K-Means Clustering: Selecting K.

W. Wang[1]

[1] Department of Mathematics
University of Houston

MATH 4323

Cluster Validation. Selecting K in K-Means.

Having introduced one approach for clustering - K**-means** algorithm - we are yet to discuss the following critical topics:

- How do we select the best value K (the # of clusters)?
- Given that there's no target variable and no obvious model validation approach (such as validation set or cross-validation), how do we **evaluate the quality** of our **clustering solution**?

Choosing K in K-Means: Data Segmentation Task.

A choice for the number of clusters K depends on the **goal** of the **application**.

In **data segmentation** tasks, K is usually **supplied** as part of the problem.

Example (data segmentation). Company may employ 10 sales people, and the goal is to partition a customer database into **10 segments**, one for each sales person, such that the customers assigned to each one are **as similar as possible**.

If we were to use K-means clustering, then $\boldsymbol{K = 10}$ is clearly warranted here.

Choosing K for K-Means: Data-based methods.

Often, however, cluster analysis is used to provide a data description via **natural** distinct groupings of observations. Here the number of such groups ***K**** is **unknown** and one requires that it, as well as the groupings themselves, be estimated **from the data**.

Data-based methods for estimating K^* typically examine the **within-cluster** variation W_K as a function of the **number of clusters** K:

1. A separate clustering solution is obtained for $K = 1, 2, \ldots, K_{max}$.
2. Calculate the corresponding values $\{W_1, \ldots, W_{K_{max}}\}$ of **within-cluster** variations.

Example: Human Tumor Microarray Data.

Example. For the human tumor microarray data *NCI*60 from previous slides, we apply *K*-means clustering with

- *K* running from 1 to 10,
- using 50 random initializations each time.

Then we computed the total within-sum of squares for each clustering.

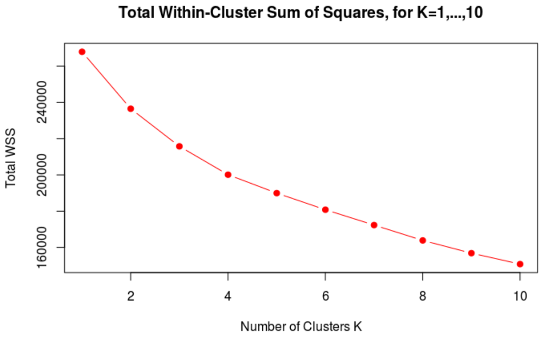
```
library(ISLR)
library(factoextra)

k.max <- 10
wss <- numeric(k.max)
for (k in 1:k.max){
  wss[k] <- eclust(NCI60$data,
                   FUNcluster="kmeans",
                   k = k,
                   nstart = 50,
                   graph=0)$tot.withinss
}
```

Example: Human Tumor Microarray Data.

Example (cont'd). Plot of WSS for resulting $K = 1, \ldots, 10$-cluster solutions is on the right.

$$W_1 > W_2 > \cdots > W_{K_{max}}$$

Values $\{W_1, \ldots, W_{K_{max}}\}$ generally decrease as $K \Uparrow$, as the more clusters created $\Rightarrow$ the higher the density of each cluster (just



imagine the case of $K = n$, with each observation being its own cluster). Thus **cross-validation** techniques, so useful for model selection in supervised learning, cannot be utilized in this context.

Choosing K for K-Means: Data-based methods.

Intuition. If there were truly K^* distinct groupings of the observations:

- Performing K-means with $K < K^*$, some clusters will contain representatives of ≥ 2 distinct groupings, which leads to them being **heterogeneous** (having **high** within-cluster variation W_K),
- With each successive increase in K, up until $K \equiv K^*$, the value W_k should decrease **substantially** $\Leftrightarrow W_K >> W_{K+1}, K < K^*$, due to the natural groups of observations being successively assigned to separate clusters.
- Splitting a natural group, within which the observations are all quite close to each other, reduces the criterion **less** than partitioning the union of two well-separated groups into their proper constituents.

