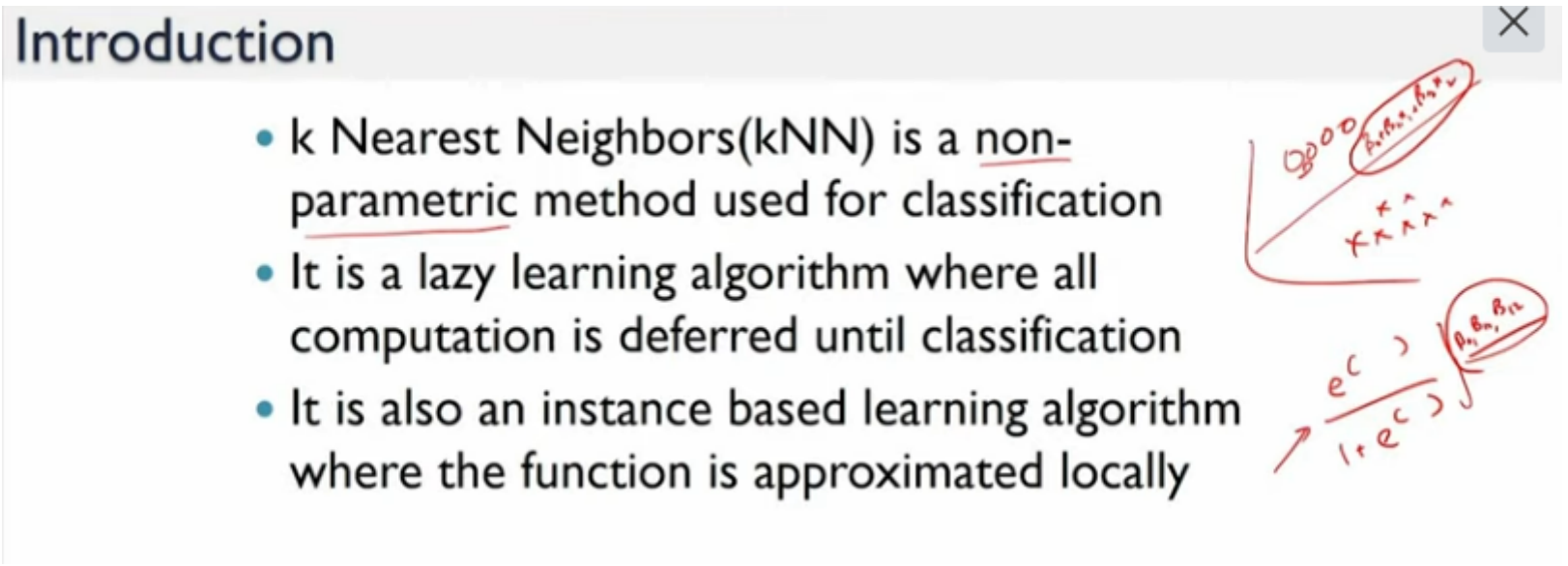
on 25th September, 2021 ~ Saturday_4

Module-1: **K-Nearest Neighbors** *--- A Supervised Learning algorithm*



- Its **Non-Parametric**

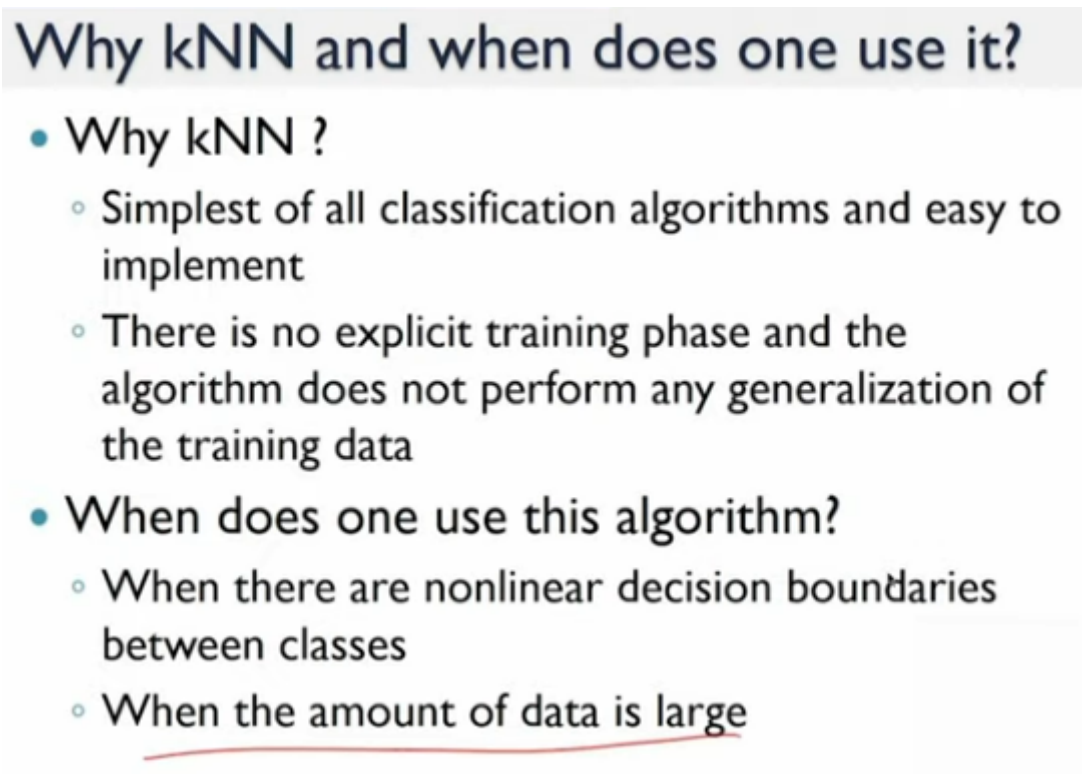
> i.e., (Recall the idea of applying *Linear Regression, Multiple Regression...*, there we derive some parameters *[informally.. ""a function""]* from the training data, and later when proposed a new test_data, those are plugged into those parameters and calculated).
>
> ----------*That's called the **Parametric Method***
>
> Here interestingly.. we don't use any such parameters. -- Hence called **Non-Parametric Method..**

Then what about telling that **KNN with 4,5,..8 neighbors..??** aren't we using any parameter there..??

> Its a **subtle difference**. That's the **tuning parameter** -- *In the sense that... these parameters are for tuning the model, and moreover, like the Linear Regression models, those parameters are derived from the given data itself..*

- Its a **Lazy Learning algorithm**

> Say for the Logistic regression.. in order to go for prediction, need to derive the parameters, then can go for prediction. Whereas here... just give the training-dataset and test-point -- it'll classify. But, a lot of work has to be done earlier to that



k Nearest Neighbors

- Input features
  - Input features can be both quantitative and qualitative
- Outputs
  - Outputs are categorical values, which typically are the classes of the data
- kNN explains a categorical value using the majority votes of nearest neighbors

Assumptions

- Being nonparametric, the algorithm does not make any assumptions about the underlying data distribution
- Select the parameter k based on the data
- Requires a distance metric to define proximity between any two data points
  - Example: Euclidean distance, Mahalanobis distance or Hamming distance

Algorithm

- The kNN classification is performed using the following four steps
  - Compute the distance metric between the test data point and all the labeled data points

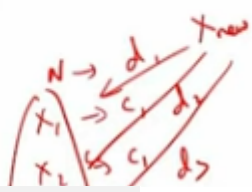


Illustration of kNN

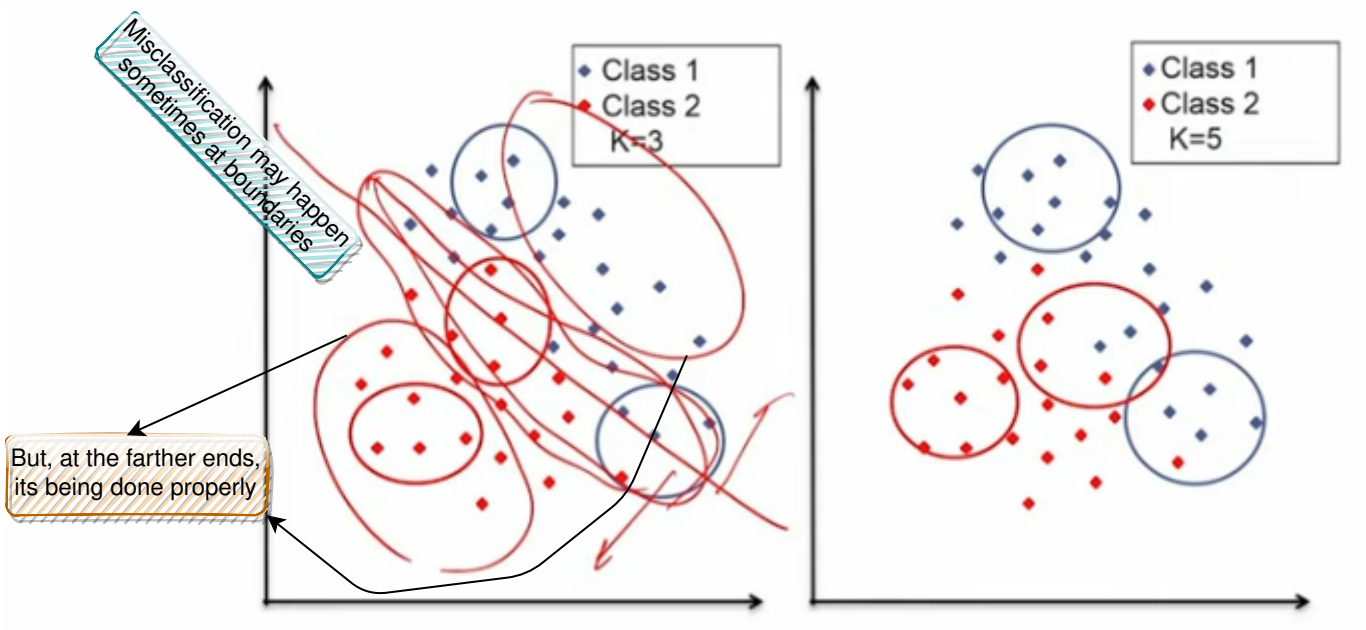
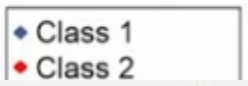
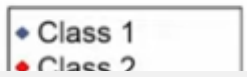


