5. Graphs and charts

Bar plots in R

- simple Bar plot
- Horizontal Bar plot
- stacked Bar plot
- Grouped Bar plot

R Bar plot

* Created by uing → barplot () funtion

* Input can be vector /matrix

* if we supply a vector the plot will have bars with their height equal to the elements in the vector

eg: temp = c(27, 26, 23, 24, 30) (R.studio)
barplot (temp)

Argument used

* main → used to give heading

xlab → x - axis name

y lab → y - axis name

col → Give colour to bar

* x lim
* y lim

```
eg: temp = c(
      barplot (temp,
            main = "max Temp in a week",
            xlab = Degree celcius",
            ylab = Day
             col = blue
```

* density → Give line inside bar

* border → border to bars

density = 20, border = "red", col = green")

* width → size of bars

* space → space b/w bars

* x <- c(1, 1, 2, 2, 2, 3, 3, 1, 1, 2, 2, 3, 4, 4, 4]

table (x)

```
x
1  2  3  4
4  5  3  3
```

plotting of categorical data

* x ← c(1, 1, 2, 2, 2, 3, 3, 1, 1, 2, 2, 3, 4, 4, 4)

y = table (x)

bar plot (height = y , width = c(3, 4, 5, 6)

* x ← c(1, 1, 2, 2, 2, 3, 3, 1, 1, 2, 2, 3, 4, 4, 4)

y = table (x)

bar plot (height = y , space = 5)

* y x ← c(1, 1, 2, 2, 3, 3, 1, 1, 2, 3, 3, 4, 4, 5)

y = table(x)

barplot (height = y, names.arg = LETTER [1:4])

barplot (height = y, names.arg = c("student 1", "student 2" "student 3", "student 4"))

* x = c(1, 1, 1, 1, 2, 1, 2, 2, 2, 3, 3, 3, 1, 1. 2, 2, 3, 3)

y = table (x)

barplot (height = y . names . arg = c ("student1", "student2"
"student3"), legend . text = T)

legend . text → is a vector of text used to construct a legend for the plot ie, used to identify what each bar represent

* x = c(1, 1, 1, 1, 2, 1, 2, 2, 2, 3, 3, 3, 1, 1, 2, 2, 3, 3)

y = table (x)

barplot (height = y, los = 1)

bar plot (height = y, las = 2)

Stacked Bar plot

The plot drawn when Matrix is given as input

* > data ("mtcars")

> names (mtcar)

[1] "mpg' "cyl" "disp" "hp" "drat" "wt" "qsec"
"vs" "am" "gear" "carb"

> ~~matrix~~ mtcars

> mtcars $cyl

[1] . . .

> table (mtcars $ cyl)

4	6	8
11	7	14

> table (mtcars $ gear)

