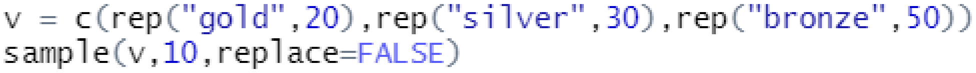
Probability and Statistics (UCS410)

Experiment 2: Descriptive statistics, Sample space, definition of probability

Name: Ketan Singh Group: 3CO14 Roll no.: 102003349

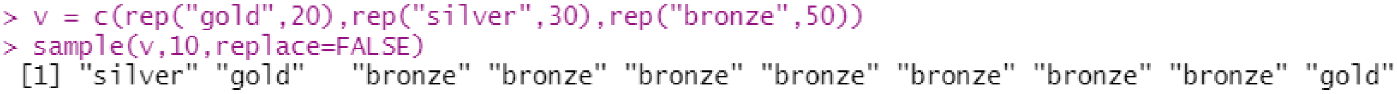
---

(1) (a) Suppose there is a chest of coins with 20 gold, 30 silver and 50 bronze coins. You randomly draw 10 coins from this chest. Write an R code which will give us the sample space for this experiment. (use of sample(): an in-built function in R)



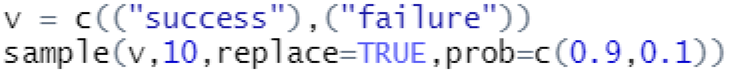
```
v = c(rep("gold",20),rep("silver",30),rep("bronze",50))
sample(v,10,replace=FALSE)
```

Output:



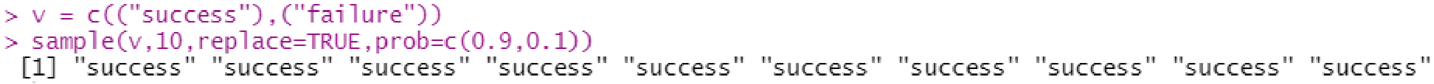
```
> v = c(rep("gold",20),rep("silver",30),rep("bronze",50))
> sample(v,10,replace=FALSE)
 [1] "silver" "gold"   "bronze" "bronze" "bronze" "bronze" "bronze" "bronze" "bronze" "gold"
```

(b) In a surgical procedure, the chances of success and failure are 90% and 10% respectively. Generate a sample space for the next 10 surgical procedures performed. (use of prob(): an in-built function in R)



```
v = c(("success"),("failure"))
sample(v,10,replace=TRUE,prob=c(0.9,0.1))
```

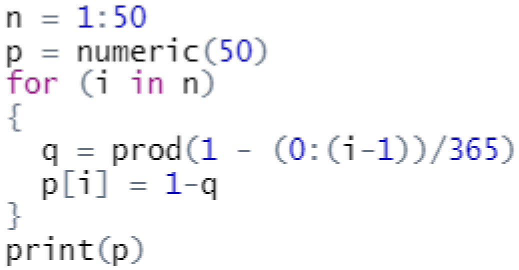
Output:



```
> v = c(("success"),("failure"))
> sample(v,10,replace=TRUE,prob=c(0.9,0.1))
 [1] "success" "success" "success" "success" "success" "success" "success" "success" "success" "success"
```

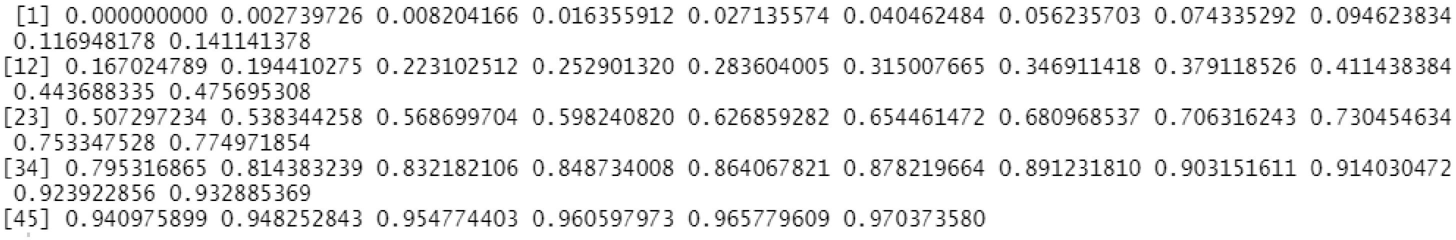
(2) A room has n people, and each has an equal chance of being born on any of the 365 days of the year. (For simplicity, we'll ignore leap years). What is the probability that two people in the room have the same birthday?

