The Business School for the World®

DS(ML)B: Data Science (& Machine Learning) for Business

Profs. Anton Ovchinnikov, Theos Evgeniou, Spyros Zoumpoulis

Sessions 09-10

Unsupervised Learning

- Clustering and Segmentation
- Dimensionality Reduction (Principal Component Analyses, PCA)
- [Optional / Time-permitting] Association Rules, Anomaly Detection

Plan for the day
Learning objectives

- Clustering and segmentation:
  - Group observations in a few segments so that data within any segment are similar while data across segments are different
- Derived attributes and dimensionality reduction:
  - Generate a small number of new variables (« principal components ») that capture most of the information in the data

- Association Rules:
  - « I don't often buy milk, but when I do, I also buy beer »

- Anomaly Detection:
  - « Is this an outlier? » (fraud, non-human log-in, abnormal drop in sales, increase in processing time, etc.)

Clustering and Segmentation

Main idea: Processes and tools to organize data into segments so that data is as **similar** as possible **within** each segment, and as **different** as possible **across** segments

Applications https://en.wikipedia.org/wiki/Cluster_analysis#Applications :

- Market, demographic, geospatial, etc. segmentation
- Feature engineering for supervised learning by "combining" data (e.g., asset classes, customer types, usage patterns, etc.) → **transfer learning** [Yahoo/Tumblr example]
- Anomaly detection [Time-permitting, later today]

Methods: many, but we will look at two most popular:

- K-means clustering
- Hierarchical clustering

How will we do it?

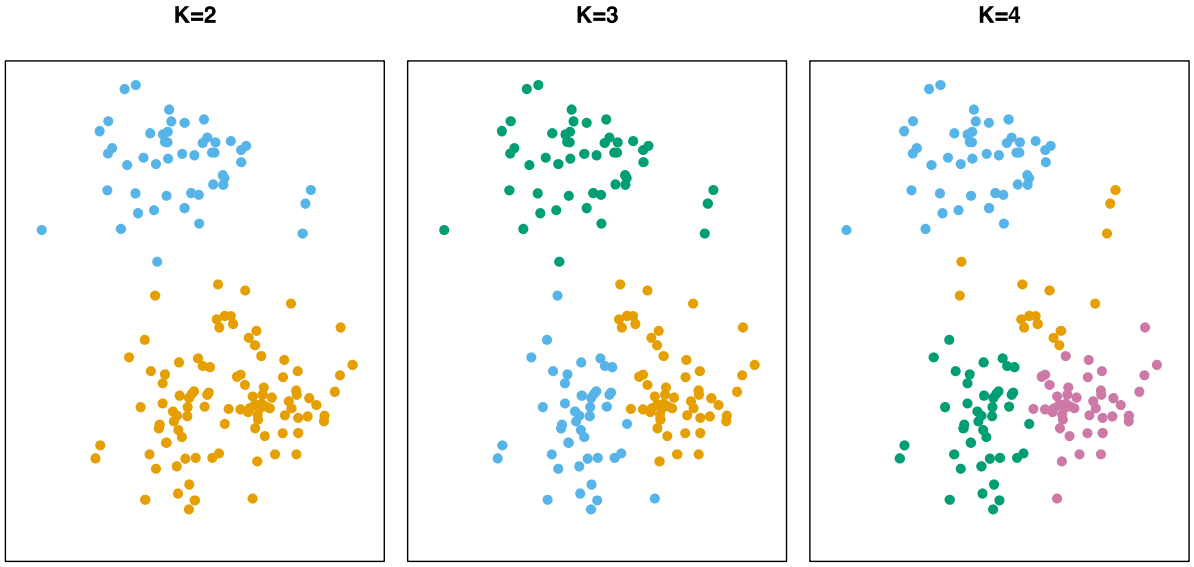
- Start with a simple "synthetic data" example – Lecture
- Then move to real data from Kaggle – Group work


INSEAD
The Business School
for the World®
Pikachu
Pika! Pika!*
* Wear your badge

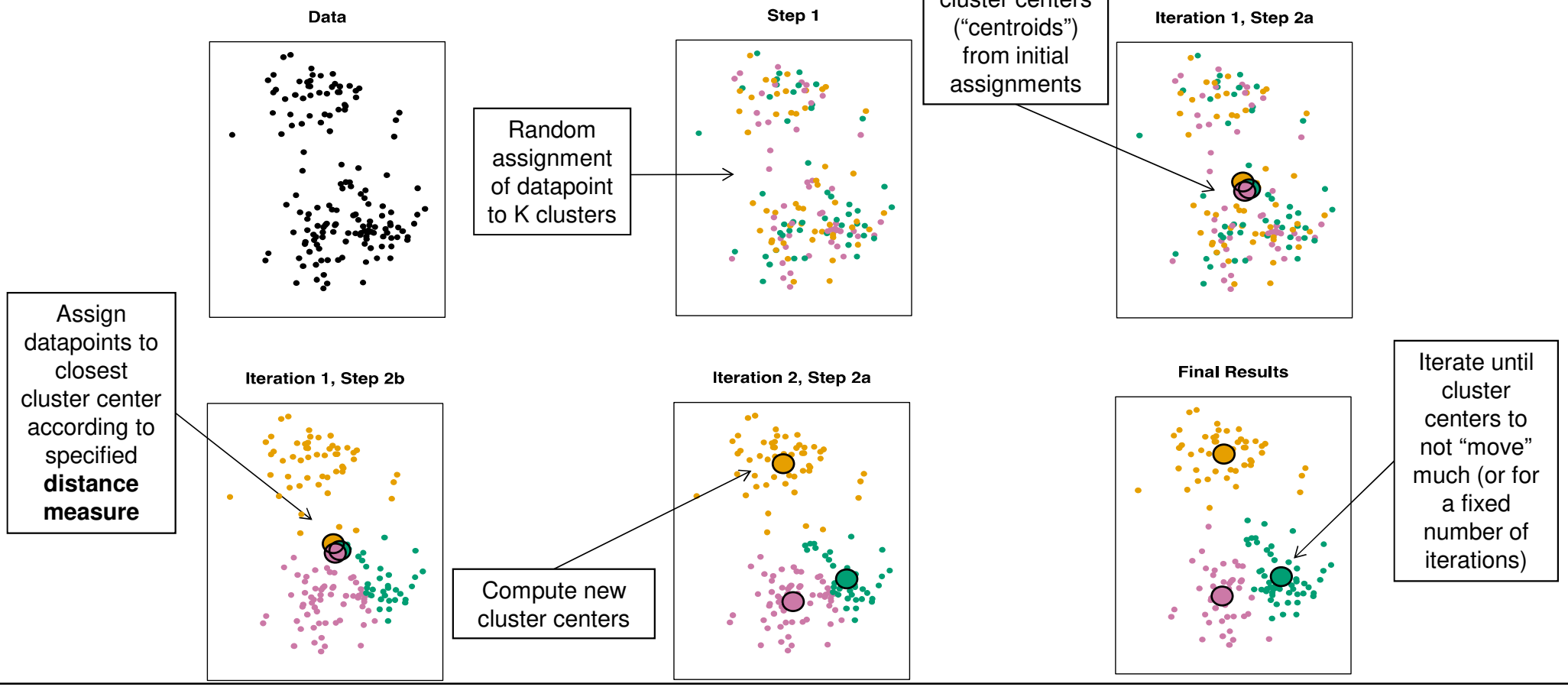
K-Means

One of the most popular clustering algorithms:

- Fast, easy to understand
- User specifies the desired number of clusters, K
- Algorithm assigns each datapoint to exactly one of the K clusters

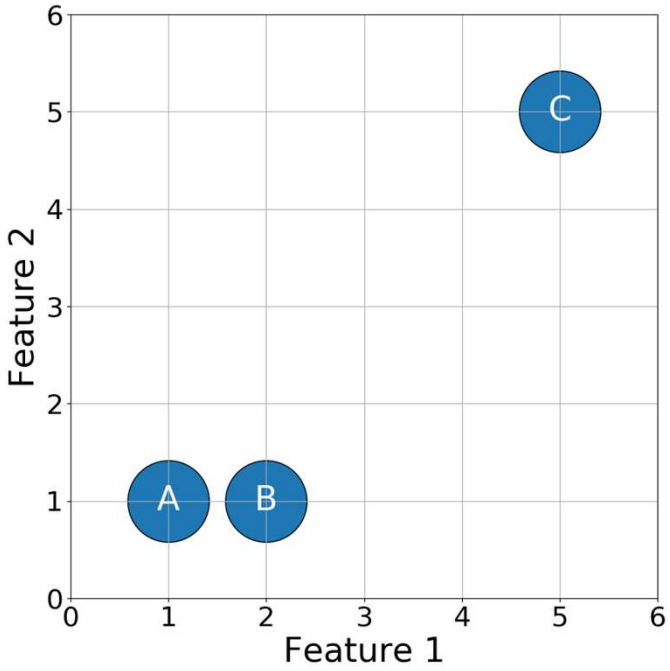


How Does it Work?
Main idea: Centroids



Distance Metrics

- Clustering algorithms require you to specify the distance metric
- ***Distance metrics*** measure how "far apart" two datapoints are from each other; equivalently: how "close" they are



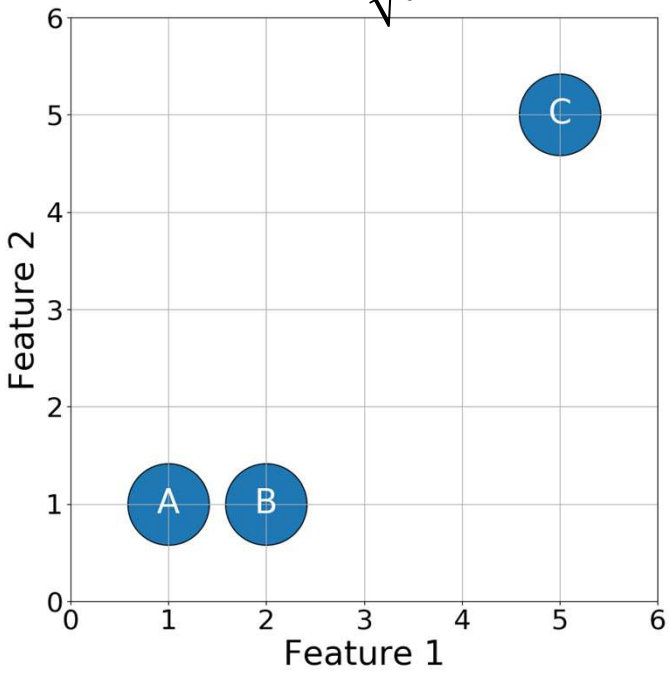
ID	Feature 1	Feature 2
A	1.0	1.0
B	2.0	1.0
C	5.0	5.0

Common distance metrics:

- Euclidean
- Cosine
- Manhattan
- Chebyshev
- Canberra
- Pearson Correlation
- Hamming
- Jaccard

Euclidean Distance
Length of the "straight line“ between two datapoints

$$dist_{euc}(A,B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

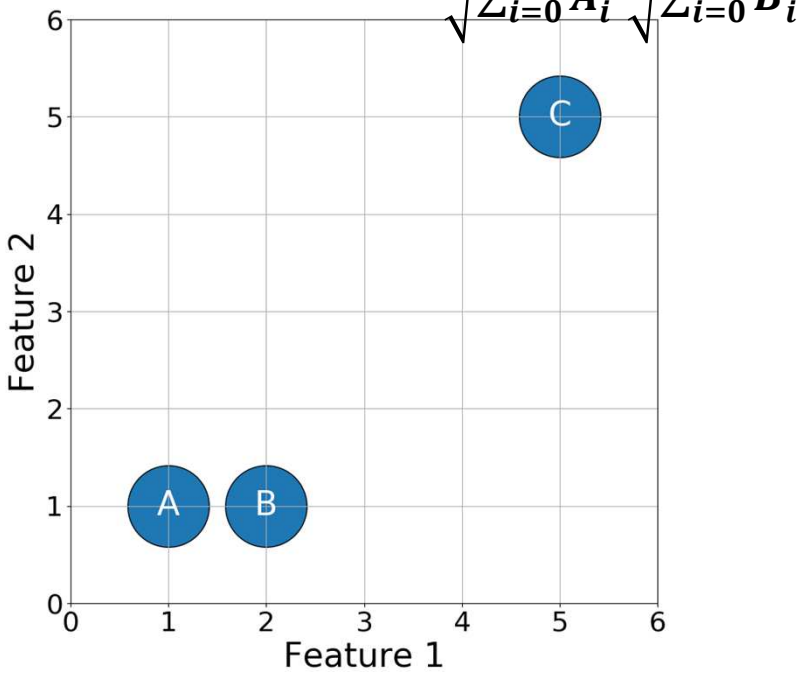


ID	Feature 1	Feature 2
A	1.0	1.0
B	2.0	1.0
C	5.0	5.0

	A, B	A, C	B, C
Euclidean	1.00	5.66	5.00

Cosine Distance
Measures the cosine of the angle between two vectors

$$dist_{cos}(A, B) = 1 - \frac{\sum_{i=0}^{n} A_i B_i}{\sqrt{\sum_{i=0}^{n} A_i^2}\sqrt{\sum_{i=0}^{n} B_i^2}}$$

