Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

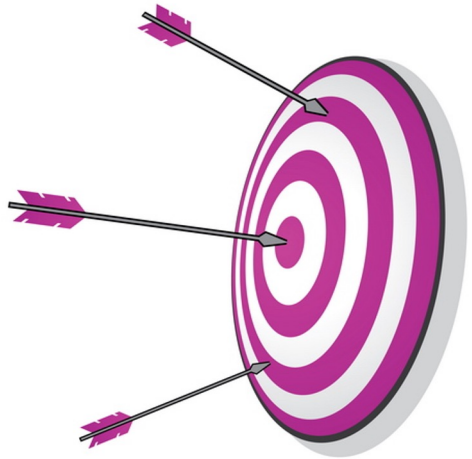
Business Analytics using Data Mining

BU7143

Dr. Nicholas P. Danks
Business Analytics
nicholas.danks@tcd.ie

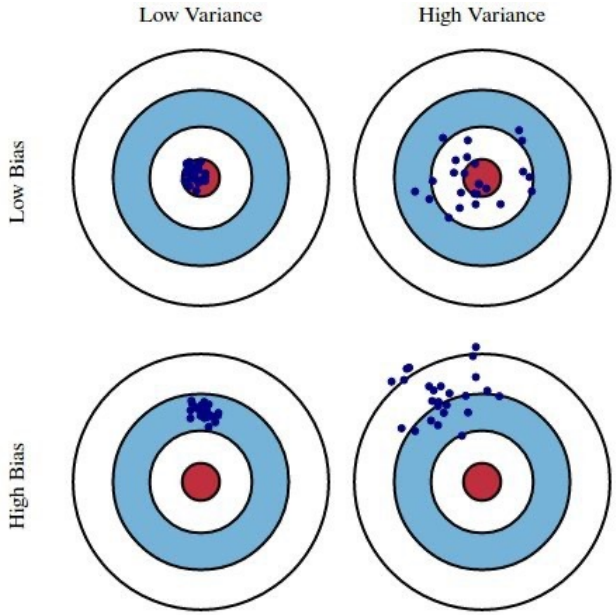
Part II
Performance Evaluation

Why Evaluate?



- Multiple methods are available to classify or predict
- For each method, multiple choices are available for settings
- To choose best model, need to assess each model's performance

Measuring Predictive error



- We want to know how well the model predicts **new data**, not how well it fits the data it was trained with
- Key component of most measures is difference between actual *y* and predicted *y* ("error")

$$e = y - \hat{y}$$

- **MAE or MAD**: Mean absolute error (deviation) Gives an idea of the magnitude of errors

$$\text{MAE} = \frac{\sum |e|}{n}$$

- **Average error** Gives an idea of systematic over- or under-prediction

$$\text{ME} = \frac{\sum e}{n}$$

- **MAPE**: Mean absolute percentage error

$$\text{MAPE} = \frac{100}{n} \sum \left| \frac{e}{y} \right|$$

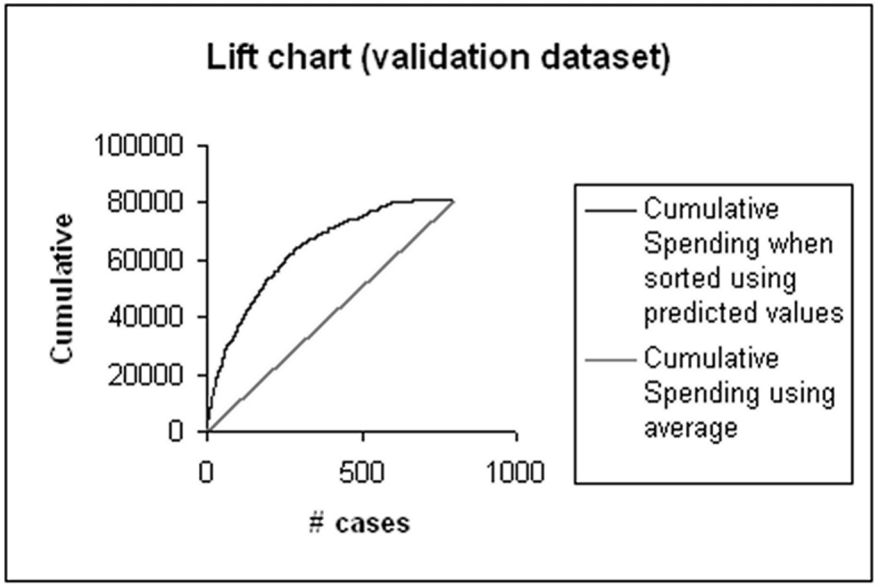
- **RMSE** (root-mean-squared-error): Square the errors, find their average, take the square root

$$\text{RMSE} = \sqrt{\frac{\sum e^2}{n}}$$

Lift Chart for Predictive Error

- Y axis is cumulative value of numeric target variable (e.g., revenue), instead of cumulative count of "responses"
- X axis is cumulative number of cases, sorted left to right in order of predicted value
- Benchmark is average numeric value per record, i.e. not using model

Lift chart example – spending



Misclassification error

- Error = classifying a record as belonging to one class when it belongs to another class.
- Error rate = percent of misclassified records out of the total records in the validation data

Predicted ($\hat{Y}$) \ Actual (Y)	0	1
0	2689	85
1	25	201

$$e = \frac{25+85}{3000} = 3.6\%$$

Naïve Rule

Naïve rule: classify all records as belonging to the most prevalent class

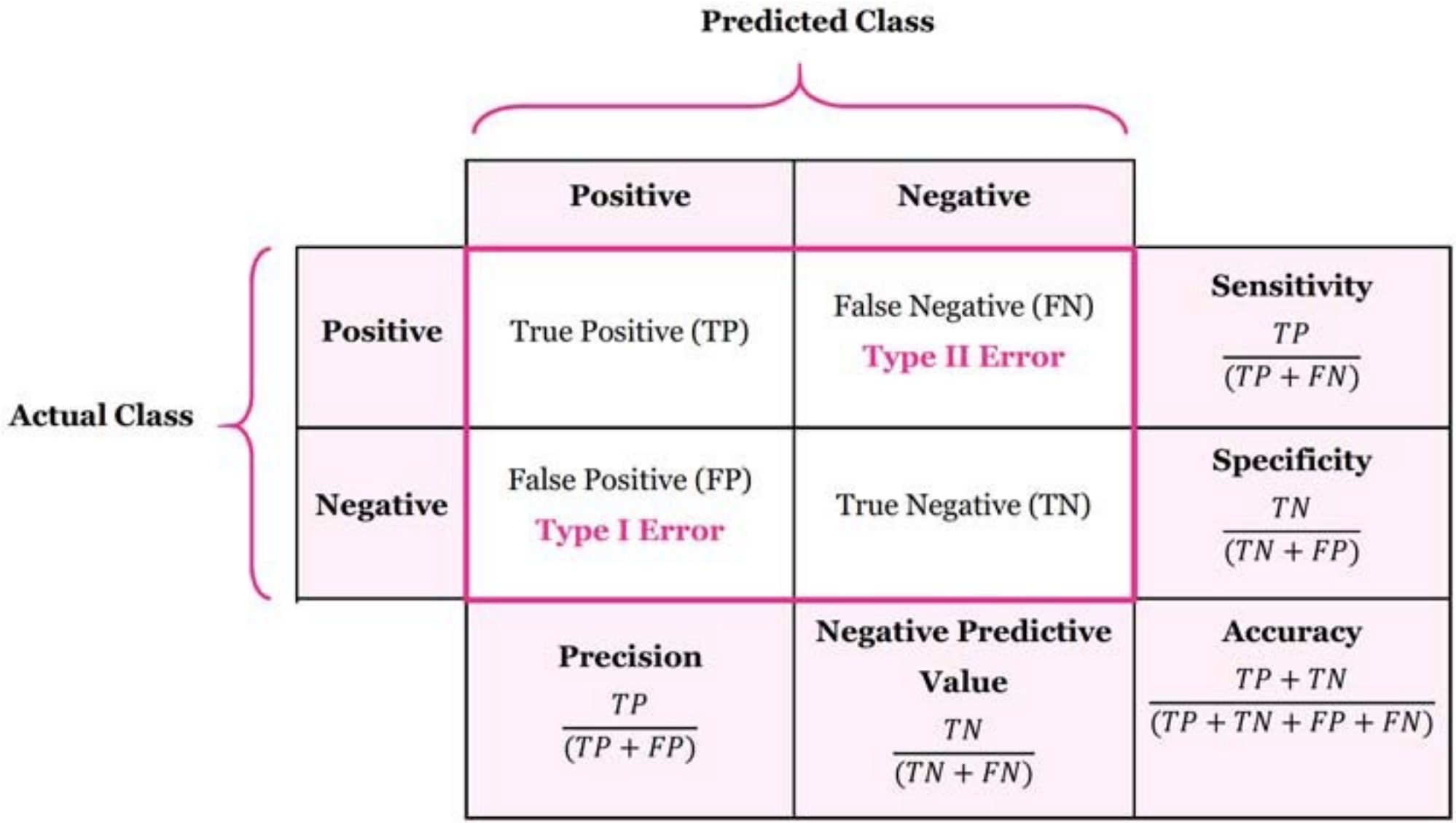
- Often used as benchmark: we hope to do better than that
- Exception: when goal is to identify high-value but rare outcomes, we may do well by doing worse than the naïve rule (see “lift” – later)

