Centroid-Based Models

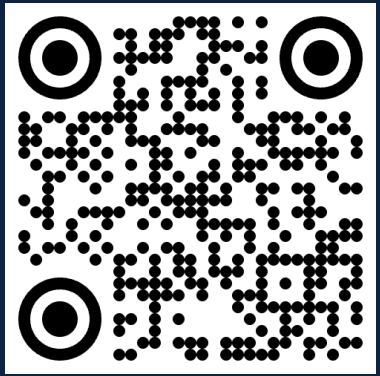
The Advanced R Series: Clustering & Classification

Slides and code created by Jesse Ghashti

January 19, 2026

Centre for Scholarly Communication
The University of British Columbia | Okanagan Campus | Syilx Okanagan Nation Territory

New Here?

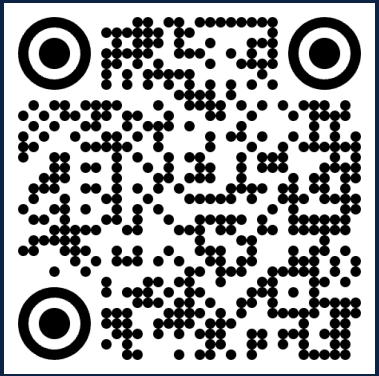


<——————-

Check out our other CSC workshops!

GitHub code and slides for today's workshop (and previous workshops)

——————>



Alternatively, code/slides available at the bottom of `https://csc-ubc-okanagan.github.io/workshops/`

Workshop Series Overview

Session	Topic	Date/Time
1	Hierarchical Models	Jan 14, 12:00 PM
2	**Centroid-Based Models**	**Jan 19, 3:30 PM**
3	Fuzzy Clustering	Jan 28, 1:00 PM
4	Distribution-Based Models	Feb 2, 3:30 PM
5	Density-Based Models	Feb 9, 3:30 PM
6	Graph-Based Models	Feb 23, 3:30 PM
7	Mixed-Type Data Quantification	Mar 4, 1:00 PM
8	Mixed-Type Data Clustering	Mar 11, 1:00 PM
9	Bias Reduction and Fairness	Mar 18, 1:00 PM
10	Dimensionality Reduction 1 of 2	Mar 23, 3:30 PM
11	Dimensionality Reduction 2 of 2	Mar 30, 3:30 PM

Last Time: Hierarchical Clustering

What did we discuss?

- Learned linkage methods: single, complete, average, Ward's, DIANA
- Used elbow method and silhouette analysis to choose k
- Applied to Iris dataset without scaling (same units)
- Evaluated clustering as classification with ARI and accuracy
- Found Ward's method performed best (89.3% accuracy, ARI=0.731)

Today: Centroid-Based Clustering

Today we will...

- Understand k-means algorithm and convergence
- Visualize k-means iterations step-by-step
- Learn classification metrics: Precision, Recall, F1, NMI
- Apply to customer segmentation dataset
- Handle data transformations for skewed distributions
- Compare k-means to k-medoids (PAM) with different distances

Today we require...

```r
library('ggplot2')     # visualization
library('cluster')     # PAM clustering
library('mclust')      # ARI metric
library('aricode')     # NMI metric
library('patchwork')   # plot arrangement
```

k-Means Algorithm

Objective Function

Minimize within-cluster sum of squares (WCSS)

$$\text{WCSS} = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$$

where μ_k is the centroid (mean) of cluster C_k

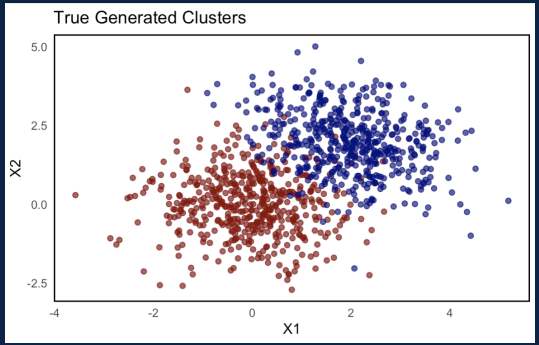
Algorithm:

1. Initialize k centroids randomly
2. Assign each point to nearest centroid
3. Update centroids as cluster means
4. Repeat until convergence

k-means assumes spherical clusters and is sensitive to initialization

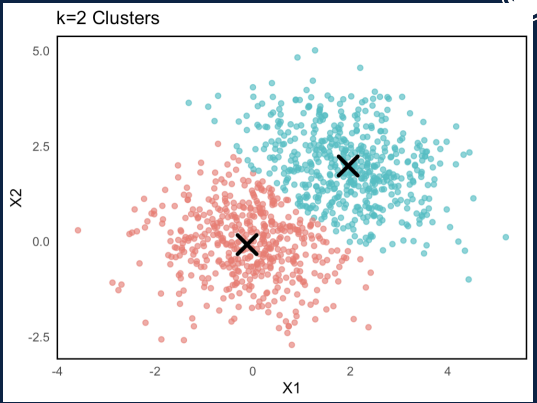
Toy Example: Two Overlapping Gaussians

```r
set.seed(42)
nPerClass <- 500
Mu1 <- c(0, 0)
Mu2 <- c(2, 2)
Sigma1 <- matrix(c(1.0, 0.0,
                   0.0, 1.0), 2, 2)
Sigma2 <- matrix(c(1.0, -0.3,
                   -0.3, 1.0), 2, 2)

X1 <- MASS::mvrnorm(nPerClass, mu = Mu1,
                    Sigma = Sigma1)
X2 <- MASS::mvrnorm(nPerClass, mu = Mu2,
                    Sigma = Sigma2)
X <- rbind(X1, X2)
TrueClass <- factor(c(rep("Class1", nPerClass),
                      rep("Class2", nPerClass)))

SimData <- data.frame(X1 = X[,1], X2 = X[,2],
                      TrueClass = TrueClass)
```

