Data structure

Qiang Shen

Sept.22,2016

Contents

math

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation
x%%y	Modulus (x mod y) 5%%2 is 1
x%/%y	Integer division 5%/%2 is 2

Figure 1:

basics of R

- variable and data structure

basics of R

- variable and data structure
- import and output of R

data structure

- variable and observations (statistics)
- records and fields (database)
- examples and attributes (machine learning)

ID	记录时间	级别	产品类别	销量（件）	是否参加双11
1	2014/11/06	红星卖家	享受型	5128	否
2	2014/10/09	钻石卖家	享受型	8928	是
3	2014/10/18	皇冠卖家	实用型	1024	是
4	2014/10/15	钻石卖家	享受型	8235	是
5	2013/10/22	皇冠卖家	实用型	2356	否
6	2014/10/28	红星卖家	实用型	45667	是

-

variable

- dynamic language vs. static language (declare)

```
x<-2
y=3
###
a=5
b<-a
a="k"
a;b #what is a, b?
assign('s',7)
```

variable

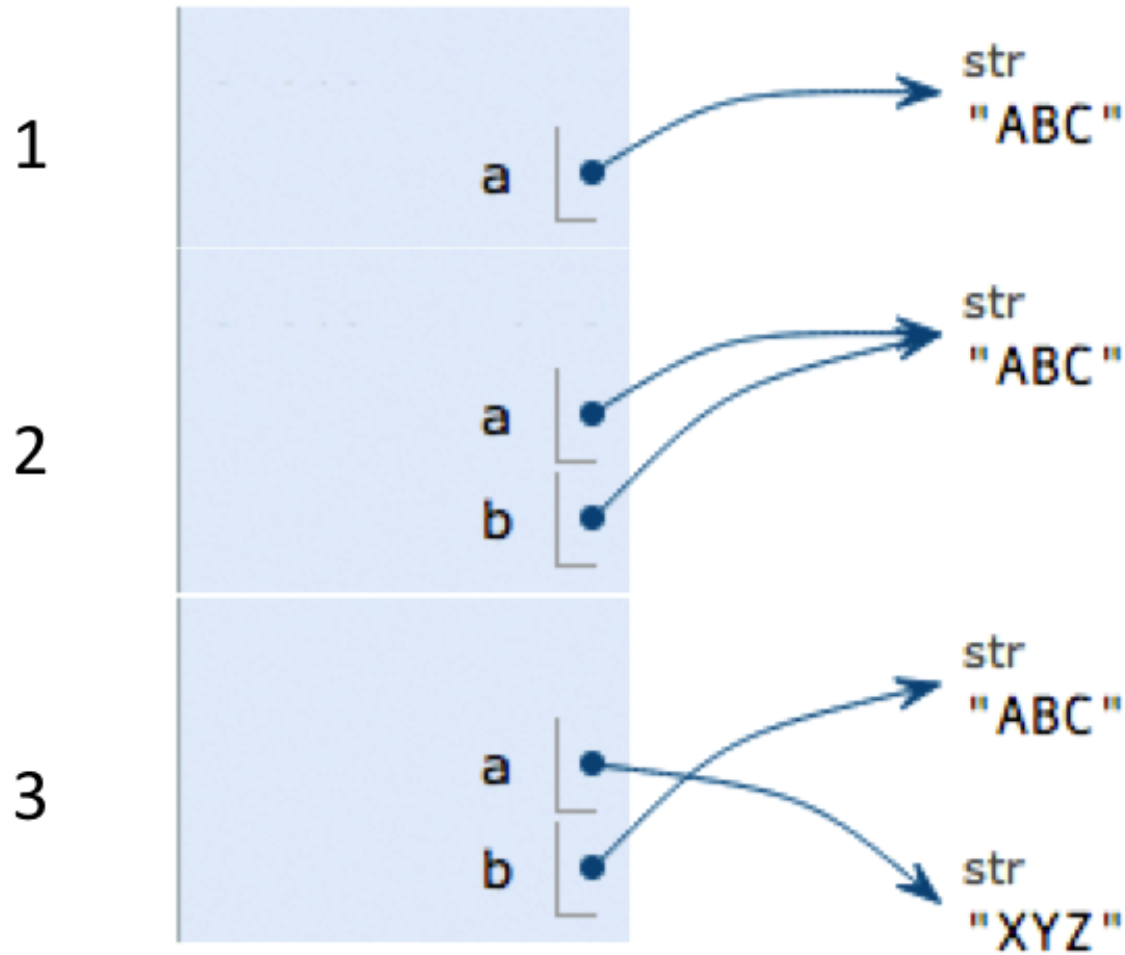


Figure 2: assignment

variable name

- case senstive (vs. VB,SQL)
- number,dot,underscore and “c”
- the arts of name variables
-

Reserved words in R

if	else	repeat	while	function
for	in	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

variable

- remove of variable

data: core concept in R.

- scales of measurement (statistics)
- atomic classes (R as a programming language)
- data structure (R as a statistical software:container)

scales of measurement in statistics.

- Categorical: nominal ordinal
- Quantitative: interval (+/-) ratio (+/-/*//)

ID	记录时间	级别	产品类别	销量（件）	是否参加双11
