Lecture 7: Unsupervised Learning

Part I: Hierarchical Clustering, EM clustering
Part II: Dimensionality Reduction, Association rule mining

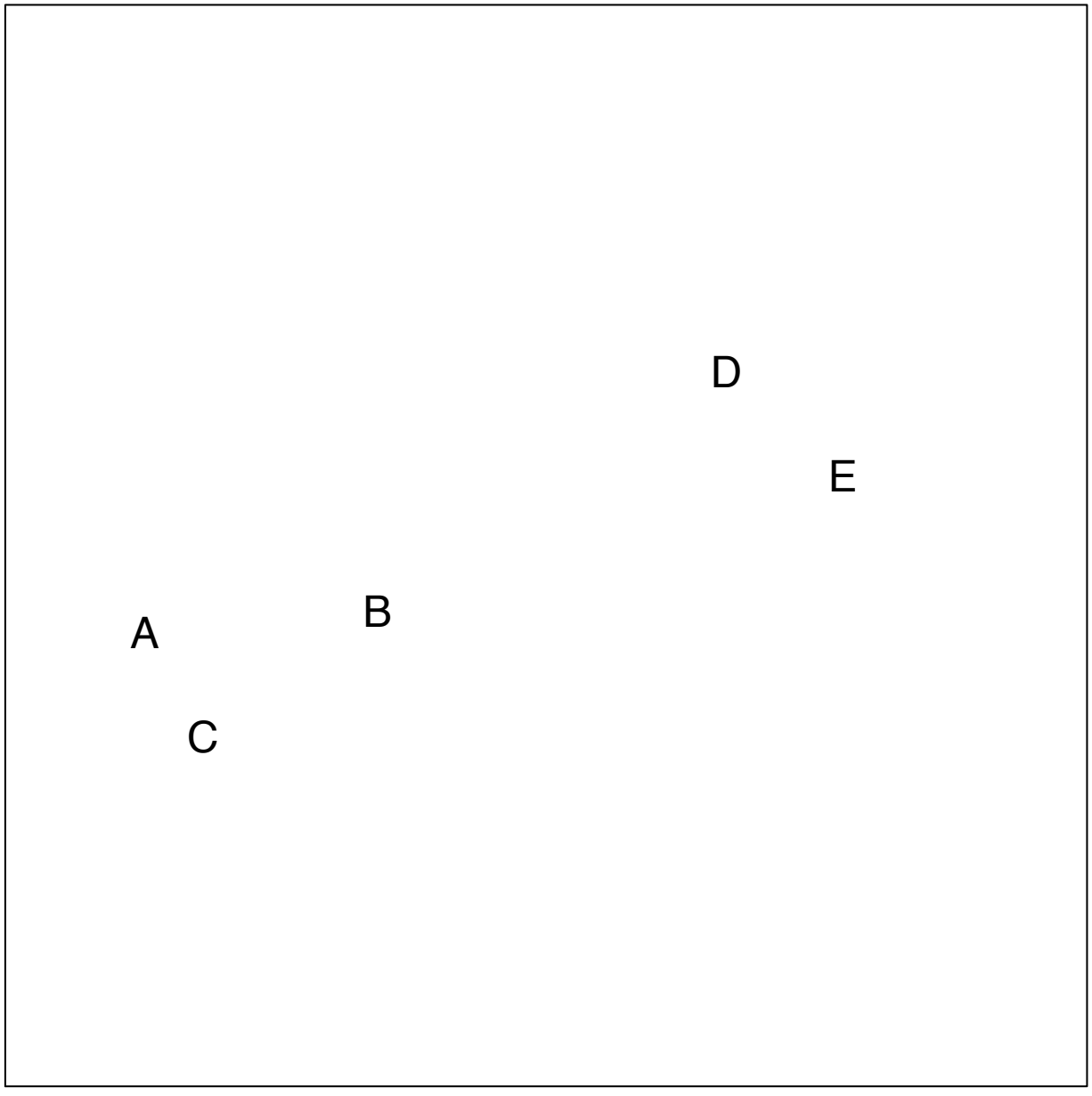
Prof. Alexandra Chouldechova
95-791: Data Mining

Agenda for Part I

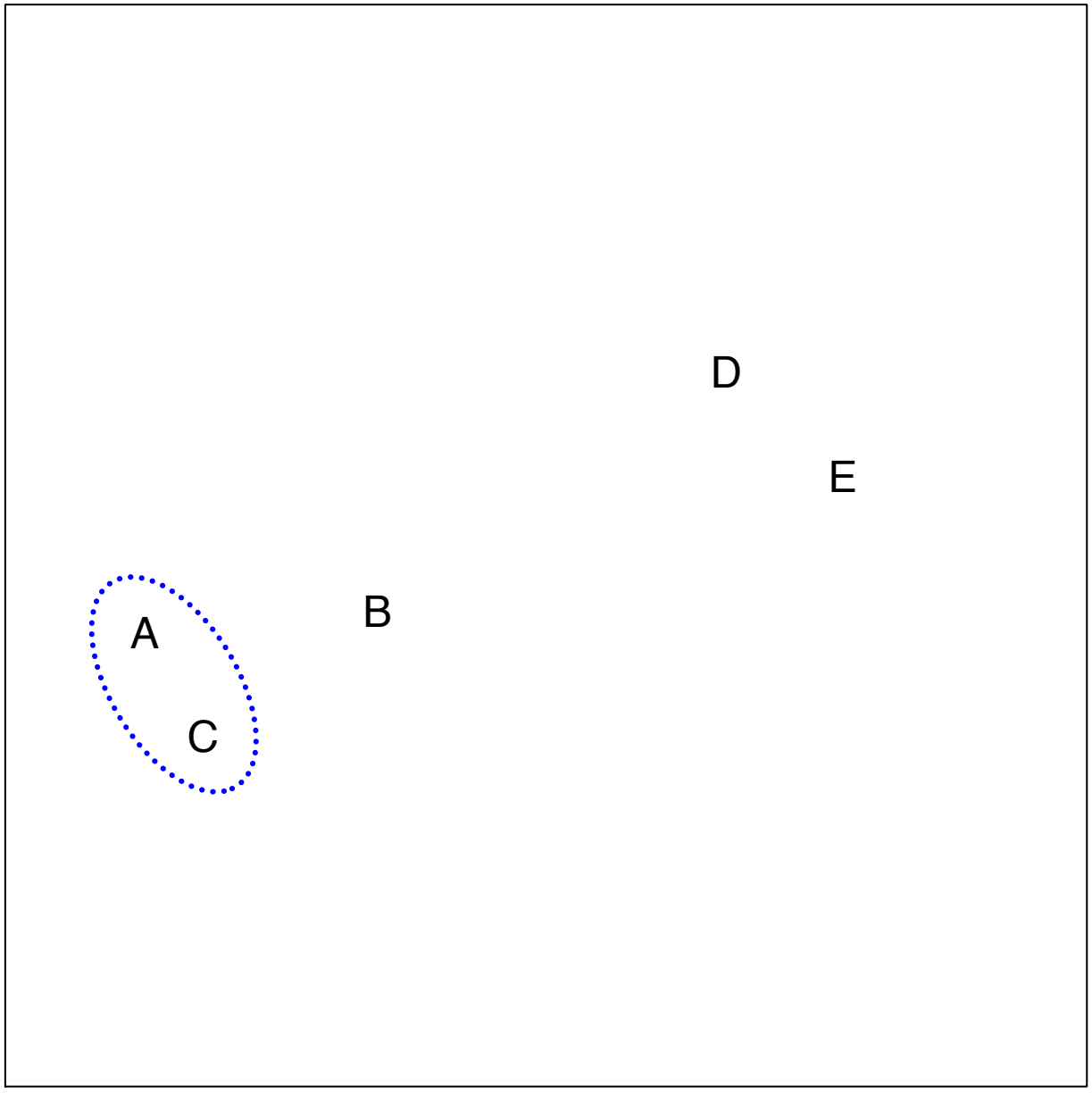
- **Hierarchical clustering**
- **Mixture models (EM clustering)**

Hierarchical clustering

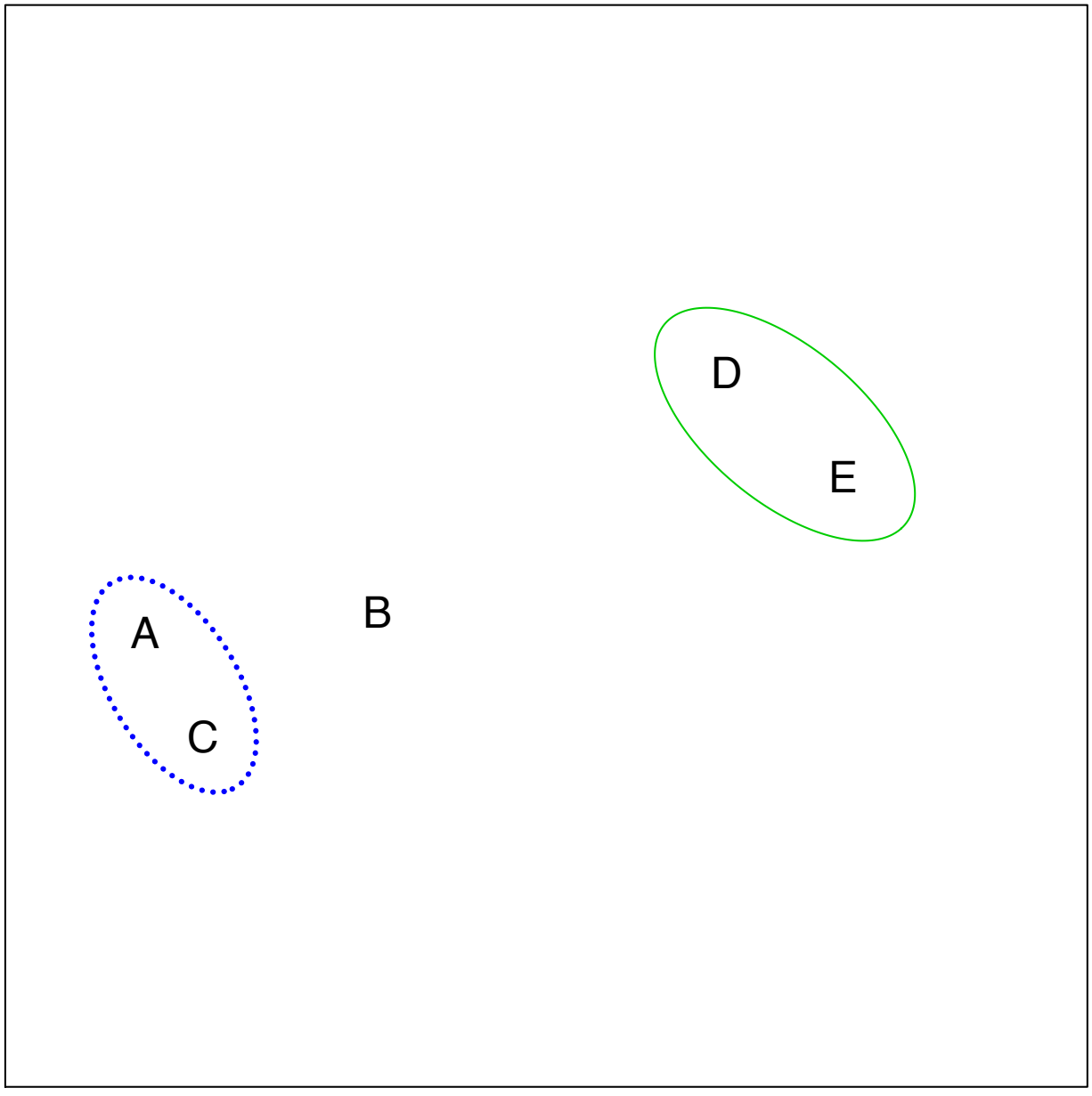
- K-means is an objective-based approach that requires us to pre-specify the number of clusters K
- The answer it gives is somewhat random: it depends on the random initialization we started with
- Hierarchical clustering is an alternative approach that does not require a pre-specified choice of K, and which provides a deterministic answer (no randomness)
- We'll focus on bottom-up or agglomerative hierarchical clustering
- top-down or divisive clustering is also good to know about, but we won't directly cover it here



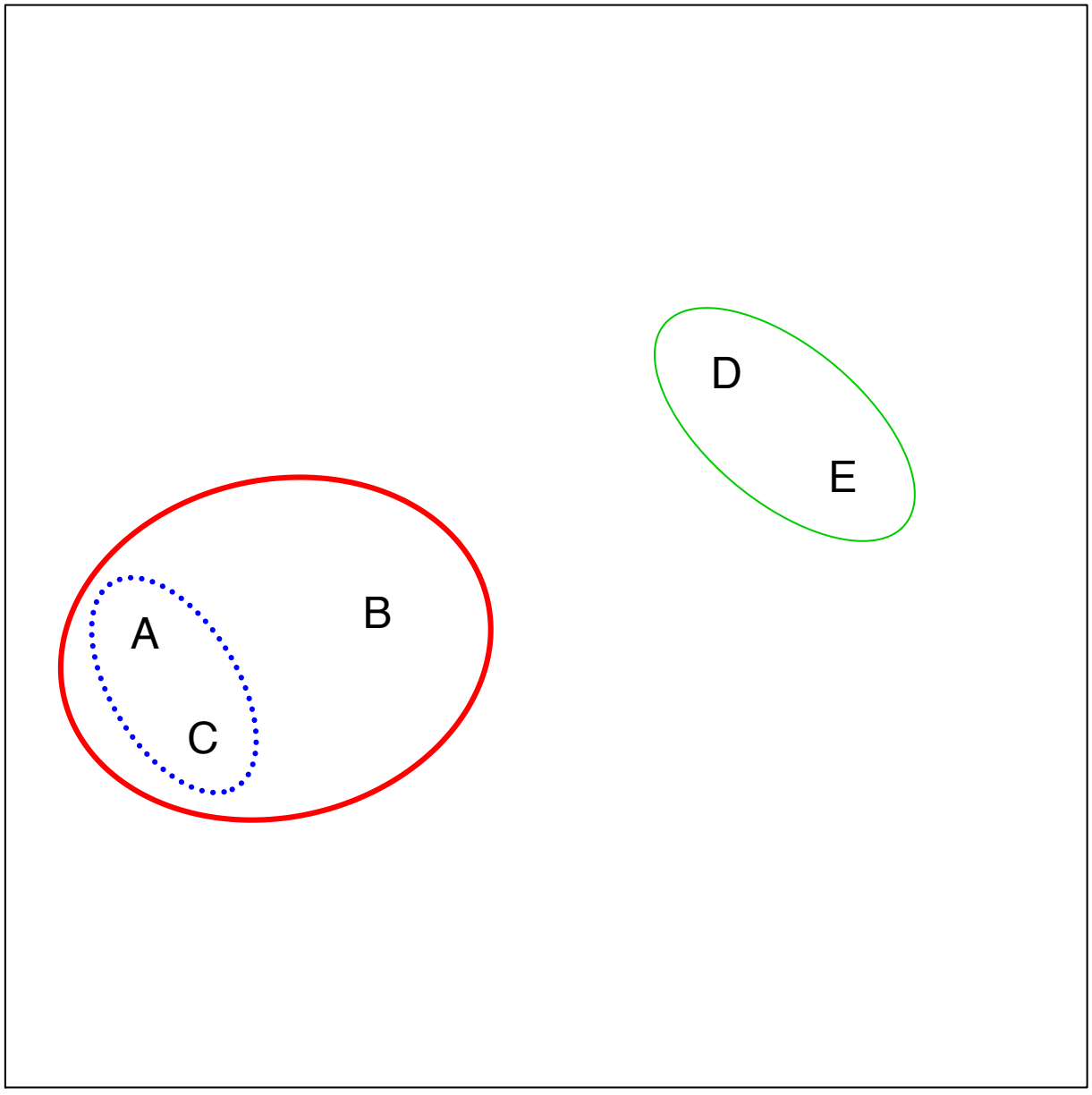
Each point starts as its own cluster



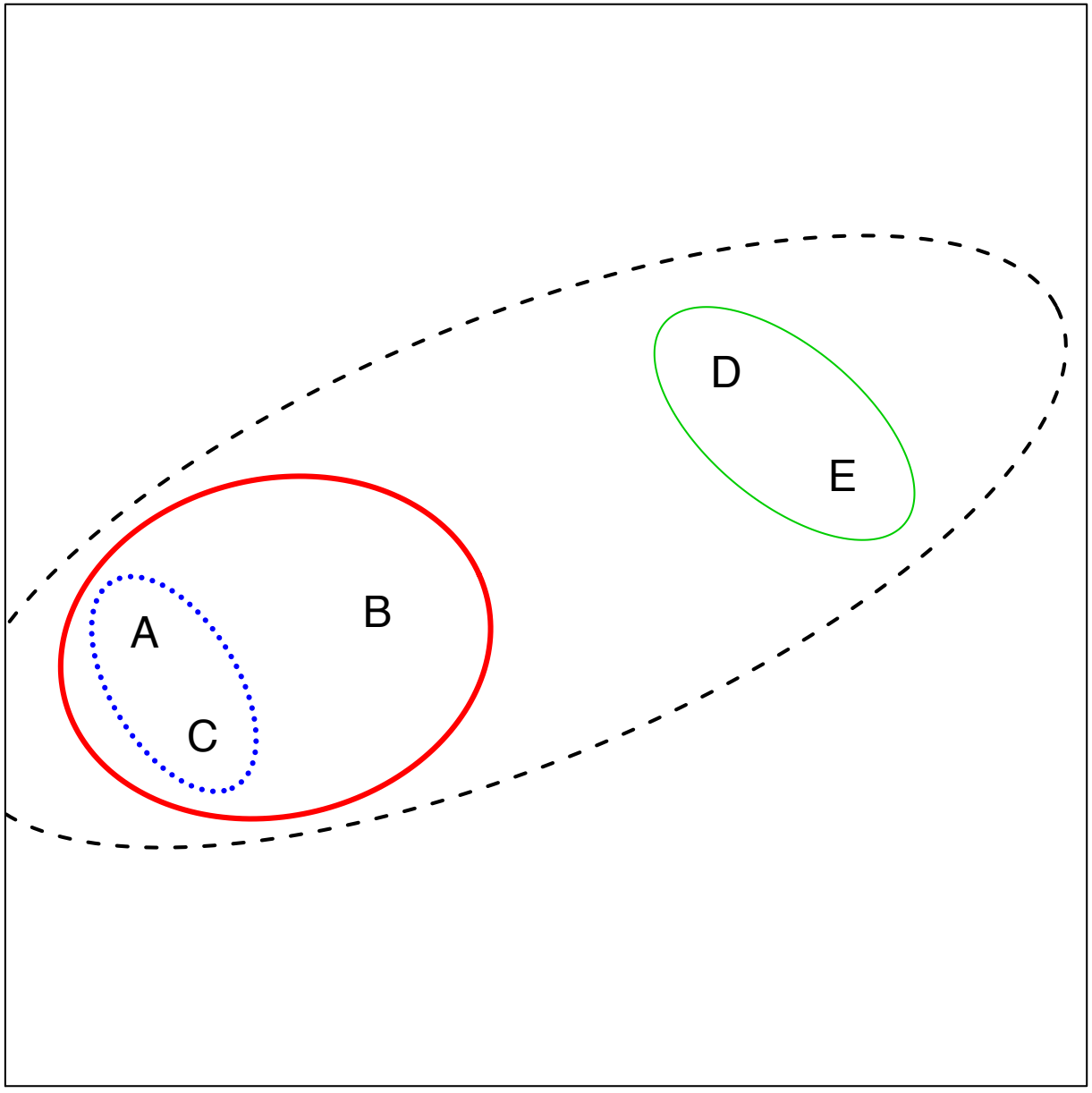
We merge the two clusters (points) that are closet to each other



Then we merge the next two closest clusters



Then the next two closest clusters…

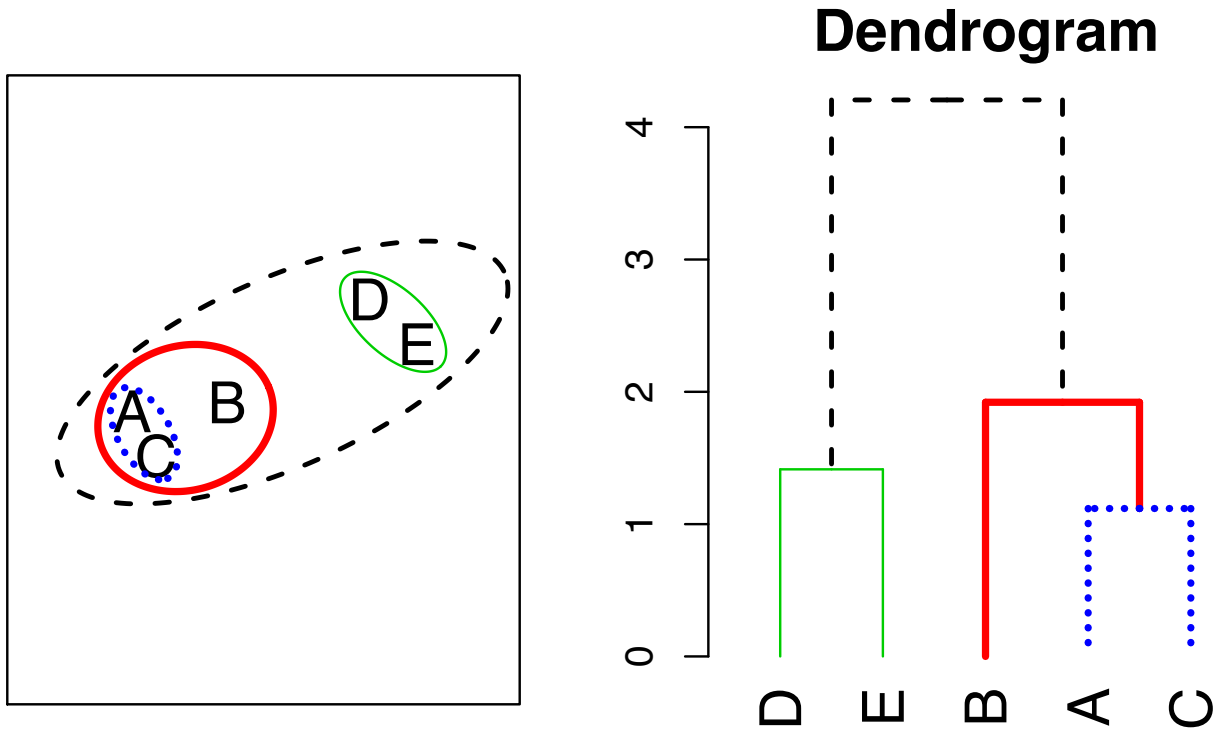


Until at last all of the points are all in a single cluster

Agglomerative Hierarchical Clustering

- Start with each point in its own cluster.
- Identify the two closest clusters. Merge them.
- Repeat until all points are in a single cluster

To visualize the results, we can look at the resulting dendrogram



y-axis on dendrogram is (proportional to) the distance between the clusters that got merged at that step

Linkages

- Let $d_{ij} = d(x_i, x_j)$ denote the dissimilarity[1] (distance) between observation x_i and x_j
- At our first step, each cluster is a single point, so we start by merging the two observations that have the *lowest dissimilarity*
- But after that…we need to think about distances not between points, but between sets (clusters)
- The dissimilarity between two clusters is called the linkage
- i.e., Given two sets of points, G and H, a linkage is a dissimilarity measure $d(G, H)$ telling us how different the points in these sets are
- Let's look at some examples

