R Programming Lab Notebook

Knowledge Engineering Lab

Sanmitra Dharmavarapu

github url: <https://github.com/sanmitraD/Knowledge-Engineering-Lab-coursework>

# 1. Exploratory data analysis using R

Load the iris dataset and display the names and type of each column. Find statistics such as min, max, range, mean, median, variance and standard deviation for each column of data.

**Iris Data Set** Iris dataset os the best known dataset to be found in pattern recognition literature. The dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

*Attribute Information* 1. sepal length in cm

1. sepal width in cm
2. petal length in cm
3. petal width in cm
4. class
   * Iris Setosa
   * Iris Versicolour
   * Iris Virginica

# the Iris dataset is an inbulit dataset avalable in RStudio and we can use it using iris identifier   
print(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa  
## 7 4.6 3.4 1.4 0.3 setosa  
## 8 5.0 3.4 1.5 0.2 setosa  
## 9 4.4 2.9 1.4 0.2 setosa  
## 10 4.9 3.1 1.5 0.1 setosa  
## 11 5.4 3.7 1.5 0.2 setosa  
## 12 4.8 3.4 1.6 0.2 setosa  
## 13 4.8 3.0 1.4 0.1 setosa  
## 14 4.3 3.0 1.1 0.1 setosa  
## 15 5.8 4.0 1.2 0.2 setosa  
## 16 5.7 4.4 1.5 0.4 setosa  
## 17 5.4 3.9 1.3 0.4 setosa  
## 18 5.1 3.5 1.4 0.3 setosa  
## 19 5.7 3.8 1.7 0.3 setosa  
## 20 5.1 3.8 1.5 0.3 setosa  
## 21 5.4 3.4 1.7 0.2 setosa  
## 22 5.1 3.7 1.5 0.4 setosa  
## 23 4.6 3.6 1.0 0.2 setosa  
## 24 5.1 3.3 1.7 0.5 setosa  
## 25 4.8 3.4 1.9 0.2 setosa  
## 26 5.0 3.0 1.6 0.2 setosa  
## 27 5.0 3.4 1.6 0.4 setosa  
## 28 5.2 3.5 1.5 0.2 setosa  
## 29 5.2 3.4 1.4 0.2 setosa  
## 30 4.7 3.2 1.6 0.2 setosa  
## 31 4.8 3.1 1.6 0.2 setosa  
## 32 5.4 3.4 1.5 0.4 setosa  
## 33 5.2 4.1 1.5 0.1 setosa  
## 34 5.5 4.2 1.4 0.2 setosa  
## 35 4.9 3.1 1.5 0.2 setosa  
## 36 5.0 3.2 1.2 0.2 setosa  
## 37 5.5 3.5 1.3 0.2 setosa  
## 38 4.9 3.6 1.4 0.1 setosa  
## 39 4.4 3.0 1.3 0.2 setosa  
## 40 5.1 3.4 1.5 0.2 setosa  
## 41 5.0 3.5 1.3 0.3 setosa  
## 42 4.5 2.3 1.3 0.3 setosa  
## 43 4.4 3.2 1.3 0.2 setosa  
## 44 5.0 3.5 1.6 0.6 setosa  
## 45 5.1 3.8 1.9 0.4 setosa  
## 46 4.8 3.0 1.4 0.3 setosa  
## 47 5.1 3.8 1.6 0.2 setosa  
## 48 4.6 3.2 1.4 0.2 setosa  
## 49 5.3 3.7 1.5 0.2 setosa  
## 50 5.0 3.3 1.4 0.2 setosa  
## 51 7.0 3.2 4.7 1.4 versicolor  
## 52 6.4 3.2 4.5 1.5 versicolor  
## 53 6.9 3.1 4.9 1.5 versicolor  
## 54 5.5 2.3 4.0 1.3 versicolor  
## 55 6.5 2.8 4.6 1.5 versicolor  
## 56 5.7 2.8 4.5 1.3 versicolor  
## 57 6.3 3.3 4.7 1.6 versicolor  
## 58 4.9 2.4 3.3 1.0 versicolor  
## 59 6.6 2.9 4.6 1.3 versicolor  
## 60 5.2 2.7 3.9 1.4 versicolor  
## 61 5.0 2.0 3.5 1.0 versicolor  
## 62 5.9 3.0 4.2 1.5 versicolor  
## 63 6.0 2.2 4.0 1.0 versicolor  
## 64 6.1 2.9 4.7 1.4 versicolor  
## 65 5.6 2.9 3.6 1.3 versicolor  
## 66 6.7 3.1 4.4 1.4 versicolor  
## 67 5.6 3.0 4.5 1.5 versicolor  
## 68 5.8 2.7 4.1 1.0 