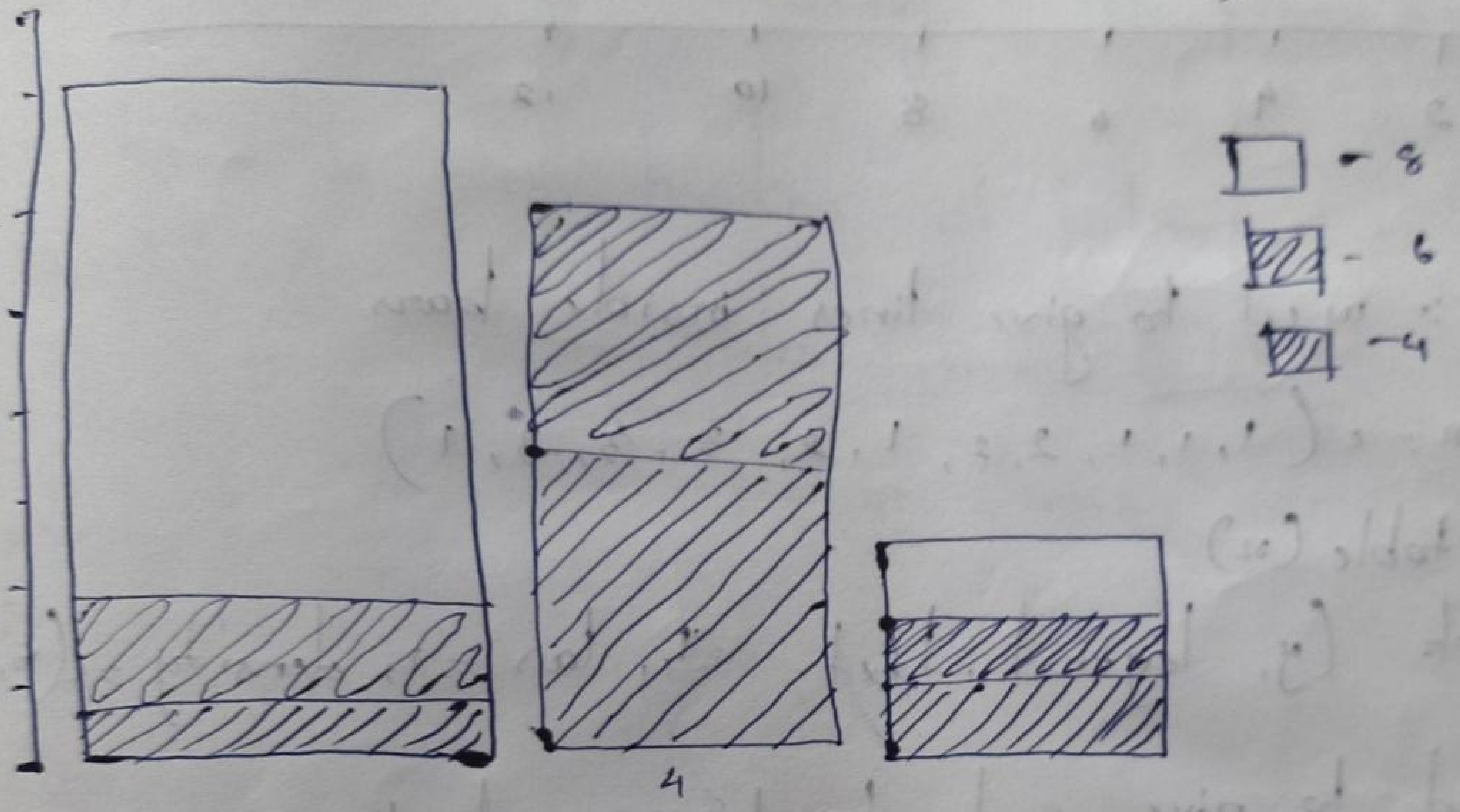
3	4	5
15	12	5

y ; mt carls $gear)

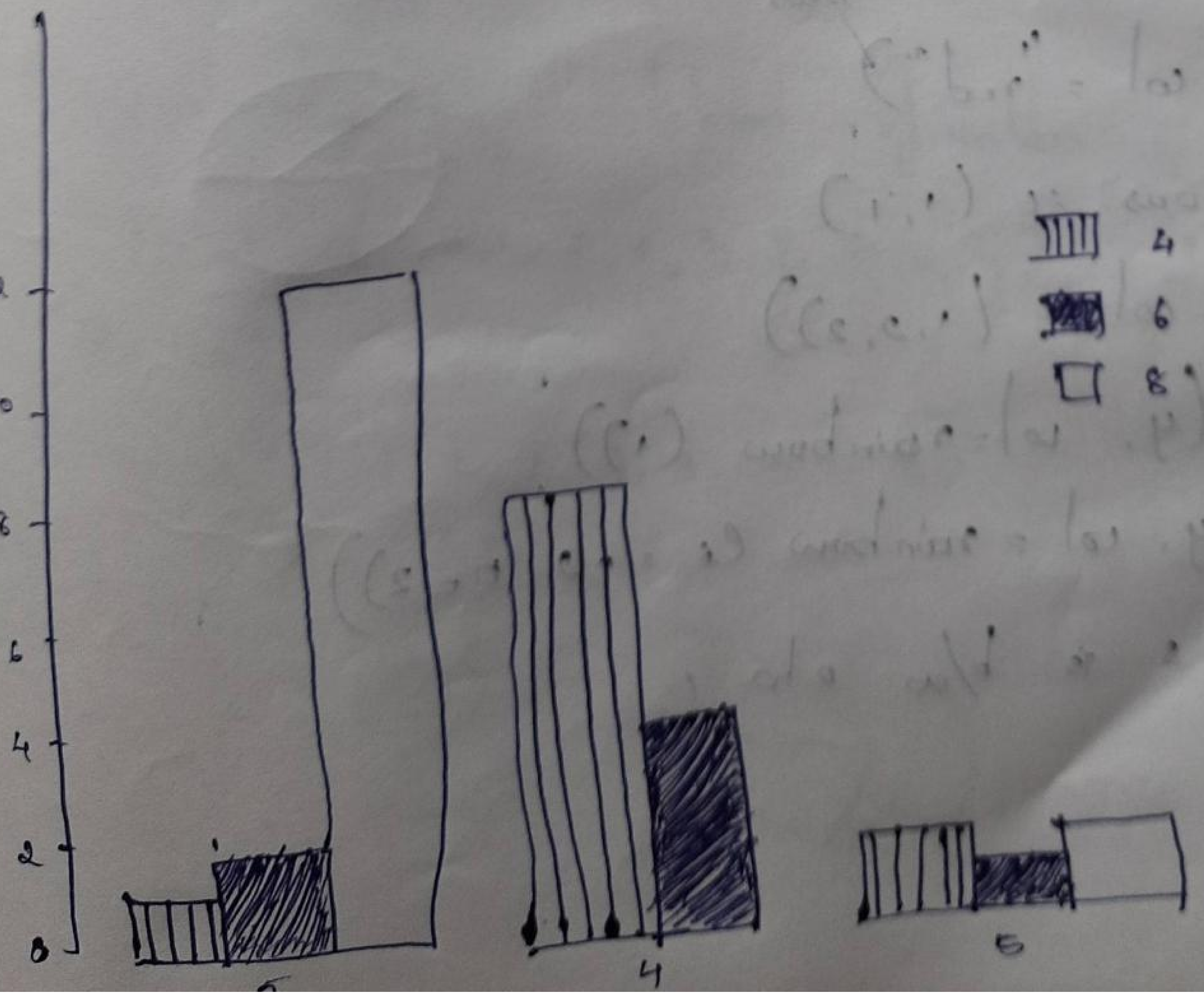
mt cars (y)