Illustration of kNN (Testing)



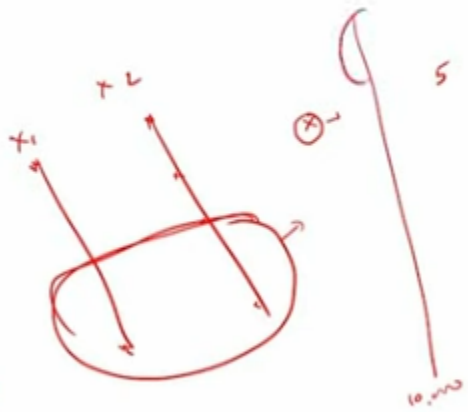


So, upon c

Things to consider

- Following are some things one should consider before applying kNN algorithm
  - Parameter selection — Like.. k=2, 4, 5...
  - Presence of noise
  - Feature selection and scaling
  - Curse of dimensionality

If had un-necessary features in the dataset, even they contribute to distance which may cause some misclassification.



(Say) had 10, 000 points in Database, and K=4, then to get 4 points, need to calculate distances for all those 10,000 points. So, there must be a smarter way of doing it.

Like, say, if had one attribute(T) is in 100's and another attribute(c) is in 0.1, 0.5...
Then distance between these two, would be very large.

So better idea is to scale-up or scale-down either the one

Parameter selection

- The best choice of k depends on the data
- Larger values of k reduce the effect of noise on classification but makes the decision boundaries between classes less distinct
- Smaller values of k tend to be affected by the noise with clear separation between classes

Feature selection and scaling

- It is important to remove irrelevant features
- When the number of features is too large, and suspected to be highly redundant, feature extraction is required
- If the features are carefully chosen then it is expected that the classification will be better

Module-2: **Implementing KNN in R**

Agenda:

In this lecture

- Case study
  - Problem statement
- Solve the case study using R
  - Read the data from a ".csv" file
  - Understand the data
  - knn() function
  - Interpret the results

Recap...

Key points from previous lecture

- knn is primarily used as a classification algorithm
- It is supervised learning algorithm
  - Data is labelled
- Non-parametric method
- No explicit training phase is involved
- Lazy learning algorithm
- Notion of distance is needed
- Majority voting method

Automotive Service Study: Problem statement

An automotive service chain is launching its new grand service station this weekend. They offer to service a wide variety of cars. The current capacity of the station is to check 315 cars thoroughly per day.

As an inaugural offer, they claim to freely check all cars that arrive on their launch day, and report whether they need servicing or not!

Unexpectedly, they get 450 cars. The service men won't work longer than the working hours but the data analysts have to!

Can you save the day for the new service station?

How can a data scientist save a day for them?

- He has been a data set which contains some attributes of car that can be easily measured and wont require much time and a conclusion that if service is needed for that or not. -"serviceTrainData.csv"
- Now for the cars they cannot check in detail, they measure those attributes- "serviceTestData.csv"
- Use knn classification technique to classify the cars they cannot test manually and say whether service is needed or not

Solution to case study using R

Getting things ready

- Setting working directory, clearing variables in the workspace
- Installing or loading required packages

```
##### knn Implementation in R #########
# Set the working directory as the directory which contains the
data files
# setwd("Path of the directory with data files")
rm(list=ls()) # to clear the environment
# install.packages("caret",dependencies = TRUE)
# install.packages("class",dependencies = TRUE)
library(caret) # for confusionMatrix
library(class) # for knn
```

In [2]:
```
install.packages("caret", dependencies=TRUE)
```

In [ ]:
```
install.pacakges("class", dependencies=TRUE)  # Contains various classification algorithms..
```

Reading the data

- Data for this case study is provided to you in files with names "serviceTrainData.csv ", "serviceTestData.csv"
- To read the data from a ".csv" file we use `read.csv()` function

.png) _1.png)

In [1]:
```
serviceTrain = read.csv("resources/datasets/serviceTrainData.csv")
serviceTest = read.csv("resources/datasets/serviceTestData.csv")   # Here even for this had the final column of answ
```

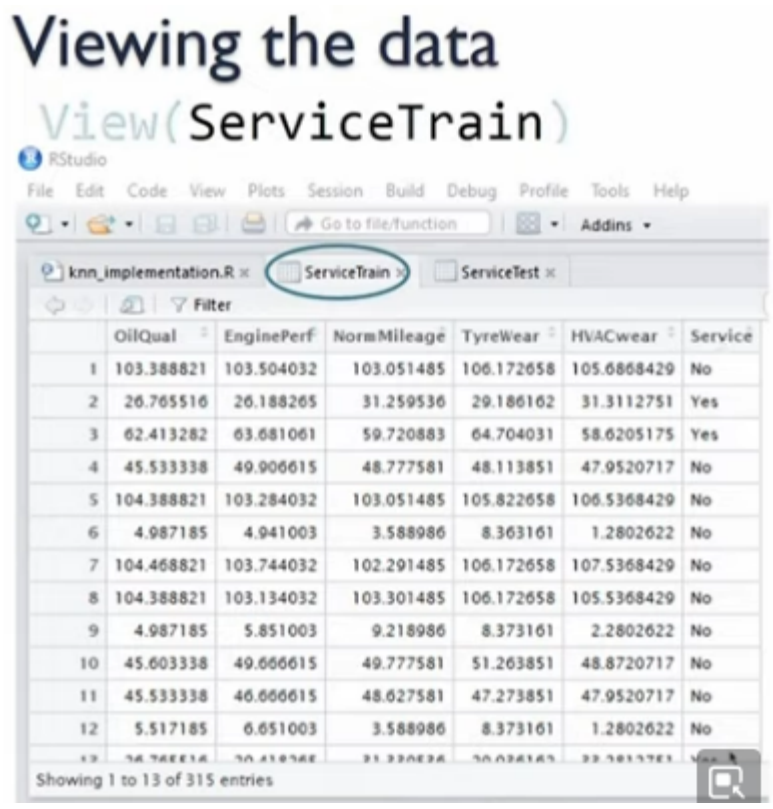
In [11]:
```
library(class)
```

In [12]:
```
library(caret)
```

```
Loading required package: lattice
Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
```

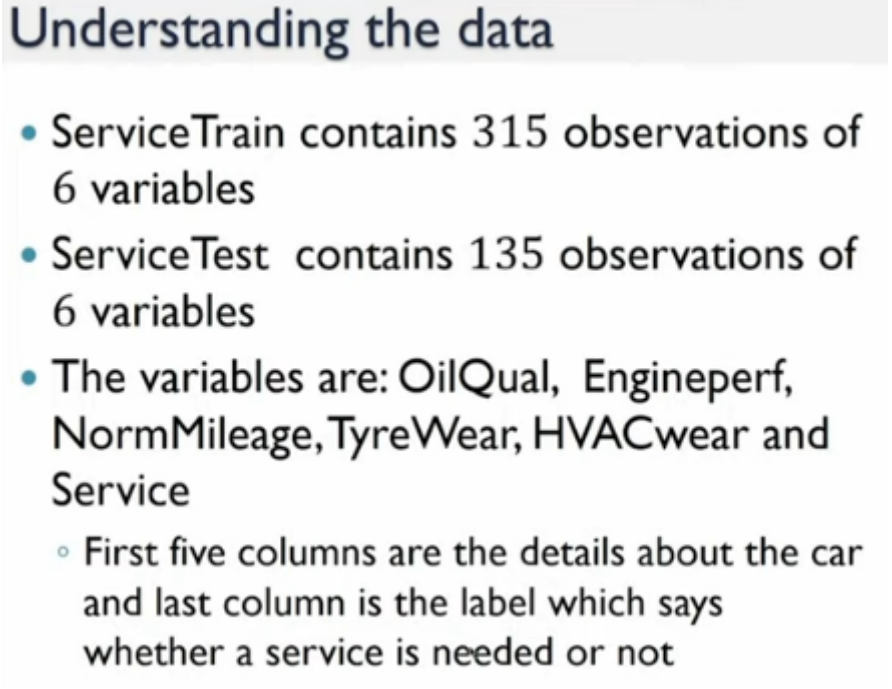
In [3]:
```
View(serviceTrain)
```

```
Error in View(serviceTrain): ‘View()’ not yet supported in the Jupyter R kernel
Traceback:

1. View(serviceTrain)
2. stop(sQuote("View()"), " not yet supported in the Jupyter R kernel")
```

