PES University, Bangalore

UE20CS312 - Data Analytics

First Steps With R!

Course Instructor : Dr. Gowri Srinivasa

Prepared by : Nishanth M S - nishanthmsathish.23@gmail.com

Basics of R

R Command Line

Just like python interpreter , one can use R Command line like a calculator

```r
#This is a comment
7+10       #Addition
```

```
## [1] 17
```

```r
28/10      #Division
```

```
## [1] 2.8
```

```r
28 %/% 10 #Integer division
```

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

```r
7%%4       #modulus
```

```
## [1] 3
```

```r
2^5        #power
```

```
## [1] 32
```

Variables and Data types

```r
a<-10 #numeric
print(a) #prints a's value
```

```
## [1] 10
```

```
print(class(a)) #prints the class of 'a' which is "numeric"
```

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

```
b<-as.integer(11) #integer
cat("b's value : ",b," b's class : ",class(b)) #Used to print multiple variables at once
```

```
## b’s value :  11  b’s class :  integer
```

```
c<-"hello" #character
cat("c's value : ",c," c's class : ",class(c))
```

```
## c’s value :  hello  c’s class :  character
```

```
d<-TRUE #logical
cat("d's value : ",d," d's class : ",class(d))
```

```
## d’s value :  TRUE  d’s class :  logical
```

Vectors and Sequences

A vector is a sequence of data elements of the same basic type.Members of a vector are called components.

```
vector_a<-c(10,20,30,40) #numeric vector
cat("vector_a  : ",vector_a," vector_a's class : ",class(vector_a),
    " length of vector_a : ",length(vector_a),"\n")
```

```
## vector_a  :  10 20 30 40  vector_a’s class :  numeric  length of vector_a :  4
```

```
vector_b<-c("Data","Analytics","R","Python") #character vector
cat("vector_b  : ",vector_b," vector_b's class : ",class(vector_b),
    " length of vector_b : ",length(vector_b))
```

```
## vector_b  :  Data Analytics R Python  vector_b’s class :  character  length of vector_b :  4
```

Sequences return a vector within the given range

```
sequence_a <- seq(4,15)
print(sequence_a)
```

```
##  [1]  4  5  6  7  8  9 10 11 12 13 14 15
```

Vectors follow 1-based indexing

```
a <- seq(4,15)
print(a[1])
```

```
## [1] 4
```

```
print(a[length(a)])
```

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

Loops and conditional statements in R

for loop

```
a <- seq(4,15)
for (digit in a) {
  print(digit)
}
```

```
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
```

while loop

```
a <- seq(4,15)
i<-1
while(i<=length(a)){
  print(a[i])
  i<-i+1
}
```

```
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
```

if else statement

```
a<-21
if(a%%2){
  print("Number is odd")
}else{
  print("Number is even")
}
```

```
## [1] "Number is odd"
```

"ifelse" in R

```
a<-20
ifelse(a%%2,"Number is odd","Number is even")
```

```
## [1] "Number is even"
```

Fuctions in R

```
isEven <-function(a){
if(a%%2){
  print("Number is odd")
}else{
  print("Number is even")
}
}
isEven(7)
```

```
## [1] "Number is odd"
```

```
isEven(10)
```

```
## [1] "Number is even"
```

Installing and Loading a package

To first load a package , one must install it. Let's try and install ggplot2

```
install.packages("ggplot2")
```

After installing , you can load it. You can check if the package is loaded by using "search()"

```
library(ggplot2)
search()
```

```
##  [1] ".GlobalEnv"        "package:ggplot2"   "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

Dataframes and Visualization

Loading a dataframe

Dataframes in R is similar to python.Let's use a preloaded dataset named **txhousing** which is data of house sales in Texas

```
df <- txhousing
```

Viewing the database

```
head(df)
```

```
## # A tibble: 6 x 9
##   city     year month sales   volume median listings inventory  date
##   <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701       6.3 2000
## 2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.
## 3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.
## 4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.
## 5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.
## 6 Abilene  2000     6   156 13910000  66900      780       6.6 2000.
```