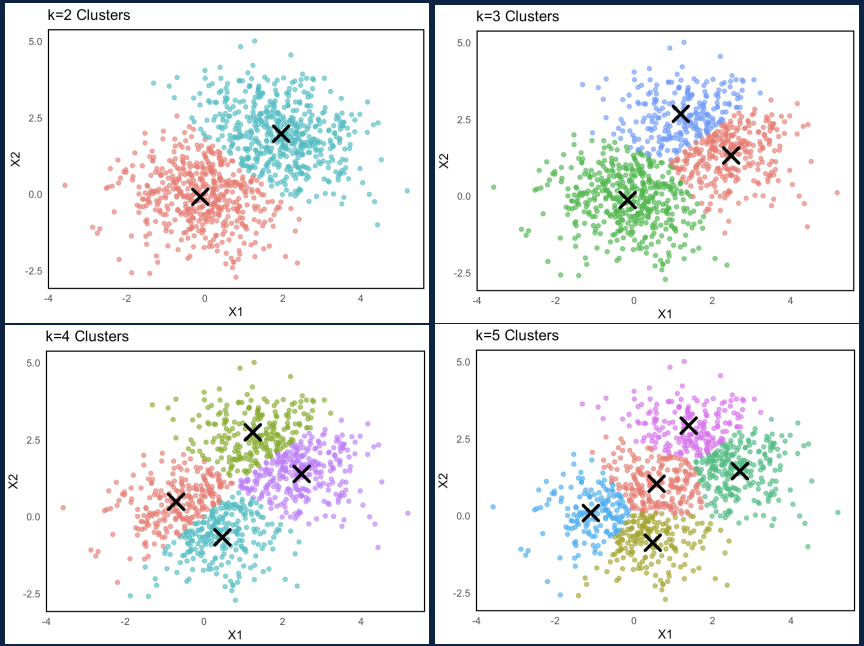


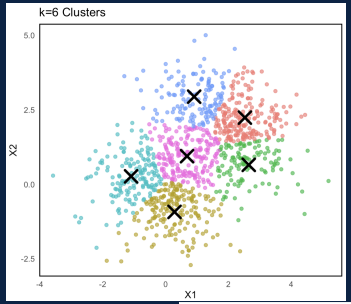
Effect of k from 2 to 10 Clusters

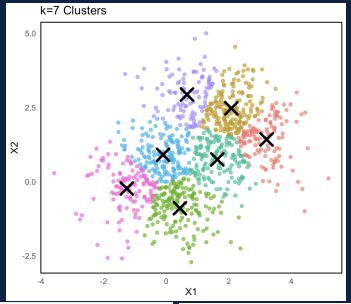
```r
Ks <- 2:10
for (k in Ks) {
  km <- kmeans(SimData[, c("X1","X2")],
               centers = k, nstart = 30,
               iter.max = 1000)
  SimData$ClusterK <- factor(km$cluster)
  Ck <- as.data.frame(km$centers)
  names(Ck) <- c("X1","X2")
  p <- ggplot(SimData, aes(X1, X2,
                           color = ClusterK)) +
    geom_point(alpha = 0.65, size = 1.5) +
    geom_point(data = Ck, aes(X1, X2),
               inherit.aes = FALSE,
               shape = 4, stroke = 2.0, size = 5) +
    labs(title = paste0("k=", k, " Clusters")) +
    theme_minimal()
  print(p)
}
```

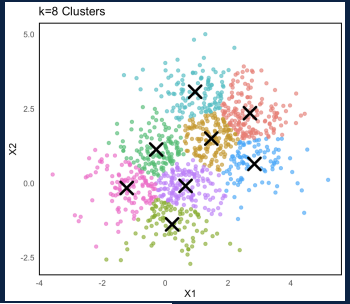


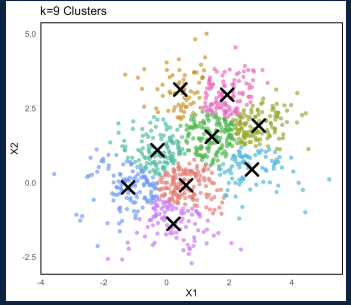
As k increases, clusters fragment the natural groups. k=2 matches the data structure best.