(a) Use an R simulation to estimate this for various n.



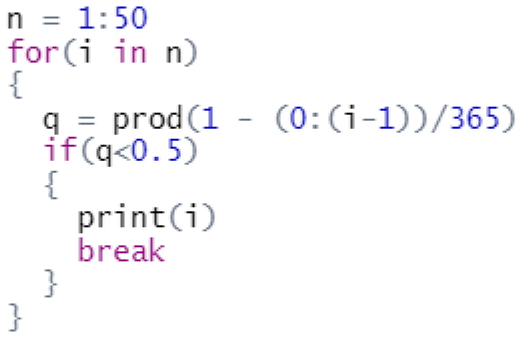
```
n = 1:50
p = numeric(50)
for (i in n)
{
  q = prod(1 - (0:(i-1))/365)
  p[i] = 1-q
}
print(p)
```

Output:



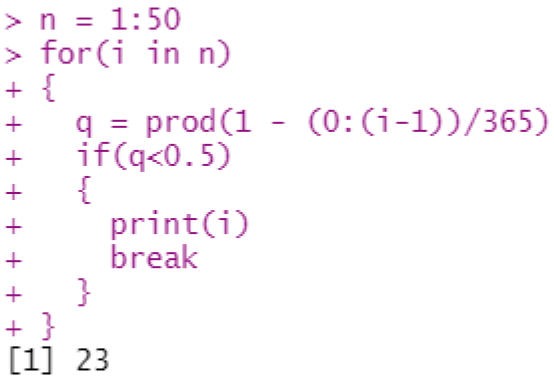
```
 [1] 0.000000000 0.002739726 0.008204166 0.016355912 0.027135574 0.040462484 0.056235703 0.074335292 0.094623834
 0.116948178 0.141141378
[12] 0.167024789 0.194410275 0.223102512 0.252901320 0.283604005 0.315007665 0.346911418 0.379118526 0.411438384
 0.443688335 0.475695308
[23] 0.507297234 0.538344258 0.568699704 0.598240820 0.626859282 0.654461472 0.680968537 0.706316243 0.730454634
 0.753347528 0.774971854
[34] 0.795316865 0.814383239 0.832182106 0.848734008 0.864067821 0.878219664 0.891231810 0.903151611 0.914030472
 0.923922856 0.932885369
[45] 0.940975899 0.948252843 0.954774403 0.960597973 0.965779609 0.970373580
```

(b) Find the smallest value of n for which the probability of a match is greater than .5.



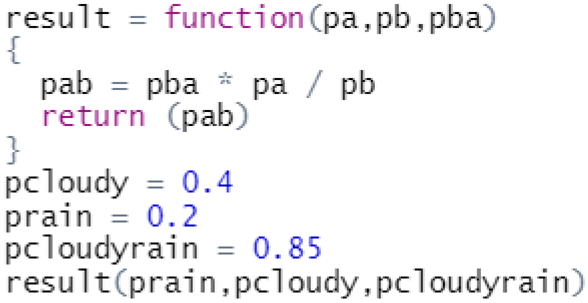
```
n = 1:50
for(i in n)
{
  q = prod(1 - (0:(i-1))/365)
  if(q<0.5)
  {
    print(i)
    break
  }
}
```

Output:



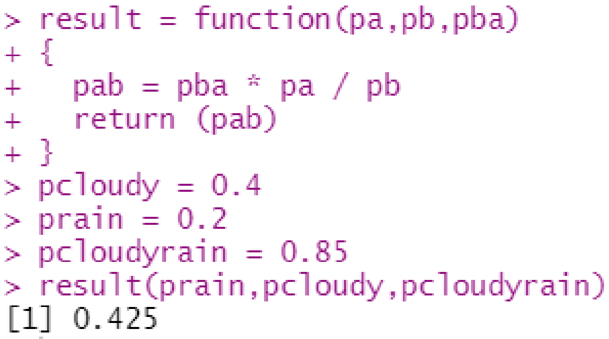
```
> n = 1:50
> for(i in n)
+ {
+   q = prod(1 - (0:(i-1))/365)
+   if(q<0.5)
+   {
+     print(i)
+     break
+   }
+ }
[1] 23
```

(3) Write an R function for computing conditional probability. Call this function to do the following problem: suppose the probability of the weather being cloudy is 40%. Also suppose the probability of rain on a given day is 20% and that the probability of clouds on a rainy day is 85%. If it’s cloudy outside on a given day, what is the probability that it will rain that day?



```
result = function(pa,pb,pba)
{
  pab = pba * pa / pb
  return (pab)
}
pcloudy = 0.4
prain = 0.2
pcloudyrain = 0.85
result(prain,pcloudy,pcloudyrain)
```

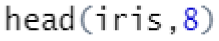
Output:



```
> result = function(pa,pb,pba)
+ {
+   pab = pba * pa / pb
+   return (pab)
+ }
> pcloudy = 0.4
> prain = 0.2
> pcloudyrain = 0.85
> result(prain,pcloudy,pcloudyrain)
[1] 0.425
```

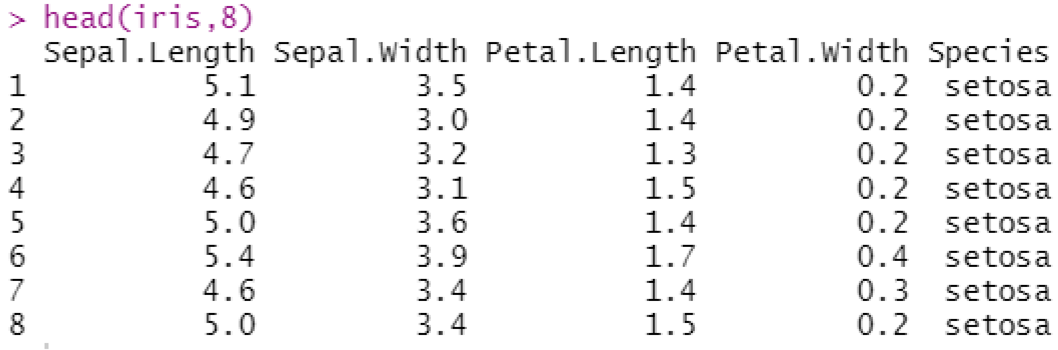
(4) The iris dataset is a built-in dataset in R that contains measurements on 4 different attributes (in centimeters) for 150 flowers from 3 different species. Load this dataset and do the following:

(a) Print first few rows of this dataset.



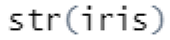
```
head(iris,8)
```

Output:



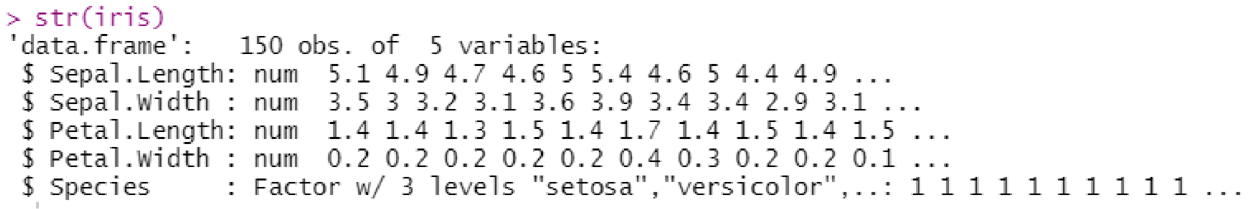
```
> head(iris,8)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
7          4.6         3.4          1.4         0.3  setosa
8          5.0         3.4          1.5         0.2  setosa
```

(b) Find the structure of this dataset.