	3	4	5
4	1	8	2
6	2	4	1
8	12	0	2

```
> y, = table (mt cars $(y), mt cyl , m t carts $gear)
> barplot (y,) , legenl . text =T)
```

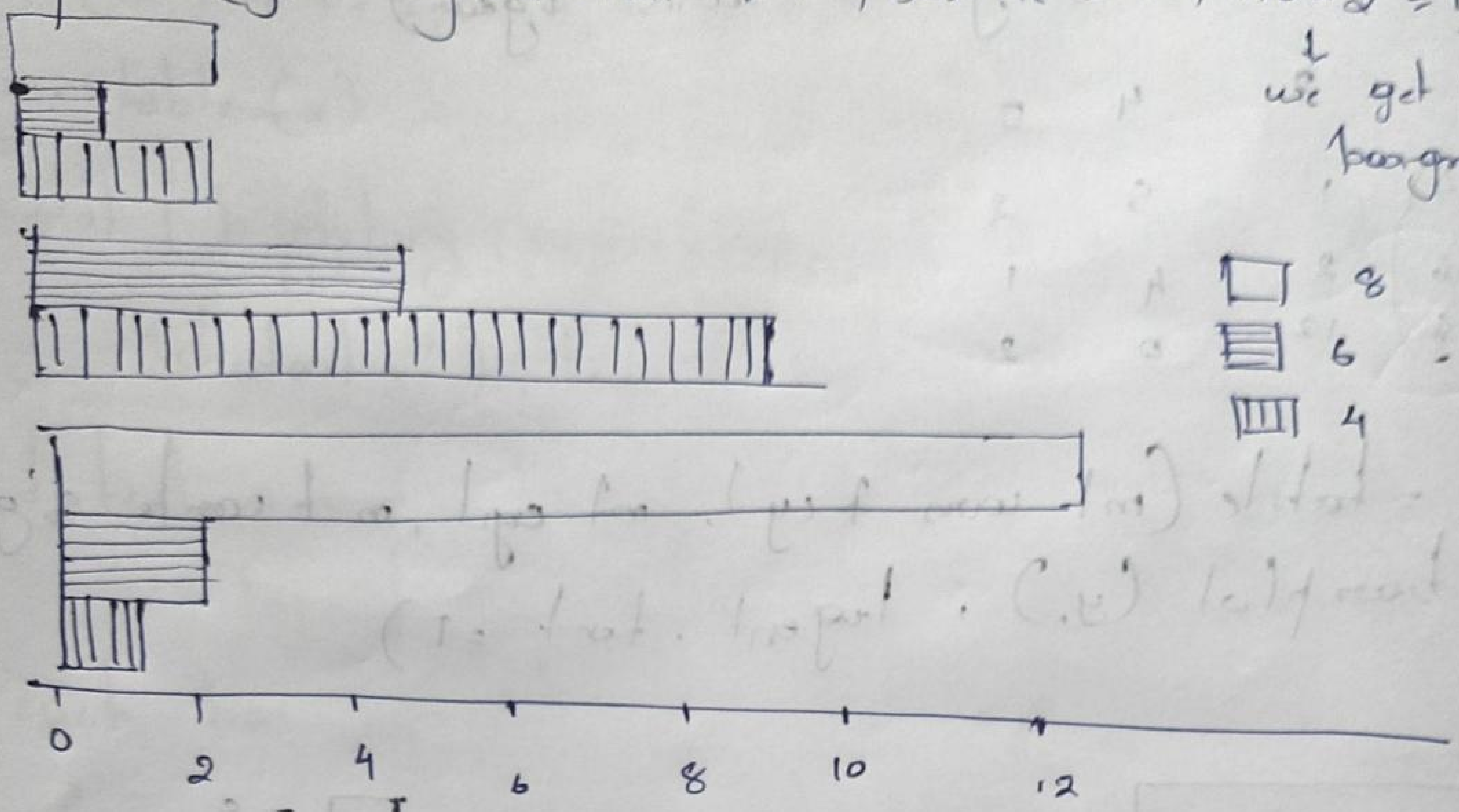


```
> bar plot (y 1 , legend . text = T , beside =T)
```



