# IMT 589 D
# Special Topics in Information Management

## Introduction to Data Sciences

# A Tutorial on R

By
Muhammad Raza Khan
mraza@uw.edu

# Disclaimer

- This tutorial is highly inspired by
  - Tutorial "Introduction to R" by

    Guy Yollin

    Principal Consulant, r-programming.org

    Affiliate Instructor, University of Washington
  - Seminar "Introducing R"

    Statistical Consulting Group
  - UCLA Institute for Digital Research and Education

    Introduction to R

    Workshop in Methods from Indiana Statistical Computing Center

# Background of R

- R: Language and Environment for Statistical Computing and Analysis
- Based on S. Also called as GNU S
- Robert Gentleman and Ross Ihaka, University of Auckland, NZ initiated the development of R
- R: Formally known as R Project for Statistical Computing (www.r-project.org)

# What can be done with R

- Data Manipulation
- Data Analysis
- Statistical Modeling
- Data Visualization

# Working in R

- Installation
  - Installer available at http://cran.r-project.org/ for MAC, Windows and Linux
- R can be compiled ( or interpreted) from the command line
- R studio : Full Fledge IDE for R Development
  - www.rstudio.com

# Assigning Values to Variables

- operator <- or ->
  - x<-5
  - 5->x
- assign function
  - assign("pi",3.14)
- equal sign
  - rn <- rnorm(n=2)
- = and <- can be interchanged except for (passing arguments to a function)

# Printing Values of Variables

- print ( variable_name)
- print ( function(args))
- variable_name

  - x<-5
  - x ( ENTER)
  - assign(“pi”,3.14)
  - print(pi)

# Objects & Data Types

- Every thing in R is an object
- ls() and objects() list all objects in the current workspace
- All R objects have a type or storage mode
- Function typeof() tells the type of an object

  - x<- 5.4
  - typeof(x)

  [1] “double”

  - typeof(“x”)

  [1] “character”

# Object Classes in R

- All R objects have a class
- Function class to display an object's class
- Basic classes in R
  - numeric
  - character
  - data.frame
  - matrix

# Vectors in R

- R is optimized for vector/ matrix operations
- Single variables/ values and Vectors can be used interchangeably most of the time
- c function is used to create vectors
- After creation of the vector members can be accessed through indices. *Indices in R start from 1*

- constants <- c(3.14,1.6)

- constants[1]

[1] 3.14

# Vectors in R

- More than one indices can be passed in []
- Vectors can grow through [] as well
- Names can also be assigned to member of the vectors through the name function

- constants[1:2]

[1] 3.14 1.6

- constants[3]<-9.8
- constants

[1] 3.14 1.6 9.8

- names(constants)<-c ("pi", "gr", "g")
- constants["pi"]

[1] 3.14

# More on Vectors Creation

- Vectors can create objects of not only numeric values but other types as well

```
x <- c(0.5, 0.6) ## numeric
x <- c(TRUE, FALSE) ## logical
x <- c(T, F) ## logical
x <- c("a", "b", "c") ## character
x <- 9:29 ## integer
x <- c(1+0i, 2+4i) ## complex
```

- ## indicates comments

# Mixing types in Vector creation

- What would happen if

➢ v<- c(“c”,1)

- Coercion occurs in this case which results in all of the objects being of the same type

➢ v<- c(“c”,1)

[1] “c” “1”

# Mathematical Operation on Vectors

- +, -, *, / can be applied just like normal values (Assuming that vector has numeric data in it)
  - vec<-c(0,1,2,5,7)
  - vec+5

```
[1]  5  6  7 10 12
```

  - vec*5

```
[1]  0  5 10 25 35
```

  - vec/2

```
[1] 0.0 0.5 1.0 2.5 3.5
```

  - vec^2

```
[1]  0  1  4 25 49
```

  - vec2<-c(1,2,3,5,7)
  - vec*vec2

```
[1]  0  2  6 25 49
```

# Mathematical Operations on Vectors

- What if the two vectors being multiplied are not of equal length
  - Recycling ( The shorter vector is recycled or expanded to the length of the longer vector)