versicolor  
## 69 6.2 2.2 4.5 1.5 versicolor  
## 70 5.6 2.5 3.9 1.1 versicolor  
## 71 5.9 3.2 4.8 1.8 versicolor  
## 72 6.1 2.8 4.0 1.3 versicolor  
## 73 6.3 2.5 4.9 1.5 versicolor  
## 74 6.1 2.8 4.7 1.2 versicolor  
## 75 6.4 2.9 4.3 1.3 versicolor  
## 76 6.6 3.0 4.4 1.4 versicolor  
## 77 6.8 2.8 4.8 1.4 versicolor  
## 78 6.7 3.0 5.0 1.7 versicolor  
## 79 6.0 2.9 4.5 1.5 versicolor  
## 80 5.7 2.6 3.5 1.0 versicolor  
## 81 5.5 2.4 3.8 1.1 versicolor  
## 82 5.5 2.4 3.7 1.0 versicolor  
## 83 5.8 2.7 3.9 1.2 versicolor  
## 84 6.0 2.7 5.1 1.6 versicolor  
## 85 5.4 3.0 4.5 1.5 versicolor  
## 86 6.0 3.4 4.5 1.6 versicolor  
## 87 6.7 3.1 4.7 1.5 versicolor  
## 88 6.3 2.3 4.4 1.3 versicolor  
## 89 5.6 3.0 4.1 1.3 versicolor  
## 90 5.5 2.5 4.0 1.3 versicolor  
## 91 5.5 2.6 4.4 1.2 versicolor  
## 92 6.1 3.0 4.6 1.4 versicolor  
## 93 5.8 2.6 4.0 1.2 versicolor  
## 94 5.0 2.3 3.3 1.0 versicolor  
## 95 5.6 2.7 4.2 1.3 versicolor  
## 96 5.7 3.0 4.2 1.2 versicolor  
## 97 5.7 2.9 4.2 1.3 versicolor  
## 98 6.2 2.9 4.3 1.3 versicolor  
## 99 5.1 2.5 3.0 1.1 versicolor  
## 100 5.7 2.8 4.1 1.3 versicolor  
## 101 6.3 3.3 6.0 2.5 virginica  
## 102 5.8 2.7 5.1 1.9 virginica  
## 103 7.1 3.0 5.9 2.1 virginica  
## 104 6.3 2.9 5.6 1.8 virginica  
## 105 6.5 3.0 5.8 2.2 virginica  
## 106 7.6 3.0 6.6 2.1 virginica  
## 107 4.9 2.5 4.5 1.7 virginica  
## 108 7.3 2.9 6.3 1.8 virginica  
## 109 6.7 2.5 5.8 1.8 virginica  
## 110 7.2 3.6 6.1 2.5 virginica  
## 111 6.5 3.2 5.1 2.0 virginica  
## 112 6.4 2.7 5.3 1.9 virginica  
## 113 6.8 3.0 5.5 2.1 virginica  
## 114 5.7 2.5 5.0 2.0 virginica  
## 115 5.8 2.8 5.1 2.4 virginica  
## 116 6.4 3.2 5.3 2.3 virginica  
## 117 6.5 3.0 5.5 1.8 virginica  
## 118 7.7 3.8 6.7 2.2 virginica  
## 119 7.7 2.6 6.9 2.3 virginica  
## 120 6.0 2.2 5.0 1.5 virginica  
## 121 6.9 3.2 5.7 2.3 virginica  
## 122 5.6 2.8 4.9 2.0 virginica  
## 123 7.7 2.8 6.7 2.0 virginica  
## 124 6.3 2.7 4.9 1.8 virginica  
## 125 6.7 3.3 5.7 2.1 virginica  
## 126 7.2 3.2 6.0 1.8 virginica  
## 127 6.2 2.8 4.8 1.8 virginica  
## 128 6.1 3.0 4.9 1.8 virginica  
## 129 6.4 2.8 5.6 2.1 virginica  
## 130 7.2 3.0 5.8 1.6 virginica  
## 131 7.4 2.8 6.1 1.9 virginica  
## 132 7.9 3.8 6.4 2.0 virginica  
## 133 6.4 2.8 5.6 2.2 virginica  
## 134 6.3 2.8 5.1 1.5 virginica  
## 135 6.1 2.6 5.6 1.4 virginica  
## 136 7.7 3.0 6.1 2.3 virginica  
## 137 6.3 3.4 5.6 2.4 virginica  
## 138 6.4 3.1 5.5 1.8 virginica  
## 139 6.0 3.0 4.8 1.8 virginica  
## 140 6.9 3.1 5.4 2.1 virginica  
## 141 6.7 3.1 5.6 2.4 virginica  
## 142 6.9 3.1 5.1 2.3 virginica  
## 143 5.8 2.7 5.1 1.9 virginica  
## 144 6.8 3.2 5.9 2.3 virginica  
## 145 6.7 3.3 5.7 2.5 virginica  
## 146 6.7 3.0 5.2 2.3 virginica  
## 147 6.3 2.5 5.0 1.9 virginica  
## 148 6.5 3.0 5.2 2.0 virginica  
## 149 6.2 3.4 5.4 2.3 virginica  
## 150 5.9 3.0 5.1 1.8 virginica