Cutoff for classification

- Most DM algorithms classify via a 2-step process:
- For each record,
  - Compute **probability of belonging to class "1"**
  - Compare to cutoff value, and classify accordingly
- Default cutoff value is 0.50
  - If >= 0.50, classify as "1"
  - If < 0.50, classify as "0"
- Can use different cutoff values
- Typically, error rate is lowest for cutoff = 0.50

Cutoff Table

Actual Class	Prob. of "1"	Actual Class	Prob. of "1"
1	0.996	1	0.506
1	0.988	0	0.471
1	0.984	0	0.337
1	0.980	1	0.218
1	0.948	0	0.199
1	0.889	0	0.149
1	0.848	0	0.048
0	0.762	0	0.038
1	0.707	0	0.025
1	0.681	0	0.022
1	0.656	0	0.016
0	0.622	0	0.004


Predicted Class
Positive
Negative
Actual Class
Positive
True Positive (TP)
False Negative (FN)
Type II Error
Sensitivity
$\frac{TP}{(TP+FN)}$
Negative
False Positive (FP)
Type I Error
True Negative (TN)
Specificity
$\frac{TN}{(TN+FP)}$
Precision
$\frac{TP}{(TP+FP)}$
Negative Predictive Value
$\frac{TN}{(TN+FN)}$
Accuracy
$\frac{TP+TN}{(TP+TN+FP+FN)}$

Confusion Matrix for Different Cutoffs

Function `confusionMatrix` requires library `caret`

```
## cutoff = 0.5
> confusionMatrix(ifelse(owner.df$Probability>0.5,
# note: "reference" = "actual"
Confusion Matrix and Statistics

          Reference
Prediction nonowner owner
  nonowner       10     1
  owner           2    11

               Accuracy : 0.875
```

```
## cutoff = 0.25
> confusionMatrix(ifelse(owner.df$Probability>0.25,
Confusion Matrix and Statistics

          Reference
Prediction nonowner owner
  nonowner        8     1
  owner           4    11

               Accuracy : 0.7916667
```

```
## cutoff = 0.75
> confusionMatrix(ifelse(owner.df$Probability>0.75,
Confusion Matrix and Statistics

          Reference
Prediction nonowner owner
  nonowner       11     5
  owner           1     7

               Accuracy : 0.75
```

When One Class is More Important

In many cases it is more important to identify members of one class

- Tax fraud
- Credit default
- Response to promotional offer
- Detecting electronic network intrusion
- Predicting delayed flights

In such cases, we are willing to tolerate greater overall error, in return for better identifying the important class for further attention

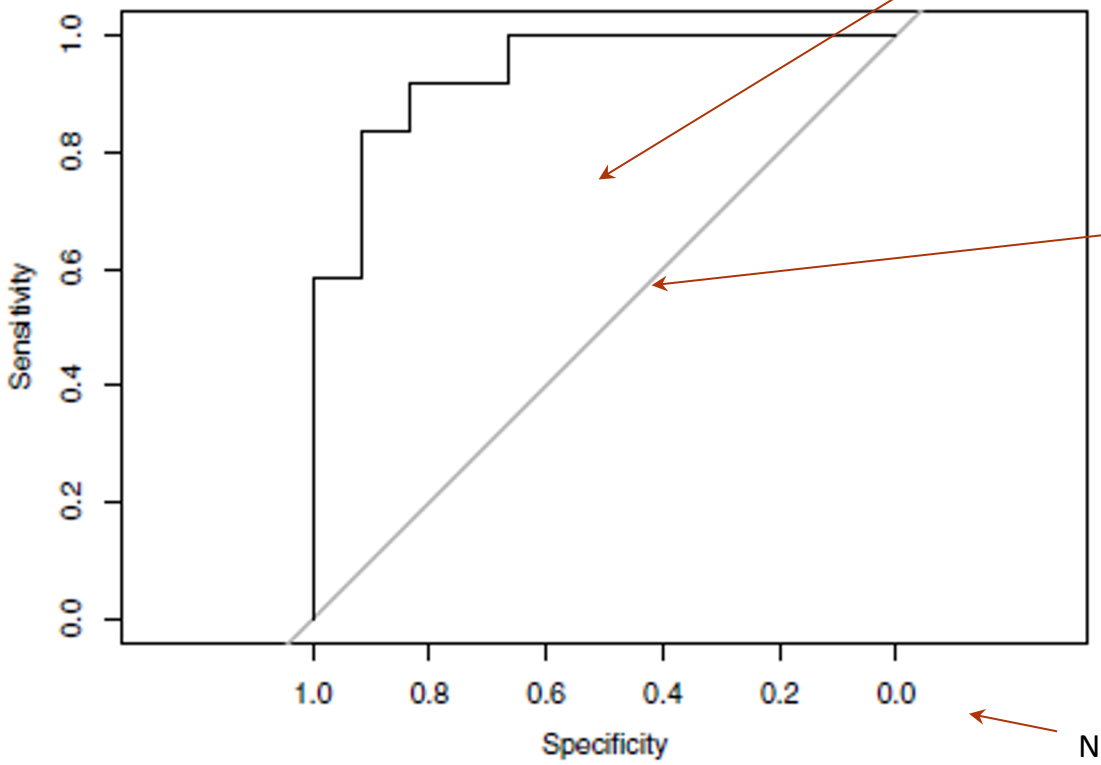
If "C_1" is the important class,

Sensitivity (also called "recall) = % of "C_1" class correctly classified

Specificity = % of "C_0" class correctly classified

Precision= % of predicted "C_1's" that are actually"C_1's"

ROC Curve (library pROC)



Area under the ROC curve ("AUC") is a useful metric

Diagonal reflects random classification on a sliding probability, from 0 to 1, of labeling a record as a "1"

Note that x-axis is in reverse order; alternative is to plot "1-specificity" in ascending order

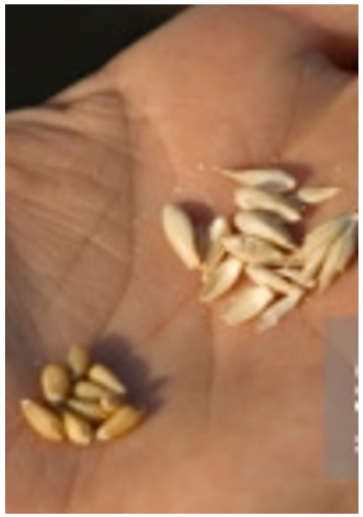
If "C_1" is the important class,

Sensitivity (also called "recall) = % of "C_1" class correctly classified

Specificity = % of "C_0" class correctly classified

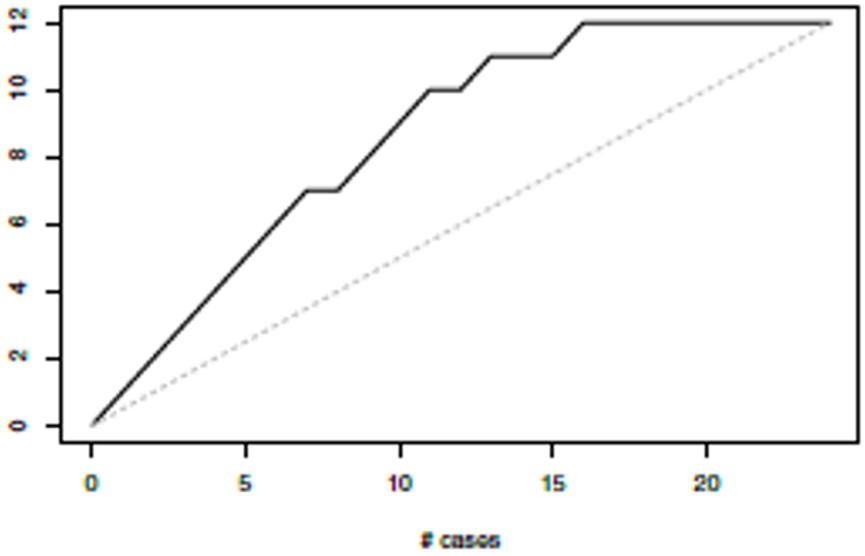
Lift (gains)

