Chapter 10: Clustering

1 Opening Story

"Flag and Family Republicans" and "Tax and Terrorism Moderates" sound like names that fall along the American political spectrum. In fact, these are labels assigned to two clusters of American voters that have been empirically identified by political campaigns.[1] Some set of voter characteristics lend themselves to discovering naturally occurring groups, or clusters. As these clusters are quantifiable, the implications are profound for the tactics and strategies employed in swaying voters. Imagine a scenario in which campaigns needs to get the word out without data on voters. This would mean a random call placed to a "Flag and Family Republic" touting the message of a liberal democrat candidate is unlikely to be successful. However, if data on voters is available and each individual can be mapped to a cluster, then campaigns can develop customized ads through relevant channels and influence votes and contributions.[2] This is the core idea behind *micro-targeting* – campaigns are tailored to distinct groups and even specific people.[3]

The idea of voter micro-targeting is not new, however. A rudimentary form of targeting was used as early as Jimmy Carter's 1976 Presidential campaign in which the country was segmented by issue and geography.[4] Surveying played a notable part in understanding which issues matter to which demographics – what emotions are associated with different social and fiscal matters. [5] Now in the information age, the possible data-driven strategies are also associated with a "creepy" factor. As it turns out, individual level data such as brand preferences are correlated with which candidate a voter is likely to vote. According to a survey conducted by a brand consultant in 2016, Bernie Sanders supporters are 82% more likely to eat at Chipotle than the average American and Donald Trump supporters are 111% more likely to eat a Sonic.[6] Thus, using data that is common in marketing databases enable campaigns to define hyper-local, finely tuned clusters for highly customized messaging.

The interest and investment in micro-targeting in campaigns has only been growing with each successive election cycle. During the 2012 Obama campaign, for example, the analytics team was five times as large as the 2008 campaign.[7] In the 2016 cycle, the campaign of libertarian candidate Gary Johnson relied on Facebook's political ideology clusters to bolster his vote share.[8] Granted, with the revelations of the Cambridge Analytica scandal, criticism is mounting with some calling the microtargetted "dark advertising" on Facebook a threat to democracy.

Nonetheless, clustering is an effective statistical technique that brings natural structure to data, enabling a broad range of use cases from political campaigns to classifying web traffic.

2 Concepts

How exactly does clustering work? Perhaps the most tangible example is a matter of human sight. Below are visible nighttime imagery taken by the Suomi NPP satellite when flying over six major U.S. cities. Can you identify where are the likely inhabited areas? It does not take much to realize that the human eye is drawn to areas of contrast in which the average difference in light between two areas is large. Contrast is basically a measure of dissimilarity.

[1] http://www.winningcampaigns.org/Winning-Campaigns-Archive-Articles/Microtargeting-Knowing-Voter-Intimately.html
[2] https://fivethirtyeight.com/features/a-history-of-data-in-american-politics-part-2-obama-2008-to-the-present/
[3] https://www.cnn.com/2012/11/05/politics/voters-microtargeting/index.html
[4] http://www.washingtonpost.com/wp-dyn/content/article/2007/07/04/AR2007070401423.html
[5] http://www.nbcnews.com/id/15292903/ns/politics-tom_curry/t/mechanics-micro-targeting/#.XANCoZNKgxc
[6] https://www.bloomberg.com/news/articles/2016-02-18/sanders-supporters-like-chipotle-while-trump-fans-prefer-sonic
[7] http://swampland.time.com/2012/11/07/inside-the-secret-world-of-quants-and-data-crunchers-who-helped-obama-win/
[8] https://www.cnbc.com/2018/02/21/facebook-targeting-tools-helped-libertarian-candidate-gary-johnson.html