1	2014/11/06	红星卖家	享受型	5128	否
2	2014/10/09	钻石卖家	享受型	8928	是
3	2014/10/18	皇冠卖家	实用型	1024	是
4	2014/10/15	钻石卖家	享受型	8235	是
5	2013/10/22	皇冠卖家	实用型	2356	否
6	2014/10/28	红星卖家	实用型	45667	是

-

basic('atomic') classes of objects

- character: between " "or' '
- numeric (real numbers)
- integer
- complex
- logical: T(TRUE) or F(FALSE)

basic('atomic') classes of objects

- character: between " "or' '
- numeric (real numbers)
- ~~integer~~
- ~~complex~~
- logical: T(TRUE) or F(FALSE)

data structure

- numeric:float/double
- integer
- complex

```
a<-53;b<-5.3;c=5.0
class(c);print(a)
1/0 #Inf
0/0 #NAN
i<-5L;class(i)
# j<-5+1i;class(j)
```

attributes

R objects have attributes, which are like metadata for the object.

- names,dimnames
- dimensions(e.g. matrices,arrays)
- class(e.g. integer, numeric)
- length
- user-defined attributes

```
attributes(mtcars)
```

Character and factor

```r
x<-'data' #or with ""
class(x)
```

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

```r
factor('data')
```

```
## [1] data
## Levels: data
```

```r
nchar(x)
```

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

logical type

```
TRUE##T,True
FALSE##F,False
is.logical(FALSE)
TRUE*5
FALSE*5
a<-2;b<-3
a!=b   ###if contional statement
"data"=="sprite"
"data"=="angel"
```

Revisit: basic('atomic') classes of objects

- character: between " "or' '
- numeric (real numbers)
- ~~integer~~
- ~~complex~~
- logical: T(TRUE) or F(FALSE)