[1] We'll talk more about dissimilarities in a moment

Common linkage types

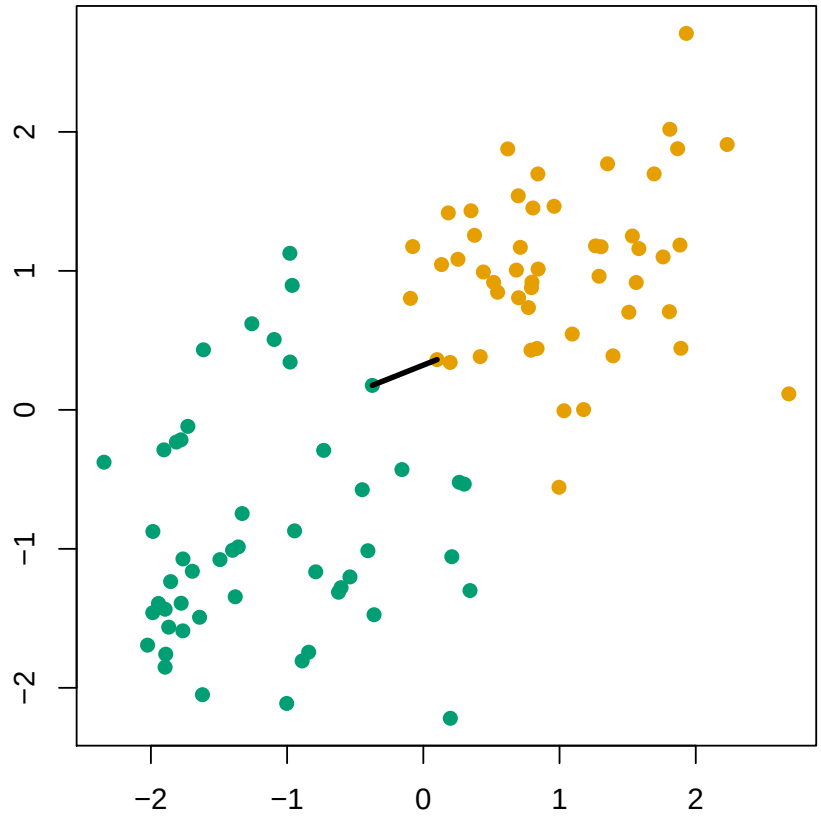
Linkage	*Description*
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Single linkage

In single linkage (i.e., nearest-neighbor linkage), the dissimilarity between G, H is the smallest dissimilarity between two points in different groups:

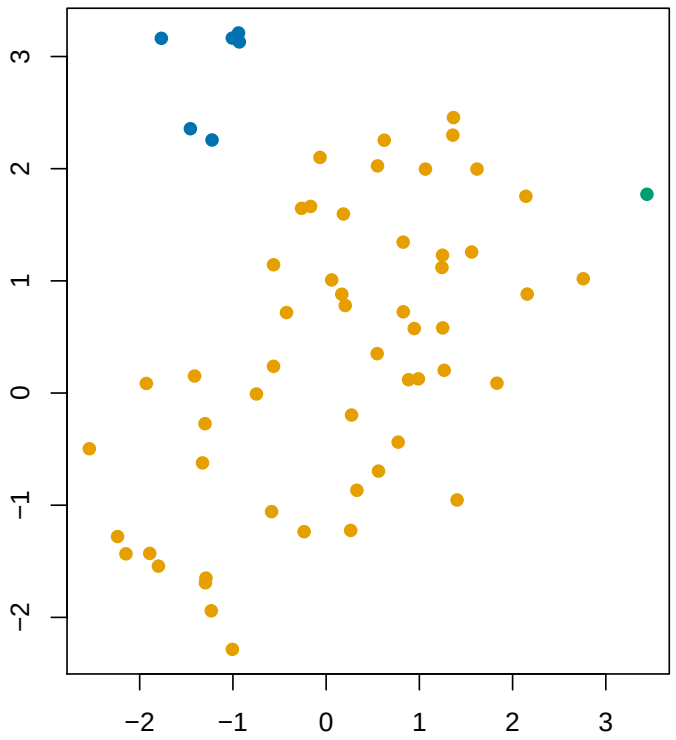
$$d_{\text{single}}(G, H) = \min_{i \in G,\, j \in H} d(x_i, x_j)$$

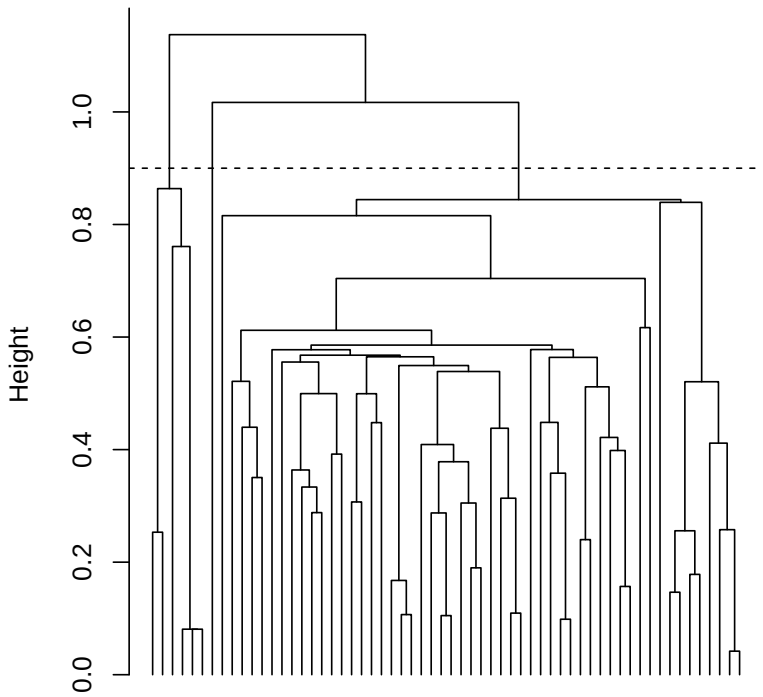
Example (dissimilarities d_{ij} are distances, groups are marked by colors): single linkage score $d_{\text{single}}(G, H)$ is the distance of the closest pair



Single linkage example

Here $n = 60$, $x_i \in \mathbb{R}^2$, $d_{ij} = \|x_i - x_j\|_2$. Cutting the tree at $h = 0.9$ gives the clustering assignments marked by colors





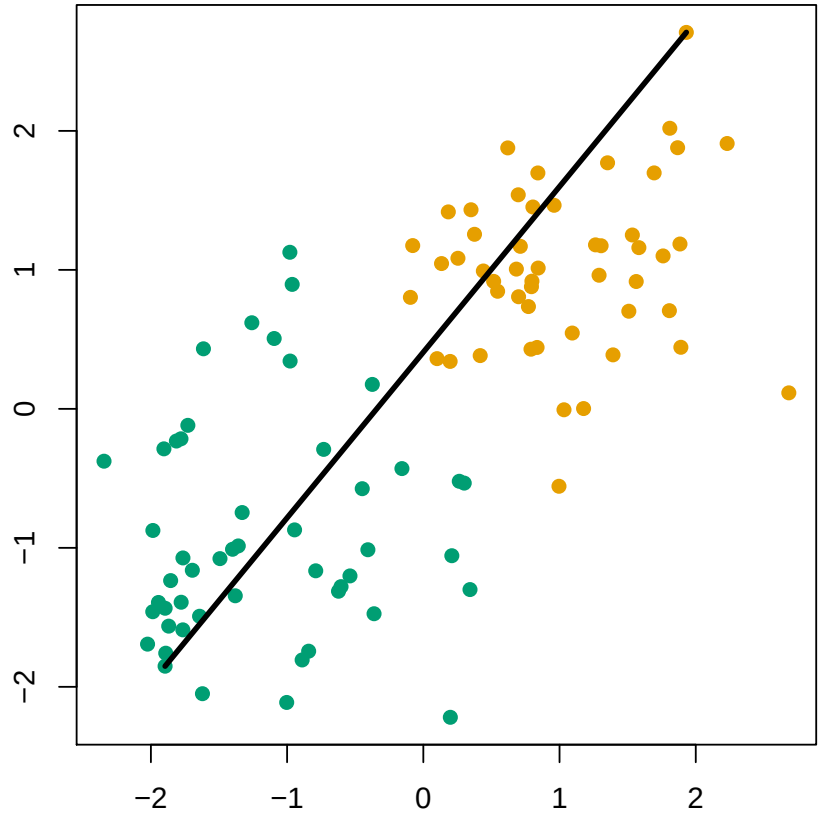
Cut interpretation: for each point x_i, there is another point x_j in its cluster such that $d(x_i, x_j) \leq 0.9$

Complete linkage

In complete linkage (i.e., furthest-neighbor linkage), dissimilarity between G, H is the largest dissimilarity between two points in different groups:

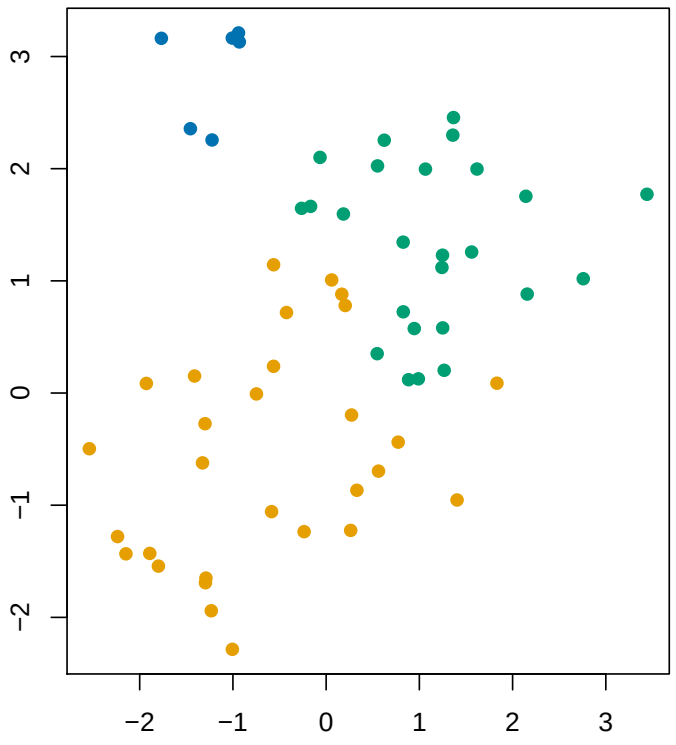
$$d_{\text{complete}}(G, H) = \max_{i \in G,\, j \in H} d(x_i, x_j)$$

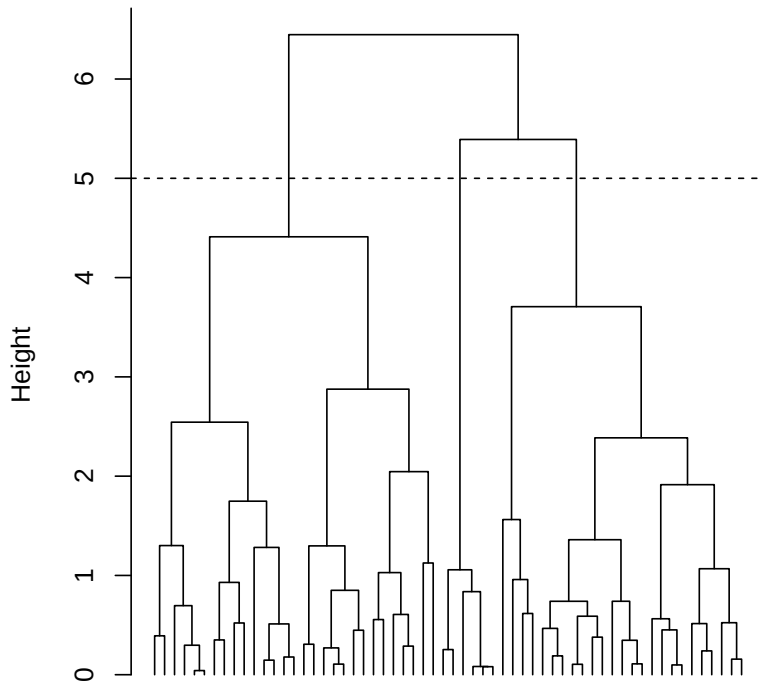
Example (dissimilarities d_{ij} are distances, groups are marked by colors): complete linkage score $d_{\text{complete}}(G, H)$ is the distance of the furthest pair



Complete linkage example

Same data as before. Cutting the tree at $h = 5$ gives the clustering assignments marked by colors





Cut interpretation: for each point x_i, every other point x_j in its cluster satisfies $d(x_i, x_j) \leq 5$

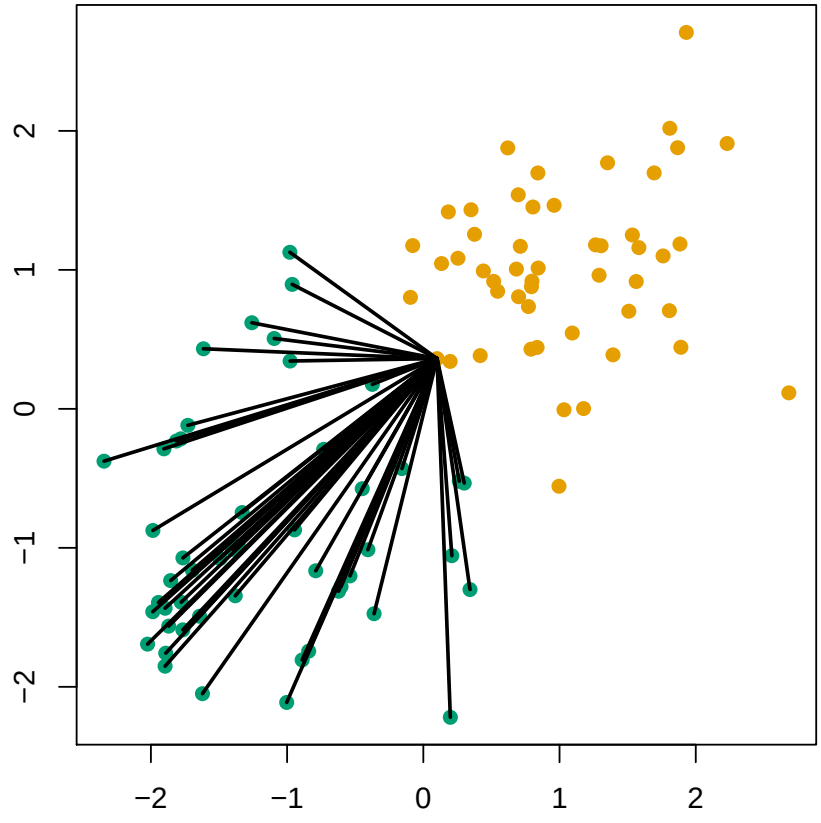
Average linkage

In average linkage, the dissimilarity between G, H is the average dissimilarity over all points in opposite groups:

$$d_{\text{average}}(G, H) = \frac{1}{|G| \cdot |H|} \sum_{i \in G,\, j \in H} d(x_i, x_j)$$

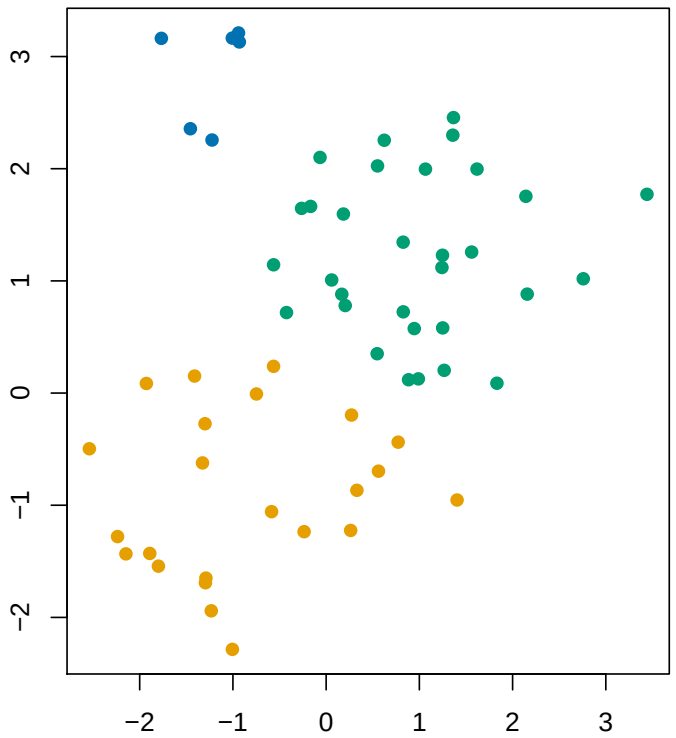
Example (dissimilarities d_{ij} are distances, groups are marked by colors): average linkage score $d_{\text{average}}(G, H)$ is the average distance across all pairs

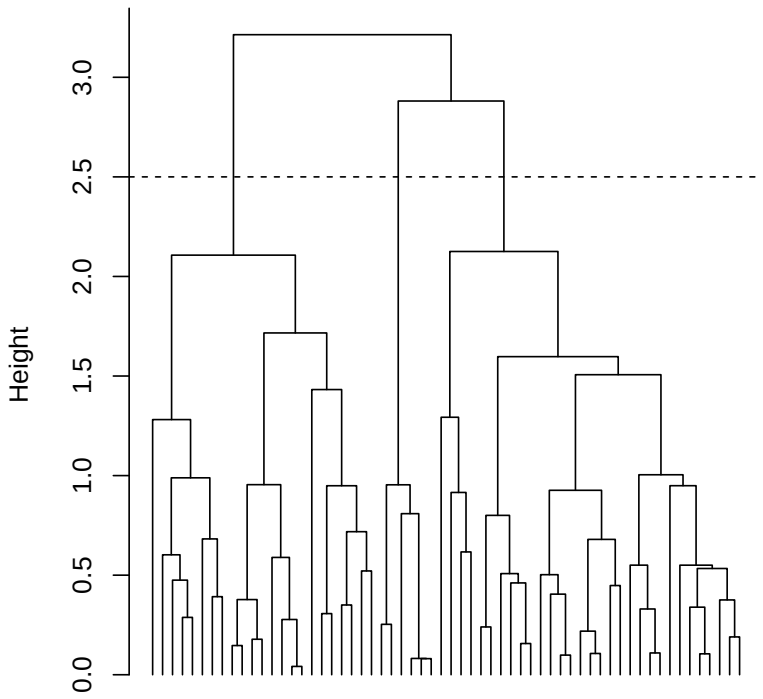
(Plot here only shows distances between the green points and one orange point)



Average linkage example

Same data as before. Cutting the tree at $h = 2.5$ gives clustering assignments marked by the colors





Cut interpretation: there really isn't a good one! ☹

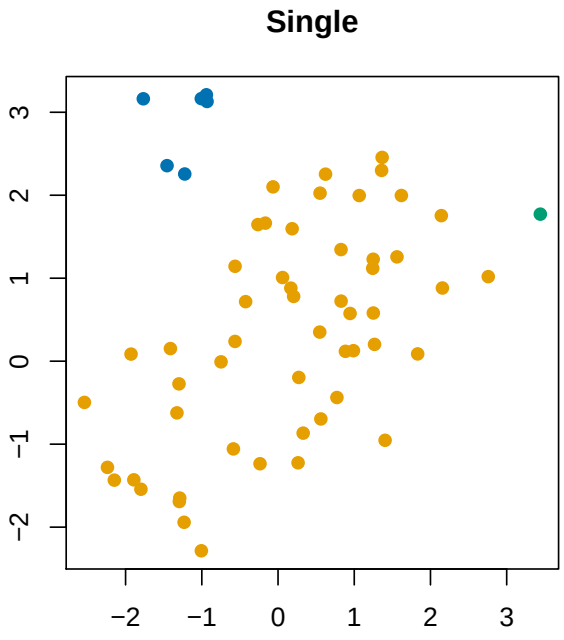
Shortcomings of Single and Complete linkage

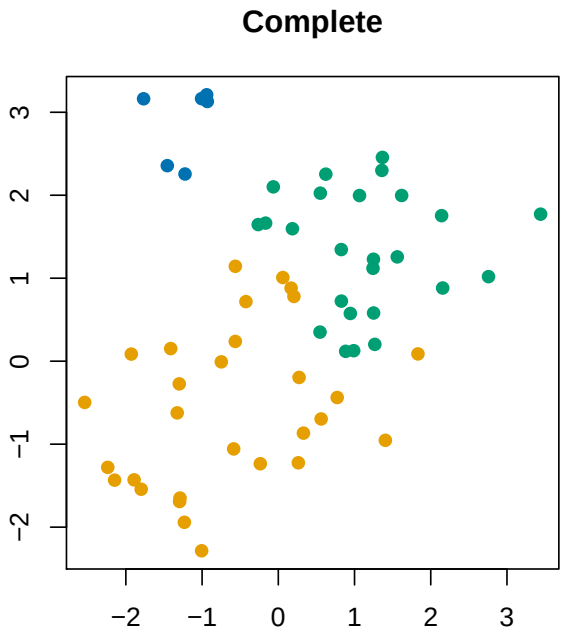
Single and complete linkage have some practical problems:

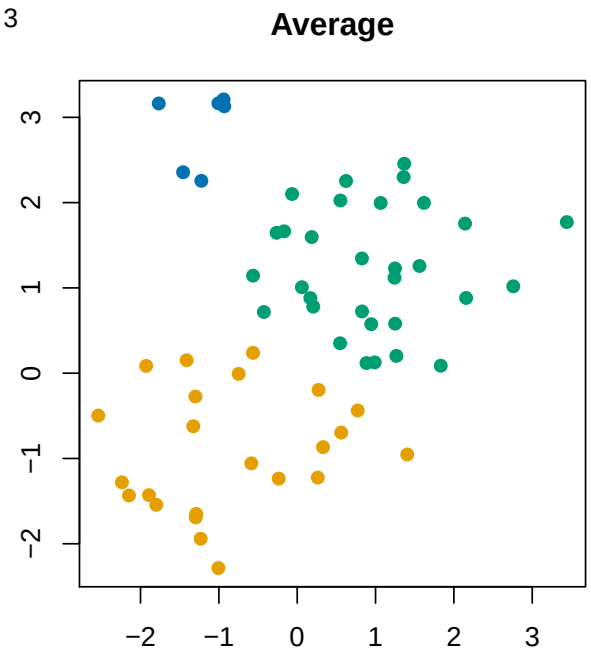
- Single linkage suffers from chaining.
  - In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore *clusters can be too spread out*, and not compact enough.
- Complete linkage avoids chaining, but suffers from crowding.
  - Because its score is based on the worst-case dissimilarity between pairs, *a point can be closer to points in other clusters than to points in its own cluster*. Clusters are compact, but not far enough apart.

Average linkage tries to strike a balance. It uses average pairwise dissimilarity, so clusters tend to be relatively compact and relatively far apart

Example of chaining and crowding





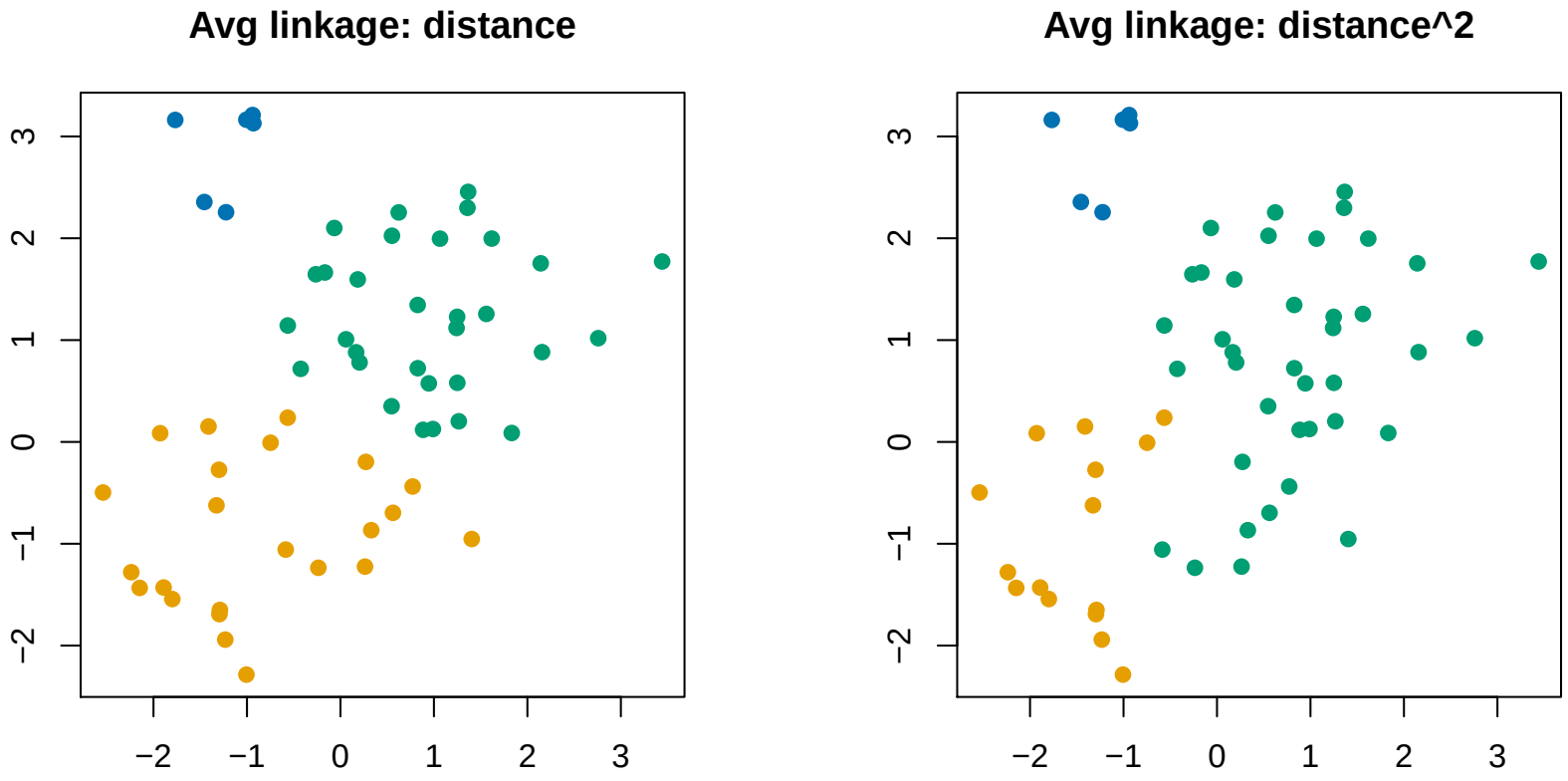


Shortcomings of average linkage

Average linkage has its own problems:

- Unlike single and complete linkage, average linkage *doesn't give us a nice interpretation* when we cut the dendrogram
- Results of average linkage clustering can change if we simply apply a monotone increasing transformation to our dissimilarity measure, our results can change
  - E.g., $d \to d^2$ or $d \to \frac{e^d}{1+e^d}$
  - This can be a big problem if we're not sure precisely what dissimilarity measure we want to use
  - Single and Complete linkage do not have this problem

Average linkage monotone dissimilarity transformation



The left panel uses $d(x_i, x_j) = \|x_i - x_j\|_2$ (Euclidean distance), while the right panel uses $\|x_i - x_j\|_2^2$. The left and right panels would be same as one another if we used single or complete linkage. For average linkage, we see that the results can be different.

