Applied Analytics: Frameworks and Methods 2

COLUMBIA UNIVERSITY
School of Professional Studies

No Phone

No Photographing

No Audio Recording

No Video Recording

Lecture Content and Materials Should Not Be Posted or Shared

Online or Offline Without Explicit Permission

COLUMBIA UNIVERSITY
School of Professional Studies

Return to In-Person Instruction on Monday January 31st, 2022

Dear SPS Students,

As our two week remote learning period comes to a close we want to ensure that all students are prepared for the return to in-person instruction. All courses that were originally set to run in a traditional "face to face" teaching modality, will resume in-person instruction on **Monday, January 31st, 2022**. In order to make this transition as smooth as possible we have outlined a few recommended action steps below.

Prepare for your return to campus

-Compliance/ReopenCU App - All students must meet the latest requirements and compliance protocols for COVID-19 before returning to campus. This includes uploading vaccination/booster documentation, completing necessary testing, signing the CU Health Compact, and completing your daily attestation on the ReopenCU App. Only students with a green pass will be allowed to enter buildings on Columbia campus.

-Appropriate Face-coverings - Ensure that you have the proper mask. Effective Tuesday, January 18, cloth masks will not be acceptable protection in indoor Columbia University settings. All members of the Columbia community are now required to wear a surgical mask, KN95 or KF94 masks. If you forget to wear a surgical mask or run out of them, there will be some available to you at various locations on campus.

-Booster Mandate- Students should follow these instructions for uploading their vaccine booster documentation. All booster documentation should be uploaded by January 31st.

-Testing- All new students must complete their necessary gateway testing. The original gateway testing period for 2022 has been extended from January 31 to February 4, 2022. After February 4, **red passes** will be applied to all those who are not compliant with the gateway testing requirement.

If you have any questions about the location of your next class session please reach out to your instructor right away.

Sincerely,

This is a message from **School Of Professional Studies**

Time and Location

COLUMBIA UNIVERSITY
School of Professional Studies

Day/Time: Tuesday, 8:10am-10:00am

Modality: ON-CAMPUS

Location: 417 Mathematics Building

- 1/18 and 1/25 (lecture 1 and 2)
  - classes meet online
  - use zoom link under "Zoom Class Session" page on canvas
- Starting 2/1 (lecture 3 and all lectures follow)
  - classes meet on campus
  - 417 Mathematics Building

Applied Analytics: Frameworks and Methods 2

Basic and Advanced Clustering

Outline

- The Concept of Clustering
- Market Segmentation
- Process of Clustering
- Clustering Techniques
  - Hierarchical Clustering
  - k-Means
  - Model-based Clustering
- Input to Predictive Modelling
- R Illustrations:

  Simple Illustration of Cluster Analysis (Intro)

  Clustering for Segmentation

  Cluster then Predict

  Simple Illustration of Cluster Analysis

  Cluster Analysis: An Iris Story

Cluster Analysis

- Clustering is a class of techniques used to classify objects or cases into relatively homogeneous groups called clusters.
- Clustering groups objects that are similar in a single cluster and those that are dissimilar in other clusters.
- Clustering groups observations based on similarity on a set of variables.

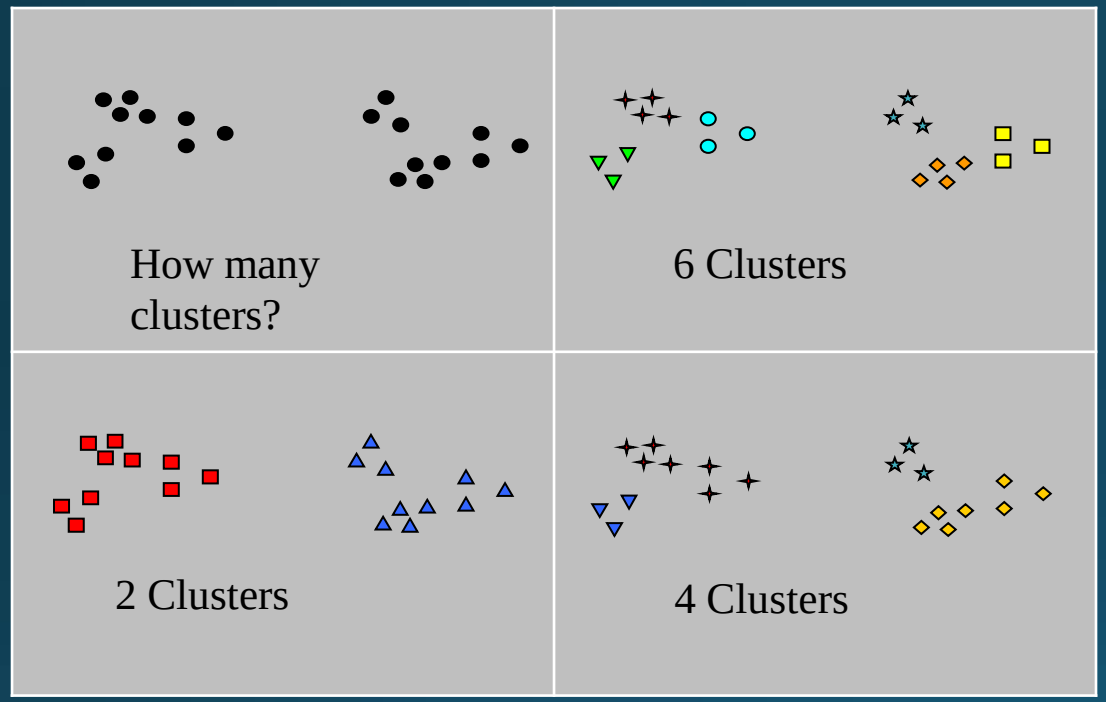
Cluster Analysis

- We are all accustomed to organizing our belongings. Clustering is about organizing data.

Cluster Analysis

- Classification technique used to group together observations typically based on spatial distance
- No statistical basis to this grouping
- Used primarily as an exploratory technique
    - So, no hypotheses
- No unique solution
- Number of clusters is usually guided by theory and inter-cluster distance

Notion of a Cluster can be Ambiguous



Find lookalike groups within a population

Combine science and business judgment

Tan, Steinbach, Kumar (2004)

Market Segmentation

Market Segmentation

- *"You can have the Ford in any color, as long as it is Black",*

Henry Ford

Available Sizes: One size fits all

Available Styles: One style fits all

Available colors: One color fits all

One Size Doesn't Fit All

Suitable for

- Girls and Boys
- Women and Men
- Teens and Senior Citizens
- Americans and Non-Americans

Market Segmentation

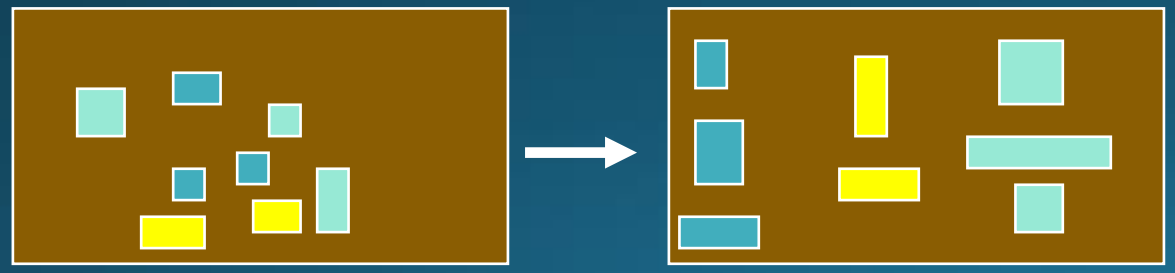
- In general
  - Customers are happier when they get what they like
  - Generic one-size offerings lead to lower satisfaction
- On the other hand, cost of catering to individual customers is very high
- Segmentation is a compromise between serving individual customers' needs and offering all customers the same product

Market Segmentation

- Market Segmentation involves dividing a market into distinct groups with
  - distinct needs, characteristics, or behavior, and
  - who respond differently to a marketing action

1. Break market down

2. Group into segments

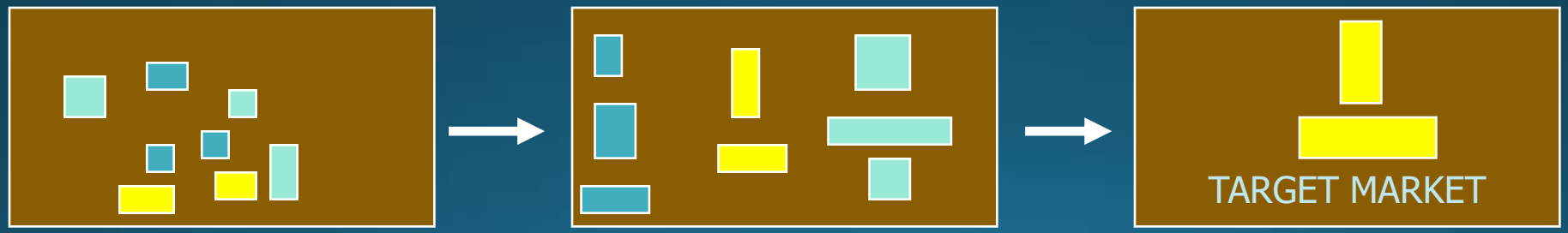


Segmentation and Targeting

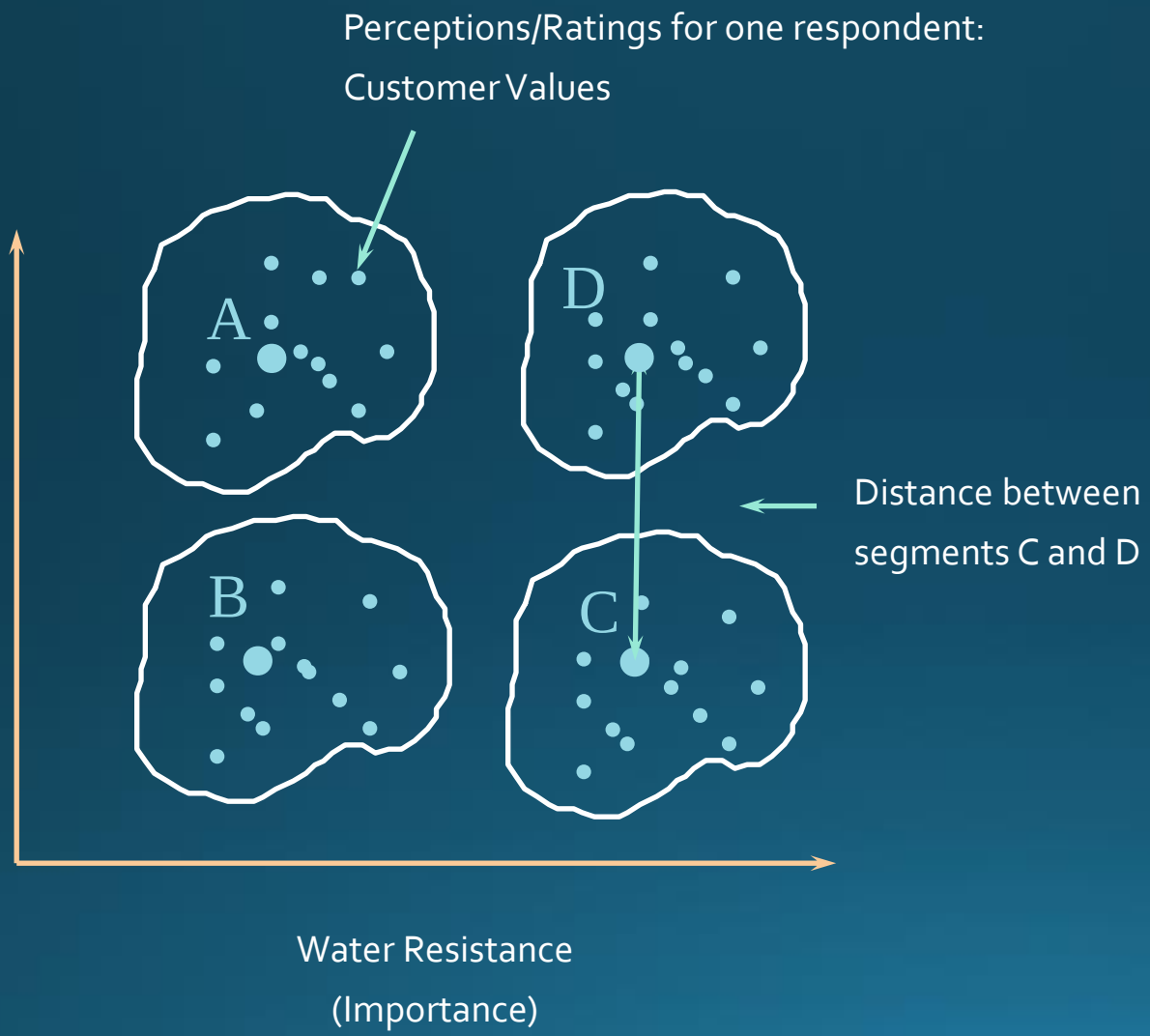
1. Break market down

2. Group into segments

3. Choose target market



Segmentation (for Carpet Fibers)



A,B,C,D:

- Location of segment centers

Typical members:

A: schools

B: light commercial

C: indoor/outdoor carpeting

D: health clubs

Coffee buyers?

COLUMBIA UNIVERSITY
School of Professional Studies

Cluster Analysis

Profile Clusters

Market Segmentation Approaches

- Heuristic Approach
  - Use observable variables to segment the market
  - Variables selected to segment are expected to be the ones on which customer needs vary
  - This is the traditional approach and does not involve cluster analysis
- Needs-based Clusters
  - Use data on customer needs to generate clusters of individuals that differ in their needs
  - Profile clusters on observable variables (e.g., gender, age, location)

Market Segmentation – Heuristic Approach

COLUMBIA UNIVERSITY
School of Professional Studies

- Here are some ways markets may be segmented using the Heuristic Approach
- Consumer markets
  - Geography (region, country, state, city)
    - E.g., Walmart, McDonalds,
  - Demographic (Age, gender, income, lifecycle, occupation)
    - E.g., Gap, Abercrombie and Fitch, American Eagle, Toyota, Walmart, Canon Powershot
  - Psychographic (personality traits such as conscientiousness, openness to Target vs. experience; athleticism, innovativeness)
    - E.g., GNC, Paragon, EMS
  - Behavioral (loyalty, purchase quantity, frequency)
    - All airlines, Costco, credit cards
- Business markets
  - Demographics (industry, company size)
  - Operating variables (technology, usage status)
  - Purchasing approaches
  - Situational factors (urgency, order size)
  - Personal characteristics (buyer-seller similarity, loyalty)

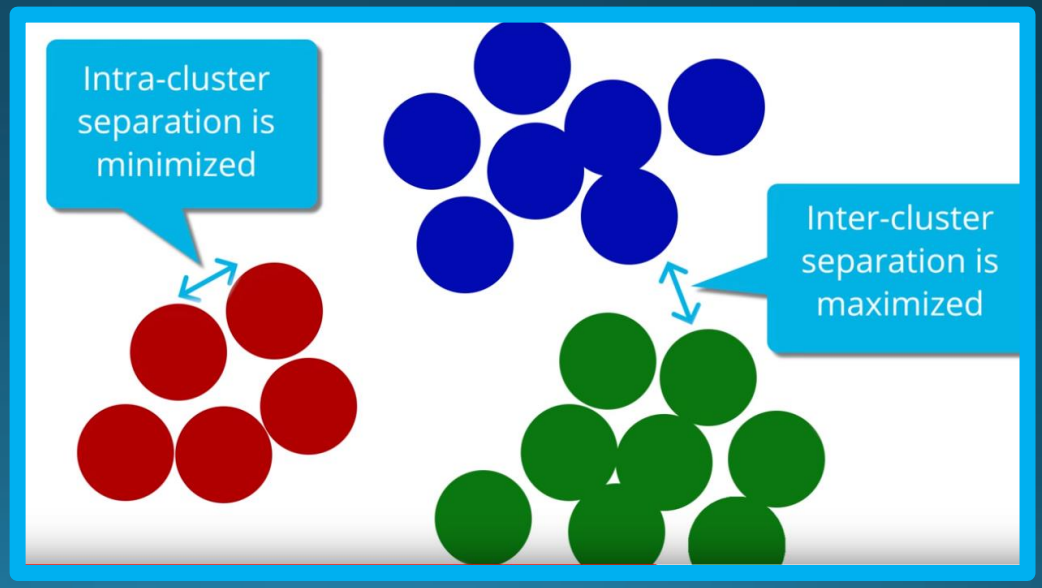
Needs-based Clusters

- Articulate a strategic rationale for segmentation (i.e., why are we segmenting this market?).
- Select a set of needs-based segmentation variables *most useful* for achieving the strategic goals.
- Select a cluster analysis technique for segmentation.
- Group customers into a defined number of different segments.
- Choose the segments that will best serve the firm's strategy, given its capabilities and the likely reactions of competitors.

Process of Cluster Analysis

Cluster Analysis

- Cluster analysis is a class of techniques used to classify objects or cases into relatively homogeneous groups called clusters.
- Objects in each cluster tend to be similar to each other and dissimilar to objects in the other clusters.
- Cluster analysis is also called classification analysis, or numerical taxonomy.



Cluster Analysis

1. Select variables for analysis
2. Prepare data
3. Compute similarity using a metric
4. Apply a clustering technique
5. Interpret results to determine number of segments (for some methods)
6. Evaluate, eliminate outliers, validate clustering scheme with another sample
7. Profile clusters and evaluate usefulness of resulting segments

Select variables for analysis

- Clusters are only as good as the variables chosen to define it.
- Software cannot offer guidance on the practical use of the variables.
- Consider relevance of variables chosen to the problem
  - In market segmentation, variables are chosen based on customer needs. For a fitness club, this could include, visit frequency, and workout goals.
  - On the other hand, is number of visits/week to the gym relevant in segmenting the market for Folgers coffee? What about variety seeking behavior? What about innovativeness?
- Use of multiple variables tapping the same dimension may overweight the dimensions' impact in defining clusters.
- Nature of variables selected influence choice of clustering algorithm

Prepare Data

- Transform format of variables
  - Clustering algorithms prefer all variables to be of the same class (e.g., numeric, factor)
- Impute missing data
  - Clustering algorithms use data on all variables. A missing observation on one variable will cause the entire row of data to be ignored for analysis. Therefore, it is important to impute missing values.
- Scale
  - For distance-based clustering methods, scale of the variable (e.g., seconds vs minutes) affects the weight assigned to the variable. Therefore, it is best to standardize variables.
- Redundant variables
  - Multiple variables measuring the same dimension should be replaced by the underlying factor or a representative variable.

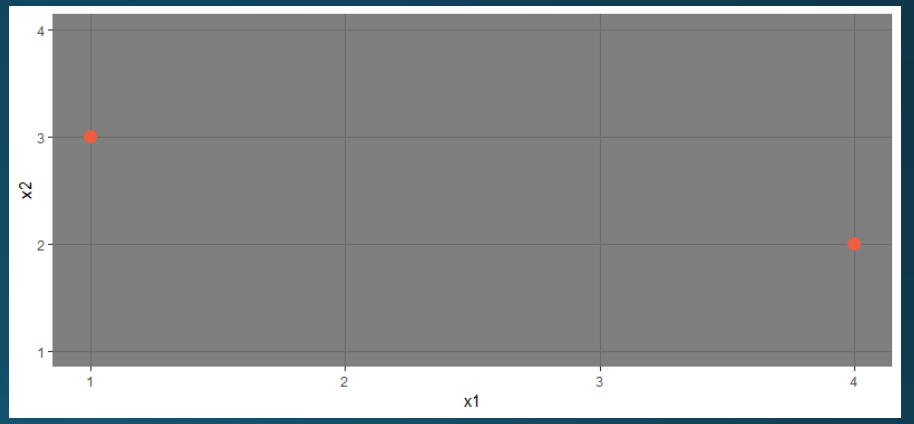
Compute Similarity

Similarity between observations is computed for distance-based methods but not for model-based methods.

- Distance-based methods
  - Find groups that minimize the distance between members within the group while maximizing the distance from members of other groups
  - Popular methods are Hierarchical clustering and k-means
  - Computation of similarity in observation is a pre-requisite
- Model-based methods
  - View the data as a mixture of groups sampled from different distributions
  - In this approach, similarity between observations is not computed

Compute Similarity

- Almost always distance-based measures (as opposed to a correlational measure)
- Distance-based similarity measures differ in how distance is computed
  - Euclidean distance
  - Maximum
  - Manhattan
  - Canberra (weighted Manhattan distance, Wikipedia)
  - Minkowski (Wikipedia)
  - Mahalanobis distance (adjusts for multicollinearity)



- Distance-based measures are sensitive to scale (e.g., minutes and seconds). Therefore, it is desirable to standardize.

Apply a Clustering Technique

- Hierarchical clustering
- k-means clustering
- Model-based clustering
- Latent class analysis

Interpret Results: Select Number of Clusters

- Domain knowledge
  - Whenever possible, the decision of number of clusters must be based on an understanding of the problem.
- Statistical criterion
  - Some algorithms require specifying number of clusters (e.g., kmeans and polCA) while others require one to infer number of clusters from the output (e.g., hclust)

Validation

- In the absence of an outcome variable, there is no "answer" to evaluate the clustering scheme
- It is best to validate the clustering scheme with another sample or a holdout sample
- To examine the potential bias from outliers, the cluster results with outliers can be compared to that without outliers.

Evaluate Usefulness

- The usefulness of results will depend on the meaningfulness of clusters in the context of the problem.

Clustering Techniques

Cluster Analysis Techniques

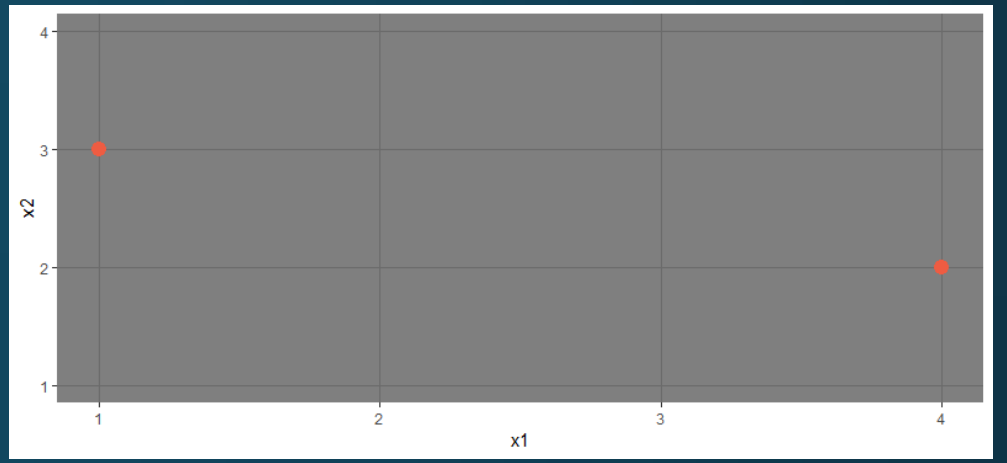
- Hierarchical clustering
- K-means clustering
- Model based clustering methods
  - Finite Mixture models

Hierarchical Clustering

Hierarchical Clustering

- Popular method that groups observations according to their similarity
- Distance-based algorithm
- Algorithm begins with each observation in its own cluster.
- Successively joins neighboring observations or clusters one at a time according to their distances from one another, and continues until all observations are linked
- Process of repeatedly joining observations is called the agglomerative method
- hclust()

Distance Measure



- Distance between observations may be measured as
  - Euclidean distance
  - Maximum
  - Manhattan
  - Canberra (weighted Manhattan distance, Wikipedia)
  - Minkowski (Wikipedia)
- dist(x = data, method = 'euclidean')
- Most distance metrics are only defined when observations are numeric. If dataset contains a mix of variable types, consider
  - library(cluster); daisy(x=data, method='euclidean')

Method	Distance
euclidean	3.162278
manhattan	4.000000
maximum	3.000000
canberra	0.800000
minkowski	3.162278

Clustering Method

- Distance between pairs of points is useful in identifying similarity between a pair of points.
- As observations are combined into clusters, distances must now be evaluated among clusters or between a point and a cluster.
- There are different approaches to grouping observations and these are called clustering methods.