data structure

5 common structure

- vector
- matrix
- arrary
- data frame
- list

data structure

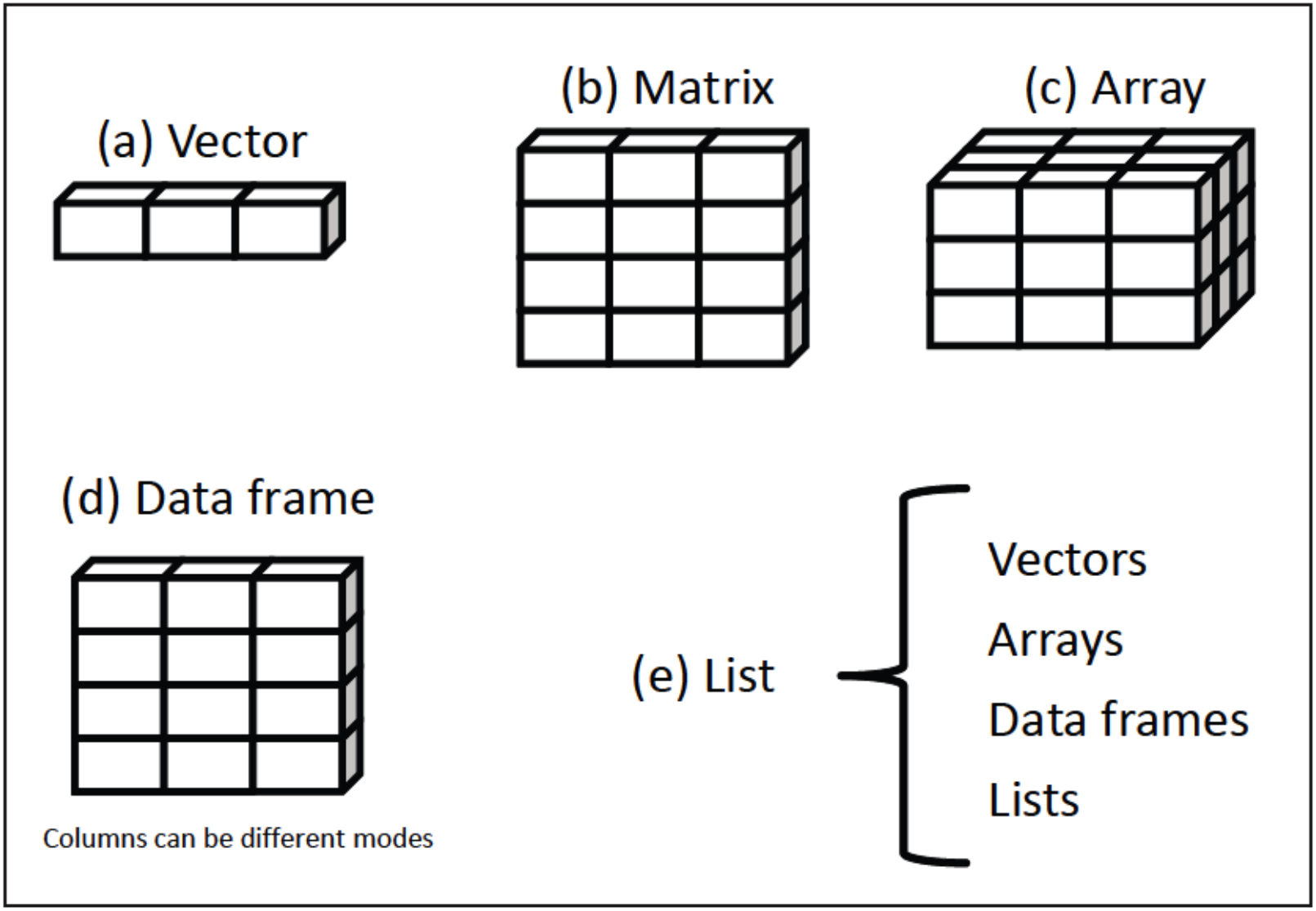


Figure 3: plot

data strucutre in R

- heterogenous or not
- dimension

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Figure 4:

Vector

- the **core data structure** in R.
- container of data objects, but **only for one class** of data
- **vectorization**

```
a<-c(1,2,3) ###no row/column
is.vector(a)
```

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

Vector

numeric type as an example

```
c(1,2,3,4)#concatenate
c(1,3,6,7,7)
c(1:4) ##colon
seq(1,100,by=2)
 #calculation
 c(1,2,3,4)+c(2,4,5,6)
 c(1,2,3,4)-c(2,4,5,6)
 c(1,2,3,4)*c(2,4,5,6)
 c(1,2,3,4)/c(2,4,5,6)
 c(1,2,3,4)+1  #vectorization
 c(1,2,3,4)+c(2,4)   #recycle
 c(1,2,3,4) %*% c(2,4,5,6) ##row times column
```

