I do it for fun

Charles Kwame Appiah

2025-03-25

x <- "Hello Charles, I hope you're working hard to become who you are destined to be by God's purpose? "  
x

## [1] "Hello Charles, I hope you're working hard to become who you are destined to be by God's purpose? "

class(x) # checking the class of variable x

## [1] "character"

y <- pi^2 #   
y

## [1] 9.869604

class(y) # checking the class of variable y

## [1] "numeric"

z <- 15L  
z

## [1] 15

class(z) # checking the class of variable z

## [1] "integer"

a <- (5 + 2i)^2  
a

## [1] 21+20i

class(a) # checking the class of variable a

## [1] "complex"

l <- TRUE  
class(l) # checking the class of variable l

## [1] "logical"

x <- list(age=c(10,21,22), weight=c(30,33,32))  
x

## $age  
## [1] 10 21 22  
##   
## $weight  
## [1] 30 33 32

names(x) # Calling the names of the list x

## [1] "age" "weight"

length(x) # checking the length of the list x

## [1] 2

xk <- data.frame(age=c(10,21,22), weight=c(30,33,32))  
xk

## age weight  
## 1 10 30  
## 2 21 33  
## 3 22 32

d <- c("Charles Kwame Appiah is my name")  
d

## [1] "Charles Kwame Appiah is my name"

class(d) # checking the class of variable d

## [1] "character"

length(d)

## [1] 1

f<- c(1,3,4,6,7)   
f

## [1] 1 3 4 6 7

class(f) # checking the class of variable f

## [1] "numeric"

fo<- c(1L,3L,4L,6L,7L)  
fo

## [1] 1 3 4 6 7

class(fo) # checking the class of variable fo

## [1] "integer"

# Initialization  
x <- vector(mode = "logical", length = 5)  
x

## [1] FALSE FALSE FALSE FALSE FALSE

class(x)

## [1] "logical"

x[1:3] <- TRUE # indexing the first to third element with TRUE  
x

## [1] TRUE TRUE TRUE FALSE FALSE

s <- c(TRUE, FALSE,TRUE, 1)  
s

## [1] 1 0 1 1

as.logical(s) # Default

## [1] TRUE FALSE TRUE TRUE

q <- list("Hello World",2015L, TRUE, 32.1)  
q

## [[1]]  
## [1] "Hello World"  
##   
## [[2]]  
## [1] 2015  
##   
## [[3]]  
## [1] TRUE  
##   
## [[4]]  
## [1] 32.1

class(q[[2]]) # Checking the class of list 2

## [1] "integer"

class(q[[4]])

## [1] "numeric"

class(q[[1]])

## [1] "character"

mat <- c(2,4,5,7)  
dim(mat) <- c(2,2) # creating a matrix with 2 rows and 2 columns  
mat

## [,1] [,2]  
## [1,] 2 5  
## [2,] 4 7

temp <- c(3,4,5,5.6,6,7)  
mati <- matrix(temp, nrow = 2, ncol = 3,byrow = TRUE) # creating a matrix with 2 rows and 3 columns  
mati

## [,1] [,2] [,3]  
## [1,] 3.0 4 5  
## [2,] 5.6 6 7

temp <- c(3,4,5,5.6,6,7, 8, 9, 10)  
mato <- matrix(temp, nrow=3, ncol=3,byrow=TRUE) # creating a matrix with 3 rows and 3 columns  
mato

## [,1] [,2] [,3]  
## [1,] 3.0 4 5  
## [2,] 5.6 6 7  
## [3,] 8.0 9 10

# Default byrow = FALSE  
temp <- c(3,4,5,5.6,6,7)  
mati <- matrix(temp, nrow=2, ncol=3,byrow=FALSE)  
mati

## [,1] [,2] [,3]  
## [1,] 3 5.0 6  
## [2,] 4 5.6 7

t1 <- c(23, 55)  
t2 <- c(34, 45)  
  
# By rows  
rbind(t1,t2) # binding them by their rows

## [,1] [,2]  
## t1 23 55  
## t2 34 45

t3 <- c(32, 50)  
t4 <- c(43, 54)  
  
# By columns  
cbind(t3,t4) # binding them by their columns

## t3 t4  
## [1,] 32 43  
## [2,] 50 54

factor <- c("Yes","No","No","Yes")  
factor # use to encode vectors

## [1] "Yes" "No" "No" "Yes"

f <- factor(c("Yes","No","No","Yes"), levels= c("Yes","No"))  
f

## [1] Yes No No Yes  
## Levels: Yes No

x <- NA # Missing number  
x

## [1] NA

is.na(x) # checking if it is a missing number

## [1] TRUE

u <- 0/0  
u

## [1] NaN

class(u) # checking for the class of variable u

## [1] "numeric"

### Dataframe  
c <- c("Charles","Richmond","Nicholas")  
d <- c(12, 23, 45)  
s <- c(FALSE,TRUE,TRUE)  
  
dfr <- data.frame(Username = c,Age = d, Adult = s)  
dfr # creating a dataframe

## Username Age Adult  
## 1 Charles 12 FALSE  
## 2 Richmond 23 TRUE  
## 3 Nicholas 45 TRUE

# First Row  
dfr[1,] # accessing row 1 of the dataframe

## Username Age Adult  
## 1 Charles 12 FALSE

# First Column  
dfr[,1] # accessing column 1 [Username] of the dataframe

## [1] "Charles" "Richmond" "Nicholas"

# Age Column  
dfr$Age

## [1] 12 23 45

# Username Column  
dfr$Username

## [1] "Charles" "Richmond" "Nicholas"

# Adult Column  
dfr$Adult

## [1] FALSE TRUE TRUE

### Importing Datasets  
library(readxl)

## Warning: package 'readxl' was built under R version 4.4.3

dat <- read.csv("C:/Users/HP/Desktop/Data Science/Machine learning/Training r.csv")  
#print(dat, na.rm = TRUE)  
head(dat)

## itching skin\_rash nodal\_skin\_eruptions continuous\_sneezing shivering chills  
## 1 1 1 1 0 0 0  
## 2 0 1 1 0 0 0  
## 3 1 0 1 0 0 0  
## 4 1 1 0 0 0 0  
## 5 1 1 1 0 0 0  
## 6 0 1 1 0 0 0  
## joint\_pain stomach\_pain acidity ulcers\_on\_tongue muscle\_wasting vomiting  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 0 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 0 0 0  
## burning\_micturition spotting\_.urination fatigue weight\_gain anxiety  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## cold\_hands\_and\_feets mood\_swings weight\_loss restlessness lethargy  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## patches\_in\_throat irregular\_sugar\_level cough high\_fever sunken\_eyes  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## breathlessness sweating dehydration indigestion headache yellowish\_skin  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## 4 0 0 0 0 0 0  
## 5 0 0 0 0 0 0  
## 6 0 0 0 0 0 0  
## dark\_urine nausea loss\_of\_appetite pain\_behind\_the\_eyes back\_pain  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## constipation abdominal\_pain diarrhoea mild\_fever yellow\_urine  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## yellowing\_of\_eyes acute\_liver\_failure fluid\_overload swelling\_of\_stomach  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## swelled\_lymph\_nodes malaise blurred\_and\_distorted\_vision phlegm  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## throat\_irritation redness\_of\_eyes sinus\_pressure runny\_nose congestion  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## chest\_pain weakness\_in\_limbs fast\_heart\_rate pain\_during\_bowel\_movements  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## pain\_in\_anal\_region bloody\_stool irritation\_in\_anus neck\_pain dizziness  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## cramps bruising obesity swollen\_legs swollen\_blood\_vessels  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## puffy\_face\_and\_eyes enlarged\_thyroid brittle\_nails swollen\_extremeties  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## excessive\_hunger extra\_marital\_contacts drying\_and\_tingling\_lips  
## 1 0 0 0  
## 2 0 0 0  
## 3 0 0 0  
## 4 0 0 0  
## 5 0 0 0  
## 6 0 0 0  
## slurred\_speech knee\_pain hip\_joint\_pain muscle\_weakness stiff\_neck  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## swelling\_joints movement\_stiffness spinning\_movements loss\_of\_balance  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## unsteadiness weakness\_of\_one\_body\_side loss\_of\_smell bladder\_discomfort  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## foul\_smell\_of.urine continuous\_feel\_of\_urine passage\_of\_gases  
## 1 0 0 0  
## 2 0 0 0  
## 3 0 0 0  
## 4 0 0 0  
## 5 0 0 0  
## 6 0 0 0  
## internal\_itching toxic\_look\_.typhos. depression irritability muscle\_pain  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## altered\_sensorium red\_spots\_over\_body belly\_pain abnormal\_menstruation  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## dischromic.\_patches watering\_from\_eyes increased\_appetite polyuria  
## 1 1 0 0 0  
## 2 1 0 0 0  
## 3 1 0 0 0  
## 4 1 0 0 0  
## 5 0 0 0 0  
## 6 1 0 0 0  
## family\_history mucoid\_sputum rusty\_sputum lack\_of\_concentration  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## visual\_disturbances receiving\_blood\_transfusion  
## 1 0 0  
## 2 0 0  
## 3 0 0  
## 4 0 0  
## 5 0 0  
## 6 0 0  
## receiving\_unsterile\_injections coma stomach\_bleeding distention\_of\_abdomen  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## history\_of\_alcohol\_consumption fluid\_overload.1 blood\_in\_sputum  
## 1 0 0 0  
## 2 0 0 0  
## 3 0 0 0  
## 4 0 0 0  
## 5 0 0 0  
## 6 0 0 0  
## prominent\_veins\_on\_calf palpitations painful\_walking pus\_filled\_pimples  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## blackheads scurring skin\_peeling silver\_like\_dusting small\_dents\_in\_nails  
## 1 0 0 0 0 0  
## 2 0 0 0 0 0  
## 3 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## inflammatory\_nails blister red\_sore\_around\_nose yellow\_crust\_ooze  
## 1 0 0 0 0  
## 2 0 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## prognosis X  
## 1 Fungal infection NA  
## 2 Fungal infection NA  
## 3 Fungal infection NA  
## 4 Fungal infection NA  
## 5 Fungal infection NA  
## 6 Fungal infection NA

data <- read\_xlsx("C:/Users/HP/Desktop/Data Science/Pandas/Copy of V1- UN Data on Refugees (AiCE \_\_ Dataset).xlsx")  
 head(data)