Where should we place cell towers?

[source: http://imagicdigital.com/, Mark & Danya Henninger]

Suppose we wanted to place cell towers in a way that ensures that no building is more than 3000ft away from a cell tower. What linkage should we use to cluster buildings, and where should we cut the dendrogram, to solve this problem?

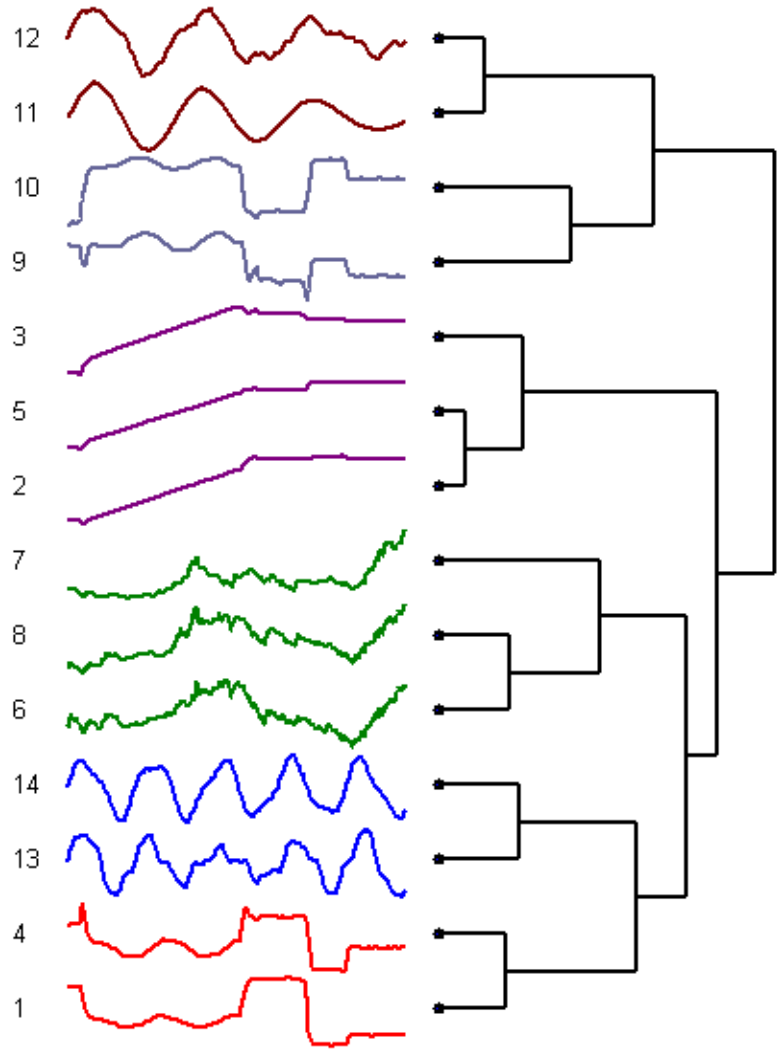
Dissimilarity measures

- The choice of linkage can greatly affect the structure and quality of the resulting clusters
- The choice of dissimilarity (equivalently, similarity) measure is *arguably* even more important
- To come up with a similarity measure, you may need to think carefully and use your intuition about what it means for two observations to be similar. E.g.,
  - What does it mean for two people to have similar *purchasing behaviour*?
  - What does it mean for two people to have similar music listening habits?
- You can apply hierarchical clustering to any similarity measure $s(x_i, x_j)$ you come up with. The difficult part is coming up with a good similarity measure in the first place.

Example: Clustering time series

Here's an example of using hierarchical clustering to cluster time series.

You can quantify the similarity between two time series by calculating the correlation between them. There are different kinds of correlations out there.



[source: A Scalable Method for Time Series Clustering, Wang et al]

K-means vs Hierarchical clustering

- K-means:
  - ☺ Low memory usage
  - ☺ Essentially $O(n)$ compute time
  - ☹ Results are sensitive to random initialization
  - ☹ Number of clusters is pre-defined
  - ?? Awkward with categorical variables

- Hierarchical clustering:
  - ☺ Deterministic algorithm
  - ☺ Dendrogram shows us clusterings for various choices of K
  - ☺ Requires only a distance matrix, quantifying how dissimilar observations are from one another
    - We can use a dissimilarity measure that gracefully handles categorical variables, missing values, etc
  - ☹ Memory-heavy, more computationally intensive than K-means

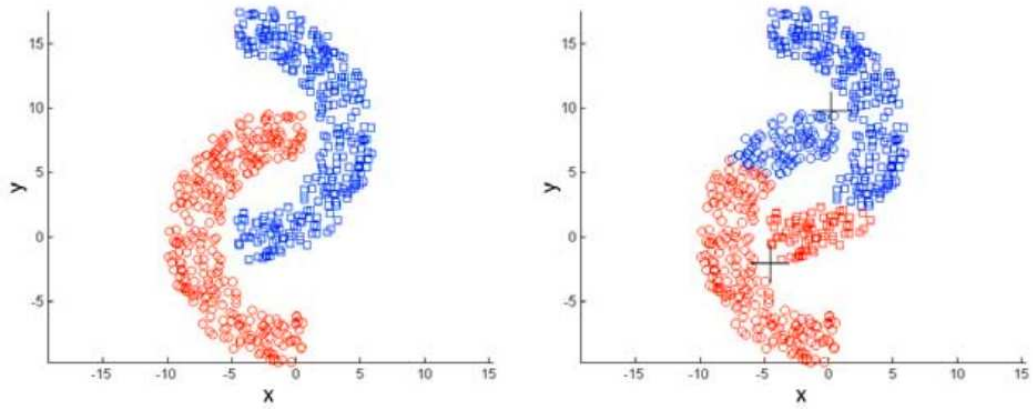
There are lots of practical considerations...

What does it mean for two things to be "similar"?

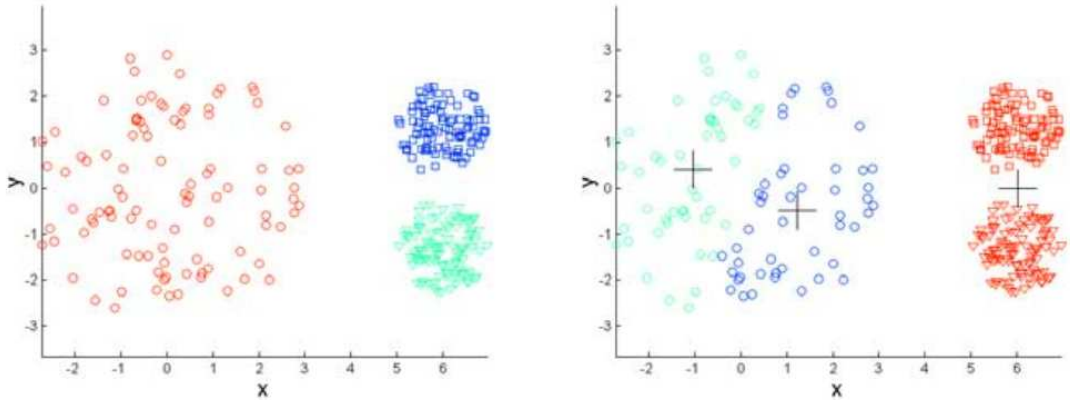
[source: photographs by Sebastian Magnani]

What shape might the clusters be expected to have?

Non-convex/non-round-shaped clusters: Standard K-means fails!



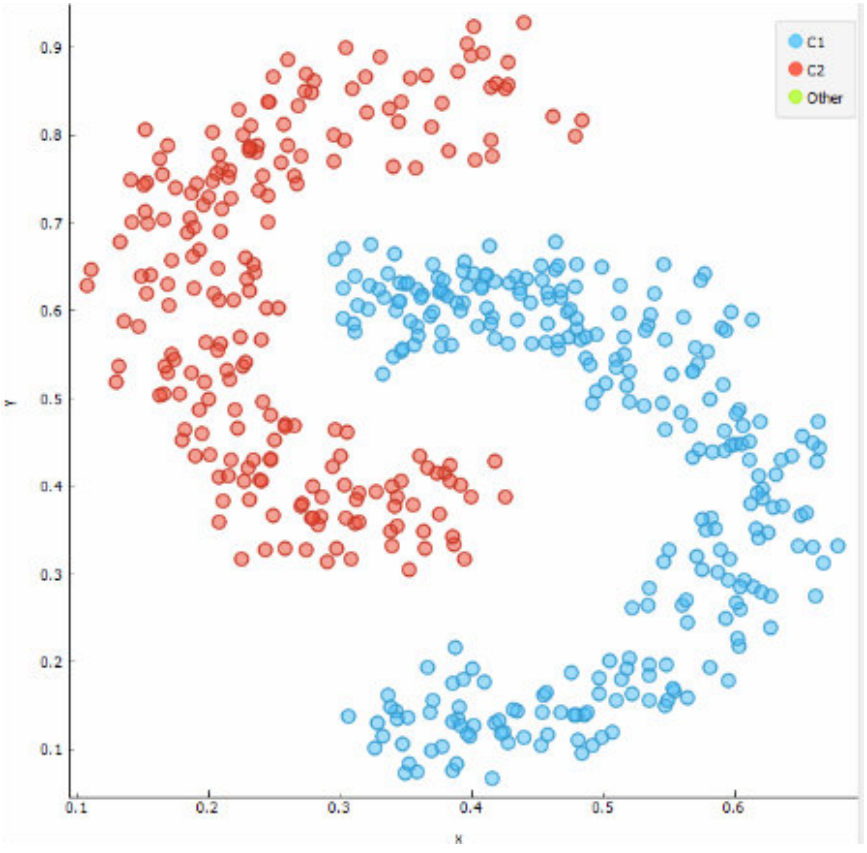
Clusters with different densities



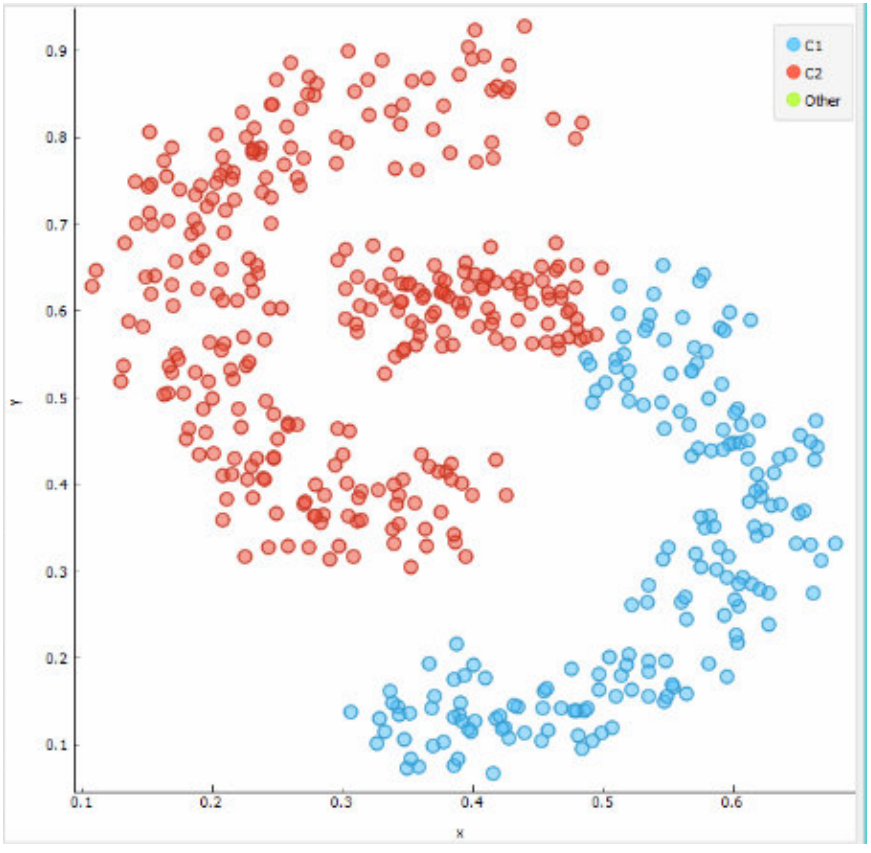
Picture courtesy: Christof Monz (Queen Mary, Univ. of London)

[source: Piyush Rai, CS5350/6350 @ Utah]

Not that your data will ever look like this, but...



Single linkage

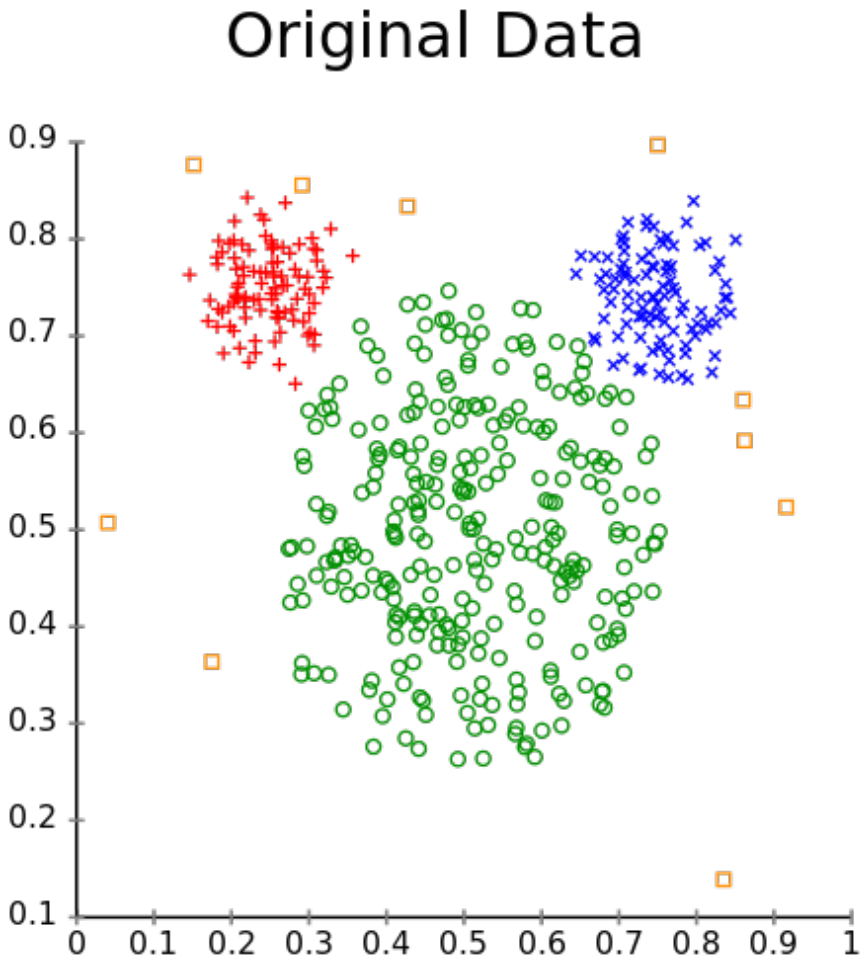


Complete linkage

The chaining property of Single linkage actually helps us in this case. Single linkage reproduces the clusters exactly, while Complete linkage strives to find *compact* clusters, and hence fails on this problem.

[source: Ajda Pretnar, Orange, University of Ljubljana]

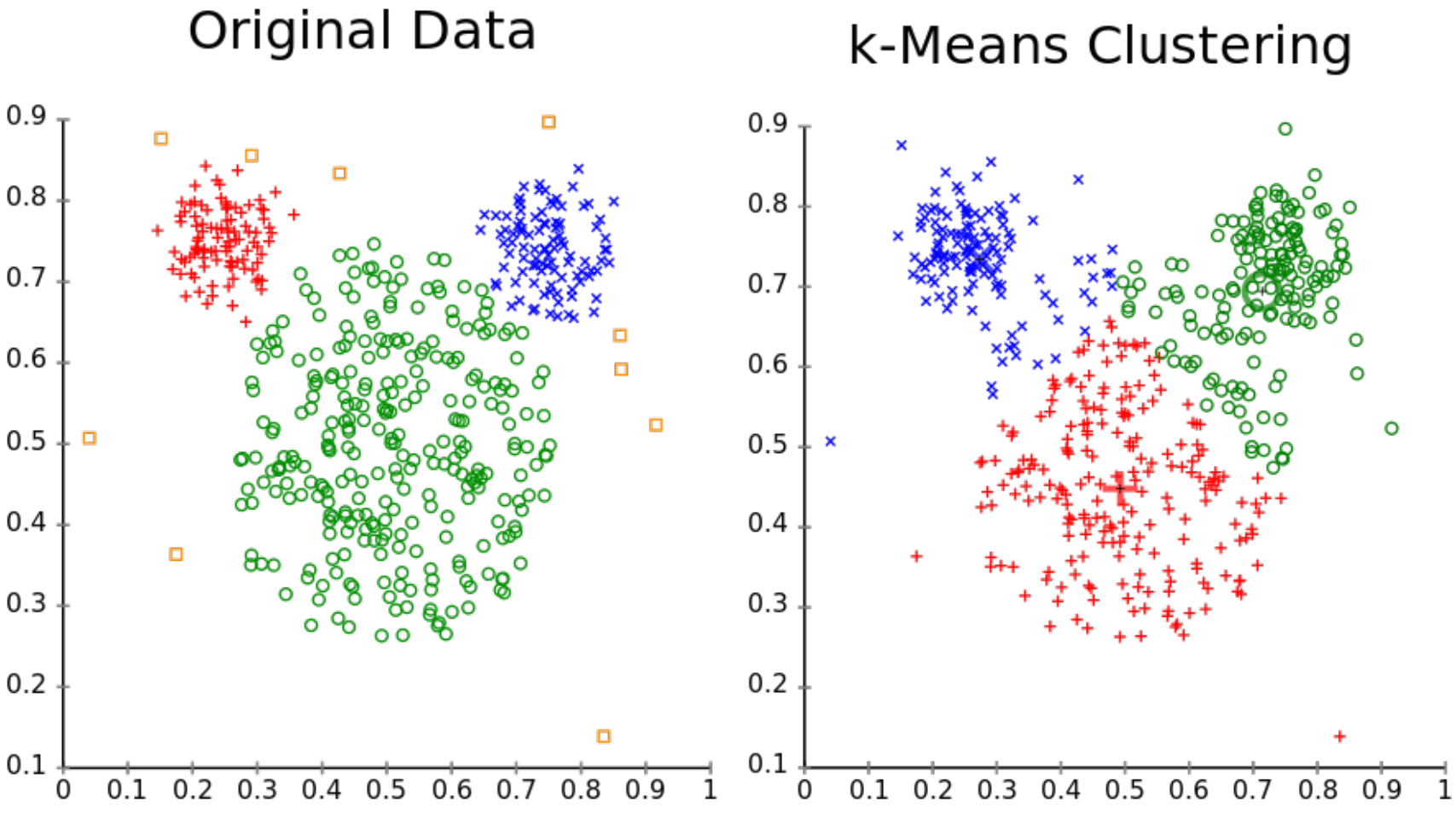
You might get data that looks like this



- Of course, we don't get to observe the labels
- This is a problem that K-means doesn't do well on

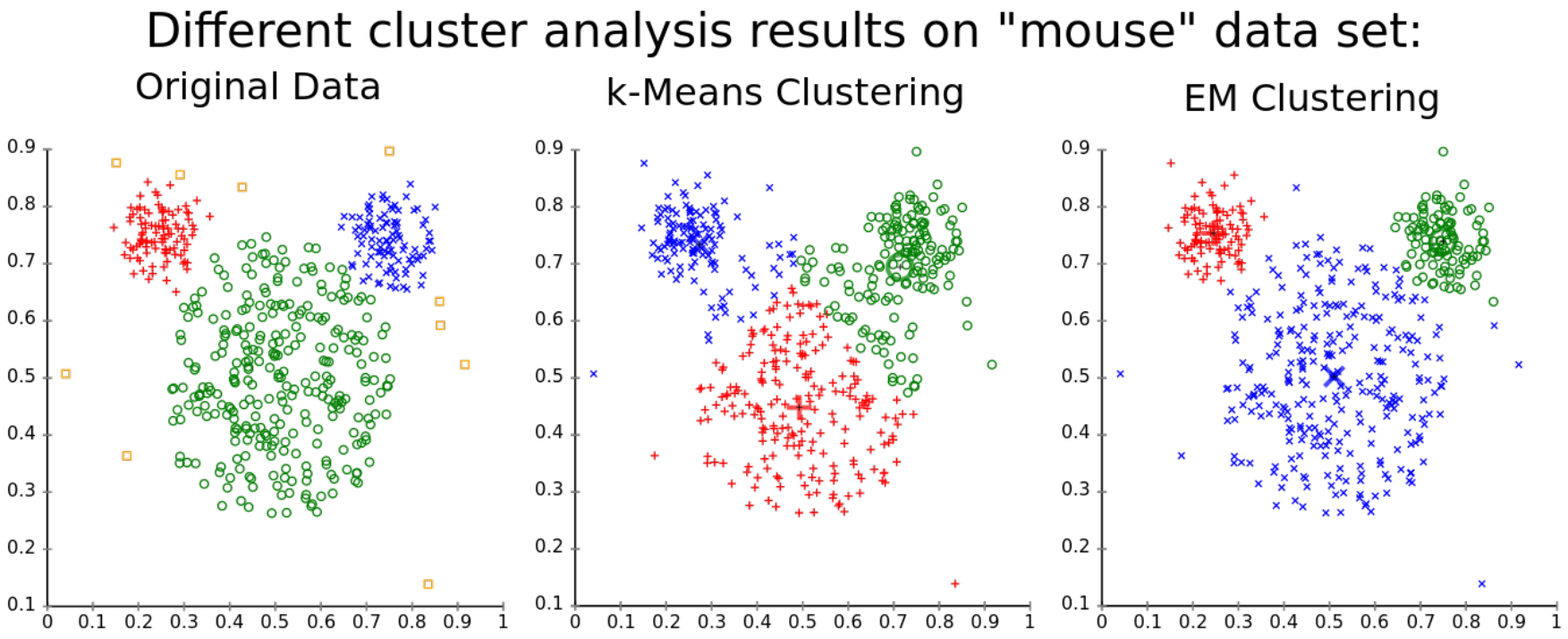
[source: Wikipedia, Public Domain image]

Here's what K-means does



- K-means struggles when the clusters have different densities

Gaussian Mixture Models (EM Clustering)

