Intro to R

#### Execute the following cells to load the libraries

library(ggplot2)  
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

#### Creating homeogenous datastructures: vectors and matrices (built on the fundamental datatypes character, numeric, integer, and logical)

myvector1 = c(1, 2, 3) # a vector of integers/numeric?  
myvector2 = c('a', 'b', 'c') # a vector of characters  
myvector3 = c(TRUE, FALSE, FALSE) # a vector of booleans  
myvector4 = c(1.5, 2.5, 3.5) # a vector of numerics  
myvector5 = 1:10 # creates a vector comprising 1 through 10  
myvector6 = seq(1, 10, by = 0.5) # Creates a vector comprising 1, 1.5, 2.0 etc  
print(myvector1)

## [1] 1 2 3

print(myvector2)

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

print(myvector3)

## [1] TRUE FALSE FALSE

print(myvector4)

## [1] 1.5 2.5 3.5

print(myvector5)

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

print(myvector6)

## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0  
## [16] 8.5 9.0 9.5 10.0

mymatrix1 = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)  
mymatrix2 = matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)  
print(mymatrix1)

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

print(mymatrix2)

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

#### Creating a list, a heterogenous datastructure

mylist1 = list(1, 'Name', c('Math', 'Physics', 'Chemistry'))  
print(mylist1)

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] "Name"  
##   
## [[3]]  
## [1] "Math" "Physics" "Chemistry"

#### Accessing elements of a datastructure

myvector1[1]

## [1] 1

#myvector7[10]  
mymatrix2[1, 3]

## [1] 3

mymatrix2[1, ]

## [1] 1 2 3

mymatrix2[, 2]

## [1] 2 5

#### Loading data into a dataframe, a heterogenous datastructure

#file = 'food-texture.csv'  
file = 'Data/food-texture.csv'  
foodData = read.csv(file, header = TRUE, row.names = 1, stringsAsFactors = FALSE)  
head(foodData, 5)

## Oil Density Crispy Fracture Hardness  
## B110 16.5 2955 10 23 97  
## B136 17.7 2660 14 9 139  
## B171 16.2 2870 12 17 143  
## B192 16.7 2920 10 31 95  
## B225 16.3 2975 11 26 143

#### Attributes of a dataframe

str(foodData)

## 'data.frame': 50 obs. of 5 variables:  
## $ Oil : num 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 ...  
## $ Density : int 2955 2660 2870 2920 2975 2790 2750 2770 2955 2945 ...  
## $ Crispy : int 10 14 12 10 11 13 13 10 11 11 ...  
## $ Fracture: int 23 9 17 31 26 16 17 26 23 24 ...  
## $ Hardness: int 97 139 143 95 143 189 114 63 123 132 ...

nrow(foodData)

## [1] 50

ncol(foodData)

## [1] 5

colnames(foodData)

## [1] "Oil" "Density" "Crispy" "Fracture" "Hardness"

rownames(foodData)

## [1] "B110" "B136" "B171" "B192" "B225" "B237" "B261" "B264" "B353" "B360"  
## [11] "B366" "B377" "B391" "B397" "B404" "B437" "B445" "B462" "B485" "B488"  
## [21] "B502" "B554" "B556" "B575" "B576" "B605" "B612" "B615" "B649" "B665"  
## [31] "B674" "B692" "B694" "B719" "B727" "B758" "B776" "B799" "B836" "B848"  
## [41] "B861" "B869" "B876" "B882" "B889" "B907" "B911" "B923" "B971" "B998"

#### Get the data type and the data structure associated with an object

typeof(myvector1)

## [1] "double"

typeof(myvector2)

## [1] "character"

str(myvector1)

## num [1:3] 1 2 3

str(myvector2)

## chr [1:3] "a" "b" "c"

str(foodData)

## 'data.frame': 50 obs. of 5 variables:  
## $ Oil : num 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 ...  
## $ Density : int 2955 2660 2870 2920 2975 2790 2750 2770 2955 2945 ...  
## $ Crispy : int 10 14 12 10 11 13 13 10 11 11 ...  
## $ Fracture: int 23 9 17 31 26 16 17 26 23 24 ...  
## $ Hardness: int 97 139 143 95 143 189 114 63 123 132 ...