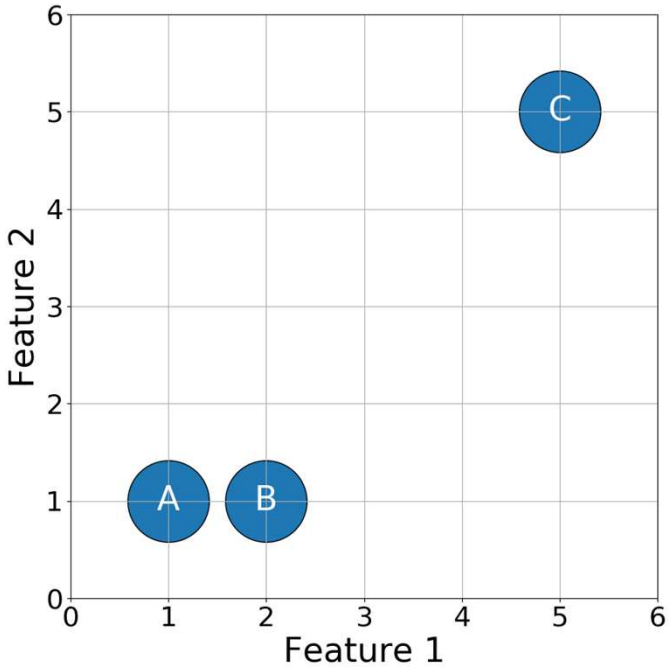


ID	Feature 1	Feature 2
A	1.0	1.0
B	2.0	1.0
C	5.0	5.0

	A, B	A, C	B, C
Euclidean	1.00	5.66	5.00
Cosine	0.05	0.00	0.05

Manhattan Distance
Length of “block driving distance” between two points

$$dist_{man}(A, B) = \sum_{i=1}^{n} |A_i - B_i|$$

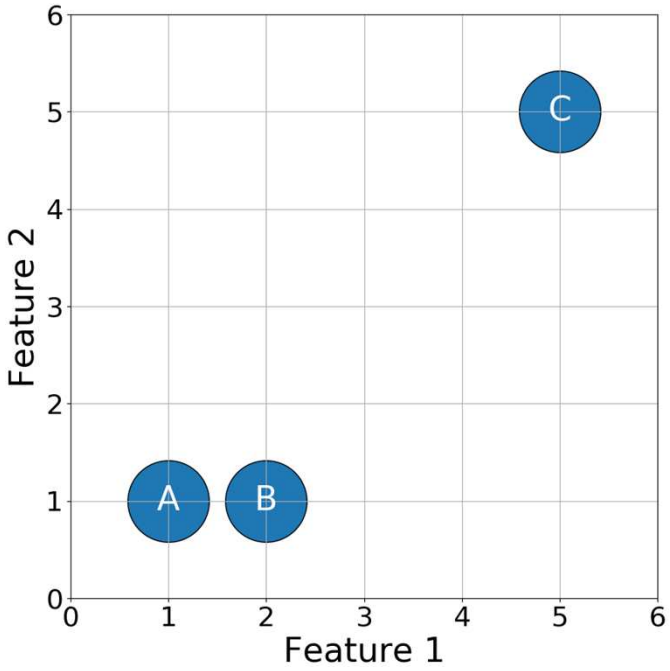


ID	Feature 1	Feature 2
A	1.0	1.0
B	2.0	1.0
C	5.0	5.0

	A, B	A, C	B, C
Euclidean	1.00	5.66	5.00
Cosine	0.05	0.00	0.05
Manhattan	1.00	8.00	7.00

Chebychev Distance
Greatest of the distances along any dimension

$$dist_{cheby}(A, B) = \max_i |A_i - B_i|$$



ID	Feature 1	Feature 2
A	1.0	1.0
B	2.0	1.0
C	5.0	5.0

	A, B	A, C	B, C
Euclidean	1.00	5.66	5.00
Cosine	0.05	0.00	0.05
Manhattan	1.00	8.00	7.00
Chebychev	1.00	4.00	4.00

Which Distance Metric is Best?

- Distance metric is very important
  - Stronger influence on results than the clustering algorithm
  - Euclidean is very common; default in most packages

- A recent research paper suggests:
  - Euclidean is best all-purpose
  - Cosine is better when absolute value is not important
    - E.g., in business: customer shopping patterns, “popularity” of items within categories
    - E.g., in sciences: gene expressions

- As always, trial-and-error is often used

Distance Measure	Equation	Time complexity	Advantages	Disadvantages	Applications
Euclidean Distance	$d_{euc} = \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}$	O(n)	Very common, easy to compute and works well with datasets with compact or isolated clusters [27,31].	Sensitive to outliers [27,31].	*K*-means algorithm, Fuzzy *c*-means algorithm [38].
Average Distance	$d_{ave} = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2\right)^{\frac{1}{2}}$	O(n)	Better than Euclidean distance [35] at handling outliers.	Variables contribute independently to the measure of distance. Redundant values could dominate the similarity between data points [37].	*K*-means algorithm
Weighted Euclidean	$d_{we} = \left(\sum_{i=1}^{n}w_i(x_i - y_i)^2\right)^{\frac{1}{2}}$	O(n)	The weight matrix allows to increase the effect of more important data points than less important one [37].	Same as Average Distance.	Fuzzy *c*-means algorithm [38]
Chord	$d_{chord} = \left(2 - 2\frac{\sum_{i=1}^{n}x_i y_i}{\|x\|_2\|y\|_2}\right)^{\frac{1}{2}}$	O(3n)	Can work with un-normalized data [27].	It is not invariant to linear transformation [33].	Ecological resemblance detection [35].
Mahalanobis	$d_{mah} = \sqrt{(x-y)S^{-1}(x-y)^T}$	O(3n)	Mahalanobis is a data-driven measure that can ease the distance distortion caused by a linear combination of attributes [35].	It can be expensive in terms of computation [33]	Hyperellipsoidal clustering algorithm [30].
Cosine Measure	$\mathrm{Cosine}(x,y) = \frac{\sum_{i=1}^{n}x_i y_i}{\|x\|_2\|y\|_2}$	O(3n)	Independent of vector length and invariant to rotation [33].	It is not invariant to linear transformation [33].	Mostly used in document similarity applications [28,33].
Manhattan	$d_{man} = \sum_{i=1}^{n}(x_i - y_i)$	O(n)	Is common and like other Minkowski-driven distances it works well with datasets with compact or isolated clusters [27].	Sensitive to the outliers. [27,31]	*K*-means algorithm
Mean Character Difference	$d_{MCD} = \frac{1}{n}\sum_{i=1}^{n}\|x_i - y_i\|$	O(n)	*Results in accurate outcomes using the K-medoids algorithm.	*Low accuracy for high-dimensional datasets using K-means.	Partitioning and hierarchical clustering algorithms.
Index of Association	$d_{IOA} = \frac{1}{n}\sum_{i=1}^{n}\left\|\frac{x_i}{\sum_{i=1}^{n}x_i} - \frac{y_i}{\sum_{i=1}^{n}y_i}\right\|$	O(3n)	-	*Low accuracy using K-means and K-medoids algorithms.	Partitioning and hierarchical clustering algorithms.
Canberra Metric	$d_{canb} = \sum_{i=1}^{n}\frac{\|x_i - y_i\|}{(x_i + y_i)}$	O(n)	*Results in accurate outcomes for high-dimensional datasets using the K-medoids algorithm.	-	Partitioning and hierarchical clustering algorithms.
Czekanowski Coefficient	$d_{czekan} = 1 - \frac{2\sum_{i=1}^{n}min(x_i, y_i)}{\sum_{i=1}^{n}(x_i + y_i)}$	O(2n)	*Results in accurate outcomes for medium-dimensional datasets using the K-means algorithm.	-	Partitioning and hierarchical clustering algorithms.
Coefficient of Divergence	$d_{canb} = \left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - y_i}{x_i + y_i}\right)^2\right)^{\frac{1}{2}}$	O(n)	*Results in accurate outcomes using the K-means algorithm.	-	Partitioning and hierarchical clustering algorithms.
Pearson coefficient	$Pearson(x,y) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}}$	O(2n)	*Results in accurate outcomes using the hierarchical single-link algorithm for high dimensional datasets.	-	Partitioning and hierarchical clustering algorithms.

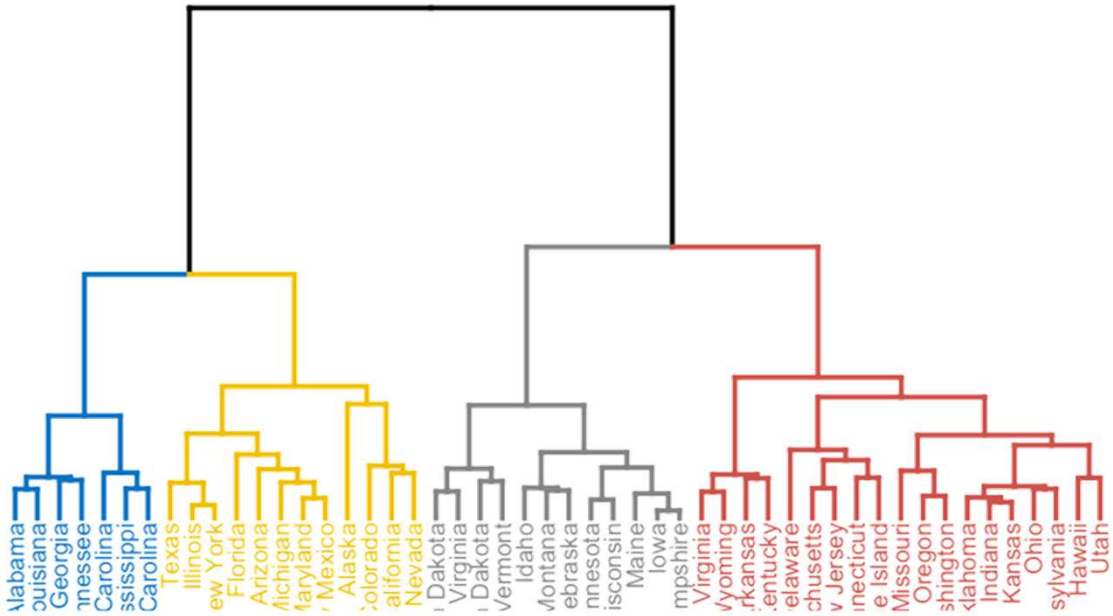
*Points marked by asterisk are compiled based on this article's experimental results.

doi:10.1371/journal.pone.0144059.t001

Shirkhorshidi AS, Aghabozorgi S, Wah TY (2015) A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144059

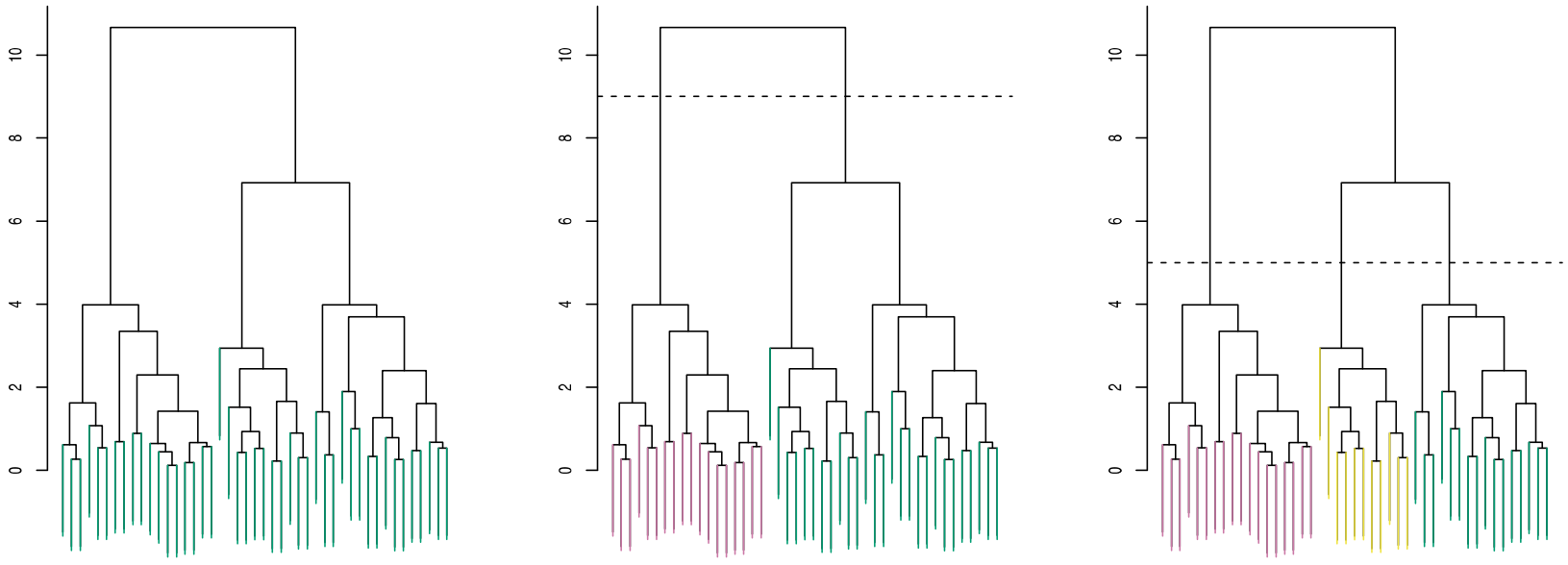
Hierarchical Clustering

- Builds a hierarchy of “nested” clusters by “fusing” datapoints together
  - Agglomerative: bottom up approach (more common)
  - Divisive: top down approach
- Resulting hierarchy is shown as a dendogram:

