I do it for fun

Charles Kwame Appiah

2025-04-09

Contents

Theme

"bootstrap", "cerulean", "cosmo", "darkly", "flatly", "journal", "lumen", "paper", "readable", "sandstone", "simplex", "spacelab", "united", "yeti"

Data Types

Data types are used to represent different types of data in R. R has several built-in data types, such as `numeric`, `character`, `integer`, `complex`, and `logical`. Data types are used to store different types of data, such as numbers, text, and logical values.

```
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 # calculating the square of pi
y
```

```
## [1] 9.869604
```

```
class(y) # checking the class of variable y
```

```
## [1] "numeric"
```

```
z <- 15L # assigning 15 to variable z
z
```

```
## [1] 15
```

```
class(z) # checking the class of variable z
```

```
## [1] "integer"
```

```
a <- (5 + 2i)^2 # calculating the square of (5 + 2i)
a
```

```
## [1] 21+20i
```

```
class(a) # checking the class of variable a
```

```
## [1] "complex"
```

```r
l <- TRUE # assigning TRUE to variable l
class(l) # checking the class of variable l
```

```
## [1] "logical"
```

```r
x <- list(age = c(10,21,22), weight = c(30,33,32)) # creating a list x
x
```

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

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

```
## [1] "age"    "weight"
```

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

```
## [1] 2
```

```r
kx <- data.frame(age = c(11,14,22), weight = c(30,33,32)) # creating a dataframe xk
kx
```

```
##   age weight
## 1  11     30
## 2  14     33
## 3  22     32
```

```r
d <- c("Charles Kwame Appiah is my name") # creating a vector
d
```

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

```r
class(d) # checking the class of variable d
```

```
## [1] "character"
```

```r
length(d) # checking the length of variable d
```

```
## [1] 1
```

```r
nchar(d) # checking the number of characters in variable d
```

```
## [1] 31
```

```
f<- c(1,3,4,6,7) # creating a vector
f
```

```
## [1] 1 3 4 6 7
```

```
class(f)# checking the class of variable f
```

```
## [1] "numeric"
```

Integers

Integers in R are stored as numeric data type. To create an integer in R, you can use the `L` suffix or the `as.integer()` function.

```
2 + 3
```

```
## 5
```

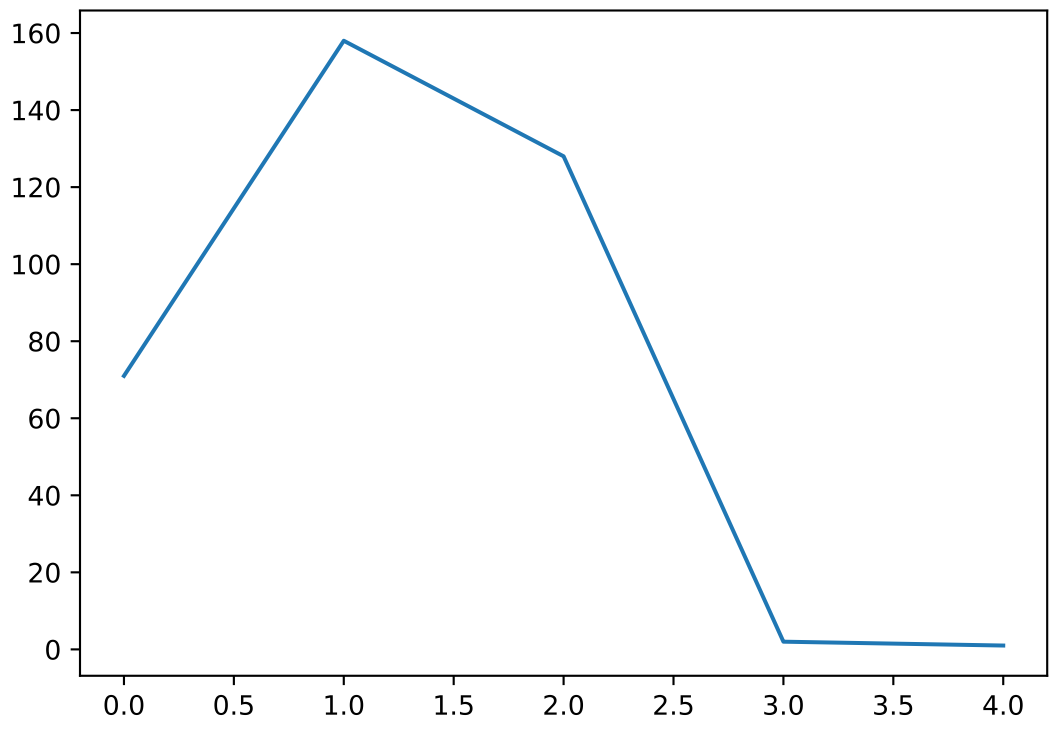
```
import pandas as pd

df = pd.read_csv("C:/Users/HP/Desktop/Data Science/Machine learning/kyphosis.csv")

df.info
```

```
## <bound method DataFrame.info of    Kyphosis  Age  Number  Start
## 0    absent   71       3      5
## 1    absent  158       3     14
## 2   present  128       4      5
## 3    absent    2       5      1
## 4    absent    1       4     15
## ..      ...  ...     ...    ...
## 76  present  157       3     13
## 77   absent   26       7     13
## 78   absent  120       2     13
## 79  present   42       7      6
## 80   absent   36       4     13
##
## [81 rows x 4 columns]>
```

```
df.describe()
```

```
##               Age     Number      Start
## count   81.000000  81.000000  81.000000
## mean    83.654321   4.049383  11.493827
## std     58.104251   1.619423   4.883962
## min      1.000000   2.000000   1.000000
## 25%     26.000000   3.000000   9.000000
## 50%     87.000000   4.000000  13.000000
## 75%    130.000000   5.000000  16.000000
## max    206.000000  10.000000  18.000000
```

```
df["Age"].head().plot()
```




```
fo<- c(1L,3L,4L,6L,7L) # creating a vector
fo
```

```
## [1] 1 3 4 6 7
```

```
class(fo) # checking the class of variable fo
```

```
## [1] "integer"
```

Vectors

Vectors can be created with the `c()` function. The `c()` function can take multiple arguments and combine them into a single vector. The `c()` function can also be used to combine multiple vectors into a single vector.

```
# Initialization
x <- vector(mode = "logical", length = 5) # creating a vector with 5 logical elements
x
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```r
class(x) # checking the class of variable x
```

```
## [1] "logical"
```

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

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

Logical

Logical vectors are created with the `c()` function. Logical vectors can be used to store TRUE or FALSE values. Logical vectors are used in R to represent binary data.Binary data is data that can take on one of two values, such as TRUE or FALSE.

```r
s <- c(TRUE, FALSE,TRUE, 1) # creating a vector
s
```

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

```r
as.logical(s) # converting the vector to logical
```

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

List

List is a collection of multiple objects. The objects can be of different classes. Lists are created with the `list()` function

```r
q <- list("Hello World",2015L, TRUE, 32.1) # creating a list
q
```

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

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

```
## [1] "integer"
```

```r
class(q[[4]]) # Checking the class of list 4
```

```
## [1] "numeric"
```

```r
class(q[[1]]) # Checking the class of list 1
```

```
## [1] "character"
```

Matrix

Matrix is a two-dimensional array. It is created with the `matrix()` function. The `matrix()` function takes a vector as input and reshapes it into a matrix. The `nrow` argument specifies the number of rows in the matrix, and the `ncol` argument specifies the number of columns in the matrix.

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

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

```r
mar = c(2,4,5,7,8,9) # creating a vector
dim(mar) <- c(3,2) # creating a matrix with 3 rows and 2 columns
mar
```

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

```r
temp <- c(3,4,5,5.6,6,7) # creating a vector
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
```

```r
temp <- c(3,4,5,5.6,6,7, 8, 9, 10) # creating a vector
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
```

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

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

```r
t1 <- c(23, 55) # creating a vector
t2 <- c(34, 45) # creating a vector

# By rows
rbind(t1,t2) # binding them by their rows
```

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

```r
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
```

Factors and Nominal Data

Factors are used to represent categorical data in R. Factors are created with the `factor()` function. The `factor()` function takes a vector as input and converts it into a factor. Factors are used to represent nominal data, which is data that has no inherent order.

```r
factor <- c("Yes","No","No","Yes") # creating a vector
factor # use to encode vectors as factors
```

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

```r
f <- factor(c("Yes","No","No","Yes"), levels = c("Yes","No"), ordered = TRUE) # creating a factor
f
```

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

Missing Values

Missing values are represented by the `NA` value in R. The `is.na()` function can be used to check if a value is missing. The `is.na()` function returns `TRUE` if the value is missing and `FALSE` otherwise.

```
x <- NA # Missing number
x
```

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

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

```
## [1] TRUE
```

```
u <- 0/0 # Missing number
u
```

```
## [1] NaN
```

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

```
## [1] "numeric"
```

Dataframe

Data frames are used to store tabular data in R. Data frames are created with the `data.frame()` function. The `data.frame()` function takes vectors as input and combines them into a data frame. Data frames are used to store data in a structured format.

```
a <- c("Charles","Richmond","Nicholas") # creating a vector
b <- c(12, 23, 45) # creating a vector
c <- c(FALSE,TRUE,TRUE) # creating a vector

dfr <- data.frame(Username = a,Age = b, Adult = c) # creating a dataframe
dfr
```

```
##   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
```

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

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

```r
# Selecting only Age Column
dfr$Age
```

```
## [1] 12 23 45
```

```r
# Selecting only Username Column
dfr$Username
```

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

```r
# Selecting only Adult Column
dfr$Adult
```

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

Reading Data From File

Reading data from a file is a common task in data analysis. R provides functions to read data from various file formats, such as CSV, Excel, and text files. The `read.csv()` function is used to read data from a CSV file. The `read.csv()` function takes the path to the CSV file as input and returns a data frame. Sample data is provided in the `Training r.csv` file.

```r
### Importing Datasets
library(readxl) # Importing the readxl package

dat <- read.csv("C:/Users/HP/Desktop/Data Science/Machine learning/Training r.csv") # reading the datas
#print(dat, na.rm = TRUE)
head(dat) # displaying the first few rows of the dataset
```

```
##   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
```

```r
data <- read_xlsx("C:/Users/HP/Desktop/Data Science/Pandas/Copy of V1- UN Data on Refugees (AiCE __ Data
 head(data) # displaying the first few rows of the dataset
```

```
## # A tibble: 6 x 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 Creation

Sequence is a collection of numbers in a specific order. Sequences can be created in R using the `:` operator, the `seq()` function, and the `rep()` function. The `:` operator is used to create a sequence of numbers from a starting value to an ending value. The `seq()` function is used to create a sequence of numbers with a specified start, end, and step size. The `rep()` function is used to repeat a sequence of numbers a specified number of times.

```
v <- (10:20) # creating a sequence
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) # creating a sequence
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) # creating a sequence
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) # creating a sequence
repe # repeat from 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) # creating a sequence
eq # repeat from Hello Ann, 5 times
```

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

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

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

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

```
## [1] 1 3 5 7 9
```

```r
class(we) # checking the class of we
```

```
## [1] "numeric"
```

```r
fo <- list("Hello","Hi","Hey") # creating a list

unlist(fo)
```

```
## [1] "Hello" "Hi"    "Hey"
```

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

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

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

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

```r
fo[[2]] # for only one element
```

```
## [1] "Hi"
```

```r
class(fo[[3]]) # checking the class of the third element
```

```
## [1] "character"
```

```r
wi <- list(age = c(12,23,45), height = c(12.3,45.4, 34.5)) # creating a list

wi
```

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

```r
class(wi) # checking the class of wi
```

```
## [1] "list"
```

```r
# creating a dataframe
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
```

```r
class(woo) # checking the class of woo
```

```
## [1] "data.frame"
```

```r
woo$age # selecting only the age list
```

```
## [1] 12 23 45
```

```r
woo$height # selecting only the height list
```

```
## [1] 12.3 45.4 34.5
```

```r
woo[["age"]] # selecting only the age list
```

```
## [1] 12 23 45
```

```r
woo[['h',exact = FALSE]] # partial matching
```

```
## [1] 12.3 45.4 34.5
```

```r
class(woo$age) # checking the class of age list
```

```
## [1] "numeric"
```

```r
class(woo$height) # checking the class of height list
```

```
## [1] "numeric"
```

```r
# creating a matrix with 3 rows and 3 columns
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) # checking the class of 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) # creating a vector

print(ch)
```

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

```
i<- is.na(ch) # checking for missing numbers

i
```

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

```
ch[!i] # removing missing numbers
```

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

Vectorization

Vectorization is a technique used in R to perform operations on vectors without using loops. Vectorization allows you to perform operations on vectors more efficiently than using loops. Vectorization is a key concept in R programming and is used to write efficient and concise code.

```
ew <- rnorm(1000) # creating a vector

er <- rnorm(1000) # creating a vector

cv <- vector(mode = "numeric", length = 1000) # creating a vector

# Iteration
start <- proc.time() # starting the timer
for (i in 1:1000){
cv[i] <- ew[i] + er[i] # adding the two vectors
}
proc.time()-start # initiating the timer
```

```
##    user  system elapsed
##    0.01    0.00    0.04
```

```r
# Vectorization
start <- proc.time() # starting the timer
cv <- ew + er # adding the two vectors
proc.time()-start #  initiating the timer
```

```
##    user  system elapsed
##       0       0       0
```

Control Structures

Control structures are used to control the flow of a program. Control structures allow you to execute different code blocks based on conditions. Control structures include if-else statements, for loops, while loops, and repeat loops.

```r
x <- 20 # assigning 20 to x

if (x < 0) {
print("Negative!") # checking if x is negative
}else if (x < 10) {
print("Positive, less than 10!") # checking if x is less than 10
}else {
print("Number greater than 10!") # checking if x is greater than 10
}
```

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

```r
x <- -20
if (x < 0) {
print("Negative!") # checking if x is negative
}else if (x < 10) {
print("Positive, less than 10!") # checking if x is less than 10
}else {
print("Number greater than 10!") # checking if x is greater than 10
}
```

```
## [1] "Negative!"
```

```r
x <- 6 # assigning 6 to x
if (x < 0) {
print("Negative!") # checking if x is negative
}else if (x < 10) {
print("Positive, less than 10!") # checking if x is less than 10
}else {
print("Number greater than 10!") # checking if x is greater than 10
}
```

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

For Loop

`For loop` is used to execute a block of code for a specified number of times. `For loops` are used when you know the number of iterations in advance. `For loops` are used to iterate over a sequence of values.

```
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 3
```

```
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

`While loops` are used to execute a block of code as long as a condition is true. `While loops` are used when you do not know the number of iterations in advance.

```
x <- -1 # assigning -1 to x
while (x < 5){
print(x) #
x <- x + 1
}
```

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

```r
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
```

```r
for (i in 1:100){
# Over ride the first 80 iterations
if (i <= 80){
next
}
print(i)
}
```

```
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
```

Functions

Functions

```r
myPrinter <- function(x){
  for (i in seq(x)){
       print("Hello, Charles")
  }
}
myPrinter(3)
```

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

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

```
## [1] 165
```

```r
volume()
```

```
## [1] 27
```

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

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

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

Iterated Functions

Iterated function systems (IFS) are a method of constructing fractals; the resulting fractals are often self-similar. IFS fractals are more related to set theory than fractal geometry. They were introduced in 1988 by Michael Barnsley using the concept of affine transformations.

lapply

`lapply` is used to apply a function to each element of a list. lapply returns a list of the same length as the input list, where each element is the result of applying the function to the corresponding element of the input list.

```r
str(lapply) # checking the structure of lapply
```

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

```r
x <- list(a = rnorm(10), b = rnorm(20), c = rnorm(30))

lapply(x, mean) # checking the mean of x
```

```
## $a
## [1] -0.4574003
##
## $b
## [1] -0.1116678
##
## $c
## [1] -0.01046405
```

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

```
## $a
## [1] 0.9401963
##
## $b
## [1] 0.6072367
##
## $c
## [1] 0.7275282
```

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

```
## $a
## [1] 0.9696372
##
## $b
## [1] 0.7792539
##
## $c
## [1] 0.8529526
```

sapply

`sapply` is used to apply a function to each element of a list and simplify the result. sapply returns a vector or matrix instead of a list.

```r
str(sapply) # checking the structure of sapply
```

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

```r
xi <- list(a = rnorm(10), b = rnorm(20), c = rnorm(30))

sapply(xi, mean) # checking the mean of xi
```

```
##           a           b           c
## -0.08244251 -0.29385749  0.08847233
```

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

```
##        a        b        c
## 0.375878 1.131216 0.850656
```

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

```
##         a         b         c
## 0.6130889 1.0635864 0.9223101
```

Split

split is used to split a data frame or vector into groups. split returns a list of vectors, where each vector contains the elements of the input data frame or vector that belong to a particular group.

```r
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) # splitting based on the adult column

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
```

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

```
##    FALSE     TRUE
## 16.00000 21.33333
```

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

```
##    FALSE     TRUE
## 1.000000 2.333333
```

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

```
##    FALSE     TRUE
## 1.000000 1.527525
```

tapply

`tapply` is used to apply a function to each group of a vector. tapply returns a vector or matrix, where each element is the result of applying the function to the corresponding group of the input vector.

```r
str(tapply)
```

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

```r
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
```

```r
tapply(x, f, mean)
```

```
##            1            2            3            4
## -0.081708982 -0.006995191  0.523500221 -0.141153999
```

```r
tapply(x, f, var)
```

```
##         1         2         3         4
## 0.4216030 0.4889995 0.8129353 1.3410572
```

```r
tapply(x, f, sd)
```

```
##         1         2         3         4
## 0.6493096 0.6992850 0.9016292 1.1580402
```

```r
### Help
#?c
#?vector
#?sapply
#?lapply
#?tapply
```

Types, Quality and Data preprocessing

`Types`, `quality` and `data preprocessing` are important steps in data analysis. `Types` refer to the data types of the variables in the dataset. `Quality` refers to the quality of the data, such as missing values and outliers. `Data preprocessing` refers to the steps taken to clean and prepare the data for analysis.

```
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

`Regularization` is a technique used to prevent overfitting in machine learning models. Regularization adds a penalty term to the loss function to prevent the model from fitting the training data too closely. Regularization is used to improve the generalization of the model.r

```
# 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

`Dplyr` is a package for data manipulation in R. `Dplyr` provides a set of functions for manipulating data frames, such as selecting columns, filtering rows, and summarizing data. `Tidyr` is a package for data tidying in R. `Tidyr` provides functions for reshaping data frames, such as gathering columns into rows and spreading rows into columns.

```r
library(dplyr) # Importing the dplyr package

data(airquality) # loading the airquality dataset

class(airquality) # checking the class of airquality
```

```
## [1] "data.frame"
```

```r
airquality <- as_tibble(airquality) # converting airquality to a tibble

class(airquality) # checking the class of airquality
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
airquality # displaying the airquality dataset
```

```
## # A tibble: 153 x 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
```

```r
select(airquality, (Ozone), Solar.R, Day) # selecting the Ozone, Solar.R and Day columns
```

```
## # A tibble: 153 x 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
```

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

```
## # A tibble: 153 x 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
```

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

```
## # A tibble: 10 x 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
```

```r
filter(airquality, Day == 1 | Day == 2) # filter values of Day = 1 or 2
```

```
## # A tibble: 10 x 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
```

```r
arrange(airquality,Ozone) # arrange based on Ozone
```

```
## # A tibble: 153 x 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      NA   6.9    74     5    11
##  5     7      48  14.3    80     7    15
##  6     7      49  10.3    69     9    24
##  7     8      19  20.1    61     5     9
##  8     9      24  13.8    81     8     2
##  9     9      36  14.3    72     8    22
## 10     9      24  10.9    71     9    14
## # i 143 more rows
```

```r
arrange(airquality, Solar.R) # arrange based on Solar.R
```

```
## # A tibble: 153 x 6
##    Ozone Solar.R  Wind  Temp Month   Day
##    <int>   <int> <dbl> <int> <int> <int>
##  1    16       7   6.9    74     7    21
##  2     1       8   9.7    59     5    21
##  3    23      13  12      67     5    28
##  4    23      14   9.2    71     9    22
##  5     8      19  20.1    61     5     9
##  6    14      20  16.6    63     9    25
##  7     9      24  13.8    81     8     2
##  8     9      24  10.9    71     9    14
##  9     4      25   9.7    61     5    23
## 10    13      27  10.3    76     9    18
## # i 143 more rows
```