#### Accessing elements of a data frame in a particular column

foodData$Oil

## [1] 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 18.0 17.4 18.4 13.9 15.8  
## [16] 16.4 18.9 17.3 16.7 19.1 13.7 14.7 18.1 17.2 18.7 18.1 16.6 17.1 17.4 19.4  
## [31] 15.9 17.1 15.5 17.7 15.9 21.2 19.5 20.5 17.0 16.7 16.8 16.8 16.3 16.2 18.1  
## [46] 16.6 16.4 15.1 21.1 16.3

foodData[['Oil']]

## [1] 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 18.0 17.4 18.4 13.9 15.8  
## [16] 16.4 18.9 17.3 16.7 19.1 13.7 14.7 18.1 17.2 18.7 18.1 16.6 17.1 17.4 19.4  
## [31] 15.9 17.1 15.5 17.7 15.9 21.2 19.5 20.5 17.0 16.7 16.8 16.8 16.3 16.2 18.1  
## [46] 16.6 16.4 15.1 21.1 16.3

foodData[, 'Oil']

## [1] 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 18.0 17.4 18.4 13.9 15.8  
## [16] 16.4 18.9 17.3 16.7 19.1 13.7 14.7 18.1 17.2 18.7 18.1 16.6 17.1 17.4 19.4  
## [31] 15.9 17.1 15.5 17.7 15.9 21.2 19.5 20.5 17.0 16.7 16.8 16.8 16.3 16.2 18.1  
## [46] 16.6 16.4 15.1 21.1 16.3

foodData['Oil']

## Oil  
## B110 16.5  
## B136 17.7  
## B171 16.2  
## B192 16.7  
## B225 16.3  
## B237 19.1  
## B261 18.4  
## B264 17.5  
## B353 15.7  
## B360 16.4  
## B366 18.0  
## B377 17.4  
## B391 18.4  
## B397 13.9  
## B404 15.8  
## B437 16.4  
## B445 18.9  
## B462 17.3  
## B485 16.7  
## B488 19.1  
## B502 13.7  
## B554 14.7  
## B556 18.1  
## B575 17.2  
## B576 18.7  
## B605 18.1  
## B612 16.6  
## B615 17.1  
## B649 17.4  
## B665 19.4  
## B674 15.9  
## B692 17.1  
## B694 15.5  
## B719 17.7  
## B727 15.9  
## B758 21.2  
## B776 19.5  
## B799 20.5  
## B836 17.0  
## B848 16.7  
## B861 16.8  
## B869 16.8  
## B876 16.3  
## B882 16.2  
## B889 18.1  
## B907 16.6  
## B911 16.4  
## B923 15.1  
## B971 21.1  
## B998 16.3

foodData[1, 'Oil']

## [1] 16.5

foodData['B110', 'Oil']

## [1] 16.5

#### Accessing elements of multiple columns of a dataframe (using base R and dplyr)

foodData[c('Oil', 'Density')]

## Oil Density  
## B110 16.5 2955  
## B136 17.7 2660  
## B171 16.2 2870  
## B192 16.7 2920  
## B225 16.3 2975  
## B237 19.1 2790  
## B261 18.4 2750  
## B264 17.5 2770  
## B353 15.7 2955  
## B360 16.4 2945  
## B366 18.0 2830  
## B377 17.4 2835  
## B391 18.4 2860  
## B397 13.9 2965  
## B404 15.8 2930  
## B437 16.4 2770  
## B445 18.9 2650  
## B462 17.3 2890  
## B485 16.7 2695  
## B488 19.1 2755  
## B502 13.7 3000  
## B554 14.7 2980  
## B556 18.1 2780  
## B575 17.2 2705  
## B576 18.7 2825  
## B605 18.1 2875  
## B612 16.6 2945  
## B615 17.1 2920  
## B649 17.4 2845  
## B665 19.4 2645  
## B674 15.9 3080  
## B692 17.1 2825  
## B694 15.5 3125  
## B719 17.7 2780  
## B727 15.9 2900  
## B758 21.2 2570  
## B776 19.5 2635  
## B799 20.5 2725  
## B836 17.0 2865  
## B848 16.7 2975  
## B861 16.8 2980  
## B869 16.8 2870  
## B876 16.3 2920  
## B882 16.2 3100  
## B889 18.1 2910  
## B907 16.6 2865  
## B911 16.4 2995  
## B923 15.1 2925  
## B971 21.1 2700  
## B998 16.3 2845

foodData %>% select(c(Oil, Density))