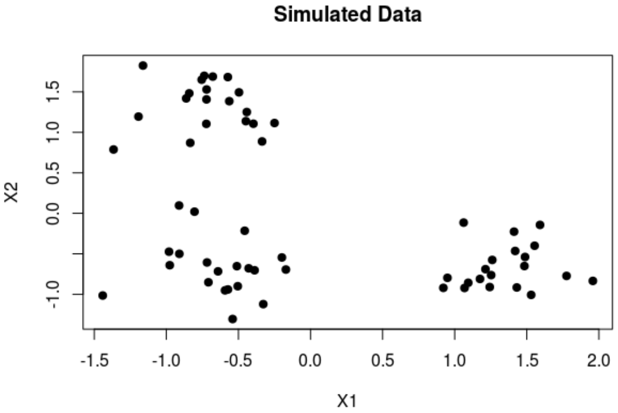
Data-based methods, Intuition: Simulated Example.

Example. Let's generate a simulated data example in $2D$ space with $K = 3$ clear, well-separated, clusters of observations.

```
> set.seed(1)
> x <- matrix(rnorm(20*2*3), ncol=2)
>
> # Shifting coordinates to separate observations
> # into 3 clusters.
> x[21:40,2] <- x[21:40, 2] + 6
> x[41:60,1] <- x[41:60, 1] + 6
> x <- scale(x)  # Just scale for consistency.
>
> plot(x, xlab = "X1", ylab = "X2",
      main="Simulated Data")
```

Data-based methods, Intuition: Simulated Example.

Example (cont'd). Below one can see three clusters of observaions: top-left, bottom-left and bottom-right parts of 2D predictor space.



Data-based methods, Intuition: Simulated Example.

Example (cont'd). We will fit K-means for $K = 1, 2, 3$ & 4, compare the resulting clusters and within-cluster sums of squares.

Library *factoextra* presents a set of functions

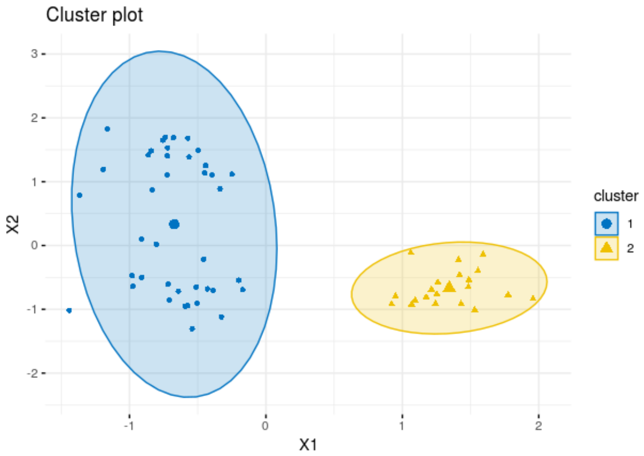
- to obtain clustering solutions (function *eclust*()),
- select a **K** for **K**-means, if needed,
- visualize the clustering solutions (function *fviz_cluster*())

To obtain a K-means solution for $K = 2$, along with visualization:

```
library(factoextra)

km.res <- eclust(data.frame(x),
                 FUNcluster = "kmeans", k=2)
print(km.res$tot.withinss)

fviz_cluster(km.res, geom = "point", ellipse.type = "norm",
             palette = "jco", ggtheme = theme_minimal())
```

Data-based methods, Intuition: Simulated Example.

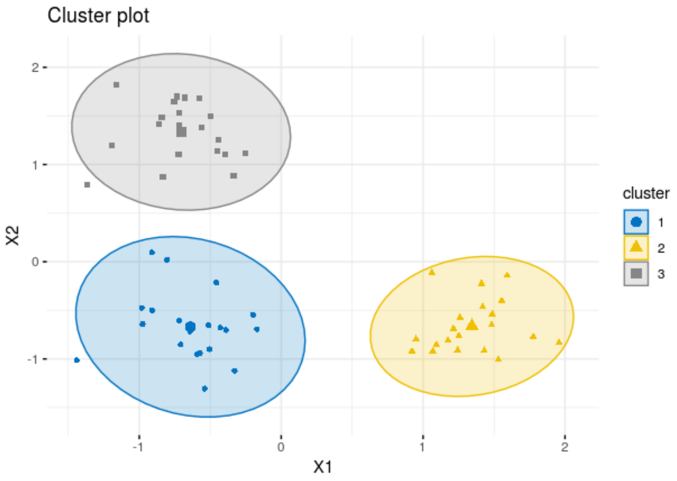
Example (cont'd). For **K=1**, we have **WSS=118**, plot not needed.
For **K=2**, we have **WSS=50.5**, and resulting plot:



See how **heterogeneous** the blue cluster is, albeit a **big** drop in **WSS**.