(separating the “wheat from the chaff”)



Lift ("gains"): Goal

- Evaluates how well a model identifies the most important class
- Helps evaluate, e.g.,
  - How many tax records to examine
  - How many loans to grant
  - How many customers to mail offer to
- Compare performance of DM model to "no model, pick randomly"
- Measures ability of DM model to identify the important class, relative to the average prevalence of the class
- Charts give explicit assessment of results over a large number of cutoffs



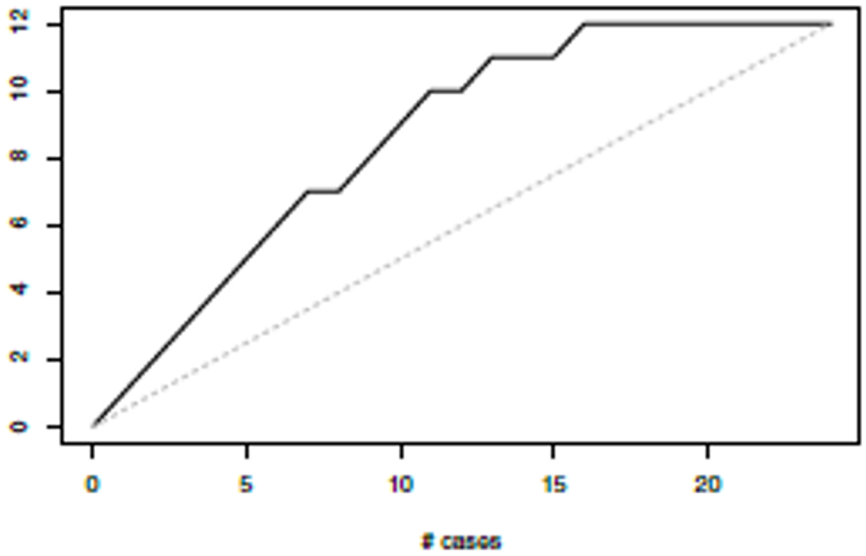
Lift and Decile Charts: How to Use

Sort records by predicted probability of belonging to the important class ("1's")

Move down the list, noting actual class

As you go, compare the number of actual 1's to the number of 1's you would expect with no model

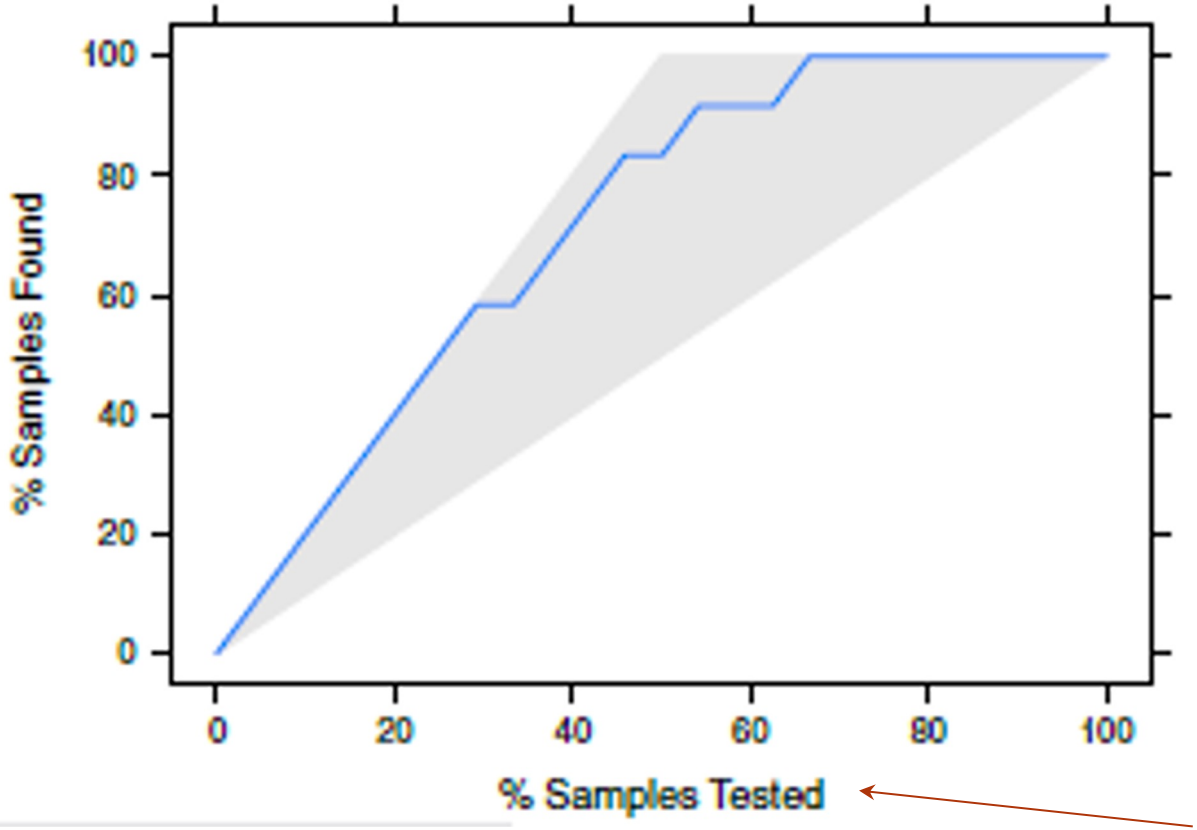
- In lift chart: compare step function to straight line
- In decile chart compare to ratio of 1



After examining (e.g.,) 10 cases (x-axis), 9 owners (y-axis) have been correctly identified

Lift Chart

using %



"Sample" = case

After examining (e.g.,) 40% = 10 of the cases (x-axis), 75% of the owners (y-axis) have been correctly identified

Decile Chart

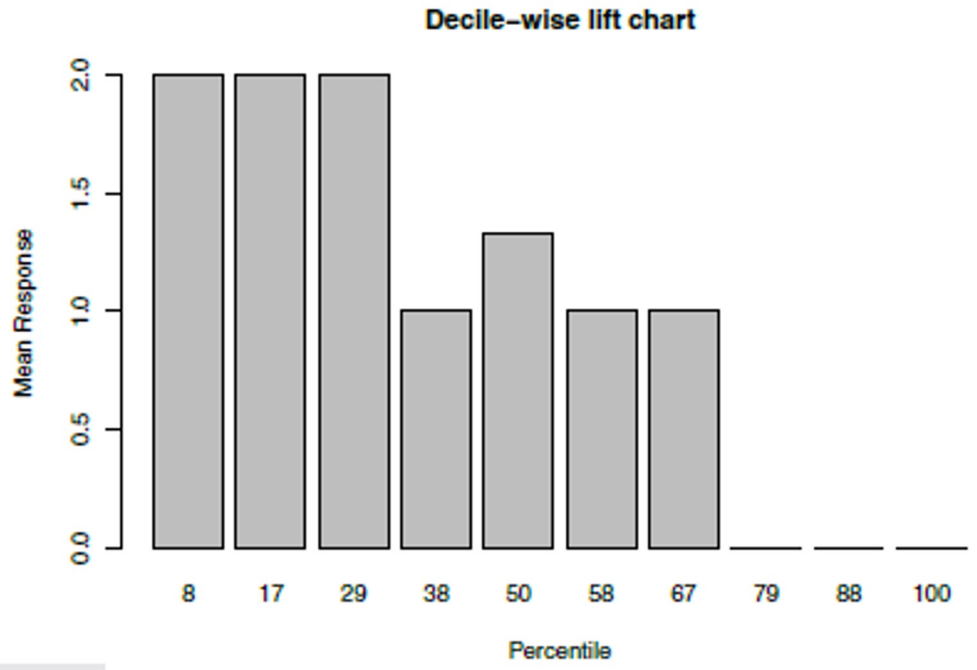


FIGURE 5.7 DECILE LIFT CHART

In “most probable” (top) decile, model is twice as likely to identify the important class compared to avg. prevalence. Percentiles do not match deciles exactly due to small sample of discrete data, with multiple records sharing same decile boundary.