## # A tibble: 6 × 8  
## Country or territory of asylum or r…¹ Country or territory…² Year `Refugees\*`  
## <chr> <chr> <dbl> <dbl>  
## 1 Afghanistan Iran (Islamic Rep. of) 2021 38  
## 2 Afghanistan Pakistan 2021 72188  
## 3 Albania China 2021 14  
## 4 Albania Egypt 2021 5  
## 5 Albania Iraq 2021 5  
## 6 Albania Serbia and Kosovo: S/… 2021 57  
## # ℹ abbreviated names: ¹​`Country or territory of asylum or residence`,  
## # ²​`Country or territory of origin`  
## # ℹ 4 more variables: `Refugees assisted by UNHCR` <dbl>,  
## # `Total refugees and people in refugee-like situations\*\*` <dbl>,  
## # `Total refugees and people in refugee-like situations assisted by UNHCR` <dbl>,  
## # `Total Administrative Cost for Host Country` <dbl>

# Sequence  
v <- (10:20)  
v # start from 10 and end at 20

## [1] 10 11 12 13 14 15 16 17 18 19 20

w <- (-5:9)  
w # start from -5 and end at 9

## [1] -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9

qw <- seq(2,34,2)  
qw # start from 2 and end at 34 with a moving step of 2

## [1] 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34

iqw <- seq(2,34,length=6)  
iqw # start from 2 and end at 34 with a length of 6

## [1] 2.0 8.4 14.8 21.2 27.6 34.0

repe <- rep(1:4,4)   
repe # repeat 1 to 4, 4 times

## [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

eq <- rep("Hello Ann", 5)   
eq # repeat Hello Ann 5 times

## [1] "Hello Ann" "Hello Ann" "Hello Ann" "Hello Ann" "Hello Ann"

we <- seq(1,15, 2)   
we # start from 1 and end at 15 with a step of 2

## [1] 1 3 5 7 9 11 13 15

we[1:5] # slicing from index 1 to 5

## [1] 1 3 5 7 9

class(we) # checking the class of we

## [1] "numeric"

fo <- list("Hello","Hi","Hey")  
fo

## [[1]]  
## [1] "Hello"  
##   
## [[2]]  
## [1] "Hi"  
##   
## [[3]]  
## [1] "Hey"

fo[c(1,2)]

## [[1]]  
## [1] "Hello"  
##   
## [[2]]  
## [1] "Hi"

fo[c(1,2,3)] # for several elements

## [[1]]  
## [1] "Hello"  
##   
## [[2]]  
## [1] "Hi"  
##   
## [[3]]  
## [1] "Hey"

fo[[2]] # for only one element

## [1] "Hi"

class(fo[[3]])

## [1] "character"

wi <- list(age=c(12,23,45), height=c(12.3,45.4, 34.5))  
wi

## $age  
## [1] 12 23 45  
##   
## $height  
## [1] 12.3 45.4 34.5

class(wi)

## [1] "list"

woo <- data.frame(age=c(12,23,45), height=c(12.3,45.4, 34.5))  
woo

## age height  
## 1 12 12.3  
## 2 23 45.4  
## 3 45 34.5

class(woo)

## [1] "data.frame"

wi$age # accessing only the age list

## [1] 12 23 45

wi$height # accessing only the height list

## [1] 12.3 45.4 34.5

wi[["age"]] # accessing only the age list

## [1] 12 23 45

wi[['h',exact=FALSE]] # partial matching

## [1] 12.3 45.4 34.5

class(wi$age) # checking the class of age list

## [1] "numeric"

class(wi$height) # checking the class of height list

## [1] "numeric"

wr <- matrix(1:9, nrow = 3, ncol = 3, by = TRUE)  
wr # creating a matrix with 3 rows and 3 columns

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

class(wr)

## [1] "matrix" "array"

class(wr[1,1]) # checking the class of row 1 column 1

## [1] "integer"

class(wr[1,1, drop=FALSE]) # checking the class of row 1 column 1

## [1] "matrix" "array"

ch <- c(1:9,NA,NA,NA)  
ch

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

i<- is.na(ch) # Locating missing numbers  
i

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE

ch[!i] # filtering non missing numbers or is not missing numbers

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

### Vectorization  
ew <- rnorm(1000)  
  
er <- rnorm(1000)  
  
cv <- vector(mode="numeric", length=1000)  
  
### Iteration  
start <- proc.time()  
for (i in 1:1000){   
cv[i] <- ew[i] + er[i]   
}  
proc.time()-start

## user system elapsed   
## 0.02 0.00 0.00

start <- proc.time()  
cv <- ew + er  
proc.time()-start

## user system elapsed   
## 0 0 0

#### Control Structures  
x <- 20  
if (x < 0) {  
print("Negative!")  
}else if (x < 10) {  
print("Positive, less than 10!")  
}else {  
print("Number greater than 10!")  
}

## [1] "Number greater than 10!"

x <- -20  
if (x < 0) {  
print("Negative!")  
}else if (x < 10) {  
print("Positive, less than 10!")  
}else {  
print("Number greater than 10!")  
}

## [1] "Negative!"

x <- 6  
if (x < 0) {  
print("Negative!")  
}else if (x < 10) {  
print("Positive, less than 10!")  
}else {  
print("Number greater than 10!")  
}

## [1] "Positive, less than 10!"

### for loop  
for (i in 1:100){  
cat(i)  
cat(" ") # inserting spaces between the numbers  
}

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

letters # lower case

## [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"  
## [20] "t" "u" "v" "w" "x" "y" "z"

LETTERS # upper case

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

class(letters)

## [1] "character"

for (x in letters){  
cat(x)  
cat(" ") # inserting spaces between the letters  
}

## 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

# while loop  
x <- -1  
while (x < 5){  
print(x)  
x <- x+1  
}

## [1] -1  
## [1] 0  
## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4

x<- 1  
repeat{  
print(x)  
if (x > 7){  
break  
}  
x <- x+1  
}

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

for (i in 1:100){  
# Over ride the first 20 iterations  
if (i <= 20){  
next   
}  
}  
  
# Functions  
myPrinter <- function(x){  
 for (i in seq(x)){  
 print("Hello, Charles")  
 }  
}  
myPrinter(3)

## [1] "Hello, Charles"  
## [1] "Hello, Charles"  
## [1] "Hello, Charles"

volume <- function(x=3, y=3, z=3){  
 print(x\*y\*z)  
}  
volume(y=3,z=5,x=11)

## [1] 165

volume()

## [1] 27

myPrinter <- function(..., mes){  
print(sum(...))  
print(mes)  
}  
myPrinter (3, 5, 11, mes= "Hi! Richmond")

## [1] 19  
## [1] "Hi! Richmond"

ls() ### displaying objects stored in R currently

## [1] "a" "c" "ch" "cv" "d" "dat"   
## [7] "data" "dfr" "eq" "er" "ew" "f"   
## [13] "factor" "fo" "i" "iqw" "l" "mat"   
## [19] "mati" "mato" "myPrinter" "q" "qw" "repe"   
## [25] "s" "start" "t1" "t2" "t3" "t4"   
## [31] "temp" "u" "v" "volume" "w" "we"   
## [37] "wi" "woo" "wr" "x" "xk" "y"   
## [43] "z"

# Iterated Functions  
# lapply  
str(lapply)

## function (X, FUN, ...)

x <- list(a=rnorm(10), b=rnorm(20), c=rnorm(30))  
lapply(x, mean) # checking the mean of x

## $a  
## [1] -0.1842628  
##   
## $b  
## [1] -0.2374803  
##   
## $c  
## [1] 0.002889615

lapply(x, var)# checking the variance of x

## $a  
## [1] 0.8126082  
##   
## $b  
## [1] 0.6125357  
##   
## $c  
## [1] 0.8958783

lapply(x, sd) # checking the standard deviation of x

## $a  
## [1] 0.9014478  
##   
## $b  
## [1] 0.7826466  
##   
## $c  
## [1] 0.9465085

### sapply  
str(sapply)

## function (X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)

xi <- list(a=rnorm(10), b=rnorm(20), c=rnorm(30))  
sapply(xi, mean) # checking the mean of xi

## a b c   
## 0.05644771 -0.41723950 -0.19260909

sapply(xi, var) # checking the variance of xi

## a b c   
## 1.0555137 0.9923723 1.2829147

sapply(xi, sd) # checking the standard deviation of xi

## a b c   
## 1.0273820 0.9961788 1.1326582

### Split  
dat <- data.frame(subject = 1:6,age = c(15, 17, 16,20,21,23),  
adult = c(FALSE,FALSE,FALSE,TRUE,TRUE,TRUE))  
s <- split(dat, dat$adult)  
s #### split them according to True and False

## $`FALSE`  
## subject age adult  
## 1 1 15 FALSE  
## 2 2 17 FALSE  
## 3 3 16 FALSE  
##   
## $`TRUE`  
## subject age adult  
## 4 4 20 TRUE  
## 5 5 21 TRUE  
## 6 6 23 TRUE

sapply(s, function(x){  
mean(x[["age"]])  
})

## FALSE TRUE   
## 16.00000 21.33333

sapply(s, function(x){  
var(x[["age"]])  
})

## FALSE TRUE   
## 1.000000 2.333333

sapply(s, function(x){  
sd(x[["age"]])  
})

## FALSE TRUE   
## 1.000000 1.527525

# tapply  
str(tapply)

## function (X, INDEX, FUN = NULL, ..., default = NA, simplify = TRUE)

x <- c(rnorm(10),rnorm(10),rnorm(10),rnorm(10))  
f <- gl(4, 10)  
f

## [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4  
## [39] 4 4  
## Levels: 1 2 3 4

tapply(x, f, mean)

## 1 2 3 4   
## 0.04848935 0.38028863 0.29299551 0.06208304

tapply(x, f, var)

## 1 2 3 4   
## 1.188120 1.737474 0.816524 1.044148

tapply(x, f, sd)

## 1 2 3 4   
## 1.0900090 1.3181327 0.9036172 1.0218355

### Help   
?c

## starting httpd help server ... done

?vector  
?sapply  
?lapply  
?tapply

### Types, Quality and Data preprocessing  
wi

## $age  
## [1] 12 23 45  
##   
## $height  
## [1] 12.3 45.4 34.5

### finding each column maximum  
m <- sapply(wi,max)  
m

## age height   
## 45.0 45.4

### finding each column minimum  
n <- sapply(wi,min)  
n

## age height   
## 12.0 12.3

### Regularization ith range [0,1]  
wi$age <- ( (wi$age - n[1])/(m[1] - n[1]))\*(1 - 0) + 0  
wi$height <- ( (wi$height - n[2])/(m[2] - n[2]))\*(1 - 0)   
  
wi

## $age  
## [1] 0.0000000 0.3333333 1.0000000  
##   
## $height  
## [1] 0.0000000 1.0000000 0.6706949

# DPLYR AND TIDYR PACKAGES  
# install.packages("dplyr")  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

data(airquality)  
class(airquality)