Data-based methods, Intuition: Simulated Example.

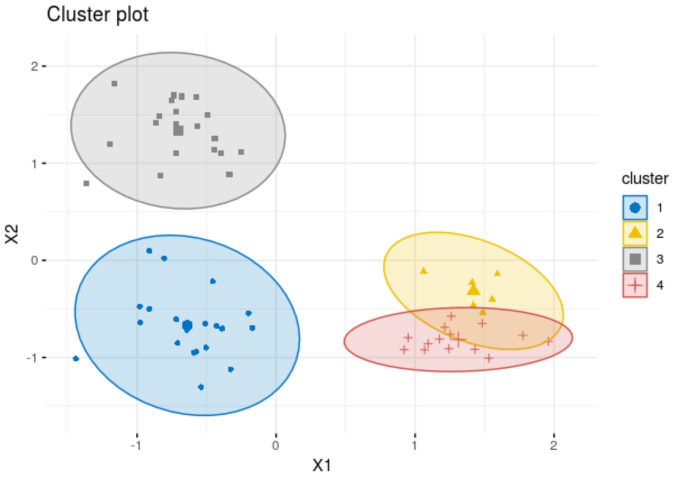
Example (cont'd). For **K=3**, we have **WSS=10.2**, and resulting plot:



All three clusters are **well-separated** and **dense**, **big** drop in **WSS**.

Data-based methods, Intuition: Simulated Example.

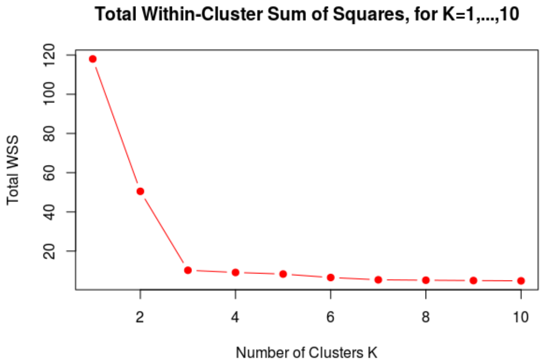
Example (cont'd). For **K=4**, we have **WSS=9.1**, and resulting plot:



Red and orange clusters are **not** well-separated, **small** drop in **WSS**.

Data-based methods, Intuition: Simulated Example.

Example. To give a better idea of WSS value progression as $K \Uparrow$, let's fit K-means for $K = 1, \ldots, 10$ and plot the WSS values:



One sees large drops in WSS for $K = 1 \Rightarrow K = 2 \Rightarrow K = 3$, followed by small drops for $K \geq 4$. Given that our simulated data has $K = 3$ natural groupings, it re-affirms the intuition from slide 7.

Data-based methods: "Elbow" method, "Gap" statistic.

So, more formally, we need to find the smallest K^* (the **"elbow"**) value such that

$$\{W_K - W_{K+1} \mid K < K^*\} >> \{W_K - W_{K+1} \mid K \geq K^*\}$$

or, in plain English,

"The drops in within-cluster variation are the highest up until K^* clusters."

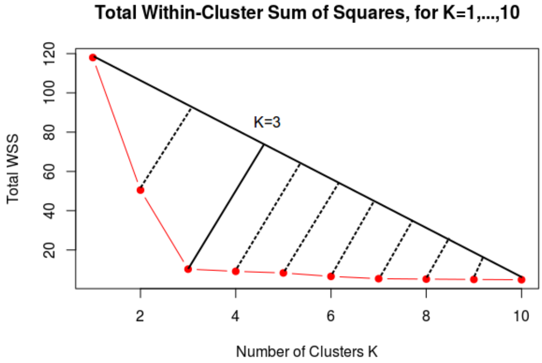
A couple of methods exists for solving this task.