> barplot (y, legend.text = T, beside = T, horiz = T)

↓
we get horizontal bargraph



* density :- used to give lines inside bars

eg :- x = c(1, 1, 1, 2, 2, 1, 2, 3, 3, 3, 1)

y = table (x)

barplot (y, legend.text = T, las = 1, density = c(5,10,15)

* Angle used to give angle to lines inside bar

* colour :- Give colour to bar

*> bar plot (y, col = "red")

* > bar (m) rows = c (1, 1)

> bar plot (y. col = c (1, 2, 2))

* > bar plot (y, col = rainbow (1))

* > bar plot (y. col = rainbow (s = .2 n = 2))

3 is b/co o to 1

border ÷ used to set border to bar

```
> barplot :(y. col = rainbow(s = .5, n = 15), border = T)
> bar (mf rows = c(1,2))
> bar plot (y. col = rainbow (s = .5, n = 13), border = F)
> barplot (y. col = rainbow (s = .5, n = 13) border = T)
> bar (mf rows = c(1, 1))
```

* main ÷ used to give heading to the particular bar plot

sub ÷ used to give heading at botem

eg : barplot (y, main = "header", sub = "footer")

- barplot (y, main = expression (sum(y))

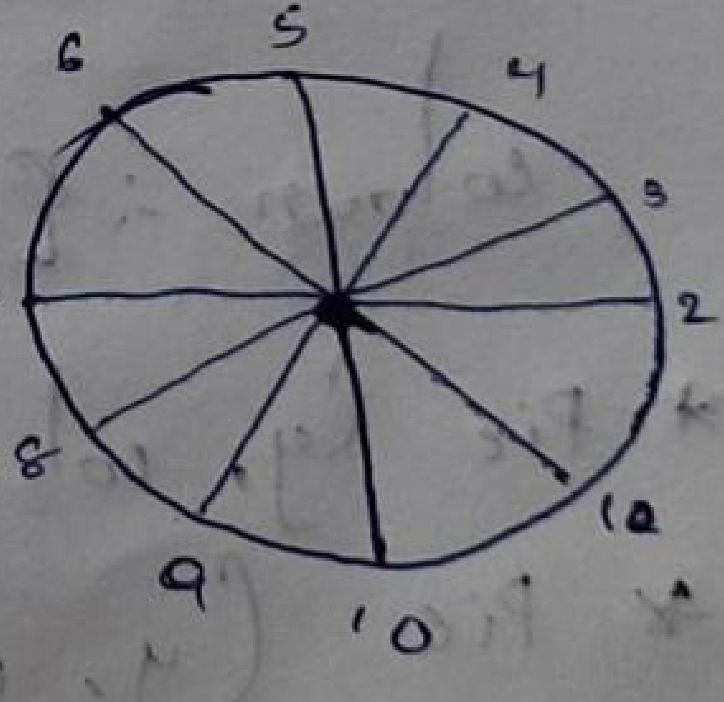
* xlimit, ylimit

barplot (y, ylim = c(0,10))

barplot (y, xlim = c(0,05))