Clustering Method

- Linkage methods generate clusters based on distance between observations
  - single
  - complete
  - average
- Variance methods generate clusters to minimize the within-cluster variance. Ward's minimum variance method is aimed at finding compact spherical clusters
  - ward.D2 (original Ward's clustering criterion, Ward, 1963)
  - ward.D
- Centroid methods compute distance between cluster centroids
  - median
  - centroid
- hclust(d, method = "complete")

Clustering Methods

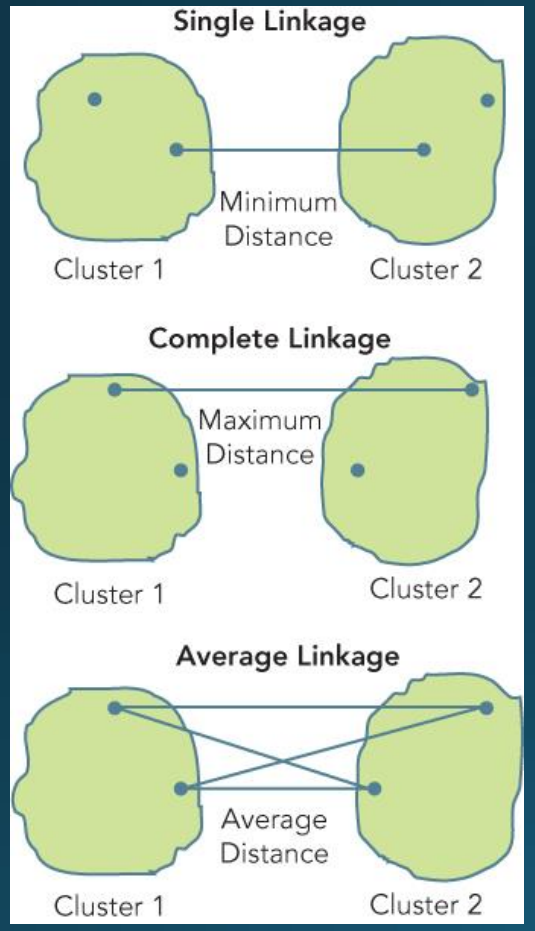
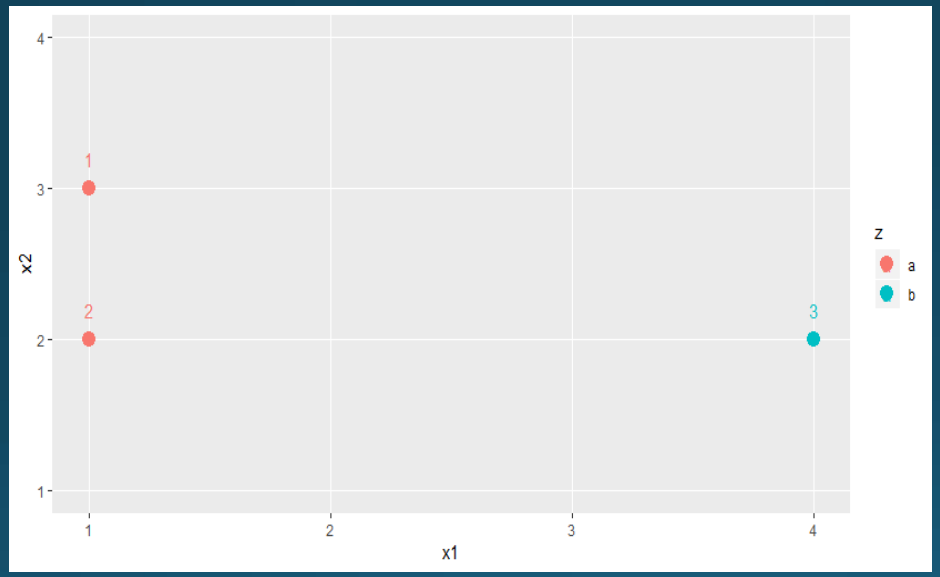


Illustration of Linkage Methods

Distances between "1-2" and "3" using each Linkage method



Method	Distance
single	3.000000
complete	3.162278
average	3.081139

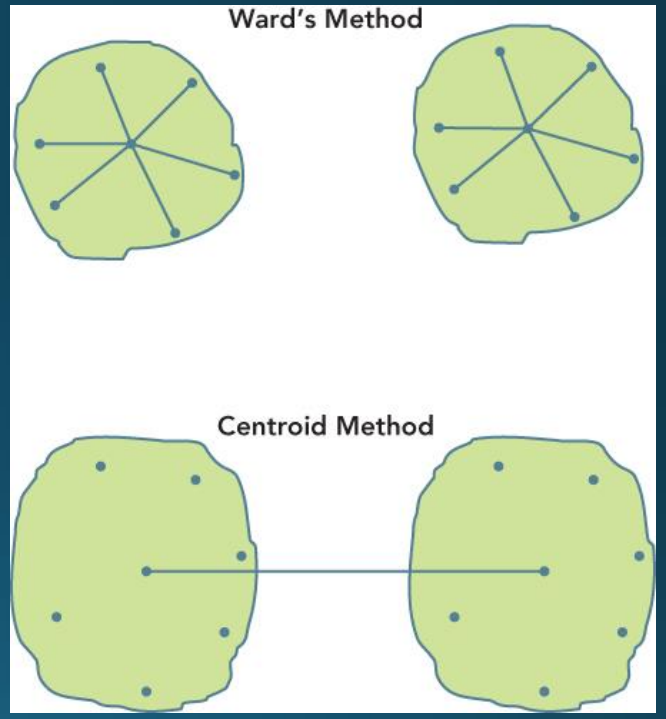


Illustration: Average Linkage Method for Clustering

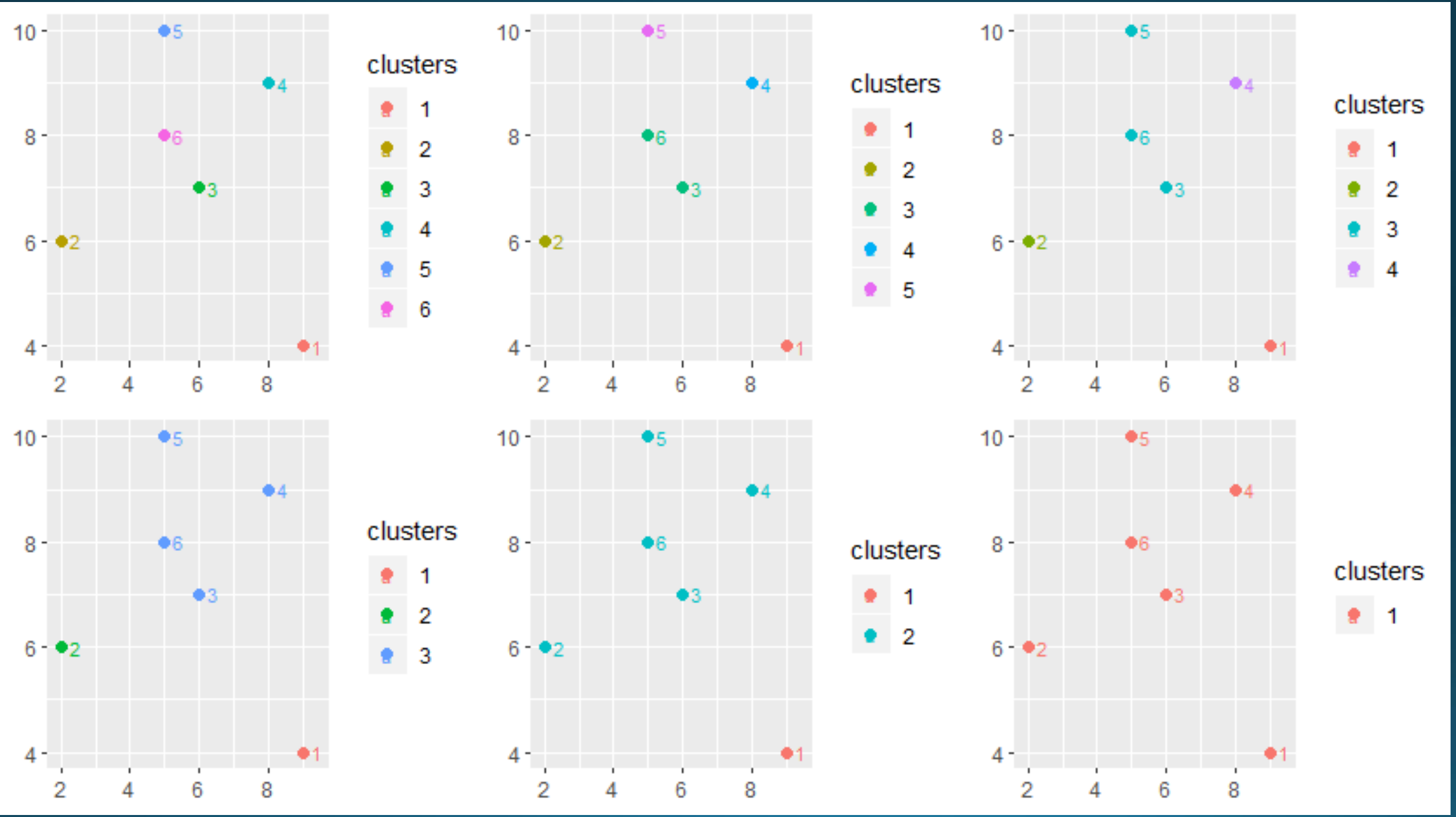
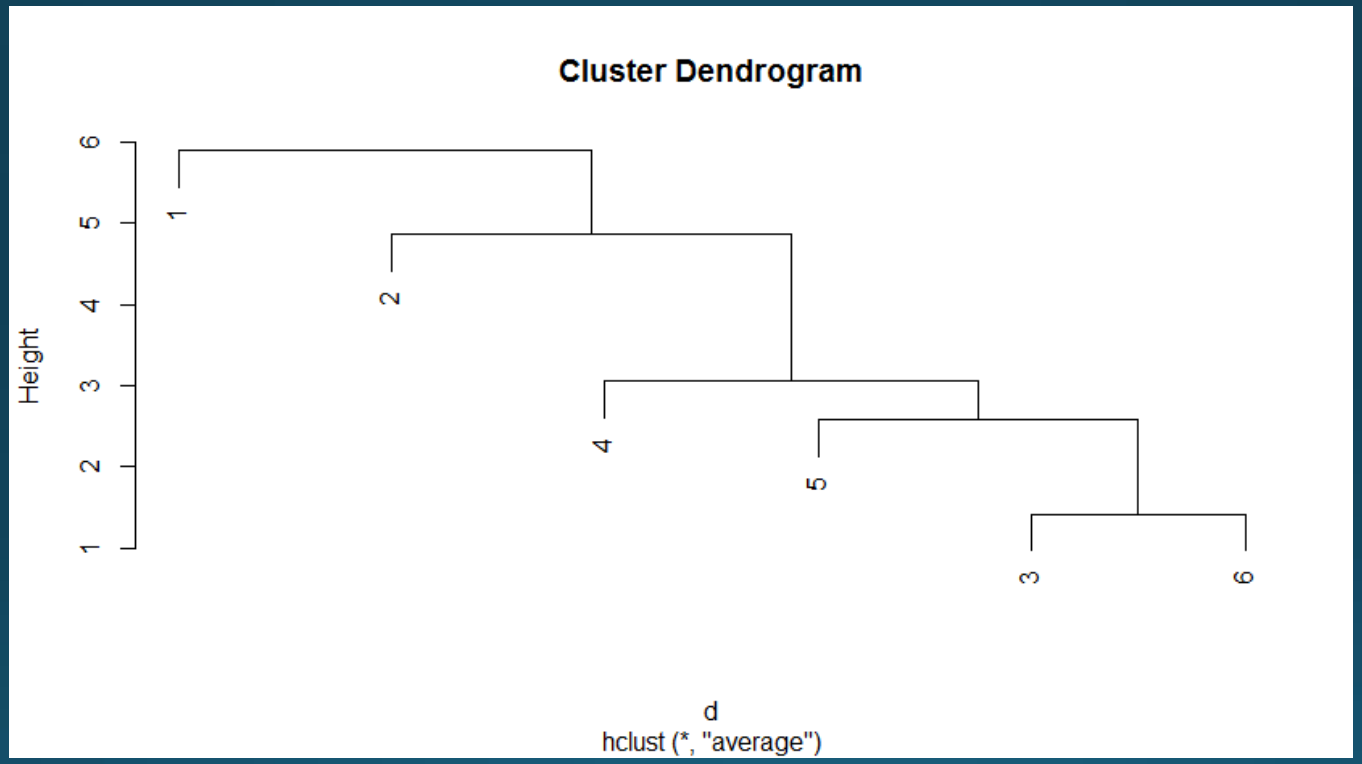


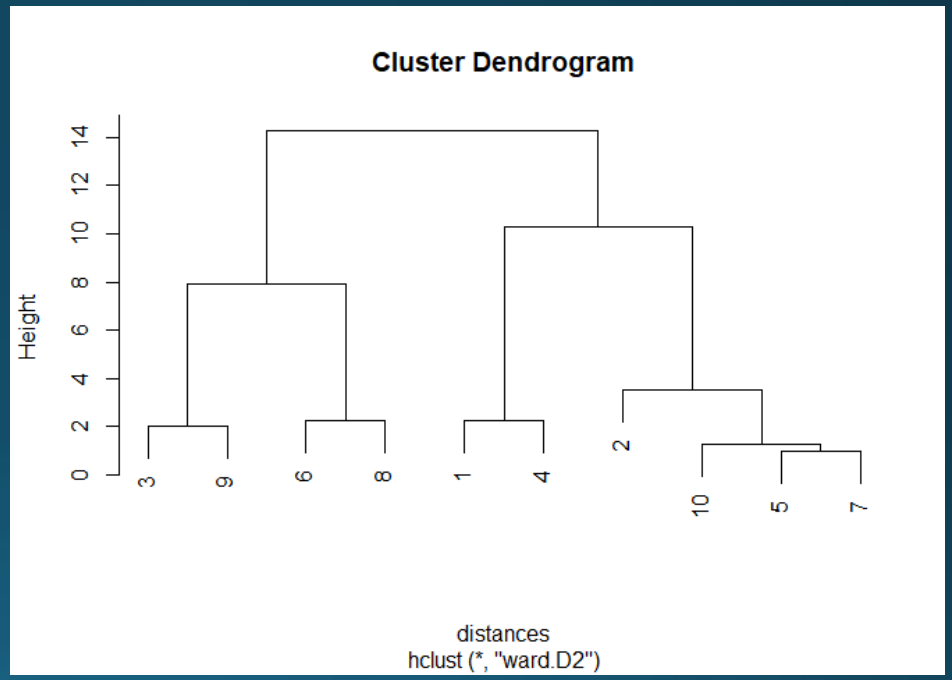
Illustration: Dendrogram



Determine Number of Clusters

- A hierarchical dendrogram is interpreted primarily by height and where observations are joined. The height represents the dissimilarity between elements that are joined.

- Cophenetic correlation coefficient (CPC) is a goodness of fit statistic for hierarchical clustering which assesses how well a dendrogram matches the true distance metric. It is interpreted similar to a correlation coefficient. CPC> 0.7 indicates relatively strong fit.



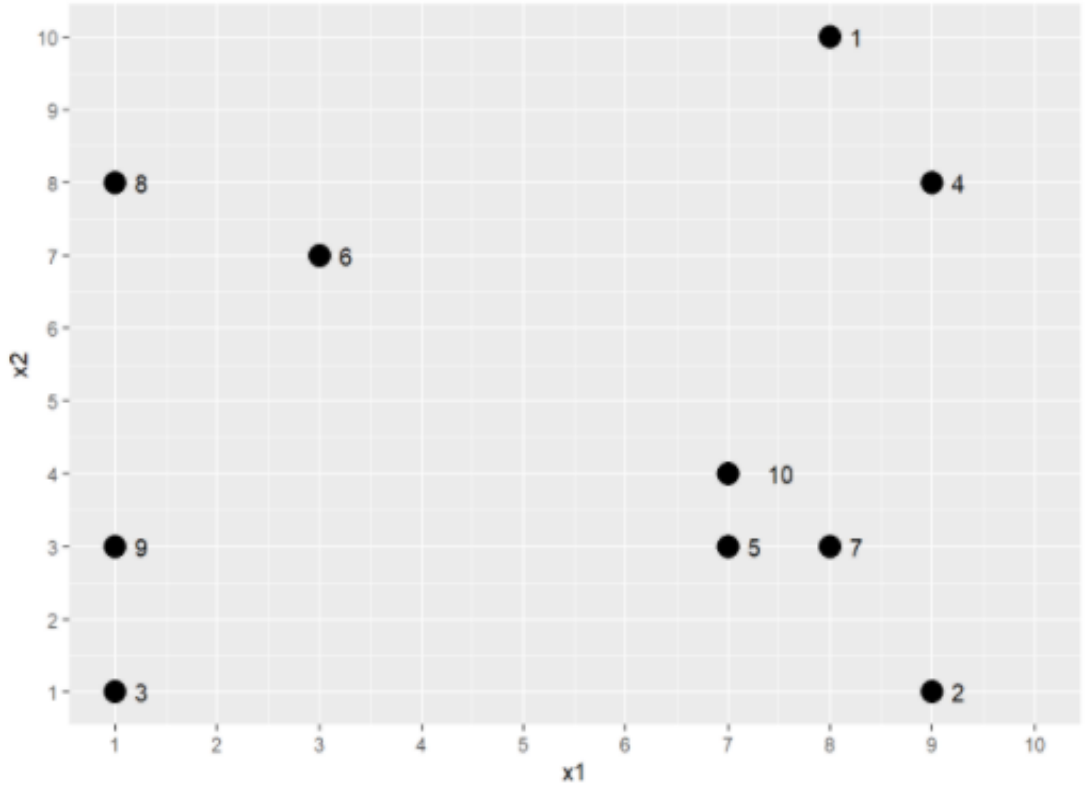
Create data

```
library(ggplot2)
set.seed(1031)
data = data.frame(x1=sample(1:10,10,replace = T),
                  x2=sample(1:10,10,replace = T))
rownames(data) = 1:10
data
```

```
##    x1 x2
## 1   8 10
## 2   9  1
## 3   1  1
## 4   9  8
## 5   7  3
## 6   3  7
## 7   8  3
## 8   1  8
## 9   1  3
## 10  7  4
```

Scatter Plot

```
ggplot(data=data,aes(x=x1,y=x2))+
  geom_point(aes(size=1.2))+
  scale_x_continuous(limits=c(1,10),breaks=1:10)+
  scale_y_continuous(limits=c(1,10),breaks=1:10)+
  guides(size=F)+
  geom_text(aes(label=rownames(data)),hjust=-1.5,vjust=0.5)
```



Hierarchical Clustering

Hierarchical Clustering

Compute distances

```
distances = round(dist(data,method = "euclidean"),2)
distances
```

```
##        1     2     3     4     5     6     7     8     9
## 2   9.06
## 3  11.40  8.00
## 4   2.24  7.00 10.63
## 5   7.07  2.83  6.32  5.39
## 6   5.83  8.49  6.32  6.08  5.66
## 7   7.00  2.24  7.28  5.10  1.00  6.40
## 8   7.28 10.63  7.00  8.00  7.81  2.24  8.60
## 9   9.90  8.25  2.00  9.43  6.00  4.47  7.00  5.00
## 10  6.08  3.61  6.71  4.47  1.00  5.00  1.41  7.21  6.08
```

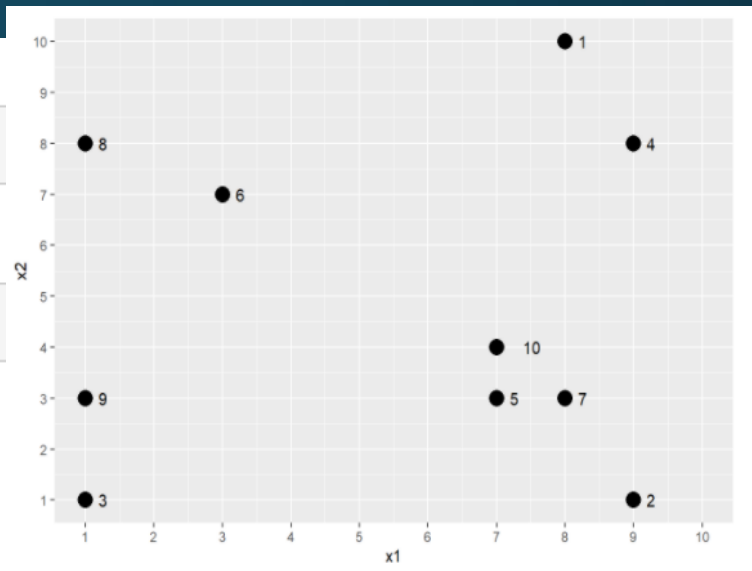
```
> length(distances)
[1] 45
```

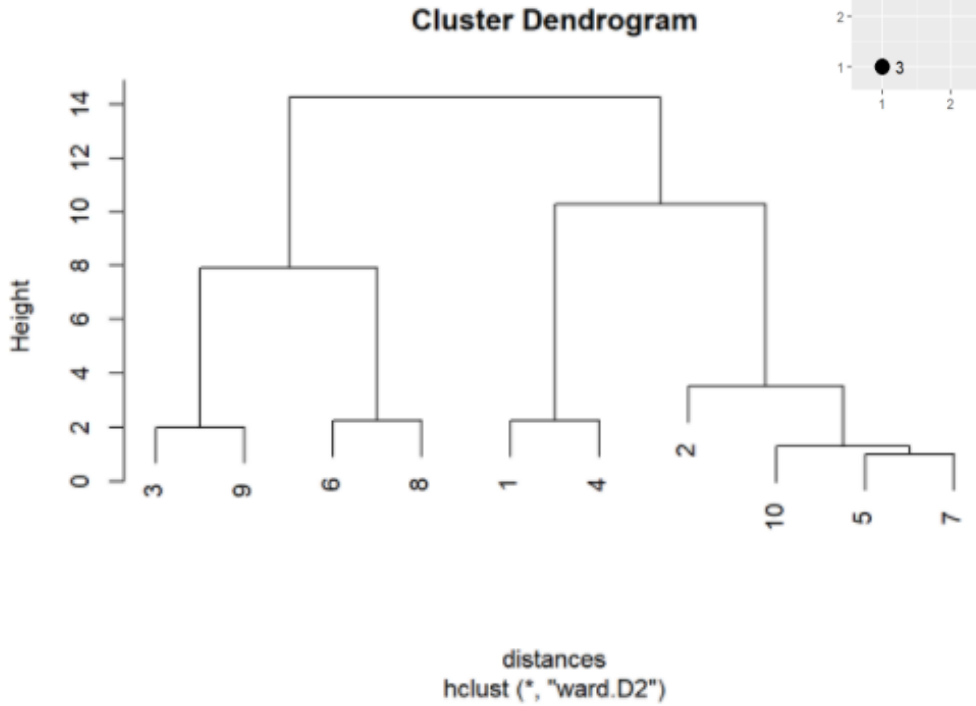
Cluster data based on distances

```
clust = hclust(distances,method = "ward.D2")
```

Dendrogram: Tree presentation of cluster results

```
plot(clust)
```





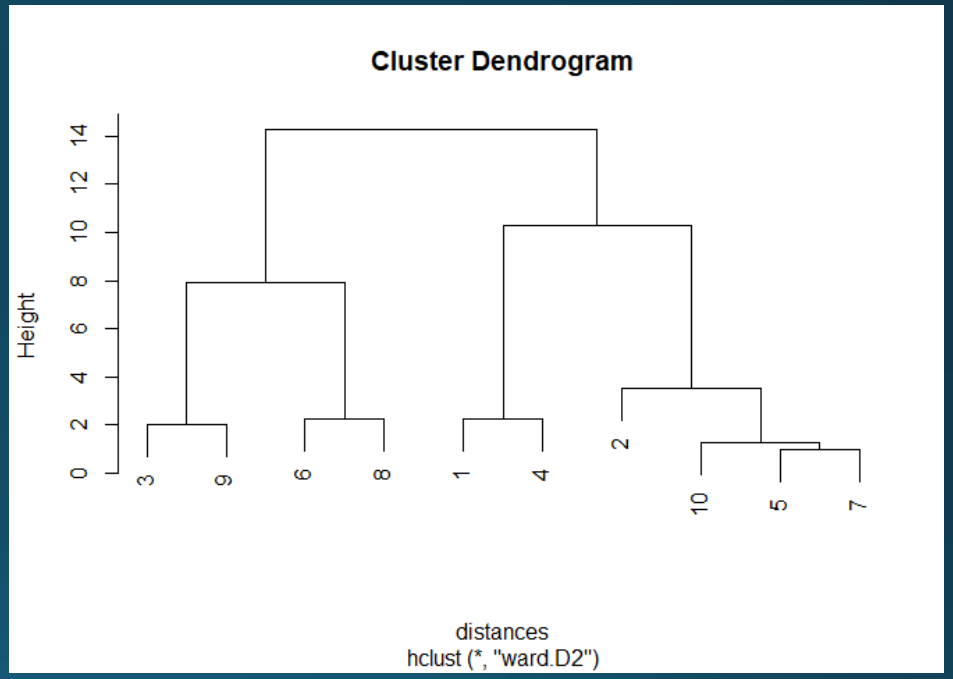

```
> clust$height
[1]  1.000000  1.287918  2.000000  2.240000  2.240000  3.516336
[7]  7.902712 10.279743 14.240875
```

```
> diff(clust$height)
[1] 0.2879182 0.7120818 0.2400000 0.0000000 1.2763357 4.3863759
[7] 2.3770315 3.9611321
```

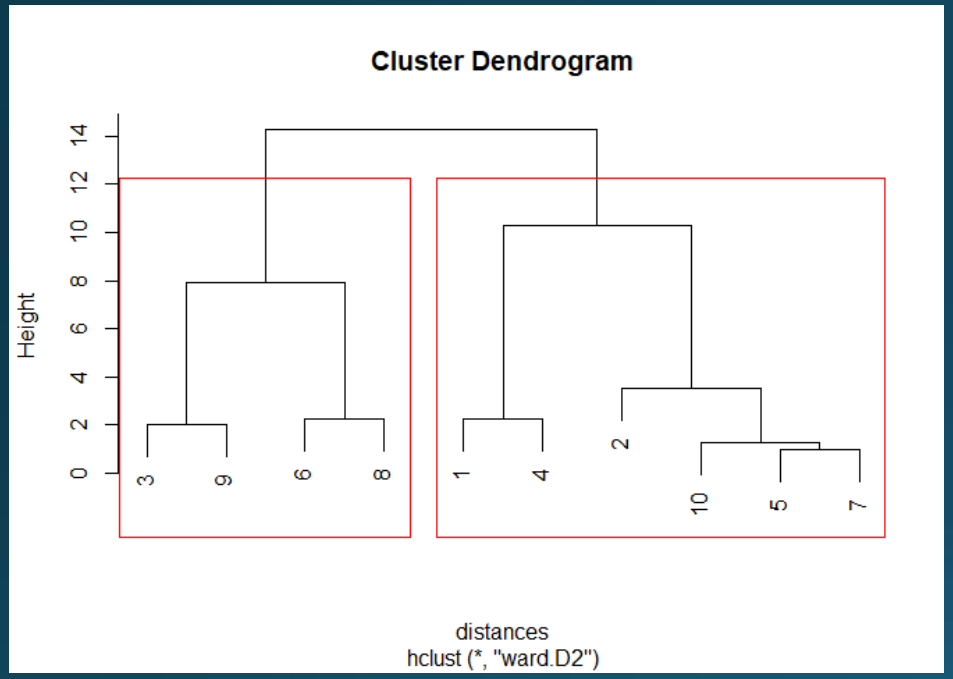
Determine Number of Clusters

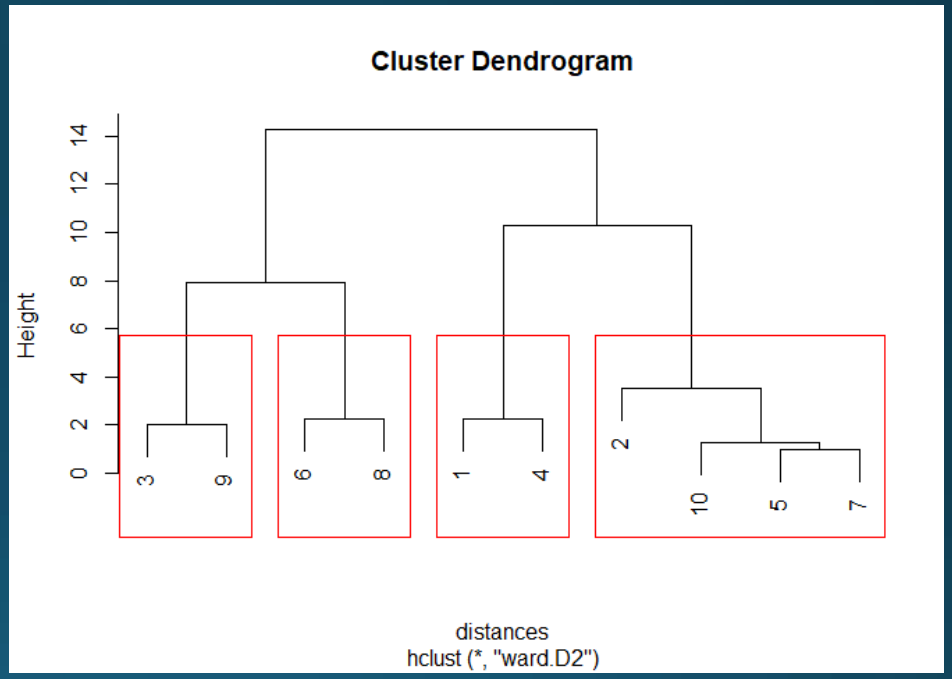
- Number of clusters may be determined by examining the height between branch splits.