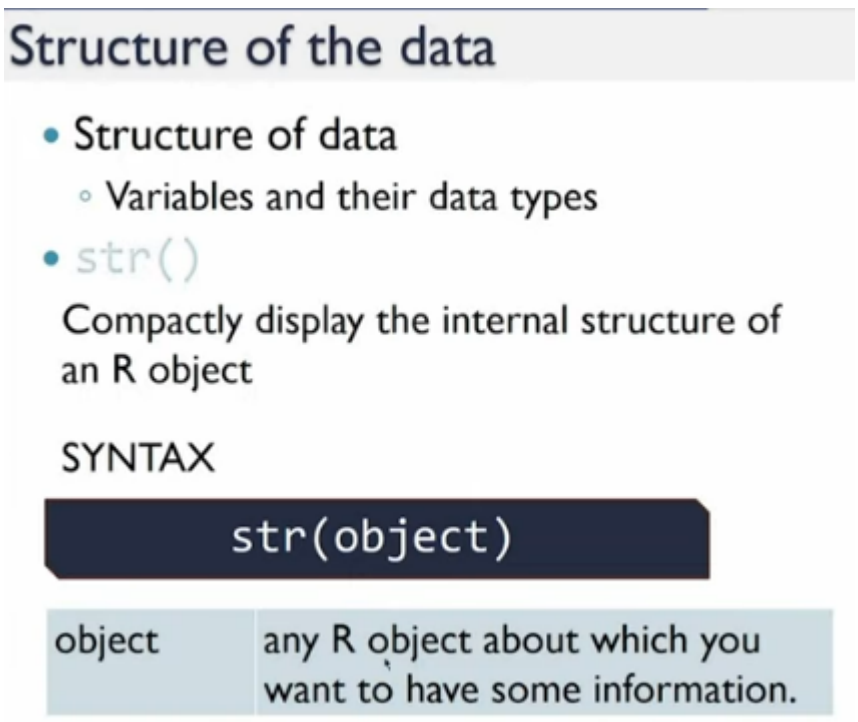


-- here in the test dataset, there ias already a column of "Whether the "

In [ ]:
```
serviceTrain
```



Ok, then what are the datatypes of each..?? How can one know about those..??

> via Structure of data as: `str(object)`
>
> 
>

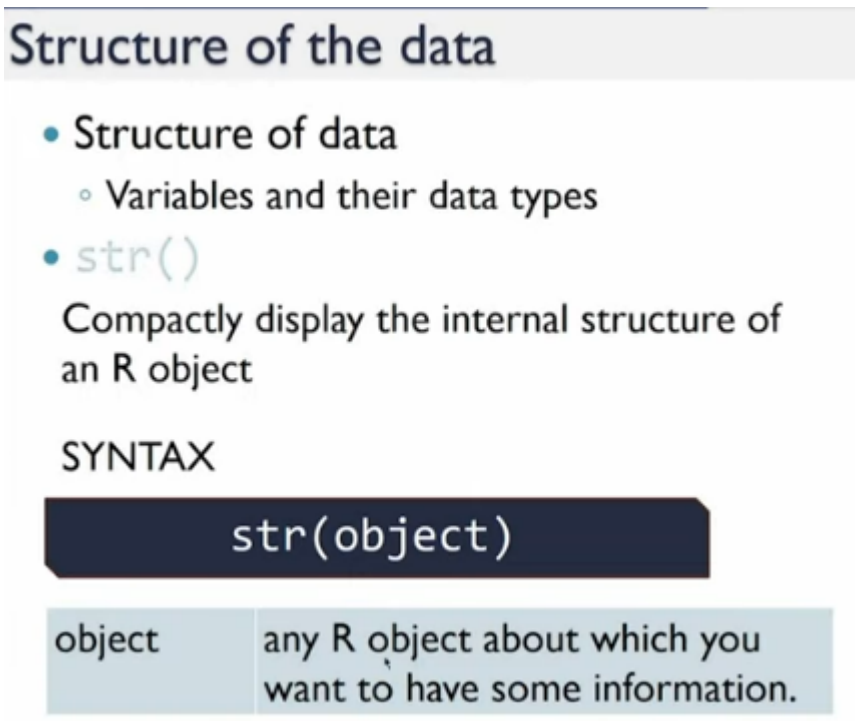
In [2]:
```
str(serviceTrain)
```

```
'data.frame':   315 obs. of  6 variables:
 $ OilQual    : num  103.4 26.8 62.4 45.5 104.4 ...
 $ EnginePerf : num  103.5 26.2 63.7 49.9 103.3 ...
 $ NormMileage: num  103.1 31.3 59.7 48.8 103.1 ...
 $ TyreWear   : num  106.2 29.2 64.7 48.1 105.8 ...
 $ HVACwear   : num  105.7 31.3 58.6 48 106.5 ...
 $ Service    : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 1 1 1 1 ...
```

In [3]:
```
str(serviceTest)
```

```
'data.frame':   135 obs. of  6 variables:
 $ OilQual    : num  45.77 4.99 4.99 106.39 104.39 ...
 $ EnginePerf : num  49.94 7.89 4.89 104.45 103.74 ...
 $ NormMileage: num  49.78 6.59 7.31 103.05 103.05 ...
 $ TyreWear   : num  48.26 9.49 8.37 106.28 106.13 ...
 $ HVACwear   : num  50.95 3.24 2.78 105.54 105.78 ...
 $ Service    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 2 1 ...
```

Is there any way of getting the summary of the data read..??

> Yes.. via `sumary(object)`
>
> 
>

In [4]:
```
summary(serviceTrain)
```

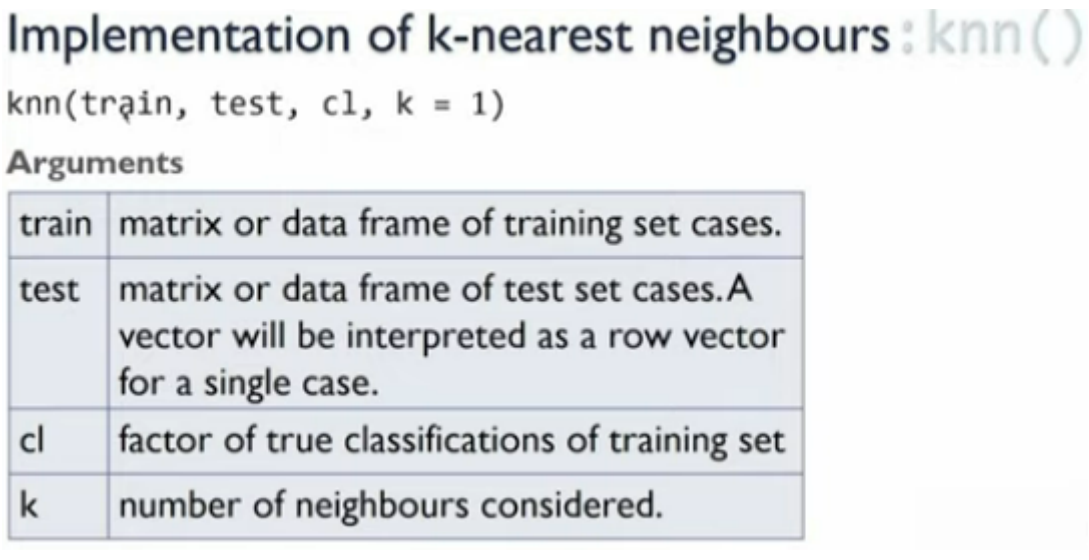
```
   OilQual          EnginePerf       NormMileage        TyreWear      
Min.   :  0.9872   Min.   :  1.891   Min.   :  3.359   Min.   :  6.213  
1st Qu.: 26.7655   1st Qu.: 27.418   1st Qu.: 31.260   1st Qu.: 29.036  
Median : 59.6633   Median : 59.741   Median : 57.221   Median : 60.304  
Mean   : 59.6493   Mean   : 60.306   Mean   : 60.297   Mean   : 61.759  
3rd Qu.:104.3888   3rd Qu.:103.744   3rd Qu.:103.051   3rd Qu.:106.173  
Max.   :106.4288   Max.   :105.744   Max.   :105.051   Max.   :108.173  
   HVACwear      Service   
Min.   : -1.72   No :232   
1st Qu.: 31.34   Yes: 83   
Median : 60.62             
Mean   : 60.39             
3rd Qu.:105.54             
Max.   :107.54             
```

In [5]:
```
summary(serviceTest)
```

```
   OilQual          EnginePerf       NormMileage        TyreWear      
Min.   :  2.597   Min.   :  1.891   Min.   :  3.589   Min.   :  6.143  
1st Qu.: 26.696   1st Qu.: 27.418   1st Qu.: 31.260   1st Qu.: 28.901  
Median : 61.023   Median : 61.501   Median : 59.351   Median : 61.304  
Mean   : 58.629   Mean   : 59.077   Mean   : 59.118   Mean   : 60.864  
3rd Qu.:104.229   3rd Qu.:103.744   3rd Qu.:103.051   3rd Qu.:106.173  
Max.   :106.389   Max.   :105.744   Max.   :105.051   Max.   :108.173  
   HVACwear      Service 
Min.   : -1.72   No :99  
1st Qu.: 31.31   Yes:36  
Median : 62.62           
Mean   : 58.99           
3rd Qu.:105.33           
Max.   :105.83           
```

As of now, thee one to be predicted: "**Service**", is already given, but now we need to that same job via **KNN**..

