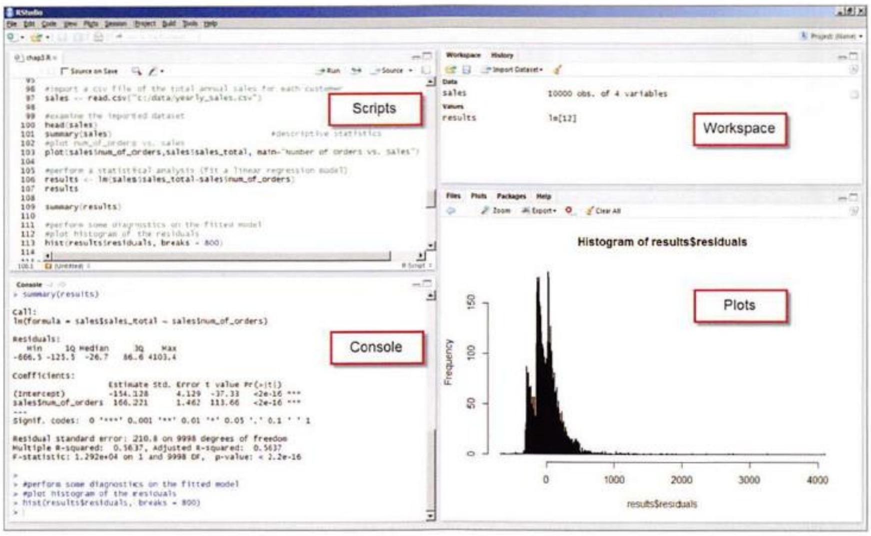
1. R Graphical User Interfaces

2.



```
#import a csv file of the total annual sales for each customer
sales <- read.csv("c:/data/yearly_sales.csv")

#examine the imported dataset
head(sales)
summary(sales)                              #descriptive statistics
#plot num_of_orders vs. sales
plot(sales$num_of_orders,sales$sales_total, main="Number of Orders vs. Sales")

#perform a statistical analysis (fit a linear regression model)
results <- lm(sales$sales_total~sales$num_of_orders)
results

summary(results)

#perform some diagnostics on the fitted model
#plot histogram of the residuals
hist(results$residuals, breaks = 800)
```

```
Data
sales          10000 obs. of 4 variables
Values
results        lm[12]
```

```
> summary(results)

Call:
lm(formula = sales$sales_total ~ sales$num_of_orders)

Residuals:
   Min     1Q Median     3Q    Max
-666.5 -125.5  -26.7   86.6 4103.4

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         -154.128      4.129  -37.33   <2e-16 ***
sales$num_of_orders  166.221      1.462  113.66   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 210.8 on 9998 degrees of freedom
Multiple R-squared:  0.5637,	Adjusted R-squared:  0.5637
F-statistic: 1.292e+04 on 1 and 9998 DF,  p-value: < 2.2e-16

>
> #perform some diagnostics on the fitted model
> #plot histogram of the residuals
> hist(results$residuals, breaks = 800)
>
```

RStudio GUI

2. R-Objects

1. **Vectors**
2. **Matrices**
3. **Arrays**
4. **Data Frames**
5. **Factors**
6. **List**

1. Vectors:

Vectors are one-dimensional arrays that can hold numeric data, character data or logical data. The combine function c() is used to form the vector.

# create a vector

> a <- c(1,2,3,4,5,-1,-2)

> b <- c("One","two","three")

> c <- c(TRUE,TRUE,FALSE,TRUE)

>print(a)

[1] 1 2 3 4 5 -1 -2

>print(b)

[1] "One"   "two"   "three"

> print(c)

```
[1]  TRUETRUE FALSE  TRUE
```

You can refer the element of vector by their position.

```
>b[2]
[1] "two"
```

The colon operator is used to generate the sequence of elements from the vector

```
>a[2:6]
[1]  2  3  4  5 -1
```

2. Matrices:

- ✓ A matrix is a two-dimensional array in which each element has same object type(numeric, integer, or logical).
- ✓ Matrices are created with the Matrix function.
- ✓ A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
>M = matrix( c("a","'a","b","c","b","a"), nrow=2,ncol=3,byrow = TRUE)
>print(M)
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"


> y <- matrix(1:20,nrow=5,ncol=4)
>y
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
>cells<- c(1,26,24,68)
>rnames<- c("R1","R2")
```

```
>cnames<- c("C1","C2")

>mymatrix<-  matrix(cells,nrow=2,ncol=2,byrow=TRUE,dimnames=list(rnames,cnames))

>mymatrix

   C1 C2

R1  1 26

R2 24 68

>mymatrix<- matrix(cells,nrow=2,ncol=2,byrow=FALSE,dimnames=list(rnames,cnames))