## Oil Density  
## B110 16.5 2955  
## B136 17.7 2660  
## B171 16.2 2870  
## B192 16.7 2920  
## B225 16.3 2975  
## B237 19.1 2790  
## B261 18.4 2750  
## B264 17.5 2770  
## B353 15.7 2955  
## B360 16.4 2945  
## B366 18.0 2830  
## B377 17.4 2835  
## B391 18.4 2860  
## B397 13.9 2965  
## B404 15.8 2930  
## B437 16.4 2770  
## B445 18.9 2650  
## B462 17.3 2890  
## B485 16.7 2695  
## B488 19.1 2755  
## B502 13.7 3000  
## B554 14.7 2980  
## B556 18.1 2780  
## B575 17.2 2705  
## B576 18.7 2825  
## B605 18.1 2875  
## B612 16.6 2945  
## B615 17.1 2920  
## B649 17.4 2845  
## B665 19.4 2645  
## B674 15.9 3080  
## B692 17.1 2825  
## B694 15.5 3125  
## B719 17.7 2780  
## B727 15.9 2900  
## B758 21.2 2570  
## B776 19.5 2635  
## B799 20.5 2725  
## B836 17.0 2865  
## B848 16.7 2975  
## B861 16.8 2980  
## B869 16.8 2870  
## B876 16.3 2920  
## B882 16.2 3100  
## B889 18.1 2910  
## B907 16.6 2865  
## B911 16.4 2995  
## B923 15.1 2925  
## B971 21.1 2700  
## B998 16.3 2845

#foodData[-c('Oil', 'Density')]  
foodData %>% select(-c(Oil, Density))

## Crispy Fracture Hardness  
## B110 10 23 97  
## B136 14 9 139  
## B171 12 17 143  
## B192 10 31 95  
## B225 11 26 143  
## B237 13 16 189  
## B261 13 17 114  
## B264 10 26 63  
## B353 11 23 123  
## B360 11 24 132  
## B366 12 15 121  
## B377 12 18 172  
## B391 14 11 170  
## B397 12 19 169  
## B404 9 26 65  
## B437 15 16 183  
## B445 14 20 114  
## B462 12 17 142  
## B485 13 13 111  
## B488 14 10 140  
## B502 10 27 177  
## B554 10 20 133  
## B556 13 14 150  
## B575 8 27 113  
## B576 13 20 166  
## B605 12 15 150  
## B612 10 25 100  
## B615 10 25 123  
## B649 13 19 129  
## B665 12 18 68  
## B674 10 23 106  
## B692 10 28 131  
## B694 7 33 92  
## B719 13 22 141  
## B727 12 21 192  
## B758 14 13 105  
## B776 13 22 101  
## B799 14 16 145  
## B836 11 22 100  
## B848 10 26 105  
## B861 10 24 144  
## B869 12 20 123  
## B876 11 22 136  
## B882 8 27 140  
## B889 12 21 120  
## B907 11 25 120  
## B911 12 20 165  
## B923 10 29 118  
## B971 13 16 116  
## B998 10 26 75

#### Load the court dataset

file = 'Data/court.csv'  
courtData = read.csv(file, header = TRUE)  
head(courtData, 5)

## Judge.A Judge.B Judge.C Judge.D Judge.E  
## 1 1 1 1 0 0  
## 2 1 1 0 1 0  
## 3 1 1 1 0 1  
## 4 1 1 1 1 1  
## 5 1 1 1 1 1

#### For-loop and if-statement

count = 0  
for (j in 1:nrow(foodData)){  
 if (foodData[j, 'Oil'] >= 16){  
 count = count +1  
 }  
}  
print(count / nrow(foodData))

## [1] 0.82

#### User-defined function

judgeEdecision = function(n){  
 decision = sample(c(0, 1), n, replace = TRUE, prob = c(0.2, 0.8))  
 return(decision)  
}

#### Calling the user-defined function

judgeEdecision(10)

## [1] 1 0 0 1 1 1 1 1 0 1

#### Matrix example - judges problem

courtdecision = function(){  
 jcp = apply(courtData, 2, mean)  
 jicp = 1-jcp # broadcasting  
 P = matrix(c(jcp, jicp), nrow = 2, ncol = length(jcp), byrow = TRUE)  
 decision = c(1, 0)  
 cdecision = vector('integer', ncol(P))  
 for (j in 1:ncol(P)){  
 cdecision[j] = sample(decision, 1, prob = P[, j])  
 }  
 if (sum(cdecision) >= 3){  
 return(1)  
 }  
 else{  
 return(0)  
}  
}