R Code: Illustration of recycling

```
> constants

      pi    euler    sqrt2  goldden
3.1416  2.7183  1.4142  1.6180

> constants*2

      pi    euler    sqrt2  goldden
6.2832  5.4366  2.8284  3.2360

> constants*c(0,1)

      pi    euler    sqrt2  goldden
0.0000  2.7183  0.0000  1.6180

> constants*c(0,1,2)

      pi    euler    sqrt2  goldden
0.0000  2.7183  2.8284  0.0000
```

last input generates a warning: longer object length is not a multiple of shorter object length

# Sequences

- Integer sequence vector can be created through : operator
- For the cases, where Stepping factor may not be one seq function is also available

R Code: seq arguments

```
> args(seq.default)

function (from = 1, to = 1, by = ((to - from)/(length.out - 1)),
    length.out = NULL, along.with = NULL, ...)
```

```
> -5:5
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
> seq(from=0, to=1, len=5)
[1] 0.00 0.25 0.50 0.75 1.00
> seq(from=0, to=20, by=2.5)
[1]  0.0  2.5  5.0  7.5 10.0 12.5 15.0 17.5 20.0
```

# Arguments to a Function

- Unnamed arguments to a function are assigned according to their position
- Positioning of named arguments can be changed

R Code: Illustration of flexibility in passing arguments

```
> seq(0, 10, 2)
[1]  0  2  4  6  8 10
> seq(by=2, 0, 10)
[1]  0  2  4  6  8 10
> seq(0, 10, len=5)
[1]  0.0  2.5  5.0  7.5 10.0
> seq(0, 10)
 [1]  0  1  2  3  4  5  6  7  8  9 10
```

# Rep Function

- Create or init vectors by repeating data
- help(rep)

R Code: Examples of rep

```
> rep(0,10)    # initialize a vector
 [1] 0 0 0 0 0 0 0 0 0 0
> rep(1:4, 2) # repeat pattern 2 times
[1] 1 2 3 4 1 2 3 4
> rep(1:4, each = 2)  # repeat each element 2 times
[1] 1 1 2 2 3 3 4 4
> rep(1:4, c(2,1,2,1))
[1] 1 1 2 3 3 4
> rep(1:4, each = 2, len = 10)    # 8 integers plus two recycled 1's.
 [1] 1 1 2 2 3 3 4 4 1 1
> rep(1:4, each = 2, times = 3)  # length 24, 3 complete replications
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

# Packages in R

- Similar Functions and datasets are stored in packages
  - Core R packages
  - Recommended Packages
  - Additional Packages
- When R is initially loaded only the functions from the core packages are included in the workspace
  - install.packages function installs the package ( it can download the package from the Internet as well)
  - Installed packaged can be loaded through the library function
  - Datasets from the installed packages can be loaded through data function

# Installing Contributed Packages

R Code: The install.packages function

```
> args(install.packages)

function (pkgs, lib, repos = getOption("repos"), contriburl = contrib.url(repos,
    type), method, available = NULL, destdir = NULL, dependencies = NA,
    type = getOption("pkgType"), configure.args = getOption("configure.args"),
    configure.vars = getOption("configure.vars"), clean = FALSE,
    Ncpus = getOption("Ncpus", 1L), libs_only = FALSE, INSTALL_opts,
    ...)
NULL

> #install.packages("nutshell")
> # or if repos needs to be specified
> #install.packages("nutshell", repos="http://cran.fhcrc.org")
```

# Data Manipulation in R

- Working Directory
  - R reads from and writes to the working directory
  - getwd() and setwd() functions can be used to get and set working directories

R Code: Getting and setting the working directory

```
> getwd()

[1] "C:/Rprojects/UW/Lecture-01"

> setwd("C:\\Rprojects\\PCA")
> getwd()

[1] "C:/Rprojects/PCA"

> setwd("C:/Rprojects/UW/Lecture-01")
> getwd()