To display the names and types of each column

# to display names of each column we use the names functon  
  
print(names(iris))

## [1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"

For diplaying the names along with their types we use the lapply function using class as second argument

print(lapply(iris,class))

## $Sepal.Length  
## [1] "numeric"  
##   
## $Sepal.Width  
## [1] "numeric"  
##   
## $Petal.Length  
## [1] "numeric"  
##   
## $Petal.Width  
## [1] "numeric"  
##   
## $Species  
## [1] "factor"

*Finding the mean of individual columns*

# we can find mean of a column using mean function  
  
# we can refer to individual columns in a datset using $ symbol as showin below  
  
print(mean(iris$Sepal.Length))

## [1] 5.843333

print(mean(iris$Sepal.Width))

## [1] 3.057333

print(mean(iris$Petal.Length))

## [1] 3.758

print(mean(iris$Petal.Width))

## [1] 1.199333

*Finding the median of each column*

# we can find median of a column using median function  
  
  
print(median(iris$Sepal.Length))

## [1] 5.8

print(median(iris$Sepal.Width))

## [1] 3

print(median(iris$Petal.Length))

## [1] 4.35

print(median(iris$Petal.Width))

## [1] 1.3

*Finding maximum value of each column*

# we can find maximum of a column using max function  
  
  
print(max(iris$Sepal.Length))

## [1] 7.9

print(max(iris$Sepal.Width))

## [1] 4.4

print(max(iris$Petal.Length))

## [1] 6.9

print(max(iris$Petal.Width))

## [1] 2.5

*Finding minimum value of each column* returns a vector containing the minimum and maximum of all the given arguments.