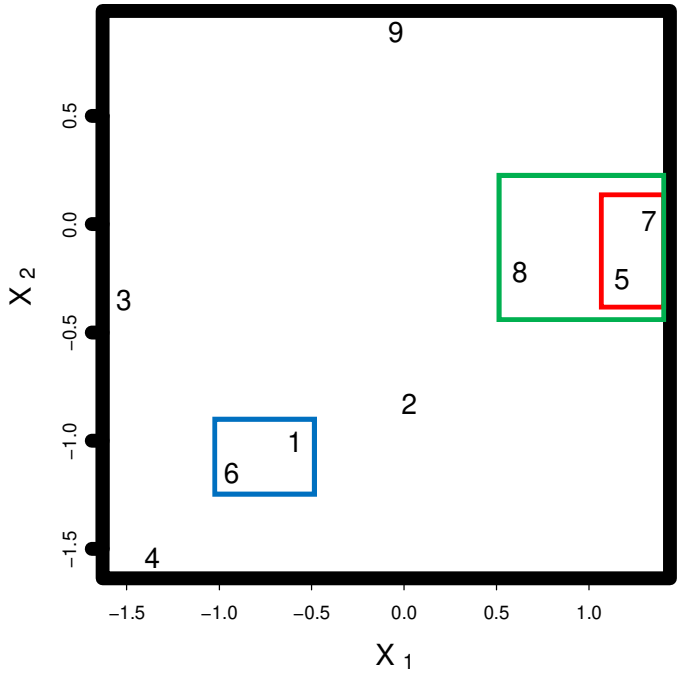


How to Use a Dendogram to Create Clusters?

- To create clusters, we cut the dendogram at a given height
- We can form any number of clusters depending on where we draw the line (aka, ***break point***)



How Does it Work?
Main idea: Dendogram



Say we have two features and 9 datapoints

Initially, each of 9 points is a cluster of its own

Then:

- First, closest two points are fused: (5, 7)
- Next closest two points are fused: (6, 1)
- Next closest two points are fused: (8, (5, 7))

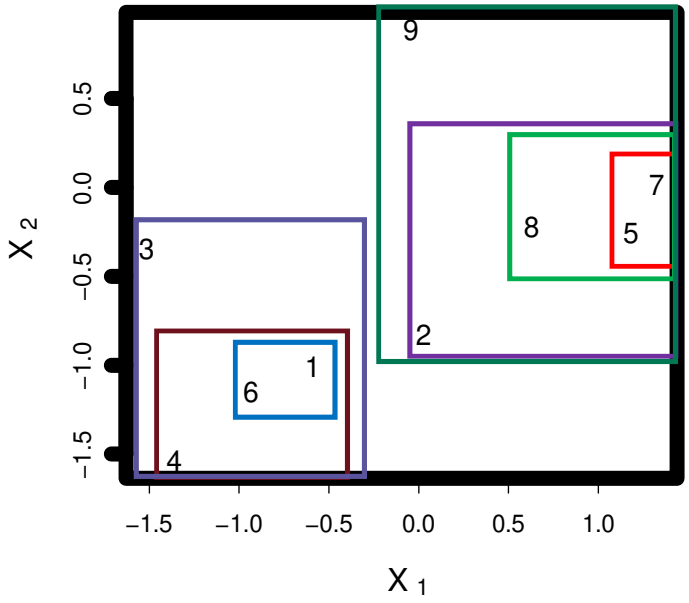
…

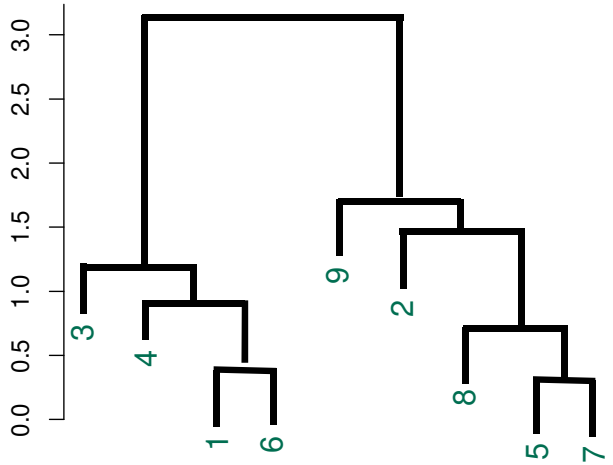
Continue until all datapoints are fused

How Does it Work?
Main idea: Dendogram

A dendogram basically "keeps track" of all the fuses

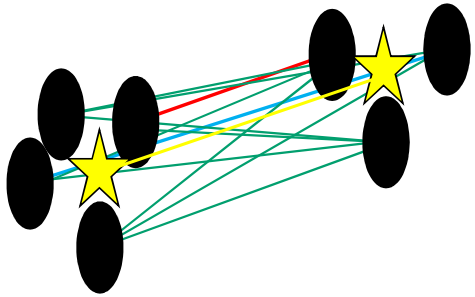
Height of fusing/merging (on vertical axis) indicates how similar the points are (per the selected distance measure)





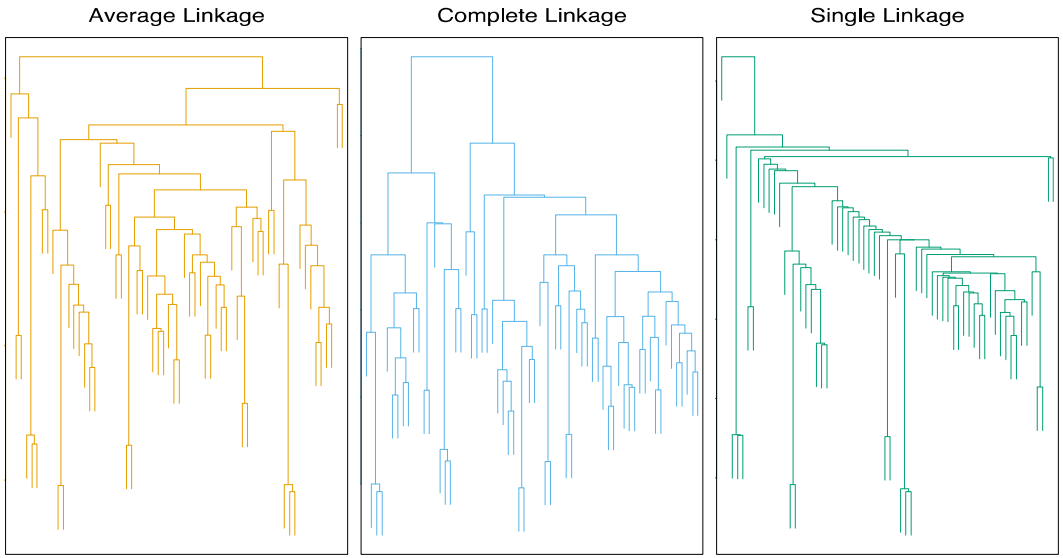
Distance measures cont.
Linkages

- How do we define the similarity/dissimilarity, or linkage, between fused points and a single point?
  - E.g., between the fused (5,7) cluster and datapoint 8?
- Use a **linkage**:
  - **Single** linkage***:*** Smallest distance between instances
  - **Complete** linkage***:*** Largest distance between instances
  - **Average** linkage***:*** Average distance between instances
  - **Centroid*:*** Distance between centroids of the instances



Linkage Can be Important

- Same dataset, same distance measure, different linkage
  - Complete and average linkage tend to yield evenly sized clusters
  - Single linkage tends to yield extended clusters to which single leaves are fused one by one
  - **Rule of thumb**: use **complete** or **average**



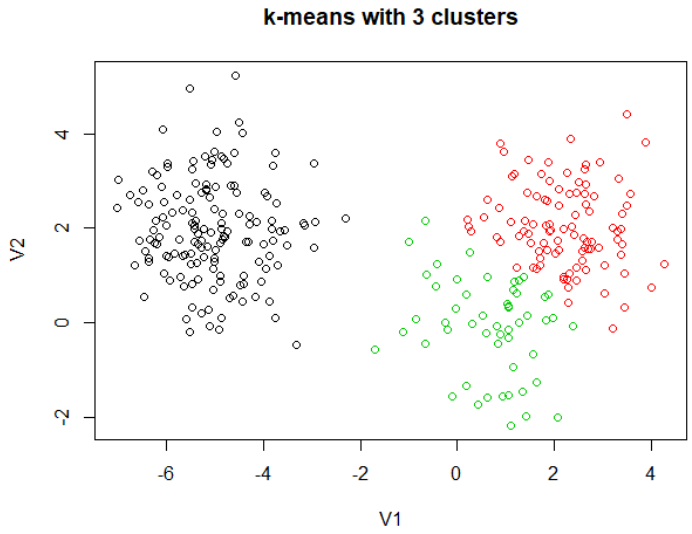
Now lets practice!
K-means

Data: 0910 CSV data -- for clustering.csv

Code: 0910 R code – clustering.R

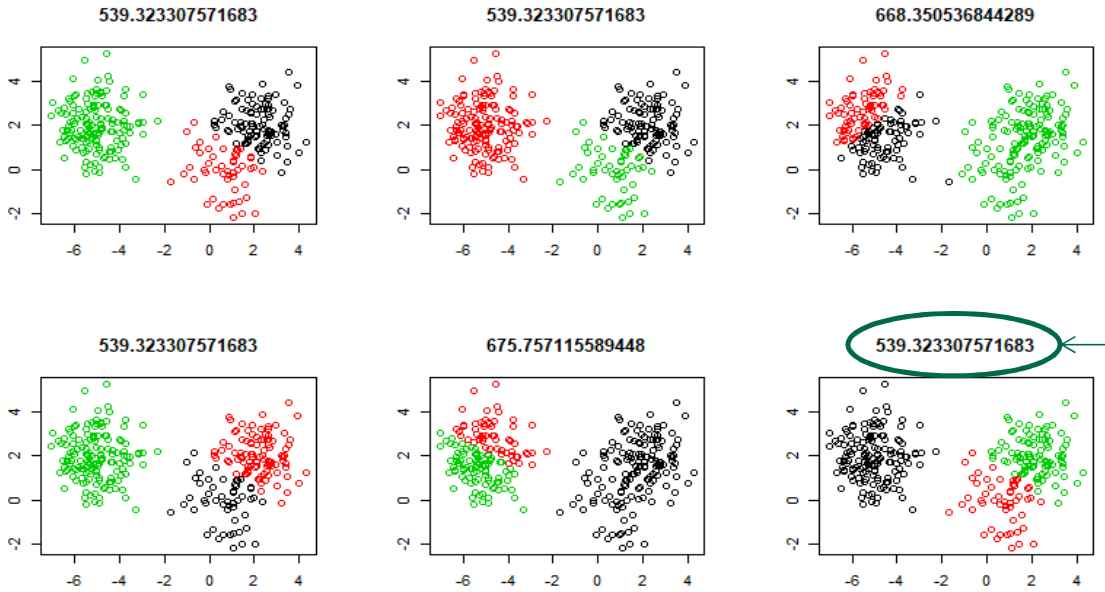
```
Clustering.data=read.csv(file.choose(), header=FALSE)
km.out<-kmeans(Clustering.data, centers=3, nstart=20, iter.max = 50)
km.out$centers # Print averages for centeroids by variable
km.out$cluster # Print the cluster memberships
plot(Clustering.data, col=km.out$cluster, main="k-means with 3 clusters")
```

```
> km.out$centers # Print averages for centeroids
         V1          V2
1 -5.0556758  1.96991743
2  2.2171113  2.05110690
3  0.6642455 -0.09132968
> km.out$cluster # Print the cluster memberships
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 3 3 2 2
 [31] 2 2 2 2 2 2 3 3 3 2 2 2 2 3 2 2 2 2 2 2 2
 [61] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2
 [91] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1
[121] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[151] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[241] 1 1 1 1 1 1 1 1 1 1 3 3 3 3 2 3 3 3 3 3 3
[271] 3 3 3 3 3 2 2 3 3 2 3 3 3 3 3 3 2 3 3 3 3
```