Pie chart

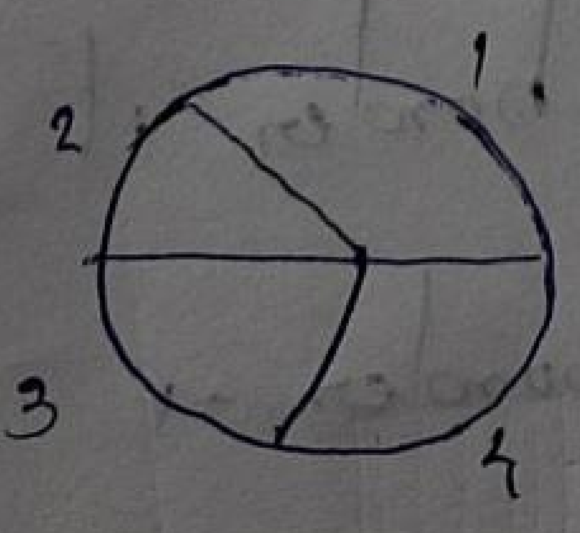
Diagramatica representation of values



eg ÷ x = c(1,1,1, 2, 2, 3, 3, 4,4, 4)

Pie (x)

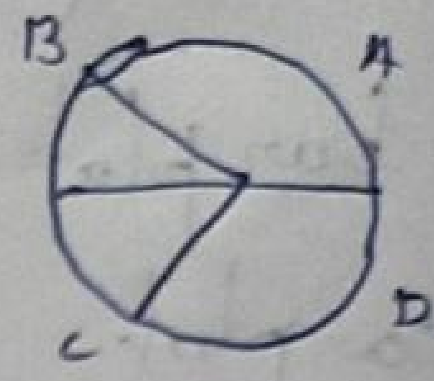
```
> x = c(1, 1, 1, 2, 2, 3, 3, 4, 4, 4)
> y = table(x)
> pie(y)
```



> Pie (y, main = "my first plot")

* x → a vector of non negative numerical quantities

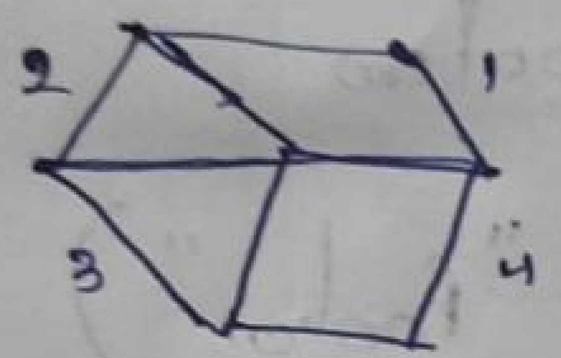
eg 1 > Pie (y = lable = LETTER [1 : 14])



eg 2 > pie(y, labels = c("red", "blue", "green", "orange")

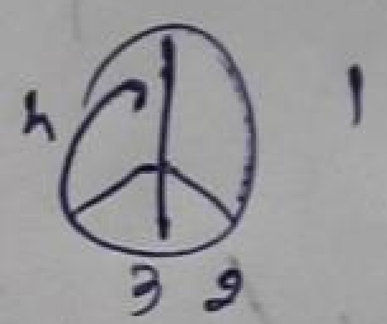
* labels are names of each slices

⊛ Pie (y. edges = 10)



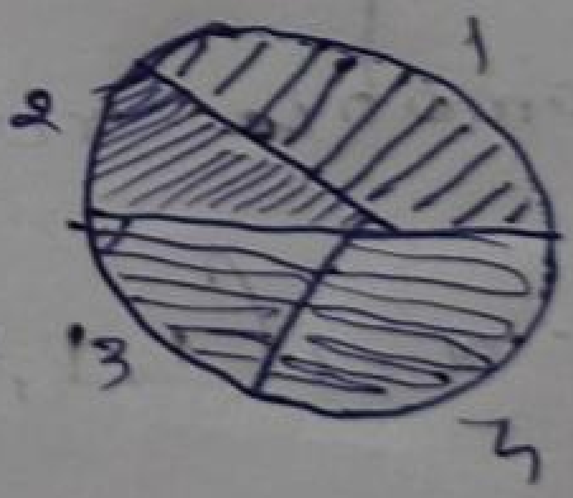
⊛ Pie (y, radius = ·5)

⊛ Pie (y. clock wise = T)



⊛ Pie (y . density = c(10, 20, 30, 40)

Density : used to give shading to each slice



colour : [col]

* Pie (y, col = rainbow (15))

* Pie (y, col = 1:4)