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

```
## # A tibble: 153 x 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
```

```r
# 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 x 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
```

```r
### print(airquality, n=143)
```

```r
# grouping by month
by_month <- group_by (airquality, Month)
by_month
```

```
## # A tibble: 111 x 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
```

```r
# Finding the minimum, mean and maximum value per Month
summarize (by_month, min(Ozone), mean(Ozone), max(Ozone))
```

```
## # A tibble: 5 x 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
```

```r
# Finding the minimum, mean and maximum value per Month
summarize (by_month, min(Day), mean(Day), max(Day))
```

```
## # A tibble: 5 x 4
##   Month `min(Day)` `mean(Day)` `max(Day)`
##   <int>      <int>       <dbl>      <int>
## 1     5          1        16.1         31
## 2     6          7        14.3         20
## 3     7          1        16.2         31
## 4     8          1        17.2         31
## 5     9          1        15.1         30
```

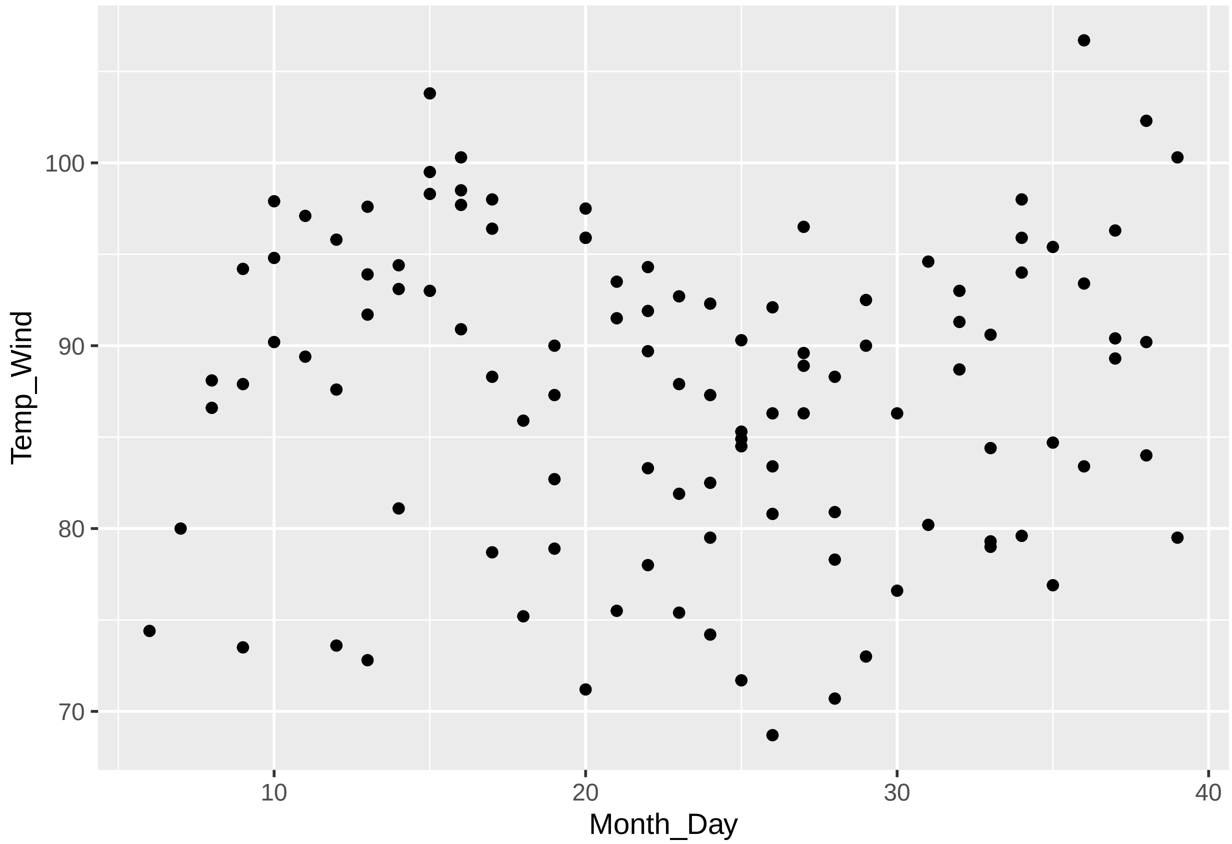
```r
# Finding the minimum, mean and maximum value per Month
summarize (by_month, min(Wind), mean(Wind), max(Wind))
```

```
## # A tibble: 5 x 4
##   Month `min(Wind)` `mean(Wind)` `max(Wind)`
##   <int>       <dbl>        <dbl>       <dbl>
## 1     5         5.7        11.5         20.1
## 2     6         8          12.2         20.7
## 3     7         4.1         8.52        14.9
## 4     8         2.3         8.86        15.5
## 5     9         2.8        10.1         16.6
```

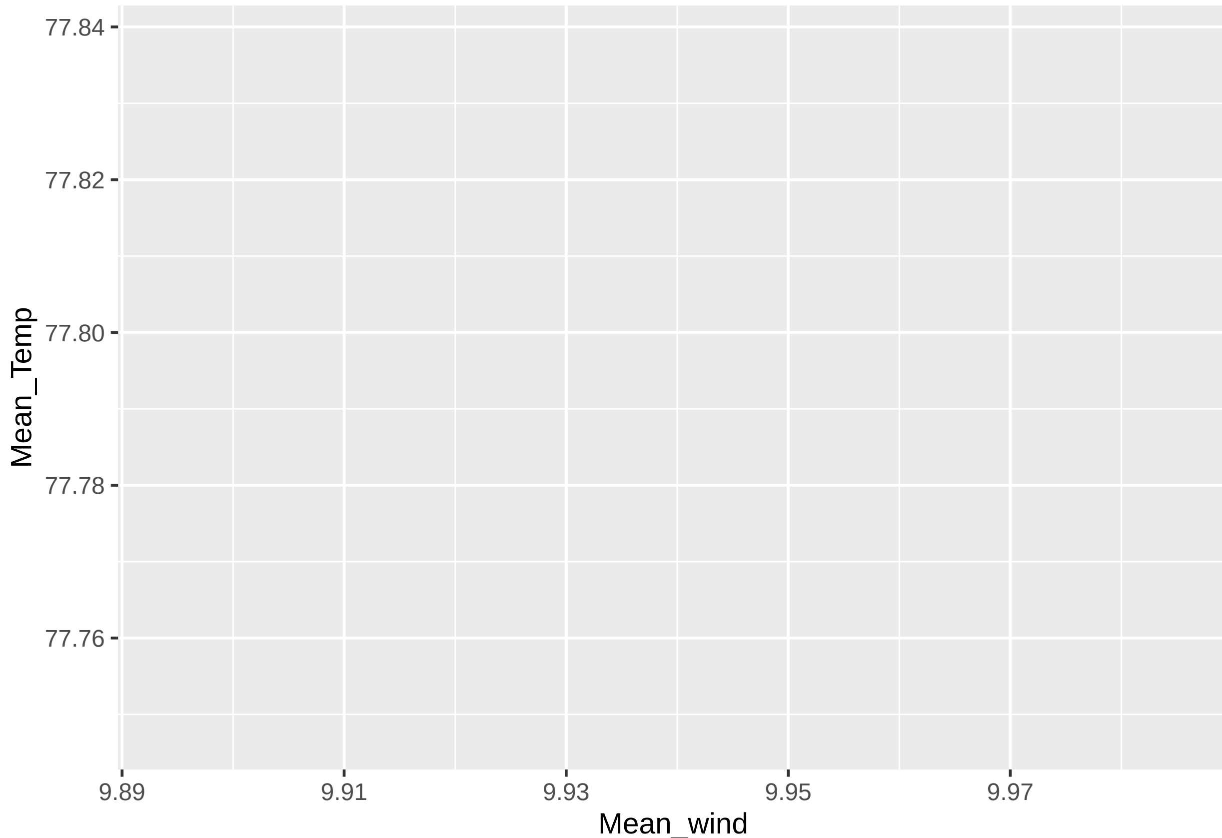
```r
library (tidyr)
library(ggplot2)

airquality %>% mutate(Month_Day = Month + Day, Temp_Wind = Temp + Wind)
```

```
## # A tibble: 111 x 8
##    Ozone Solar.R  Wind  Temp Month   Day Month_Day Temp_Wind
##    <int>   <int> <dbl> <int> <int> <int>     <int>     <dbl>
##  1    41     190   7.4    67     5     1         6      74.4
##  2    36     118   8      72     5     2         7      80
##  3    12     149  12.6    74     5     3         8      86.6
##  4    18     313  11.5    62     5     4         9      73.5
##  5    23     299   8.6    65     5     7        12      73.6
##  6    19      99  13.8    59     5     8        13      72.8
##  7     8      19  20.1    61     5     9        14      81.1
##  8    16     256   9.7    69     5    12        17      78.7
##  9    11     290   9.2    66     5    13        18      75.2
## 10    14     274  10.9    68     5    14        19      78.9
## # i 101 more rows
```

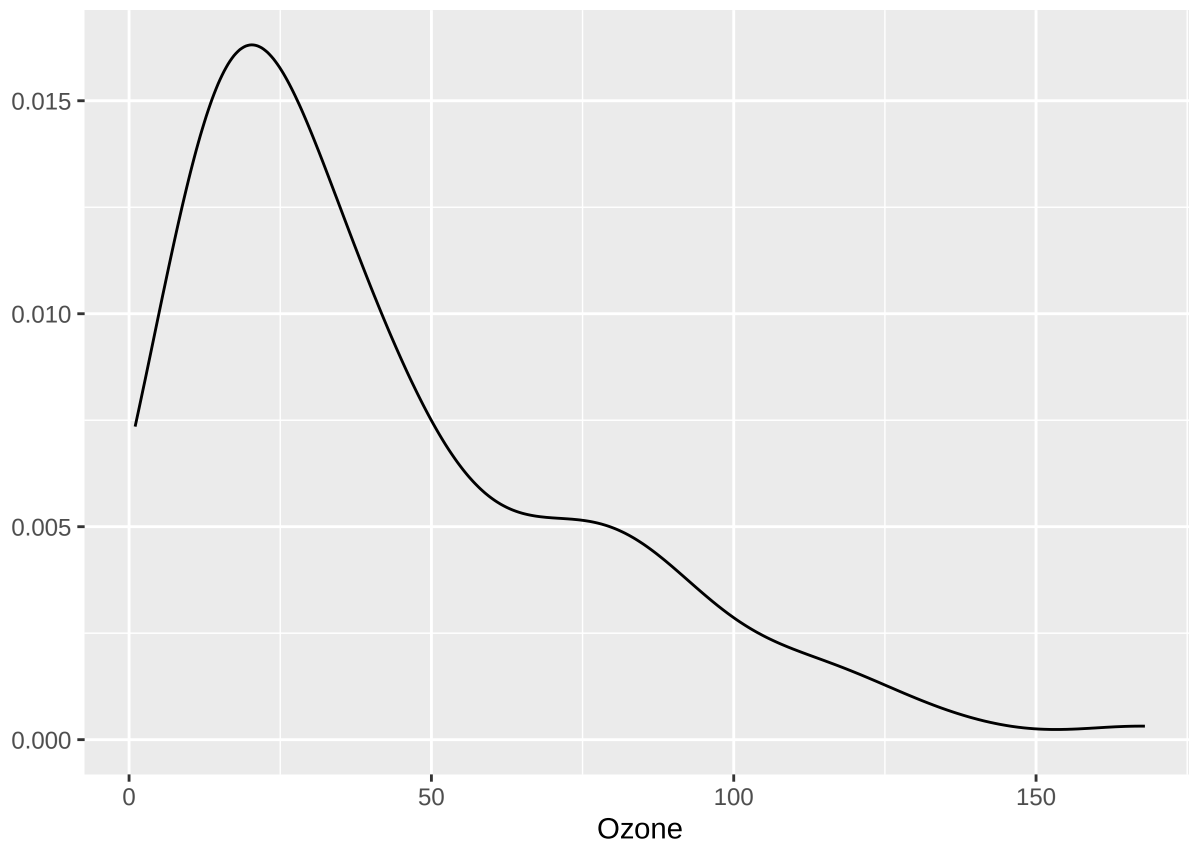
```r
airquality %>% transmute(Month_Day = Month + Day, Temp_Wind = Temp + Wind) %>% ggplot() + geom_point(ae
```



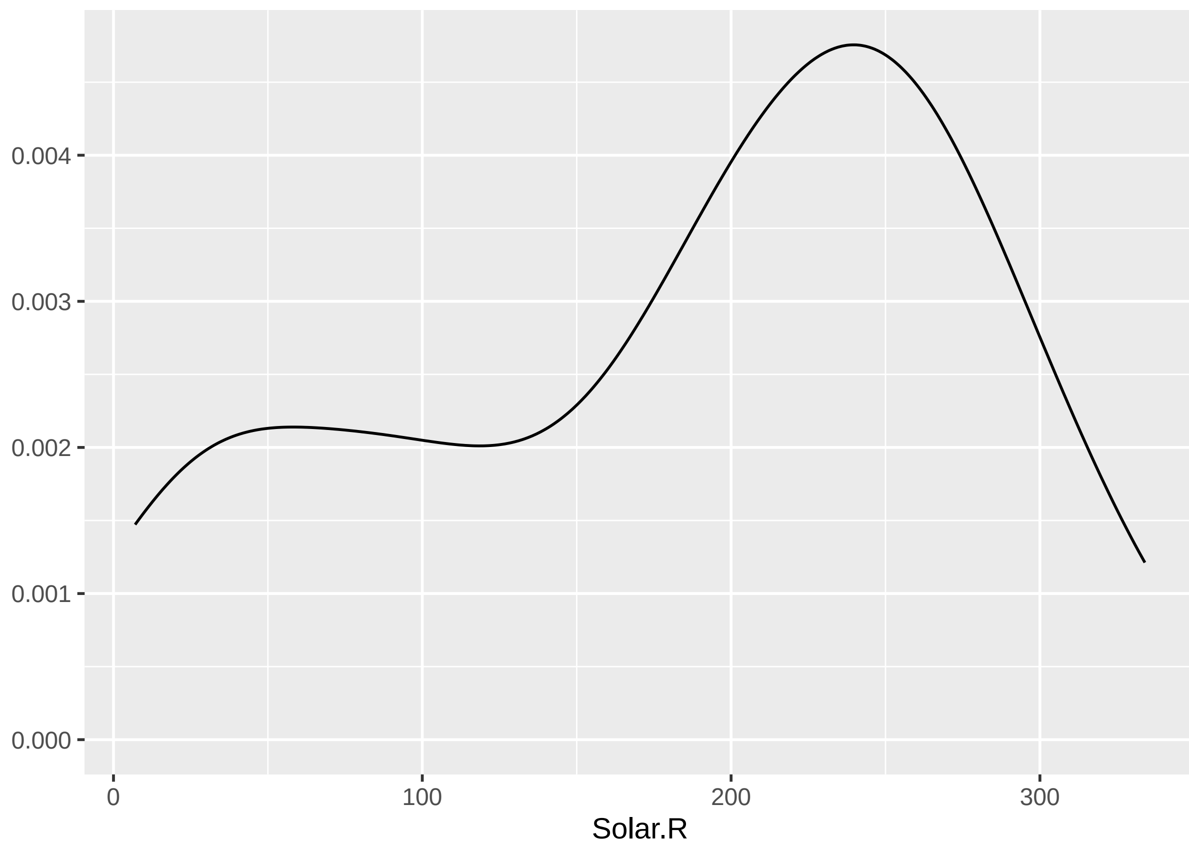

```
airquality %>% transmute(Mean_wind = mean(Wind), Mean_Temp = mean(Temp)) %>% ggplot(aes(x = Mean_wind, y
```



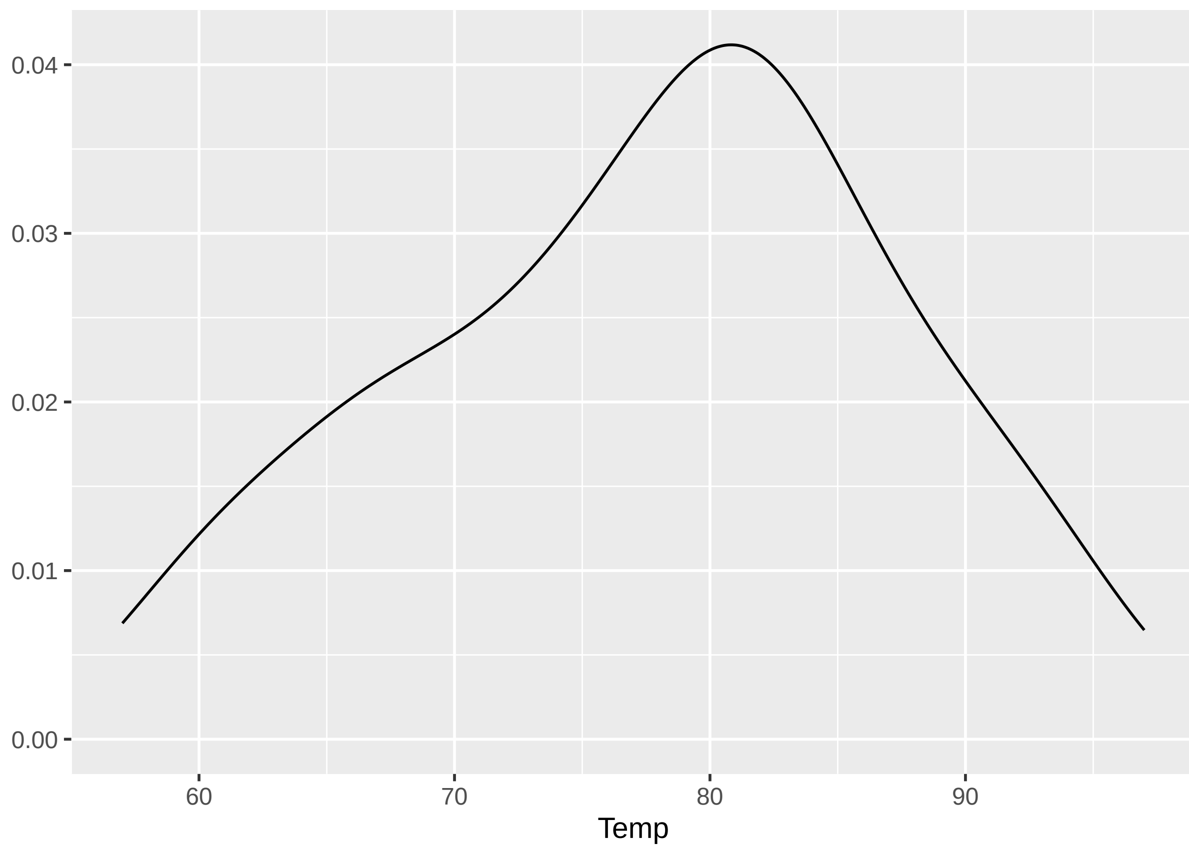

```
airquality %>% qplot (Ozone, data = ., geom = "density")
```