## [1] "data.frame"

airquality <- tibble(airquality)  
class(airquality)

## [1] "tbl\_df" "tbl" "data.frame"

airquality

## # A tibble: 153 × 6  
## Ozone Solar.R Wind Temp Month Day  
## <int> <int> <dbl> <int> <int> <int>  
## 1 41 190 7.4 67 5 1  
## 2 36 118 8 72 5 2  
## 3 12 149 12.6 74 5 3  
## 4 18 313 11.5 62 5 4  
## 5 NA NA 14.3 56 5 5  
## 6 28 NA 14.9 66 5 6  
## 7 23 299 8.6 65 5 7  
## 8 19 99 13.8 59 5 8  
## 9 8 19 20.1 61 5 9  
## 10 NA 194 8.6 69 5 10  
## # ℹ 143 more rows

select(airquality, Ozone, Solar.R, Day)

## # A tibble: 153 × 3  
## Ozone Solar.R Day  
## <int> <int> <int>  
## 1 41 190 1  
## 2 36 118 2  
## 3 12 149 3  
## 4 18 313 4  
## 5 NA NA 5  
## 6 28 NA 6  
## 7 23 299 7  
## 8 19 99 8  
## 9 8 19 9  
## 10 NA 194 10  
## # ℹ 143 more rows

select(airquality, -(Wind:Month)) # offsetting Wind and Month from the airquality dataset

## # A tibble: 153 × 3  
## Ozone Solar.R Day  
## <int> <int> <int>  
## 1 41 190 1  
## 2 36 118 2  
## 3 12 149 3  
## 4 18 313 4  
## 5 NA NA 5  
## 6 28 NA 6  
## 7 23 299 7  
## 8 19 99 8  
## 9 8 19 9  
## 10 NA 194 10  
## # ℹ 143 more rows

filter(airquality, Month > 5, Month < 9, Day < 3) # values in Month greater than 5 and less than 9, Day values less than 3

## # A tibble: 6 × 6  
## Ozone Solar.R Wind Temp Month Day  
## <int> <int> <dbl> <int> <int> <int>  
## 1 NA 286 8.6 78 6 1  
## 2 NA 287 9.7 74 6 2  
## 3 135 269 4.1 84 7 1  
## 4 49 248 9.2 85 7 2  
## 5 39 83 6.9 81 8 1  
## 6 9 24 13.8 81 8 2

filter(airquality, Day==1 | Day == 2) # Values of Day = 1 or 2

## # A tibble: 10 × 6  
## Ozone Solar.R Wind Temp Month Day  
## <int> <int> <dbl> <int> <int> <int>  
## 1 41 190 7.4 67 5 1  
## 2 36 118 8 72 5 2  
## 3 NA 286 8.6 78 6 1  
## 4 NA 287 9.7 74 6 2  
## 5 135 269 4.1 84 7 1  
## 6 49 248 9.2 85 7 2  
## 7 39 83 6.9 81 8 1  
## 8 9 24 13.8 81 8 2  
## 9 96 167 6.9 91 9 1  
## 10 78 197 5.1 92 9 2

arrange(airquality,Ozone, desc(Solar.R))

## # A tibble: 153 × 6  
## Ozone Solar.R Wind Temp Month Day  
## <int> <int> <dbl> <int> <int> <int>  
## 1 1 8 9.7 59 5 21  
## 2 4 25 9.7 61 5 23  
## 3 6 78 18.4 57 5 18  
## 4 7 49 10.3 69 9 24  
## 5 7 48 14.3 80 7 15  
## 6 7 NA 6.9 74 5 11  
## 7 8 19 20.1 61 5 9  
## 8 9 36 14.3 72 8 22  
## 9 9 24 13.8 81 8 2  
## 10 9 24 10.9 71 9 14  
## # ℹ 143 more rows

mutate(airquality, Temp.C = round((Temp - 32) \* 5/9)) # creating a new column for Temp.C

## # A tibble: 153 × 7  
## Ozone Solar.R Wind Temp Month Day Temp.C  
## <int> <int> <dbl> <int> <int> <int> <dbl>  
## 1 41 190 7.4 67 5 1 19  
## 2 36 118 8 72 5 2 22  
## 3 12 149 12.6 74 5 3 23  
## 4 18 313 11.5 62 5 4 17  
## 5 NA NA 14.3 56 5 5 13  
## 6 28 NA 14.9 66 5 6 19  
## 7 23 299 8.6 65 5 7 18  
## 8 19 99 13.8 59 5 8 15  
## 9 8 19 20.1 61 5 9 16  
## 10 NA 194 8.6 69 5 10 21  
## # ℹ 143 more rows

# Removing rows with missing values on the Ozone and Solar.R features  
airquality <- filter(airquality, !is.na(Ozone),!is.na(Solar.R))  
airquality

## # A tibble: 111 × 6  
## Ozone Solar.R Wind Temp Month Day  
## <int> <int> <dbl> <int> <int> <int>  
## 1 41 190 7.4 67 5 1  
## 2 36 118 8 72 5 2  
## 3 12 149 12.6 74 5 3  
## 4 18 313 11.5 62 5 4  
## 5 23 299 8.6 65 5 7  
## 6 19 99 13.8 59 5 8  
## 7 8 19 20.1 61 5 9  
## 8 16 256 9.7 69 5 12  
## 9 11 290 9.2 66 5 13  
## 10 14 274 10.9 68 5 14  
## # ℹ 101 more rows

### print(airquality, n=143)

### Grouping by month  
by\_month <- group\_by (airquality, Month)  
by\_month

## # A tibble: 111 × 6  
## # Groups: Month [5]  
## Ozone Solar.R Wind Temp Month Day  
## <int> <int> <dbl> <int> <int> <int>  
## 1 41 190 7.4 67 5 1  
## 2 36 118 8 72 5 2  
## 3 12 149 12.6 74 5 3  
## 4 18 313 11.5 62 5 4  
## 5 23 299 8.6 65 5 7  
## 6 19 99 13.8 59 5 8  
## 7 8 19 20.1 61 5 9  
## 8 16 256 9.7 69 5 12  
## 9 11 290 9.2 66 5 13  
## 10 14 274 10.9 68 5 14  
## # ℹ 101 more rows

#Finding the minimum, average and maximum value per Month  
summarize (by\_month, min(Ozone), mean(Ozone), max(Ozone))

## # A tibble: 5 × 4  
## Month `min(Ozone)` `mean(Ozone)` `max(Ozone)`  
## <int> <int> <dbl> <int>  
## 1 5 1 24.1 115  
## 2 6 12 29.4 71  
## 3 7 7 59.1 135  
## 4 8 9 60 168  
## 5 9 7 31.4 96

#install.packages (“tidyr”) #library (tidyr)

#dat #gather(dat, sex, count, -subject)

#dat <- gather(dat, sex, class, count, -subject) #dat

#dat #separate(dat, sex, class, c(“sex”, “class”))

#dat #dat <- gather(dat, lesson, grade, lesson1:lesson4, na.rm = TRUE) #dat

#dat <- spread(dat, quarter, grade) #dat

#mutate(dat, lesson = extract\_numeric(lesson))

### Statistical Summary and Visualization  
  
### Mean  
internet\_usage = c(22,0, 7,12,5, 33, 14, 8, 0, 9)  
#internet\_usage  
mean(internet\_usage) # finding the mean of internet\_usage

## [1] 11

net\_usage = c(22,0,7,12,5,NA,33,14,8,NA,0,9)  
#net\_usage  
mean(na.omit(net\_usage)) # finding the mean of net\_usage with missing numbers

## [1] 11

# Median  
median(net\_usage, na.rm = TRUE)

## [1] 8.5

# Minimum, Maximum and Range  
A = c(49,33,63,48,54,62,52,64,71,68)  
min(A)

## [1] 33

max(A)

## [1] 71

which.min(A)

## [1] 2

which.max(A)

## [1] 9

print(max(A) - min(A))

## [1] 38

range(A)

## [1] 33 71

print(range(A)[2] - range(A)[1])

## [1] 38

summary(A)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 33.00 49.75 58.00 56.40 63.75 71.00

### Percentile Values  
X = c(3,4,5,6,7,8,10,10,11,12,14,14,14,15,16,17,21,25,27,32)  
quantile(X,0.80)

## 80%   
## 17.8

quantile(X,0.50,type = 7)

## 50%   
## 13

quantile(X,0.25,type = 7)

## 25%   
## 7.75

quantile(X,0.75,type = 7)

## 75%   
## 16.25

median(X)

## [1] 13

summary(X)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.00 7.75 13.00 13.55 16.25 32.00

sd(X)

## [1] 7.843703

###cv(X)

# Interquartile Range  
irq = function(X) (quantile(X,0.75) - quantile(X,0.25))  
irq(X)

## 75%   
## 8.5

### Variance and Standard Deviation  
course = c(6, 2, 1, 9, 17, 4, 3, 2, 1, 5, 11 ,4, 3, 1, 2, 2, 5, 4, 3, 6)  
1 / course

## [1] 0.16666667 0.50000000 1.00000000 0.11111111 0.05882353 0.25000000  
## [7] 0.33333333 0.50000000 1.00000000 0.20000000 0.09090909 0.25000000  
## [13] 0.33333333 1.00000000 0.50000000 0.50000000 0.20000000 0.25000000  
## [19] 0.33333333 0.16666667

cf = c(course, 0, course)  
cf

## [1] 6 2 1 9 17 4 3 2 1 5 11 4 3 1 2 2 5 4 3 6 0 6 2 1 9  
## [26] 17 4 3 2 1 5 11 4 3 1 2 2 5 4 3 6

vw <- 2 \* course + cf + 1  
vw

## [1] 19 7 4 28 52 13 10 7 4 16 34 13 10 4 7 7 16 13 10 19 13 11 5 20 44  
## [26] 26 11 8 5 12 28 20 11 6 6 7 13 14 11 16 19

var(course)