Asymmetric Costs

Misclassification Costs May Differ

Example – Response to Promotional Offer

The cost (benefit) of making a misclassification error may be higher for one class than the other(s)

Suppose we send an offer to 1000 people, with 1% average response rate ("1" = response, "0" = nonresponse)

1. "Naïve rule" (classify everyone as "0")
- error rate of 1% (seems good)

2. Using DM we can correctly classify eight 1's as 1's
- It comes at the cost of misclassifying twenty 0's as 1's and two 1's as 0's.
- Error rate 2.2%

	Actual 0	Actual 1
Predicted 0	970	2
Predicted 1	20	8

Introducing Costs & Benefits

Suppose:

Profit from a "1" is $10

Cost of sending offer is $1

Then:

Under naïve rule, all are classified as "0", so no offers are sent: no cost, no profit

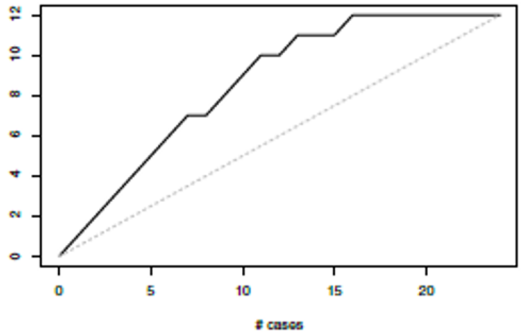
Under DM predictions, 28 offers are sent.

- 8 respond with profit of $10 each
- 20 fail to respond, cost $1 each
- 972 receive nothing (0 cost, 0 profit)

Net profit = $60

Profit Matrix

	Actual 0	Actual 1
Predicted 0	$0	$0
Predicted 1	($20)	$80



Adding costs to the mix, as above, does not change the actual classifications

Better: Use the lift curve and change the cutoff value for "1" to maximize profit

Rare Cases

Asymmetric costs/benefits typically go hand in hand with presence of rare but important class

- Responder to mailing
- Someone who commits fraud
- Debt defaulter

- Often we oversample rare cases to give model more information to work with

- Typically use 50% “1” and 50% “0” for training

Clustering: The Main Idea

Goal: Form groups (clusters) of similar records

Used for **segmenting markets** into groups of similar customers

Example: Claritas segmented US neighborhoods based on demographics & income: “Furs & station wagons,” “Money & Brains”, …

Other Applications

- Periodic table of the elements
- Classification of species
- Grouping securities in portfolios
- Grouping firms for structural analysis of economy
- Army uniform sizes

Example: Public Utilities

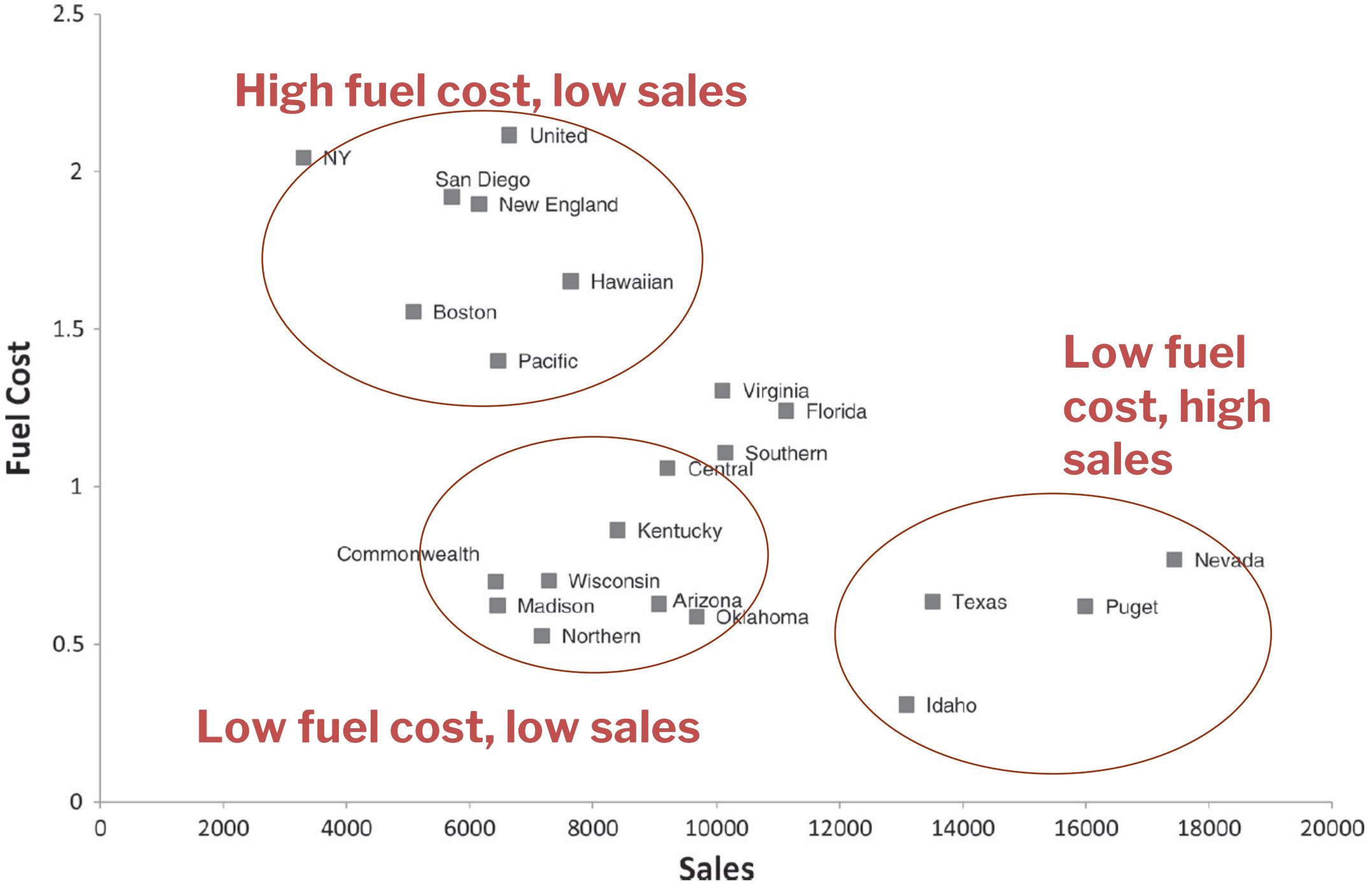
Goal: find clusters of similar utilities

Data: 22 firms, 8 variables

- Fixed-charge covering ratio
- Rate of return on capital
- Cost per kilowatt capacity
- Annual load factor
- Growth in peak demand
- Sales
- % nuclear
- Fuel costs per kwh

