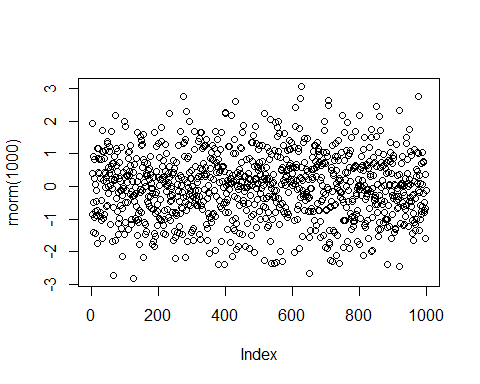
R-code-Topic-00.R

duter

2021-02-01

# Sample code to accompany Slides on Topic 01  
plot(rnorm(1000))



# If no additional arguments are provided rnorm creates draws from a standard-normal distribution  
# assign values to variables  
a = 5  
b <- 10  
  
# simple math calculations  
a + b - a \* b

## [1] -35

# Display the names of the objects  
objects()

## [1] "a" "b"

ls()

## [1] "a" "b"

# remove variables  
rm(a)  
# a  
  
# Variable names should be short, but descriptive  
#Camel caps:  
MyMathScore = 95  
#Underscore:  
my\_math\_score = 95  
#Dot separated:  
my.math.score = 95  
  
# If you know the name of the function or object on which you want  
# help:  
# help(read.csv)  
# help('read.csv')  
# ?read.csv  
  
# If you do not know the name of the function or object on which you  
# want help:  
# help.search('input')  
# RSiteSearch('input')  
# ??input  
  
# Assignment using function c()  
x <- c(1, 2, 3)  
c(1, 2, 3) -> x  
y <- c(x, 2, x)  
z <- c(1e3, 100)  
  
  
# Vector arithmetic  
x2 <- c(2, 4, 6)  
x3 <- x + x2  
x4 <- x - x2  
x5 <- x \* x2  
x6 <- x / x2  
  
  
# Generating regular sequences  
seq(-5, 5, by = 1) -> x  
x <- seq(length = 10, from = -5, by = .5)  
x <- rep(x, times = 5)  
x <- rep(x, each = 5)  
  
# Vector Operations  
x = 1:6  
y = 2  
x \* y

## [1] 2 4 6 8 10 12

y = c(1, 10)  
x \* y

## [1] 1 20 3 40 5 60

x / y

## [1] 1.0 0.2 3.0 0.4 5.0 0.6

# Logical vectors are generated by conditions:  
x <- 5 > 4  
  
  
# missing value: NA,  
z <- c(1:5, NA)  
is.na(z)

## [1] FALSE FALSE FALSE FALSE FALSE TRUE

# miss value: NULL  
# NULL cannot exist within a vector; if used, it simply disappears  
z <- c(1, NULL, 3)  
z

## [1] 1 3

#create a numerical vector  
x = 1:6  
#test whether x>3, create a logical vector  
x > 3

## [1] FALSE FALSE FALSE TRUE TRUE TRUE

x <= 3

## [1] TRUE TRUE TRUE FALSE FALSE FALSE

x == 3

## [1] FALSE FALSE TRUE FALSE FALSE FALSE

y = (x > 3)  
  
x = seq(0, 2, .2)  
#create a new vector from the 5th element of x  
x[4]

## [1] 0.6

#y=x[4:]  
y = x[4:length(x)]  
x[c(5, 6, 8)]

## [1] 0.8 1.0 1.4

z = 2:5  
  
x = 1:10  
y = (x > 5)  
summary(y)

## Mode FALSE TRUE   
## logical 5 5

x[y]

## [1] 6 7 8 9 10

x[!y]

## [1] 1 2 3 4 5

# Factors: Examples  
country <- c("US", "UK", "China", "India", "Japan", "Korea", "Canada")  
# convert to factor  
countryf <- factor(country)  
country