Come, let's go ahead..



Applying knn algorithm on data

```
# Applying k-NN algorithm
# K Nearest neighbour is a lazy algorithm and can do prediction directly with the testing
dataset, command "knn", accepts training and testing datasets the class variable of interest
i.e outcome categorical variable is provided for the parameter "cl". parameter "k" is to
specify the number of nearest neighbours required.

predictedknn <- knn(train = ServiceTrain[,-6],
                    test = ServiceTest[,-6],
                    cl = ServiceTrain$Service,
                    k  = 3)
```

- ServiceTrain[,-6] gives information in ServiceTrain except the last column
- ServiceTest[,-6] gives information in ServiceTestexcept the last column
- ServiceTrain$Service gives the last column of training data as a classification factor to the algorithm

In [14]:
```
KNN_prediction = knn(train = serviceTrain[, -6],     # Pass the whole training data, except the one, to be predicted
                     test  = serviceTest[, -6],      # for us, they gave the: finally predicted value, but we need to
                     cl = serviceTrain$Service,      # Column which contains the target-predictions of the training_d
                     k=3)                            # num_neighbors
```

In [ ]:
```
KNN_prediction
```

Results: predicted classes

- "predictedknn" is the output from the algorithm, which has a categorical variable "Yes" or "No", indicating whether service is needed or not for each case in Test data

```
> # printing the information in predictedknn
> predictedknn
  [1] No  No  No  No  No  No  No  Yes Yes No  No
 [12] No  No  No  No  No  Yes No  No  Yes Yes No
 [23] Yes No  No  No  No  No  No  No  No  No  No
 [34] No  No  Yes No  No  No  No  No  Yes No  Yes
 [45] No  No  No  Yes Yes No  Yes No  Yes No  No
 [56] No  No  No  No  No  No  No  No  No  Yes Yes
 [67] Yes No  Yes No  No  Yes No  No  No  No  No
 [78] No  Yes Yes Yes Yes No  Yes No  No  Yes Yes
 [89] Yes No  No  No  Yes No  Yes No  No  No  No
[100] No  No  No  No  No  No  Yes No  No  No  No
[111] No  Yes No  Yes No  Yes Yes No  Yes No  No
[122] No  No  No  Yes No  No  No  No  No  No  No
[133] Yes No  No
Levels: No Yes
```

How well this classification happened.....?? .. is there any such metric to know..??

> Yes..!! its the **Confusion Matrix**.
> Can obtain manually or via `caret` package..

Manual approach..!!

Results: generating confusion matrix manually

```
# Command to develop and print a confusion matrix
conf_matrix = table(predictedknn,ServiceTest[,6])

predictedknn No Yes
         No  99   0
         Yes  0  36
```

```
# Ameasure of accuracy is calculated by summing the true positives and true negatives and dividing them by total number of samples
knn_accuracy = sum(diag(conf_matrix))/nrow(ServiceTest)

> knn_accuracy
[1] 1
```

In [19]:
```
conf_matrix = table(KNN_prediction, serviceTest$Service)
conf_matrix
```

```
KNN_prediction No Yes
           No  99   0
           Yes  0  36
```

Oh...!! Its quite good..!! Predicted **Yes** as **Only YES** and **False** as **Only FALSE**..

In [21]:
```
# Let's find out the accuracy (Consider the diagonal elements: refer to the Performance Metric Lecture of Week--????
prediction_accuracy <- sum(diag(conf_matrix)/nrow(serviceTest))
prediction_accuracy
```

1

That's what connotated here numerically... right

Via `caret` package..

Results

```
# confusionMatrix command shown below used from caret package

COnF_Matrix <-confusionMatrix(data = predictedknn,ServiceTest$Service)
```

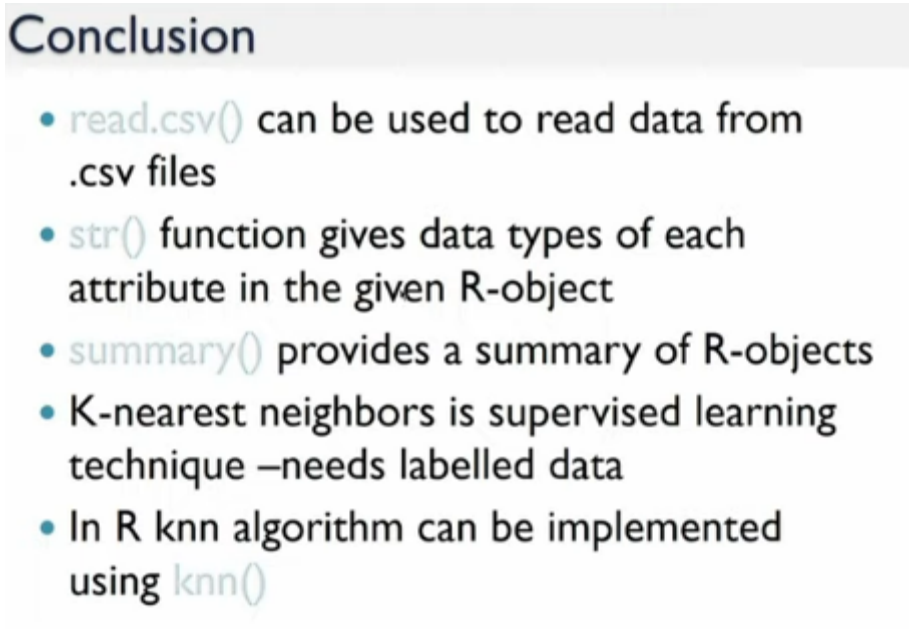
```
> COnF_Matrix

Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  99   0
       Yes  0  36

               Accuracy : 1
                 95% CI : (0.973, 1)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : < 2.2e-16
```

In [24]:
```
library('caret')
conf_matrix <- confusionMatrix(data = KNN_prediction,
                               serviceTest$Service)
conf_matrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  99   0
       Yes  0  36

               Accuracy : 1
                 95% CI : (0.973, 1)
    No Information Rate : 0.7333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.7333
         Detection Rate : 0.7333
   Detection Prevalence : 0.7333
      Balanced Accuracy : 1.0000

       'Positive' Class : No
```

`Sensitivity` and `Specificity` are 1, because all TRUE's are classified as TRUE and respectively for All FALSE's..
and the

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} = \frac{1+1}{2} = 1$$



---

on 24th September, 2021 ~ Friday_4

Done in Laptop, need to transfer to desktop to merge..

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

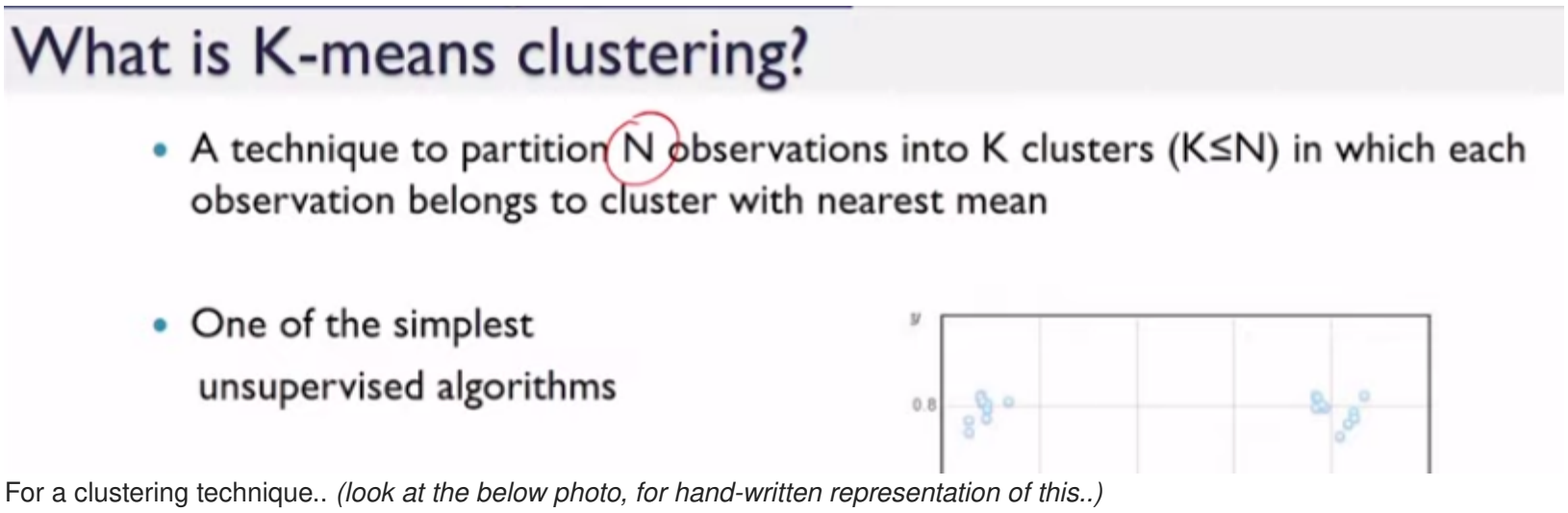
In [ ]:

K-Means clustering algorithm (Un-Supervised)

What is K-Means Clustering?