Company	Fixed_charge	RoR	Cost	Load	Δ Demand	Sales	Nuclear	Fuel_Cost
Arizona	1.06	9.2	151	54.4	1.6	9077	0	0.628
Boston	0.89	10.3	202	57.9	2.2	5088	25.3	1.555
Central	1.43	15.4	113	53	3.4	9212	0	1.058
Commonwealth	1.02	11.2	168	56	0.3	6423	34.3	0.7
Con Ed NY	1.49	8.8	192	51.2	1	3300	15.6	2.044
Florida	1.32	13.5	111	60	-2.2	11127	22.5	1.241
Hawaiian	1.22	12.2	175	67.6	2.2	7642	0	1.652
Idaho	1.1	9.2	245	57	3.3	13082	0	0.309
Kentucky	1.34	13	168	60.4	7.2	8406	0	0.862
Madison	1.12	12.4	197	53	2.7	6455	39.2	0.623
Nevada	0.75	7.5	173	51.5	6.5	17441	0	0.768
New England	1.13	10.9	178	62	3.7	6154	0	1.897
Northern	1.15	12.7	199	53.7	6.4	7179	50.2	0.527
Oklahoma	1.09	12	96	49.8	1.4	9673	0	0.588
Pacific	0.96	7.6	164	62.2	-0.1	6468	0.9	1.4
Puget	1.16	9.9	252	56	9.2	15991	0	0.62
San Diego	0.76	6.4	136	61.9	9	5714	8.3	1.92
Southern	1.05	12.6	150	56.7	2.7	10140	0	1.108
Texas	1.16	11.7	104	54	-2.1	13507	0	0.636
Wisconsin	1.2	11.8	148	59.9	3.5	7287	41.1	0.702
United	1.04	8.6	204	61	3.5	6650	0	2.116
Virginia	1.07	9.3	174	54.3	5.9	10093	26.6	1.306

Sales & Fuel Cost: 3 rough clusters can be seen



Extension to More Than 2 Dimensions

In prior example, clustering was done by eye

Multiple dimensions require formal algorithm with

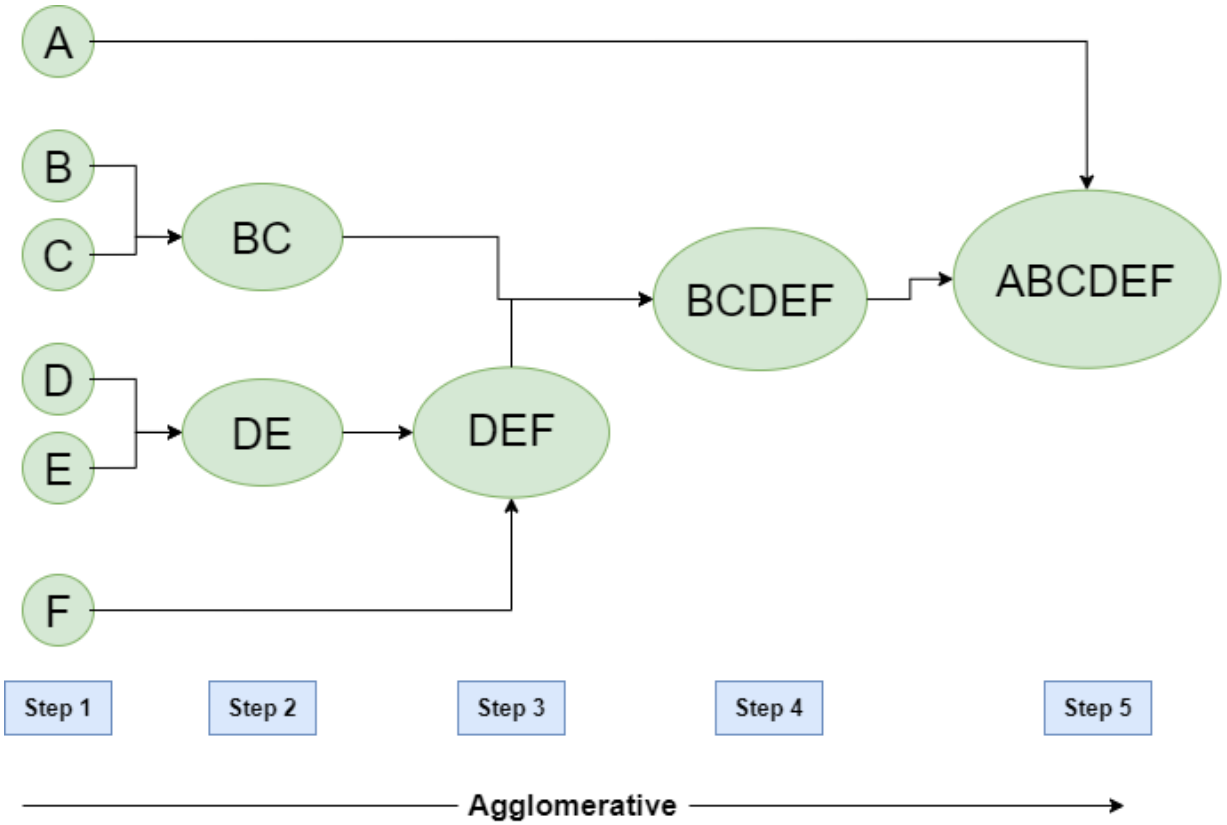
- A **distance measure**
- A way to use the distance measure in forming clusters

We will consider two algorithms: **hierarchical** and **non-hierarchical**

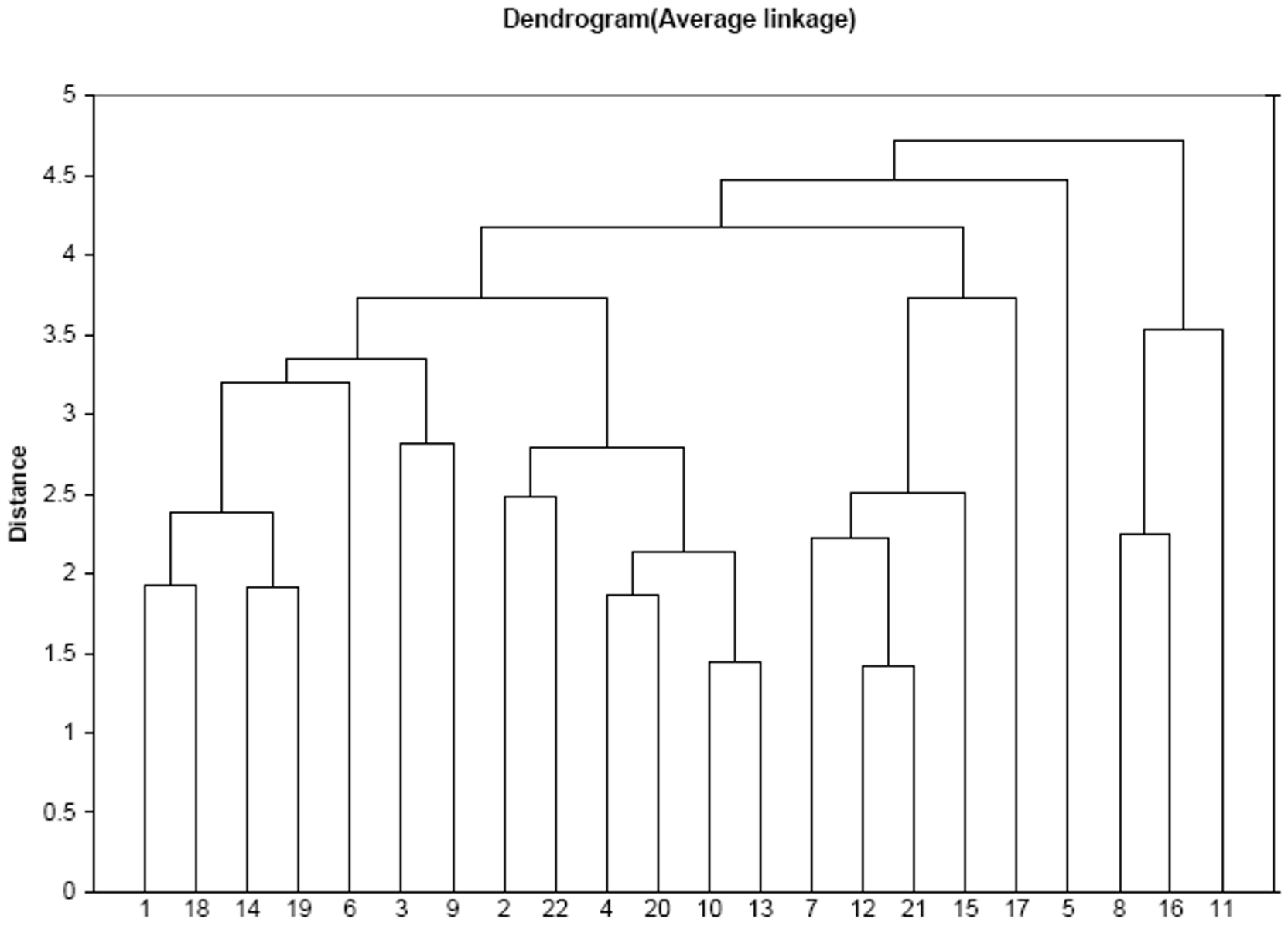
Hierarchical Clustering

Hierarchical Methods

- Begin with n-clusters (each record its own cluster)
- Keep joining records into clusters until one cluster is left (the entire data set)
- Most popular