[1] "C:/Rprojects/UW/Lecture-01"
```

# Reading a Text File

- help(read.table)
- For simple space separated files read.table("filename") should work
- Class Activity 1
  - Create a text files having 3 rows and 3 columns with dummy data
  - Call read.table function to load the data from the file and then print it over the R shell

# Reading a CSV File

- help(read.csv)
- Activity 2
  - Read the csv from the URL (http://kanspra.org/memberdirectory.csv)
  - What is the impact of passing header=TRUE and header =FALSE as arg to the read.csv function
- DIY Activity
  - Load a data file having numeric files
  - Calculate square of each value and save the squared values in a different data file
  - Hint: write.table or write.csv
- Question: Can we read the csv file through read.table function?

# Analyzing the Loaded Data

- mydata<-read.table("input.txt", header=FALSE)
- typeof(mydata)

[1] “list”

- class(mydata)

[1] data.frame

Read.data and Read.csv return a data.frame object which is sort of a matrix having columns of different types

# Analyzing the Loaded Data

| Function | Description |
|---|---|
| dim | return dimensions of a multidimensional object |
| nrow | number of rows of a multidimensional object |
| ncol | number of columns of a multidimensional object |
| length | length a vector or list |
| head | display first n rows (elements) |
| tail | display last n rows (elements) |
| str | summarize structure of an object |

# List Object in R

- A container that can hold objects of different types
  - obj< - list(name="k1", value=35)
- Accessing elements of a list
  - [] returns a sublist
    - obj["name"]

      $name

      [1] "k1
  - [[]] return a single element
    - Either name or index can be specified
  - $ returns a single element
    - obj$name

# Function to get Details of an Object

| | |
|---|---|
| class | query an objects class |
| str | reports structure of an object |
| tributes | returns list of objects attributes |
| attr | get/set attributes of an object |
| names | gets the names of a list, vector, data.frame, etc. |
| dimnames | gets the row and column names of a data.frame or matr |
| colnames | column names of a data.frame or matrix |
| rownames | row names of a data.frame or matrix |

# Matrices

- Matrix Function can be used to create matrices in R
  - m<- matrix(nrow=2,ncol=5)
  - m

```
     [,1] [,2] [,3] [,4] [,5]
[1,]   NA   NA   NA   NA   NA
[2,]   NA   NA   NA   NA   NA
```

  - m2<-matrix(1:4,nrow=2,ncol=2) ## First arg is specifying data now
  - m2

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

- cbind and rbind functions can be used to assign columns and row values to a matrix after creating it

# Apply Function and Its Variants

- apply: Apply a function over the margins of an array
- lapply: Loop over a list and evaluate a function on each element
- sapply: Same as lapply but try to simplify the result
- tapply: Apply a function over subsets of a vector

# Apply Function

- Applies a function to margins of a matrix i.e. either on ROW OR COLUMN of a matrix or array
- Usage
  - apply(X, MARGIN, FUN, ...)
  - The value 1 for margin specifies row-wise operation and 2 specifies column wise operation
  - set.seed(1)
  - m2<-matrix(sample(100),nrow=2,ncol=2)
  - apply(m2,2,sum)

# Lapply and sapply functions

- Sapply tries to simplify the results of lapply by converting the results to a vector if possible otherwise it returns a list