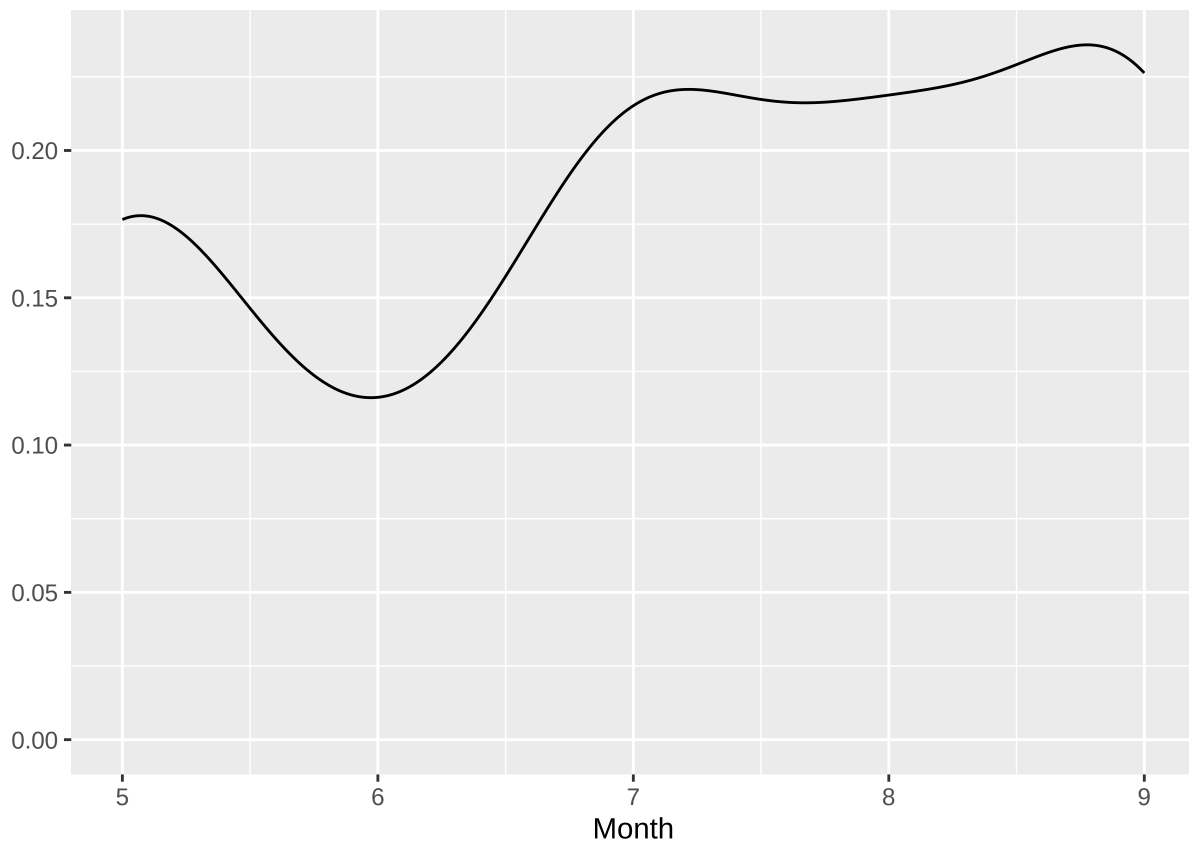

```
airquality %>% qplot (Solar.R, data = ., geom = "density")
```



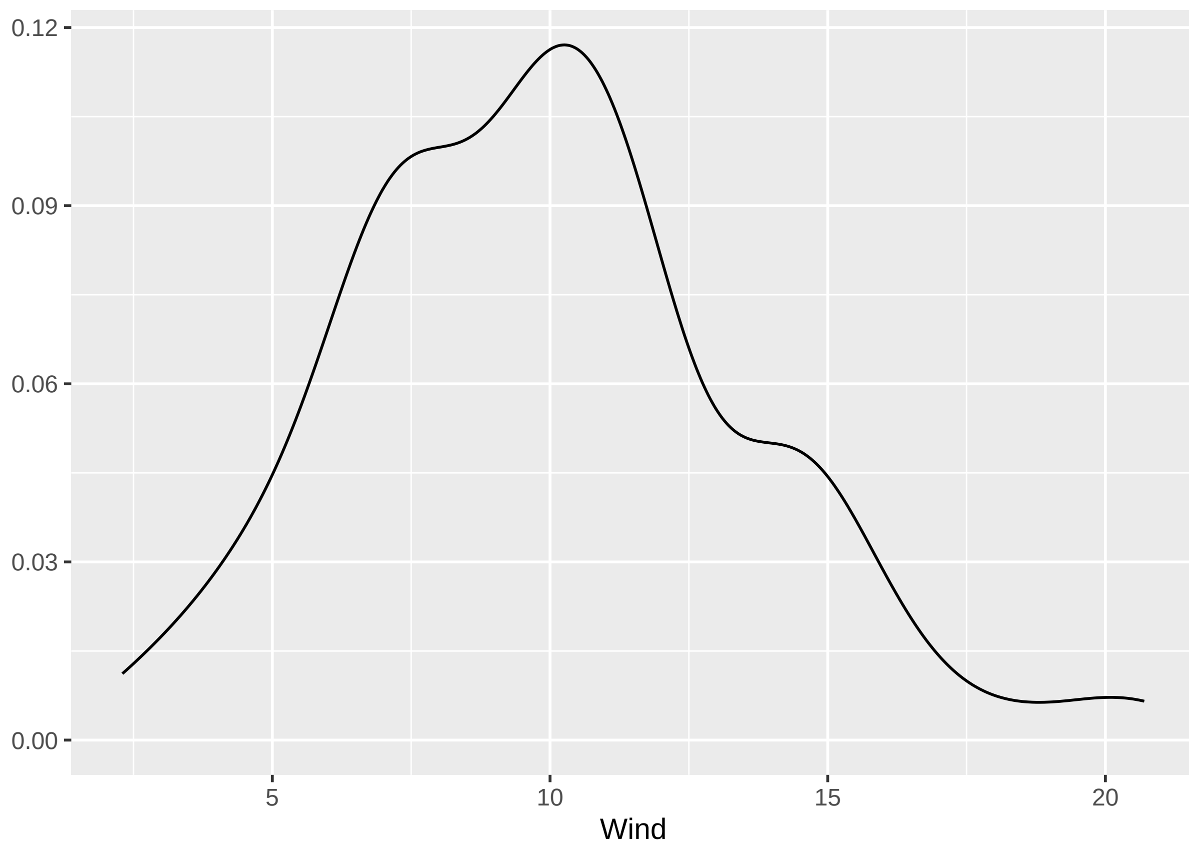

```
airquality %>% qplot (Temp, data = ., geom = "density")
```



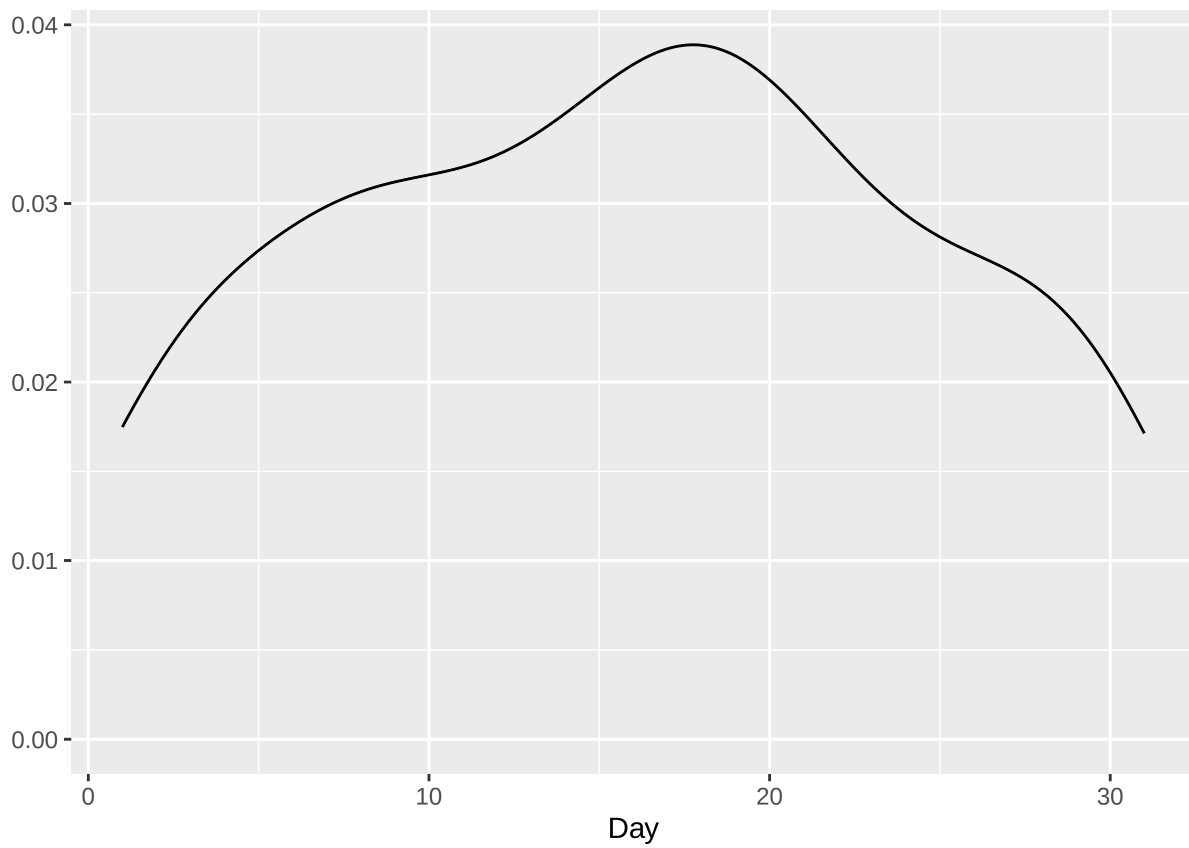

```r
airquality %>% qplot (Month, data = ., geom = "density")
```




```
airquality %>% qplot (Wind, data = ., geom = "density")
```




```r
airquality %>% qplot (Day, data = ., geom = "density")
```



Statistical Summary and Visualization

`Statistical summary` and `visualization` are important steps in data analysis. Statistical summary provides a summary of the data, such as the mean, median, and standard deviation. Visualization provides a visual representation of the data, such as bar charts and histograms.

```
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(net_usage, na.rm = TRUE)
```

```
## [1] 8.5
```

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

min(A)
```

```
## [1] 33
```

```r
max(A)
```

```
## [1] 71
```

```r
which.min(A)
```

```
## [1] 2
```

```r
which.max(A)
```

```
## [1] 9
```

```r
print(max(A) - min(A))
```

```
## [1] 38
```

```r
range(A)
```

```
## [1] 33 71
```

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

```
## [1] 38
```

```r
summary(A)
```

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

Percentile Values

`Percentile` values are used to divide a dataset into equal parts. The `quantile()` function is used to calculate percentile values in R. The `quantile()` function takes a vector as input and returns the percentile value for the specified quantile.

```r
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
```

```r
quantile(X,0.50,type = 7)
```

```
## 50%
##  13
```

```r
quantile(X,0.25,type = 7)
```

```
##  25%
## 7.75
```

```r
quantile(X,0.75,type = 7)
```

```
##   75%
## 16.25
```

```r
median(X)
```

```
## [1] 13
```

```r
summary(X)
```

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

```r
sd(X)
```

```
## [1] 7.843703
```

```r
# cv(X)
```

Interquartile Range

The interquartile range (IQR) is a measure of statistical dispersion, or how spread out the values in a dataset are. The IQR is calculated as the difference between the 75th percentile (Q3) and the 25th percentile (Q1) of the dataset. The IQR is used to identify outliers in a dataset.

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

```
## 75%
## 8.5
```

Variance and Standard Deviation

`Variance` and `standard deviation` are measures of statistical dispersion. Variance is the average of the squared differences between each value in the dataset and the mean of the dataset. Standard deviation is the square root of the variance. Variance and standard deviation are used to measure the spread of the values in a dataset.

```r
# 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

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
```

```r
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
```

```r
var(course)
```

```
## [1] 15.41842
```

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

```
## [1] 15.41842
```

```r
sd(course)
```

```
## [1] 3.92663
```

```r
sqrt(var(course))
```

```
## [1] 3.92663
```

```r
std = function(x) sqrt(var(x))
```

```r
std(course)
```

```
## [1] 3.92663
```

```r
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
```

```r
sum(course)
```

```
## [1] 91
```

```r
prod(course)
```

```
## [1] 41878425600
```

```r
sort(course)
```

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

```r
order(course)
```

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

```r
sqrt(-14 + 9i)
```

```
## [1] 1.149634+3.914289i
```

Coefficient of Variation

`Coefficient of variation (CV)` is a measure of relative variability. The CV is calculated as the standard deviation divided by the mean of the dataset. The CV is used to compare the variability of datasets with different units of measurement.

```r
cv = function(x) ( sd(x) / mean(x) )
cv(course)
```

```
## [1] 0.8629955
```

Visualization of Qualitative Data

`Visualization of qualitative data` is an important step in data analysis. Qualitative data is data that is categorical or non-numeric. Visualization of qualitative data can help identify patterns and trends in the data. Bar charts, pie charts, and contingency matrices are commonly used to visualize qualitative data.

```r
mo = c("car","car","bus", "metro","metro","car","metro","metro","foot","car","foot","bus","bus","metro"
mo
```

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

```r
table(mo) # creating a table of mo
```

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

```r
prop.table(table(mo)) # creating a table of mo
```

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

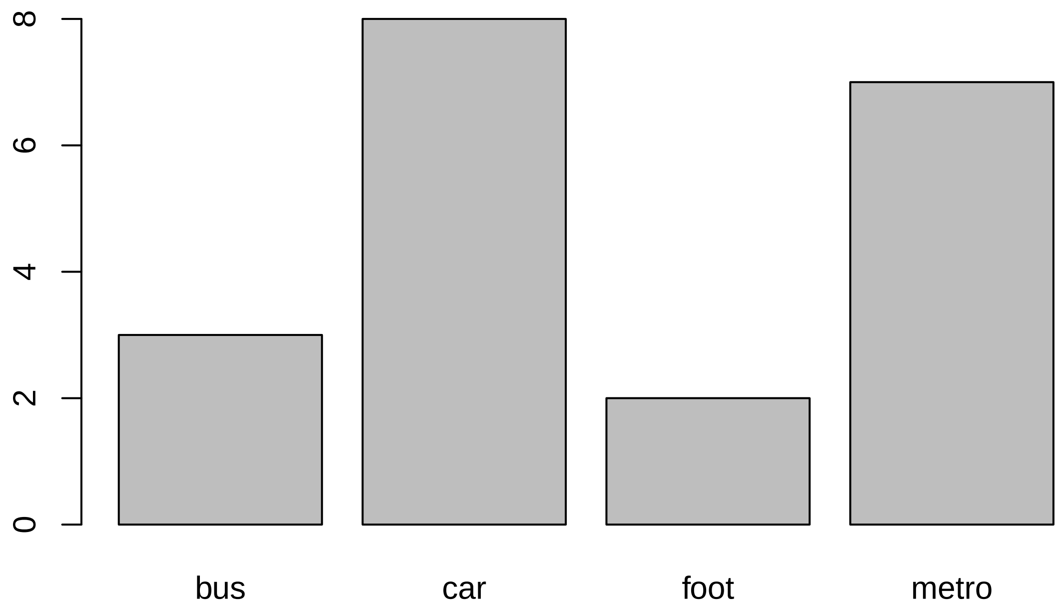
```r
data.frame(mo)# creating a dataframe of mo
```

```
##       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
```

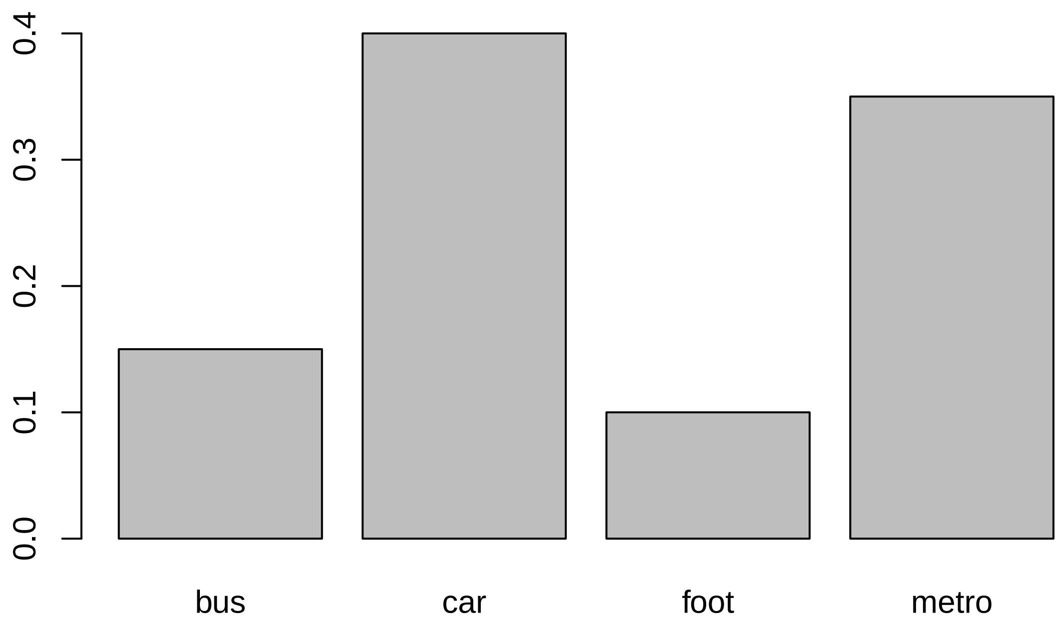
Bar Charts

`Bar charts` are used to visualize the frequency of categorical data. Bar charts are used to compare the frequency of different categories in a dataset. Bar charts are used to visualize the distribution of categorical data.

```r
barplot(table(mo)) # creating a bar chart of mo
```



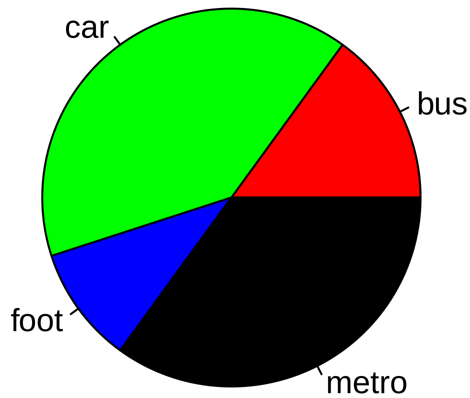

```r
barplot(prop.table(table(mo))) # creating a bar chart of mo
```



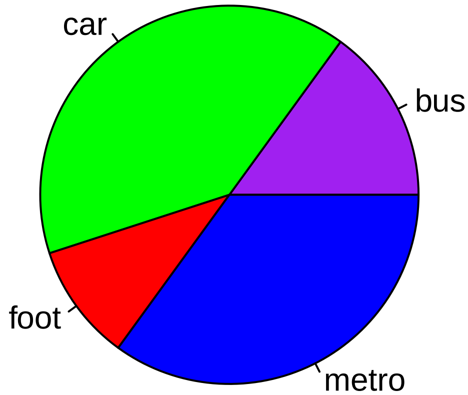
Pie Chart

`Pie charts` are used to visualize the proportion of different categories in a dataset. Pie charts are used to compare the proportion of different categories in a dataset. Pie charts are used to visualize the distribution of categorical data.

```r
pie(table(mo), col = c ("red", "green", "blue", "black")) # creating a pie chart of mo
```




```
pie(prop.table(table(mo)), col = c("purple", "green", "red", "blue")) # creating a pie chart of mo
```



Contingency Matrix

`A contingency matrix` is used to visualize the relationship between two categorical variables. A contingency matrix is a two-dimensional table that shows the frequency of each combination of categories in the two variables. A contingency matrix is used to visualize the relationship between two categorical variables.