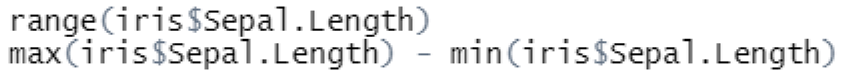
```
str(iris)
```

Output:



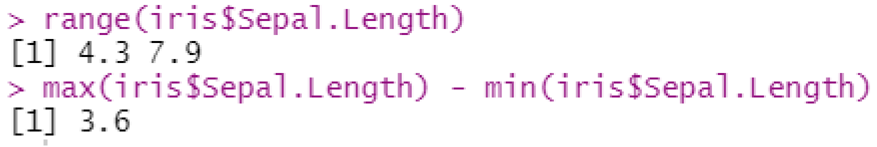
```
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

(c) Find the range of the data regarding the sepal length of flowers.



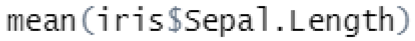
```
range(iris$Sepal.Length)
max(iris$Sepal.Length) - min(iris$Sepal.Length)
```

Output:



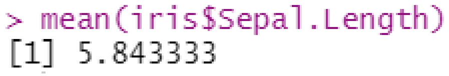
```
> range(iris$Sepal.Length)
[1] 4.3 7.9
> max(iris$Sepal.Length) - min(iris$Sepal.Length)
[1] 3.6
```

(d) Find the mean of the sepal length.



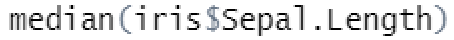
```
mean(iris$Sepal.Length)
```

Output:



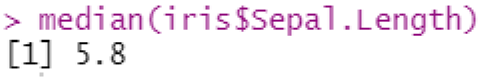
```
> mean(iris$Sepal.Length)
[1] 5.843333
```

(e) Find the median of the sepal length.



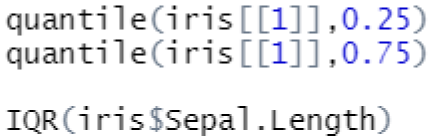
```
median(iris$Sepal.Length)
```

Output:



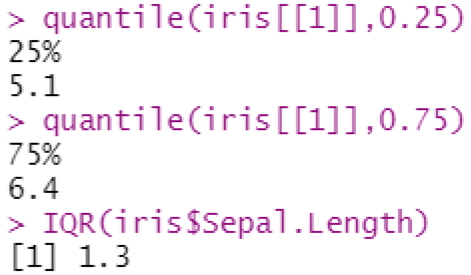
```
> median(iris$Sepal.Length)
[1] 5.8
```

(f) Find the first and the third quartiles and hence the interquartile range.



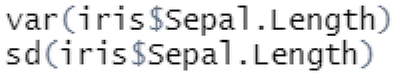
```
quantile(iris[[1]],0.25)
quantile(iris[[1]],0.75)