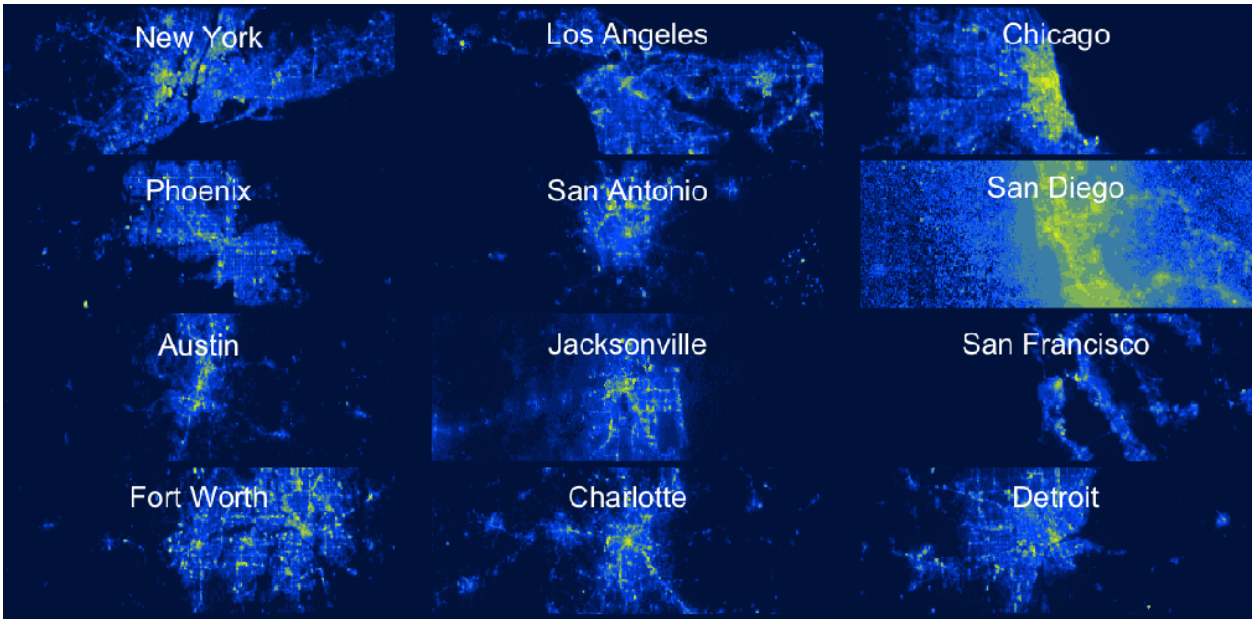


Figure 1: Night time imagery from the NOAA-NASA Suomi NPP Satellite's Visible Infrared Imaging Radiometer Suite (VIIRS)

Unfortunately, harnessing the human eye on a large scale is not an easy or convenient task – it can be labor intensive to hire an army of people to manually inspect imagery. Instead, what if we treated each pixel as data? A pixel is an element in a vector representing light intensity, on top of which statistical techniques can be applied to scalably approximate the process of visual analysis. In the San Francisco and the Bay Area, for example, the bulk of human activity is located along two strips of land. The *radiances*, or emitted light, are visualized in a kernel density plot showing that there may be a way to separate the dark pixels (the peak to the left) from brighter pixels (the tail to the right). The distribution can be color quantized, meaning that clustering can naturally reduce the radiance distribution into a few distinct groups of light intensity or color. In this case, we cluster pixels into two groups, light and dark. We do not know the true demarcation between light and dark, but we can pick two radiance values that approximate the average radiance on the left and right side of the distribution. For simplicity, let's assume they happen to be the "right" values.

Each pixel is compared with the reference points using a dissimilarity measure. For continuous values like radiances, dissimilarity can be defined as a simple Euclidean Distance:

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{n} |z_1 - z_2|^2}$$

Manhattan or binary distances would be appropriate for binary or discrete variables. Regardless of distance measure, each pixel is assigned to the same cluster as the closest reference point. In the case of San Francisco and the Bay Area, we can infer that 76% of pixels are likely inhabited by people with ocean and wilderness covering the remainder

There are many clustering techniques that can be applied to simple everyday problems to complex strategic endeavors. We explore two of the most commonly employed clustering algorithms: K-means clustering and hierarchical clustering. Each clustering technique is computationally intensive, constructed on different assumptions that help it accomplish this task, but the core differences between the techniques enable very different use cases.