## [1] 15.41842

sum((course - mean(course)) ^ 2 / (length(course) - 1))

## [1] 15.41842

sd(course)

## [1] 3.92663

sqrt(var(course))

## [1] 3.92663

std = function(x) sqrt(var(x))  
std(course)

## [1] 3.92663

sqrt(course)

## [1] 2.449490 1.414214 1.000000 3.000000 4.123106 2.000000 1.732051 1.414214  
## [9] 1.000000 2.236068 3.316625 2.000000 1.732051 1.000000 1.414214 1.414214  
## [17] 2.236068 2.000000 1.732051 2.449490

sum(course)

## [1] 91

prod(course)

## [1] 41878425600

sort(course)

## [1] 1 1 1 2 2 2 2 3 3 3 4 4 4 5 5 6 6 9 11 17

order(course)

## [1] 3 9 14 2 8 15 16 7 13 19 6 12 18 10 17 1 20 4 11 5

sqrt(-14 + 9i)

## [1] 1.149634+3.914289i

### Coefficient of Variation  
cv = function(x) ( sd(x) / mean(x) )  
cv(course)

## [1] 0.8629955

### Visualization of Qualitative Data  
mo = c("car","car","bus", "metro","metro","car","metro","metro","foot","car","foot","bus","bus","metro","metro","car","car","car","metro","car") ### dataset of employee's   
mo

## [1] "car" "car" "bus" "metro" "metro" "car" "metro" "metro" "foot"   
## [10] "car" "foot" "bus" "bus" "metro" "metro" "car" "car" "car"   
## [19] "metro" "car"

table(mo)

## mo  
## bus car foot metro   
## 3 8 2 7

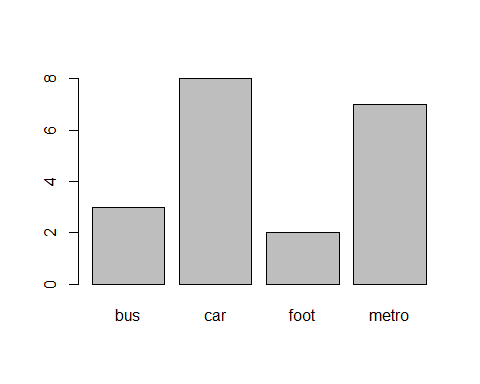
prop.table(table(mo))

## mo  
## bus car foot metro   
## 0.15 0.40 0.10 0.35

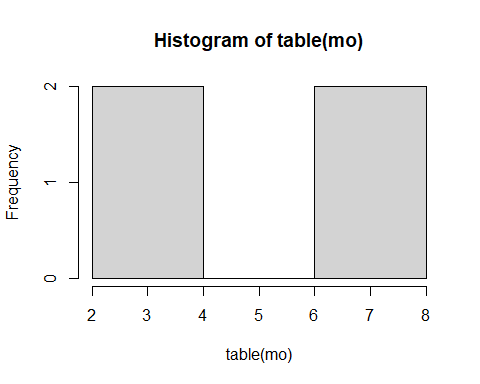
data.frame(mo)

## mo  
## 1 car  
## 2 car  
## 3 bus  
## 4 metro  
## 5 metro  
## 6 car  
## 7 metro  
## 8 metro  
## 9 foot  
## 10 car  
## 11 foot  
## 12 bus  
## 13 bus  
## 14 metro  
## 15 metro  
## 16 car  
## 17 car  
## 18 car  
## 19 metro  
## 20 car

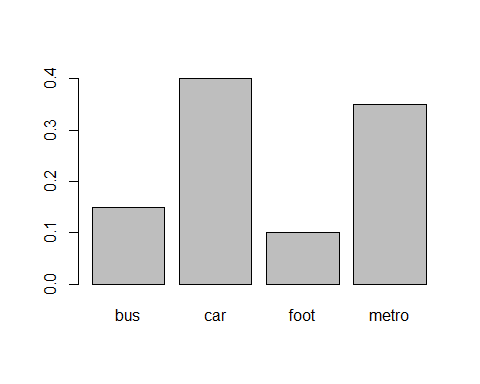
### Bar Charts  
barplot(table(mo))



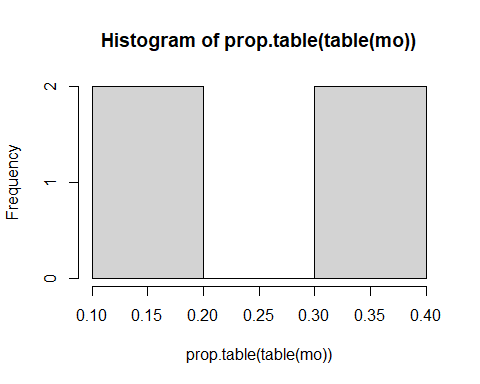
hist(table(mo))



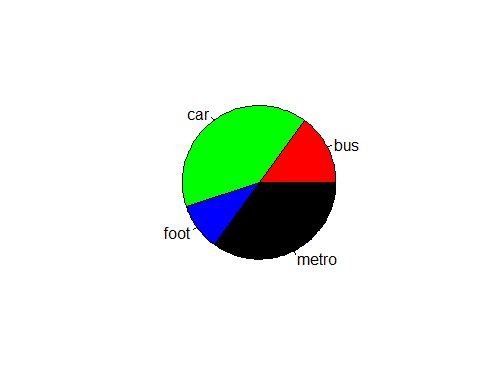
barplot(prop.table(table(mo)))



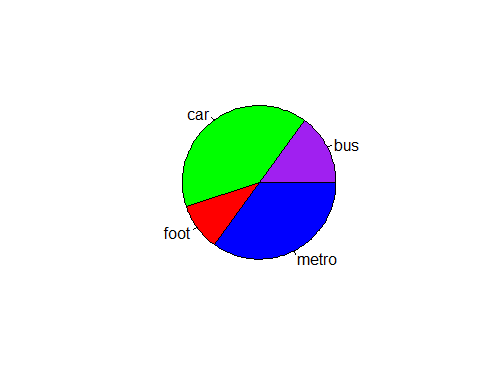
hist(prop.table(table(mo)))



### Pie Chart  
pie(table(mo), col = c ("red", "green", "blue", "black"))



pie(prop.table(table(mo)), col = c("purple", "green", "red", "blue"))



### Contingency Matrix  
g = c(rep("Male",8), rep("Female",12))  
g

## [1] "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male"   
## [9] "Female" "Female" "Female" "Female" "Female" "Female" "Female" "Female"  
## [17] "Female" "Female" "Female" "Female"

mg = table(mo,g)  
mg

## g  
## mo Female Male  
## bus 2 1  
## car 5 3  
## foot 2 0  
## metro 3 4

gm = data.frame(mo,g)  
gm

## mo g  
## 1 car Male  
## 2 car Male  
## 3 bus Male  
## 4 metro Male  
## 5 metro Male  
## 6 car Male  
## 7 metro Male  
## 8 metro Male  
## 9 foot Female  
## 10 car Female  
## 11 foot Female  
## 12 bus Female  
## 13 bus Female  
## 14 metro Female  
## 15 metro Female  
## 16 car Female  
## 17 car Female  
## 18 car Female  
## 19 metro Female  
## 20 car Female

margin.table(mg, 1)

## mo  
## bus car foot metro   
## 3 8 2 7

margin.table(mg,2)

## g  
## Female Male   
## 12 8

prop.table(mg)

## g  
## mo Female Male  
## bus 0.10 0.05  
## car 0.25 0.15  
## foot 0.10 0.00  
## metro 0.15 0.20

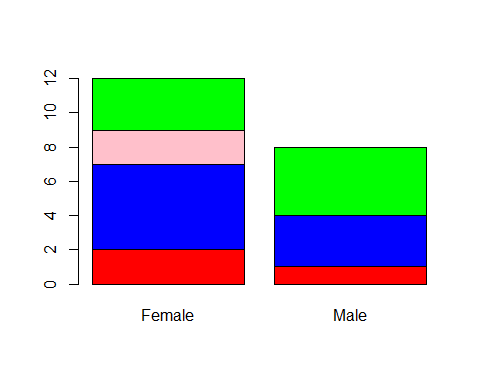
prop.table(mg,1)

## g  
## mo Female Male  
## bus 0.6666667 0.3333333  
## car 0.6250000 0.3750000  
## foot 1.0000000 0.0000000  
## metro 0.4285714 0.5714286

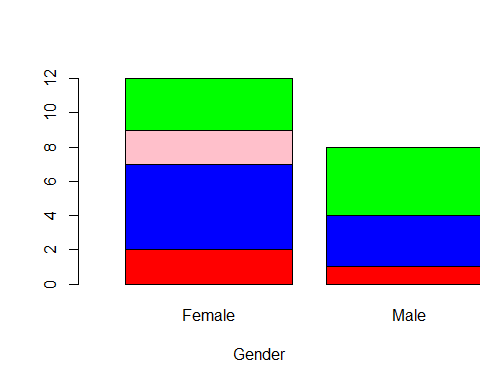
prop.table(mg,2)

## g  
## mo Female Male  
## bus 0.1666667 0.1250000  
## car 0.4166667 0.3750000  
## foot 0.1666667 0.0000000  
## metro 0.2500000 0.5000000

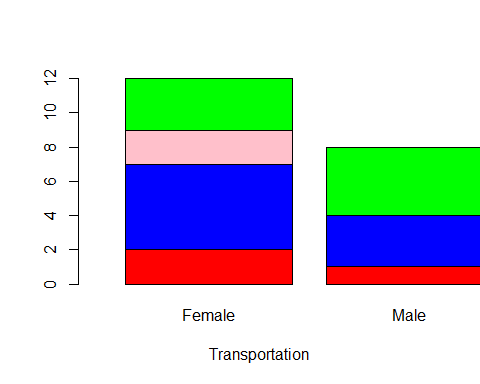
### Stacked Bar Charts and Grouped Bar Charts  
barplot(mg, col = c("red","blue", "pink","green"))



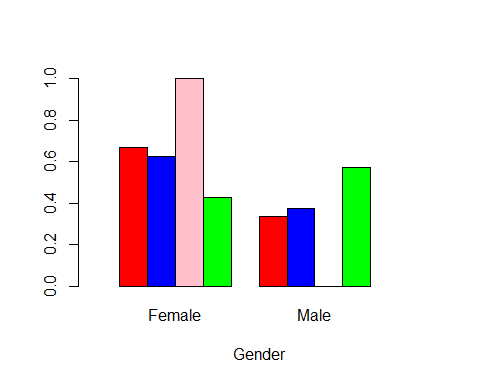
barplot(mg,xlim=c(0,2), xlab="Gender", length=levels(mo), col = c("red","blue", "pink","green"))