K-means: process & randomness in cluster creation

```
par(mfrow = c(2, 3))
for(i in 1:6) {
  # Run kmeans() with three clusters and one start
  km.out <- kmeans(Clustering.data, 3, 1, iter.max = 50)
  # Plot clusters
  plot(Clustering.data, col = km.out$cluster,main = km.out$tot.withinss)}
```

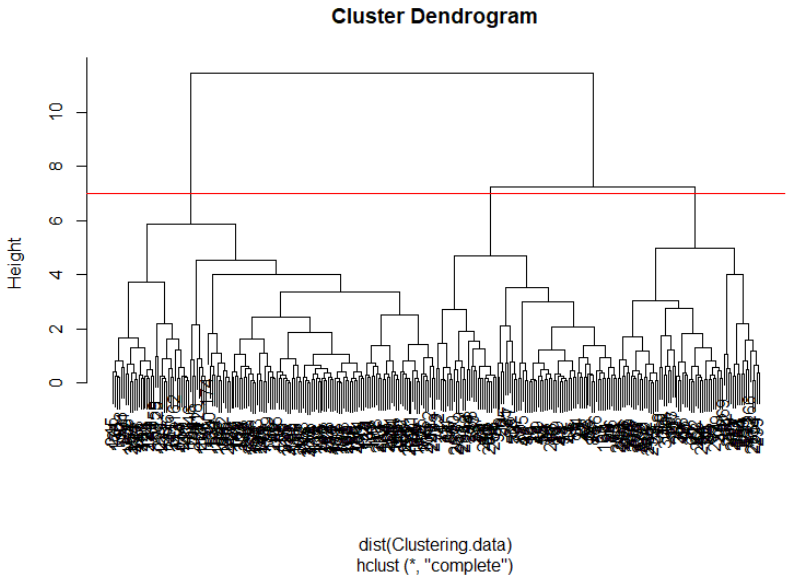


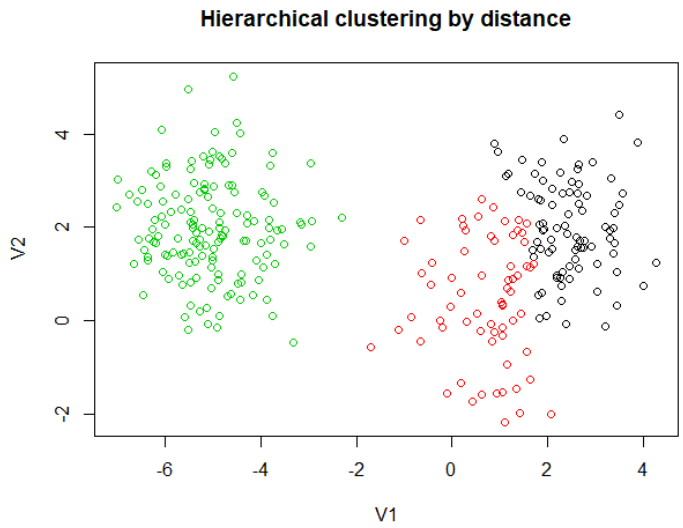
Total within-cluster sum of squares (wss)

The lower, the better

Now lets practice!
Hierarchical clustering

```
hclust.out<-hclust(d=dist(Clustering.data), method="complete")
plot(hclust.out) abline(h=7, col="red") # Plot the dendogram and a cut
clusters_h_method<-cutree(hclust.out, h=7) # Cut by height
plot(Clustering.data, col=clusters_h_method, main="Hierarchical clustering by
distance")
clusters_k_method<-cutree(hclust.out, k=3) # Cut by number of clusters
plot(Clustering.data, col=clusters_k_method, main="Hierarchical clustering by number")
```





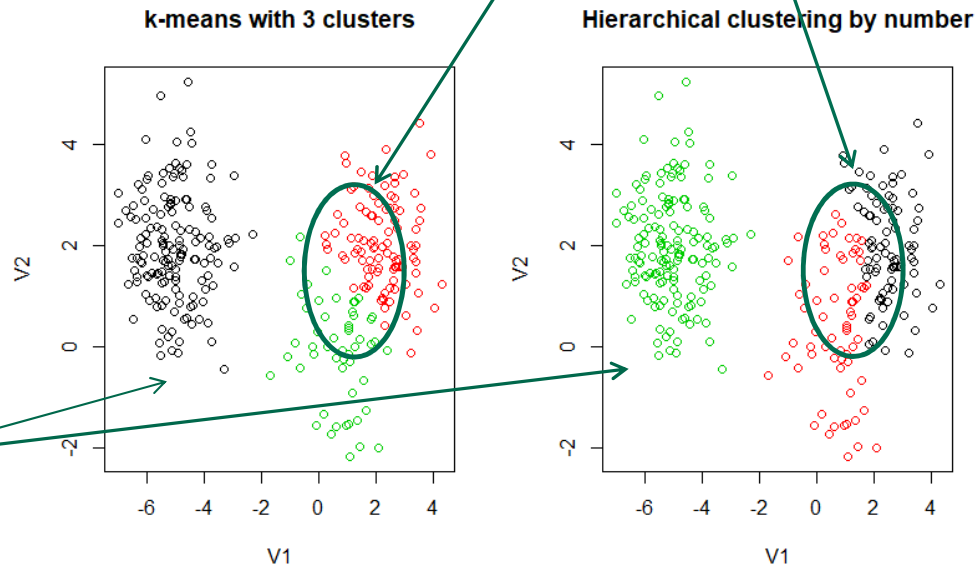
Comparing the resultant clusters

```
table(km.out$cluster)
table(clusters_k_method)
table(km.out$cluster, clusters_k_method)
par(mfrow = c(1, 2))
plot(Clustering.data, col=km.out$cluster, main="k-means with 3 clusters")
plot(Clustering.data, col=clusters_k_method, main="Hierarchical clustering by number")
```

"Semantic" differences in colors/labels aside, this is where the real difference is

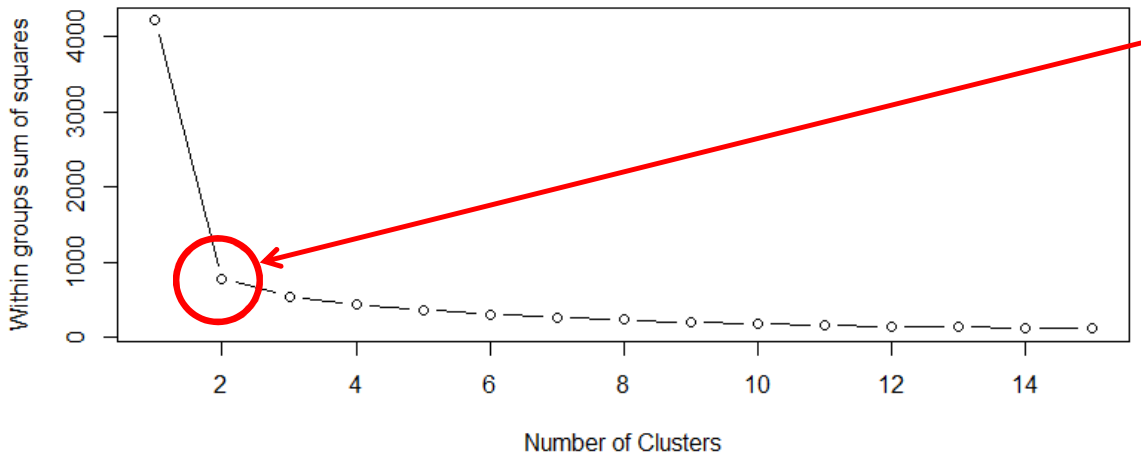
```
> table(km.out$cluster)

  1   2   3
150  98  52
> table(clusters_k_method)
clusters_k_method
  1   2   3
 84  66 150
> table(km.out$cluster, clusters_k_method)
   clusters_k_method
     1   2   3
  1  0   0 150
  2 79  19   0
  3  5  47   0
```

The match of cluster labels ("colors") is not exact b/c of randomness



How many clusters?
“Elblow” plot

```
wss <- 0 # Initialize total within sum of squares error: wss
# Test for k = 1 to 15 cluster centers
for (i in 1:15) {
  km.out <- kmeans(Clustering.data, centers = i, nstart=20)
  wss[i] <- km.out$tot.withinss }
# Plot total within sum of squares vs. number of clusters
par(mfrow = c(1, 1))
plot(1:15, wss, type = "b", xlab = "Number of Clusters", ylab = "Within groups sum of
squares")
```



Rule of thumb: set number of clusters at the “elbow” of the plot

In practice: start with above rule, then explore different numbers of clusters based on interpretability / business usefulness

Interpreting the results
"Snake" plot

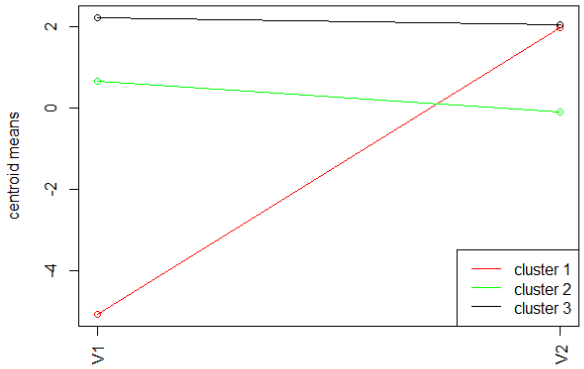

```
ul<-max(km.out$centers)# upper and lower limits for plotting
ll<-min(km.out$centers)
plot(km.out$centers[1,], type = "o", col="red", ylim=(c(ll,ul)), xlab=NA,
xaxt="n", ylab="centroid means")
axis(1,at=1:ncol(Clustering.data), las=2, labels=c(colnames(Clustering.data)))
lines(km.out$centers[2,], type = "o", col="green")
lines(km.out$centers[3,], type = "o", col="black")
legend("bottomright", legend=c("cluster 1", "cluster 2","cluster 3"),
col=c("red", "green", "black"), lty=1)
```

"base" plot for cluster 1

axis with var. names

add lines for clusters 2, 3

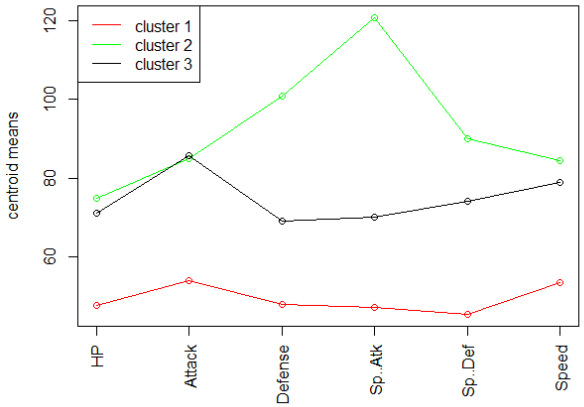
add legend



In our example: not much of a "snake" since we have 2 variables only

In general: clear "business" intuition:

cluster 1 are the datapoints when all variables are "low")



Summary: clustering

An **unsupervised learning** technique of grouping a set of objects in a such a way that objects in the same group (called a cluster) are more similar to each than to those in other groups (clusters)

Applications
- Customer segmentation, outlier detection, feature engineering, …

Algorithms
- Centroid: k-mean, k-medians, k-modes, etc.
- Connectivity: Hierarchical
- Distribution: Gaussian Mixture Models, Density: DBSCAN, OPTICS

Distance metrics
- Euclidean, Cosine, Manhattan, Chebyshev, Canberra, Pearson Correlation, Hamming, Jaccard

Evaluation
- Measure internal validation metric, like within sum of squares
- How many clusters: “Elbow” plot
- Interpretability: class memberships, “Snake” plot

Robustness: The segments found should be relatively robust to changes in the clustering methodology; large changes indicate that segmentation may not be valid
- How much overlap is there between the clusters found using different approaches (clustering methods, distance metrics, etc.)? How similar are the profiles of the segments found?
- Is the result sensitive to subsets of the original rows (“new data”)?