String vector

```
c('1','2','3','4')
c('bj','sh','gz','sz') ##first tier cities
temp<-c('bj','sh','gz','shenzheng')
nchar(temp)  #vectorization
```

logical vector

```
c(T,F,T,T)
age<-c(23,21,20,24,18,15,25) ##legal age of marriage
age>20
price<-c(3,6,5,7,4,11,14)
price>5
# age[age>20] ## []
```

Explicit and implicit data coercion

```
#explicit
c('1','2','3','4')
as.numeric(c('1','2','3','4')) ##add 'h'
as.character(c(3,5,6,7,8,11,14))
a<-c('x','y','z')
as.logical(a)



##implicit: represent all objects  in a reasonable fashion.
a<-c(1.2,'a')
a<-c(TRUE,2)
a<-c(FALSE,'a')
```

vector manipulation

```
age<-c(3,51,6,74,58,21,14)
age[2] ###[] vs. ()
age[2:3]
age[-1]
age[age>20]

## age vs. newage
newage<-age[-3]
newage

city<-c('bj','sh','canton','sz')
nchar(city)

class(age);length(age);max(age)
names(age)<-c('a','b','c','d','e','f','g')
age
names(age)<-letters[1:7]
age #names(temp)<-NULL
```

matrix

- two dimensions

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
vector<-c(1:12)
matrix(vector,nrow = 3,ncol = 4)
matrix(vector,nrow = 3,ncol = 4,byrow = T)
mt<-matrix(vector,nrow = 3,ncol = 4,T)
dimnames(mt) = list(c('gain1','gain2','gain3'),
            c('loss1','loss2','loss3','loss4')) #
mt
attributes(mt)
rownames(mt)
colnames(mt)
v<-1:12
dim(v)<-c(3,4)
```

matrix

```
mt
mt[2,3]
mt[2,]##colon in matlab
mt[,3]
mt[,'loss2']
```

magic square

$$\textbf{MagicMatrix} = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

```
my_mat<-matrix(c(8,3,4,1,5,9,6,7,2),ncol=3)
my_mat[1,1]+my_mat[1,2]+my_mat[1,3]
sum(my_mat[1,])
rowSums(my_mat)
colSums(my_mat)
sum(diag(my_mat))
```

arrary

```
dim1<-c("A1","A2")
dim2<-c("B1","B2","B3","B4")
dim3<-c("C1","C2","C3")
a<-array(c(1:24),c(2,4,3),
         dimnames=list(dim1,dim2,dim3))
a
a[1,2,3]
```

data frame

- data frame
- combination of vectors and factors.
- attributes: nrow;ncol;dim

```
city<-c('bj','sh','gz','sz')
age<-c(22,34,34,24)
sex<-c('M','F','M','F')
people<-data.frame(city,age,sex)
people[2,3];people[,3]
people$age
people$age>30
people[people$age>30,]
```

data frame

```
city<-c('bj','sh','gz','sz')
age<-c(22,34,34,24)
sex<-c('M','F','M','F')
people<-data.frame(city,age,sex)
dim(iris)
ncol(iris);nrow(iris)
rownames(people)<-letters[1:4]
rownames(people)<-c(1,4,5,8)
rownames(people)<-NULL ##set back to generic index
colnames(people)[3]
```

data frame

```
head(iris);tail(iris)
str(iris)
iris[c(50:100),c(1:2),]
iris[,c(1:2)]
iris[,c('Sepal.Length','Sepal.Width')]

iris$Sepal.Width
iris[,'Sepal.Length']
class(iris[,'Sepal.Length'])###reduced
class(iris['Sepal.Length'])
class(iris[,'Sepal.Length',drop=F])
class(iris[['Sepal.Length']])###reduced