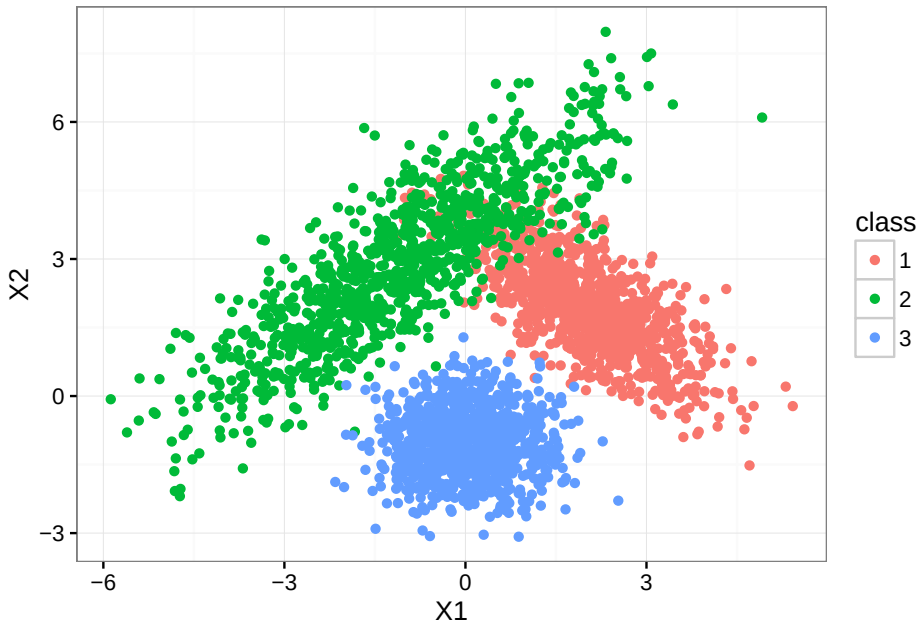


[source: Wikipedia, Public Domain image]

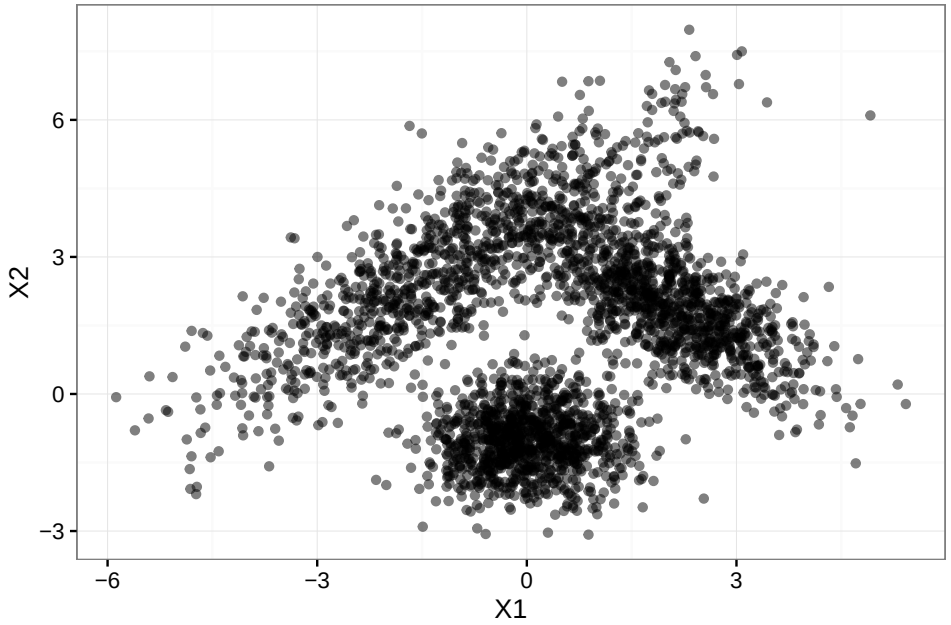
- If this was a classification problem, we'd try to use QDA
- The unsupervised setting, we can apply the same kind of intuition
- Basic idea: Let's assume that the data generating process is a mixture of Multivariate Normals

Gaussian Mixture Models

- Assume each observation has probability π_k of coming from cluster k
- Assume that all observations from cluster k a drawn randomly from a $\text{MVN}(\mu_k, \Sigma_k)$ distribution



i.e., We're assuming that there are latent class labels that we don't observe.



- We don't know what the mixture components look like ahead of time
- We'll use an iterative algorithm that feels kind of like K-means
- Start by fixing K at some value
- Now, if we have estimates $\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi}_k$, we can calculate the posterior probability that observation i comes from class k (E-step)
- If we have the posterior probabilities, we can get updated estimates of $\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi}_k$ (M-step)

EM Algorithm (sketch) for Gaussian mixtures

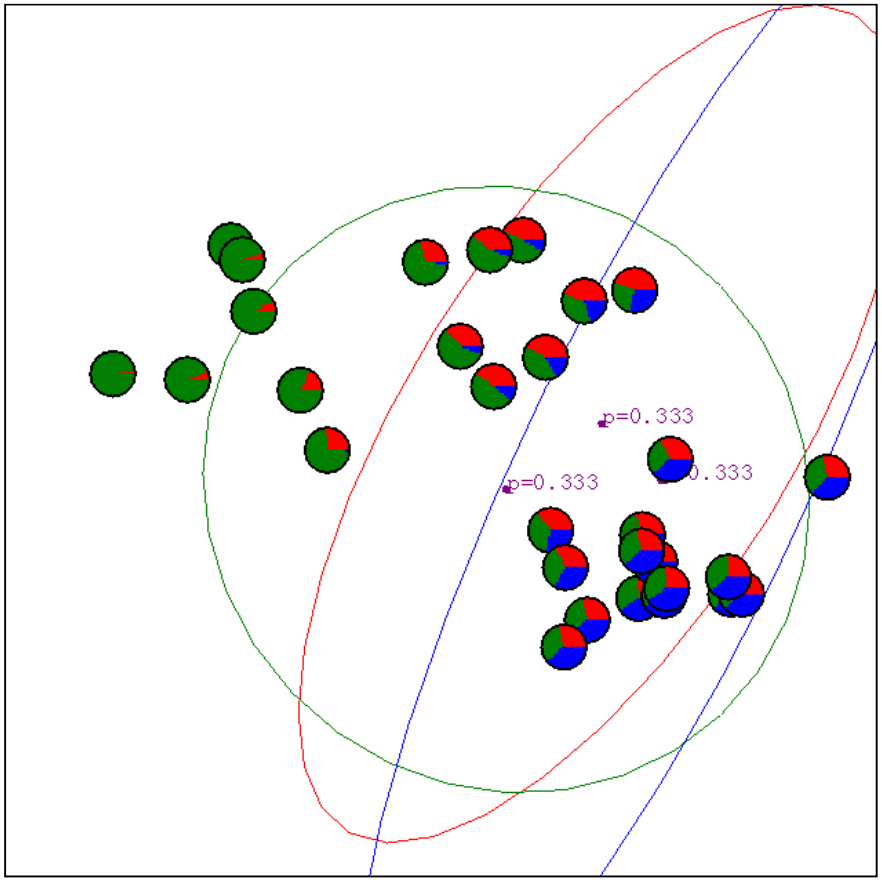
- Take initial guesses for the parameters $\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi}_k$
- Repeat until convergence:
- (Expectation step) Using the *current parameter estimates*, calculate the responsibilities

$$\hat{\theta}_{i,k} = \mathbb{P}(i \in C_k \mid x_i)$$

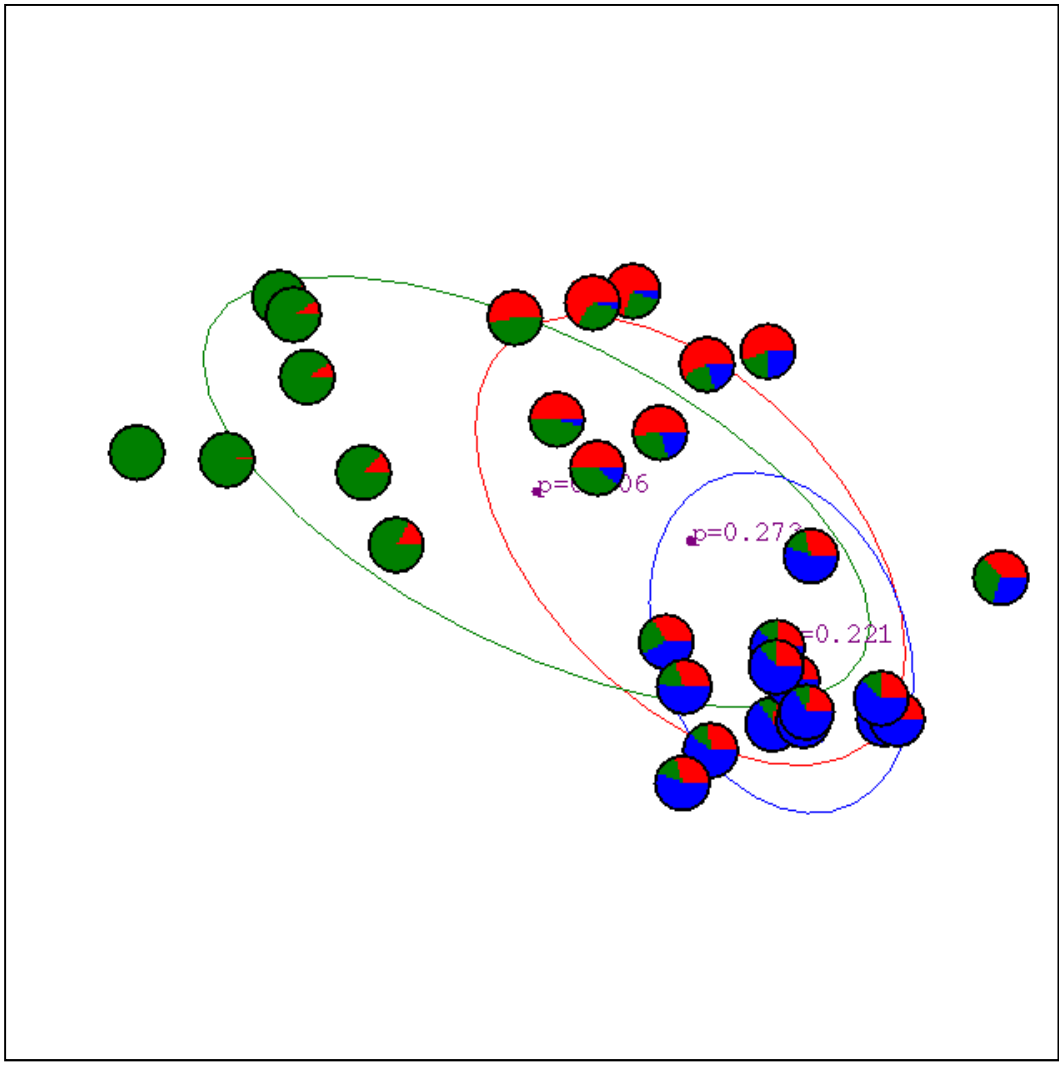
- (Maximization step) Using the *current responsibilities*, re-estimate the parameters.
  - This is done with *weighted averaging*

E.g., the mean estimates work out to

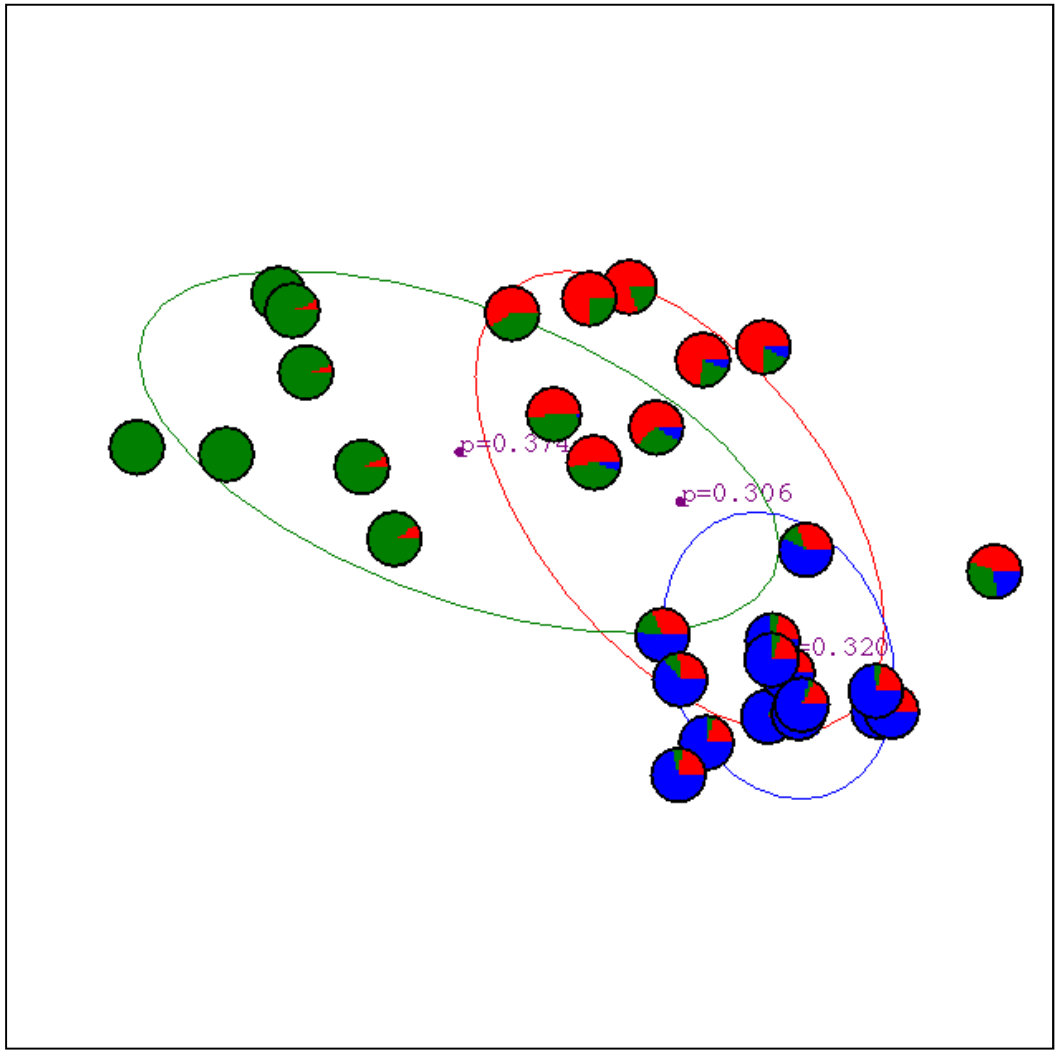
$$\hat{\mu}_k = \frac{\sum_{i=1}^{n} \hat{\theta}_{i,k} x_i}{\sum_{i=1}^{n} \theta_{i,k}}$$



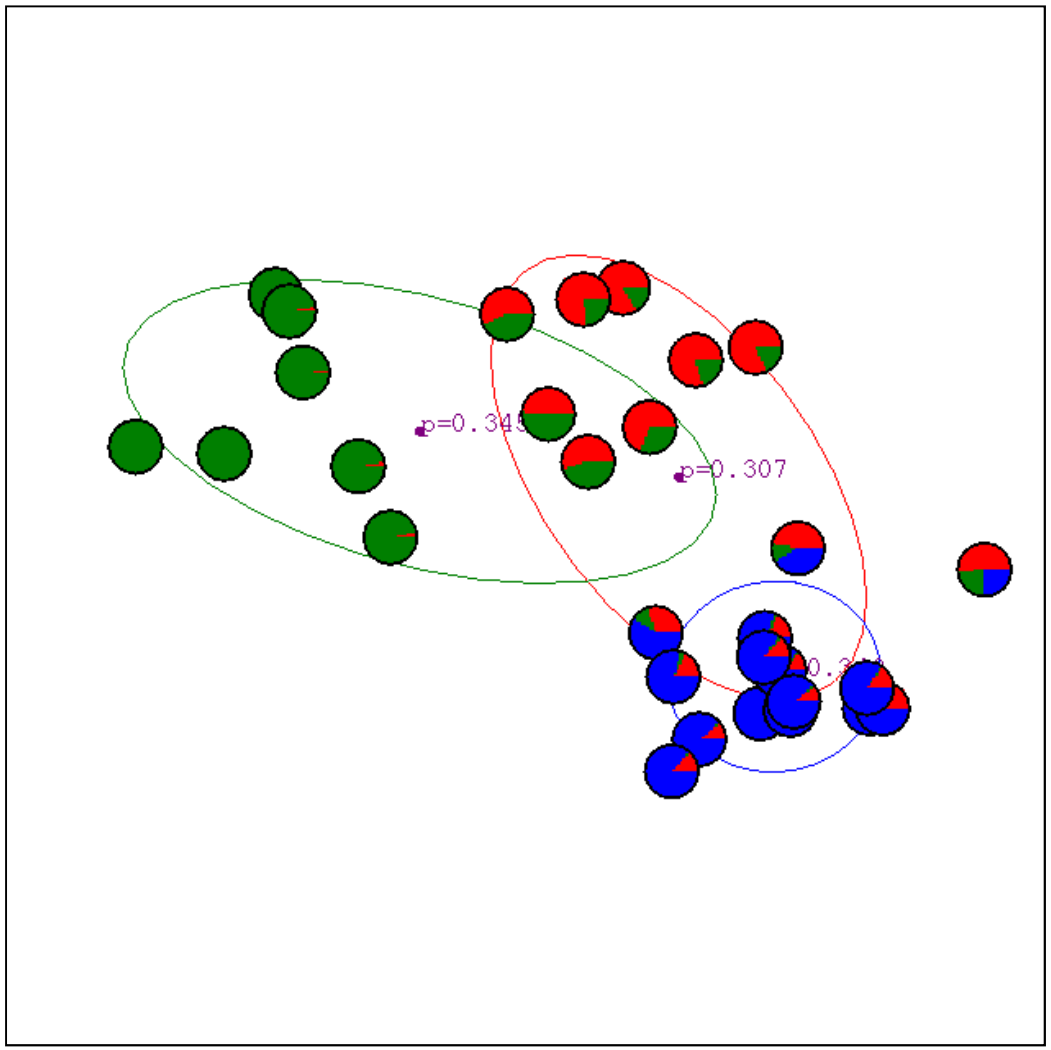
- Each circle (mini pie-chart) is an observation
- Large ovals in the background represent initial $\hat{\mu}_k$, $\hat{\Sigma}_k$.
  $\hat{\pi}_k = 1/3$ for all 3 classes
- Pie chart segments correspond to responsibilities estimates from current $\hat{\mu}_k$, $\hat{\Sigma}_k$, $\hat{\pi}_k$.



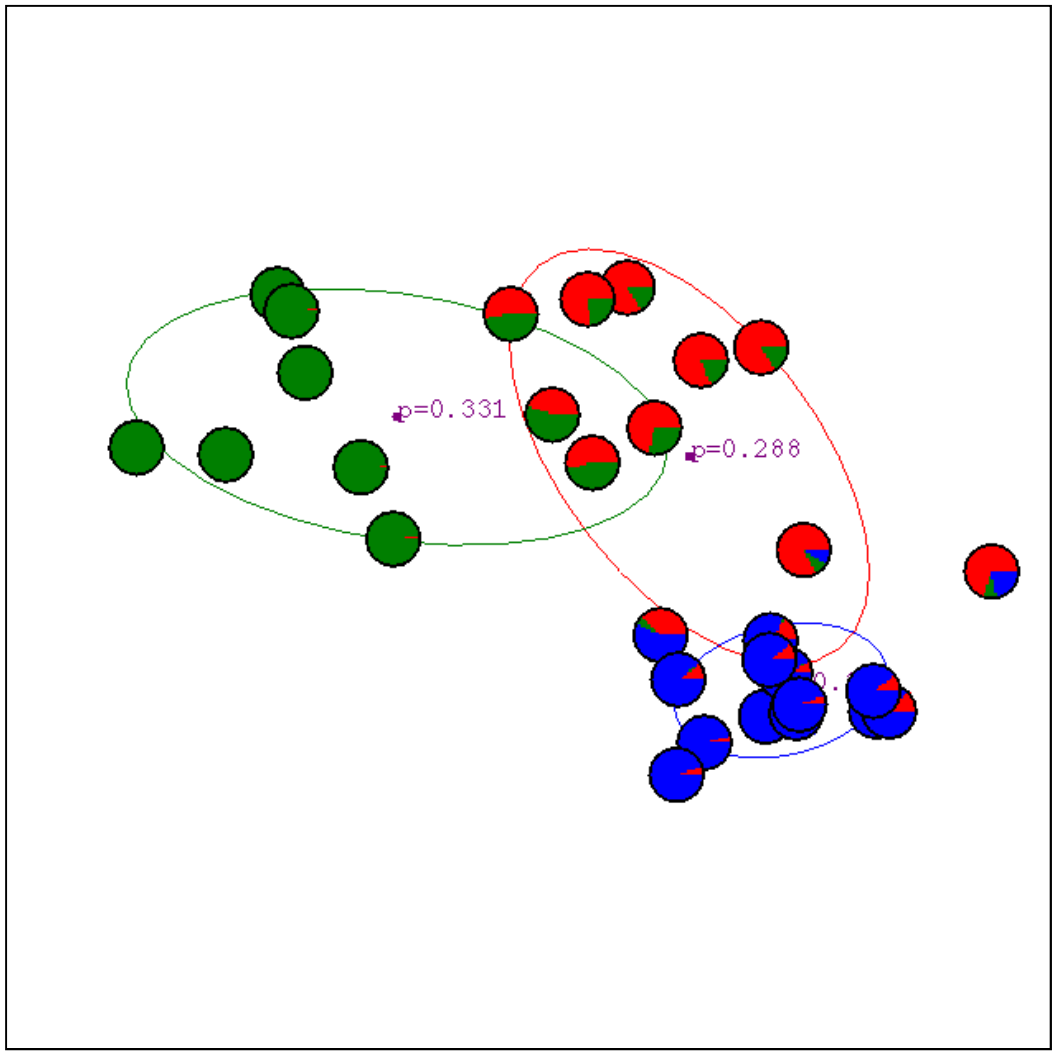
1 iteration of both the E-step and M-step



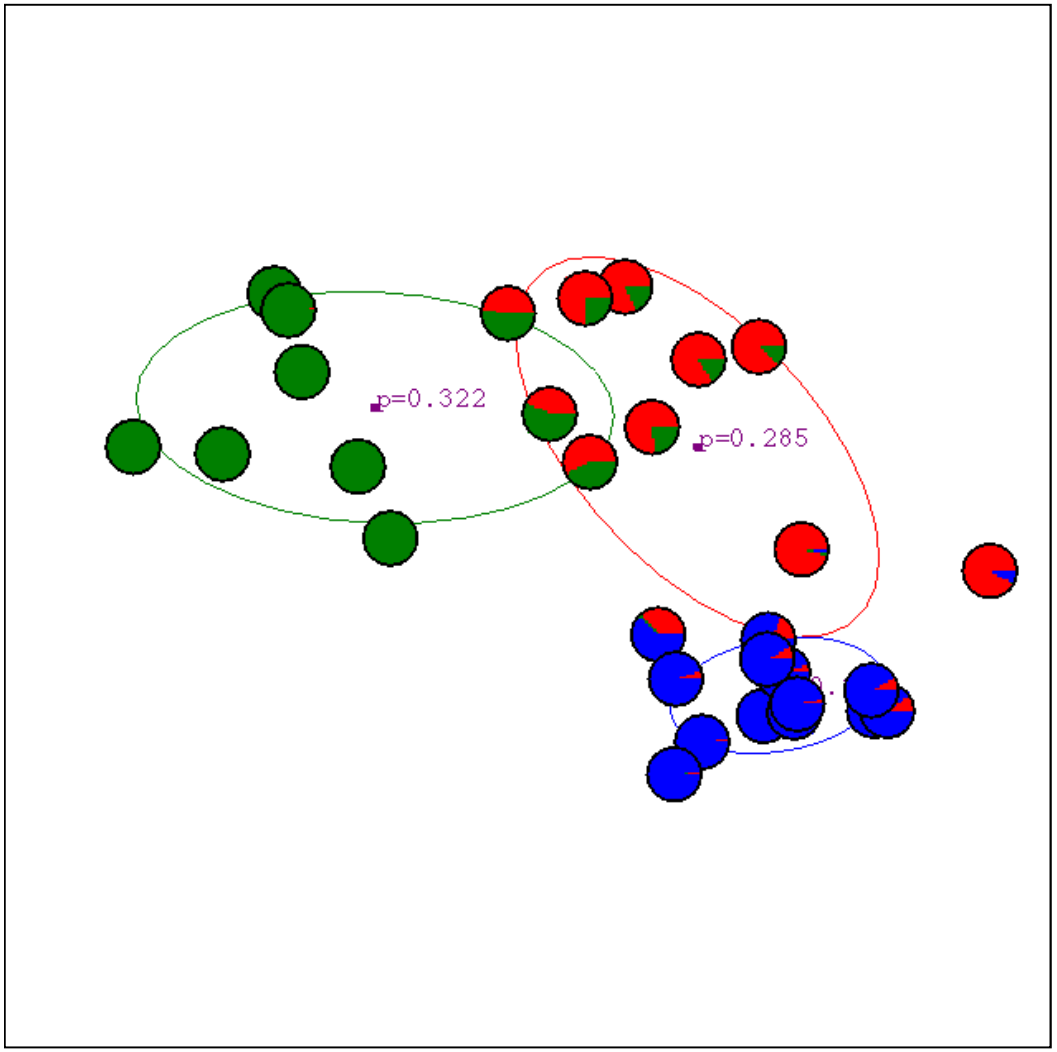
another iteration of both the E-step and M-step