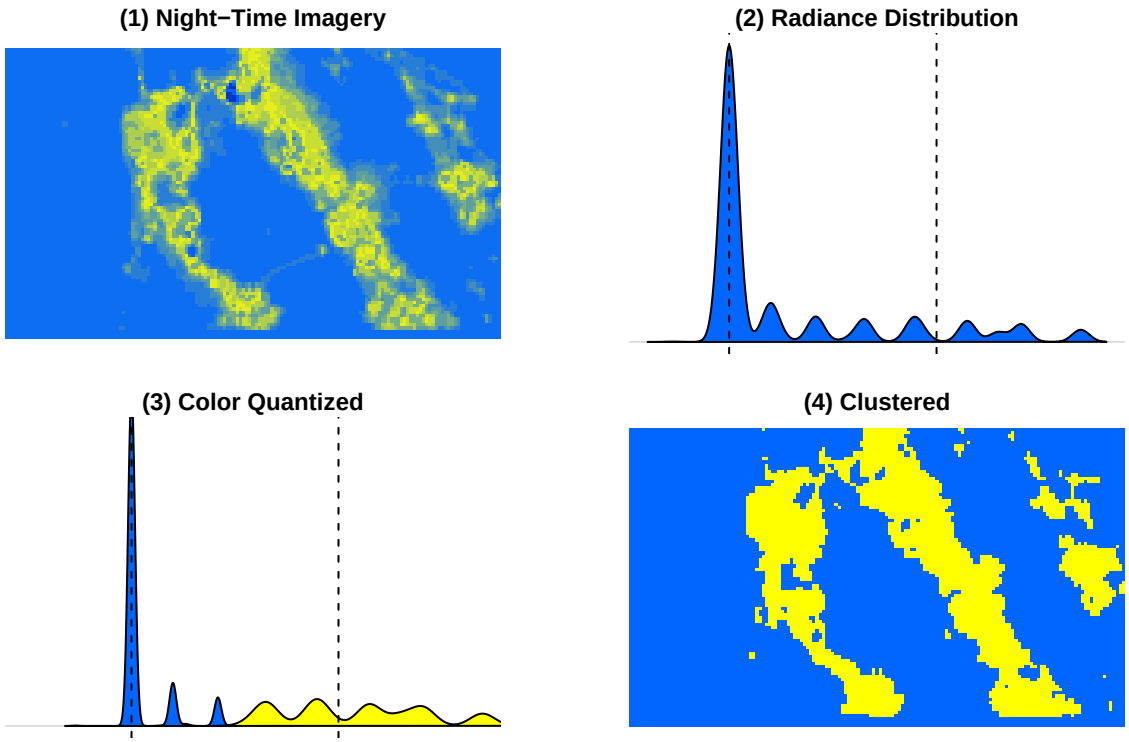


Figure 2: Applying clustering to extract likely inhabited areas in the area around San Francisco: (1) False color night-time imagery, (2) Kernel density of radiances, (3) Clustered or color quantized distribution, (4) Imagery classified into light and dark areas.

3 K-Means

K-means clustering identifies observations that belong to an unlabeled latent group by treating variables as coordinates. The coordinates can then be used to calculate distance between points. The k in *k-means* is a number of clusters that the analysts is interested. Without definition the value of k, the algorithm cannot run.

3.1 How It Works

The San Francisco example resembles the k-means process. Given k, the goal is to assign each observation to a cluster C. Each cluster C is defined by a set of centroids that are the means of input variables X for each cluster set. The optimal set of clusters minimizes the total cluster variance:

$$argmin \sum_{j=1}^{k} \sum_{i=1}^{n} ||X_{i,j} - \mu_j||^2$$

where the sum of the distance X of each point i in cluster j to its corresponding centroid of j. Distance is calculated in terms of input variables X and the j^{th} cluster centroid μ.

To identify clusters, the algorithm is a simple iterative procedure:

1. Initialize by randomly generating k-centroids.
2. Assign each point i to the closest cluster C.

- Recalculate the centroid coordinates for each C.

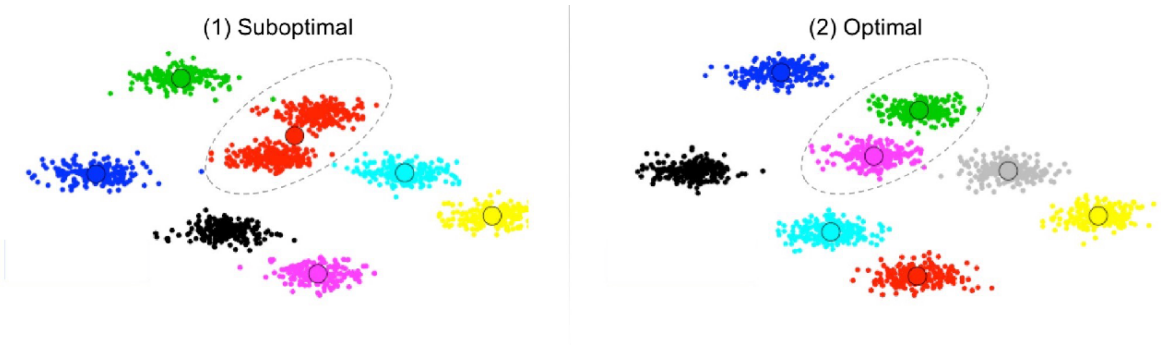


Figure 3: Comparison of a suboptimal result and optimal result

- Repeat step 2 until each point's cluster assignment no longer changes.

The first step is initialization in which k centroids are randomly generated. For each observation, calculate the Euclidean distance from each point to each centroid and assign to the closest centroid. This is known as the *assignment* step – all points take the label of its closest centroid. It is unlikely that this initial assignment is the "right" cluster, thus the algorithm will *update* the centroid coordinates by calculating the mean of each variable for each cluster. Upon doing so, this assignment-update procedure is iteratively repeated until the centroid coordinates no longer change between iterations.

3.2 Assumptions

While k-means is a simple algorithm, its performance and effectiveness is guided by a number of key assumptions at each step of computation.

- *Scale.* Similar to k-nearest neighbors, k-means treats sets of variables as coordinates. Each variable is assumed to have equal importance, which is unlikely to be the case with most data. To remedy this problem for cases where only continuous values are considered, input variables should be mean-centered and standardized ($\frac{x_i - \mu}{\sigma}$) to minimized unwanted biases due to scale. In cases with only discrete and categorical variables, each variable can be converted into a dummy variable matrix. In scenarios with mixed variable types, the authors recommend discretizing continuous values, then convert all variables into a dummy variable matrix so that all inputs are in a binary scale.
- *Missing Values.* K-means are unable to accommodate missing values as every coordinate is a necessary input. Thus, often times k-means models are usually reserved for complete data sets.
- *Stability of Clusters.* The initialization step creates k centroids at random, which can result in suboptimal and unstable clusters. The instability can be observed when comparing two sequential runs of the same algorithm with the same data – the points in each cluster may be entirely different! For example, a cluster that is visible to the eye may actually be divided among two or more clusters. While a number of factors influence unstable outcomes, we cover two key issues. First, the choice of k needs to be tuned, requiring a search for the value of k that optimizes some measure of quality. Second, as all variables have equal weight, highly dimensional training sets can have many local optima – there may simply be more nooks and cranieson the optimization surface into which the model may fall.
- *Choice of K.* Selecting the best value of k is arguably a subjective affair: there is a lack of consensus regarding how to identify k. In some cases, analysts choose an arbitrarily small value of k so that each cluster can inform a storyline. While this is convenient, it is quite subjective and some audiences may be more amenable than others. The *Elbow method* is a more technical approach in which k is chosen at the inflection point where an additional cluster does not significantly reduce the variance explained or reduction of error. By testing values a range of values of k, the elbow is the inflection point along the

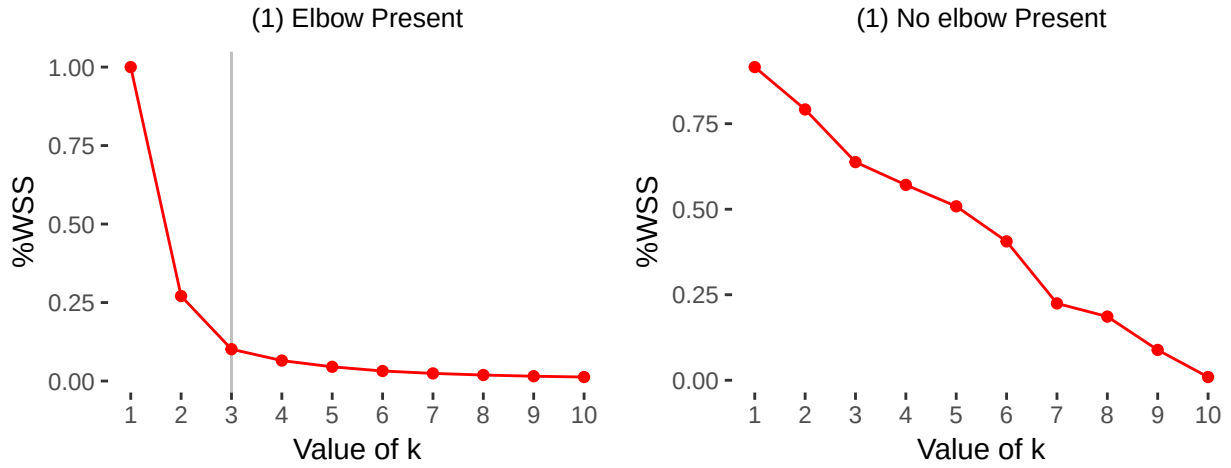


Figure 4: Elbow method: Choose k at the inflection point. (1) Inflection point identified at k = 3, (2) No inflection point identified.

total sum of squares curve. While the Elbow method sounds like a reasonable rule of thoumb, a clear inflection point is not always present.