IQR(iris$Sepal.Length)
```

Output:



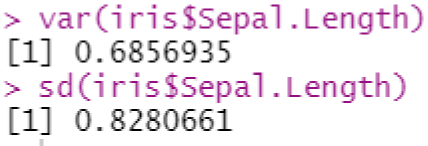
```
> quantile(iris[[1]],0.25)
25% 
5.1 
> quantile(iris[[1]],0.75)
75% 
6.4 
> IQR(iris$Sepal.Length)
[1] 1.3
```

(g) Find the standard deviation and variance.



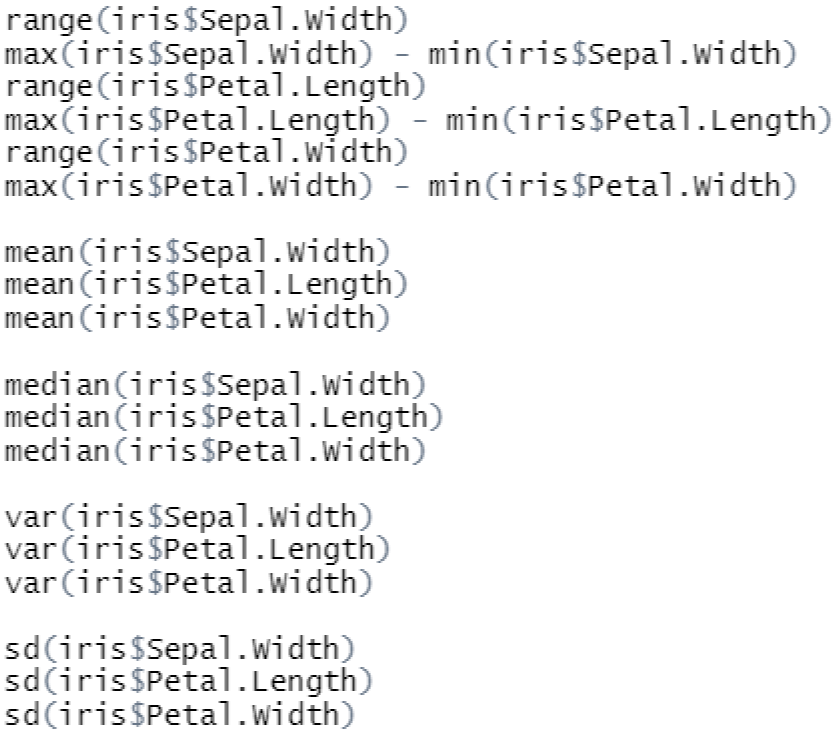
```
var(iris$Sepal.Length)
sd(iris$Sepal.Length)
```

Output:



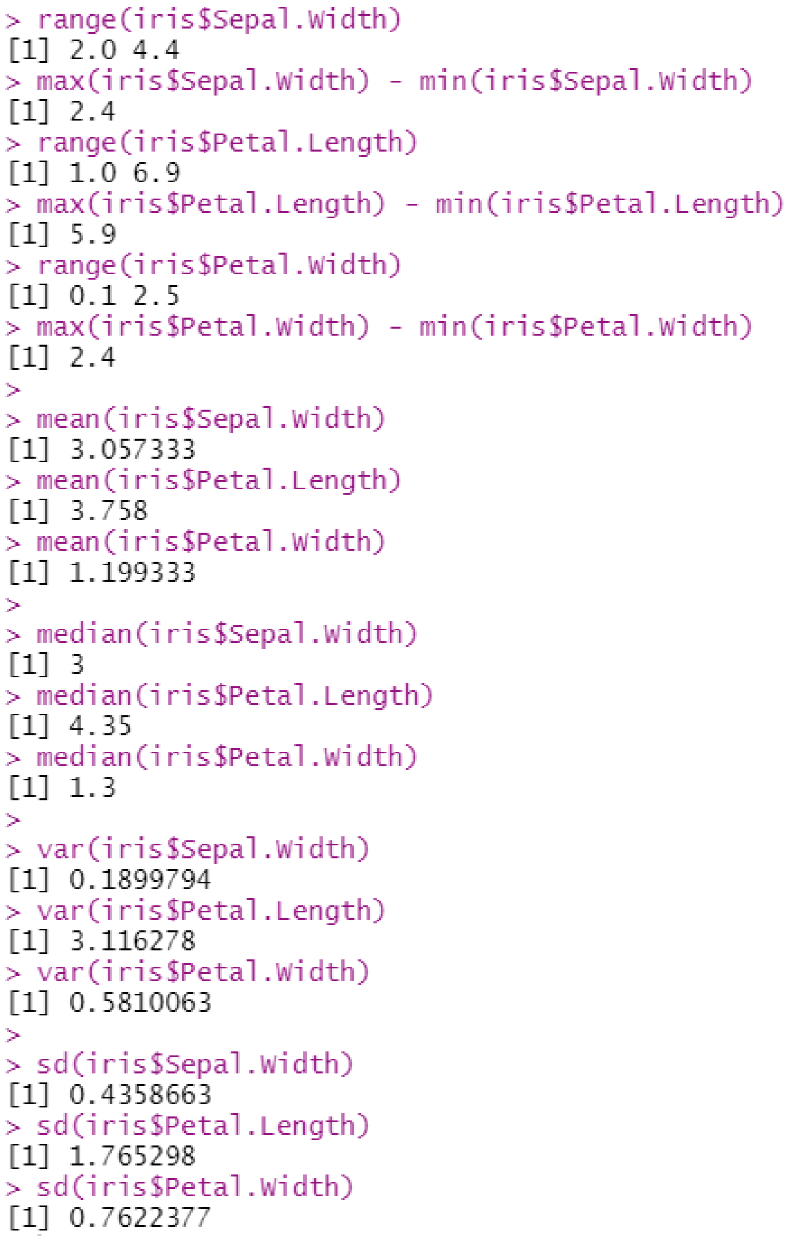
```
> var(iris$Sepal.Length)
[1] 0.6856935
> sd(iris$Sepal.Length)
[1] 0.8280661
```

(h) Try doing the above exercises for sepal.width, petal.length and petal.width.



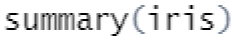
```
range(iris$Sepal.Width)
max(iris$Sepal.Width) - min(iris$Sepal.Width)
range(iris$Petal.Length)
max(iris$Petal.Length) - min(iris$Petal.Length)
range(iris$Petal.Width)
max(iris$Petal.Width) - min(iris$Petal.Width)