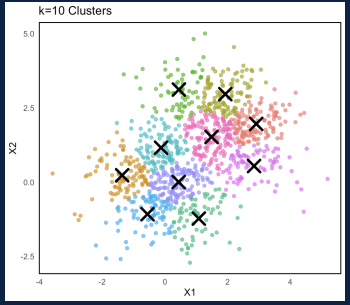

k=2 Clusters
k=3 Clusters
k=4 Clusters
k=5 Clusters
X1
X2
5.0
2.5
0.0
-2.5
-4
-2
0
2
4


k=6 Clusters
X1
X2


k=7 Clusters
X1
X2


k=8 Clusters
X1
X2


k=9 Clusters
X1
X2


k=10 Clusters
X1
X2

k-Means Algorithm Step-by-Step

Tracking the Algorithm

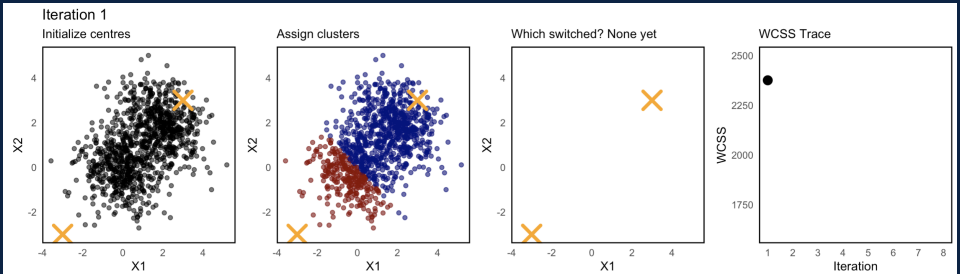
We'll visualize 8 iterations showing:

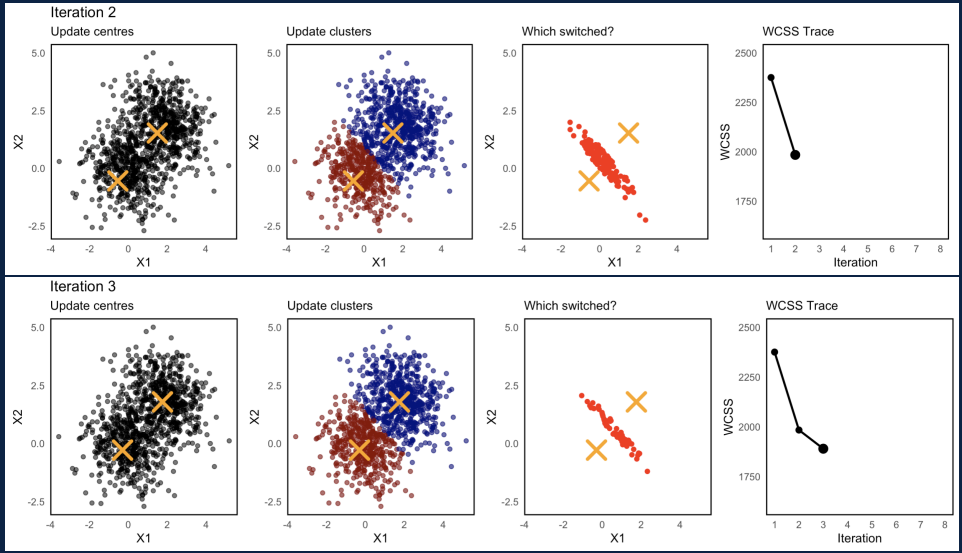
1. **Initialize/Update centers:** Position of centroids
2. **Assign clusters:** Each point to nearest centroid
3. **Which switched:** Points that changed clusters
4. **WCSS trace:** Objective function decrease