```
g = c(rep("Male",8), rep("Female",12)) # creating a vector
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) # creating a contingency matrix of mo and g
mg
```

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

```r
gm = data.frame(mo,g) # creating a dataframe of mo and 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
```

```r
margin.table(mg, 1) # creating a margin table of mg
```

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

```r
margin.table(mg,2) # creating a margin table of mg
```

```
## g
## Female   Male
##     12      8
```

```r
prop.table(mg)     # creating a table of 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
```

```r
prop.table(mg,1)   # creating a table of mg
```

```
##        g
## mo         Female      Male
```

```
##   bus   0.6666667 0.3333333
##   car   0.6250000 0.3750000
##   foot  1.0000000 0.0000000
##   metro 0.4285714 0.5714286
```

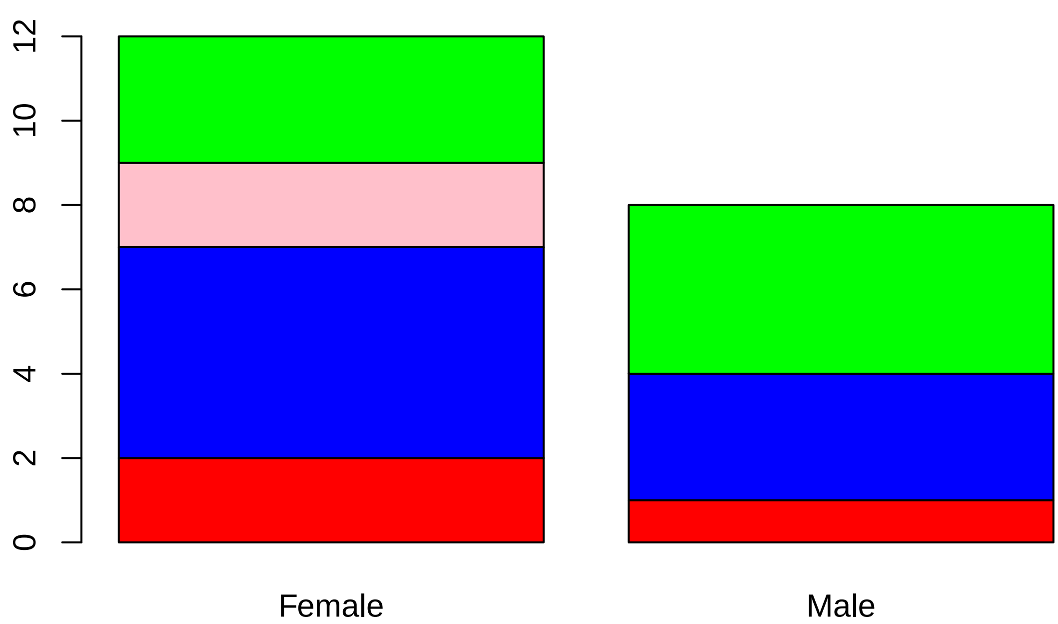
```
prop.table(mg,2)    # creating a table of mg
```

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

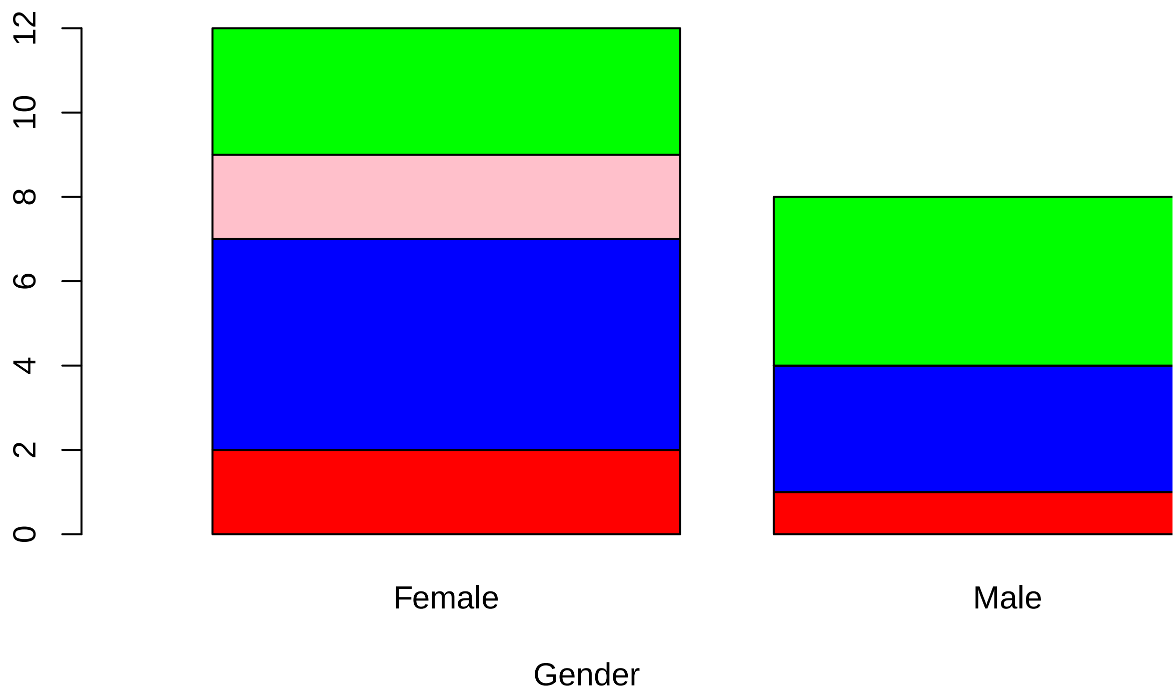
Stacked Bar Charts and Grouped Bar Charts

`Stacked bar charts` are used to visualize the relationship between two categorical variables. Stacked bar charts are used to compare the frequency of different categories in one variable for each category in another variable. Stacked bar charts are used to visualize the relationship between two categorical variables.

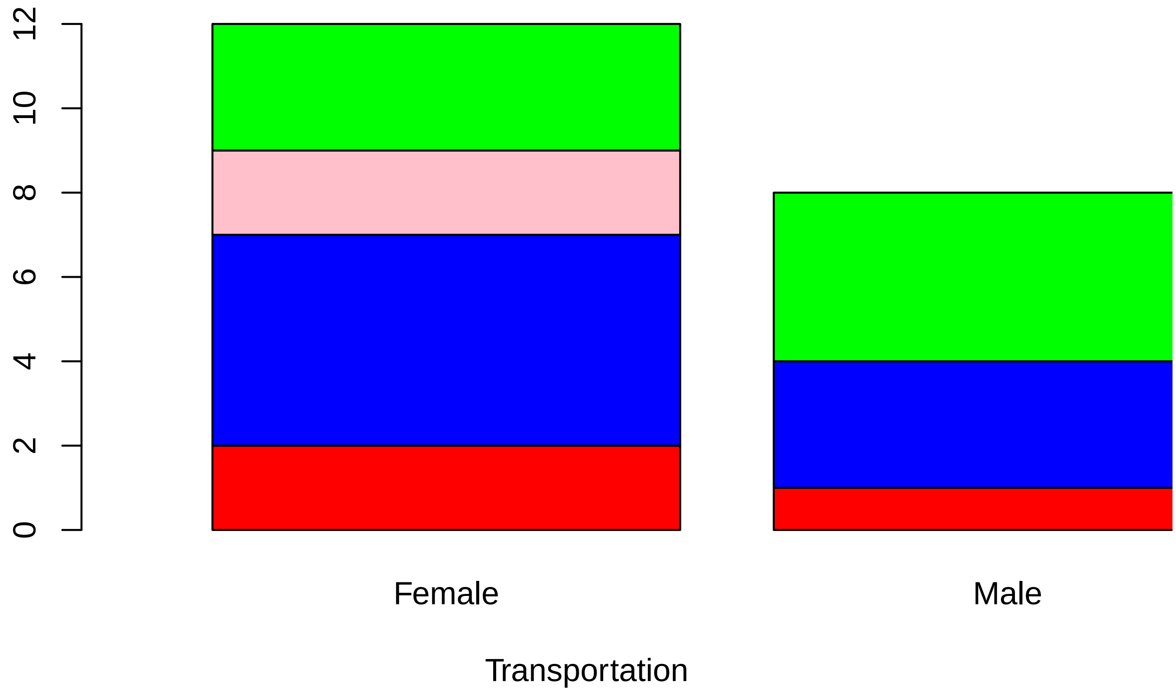
```
barplot(mg, col = c("red","blue", "pink","green")) # creating a bar chart of mg
```



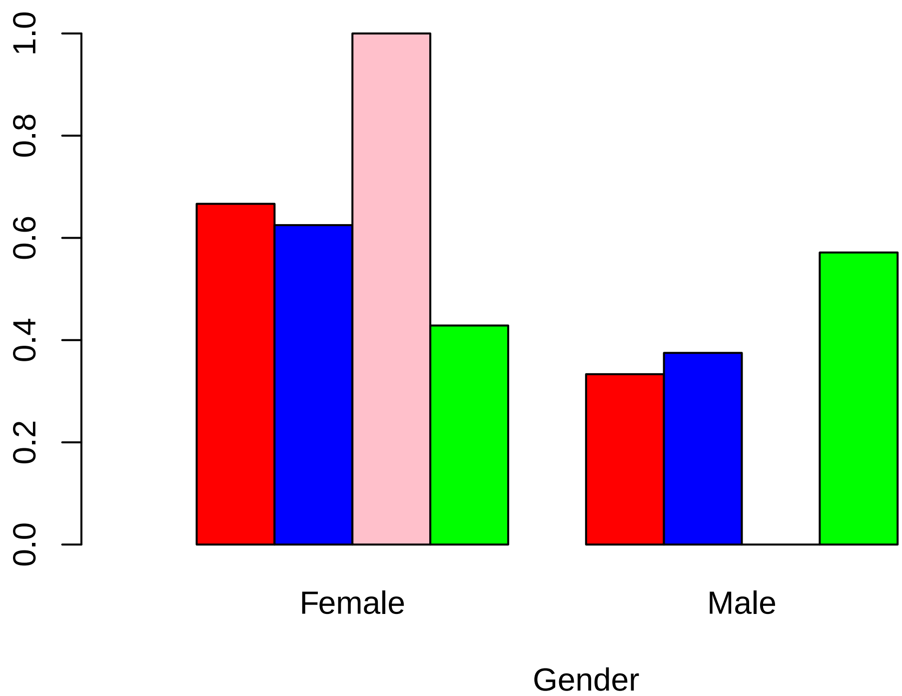
```r
barplot(mg,xlim=c(0,2), xlab="Gender", length=levels(mo), col = c("red","blue", "pink","green")) # crea
```



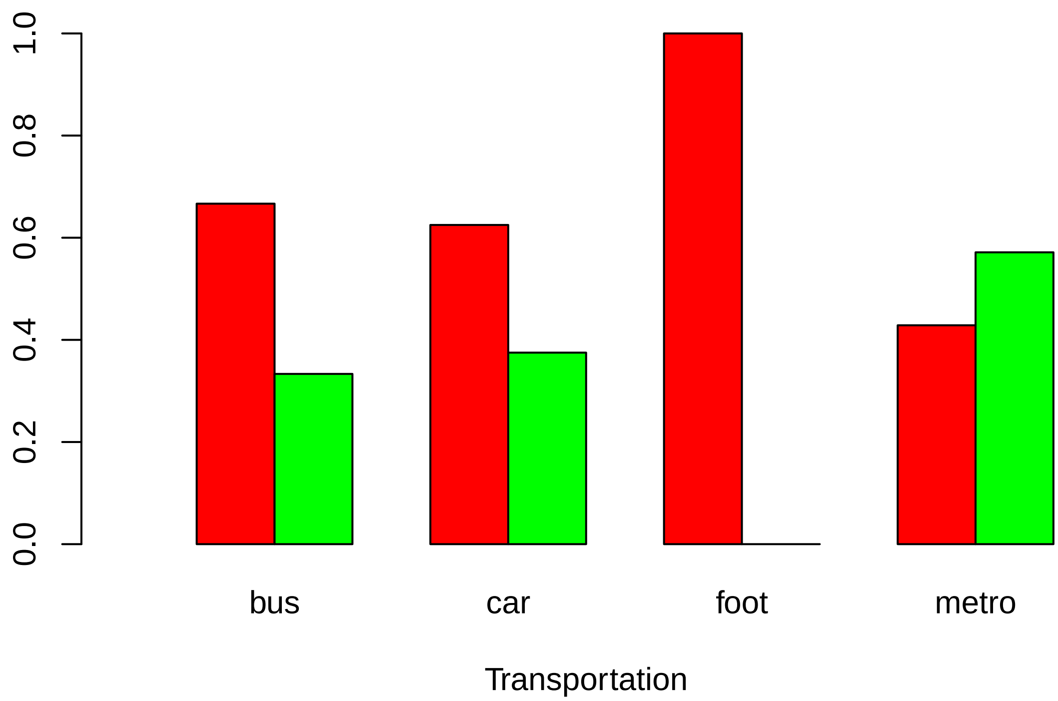

```r
barplot(mg,xlim=c(0,2), xlab="Transportation", length=levels(g), col = c("red","blue", "pink","green"))
```




```
barplot(prop.table(mg,1), width=0.25, xlim=c(0,3), ylim=c(0,1), xlab="Gender", legend=levels(mo), beside
```




```r
mg = table(g,mo) # creating a contingency matrix of g and mo
barplot(prop.table(mg,2), width=0.25, xlim=c(0,3), ylim=c(0,1), xlab="Transportation", legend=levels(g)
```



Histograms

`Histograms` are used to visualize the distribution of numerical data. Histograms are used to show the frequency of different values in a dataset. Histograms are used to visualize the distribution of numerical data.

```
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) # creating a vect

table(xo) # creating a table of xo
```

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

```
data.frame(xo) # creating a dataframe of xo
```

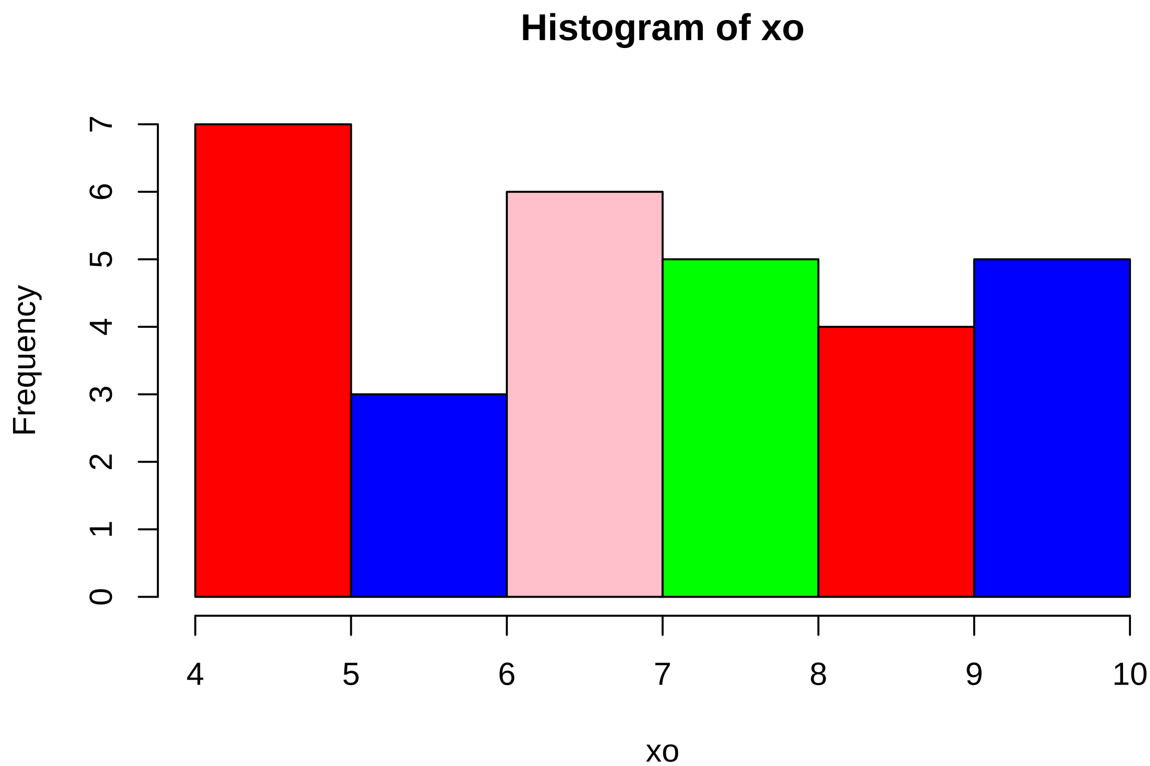
```
##    xo
## 1  10
## 2  10
## 3   5
## 4   9
## 5   7
## 6   6
## 7   8
```

```
## 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
```

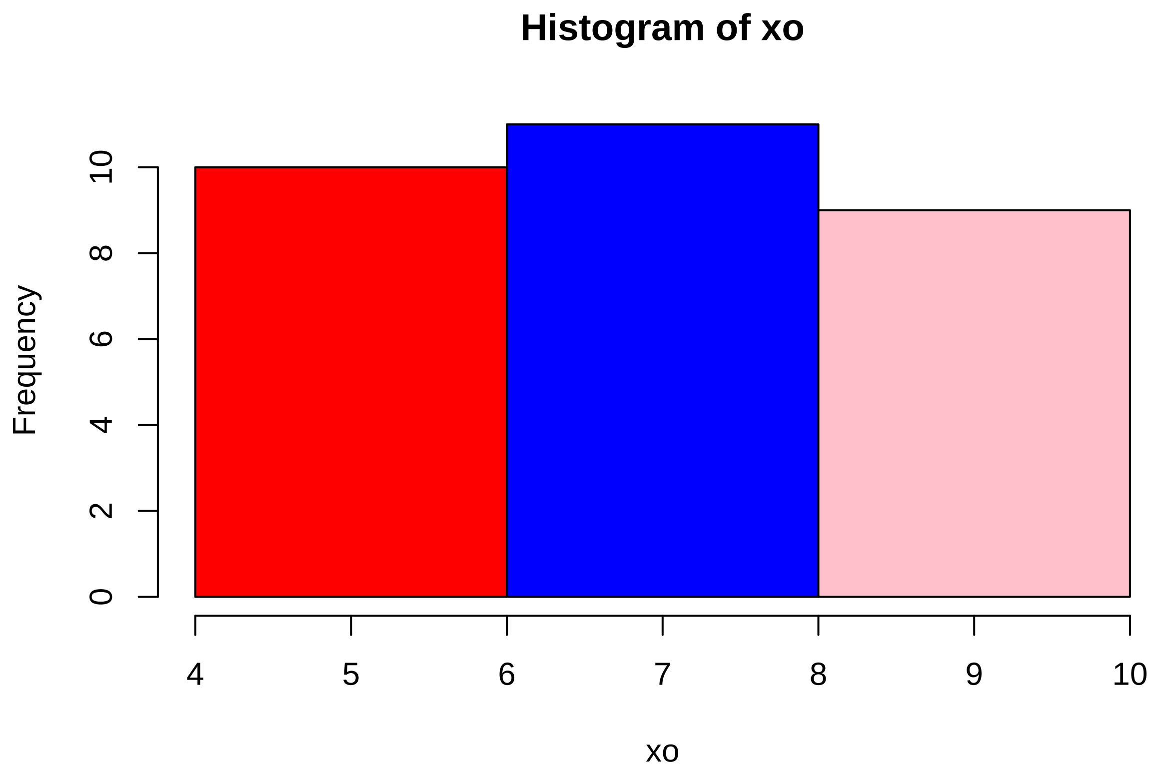
```r
prop.table(table(xo)) # creating a table of xo
```

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

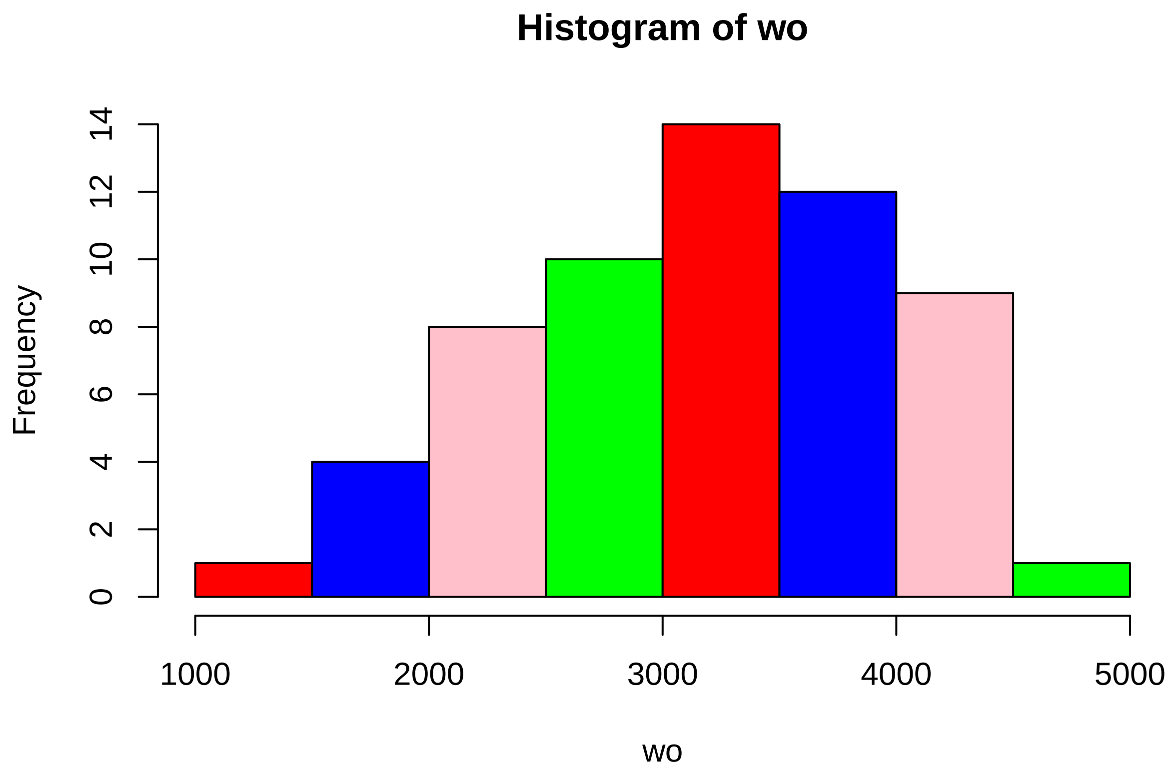
```r
#?prop.table