- It is a technique to cluster or partition a certain no. of observations (Say, N observations into K clusters)
    - that "K" canbe choosen manually or via algorithmically.
- For the classification algorithms, we're given with the labelled data points, and their job is to find the """**Decision Boundaries between different classes**""" -- These are supervised algos - Ex: KNN. But many of the clustering algos are "UnSupervised algos" -- i.e., they are not labelled into classes.

    > So, can think that... Clustering is slightly diff. from Classification.



For a clustering technique.. *(look at the below photo, for hand-written representation of this..)*

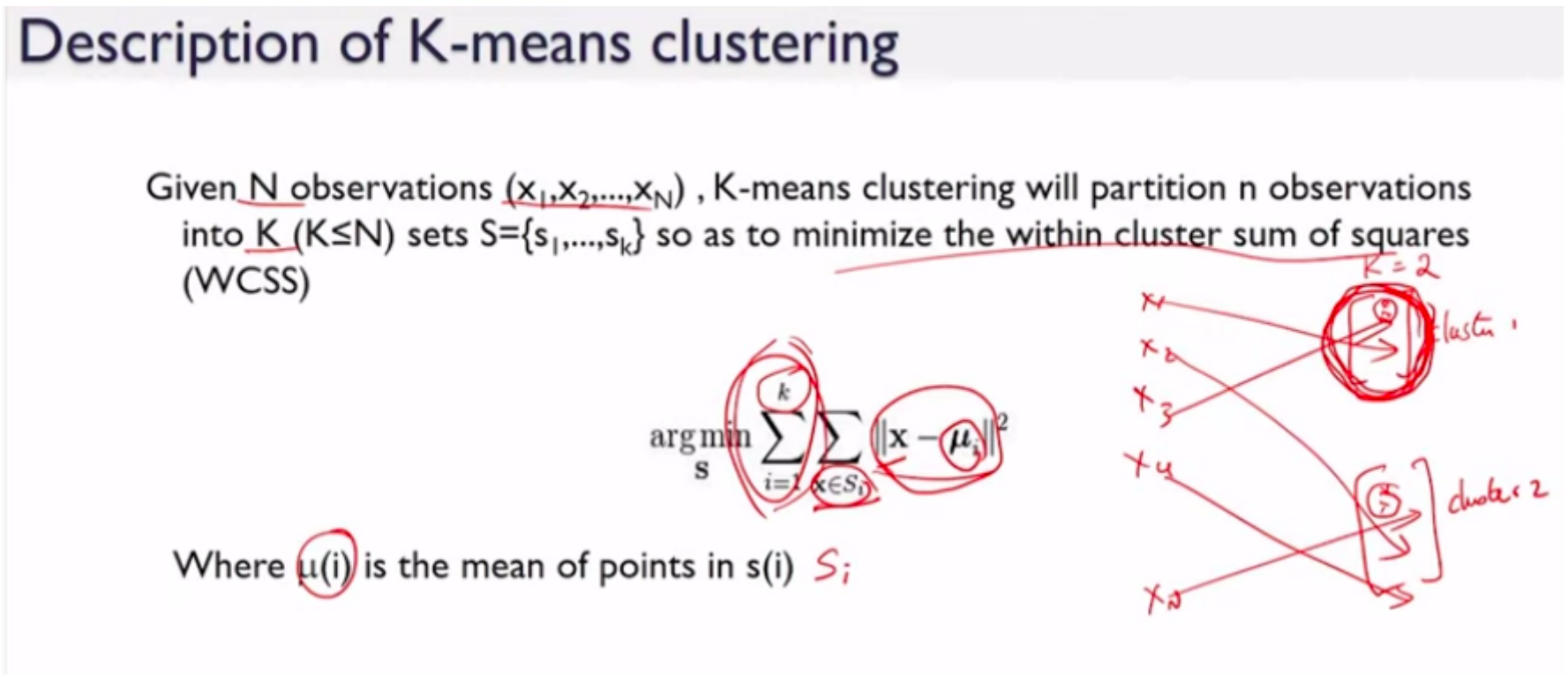
1. Primarily.., after categorization or classification into diff groups..
2. Next, looking at, what characteristics these groups pop-out from each group to understand and label them

to put it together..

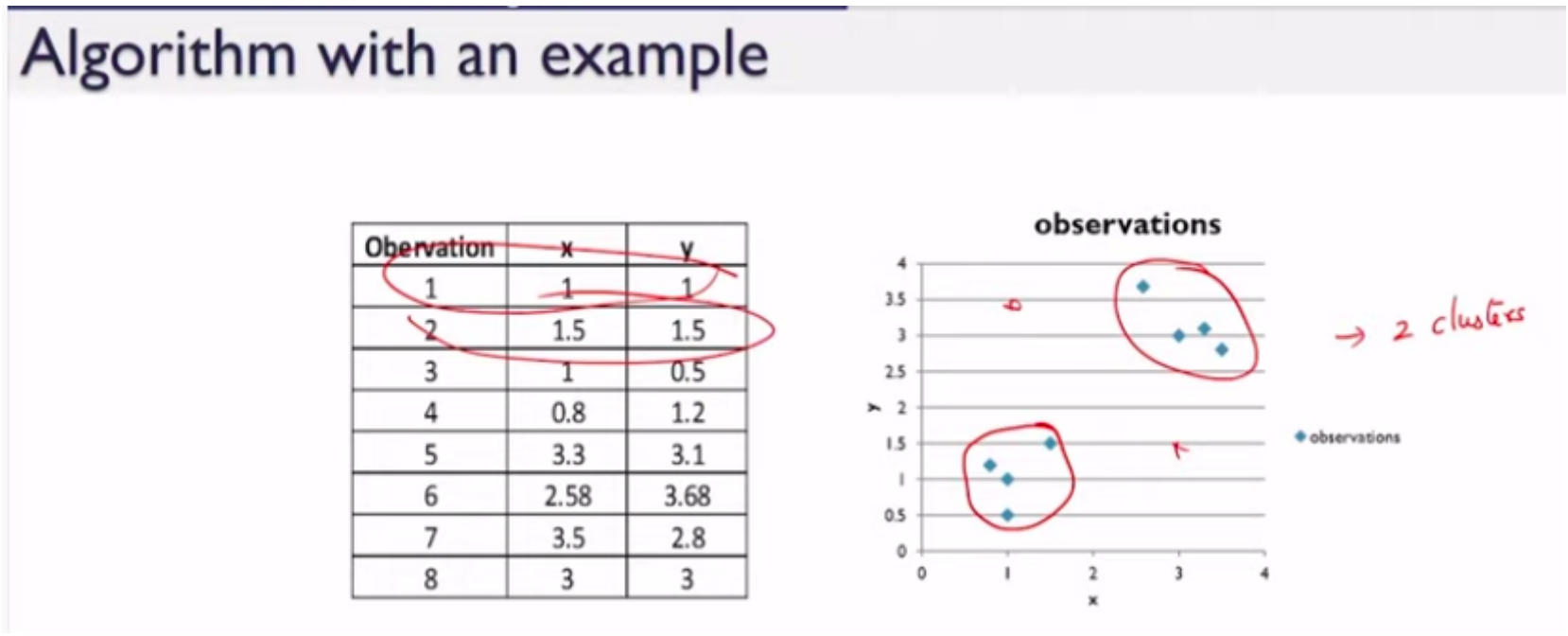
> K-Means is one of the simplest un-supervised algo, where if you give no. of clusters/groups that exist, then it partitions N observations into those categories.

As any machine learning algo contains some optimization approach, this too had it. It does **as the summation expression works**..
Can be interpreted as..

> The first summation works on each grouped set(S_i), and the second summation works on all such K sets.



Let's look at some example.. *(for the sake of illustration, considering in 2-dimensions for identifying clusters, but that becomes non-obvious for higher dimensions)* ..