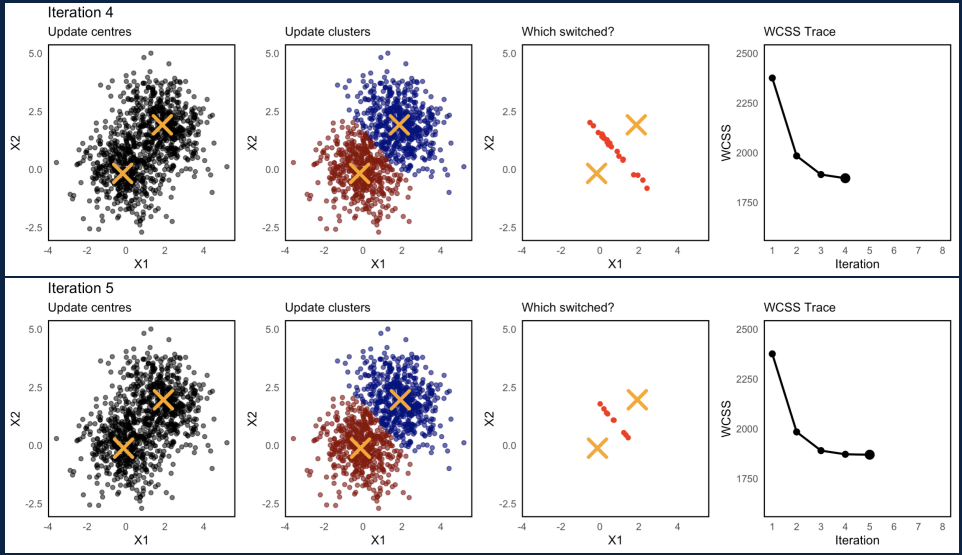
k-Means Iterations

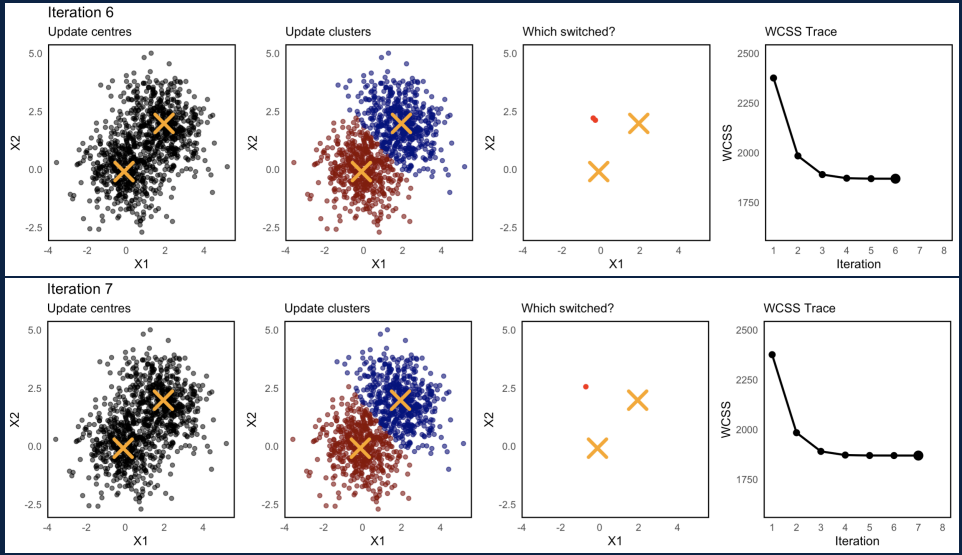
```r
wcssValue <- function(X, cluster, centers) {
  X <- as.matrix(X)
  k <- nrow(centers)
  s <- 0
  for (j in 1:k) {
    members <- which(cluster == j)
    if (length(members) > 0) {
      diffs <- X[members, , drop = FALSE] -
               matrix(centers[j, ], length(members),
                      ncol(X), byrow = TRUE)
      s <- s + sum(diffs^2)
    }}s
}
```

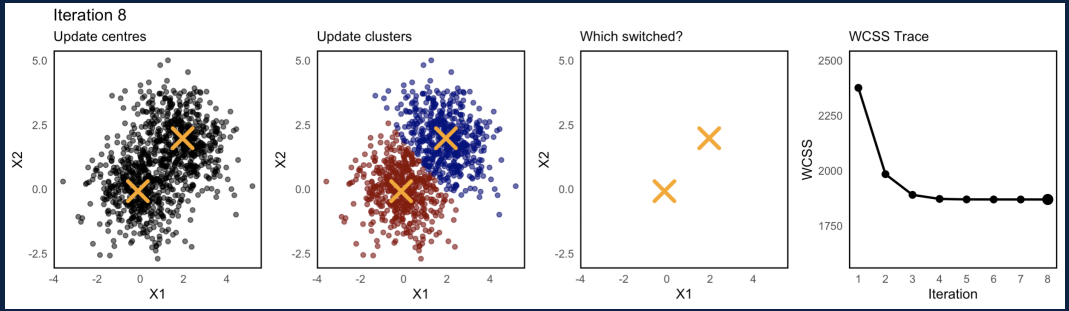
```r
kmeansSteps <- function(X, k, iters = 5) {
  X <- as.matrix(X)
  n <- nrow(X)
  centers <- matrix(c(3,3,-3,-3), nrow = 2,
                    byrow = T)
  for (it in 1:iters) {
    d2 <- sapply(1:k, function(j)
      rowSums((X - matrix(centers[j, ], n,
               ncol(X), byrow = TRUE))^2))
    cluster <- max.col(-d2)
  }
}
```



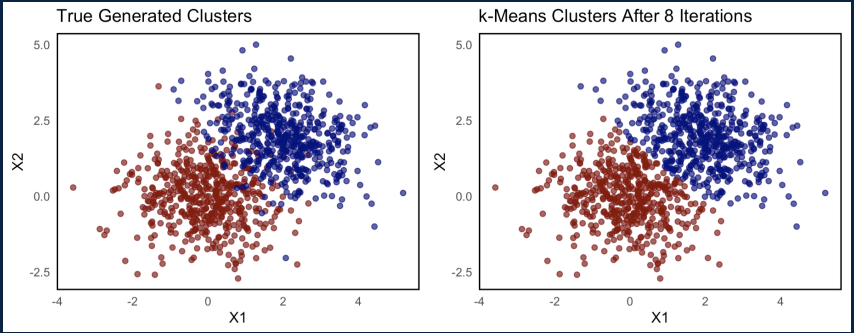

Iteration 2
Update centres
Update clusters
Which switched?
WCSS Trace
Iteration 3
Update centres
Update clusters
Which switched?
WCSS Trace
X1
X2
WCSS
Iteration


Iteration 4
Update centres
Update clusters
Which switched?
WCSS Trace
Iteration 5
Update centres
Update clusters
Which switched?
WCSS Trace
X1
X2
WCSS
Iteration


Iteration 6
Update centres
Update clusters
Which switched?
WCSS Trace
Iteration 7
Update centres
Update clusters
Which switched?
WCSS Trace
X1
X2
WCSS
Iteration


Iteration 8
Update centres
Update clusters
Which switched?
WCSS Trace
X1
X2
WCSS
Iteration

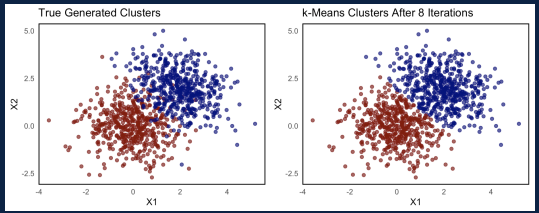
Final Clustering

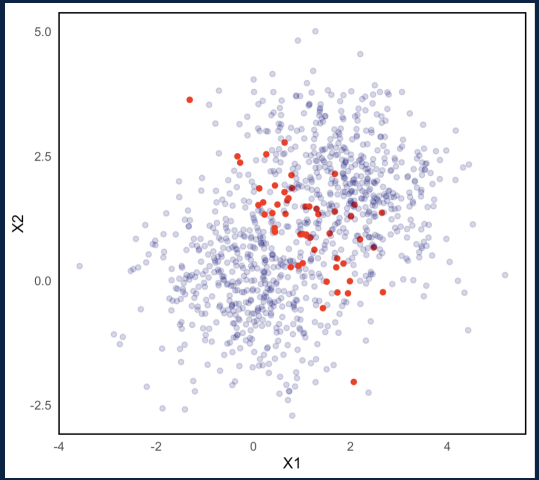


Notice the hard cut of the two clusters.

Think about if the left or right plots makes more intuitive sense...

Final Clustering II





Evaluating Clustering as Classification

Confusion Matrix

		Predicted
	Positive	**Negative**
Positive	TP	FN
Negative	FP	TN

- TP = True Positives (correct)
- TN = True Negatives (correct)
- FP = False Positives (error)
- FN = False Negatives (error)

Classification Metrics

Accuracy (overall): $\frac{TP+TN}{TP+TN+FP+FN}$

Precision (one cluster): $\frac{TP}{TP+FP}$

Recall (one cluster): $\frac{TP}{TP+FN}$

F1 Score (one cluster): $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Don't forget you need the best matching between cluster labels and true classes first.

Clustering Agreement Metrics

Label-Invariant Metrics

Adjusted Rand Index (ARI): Reviewed last time

$$ARI = \frac{RI - E[RI]}{1 - E[RI]}$$

Normalized Mutual Information (NMI):

$$NMI = \frac{2 \cdot I(X;Y)}{H(X) + H(Y)}$$

- $I(X;Y)$ = mutual information between partitions
- $H(X), H(Y)$ = entropies of each partition
- Range: 0 (independent) to 1 (perfect agreement)

Classification Metrics

```r
km2 <- kmeans(SimData[, c("X1","X2")],
              centers = 2, nstart = 50,
              iter.max = 10000)
PredCluster <- factor(km2$cluster)

Tab <- table(PredCluster, TrueClass)

accA <- (Tab[1,"Class1"] + Tab[2,"Class2"]) /
        sum(Tab)
accB <- (Tab[1,"Class2"] + Tab[2,"Class1"]) /
        sum(Tab)

if (accA >= accB) {
  Map <- c("1" = "Class1", "2" = "Class2")
} else {
  Map <- c("1" = "Class2", "2" = "Class1")
}

PredClass <- factor(Map[as.character(PredCluster)],
                    levels = levels(TrueClass))

ARI <- adjustedRandIndex(PredCluster, TrueClass)
NMI <- NMI(as.integer(PredCluster),
           as.integer(TrueClass))
```

Results for k=2:

Accuracy (94.9%): Overall classification correctness is high.

Precision (92.9%): Low false-positive rate.

Recall (97.2%): Very low false-negative rate.

F1 (95.0%): Strong balance between precision and recall.

ARI (0.806): Clustering shows strong agreement with ground truth after chance correction.

NMI (0.716): A lot of shared information between clustering and true labels.

Application: Customer Segmentation

UCI Wholesale Customers Dataset

Dataset: 440 customers from a wholesale distributor

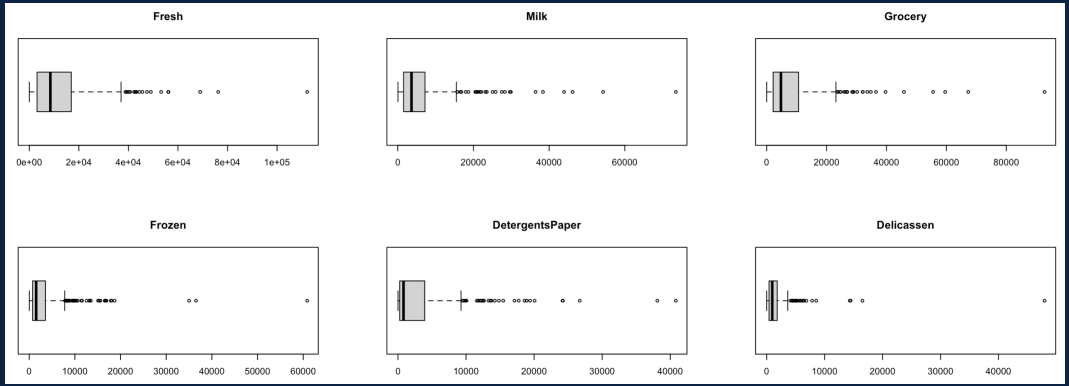
Variables: Annual spending on

- Fresh products
- Milk products
- Grocery products
- Frozen products
- Detergents and paper
- Delicatessen products

Goal: Segment customers based on purchasing behaviour

Exploration

```r
set.seed(448)
DataUrl <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20customers%20data.csv"
CustomerData <- read.csv(DataUrl)
names(CustomerData) <- c("Channel","Region", "Fresh","Milk","Grocery","Frozen", "DetergentsPaper","Delicassen")
dim(CustomerData); str(CustomerData); summary(CustomerData); table(CustomerData$Channel); table(CustomerData$Region)  # check these out
SpendVars <- c("Fresh","Milk","Grocery", "Frozen","DetergentsPaper", "Delicassen")
SpendData <- CustomerData[, SpendVars]
```



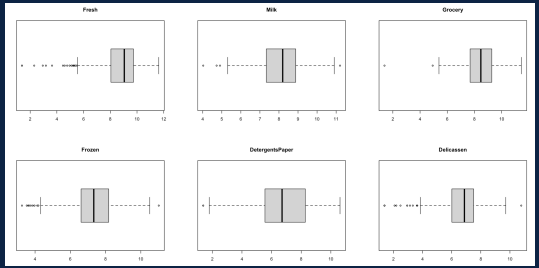
Transformation for Skew

Why log transform?

- Spending data is highly skewed
- Log reduces influence of outliers
- Makes distributions more normal

Why scale?

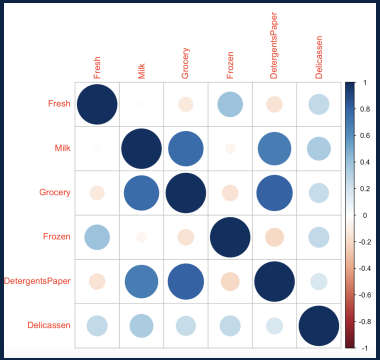
- Variables have different ranges
- Equal contribution to distance
- Standard practice for k-means

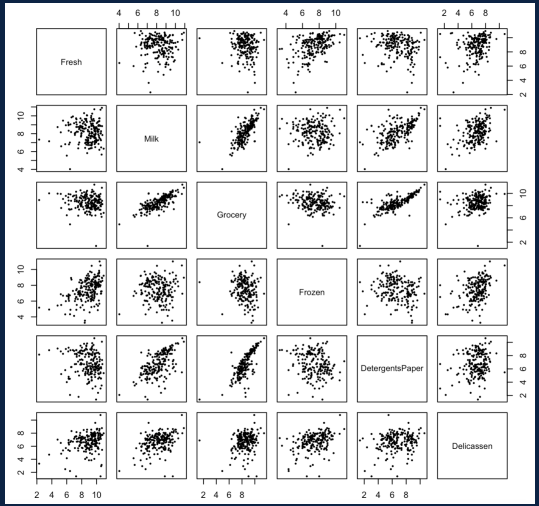



```r
par(mfrow = c(2,3))
for (v in SpendVars) hist(SpendData[[v]], main = v, xlab = v, breaks = 25)
par(mfrow = c(1,1))
SpendLog <- log1p(SpendData); X <- scale(SpendLog)
par(mfrow = c(2,3))
for (v in SpendVars) boxplot(X[[v]], main = v, horizontal = TRUE)
```

Explore More

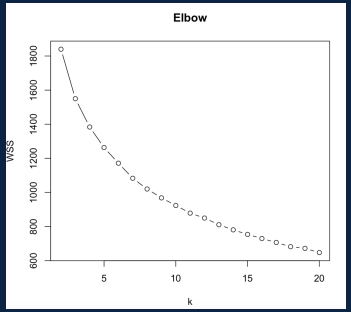
```r
corrplot::corrplot(round(cor(SpendLog),2))
# sample of the data to avoid cluttered plot
set.seed(448)
idx <- sample(seq_len(nrow(SpendLog)),
              size = min(200, nrow(SpendLog)))
pairs(SpendLog[idx, ], pch = 19, cex = 0.3)
```

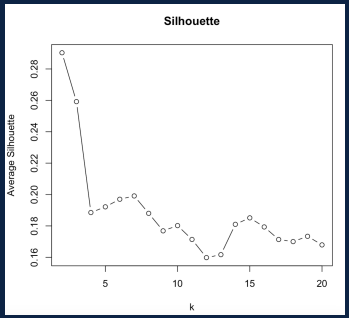




Choosing k: Elbow and Silhouette

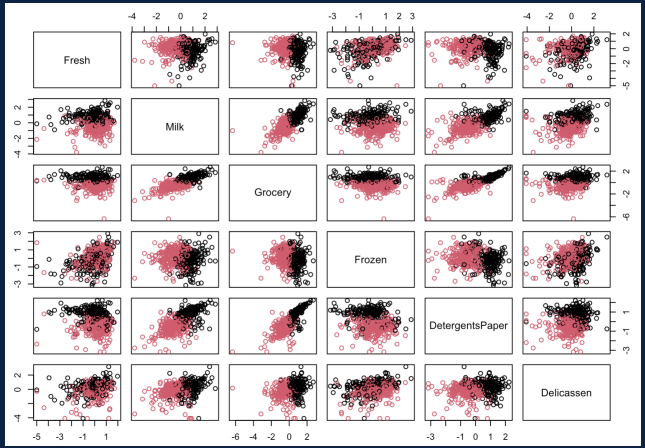
```r
avgSilhouette <- function(X, cluster) {
  sil <- silhouette(cluster, dist(X))
  mean(sil[, 3])
}

KGrid <- 2:20
ElbowWCSS <- numeric(length(KGrid))
SilScores <- numeric(length(KGrid))

for (i in seq_along(KGrid)) {
  k <- KGrid[i]
  km <- kmeans(X, centers = k, nstart = 20,
               iter.max = 1000)
  ElbowWCSS[i] <- km$tot.withinss
  SilScores[i] <- avgSilhouette(X, km$cluster)
}

plot(KGrid, ElbowWCSS, type = "b",
     xlab = "k", ylab = "WSS", main = "Elbow")
plot(KGrid, SilScores, type = "b",
     xlab = "k", ylab = "Average Silhouette",
     main = "Silhouette")

k <- 2
```





k-Means: Final Model

```
kmres <- kmeans(X, centers = k, nstart = 50,
                iter.max = 10000)
kmres$size
kmCenters <- kmres$centers

CustomerData$KMeansCluster <-
  factor(kmres$cluster)

aggregate(CustomerData[, SpendVars],
  list(Cluster = CustomerData$KMeansCluster),
  mean)

aggregate(CustomerData[, SpendVars],
  list(Cluster = CustomerData$KMeansCluster),
  median)

prop.table(table(CustomerData$KMeansCluster,
                 CustomerData$Channel), margin = 1)
prop.table(table(CustomerData$KMeansCluster,
                 CustomerData$Region), margin = 1)
```



k-Medoids (PAM): A Good Alternative

Partitioning Around Medoids

Key differences from k-means:

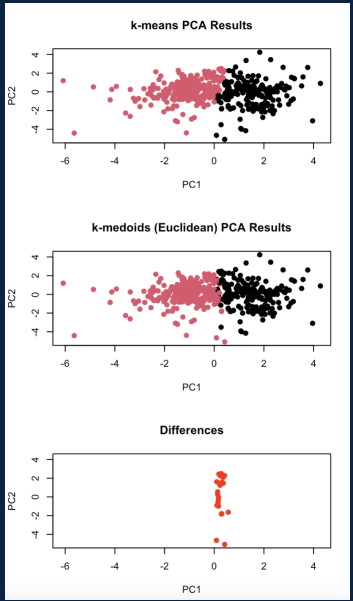
- Medoid = actual data point (not mean)
- Can use any distance metric
- More robust to outliers
- Computationally more expensive

Distance metrics:

- **Euclidean:** $d = \sqrt{\sum (x_i - y_i)^2}$ (spherical clusters)
- **Manhattan:** $d = \sum |x_i - y_i|$ (robust to outliers)

PAM Implementation

```r
pamresEuc <- pam(X, k = k,
                 metric = "euclidean")
pamresMan <- pam(X, k = k,
                 metric = "manhattan")

table(pamresEuc$clustering)
table(pamresMan$clustering)

CustomerData$PAMEucCluster <-
  factor(pamresEuc$clustering)

pca <- prcomp(X, center = FALSE,
              scale. = FALSE)
PC <- pca$x[, 1:2]

plot(PC, col = as.integer(
  CustomerData$KMeansCluster), pch = 19,
  xlab = "PC1", ylab = "PC2",
  main = "k-means clusters in PCA space")

plot(PC, col = as.integer(
  CustomerData$PAMEucCluster), pch = 19,
  xlab = "PC1", ylab = "PC2",
  main = "k-medoids clusters in PCA space")
```