```
> x<-list(a=1:4, b=6:10, c=-3:4)
> lapply(x,mean)
$a
[1] 2.5
$b
[1] 8
$c
[1] 0.5
> sapply(x,mean)
  a   b   c
2.5 8.0 0.5
```

# Tapply

- Apply a function to a substring based on unique combination of levels of factors

➢ levels(CHFLS$R_health)

[1] "Poor" "Not good" "Fair" "Good" "Excellent"

➢ tapply(CHFLS$R_income, CHFLS$R_health, mean)

Poor Not good Fair Good Excellent

338.0000 419.4245 649.0456 645.4639 613.8304

# Basic Stats Function Available in R

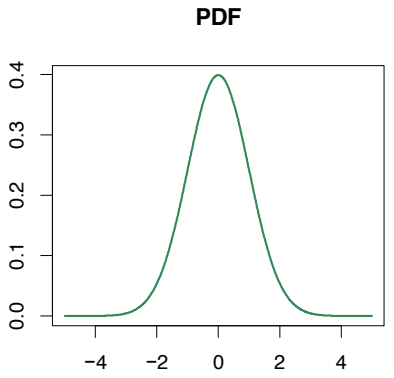
| | |
|---|---|
| mean | mean of a vector or matrix |
| median | median of a vector or matrix |
| mad | median absolute deviation of a vector or matrix |
| var | variance of a vector or matrix |
| sd | standard deviation of a vector |
| cov | covariance between vectors |
| cor | correlation between vectors |
| diff | difference between elements in a vector |
| log | log of a vector or matrix |
| exp | exponentiation of a vector or matrix |
| abs | absolute value of a vector or matrix |

# Probability Distributions

- Probability density function (PDF)

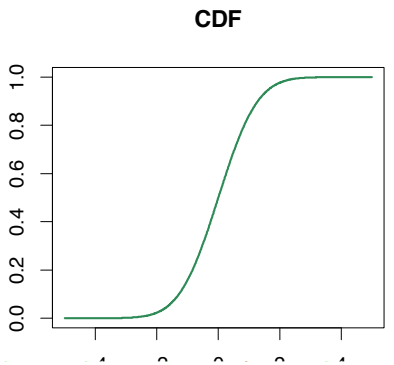
$$Pr(a < Y < b) = \int_a^b f_Y(y)$$

$$\int_{-1}^{1} f_Y(y)dy = 1$$




- Cumulative distribution function (CDF)

$$F_Y(y) = Pr(Y \leq y) = \int_{-1}^{y} f_Y(y)$$



# Probability Distributions

- x<- -5:5
- y<-dnorm(x)
- layout(matrix(1:1))
- c_mar=par("mar")
- n_mar=c_mar+c(0,1,0,0)
- plot(x=x,y=y, type="l", col="seagreen",lwd="2", xlab="quantile", ylab="dnorm(x)")
- grid(col="darkgrey",lwd=2)
- title(main="PDF")

# Pnorm, qnorm and rnorm

- Pnorm calculates the CDF
- Qnorm calculates the inverse of normal CDF
- Rnorm generates random numbers from a normal distribution

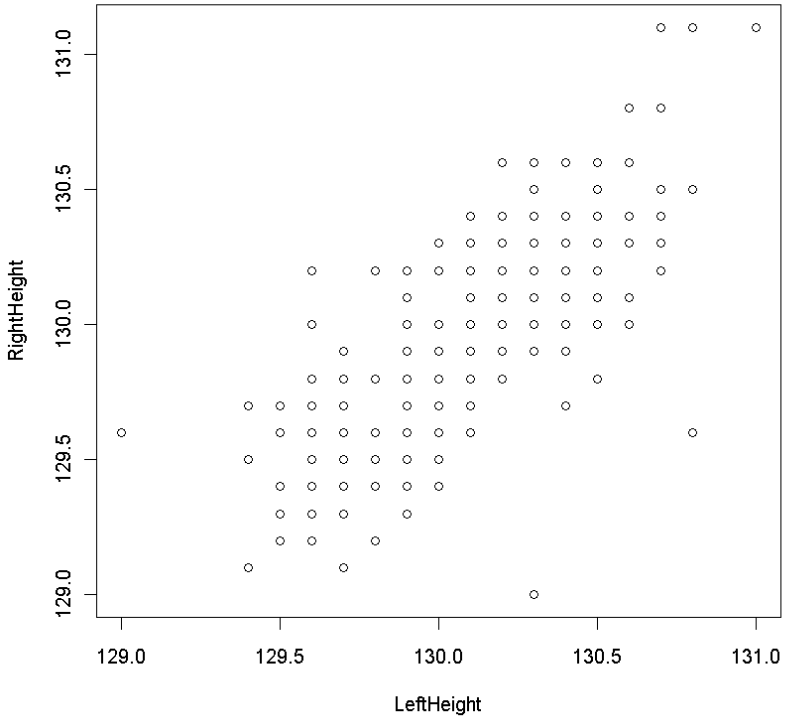
# plot() function

- plot() is the primary plotting function
- Calling plot will open a new plotting window
- Documentation:  ?plot
- For complete list of graphical parameters to manipulate:  ?par

# plot() function

- Assume a Data having LeftHeight and RightHeight as members
- Let's try a scatter plot of LeftHeight by RightHeight.