hist(xo, col = c("red","blue", "pink","green")) # creating a histogram of xo
```



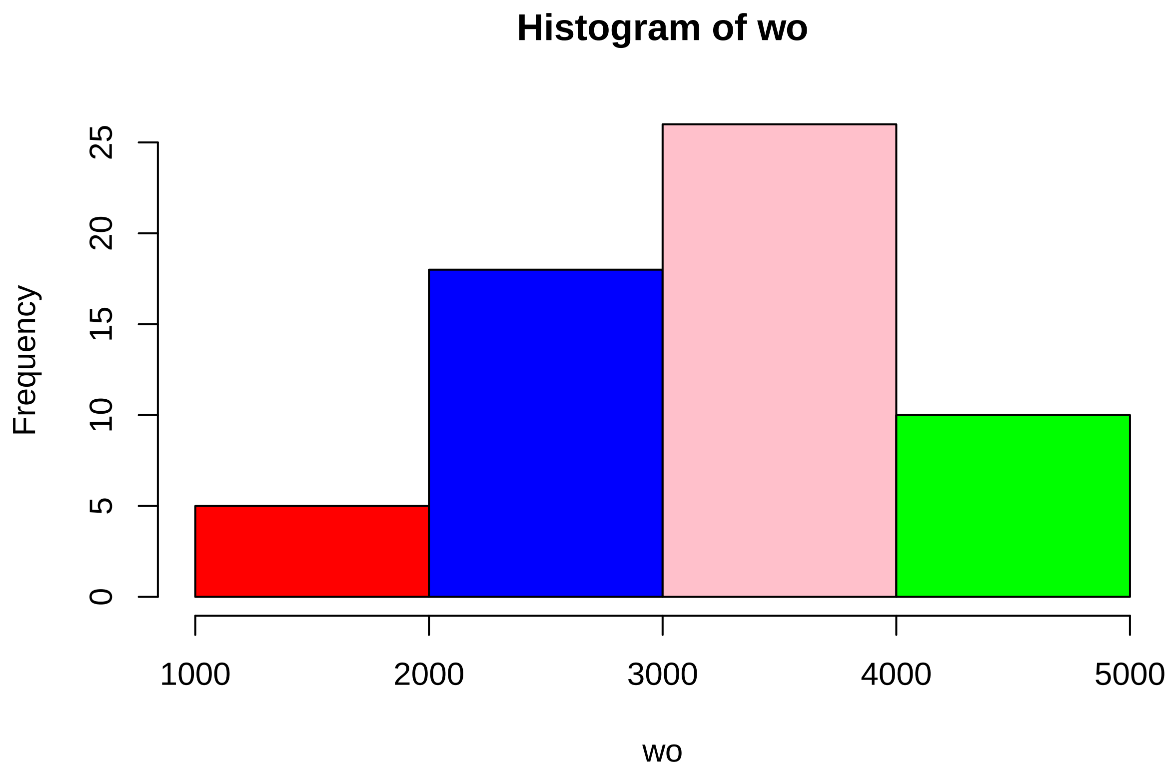

```r
hist(xo, nclass = 3, col = c("red","blue", "pink","green")) # creating a histogram of xo
```



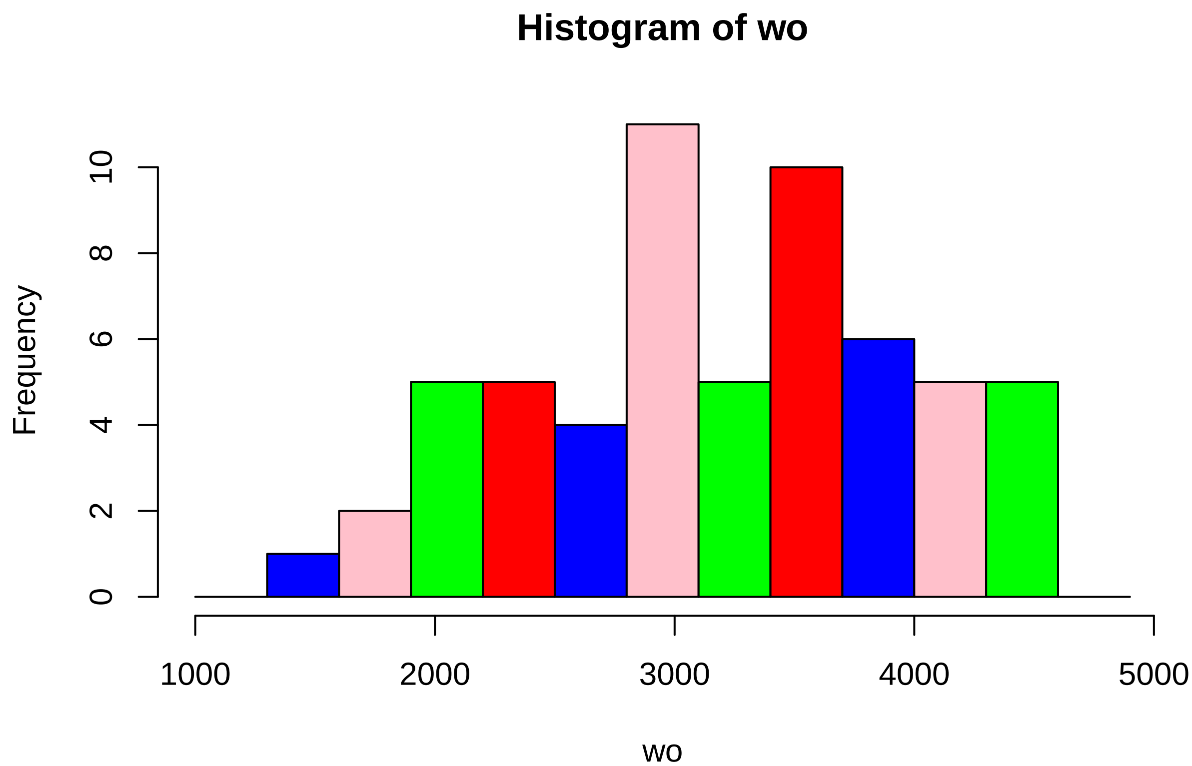

```
wo = c (1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550, 3050, 2350, 1850, 2820, 3670, 2950,

hist(wo, col = c("red","blue", "pink","green")) # creating a histogram of wo
```



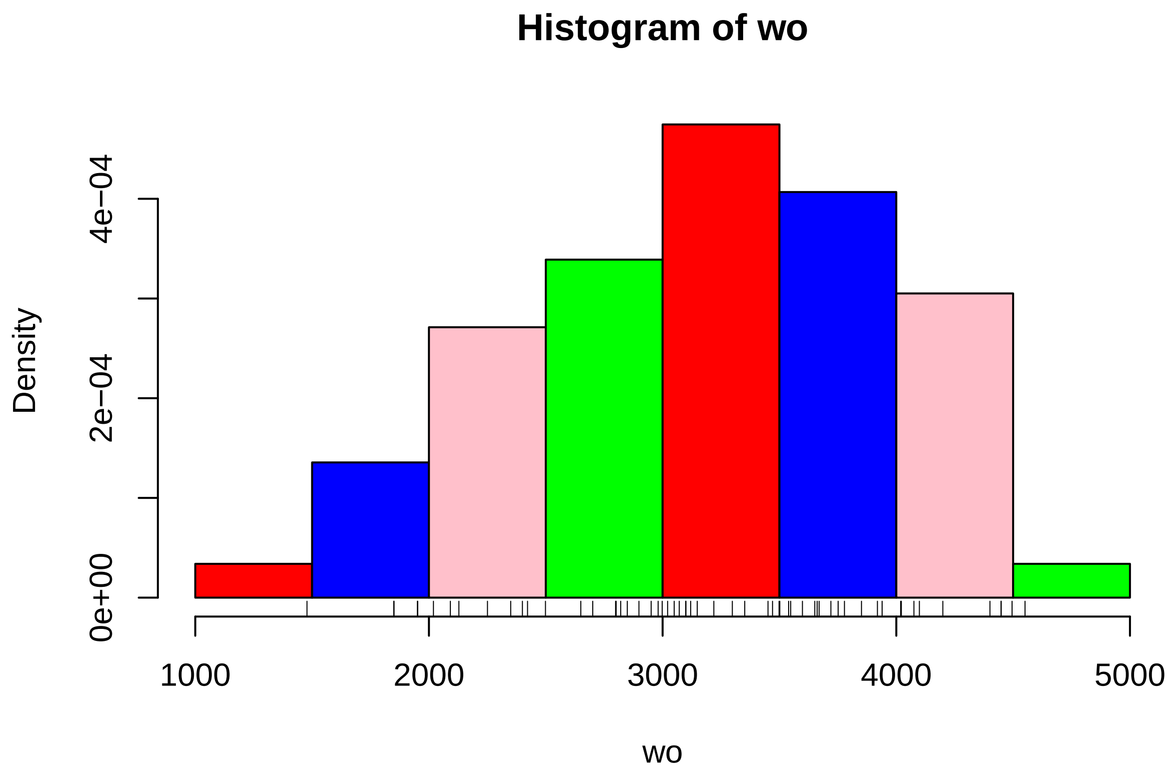

```
hist(wo, nclass=4, col = c("red","blue", "pink","green")) # creating a histogram of wo
```




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




```r
hist(wo, probability=T, col = c("red","blue", "pink","green")) # creating a histogram of wo
rug(jitter(wo)) # creating a histogram of wo
```



Frequency Polygon

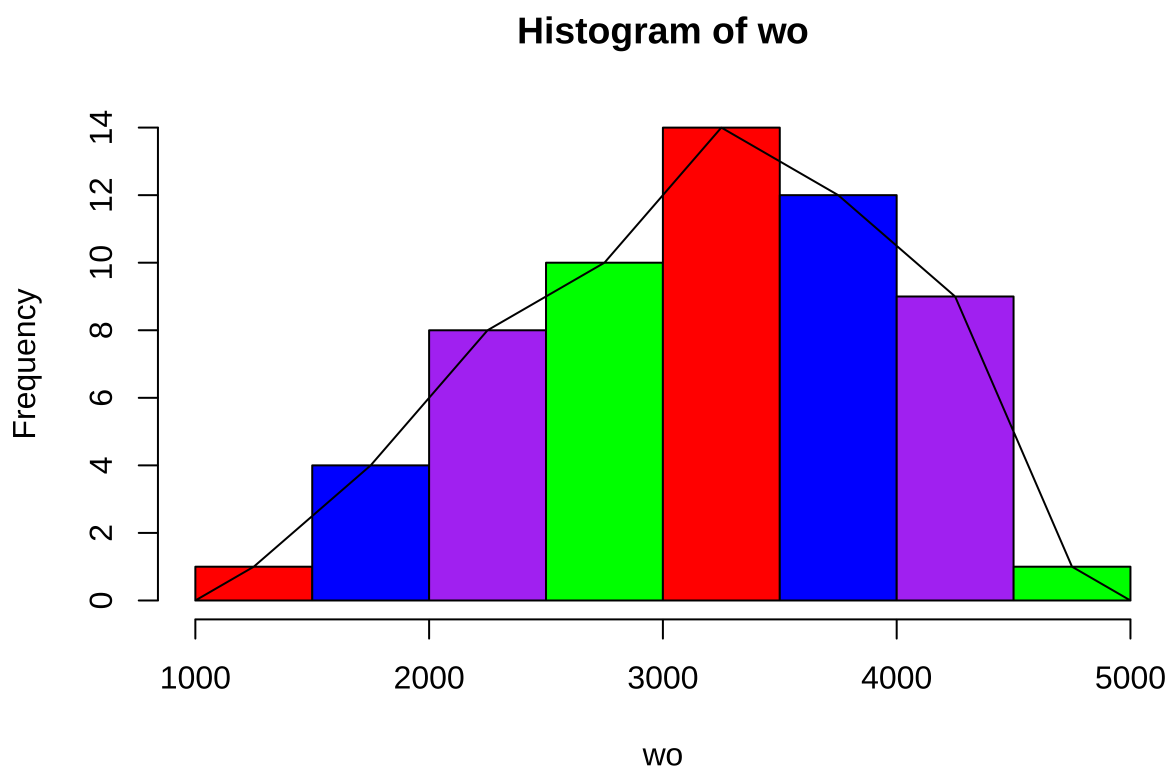
`Frequency polygons` are used to visualize the distribution of numerical data. Frequency polygons are used to show the frequency of different values in a dataset. Frequency polygons are used to visualize the distribution of numerical data.

```
temp = hist(wo, col = c("red","blue", "purple","green")) # creating a histogram of wo
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"
```

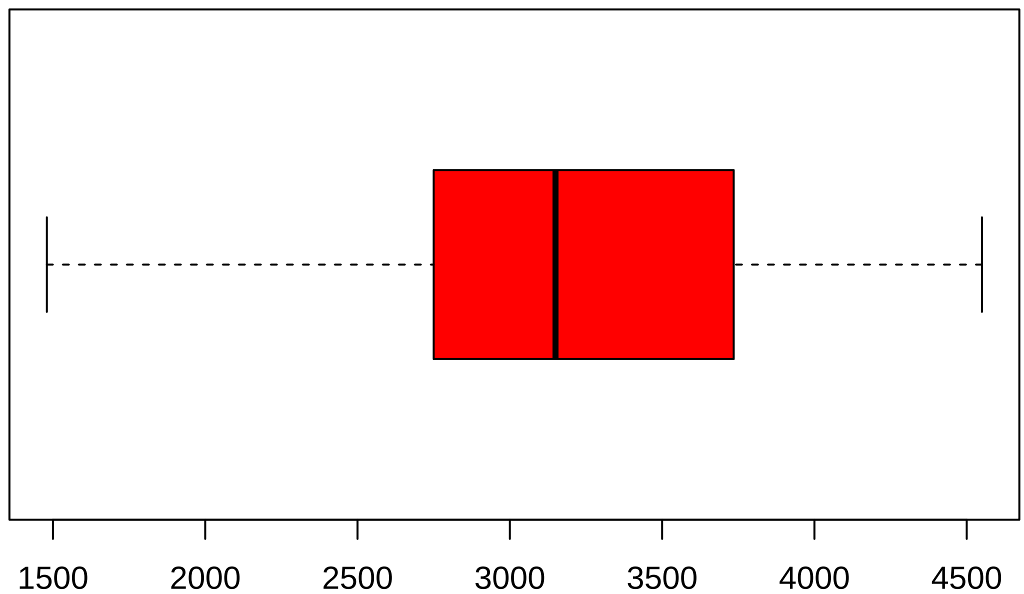
```r
lines(c(min(temp$breaks), (temp$mids),max(temp$breaks)), c(0,temp$counts,0),type="l") # creating a freq
```



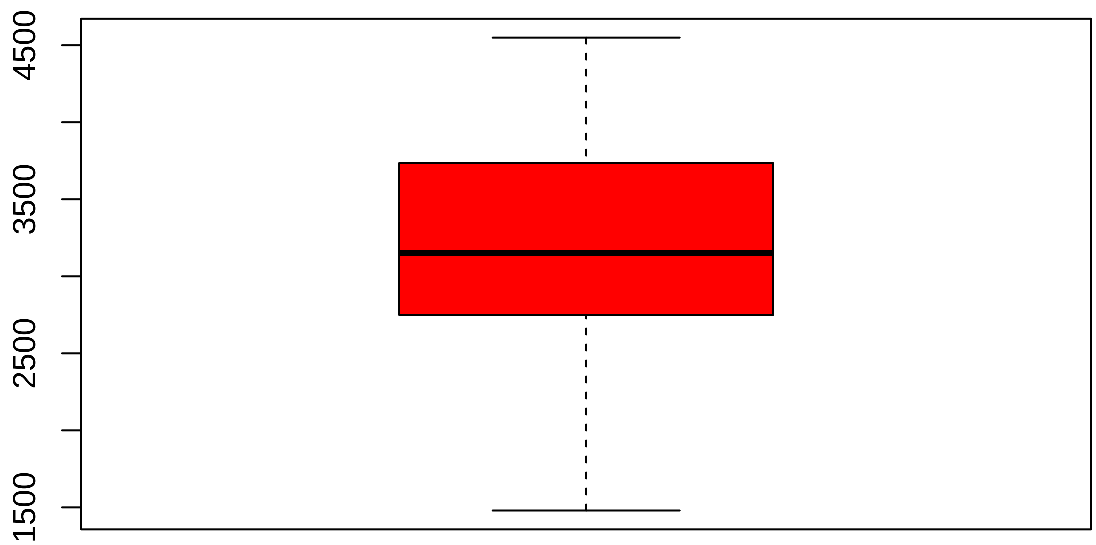
Boxplot

`Boxplots` are used to visualize the distribution of numerical data. `Boxplots` are used to show the median, quartiles, and outliers in a dataset. `Boxplots` are used to visualize the distribution of numerical data.

```r
boxplot(wo, horizontal = T, col = c("red")) # creating a boxplot of wo
```




```r
boxplot(wo, vertical = T, col = c("red")) # creating a boxplot of wo
```




```r
fivenum(wo) # summary  of wo
```

```
## [1] 1480 2750 3150 3735 4550
```

```r
summary(wo) # summary of wo
```

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

```r
#?fivenum
```

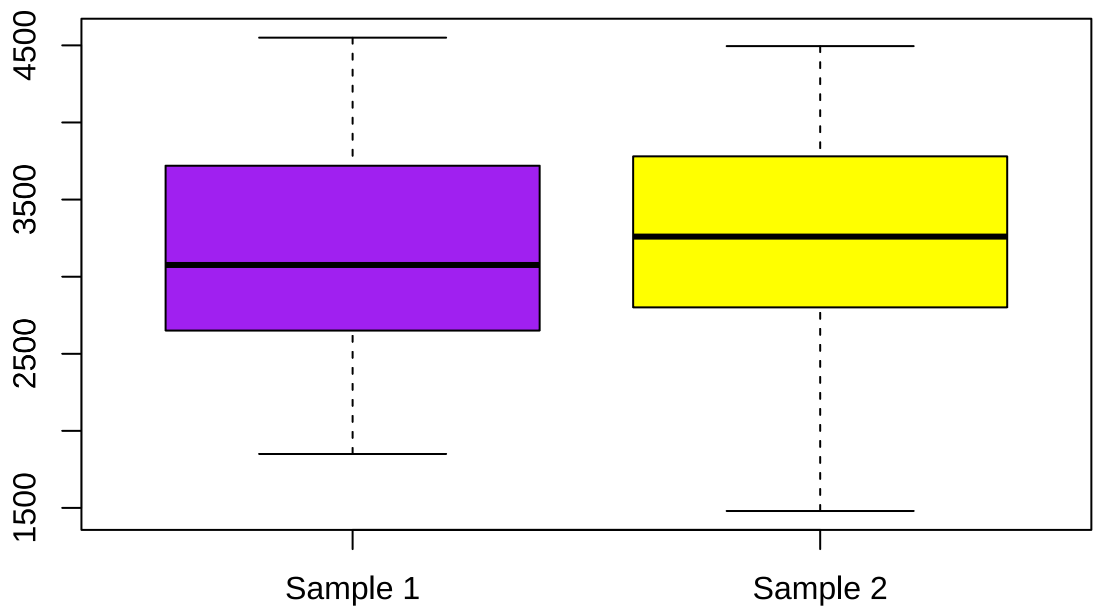
```r
w1 = c(1950, 2090, 2700, 3350, 4200, 3720, 4400, 2980, 3850, 4550, 3050, 2350,1850, 2820, 3670, 2950, 3
fivenum(w1) # summary of w1
```

```
## [1] 1850 2650 3075 3720 4550
```

```r
w2 = c(4450, 3120, 3660, 3070, 3550, 2020, 3500, 2500, 3780, 3940, 3340, 2800, 2850, 4450, 1950,3020,28
fivenum(w2) # summary of w2
```

```
## [1] 1480 2800 3260 3780 4495
```

```
boxplot(w1, w2, names = c("Sample 1", "Sample 2"), col = c("purple", "yellow" )) # creating a boxplot o
```



Classification and Prediction

`Classification and prediction` are important tasks in data analysis. Classification is the process of categorizing data into different classes. Prediction is the process of predicting the value of a target variable based on the values of other variables. `Classification` and `prediction` are used to make decisions based on data.

```
computeCost <- function(X, y, th){
m <- length(y)
return(1/(2*m) * sum((X%*%th - y)^2))
}

computeCost(5,6,7) # computing the cost of 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)) # creating a vector of zeros

for (iter in c(1:num_iters)){
```

```r
  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) # computing the output
```

```
## [1] 1.540012
```

```
## $theta
##        [,1]
## [1,] 0.6225
##
## $F_history
## [1] 5.445000 1.540012
```

Clustering

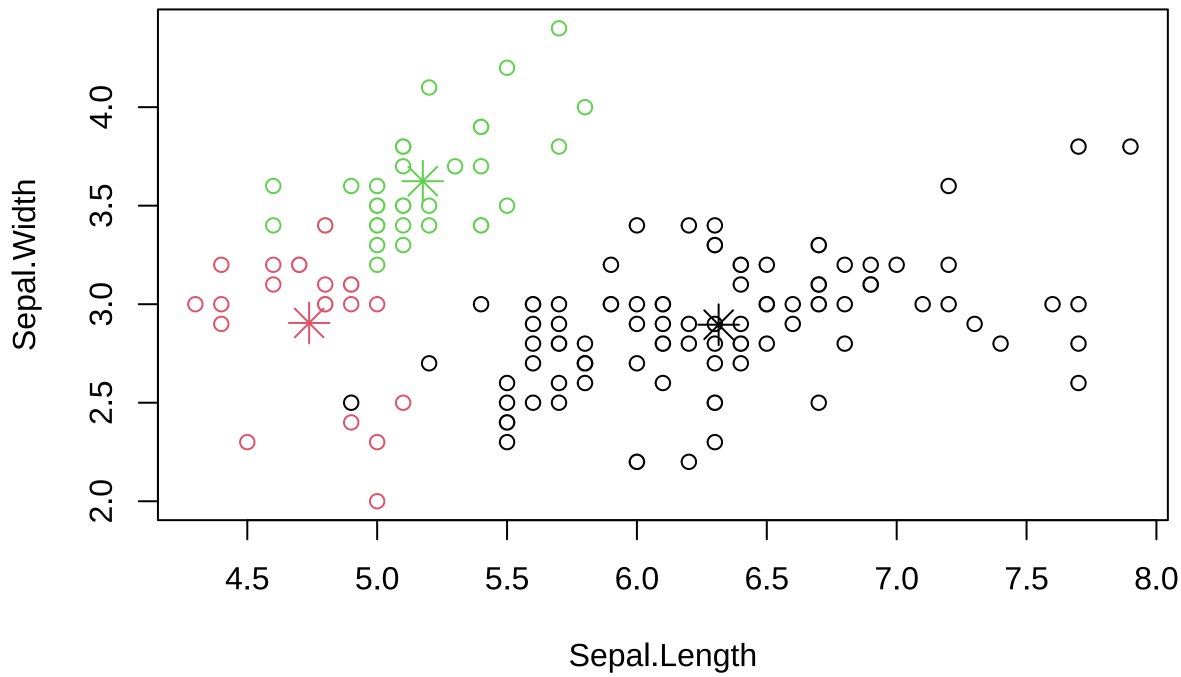
`Clustering` is a technique used to group similar data points together. Clustering is used to identify patterns and relationships in data. Clustering is used to group data points into clusters based on their similarity. Clustering is used to analyze and explore data.

`kmeans` is a popular clustering algorithm that is used to group data points into k clusters. kmeans is an unsupervised learning algorithm that is used to identify patterns and relationships in data. kmeans is used to group data points into clusters based on their similarity.

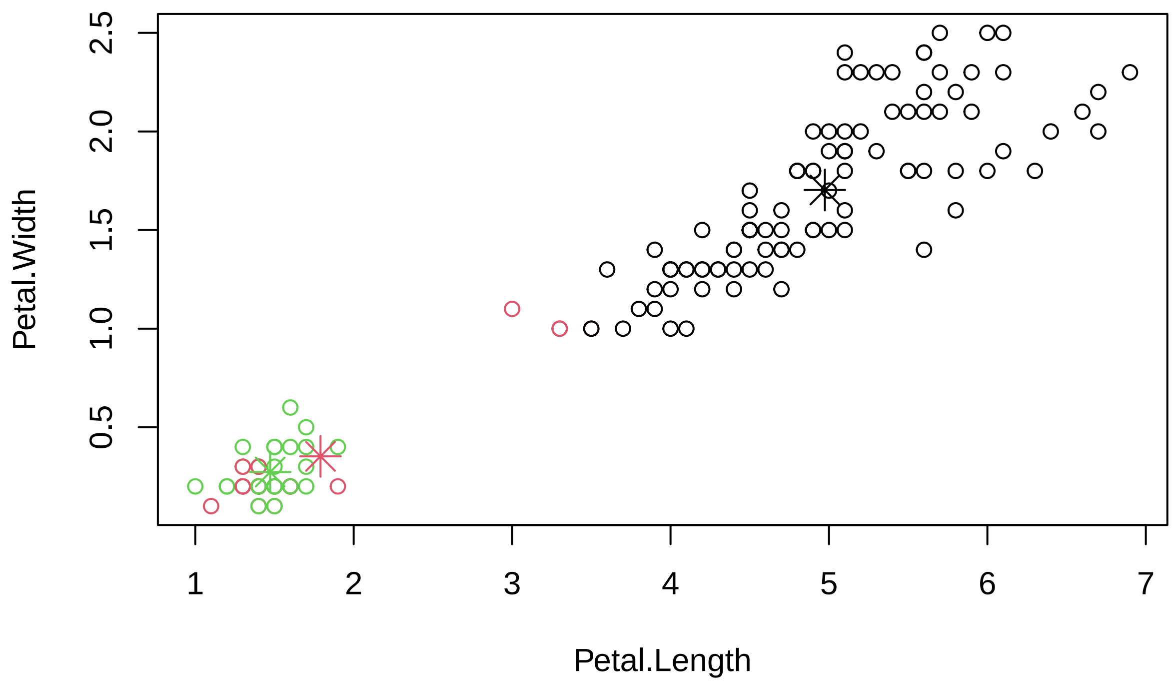
```r
iris_new <- iris # assigning iris to iris_new
### iris_new

iris_new$Species <- NULL # assigning the species class to null

kc <- kmeans(iris_new, 3) # creating a kmeans of iris_new

table(iris$Species, kc$cluster) # creating a table of iris and kc
```

```
##
##                 1  2  3
##   setosa        0 17 33
##   versicolor 46  4  0
##   virginica  50  0  0
```

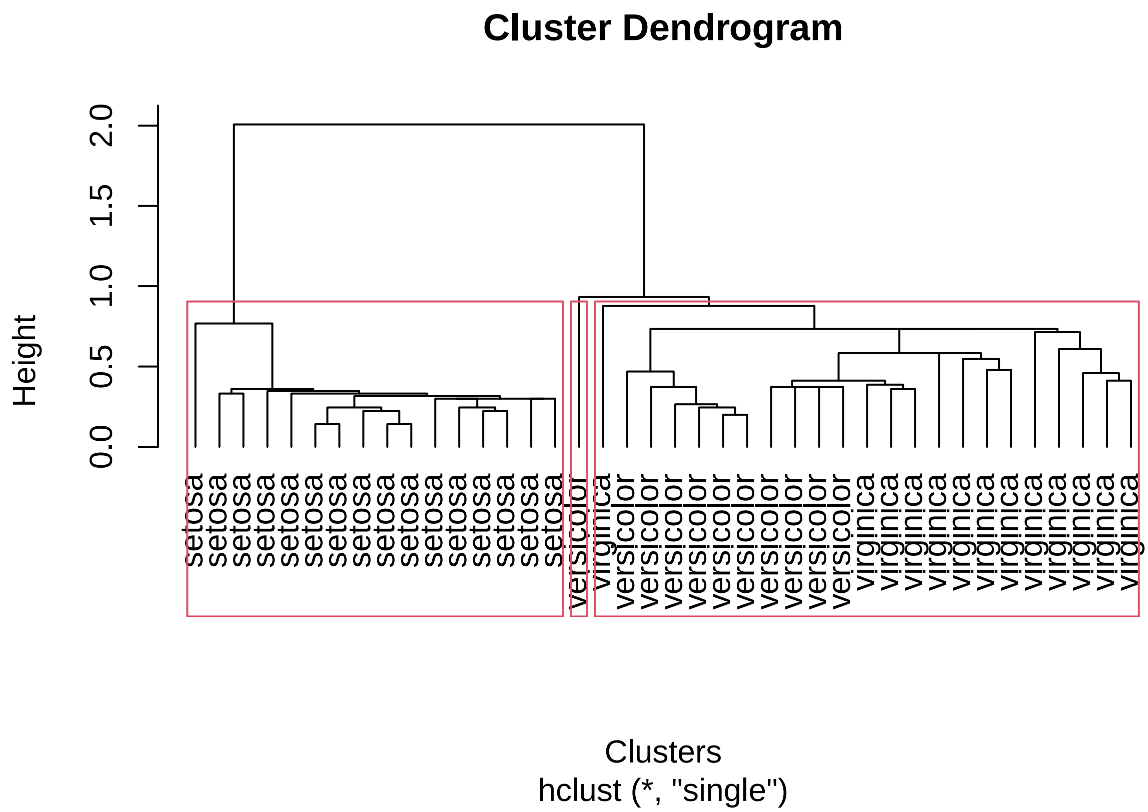
```r
plot(iris_new[c("Sepal.Length", "Sepal.Width")], col = kc$cluster) # visualizing the  iris datasets

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



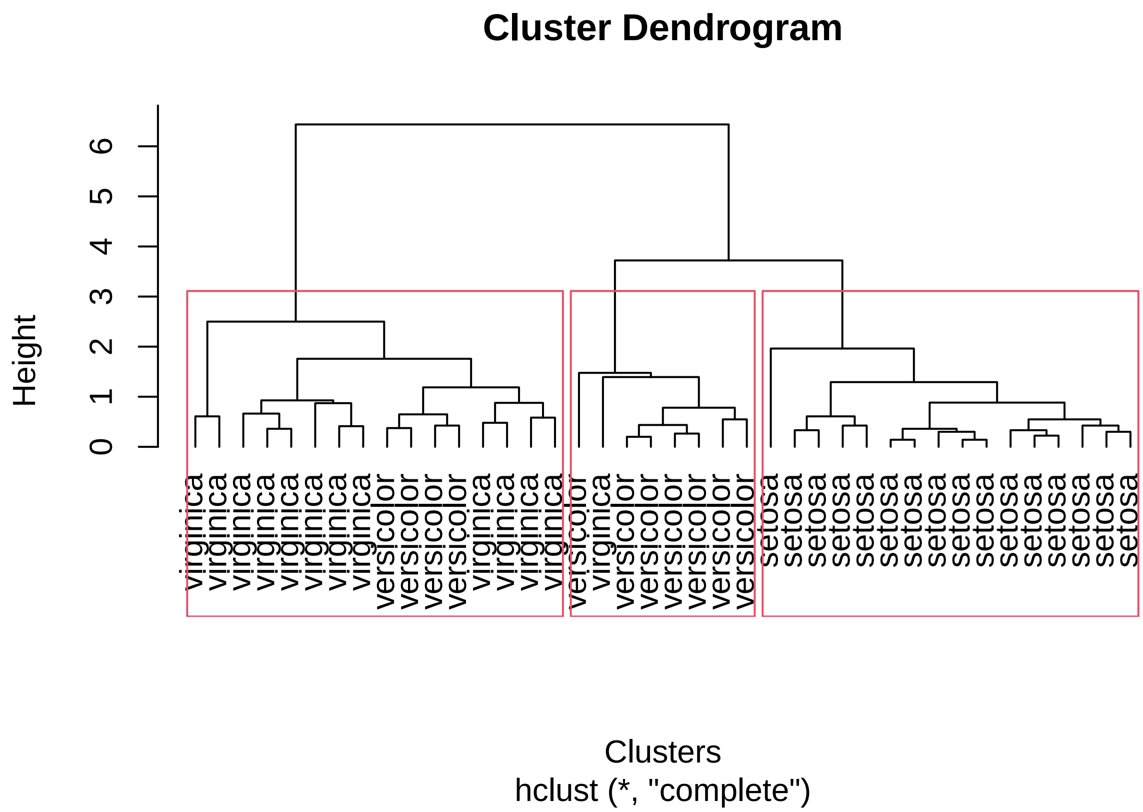

```
plot(iris_new[c("Petal.Length", "Petal.Width")], col = kc$cluster) # visualizing the iris dataset

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



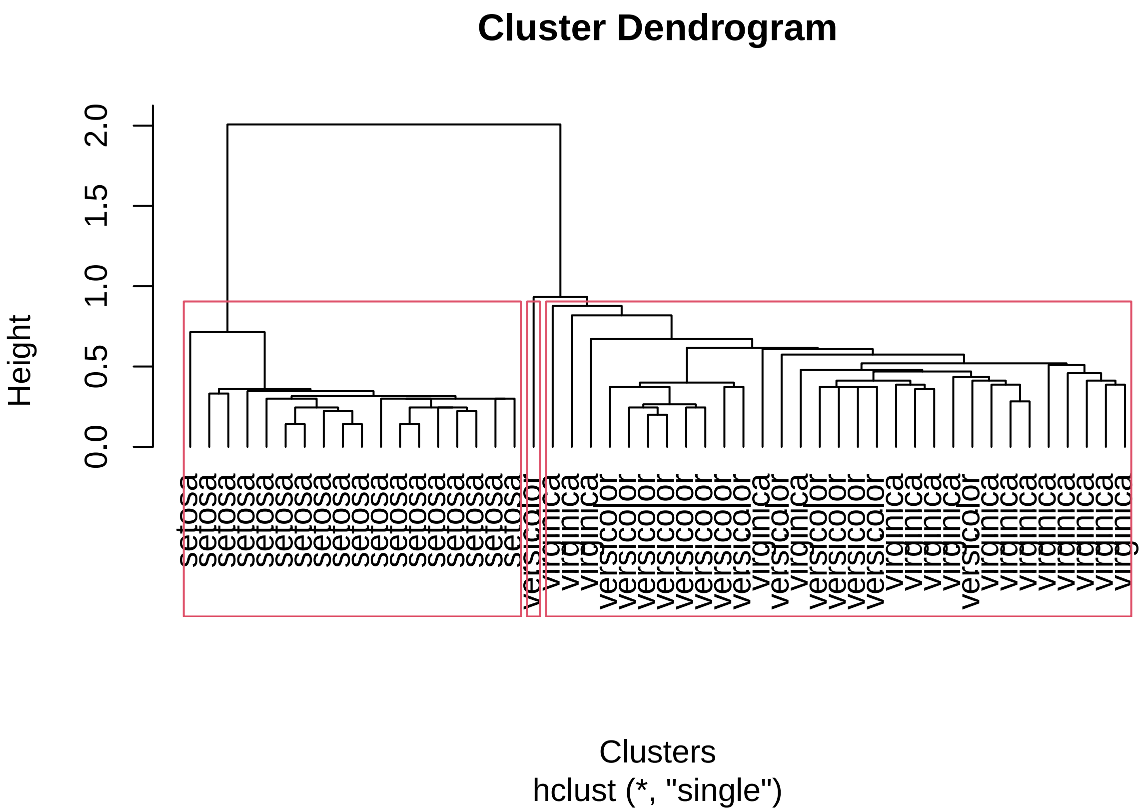