Border

used to set border it can be either T or F

Pie (y, col = 1:4, border = F

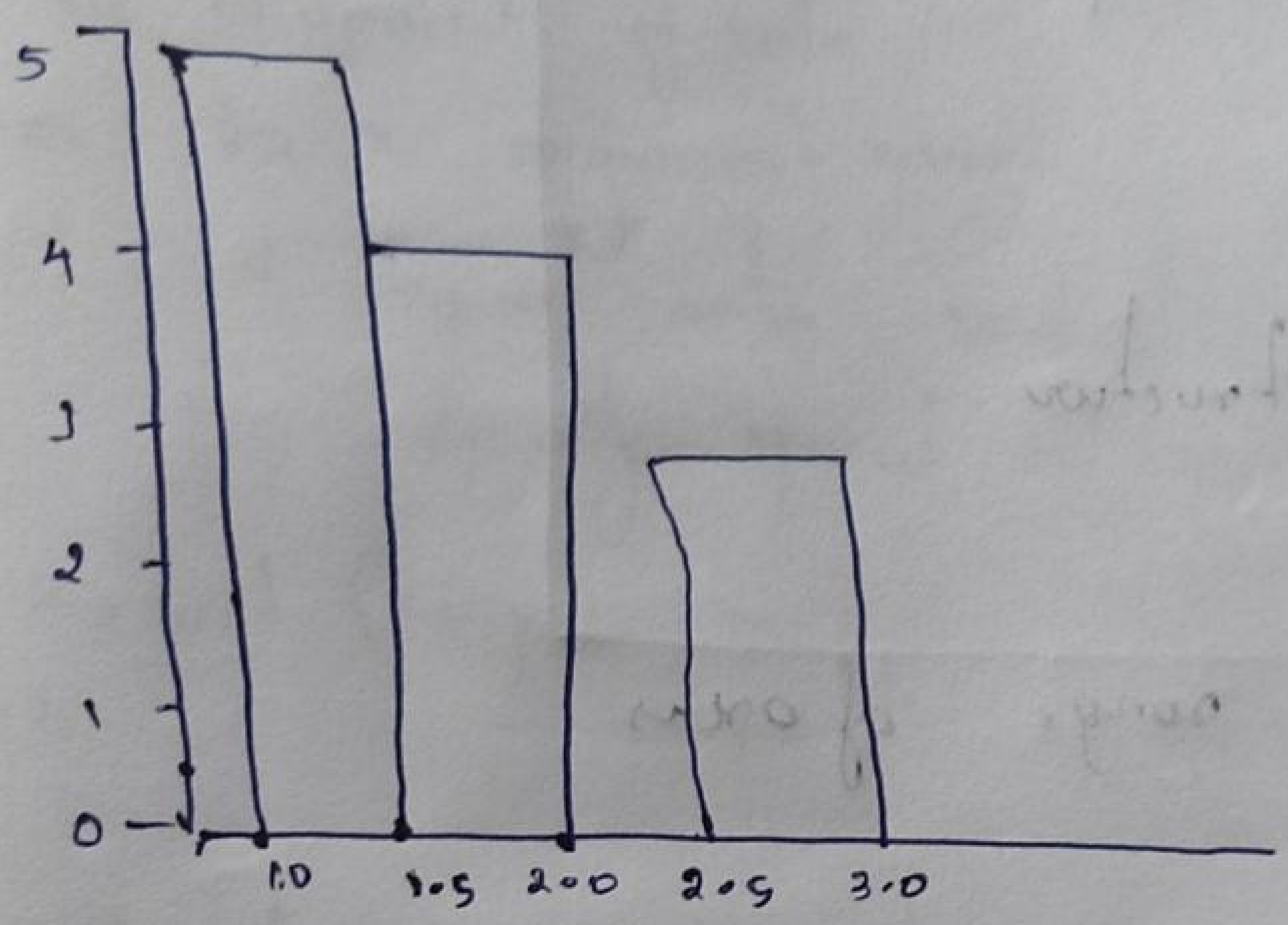
Histogram

Used to plot Quantitative data

funtion : hist()

* input are vector input

```
> x = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4)
> hist (x)
```



* for viewing the grouping arrangement use the function cut()

```
[1]. [ , ] , [ , ]
> data frame (x, cut(x, 6))
```

* data ("cars")

```
> head (cars)
   speed , dist
1    4      2
2    4      10
3    7      4
4    7      22
5    8      16
6    9      10
> car $ speed
  [1] ..
> hist (cars $ speed
```

Arguments

* breaks :- >hist (cars $ speed , x lab= "dist" , ylab = "no of times")

eg : > air quality
> head (air quality)

>temp = air quality . temp
> hist (temp)
> str (air quality)

str : used to display structure

* xlim, ylim :-

used to provide range of axes

* col :-

used to define color

* with the argument freq = false we can get the probability distribution instead of the frequency

> hist (temp , freq = F)

Eg : hist (temp , main = "maximum Daily Temperature" . xlab = "Temp in Degree " xlim = c (50, 100)

col = rainbow [20] , freq = false , las = 1

* Border = (T/F)

* density :-

> ~~hist ltr~~ hist (temp , border = "blue" . density = 20)

Return value of hist()

Display the value in hist ()

* breaks : places when the breaks occur

* counts : The no of observation falling in that cell

* density : The density of cells

* mids: The midpoint of cells

* x names : The x argument name

* equidist : A logical value indicating if the break are equally spaced or not

eg : h=hist (temp)
 > h

* with the break argument we can specify the no of cells we want in the histogram.