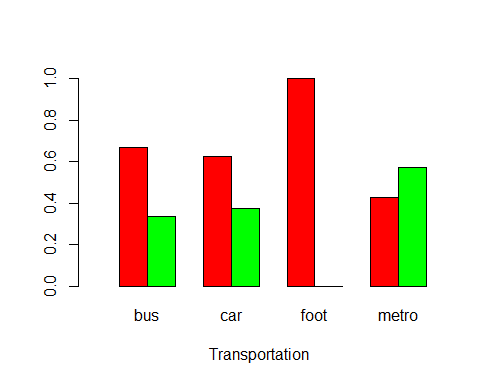
barplot(mg,xlim=c(0,2), xlab="Transportation", length=levels(g), col = c("red","blue", "pink","green"))



barplot(prop.table(mg,1), width=0.25, xlim=c(0,3), ylim=c(0,1), xlab="Gender", legend=levels(mo), beside=T, col = c("red","blue", "pink","green"))



mg = table(g,mo)  
barplot(prop.table(mg,2), width=0.25, xlim=c(0,3), ylim=c(0,1), xlab="Transportation", legend=levels(g), beside=T, col= c("red", "green"))



xo = c(10, 10, 5, 9, 7, 6,8,6,5,8, 10, 7, 7,8, 5, 6,4,7,9,7, 4,8, 10,10, 7,4,9,5,8,9)  
table(xo)

## xo  
## 4 5 6 7 8 9 10   
## 3 4 3 6 5 4 5

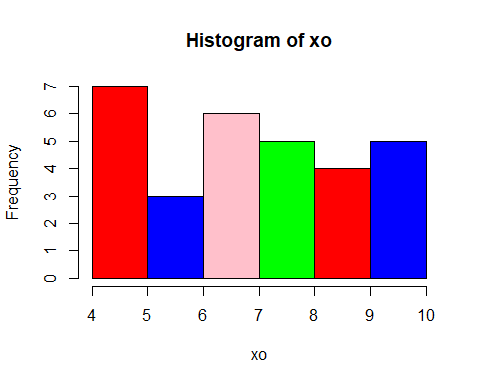
data.frame(xo)

## xo  
## 1 10  
## 2 10  
## 3 5  
## 4 9  
## 5 7  
## 6 6  
## 7 8  
## 8 6  
## 9 5  
## 10 8  
## 11 10  
## 12 7  
## 13 7  
## 14 8  
## 15 5  
## 16 6  
## 17 4  
## 18 7  
## 19 9  
## 20 7  
## 21 4  
## 22 8  
## 23 10  
## 24 10  
## 25 7  
## 26 4  
## 27 9  
## 28 5  
## 29 8  
## 30 9

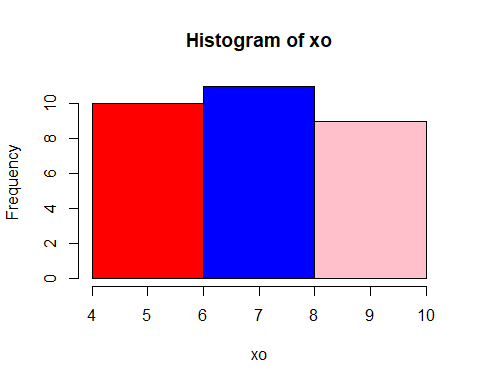
prop.table(table(xo))

## xo  
## 4 5 6 7 8 9 10   
## 0.1000000 0.1333333 0.1000000 0.2000000 0.1666667 0.1333333 0.1666667

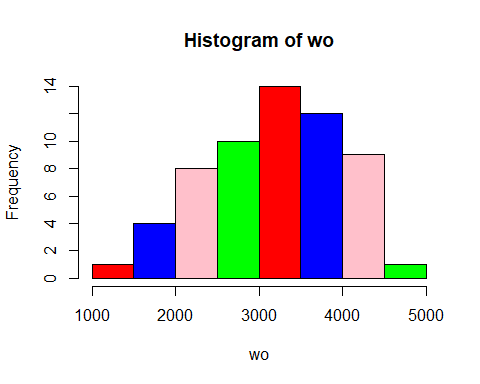
?prop.table  
  
### Histograms  
hist(xo, col = c("red","blue", "pink","green"))



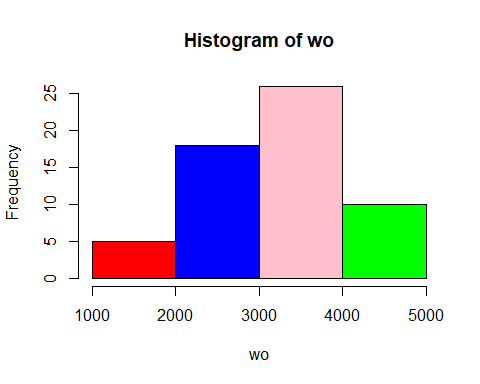
hist(xo, nclass=3, col = c("red","blue", "pink","green"))



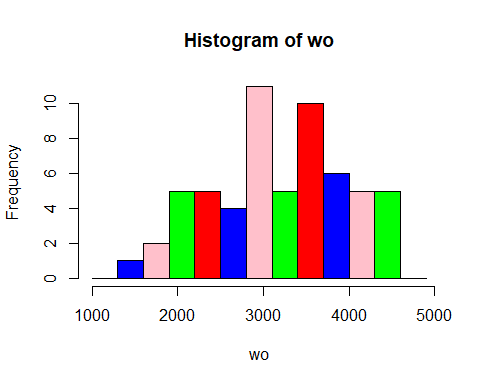
wo = c (1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550, 3050, 2350, 1850, 2820, 3670, 2950, 3750, 1850, 2420, 3150, 3000, 3470, 3920, 3100, 2400, 2900, 2650, 3450, 3650, 4020, 4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940, 3540, 2800, 4450, 1950, 3020, 2800, 3500, 1480, 4495,2850, 3100, 2250,3300, 4100, 3220, 3600,2130, 4020, 4075)  
hist(wo, col = c("red","blue", "pink","green"))



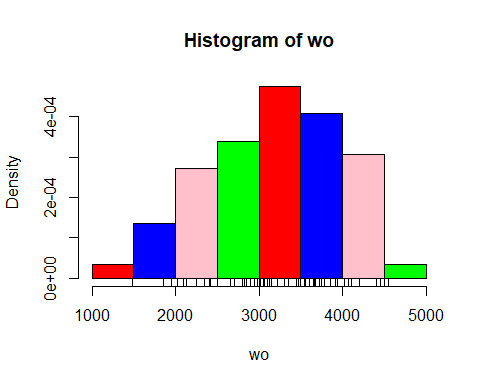
hist(wo, nclass=4, col = c("red","blue", "pink","green"))



hist(wo, breaks= seq(from = 1000, to=5000, by=300), col = c("red","blue", "pink","green"))



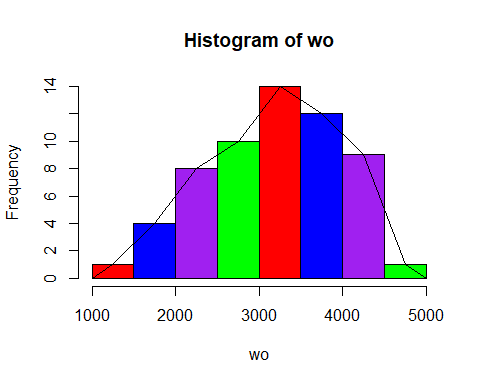
hist(wo, probability=T, col = c("red","blue", "pink","green"))  
rug(jitter(wo))



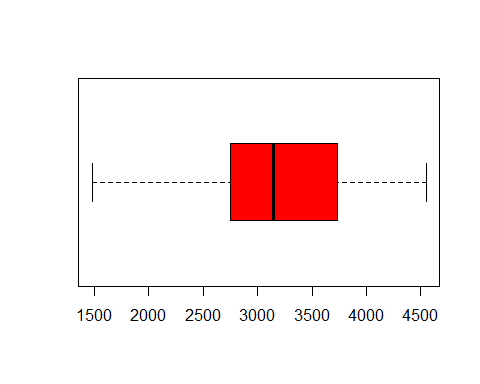
### Frequency Polygon  
temp = hist(wo, col = c("red","blue", "purple","green"))  
temp

## $breaks  
## [1] 1000 1500 2000 2500 3000 3500 4000 4500 5000  
##   
## $counts  
## [1] 1 4 8 10 14 12 9 1  
##   
## $density  
## [1] 3.389831e-05 1.355932e-04 2.711864e-04 3.389831e-04 4.745763e-04  
## [6] 4.067797e-04 3.050847e-04 3.389831e-05  
##   
## $mids  
## [1] 1250 1750 2250 2750 3250 3750 4250 4750  
##   
## $xname  
## [1] "wo"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

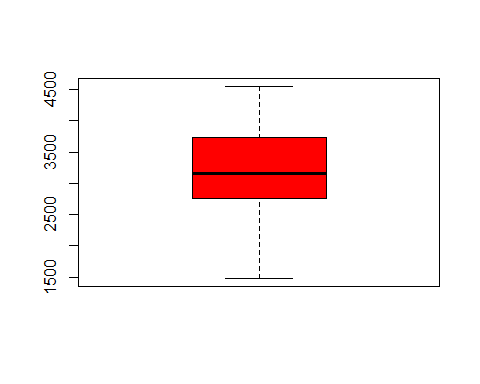
lines(c(min(temp$breaks), (temp$mids),max(temp$breaks)), c(0,temp$counts,0),type="l")



boxplot(wo, horizontal = T, col = c("red"))



boxplot(wo, vertical = T, col = c("red"))



fivenum(wo)

## [1] 1480 2750 3150 3735 4550

summary(wo)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1480 2750 3150 3195 3735 4550