```
>plot(LeftHeight, RightHeight)
```

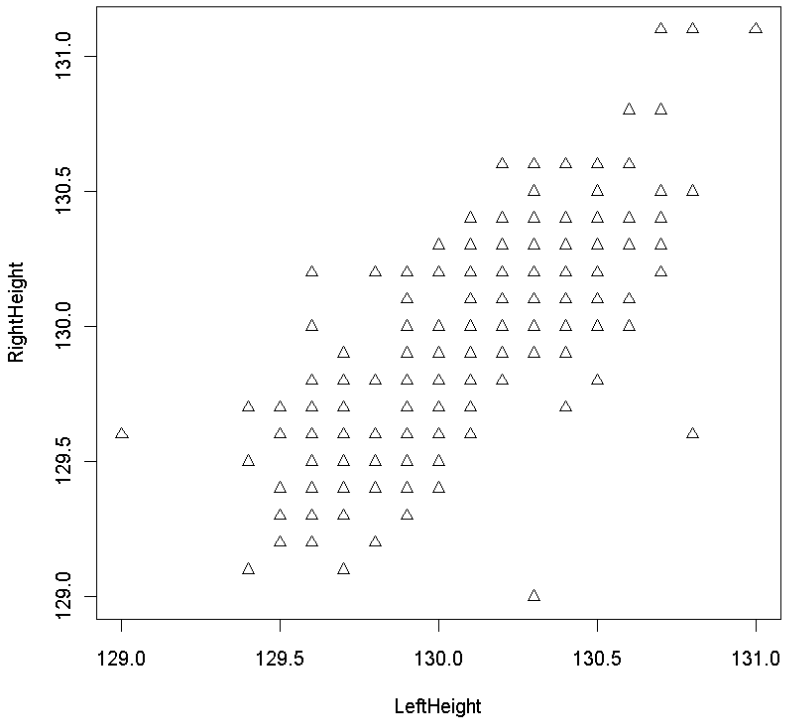


# plot() function

Change symbols: Option pch=.

See ?par for details.

>plot(LeftHeight,RightHeight,pch=2)



# plot() Function

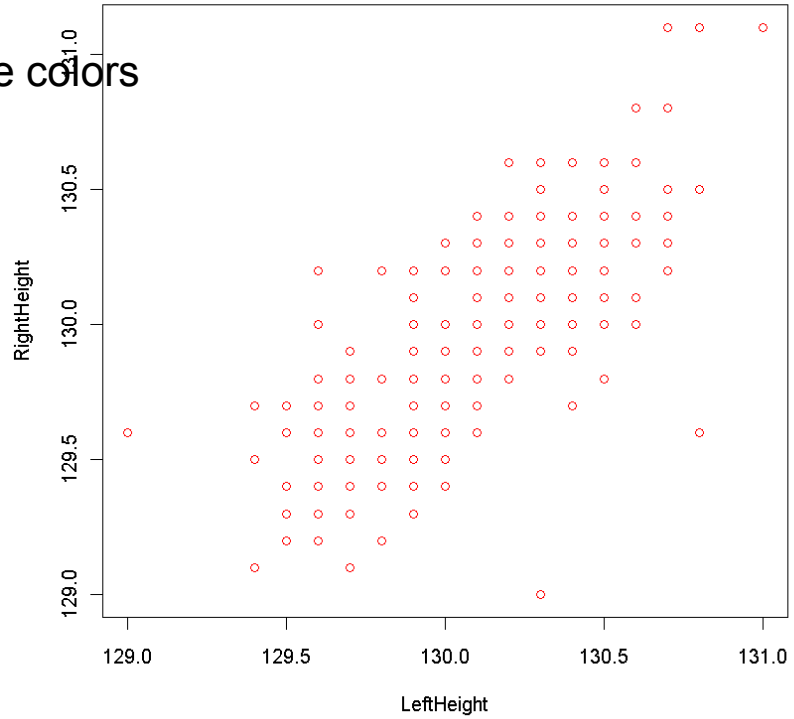
Change symbol color: Option col=

Specify by number or by name: col=2 or col="red"

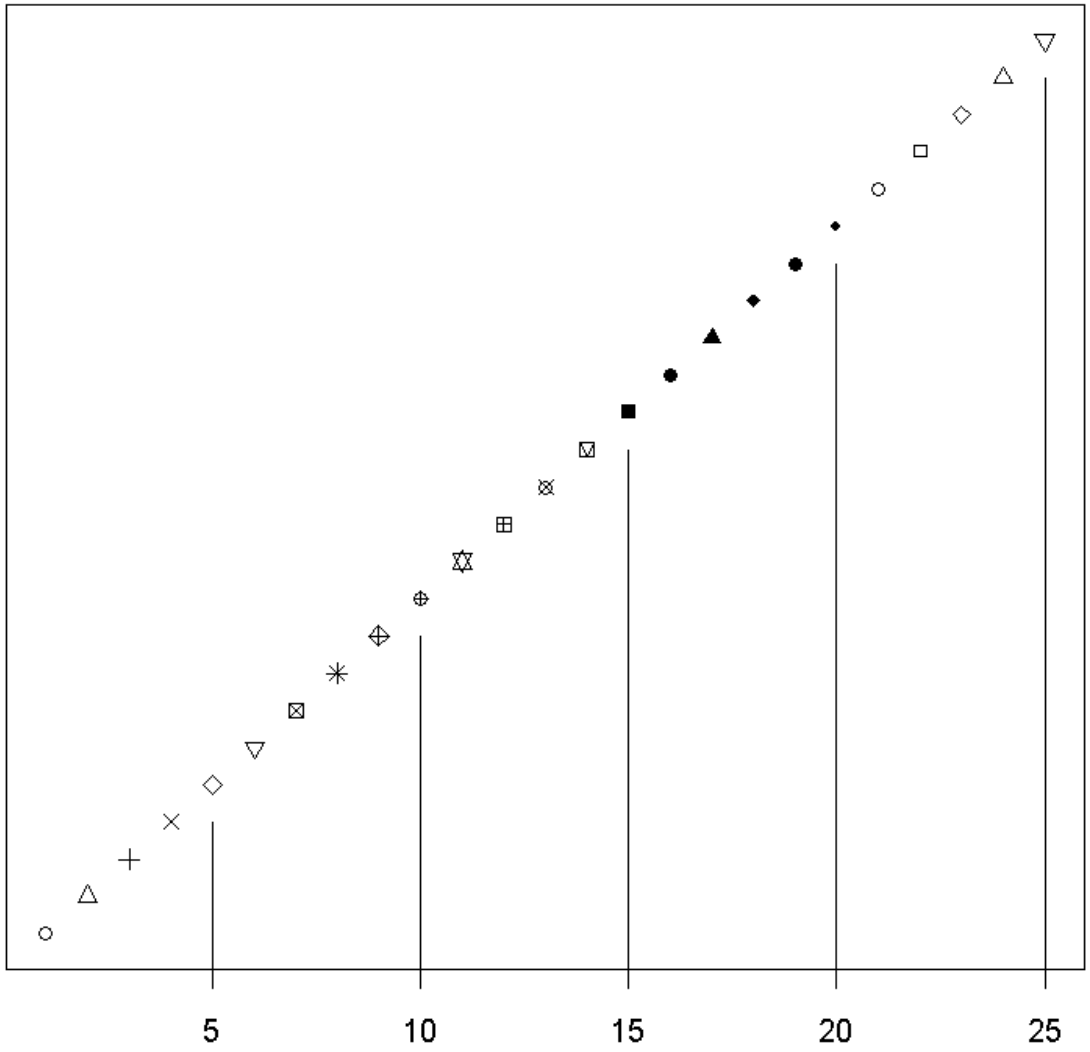
Hint: Type palette() to see colors associated with number

Type colors() to see all possible colors