- In this dendrogram, a cut between 11 and 13 would generate two clusters and a cut between 4 and 7 would create a four cluster solution. Based on distances, a three cluster solution doesn't seem viable.



Determine Number of Clusters

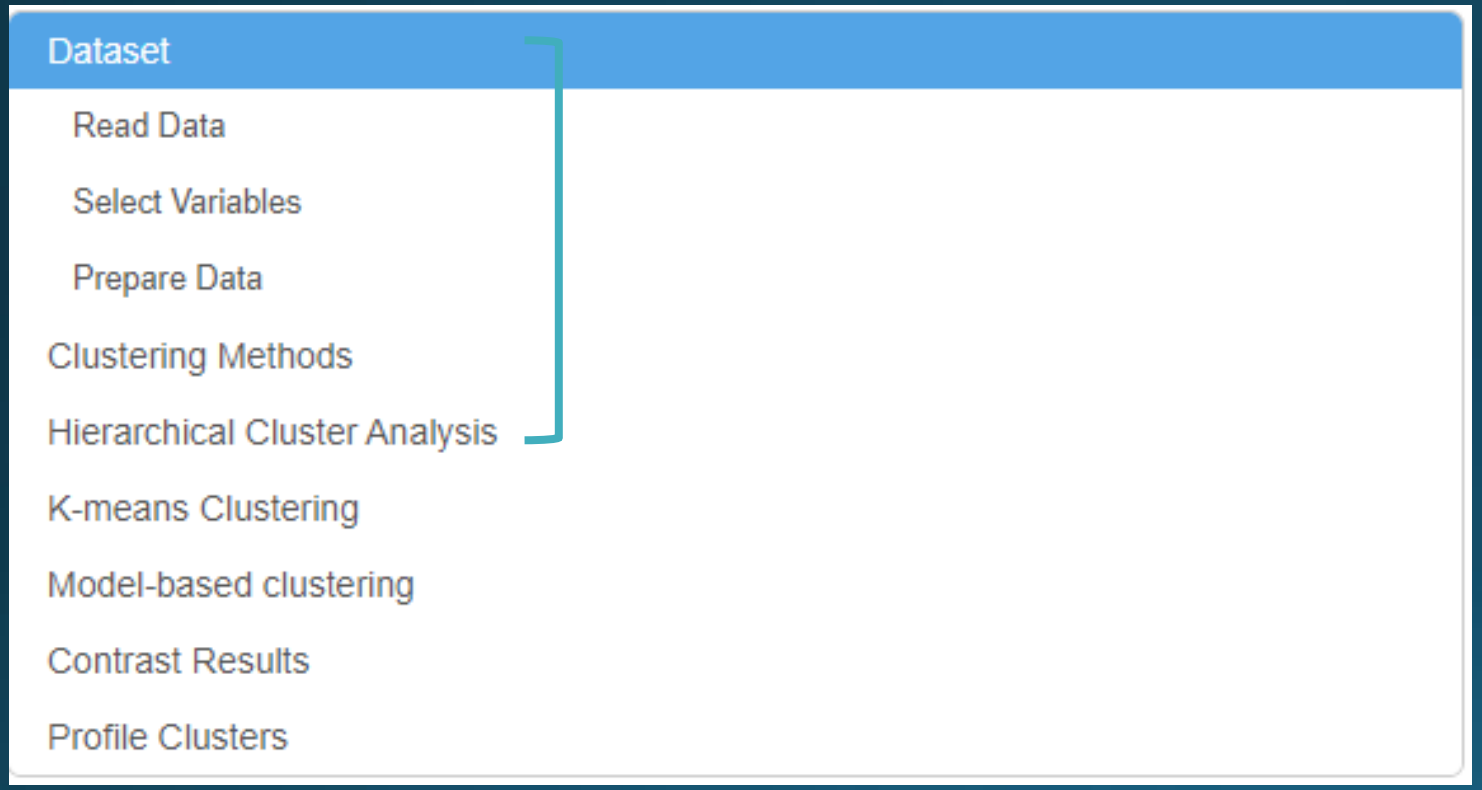




Meaningfulness of Clusters

- Hierarchical clustering will seldom converge on a single cluster solution.
- Ultimate decision on the number of clusters to pick should be made based on distances and meaningfulness of clusters
- The characteristics of the clusters should be examined based on the features used for clustering and other variables not used for clustering.
- In conducting market segmentation, it is common to segment based on needs-based variables (e.d., attitude, preference, and product usage), but profile segments based on demographic variables (e.g., age, gender, income).

R Illustration:
Clustering for Segmentation



Clustering for Segmentation

Dataset

This dataset is derived from a survey of customers of financial services.

Respondents were asked to rate the importance of 12 factors in selecting a primary financial provider

These include:

- performance of investments,
- fees or commissions charged,
- depth of products,
- ability to resolve problems,
- online services offered,
- multiple providers' products to choose from,
- quality of advice,
- knowledge of representatives,
- representative knowing needs,
- access to other professional services,
- degree to which providers knows customer, and
- quality of service.

Each item was rated on a 1-5 scale where 1 is not at all important and 5 is extremely important.

Data also includes (i) age, (ii) marital status and (iii) education of respondents.

Source: Malhotra, Naresh (2010), Marketing Research, 6th ed, Pearson, p. 808-813

Read Data

You must read the data before trying to run code on your own machine. To read data use the following code after setting your working directory. To set your working directory, modify the following to set the file path for the folder where the data file resides. setwd('C:/Users/Columbia/AAFM2')

```
data = read.csv(file = 'jpm_cluster.csv',stringsAsFactors = F)
```

```
str(data)
```

```
## 'data.frame':    500 obs. of  16 variables:
##  $ id                     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Q1_a                   : int  3 NA 5 5 5 5 5 4 5 4 ...
##  $ Q1_b                   : int  3 NA 5 3 5 3 4 4 3 4 ...
##  $ Q1_c                   : int  NA 4 3 5 4 5 5 3 5 3 ...
##  $ Q1_d                   : int  5 NA 2 5 4 4 5 4 5 3 ...
##  $ Q1_e                   : int  1 NA 3 4 4 NA 3 3 4 1 ...
##  $ Q1_f                   : int  3 NA 1 4 5 3 3 3 4 4 ...
##  $ Q1_g                   : int  5 NA 5 4 5 5 5 4 5 5 ...
##  $ Q1_h                   : int  5 4 5 5 5 5 5 4 5 5 ...
##  $ Q1_i                   : int  5 NA 2 5 5 4 5 4 5 4 ...
##  $ Q1_j                   : int  4 NA 2 4 4 5 5 3 3 2 ...
##  $ Q1_k                   : int  5 NA 4 5 5 5 5 4 5 4 ...
##  $ Q1_l                   : int  4 4 5 5 4 5 5 4 5 4 ...
##  $ Age_Recoded            : int  3 2 1 2 4 NA 1 2 2 2 ...
##  $ Marital_Status_Recoded: int  1 1 1 1 1 NA 1 1 1 1 ...
##  $ Education_Recoded      : int  2 1 5 3 2 NA 3 6 3 5 ...
```

- Age
  - '27-57'
  - '58-68'
  - '69-75'
  - '75+'
- Marital Status
  - 'married'
  - 'not married'
- Education
  - 'no college'
  - 'some college'
  - 'college graduate'
  - 'some graduate school'
  - 'masters degree'
  - 'doctorate'

Select Variables

Label Data

Naming columns based on survey. Also, assigning value labels to categories of demographic variables.

```
names(data) = c('id','performance','fees_commissions','depth_of_products','ability_resolve_problems',
                'online_services','choice_of_providers','quality_of_advice','knowledge_of_reps',
                'rep_knowing_your_needs','professional_services','provider_knows_me',
                'quality_of_service','age','marital_status','education')
data$age = factor(data$age,labels = c('27-57','58-68','69-75','75+'))
data$marital_status = factor(data$marital_status,labels=c('married','not married'))
data$education = factor(data$education,labels=c('no college','some college','college graduate',
                                                'some graduate school','masters degree','doctorate'))
str(data)
```

```
## 'data.frame':    500 obs. of  16 variables:
##  $ id                     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ performance            : int  3 NA 5 5 5 5 5 4 5 4 ...
##  $ fees_commissions       : int  3 NA 5 3 5 3 4 4 3 4 ...
##  $ depth_of_products      : int  NA 4 3 5 4 5 5 3 5 3 ...
##  $ ability_resolve_problems: int  5 NA 2 5 4 4 5 4 5 3 ...
##  $ online_services        : int  1 NA 3 4 4 NA 3 3 4 1 ...
##  $ choice_of_providers    : int  3 NA 1 4 5 3 3 3 4 4 ...
##  $ quality_of_advice      : int  5 NA 5 4 5 5 5 4 5 5 ...
##  $ knowledge_of_reps      : int  5 4 5 5 5 5 5 4 5 5 ...
##  $ rep_knowing_your_needs : int  5 NA 2 5 5 4 5 4 5 4 ...
##  $ professional_services  : int  4 NA 2 4 4 5 5 3 3 2 ...
##  $ provider_knows_me      : int  5 NA 4 5 5 5 5 4 5 4 ...
##  $ quality_of_service     : int  4 4 5 5 4 5 5 4 5 4 ...
##  $ age                    : Factor w/ 4 levels "27-57","58-68",..: 3 2 1 2 4 NA 1 2 2 2 ...
##  $ marital_status         : Factor w/ 2 levels "married","not married": 1 1 1 1 1 NA 1 1 1 1 ...
##  $ education              : Factor w/ 6 levels "no college","some college",..: 2 1 5 3 2 NA 3 6 3 5 ...
```

- Age
  - '27-57'
  - '58-68'
  - '69-75'
  - '75+'

- Marital Status
  - 'married'
  - 'not married'

- Education
  - 'no college'
  - 'some college'
  - 'college graduate'
  - 'some graduate school'
  - 'masters degree'
  - 'doctorate'

Subset needs-based variables

Variables in columns 2 - 13 include the needs based-variables. These variables reflect respondent importance rating of twelve features of financial services. We will subset these twelve variables. Five of the 12 variables are displayed.

```
data_cluster = data[,2:13]
head(data_cluster[,1:5])
```

	performance <int>	fees_commissions <int>	depth_of_products <int>	ability_resolve_problems <int>	online_services <int>
1	3	3	NA	5	1
2	NA	NA	4	NA	NA
3	5	5	3	2	3
4	5	3	5	5	4
5	5	5	4	4	4
6	5	3	5	4	NA

Examine the subset data

How many observations are in the subset.

```
str(data_cluster)
```

```
## 'data.frame':    500 obs. of  12 variables:
##  $ performance              : int  3 NA 5 5 5 5 5 4 5 4 ...
##  $ fees_commissions         : int  3 NA 5 3 5 3 4 4 3 4 ...
##  $ depth_of_products        : int  NA 4 3 5 4 5 5 3 5 3 ...
##  $ ability_resolve_problems: int  5 NA 2 5 4 4 5 4 5 3 ...
##  $ online_services          : int  1 NA 3 4 4 NA 3 3 4 1 ...
##  $ choice_of_providers      : int  3 NA 1 4 5 3 3 3 4 4 ...
##  $ quality_of_advice        : int  5 NA 5 4 5 5 5 4 5 5 ...
##  $ knowledge_of_reps        : int  5 4 5 5 5 5 5 4 5 5 ...
##  $ rep_knowing_your_needs   : int  5 NA 2 5 5 4 5 4 5 4 ...
##  $ professional_services    : int  4 NA 2 4 4 5 5 3 3 2 ...
##  $ provider_knows_me        : int  5 NA 4 5 5 5 5 4 5 4 ...
##  $ quality_of_service       : int  4 4 5 5 4 5 5 4 5 4 ...
```

```
nrow(data_cluster)
```

```
## [1] 500
```

How many missing values are there for the variable performance are in the subset

```
sum(is.na(data_cluster$performance))
```

```
## [1] 42
```

How many rows of data would be left if rows corresponding to missing values on any of the variables were removed. Let's use na.omit() and not to overwirte the original subset

```
nrow(na.omit(data_cluster))
```

```
## [1] 396
```

Prepare Data

Transform format of variables

Clustering algorithms prefer all variables to be of the same class (e.g., numeric, factor). Furthermore, algorithms differ in the class of variables required (e.g., hclust and k-means work with numeric while polCA works with factor).

Since all the survey responses in this dataset are numeric variables, there is no need to change the class of the variables.

Note: It is common to treat survey responses gathered on 1-5 scale to be numeric, as we have done here. However, some researchers contend that responses to itemized rating scales are ordinal, which in R are represented by an ordered factor.

Impute missing data

Clustering algorithms use data on all variables. A missing observation on one variable will cause the entire row of data to be ignored for analysis. Therefore, it is important to impute missing values.

Let us impute the missing values. There are many packages and functions in R to use for imputation. We are going to make use of the mice package with the default method, predictive mean matching. Setting the seed is critical for getting consistent results. In newer version of the `mice` library, it implemented a new match index C function that makes predictive mean matching 50 to 600 times faster, however this affects reproducibility of the algorithm. Read more about the latest version here. In order to reproduce the original behavior, include `use.matcher=T` in the `mice` function. This is critically important, especially when completing the associated assignment. Finally, `tidyr` has an identically named `complete` function. To prevent conflicts, it is best to include the package reference, `mice::complete(...)`

```
library(mice)
set.seed(617)
data_cluster = mice::complete(mice(data_cluster,use.matcher=T))
```

Running the `mice` function in interactive mode will generate a lengthy output of the iterative process. If you would rather not see this output, you can gather the output of the iterative process in a garbage collector variable as illustrated below.

```
garbage = capture.output(data_cluster <- mice::complete(mice(data_cluster,use.matcher=T)))
```

Now, let's examine the imputed values for the first few rows of the first four variables.

```
head(data_cluster[,1:4])
```

	performance <int>	fees_commissions <int>	depth_of_products <int>	ability_resolve_problems <int>
1	3	3	4	5
2	4	4	4	4
3	5	5	3	2
4	5	3	5	5
5	5	5	4	4
6	5	3	5	4

6 rows

```
data_cluster$performance[2]
```

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

```
> data_cluster[2,1]
[1] 4
> data_cluster[2,"performance"]
[1] 4
```

Scale

For distance-based clustering methods, scale of the variable (e.g., seconds vs minutes) affects the weight assigned to the variable. Therefore, it is best to standardize variables.

There are many ways to standardize a variables. Here, we are using scale() which will by default subtract mean and divide by standard deviation. One thing to be cautious of is that scale() returns a matrix. Examine the standardized values of the first four variables and the second observation of the variable performance.

```
data_cluster = scale(data_cluster)
head(data_cluster[,1:4])
```

```
##      performance fees_commissions depth_of_products
## [1,]  -1.3928295       -0.8440928         0.2078544
## [2,]  -0.3147881        0.2549864         0.2078544
## [3,]   0.7632533        1.3540656        -0.8314177
## [4,]   0.7632533       -0.8440928         1.2471266
## [5,]   0.7632533        1.3540656         0.2078544
## [6,]   0.7632533       -0.8440928         1.2471266
##      ability_resolve_problems
## [1,]               1.09390272
## [2,]               0.05209061
## [3,]              -2.03153363
## [4,]               1.09390272
## [5,]               0.05209061
## [6,]               0.05209061
```

For variable "performance",
the standardize value of second observation.

```
data_cluster[2,1]
```

```
## performance
##  -0.3147881
```

Redundant variables

Multiple variables measuring the same dimension should be replaced by the underlying factor or a representative variable. Imagine a survey that uses 3 questions to measure restaurant service and 1 question to measure food quality in the restaurant. Used as such cluster analysis will on average weight restaurant service three times as much as food quality. When faced with multiple variables measuring an underlying factor, it is best to run a factor analysis and use factors instead of variables. Alternatively, one could combine correlated variables into a single component based on the results of a principal components analysis.

Since we have not examined dimension reduction methods yet, let us assume the survey items used here do not tap any underlying factor and move along.

Clustering Methods

Clustering algorithms can be broadly categorized into

- **Distance-based methods**: Find groups that minimize the distance between members within the group while maximizing the distance from members of other groups. Popular methods are Hierarchical clustering and k-means.
- **Model-based methods**: View the data as a mixture of groups sampled from different distributions

Let us begin with Hierarchical Cluster Analysis

Hierarchical Cluster Analysis

- Popular method that groups observations according to their similarity
- Distance-based algorithm that operates on a dissimilarity matrix
- Algorithm begins with each observation in its own cluster
- Successively joins neighboring observations or clusters on at a time according to their distances from one another, and continues until all observations are linked
- Process of repeatedly joining observations is called the agglomerative method

Compute Similarity Measure

Similarity is generally measured using a distance-based measure (as opposed to a correlational measure). Here, we will make use of Euclidean distance to assess similarity. Now, it is important to note that Euclidean distance only works with integer or numeric data. If data also includes a factor, an alternative is to use daisy() from library(cluster).

```
d = dist(x = data_cluster,method = 'euclidean')
```

How many elements are in the distance matrix?

```
length(d)
```

```
## [1] 124750
```

Clustering Method

Distance between pairs of points is useful in identifying similarity between a pair of points. As observations are combined into clusters, distances must now be evaluated among clusters or between a point and a cluster. There are different approaches to grouping observations and these are called clustering methods. These methods differ based on whether they are based on distance (single, complete, linkage), variance (ward.D2, ward.D) or centroid (median, centroid).

Let us conduct hierarchical clustering using Ward's method.

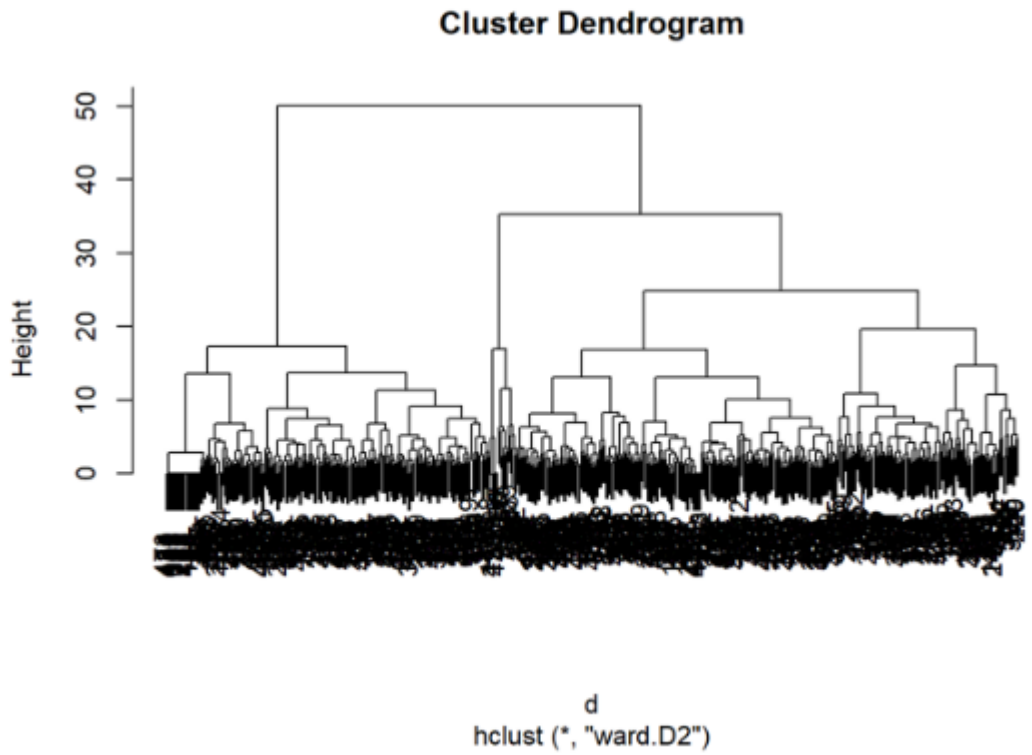
```
clusters = hclust(d = d,method='ward.D2')
```

Intepret Results

Examine Dendrogram

The ultimate result of cluster analysis is a dendrogram which looks a bit like an inverted tree. A hierarchical dendrogram is interpreted primarily by height and where observations are joined. The height represents the dissimilarity between elements that are joined.

```
plot(clusters)
```



Goodness of Fit

Before moving on to deciphering the dendrogram, it is useful to look at the Cophenetic correlation coefficient. Cophenetic correlation coefficient (CPC) is a goodness of fit statistic for hierarchical clustering which assesses how well a dendrogram matches the true distance metric. It is interpreted similar to a correlation coefficient. CPC> 0.7 indicates relatively strong fit, 0.3<CPC<0.7 indicates moderate fit.

```
cor(cophenetic(clusters),d)
```

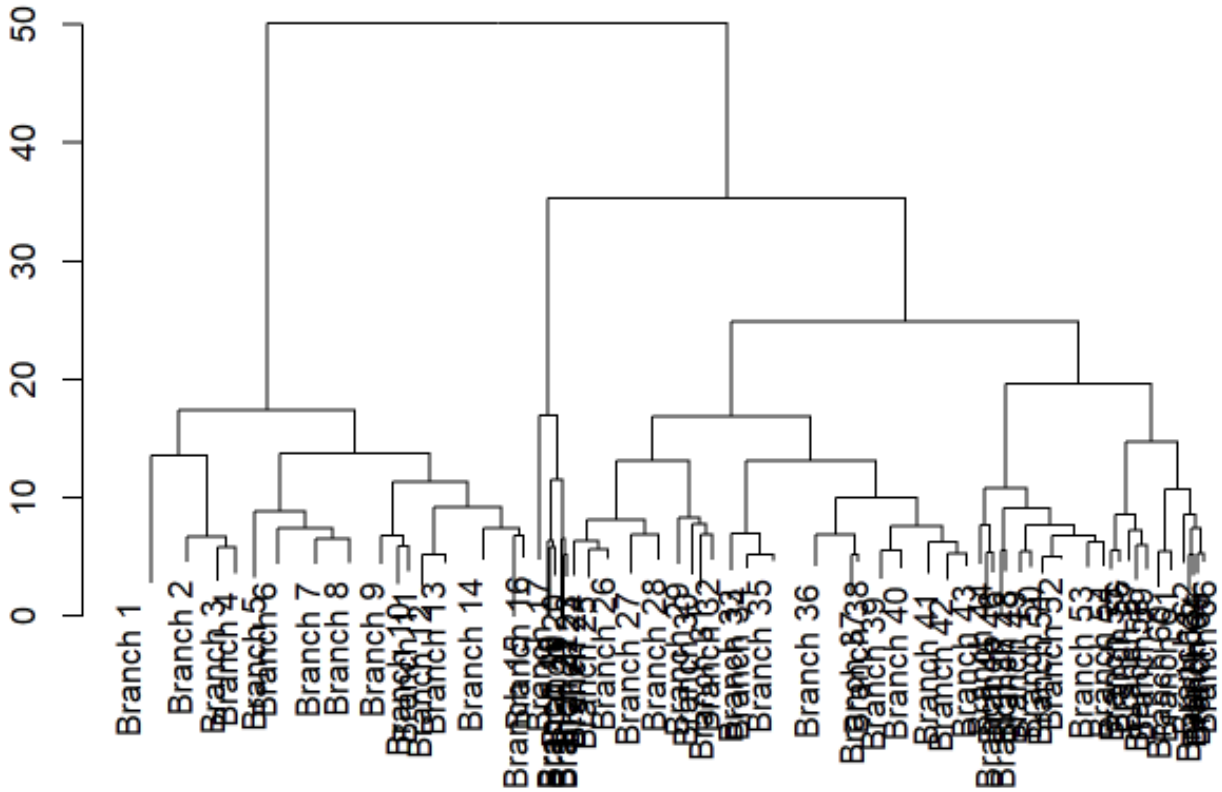
```
## [1] 0.3479761
```

Number of Clusters

Even a moderate sized data will make the leaves of the dendrogram look very busy. To clarify the picture we cut the dendrogram to only display the tree above a certain height.