>mymatrix

   C1 C2

R1  1 24

R2 26 68
```

```
> x <- matrix(1:10, nrow=2)

>x

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

[1,]    1    3    5    7    9

[2,]    2    4    6    8   10

>x[2,]

[1]  2  4  6  8 10

>x[ ,2]

[1] 3 4

>x[1,4]

[1] 7

>x[1, c(4,5)]

[1] 7 9
```

3. Arrays

- ✓ While matrices are confined to two dimensions, arrays can be of any number of dimensions.
- ✓ The array function takes a dim attribute which creates the required number of dimension.

✓ In the below example we create an array with two elements which are 3x3 matrices each. Like matrices, they must be a single mode.

```
# Create an array.

> a <- array(c('green','yellow'),dim=c(3,3,2))

>print(a)

, , 1

     [,1]     [,2]     [,3]

[1,] "green"  "yellow" "green"

[2,] "yellow" "green"  "yellow"

[3,] "green"  "yellow" "green"


, , 2

     [,1]     [,2]     [,3]

[1,] "yellow" "green"  "yellow"

[2,] "green"  "yellow" "green"

[3,] "yellow" "green"  "yellow"


> dim1 <- c("A1","A2")

> dim2 <- c("B1","B2","B3")

> dim3 <- c("C1","C2","C3","C4")

> Z <- array(1:24,c(2,3,4), dimnames=list(dim1,dim2,dim3))

> Z

, , C1

   B1 B2 B3
A1  1  3  5
A2  2  4  6


, , C2

   B1 B2 B3
  A1  7  9 11
 A2  8 10 12
```

```
, , C3

   B1 B2 B3

A1 13 15 17

A2 14 16 18


, , C4

   B1 B2 B3

A1 19 21 23

A2 20 22 24
```

4. Data Frames

- A data frame is more general than a matrix in that different columns can contains different data objects (numeric, character and so on)
- Data frames are the most common data structure used in R. A data frame is created using data. Frame function

```
>

>patientdata<-data.frame(patientid,age,diabetes,status)

>patientdata

patientid age diabetes    status

1         1  25    Type1      Poor

2         2  34    Type1 Excellent

3         3  48    Type1   Average

4         4  52    Type1      High
```

```
>patientdata[1:2]

patientid age

1         1  25
```

```
2        2  34

3        3  48

4        4  52
```

```
>patientdata[c("diabetes","status")]

diabetes    status

1    Type1      Poor

2    Type1 Excellent

3    Type1   Average

4    Type1      High
```

5. Factors

- Variables can be described as nominal, ordinal and continuous.
- In categorical variable, there won't be any implied order. Eg. Diabetes (Type1, Type2)
- Ordinal variables imply some order of information. Eg. Status (Poor, Improved, Excellent)
- Variables having continuous values have been called as Continuous attributes.
  Eg. Age(21,23,34,45)
- Categorical and ordinal variables in R are called as factors. Factors are crucial in R because, they determine how data are analysed and presented visually.

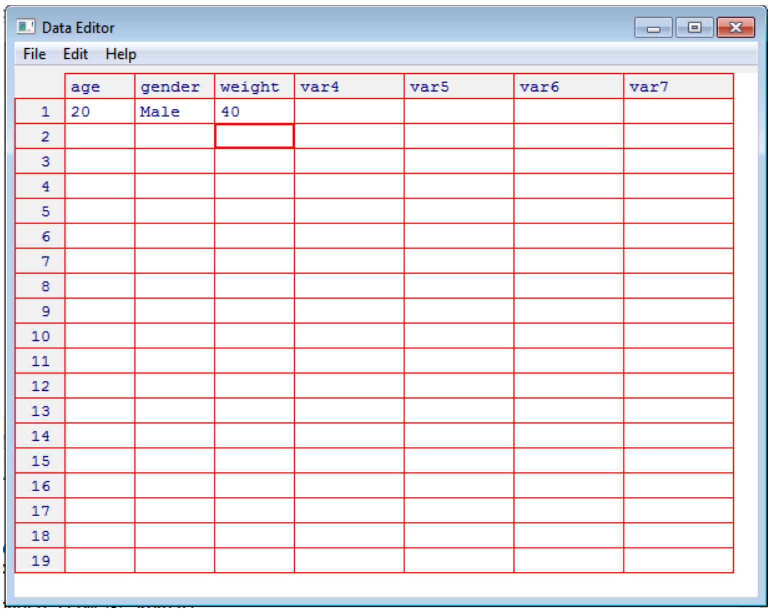
```
> patientcodes<- c(1,2,3,4)
> age<- c(30,32,34,40)
> diabetes<- c("Type1","Type2","Type1","Type1")
> status<- c("Poor","Improved","Excellent","Poor")
> diabetes<- factor(diabetes)
> status<- factor(status)
> patientdata<- data.frame(patientcodes,age,diabetes,status)
> patientdata

  patientcodes age diabetes    status
1            1  30    Type1      Poor
2            2  32    Type2  Improved
3            3  34    Type1 Excellent
4            4  40    Type1      Poor