A Dendrogram shows the cluster hierarchy



Measuring Distance Between Records

Euclidean Distance is most popular:

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

Normalizing can be important

Problem: Raw distance measures are highly influenced by scale of measurements

Solution: normalize (standardize) the data first

Subtract mean, divide by std. deviation

Also called **z-scores**

For 22 utilities:

Avg. sales = 8,914

Std. dev. = 3,550

Normalized score for Arizona sales: (9,077-8,914)/3,550 = 0.046

Compute Distance Matrix for Utility Pairs

```
utilities.df <- read.csv("Utilities.csv")
# set row names to the utilities column
row.names(utilities.df) <- utilities.df[,1]
# remove the utility column
utilities.df <- utilities.df[,-1]
# compute Euclidean distance
d <- dist(utilities.df, method = "euclidean")
```

Could use a variety of metrics here,
see R documentation for `dist`

Output of Distance Matrix (showing first 5 pairs)

	Arizona	Boston	Central	Commonwealth	NY
Boston	3989.40808				
Central	140.40286	4125.04413			
Commonwealth	2654.27763	1335.46650	2789.75967		
NY	5777.16767	1788.06803	5912.55291	3123.15322	
Florida	2050.52944	6039.68908	1915.15515	4704.36310	7827.42921

Code for Normalizing then Computing Distance

`scale`, without further arguments, normalizes each column

```
# normalize input variables
utilities.df.norm <- sapply(utilities.df, scale)
# add row names: utilities
row.names(utilities.df.norm) <- row.names(utilities.df)
# compute normalized distance based on Sales (column
     6) and Fuel Cost (column 8)
d.norm <- dist(utilities.df.norm[,c(6,8)], method =
     "euclidean")
```

For Categorical Data: **Similarity**

To measure the distance between records in terms of two 0/1 variables, create table with counts:

	0	1
0	a	b
1	c	d

Similarity metrics based on this table:

- Matching coef. = (a+d)/p
- Jaquard's coef. = d/(b+c+d)
  - Use in cases where a matching "1" is much greater evidence of similarity than matching "0" (i.e. important attribute)

Other Distance Measures

- Correlation-based similarity
- Statistical distance (Mahalanobis)
- Manhattan distance (absolute differences)
- Maximum coordinate distance
- Gower's similarity (for mixed variable types: continuous & categorical)

Measuring Distance Between Clusters

Minimum Distance (single linkage)

- Distance between two clusters is the distance between the pair of records A_i and B_j that are closest

Maximum Distance (complete linkage)

- Distance between two clusters is the distance between the pair of records A_i and B_j that are farthest from each other

Average Distance (average linkage)

- Distance between two clusters is the average of all possible pair-wise distances

Centroid Distance

- Distance between two clusters is the distance between the two cluster centroids
- Centroid is the vector of variable averages for all records in a cluster

The Hierarchical Clustering Steps (Using Agglomerative Method)

1. Start with *n* clusters (each record is its own cluster)
2. Merge two closest records into one cluster
3. At each successive step, the two clusters closest to each other are merged

Dendrogram, from bottom up, illustrates the process

Code for Hierarchical Clustering and Plotting the Dendrogram

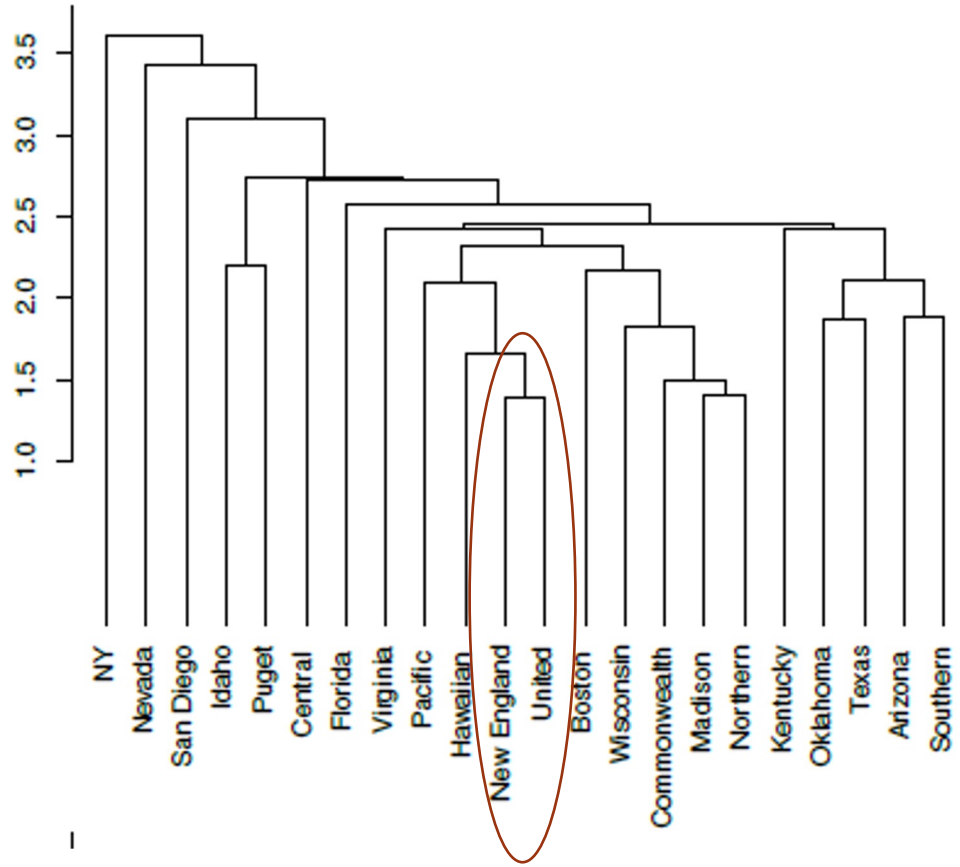
```
# in hclust() set argument method =
# to "ward.D", "single", "complete", "average",
     "median", or "centroid"

hc1 <- hclust(d.norm, method = "single")
plot(hc1, hang = -1, ann = FALSE)
```

causes x-axis labels to hang below the 0 line

turns off annotation (plot and axis titles)

Dendrogram for Single Linkage



two closest records form first cluster

Reading the Dendrogram

See process of clustering:

Lines connected lower down are merged earlier

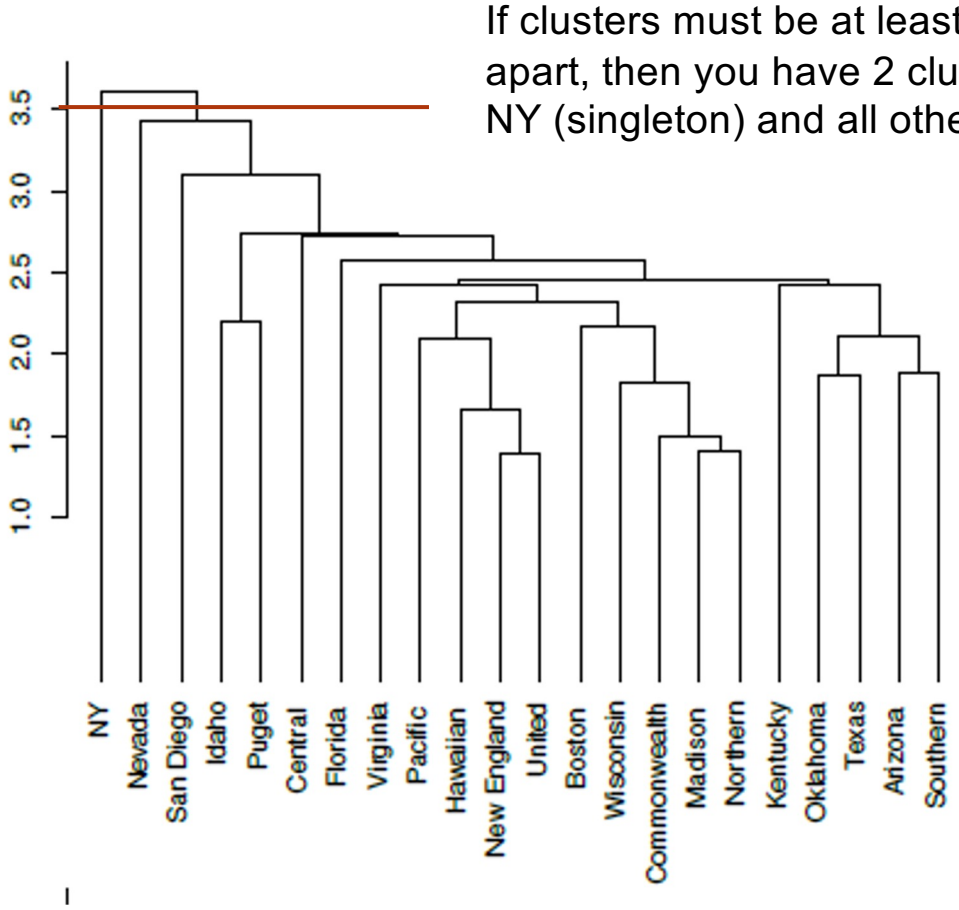
- New England and United will be merged first, then Madison and Northern

Determining number of clusters:

For a given "distance between clusters", a horizontal line intersects the clusters that are that far apart, to create clusters

- E.g., at distance of 3.5 (**red line** in next slide), data can be reduced to 2 clusters – NY (singleton cluster) and all others
- At distance of 2.6 (**green line in following slide**) data can be reduced to 6 clusters – 4 singletons, 1 doubleton, and all others

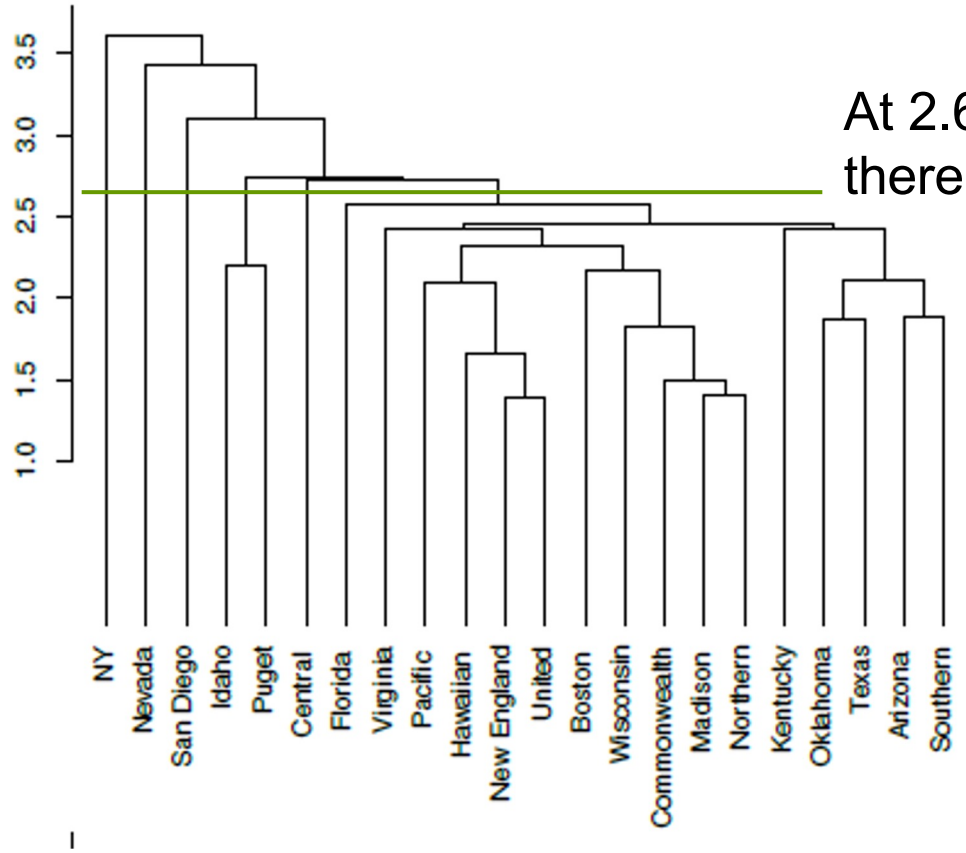
Dendrogram for Single Linkage



If clusters must be at least 3.5 units apart, then you have 2 clusters – NY (singleton) and all others

two closest records form first cluster

Dendrogram for Single Linkage

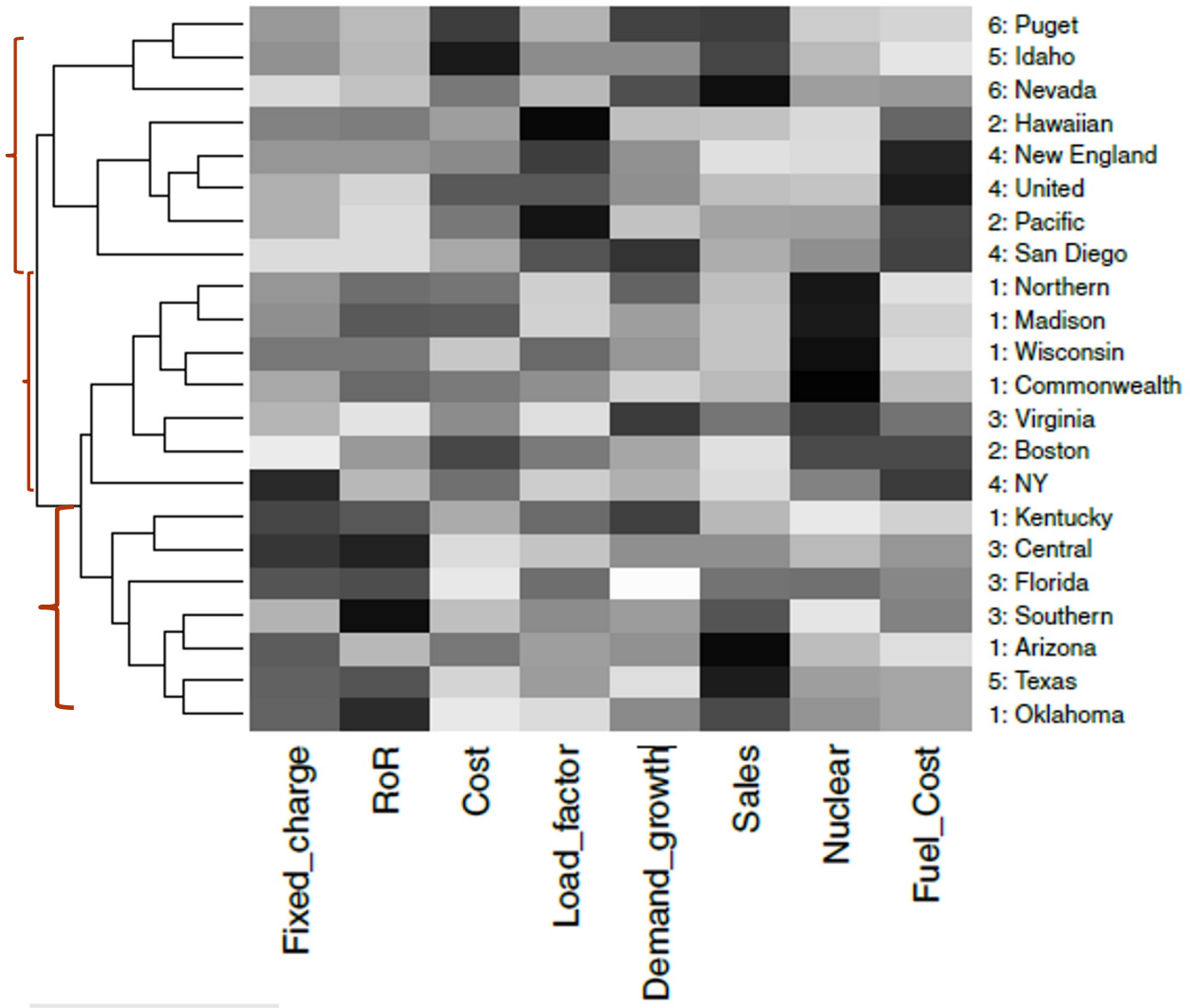


At 2.6 units of distance, there are 6 clusters

two closest records form first cluster

Visualize Cluster Features with Heatmap

(darker = higher values)



Considering 3 clusters, bottom cluster has higher Fixed_charge and RoR than the other two

Validating Clusters

Interpretation

Goal: obtain meaningful and useful clusters

Caveats:

(1) Random chance can often produce apparent clusters

(2) Different cluster methods produce different results

Solutions:

- Obtain summary statistics
- Also review clusters in terms of variables **not** used in clustering
- Label the cluster (e.g. clustering of financial firms in 2008 might yield label like “midsize, sub-prime loser”)

Desirable Cluster Features

Stability – are clusters and cluster assignments sensitive to slight changes in inputs? Are cluster assignments in partition B similar to partition A?