Next:
Additional considerations + practice on “non-toy” data

1. Scaling of data
2. Clustering with categorical variables (“factors” in R)

   … both will be discussed a bit later

Next: practice: **~30 mins** in **BORs**

- Have you downloaded the Pokemon data from Kaggle?
- Use numerical columns only, use first 100 rows only

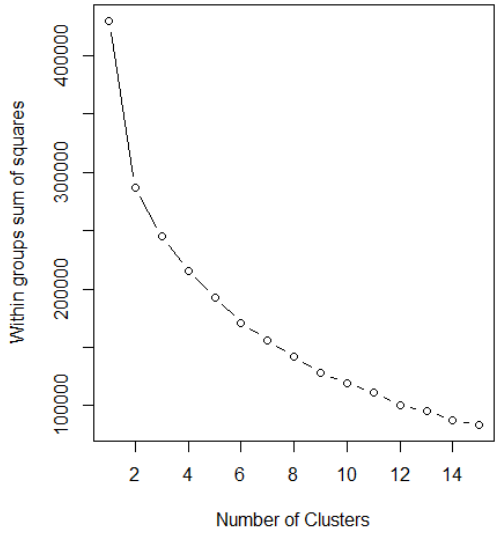
```
Pokemon.data<-Pokemon.data[c(1:100),]
Pokemon.numerical.data<-Pokemon.data[,c(6:11)]
rownames(Pokemon.numerical.data)<-Pokemon.data$Name
```

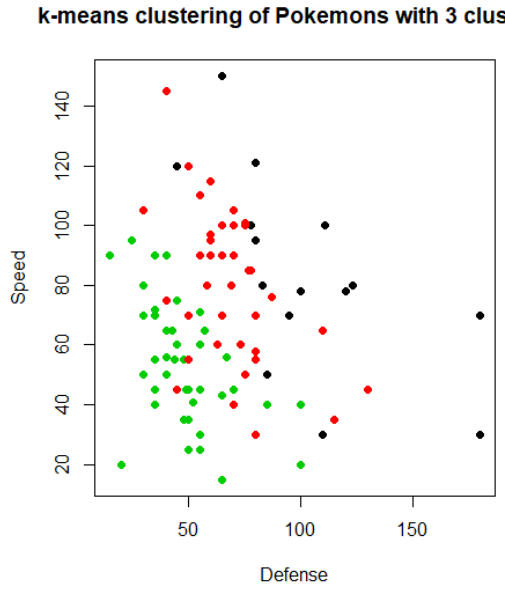
- **Q**: What does the data tell us about the different kinds of Pokemon?

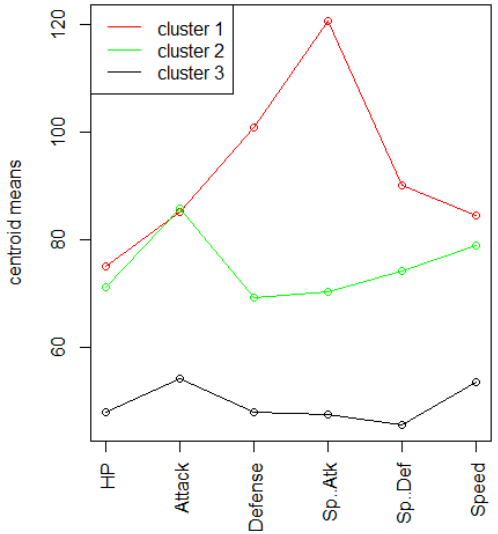
- Hint: split: ~2ppl work on visuals (Excel, Tableau), the rest do clustering

Pokemon Analyses, 1/3

K-means method
No evident “elbow”, lets pick k=3 clusters for now
2D plot makes sense
Snake plot also makes sense (clusters are indeed different)





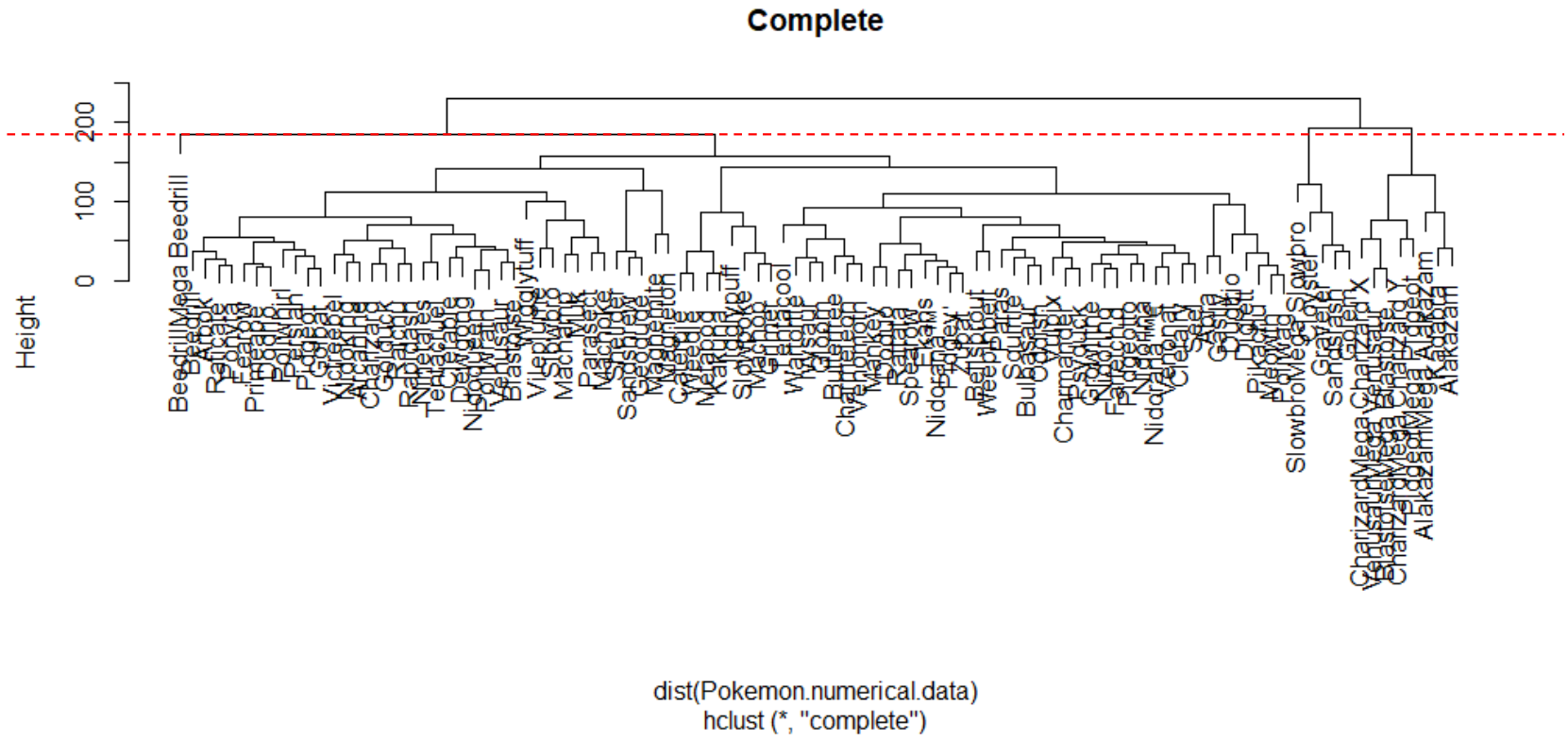


Code: 0910 R code – pokemon.R

Pokemon Analyses, 2/3

Hierarchical method.

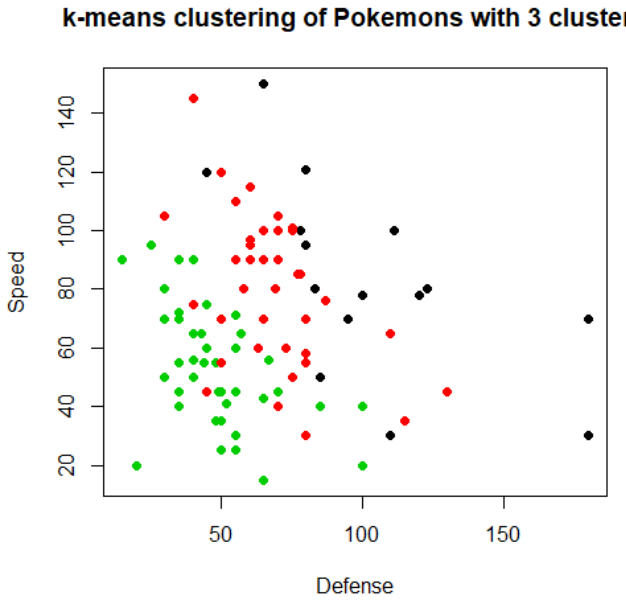
Maybe 3 clusters are not the best (very narrow height when k=3)

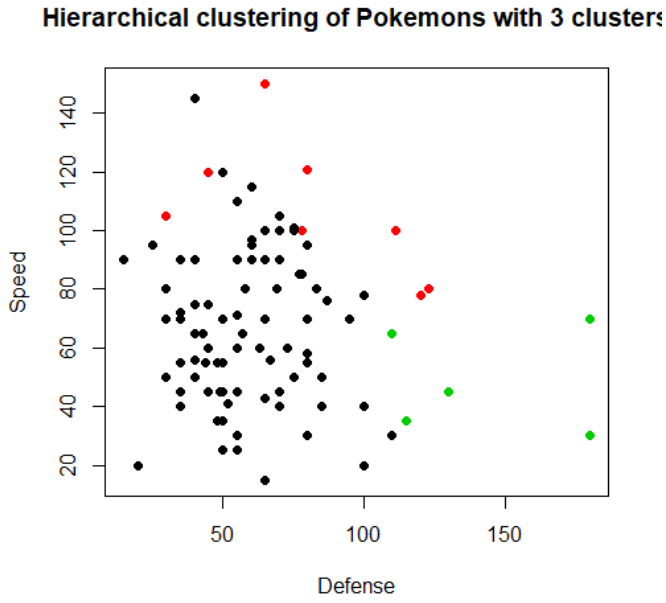


Pokemon Analyses, 3/3

Two methods result in very different clusters
Not robust; need to continue exploring, e.g., other k

```
> table(km.pokemon$cluster)

 1  2  3
16 40 44
> table(cut.pokemon)
cut.pokemon
 1  2  3
87  8  5
> table(km.pokemon$cluster, cut.pokemon)
   cut.pokemon
     1  2  3
  1  7  7  2
  2 36  1  3
  3 44  0  0
```





Next:
Additional considerations

1. Scaling of data
2. Clustering with categorical variables ("factors" in R)

   … for that – Dimensionality Reduction and Principal Component Analyses

Next: practice: ~**30 mins** in **BORs**

- Have you downloaded the Pokemon data from Kaggle?
- Use numerical columns only, use first 100 rows only

```
Pokemon.data<-Pokemon.data[c(1:100),]
Pokemon.numerical.data<-Pokemon.data[,c(6:11)]
rownames(Pokemon.numerical.data)<-Pokemon.data$Name
```

- **Q**: What does the data tell us about the different kinds of Pokemon?

- Hint: split: ~2ppl work on visuals (Excel, Tableau), the rest do clustering

Additional considerations
Scaling

Main idea: PCA (and clustering) minimize sum of distances, so if the data has very different units of measurement (different variances) then the results will be meaningless. E.g., Price (1000s) vs fuel economy (10s) vs # of cylinders (~3-4-6) in a car

```
# View column means
colMeans(Pokemon.numerical.data)

# View column standard deviations
apply(Pokemon.numerical.data, 2, sd)

# Scale the data
Pokemon.numerical.data.scaled<-scale(Pokemon.numerical.data)
colMeans(Pokemon.numerical.data.scaled)
```

```
> # View column means
> colMeans(Pokemon.numerical.data)
     HP  Attack Defense Sp..Atk Sp..Def   Speed
  61.52   71.69   64.86   68.21   64.16   68.63
>
> # View column standard deviations
> apply(Pokemon.numerical.data, 2, sd)
      HP   Attack  Defense  Sp..Atk  Sp..Def    Speed
21.58773 24.98027 28.98381 32.45706 23.69918 28.18648
>
> # Scale the data
> Pokemon.numerical.data.scaled<-scale(Pokemon.numerical.data)
> colMeans(Pokemon.numerical.data.scaled)
           HP        Attack       Defense       Sp..Atk       Sp..Def         Speed
-1.387432e-16  1.060090e-16  2.165802e-17  1.808970e-16  1.334002e-16  1.590394e-16
```

Derived Attributes and Dimensionality Reduction

Main idea: Generate (a small) number of new variables that are (linear) combinations of the original ones, and capture most of the information in the original data

Why do dimensionality reduction?