> str(patientdata)
'data.frame':	4 obs. of  4 variables:
 $ patientcodes: num  1 2 3 4
 $ age         : num  30 32 34 40
 $ diabetes    : Factor w/ 2 levels "Type1","Type2": 1 2 1 1
 $ status      : Factor w/ 3 levels "Excellent","Improved",..: 3 2 1 3
```

```
>summary(patientdata)

patientcodes      age        diabetes      status
Min.:1.00   Min.   :30.0   Type1:3   Excellent:1
 1st Qu.:1.75   1st Qu.:31.5   Type2:1   Improved :1
Median :2.50   Median :33.0            Poor     :2
 Mean   :2.50   Mean   :34.0
 3rd Qu.:3.25   3rd Qu.:35.5
 Max.   :4.00   Max.   :40.0
```

6. Entering data from the key board

- ✓ The edit() function in R invokes a text editor that lets you enter the data manually.

```
>mydata<-data.frame(age=numeric(0),gender=character(0),weight=numeric(0))
>mydata<- edit(mydata)
```



	age	gender	weight	var4	var5	var6	var7
1	20	Male	40				
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

7. Lists

- ✓ Lists are the most complex data types. Basically it is an ordered collection of objects.
- ✓ A list allowsto gather variety of (possibly unrelated) objects under one name). It may contain combination of vectors, matrices and data frames and even other lists.

```
> g <- "myfirstlist"
> h <- c(25,26,27,18,13)
> j <- matrix(1:10, nrow=5)
> k <- c("one", "Two", "Three")
>mylist<- list(title=g, ages=h, j,k)
>mylist

$title
```

```
[1] "myfirstlist"

$ages
[1] 25 26 27 18 13

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

[[4]]
[1] "one"   "Two"   "Three"
```

3. Importing data from CSV file

Create the following CSV (Comma Separated Values) and save the file as result.csv

```
cust_id,sales_total,num_of_orders,gender
10001,800.64,3,F
10002,217.53,3,F
10003,74.58,2,M
10004,498.6,3,M
10005,723.11,4,F
10006,69.43,2,F
```

```
>result<- read.csv("C:/Users/Home/Desktop/sales.csv",header=TRUE, sep=",")

> result
  cust_id sales_total num_of_orders gender
1   10001      800.64             3      F
2   10002      217.53             3      F
3   10003       74.58             2      M
4   10004      498.60             3      M
5   10005      723.11             4      F
6   10006       69.43             2      F
```

```
> results <- read.table ("C:/Users/home/Desktop/sales.csv", header=TRUE, sep=",")
> results
  cust_id sales_total num_of_orders gender
1   10001      800.64             3      F
2   10002      217.53             3      F
```

```
3   10003       74.58            2      M
4   10004      498.60             3      M
5   10005      723.11             4      F
6   10006       69.43            2      F
> summary(results)
    cust_id       sales_total     num_of_orders   gender
 Min.   :10001   Min.   : 69.43   Min.   :2.000   F:4
 1st Qu.:10002   1st Qu.:110.32   1st Qu.:2.250   M:2
 Median :10004   Median :358.06   Median :3.000
 Mean   :10004   Mean   :397.31   Mean   :2.833
 3rd Qu.:10005   3rd Qu.:666.98   3rd Qu.:3.000
 Max.   :10006   Max.   :800.64   Max.   :4.000
```

```
>install.packages("RMySQL")
> library("RMySQL")
> conn1<-dbConnect(MySQL(),user="root",password="",host="127.0.0.1",dbname="empinfo")
   > sqlquery<-dbGetQuery(conn=conn1,statement = "select * from info”;)

>sqlquery
  rollno ename esalary
1       1  Raja   20000
2       2  Raju   10000
3       1  Raja   20000
```

4. Contingency Tables

```
> sales<- read.csv("C:/Users/Home/Desktop/sales.csv",header=TRUE, sep=",")
> sales_group[sales$sales_total<100] <- "small"
> sales_group[sales$sales_total>=100 & sales$sales_total<500] <- "medium"
> sales_group[sales$sales_total>=500] <- "big"
> spender<- factor(sales_group,levels=c("small", "medium", "big"), ordered = TRUE)
> sales <- cbind(sales,spender)
> str(sales$spender)

 Ord.factor w/ 3 levels "small"<"medium"<..: 3 2 1 2 3 1

> head(sales$spender)