```
tail(df)
```

```
## # A tibble: 6 x 9
##   city           year month sales   volume median listings inventory  date
##   <chr>         <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Wichita Falls  2015     2   100 11646765  94000      795       6.8 2015.
## 2 Wichita Falls  2015     3   152 16716584  89200      818       6.8 2015.
## 3 Wichita Falls  2015     4   129 15482194 105300      760       6.4 2015.
## 4 Wichita Falls  2015     5   174 19188181 100000      776       6.4 2015.
## 5 Wichita Falls  2015     6   143 18820752 118800      770       6.2 2015.
## 6 Wichita Falls  2015     7   172 23850905 116700      811       6.5 2016.
```

Basic operations

Viewing column names

```
colnames(df)
```

```
## [1] "city"      "year"      "month"     "sales"     "volume"    "median"
## [7] "listings"  "inventory" "date"
```

Finding dimensions

```
dim(df)
```

```
## [1] 8602    9
```

Slicing

```
top5 <- df[1:5,]
top5
```

```
## # A tibble: 5 x 9
##   city     year month sales   volume median listings inventory  date
##   <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701       6.3 2000
## 2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.
## 3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.
## 4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.
## 5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.
```

Selecting a single column(2 methods)

```
cities <- df$city
cities2 <- df[,"city"]
cities[1:10]
```

```
##  [1] "Abilene" "Abilene" "Abilene" "Abilene" "Abilene" "Abilene" "Abilene"
##  [8] "Abilene" "Abilene" "Abilene"
```

```
head(cities2)
```

```
## # A tibble: 6 x 1
##   city
##   <chr>
## 1 Abilene
## 2 Abilene
## 3 Abilene
## 4 Abilene
## 5 Abilene
## 6 Abilene
```

Preliminary Analysis

```
mean(df$sales,na.rm=TRUE) #na.rm will remove missing values
```

```
## [1] 549.5646
```

```
median(df$sales,na.rm=TRUE)
```

```
## [1] 169
```

```
min(df$sales,na.rm=TRUE)
```

```
## [1] 6
```

```
max(df$sales,na.rm=TRUE)
```

```
## [1] 8945
```

Calculating the summary

```
summary(df)
```

```
##      city                year          month            sales       
##  Length:8602        Min.   :2000   Min.   : 1.000   Min.   :   6.0  
##  Class :character   1st Qu.:2003   1st Qu.: 3.000   1st Qu.:  86.0  
##  Mode  :character   Median :2007   Median : 6.000   Median : 169.0  
##                     Mean   :2007   Mean   : 6.406   Mean   : 549.6  
##                     3rd Qu.:2011   3rd Qu.: 9.000   3rd Qu.: 467.0  
##                     Max.   :2015   Max.   :12.000   Max.   :8945.0  
##                                                     NA's   :568     
##      volume              median          listings       inventory     
##  Min.   :8.350e+05   Min.   : 50000   Min.   :    0   Min.   : 0.000  
##  1st Qu.:1.084e+07   1st Qu.:100000   1st Qu.:  682   1st Qu.: 4.900  
##  Median :2.299e+07   Median :123800   Median : 1283   Median : 6.200  
##  Mean   :1.069e+08   Mean   :128131   Mean   : 3217   Mean   : 7.175  
##  3rd Qu.:7.512e+07   3rd Qu.:150000   3rd Qu.: 2954   3rd Qu.: 8.150  
##  Max.   :2.568e+09   Max.   :304200   Max.   :43107   Max.   :55.900  
##  NA's   :568         NA's   :616      NA's   :1424    NA's   :1467    
##       date     
##  Min.   :2000  
##  1st Qu.:2004  
##  Median :2008  
##  Mean   :2008  
##  3rd Qu.:2012  
##  Max.   :2016  
## 
```

Sorting a DataFrame