## [1] "US" "UK" "China" "India" "Japan" "Korea" "Canada"

countryf

## [1] US UK China India Japan Korea Canada  
## Levels: Canada China India Japan Korea UK US

# convert factor back to character vector  
as.character(countryf)

## [1] "US" "UK" "China" "India" "Japan" "Korea" "Canada"

# convert to numeric vector  
as.numeric(countryf)

## [1] 7 6 2 3 4 5 1

as.numeric(country)

## Warning: NAs introduced by coercion

## [1] NA NA NA NA NA NA NA

# Matrices and Data Frames  
aa = 1:6  
dim(aa) <- c(2, 3)  
aa

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

# Create a Matrix  
a <- 1:5  
b <- rnorm(5)  
# make a matrix by column binding  
c.matrix <- cbind(a, b)  
# names of rows and columns  
rownames(c.matrix)

## NULL

colnames(c.matrix)

## [1] "a" "b"

# Indexing for matrices  
c.matrix[4, 2]

## b   
## 0.4632554

b

## [1] 0.2787288 0.1042588 -0.9901384 0.4632554 0.7770140

c.matrix[1, ]

## a b   
## 1.0000000 0.2787288

a

## [1] 1 2 3 4 5

b

## [1] 0.2787288 0.1042588 -0.9901384 0.4632554 0.7770140

c.matrix[, 2]

## [1] 0.2787288 0.1042588 -0.9901384 0.4632554 0.7770140

c.matrix[c.matrix > 1]

## [1] 2 3 4 5

# Matrix Operations  
# create a matrix with 2 columns and 3 rows  
# filled with random normal values  
m.normal = matrix(rnorm(6), nrow = 3)  
m2 = m.normal \* 10  
m2

## [,1] [,2]  
## [1,] 7.622055 0.2465089  
## [2,] 5.241296 7.3179152  
## [3,] -6.793609 -11.9298834

m2[, 2] = m2[, 2] + 50  
summary(m2)

## V1 V2   
## Min. :-6.7936 Min. :38.07   
## 1st Qu.:-0.7762 1st Qu.:44.16   
## Median : 5.2413 Median :50.25   
## Mean : 2.0232 Mean :48.54   
## 3rd Qu.: 6.4317 3rd Qu.:53.78   
## Max. : 7.6221 Max. :57.32

# Matrices Versus Data Frames  
x = 1:10  
y = rnorm(10)  
mat <- cbind(x, y)  
class(mat[, 1])

## [1] "numeric"

z = paste0('a', 1:10)  
tab <- cbind(x, y, z)  
class(tab)

## [1] "matrix" "array"

mode(tab[, 1])

## [1] "character"

head(tab, 4)

## x y z   
## [1,] "1" "-0.437981678038827" "a1"  
## [2,] "2" "-0.398891132010042" "a2"  
## [3,] "3" "0.74101918981931" "a3"  
## [4,] "4" "0.514997496180661" "a4"

tab <- data.frame(x, y, z)  
class(tab)

## [1] "data.frame"

head(tab)

## x y z  
## 1 1 -0.4379817 a1  
## 2 2 -0.3988911 a2  
## 3 3 0.7410192 a3  
## 4 4 0.5149975 a4  
## 5 5 1.9011509 a5  
## 6 6 -1.2861437 a6

mode(tab[, 1])

## [1] "numeric"

rownames(tab)

## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"

rownames(tab) <- paste0("row", 1:10)  
rownames(tab)

## [1] "row1" "row2" "row3" "row4" "row5" "row6" "row7" "row8" "row9"   
## [10] "row10"

# Data frame columns can be refered to by name using the "dollar sign" operator $  
tab$x

## [1] 1 2 3 4 5 6 7 8 9 10

attach(tab)

## The following objects are masked \_by\_ .GlobalEnv:  
##   
## x, y, z

x

## [1] 1 2 3 4 5 6 7 8 9 10

# Column names can be set, which can be useful for referring to data later  
colnames(tab)

## [1] "x" "y" "z"

colnames(tab) <- c('a', 'b', 'c')  
colnames(tab)