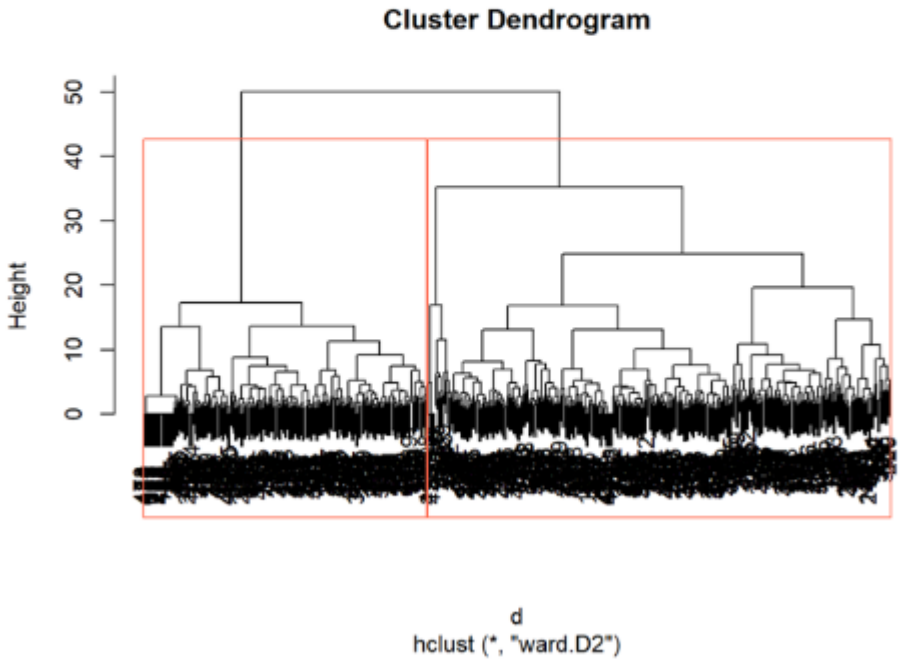
Let's only disply the tree above 5. Based on the distances, a two or three cluster solution looks good.

```
plot(cut(as.dendrogram(clusters),h=5)$upper)
```



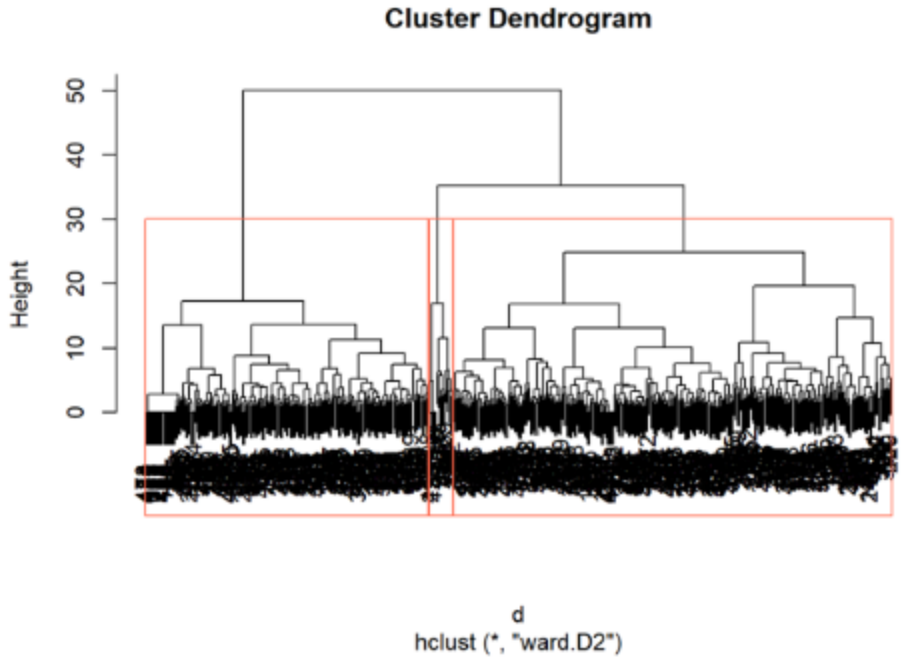
Let us draw rectangles around a two cluster solution

```
plot(clusters)
rect.hclust(tree=clusters, k=2, border='tomato')
```



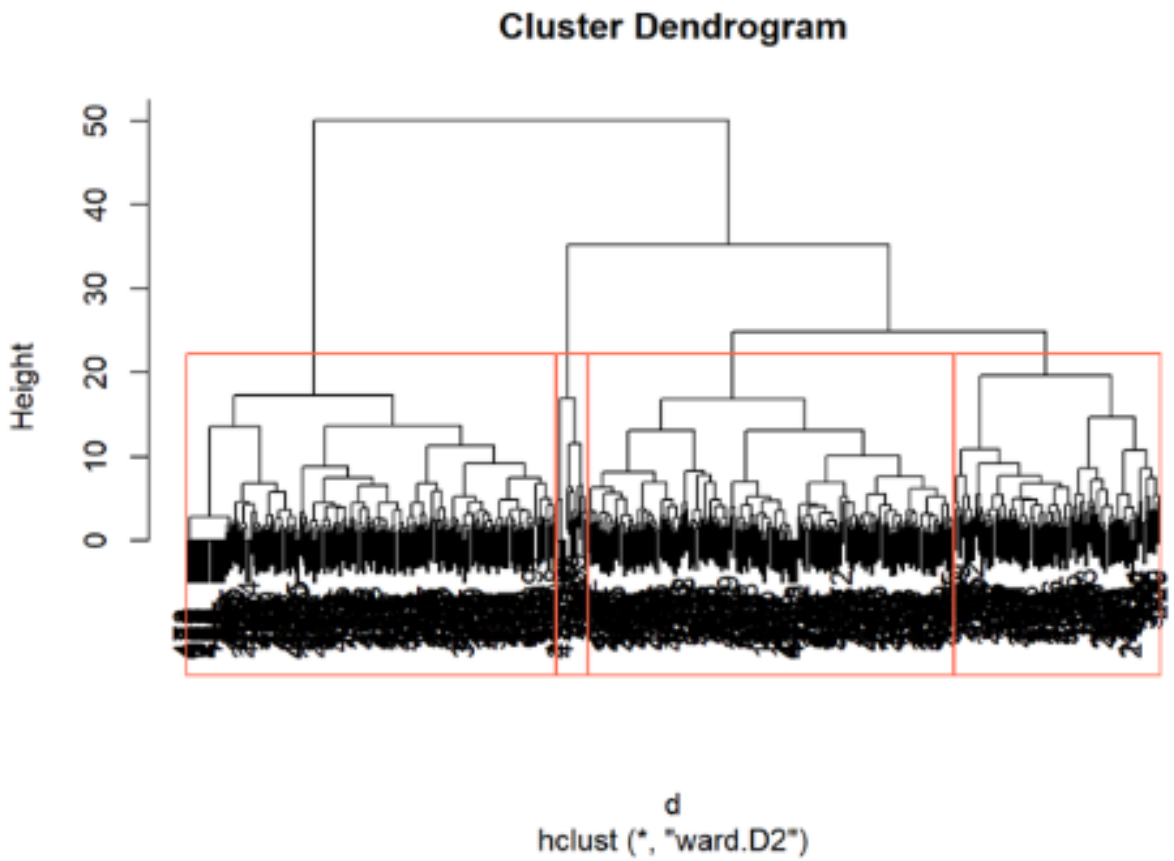
Now, let us do the same for a three cluster solution

```
plot(clusters)
rect.hclust(tree=clusters, k=3, border='tomato')
```



Now, let us do the same for a four cluster solution. The solution indicates three major segments and one niche-segment.

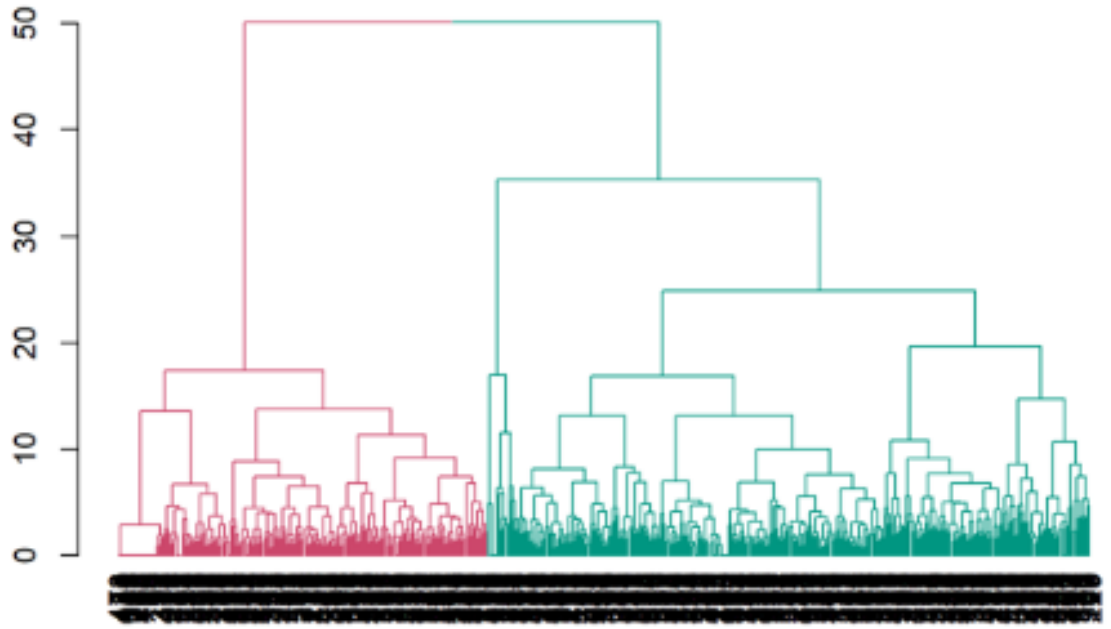
```
plot(clusters)
rect.hclust(tree=clusters, k=4, border='tomato')
```



Use dendextend

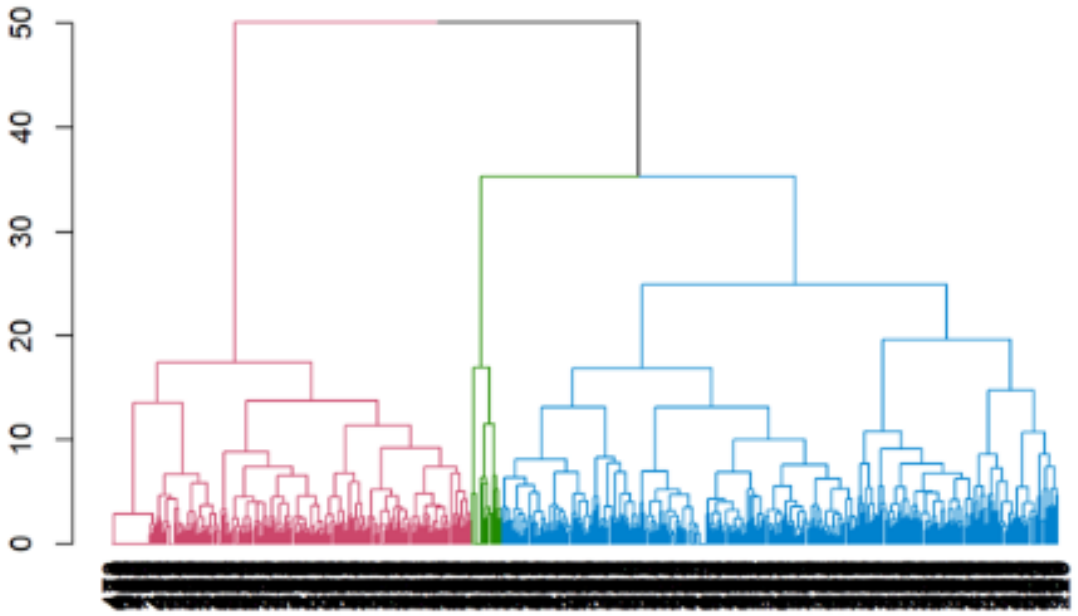
Let us highlight a two cluster solution using dendextend

```
library(dendextend)
plot(color_branches(as.dendrogram(clusters), k=2, groupLabels = F))
```



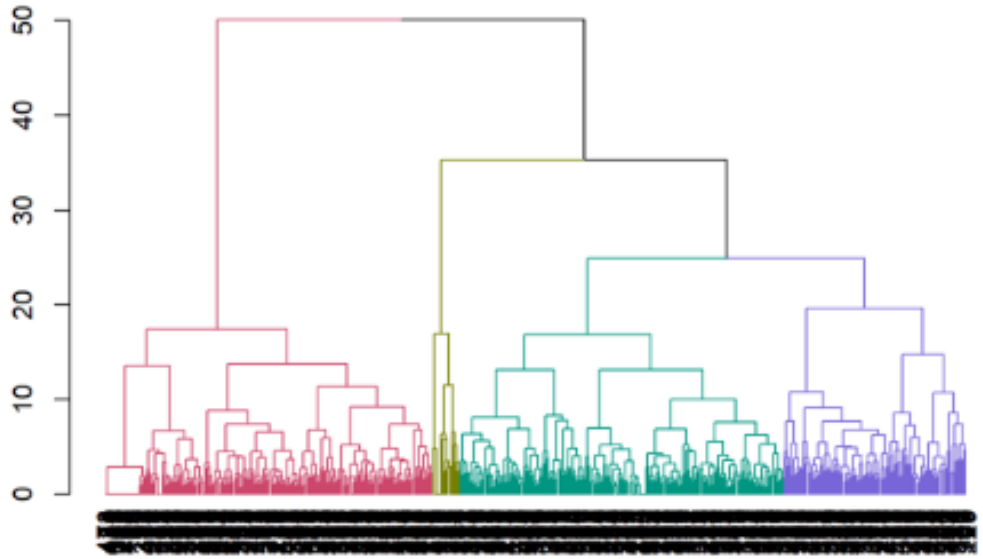
Let us highlight a three cluster solution using dendextend

```
library(dendextend)
plot(color_branches(as.dendrogram(clusters), k=3, groupLabels = F))
```



And, now a four cluster solution using dendextend

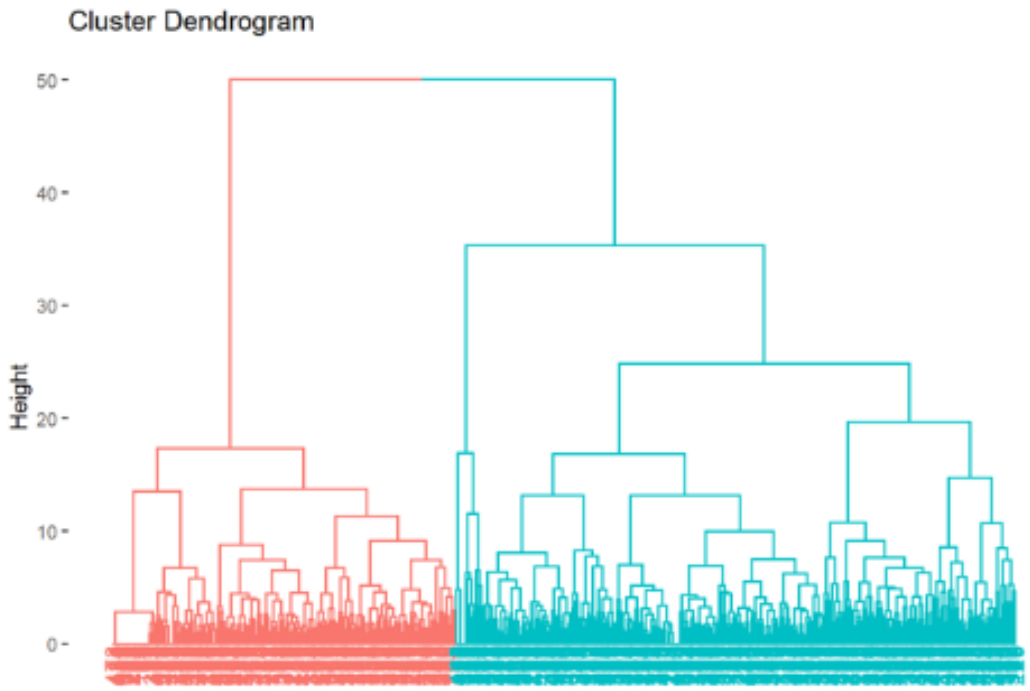
```
library(dendextend)
plot(color_branches(as.dendrogram(clusters), k=4, groupLabels = F))
```



Use factoextra

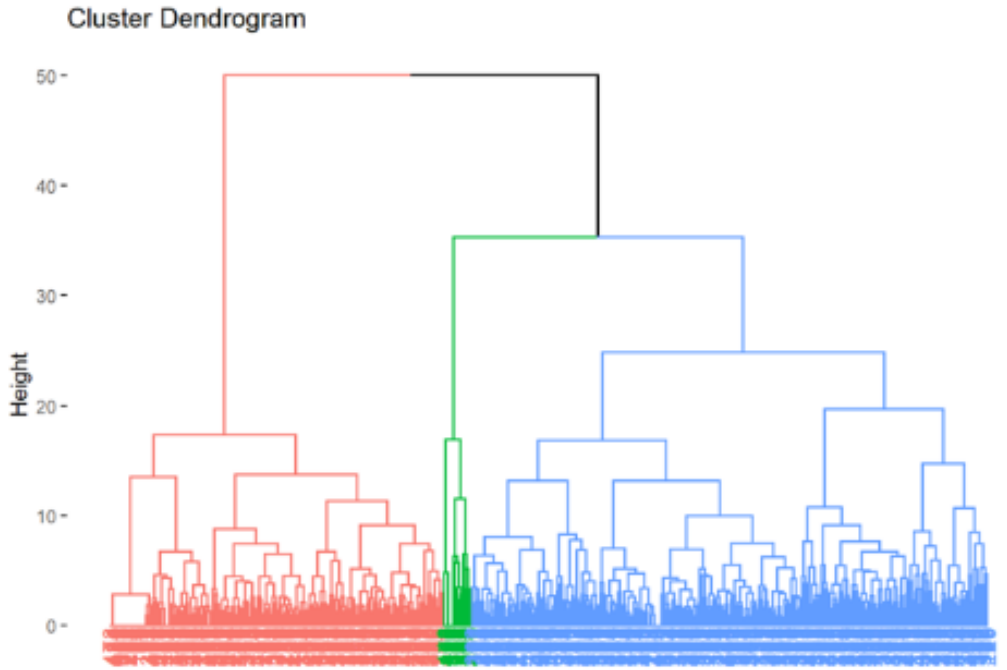
Another useful library for visualizing a dendrogram is the fviz_dend from library(factoextra). Highlight two clusters.

```
library(factoextra)
fviz_dend(x = clusters,k=2)
```



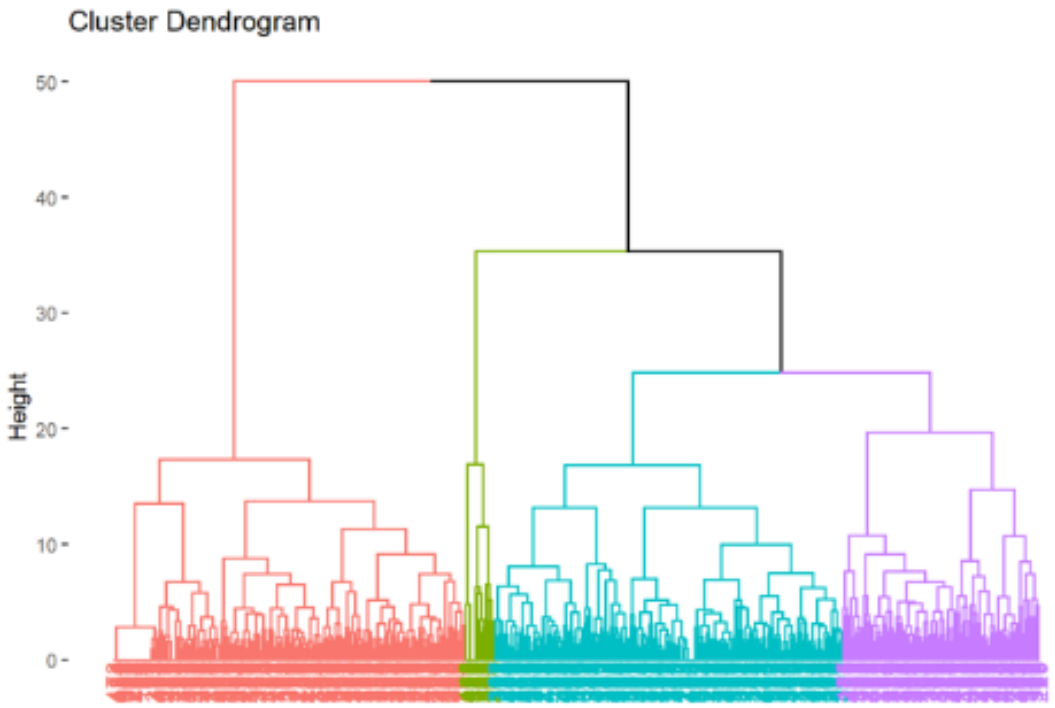
Three Clusters

```
library(factoextra)
fviz_dend(x=clusters,k=3)
```



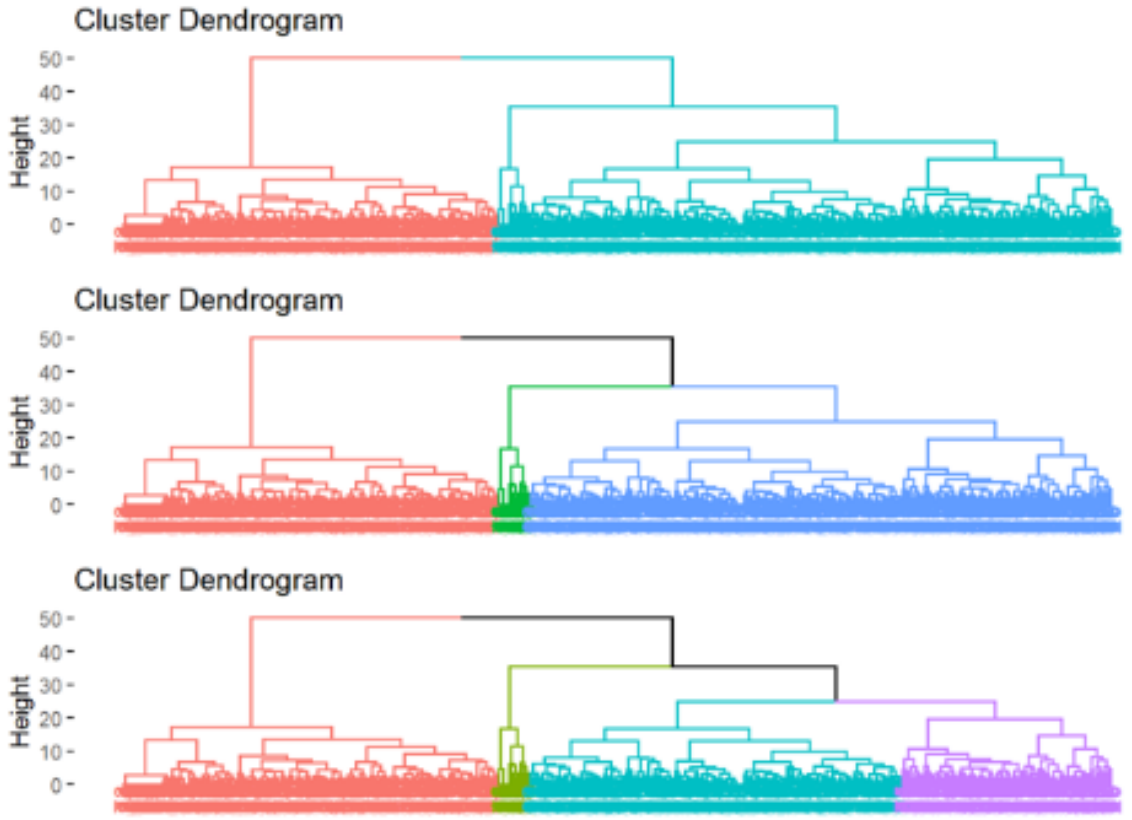
Four Clusters

```
library(factoextra)
fviz_dend(x=clusters,k=4)
```



From the default color palette of above charts, it may be obvious that this fviz_dend generates a ggplot2 object. This makes other packages such as gridExtra that use ggplot objects accessible.

```
library(gridExtra)
grid.arrange(fviz_dend(x=clusters, k=2),
             fviz_dend(x=clusters, k=3),
             fviz_dend(x=clusters, k=4)
)
```



Selecting Clusters

Number of clusters may be determined by examining the height between branch splits. Combining distances and domain knowledge, a four-cluster solution seems to be best. Most of the respondents seem to be clustering into three groups with the fourth cluster containing a very small number of respondents. For a financial services company, three major segments and one niche segment is a reasonable number of segments to cater to. Based on these considerations, we cut the data into four clusters.

```
h_segments = cutree(tree=clusters, k=4)
h_segments
```

```
##   [1] 1 2 1 3 3 3 3 1 3 1 1 2 1 1 1 1 3 1 1 3 2 3 3 1 2 2 3 3 3 1 2 1 1 3 3 3 3
##  [38] 3 2 1 1 3 2 4 2 1 2 2 3 3 3 1 1 2 3 3 2 1 1 3 3 3 1 3 3 3 3 1 2 2 2 3 1 1
##  [75] 3 2 2 2 1 1 1 3 1 2 2 2 3 3 1 1 1 2 2 2 2 3 3 3 3 1 3 2 3 1 3 1 3 3 2 3 2
## [112] 3 3 3 1 1 3 3 1 3 3 3 4 2 1 3 3 3 2 3 1 1 2 1 2 4 4 1 3 2 3 3 1 2 1 3 1 2
## [149] 3 3 2 3 1 3 1 3 3 4 3 2 1 3 2 3 3 1 2 3 1 1 1 1 1 2 2 1 1 3 3 3 1 2 2 1 2
## [186] 1 1 1 1 2 1 3 3 1 2 1 3 3 1 1 1 1 2 3 2 3 3 4 2 2 2 3 3 1 2 2 3 1 3 1 2 1
## [223] 1 3 3 1 3 1 2 3 3 1 1 3 3 3 2 4 2 1 2 1 1 1 1 4 3 2 1 3 1 2 4 3 1 3 3 1 3
## [260] 2 1 3 1 1 3 3 1 3 1 1 2 1 1 1 1 3 2 1 1 2 1 3 1 2 1 3 1 3 3 4 2 3 3 3 3 3
## [297] 1 3 2 1 1 1 2 3 3 2 2 1 1 3 1 3 2 1 1 2 3 1 3 3 1 1 1 1 3 2 1 1 1 1 1 2 3
## [334] 1 3 2 3 1 1 1 1 3 3 3 1 2 3 1 1 2 1 2 1 1 3 3 3 3 4 2 3 1 1 1 1 3 4 3 3 3
## [371] 2 1 1 3 3 3 1 1 1 2 1 3 3 4 3 1 1 1 2 3 1 2 3 3 1 1 3 3 1 1 1 3 3 2 1 1 3
## [408] 1 3 2 3 3 2 1 1 1 3 4 2 1 2 1 2 1 3 2 1 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 2
## [445] 3 1 2 1 1 1 2 3 2 3 3 3 2 1 4 2 1 2 3 1 3 1 2 1 1 1 3 2 3 1 4 1 1 3 3 1 2
## [482] 1 1 2 3 2 3 1 2 3 1 2 3 1 3 1 2 3 1 3
```

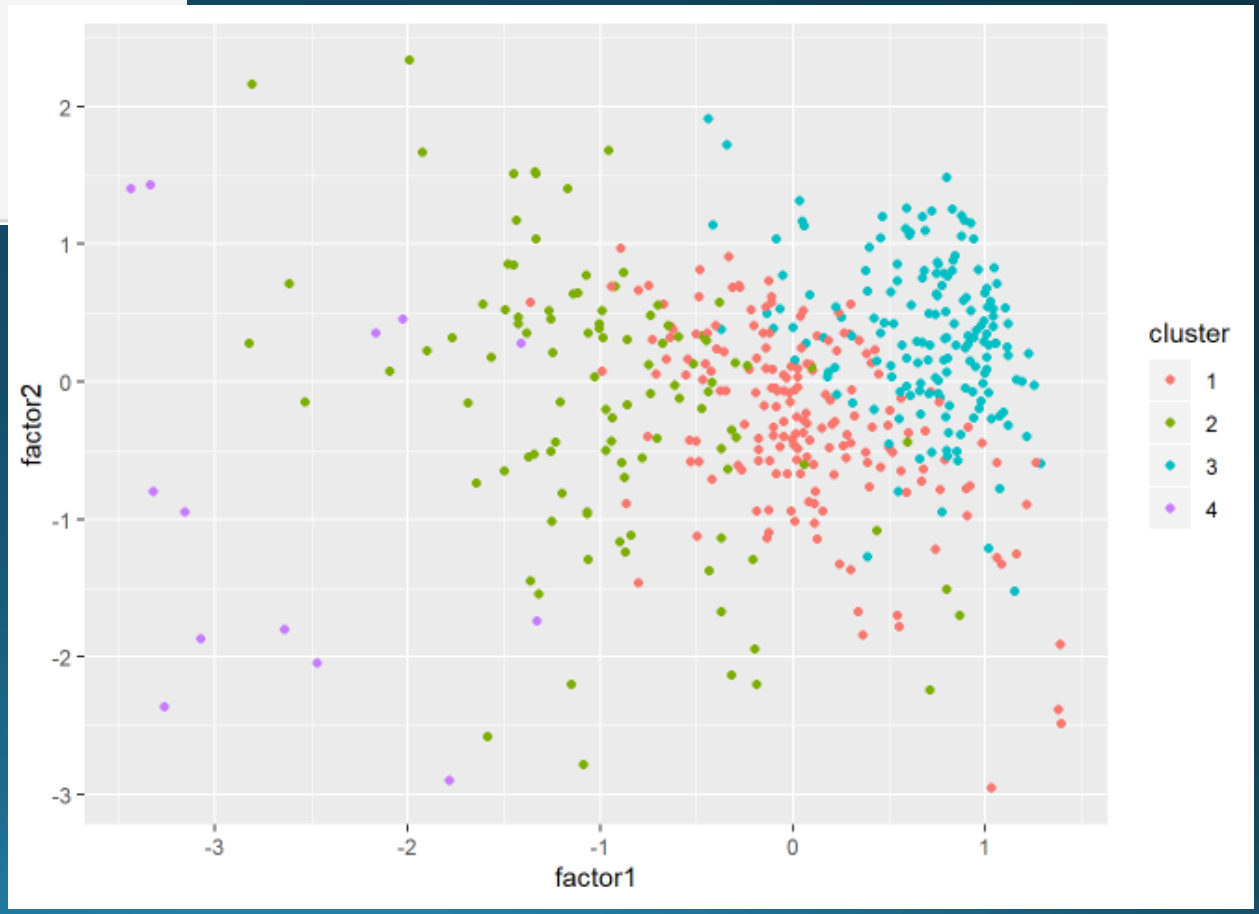
```
table(h_segments)
```

```
## h_segments
##   1   2   3   4
## 188 106 190  16
```

Visualize

To express the clusters on a scatterplot, we flatten the data from 12 dimensions onto 2 by conducting a factor analysis with varimax rotation. This is done because it is not possible to visualize 12-dimensional data.