Ok, that's fine through a visual manner.. but how do we do it computationally..?? *as follows..*

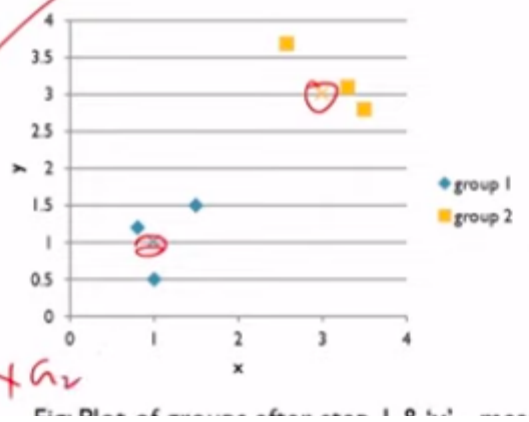
Example continued…

Step 1: Randomly choose two points as the cluster centers

	Individual	Mean x	Mean y
Group 1	1	1	1
Group 2	8	3	3

Step 2: Compute the distances and group the closest ones

Observation	distance 1	distance 2	Group
1	0	2.8284271	1
2	0.7071068	2.1213203	1
3	0.5	3.2015621	1
4	0.2828427	2.8425341	1
5	3.1144823	0.3162278	2
6	3.111077	0.7992496	2
7	3.0805844	0.5385165	2



By now, we classified the observations into groups, it would be better to re-choose the cluster center right..!!, so that we can represent whole cluster by that one point.

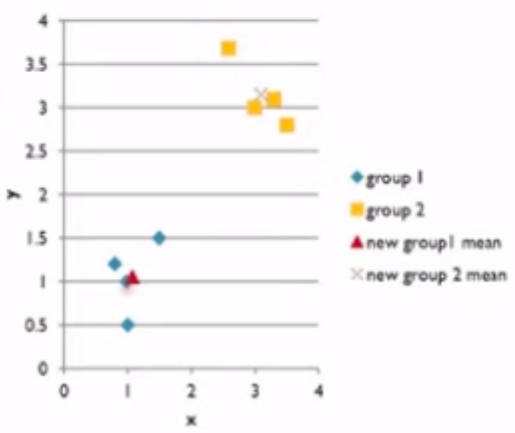
Example continued…

Step 3: Compute the new mean and repeat step 2

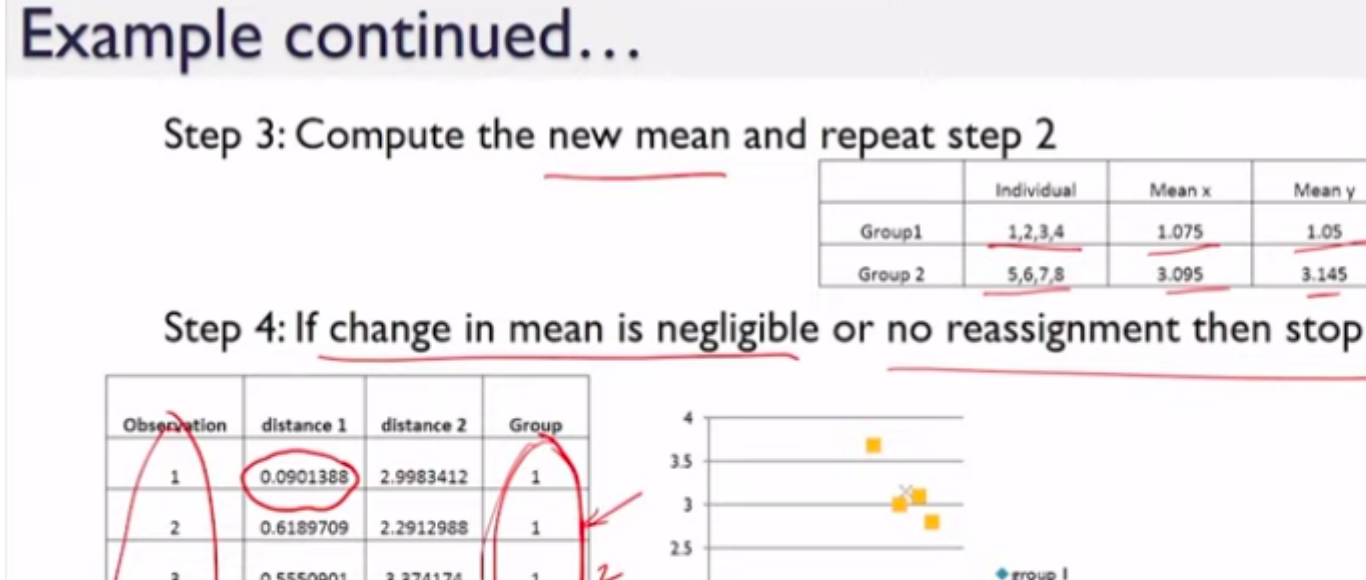
	Individual	Mean x	Mean y
Group1	1,2,3,4	1.075	1.05
Group 2	5,6,7,8	3.095	3.145

Step 4: If change in mean is negligible or no reassignment then stop the process

Observation	distance 1	distance 2	Group
1	0.0901388	2.9983412	1
2	0.6189709	2.2912988	1
3	0.5550901	3.374174	1
4	0.3132491	3.0083301	1
5	3.0254132	0.2098809	2
6	3.0301691	0.7425968	2
7	2.9905058	0.5320244	2
8	2.7400958	0.1733494	2

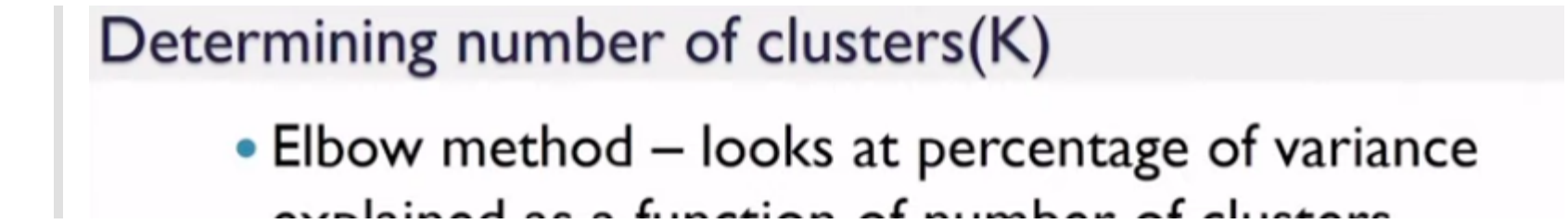


To sum it up..



But, in all the cases, the data won't be of two-dimensional right... then how about handling those cases..?

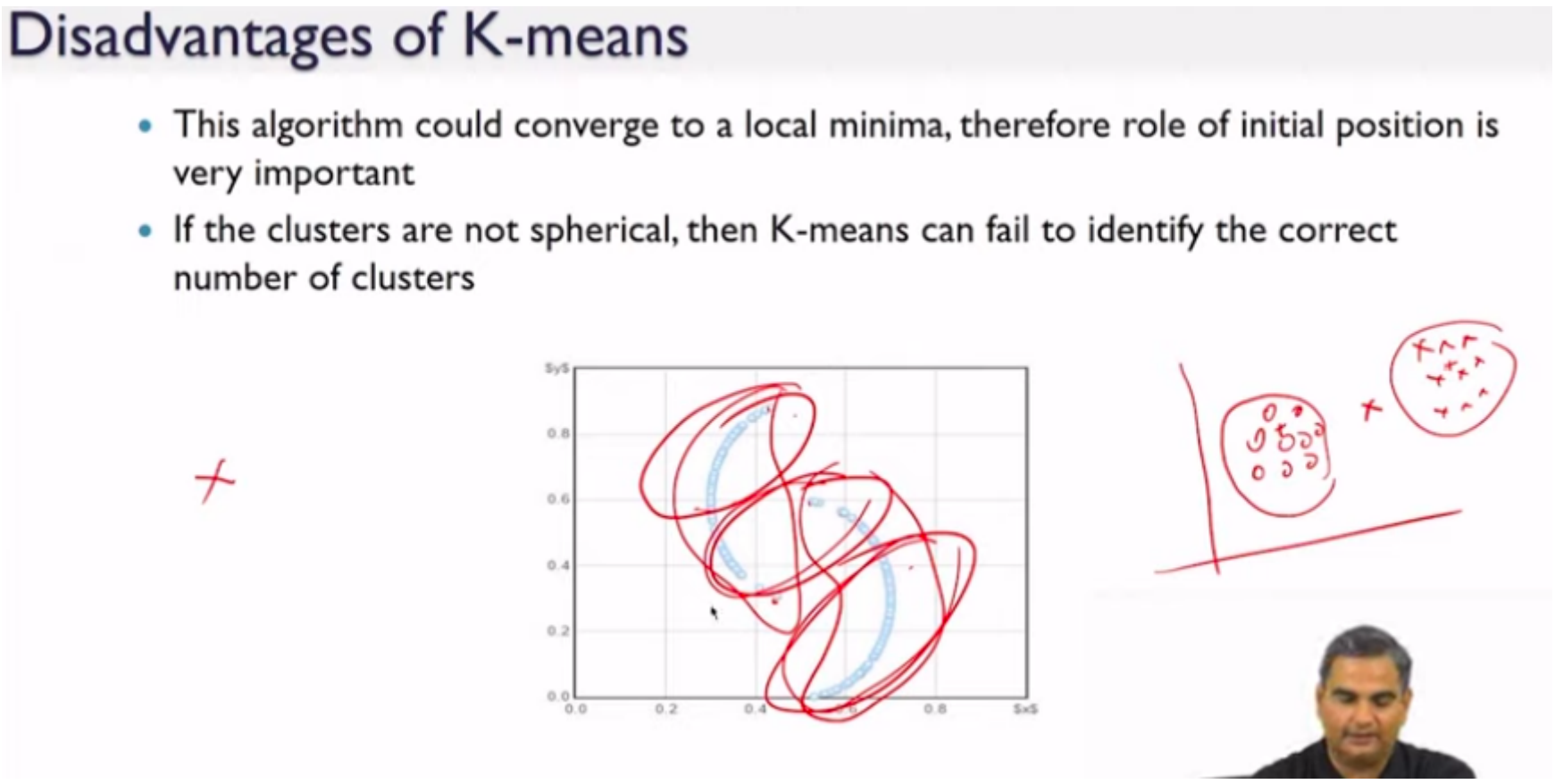
> via **Elbow Method**.