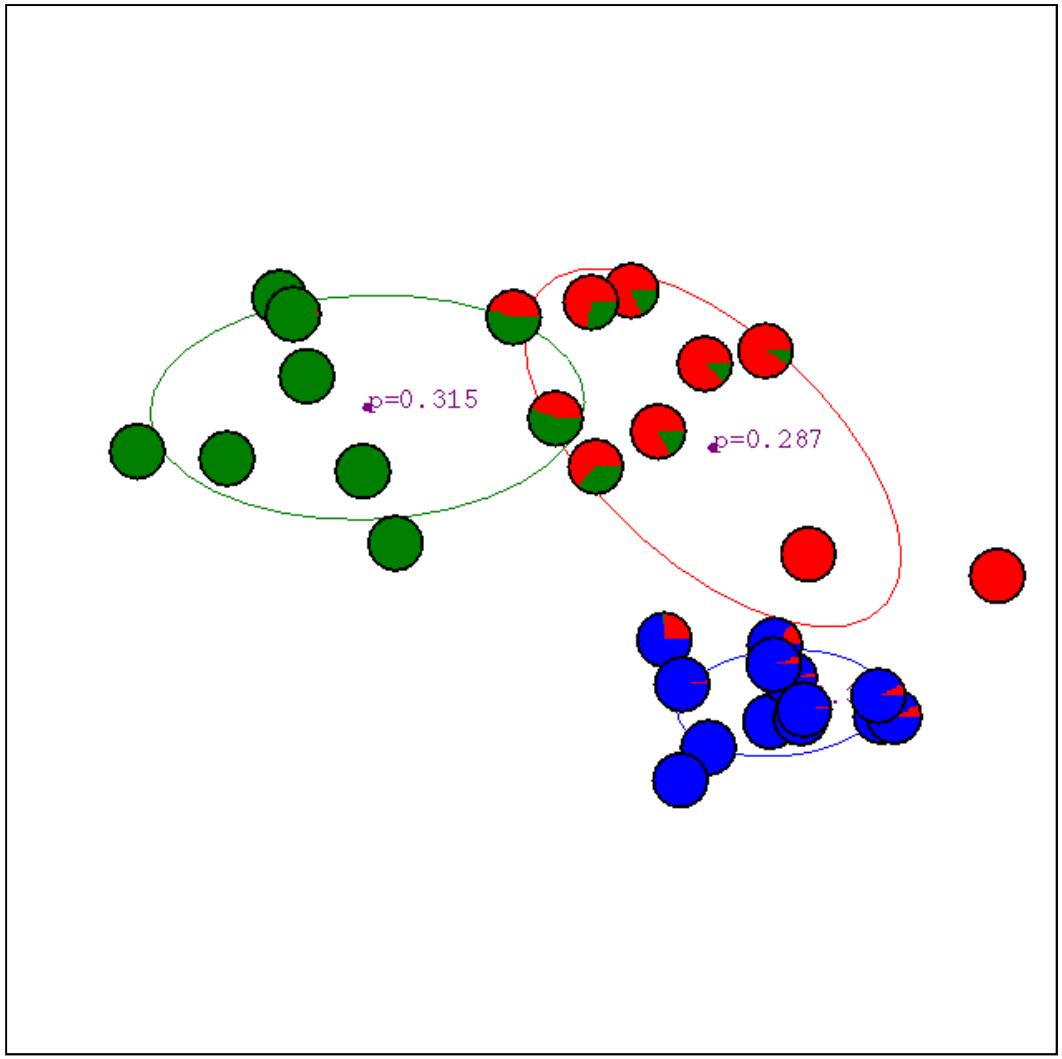
another iteration of both the E-step and M-step.



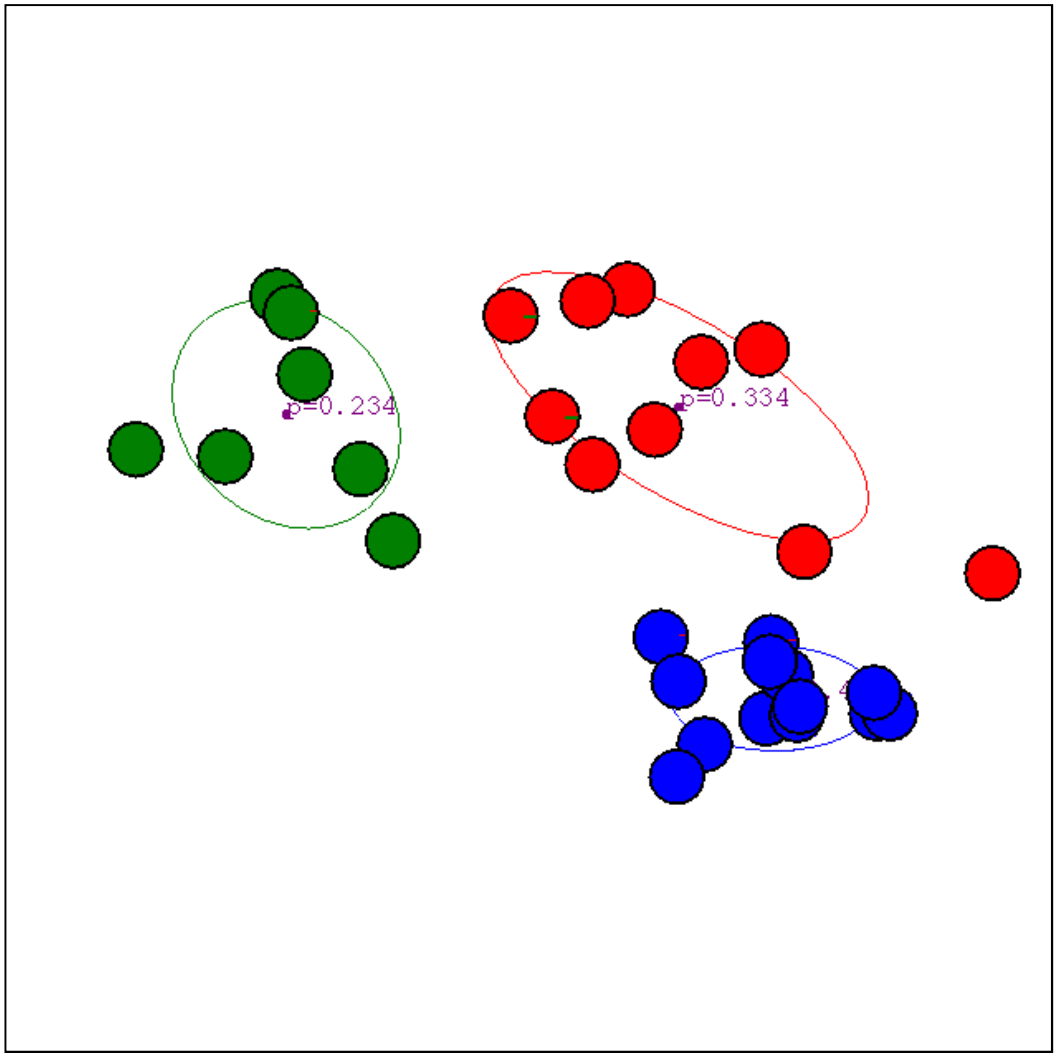
another iteration of both the E-step and M-step..



another iteration of both the E-step and M-step...

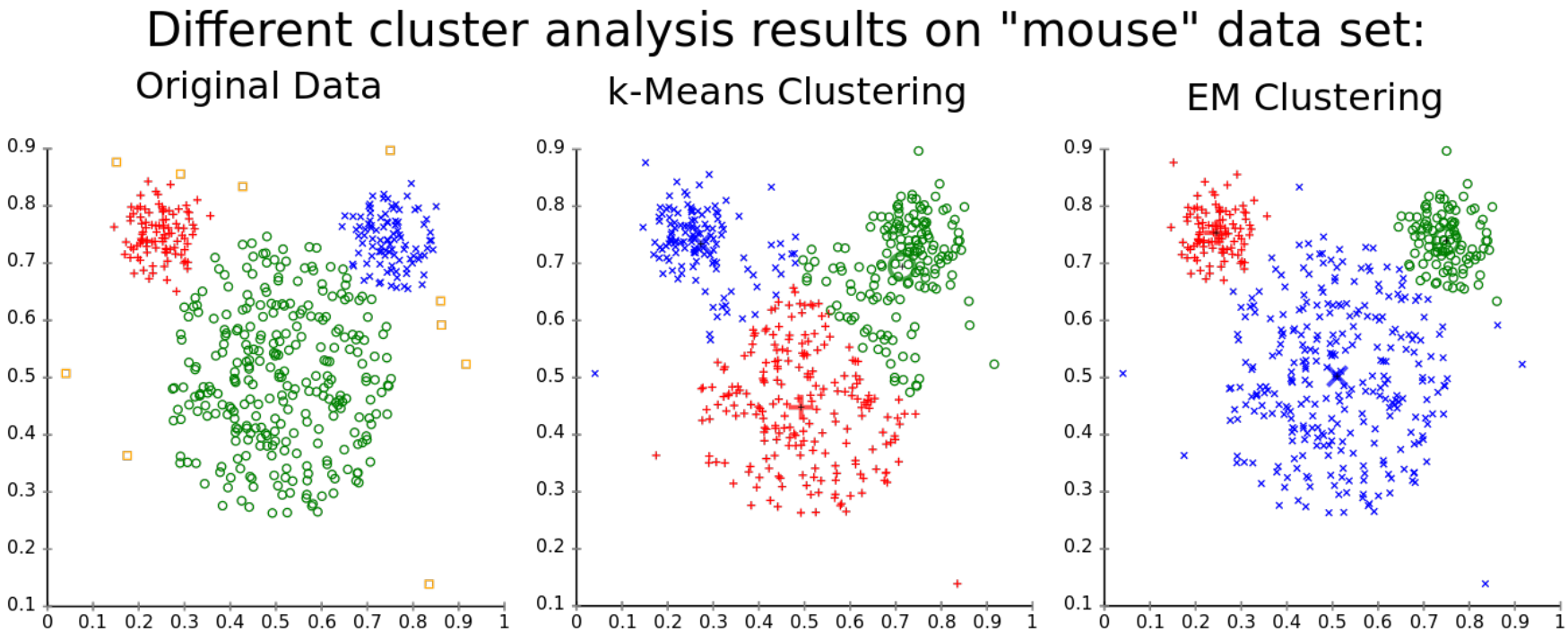


another iteration of both the E-step and M-step....



Final picture: algorithm has converged

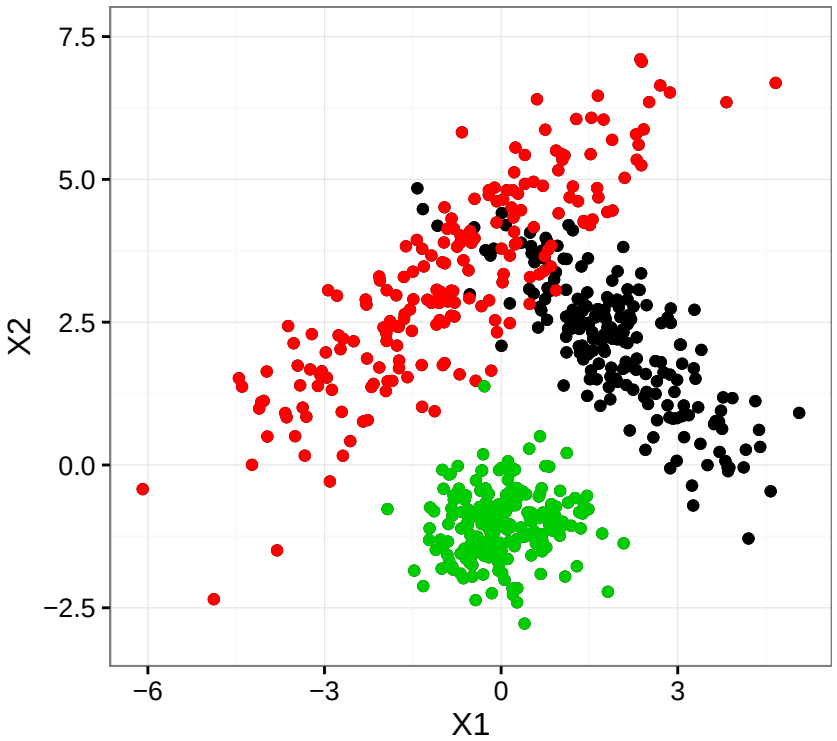
Gaussian Mixture Modeling vs. K-means



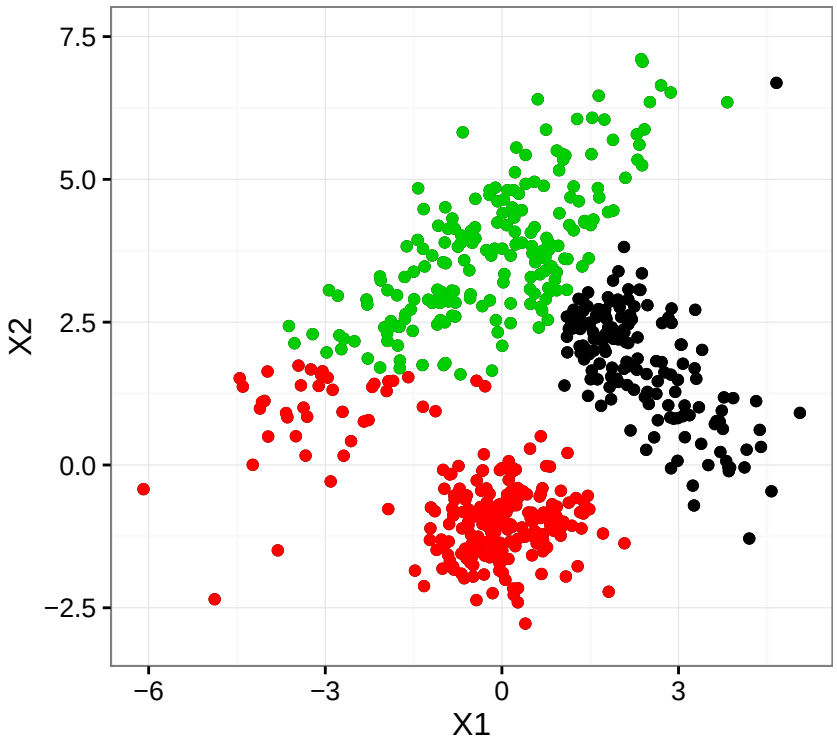
- GMM's do better on this example because they essentially allow for a data-adaptive notion of *distance* when assigning points to centroids
- i.e., In the original data, we have 2 clumps with small variance, and one clump with large variance
- K-means can't capture this added information
- GMM's say: An observation belongs to C_k if its *variance-adjusted* (i.e., Σ_k-adjusted) distance to μ_k is small

A note on variable scaling

- Variable scaling can matter a lot
- Here's an example of data in some original scaling.



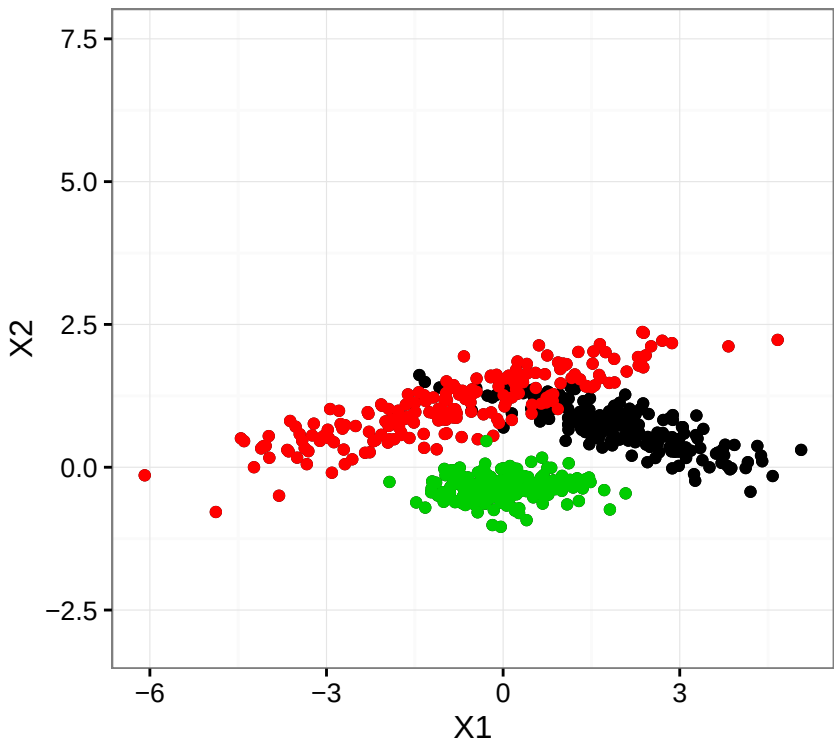
Labeled data



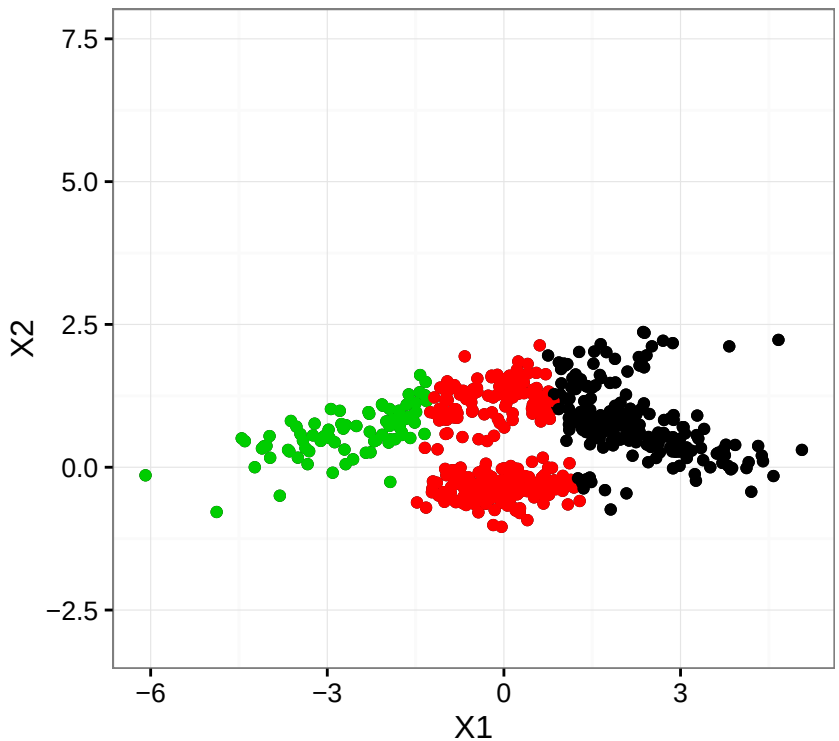
$K = 3$-means clustering

A note on variable scaling

- Here's what happens if we rescale X_2 via $X_2 \leftarrow X_2/3$



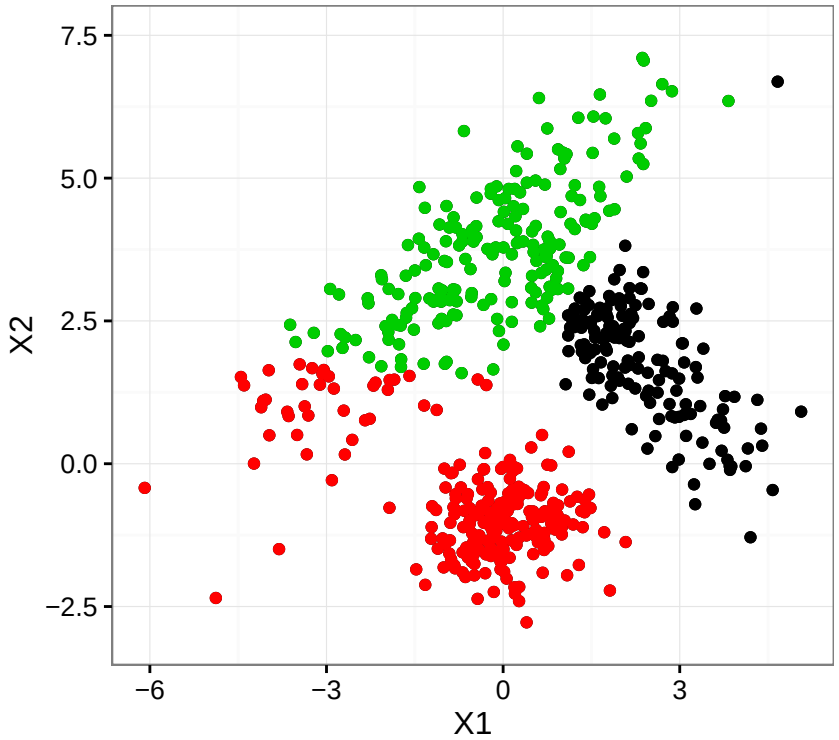
Labeled rescaled data



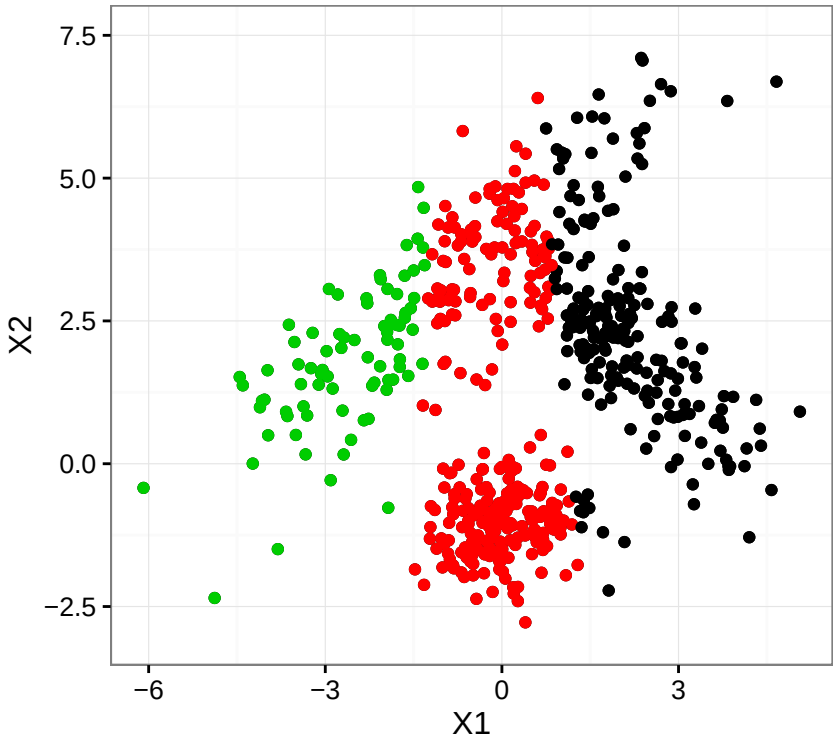
K-means clustering

A note on variable scaling

- To make it easier to compare the results, in the right panel of the Figure below we colour the points in the original data based on the clustering obtained from the *scaled data*



K-means from *unscaled* data



K-means from *scaled* data

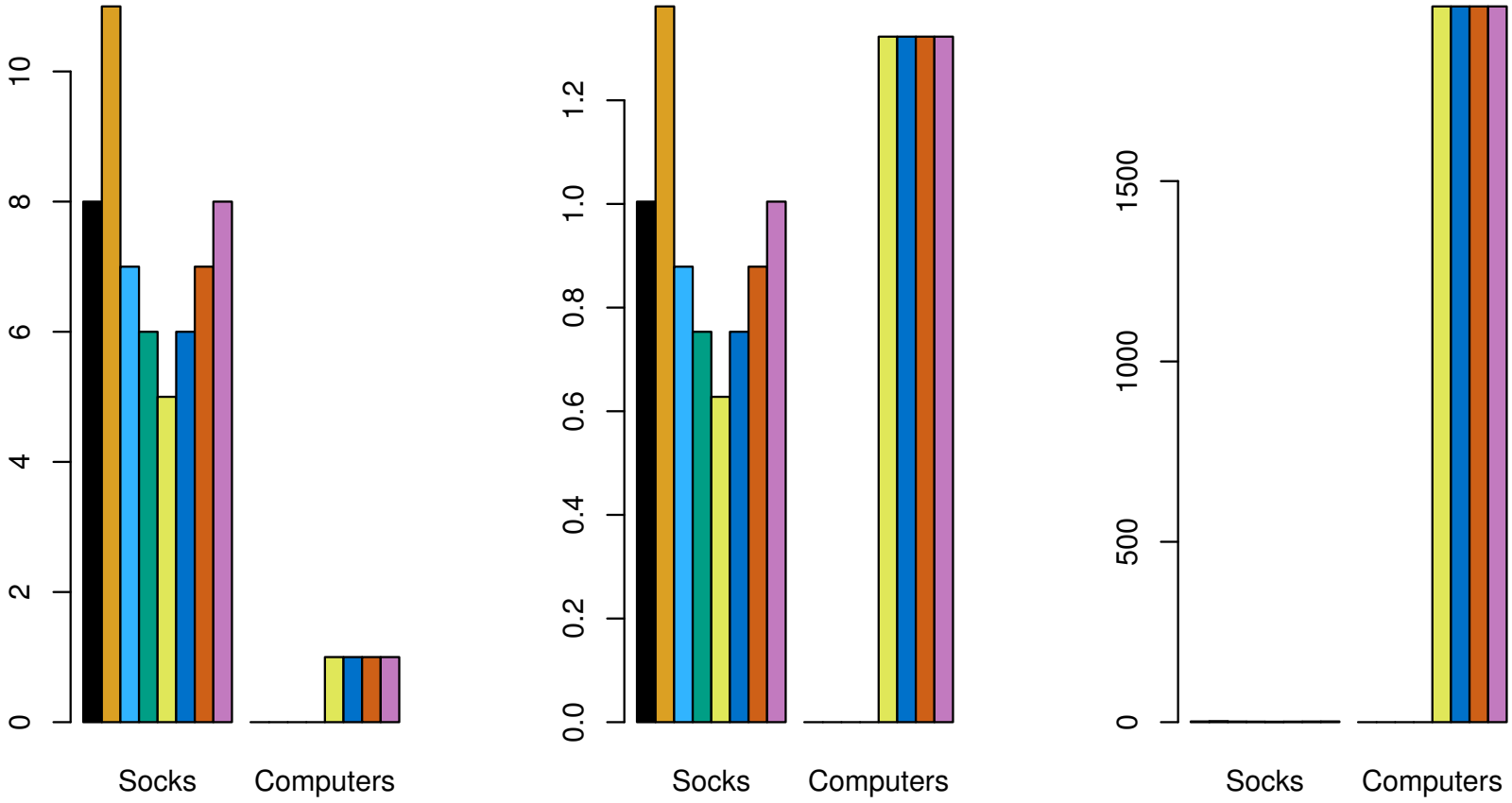


Figure 10.14 from ISL. Here we have # socks and computers purchased. Each observation is represented by a coloured bar. How we scale the `Socks` and `Computers` variables affects how dissimilar we view the people to be.