```
sortdf <- df[order(df$sales, decreasing = TRUE),]
head(sortdf)
```

```
## # A tibble: 6 x 9
##   city     year month sales     volume median listings inventory  date
##   <chr>   <int> <int> <dbl>      <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Houston  2015     7  8945 2568156780 217600    23875       3.4 2016.
## 2 Houston  2006     6  8628 1795898108 155200    36281       5.6 2006.
## 3 Houston  2013     7  8468 2168720825 187800    21497       3.3 2014.
## 4 Houston  2015     6  8449 2490238594 222400    22311       3.2 2015.
## 5 Houston  2013     5  8439 2121508529 186100    20526       3.3 2013.
## 6 Houston  2014     6  8391 2342443127 211200    19725       2.9 2014.
```

Filtering a DataFrame. Let's choose records belonging to the city *Houston*

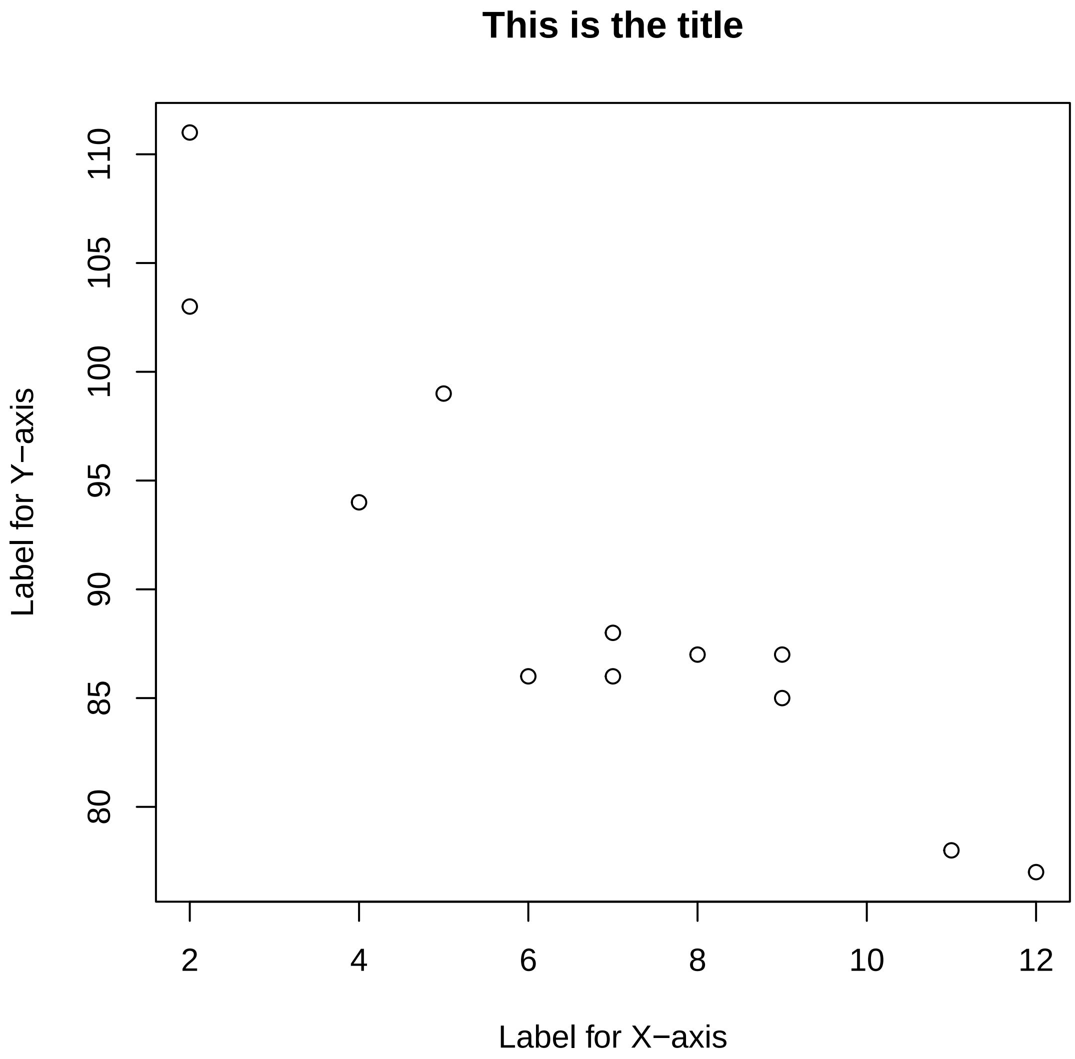
```
houston_data <- df[df$city=="Houston",]
head(houston_data)
```

```
## # A tibble: 6 x 9
##   city     year month sales    volume median listings inventory  date
##   <chr>   <int> <int> <dbl>     <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Houston  2000     1  2653 381805283 102500    16768       3.9 2000
## 2 Houston  2000     2  3687 536456803 110300    16933       3.9 2000.
## 3 Houston  2000     3  4733 709112659 109500    17058       3.9 2000.
## 4 Houston  2000     4  4364 649712779 110800    17716       4.1 2000.
## 5 Houston  2000     5  5215 809459231 112700    18461       4.2 2000.
## 6 Houston  2000     6  5655 887396592 117900    18959       4.3 2000.
```

Visualization

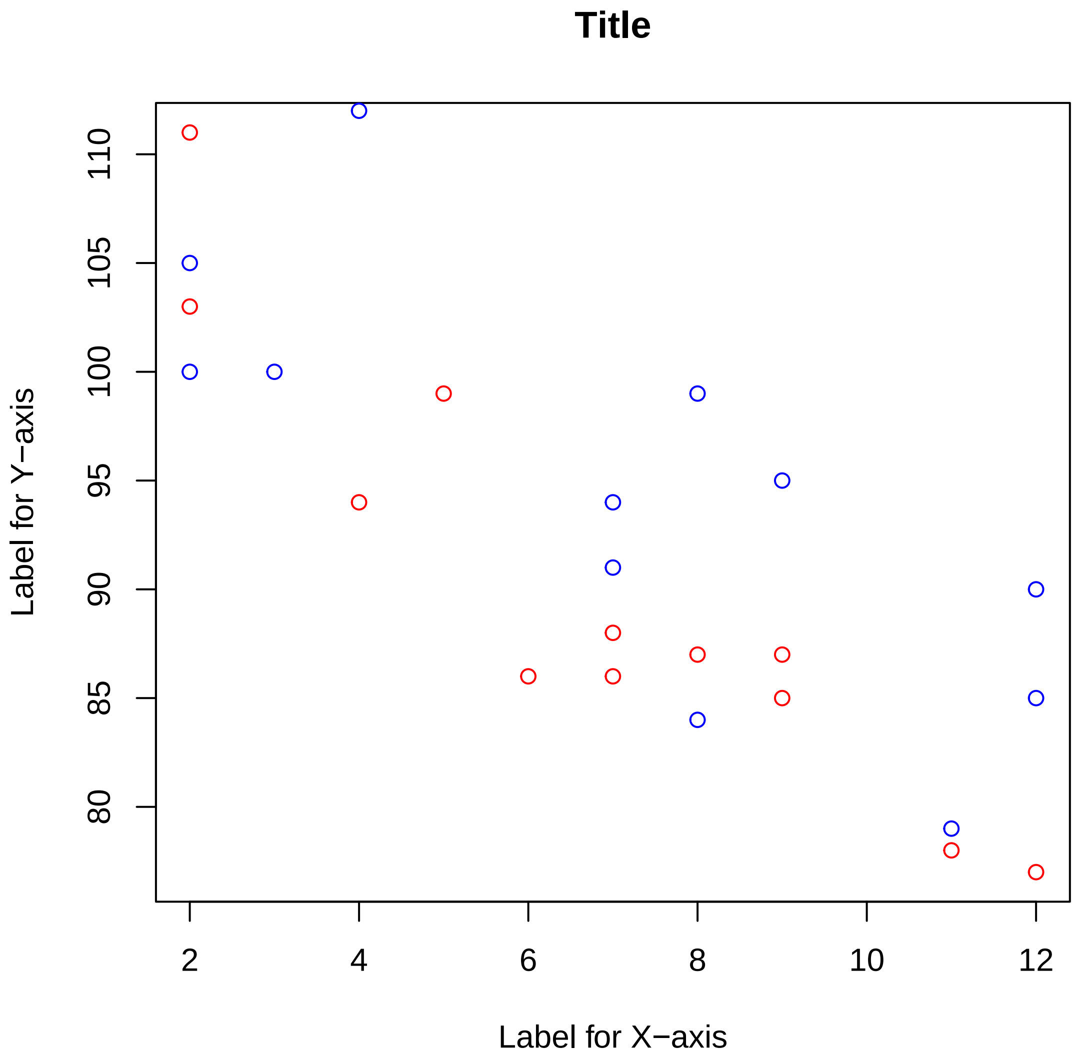
Scatter plot

```
x <- c(5,7,8,7,2,2,9,4,11,12,9,6)
y <- c(99,86,87,88,111,103,87,94,78,77,85,86)

plot(x, y, main="This is the title", xlab="Label for X-axis", ylab="Label for Y-axis")
```



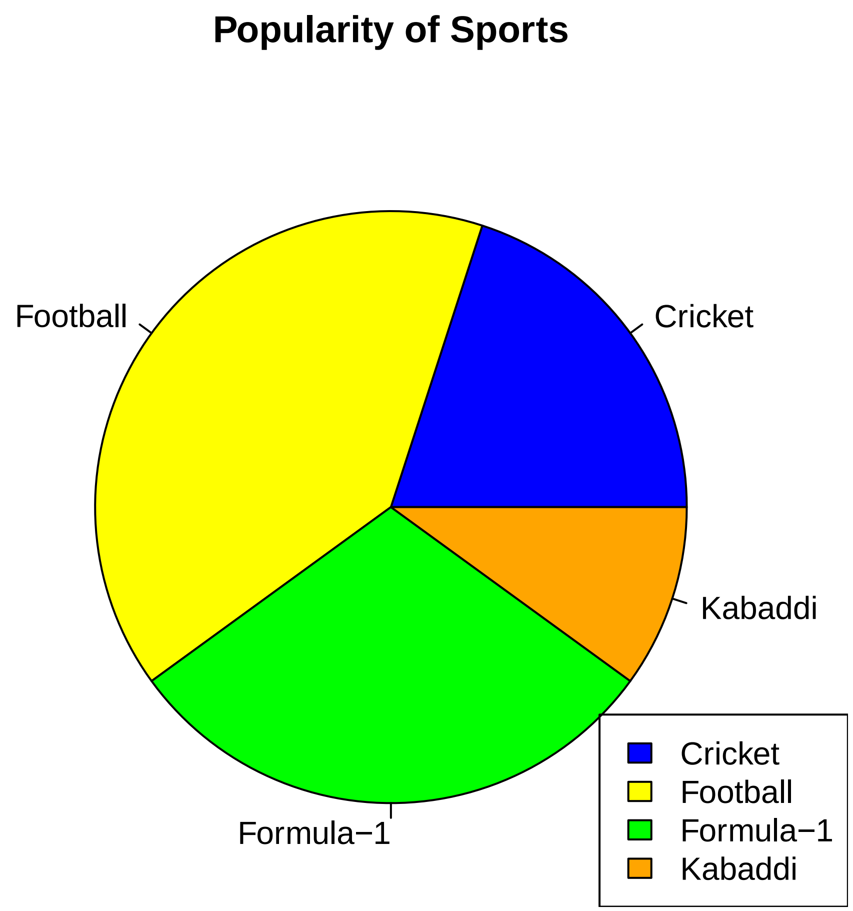
What if we want to plot two different sets of data?

```
x1 <- c(5,7,8,7,2,2,9,4,11,12,9,6)
y1 <- c(99,86,87,88,111,103,87,94,78,77,85,86)


x2 <- c(2,2,8,1,15,8,12,9,7,3,11,4,7,14,12)
y2 <- c(100,105,84,105,90,99,90,95,94,100,79,112,91,80,85)

plot(x1, y1, main="Title", xlab="Label for X-axis", ylab="Label for Y-axis", col="red")
points(x2, y2, col="blue")
```



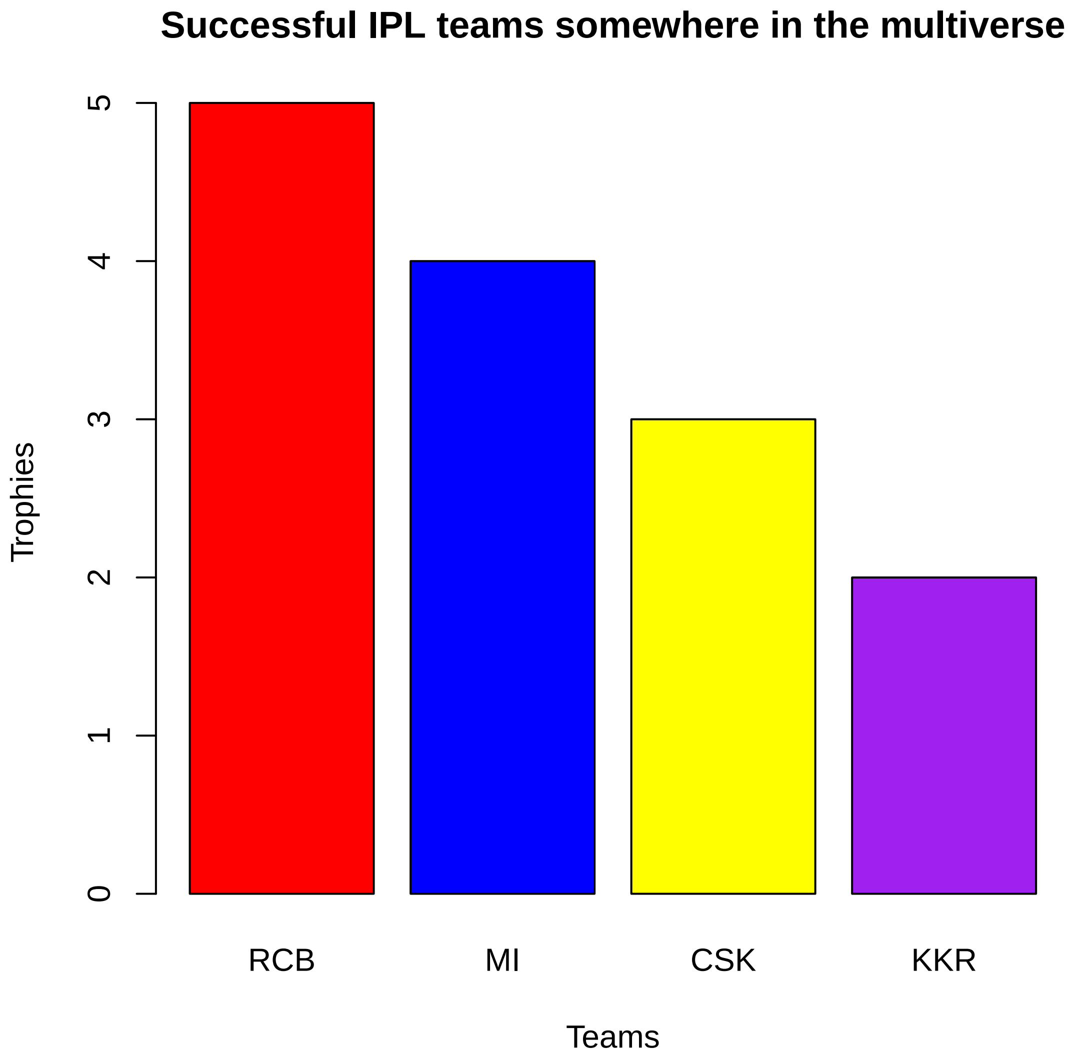
Pie Chart

```
# Create a vector of pies
x <- c(20,40,30,10)
# Create a vector of labels
mylabel <- c("Cricket", "Football", "Formula-1", "Kabaddi")

# Create a vector of colors
colors <- c("blue", "yellow", "green", "orange")

# Display the pie chart with colors
pie(x, label = mylabel, main = "Popularity of Sports", col = colors)

# Display the explanation box
legend("bottomright", mylabel, fill = colors)
```



Bar Plot

```
x <- c("RCB", "MI", "CSK", "KKR")
y <- c(5, 4, 3, 2)

barplot(y, names.arg = x, col = c("red","blue","yellow","purple"),main="Successful IPL teams somewhere
```



Resources To gain more experience and understanding in R , you can check these resources out!

- Check out this beautifully comprehensive resource for everything you need to get started with R.
- Interactive R Tutorials at W3 Schools
- The R documentation