Disadvantage of K-Means..

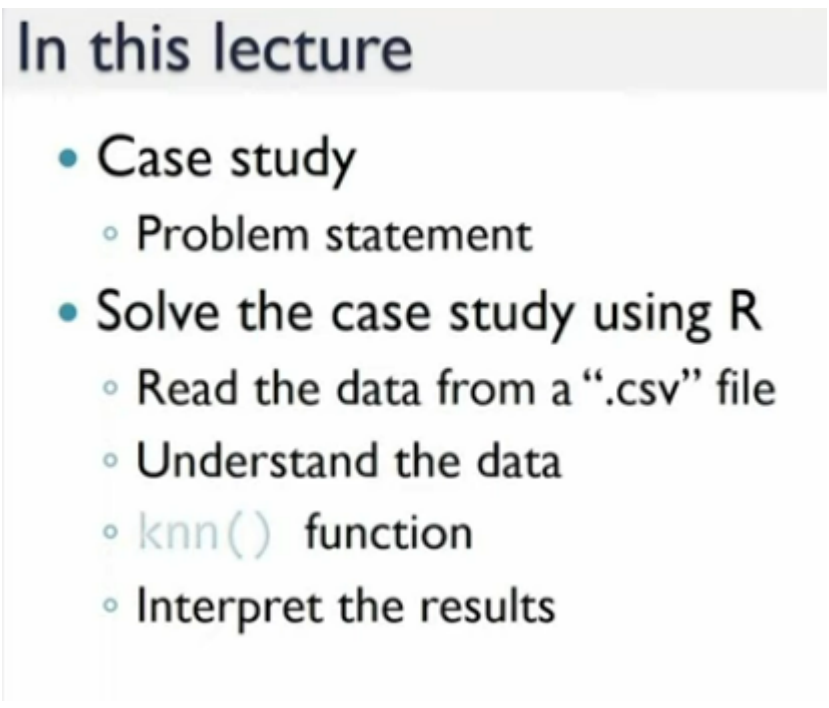
- Could converge to **Local Minima**, so initial choosing of centers MATTERS A LOT (~~ Highly sensitive)

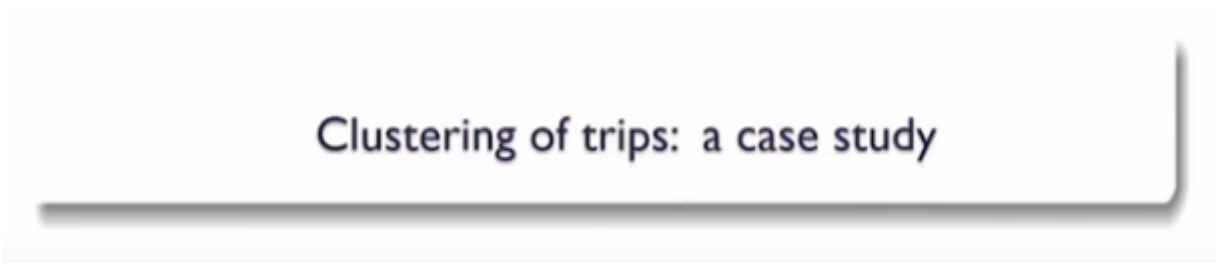
> i.e., If centers are not chosen carefully *(Say one choosen in the center of [Say 2] clusters and one farther than these.. then those two individual clusters becomes 1 cluster) [Refer to the sir's representation in below figure]* then classification goes wrong.



That's it, the end of the theory part of this course

Let's see the practical implementation of this algorithm with a case study..





Automotive Service Study: Problem statement

An automotive service chain is launching its new grand service station this weekend. They offer to service a wide variety of cars. The current capacity of the station is to check 315 cars thoroughly per day.

As an inaugural offer, they claim to freely check all cars that arrive on their launch day, and report whether they need servicing or not!

Unexpectedly, they get 450 cars. The service men won't work longer than the working hours but the

Reading the data

- Data for this case study is provided to you file with name "tripDetails.csv"
- To read the data from a ".csv" file we use `read.csv()` function

.png)

Reading the data

- Data for this case study is provided to you file with name "tripDetails.csv"

```
#Reading the data
tripDetails = read.csv("tripDetails.csv",
                       row.names=1)
```

In [8]:

```
tripDetails = read.csv("resources/datasets/tripDetails.csv", row.names=1)  # For the given dataset, the 1st row cont
tripDetails      # View(tripDetails) GUI's work in command-line..
```

TripLength	MaxSpeed	MostFreqSpeed	TripDuration	Brakes	IdlingTime	Honking
21	51	14	93	307	27	112
148	130	106	156	226	5	114
18	38	16	100	351	26	107
22	43	48	36	17	4	5
183	108	90	171	88	5	29
18	43	13	64	136	25	21
20	37	15	85	121	26	23
21	38	14	69	114	25	20
181	99	108	155	86	5	25
174	100	92	133	106	5	34
177	130	85	152	210	5	128
17	67	41	30	33	4	17
19	42	14	102	429	27	97
18	39	39	37	20	4	5

TripLength	MaxSpeed	MostFreqSpeed	TripDuration	Brakes	IdlingTime	Honking
17	39	16	87	115	25	26
193	122	101	150	183	6	94
17	61	43	26	40	5	23
20	35	43	42	15	4	5
21	48	15	88	384	27	98
21	39	15	92	131	24	23
181	111	100	147	88	6	33
20	35	15	88	128	26	22
22	35	14	79	343	28	100
20	43	16	91	361	26	96
20	41	14	87	405	27	84
21	37	16	94	120	27	20
21	44	45	42	17	5	5
19	48	16	86	311	24	90
176	126	118	162	174	6	150
151	120	105	128	195	5	108
...	...	...	...	...	...	...
19	43	40	34	16	4	6
210	99	90	158	195	4	155
163	119	99	142	243	5	135
21	41	15	87	308	26	102
21	47	38	31	18	4	5
18	69	44	32	39	6	21
19	64	38	32	36	5	18
159	138	84	151	94	4	32
22	38	14	92	137	22	21
20	36	14	106	149	24	22
21	47	17	84	345	24	90
16	42	42	34	17	5	5
23	41	44	36	20	5	5
20	60	40	30	37	5	20
176	136	99	138	206	6	118
183	106	102	142	211	5	115
163	116	93	151	87	4	26
21	49	43	33	19	4	5
19	62	39	33	36	4	17
164	105	101	166	96	4	30
154	128	107	152	201	5	148
179	118	100	139	168	4	121
196	107	89	142	79	5	31
16	61	42	35	38	5	17
20	59	42	28	41	6	20
21	42	13	92	360	24	98
183	126	89	149	189	5	130
22	44	12	72	349	24	93
163	135	91	129	202	5	144

.png)

In [9]:
```
str(tripDetails)
```

```
'data.frame':   91 obs. of  7 variables:
 $ TripLength   : int  21 148 18 22 183 18 20 21 181 174 ...
 $ MaxSpeed     : int  51 130 38 43 108 43 37 38 99 100 ...
 $ MostFreqSpeed: int  14 106 16 48 90 13 15 14 108 92 ...
 $ TripDuration : int  93 156 100 36 171 64 85 69 155 133 ...
 $ Brakes       : int  307 226 351 17 88 136 121 114 86 106 ...
 $ IdlingTime   : int  27 5 26 4 5 25 26 25 5 5 ...
 $ Honking      : int  112 114 107 5 29 21 23 20 25 34 ...
```

.png)

In [11]:
```
summary(tripDetails)  # Gives the five-point summary .. as same as for the box-plot / candle-stick plot..
```

```
  TripLength        MaxSpeed      MostFreqSpeed     TripDuration   
Min.   : 16.00   Min.   : 35.00   Min.   : 12.00   Min.   : 22.00  
1st Qu.: 20.00   1st Qu.: 42.00   1st Qu.: 15.50   1st Qu.: 34.50  
Median : 21.00   Median : 54.00   Median : 42.00   Median : 88.00  
Mean   : 70.77   Mean   : 70.36   Mean   : 50.65   Mean   : 87.37  
3rd Qu.:163.00   3rd Qu.:105.50   3rd Qu.: 89.00   3rd Qu.:133.00  
Max.   :210.00   Max.   :138.00   Max.   :118.00   Max.   :171.00  
    Brakes         IdlingTime       Honking      
Min.   : 14.0   Min.   : 4.00   Min.   :  4.00  
1st Qu.: 36.5   1st Qu.: 5.00   1st Qu.: 20.00  
Median :100.0   Median : 5.00   Median : 25.00  
Mean   :135.4   Mean   :11.59   Mean   : 49.92  
3rd Qu.:198.0   3rd Qu.:24.00   3rd Qu.: 97.50  
Max.   :429.0   Max.   :32.00   Max.   :155.00  
```

NOTICE that... the dataset given to us, hasn't labelled as **Short-trip**, **Long-trip**..,, so if knowing these are the target from the dataset, then the un-supervised K-means would be the good choice.