One general approach: Rescale all variables to have variance 1.

End of Part I

10 minute break

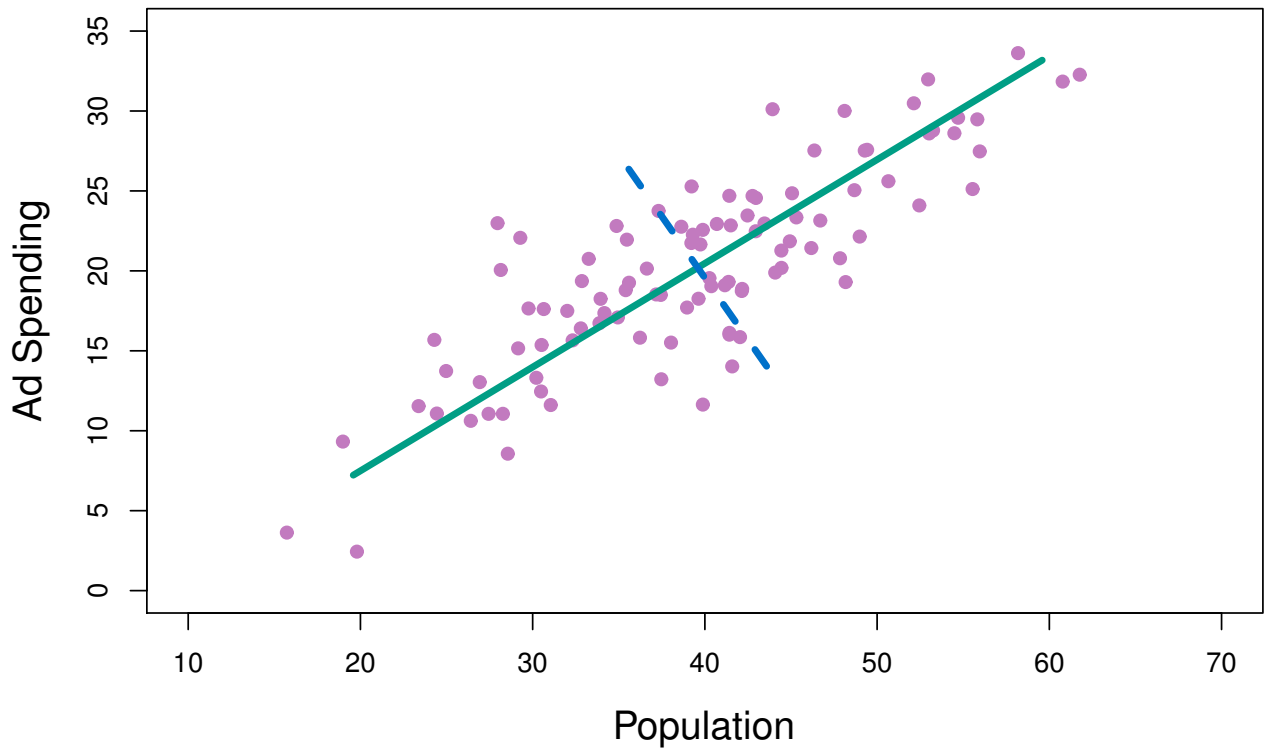
Agenda for Part II

- **Dimensionality reduction**
  - Principal Components Analysis
  - Correspondence analysis
  - Multidimensional scaling

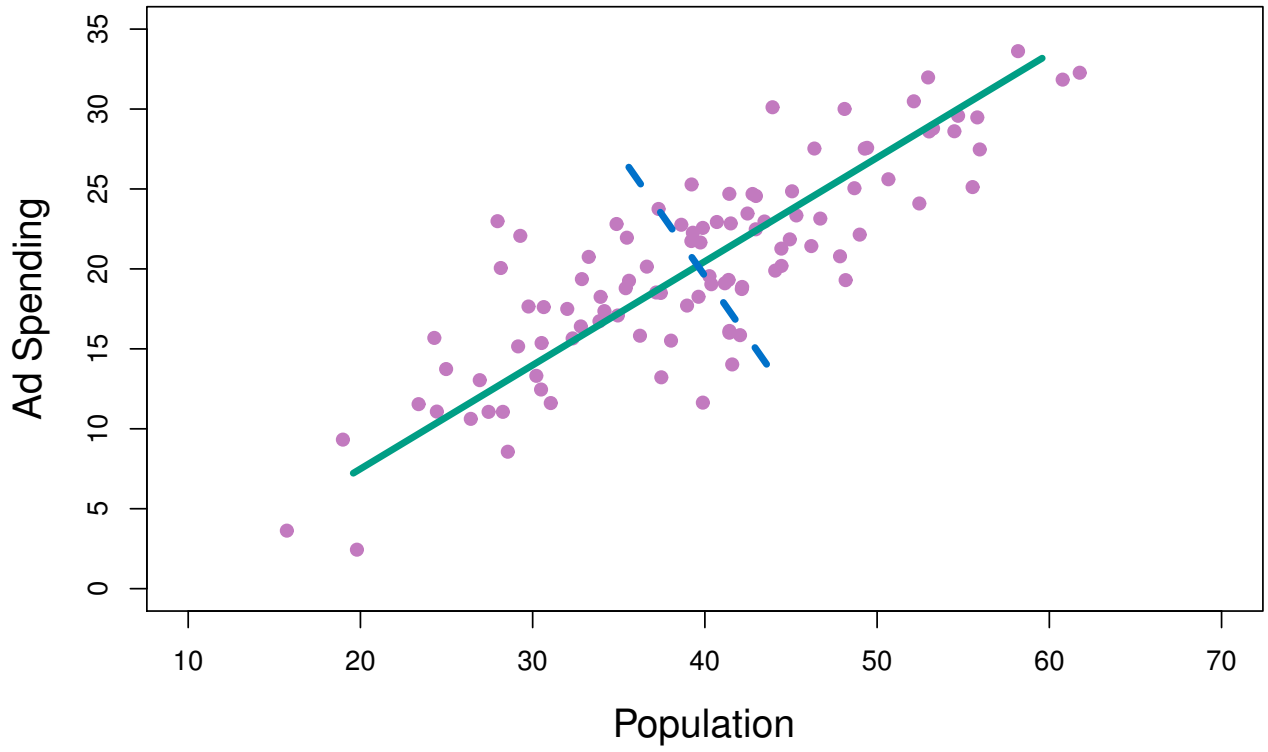
- **Association rules**

Dimensionality reduction

- Dimensionality reduction describes a family of methods for identifying *directions* along with the data varies most highly
- E.g., the plot below shows `Ad Spending` vs. `Population` for $n = 100$ different cities.
- Most of the variation is along the direction of the green diagonal line
- A smaller amount of variation is in the direction of the dashed blue line



Principal components analysis



- Principal components analysis (PCA) takes a data frame and computes the directions of greatest variation
- Essentially: PCA finds linear combinations of the original features that explain as much of the variation in the data as possible
- The green diagonal line in the Figure above is called the first principal direction

Computation of Principal Components

- Start with an $n \times p$ data set $\mathbf{X}$. We only care about variation, so assume all of the columns (variables) have mean 0.
- To find the first principal component, look for the linear combination of features

$$z_{i1} = \phi_{11} x_{i1} + \phi_{12} x_{i2} + \ldots + \phi_{p1} x_{ip}$$

  for $i = 1, \ldots, n$ that has the largest sample variance, subject to the constraint $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- We started by assuming that $\frac{1}{n} \sum_{i=1}^{n} x_{ij} = 0$, which ensures that the sample mean $\frac{1}{n} \sum_{i=1}^{n} z_{i1} = 0$ also
- So we just need to find values of ϕ_{j1} to maximize the sample variance:

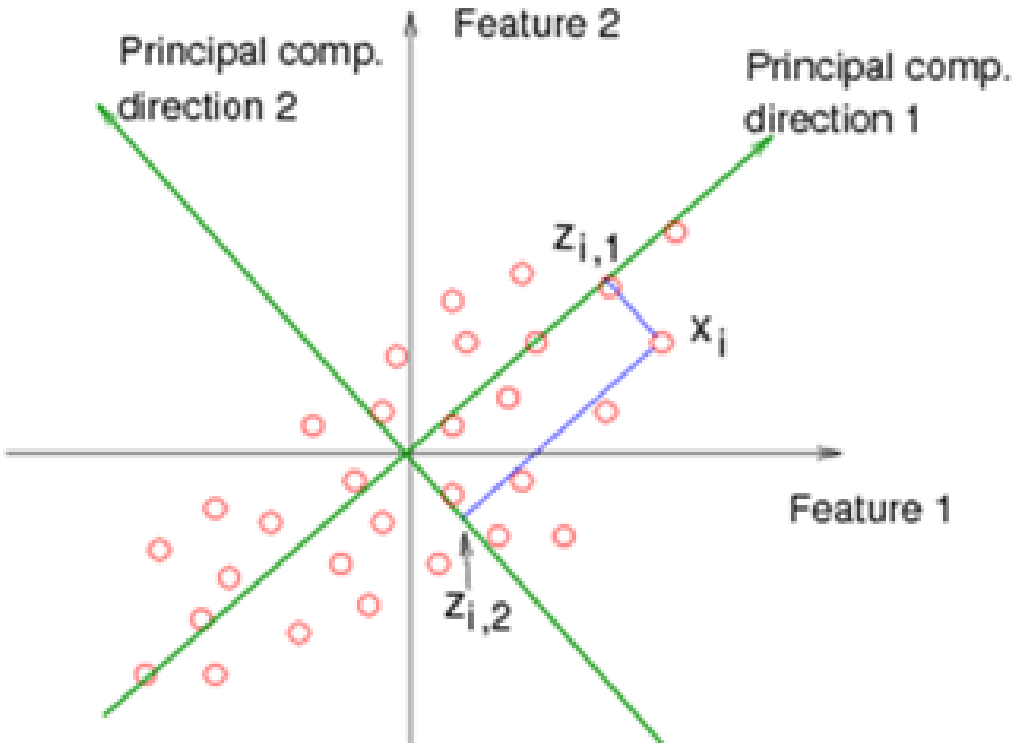
$$\frac{1}{n} \sum_{i=1}^{n} z_{i1}^2$$

  subject to $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.

What does this give us?

- We get a vector $Z_1 = (z_{11}, z_{21}, \ldots, z_{n1})$ called the first principal component
- The vector $\phi_1 = (\phi_{11}, \phi_{21}, \ldots, \phi_{p1})$ is called a loading vector, and defines a direction in feature space along which the data varies the most
- By projecting the n data points $x_1, \ldots, x_n$ onto this direction, we get the principal component scores $(z_{11}, z_{21}, \ldots, z_{n1})$
- To find the second principal component Z_2, we repeat the process, further requiring that Z_2 be uncorrelated with Z_1
  - This amounts to requiring that the second principal direction ϕ_2 be orthogonal to the first direction ϕ_1
- In general, for the kth principal component Z_k, we figure out the direction ϕ_k that maximizes the variance, requiring that ϕ_k be *orthogonal* to $\phi_1, \phi_2, \ldots, \phi_{k-1}$

A picture



[source: https://onlinecourses.science.psu.edu/stat857]

- This Figure shows an observation $x_i = (x_{i1}, x_{i2})$ along with z_{i1}, its projection onto the first principal component direction, and z_{i2}, its projection onto the second principal component direction

Example: USArrests Data

- `USArrests data`: For each of the $n = 50$ states in the United States, the data set contains the number of arrests per $100{,}000$ residents for each of three crimes: `Assault`, `Murder`, and `Rape`.

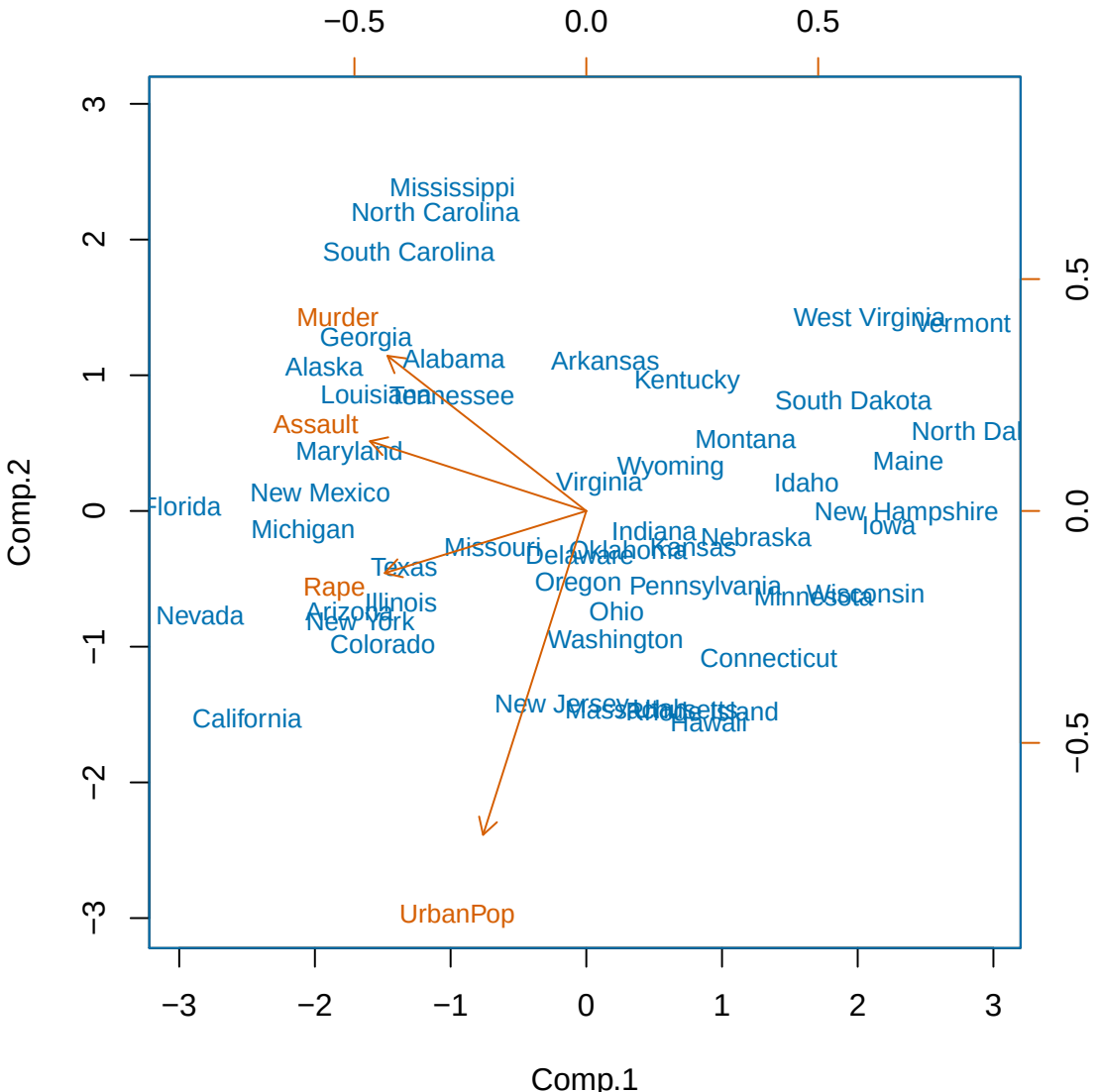
- We also record `UrbanPop`, the percent of the population in each state living in urban areas.

- The principal component score vectors have length $n = 50$, and the principal component loading vectors (directions) have length $p = 4$.

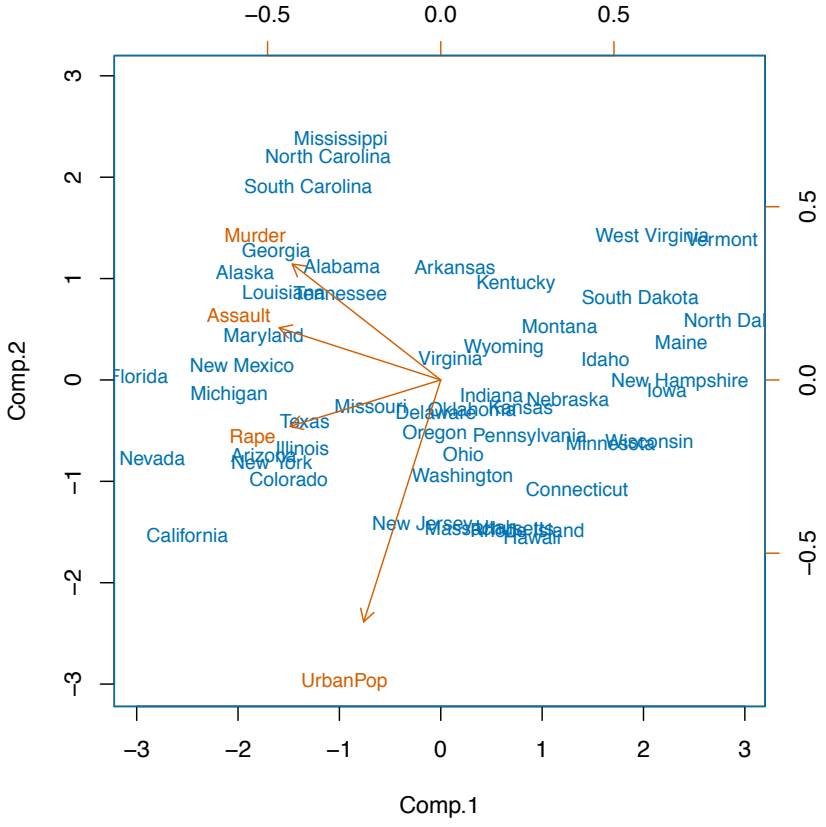
- PCA was performed after standardizing each variable to have mean zero and standard deviation one.
  - Normalization is very important!

USArrests biplot

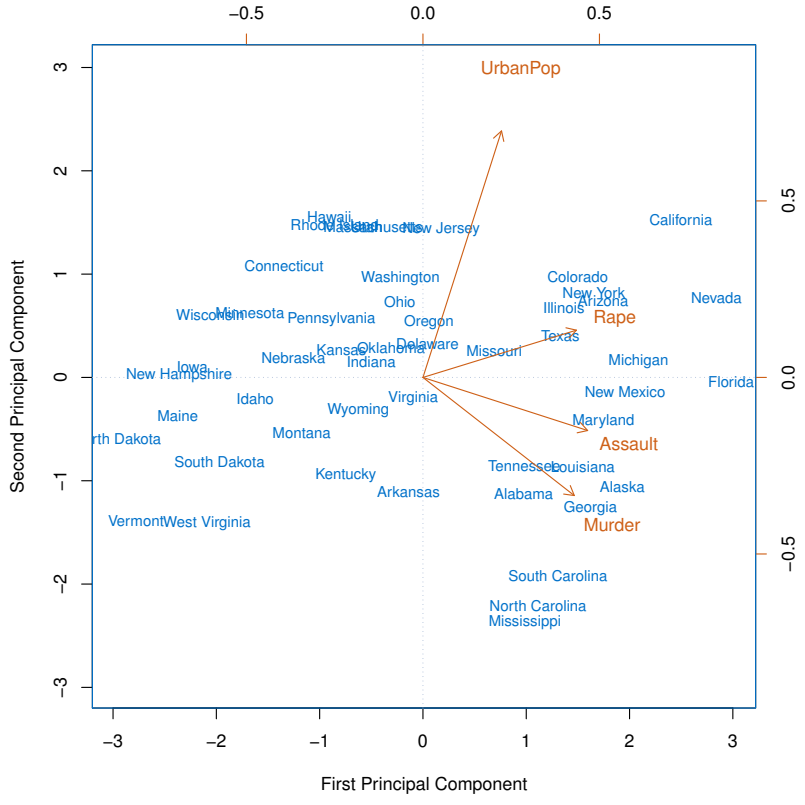
```
arrests.scaled <- scale(USArrests) # Normalize the data
arrests.pca <- princomp(arrests.scaled) # Perform PCA
biplot(arrests.pca, scale = 0) # Construct biplot
```



What happened?



Our biplot



ISL Figure 10.1

- The direction of the arrows and location of the points is now flipped...
- The PCA solution is *non-unique* in this sense: Loading vectors ϕ_1 and $-\phi_1$ will produce the same variance for the scores z_{i1}, but will result in opposite signs.