Separation – check ratio of between-cluster variation to within-cluster variation (higher is better)

Nonhierarchical Clustering:
K-Means Clustering

K-Means Clustering Algorithm

1. Choose # of clusters desired, *k*
2. Start with a partition into k clusters
   Often based on random selection of k centroids
3. At each step, move each record to cluster with closest centroid
4. Recompute centroids, repeat step 3
5. Stop when moving records increases within-cluster dispersion

K-means Algorithm:
Choosing k and Initial Partitioning

Choose *k* based on the how results will be used

e.g., “How many market segments do we want?”

Also experiment with slightly different *k*’s

Initial partition into clusters can be random, or based on domain knowledge

If random partition, repeat the process with different random partitions

Code for K-means Clustering

(data prep as for hierarchical)

ask for 6 clusters

```
km <- kmeans(utilities.df.norm, 6)

# show cluster membership
km$cluster
```

Output

```
> km$cluster
    Arizona      Boston     Central Commonwealth          NY
          3           2           3           4           1
    Florida    Hawaiian       Idaho    Kentucky     Madison
          3           2           6           3           4
     Nevada New England    Northern    Oklahoma     Pacific
          6           2           4           3           2
      Puget   San Diego    Southern       Texas   Wisconsin
          6           5           3           3           4
     United    Virginia
          2           4
```

Cluster Characteristics

Centroids = each cluster has a vector of variable means

```
> km$centers
  Fixed_charge        RoR       Cost Load_factor Demand_growth
1   2.03732429 -0.8628882  0.5782326  -1.2950193   -0.7186431
2  -0.35819462 -0.3637904  0.3985832   1.1572643   -0.3017426
3   0.50431607  0.7795509 -0.9858961  -0.3375463   -0.4895769
4  -0.01133215  0.3313815  0.2189339  -0.3580408    0.1664686
5  -1.91907572 -1.9323833 -0.7812761   1.1034665    1.8468982
6  -0.60027572 -0.8331800  1.3389101  -0.4805802    0.9917178
       Sales    Nuclear  Fuel_Cost
1 -1.5814284  0.2143888  1.6926380
2 -0.7080723 -0.4025746  1.1171999
3  0.3518600 -0.5232108 -0.4105368
4 -0.4018738  1.5650384 -0.5954476
5 -0.9014253 -0.2203441  1.4696557
6  1.8565214 -0.7146294 -0.9657660

> # within-cluster sum of squares
> km$withinss
[1]  0.000000 11.935249 26.507769 10.177094  0.000000  9.533522

> # cluster size
> km$size
[1] 1 5 7 5 1 3
```

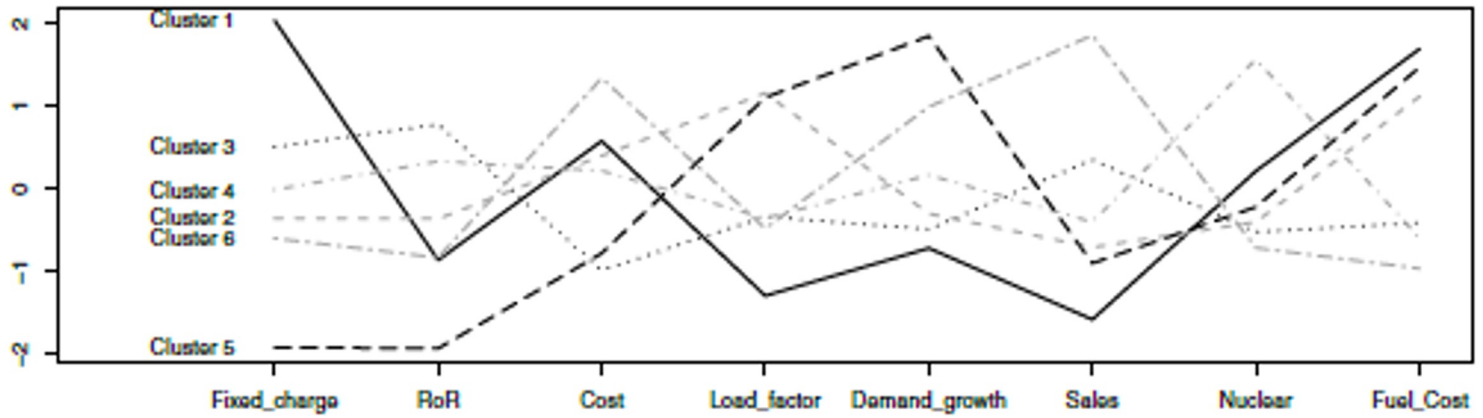
Custer 1 has normalized average demand growth of -0.7186

Clusters 1 and 5 are each singletons, so within-cluster distance = 0

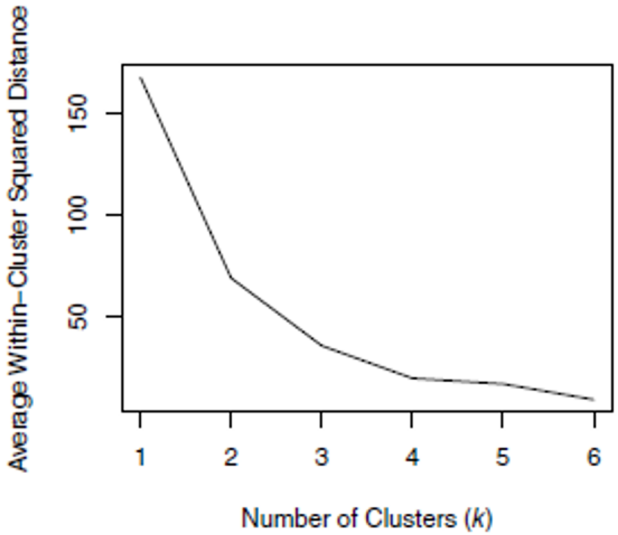
number of utilities in each cluster

Visualize Cluster Centroids with Profile Plot

```
# plot an empty scatter plot
plot(c(0), xaxt = 'n', ylab = "", type = "l", ylim =
     c(min(km$centers), max(km$centers)), xlim = c(0, 8))
# label x-axes
axis(1, at = c(1:8), labels = names(utilities.df))
# plot centroids
for (i in c(1:6))
lines(km$centers[i,], lty = i, lwd = 2, col = ifelse(i %in% c(1, 3,
5),"black", "dark grey"))
# name clusters
text(x = 0.5, y = km$centers[, 1], labels = paste("Cluster", c(1:6)))
```



Different Choices for K



As the number of clusters increases, the cluster members are closer to one another.

Question: What happens to the distance *between* clusters?

Summary

- Cluster analysis is an exploratory tool.
- Useful only when it produces **meaningful** clusters
- **Hierarchical** clustering gives visual representation of different levels of clustering
  - On other hand, due to non-iterative nature, it can be unstable, can vary highly depending on settings, and is computationally expensive
- **Non-hierarchical** is computationally cheap and more stable; requires user to set *k*
- Can use both methods
- Be wary of chance results; data may not have definitive “real” clusters