K-means clustering

- Given the dataset of trip details, Mr. Sam's job is to segregate these trips into clusters
  - We seek an answer through k-means clustering
- Using k-means clustering on data
  - k-means clustering in R can be applied on data using "`kmeans()`" function

kmeans()

```
object = kmeans(x, centers, iter.max = 10, nstart = 1)
```

Arguments

x	numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
centers	either the number of clusters, say k, or a set of initial (distinct) cluster centers. If a number, a random set of (distinct) rows in x is chosen as the initial centers.
iter.max	the maximum number of iterations allowed.
nstart	if centers is a number, how many random sets should be chosen?
object	an R object of class "kmeans", typically the result "ob" of ob < − kmeans(..).

As, K-Means is an
Iterative algorithm

In [16]:
```
# Implementing kmeans.
tripCluster = kmeans(tripDetails, 3)  # with 3 Clusters...
#tripCluster  # See below image with interpretations
```

```
K-means clustering with 3 clusters of sizes 15, 31, 45

Cluster means:
  TripLength MaxSpeed MostFreqSpeed TripDuration    Brakes IdlingTime  Honking
1   20.06667 38.73333      14.53333     87.33333 127.66667  25.266667 22.00000
2   19.83871 52.80645      41.67742     32.12903  27.16129   4.709677 12.51613
3  122.75556 93.00000      68.86667    125.44444 212.62222  11.777778 85.00000

Clustering vector:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 3  3  3  2  3  1  1  1  3  3  3  2  3  2  1  3  2  2  3  1  3  1  3  3  3  1
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
 2  3  3  3  2  1  2  3  2  3  3  3  2  3  3  2  2  1  3  1  3  2  3  2  1  2
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
 1  2  3  3  1  2  3  2  2  2  3  3  3  2  2  2  3  1  1  3  2  2  2  3  3  3
79 80 81 82 83 84 85 86 87 88 89 90 91
 2  2  3  3  3  3  2  2  3  3  3  3  2

Within cluster sum of squares by cluster:
[1]    4353.200    8837.613 1038860.978
 (between_SS / total_SS =  53.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Each row value for a column is for **each cluster**

Variance within each cluster.
Lesser the variance the good are the clusters

Legend

`1 2 3 4 ..` Indices

`3 3 3 2 ...` to which cluster it became part of

The biggest problem with the K-means is that.. **Choosing the no. of clusters** -- that's where **elbow method** comes to rescue..

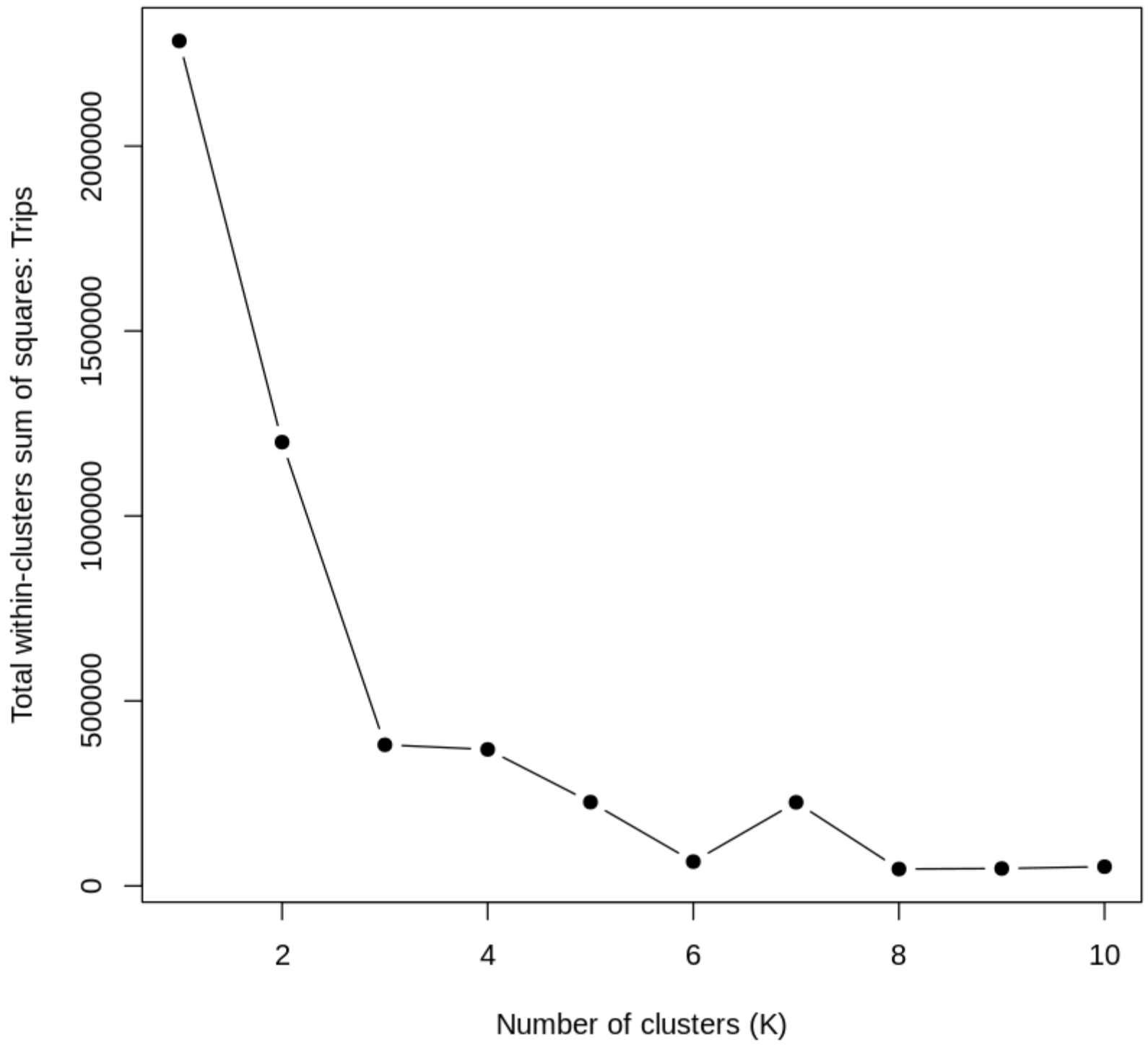
Elbow method: Optimal method to calculate `k`

Strategy:

- In a loop apply the `kmeans` over a range of cluster_sizes and take that metric of `sum of squares`, and plot it.
- The elbow point tells the optimal cluster-value.

In [32]:
```
k.max=10                  # Assuming the max-clusters can be 10..
wss <- rep(NA, k.max)     # Allocating a vector of the req. size x 1 and filling with NA's.. wss: within_sum_of_squa
#num_clusters <- list()    # A empty list to store the clusters....??
for(i in 1:k.max)
{
    result_obj <- kmeans(tripDetails, i)   # Applying the kmeans ..
    wss[i] <- result_obj$tot.withinss      # Calculating within-sum_of_squares..
}
```

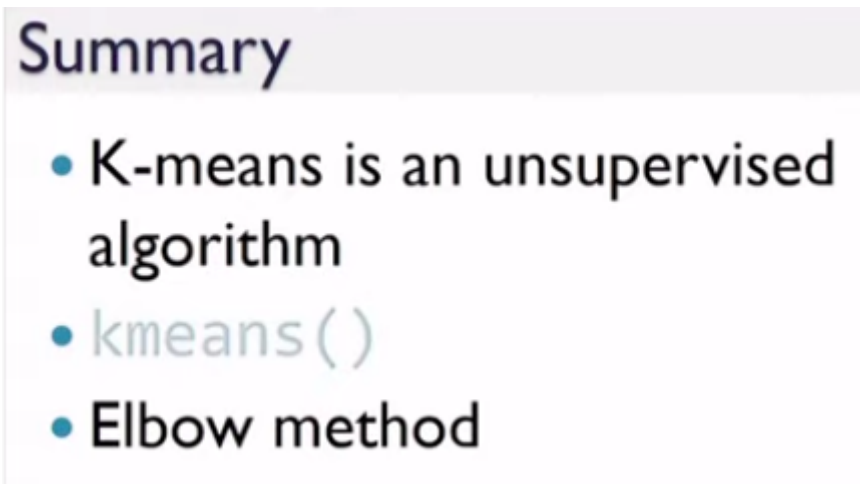
In [36]:
```
plot(1:k.max, wss,
     type="b", pch=19,
     xlab = "Number of clusters (K)",
     ylab = "Total within-clusters sum of squares: Trips")
```



The elbo-point.. seems to be 3, hence that becomes the optimal point.

Why shouldn't we take other ones like.. 4, 5, 6..??

> *[May be..]* we'll get too many clusters (probably may become difficult to make labellings to those clusterings)



END

In [ ]: