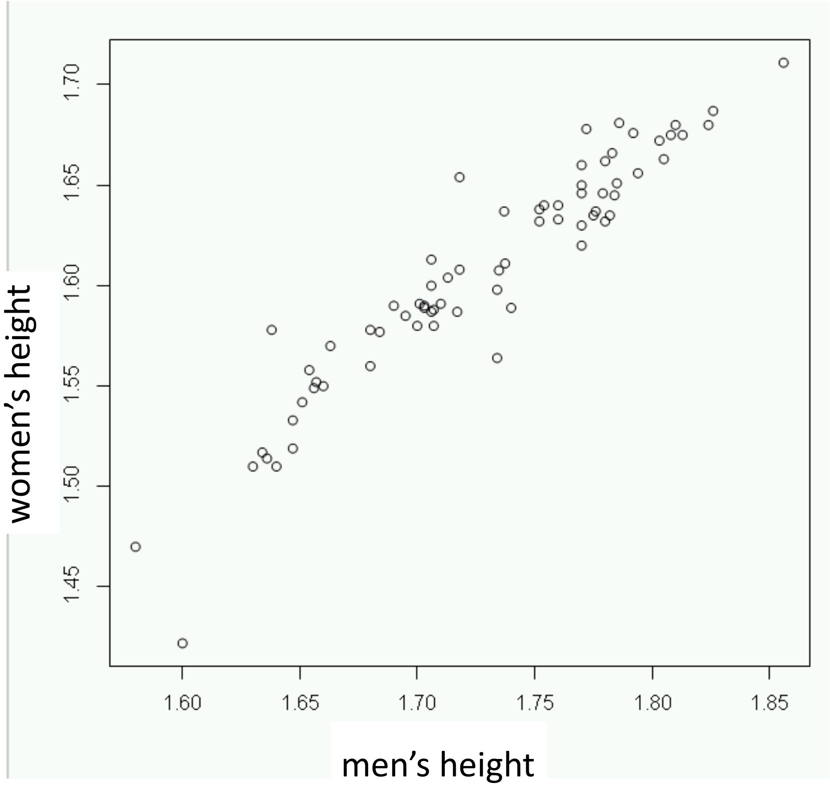
  * data is from Wikipedia

- Step2 Create a function
  - Calculating the world averages of (1) men (2) women
  - Generating scatter plot of men's height over women's height

- Solution

```
myfunc3 <- function(tb){
         myvec <- c(0,0);
         myvec[1] = mean(tb[,1]);
         myvec[2] = mean(tb[,2]);
         print(myvec);
         plot(tb[,1],tb[,2]);
}
```

# Plot from Exercise 3



- Heights of men and women look correlated. How can we analyze?
    1. Covariance
    2. Regression

# Covariance

- Variance
  > var(height[,1])

- Correlation Coefficient : $cor = \frac{Cov(x,y)}{\sqrt{var(x)var(y)}}$
  > cor(height[,1],height[,2])
  > cor(height)

- Covariance : $\mathrm{cor}\sqrt{var(x)var(y)}$
  > try!

# Linear Regression

- Regression model for EX3

Women’s height = $\beta_0$+$\beta_1$*men’s height

- Building linear regression model

```
> myReg <- lm(height[,2] ~ height[,1]);
```

- Summary stats and plot with fitted line

```
> summary(myReg)
> plot(height[,1],height[,2]);
> abline(coef(myReg));
```

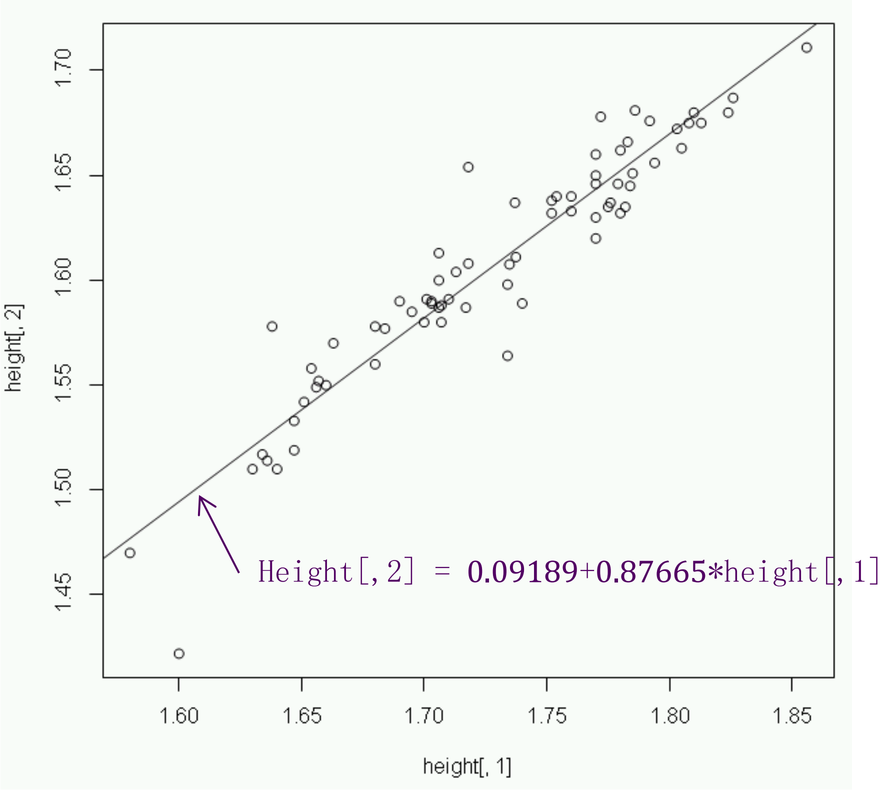
# Linear Regression (Cont.)

```
Call:
lm(formula = ht[, 2] ~ ht[, 1])

Residuals:
     Min        1Q    Median        3Q       Max
-0.072529 -0.009347 -0.000376  0.009851  0.056027

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.09189    0.06278   1.464    0.148
ht[, 1]      0.87665    0.03631  24.144   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01846 on 69 degrees of freedom
Multiple R-squared: 0.8942,   Adjusted R-squared: 0.8926
F-statistic: 582.9 on 1 and 69 DF,  p-value: < 2.2e-16
```



Height[,2] = 0.09189+0.87665*height[,1]

Women’s height = $\beta_0$+$\beta_1$*men’s height

⇩

Women’s height = 0.09189+0.87665*men’s height

# Packages

- "R" contains libraries of packages
  - Packages contain functions and data sets
  - Some packages are part of the basic installation. (mean, lm, var, etc.)
  - Others can be downloaded from CRAN.

- Installing a package: currently, you are not allowed to install at the server

  ```
  > install.packages("PackageName")
  ```

- Note: to use a function in a package,

  you must load the package to the workspace before use