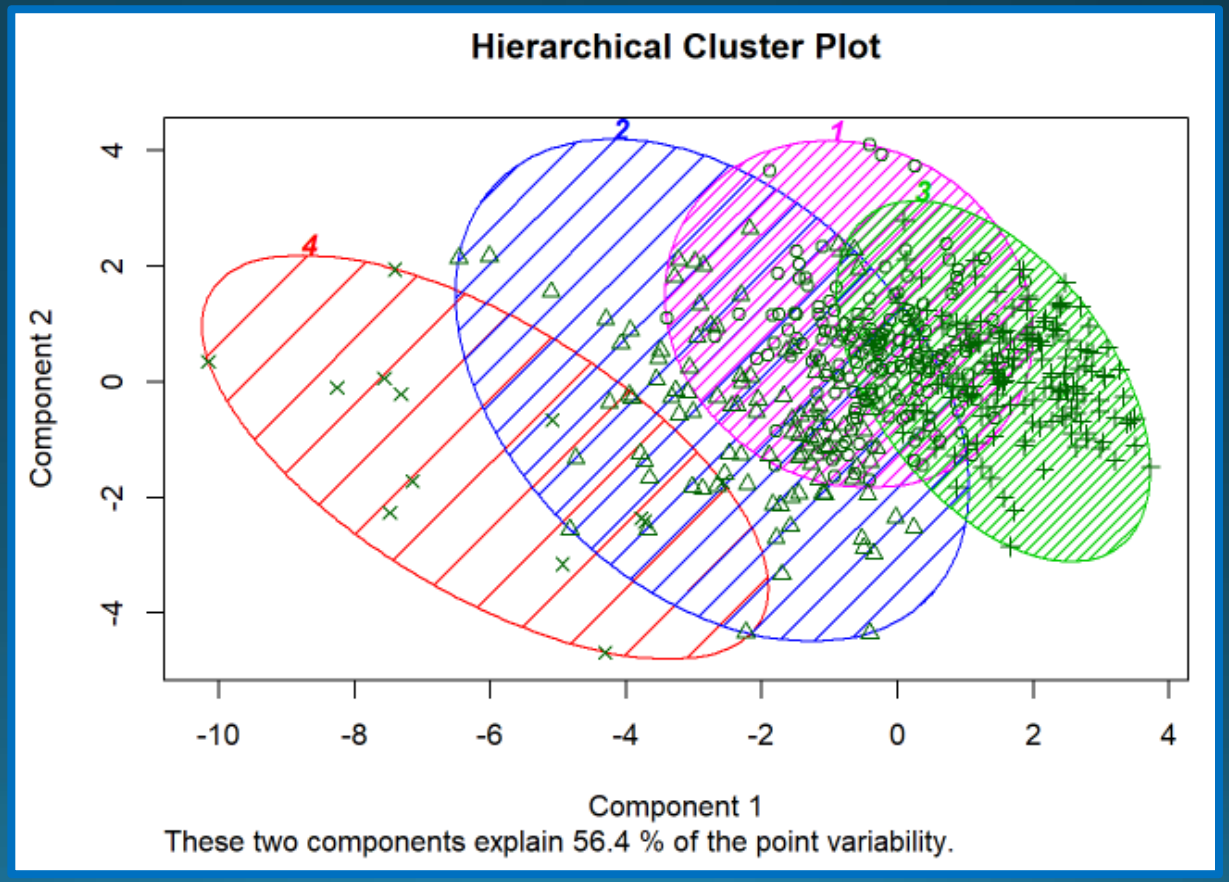
```
library(psych)
temp = data.frame(cluster = factor(h_segments),
           factor1= fa(data_cluster,nfactors=2, rotate ='varimax')$scores[,1],
           factor2= fa(data_cluster,nfactors=2, rotate ='varimax')$scores[,2])

ggplot(temp,aes(x=factor1,y=factor2,col=cluster))+
  geom_point()
```



Using clusplot

`clusplot` does something similar. Rather than running a factor analysis, it runs a principal components analysis.

```
library(cluster)
clusplot(data_cluster,
         h_segments,
         color=T, shade=T, labels=4, lines=0, main='Hierarchical Cluster Plot')
```



Hierarchical Clustering

- Hierarchical clustering is simple and intuitive.
- However, this technique does not scale well to large datasets. This is because, the technique requires computing distances between every pair of observations. For large datasets, this can be a very expensive process.
- The computation of the distance matrix can consume significant computing resources and the distance matrix itself can overwhelm system memory.
- Let us turn to another technique that is less resource intensive.

K-means Clustering

k-means Clustering

- Attempts to find groups that are most compact, in terms of the mean sum-of-squares deviation of each observation from the multivariate center (centroid) of its assigned group.
- Non-hierarchical process that reaches solution through an iteration.
- K-means clustering begins by arbitrarily placing centroids in the data and then iterating from that point to the final solution. This disorganized approach to clustering produces similar quality of clusters to hierarchical clustering but much faster.

k-means Process

- Requires an a priori definition of number of clusters
- Begins with an arbitrary assignment of observations to cluster centroids. Seed is important for reproducibility.
- Iterative process involves
  - Updating cluster centers
  - Reassigning observations
- Updating is done to minimize sum of squares from observations to cluster centers
- Since it explicitly computes a mean deviation, k-means clustering relies on Euclidean distance
- It is only suitable for numerical data
- Distance is computed using Euclidean's method
- Method of updating is defined by algorithms including
  - "Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"

k-means

- Initial assignment of centers is influenced by seed
- Sensitive to scale of variables as similarity is based on Euclidean distance

Visualizing k-means

k-means clustering algorithm involves the following steps

1. Identify number of centers, k, to be used
2. Randomly place centers in data
3. Assign observations to nearest center
4. Update cluster centers
5. Reassign cases
6. Repeat steps 4 and 5 until convergence

Next few slides visually illustrate the process until convergence is reached.

Nonhierarchical Clustering: K-means Algorithm

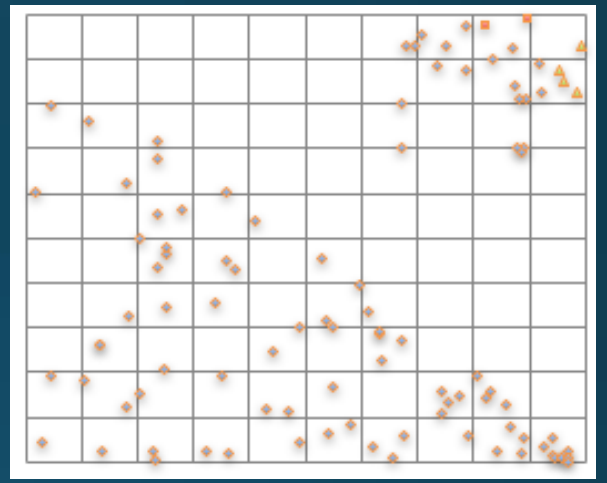
The *centroid* is the "average position" inside a cluster

Set *k* initial seeds (centroids)

Compute the centroid

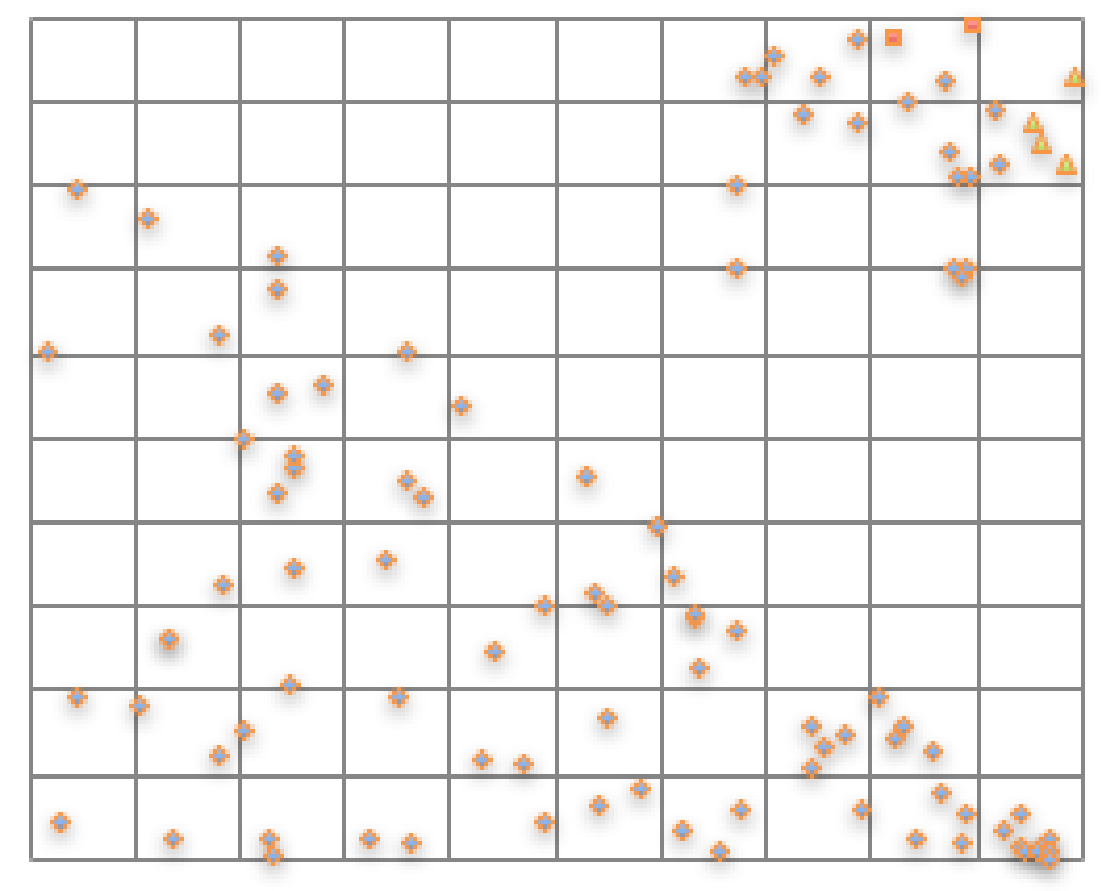
Assign each item to its nearest centroid

Exit loop when cluster structure stabilizes

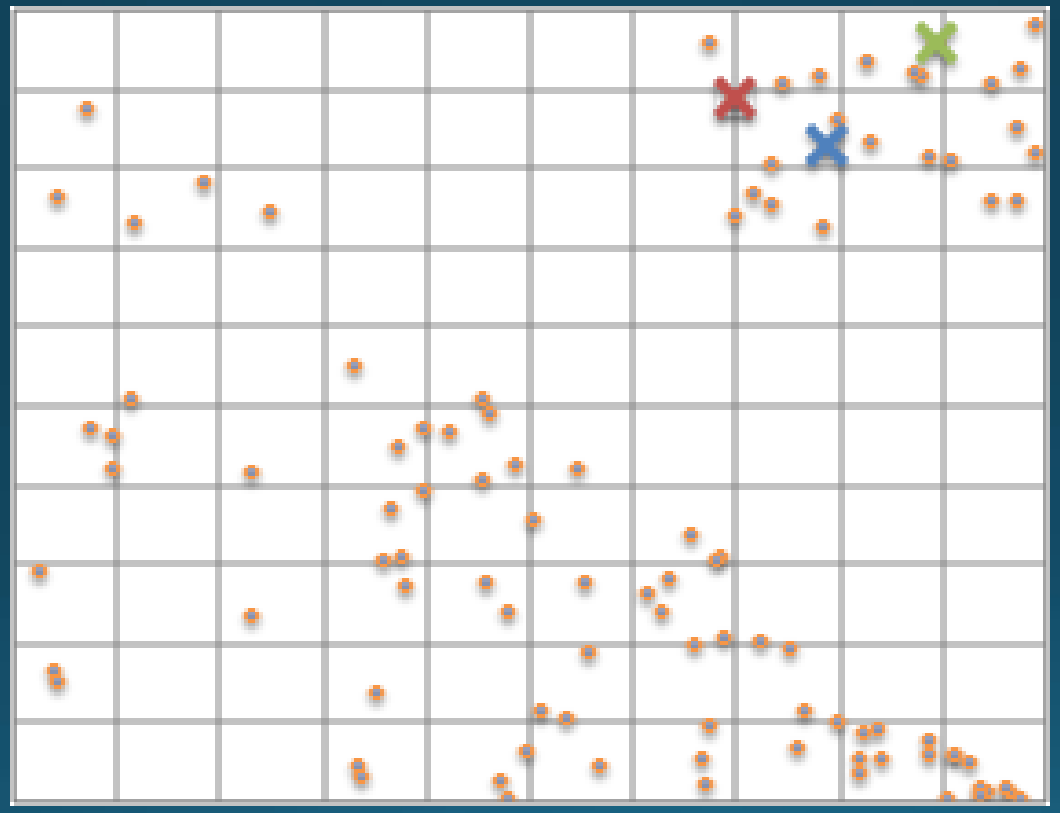


k-means initializes observations to random clusters, it is important to set the random number generator seed for reproducibility.

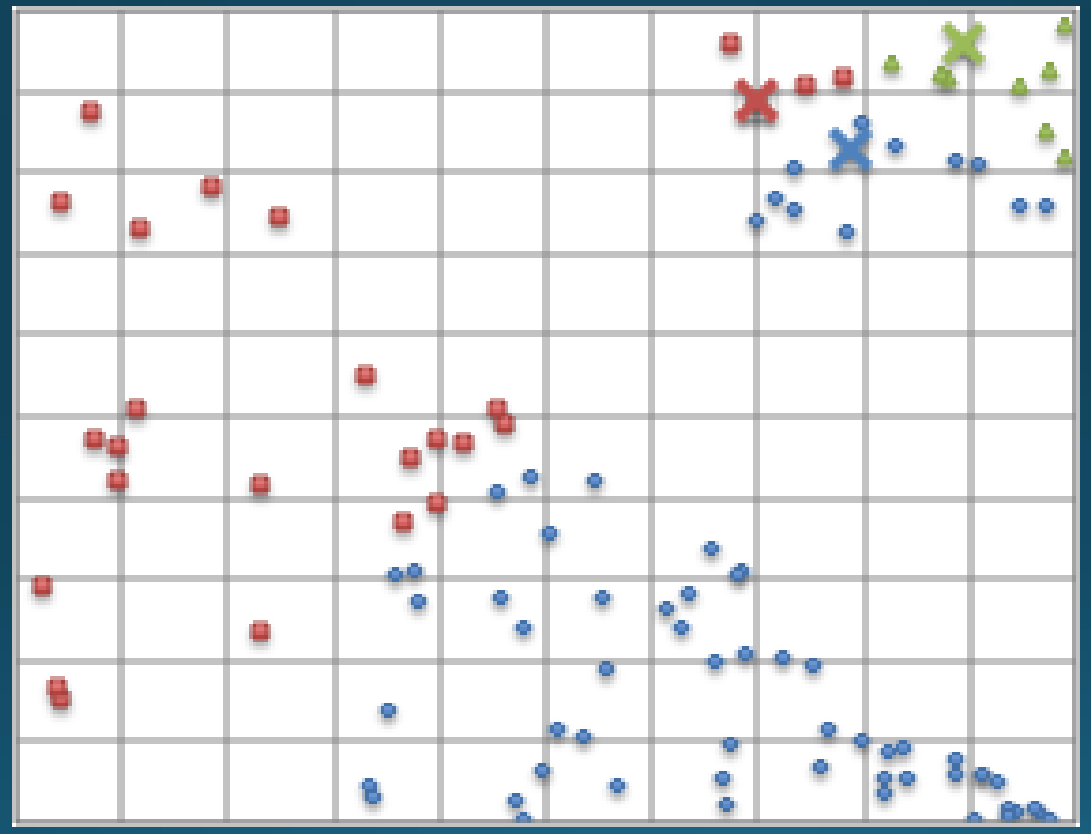
k-means Algorithm



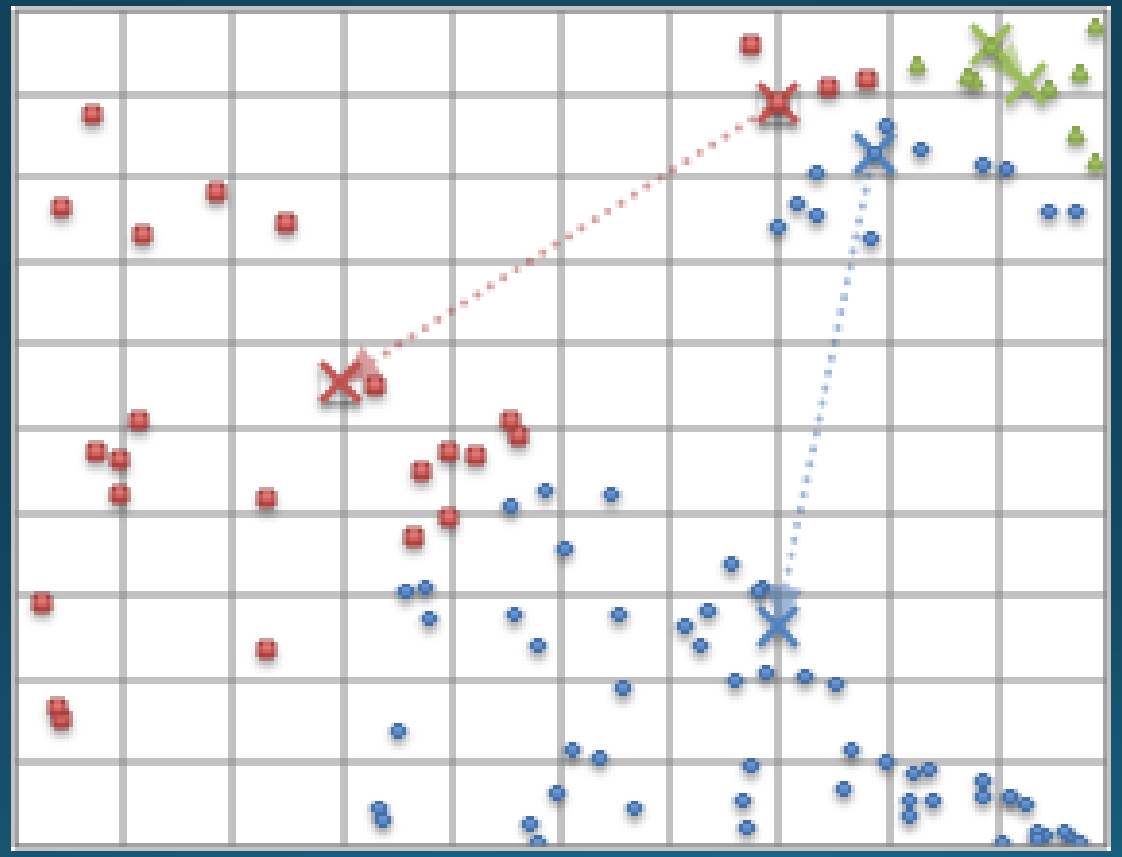
The k-means Algorithm Starts by Choosing k Centers



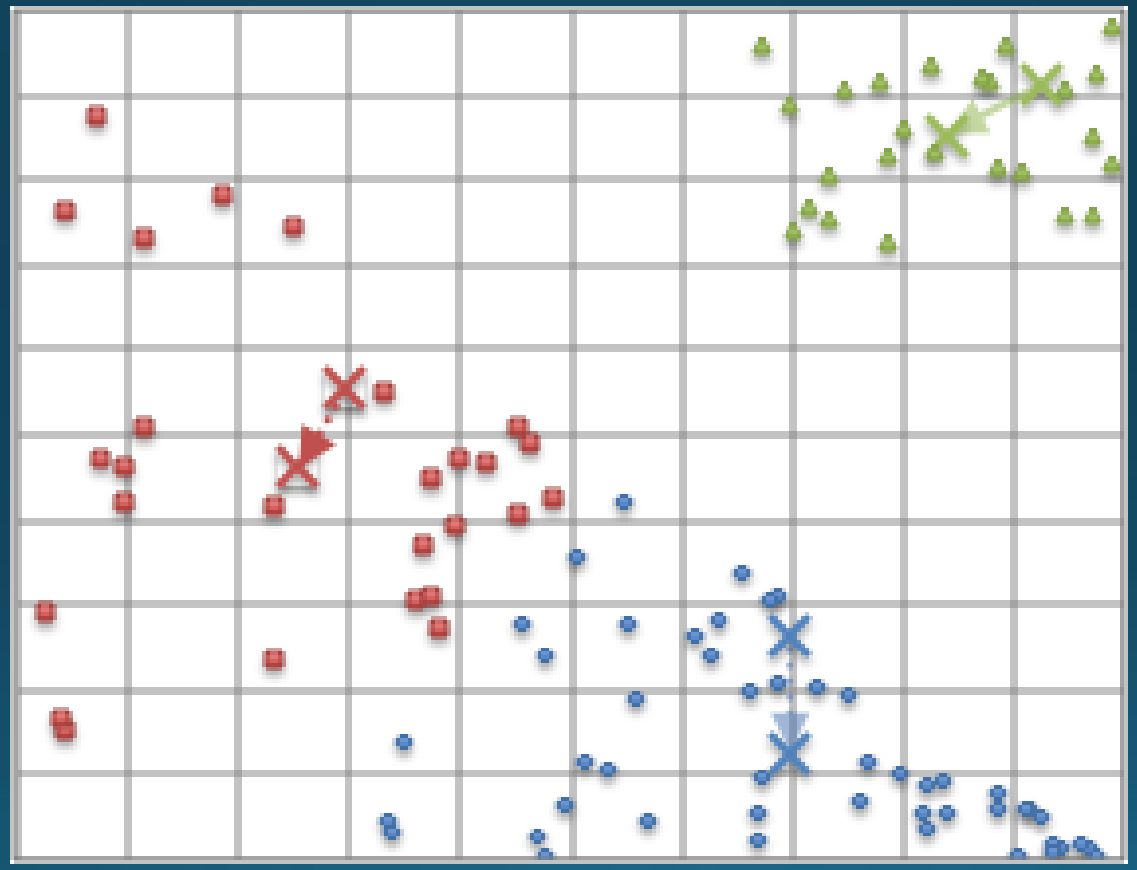
Assign Each Point to the Nearest Centroid



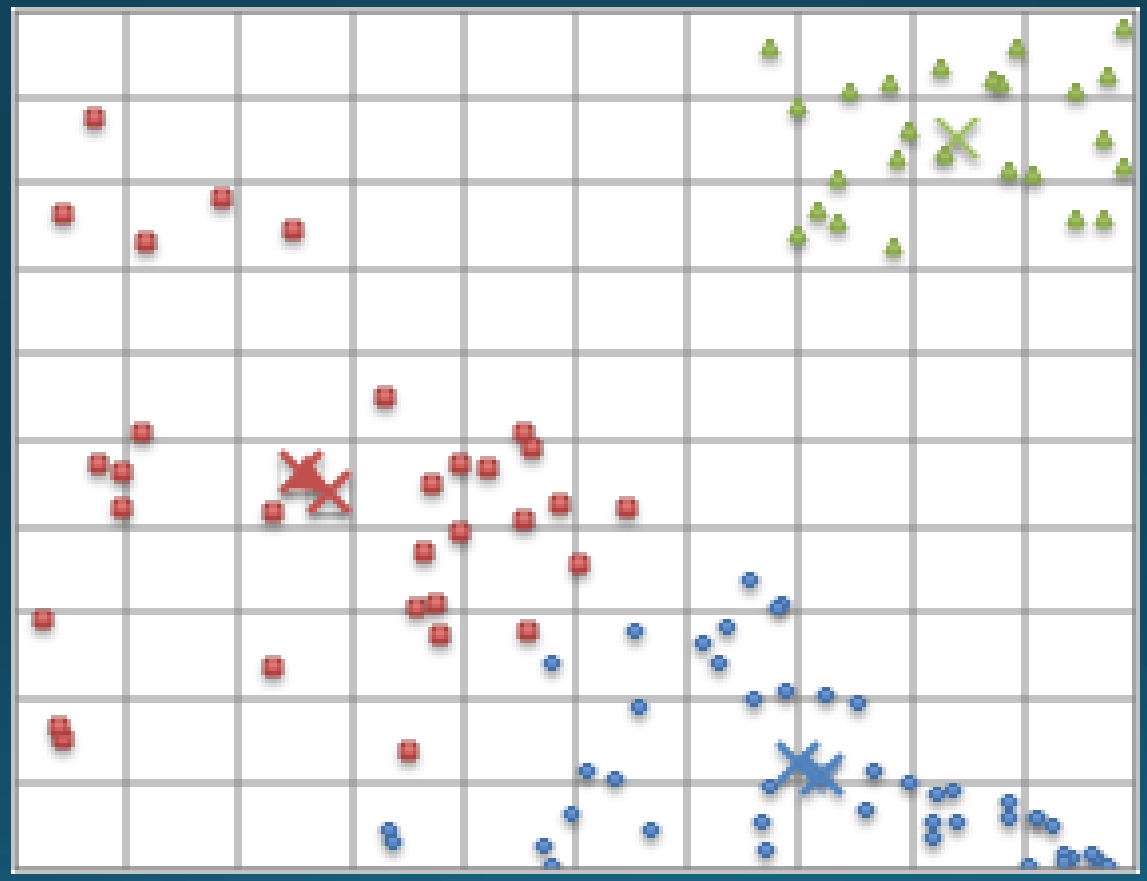
Move Centroids to the Center of the Clusters