```r
data(iris) # importing the iris dataset

set.seed(500) # setting the seed to 500 to avoid randomness

idx <- sample(1:dim(iris)[1], 40) # sampling the iris dataset

iris_Sample <- iris[idx,] # assigning the sampled iris dataset to iris_Sample

iris_Sample$Species <- NULL # assigning the species class to null

hc <- hclust(dist(iris_Sample), method = "single") # creating a hclust of iris_Sample

plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters") # visualizing the iris dataset

rect.hclust(hc, 3) # creating a rect.hclust of 3
```



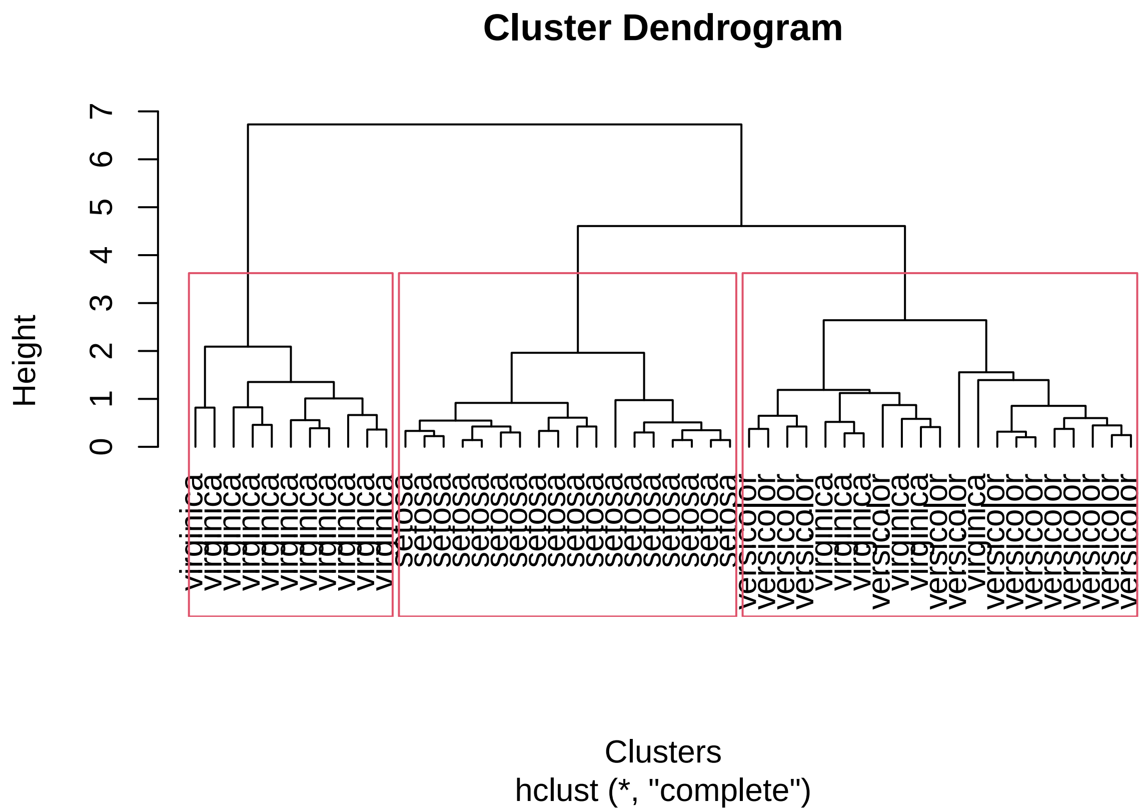

```r
hc <- hclust(dist(iris_Sample),method = "complete") # creating a hclust of iris_Sample

plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters") # visualizing the iris dataset

rect.hclust(hc, 3) # creating a rect.hclust of 3
```




```r
data(iris) # initiating iris dataset

set.seed(500) # setting seed to 500

idx <- sample(1:dim(iris)[1], 50) # sampling the dataset

iris_Sample <- iris[idx,] # assigning the sampled iris dataset to iris_Sample

iris_Sample$Species <- NULL # setting the species column to null

hc <- hclust(dist(iris_Sample), method = "single") # creating a hclust of iris_Sample

plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters") # visualizing the iris dataset

rect.hclust(hc, 3) # creating a rect.hclust of 3
```




```r
hc <- hclust(dist(iris_Sample),method = "complete") # creating a hclust of iris_Sample

plot(hc, hang = -1, labels = iris$Species[idx], xlab = "Clusters") # visualizing the iris dataset

rect.hclust(hc, 3) # creating a rect.hclust of 3
```

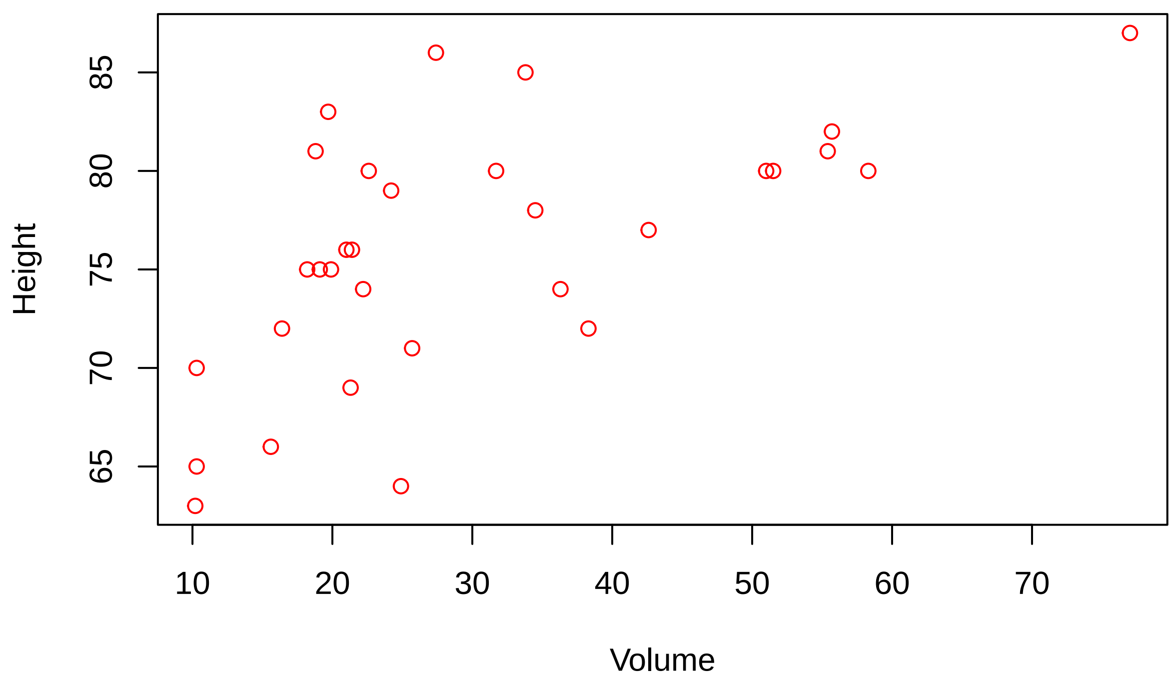



```r
tr = trees # assigning trees to tr

as.data.frame(tr) # converting tr to a dataframe
```

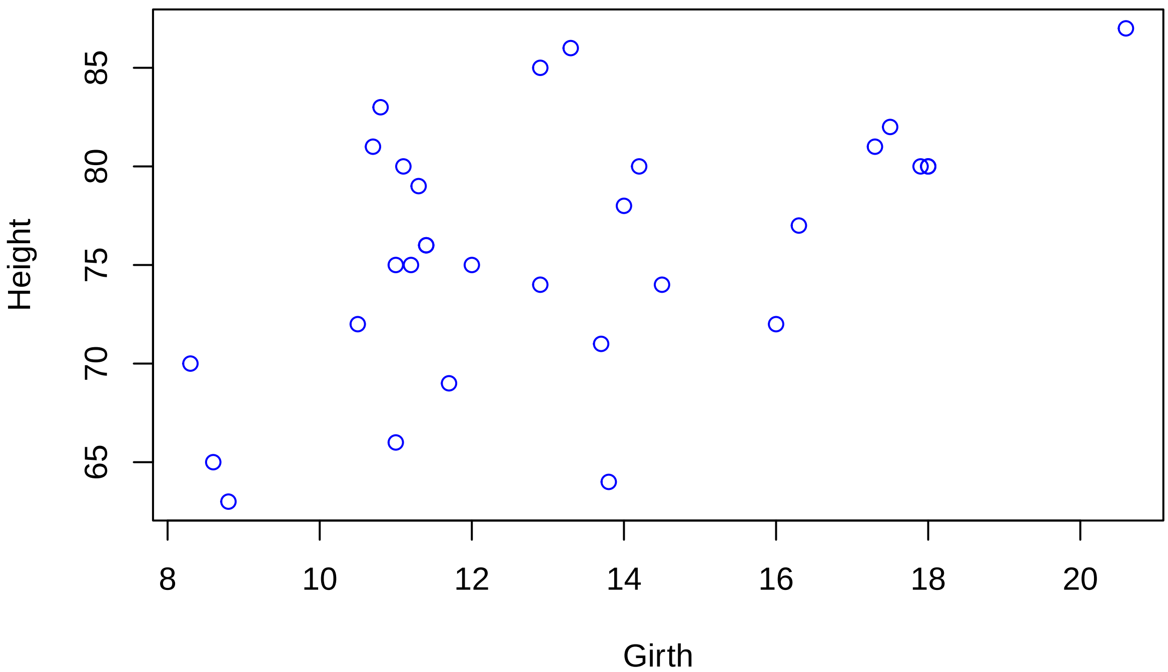
```
##    Girth Height Volume
## 1    8.3     70   10.3
## 2    8.6     65   10.3
## 3    8.8     63   10.2
## 4   10.5     72   16.4
## 5   10.7     81   18.8
## 6   10.8     83   19.7
## 7   11.0     66   15.6
## 8   11.0     75   18.2
## 9   11.1     80   22.6
## 10  11.2     75   19.9
## 11  11.3     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
```

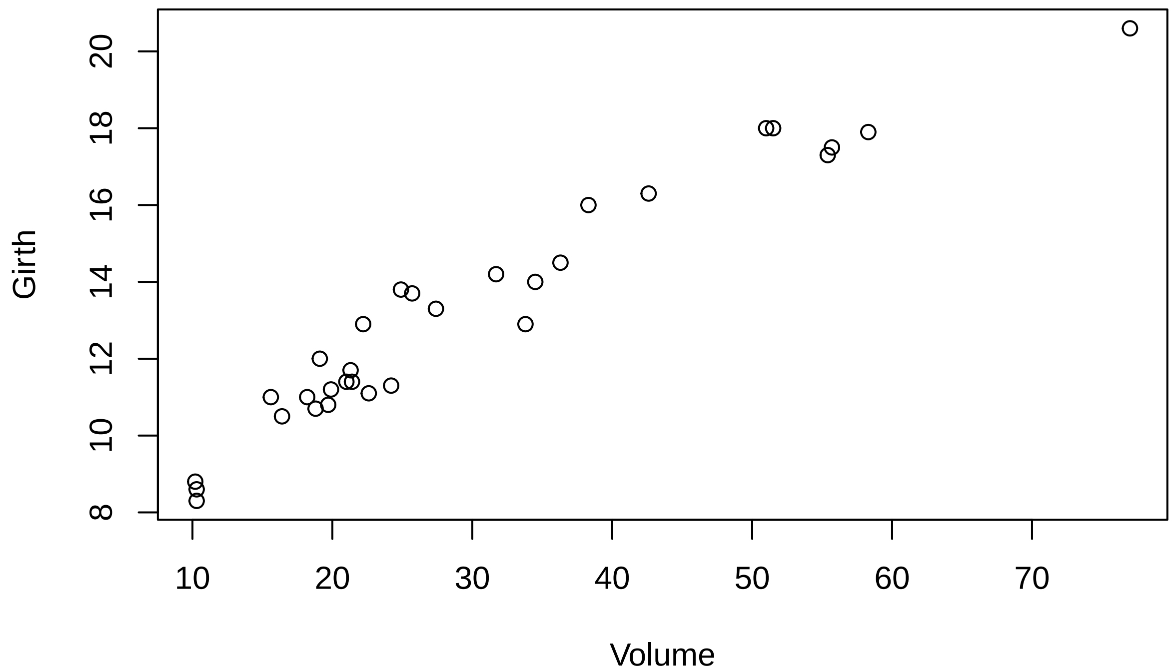
```r
plot(tr[c("Volume","Height")], col = "red") # visualizing the trees dataset
```




```r
plot(tr[c("Girth","Height")], col = "blue" ) # visualizing the trees dataset
```




```
plot(tr[c("Volume","Girth")], col = "black" ) # visualizing the trees dataset
```

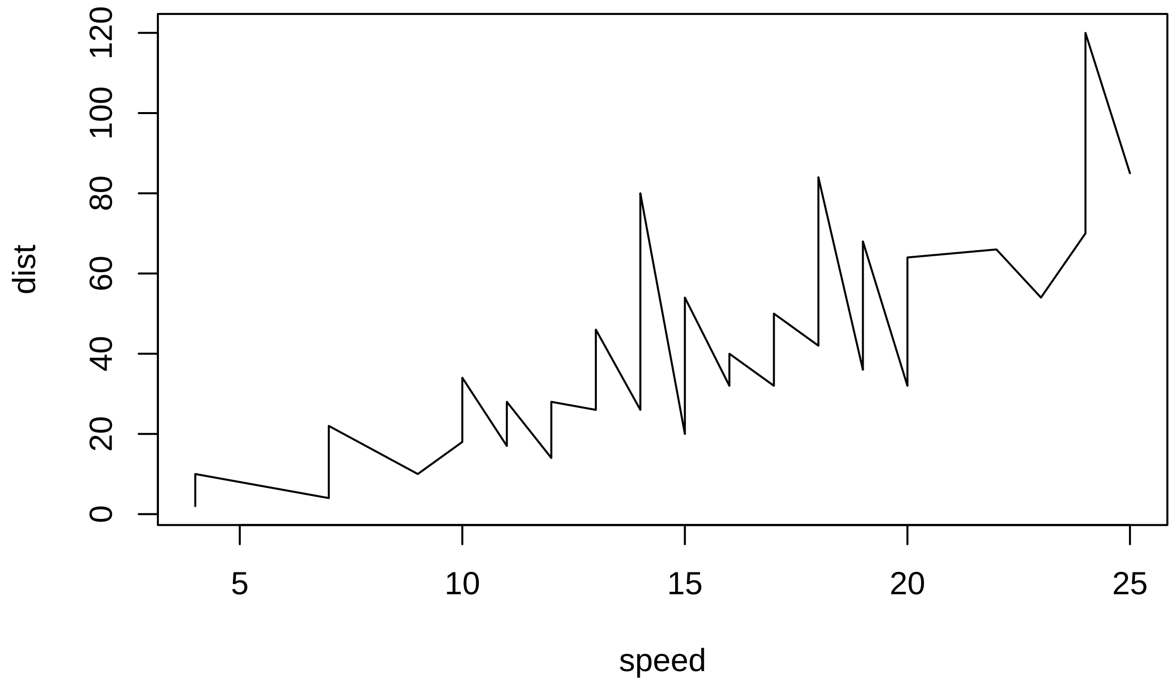



```r
ca = cars # assigning cars to ca

as.data.frame(ca) # converting ca to a dataframe
```

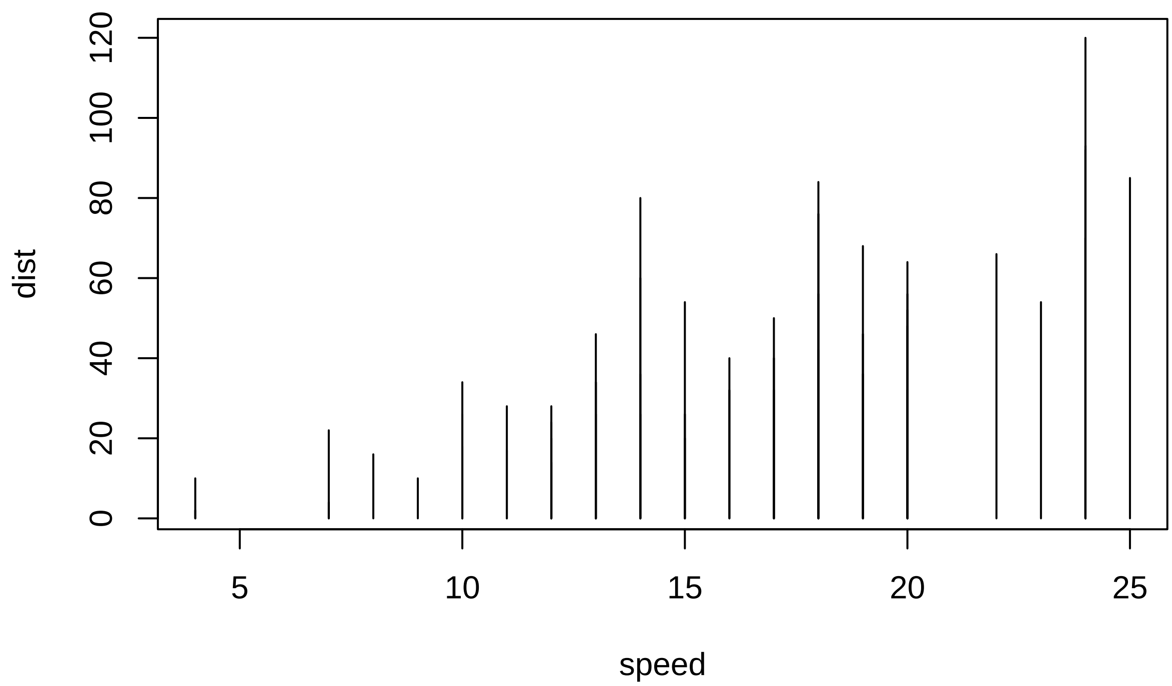
```
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
## 6      9   10
## 7     10   18
## 8     10   26
## 9     10   34
## 10    11   17
## 11    11   28
## 12    12   14
## 13    12   20
## 14    12   24
## 15    12   28
## 16    13   26
## 17    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
```

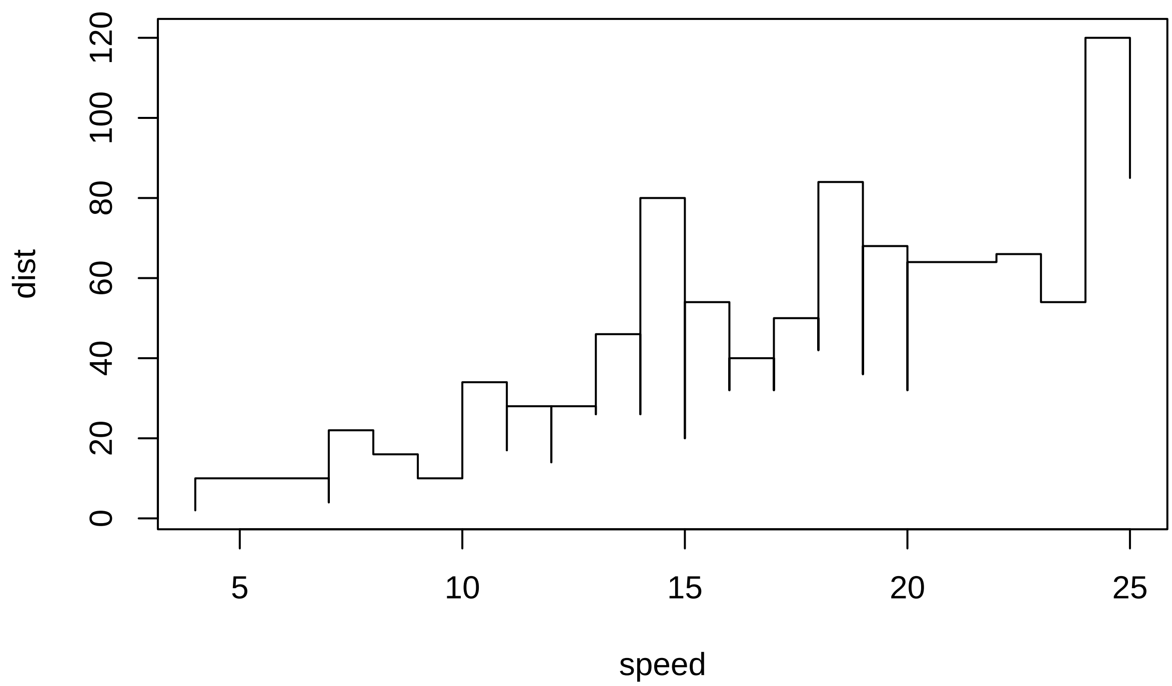
```r
plot(ca[c("speed","dist")], type = "l") # visualizing the cars dataset
```



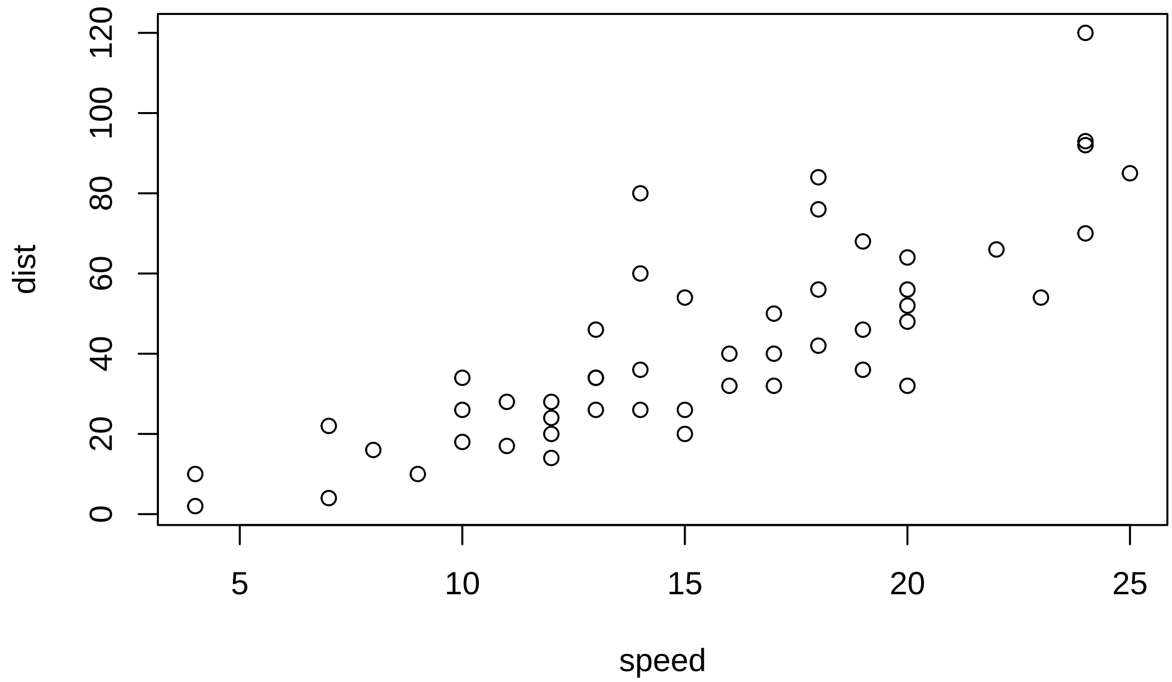

```
plot(ca[c("speed","dist")], type = "h") # visualizing the cars dataset
```




```r
plot(ca[c("speed","dist")], type = "s") # visualizing the cars dataset
```



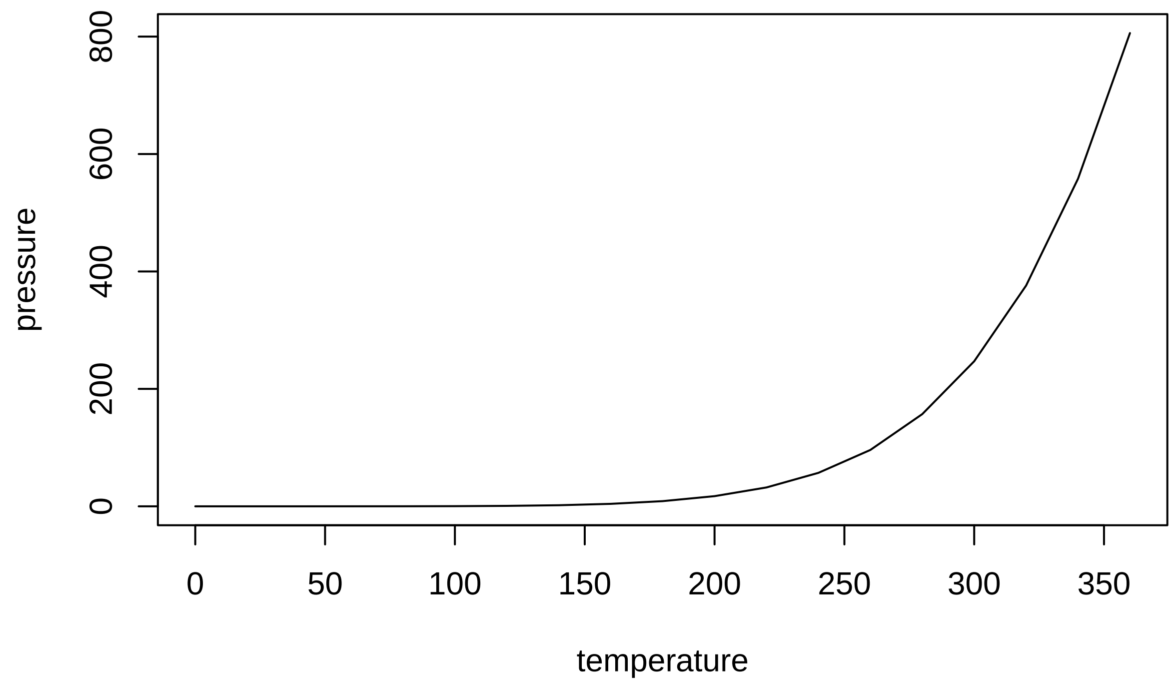

```r
plot(ca[c("speed","dist")], type = "p") # visualizing the cars dataset
```




```r
pre = pressure # assigning pressure to pre

as.data.frame(pre) # converting pre to a dataframe
```

```
##    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
```

```r
plot(pre[c("temperature","pressure")], type="l") # visualizing the pressure dataset
```



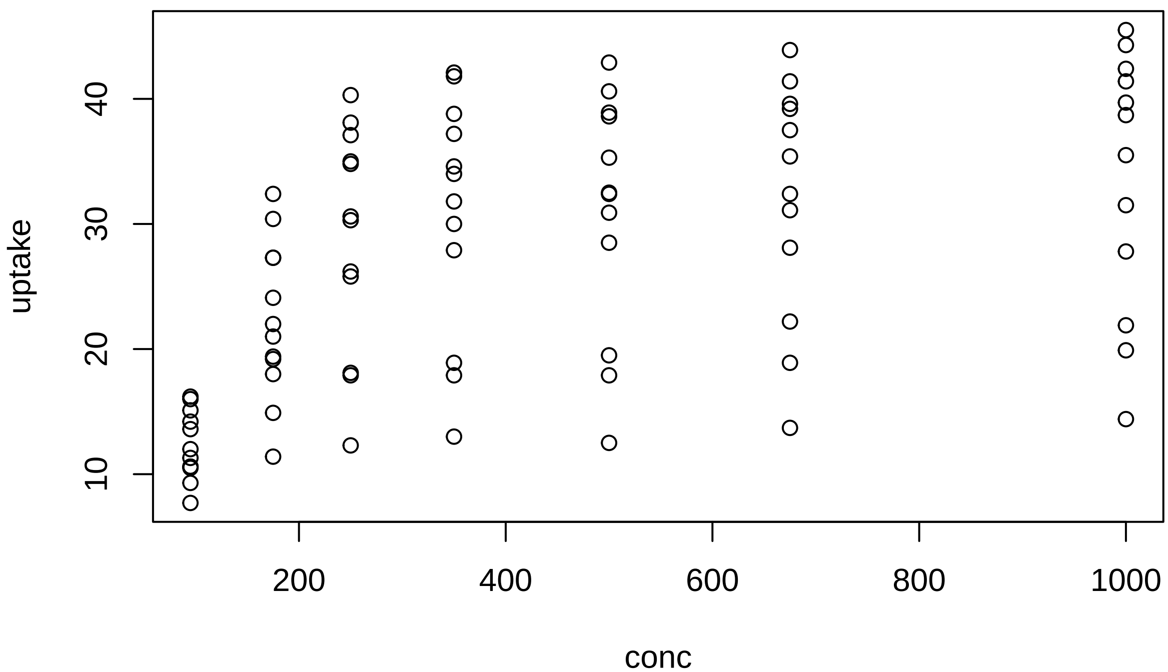

```r
cab = CO2 # assigning CO2 to cab

as.data.frame(cab) # converting cab to a dataframe
```

```
##    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
```

```r
plot(cab[c("conc","uptake")]) # visualizing the CO2 dataset
```

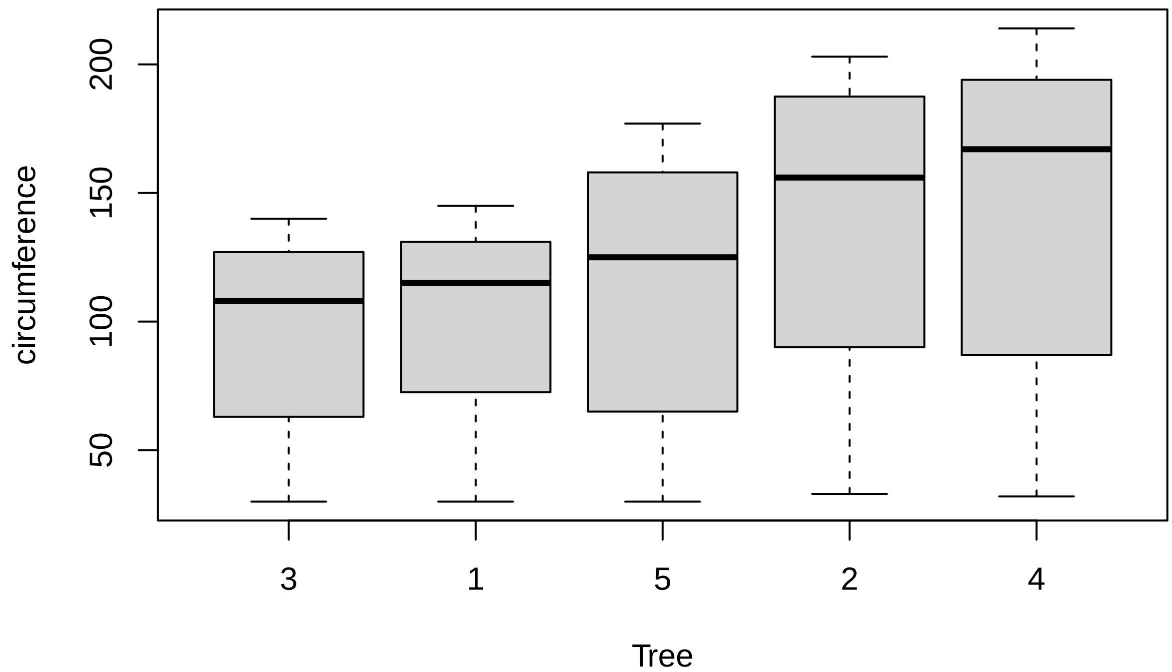



```r
oran = Orange # assigning Orange to oran

as.data.frame(oran) # converting oran to a dataframe
```

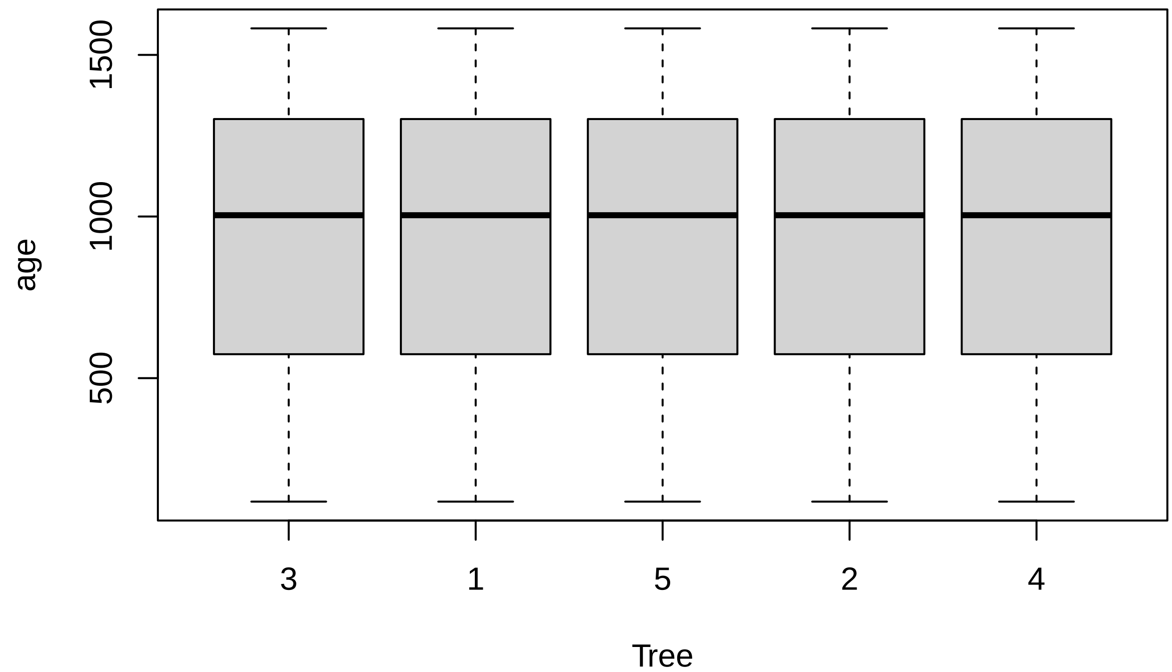
```
##     Tree  age circumference
## 1      1  118            30
## 2      1  484            58
## 3      1  664            87
## 4      1 1004           115
## 5      1 1231           120
```

```
## 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
```

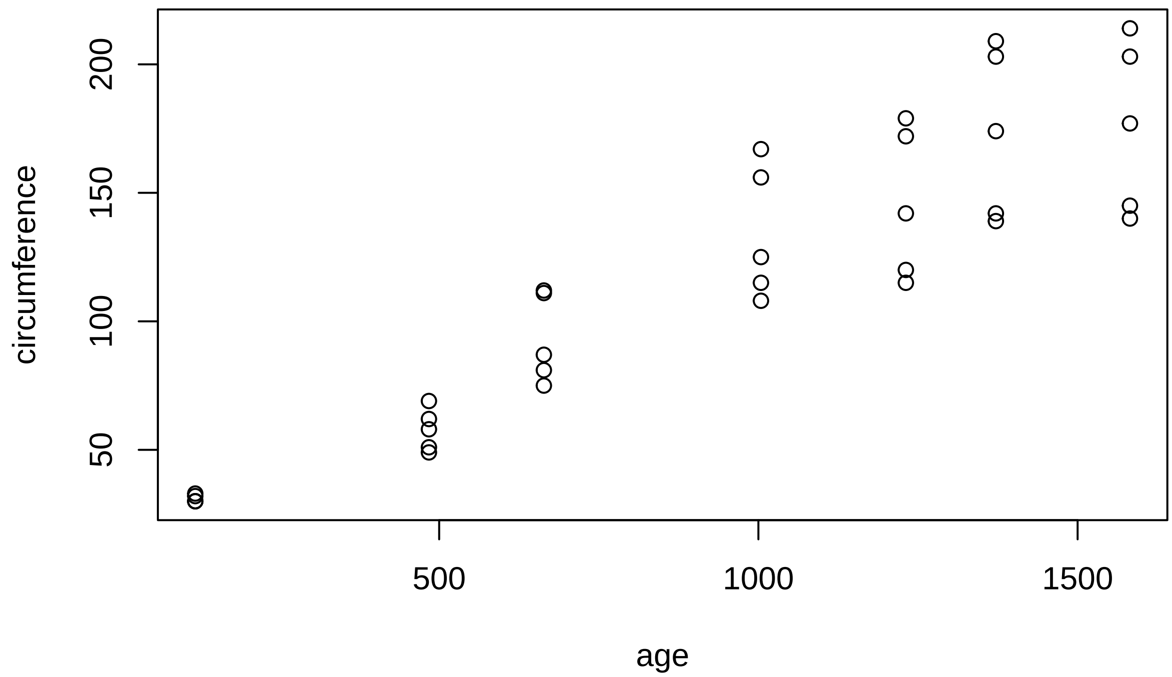
```
plot(oran[c("Tree","circumference")]) # visualizing the Orange dataset
```




```r
plot(oran[c("Tree","age")]) # visualizing the Orange dataset
```




```r
plot(oran[c("age","circumference")]) # visualizing the Orange dataset
```



Mining of frequent itemsets and Association Rules

`Mining of frequent itemsets` and `association rules` is an important task in data analysis. Frequent `itemsets` are sets of items that occur together frequently in a dataset. Association rules are rules that describe the relationships between items in a dataset. Mining of frequent `itemsets` and association rules is used to identify patterns and relationships in data.

```
library(arules) # importing the arules package

db <- list(c("A", "B", "D", "E"), c("B", "C", "E"), c("A", "B", "D", "E"), c("A", "B", "C", "E"), c("A"

frequent <- apriori(db, parameter = list(supp = 0.5, conf = 1, target="frequent itemsets")) # creating
```

```
## 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].
```

```r
inspect(frequent) # inspecting the frequent itemset
```

```
##      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
```

```r
cl <- apriori(db, parameter = list(supp = 0.5, conf = 1, target = "closed")) # creating a closed itemse
```

```
## 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].
```

```r
inspect(cl) # inspecting the closed itemset
```

```
##     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
```

```r
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].
```

```r
# creating a maximal itemset of db

inspect(mx) # inspecting the maximal itemset
```

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

```r
rules <- apriori(db, parameter=list(supp=0.5, conf=1, target="rules")) # creating a rule of db
```

```
## 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].
```

```r
inspect(rules) # inspecting the rule
```

```
##      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
```

```r
data(Adult) # importing the Adult dataset

dim(Adult) # checking the dimensions of Adult
```

```
## [1] 48842   115
```

```r
inspect(apriori(Adult, parameter=list(supp = 0.75))) # inspecting the Adult dataset
```

```
## 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
## [1]  {}                              => {race=White}                    0.8550428  0.8550428 1.0000000
## [2]  {}                              => {native-country=United-States} 0.8974243  0.8974243 1.0000000
## [3]  {}                              => {capital-gain=None}             0.9173867  0.9173867 1.0000000
## [4]  {}                              => {capital-loss=None}             0.9532779  0.9532779 1.0000000
## [5]  {race=White}                    => {native-country=United-States} 0.7881127  0.9217231 0.8550428
## [6]  {native-country=United-States} => {race=White}                    0.7881127  0.8781940 0.8974243
## [7]  {race=White}                    => {capital-gain=None}             0.7817862  0.9143240 0.8550428
## [8]  {capital-gain=None}             => {race=White}                    0.7817862  0.8521883 0.9173867
## [9]  {race=White}                    => {capital-loss=None}             0.8136849  0.9516307 0.8550428
## [10] {capital-loss=None}             => {race=White}                    0.8136849  0.8535653 0.9532779
## [11] {native-country=United-States} => {capital-gain=None}             0.8219565  0.9159062 0.8974243
## [12] {capital-gain=None}             => {native-country=United-States} 0.8219565  0.8959761 0.9173867
## [13] {native-country=United-States} => {capital-loss=None}             0.8548380  0.9525461 0.8974243
## [14] {capital-loss=None}             => {native-country=United-States} 0.8548380  0.8967354 0.9532779
## [15] {capital-gain=None}             => {capital-loss=None}             0.8706646  0.9490705 0.9173867
## [16] {capital-loss=None}             => {capital-gain=None}             0.8706646  0.9133376 0.9532779
## [17] {capital-gain=None,
##       native-country=United-States} => {capital-loss=None}             0.7793702  0.9481891 0.8219565
## [18] {capital-loss=None,
##       native-country=United-States} => {capital-gain=None}             0.7793702  0.9117168 0.8548380
## [19] {capital-gain=None,
##       capital-loss=None}             => {native-country=United-States} 0.7793702  0.8951440 0.8706646
```

```r
inspect(apriori(Adult, parameter = list(supp = 0.75), appearance = list(rhs = "capital-gain=None", defa
```

```
## 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 [0.06s].
## 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 ... [5 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
##     lhs                               rhs                    support confidence  coverage      lift c
## [1] {}                             => {capital-gain=None} 0.9173867  0.9173867 1.0000000 1.0000000 4
## [2] {race=White}                   => {capital-gain=None} 0.7817862  0.9143240 0.8550428 0.9966616 3
## [3] {native-country=United-States} => {capital-gain=None} 0.8219565  0.9159062 0.8974243 0.9983862 4
## [4] {capital-loss=None}            => {capital-gain=None} 0.8706646  0.9133376 0.9532779 0.9955863 4
## [5] {capital-loss=None,
##      native-country=United-States} => {capital-gain=None} 0.7793702  0.9117168 0.8548380 0.9938195 3
```