# we can find maximum of a column using min function  
  
  
print(min(iris$Sepal.Length))

## [1] 4.3

print(min(iris$Sepal.Width))

## [1] 2

print(min(iris$Petal.Length))

## [1] 1

print(min(iris$Petal.Width))

## [1] 0.1

*Finding the range of each column* We use range() to finding the range of each column. It returns a vector containing the minimum and maximum of all the given arguments.

print(range(iris$Sepal.Length))

## [1] 4.3 7.9

print(range(iris$Sepal.Width))

## [1] 2.0 4.4

print(range(iris$Petal.Length))

## [1] 1.0 6.9

print(range(iris$Petal.Width))

## [1] 0.1 2.5

*Finding variance of each column* we use var() function to find the variance of each column

print(var(iris$Sepal.Length))

## [1] 0.6856935

print(var(iris$Sepal.Width))

## [1] 0.1899794

print(var(iris$Petal.Length))

## [1] 3.116278

print(var(iris$Petal.Width))

## [1] 0.5810063

*Finding standard deviation of each column* We use sd() function to find the variance of each column

print(sd(iris$Sepal.Length))

## [1] 0.8280661

print(sd(iris$Sepal.Width))

## [1] 0.4358663

print(sd(iris$Petal.Length))

## [1] 1.765298

print(sd(iris$Petal.Width))

## [1] 0.7622377

**using summary function** we can use summary() function to get all the above details in a single go.

summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

# 2. R program to normalize the variables into 0 to 1 scale using min-max normalization

the formula to achieve min max normalization is y = (x-min)/(max-min)

#dummy data  
x = sample(-100:100,50)  
print("original data")

## [1] "original data"

print(x)

## [1] -5 96 13 51 48 -54 74 -28 81 76 -99 14 63 -41 -21  
## [16] -97 80 -94 72 68 -6 -56 -13 93 84 67 89 40 -33 -60  
## [31] -20 49 -32 87 -100 35 -66 -4 24 -79 60 -58 37 -48 -38  
## [46] -63 33 -12 -71 -26

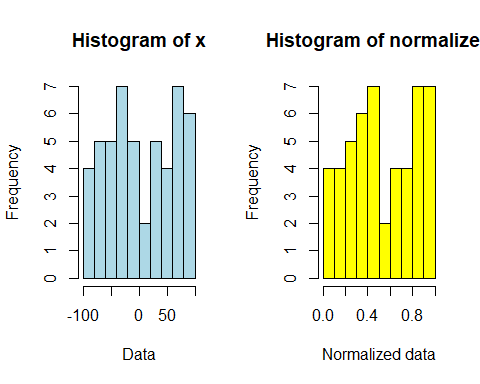
maximum = max(x)  
minimum = min(x)  
normalized = (x-minimum)/(maximum-minimum)  
print("Normalized data")

## [1] "Normalized data"

print(normalized)

## [1] 0.484693878 1.000000000 0.576530612 0.770408163 0.755102041 0.234693878  
## [7] 0.887755102 0.367346939 0.923469388 0.897959184 0.005102041 0.581632653  
## [13] 0.831632653 0.301020408 0.403061224 0.015306122 0.918367347 0.030612245  
## [19] 0.877551020 0.857142857 0.479591837 0.224489796 0.443877551 0.984693878  
## [25] 0.938775510 0.852040816 0.964285714 0.714285714 0.341836735 0.204081633  
## [31] 0.408163265 0.760204082 0.346938776 0.954081633 0.000000000 0.688775510  
## [37] 0.173469388 0.489795918 0.632653061 0.107142857 0.816326531 0.214285714  
## [43] 0.698979592 0.265306122 0.316326531 0.188775510 0.678571429 0.448979592  
## [49] 0.147959184 0.377551020

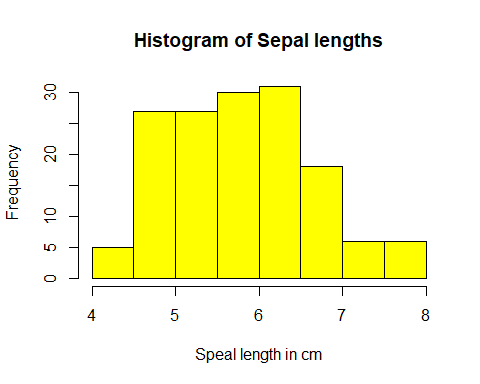
#using par function to fix multiple graphs in same plot  
  
par(mfrow=c(1,2))  
hist(x,breaks = 10, xlab = "Data",col = "lightblue", )  
hist(normalized, breaks = 10, xlab = "Normalized data", col = "yellow")



# 3.Generate histograms for any one variable and generate scatter plots for every pair of variables showing each species in different colour on iris dataset.