###comparison with stata
# plot(Sepal.Length,Petal.Length)
plot(iris$Sepal.Length,iris$Petal.Length)
```

data frame

```
attach(iris)
plot(Sepal.Length,Petal.Length)
detach(iris)

with(iris,plot(Sepal.Length,Petal.Length))
```

list

- list

```
mylist<-list(city=city,age=age,age=sex)
str(mylist)
mylist[1]
mylist$city
mylist[[1]]
```

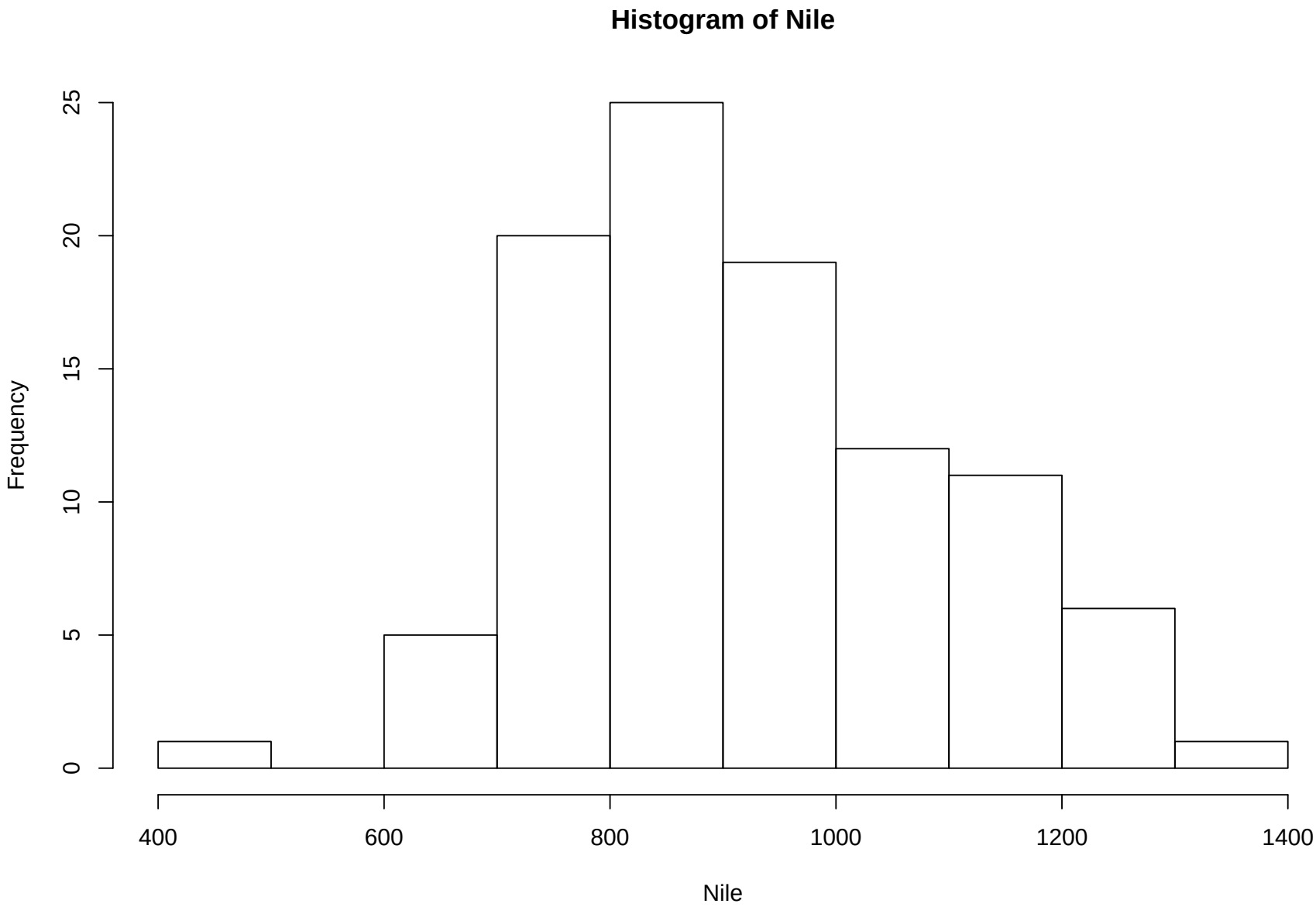
The importance of list

- 1.any combinations

