# Fundamentals-02 Compound Data Structures

Created by H. M. Samadhi Chathuranga Rathnayake

```r
#Compound data structures
#Vectors
#Creating vectors with c function
a=c(12,22,23,34,35)
a

class(a)

b=c("Dog","Cat","Rat")
b

class(b)

x=c(12,22,23,"Man","Car")
x  #This is called the coercion

#logical < integer < numeric < complex < character (This is the order)
class(x)

y=c(TRUE,FALSE,12)
y

class(y)

#Other ways of creating vectors
d=1:10
d

p=15:5
p

k=seq(from=10,to=50,by=5)
k

h=seq(from=10,to=50,length.out = 20)
h

f=seq_len(20)
f

h=replicate(10,c("Dog","Cow"))
h

#Indexing & slicing vector elements
a=c(23,33,34,32,45,50,65)
```

```r
a[1]

a[c(1,3,6)]

a[-c(1,3,6)] #All other elements except these indexes

a[1:3]

a[c(T,T,T,F,T,T,F)] #Boolean masking

a[a>40]

a[a%%2==0]

#Element wise operations in numerical vectors
a=c(2,3,4,6,3)
b=c(3,4,1,6,4)

a+10

a*2

a/3

a^2

a>3

a+b

a*b

a/b

a^b

a>b

a%in%b #This is the membership

#Element wise operations with recursive property in vectors
p=c(10,20,30,40,50)
q=c(100,200,300)

p+q

#Vector properties with functions
a=c(2,3,5,3,6,7,4,5,8,1,3,4,2)

#Basic summary functions
length(a)

summary(a)
```

```r
str(a)

min(a)

max(a)

sum(a)

mean(a)

median(a)

var(a)

sd(a)

range(a)

#Cumulative functions
cumsum(a)

cumprod(a)

cummin(a)

cummax(a)

#Vector operations with functions
a=c(20,30,40)
b=append(a,100)
b

d=append(x = a,values = 200,after = 2)
d

k=append(x = a,values = c(100,200,300),after = 2)
k

h=c(a,100,200,300) #This another way of appending
h

a=c(20,30,40)
rep(a,3)

b=c(10,20,25,30,35,40)
all(b>20)

all(b<50)

any(b>20)

any(b>50)

b=c(100,50,25,30,15,40,15,40)
sort(b)
```

```r
order(b)

rev(b)

unique(b)

h=c(12,22,23,34,45,43,32,33,21,23,45,56,67,56,70)
s=sample(h,5)
s

set.seed(1000)
h=c(12,22,23,34,45,43,32,33,21,23,45,56,67,56,70)
s=sample(h,5)
s

p=sample(h,5,replace = T)
p

f=c(12,22,23)
q=sample(f,10,replace = T)
q

k=c(20,12,25,30,32,22,33)
which(k>=25)

which.min(k)

which.max(k)

a=c(12,22,23,34,45,56,43,43,33,34)
idx=which(a%in%c(22,23,34))
b=a[-idx]
b

g=c("Male","Female","Female","Female","Male")
table(g)

g=c("Male","Female","Female","Female","Male")
replace(x = g,list = c(1,5),values = "M")

#Matrices
#Matrix creation
a=matrix(c(12,23,32,33,45,43),3,2)
a

b=matrix(c(12,23,32,33,45,43),3,2,byrow = TRUE)
b

c=array(c(12,23,32,33,45,43),c(3,2))
c
```

```r
a=matrix(10,3,2)
a

#Matrix properties
b=matrix(c(12,23,32,33,45,43),3,2,byrow = TRUE)
b

dim(b)

nrow(b)

ncol(b)

str(b)

summary(b)

#Merging matrices
a=matrix(c(12,23,32,33),2,2)
b=matrix(c(22,26,37,43),2,2)

c=cbind(a,b)
c

d=rbind(a,b)
d

#Indexing & slicing matrices
a=matrix(c(12,23,32,33,45,43,34,55,56),3,3)
a

a[1,3]

a[1,]

a[,2]

a[c(1,3),2]

a[,2:3]

a>40

a[a>40]

#Matrix operations
a=matrix(c(12,23,32,33),2,2)
b=matrix(c(22,26,37,43),2,2)
c=matrix(c(12,23,32,33,45,43,34,55,56),3,3)

a

b
```

```r
a+b

a-b

a*b

a/b

a%*%b

c

t(c)

det(c)

solve(c)

diag(c)

#Lists
#Creating a list
a=c(12,22,23,34)
b=c("Cat","Dog")
c=100
d="Man"
m=matrix(c(12,22,23,34,45,32),3,2)


L=list(a,b,c,d,m)
L

#A name can be given for each segment of the list
names(L)=c("Part1","Part2","Part3","Part4","Part5")
L

#Accessing items and elements in items inside the list
L[[1]] #Accessing the first items

L$Part1

L[[1]][2] #Accessing elements inside the items

#Adding elements to a list
L$Part6="Cow"
L

#Removing elements from a list
L$Part3=NULL
L
```

```r
#Convert a list into a vector
v1=unlist(L)
v1



#Merge lists
L1=list("Man",c(12,22,23),TRUE)
L2=list(matrix(1:20,4,5),c("Cat","Dog"))

L1

L2

L3=c(L1,L2)
L3

#Splitting a list
a=c(12,22,23,34)
b=c("Cat","Dog")
c=100
d="Man"
m=matrix(c(12,22,23,34,45,32),3,2)



L=list(a,b,c,d,m)
L

L1=L[1:2]
L2=L[3:5]
L1

L2

#Factors
#Un Ordered factors
g=c("Male","Female","Male","Male","Male","Female")
g

fg=factor(g)
fg

#Ordered factors
h=c("first","first","fifth","fourth","second","fifth","third","second","fourt
h")
h

fh=factor(h)
fh

ofh=factor(h,levels = c("first","second","third","fourth","fifth"))
ofh
```

```r
#Data Frames
name=c("Kane","Jane","David","Harry","Larry","Mary","John","Jessy","Anne","Li