?fivenum

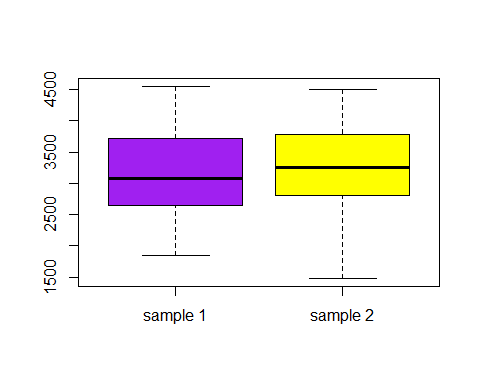
w1 = c(1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550, 3050, 2350,1850, 2820, 3670, 2950, 3750, 1850, 2420,3150, 3000, 3470, 3920, 3100, 2400, 2900, 2650, 3450, 3650, 4020)  
fivenum(w1)

## [1] 1850 2650 3075 3720 4550

w2 = c(4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940, 3340, 2800, 2850, 4450, 1950,3020,2800,3500,1480,4495,2850,3100,2250,3300,4100,3220,3600,2130,4020,4075)  
fivenum(w2)

## [1] 1480 2800 3260 3780 4495

boxplot(w1, w2, names=c("sample 1", "sample 2"), col = c("purple", "yellow" ))



# Classification and Prediction  
  
computeCost <- function(X, y, th){  
m <- length(y)  
return(1/(2\*m) \* sum((X%\*%th - y)^2))  
}  
  
computeCost(5,6,7)

## [1] 420.5

grad\_desc <- function(X, y, theta,alpha, lambda, num\_iters){  
m <- length(y)  
F\_history <- c(rep(0, num\_iters))  
  
for (iter in c(1:num\_iters)){  
 temp <- vector()  
 temp <- theta \* (1 - ((alpha\*lambda)/m)) - alpha\*(1/m) \* (t(X) %\*% (X %\*% theta - y))  
 theta <- temp  
 F\_history[iter] <- computeCost(X, y, theta)  
}  
print(F\_history[num\_iters])  
return(list("theta" = theta, "F\_history" = F\_history))  
}  
  
grad\_desc(2,3,5,0.1,7.5,2)

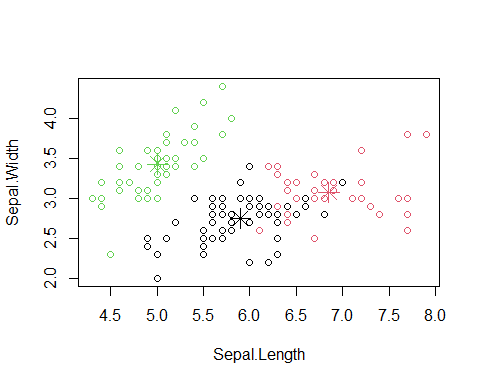
## [1] 1.540012

## $theta  
## [,1]  
## [1,] 0.6225  
##   
## $F\_history  
## [1] 5.445000 1.540012

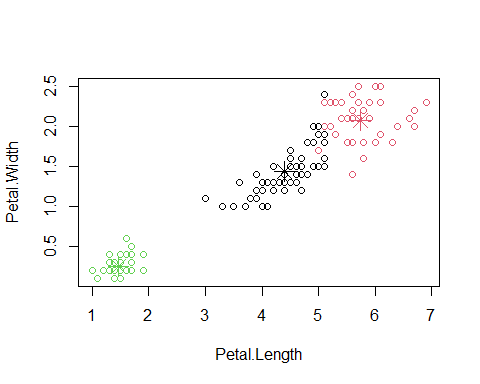
### Clustering  
iris\_new <- iris  
### iris\_new  
iris\_new$Species <- NULL  
kc <- kmeans(iris\_new, 3)  
table(iris$Species, kc$cluster)

##   
## 1 2 3  
## setosa 0 0 50  
## versicolor 48 2 0  
## virginica 14 36 0

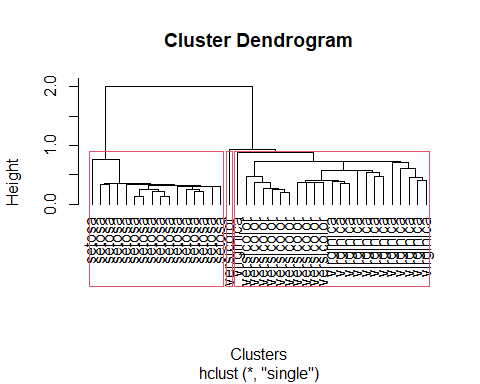
plot(iris\_new[c("Sepal.Length", "Sepal.Width")], col = kc$cluster)  
points(kc$centers[,c("Sepal.Length","Sepal.Width")], col = 1:3, pch=8, cex=2)



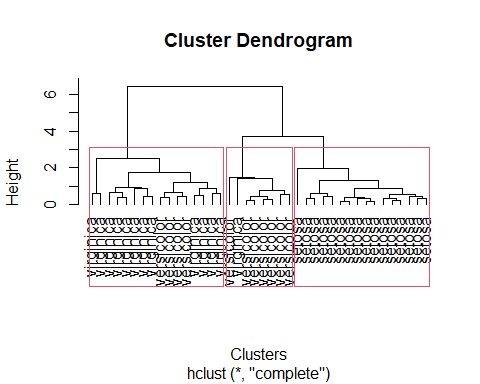
plot(iris\_new[c("Petal.Length", "Petal.Width")], col = kc$cluster)  
points(kc$centers[,c("Petal.Length","Petal.Width")], col=1:3, pch=8, cex=2)



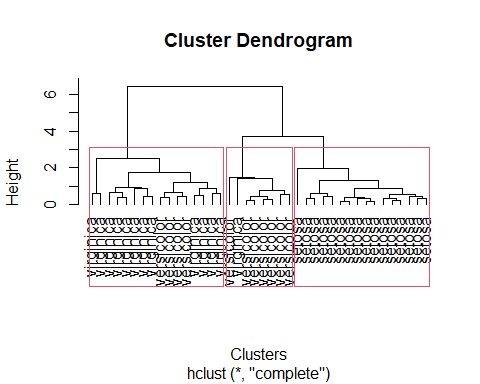
data(iris)  
set.seed(500)  
idx <- sample(1:dim(iris)[1], 40)  
iris\_Sample <- iris[idx,]  
iris\_Sample$Species <- NULL  
hc <- hclust(dist(iris\_Sample), method = "single")  
plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters")  
rect.hclust(hc, 3)



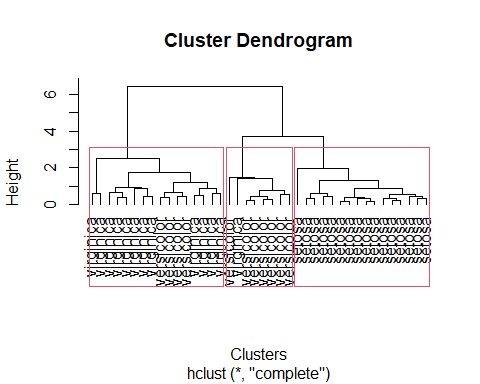
hc <- hclust(dist(iris\_Sample),method = "complete")  
plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters")  
rect.hclust(hc, 3)



data(iris)  
set.seed(500)  
idx <- sample(1:dim(iris)[1], 40)  
iris\_Sample <- iris[idx,]  
iris\_Sample$Species <- NULL  
hc <- hclust(dist(iris\_Sample), method = "complete")  
plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters")  
rect.hclust(hc, 3)



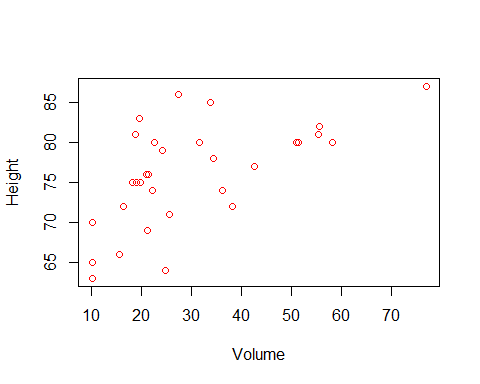
hc <- hclust(dist(iris\_Sample),method = "complete")  
plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters")  
rect.hclust(hc, 3)



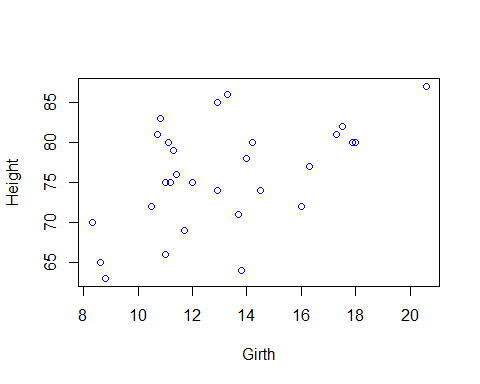
tr = trees  
tr

## Girth Height Volume  
## 1 8.3 70 10.3  
## 2 8.6 65 10.3  
## 3 8.8 63 10.2  
## 4 10.5 72 16.4  
## 5 10.7 81 18.8  
## 6 10.8 83 19.7  
## 7 11.0 66 15.6  
## 8 11.0 75 18.2  
## 9 11.1 80 22.6  
## 10 11.2 75 19.9  
## 11 11.3 79 24.2  
## 12 11.4 76 21.0  
## 13 11.4 76 21.4  
## 14 11.7 69 21.3  
## 15 12.0 75 19.1  
## 16 12.9 74 22.2  
## 17 12.9 85 33.8  
## 18 13.3 86 27.4  
## 19 13.7 71 25.7  
## 20 13.8 64 24.9  
## 21 14.0 78 34.5  
## 22 14.2 80 31.7  
## 23 14.5 74 36.3  
## 24 16.0 72 38.3  
## 25 16.3 77 42.6  
## 26 17.3 81 55.4  
## 27 17.5 82 55.7  
## 28 17.9 80 58.3  
## 29 18.0 80 51.5  
## 30 18.0 80 51.0  
## 31 20.6 87 77.0

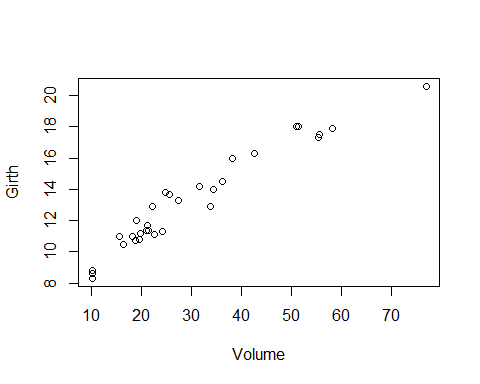
plot(tr[c("Volume","Height")], col = "red")



plot(tr[c("Girth","Height")], col = "blue" )