- "Elbow" method.
- "Gap" statistic.

Data-based methods: "Elbow" method.

"Elbow" method. Draw a line connecting points $(1, WSS_1)$ and $(K^{max}, WSS_{K^{max}})$. The point (K^*, WSS_{K^*}) which has the largest perpendicular distance from that line is called the **"elbow"** $\Rightarrow$ K^* is the **optimal** # of clusters.

Example (cont'd). Applying this to our simulated example yields $K^* = 3$, due to point $(K = 3, WSS = 10.2)$ being furthest from the line.



Data-based methods: "Gap" statistic.

"Gap" statistic (a more formal method) compares

- the curve of $\log(WSS_K)$ for your original data $\mathbf{x}$, to
- the curve of $\log(WSS_K^{Uniform})$ obtained from data $\mathbf{x}^{Uniform}$, uniformly distributed over a rectangle containing the original data $\mathbf{x}$ (representing the situation of "no natural groupings of observations").

It estimates the optimal number K^* of clusters to be the place where the **gap** between the two curves is **largest**. Intuitively, that means we have K^* natural groupings (because we are **far** from the "no natural groupings" situation).

Data-based methods: "Gap" statistic.

Example (cont'd). For our simulated example of 60 data points, let's generate 60 data points uniformly distributed over the range of those original 60 points, meaning:

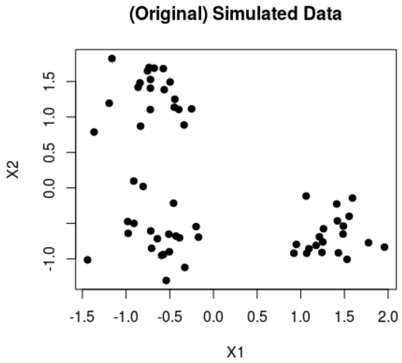
- $\mathbf{x}_1^{Uniform} \sim Uniform(min(\mathbf{x}_1), max(\mathbf{x}_1))$,
- $\mathbf{x}_2^{Uniform} \sim Uniform(min(\mathbf{x}_2), max(\mathbf{x}_2))$,

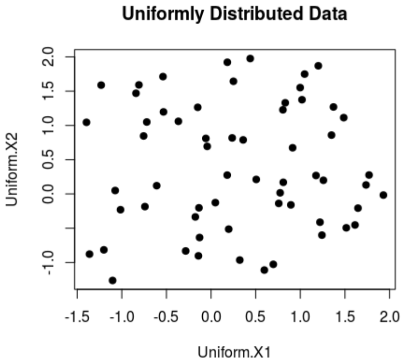
This is produced by the code below:

```
set.seed(1)
data.uniform.x1 <- runif(20*3,
                         min(x[,1]), max(x,1))
data.uniform.x2 <- runif(20*3,
                         min(x[,2]), max(x,2))
data.uniform <- cbind(data.uniform.x1,
                      data.uniform.x2)
```

Data-based methods: "Gap" statistic.

Example (cont'd). The plots of original and uniform data:

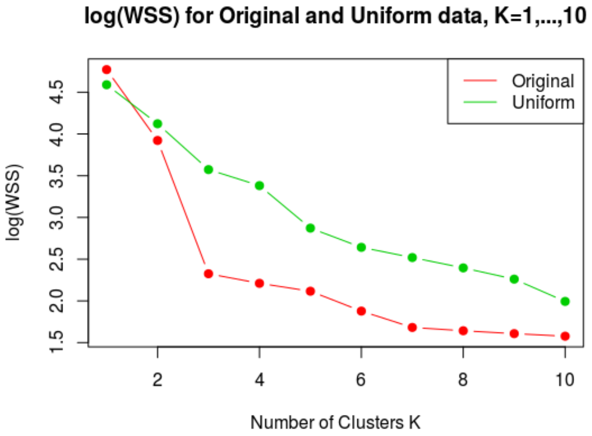




One sees how uniform data plot represents situation of **no** natural groupings - points are just uniformly scattered across the box (box is of the same size as for original data).