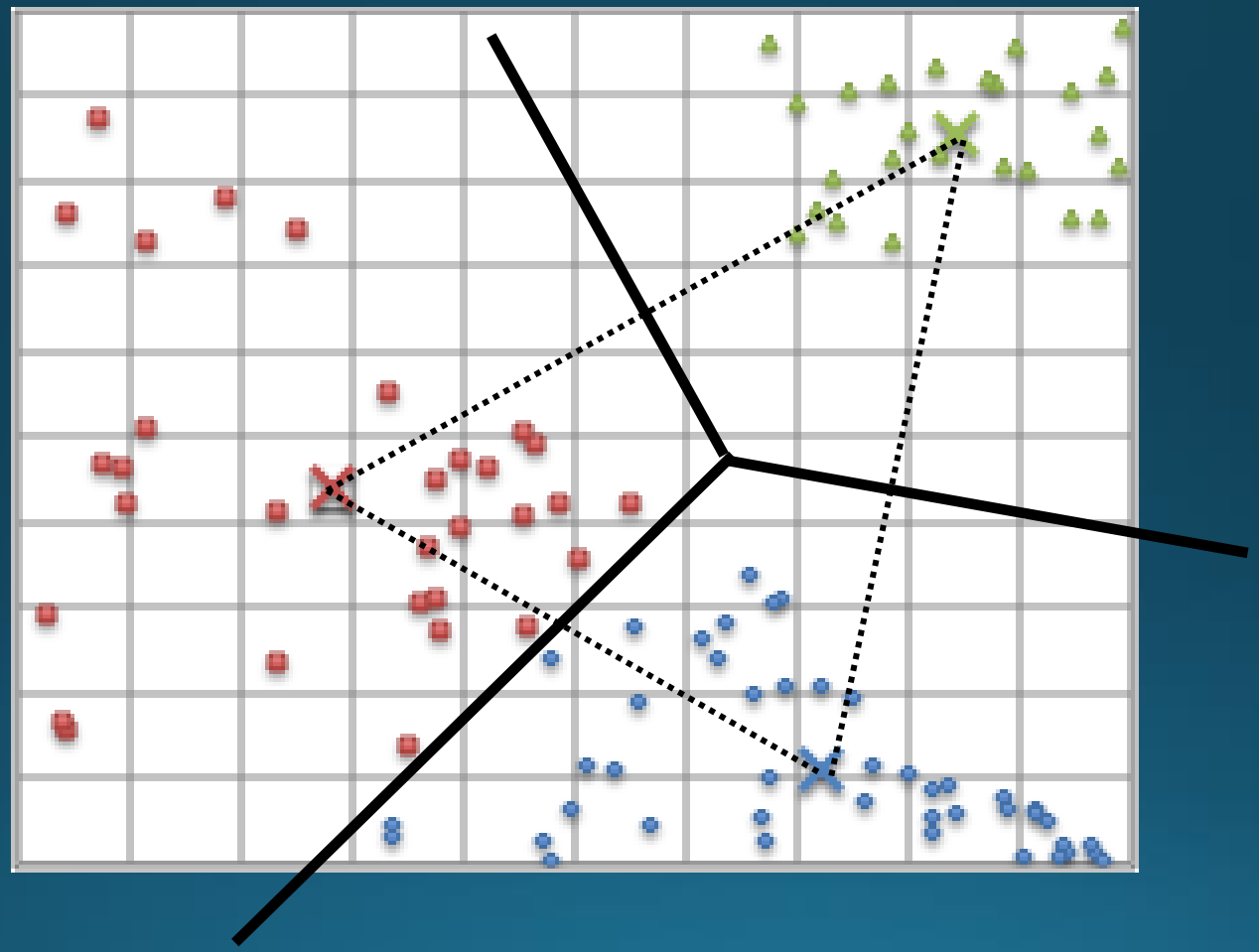
And the Process Continues



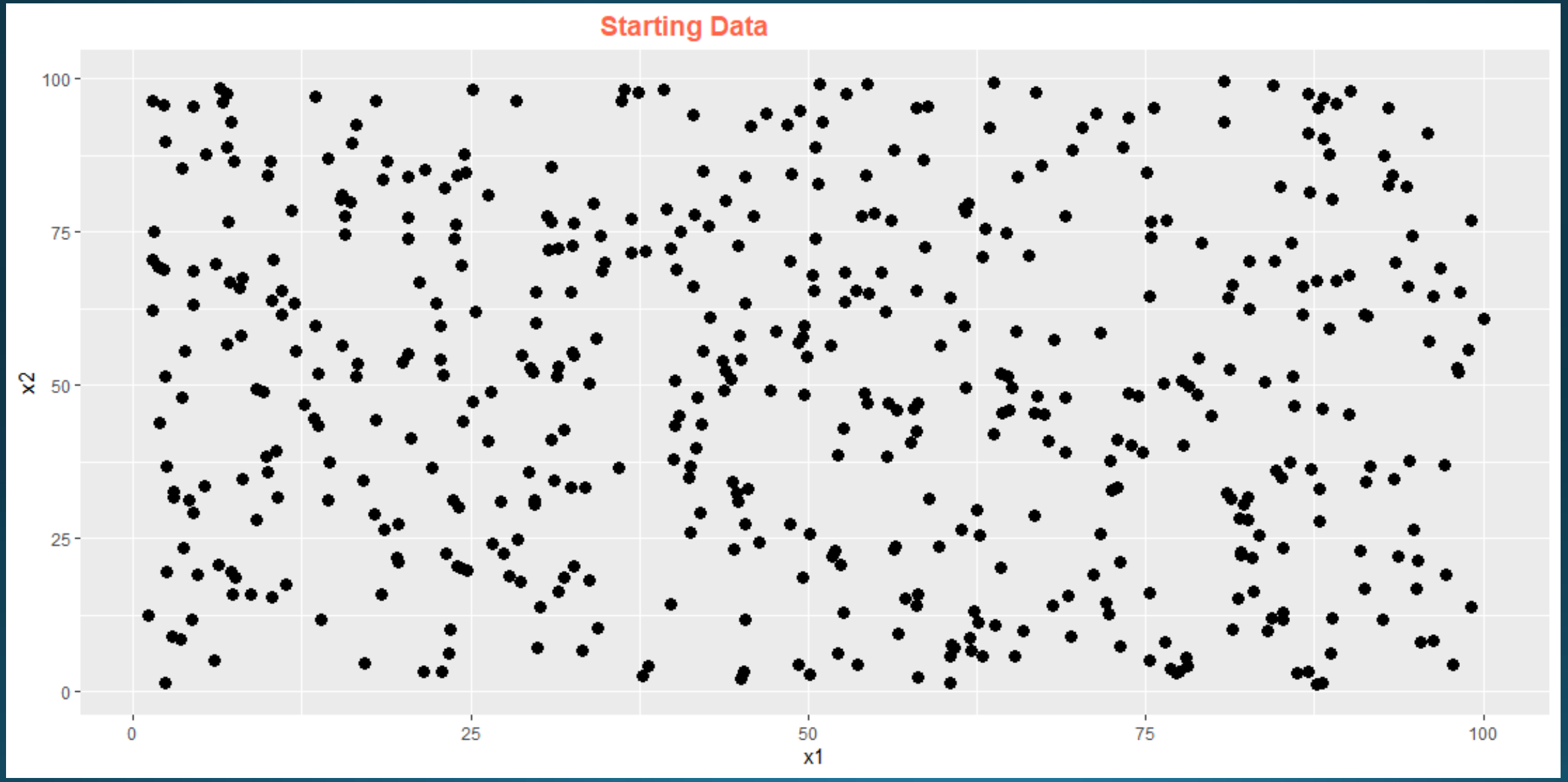
Until the Clusters Are Stable

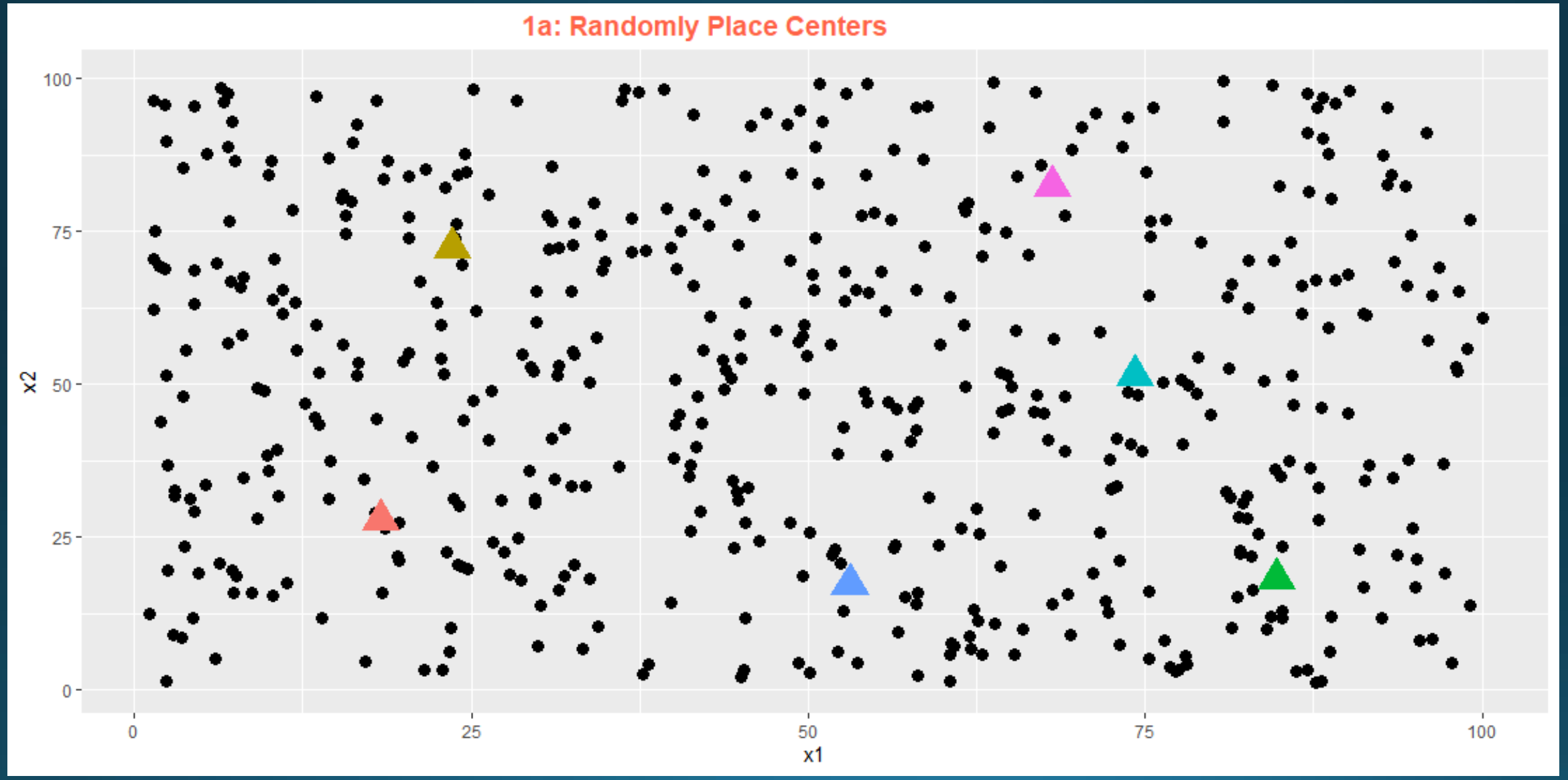


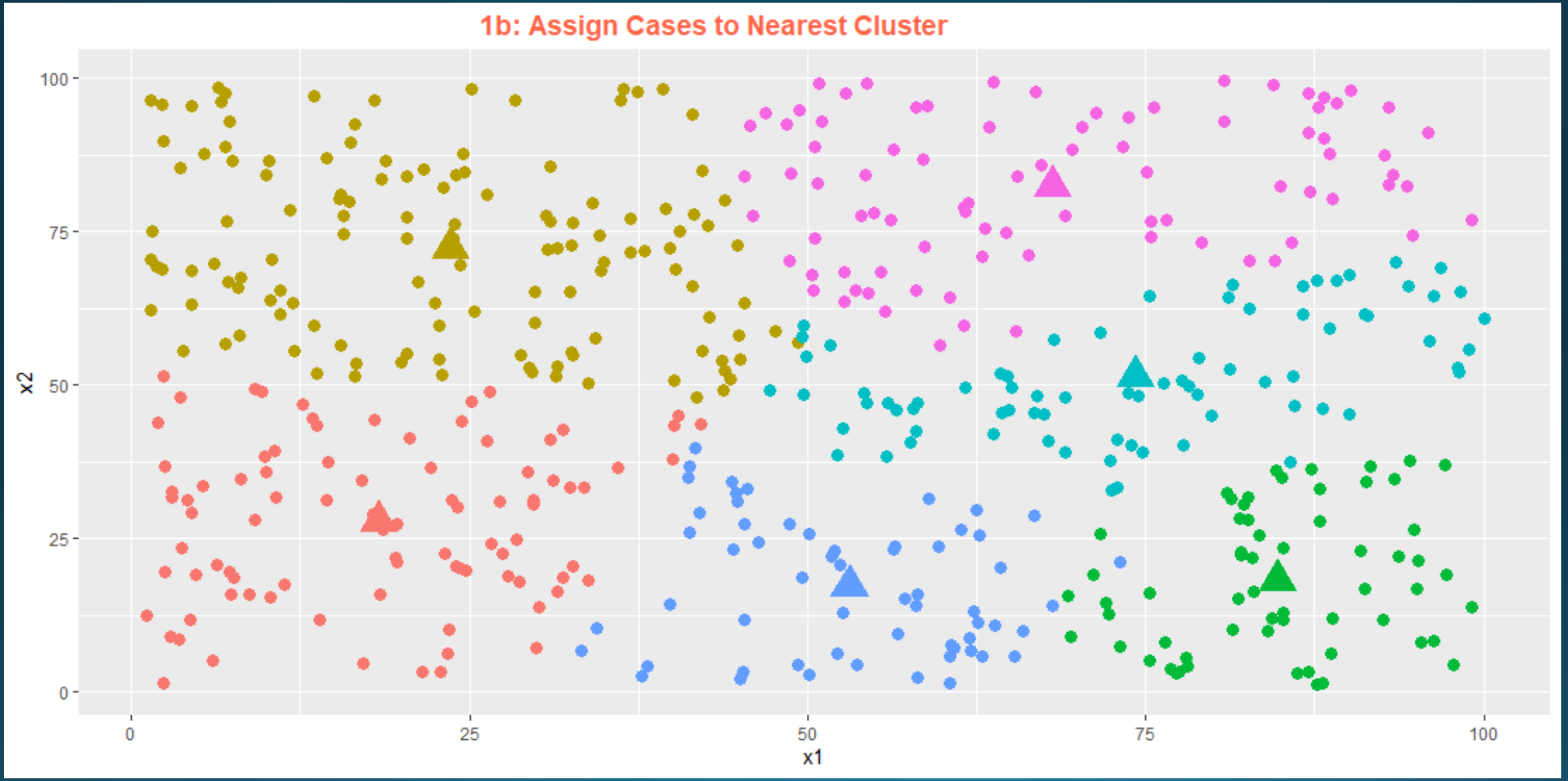
The Final Cluster Assignments

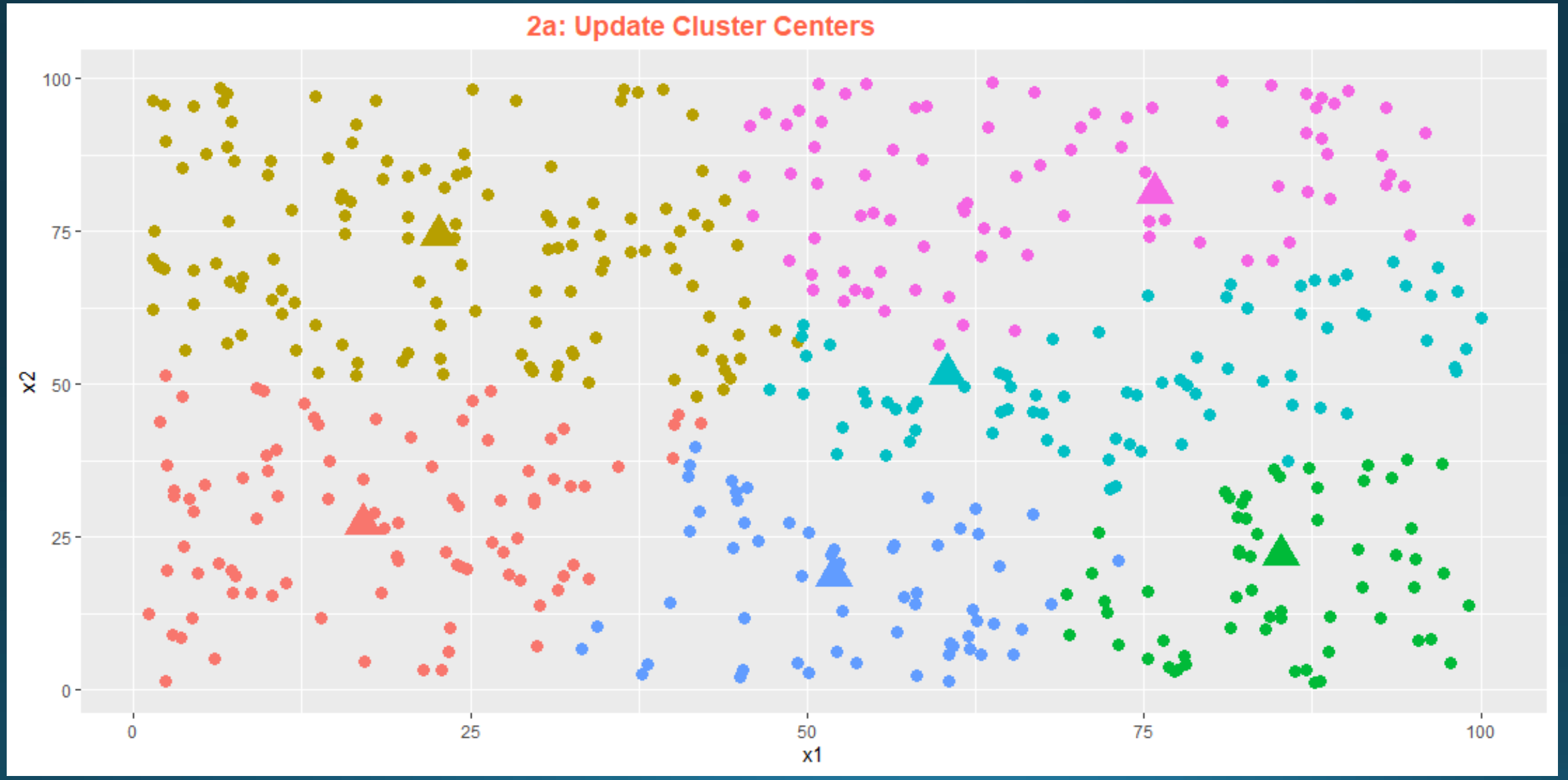


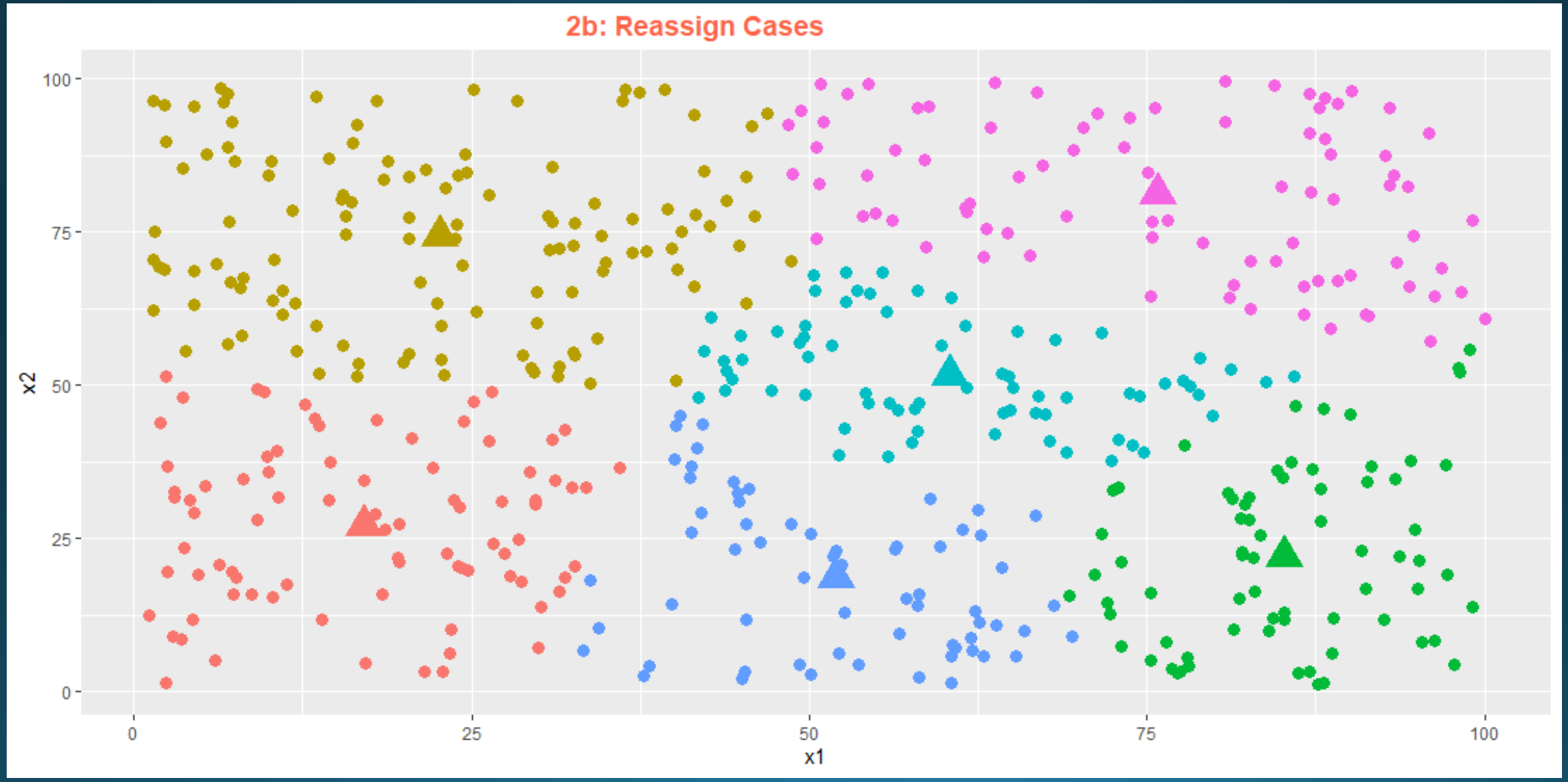
k_means_process.gif

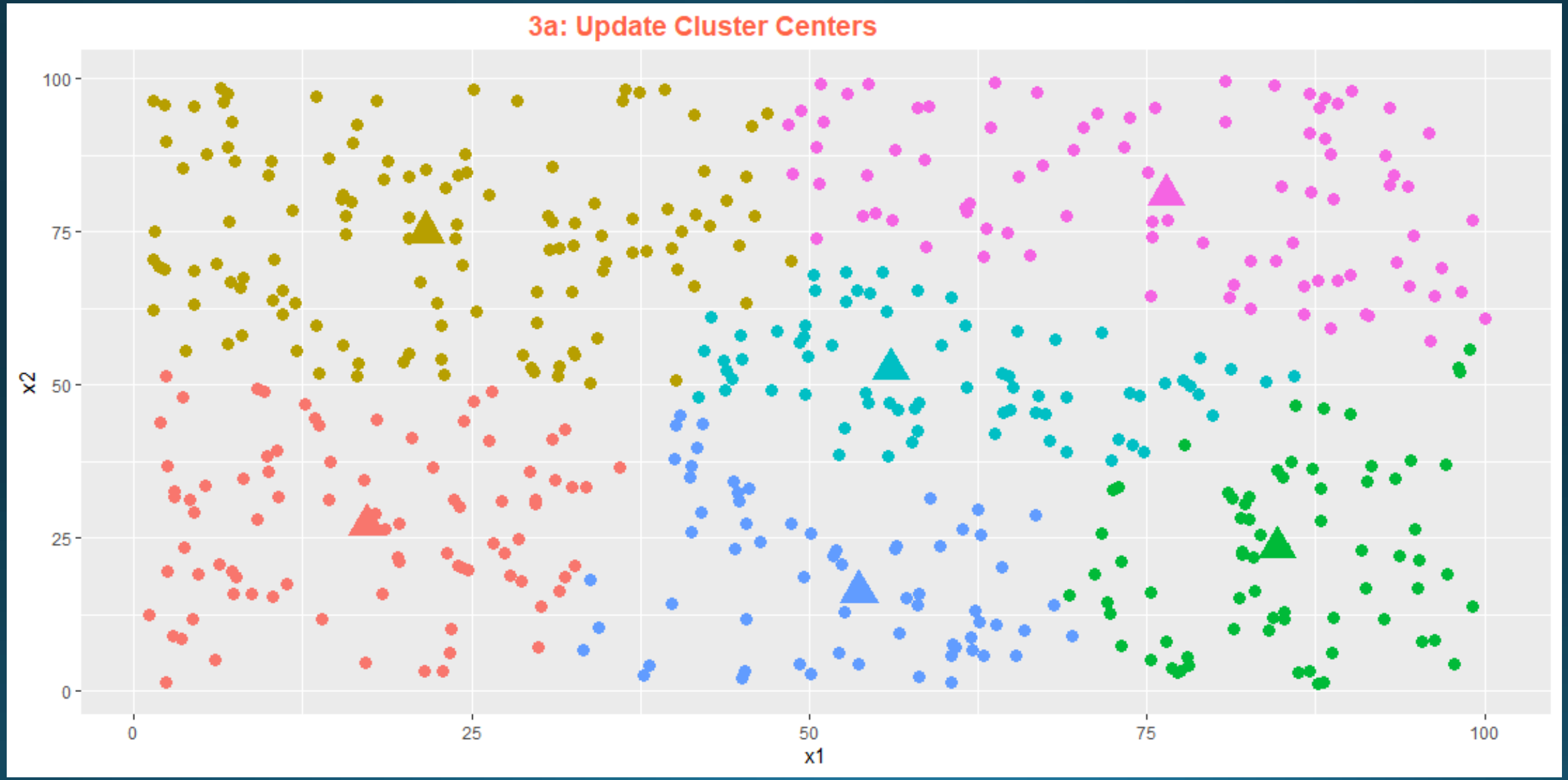


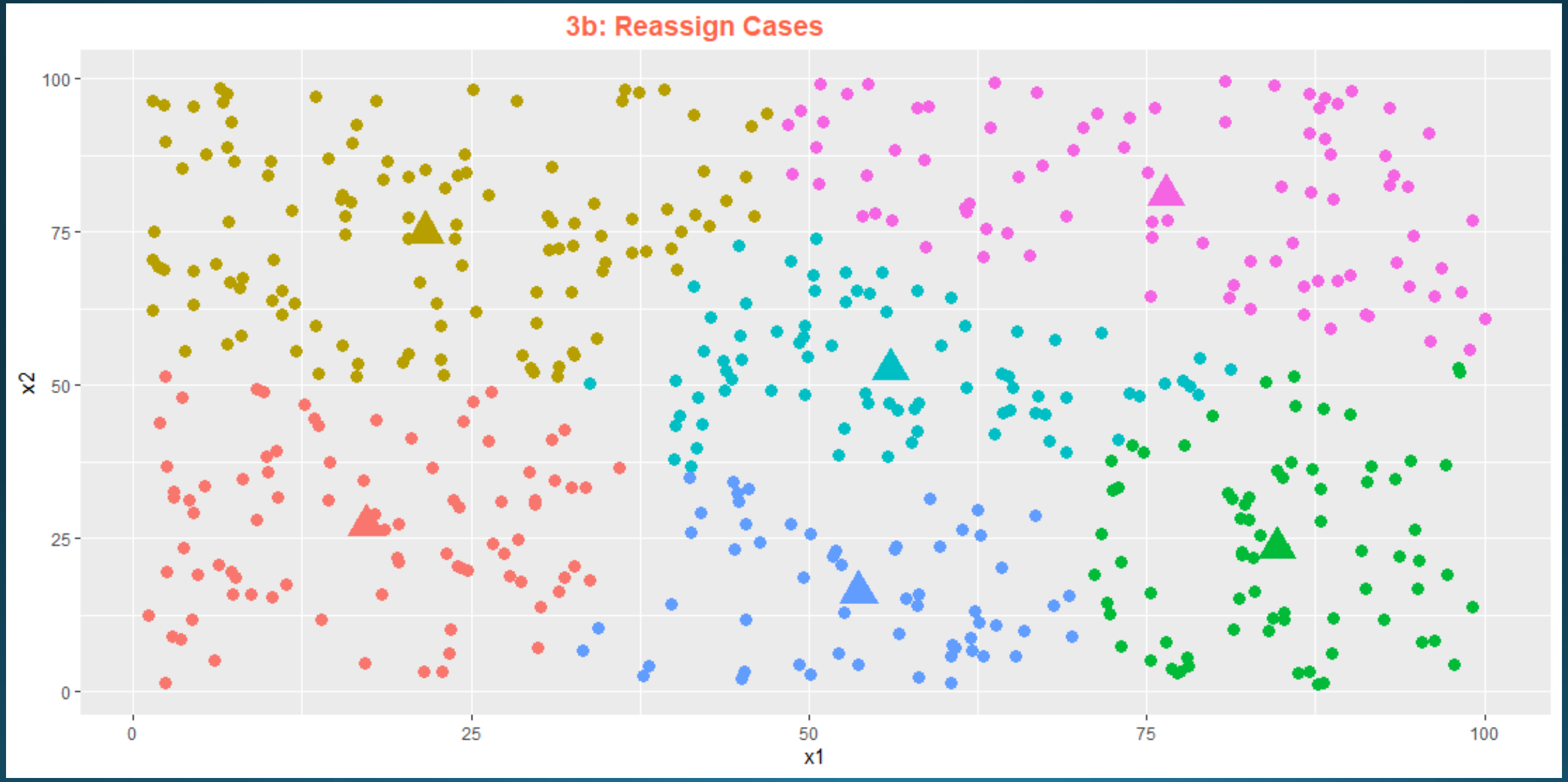

1a: Randomly Place Centers
100
75
50
25
0
x2
0
25
50
75
100
x1

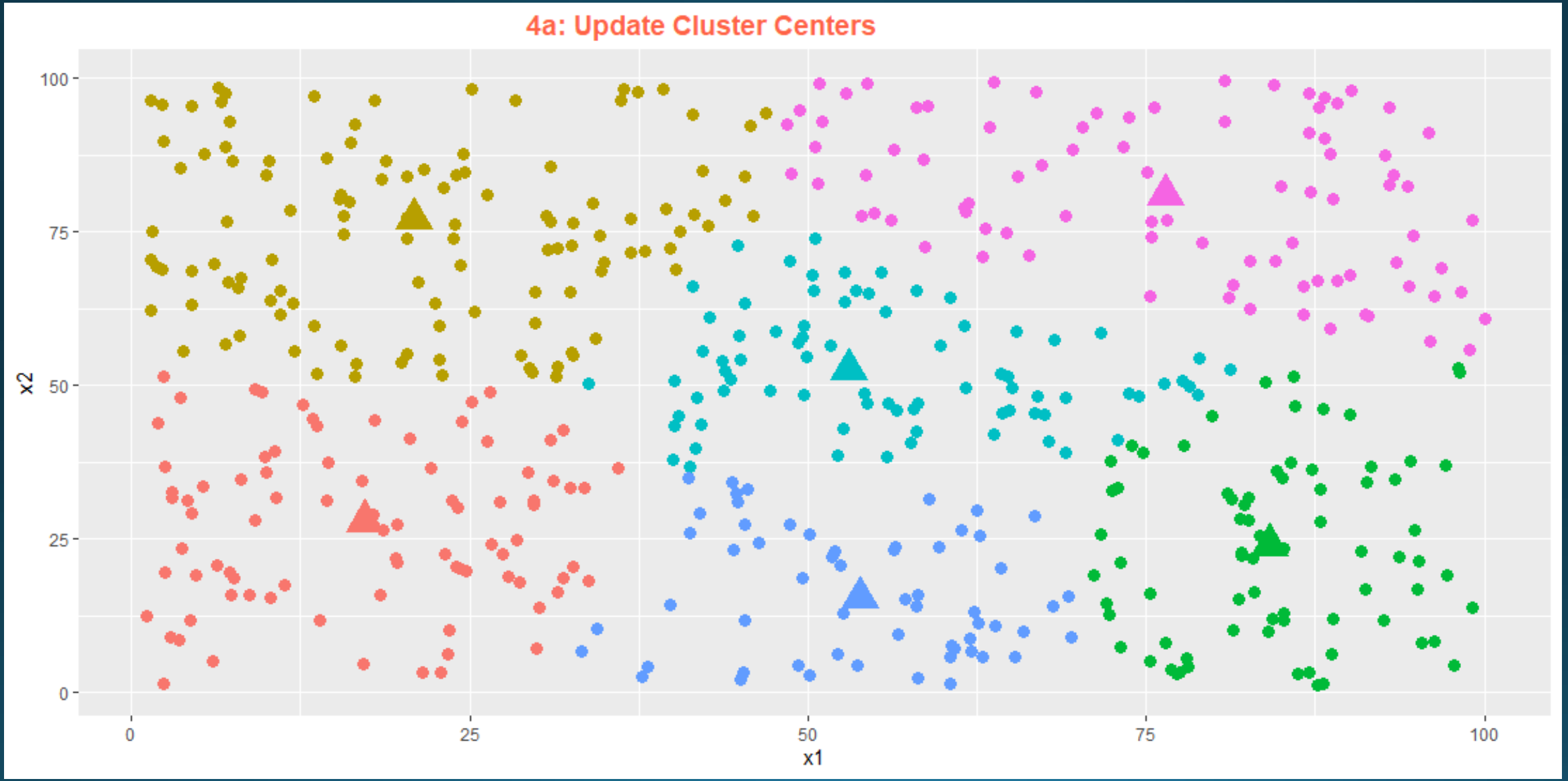

1b: Assign Cases to Nearest Cluster
100
75
50
25
0
x2
0
25
50
75
100
x1

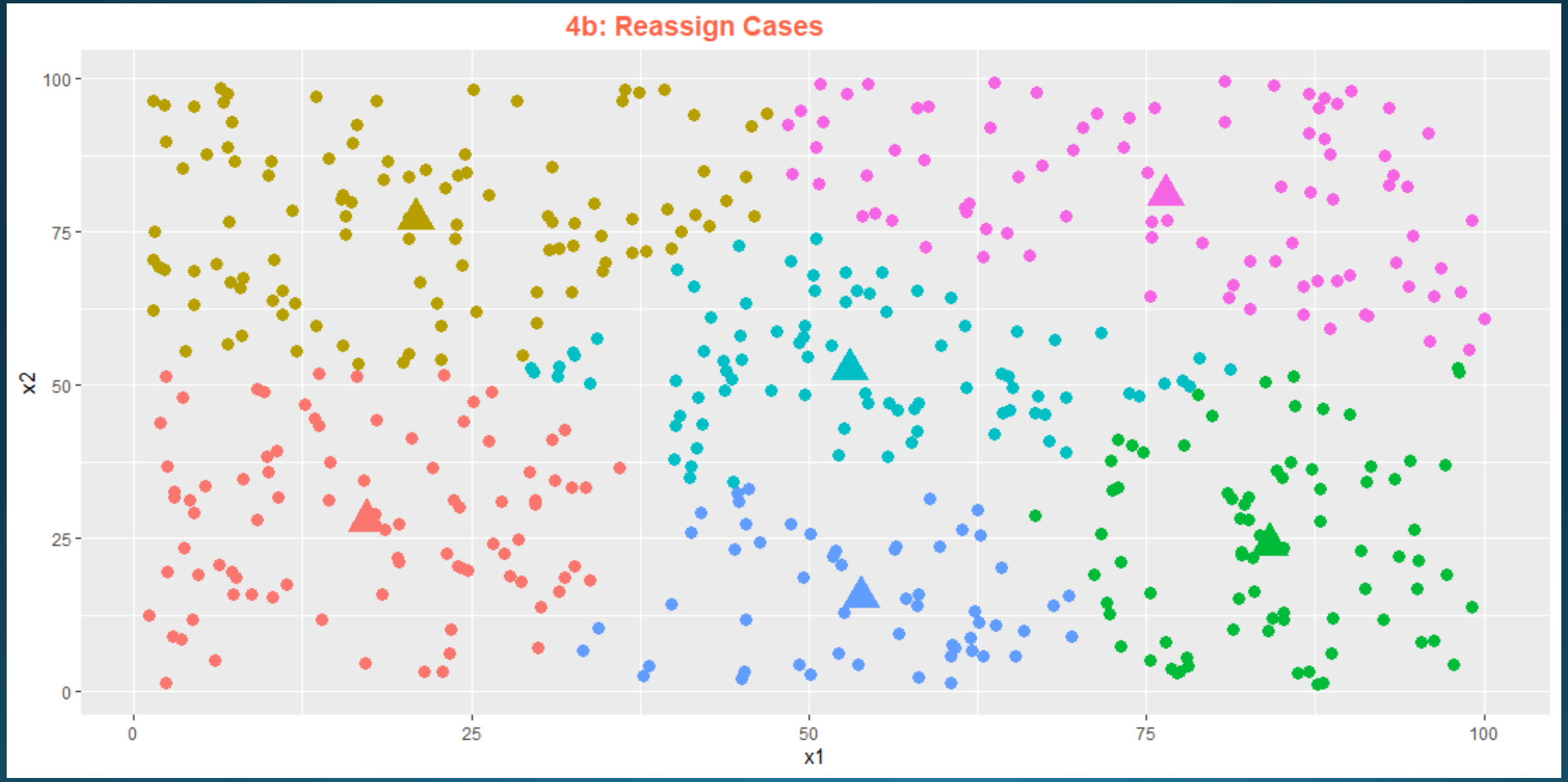

2a: Update Cluster Centers
100
75
50
25
0
x2
0
25
50
75
100
x1

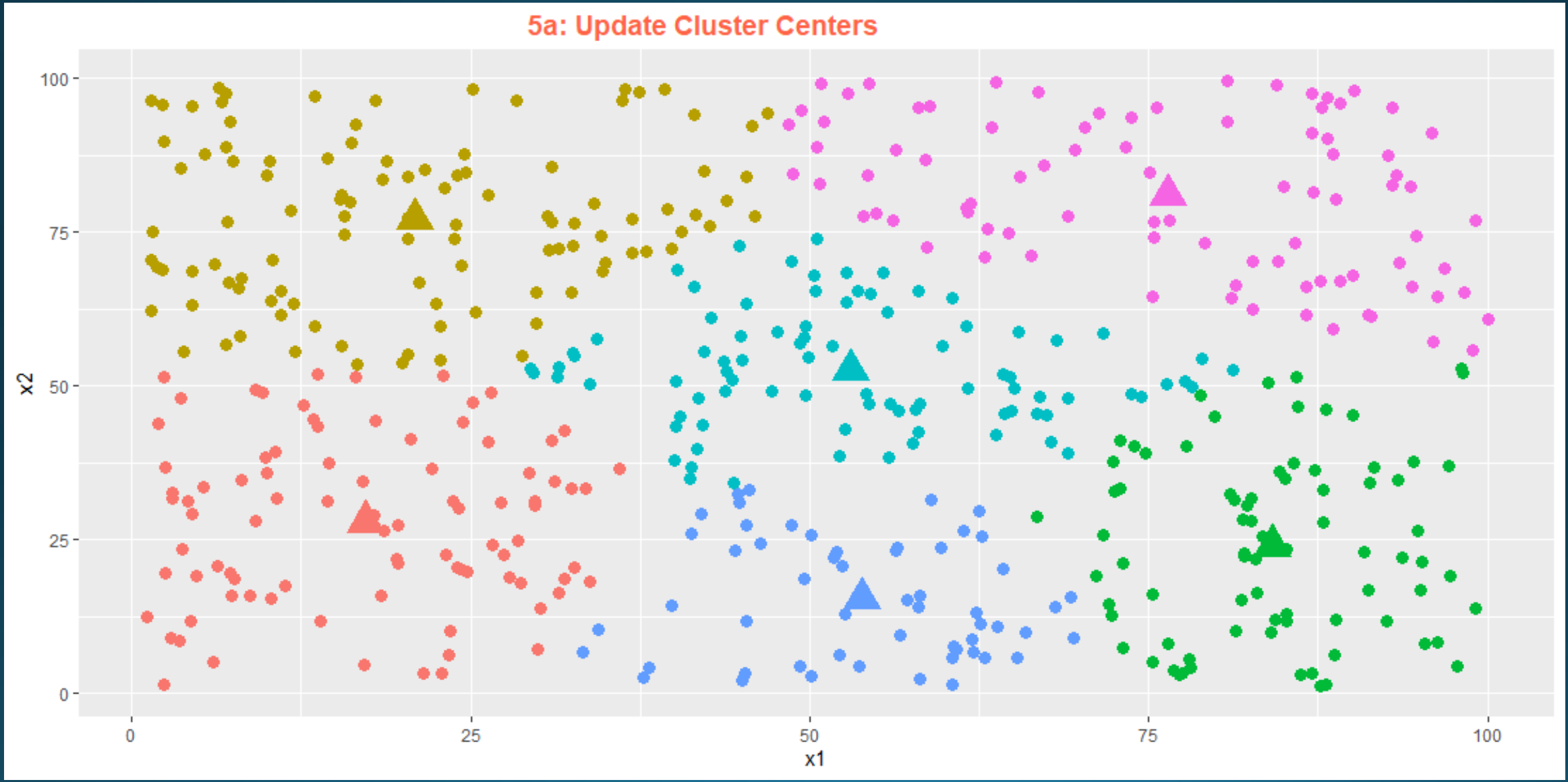

2b: Reassign Cases
100
75
50
25
0
x2
0
25
50
75
100
x1

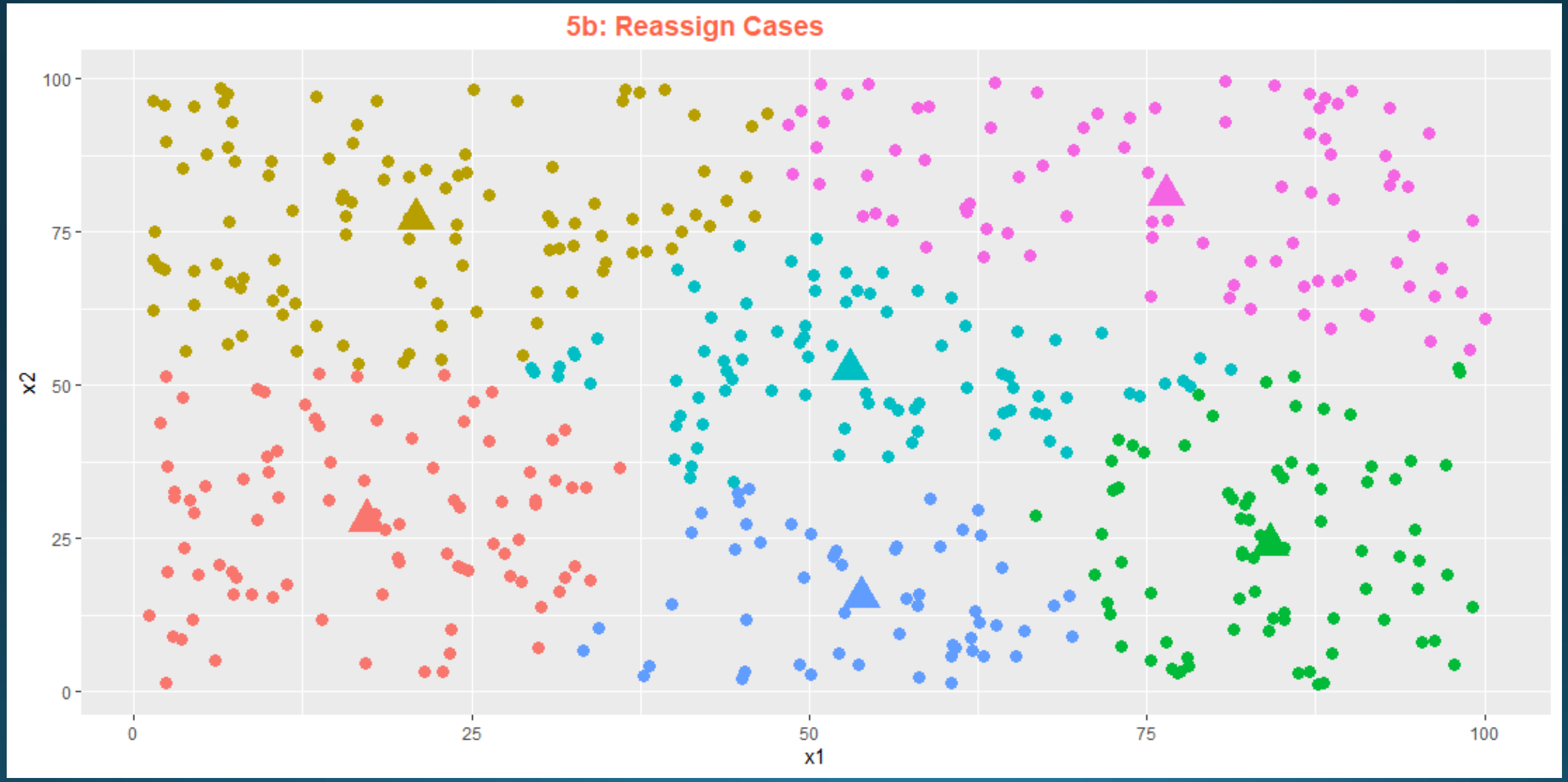

3a: Update Cluster Centers
100
75
50
25
0
x2
0
25
50
75
100
x1


3b: Reassign Cases
100
75
50
25
0
x2
0
25
50
75
100
x1


4a: Update Cluster Centers
100
75
50
25
0
x2
0
25
50
75
100
x1


4b: Reassign Cases
100
75
50
25
0
x2
0
25
50
75
100
x1


5a: Update Cluster Centers
100
75
50
25
0
x2
0
25
50
75
100
x1


5b: Reassign Cases
100
75
50
25
0
x2
0
25
50
75
100
x1

Entire iterative process

Take a look at *k_means_process.gif* for an animated version of the iterative process.

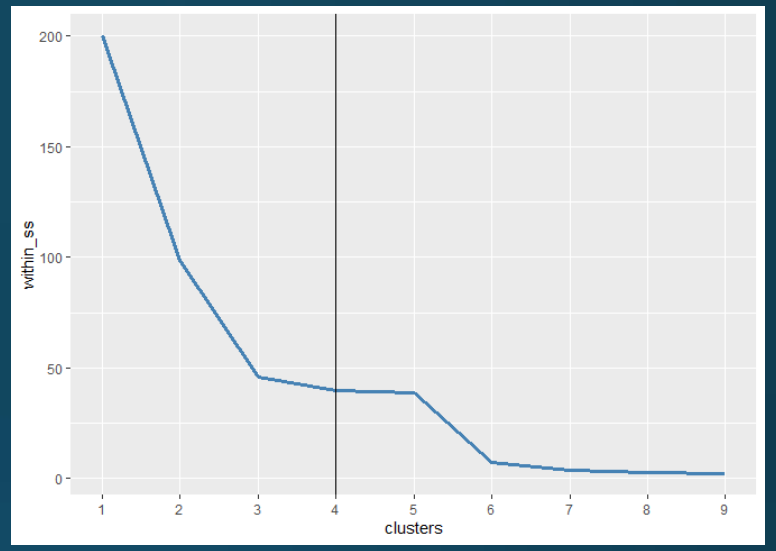
k: Number of Clusters

- For k-means, the decision of k must be made before running the algorithm
- Initial choice of k must be well thought-out and rely on domain knowledge as it drives the cluster solution