* we can also define break points b/w the cells as a vector, this makes it possible to plot a histogram with enequal intervals

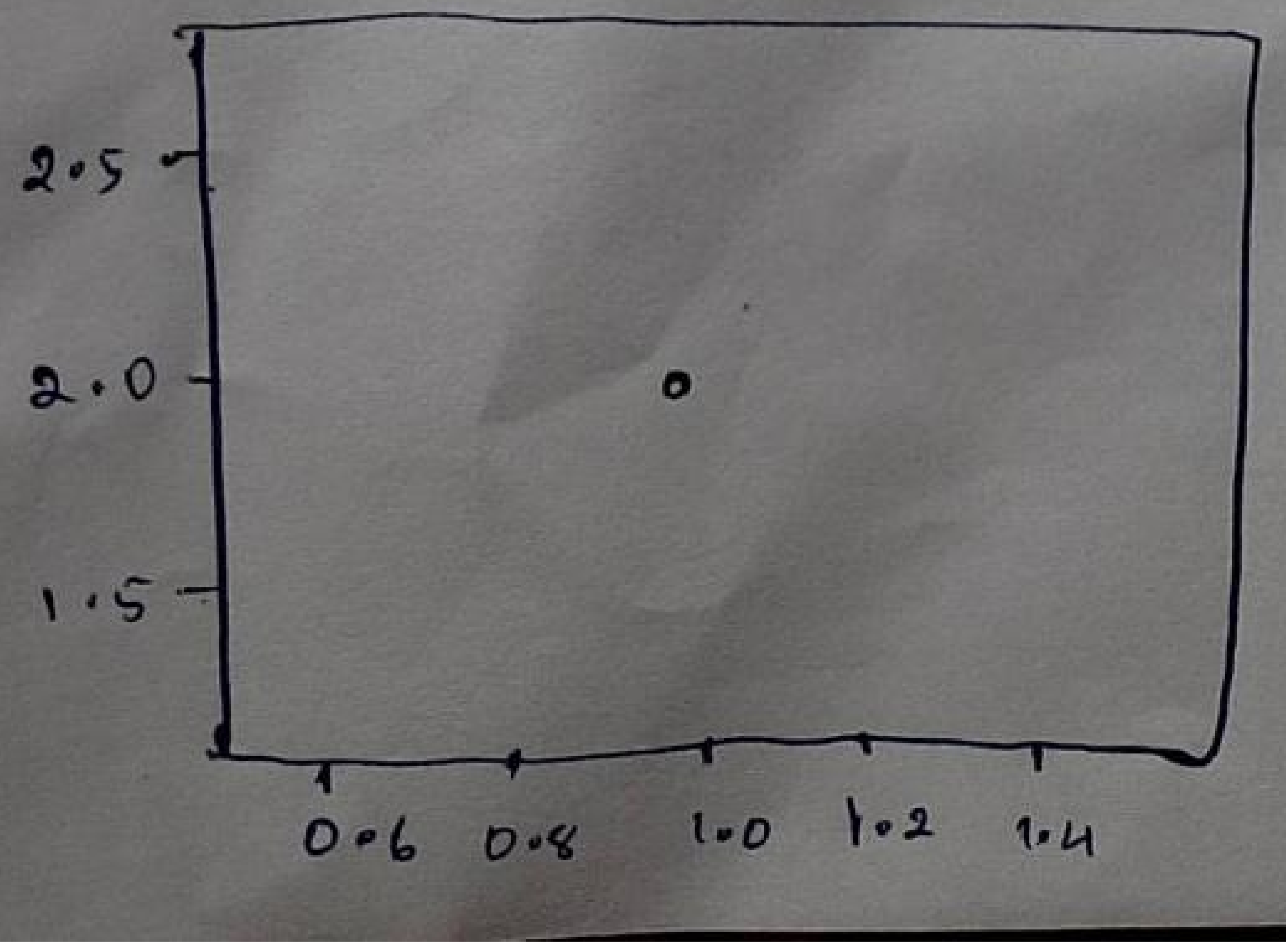
> hist (temp , border = "blue , breaks = c(55,60, 70, 75, 80, 100))