#### Court incorrect probability

results = replicate(10000, courtdecision())  
1-mean(results)

## [1] 0.0061

#### Accessing rows of a dataframe satisfying certain conditions

foodData[foodData$Oil > 17, ]

## Oil Density Crispy Fracture Hardness  
## B136 17.7 2660 14 9 139  
## B237 19.1 2790 13 16 189  
## B261 18.4 2750 13 17 114  
## B264 17.5 2770 10 26 63  
## B366 18.0 2830 12 15 121  
## B377 17.4 2835 12 18 172  
## B391 18.4 2860 14 11 170  
## B445 18.9 2650 14 20 114  
## B462 17.3 2890 12 17 142  
## B488 19.1 2755 14 10 140  
## B556 18.1 2780 13 14 150  
## B575 17.2 2705 8 27 113  
## B576 18.7 2825 13 20 166  
## B605 18.1 2875 12 15 150  
## B615 17.1 2920 10 25 123  
## B649 17.4 2845 13 19 129  
## B665 19.4 2645 12 18 68  
## B692 17.1 2825 10 28 131  
## B719 17.7 2780 13 22 141  
## B758 21.2 2570 14 13 105  
## B776 19.5 2635 13 22 101  
## B799 20.5 2725 14 16 145  
## B889 18.1 2910 12 21 120  
## B971 21.1 2700 13 16 116

foodData %>% filter(Oil > 17)

## Oil Density Crispy Fracture Hardness  
## B136 17.7 2660 14 9 139  
## B237 19.1 2790 13 16 189  
## B261 18.4 2750 13 17 114  
## B264 17.5 2770 10 26 63  
## B366 18.0 2830 12 15 121  
## B377 17.4 2835 12 18 172  
## B391 18.4 2860 14 11 170  
## B445 18.9 2650 14 20 114  
## B462 17.3 2890 12 17 142  
## B488 19.1 2755 14 10 140  
## B556 18.1 2780 13 14 150  
## B575 17.2 2705 8 27 113  
## B576 18.7 2825 13 20 166  
## B605 18.1 2875 12 15 150  
## B615 17.1 2920 10 25 123  
## B649 17.4 2845 13 19 129  
## B665 19.4 2645 12 18 68  
## B692 17.1 2825 10 28 131  
## B719 17.7 2780 13 22 141  
## B758 21.2 2570 14 13 105  
## B776 19.5 2635 13 22 101  
## B799 20.5 2725 14 16 145  
## B889 18.1 2910 12 21 120  
## B971 21.1 2700 13 16 116

#### Accessing row of a dataframe satisfying certain condition and particular columns

foodData[foodData$Oil > 17, c('Crispy', 'Hardness')]

## Crispy Hardness  
## B136 14 139  
## B237 13 189  
## B261 13 114  
## B264 10 63  
## B366 12 121  
## B377 12 172  
## B391 14 170  
## B445 14 114  
## B462 12 142  
## B488 14 140  
## B556 13 150  
## B575 8 113  
## B576 13 166  
## B605 12 150  
## B615 10 123  
## B649 13 129  
## B665 12 68  
## B692 10 131  
## B719 13 141  
## B758 14 105  
## B776 13 101  
## B799 14 145  
## B889 12 120  
## B971 13 116

foodData %>% filter(Oil > 17) %>% select(Crispy, Hardness)

## Crispy Hardness  
## B136 14 139  
## B237 13 189  
## B261 13 114  
## B264 10 63  
## B366 12 121  
## B377 12 172  
## B391 14 170  
## B445 14 114  
## B462 12 142  
## B488 14 140  
## B556 13 150  
## B575 8 113  
## B576 13 166  
## B605 12 150  
## B615 10 123  
## B649 13 129  
## B665 12 68  
## B692 10 131  
## B719 13 141  
## B758 14 105  
## B776 13 101  
## B799 14 145  
## B889 12 120  
## B971 13 116

#### Accessing rows of a dataframe satisfying certain conditions

#Filter samples with Crispy index 9 or 15 (which are very rare)  
foodData[(foodData$Crispy == 9) | (foodData$Crispy == 15), ]