mean(iris$Sepal.Width)
mean(iris$Petal.Length)
mean(iris$Petal.Width)

median(iris$Sepal.Width)
median(iris$Petal.Length)
median(iris$Petal.Width)

var(iris$Sepal.Width)
var(iris$Petal.Length)
var(iris$Petal.Width)

sd(iris$Sepal.Width)
sd(iris$Petal.Length)
sd(iris$Petal.Width)
```

Output:



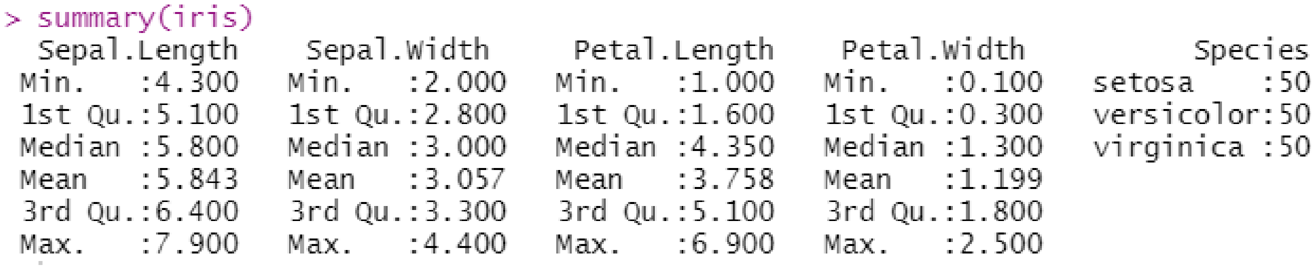
```
> range(iris$Sepal.Width)
[1] 2.0 4.4
> max(iris$Sepal.Width) - min(iris$Sepal.Width)
[1] 2.4
> range(iris$Petal.Length)
[1] 1.0 6.9
> max(iris$Petal.Length) - min(iris$Petal.Length)
[1] 5.9
> range(iris$Petal.Width)
[1] 0.1 2.5
> max(iris$Petal.Width) - min(iris$Petal.Width)
[1] 2.4
>
> mean(iris$Sepal.Width)
[1] 3.057333
> mean(iris$Petal.Length)
[1] 3.758
> mean(iris$Petal.Width)
[1] 1.199333
>
> median(iris$Sepal.Width)
[1] 3
> median(iris$Petal.Length)
[1] 4.35
> median(iris$Petal.Width)
[1] 1.3
>
> var(iris$Sepal.Width)
[1] 0.1899794
> var(iris$Petal.Length)
[1] 3.116278
> var(iris$Petal.Width)
[1] 0.5810063
>
> sd(iris$Sepal.Width)
[1] 0.4358663
> sd(iris$Petal.Length)
[1] 1.765298
> sd(iris$Petal.Width)
[1] 0.7622377
```

(i) Use the built-in function summary on the dataset Iris.



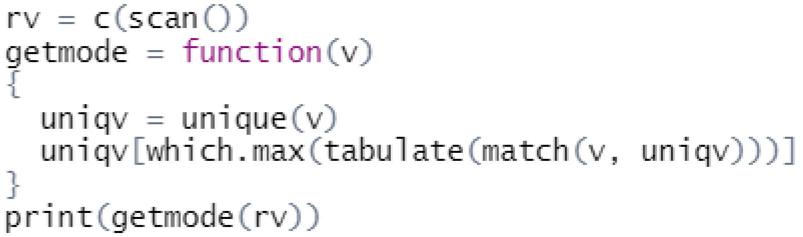
```
summary(iris)
```

Output:



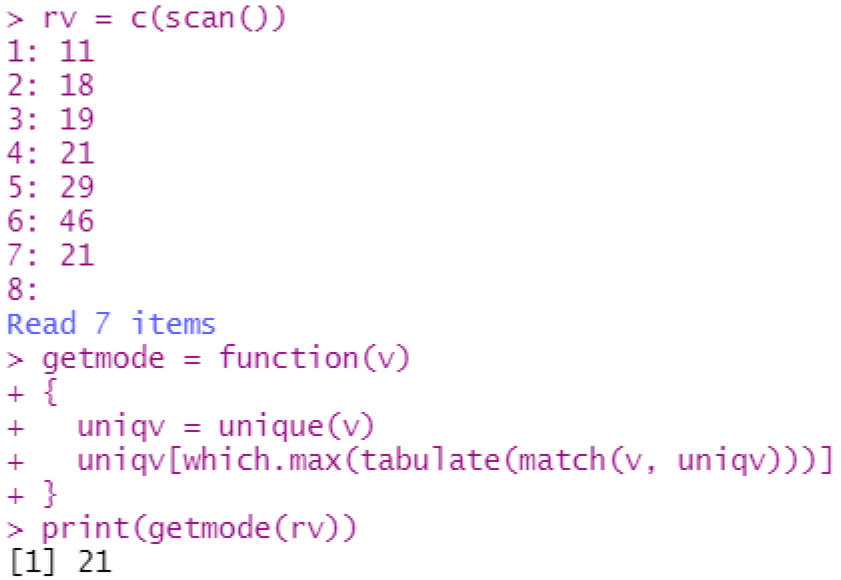
```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
```

(5) R does not have a standard in-built function to calculate mode. So we create a user function to calculate mode of a data set in R. This function takes the vector as input and gives the mode value as output.



```
rv = c(scan())
getmode = function(v)
{
  uniqv = unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
print(getmode(rv))
```

Output:



```
> rv = c(scan())
1: 11
2: 18
3: 19
4: 21
5: 29
6: 46
7: 21
8: 
Read 7 items
> getmode = function(v)
+ {
+   uniqv = unique(v)
+   uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
> print(getmode(rv))
[1] 21
```