Data-based methods: "Gap" statistic.

Example (cont'd). Fitting K-means clustering solutions to both original and uniform data scenarios for $K = 1, \ldots, 10$, below is the plot of $\log(WSS)$ progressions:



Data-based methods: "Gap" statistic.

Intuition: Large gap between our clustering solution (curve 1) and the uniformly distributed data scenario (curve 2)

$$\Updownarrow$$

Our clustering is **far from** the "**no** natural grouping" scenario

$$\Updownarrow$$

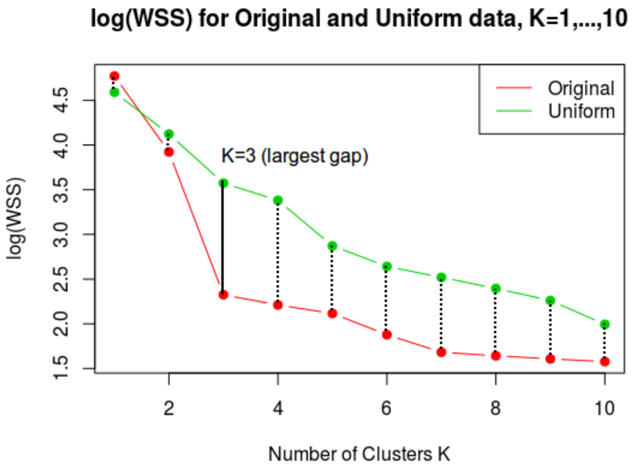
Our clustering is **close** to representing natural groupings of observations.

$$\Downarrow$$

To find optimal # of clusters K^*, we look for the K with **largest** gap bwetween two curves.

Data-based methods: "Gap" statistic.

Example (cont'd). Depicting the gaps at various values of K:



Data-based methods: "Gap" statistic.

Question: Why is it called gap "**statistic**", and not a gap "method"?

Answer: Because we don't stop at just generating **one** uniformly distributed data set and calculating gaps for it. We repeat the process of

1. Randomly generating uniform data;
2. Calculating the gaps for each value of K;

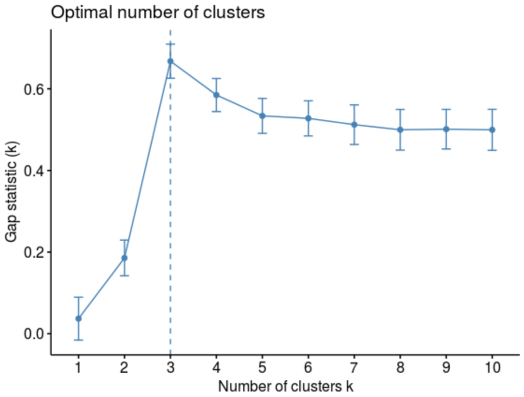
multiple B times, and then, for each K, **average** the gap values across those B replicates, eventually giving us a gap **statistic.**

Moreover, we also have access to **standard deviation** of each gap statistic, which allows to judge the **level of uncertainty** around that particular gap value.

Gap statistic in *R*: *fviz_nbclust*() function.

Function *fviz_nbclust*() of library *factoextra* calculates gap statistic, allows to specify the # of uniform data generations (via option *nboot*):

```
fviz_nbclust(data.frame(x),
                kmeans, nstart = 50,
                method = "gap_stat", nboot = 50)
```



Gap statistic in *R*: *eclust*() function.

Another way to access gap statistic calculations in *R* is by running *eclust*() function for *K*-Means **without** specifying *K*:

```
> ec.obj <- eclust(data.frame(x),
                   FUNcluster = "kmeans",
                   nstart=50,
                   nboot=50,
                   graph=0)
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 50)  [one "." per sample]:
.................................................. 50
> ec.obj$gap_stat
...
 --> Number of clusters (method 'firstSEmax', SE.factor=1): 3
          logW   E.logW        gap     SE.sim
 [1,] 3.248612 3.285550 0.03693747 0.05253545
 [2,] 2.718404 2.904052 0.18564860 0.04370008
 [3,] 2.021056 2.688716 0.66766003 0.04184520
 ...
```