Class 11

BUS 696

Prof. Jonathan Hersh

BUS 696: Class 11 Announcements

1. Final Projects
    - Nov 14th summary stats due
    - Must summarize training data -> having performed feature transformation and cleaning
2. Next Class
    - “appetizers” at Provisions

BUS 696: Class 11 Outline

1. HBR AI Articles
2. Merging Datasets
3. Cleaning Factor Levels
4. Sentiment Data
5. Unsupervised Learning
    - K-means Clustering
    - Hierarchical Clustering
    - Principal Component Analysis

Merging Datasets – Primary Keys

Artist Table

Table Key

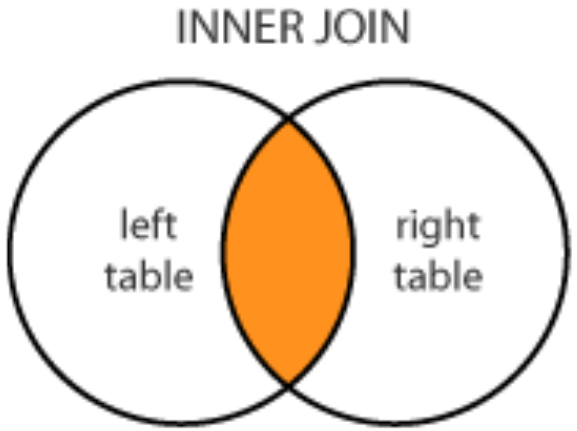
Artist	Artist ID
Justin Bieber	1
Drake	2
The Smiths	3

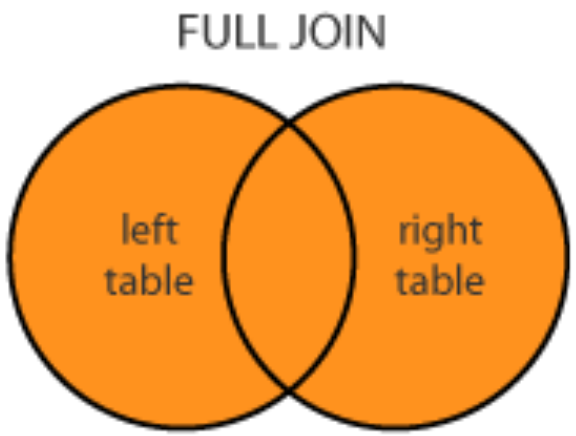
- Real world datasets are stored in separate tables to save storage space
- The unique (row) identifier for every observation in a table is called the **primary key** (or primary keys)

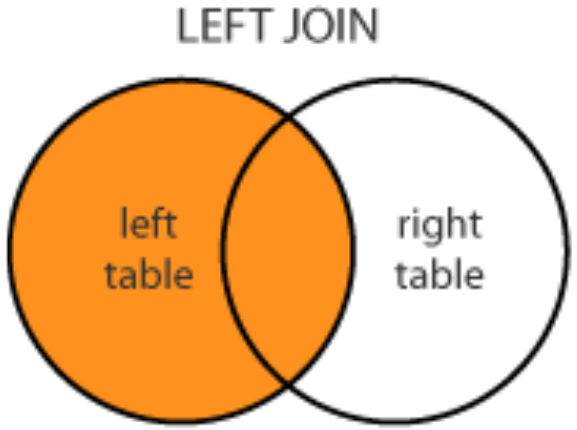
- Album Table

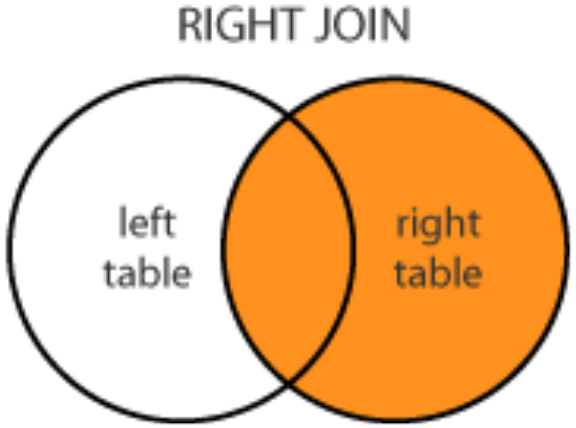
Album Name	Table Key Album ID	Artist ID
Thank Me Later	1	2
Take Care	2	2
Nothing Was The Same	3	2
Views	4	2
Scorpion	5	2
Strangeways Here We Come	6	3

Merge Datasets – Types of Joins









Let’s get some dummy data for merging

```r
library(tidyverse)

artist_DF <- data.frame(
  artist_name = c("Justin Bieber",
                  "Drake",
                  "The Smiths"),
  artist_ID = 1:3
)
```

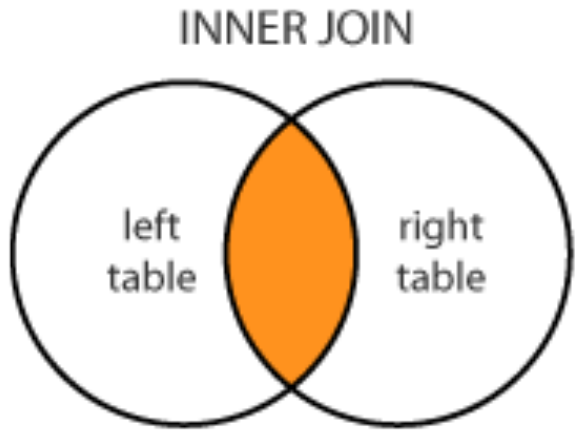
```r
> artist_DF
          artist artist_ID
1 Justin Bieber         1
2          Drake         2
3     The Smiths         3
```

```r
album_DF <- data.frame(
  album_name = c("Thank Me Later", "Take Care",
                 "Nothing Was The Same", "Views",
                 "Scorpion",
                 "Strangeways Here We Come",
                 "Thank U Next"),
  album_ID = 1:7,
  artist_ID = c(rep(2,5),3,4)
)
```

```r
> album_DF
                album_name album_ID artist_ID
1           Thank Me Later        1         2
2                Take Care        2         2
3     Nothing Was The Same        3         2
4                    Views        4         2
5                 Scorpion        5         2
6 Strangeways Here We Come        6         3
7           Thank U Next        7         4
```

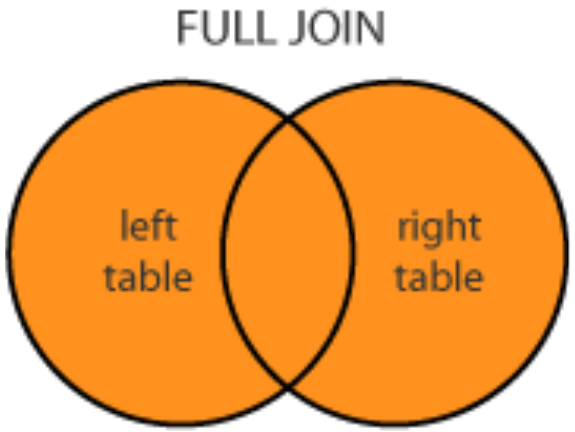
Inner Join: Returns All Albums That Match on Artist ID

```
# inner join
DF <- inner_join(x = artist_DF,
                 y = album_DF,
                 by = "artist_ID")
```




```
> merged_DF
      artist artist_ID                album_name album_ID
1      Drake         2            Thank Me Later        1
2      Drake         2                Take Care         2
3      Drake         2     Nothing Was The Same         3
4      Drake         2                     Views        4
5      Drake         2                  Scorpion        5
6 The Smiths         3 Strangeways Here We Come         6
```

Full Join: Returns All Albums and All Artists, Regardless of Match

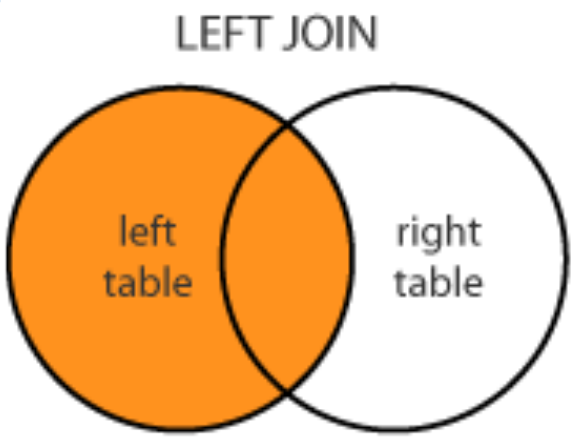


"NA"s introduced because information unknown across tables

```
# full join
merged_DF <-
  full_join(x = artist_DF,
            y = album_DF,
            by = "artist_ID")

merged_DF
```

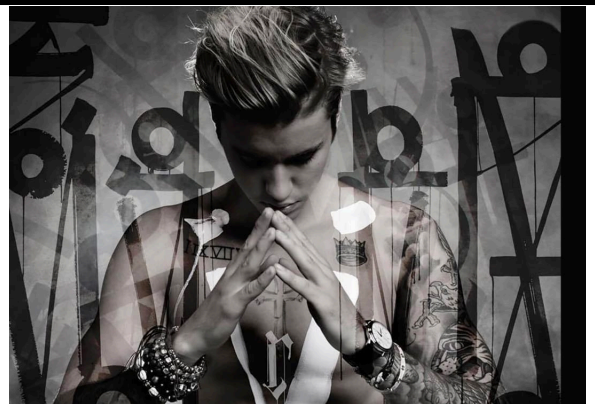
```
> merged_DF
         artist artist_ID                album_name album_ID
1 Justin Bieber         1                      <NA>       NA
2         Drake         2         Thank Me Later          1
3         Drake         2              Take Care          2
4         Drake         2  Nothing Was The Same           3
5         Drake         2                 Views           4
6         Drake         2               Scorpion          5
7    The Smiths         3 Strangeways Here We Come         6
8          <NA>         4          Thank U Next           7
```

Left Join: Returns All Artists, and Matching Album ID Information



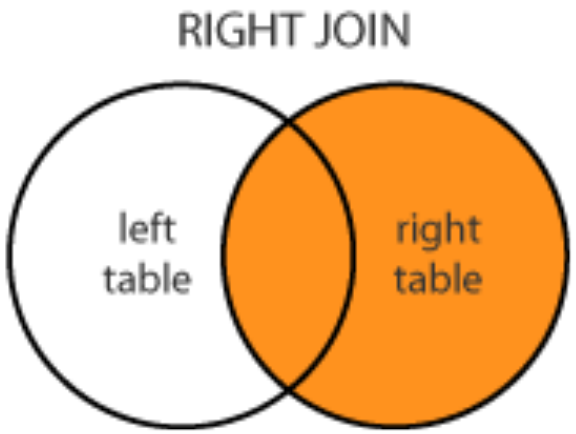
"NA"s for Justin's album because no information in album table on his masterpieces

```
# left join
merged_DF <-
  left_join(x = artist_DF,
            y = album_DF,
            by = "artist_ID")
```



```
> merged_DF
         artist artist_ID                album_name album_ID
1 Justin Bieber         1                      <NA>       NA
2         Drake         2         Thank Me Later          1
3         Drake         2             Take Care           2
4         Drake         2  Nothing Was The Same           3
5         Drake         2                 Views           4
6         Drake         2              Scorpion           5
7    The Smiths         3 Strangeways Here We Come         6
```

Right Join: Returns All Albums and Matching Artist Information




```
# right join
merged_DF <-
  right_join(x = artist_DF,
             y = album_DF,
             by = "artist_ID")
```

```
> merged_DF
      artist artist_ID                album_name album_ID
1      Drake         2          Thank Me Later        1
2      Drake         2               Take Care        2
3      Drake         2   Nothing Was The Same        3
4      Drake         2                  Views        4
5      Drake         2               Scorpion        5
6 The Smiths         3 Strangeways Here We Come        6
7       <NA>         4             Thank U Next        7
```

"NA"s for albums that don't have an artist match

BUS 696: Class 10 Outline

1. HBR AI Articles
2. Merging Datasets
3. Cleaning Factor Levels
4. Sentiment Data
5. Unsupervised Learning
    - K-means Clustering
    - Hierarchical Clustering
    - Principal Component Analysis

Wine Labels Example

```r
#-----------------------------------------------
# Factor Levels - Wine Labels Example
#-----------------------------------------------
# see more on forcats https://forcats.tidyverse.org/
# recoding factor labels by hand
library('tidyverse')
WineDF <- read_csv(here::here("datasets","WineExample.csv"))
glimpse(WineDF)
summary(WineDF)

# View(WineDF)
table(WineDF$Variety, WineDF$Color)
```

```
> table(WineDF$Variety, WineDF$Color)

     RED WHITE
  V1  10     0
  V2   5     0
  V3   4     0
  V4   1     0
  V5   0     9
  V6   0     3
  V7   0     2
  V8   0     1
```

Relabeling Factor Levels Manually Using fct_recode()

```
# relabel by hand but only have to relabel changes
library('forcats')
WineDF <- WineDF %>%
  mutate(Variety_lbl1 = fct_recode(Variety,
                                   "OtherRed" = "V3",
                                   "OtherRed" = "V4",
                                   "OtherWhite" = "V7",
                                   "OtherWhite" = "V8"))

table(WineDF$Variety, WineDF$Variety_lbl1)
```

```
> table(WineDF$Variety, WineDF$Variety_lbl1)

    V1 V2 OtherRed V5 V6 OtherWhite
  V1 10  0        0  0  0          0
  V2  0  5        0  0  0          0
  V3  0  0        4  0  0          0
  V4  0  0        1  0  0          0
  V5  0  0        0  9  0          0
  V6  0  0        0  0  3          0
  V7  0  0        0  0  0          2
  V8  0  0        0  0  0          1
```

“Lumping” Factors Into Most Common and Creating an “Other” Category for the Rest

```r
# lump into top 5 factors
WineDF <- WineDF %>%
  mutate(Variety_lbl3 = fct_lump(Variety, 5))

table(WineDF$Variety, WineDF$Variety_lbl3)
```

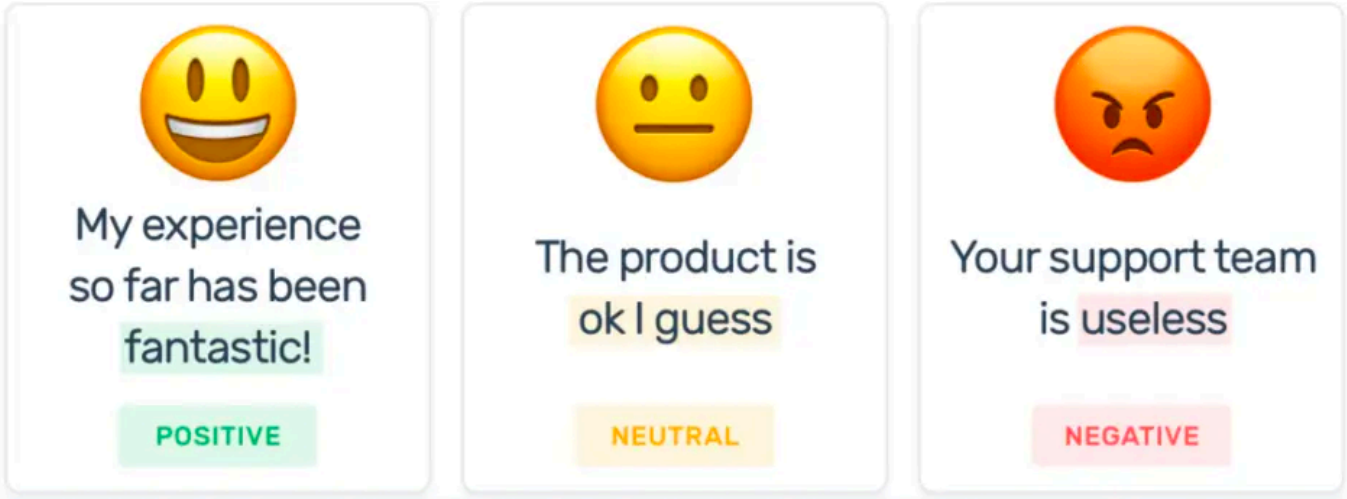
```
> table(WineDF$Variety, WineDF$Variety_lbl3)

    V1 V2 V3 V5 V6 Other
  V1 10  0  0  0  0     0
  V2  0  5  0  0  0     0
  V3  0  0  4  0  0     0
  V4  0  0  0  0  0     1
  V5  0  0  0  9  0     0
  V6  0  0  0  0  3     0
  V7  0  0  0  0  0     2
  V8  0  0  0  0  0     1
```

BUS 696: Class 10 Outline

1. HBR AI Articles
2. Merging Datasets
3. Cleaning Factor Levels
4. Sentiment Data
5. Unsupervised Learning
    - K-means Clustering
    - Hierarchical Clustering
    - Principal Component Analysis

Word Sentiments

- Existing Libraries have mapped each word to a given “sentiment”
- Sentiment is the average feeling invoked by a word



Extracting Sentiments from Words Using 'Sentimentr'

```
#-------------------------------------------------
# Text Sentiment
#-------------------------------------------------
library('devtools')
devtools::install_github("trinker/sentimentr")


# average sentiment score
library('sentimentr')
library('tidyverse')

sentiment_by("I am very scared of the dark")

sentiment('I was very scared of the dark.
          Now I glow at night and life is amazing')
```

```
> sentiment_by("I am very scared of the dark")
   element_id word_count sd ave_sentiment
1:          1          7 NA    -0.7483697
> sentiment('I was very scared of the dark.
+           Now I glow at night and life is amazing')
   element_id sentence_id word_count  sentiment
1:          1           1          7 -0.7483697
2:          1           2          9  0.2500000
```

- Sentence 1 has negative sentiment
- Sentence 2 has positive sentiment

Which Words Affect Sentiment?

```
"I am very scared of the dark" %>%
  extract_sentiment_terms()

"Now I glow at night and life is amazing" %>%
  extract_sentiment_terms()

"and I was like baby, baby, baby oh" %>%
  extract_sentiment_terms()
```

```
> "I am very scared of the dark" %>%
+   extract_sentiment_terms()
   element_id sentence_id    negative
1:          1           1 scared,dark
>
> "Now I glow at night and life is amazing" %>%
+   extract_sentiment_terms()
   element_id sentence_id     positive
1:          1           1 glow,amazing
>
> "and I was like baby, baby, baby oh" %>%
+   extract_sentiment_terms()
   element_id sentence_id       positive
1:          1           1 baby,baby,baby
```

Visualizing Sentiment Scores: Red Negative Sentiment Green Positive

```
She woke me up daily.
Don't need no Starbucks.
She made my heart pound.
And skip a beat when I see her in the street and.
At school on the playground.
But I really wanna see her on the weekend.
She know she got me dazin'.
'Cause she was so amazin'.
And now my heart is breakin'.
But I just keep on sayin'.
Baby, baby, baby oh.
Like baby, baby, baby no.
Like baby, baby, baby oh.
I thought you'd always be mine (mine).
Baby, baby, baby oh.
Like baby, baby, baby no.
Like baby, baby, baby ooh.
I thought you'd always be mine.
Now I'm gone.
Yeah, yeah, yeah.
Yeah, yeah, yeah (now I'm all gone).
Yeah, yeah, yeah.
Yeah, yeah, yeah (now I'm all gone).
Yeah, yeah, yeah.
Yeah, yeah, yeah (now I'm all gone).
Gone, gone, gone, I'm gone." %>%
  sentiment_by(by = NULL) %>% highlight()
```

1: ***+.470***

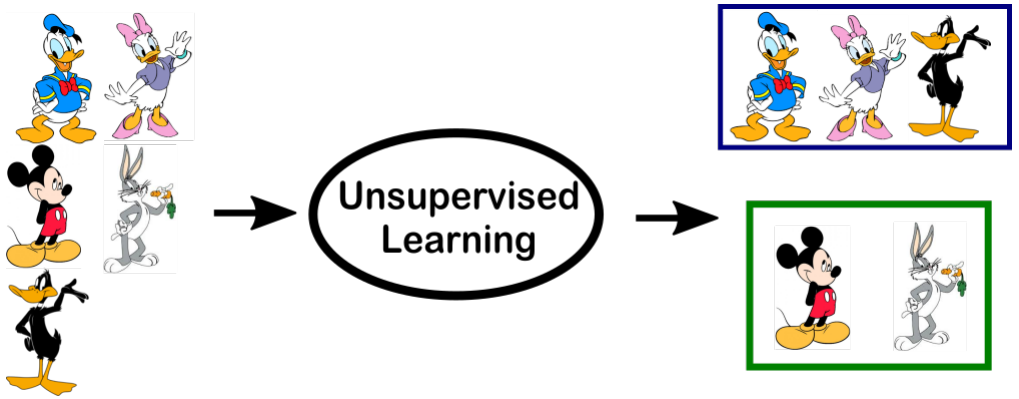
Ooh whoa, ooh whoa, ooh whoa. You know you love me, I know you care. Just shout whenever and I'll be there. You are my love, you are my heart. And we will never, ever, ever be apart. Are we an item? Girl quit playin'. We're just friends, what are you sayin'. Said there's another, look right in my eyes. My first love, broke my heart for the first time. And I was like baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby oh. I thought you'd always be mine (mine). Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby ooh. I thought you'd always be mine. Oh for you, I would have done whatever. And I just can't believe we ain't together. And I wanna play it cool. But I'm losin' you. I'll buy you anything. I'll buy you any ring. And I'm in pieces, baby fix me. And just shake me, 'til you wake me from this bad dream. I'm goin' down, down, down, down. And I can't just believe my first love won't be around. And I'm like baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby oh. I thought you'd always be mine (mine). Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby ooh. I thought you'd always be mine. Luda, when I was thirteen, I had my first love. There was nobody that compared to my baby. And nobody came between us no one could ever come above. She had me goin' crazy. Oh, I was starstruck. She woke me up daily. Don't need no Starbucks. She made my heart pound. And skip a beat when I see her in the street and. At school on the playground. But I really wanna see her on the weekend. She know she got me dazin'. 'Cause she was so amazin'. And now my heart is breakin'. But I just keep on sayin'. Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby oh. I thought you'd always be mine (mine). Baby, baby, baby oh. Like baby, baby, baby no. Like baby, baby, baby ooh. I thought you'd always be mine. Now I'm gone. Yeah, yeah, yeah. Yeah, yeah, yeah (now I'm all gone). Yeah, yeah, yeah. Yeah, yeah, yeah (now I'm all gone). Yeah, yeah, yeah. Yeah, yeah, yeah (now I'm all gone). Gone, gone, gone, I'm gone.

BUS 696: Class 10 Outline

1. HBR AI Articles
2. Merging Datasets
3. Cleaning Factor Levels
4. Sentiment Data
5. **Unsupervised Learning**
    - K-means Clustering
    - Hierarchical Clustering
    - Principal Component Analysis

What is unsupervised learning?

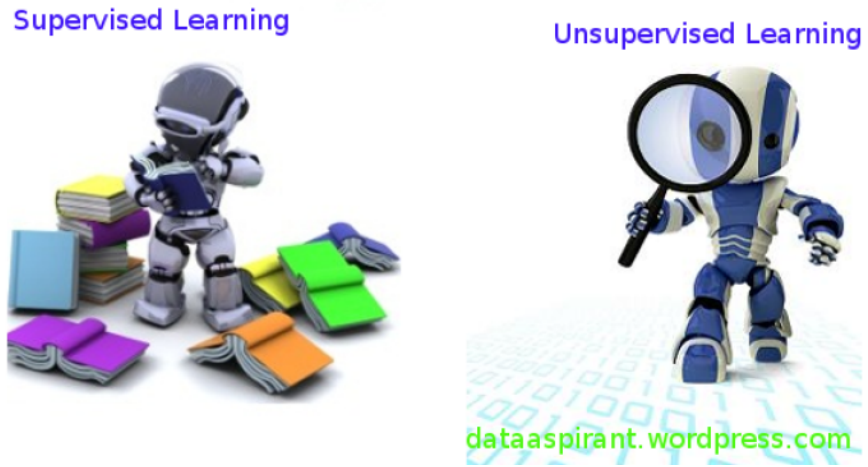
- All of the machine learning we've encountered so far has been supervised learning such as regression
- This last lecture will describe **unsupervised learning**
- In unsupervised learning, we observe x_1, x_2, x_p, features but we don't observe any Ys



Goals of unsupervised learning

- **Since we don't observe $Y's$, we can't predict anything**
- The goal is more subtle here: can we discover interesting patterns in the data? Can we discover useful subgroups?

- We'll discuss two methods
  - **clustering**: a broad class of methods for discovering unknown subgroups
  - **principal component analysis**: a tool used for data visualization and dimension reduction



Challenge of unsupervised learning

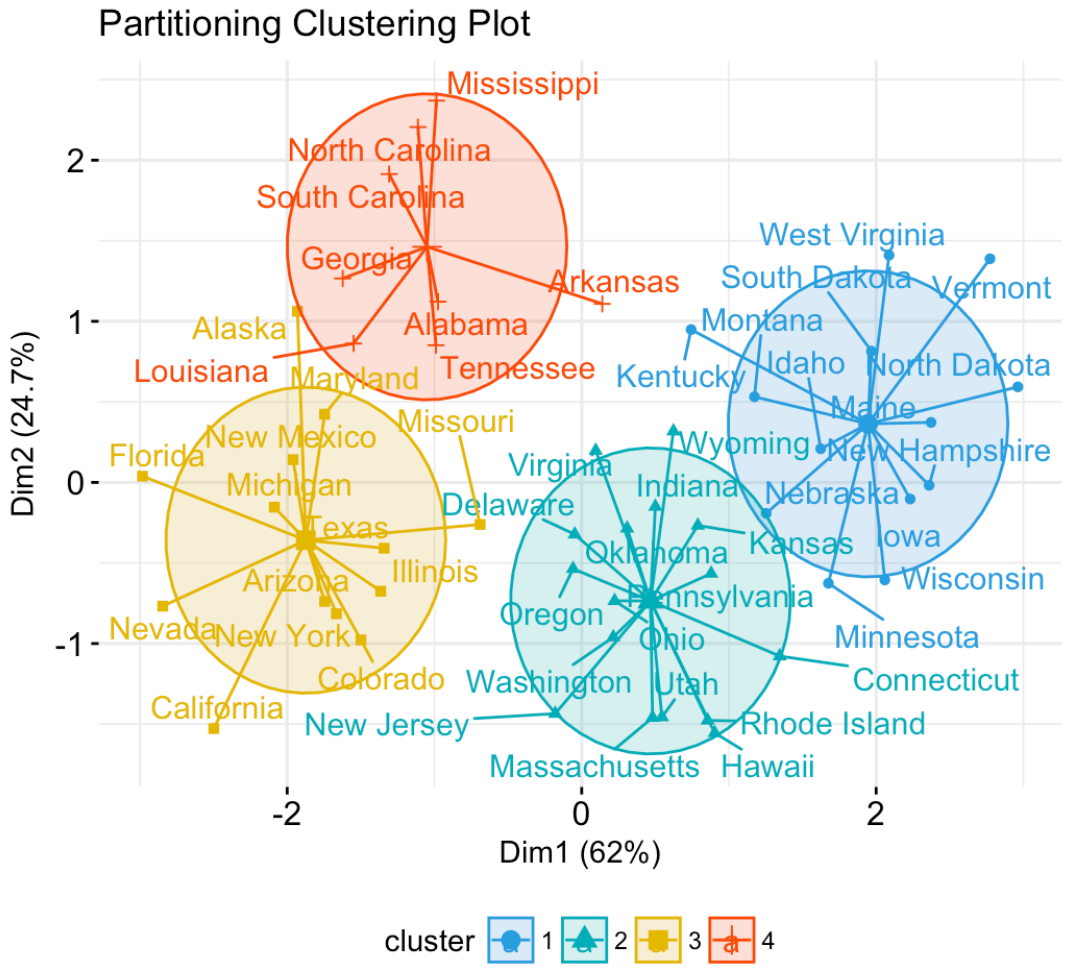
- **Because we have no “truth”, the end result is more subjective**

- **Some examples of unsupervised learning**
  - subgroup of breast cancer patients that by gene expression are drug resistant
  - groups of shoppers characterized by browsing and purchase histories
  - movies grouped by ratings assigned by movie viewers

K-means clustering as a “game”

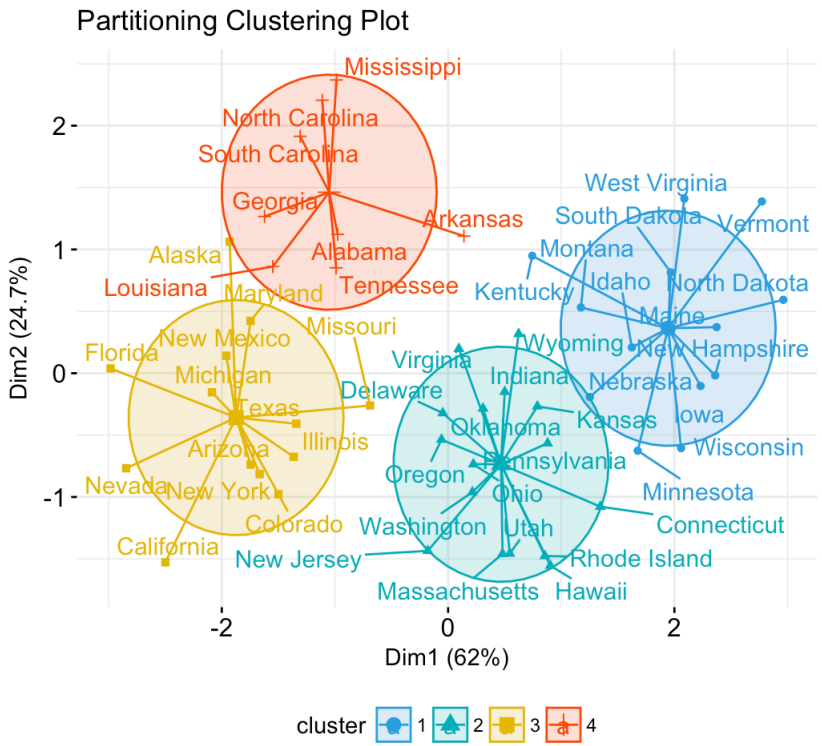
- Tell the computer how many groups (k) you think the data should be split into

- The computer splits the objects into k groups such that the groups are most similar

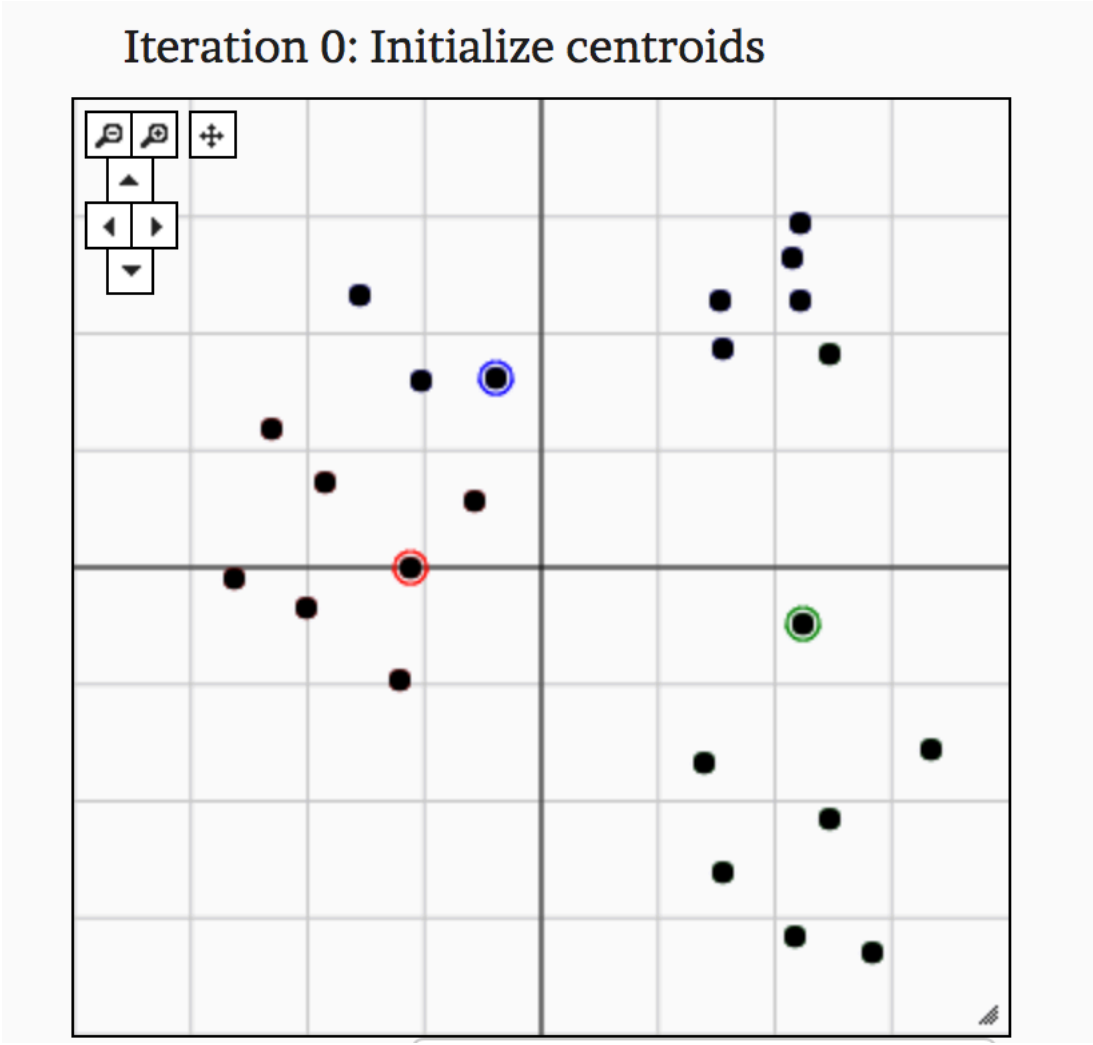


K-means clustering algorithm

1. Decide how many clusters we want. Call this *K*
2. Randomly assign a number, 1, . . . ,*K*, to each of the observations. (Initial cluster assignment)
3. Iterate until clusters stop changing:
   - **Expectation-step:** For each of the *K* clusters, compute the cluster centroid (center point, i.e. means for the *k*-th cluster)
   - **Maximization-step:** Assign each observation to the cluster whose centroid is closest (in Euclidean distance)

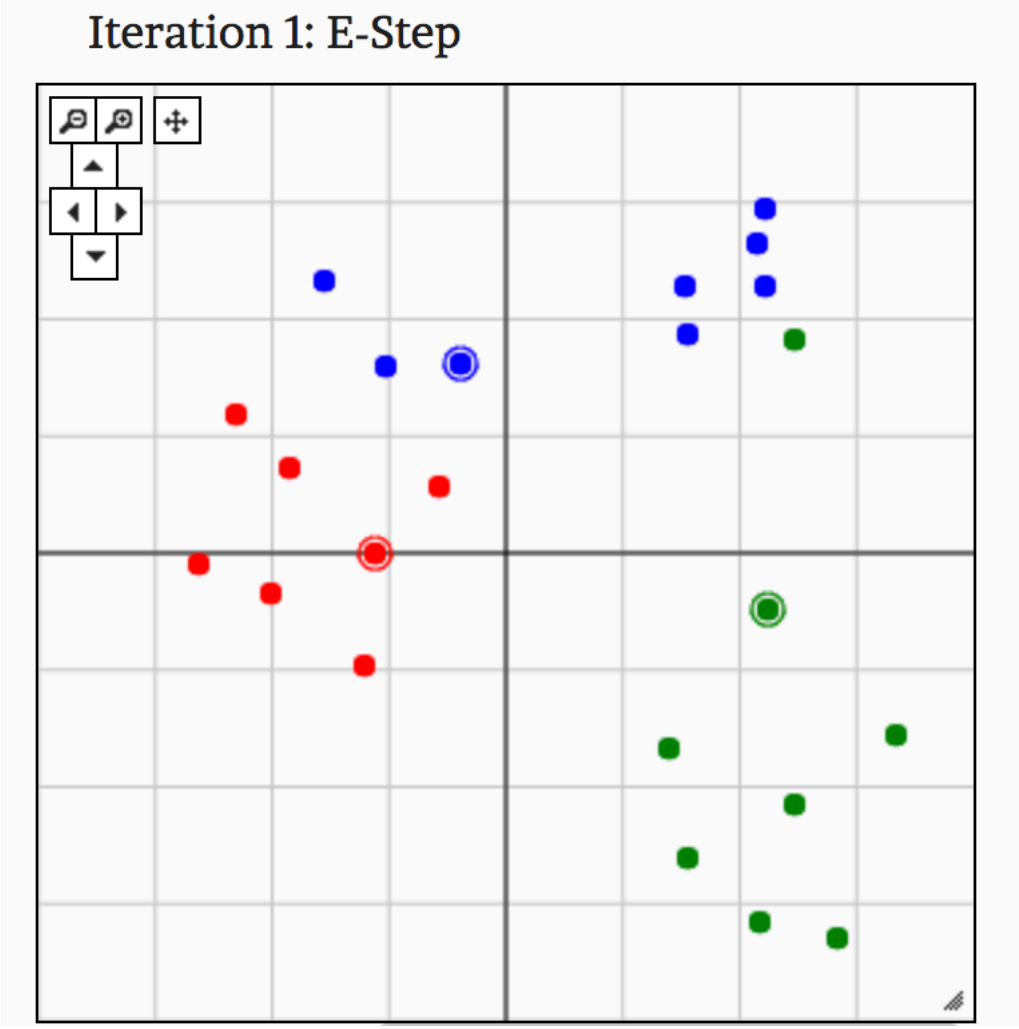


K-means clustering in action



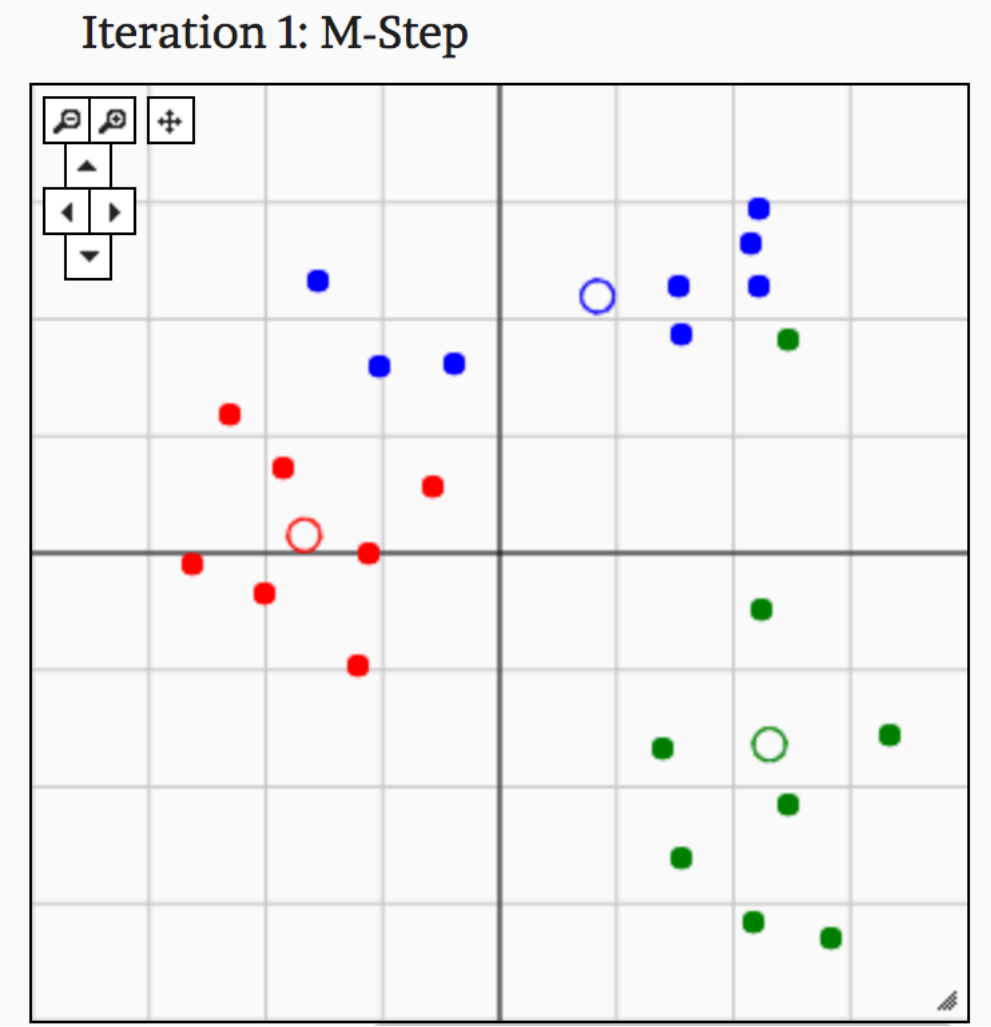
- From: http://util.io/k-means

K-means clustering in action



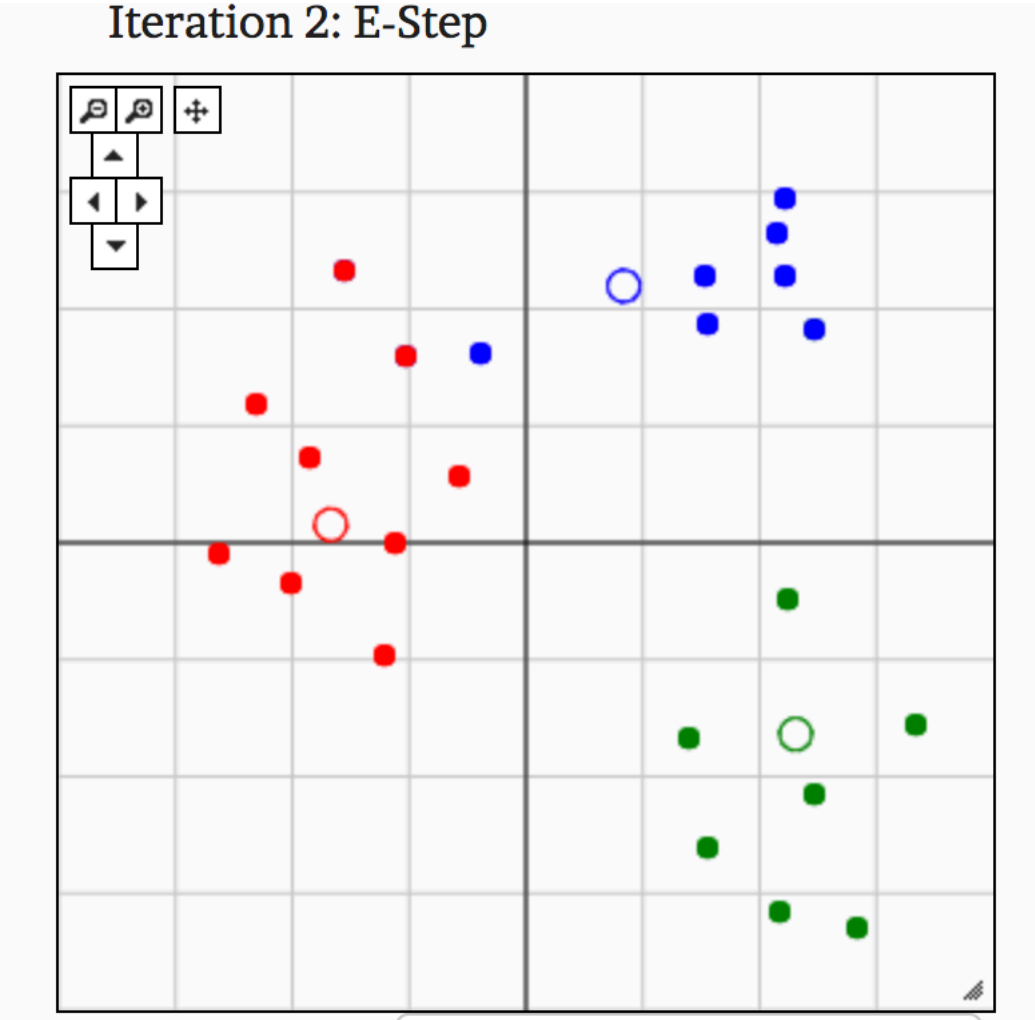
- From: http://util.io/k-means

K-means clustering in action



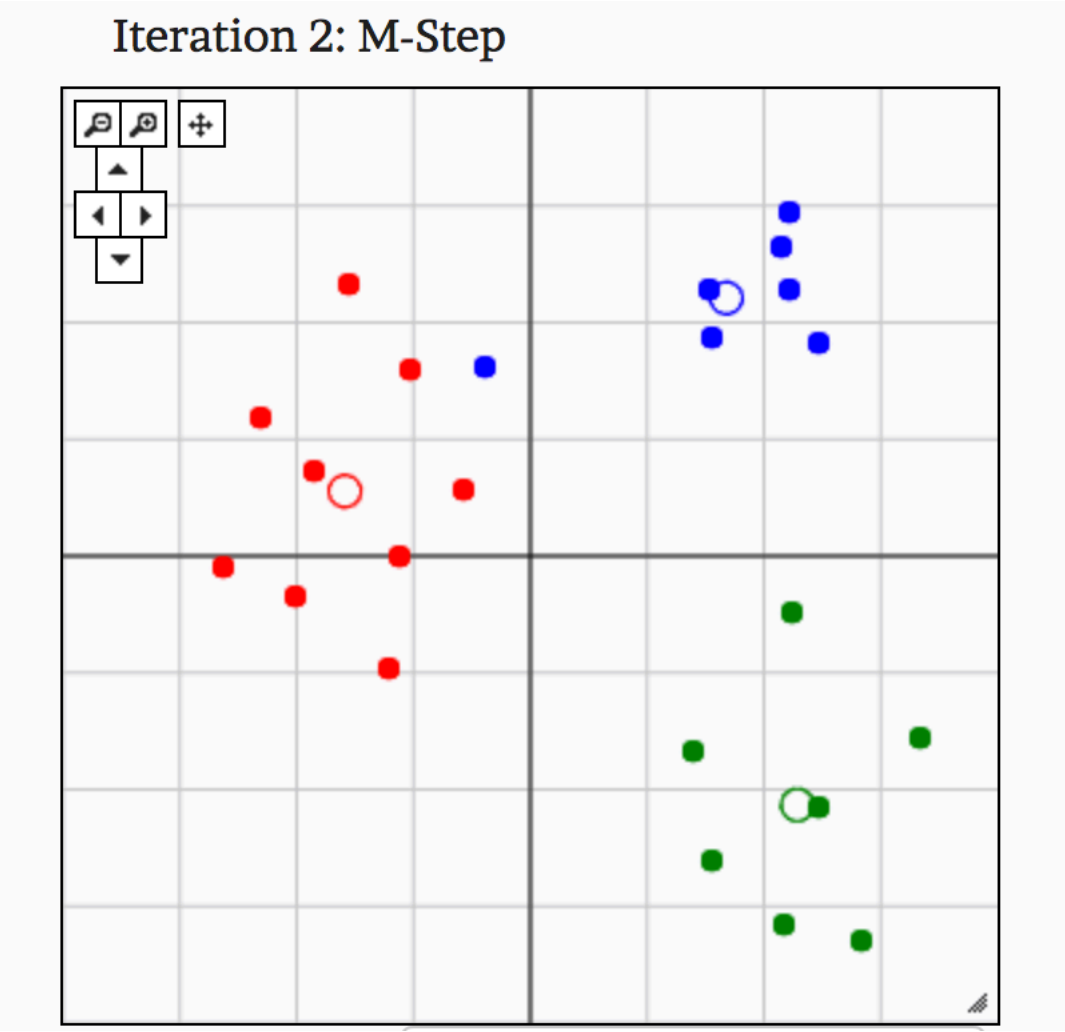
- From: http://util.io/k-means

K-means clustering in action



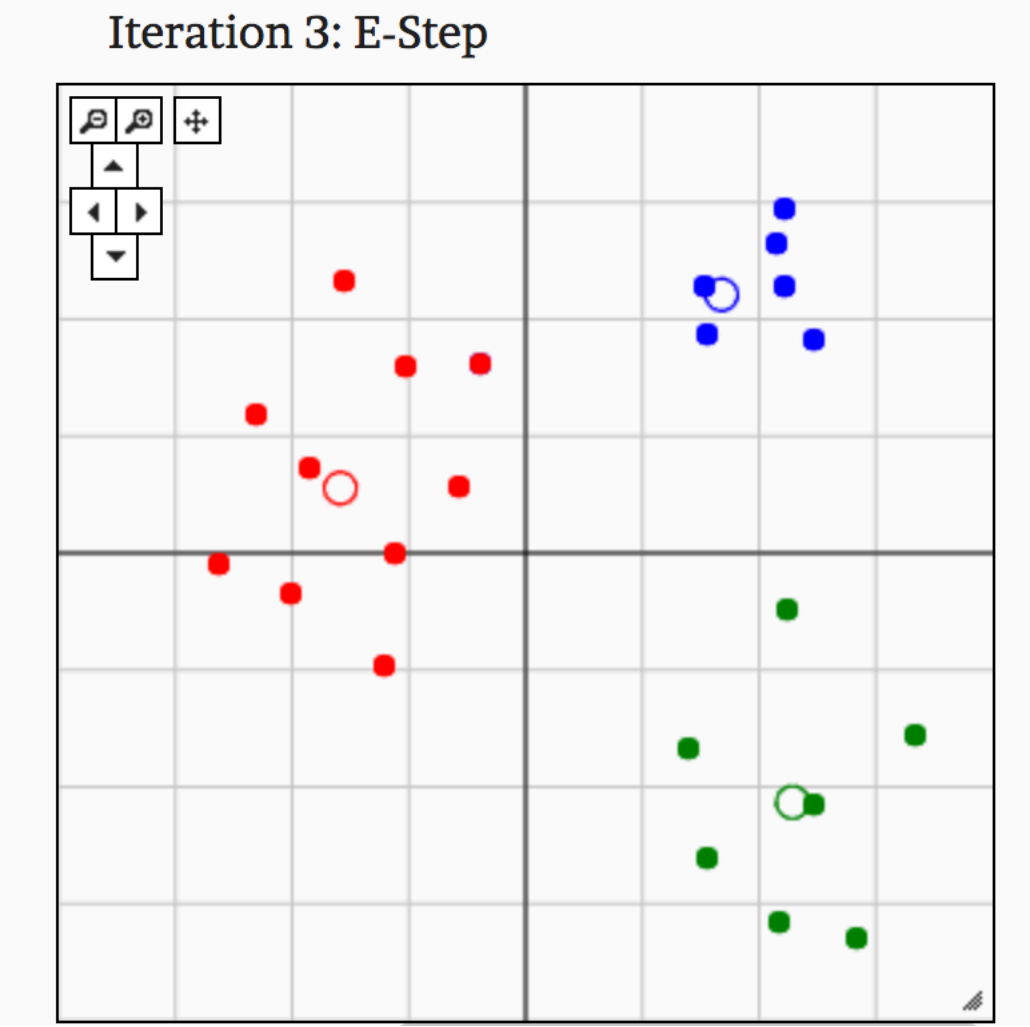
- From: http://util.io/k-means

K-means clustering in action



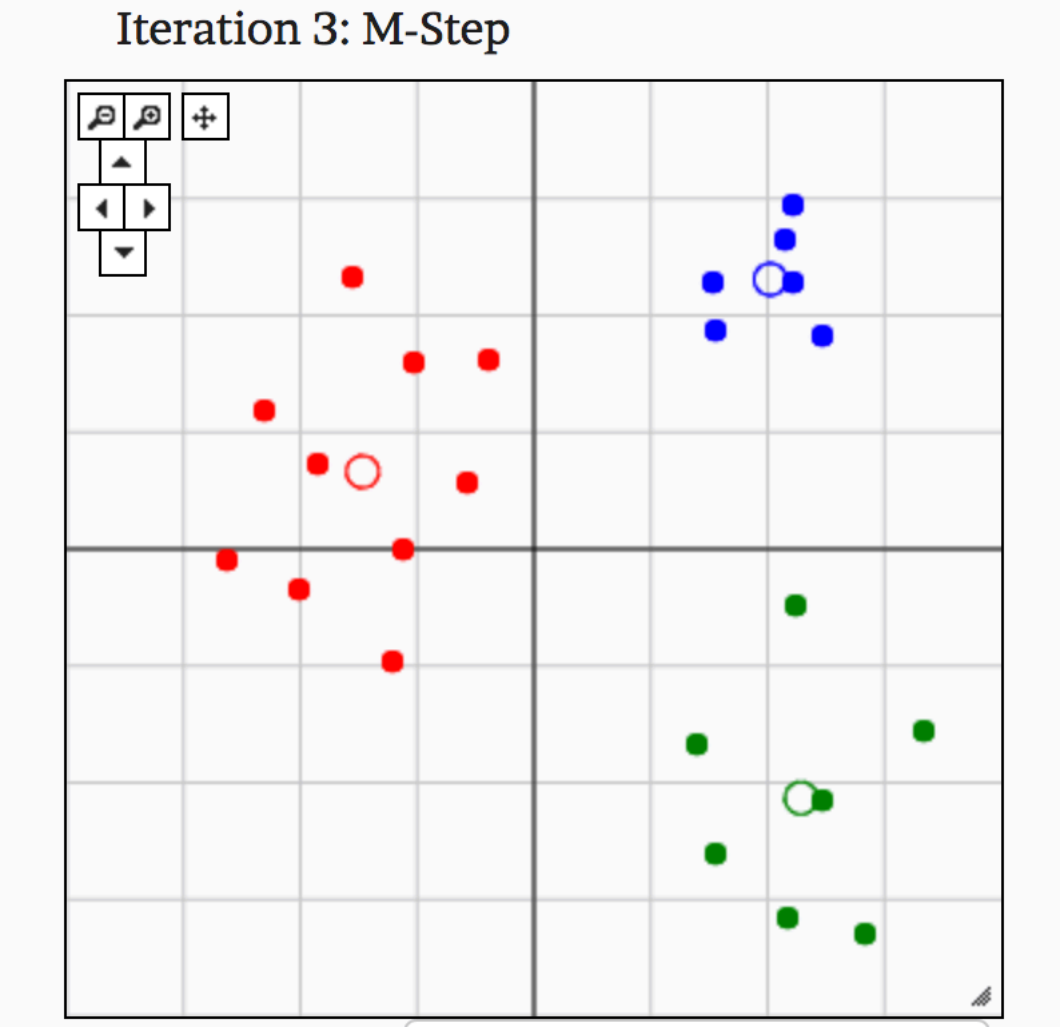
- From: http://util.io/k-means

K-means clustering in action



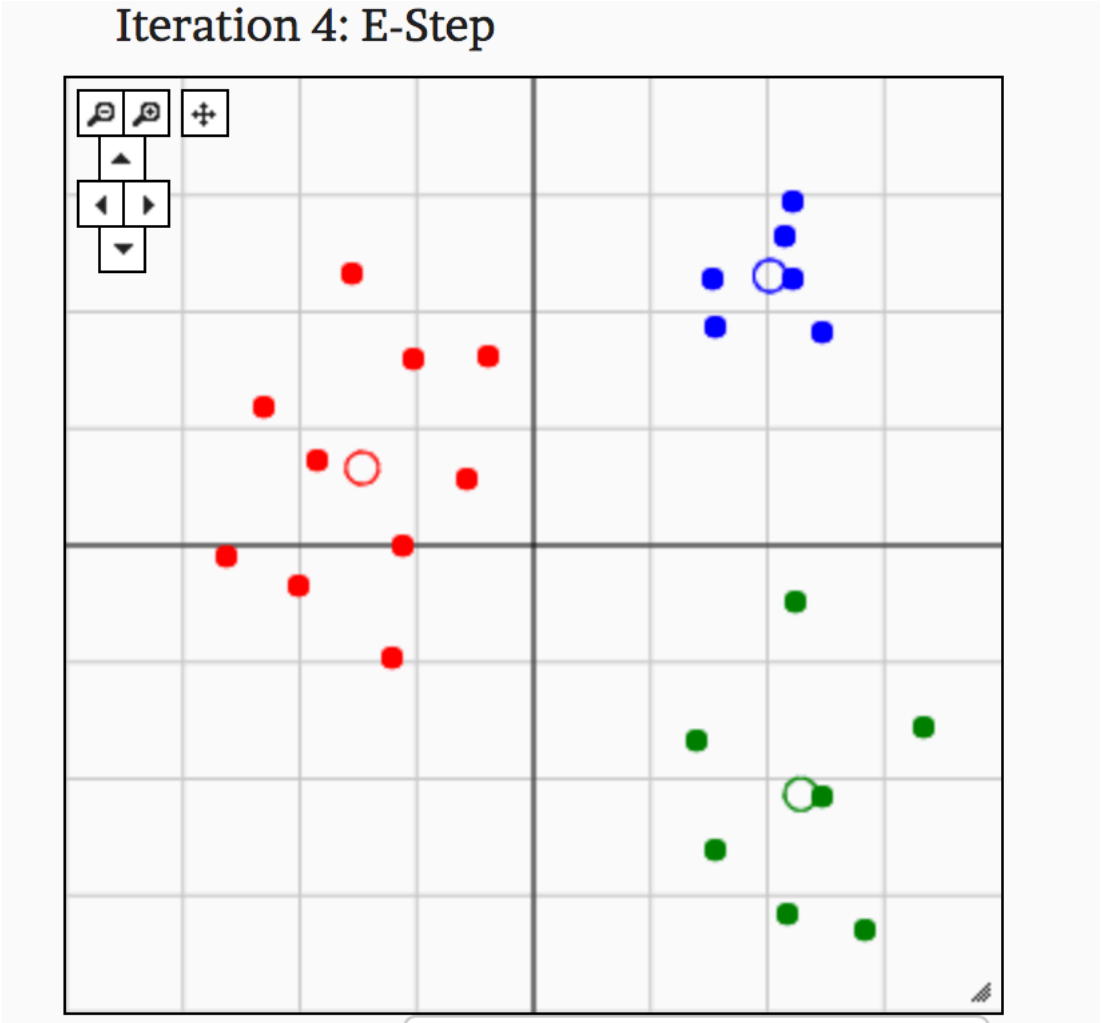
- From: http://util.io/k-means

K-means clustering in action



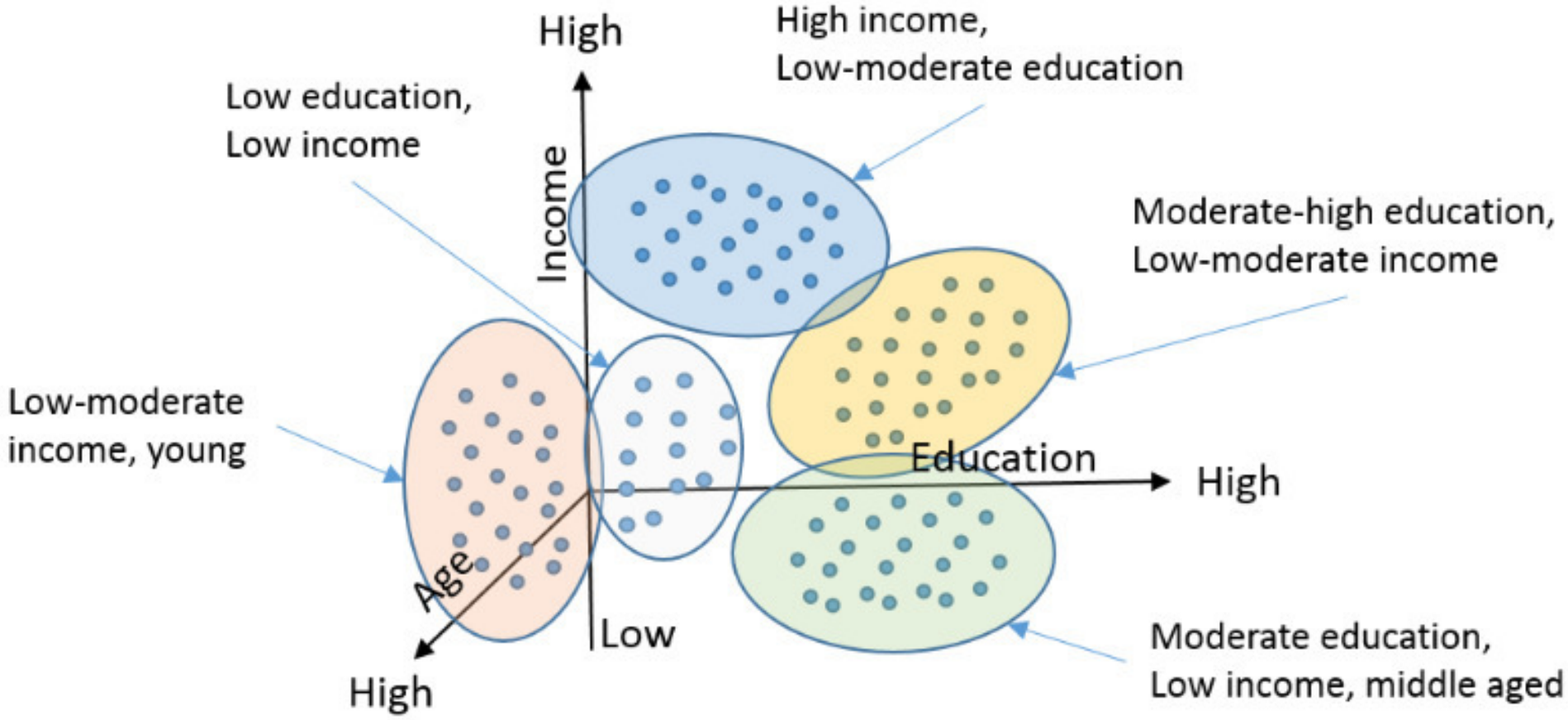
- From: http://util.io/k-means

K-means clustering in action



- From: http://util.io/k-means

Examples of clustering: Customer Segmentation



Examples of clustering: Clustering Wide Receivers

Using k-means clustering to find similar players

Wed 23 September 2015

Most of the posts so far have focused on what data scientists call *supervised* methods -- you have some outcome you're trying to predict and you use a combination of predictor variables to do so. Another important class of methods are called *unsupervised*. In this case, you might not know what exactly you're looking for or what metric you want to optimize for, but you want to explore the data and identify similarities among cases. For example, you might want to identify a list of "similar" players for your fantasy draft. This is a little late for the start of fantasy season, but with the rise of daily fantasy sports, perhaps not. However, maybe you don't know what "similar" means in this case or you don't have a single number or index that you want to match on. Perhaps you just want to find players with similar production to hedge against bye weeks or injuries.

This is where unsupervised methods come in. We'll be focusing on a popular unsupervised method called *clustering*. You'll see these kinds of methods used on a number of sports sites. Boris Chen, a data scientist at the New York Times, uses a kind of clustering to produce his fantasy football player tiers. Krishna Narsu recently used a kind of clustering to redefine the defensive positions in the NBA.

One popular method is called *k-means clustering*. (Note, this isn't the same *k* as in *k*-fold cross-validation, *k* is just a common stand-in for an unknown integer value.) I'll be working through an example clustering wide receivers using their 2013 statistics. K-means is really beautifully simple. The basic idea is that we want to take our entire data set and divide the observations into *k* sections and have each of the observations be as similar to each other as possible (and potentially as dissimilar to every other cluster as possible). Each cluster has what's known as a 'center' or 'centroid', which is the point against which all of the observations in that cluster are compared. You can think of it as the "ideal" or "prototypical" observation that typifies each cluster.

EDIT: As always, code for this example is up on GitHub.

- From: http://thespread.us/clustering.html

Examples of clustering: Clustering Wide Receivers

Cluster	Targets	Receptions	Yards	TDs	Fumbles	Fantasy Points
0	-0.84	-0.82	-0.81	-0.72	-0.4	-0.82
1	0.49	0.47	0.43	0.41	-0.38	0.45
2	1.74	1.82	1.9	1.72	0.47	1.93
3	0.21	0.12	0.08	-0.07	2.46	0.01

Player	cluster
A.Hawkins	0
A.Robinson	0
B.Golden	0
C.Johnson	2
E.Bennett	0
E.Weems	0
J.Boyce	0
J.Criner	1
J.Ebert	0
K.Allen	2
K.Martin	0
L.Brazill	1
L.Hankerson	1
L.Moore	1
R.Shepard	0
R.Shepard	0
R.Woods	1
R.Woods	1
S.Smith	1
V.Jackson	2

- From: http://thespread.us/clustering.html

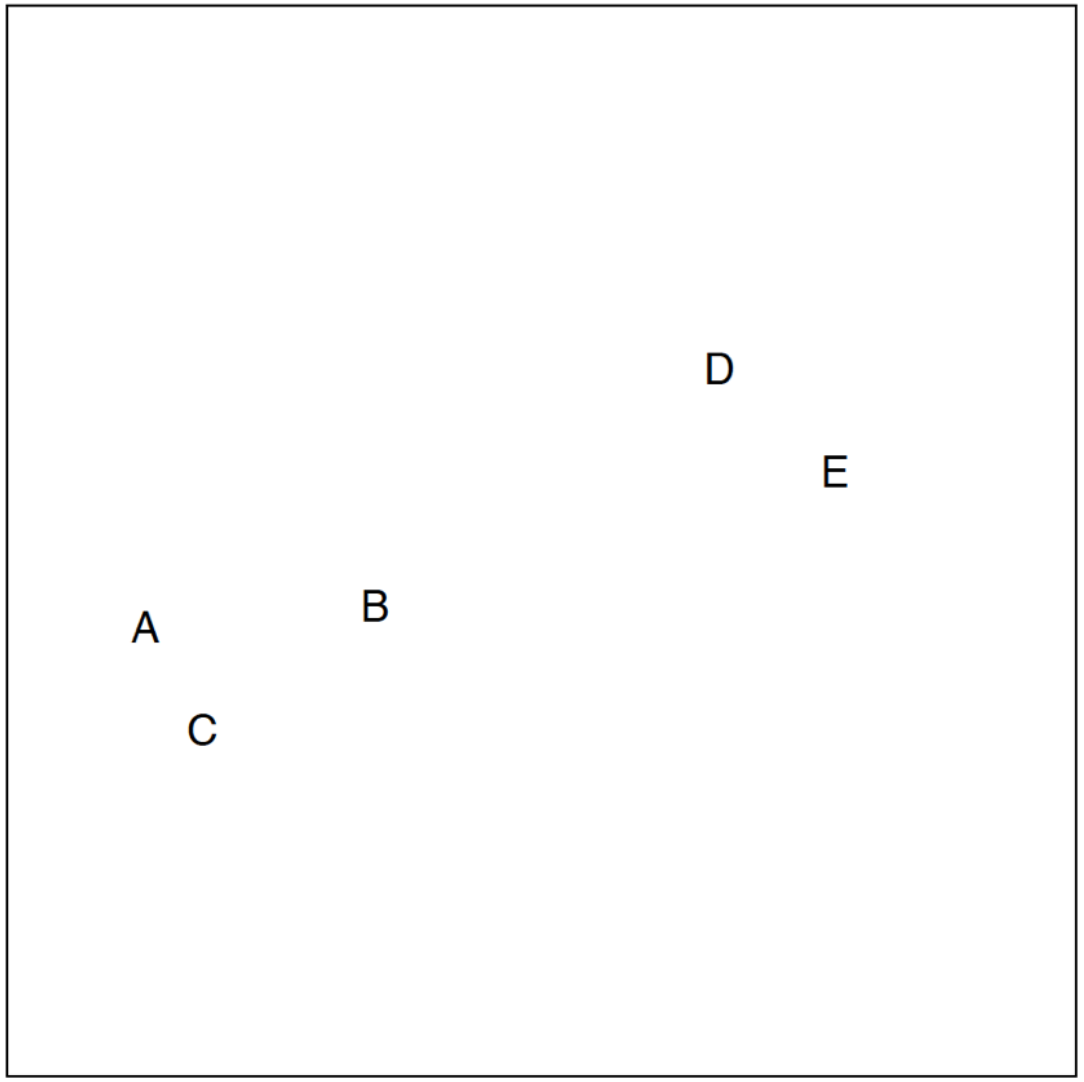
Plan for Today

- What topics to cover for Data Management?
  - Merging data
  - Dplyr basics
  - What else?
- No problem set 11 (is that okay?)
- Signup for presentations
  - Monday, Dec 3rd
  - Wednesday, Dec 5th
- Hierarchical clustering
- K-means code
- PCA
- PCA code

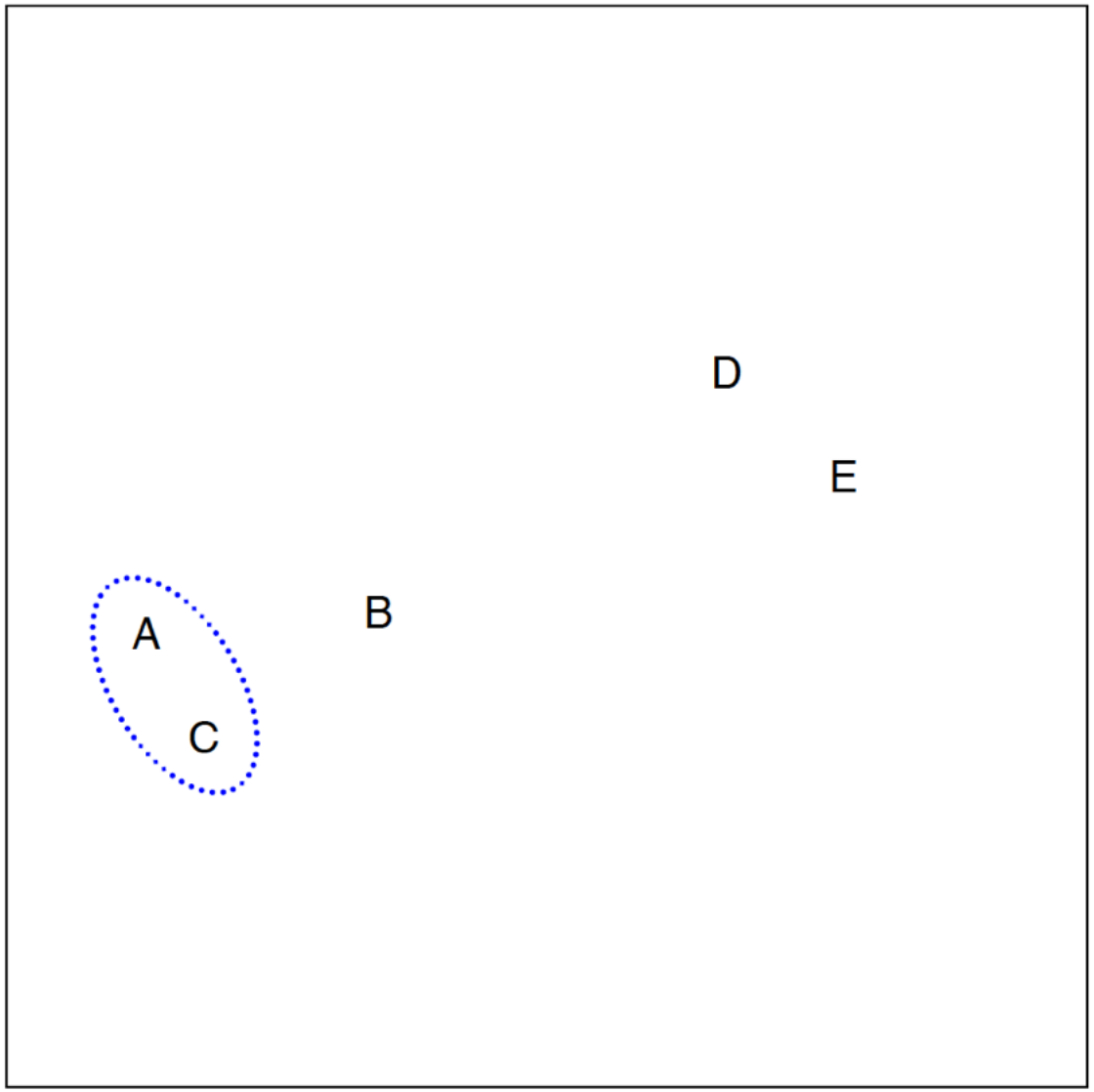
Hierarchical clustering algorithm

1. Begin with n observations and calculate all of the pairwise dissimilarities. Treat each observation as its own cluster
2. For $i = n, n - 1, \ldots, 2$:
   - Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the clusters that are most similar. Fuse these two clusters.
   - Compute the new pairwise inter-cluster dissimilarities among the $i - 1$ remaining clusters

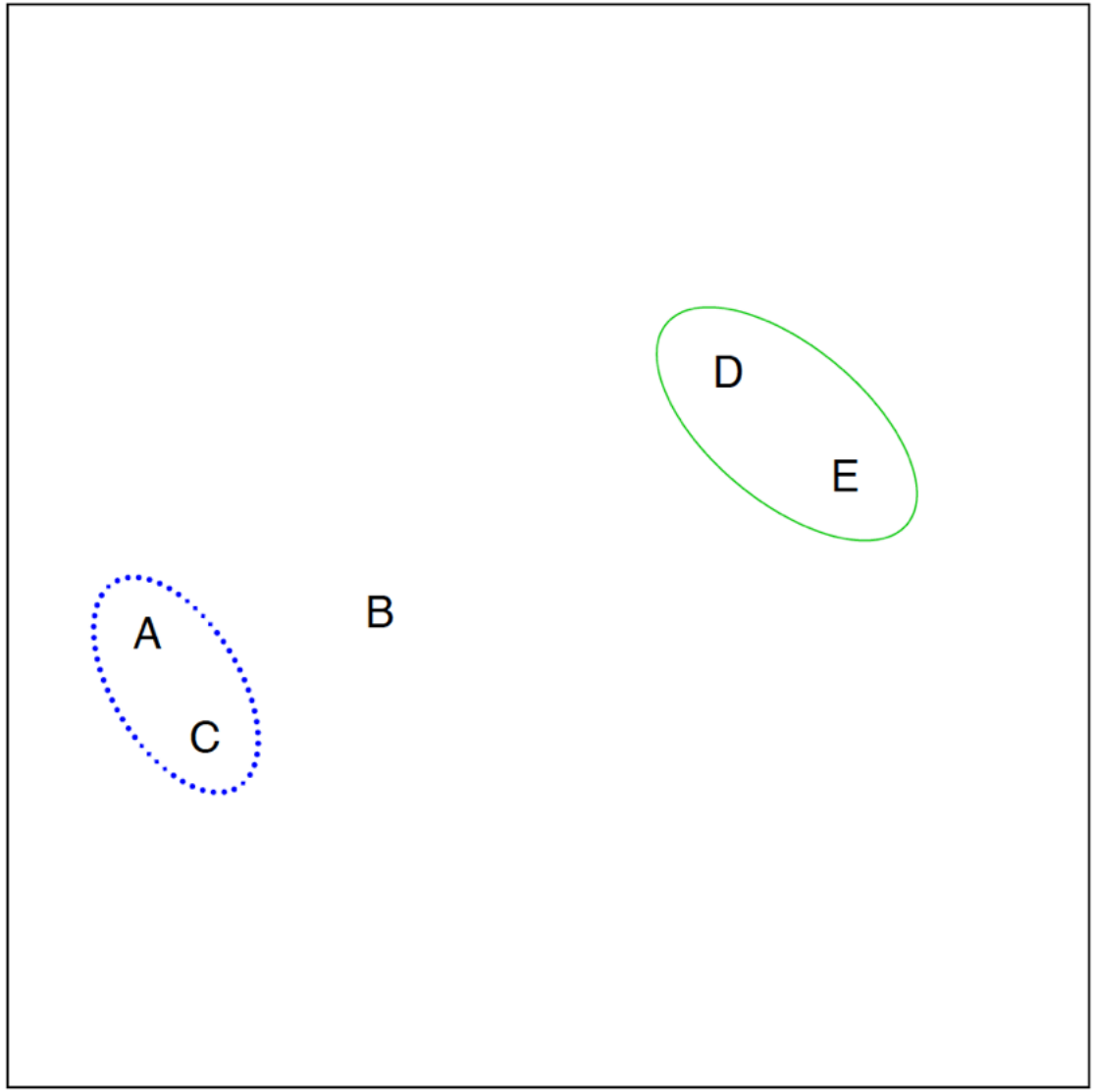
Hierarchical clustering in action



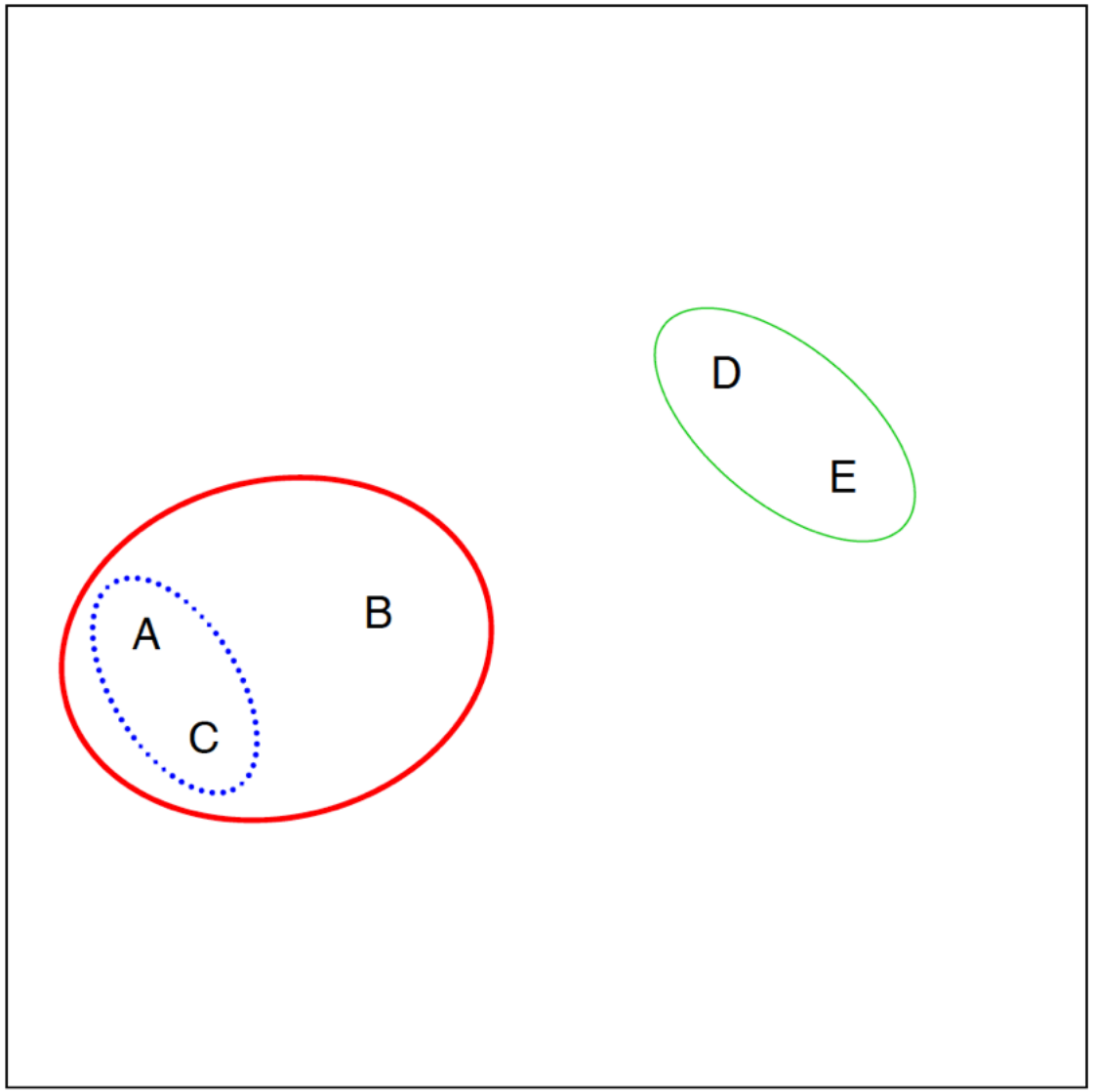
Hierarchical clustering in action



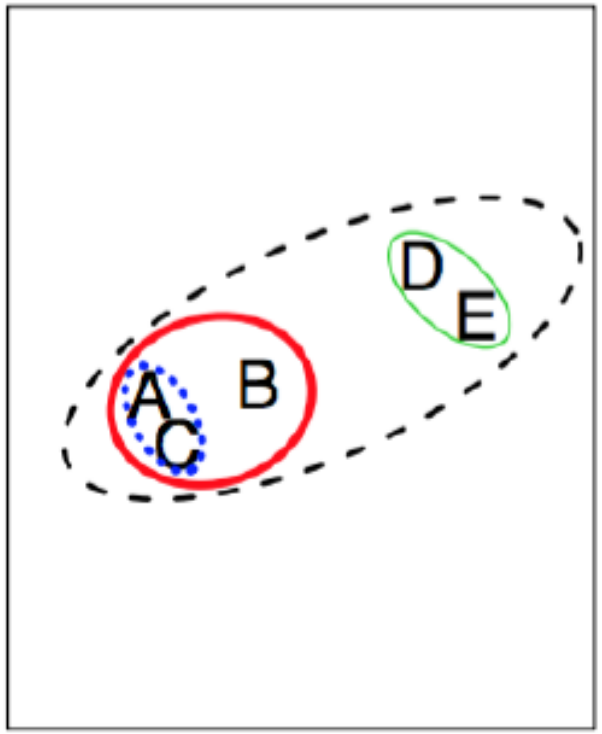
Hierarchical clustering in action

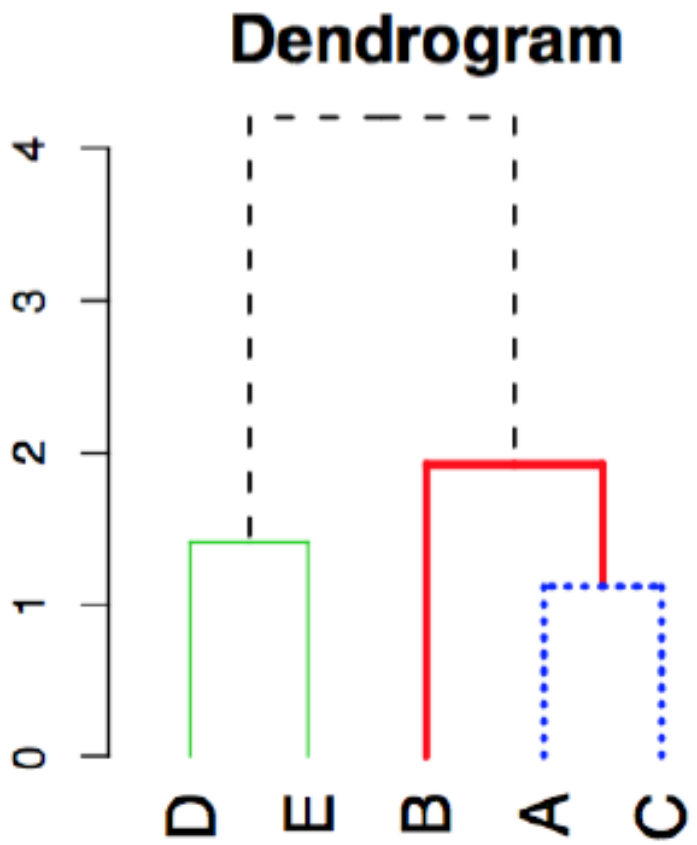


Hierarchical clustering in action



Connection to “dendrograms”





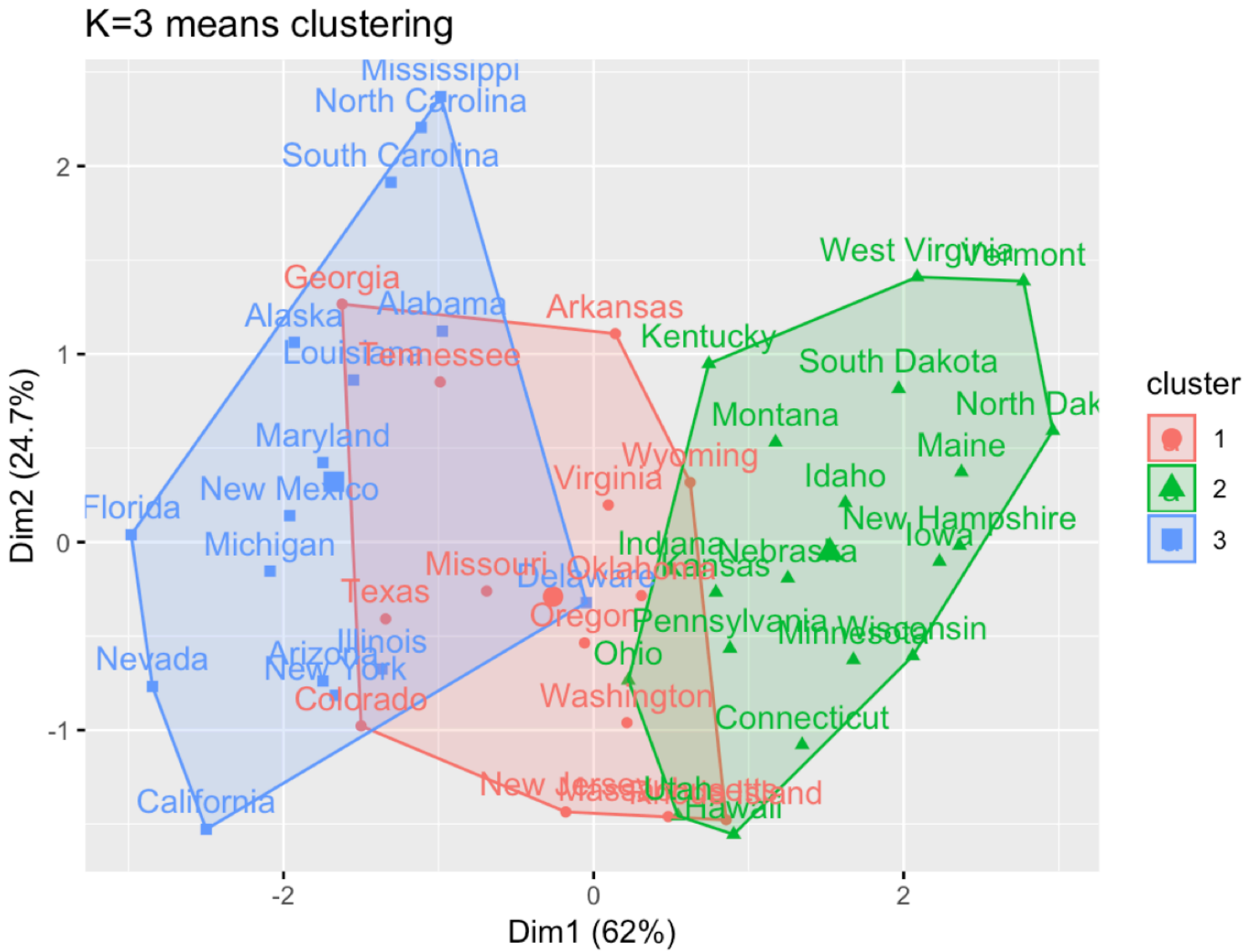
K-means in R

```
library('ISLR')
library('factoextra')
library('cluster')
data("USArrests")
kmeans3 <- kmeans(USArrests, centers = 3, nstart = 25)
str(kmeans3)
kmeans3
```

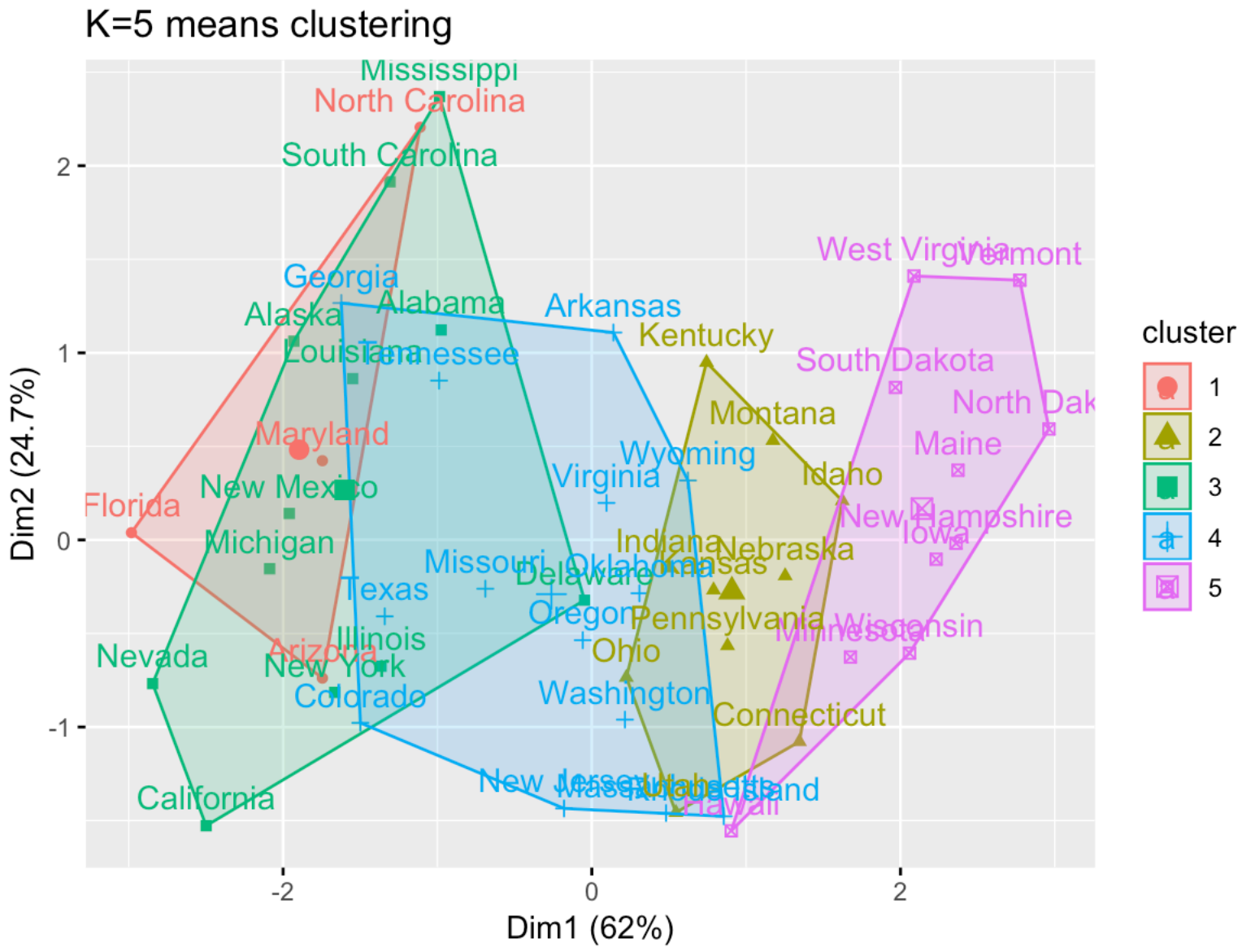
K-means in R

```
> kmeans3
K-means clustering with 3 clusters of sizes 14, 20, 16

Cluster means:
     Murder  Assault UrbanPop     Rape
1  8.214286 173.2857 70.64286 22.84286
2  4.270000  87.5500 59.75000 14.39000
3 11.812500 272.5625 68.31250 28.37500
```

```
Within cluster sum of squares by cluster:
[1]  9136.643 19263.760 19563.863
 (between_SS / total_SS =  86.5 %)
```




```
fviz_cluster(kmeans3, data = USArrests, main = "K=3 means clustering")
```



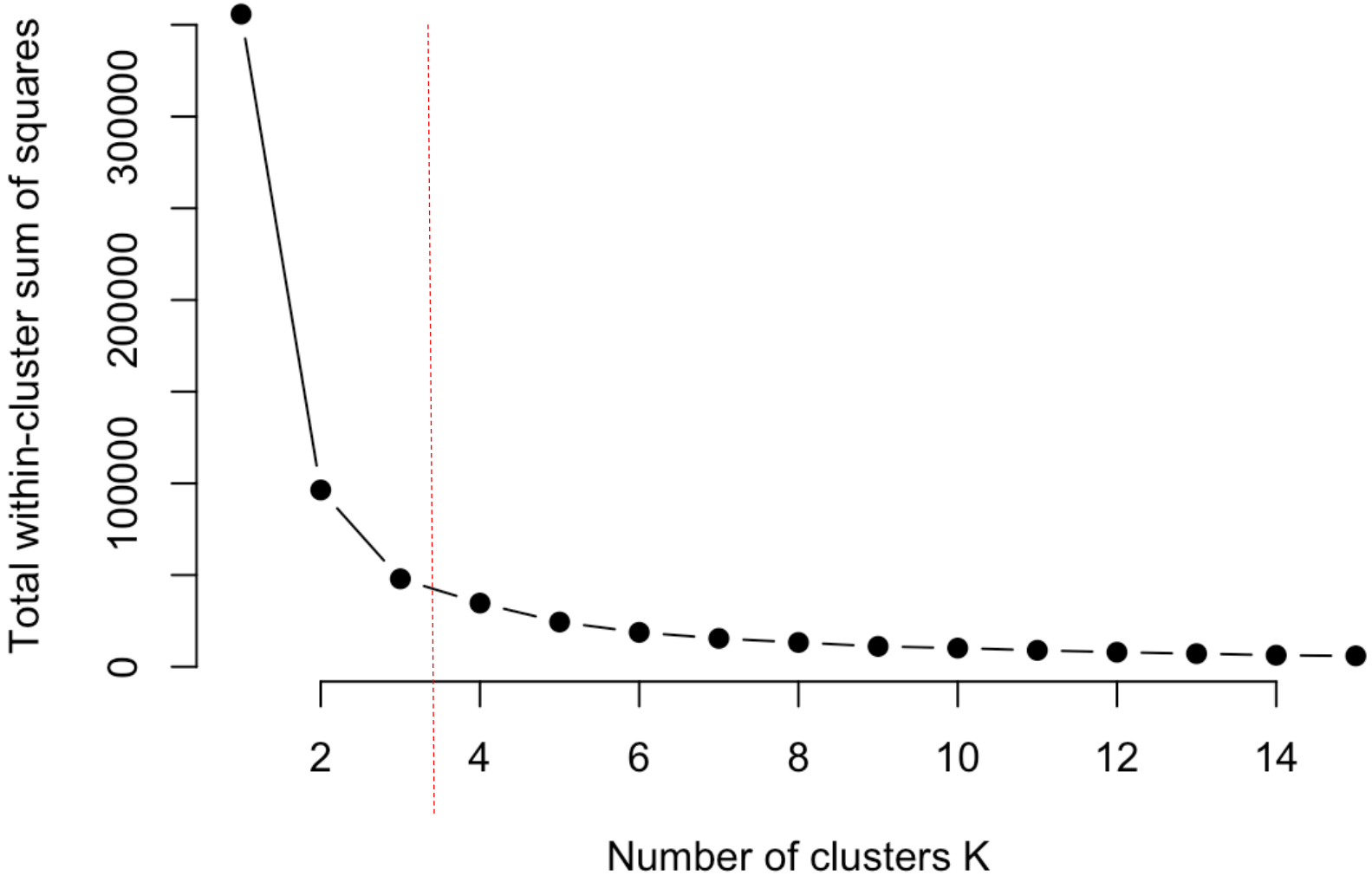

```
kmeans5 <- kmeans(USArrests, centers = 5, nstart = 25)
fviz_cluster(kmeans5, data = USArrests, main = "K=5 means clustering")
```

How to pick k? “Elbow Method”

- Plot explained variance by # of clusters
- Look for the “bend” in the plot

```
# how many clusters? Elbow method
wss <- function(k){
  kmeans(USArrests, k, nstart = 25)$tot.withinss
}
wss_values <- map_dbl(1:15, wss)
plot(1:15, wss_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Total within-cluster sum of squares")
```

Within-cluster sum of squares by # of clusters



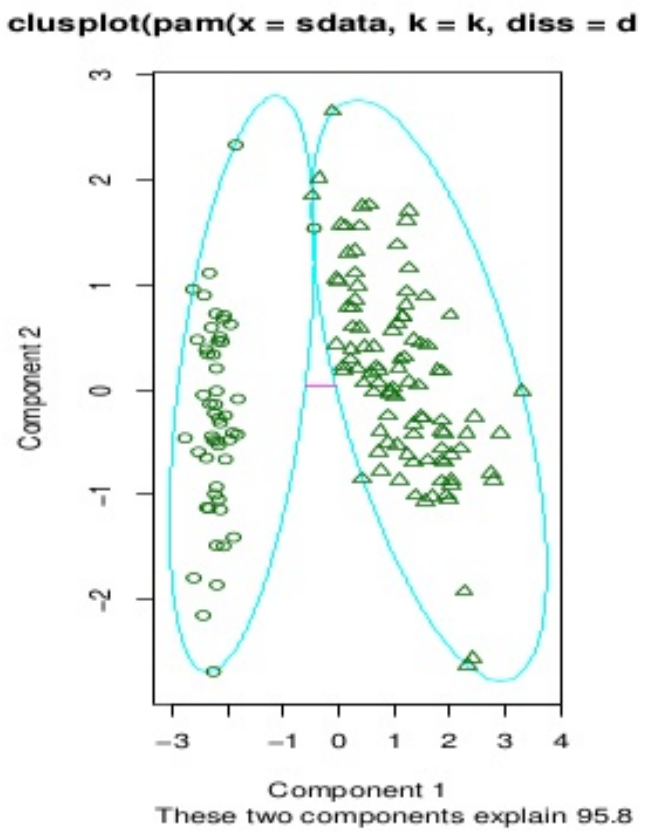
How to pick k?

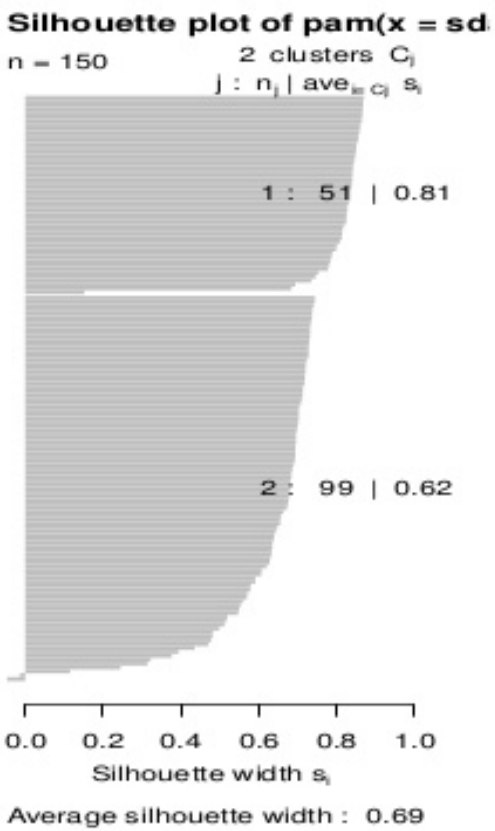
Silhouette score:

- a_i = measure of dissimilarity between point i and other observations in its cluster.
  - 1 = good match with cluster
  - -1 = really bad match
- b_i = measure of dissimilarity between i and all other points in any cluster
- $s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$

Silhouette Score

```
layout(matrix(c(1, 2), 1, 2))  # 2 graphs per page
plot(pamk.result$pamobject)
```



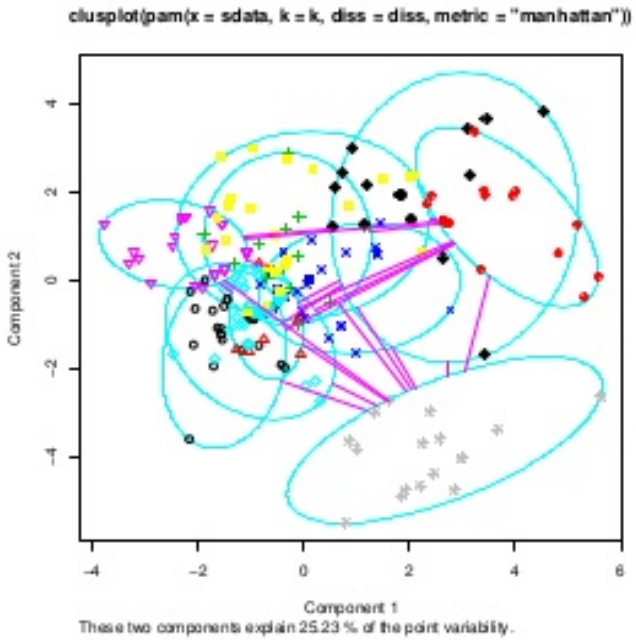


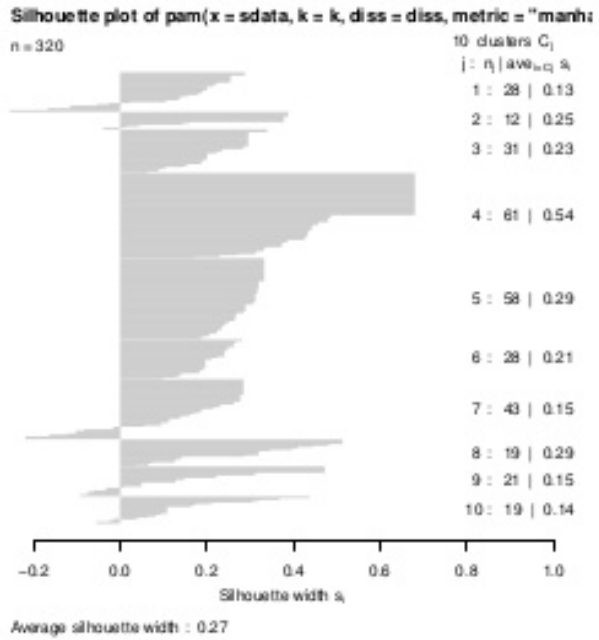

```
layout(matrix(1))  # change back to one graph per page
```

11 / 30

Silhouette Score

```
# plot clustering result
layout(matrix(c(1, 2), 1, 2))  # set to two graphs per page
plot(pamResult, col.p = pamResult$clustering)
```






```
layout(matrix(1))  # change back to one graph per page
```

Motivation for Dimension Reduction 1

- Suppose we want to visualize n observations that have p variables, $X_1, \ldots, X_n$
- We could examine every two-dimensional sub-plot

- However, that would be $(how\ many)$ subplots?
  - $\binom{p}{2} = \frac{p(p-1)}{2}$
- How do we visualize data to see how "far apart" certain observations are from one another?

Motivation for Dimension Reduction 2

- Sometimes in a regression setting, we care about controlling for many characteristics, but don't care about interpreting those characteristics.
- We can take those control variables, project them down into a lower dimensional subspace, then control for only those lower dimensional variables
- E.g. take 100 controls variables, project down to 2 dimensions, and only control for those 2 variables

Enter Principal Component Analysis

- **PCA produces a low-dimensional representation of a dataset**.
- It does so by finding directions of maximal variance that are mutually orthogonal

- In other words it asks the question

If I had to pick a set of direction such that if I projected the data onto those directions, and then computed the variance of the points, which directions would maximize explained variance?

First Principal Component

- PCA finds sequence of linear combinations of the variables that maximize variance and are mutually uncorrelated
- The first principal component of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

  That have the largest variance

- Normalized here means $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- $\boldsymbol{\phi_1} = \left(\phi_{11}\phi_{21} \ldots \phi_{p1}\right)^T$ is the vector of factor loadings of the first principal component

First Principal Component

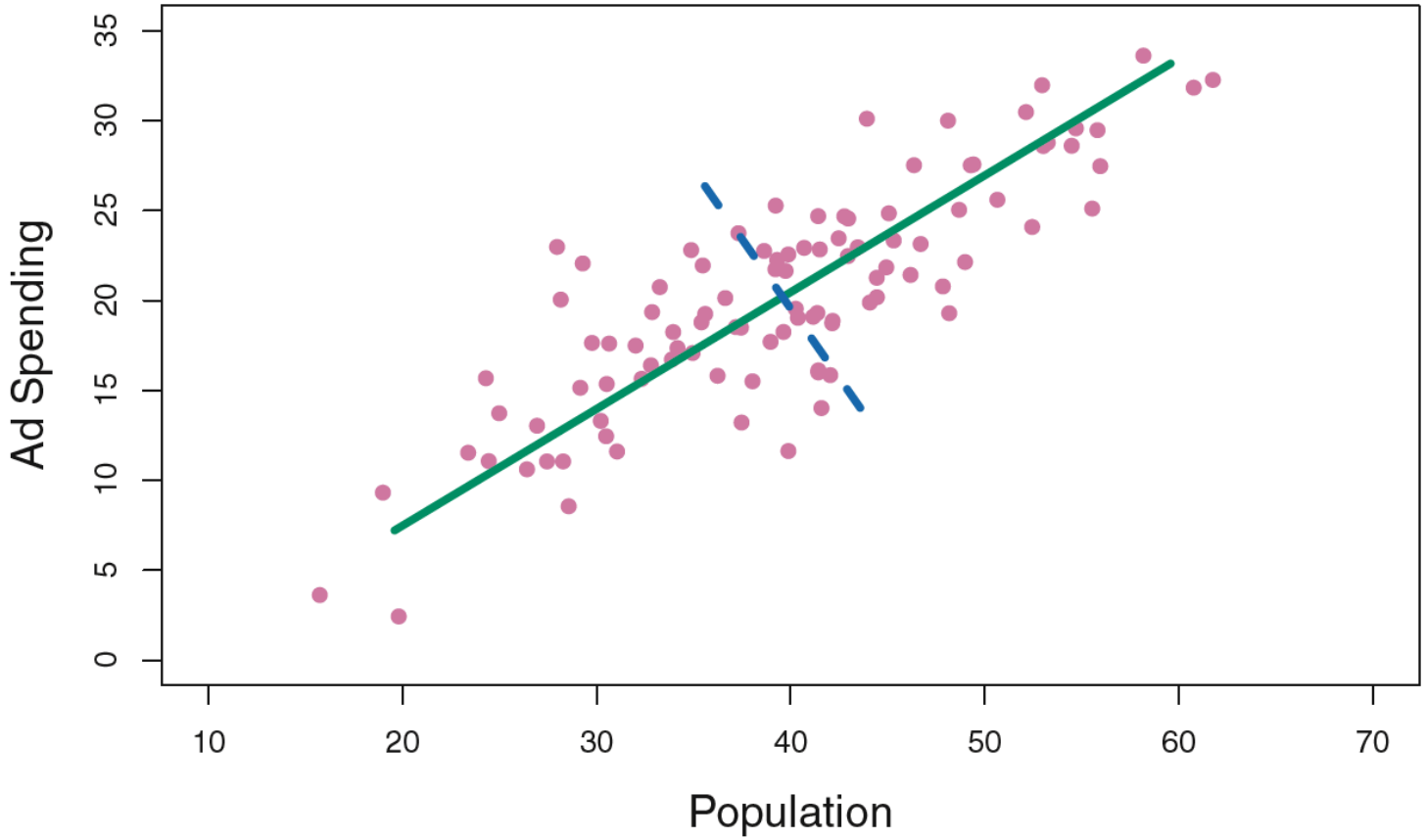


FIGURE 6.14. *The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.*

Second Principal Component

- After we have calculated the first principal component,Z_1, the second principle component is the linear combination of $X_1, \cdots, X_p$ that has maximal variance out of all linear combinations that are uncorrelated with Z_1
- The second principal component takes the form

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + \cdots + \phi_{p2}X_p$$

$\boldsymbol{\phi_2} = \left(\phi_{12}\phi_{22} \ldots \phi_{p2}\right)^T$ is the vector of factor loadings of the second principal component

- Constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining X_1 to be orthogonal (perpendicular) to X_2

Second Principal Component

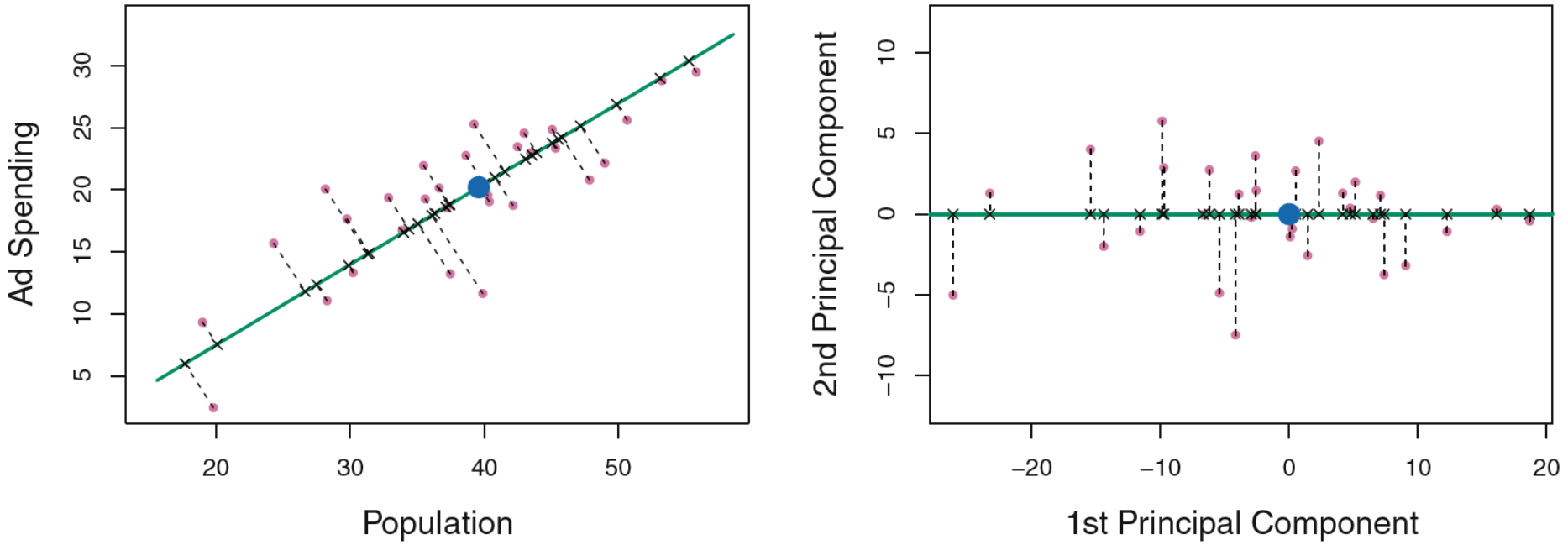


FIGURE 6.15. *A subset of the advertising data. The mean* pop *and* ad *budgets are indicated with a blue circle.* Left: *The first principal component direction is shown in green. It is the dimension along which the data vary the most, and it also defines the line that is closest to all* n *of the observations. The distances from each observation to the principal component are represented using the black dashed line segments. The blue dot represents* $(\overline{\texttt{pop}}, \overline{\texttt{ad}})$. Right: *The left-hand panel has been rotated so that the first principal component direction coincides with the* x*-axis.*

Second Principal Component

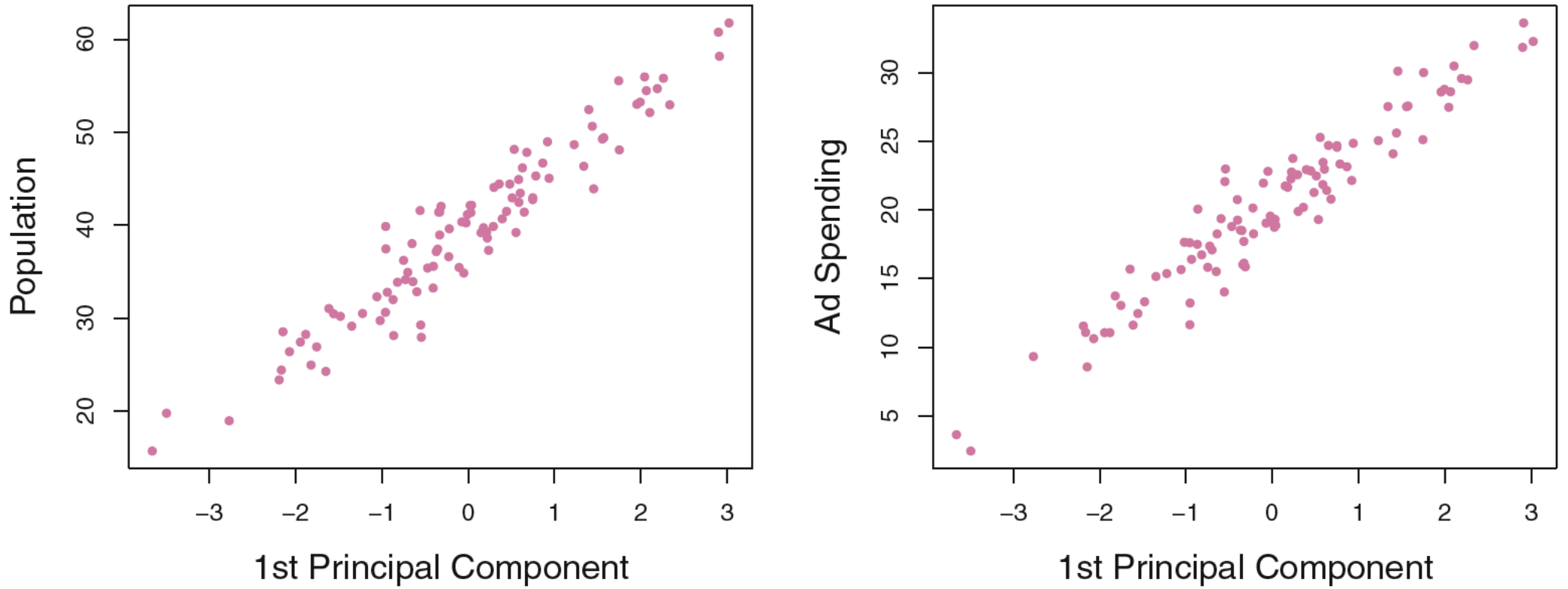


FIGURE 6.16. *Plots of the first principal component scores* z_{i1} *versus* pop *and* ad. *The relationships are strong.*

PCA Illustration

- USAarrests data: For each 50 state in the US, number of arrests per 100,000 residents for each of the three crimes: assault, murder, and rape. Also record UrbanPop.
- Principal component score vectors have a length of n = 50, and loading vectors have length p = 4
- PCA was performed after standardizing variables to have a mean of 0 and standard deviation of 1.

PCA Illustration

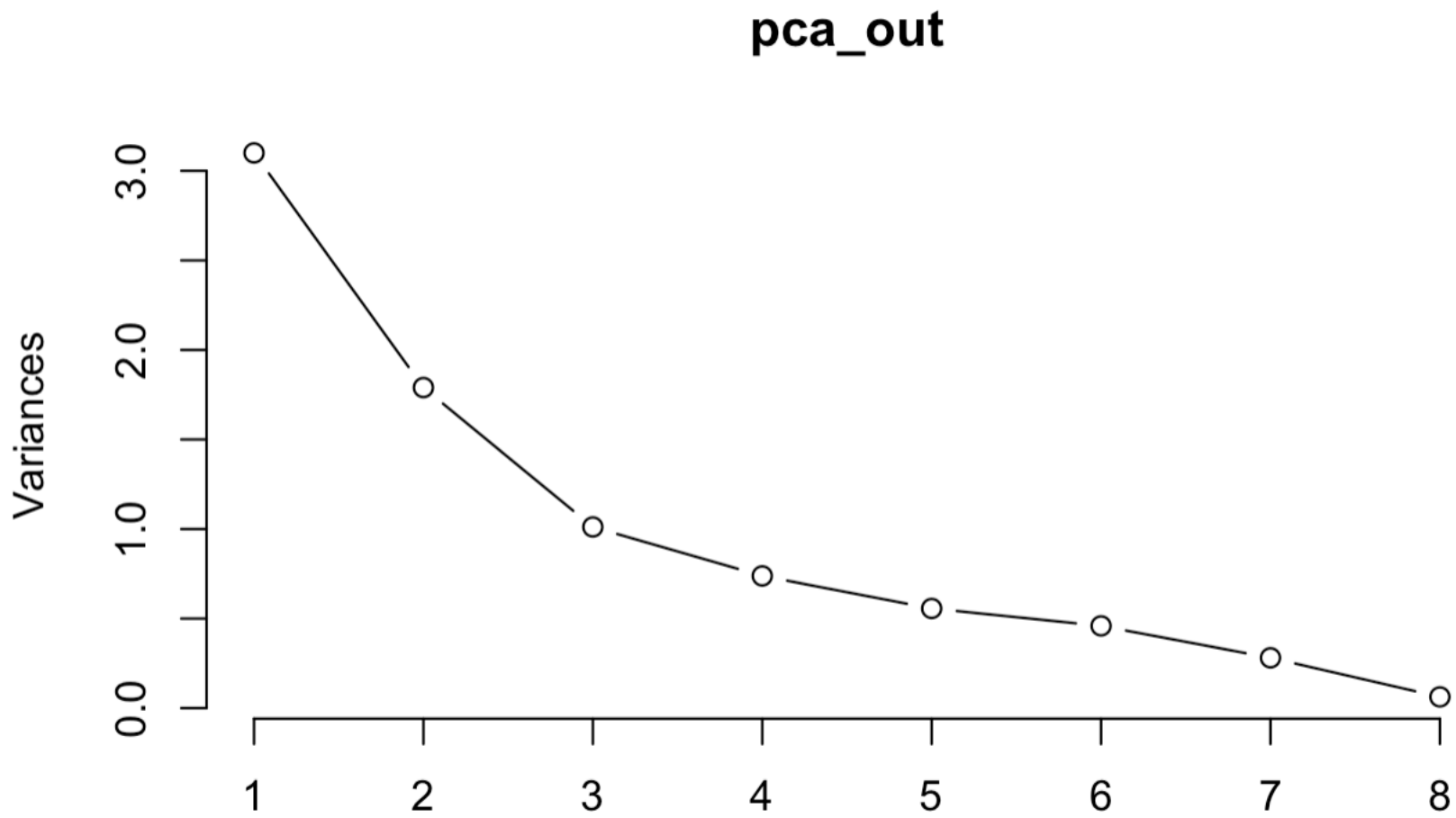
```
require('ISLR')
dimnames(USArrests)
apply(USArrests,2,mean)
apply(USArrests,2,var)

## Standardize the variables and PCA the vars
pca.out <- prcomp(USArrests, scale=TRUE)
pca.out
```

PCA Illustration

```
> pca.out
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

Factor loadings

How many PCAs to use? “Elbow Method” from “Skree” plot

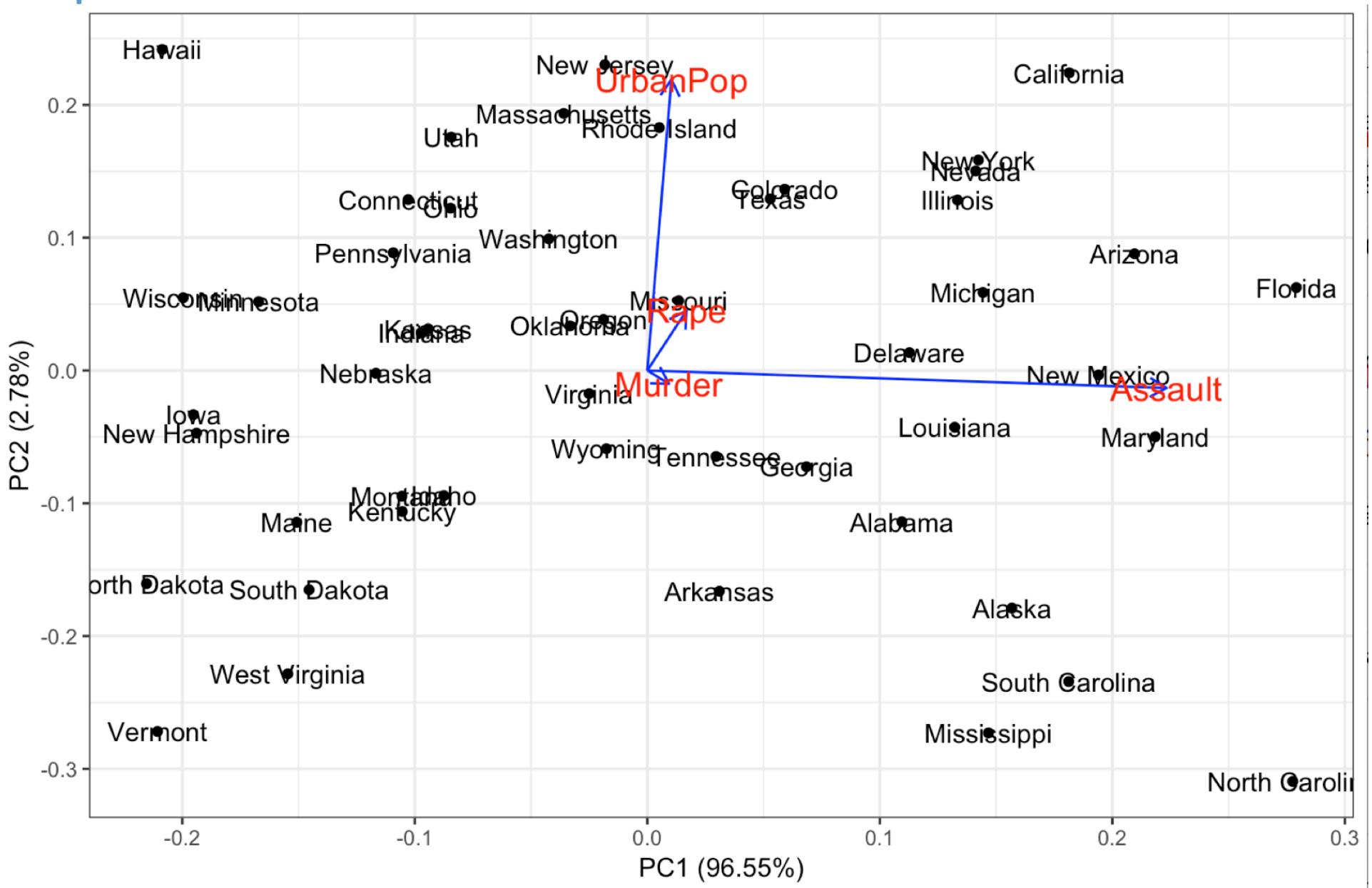


Plotting PC1 and PC2

- As promised, we can now use the 1st and 2nd principal component to visualize the P-dimensional data on a 2-dimensional plot
- This is called “projecting” our data in 2 dimensions

```
# boring biplot plot
biplot(pca.out)

# ggplot biplot
library('ggfortify')
autoplot(prcomp(USArrests), data = USArrests,
         loadings = TRUE, loadings.colour = 'blue',
         loadings.label = TRUE, loadings.label.size = 5,
         label = TRUE) + theme_bw()
```

PCA “Biplot”


UrbanPop
Rape
Murder
Assault
Hawaii
New Jersey
California
Massachusetts
Rhode Island
Utah
New York
Nevada
Colorado
Texas
Illinois
Connecticut
Ohio
Washington
Pennsylvania
Arizona
Wisconsin
Minnesota
Michigan
Florida
Missouri
Oregon
Oklahoma
Kansas
Indiana
Delaware
Nebraska
New Mexico
Virginia
Iowa
New Hampshire
Louisiana
Maryland
Wyoming
Tennessee
Georgia
Montana
Idaho
Kentucky
Maine
Alabama
North Dakota
South Dakota
Arkansas
Alaska
West Virginia
South Carolina
Vermont
Mississippi
North Carolina
0.2
0.1
0.0
-0.1
-0.2
-0.3
PC2 (2.78%)
-0.2
-0.1
0.0
0.1
0.2
0.3
PC1 (96.55%)