```
g<-"my list"
h<-c(25,26,18,39)
j<-matrix(1:10,nrow=5)
k<-c("one","five","eight")
mylist<-list(titile=g,age=h,j,k)
mylist
```

The importance of list

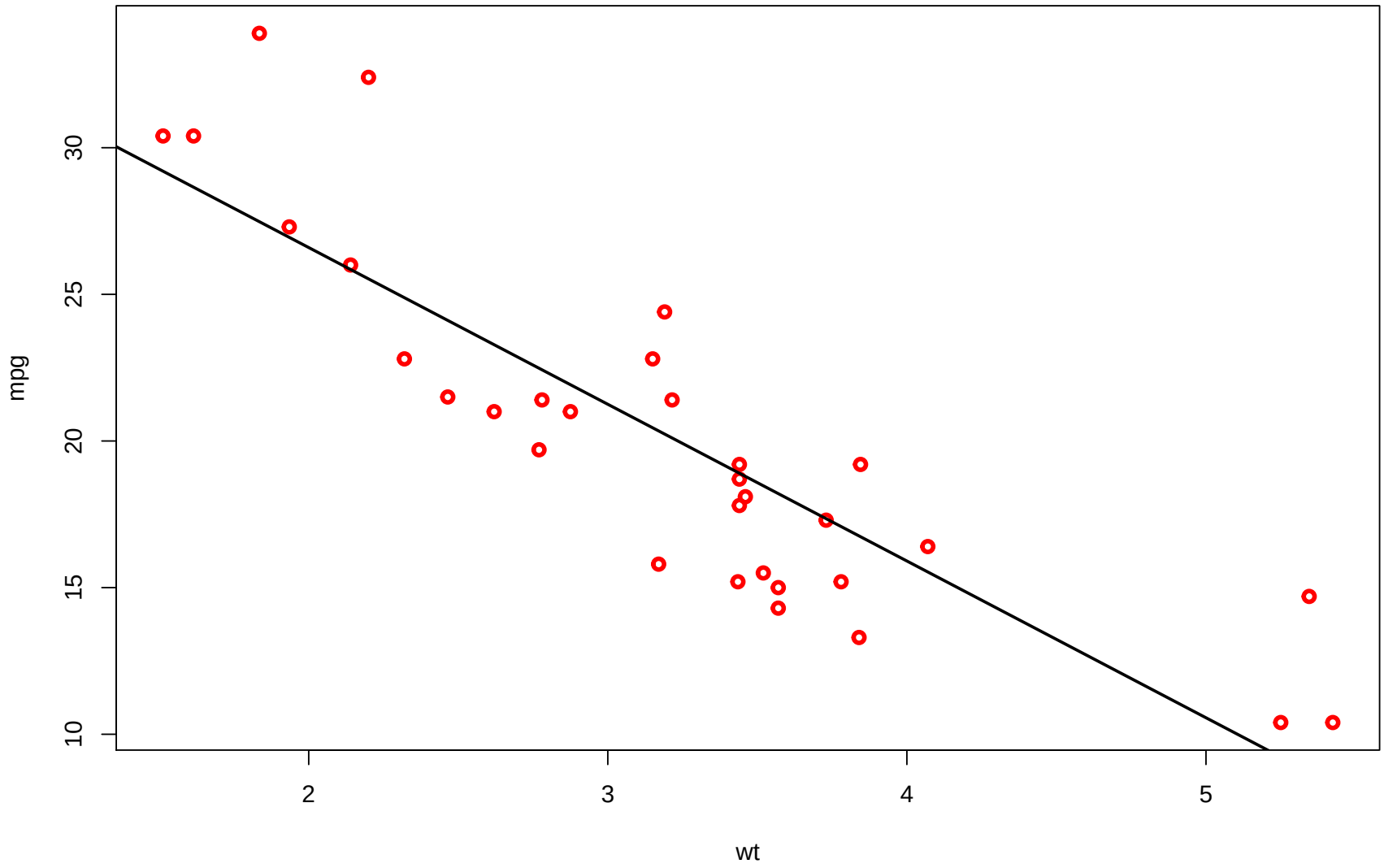
- 2. many output/return of R function is list