Scatter plot

Creating by using plot() funtion

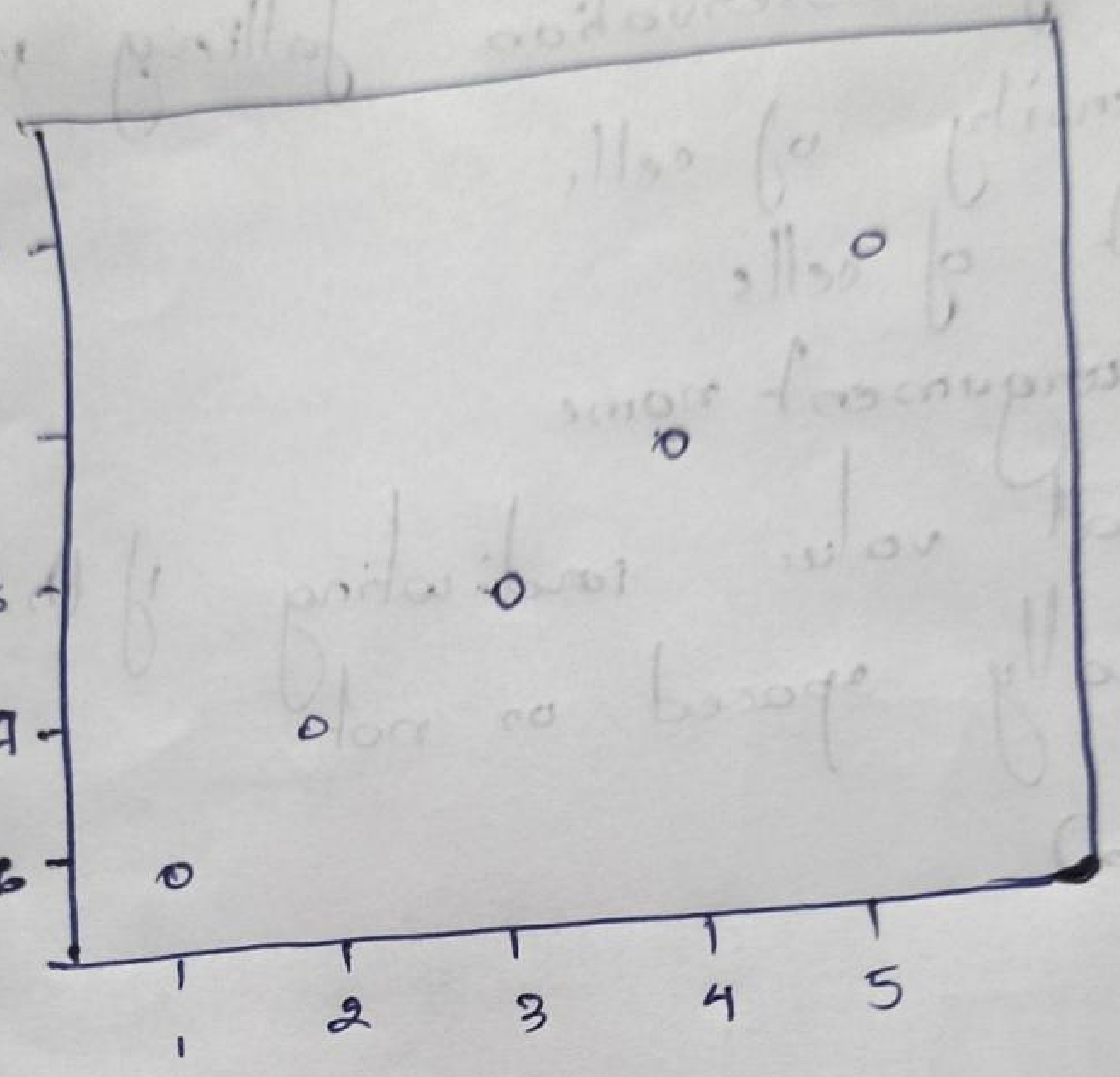
eg > plot (2)

>



eg > plot (c(5,6,7,8))

```
Eg- > x = 1:5
    > y = 6:10
    > plot (x, y)
```



```
eg > air Quality
   > head (air, quality)
   1
   2
   3
   4
   5
   6
   >day = air quality $day
   >temp = air quality $temp
   > plot (day, temp)
```