- There are also a few data-driven approaches to determining clusters. Two of them are
  - Total within cluster sum of squares
  - Silhouette Method

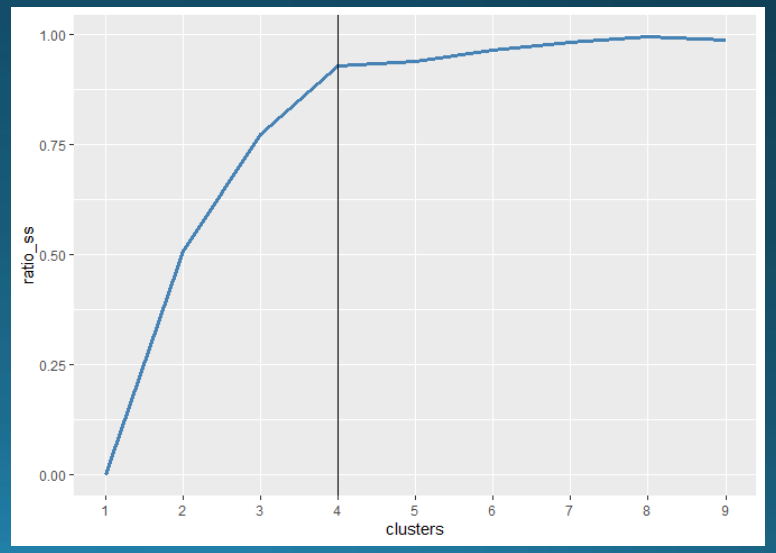
k: Number of Clusters - Within sum of squares

- Total within sum of squares plot
  - Compute total within sum of squares for a number of values of k.
  - Plot a line graph of k (on x-axis) against total within sum of squares (on y-axis).
  - Ideal number of clusters is inferred from a sudden change in the line graph or what is commonly known as the "elbow".
- Ratio plot
  - Another alternative that generates a similar conclusion
  - Compute the ratio of between cluster sum of squares and total sum of squares for a number of values of k.
  - Ideal number of clusters is inferred from the elbow in the graph.

- Total within sum of squares plot

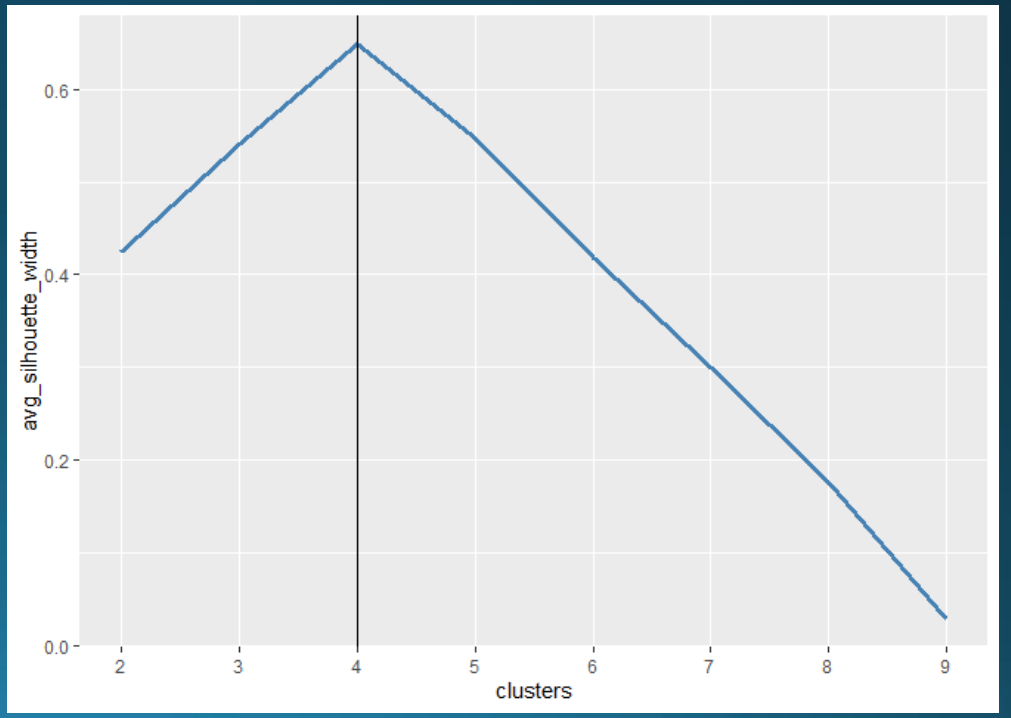


- Ratio plot

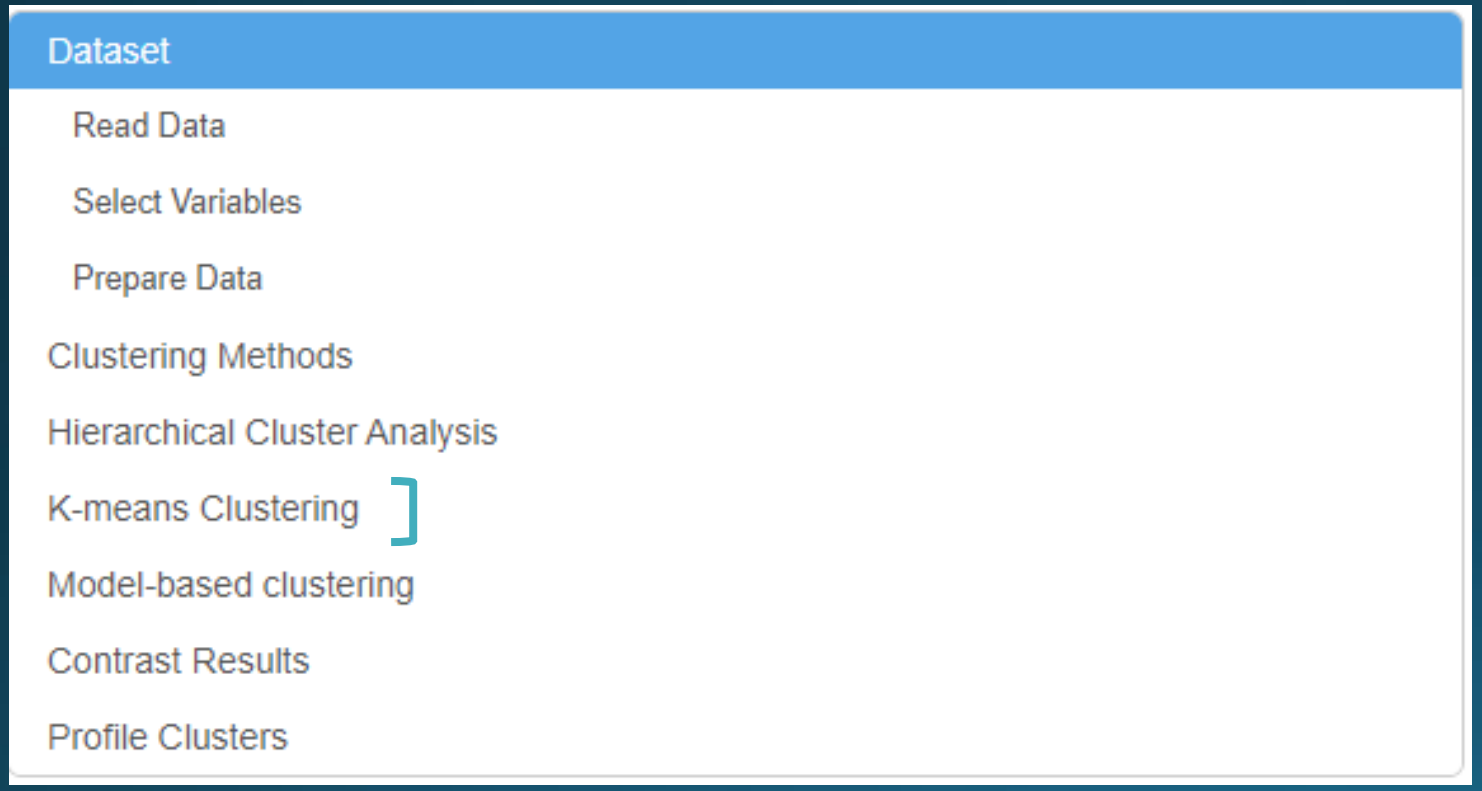


k: Number of Clusters
Silhouette Method

- Silhouette analysis reflects how well points fit in their respective clusters
- It involves computing the silhouette width(s(i)) for each point
- s(i) = (b(i) – a(i))/max{a(i),b(i)}
  - where
    - a(i) is average distance of a point to all observations within its cluster
    - b(i) is the average distance of a point to all observations in the nearest cluster
- -1 <= s(i) <= 1
  - s(i) = 1 implies i is in the right cluster
  - s(i) = 0 implies i is between two clusters
  - s(i) = -1 implies i should be in the nearest cluster