## [1] "a" "b" "c"

colnames(tab) <- paste0('col', 1:3)  
colnames(tab)

## [1] "col1" "col2" "col3"

# A list is a collection of objects that may be the same or different types.  
# A data frame is a list of matched column vectors.  
# Create a list  
x = list(1, "y", c(2, 4, 6))  
x

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] "y"  
##   
## [[3]]  
## [1] 2 4 6

length(x)

## [1] 3

class(x)

## [1] "list"

x[[2]]

## [1] "y"

is.list(tab)

## [1] TRUE

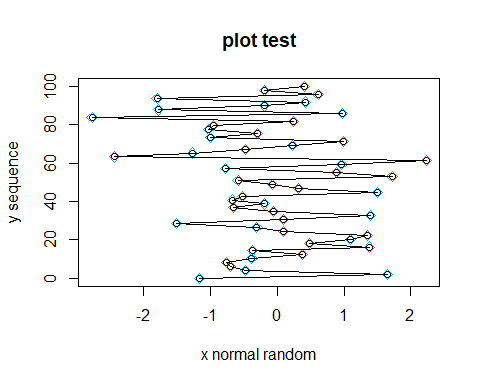
tab[[2]]

## [1] -0.4379817 -0.3988911 0.7410192 0.5149975 1.9011509 -1.2861437  
## [7] 1.2802239 -0.4912208 -1.0151303 0.7941247

names(tab)

## [1] "col1" "col2" "col3"

# Basic Plot Functions  
x = rnorm(50)  
y = seq(from = 0,  
 to = 100,  
 length.out = 50)  
plot(  
 x,  
 y,  
 xlab = 'x normal random',  
 ylab = 'y sequence',  
 main = 'plot test',  
 pch = 5,  
 col = 4  
)  
# plot(x)  
lines(x, y)  
points(x, y)



# Save a png image to a file  
png("my.first.plot.png", width = 480, height = 360)  
dev.off(3)

## png   
## 2

dev.list()

## png   
## 2

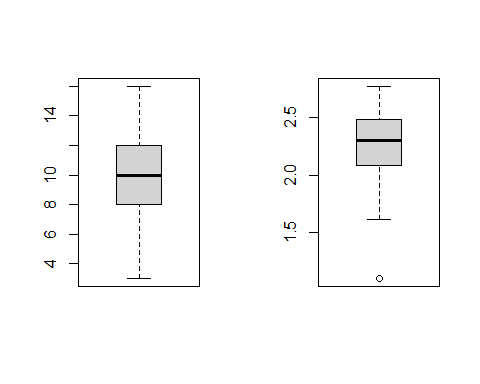
dev.set(2)

## png   
## 2

# setwd()  
getwd()

## [1] "D:/duter/Documents/JHU MSIS/Spring 2021/Data Analytics/Data\_Analytics/Week 1"

# boxplot, (try larger size)  
x = rpois(lambda = 10, 50)  
boxplot(x)  
par(mfrow = c(1, 2))  
boxplot(x)  
boxplot(log(x))



par(mfrow = c(1, 1))  
  
# Read Data  
# Use read.table() or read.csv() to read data into R  
Auto = read.csv("Auto.csv", header = T, na.strings = "?")  
  
# read data from the Internet  
theURL <- "http://www.jaredlander.com/data/Tomato%20First.csv"  
tomato <- read.table(file = theURL, header = TRUE, sep = ",")  
head(tomato)

## Round Tomato Price Source Sweet Acid Color Texture Overall  
## 1 1 Simpson SM 3.99 Whole Foods 2.8 2.8 3.7 3.4 3.4  
## 2 1 Tuttorosso (blue) 2.99 Pioneer 3.3 2.8 3.4 3.0 2.9  
## 3 1 Tuttorosso (green) 0.99 Pioneer 2.8 2.6 3.3 2.8 2.9  
## 4 1 La Fede SM DOP 3.99 Shop Rite 2.6 2.8 3.0 2.3 2.8  
## 5 2 Cento SM DOP 5.49 D Agostino 3.3 3.1 2.9 2.8 3.1  
## 6 2 Cento Organic 4.99 D Agostino 3.2 2.9 2.9 3.1 2.9  
## Avg.of.Totals Total.of.Avg  
## 1 16.1 16.1  
## 2 15.3 15.3  
## 3 14.3 14.3  
## 4 13.4 13.4  
## 5 14.4 15.2  
## 6 15.5 15.1

# Probability Distributions  
pnorm(2, mean = 5, sd = 10)

## [1] 0.3820886

dnorm(2, mean = 5, sd = 10)

## [1] 0.03813878

qnorm(.38, mean = 5, sd = 10)

## [1] 1.945192

z = rnorm(mean = 5, sd = 100, n = 10)  
  
# Example: Uniform Distribution  
dunif(x = 8, min = 5, max = 15)

## [1] 0.1

punif(10, min = 5, max = 15)

## [1] 0.5

qunif(.8, min = 5, max = 15)

## [1] 13

runif(10, min = 5, max = 15)

## [1] 13.070634 13.282950 11.688336 14.940027 6.469789 11.216427 5.056671  
## [8] 5.792931 11.707275 5.426582

# Example: Normal Distribution  
set.seed(5)  
rnorm(3, mean = 10, sd = 20)

## [1] -6.81711 37.68719 -15.10984

rnorm(3, mean = 10, sd = 20)

## [1] 11.40286 44.22882 -2.05816

set.seed(5)  
rnorm(3, mean = 10, sd = 20)

## [1] -6.81711 37.68719 -15.10984

# Sample Function  
sample(1:40, 5)

## [1] 21 30 7 19 3

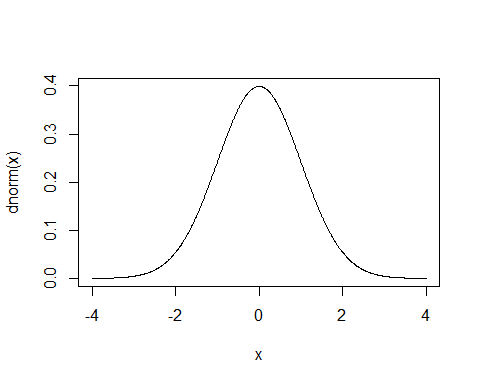
sample(c("H", "T"), 10, replace = T)

## [1] "T" "T" "H" "H" "T" "T" "T" "H" "T" "H"

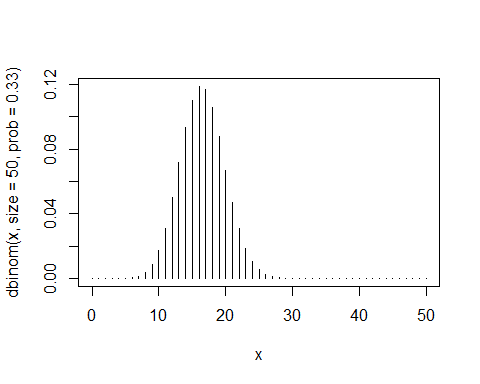
sample(c("success", "failure"),  
 10,  
 replace = T,  
 prob = c(0.8, 0.2))

## [1] "success" "success" "success" "success" "failure" "success" "failure"  
## [8] "success" "success" "success"

# Probability  
x <- seq(-4, 4, 0.01)  
plot(x, dnorm(x), type = "l")



x <- 0:50  
plot(x, dbinom(x, size = 50, prob = .33), type = "h")



# Define a Function  
f <- function(x) {  
 3 \* x ^ (-4)  
}  
f(2)

## [1] 0.1875

# Verify whether a function is a well-defined density function  
integrate(f, 1, Inf)

## 1 with absolute error < 1.1e-14

# Simulation: Generate Random Variables Following Any Distribution  
set.seed(13)  
U = runif(1000)  
X = (1 - U) ^ (-1 / 3)