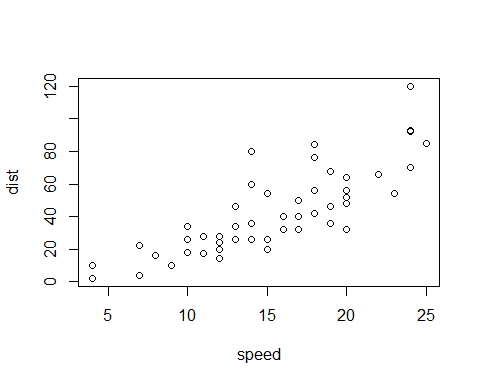
plot(tr[c("Volume","Girth")], col = "black" )



ca = cars  
ca

## speed dist  
## 1 4 2  
## 2 4 10  
## 3 7 4  
## 4 7 22  
## 5 8 16  
## 6 9 10  
## 7 10 18  
## 8 10 26  
## 9 10 34  
## 10 11 17  
## 11 11 28  
## 12 12 14  
## 13 12 20  
## 14 12 24  
## 15 12 28  
## 16 13 26  
## 17 13 34  
## 18 13 34  
## 19 13 46  
## 20 14 26  
## 21 14 36  
## 22 14 60  
## 23 14 80  
## 24 15 20  
## 25 15 26  
## 26 15 54  
## 27 16 32  
## 28 16 40  
## 29 17 32  
## 30 17 40  
## 31 17 50  
## 32 18 42  
## 33 18 56  
## 34 18 76  
## 35 18 84  
## 36 19 36  
## 37 19 46  
## 38 19 68  
## 39 20 32  
## 40 20 48  
## 41 20 52  
## 42 20 56  
## 43 20 64  
## 44 22 66  
## 45 23 54  
## 46 24 70  
## 47 24 92  
## 48 24 93  
## 49 24 120  
## 50 25 85

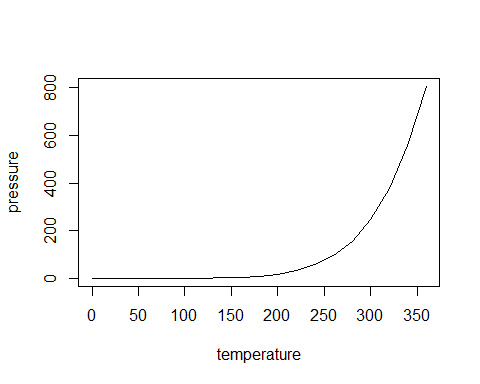
plot(ca[c("speed","dist")])



pre = pressure  
pre

## temperature pressure  
## 1 0 0.0002  
## 2 20 0.0012  
## 3 40 0.0060  
## 4 60 0.0300  
## 5 80 0.0900  
## 6 100 0.2700  
## 7 120 0.7500  
## 8 140 1.8500  
## 9 160 4.2000  
## 10 180 8.8000  
## 11 200 17.3000  
## 12 220 32.1000  
## 13 240 57.0000  
## 14 260 96.0000  
## 15 280 157.0000  
## 16 300 247.0000  
## 17 320 376.0000  
## 18 340 558.0000  
## 19 360 806.0000

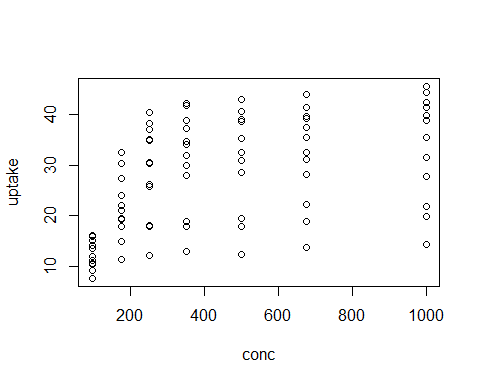
plot(pre[c("temperature","pressure")], type="l")



cab = CO2  
cab

## Plant Type Treatment conc uptake  
## 1 Qn1 Quebec nonchilled 95 16.0  
## 2 Qn1 Quebec nonchilled 175 30.4  
## 3 Qn1 Quebec nonchilled 250 34.8  
## 4 Qn1 Quebec nonchilled 350 37.2  
## 5 Qn1 Quebec nonchilled 500 35.3  
## 6 Qn1 Quebec nonchilled 675 39.2  
## 7 Qn1 Quebec nonchilled 1000 39.7  
## 8 Qn2 Quebec nonchilled 95 13.6  
## 9 Qn2 Quebec nonchilled 175 27.3  
## 10 Qn2 Quebec nonchilled 250 37.1  
## 11 Qn2 Quebec nonchilled 350 41.8  
## 12 Qn2 Quebec nonchilled 500 40.6  
## 13 Qn2 Quebec nonchilled 675 41.4  
## 14 Qn2 Quebec nonchilled 1000 44.3  
## 15 Qn3 Quebec nonchilled 95 16.2  
## 16 Qn3 Quebec nonchilled 175 32.4  
## 17 Qn3 Quebec nonchilled 250 40.3  
## 18 Qn3 Quebec nonchilled 350 42.1  
## 19 Qn3 Quebec nonchilled 500 42.9  
## 20 Qn3 Quebec nonchilled 675 43.9  
## 21 Qn3 Quebec nonchilled 1000 45.5  
## 22 Qc1 Quebec chilled 95 14.2  
## 23 Qc1 Quebec chilled 175 24.1  
## 24 Qc1 Quebec chilled 250 30.3  
## 25 Qc1 Quebec chilled 350 34.6  
## 26 Qc1 Quebec chilled 500 32.5  
## 27 Qc1 Quebec chilled 675 35.4  
## 28 Qc1 Quebec chilled 1000 38.7  
## 29 Qc2 Quebec chilled 95 9.3  
## 30 Qc2 Quebec chilled 175 27.3  
## 31 Qc2 Quebec chilled 250 35.0  
## 32 Qc2 Quebec chilled 350 38.8  
## 33 Qc2 Quebec chilled 500 38.6  
## 34 Qc2 Quebec chilled 675 37.5  
## 35 Qc2 Quebec chilled 1000 42.4  
## 36 Qc3 Quebec chilled 95 15.1  
## 37 Qc3 Quebec chilled 175 21.0  
## 38 Qc3 Quebec chilled 250 38.1  
## 39 Qc3 Quebec chilled 350 34.0  
## 40 Qc3 Quebec chilled 500 38.9  
## 41 Qc3 Quebec chilled 675 39.6  
## 42 Qc3 Quebec chilled 1000 41.4  
## 43 Mn1 Mississippi nonchilled 95 10.6  
## 44 Mn1 Mississippi nonchilled 175 19.2  
## 45 Mn1 Mississippi nonchilled 250 26.2  
## 46 Mn1 Mississippi nonchilled 350 30.0  
## 47 Mn1 Mississippi nonchilled 500 30.9  
## 48 Mn1 Mississippi nonchilled 675 32.4  
## 49 Mn1 Mississippi nonchilled 1000 35.5  
## 50 Mn2 Mississippi nonchilled 95 12.0  
## 51 Mn2 Mississippi nonchilled 175 22.0  
## 52 Mn2 Mississippi nonchilled 250 30.6  
## 53 Mn2 Mississippi nonchilled 350 31.8  
## 54 Mn2 Mississippi nonchilled 500 32.4  
## 55 Mn2 Mississippi nonchilled 675 31.1  
## 56 Mn2 Mississippi nonchilled 1000 31.5  
## 57 Mn3 Mississippi nonchilled 95 11.3  
## 58 Mn3 Mississippi nonchilled 175 19.4  
## 59 Mn3 Mississippi nonchilled 250 25.8  
## 60 Mn3 Mississippi nonchilled 350 27.9  
## 61 Mn3 Mississippi nonchilled 500 28.5  
## 62 Mn3 Mississippi nonchilled 675 28.1  
## 63 Mn3 Mississippi nonchilled 1000 27.8  
## 64 Mc1 Mississippi chilled 95 10.5  
## 65 Mc1 Mississippi chilled 175 14.9  
## 66 Mc1 Mississippi chilled 250 18.1  
## 67 Mc1 Mississippi chilled 350 18.9  
## 68 Mc1 Mississippi chilled 500 19.5  
## 69 Mc1 Mississippi chilled 675 22.2  
## 70 Mc1 Mississippi chilled 1000 21.9  
## 71 Mc2 Mississippi chilled 95 7.7  
## 72 Mc2 Mississippi chilled 175 11.4  
## 73 Mc2 Mississippi chilled 250 12.3  
## 74 Mc2 Mississippi chilled 350 13.0  
## 75 Mc2 Mississippi chilled 500 12.5  
## 76 Mc2 Mississippi chilled 675 13.7  
## 77 Mc2 Mississippi chilled 1000 14.4  
## 78 Mc3 Mississippi chilled 95 10.6  
## 79 Mc3 Mississippi chilled 175 18.0  
## 80 Mc3 Mississippi chilled 250 17.9  
## 81 Mc3 Mississippi chilled 350 17.9  
## 82 Mc3 Mississippi chilled 500 17.9  
## 83 Mc3 Mississippi chilled 675 18.9  
## 84 Mc3 Mississippi chilled 1000 19.9

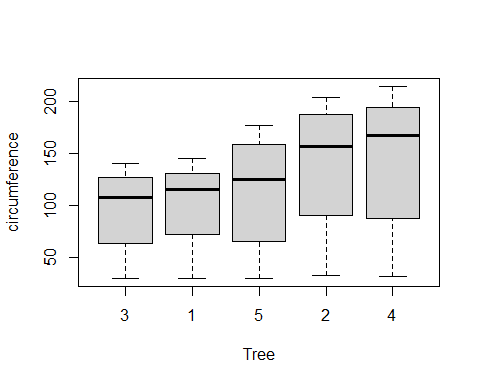
plot(cab[c("conc","uptake")])