Let's look at the loadings

Here's what ISL reports for the loadings ϕ_1, ϕ_2:

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

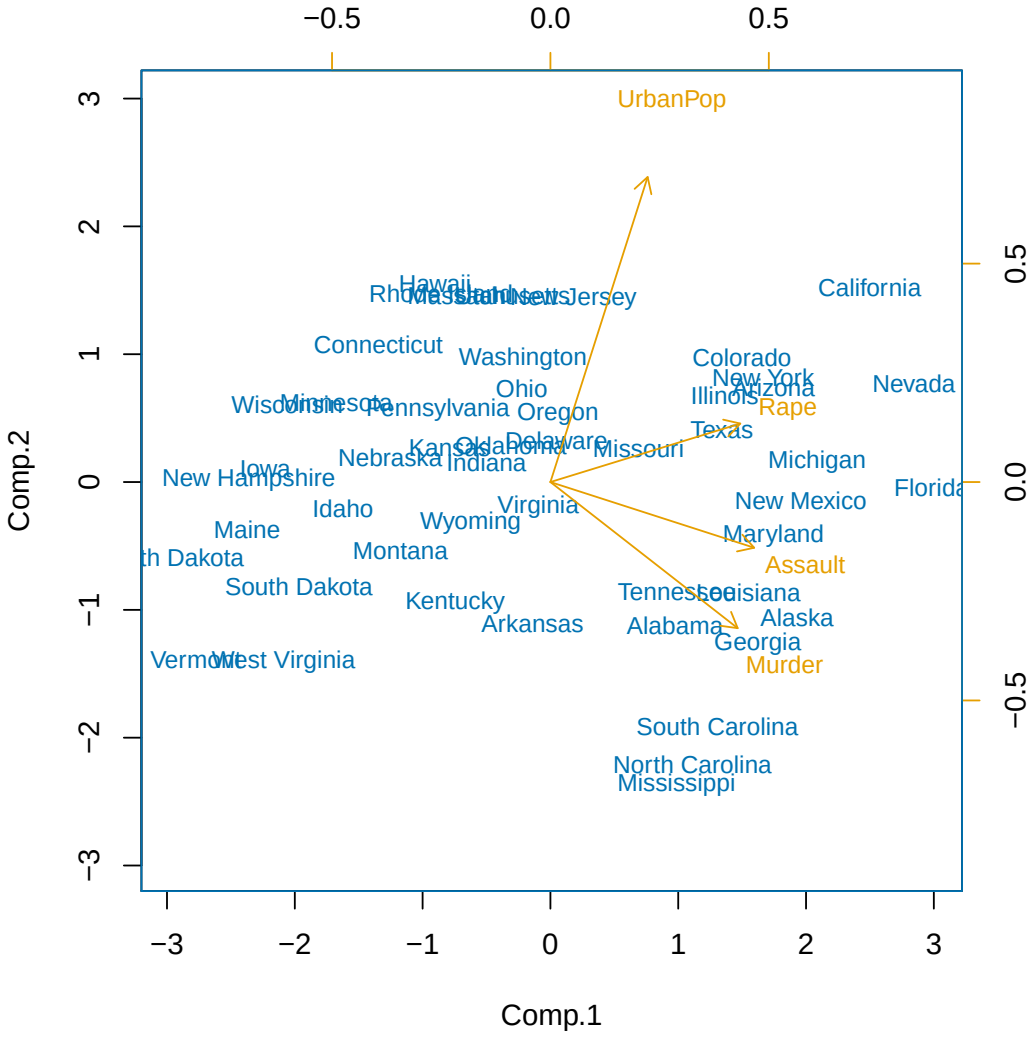
Here's what we got (all 4 loading vectors are shown):

	Comp.1	Comp.2	Comp.3	Comp.4
Murder	-0.53590	0.41818	-0.34123	0.64923
Assault	-0.58318	0.18799	-0.26815	-0.74341
UrbanPop	-0.27819	-0.87281	-0.37802	0.13388
Rape	-0.54343	-0.16732	0.81778	0.08902

- If we flip the sign of all the values in our table, we'll get the same answer as ISL.
- We'll want to flip the signs on the loadings ϕ_j and the scores z_j

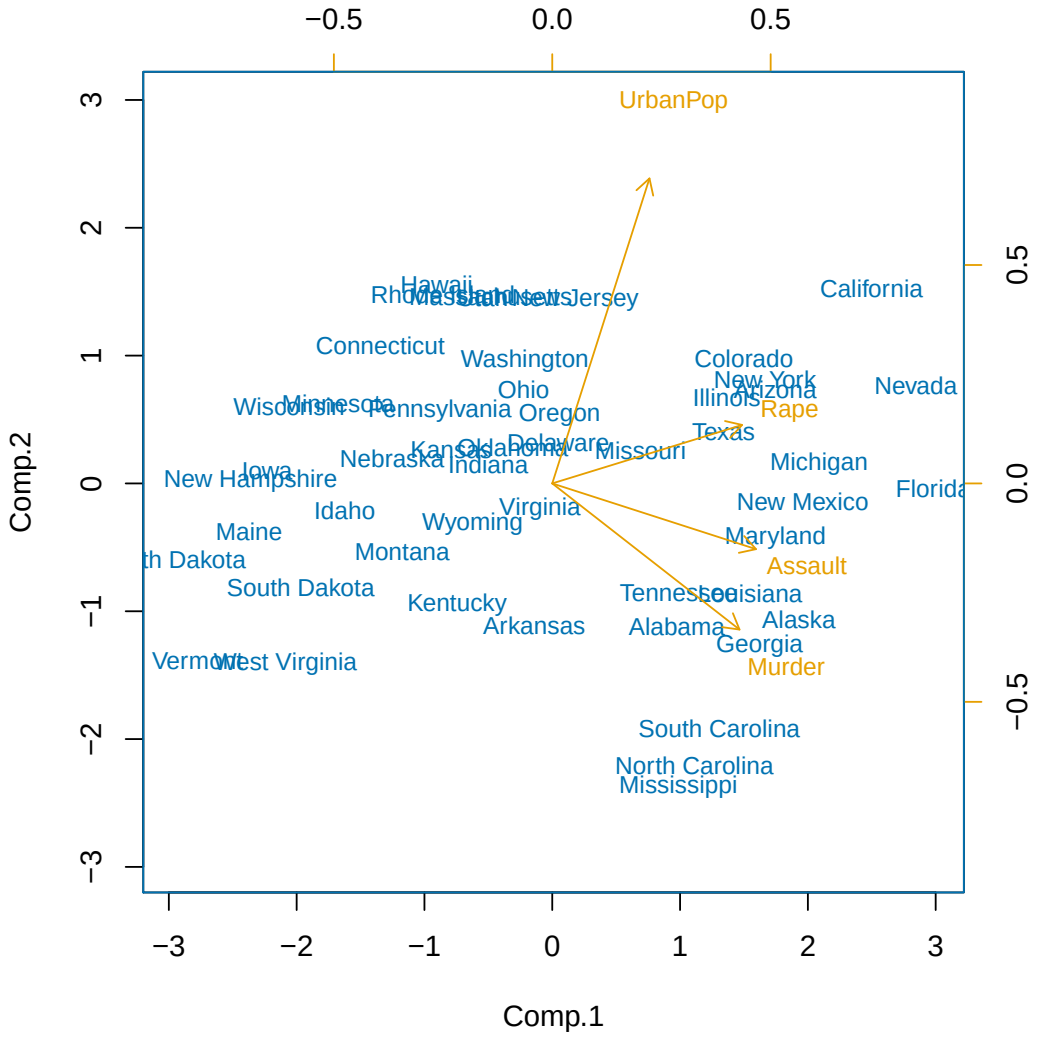
Here's our new biplot

```
arrests.pca$loadings = -arrests.pca$loadings # Flip loadings (phi's)
arrests.pca$scores = -arrests.pca$scores # Flip scores (z's)
biplot(arrests.pca, scale = 0) # Construct biplot
```



Loadings

	ϕ_1	ϕ_2
Murder	0.54	-0.42
Assault	0.58	-0.19
UrbanPop	0.28	0.87
Rape	0.54	0.17



- The word UrbanPop is centered at $(0.28, 0.87)$ (in terms of the top and right side coordinate axes)

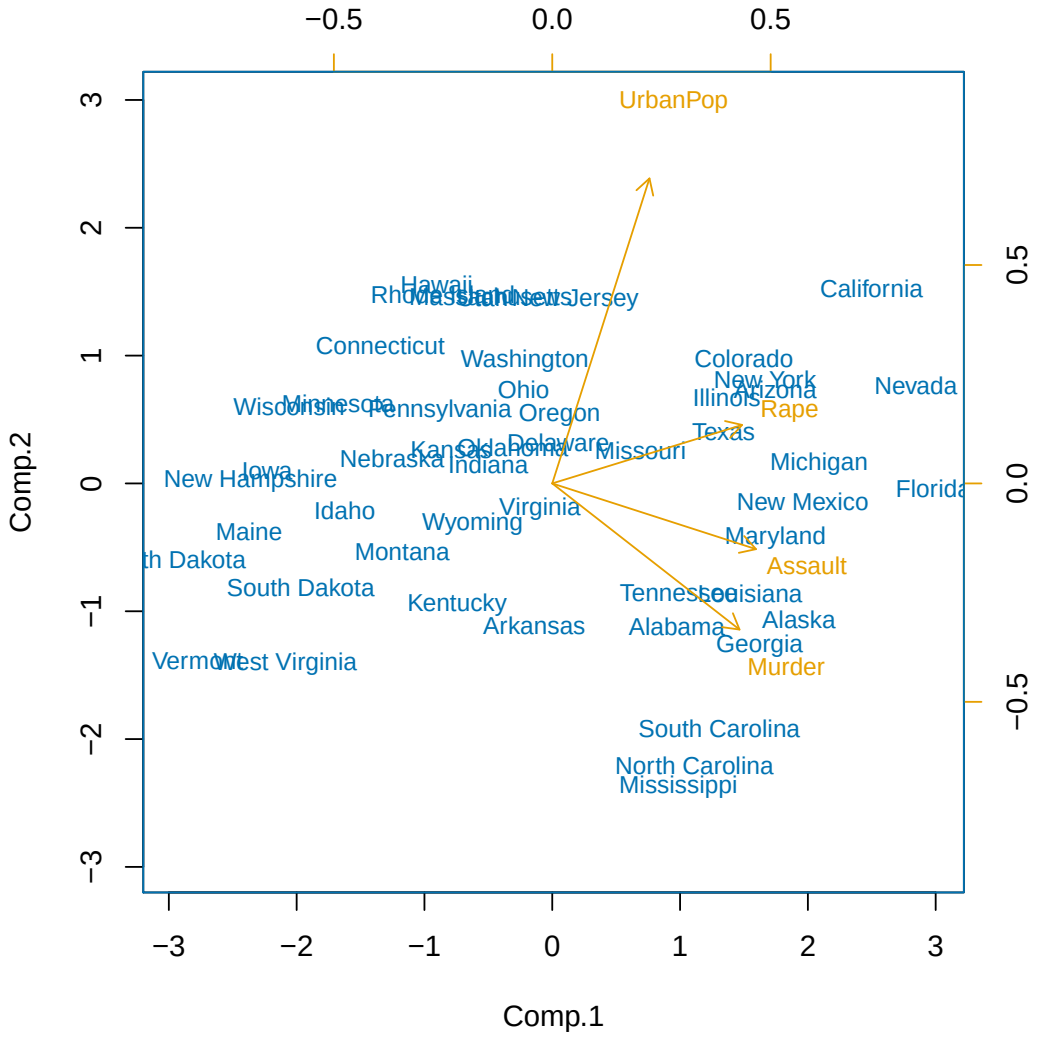
Let's look at the scores (z's)

	Comp.1	Comp.2	Comp.3	Comp.4
Alabama	0.98	-1.12	0.44	-0.15
Alaska	1.93	-1.06	-2.02	0.43
Arizona	1.75	0.74	-0.05	0.83
Arkansas	-0.14	-1.11	-0.11	0.18
California	2.50	1.53	-0.59	0.34
Colorado	1.50	0.98	-1.08	-0.00

- Just like we get 4 loading vectors, we get 4 score vectors.
- The table above shows the scores for all 4 principal components
- The biplot is constructed by plotting the points with Comp.1 on the x-axis and Comp.2 on the y-axis

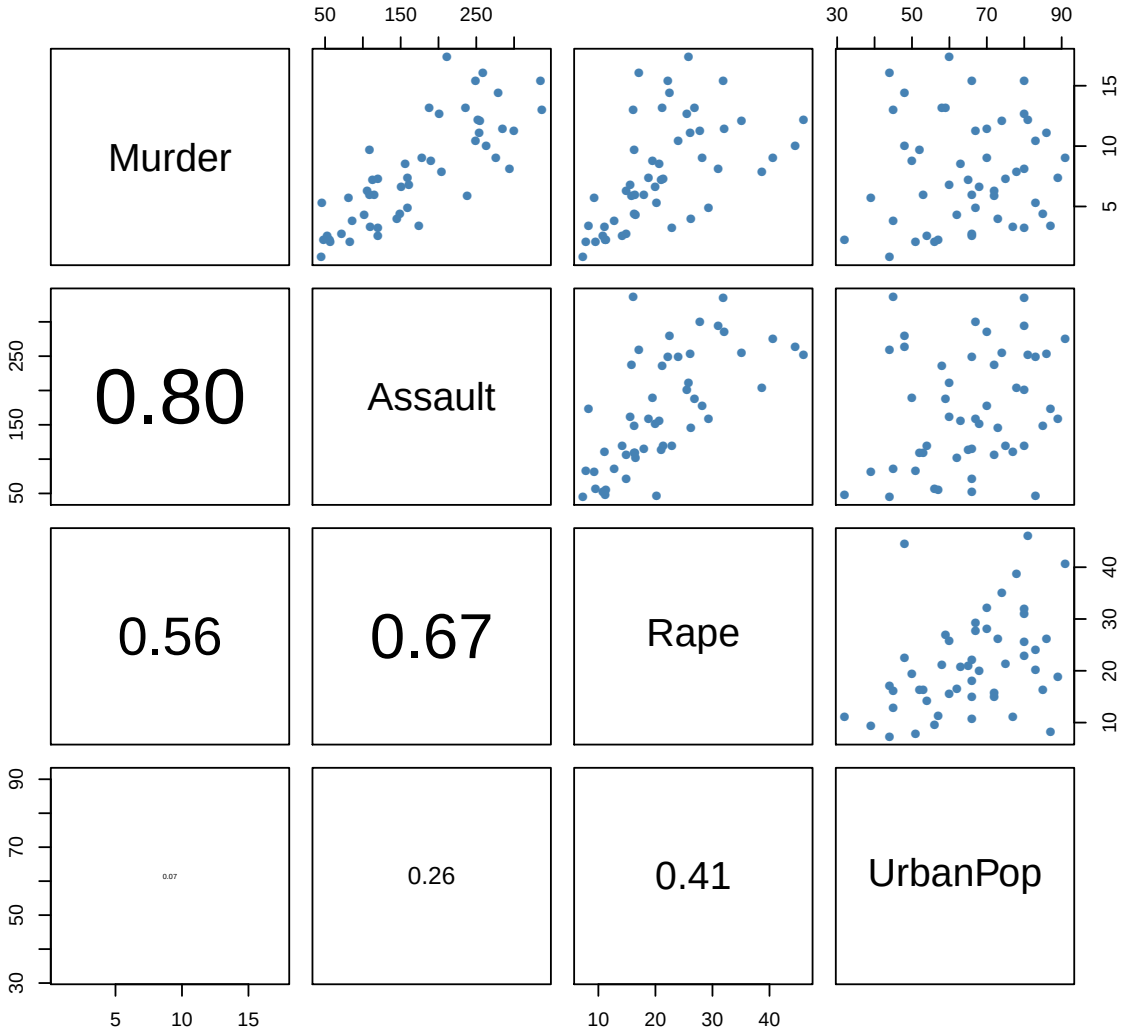
Scores

	Comp.1	Comp.2
Alabama	0.98	-1.12
Alaska	1.93	-1.06
Arizona	1.75	0.74
Arkansas	-0.14	-1.11
California	2.50	1.53
Colorado	1.50	0.98
⋮	⋮	⋮



- The word California is centered at $(2.5, 1.53)$ (in terms of the bottom and left side coordinate axes)

Why does PCA work well on this data?



In this pairs plot we can clearly see that the crime rate variables—`Murder`, `Assault`, and `Rape`—are highly correlated with one another. They provide redundant information. The first loading vector winds up forming a combination of these three features, essentially compressing 3 features into 1.

Proportion Variance Explained

- Dimensionality reduction techniques such as PCA work well when the data is essentially low dimensional
  - i.e., when there are many groups of highly correlated features

- i.e., We may have p features, but we might be able to describe the data with just $k \ll p$ linear combinations of them
  - For instance, if we have data on children, `height`, `weight` and `age` will all be highly correlated
  - PCA will be able to identify a linear combination of these features that we we'll roughly be able to interpret as the child's size

- In general, to understand how well PCA is doing, we look at the proportion of variance explained (PVE) of each principal component.

Proportion Variance Explained

- Assume as usual that all the variables have been centered to have mean 0.
- The total variance present in the data is defined as

$$\sum_{j=1}^{p} var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

and the variance explained by the mth principal component is

$$var(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

- The PVE of the mth component is the ratio of these quantities:

$$\text{PVE}(Z_m) = \frac{var(Z_m)}{\text{total variance}}$$

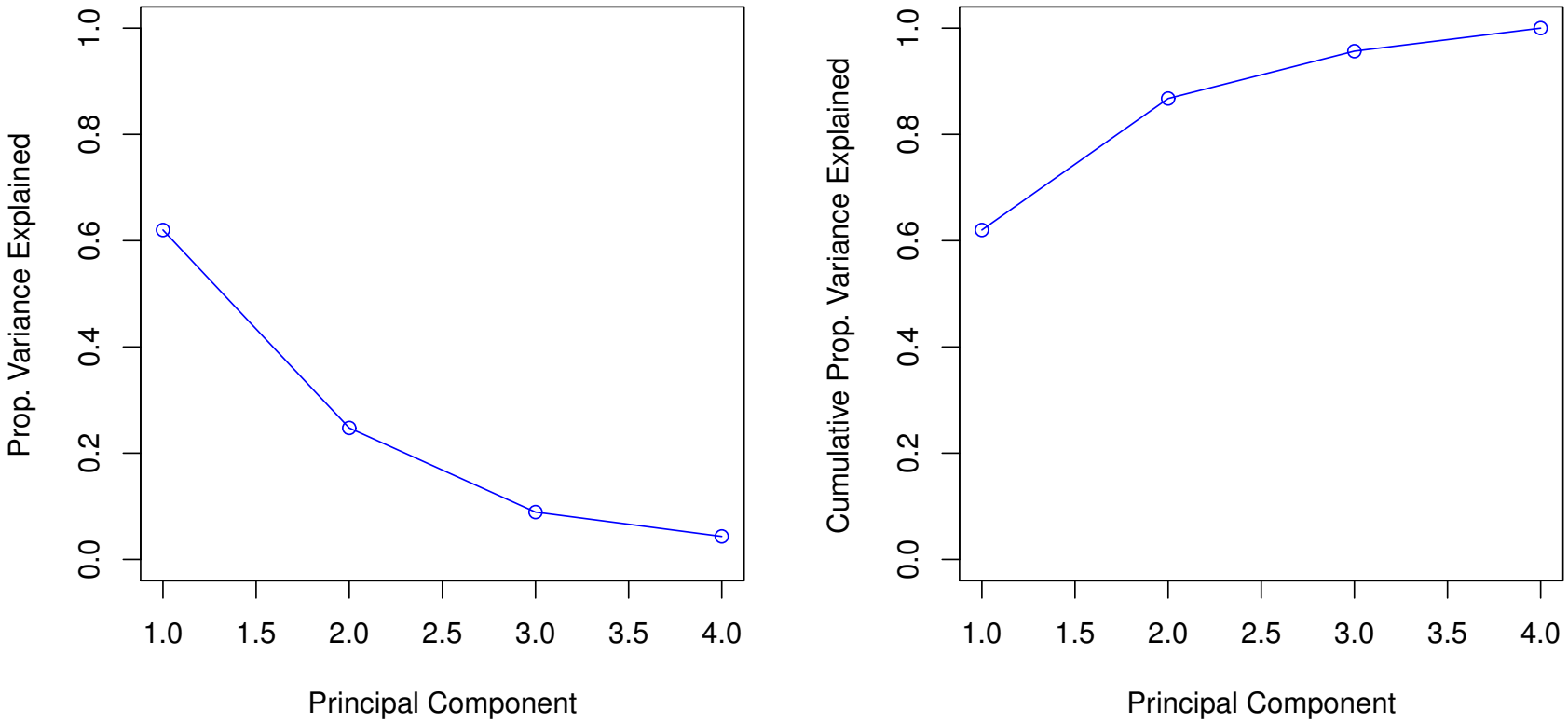


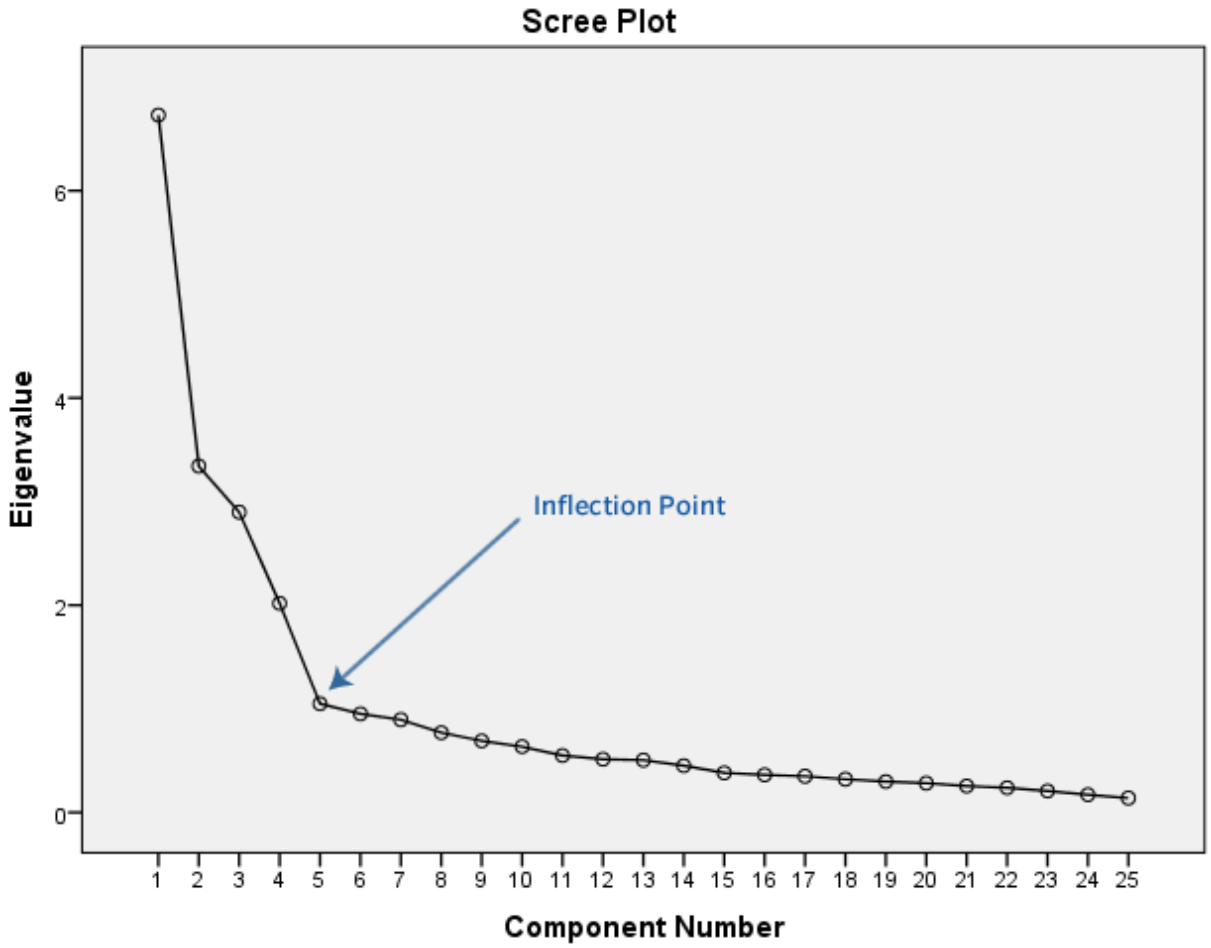
Figure 10.4 from ISL

- Left panel shows $\mathrm{PVE}(Z_m)$ for all 4 principal components in the `USArrests` data
- Right panel shows cumulative PVE: i.e., values of $\sum_{m \leq k} \mathrm{PVE}(Z_m)$ for $k = 1, 2, 3, 4$

How many principal components should we use?