```
## $breaks
```

The importance of list

- 3. Using output as input.




```
##
## Call:
```

data mode

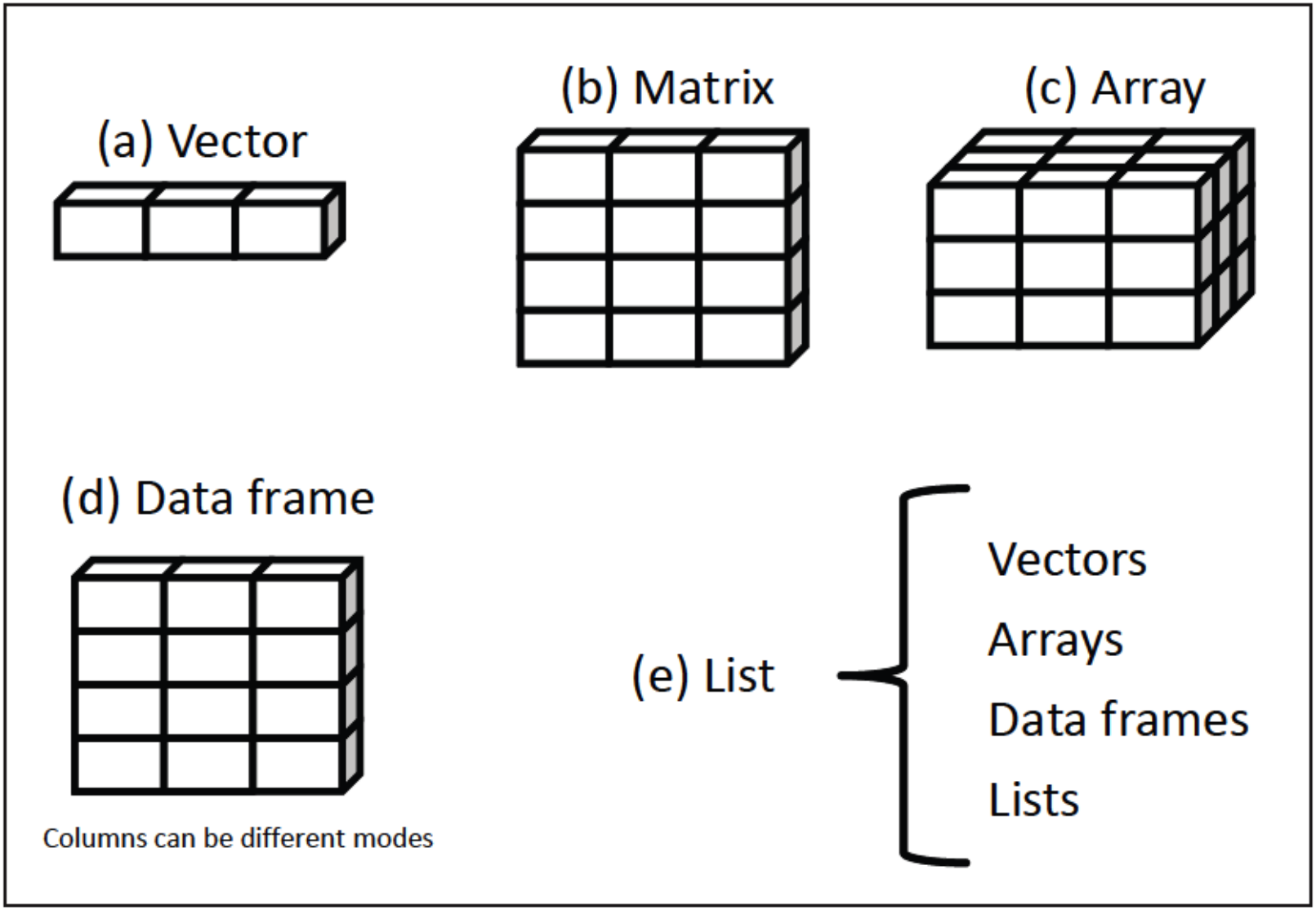


Figure 5: plot

Factor

- scales of measurement: **nominal** and **ordinal** data.
- **common** and **ordered** factor