> plot(LeftHeight, RightHeight, col="red")



# What types of points can we get?



# plot() Function

Change plot type:  Option type =

“p” for points

“l” for lines

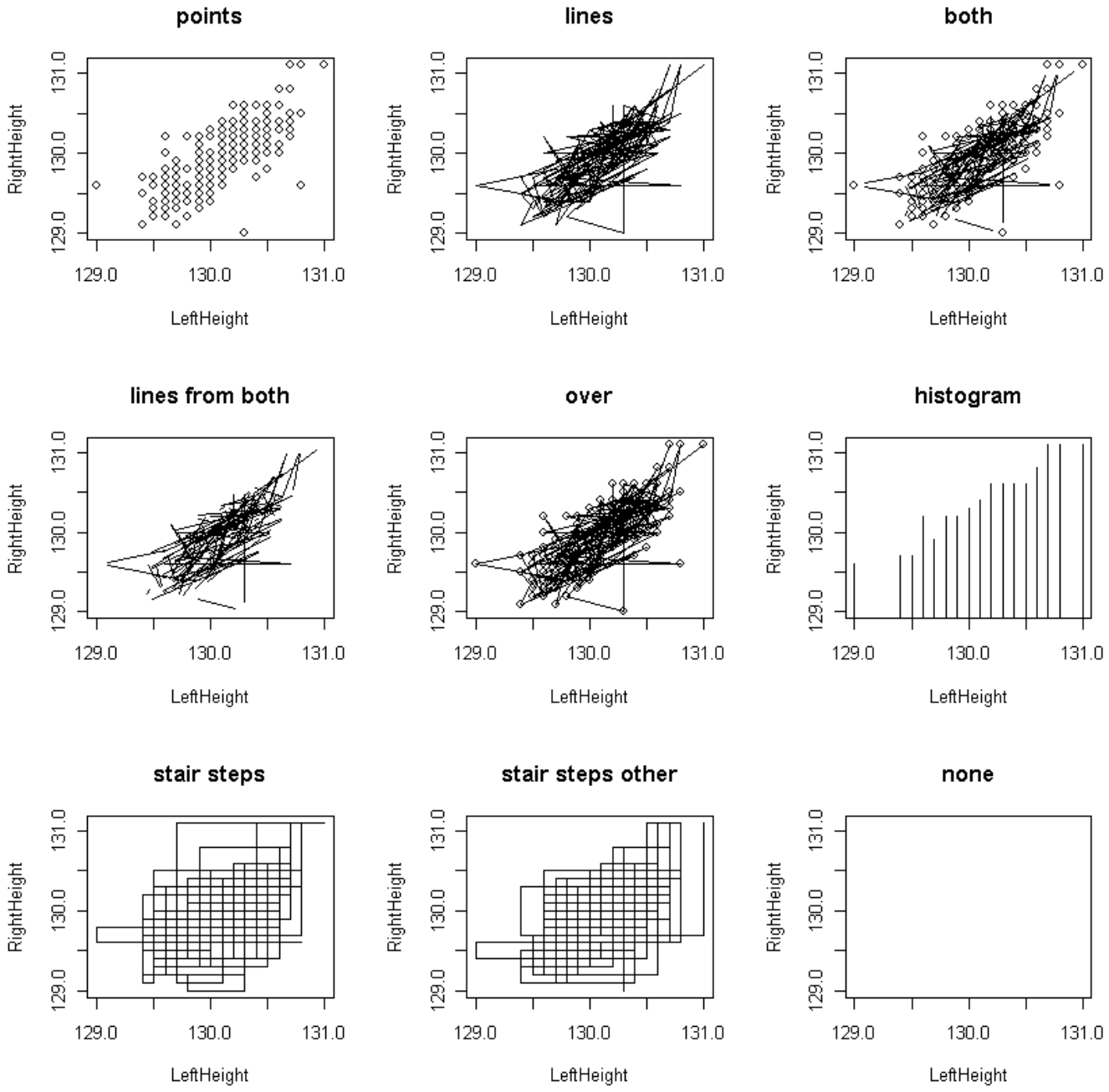
“b” for both

“c” for lines part alone of “b”

“o” for both overplotted

“h” for histogram like (or high-density) vertical lines

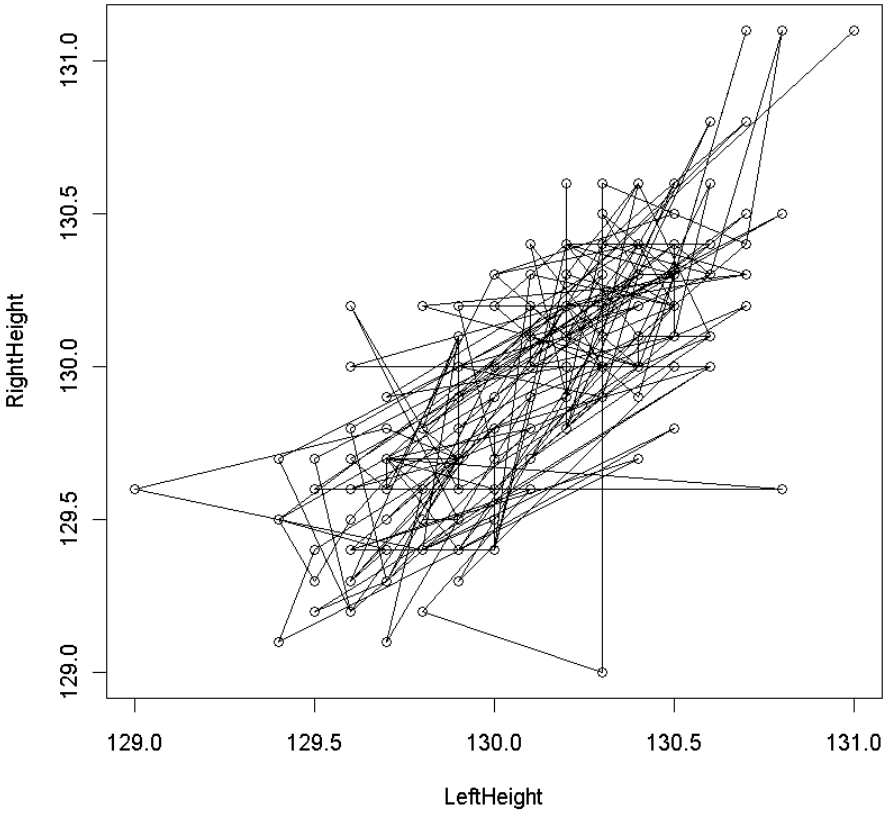
“s” for stair steps


points
lines
both
lines from both
over
histogram