## Oil Density Crispy Fracture Hardness  
## B404 15.8 2930 9 26 65  
## B437 16.4 2770 15 16 183

foodData %>% filter(Crispy == 9 | Crispy == 15)

## Oil Density Crispy Fracture Hardness  
## B404 15.8 2930 9 26 65  
## B437 16.4 2770 15 16 183

foodData %>% filter(Crispy %in% c(9, 15))

## Oil Density Crispy Fracture Hardness  
## B404 15.8 2930 9 26 65  
## B437 16.4 2770 15 16 183

#Filter samples with Crispy index not 9 or 15  
foodData %>% filter(!(Crispy %in% c(9, 15)))

## Oil Density Crispy Fracture Hardness  
## B110 16.5 2955 10 23 97  
## B136 17.7 2660 14 9 139  
## B171 16.2 2870 12 17 143  
## B192 16.7 2920 10 31 95  
## B225 16.3 2975 11 26 143  
## B237 19.1 2790 13 16 189  
## B261 18.4 2750 13 17 114  
## B264 17.5 2770 10 26 63  
## B353 15.7 2955 11 23 123  
## B360 16.4 2945 11 24 132  
## B366 18.0 2830 12 15 121  
## B377 17.4 2835 12 18 172  
## B391 18.4 2860 14 11 170  
## B397 13.9 2965 12 19 169  
## B445 18.9 2650 14 20 114  
## B462 17.3 2890 12 17 142  
## B485 16.7 2695 13 13 111  
## B488 19.1 2755 14 10 140  
## B502 13.7 3000 10 27 177  
## B554 14.7 2980 10 20 133  
## B556 18.1 2780 13 14 150  
## B575 17.2 2705 8 27 113  
## B576 18.7 2825 13 20 166  
## B605 18.1 2875 12 15 150  
## B612 16.6 2945 10 25 100  
## B615 17.1 2920 10 25 123  
## B649 17.4 2845 13 19 129  
## B665 19.4 2645 12 18 68  
## B674 15.9 3080 10 23 106  
## B692 17.1 2825 10 28 131  
## B694 15.5 3125 7 33 92  
## B719 17.7 2780 13 22 141  
## B727 15.9 2900 12 21 192  
## B758 21.2 2570 14 13 105  
## B776 19.5 2635 13 22 101  
## B799 20.5 2725 14 16 145  
## B836 17.0 2865 11 22 100  
## B848 16.7 2975 10 26 105  
## B861 16.8 2980 10 24 144  
## B869 16.8 2870 12 20 123  
## B876 16.3 2920 11 22 136  
## B882 16.2 3100 8 27 140  
## B889 18.1 2910 12 21 120  
## B907 16.6 2865 11 25 120  
## B911 16.4 2995 12 20 165  
## B923 15.1 2925 10 29 118  
## B971 21.1 2700 13 16 116  
## B998 16.3 2845 10 26 75