- There's no simple answer to this question
- Cross-validation is not available for this problem
  - CV allows us to estimate test error... but we're doing unsupervised learning here and we don't really have a notion of *test error* to work with
  - If we treated our principal components as *derived features* in a regression or classification task, we could certainly run Cross-validation in that setting
- Often, people like to look at so-called scree plots, which is what we showed on the previous slide

Finding elbows in scree plots



[source: https://gugginotes.wordpress.com/]

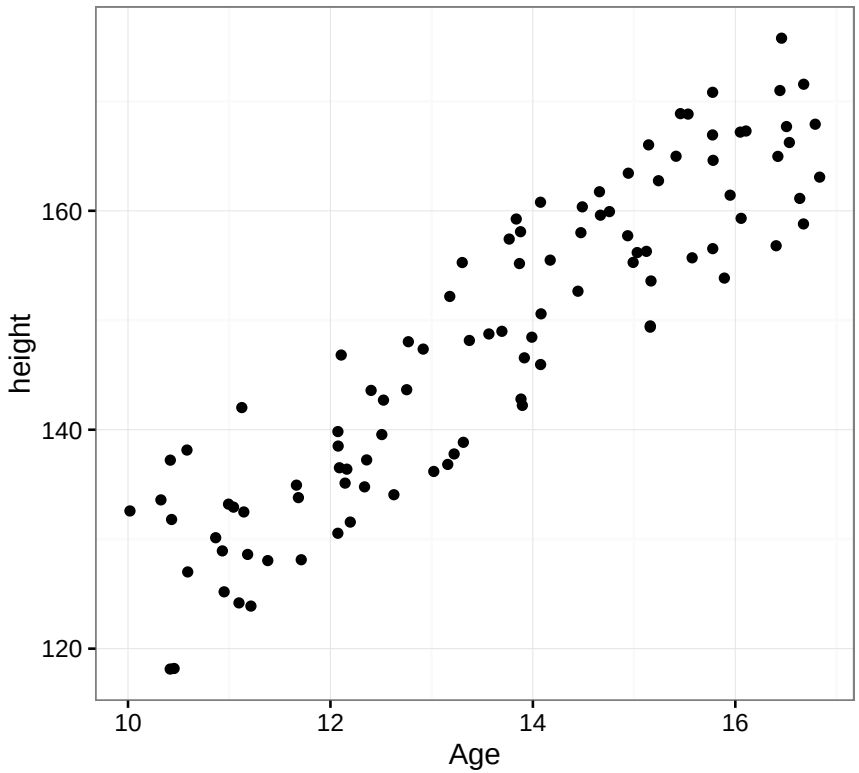
- *Eigenvalue* y-axis label should be interpreted as PVE
- Rule-of-thumb: Stop at the *elbow* in the Scree plot. ($k = 5$ here)

Principal Components Regression

- Suppose we're back in the supervised learning setting where we have observations (x_i, y_i)
- We have a lot of feature (p is large), and we suspect that many of them may be redundant/highly correlated
- We can perform PCA on the data matrix $\mathbf{X}$, treating this as a feature engineering step
- This will give us $k < p$ new features $z_1, \ldots, z_k$, corresponding to the top k principal components
- Then we can model y on $z_1, \ldots, z_k$ instead of on the x_j
- This method is called Principal Components Regression (PCR)
  - Of course, there's a classification version of PCR as well

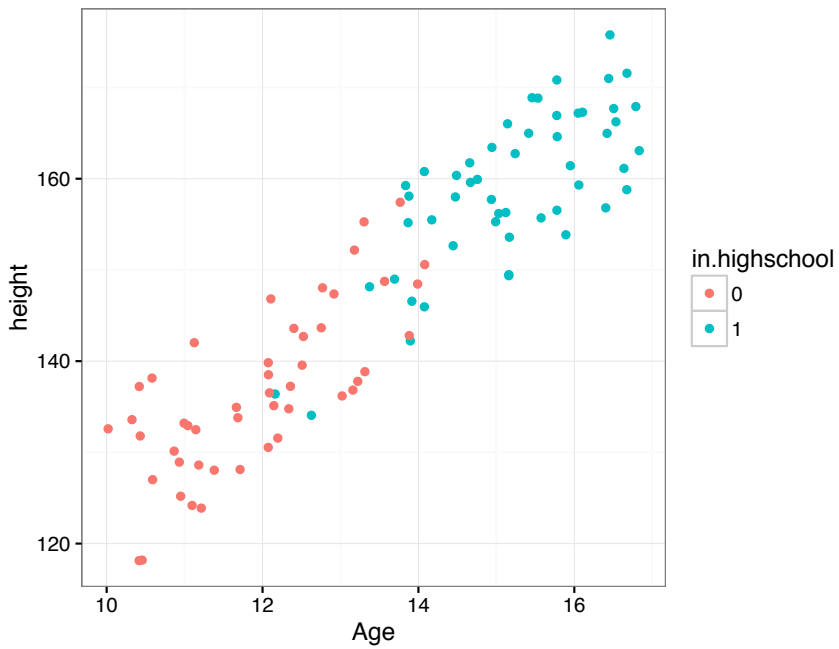
Simple PCR example

- Suppose that we have two features: `age` and `height`
- Everyone in our sample is between 10 and 17 years old, so these features are highly correlated
- Instead of using both features in our models, we could just use the 1st principal component.



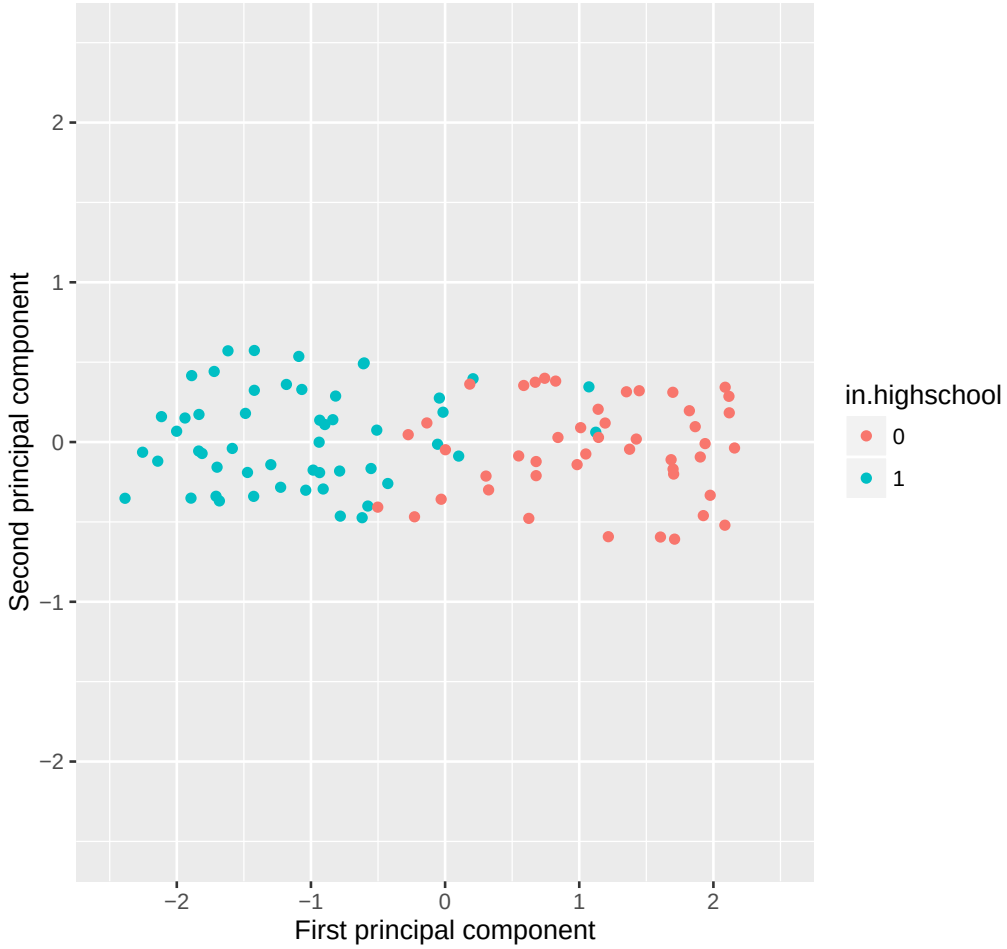
PCR Success case

- Will PCR work? This depends on the outcome, y
- Let's look at a classification setting where $y = 1$ if the individual is in high school, and 0 otherwise.



- Using just the first principal component for classification will work really well here!

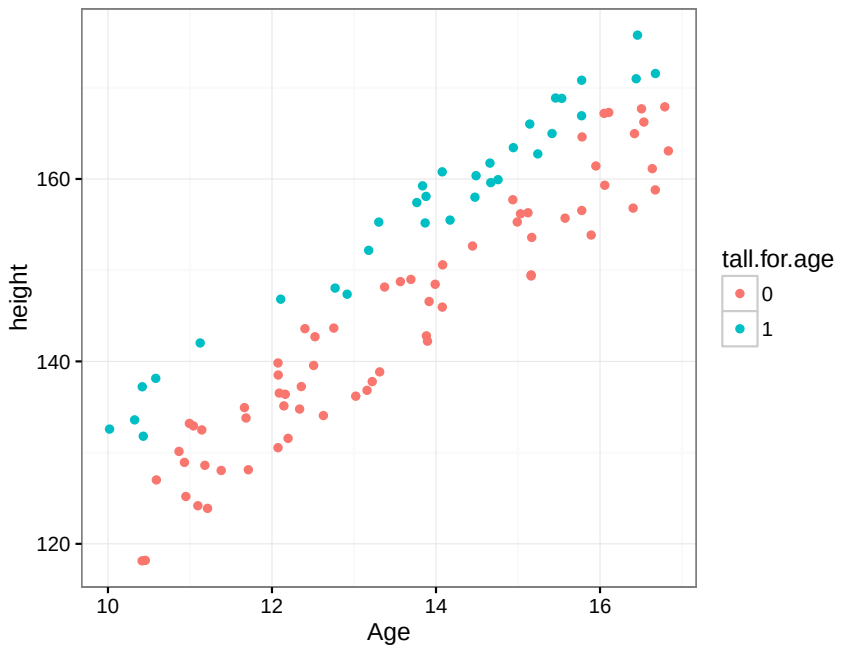
PCR Success case



- This is what the data looks like when plotted in terms of the two principal components
- We can clearly see that a logistic model with $\mathrm{y} \sim z_1$ will classify really well

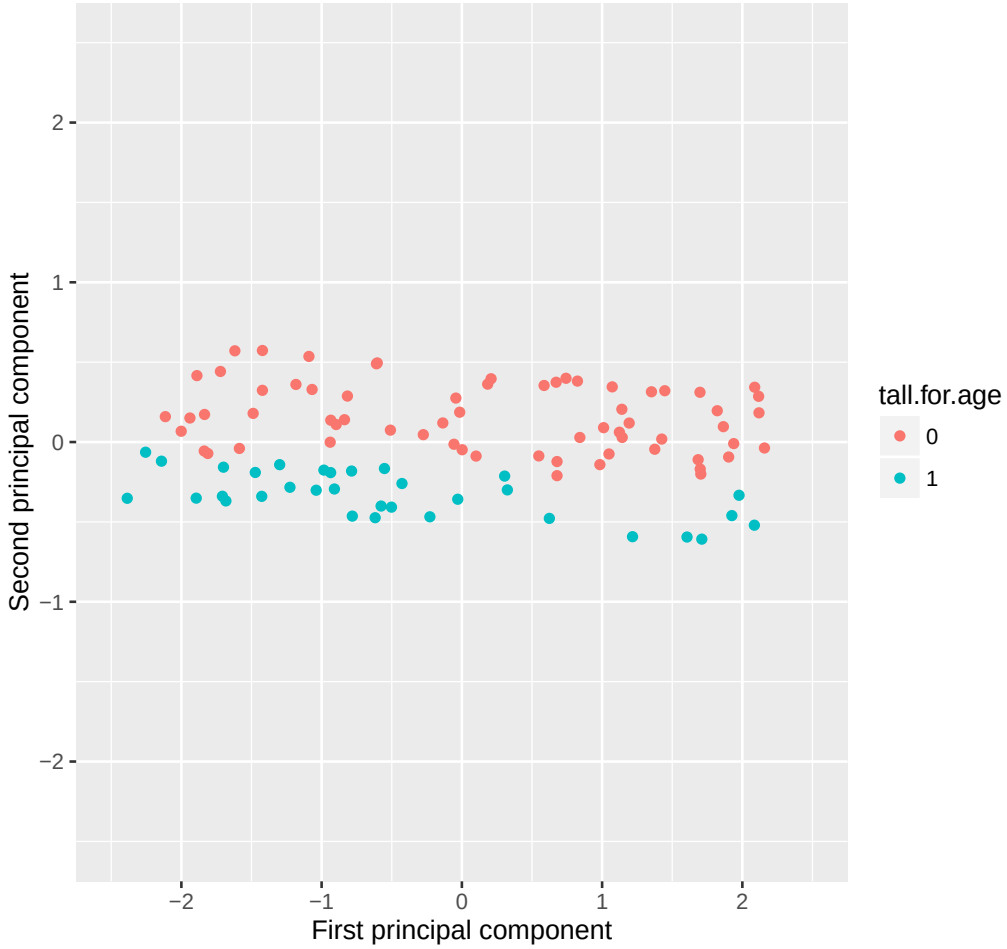
PCR Failure case

- Now what if our outcome was instead $y = 1$ if the person is *tall for their age*, and 0 otherwise.
- Here's a scatterplot of the data, colour-coded by outcome.



- The first principal component is going to be entirely orthogonal to the interesting direction for classification!

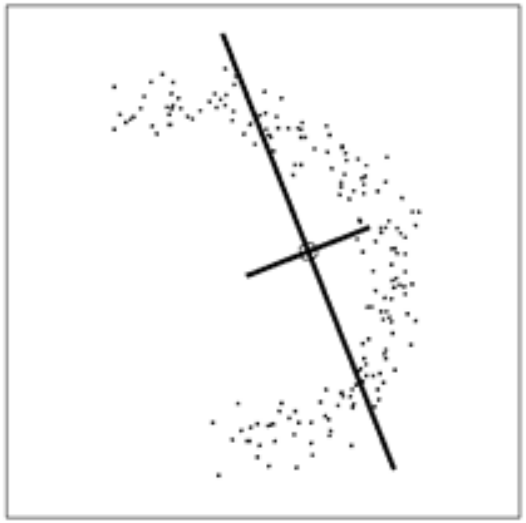
PCR Failure case



- A logistic model with $\mathrm{y} \sim z_1$ will fail completely!
- Take-away: PCR will work well when the leading principal components define directions that capture variation in the outcome y

PCA captures linear directions of variation

- PCA works well when the relationships between the features are linear (e.g., when features are linearly correlated)
- Shown below is an example where the two axes are clearly strongly associated, but the association is non-linear
- The PCA directions are reasonable... but don't *really* capture the key trend in the data
- Principal curves can be applied in such settings: This method generalizes PCA by fitting 1-dimensional *curves* instead of *lines*



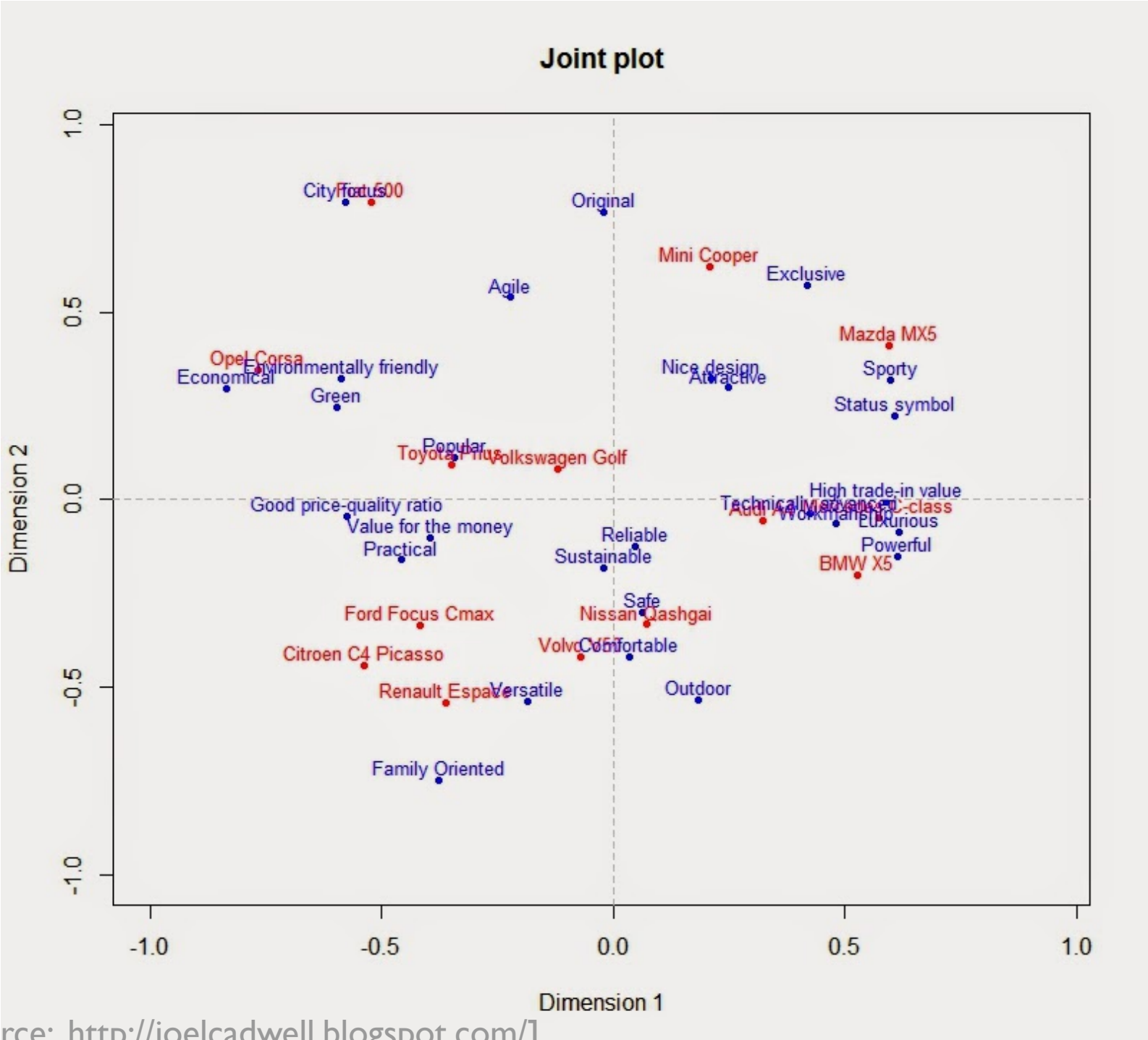
PCA

Principal curve

[img source: http://what-when-how.com/]

Correspondence Analysis

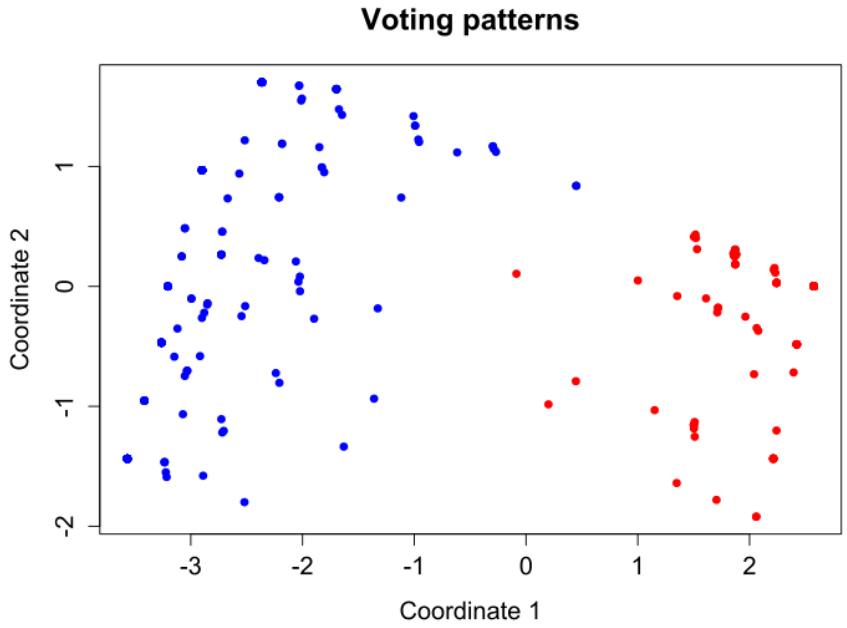
- PCA is great and widely used
- But it's limited to numeric or *ordinal* data
- We may want to perform dimensionality reduction with categorical features
- Correspondence analysis is an extension of PCA that gracefully handles categorical features
- The figure on the next page shows a biplot obtained by running Correspondence analysis on a dataset where consumers were asked to check whether or not they associated particular attributes (blue points) with various car models (red points)



[source: http://joelcadwell.blogspot.com/]

Multidimensional scaling

- Multidimensional scaling (MDS) is a dimensionality reduction methods for visualizing the level of similarity among individuals in a dataset
- Instead of feeding in the data set $\mathbf{X}$, we feed in a distance matrix specifying the pairwise distances between all observations
  - Recall: For hierarchical clustering, we also don't need $\mathbf{X}$. We just operate on the distance matrix
- Here's MDS output from an analysis of voting similarity between Republicans and Democrats (blue) in the house of representatives



Multidimensional scaling: US cities

- Here's an example of what MDS would do if applied to a matrix giving pairwise distances between all of the "major" cities in the US

- Awesome! It doesn't get the right rotation, but we can't expect it to. The reconstruction is otherwise excellent.

Association rules (Market Basket Analysis)

- Association rule learning has both a supervised and unsupervised learning flavour
- We didn't discuss the supervised version when we were talking about regression and classification, but you should know that it exists.
  - Look up: Apriori algorithm (Agarwal, Srikant, 1994)
  - In **R**: `apriori` from the `arules` package
- Basic idea: Suppose you're consulting for a department store, and your client wants to better understand patterns in their customers' purchases
- patterns or rules look something like:

$$\{\texttt{suit}, \texttt{belt}\} \Rightarrow \{\texttt{dress shoes}\}$$

$$\underbrace{\{\texttt{bath towels}\}}_{\text{LHS}} \Rightarrow \underbrace{\{\texttt{bed sheets}\}}_{\text{RHS}}$$

  - In words: People who buy a new `suit` and `belt` are more likely to also by `dress shoes`.
  - People who by `bath towels` are more likely to buy `bed sheets`

Basic concepts

- Association rule learning gives us an automated way of identifying these types of patterns
- There are three important concepts in rule learning: support, confidence, and lift
- The support of an *item* or an *item set* is the fraction of transactions that contain that item or item set.
  - We want rules with high support, because these will be applicable to a large number of transactions
  - {`suit`, `belt`, `dress shoes`} likely has sufficiently high support to be interesting
  - {`luggage`, `dehumidifer`, `teapot`} likely has low support
- The confidence of a rule is the probability that a new transaction containing the LHS item(s) {`suit`, `belt`} will also contain the RHS item(s) {`dress shoes`}
- The lift of a rule is

$$\frac{\text{support(LHS, RHS)}}{\text{support(LHS)} \cdot \text{support(RHS)}} = \frac{\mathbb{P}(\{\texttt{suit}, \texttt{belt}, \texttt{dress shoes}\})}{\mathbb{P}(\{\texttt{suit}, \texttt{belt}\})\mathbb{P}(\{\texttt{dress shoes}\})}$$

An example

```
> a_list<-list(
+   c("CrestTP","CrestTB"),
+   c("OralBTB"),
+   c("BarbSC"),
+   c("ColgateTP","BarbSC"),
+   c("OldSpiceSC"),
+   c("CrestTP","CrestTB"),
+   c("AIMTP","GUMTB","OldSpiceSC"),
+   c("ColgateTP","GUMTB"),
+   c("AIMTP","OralBTB"),
+   c("CrestTP","BarbSC"),
```

- A subset of drug store transactions is displayed above
- First transaction: Crest ToothPaste, Crest ToothBrush
- Second transaction: OralB ToothBrush
- etc…

[source: Stephen B. Vardeman, STAT502X at Iowa State University]

```
> rules<-apriori(trans,parameter=list(supp=.02, conf=.5, target="rules"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE    0.02      1     10  rules FALSE
```

- This says: Consider only those rules where the item sets have support *at least* 0.02, and confidence *at least* 0.5
- Here's what we wind up with

```
> inspect(head(sort(rules,by="lift"),n=20))
  lhs                rhs           support confidence     lift
1 {GilletteSC} => {ColgateTP}      0.03  1.0000000 20.00000
2 {ColgateTP}  => {GilletteSC}     0.03  0.6000000 20.00000
3 {CrestTB}    => {CrestTP}        0.03  0.7500000 15.00000
4 {CrestTP}    => {CrestTB}        0.03  0.6000000 15.00000
5 {GUMTB}      => {AIMTP}          0.02  0.6666667 13.33333
```

[source: Stephen B. Vardeman, STAT502X at Iowa State University]

Acknowledgements

All of the lectures notes for this class feature content borrowed with or without modification from the following sources:

- 36-462/36-662 Lecture notes (Prof. Tibshirani, Prof. G'Sell, Prof. Shalizi)
- 95-791 Lecture notes (Prof. Dubrawski)
- *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani
- *Applied Predictive Modeling*, (Springer, 2013), Max Kuhn and Kjell Johnson