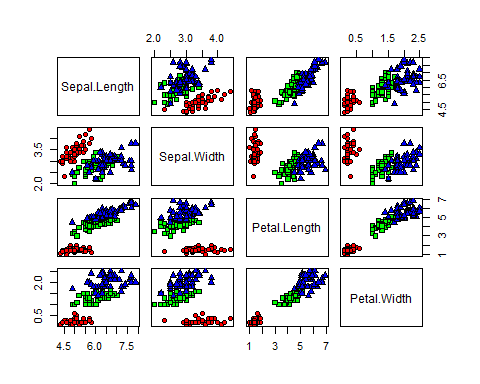
Generating histogram for any one variable let it be sepal length

hist(iris$Sepal.Length, col="yellow", xlab = "Speal length in cm", main = "Histogram of Sepal lengths")

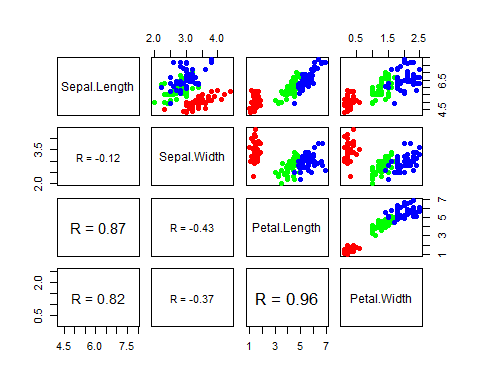


Let us use red, green, blue as the colours for 3 species

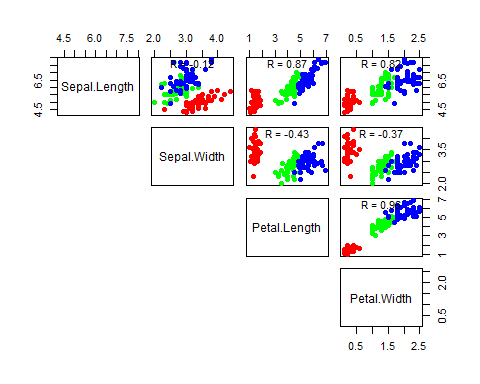
my\_cols=c("red","green","blue")  
#21 is for circle, 22 is for squares 24 is for triangles  
pairs(iris[1:4],pch=c(21,22,24)[iris$Species],bg=my\_cols[iris$Species])



#correlation panel  
panel.cor = function(x,y){  
 usr<- par("usr");on.exit(par(usr))  
 par(usr = c(0,1,0,1))  
 r = round(cor(x,y), digits = 2)  
 txt = paste0("R = ",r)  
 cex.cor = 0.8/strwidth(txt)  
 text(0.5,0.5,txt,cex=cex.cor\*r)  
}  
  
#customizing panels and printing correlations  
#customize upper panel  
upper.panel = function(x,y){  
 points(x,y,pch=19,col=my\_cols[iris$Species])  
}  
  
#create the plots  
pairs(iris[1:4],lower.panel = panel.cor, upper.panel = upper.panel)



#customize upper panel  
upper.panel = function(x,y){  
 points(x,y,pch=19,col = my\_cols[iris$Species])  
 r = round(cor(x,y),digits=2)  
 txt = paste0("R = ",r)  
 usr = par("usr");on.exit(par(usr))  
 par(usr = c(0,1,0,1))  
 text(0.5,0.9,txt)  
}  
  
pairs(iris[1:4],lower.panel = NULL,upper.panel = upper.panel)