```
http://taobaoshopping.org/how-to-view-taobao-seller-rating/
##nominal
sex<-c('male','female','male','female','male')
sex_new<-factor(sex)#levels=c('male','female')
str(sex_new)
##ordinal
rating<-c('heart','crown','diamond','heart','diamond')
factor(rating,ordered=T,levels=c('heart','diamond',
                                 'crown'))
```

labels for factors

```
patientID <- c(1, 2, 3, 4)
age <- c(25, 34, 28, 52)
gender<-c(1,1,2,1)
diabetes <- c("Type1", "Type2", "Type1", "Type1")
status <- c("Poor", "Improved", "Excellent", "Poor")
diabetes <- factor(diabetes)
gender <- factor(gender)
status <- factor(status, order = TRUE,levels=c('Poor','Impr
                                                'Excellent'))
patientdata <- data.frame(patientID,gender,age,diabetes,
status)
patientdata
patientdata$gender<-with(patientdata,factor(gender,
          levels=c(1,2),labels=c('male','female')))
patientdata
str(patientdata)
summary(patientdata)
```

missing: NA vs. NULL vs. NaN

NULL means that there is no value,while NA and NaN mean that there is some value,although one that is perhaps not usable.

- NaN not-a-number
- NA not available

example 1

```
v1 <-  c(1, NA, NULL,NaN)
v1
```

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

```
v2 <-  c("1", NA, NULL,NaN)
v2  ##coercion
```

```
## [1] "1"   NA    "NaN"
```

```
l <-  list("1", NA, NULL,NaN)
length(l[4])
```

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

NULL: values removal for data analysis

Assigning NULL to list items, removes them.

```
head(iris)[1:3,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Spec
## 1          5.1         3.5          1.4         0.2  set
## 2          4.9         3.0          1.4         0.2  set
## 3          4.7         3.2          1.3         0.2  set
```

```
iris_L<-sapply(iris,list)
length(iris_L)
```

```
## [1] 5
```

```
iris_L$Sepal_Width<-NULL
length(iris_L)
```

```
## [1] 5
```

NA: caveat 1

- na.rm

```
vy <- c(1, 2, 3, NA, 5)
mean(vy)
```

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

```
mean(vy,na.rm = T)
```

```
## [1] 2.75
```

NA: caveat 2

- stata: gen x=var1==var2/ gen x=1 if var1==var2
- R: NA==NA—>NA

```
a<-c(1,2,NA,NA)
b<-c(1,3,4,NA)
a==b
```

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

```
vy <- c(1, 2, 3, NA, 5)
vy[!is.na(vy)]
```

```
## [1] 1 2 3 5
```

common errors:

- lower case and upper case difference.
- forget (): eg. help(lm)
- confound [] with ()
- forget “”
- forget “c”" for vector generation.
- use “" in Windows OS for directory
- myvar==3 or myvar==“Jack” but not myvar==NA, should be.is.na()
- forget to load packages needed.