- **Computational** and statistical reasons: with thousands of features, it is computationally very hard to estimate a good model
- **Managerial** reason: the new variables may be interpretable and actionable
- **Visualizations**:
  - Visual cognition is very powerful but limited to ~3 dimensions
  - With 100s of variables, no obvious way to decide which 3 to plot …
  - But if, e.g., 3 such variables capture ~90% of information in the data then visually inspecting them may be very insightful

Dimensionality Reduction: Applications

MBA admissions

Data:

1. GPA
2. GMAT score
3. Scholarships, fellowships won
4. Communications skills
5. Prior job experience
6. Extra curricular achievements

How to capture all this with 2-3 variables?

Market response

("Boats" case assignment)

Data:

1. Questionnaire responses (numeric): 29 attitude questions, 18 "other" questions
2. Categorical data ("own a boat: Yes/No")

How to capture all this with just a few variables?

Product segmentation.

Pokemon data:

1. Multiple numerical characteristics ("Speed")
2. Multiple categorical characteristics ("Legendary")

How to capture all this with just a few variables?

Dimensionality Reduction: Approach and Key Questions

Dimensionality reduction is sometimes called “factor analyses” [not to confuse with “factors” in R ☹]

The derived attributes/variables are then called “factors” and the weights of the original variables are called “factor loadings”

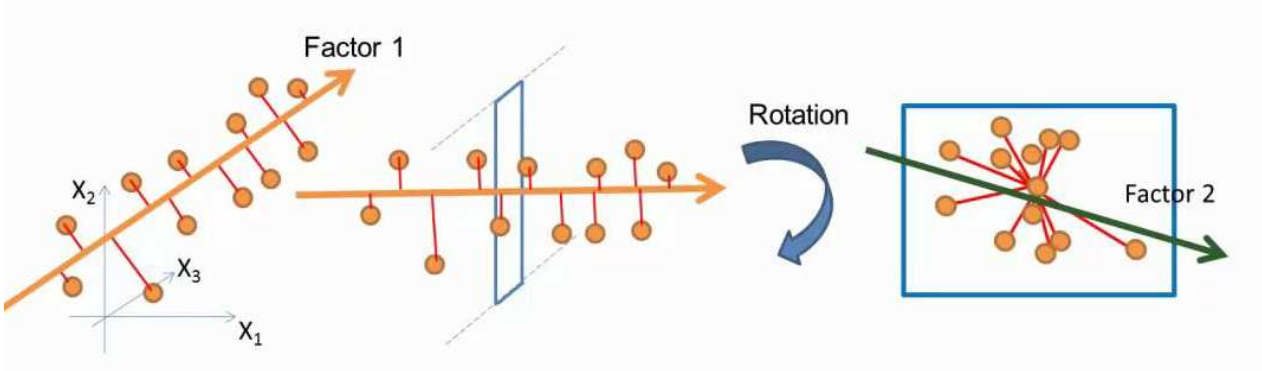
1. How many factors (“new variables”) do we need?
2. How would you name the factors (“new variables”) ? What do they mean?
3. How interpretable and actionable are the factors (“new variables”) we found?

One (popular) approach to construct derived attributes is called “Principal Component Analyses”, PCA

1. How many principal components do we need?
2. How would you name the principal components ? What do they mean?
3. How interpretable and actionable are the principal components we found?

How Does it Work?

Main idea: principal components represent the new set of coordinates, such that there is less variability in the data with respect to these coordinates then with respect to the original ones:
PC = w1*X1+w2*x2+w3*x3+…



By design:

PC1 will explain most of the variance in the data that any one (linear) coordinate / "transformation" can. It will be "most parallel" to the data

PC2 will explain less, PC3 will explain even less, etc.

Now lets practice!
PCA on Pokemon (num) data


```
pr.out<-prcomp(Pokemon.numerical.data, scale=TRUE, center=TRUE) # Perform "base-case" PCA
summary(pr.out) # Inspect model output
pr.out$rotation # Inspect PCA factor loadings
```

```
> # Inspect model output
> summary(pr.out)
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.6850 1.1406 0.8828 0.7760 0.53632 0.43665
Proportion of Variance 0.4732 0.2168 0.1299 0.1003 0.04794 0.03178
Cumulative Proportion  0.4732 0.6900 0.8199 0.9203 0.96822 1.00000
> pr.out$rotation
                PC1        PC2         PC3         PC4        PC5         PC6
HP       0.4043664  0.3746280 -0.04800283  0.71394743 -0.3107375 -0.29590317
Attack   0.4171666  0.2329212  0.69461546 -0.09977707 -0.0158832  0.52822475
Defense  0.3949229  0.4258120 -0.14162973 -0.66647368 -0.0620183 -0.44116623
Sp..Atk  0.4376205 -0.2999302 -0.55691556 -0.09981442 -0.4002151  0.48809859
Sp..Def  0.5028835 -0.2229758 -0.14278841  0.14953654  0.8069769 -0.05855391
Speed    0.2491833 -0.6959953  0.40571251 -0.06174424 -0.2965801 -0.44398591
```

Math Interpretation:

PC1 =
0.404*HP +
0.417*Attack +
0.3949*Defense + …

Code: 0910 R code – pokemon.R

Now lets practice!
PCA on Pokemon (num) data


```
pr.out<-prcomp(Pokemon.numerical.data, scale=TRUE, center=TRUE) # Perform "base-case" PCA
summary(pr.out) # Inspect model output
pr.out$rotation # Inspect PCA factor loadings
```

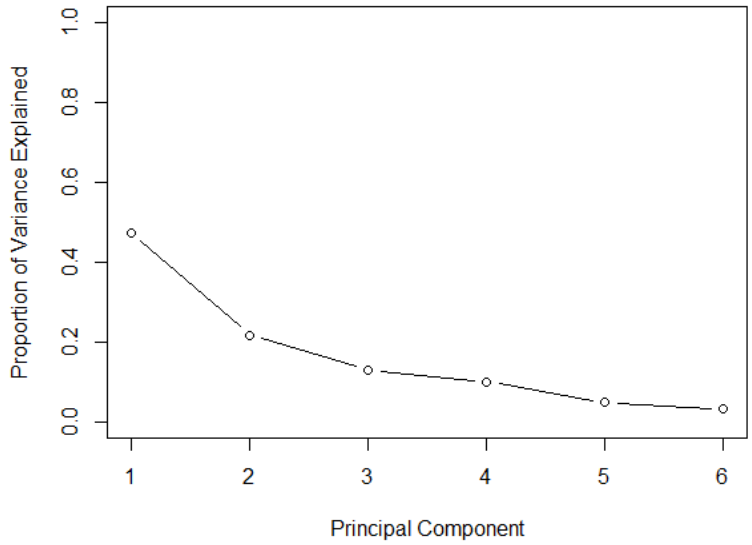
```
> # Inspect model output
> summary(pr.out)
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.6850 1.1406 0.8828 0.7760 0.53632 0.43665
Proportion of Variance 0.4732 0.2168 0.1299 0.1003 0.04794 0.03178
Cumulative Proportion  0.4732 0.6900 0.8199 0.9203 0.96822 1.00000
> pr.out$rotation
                PC1        PC2        PC3        PC4        PC5        PC6
HP        0.4043664  0.3746280         NA  0.7139474 -0.3107375         NA
Attack    0.4171666         NA  0.6946155         NA         NA  0.5282247
Defense   0.3949229  0.4258120         NA -0.6664737         NA -0.4411662
Sp..Atk   0.4376205         NA -0.5569156         NA -0.4002151  0.4880986
Sp..Def   0.5028835         NA         NA         NA  0.8069769         NA
Speed           NA -0.6959953  0.4057125         NA         NA -0.4439859
```

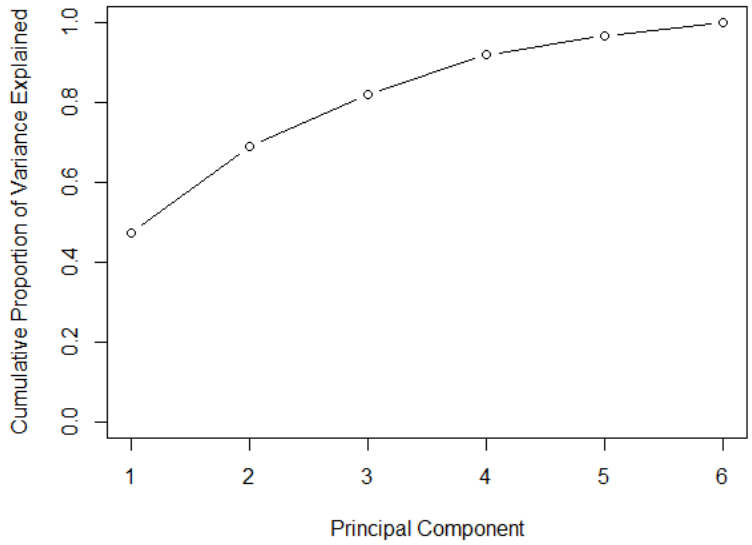
"Business" interpretation:	Total of all but speed	Slow speed + defence & HP	Strong attack and speed but slow Special attack	…

Code: 0910 R code – pokemon.R

PCA: understanding percent of variance explained

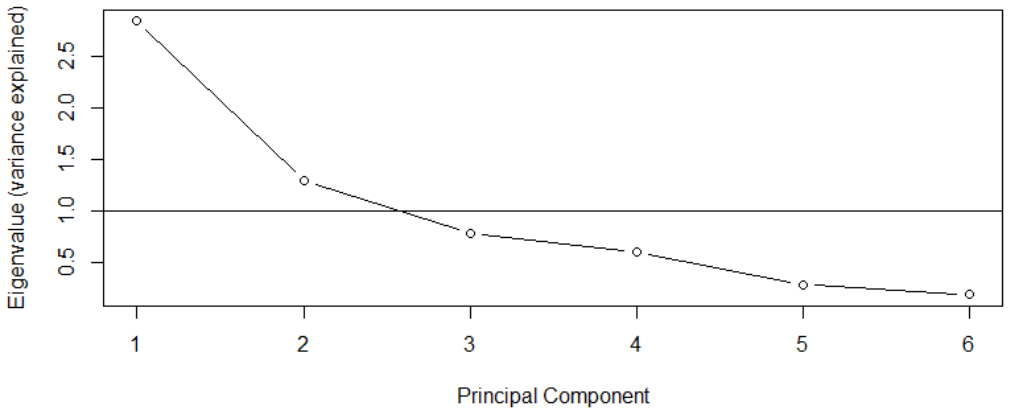

```
# Plot variance explained for each principal component
pr.var <- pr.out$sdev^2 #calculate variance explained
pve <- pr.var / sum(pr.var) #calculate % variance explained
par(mfrow = c(1, 2))
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim
= c(0, 1), type = "b")
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of
Variance Explained", ylim = c(0, 1), type = "b")
```





PCA: understanding how many components to use

```
# Plot variance explained, "eigenvalues"
pr.var <- pr.out$sdev^2
plot(pr.var, xlab = "Principal Component", ylab = "Eigenvalue (variance explained)",
type = "b")
abline(1,0)
```



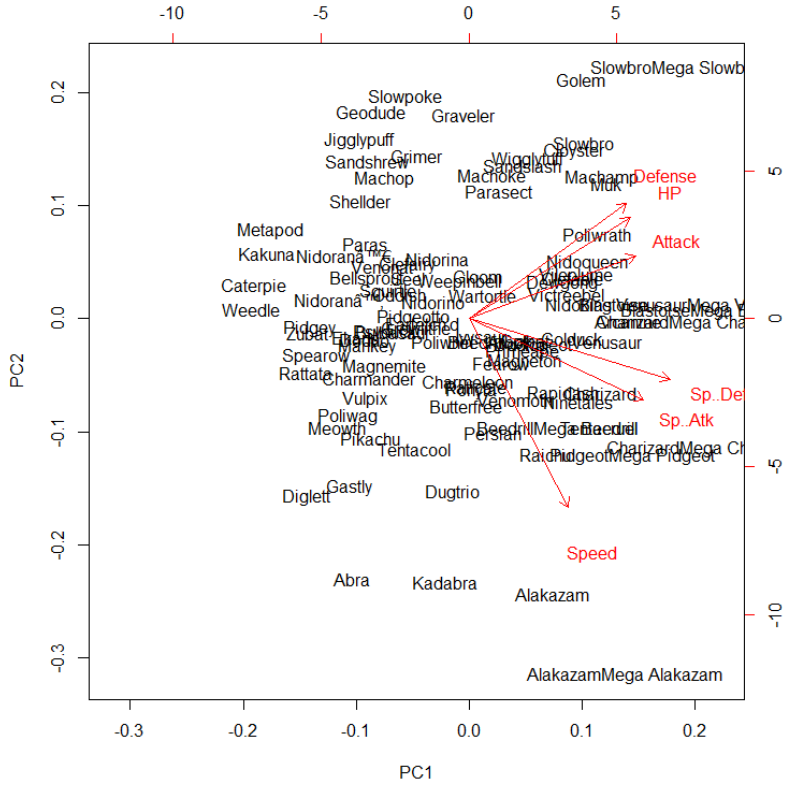
Rules of thumb:

1. Use PCs with eigenvalues > 1
2. Use enough PCs to have a cumulative % of variance explained > some desired threshold (must be >50%, ideally, ~80+)
3. Ideally we want to see an "elbow" (same logic as in clustering)

Side note: if the goal of PCA is feature engineering, then select the number of PCs through cross-validation ("principal components regression" PCR)

PCA: understanding components – biplots

```
par(mfrow = c(1, 1))
biplot(pr.out) # Biplot of the first two components
```

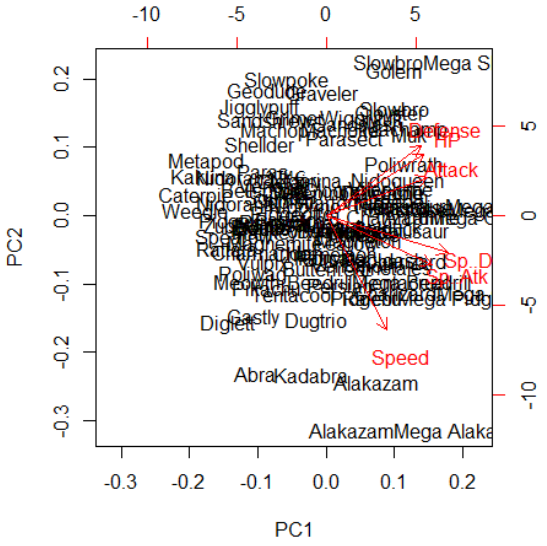


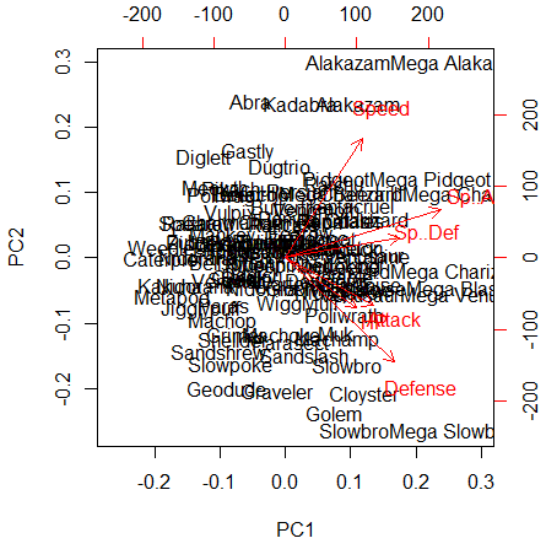
Which two pokemons are most different wrp PC2?

PCA: impact of scaling

```
pr.with.scaling<-prcomp(Pokemon.numerical.data, scale=TRUE)
pr.without.scaling<-prcomp(Pokemon.numerical.data, scale=FALSE)
par(mfrow=c(1,2)) # Create biplots of both for comparison
biplot(pr.with.scaling)
biplot(pr.without.scaling)
```

Rule of thumb: always use scaling





Our data is not terribly off-scale, so results are not that different [but try adding "Total" …c(5:11) and you will see big difference]

Additional considerations
Categorical variables

[idea before] **Main idea**: how to calculate the distance between:

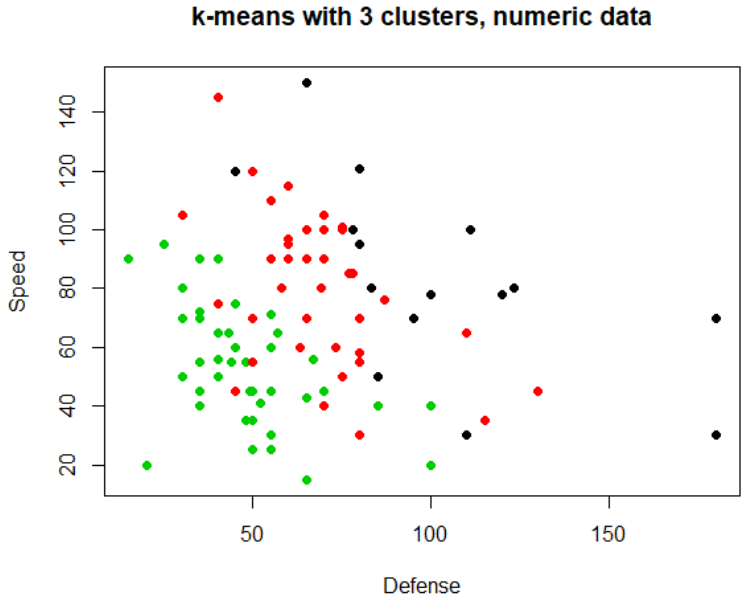
- Speed=55, Legendary=TRUE and Speed=45, Legendary=FALSE?

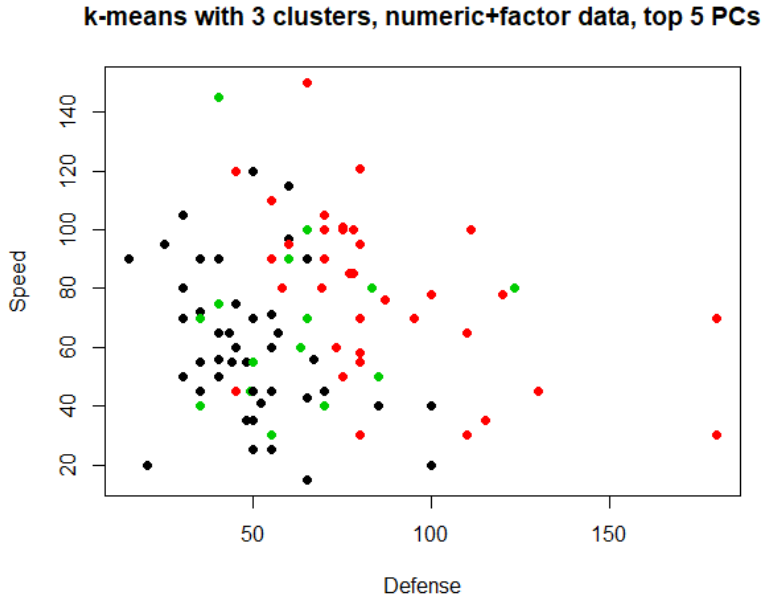
Main idea: this cannot be done in the original coordinates, i.e., where Legendary (a categorical variable) is one of the dimensions, but it could be done in new coordinates (with new / "derived" variables/attributes)

1. Pre-process data (redefine categories, fix missing, combine rare)
2. Convert categorical to dummies ("one-hot encoding")
3. Perform PCA
4. Cluster / analyze with PCA variables as opposed to the original ones

```
Pokemon.data<-combinerarecategories(Pokemon.data,5)
Pokemon.factors.onehot.encoding<-model.matrix(X. ~ Type.1 + Type.2,
data=Pokemon.data)[,-1] #+ Generation + Legendary
Pokemon.data.matrix<-cbind(Pokemon.numerical.data,Pokemon.factors.onehot.encoding)
pr.with.scaling.PCA<-prcomp(Pokemon.data.matrix, scale=TRUE, center=TRUE))
```

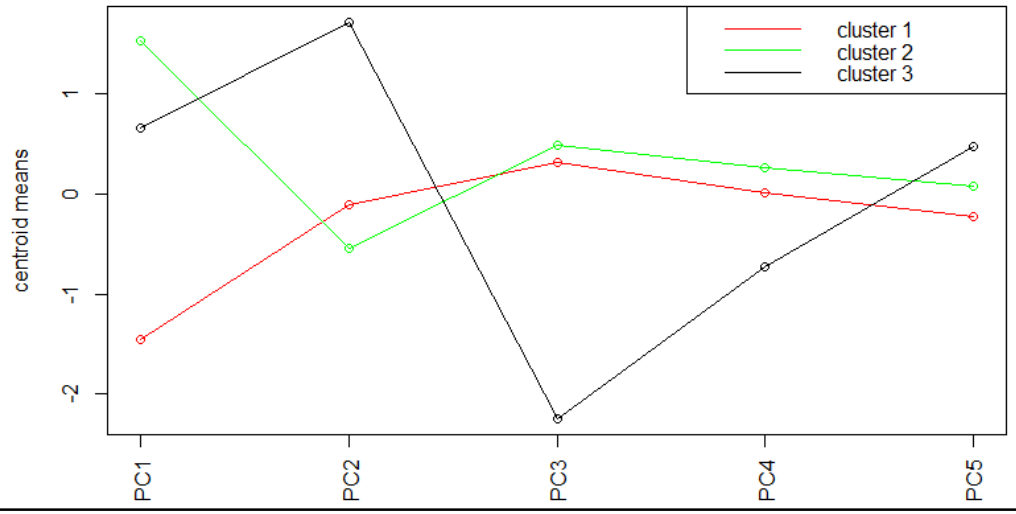
Results:
K-Means with Categorical variables (top 5 PCs)





Results: K-Means with top 5 PCs

```
> PCA.factor.loadings<-pr.with.scaling.PCA$rotation[,c(1:5)]
> PCA.factor.loadings[abs(PCA.factor.loadings)<0.3]<-NA
> PCA.factor.loadings
                          PC1        PC2        PC3        PC4        PC5
HP                  0.3883247         NA         NA         NA         NA
Attack              0.3765655         NA         NA         NA  0.3947831
Defense             0.4190806         NA         NA         NA         NA
Sp..Atk             0.4104906         NA         NA         NA         NA
Sp..Def             0.4660470         NA         NA         NA         NA
Speed                      NA -0.4147289 -0.3188637         NA         NA
Type.1Other.Type.1         NA         NA         NA  0.6210843         NA
Type.1Fighting             NA         NA         NA         NA         NA
Type.1Fire                 NA         NA         NA         NA         NA
Type.1Grass                NA  0.3613923 -0.4349960         NA         NA
Type.1Normal               NA -0.4163733         NA -0.3616799         NA
Type.1Poison               NA         NA         NA         NA  0.4337200
Type.1Water                NA         NA         NA         NA -0.3729508
Type.2Other.Type.2         NA         NA  0.3582637         NA -0.3224669
Type.2Flying               NA -0.4800873         NA         NA         NA
Type.2Ground               NA         NA         NA  0.3455778  0.3943569
Type.2Poison               NA  0.3847931 -0.4721108         NA         NA
```



Summary: Process for Dimensionality Reduction / PCA

1. Clean the data (properly define types, missing values, combine rare)
2. Choose number of factors (components)
3. Run PCA (with scaling of the numerical the data)
4. Interpret the factor loadings
5. Save the resultant PC values (if desired to be used as features in further analyses)

[Optional/Time-permitting]
Association Rules

Software

The parable of the beer and diapers

Never let the facts get in the way of a good story

By Mark Whitehorn 15 Aug 2006 at 13:20 5 SHAI

Association Rules

What is it? A prevalence of joint appearance of two (or more) phenomena in the data: **Rule: {event X} → {event Y}**

- People who bought {Diapers, Milk} tend to buy {Beer}
- CNN.com → BMW.com, etc.

Business applications:

- Put X and Y close together (in a store, on display, on webpage)
- Package X with Y; recommend X for Y, Y for X
- Package X and Y with a poorly selling item
- Give discount on only X *or* Y
- Increase the price of X *and* lower the price of Y
- Advertise only X *or* Y
- If X is a toy and Y is a candy, then offer a candy in the shape/form of toy X
- …

Example Transaction Data

Code: 0910 R code – association rules.R

```
#install.packages("arules")
library(arules)
data("Groceries")
inspect(Groceries[1:10,]) #look at the data
```

```
> inspect(Groceries[1:10,]) #look at the data
     items
[1]  {citrus fruit,semi-finished bread,margarine,ready soups}
[2]  {tropical fruit,yogurt,coffee}
[3]  {whole milk}
[4]  {pip fruit,yogurt,cream cheese ,meat spreads}
[5]  {other vegetables,whole milk,condensed milk,long life bakery product}
[6]  {whole milk,butter,yogurt,rice,abrasive cleaner}
[7]  {rolls/buns}
[8]  {other vegetables,UHT-milk,rolls/buns,bottled beer,liquor (appetizer)}
[9]  {pot plants}
[10] {whole milk,cereals}
```

Note:

- This is a VERY different format of data, not like a dataframe at all... [benefits?]
- Such a format is called JSON (JavaScript Object Notation)

Example Rules and Measures

LHS	→	RHS	Support	Confidence	Lift
{Canned Beer}	→	{Milk}	5%	20%	1.0
{Canned Beer}	→	{Berries}	0.1%	1%	0.3
{Canned Beer}	→	{Chips}	3%	79%	1.5
{Sausage}	→	{Mustard}	2%	85%	1.9
{Sausage}	→	{Ketchup}	2%	55%	1.4
{Sausage}	→	{Milk}	8%	30%	0.8
{Sausage, Chips}	→	{Canned Beer}	1%	90%	1.6

Association Rules Measures

Support:

- $S(X \rightarrow Y) = \frac{\#\text{ transactions with X and Y}}{\#\text{ transactions}}$, percentage of transactions containing X and Y

Confidence:

- $C(X \rightarrow Y) = \frac{S(X \,\&\, Y)}{S(X)}$, how frequently Y appears, in transactions that contain X

Lift:

- $L(X \rightarrow Y) = \frac{C(X \rightarrow Y)}{S(Y)}$, how much more often Y appears with X than just by random chance

Many others: conviction [1-S(Y)]/[1- C(X → Y)], addedValue, chiSquared, certainty, collectiveStrength, confidence,, cosine, coverage, confirmedConfidence, varyingLiason, yuleQ, yuleY …)

Example Measures

You usually want to find rules that are high in all three.

LHS		RHS	Support	Confidence	Lift
{Canned Beer}	→	{Milk}	5%	20%	1.0
{Canned Beer}	→	{Berries}	0.1%	1%	0.3
{Canned Beer}	→	{Chips}	0.3%	79%	1.5
{Sausage}	→	{Mustard}	3%	85%	1.9
{Sausage}	→	{Ketchup}	2%	55%	1.1
{Sausage}	→	{Milk}	1%	30%	0.8
{Sausage, Chips}	→	{Canned Beer}	1%	90%	1.6

Support *vs* Confidence *vs* Lift

Consider the rule {**Sausage**} → {**Mustard**}

Measure	**Meaning**	**Low Value Means…**	**High Value Means…**
Support [0, 1]	How often Sausage and Mustard appear together	Sausage and Mustard rarely appear together	Sausage and Mustard appear together often
Confidence [0, 1]	How often Mustard appears, when Sausage appears	when Sausage appears, Mustard is unlikely to appear	when Sausage appears, Mustard is very likely to appear as well
Lift [0, ∞)	Whether Mustard appears with Sausage more often than random chance	Sausage and Mustard appear together less often than random chance (negatively correlated)	Sausage and Mustard appear together more often than random chance (positively correlated)

Association Rules
How does this work?

Given:

- A database of transactions
- Minimum support s
- Minimum confidence c (optional)

Goal: find all association rules **X → Y** with a minimum support s and confidence c

General approach:

- Find all frequent itemsets
- Use the frequent itemsets to generate the desired rules

Apriori algorithm: Makes many passes over database, each time collecting frequent 2-itemsets, 3-itemsets, 4-itemsets, etc.

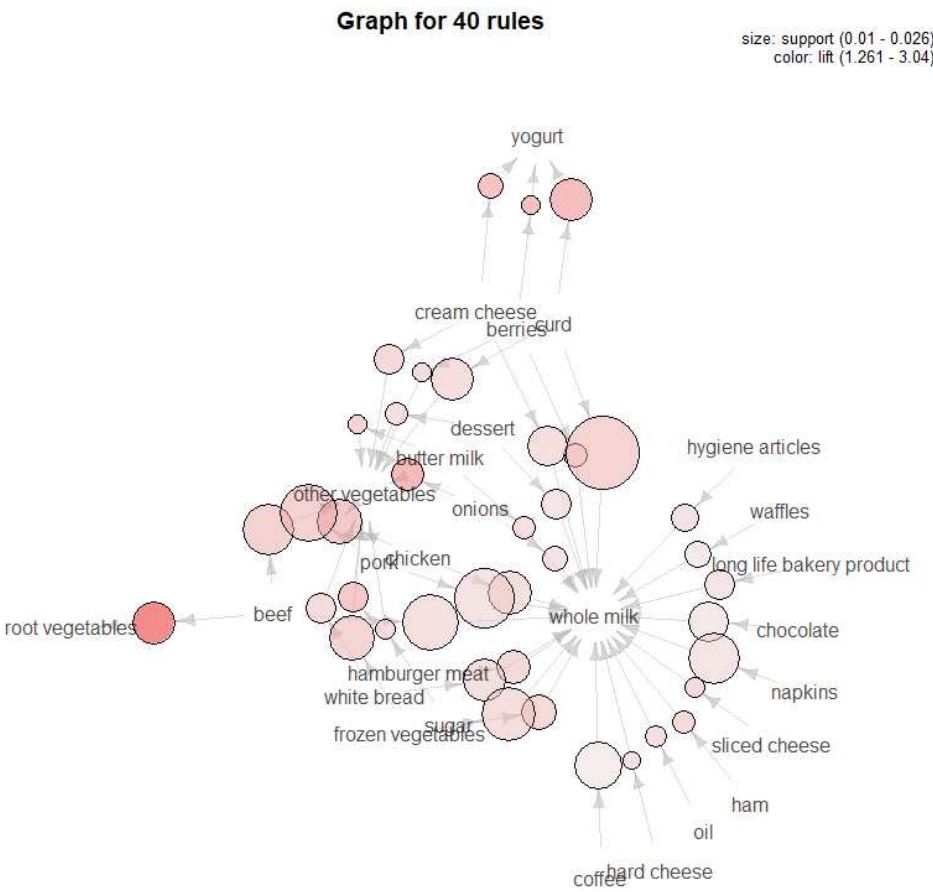
- Uses the Apriori principle: if {A, B} is frequent, then {A} and {B} must both be frequent. Equivalently, if {A} is not frequent, then {A, B} cannot be frequent.

Implementation in R: package `arules`


```
rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.3, target =
"rules"))
inspect(head(rules,40, by = "lift")) #lets see which rules were identified
quality(rules) #table with quality measures for the identified rules
```

```
> inspect(head(rules,40, by = "lift")) #lets see which rules were identified
     lhs                                rhs                  support    confidence lift     count
[1]  {citrus fruit,other vegetables} => {root vegetables}   0.01037112 0.3591549  3.295045 102
[2]  {tropical fruit,other vegetables} => {root vegetables} 0.01230300 0.3427762  3.144780 121
[3]  {beef}                         => {root vegetables}   0.01738688 0.3313953  3.040367 171
[4]  {citrus fruit,root vegetables} => {other vegetables}  0.01037112 0.5862069  3.029608 102
[5]  {tropical fruit,root vegetables} => {other vegetables} 0.01230300 0.5845411 3.020999 121
[6]  {other vegetables,whole milk}  => {root vegetables}   0.02318251 0.3097826  2.842082 228
[7]  {whole milk,curd}              => {yogurt}            0.01006609 0.3852140  2.761356  99
```

Visualizing Association Rules package `arulesViz`




```
#install.packages("arulesViz")
library(arulesViz)
plot(rules[1:40,],
     method = "graph")
```

Useful for showing rules that share items

- Circles: rules
- Links: associations
- Size: support
- Color: lift

[Optional/Time-permitting]
Anomaly Detection

What is it? A technique used to identify unusual patterns ("outliers") that do not conform to expected behavior.

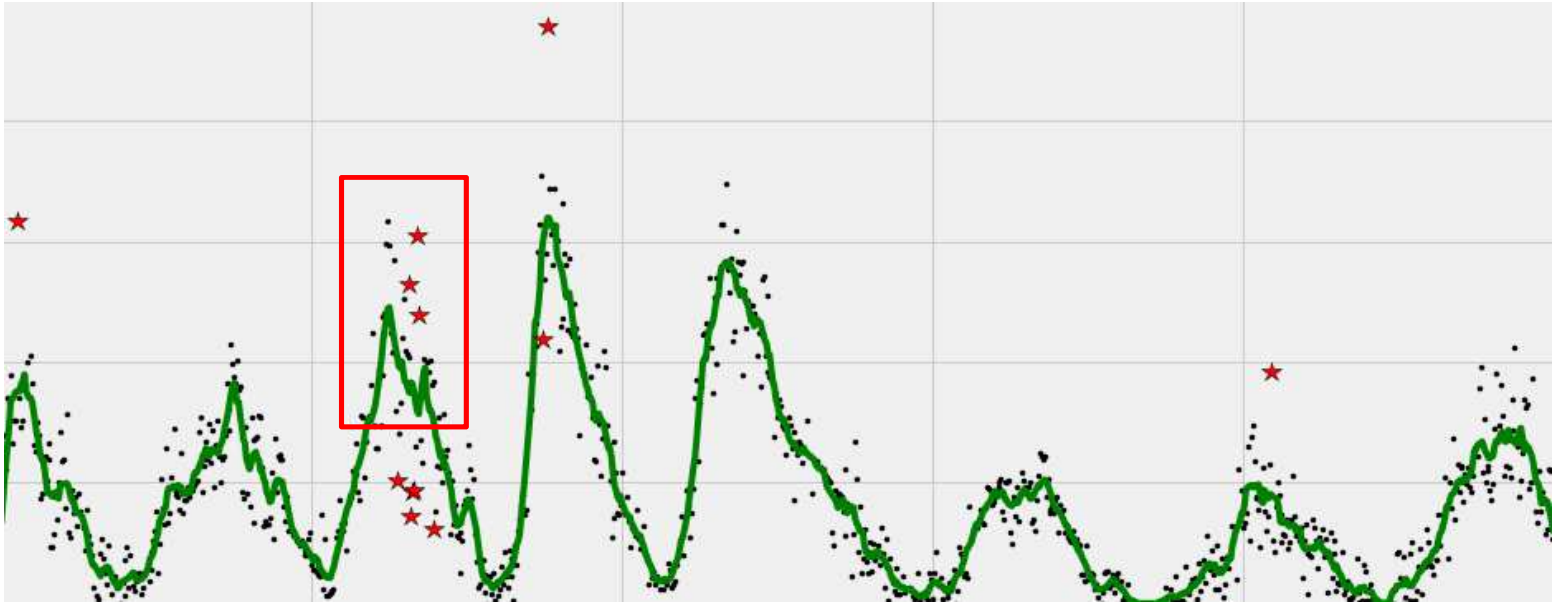
Business applications: [numerous]:

- Most common "raising red flags"
  - "sales in location A are low, is there a problem?"
  - "rep A has long "average hold time", does he/she needs a training?"
- Intrusion/adversary/non-human activity detection: is a strange pattern in network traffic a cyber-attack?
- System "health": [near/approaching] failure of a machine
- Fraud in credit card transactions or operations (robo-signing to open new accounts, unusually large/small/frequent payments to vendor X, etc.)

[Optional/Time-permitting]
Anomaly Detection

How it works: build a model to predict expected ("normal") behavior and its variance/standard deviation:

- IF[Distance(**new** datapoint(s) – to – **expected given other/known/past datapoints**) > Threshold * StDev, THEN **Anomaly**, OTHERWISE, all is fine]



[Optional/Time-permitting]
Anomaly Detection

Popular techniques:

- For time-series data:
  - “low-pass filter”: the expected value is calculated by an n-period moving average
  - Kalman filter: the expected value is calculated using a [very] complex dynamic “state space” model. `KFAS` package in r
- For unsupervised data:
  - K-means: the expected value is the centroid of the nearest cluster
- For supervised data:
  - Essentially any model that has “well understood” error structure; e.g., regression, logistic regression, regularizations (LASSO/Ridge/SVM). Tree based model are not used much (not clear what error distributions look like). ANN models are combined with PCA

Summary of Sessions 9-10

Our one and only glimpse at **unsupervised learning** (where there is no « Y » variable, only Xs). What can we do with such data?

Clustering and segmentation

- Methods: k-means, hierarchical
- Distance/similarity measures. Scaling, Elbow plots, Snake plots

Derived attributes and **dimensionality reduction** and PCA

- PCA creates a small number of new variables that capture most information
- Used as new features in supervised learning, for clustering (e.g., to include categorical variables), or for visualization
- Factor loadings, rotation, interpretation

Association Rules: What features appear in the data together

- Rules, Measures (Support, Confidence, Lift, etc.). Apriori algorithm/principle

Anomaly Detection: Is this an outlier?

- Build a model to predict « normal » behavior (ts moving average, nearest k-means centroid, fractile of residual distribution) and label as anomaly if > T SDs away

Next…

Proposal for final project, due TBA
- I will do my best to provide feedback ASAP

"Tutorial 3" – DataRobot soft demo

Sessions 11-12
- Guest speaker from BCG, nothing to do/pre-read (hurray!)

Assignment 3
- "Boats" case and data on portal, questions in the case, due by session 11-12

Sessions 13-14
- Final project presentations

The Business School
for the World®

Europe | Asia | Middle East