#### Modifying a column of a dataframe

# Insert a new column Densitynew with density values in gm/cm^3  
foodData %>% mutate(Densitynew = Density\*1000)

## Oil Density Crispy Fracture Hardness Densitynew  
## B110 16.5 2955 10 23 97 2955000  
## B136 17.7 2660 14 9 139 2660000  
## B171 16.2 2870 12 17 143 2870000  
## B192 16.7 2920 10 31 95 2920000  
## B225 16.3 2975 11 26 143 2975000  
## B237 19.1 2790 13 16 189 2790000  
## B261 18.4 2750 13 17 114 2750000  
## B264 17.5 2770 10 26 63 2770000  
## B353 15.7 2955 11 23 123 2955000  
## B360 16.4 2945 11 24 132 2945000  
## B366 18.0 2830 12 15 121 2830000  
## B377 17.4 2835 12 18 172 2835000  
## B391 18.4 2860 14 11 170 2860000  
## B397 13.9 2965 12 19 169 2965000  
## B404 15.8 2930 9 26 65 2930000  
## B437 16.4 2770 15 16 183 2770000  
## B445 18.9 2650 14 20 114 2650000  
## B462 17.3 2890 12 17 142 2890000  
## B485 16.7 2695 13 13 111 2695000  
## B488 19.1 2755 14 10 140 2755000  
## B502 13.7 3000 10 27 177 3000000  
## B554 14.7 2980 10 20 133 2980000  
## B556 18.1 2780 13 14 150 2780000  
## B575 17.2 2705 8 27 113 2705000  
## B576 18.7 2825 13 20 166 2825000  
## B605 18.1 2875 12 15 150 2875000  
## B612 16.6 2945 10 25 100 2945000  
## B615 17.1 2920 10 25 123 2920000  
## B649 17.4 2845 13 19 129 2845000  
## B665 19.4 2645 12 18 68 2645000  
## B674 15.9 3080 10 23 106 3080000  
## B692 17.1 2825 10 28 131 2825000  
## B694 15.5 3125 7 33 92 3125000  
## B719 17.7 2780 13 22 141 2780000  
## B727 15.9 2900 12 21 192 2900000  
## B758 21.2 2570 14 13 105 2570000  
## B776 19.5 2635 13 22 101 2635000  
## B799 20.5 2725 14 16 145 2725000  
## B836 17.0 2865 11 22 100 2865000  
## B848 16.7 2975 10 26 105 2975000  
## B861 16.8 2980 10 24 144 2980000  
## B869 16.8 2870 12 20 123 2870000  
## B876 16.3 2920 11 22 136 2920000  
## B882 16.2 3100 8 27 140 3100000  
## B889 18.1 2910 12 21 120 2910000  
## B907 16.6 2865 11 25 120 2865000  
## B911 16.4 2995 12 20 165 2995000  
## B923 15.1 2925 10 29 118 2925000  
## B971 21.1 2700 13 16 116 2700000  
## B998 16.3 2845 10 26 75 2845000

# Overwrite the dataframe to add the new column in place  
foodData = foodData %>% mutate(Densitynew = Density\*1000)  
colnames(foodData)

## [1] "Oil" "Density" "Crispy" "Fracture" "Hardness"   
## [6] "Densitynew"

#### Modify Crispy column to reflect high (0) or low (1) crispiness

foodData = foodData %>% mutate(Crispylevel = ifelse(Crispy > 11, 'High', 'Low'))

##### Change Crispy and Crispylevel columns to factor (categorical) type

# Continuous features -> Oil, Density, Hardness, Fracture  
# Categorical features -> Crispy (8 levels 8 through 15) and Crispylevel (2 levels 0 and 1)  
# Crispy is a categorical feature with an order  
categorical\_cols = c('Crispy', 'Crispylevel')  
str(foodData)

## 'data.frame': 50 obs. of 7 variables:  
## $ Oil : num 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 ...  
## $ Density : int 2955 2660 2870 2920 2975 2790 2750 2770 2955 2945 ...  
## $ Crispy : int 10 14 12 10 11 13 13 10 11 11 ...  
## $ Fracture : int 23 9 17 31 26 16 17 26 23 24 ...  
## $ Hardness : int 97 139 143 95 143 189 114 63 123 132 ...  
## $ Densitynew : num 2955000 2660000 2870000 2920000 2975000 ...  
## $ Crispylevel: chr "Low" "High" "High" "Low" ...

foodData[categorical\_cols] = lapply(foodData[categorical\_cols], as.factor)  
str(foodData)

## 'data.frame': 50 obs. of 7 variables:  
## $ Oil : num 16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 ...  
## $ Density : int 2955 2660 2870 2920 2975 2790 2750 2770 2955 2945 ...  
## $ Crispy : Factor w/ 9 levels "7","8","9","10",..: 4 8 6 4 5 7 7 4 5 5 ...  
## $ Fracture : int 23 9 17 31 26 16 17 26 23 24 ...  
## $ Hardness : int 97 139 143 95 143 189 114 63 123 132 ...  
## $ Densitynew : num 2955000 2660000 2870000 2920000 2975000 ...  
## $ Crispylevel: Factor w/ 2 levels "High","Low": 2 1 1 2 2 1 1 2 2 2 ...

#### Visualize the OilPercentage feature

#### In-built functions for dataframes

# Mean oil percentage across all samples  
  
# Mean-centering of OilPercentage  
  
# Sum of the squared deviation from the mean  
  
# Average of the squared deviation from the mean  
  
# Variance of OilPercentage  
  
# Standard deviation of OilPercentage

#### Scatter plot between OilPercentage and Density

#### Scatter plot between OilPercentage and Density color coded with Crispylevel

#### Scale continuous columns of the dataframe

#### Calculate the correlation matrix for the continuous features

#### Scatter plot between Density and Hardness

#### Scatter plot between Density and Fracture