stair steps
stair steps other
none
RightHeight
LeftHeight
129.0
130.0
131.0

# Plot() Function

Points with lines…works better on sorted list of points

>plot(LeftHeight,RightHeight,type="o")



# Axis Labels and Plot Titles

The plot() command call has options to

- Specify x-axis label:  xlab = “X Label”
- Specify y-axis label:  ylab = “Y Label”
- Specify plot title:  main = “Main Title”
- Specify subtitle: sub = “Subtitle”

➢ legend(“topleft”,c(“Happiness”,

”Healthiness”),pch=c(21,21),col=c(“red”,”blue”
))

# Adding Lines

To add straight lines to plot: abline()

abline() refers to standard equation for a line:

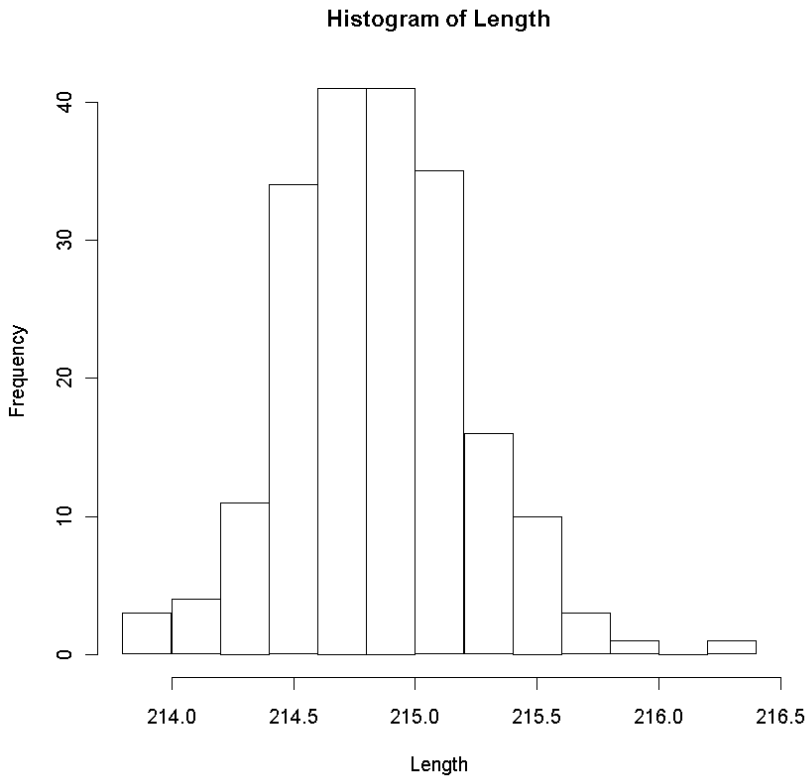
$$y = bx + a$$

- Horizontal line: abline(h= )
- Vertical Line: abline(v= )
- Otherwise: abline(a= , b= ) or abline(coef=c(a,b))

# Histograms

Histograms are another popular plotting option.

> hist(Length)



# pairs() Function

Using the SwissNote Data

> pairs(swiss)