lly","Julia","Pale")
age=c(23,33,34,32,21,22,23,34,32,18,21,23)
marks=c(89,78,88,59,67,78,88,90,59,75,77,69)

df=data.frame(name,age,marks)
df

#Accessing columns in a data frame
df["name"] #This shows a sub data frame

df["age"]

df$name #This is giving the output as a vector

attach(df) #This will make these columns as global variables

name

age

marks

detach(df) #Make them again local variables

#Selecting several columns
df[c("name","age")]

#Accessing the elements in a data frame
df

df[1,1] #First column first element

df[1,"age"] #First row age value

df[c(1,5),"age"] #First and fifth rows age value

df[2,] #All values in the 2nd row

df[,3] #All values in the 3rd column

df[,-3] #All values except 3rd column

df[2:6,2] #2nd to 6th row of 2nd column

df$name[1:5] #1st to 5th name in name column

#Change elements in a data frame
df

df[1,1]="Sammie"
df
```

```r
df[2:6,2]=c(50,50,50,50,50)
df

#Adding a new column
df$class=c("C1","C2","C3","C4","C5","C6","C1","C2","C3","C4","C5","C6")
df

#Removing an existing column
df$age=NULL
df

#Basic column operations
df$marks_new=df$marks+5
df

df$marks_diff=df$marks_new-df$marks
df

#Checking conditions
df$marks>=70 #Returns TRUE for marks greater than 70

#Boolean masking for data frames
df_new=df[df$marks>=70,]
df_new

#Data frame functions
View(df) #View the data frame

head(df) #Top elements

head(x = df,n = 8)

tail(df) #Bottom elements

tail(x = df,n = 8)

dim(df) #Dimensions

nrow(df)

ncol(df)

#Row and column names
row.names(df)

colnames(df)

colnames(df)=c("Student_Name","Previous_Marks","Class","New_Marks","Marks_Dif
ference")
df

#Column and row sums
colSums(df[c("Previous_Marks","New_Marks")])
```

```r
rowSums(df[c("Previous_Marks","New_Marks")])

#Column and row means
colMeans(df[c("Previous_Marks","New_Marks")])

rowMeans(df[c("Previous_Marks","New_Marks")])

#edit function can be used for editing a data frame manually
df1=edit(df)
df1

#str function will give all the data types in the data frame
str(df)

#summary function will give a summary about the data frame
summary(df)

#which function can be used for identifying the indexes of some criteria
which(df["Previous_Marks"]>=70)

#table function will give frequencies in a categorical column
table(df$Class)

#Factorize the categorical columns
df=data.frame(Name=c("Sam","Kane","Jane"),Gender=c("M","M","F"),Age=c(23,32,2
1),University_Year=c(2,3,1))
df

str(df)

summary(df)

df$Gender=factor(df$Gender)
df$University_Year=factor(df$University_Year,levels = c(1,2,3))

str(df)

summary(df)

#Working with external data sets
#CSV files
#Importing CSV files
data=read.csv("D:\\Workshops\\R Programming for Data Science Workshop\\Part
01 - Fundamentals of R Programming\\Datasets\\default.CSV")

getwd()

setwd("D:\\Workshops\\R Programming for Data Science Workshop\\Part 01 -
Fundamentals of R Programming\\Datasets")
data=read.csv("default.CSV")
data
```

```r
#Now we can treat this as a data frame
head(data)

dim(data)

colnames(data)

str(data)

table(data$Loan.Offered)

table(data$Own.house)

data$Gender=factor(data$Gender)
data$Loan.Offered=factor(data$Loan.Offered)
data$Job=factor(data$Job)
data$Status=factor(data$Status)
data$Credit.History=factor(data$Credit.History)
data$Own.house=factor(data$Own.house)
data$Purpose=factor(data$Purpose)

str(data)

head(data)

summary(data)

#We can perform any data frame operation
data_male=data[data$Gender=="Male",]
data_female=data[data$Gender=="Female",]
data_female=data[!data$Gender=="Male",]

summary(data_female)

data_female$CS_Ex_Ratio=data_female$Credit.Score/data_female$Work.Exp
head(data_female)

summary(data_male)

data_male$Exp_Level="Low"
head(data_male)

data_male[data_male$Work.Exp>=15,]$Exp_Level="High"
head(data_male)

#Data simulation
datanorm=rnorm(100) #Standard normal distribution
datanorm

datanorm2=rnorm(n = 100,mean = 20,sd = 5) #Standard normal distribution
datanorm2
```

```r
datauni=runif(n = 100,min = 10,max = 20) #Uniform distribution
datauni

datapois=rpois(n = 100,lambda = 5)
datapois

databin=rbinom(n = 100,size = 1, prob = 0.5)
databin

#rnbinom(),rgamma(),rhyper(),rbeta()
```