R Illustration:
Clustering for Segmentation



Compute Similarity Measure

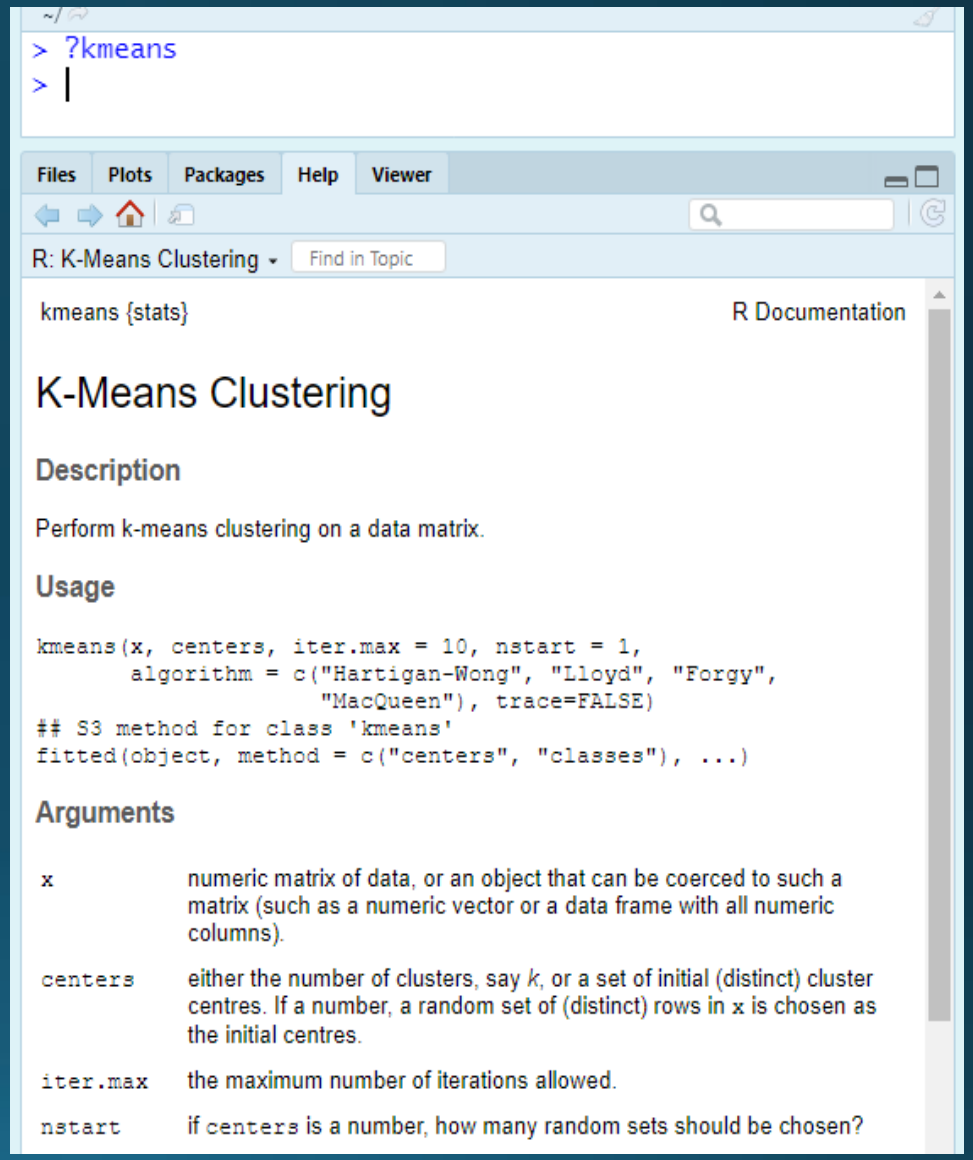
For k-means, the distance is computed using Euclidean distance.

Clustering Method

Unlike the agglomerative process of hierarchical clustering, k-means begins with an arbitrary assignment of observations to cluster centroids. (Seed is important for reproducibility). Then it uses an iterative process that involves updating cluster centers and reassigning observations. Updating is done to minimize sum of squares from observations to cluster centers. Method of updating is defined by algorithms including "Hartigan-Wong" (default), "Lloyd", "Forgy", "MacQueen"

Let us begin with an arbitrary choice of 3 centers. Later, we will examine objective methods for picking the number of centers.

```
set.seed(617)
km = kmeans(x=data_cluster, centers=3, iter.max=10000, nstart=25)
```



Here are the number of observations in the resulting clusters

```
k_segments = km$cluster
table(k_segments)
```

```
## k_segments
##   1   2   3
## 193 253  54
```

k-means examines sum of squared distances in evaluating cluster solutions. The total sum of squares is the sum of total within-cluster sum of squares and the between cluster sum of squares

```
paste(km$totss, '=', km$betweenss, '+', km$tot.withinss, sep = ' ')
```

```
## [1] "5988 = 2098.09016012787 + 3889.90983987213"
```

```
km$totss == km$betweenss + km$tot.withinss
```

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

Intepret Results

Number of Clusters

Aside from domain knowledge driven a priori choices of number of clusters, there are also a few data-driven approaches to determine cluster solutions. These methods rely on measures of distance of points from the clusters they are assigned to. The three methods we will look at include

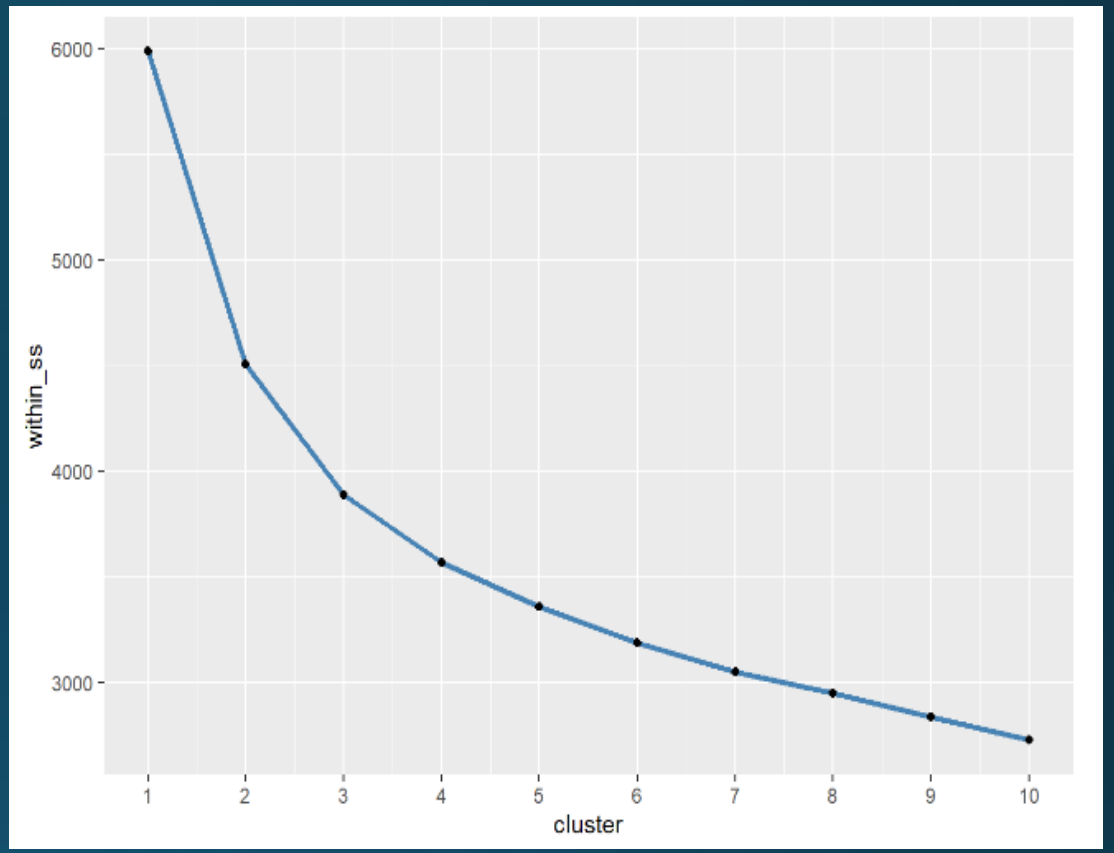
- Total within sum of squares plot
- Ratio plot
- Silhouette plot

Total within sum of squares Plot

Compute total within sum of squares for a number of values of k. Plot a line graph of k (on x-axis) against total within sum of squares (on y-axis). Ideal number of clusters is inferred from a sudden change in the line graph or what is commonly known as the "elbow".

To construct the plot, we will get the total within sum of squares for a set of 10 cluster solutions.

```
within_ss = sapply(1:10,FUN = function(x){
  set.seed(617)
  kmeans(x=data_cluster, centers=x, iter.max=1000, nstart=25)$tot.withinss})

ggplot(data=data.frame(cluster = 1:10,within_ss), aes(x=cluster,y=within_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))
```



The total within cluster sum of squares for a two-cluster and three cluster solution?

```
within_ss[2]
```

```
## [1] 4504.764
```

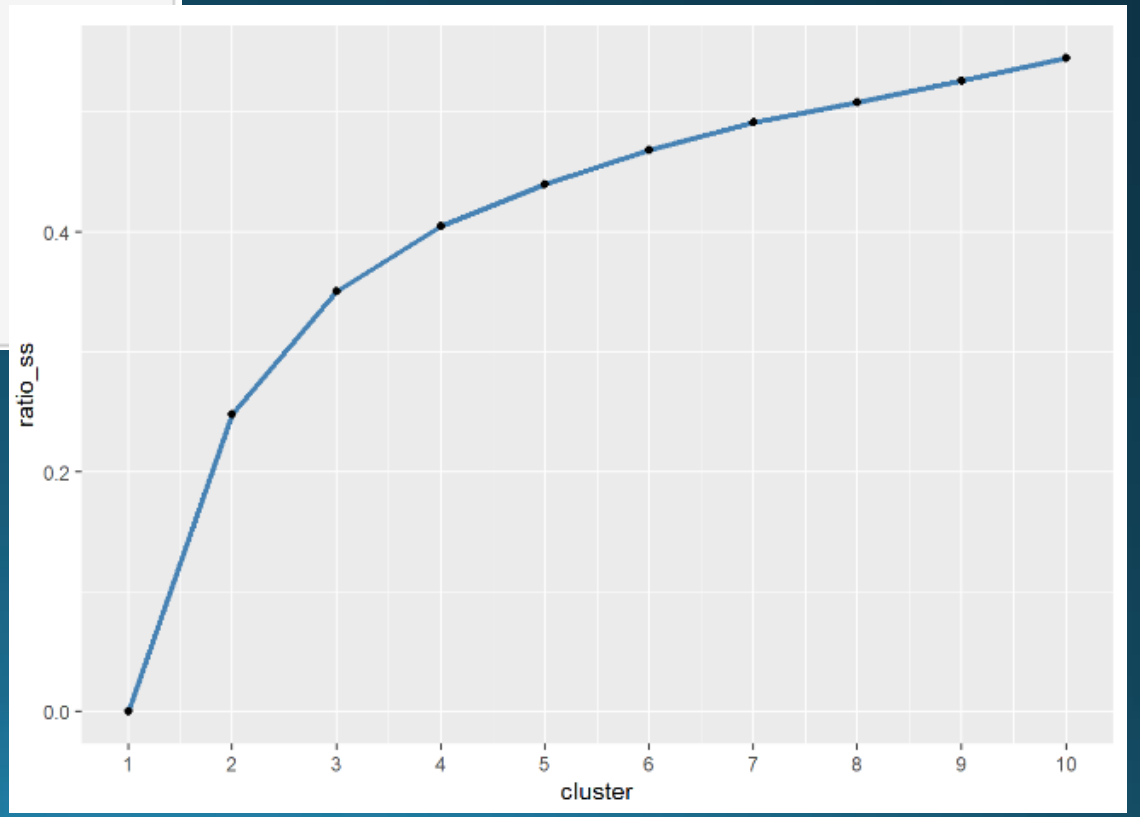
```
within_ss[3]
```

```
## [1] 3889.91
```

Ratio Plot

Another alternative that generates a similar conclusion is to compute the ratio of between cluster sum of squares and total sum of squares for a number of values of k. Ideal number of clusters is inferred from the elbow in the graph.

```
ratio_ss = sapply(1:10,FUN = function(x) {
  set.seed(617)
  km = kmeans(x=data_cluster, centers=x, iter.max=1000, nstart=25)
  km$betweenss/km$totss} )

ggplot(data=data.frame(cluster = 1:10,ratio_ss), aes(x=cluster,y=ratio_ss))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(1,10,1))
```



Silhouette Plot

Silhouette analysis reflects how well points fit in their respective clusters. It involves computing the silhouette width(s(i)) for each point

s(i) = (b(i) - a(i))/max{a(i),b(i)}

where

- a(i) is average distance of a point to all observations within its cluster
- b(i) is the average distance of a point to all observations in the nearest cluster

-1 <= s(i) <= 1

- s(i) = 1 implies i is in the right cluster
- s(i) = 0 implies i is between two clusters
- s(i) = -1 implies i should be in the nearest cluster

Silhoette width is easily computed using pam() from library(cluster). Partitioning around medoids (pam) is a more robust verion of k-means that clusters data using medoids rather than centroids. This means, the Silhouette width we compute using pam() is not exactly the same as that from kmeans() but the numbers are pretty close. Since we don't have a handy function to compute silhouette width for k-means, we will use the one computed on pam() and use it instead. (You can examine the result of pam() if you are curious about this function and its attributes).

```
library(cluster)
pam(data_cluster, k=3)$silinfo$avg.width
```

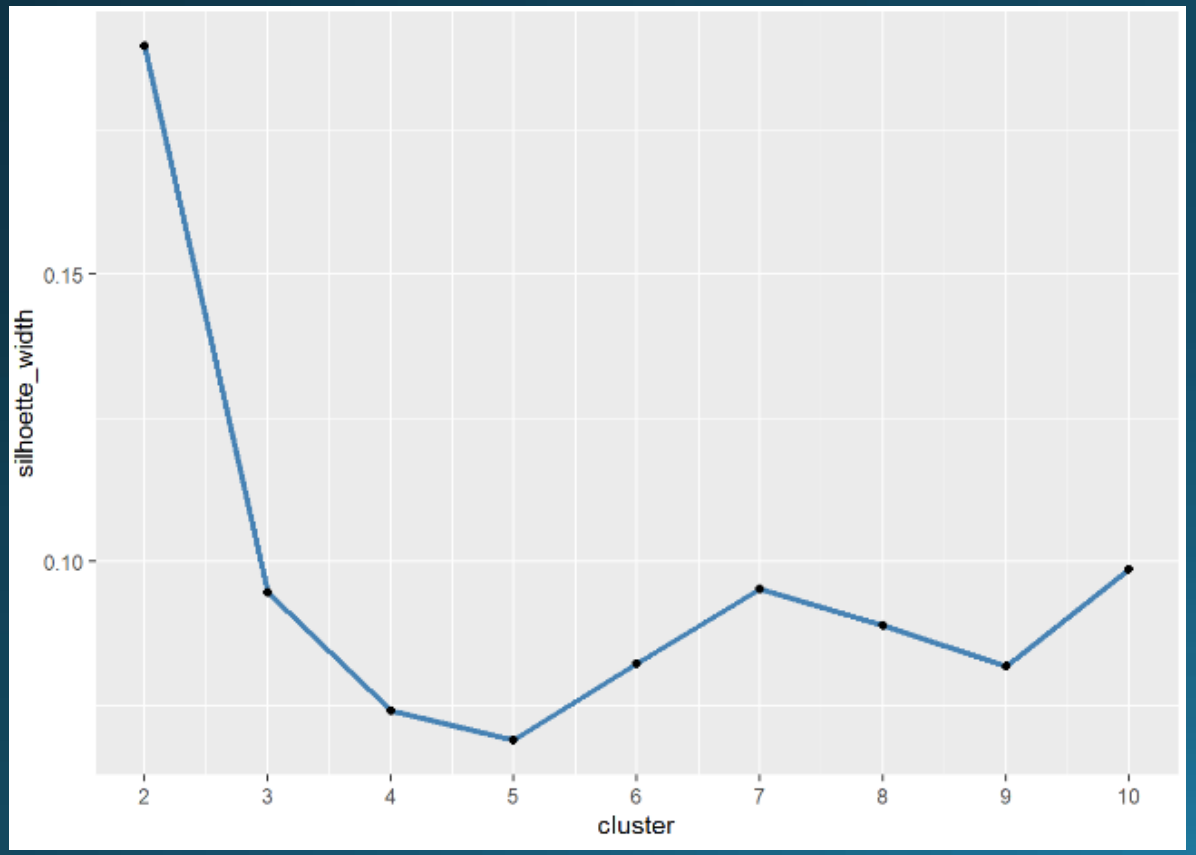
```
## [1] 0.09462411
```

We can easily modify the above code to find Silhouette width for a four-cluster solution

```
library(cluster)
pam(data_cluster, k=4)$silinfo$avg.width
```

```
## [1] 0.07399345
```

```
library(cluster)
silhoette_width = sapply(2:10,
                         FUN = function(x) pam(x=data_cluster, k=x)$silinfo$avg.width)

ggplot(data=data.frame(cluster = 2:10,silhoette_width), aes(x=cluster,y=silhoette_width))+
  geom_line(col='steelblue',size=1.2)+
  geom_point()+
  scale_x_continuous(breaks=seq(2,10,1))
```



The total within sum of square and ratio plots support a two-, three- and four-cluster solution while silhoette supports a two-cluster solution. Of these alternatives, the two- and three- cluster solutions combine a niche segment into a larger segment. (See number of observations in smallest segment from a 4-cluster solution). In order to tease out this niche segment, and focus on three dominant segments, we will go with a four-cluster solution.

```
set.seed(617)
km = kmeans(x=data_cluster, centers=4, iter.max=10000, nstart=25)
```

Size of the clusters

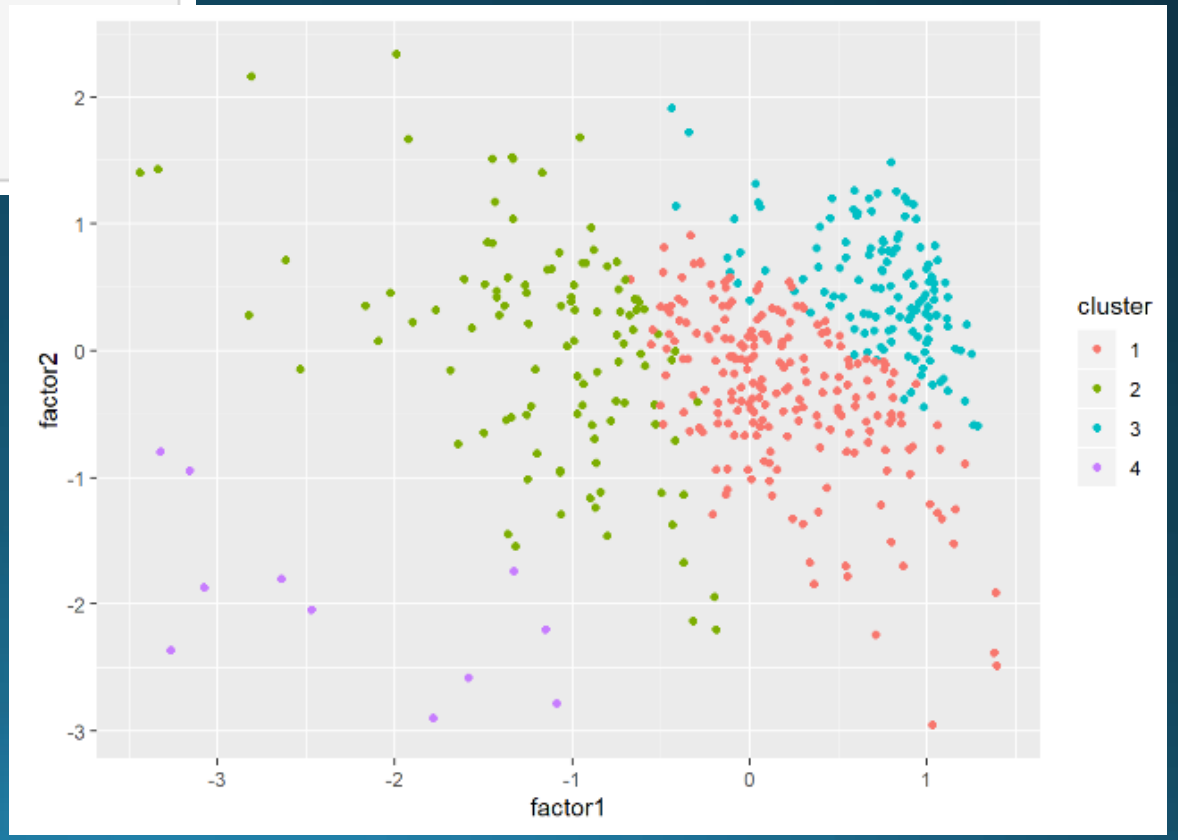
```
k_segments = km$cluster
table(k_segments)
```

```
## k_segments
##   1   2   3   4
## 221 110 155  14
```

Visualize

To express the clusters on a scatterplot, we flatten the data from 12 dimensions onto 2 by conducting a factor analysis with varimax rotation. This is done because it is not possible to visualize 12-dimensional data.

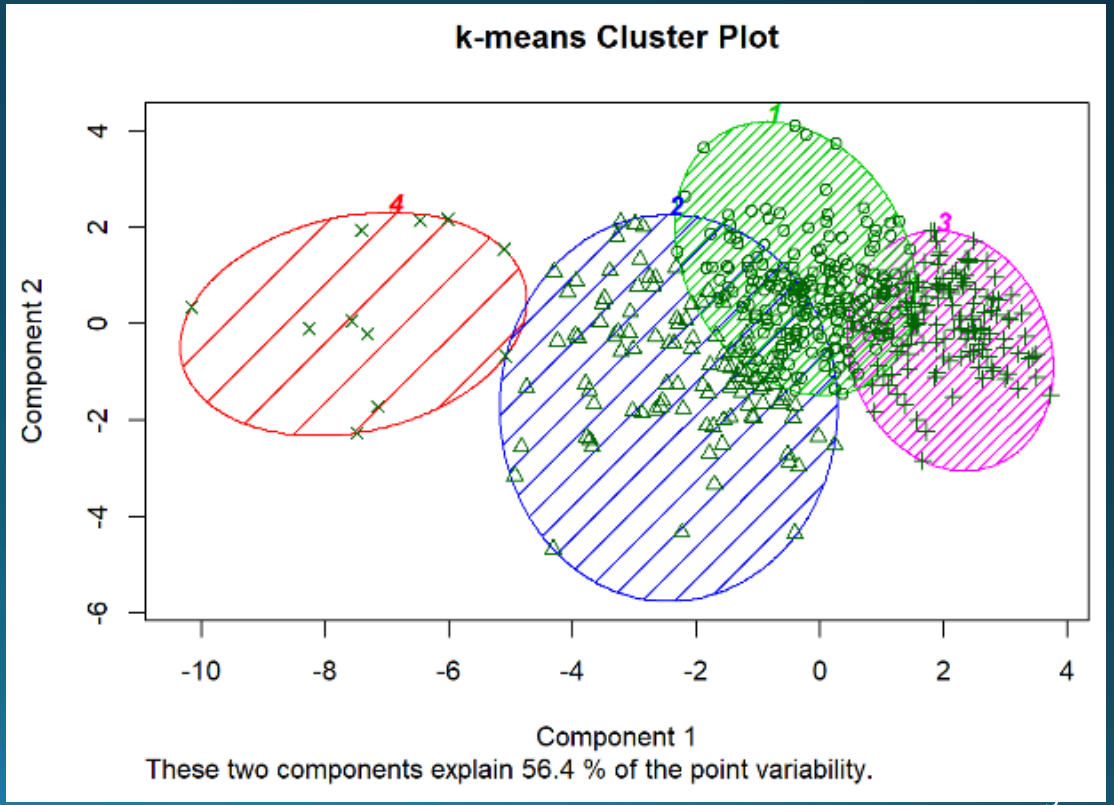
```
library(psych)
temp = data.frame(cluster = factor(k_segments),
           factor1= fa(data_cluster, nfactors=2, rotate='varimax')$scores[,1],
           factor2= fa(data_cluster, nfactors=2, rotate='varimax')$scores[,2])

ggplot(temp, aes(x=factor1,y=factor2,col=cluster))+
  geom_point()
```



Using clusplot

`clusplot` does something similar.

```
library(cluster)
clusplot(data_cluster,
         k_segments,
         color=T, shade=T, labels=4, lines=0, main='k-means Cluster Plot')
```



k-Means

- Relatively low computational power is the most important benefit of k-Means over hierarchical clustering
- However, k-means
  - is sensitive to starting values, which are based on the seed
  - only allows the use of one distance metric – Euclidean
  - lacks the benefit of a dendrogram for inferring clusters