[1] big    medium small  medium big    small
Levels: small < medium < big
```

```
> sales_table <- table(sales$gender,sales$spender)
> sales_table

    small medium big
  F     1      1   2
  M     1      1   0

> class(sales_table)
[1] "table"

> typeof(sales_table)
[1] "integer"

> dim(sales_table)
[1] 2 3

> summary(sales_table)
Number of cases in table: 6
Number of factors: 2
Test for independence of all factors:
        Chisq = 1.5, df = 2, p-value = 0.4724
        Chi-squared approximation may be incorrect
```

5. Descriptive Statistics

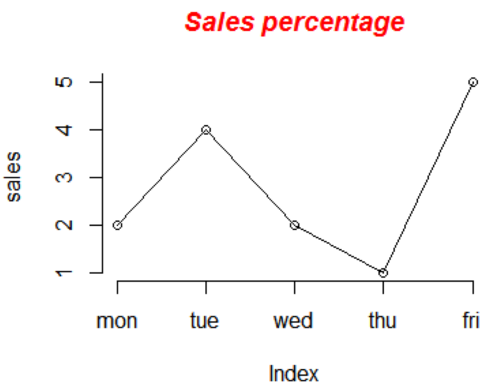
```
> summary(sales)
    cust_id       sales_total     num_of_orders   gender
 Min.   :10001   Min.   : 69.43   Min.   :2.000   F:4
 1st Qu.:10002   1st Qu.:110.32   1st Qu.:2.250   M:2
 Median :10004   Median :358.06   Median :3.000
 Mean   :10004   Mean   :397.31   Mean   :2.833
 3rd Qu.:10005   3rd Qu.:666.98   3rd Qu.:3.000
 Max.   :10006   Max.   :800.64   Max.   :4.000
   spender
 small :2
 medium:2

> x <- sales$sales_total
> y <- sales$num_of_orders
> cor(x,y)
[1] 0.8020646

> cov(x,y)
[1] 195.283
```

```
> IQR(x)
[1] 556.665

> mean(x)
[1] 397.315

> median(x)
[1] 358.065

> range (x )
[1]  69.43 800.64

> sd(x)
[1] 323.4382
> var(x)
[1] 104612.2
```

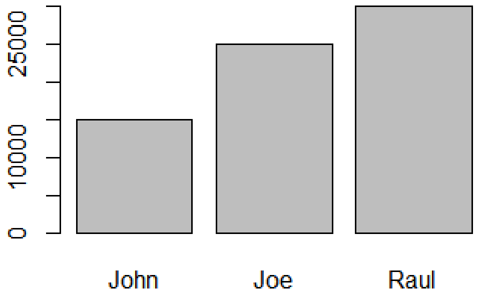
6.Graph visualization

Summary Plots

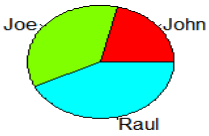
```
> sales<-c(2,4,2,1,5)
> plot(sales,type="o",color="blue",axes=FALSE)
> axis(1,at=1:5,lab=c("mon","tue","wed","thu","fri"))
> axis(2,at=1:10)
> title(main="Sales percentage",col.main="red",font.main=4)
```



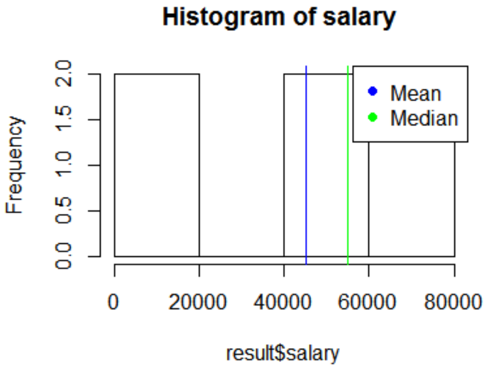
```
result<- read.csv("C:/Users/Home/Desktop/emp.csv",header=TRUE, sep=",")
barplot (result$salary,names.arg=result$name)
```



```
result<- read.csv("C:/Users/Home/Desktop/emp.csv",header=TRUE, sep=",")
pie(result$salary,labels=result$name,col=rainbow(length(result)))
```



```
> result<- read.csv("C:/Users/Home/Desktop/emp.csv",header=TRUE, sep=",")
> hist ( result$salary , main ="Histogram of salary ")
> result<- read.csv("C:/Users/Home/Desktop/emp.csv",header=TRUE, sep=",")
> hist ( result$salary , main ="Histogram of salary ")
> abline (v= mean ( result$salary), col = " blue ")
> abline (v= median (result$salary), col = " green ")
> legend ("topright", c("Mean", "Median"), pch = 16,col = c("blue", " green"))
```


Histogram of salary
Frequency
result$salary
● Mean
● Median