Performance Comparison Code

```r
calinskiHarabasz <- function(X, cluster) {
  X <- as.matrix(X)
  n <- nrow(X)
  k <- length(unique(cluster))
  overall <- colMeans(X)

  W <- 0
  for (g in sort(unique(cluster))) {
    Xg <- X[cluster == g, , drop = FALSE]
    if (nrow(Xg) > 0) {
      cg <- colMeans(Xg)
      W <- W + sum((Xg - matrix(cg, nrow(Xg),
                    ncol(X), byrow = TRUE))^2)
    }
  }

  B <- 0
  for (g in sort(unique(cluster))) {
    Xg <- X[cluster == g, , drop = FALSE]
    if (nrow(Xg) > 0) {
      cg <- colMeans(Xg)
      ng <- nrow(Xg)
      B <- B + ng * sum((cg - overall)^2)
    }
  }

  (B / (k - 1)) / (W / (n - k))
}
```

```r
daviesBouldin <- function(X, cluster) {
  X <- as.matrix(X)
  groups <- sort(unique(cluster))
  k <- length(groups)
  C <- matrix(NA, nrow = k, ncol = ncol(X))
  rownames(C) <- groups
  for (i in seq_along(groups)) {
    g <- groups[i]
    C[i, ] <- colMeans(X[cluster == g, , drop = FALSE])
  }
  S <- rep(NA, k)
  for (i in seq_along(groups)) {
    g <- groups[i]
    Xg <- X[cluster == g, , drop = FALSE]
    diffs <- Xg - matrix(C[i, ], nrow(Xg), ncol(X), byrow = TRUE)
    S[i] <- mean(sqrt(rowSums(diffs^2)))
  }
  M <- as.matrix(dist(C))
  R <- rep(NA, k)
  for (i in seq_len(k)) {
    ratios <- rep(-Inf, k)
    for (j in seq_len(k)) {
      if (i != j) ratios[j] <- (S[i] + S[j]) / M[i, j]
    }
    R[i] <- max(ratios[is.finite(ratios)])
  }
  mean(R)
}
```

Internal Validation Metrics I

Calinski-Harabasz Index

$$CH = \frac{SS_B/(k-1)}{SS_W/(n-k)}$$

- SS_B = between-cluster variance
- SS_W = within-cluster variance
- **Higher is better**
- Ratio of separation to compactness

Davies-Bouldin Index

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i} \left(\frac{s_i + s_j}{d_{ij}} \right)$$

- s_i = avg distance to centroid in cluster i
- d_{ij} = distance between centroids
- **Lower is better**
- Measures worst-case cluster overlap

Internal Validation Metrics II

Dunn Index

$$D = \frac{\min_{i \neq j} d(C_i, C_j)}{\max_k \text{diam}(C_k)}$$

- Min inter-cluster distance
- Max intra-cluster diameter
- **Higher is better**

Silhouette (Review)

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- $a(i)$ = avg dist to own cluster
- $b(i)$ = avg dist to nearest cluster
- **Higher is better** (range: -1 to 1)

Comparing Results

Method	Silhouette	CH	DB
k-Means	0.290	189.050	1.352
PAM (Euclidean)	0.282	184.168	1.368
PAM (Manhattan)	0.318	181.860	1.384

PAM (Euclidean) vs PAM (Manhattan)

	1	2
1	188	5
2	13	234

PAM (Euclidean) vs K-means

	1	2
1	179	14
2	9	238

PAM (Manhattan) vs K-means

	1	2
1	182	19
2	6	233

Interpretation I

Cluster	HoReCa	Retail
1	244	8
2	54	134

- Cluster 1 is overwhelmingly HoReCa ($244/252 \approx 96.8\%$).
- Cluster 2 is mostly Retail ($134/188 \approx 71.3\%$).
- Even though k-means used only spending variables, it nearly recovers Channel, suggesting the segments are meaningful.

Interpretation II

Cluster	Fresh	Milk	Grocery	Frozen	Det. Paper	Delic.
1	13973.12	2401.75	2918.70	3705.67	491.94	1038.04
2	9355.86	10346.36	14697.06	2222.45	6084.50	2177.42

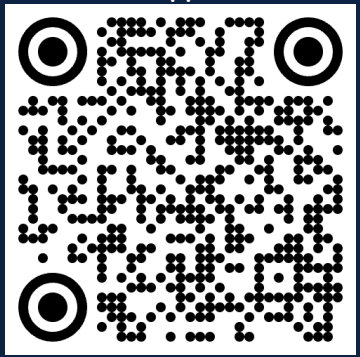
- Cluster 1 (HoReCa-heavy): higher **Fresh** and lower **Milk/Grocery/Detergents+Paper** $\Rightarrow$ fresh-food oriented purchasing.
- Cluster 2 (Retail-heavy): much higher **Milk/Grocery/Detergents_Paper** $\Rightarrow$ shelf-stable + household-goods basket typical of retail.
- These centroids might explain why Channel separates so well.

Key Takeaways

- **k-means** minimizes within-cluster sum of squares iteratively
- **Algorithm converges** but can find local minima (use nstart > 1)
- **Data transformation** crucial for skewed data (log transform)
- **Scaling** ensures equal variable contribution to distance
- **Elbow method** shows diminishing returns in WCSS
- **Silhouette** measures cluster quality (cohesion vs separation)
- **k-medoids (PAM)** more robust to outliers, uses actual data points
- **Manhattan distance** alternative for robustness
- **Multiple metrics** needed: Silhouette, Calinski-Harabasz, Davies-Bouldin
- **Stability analysis** reveals sensitivity to initialization

What's Next?

Additional Questions?
Book an Appointment!



Next Workshop: January 28, 1:00 pm

Fuzzy Clustering
– motivated by the red points in slide 17.

Thank You!

Questions?

Workshop Materials:
`https://csc-ubc-okanagan.github.io/workshops/`

Contact:
jesse.ghashti@ubc.ca