# 4. Generate box plots for each of the numerical attribute. Identify the attribute with highest variance.

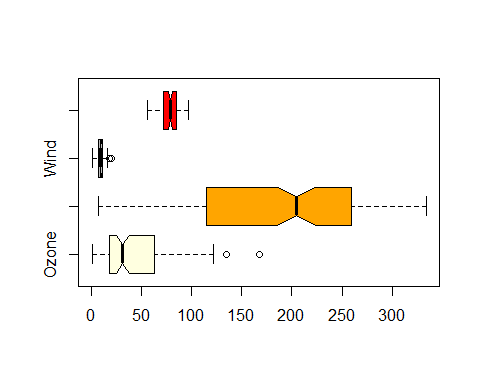
We will be using buliting dataset “airquality” dataset for this program. It is a Daily air quality measurements in New York, May to September 1973.

Finding variance is simple. The spread of the boxplot indicates the variance. The more the spread of boxplot then it have more variance.

#str(airquality)  
#Here we dont need the Day and Month values  
#removing rows with null values  
#boxplot does not consider the null values (NAs) so no need to remove them  
airq = airquality[1:4]  
str(airq)

## 'data.frame': 153 obs. of 4 variables:  
## $ Ozone : int 41 36 12 18 NA 28 23 19 8 NA ...  
## $ Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...  
## $ Wind : num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...  
## $ Temp : int 67 72 74 62 56 66 65 59 61 69 ...

boxplot(airq,notch = TRUE,horizontal = TRUE,col=(c("lightyellow","orange","gray","red")))



As we can see Solar.R which refers to Solar Radiation has more area of spread. Hence it has greater variance compared to other attributes

# 5.Study of homogeneous and heterogeneous data structures such as vector, matrix, array, list and dataframe in R.

**Vector**

Vector is a basic data structure in R that contains the element of similar type.The data types of components of vector can be logical, integer, double, character, complex or raw.

#Creating a vector  
studentname = c("Sanmitra", "Dharmavarapu")  
print(studentname)

## [1] "Sanmitra" "Dharmavarapu"

#Getting the class of the vector  
print(class(studentname))

## [1] "character"

**Lists** A list is an R object that can contain many different type of components in it. The components can vectors, fuctions and even another list and many more.

#Creating a list.  
list = list(c("Sanmitra","Dharmavarapu"),"CSE",316129510013,sin)  
print(list)

## [[1]]  
## [1] "Sanmitra" "Dharmavarapu"  
##   
## [[2]]  
## [1] "CSE"  
##   
## [[3]]  
## [1] 316129510013  
##   
## [[4]]  
## function (x) .Primitive("sin")

**Matrices**

A matrix is a two dimensional reactangular data structure that contain list of homogeneous data in tabular format.It is possible to perform arithmetic operations on whole or a part of a matrix. A matrix can created using a vector input to the matrix() function

M = matrix(c(1,2,3,4,5,6,7,8,9,10),nrow = 2,ncol = 5,byrow = TRUE)  
print(M)

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

**Array** While matrices are confined to two dimensions, arrays can be of any number of dimensions.It can store data in more than one dimensions.

For example if we create an array of dimension(10,20,30) then it creates 30 matrices each with 10 rows and 20 columns. These can store elements of only type.

v1 = c(1,2,3)  
v2 = c(4,5,6,7,8,9)  
  
arr = array(c(v1,v2),dim=c(3,3,2))  
print(arr)

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

**Data Frames**

Data frame is a tabular data object or two dimensional array like structure in which each column, contains values of one variable and each row contains one set of values from each column.

Unlike matrices, each column of a data frame can contain different modes of data.

costs = data.frame(  
 name = c("carrot","apple","sugar"),  
 costPerKG = c(50.00,60.00,39.50),  
 QuantityAvailableinKGs = c(10,5,50))  
print(costs)

## name costPerKG QuantityAvailableinKGs  
## 1 carrot 50.0 10  
## 2 apple 60.0 5  
## 3 sugar 39.5 50