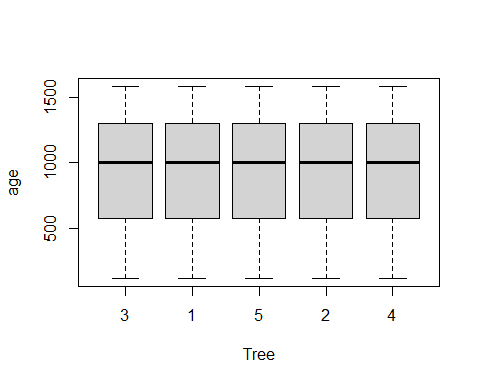
oran = Orange  
oran

## Tree age circumference  
## 1 1 118 30  
## 2 1 484 58  
## 3 1 664 87  
## 4 1 1004 115  
## 5 1 1231 120  
## 6 1 1372 142  
## 7 1 1582 145  
## 8 2 118 33  
## 9 2 484 69  
## 10 2 664 111  
## 11 2 1004 156  
## 12 2 1231 172  
## 13 2 1372 203  
## 14 2 1582 203  
## 15 3 118 30  
## 16 3 484 51  
## 17 3 664 75  
## 18 3 1004 108  
## 19 3 1231 115  
## 20 3 1372 139  
## 21 3 1582 140  
## 22 4 118 32  
## 23 4 484 62  
## 24 4 664 112  
## 25 4 1004 167  
## 26 4 1231 179  
## 27 4 1372 209  
## 28 4 1582 214  
## 29 5 118 30  
## 30 5 484 49  
## 31 5 664 81  
## 32 5 1004 125  
## 33 5 1231 142  
## 34 5 1372 174  
## 35 5 1582 177

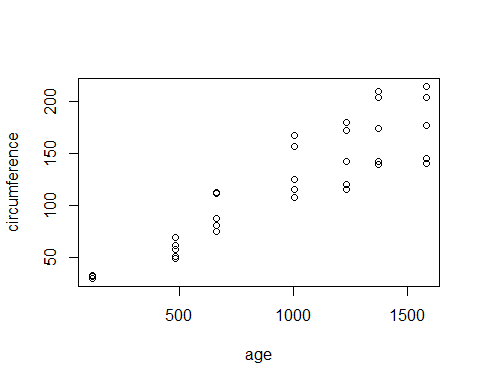
plot(oran[c("Tree","circumference")])



plot(oran[c("Tree","age")])



plot(oran[c("age","circumference")])



## MINING OF FREQUENT ITEMSETS AND ASSOCIATION RULES  
### Arules algorithm   
### install.packages('arules')  
library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':  
##   
## recode

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

db <- list(c("A", "B", "D", "E"), c("B", "C", "E"), c("A", "B", "D", "E"), c("A", "B", "C", "E"), c("A", "B", "C", "D", "E"), c("B", "C", "D"))  
  
frequent <- apriori(db, parameter=list(supp=0.5, conf=1, target="frequent itemsets"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.5 1  
## maxlen target ext  
## 10 frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[5 item(s), 6 transaction(s)] done [0.00s].  
## sorting and recoding items ... [5 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [19 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(frequent)

## items support count  
## [1] {C} 0.6666667 4   
## [2] {D} 0.6666667 4   
## [3] {A} 0.6666667 4   
## [4] {E} 0.8333333 5   
## [5] {B} 1.0000000 6   
## [6] {C, E} 0.5000000 3   
## [7] {B, C} 0.6666667 4   
## [8] {A, D} 0.5000000 3   
## [9] {D, E} 0.5000000 3   
## [10] {B, D} 0.6666667 4   
## [11] {A, E} 0.6666667 4   
## [12] {A, B} 0.6666667 4   
## [13] {B, E} 0.8333333 5   
## [14] {B, C, E} 0.5000000 3   
## [15] {A, D, E} 0.5000000 3   
## [16] {A, B, D} 0.5000000 3   
## [17] {B, D, E} 0.5000000 3   
## [18] {A, B, E} 0.6666667 4   
## [19] {A, B, D, E} 0.5000000 3

cl <- apriori(db, parameter=list(supp=0.5, conf=1, target="closed"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.5 1  
## maxlen target ext  
## 10 closed frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[5 item(s), 6 transaction(s)] done [0.00s].  
## sorting and recoding items ... [5 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## filtering closed item sets ... done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [7 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(cl)

## items support count  
## [1] {B} 1.0000000 6   
## [2] {B, C} 0.6666667 4   
## [3] {B, D} 0.6666667 4   
## [4] {B, E} 0.8333333 5   
## [5] {B, C, E} 0.5000000 3   
## [6] {A, B, E} 0.6666667 4   
## [7] {A, B, D, E} 0.5000000 3

mx <- apriori(db, parameter=list(supp=0.5, conf=1, target="maximal"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## NA 0.1 1 none FALSE TRUE 5 0.5 1  
## maxlen target ext  
## 10 maximally frequent itemsets TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[5 item(s), 6 transaction(s)] done [0.00s].  
## sorting and recoding items ... [5 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## filtering maximal item sets ... done [0.00s].  
## sorting transactions ... done [0.00s].  
## writing ... [2 set(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(mx)

## items support count  
## [1] {B, C, E} 0.5 3   
## [2] {A, B, D, E} 0.5 3

rules <- apriori(db, parameter=list(supp=0.5, conf=1, target="rules"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 1 0.1 1 none FALSE TRUE 5 0.5 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[5 item(s), 6 transaction(s)] done [0.00s].  
## sorting and recoding items ... [5 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 done [0.00s].  
## writing ... [16 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(rules)

## lhs rhs support confidence coverage lift count  
## [1] {} => {B} 1.0000000 1 1.0000000 1.0 6   
## [2] {C} => {B} 0.6666667 1 0.6666667 1.0 4   
## [3] {D} => {B} 0.6666667 1 0.6666667 1.0 4   
## [4] {A} => {E} 0.6666667 1 0.6666667 1.2 4   
## [5] {A} => {B} 0.6666667 1 0.6666667 1.0 4   
## [6] {E} => {B} 0.8333333 1 0.8333333 1.0 5   
## [7] {C, E} => {B} 0.5000000 1 0.5000000 1.0 3   
## [8] {A, D} => {E} 0.5000000 1 0.5000000 1.2 3   
## [9] {D, E} => {A} 0.5000000 1 0.5000000 1.5 3   
## [10] {A, D} => {B} 0.5000000 1 0.5000000 1.0 3   
## [11] {D, E} => {B} 0.5000000 1 0.5000000 1.0 3   
## [12] {A, E} => {B} 0.6666667 1 0.6666667 1.0 4   
## [13] {A, B} => {E} 0.6666667 1 0.6666667 1.2 4   
## [14] {A, D, E} => {B} 0.5000000 1 0.5000000 1.0 3   
## [15] {A, B, D} => {E} 0.5000000 1 0.5000000 1.2 3   
## [16] {B, D, E} => {A} 0.5000000 1 0.5000000 1.5 3

data(Adult)  
inspect(apriori(Adult, parameter=list(supp=0.75)))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.75 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 36631   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.06s].  
## sorting and recoding items ... [4 item(s)] done [0.00s].  
## creating transaction tree ... done [0.01s].  
## checking subsets of size 1 2 3 done [0.00s].  
## writing ... [19 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].  
## lhs rhs support confidence coverage lift count  
## [1] {} => {race=White} 0.8550428 0.8550428 1.0000000 1.0000000 41762  
## [2] {} => {native-country=United-States} 0.8974243 0.8974243 1.0000000 1.0000000 43832  
## [3] {} => {capital-gain=None} 0.9173867 0.9173867 1.0000000 1.0000000 44807  
## [4] {} => {capital-loss=None} 0.9532779 0.9532779 1.0000000 1.0000000 46560  
## [5] {race=White} => {native-country=United-States} 0.7881127 0.9217231 0.8550428 1.0270761 38493  
## [6] {native-country=United-States} => {race=White} 0.7881127 0.8781940 0.8974243 1.0270761 38493  
## [7] {race=White} => {capital-gain=None} 0.7817862 0.9143240 0.8550428 0.9966616 38184  
## [8] {capital-gain=None} => {race=White} 0.7817862 0.8521883 0.9173867 0.9966616 38184  
## [9] {race=White} => {capital-loss=None} 0.8136849 0.9516307 0.8550428 0.9982720 39742  
## [10] {capital-loss=None} => {race=White} 0.8136849 0.8535653 0.9532779 0.9982720 39742  
## [11] {native-country=United-States} => {capital-gain=None} 0.8219565 0.9159062 0.8974243 0.9983862 40146  
## [12] {capital-gain=None} => {native-country=United-States} 0.8219565 0.8959761 0.9173867 0.9983862 40146  
## [13] {native-country=United-States} => {capital-loss=None} 0.8548380 0.9525461 0.8974243 0.9992323 41752  
## [14] {capital-loss=None} => {native-country=United-States} 0.8548380 0.8967354 0.9532779 0.9992323 41752  
## [15] {capital-gain=None} => {capital-loss=None} 0.8706646 0.9490705 0.9173867 0.9955863 42525  
## [16] {capital-loss=None} => {capital-gain=None} 0.8706646 0.9133376 0.9532779 0.9955863 42525  
## [17] {capital-gain=None,   
## native-country=United-States} => {capital-loss=None} 0.7793702 0.9481891 0.8219565 0.9946618 38066  
## [18] {capital-loss=None,   
## native-country=United-States} => {capital-gain=None} 0.7793702 0.9117168 0.8548380 0.9938195 38066  
## [19] {capital-gain=None,   
## capital-loss=None} => {native-country=United-States} 0.7793702 0.8951440 0.8706646 0.9974590 38066

inspect(apriori(Adult, parameter=list(supp=0.75), appearance=list(rhs="capital-gain=None", default="lhs")))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.75 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 36631   
##   
## set item appearances ...[1 item(s)] done [0.00s].  
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.07s].  
## sorting and recoding items ... [4 item(s)] done [0.00s].  
## creating transaction tree ... done [0.01s].  
## checking subsets of size 1 2 3 done [0.00s].  
## writing ... [5 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].  
## lhs rhs support confidence coverage lift count  
## [1] {} => {capital-gain=None} 0.9173867 0.9173867 1.0000000 1.0000000 44807  
## [2] {race=White} => {capital-gain=None} 0.7817862 0.9143240 0.8550428 0.9966616 38184  
## [3] {native-country=United-States} => {capital-gain=None} 0.8219565 0.9159062 0.8974243 0.9983862 40146  
## [4] {capital-loss=None} => {capital-gain=None} 0.8706646 0.9133376 0.9532779 0.9955863 42525  
## [5] {capital-loss=None,   
## native-country=United-States} => {capital-gain=None} 0.7793702 0.9117168 0.8548380 0.9938195 38066