An far more computationally intensive alternative relies on the *silhouette value*, which compares the similarity of a given observation i to observations within and outside its cluster. The silhouette $s(i)$ is defined as:

$$s(i) = \frac{b_i - a_i}{max(a_i, b_i)}$$

where a_i is the Euclidean distance between a point i and other points in the same cluster, b_i is the minimum distance between i and any other cluster the sample. The values of $s(i)$ fall between -1 and 1, where 1 indicates that an observation is well-matched with its cluster and -1 indicates that fewer or more clusters may be required to achieve a better match.

In some regards, k-means can be viewed as a *quick and dirty* technique for if a set of clusters are needed right away. It is surefire, processes quickly, and gets the job done. However, the random initialization may lead to inconsistent results. Also, the cluster results are devoid of context: which points are more associated than others? Are there subclusters that are more similar? Some of these shortcomings are overcome using Hierarchical Clustering.

3.3 DIY: Clustering for Economic Development

K-means can be used to segment the economic development market. Economic development corporations and chambers of commerce support local communities by attracting jobs and investment. Given the need for more jobs around the country grows, economic development initiatives are fierce affairs, sometimes pitting one community against another in bidding wars over tax benefits. In recent memory, Amazon.com announced new secondary headquarters in New York City and Arlington, VA after an exhaustive 20 city search.[9] The global manufacturer Foxxconn announced it will bring high tech manufacturing to Racine, WS.[10] And a long-standing 'border war' between Kansas City, MO and Kansas City, KS has seen a number of high profile companies like AMC Theaters move headquarters a mere miles, chasing economic benefits.[11]

Beyond the bidding war spectacle, there are other issues that factor into these siting decisions. Also, not all companies are as high profile as the ones described above, but are nonetheless important due to their

[9] https://blog.aboutamazon.com/company-news/amazon-selects-new-york-city-and-northern-virginia-for-new-headquarters
[10] https://www.jsonline.com/story/money/business/2018/10/02/foxconn-develop-downtown-racine-site/1499783002/
[11] https://www.economist.com/united-states/2014/03/22/the-new-border-war

contributions to the economy. For one thing, the prospective host region of new jobs should have the right economic conditions to sustain and foster the new opportunity. Suppose a tech executive in Santa Clara or Alameda in the Bay Area in California wanted to find another county with similar socioeconomic conditions. *Based on available data, how would one find a list of comparables?* The same question could be asked in reverse for economic developers: *what are other areas that are in direct competition?*

An analysis begins by first considering what observable variables are selling points for prospective businesses. Is it the size of the labor force? Is it the relative size of the target industry? Or perhaps it is related to education of the labor force or the local cost of employment? In any of these cases, publicly available economic data can be clustered using the k-means technique. Below, we illustrate a simple process of finding clusters of comparable economic activity, focusing on finding clusters associated with online tech industries.[12]

Set up. We start by loading the `cluster` library that has utilities for evaluating clustering results, then import a county-level data set that contains information for over 3,100 US counties.

```
library(cluster)
load("data/county_compare.Rda")
```

The underlying data is constructed from a variety of U.S. Census Bureau programs, in particular the American Community Survey, County Business Patterns, and the Small Area Income & Poverty Estimates.

- `fips`: Federal Information Processing System code that assigns a unique ID to each county
- `all.emp`: total employment[13]
- `pct.tech`: percent of the employed population in tech industry
- `est`: percent of company establishments in that industry
- `pov`: the poverty rate[14]
- `inc`: median household income[15]
- `ba`: percent that is college educated

To take a quick peek, we use the `head` function to extract the first three rows.

```
head(cty, 3)
```

fips	state	name	ba	all.emp	pct.tech	est	pov	inc
01001	AL	Autauga County	24.593	10790	0.399	0.235	14	54487
01003	AL	Baldwin County	29.547	61341	0.386	0.306	12	56460

Clustering. Before we apply k-means to the data, the data should be mean-centered and standardized so that all inputs are on the same scale. This can be easily done by using the `scale` function, the assigning the output to a new data frame `inputs`.

```
inputs <- scale(cty[,4:ncol(cty)], center = TRUE, scale = TRUE)
```

Let's get comfortable with the clustering process. As a dry run, we apply the `kmeans` function to the scaled inputs, specifying $k = 5$ for five clusters, and setting the seed to a constant so the analysis is replicable. The resulting object `cl` contains diagnostics about the clustering process, but also the coordinates of the centroids and the cluster assignment for each county (`cl$cluster`). When we tabulate `cl$cluster`, we find that each cluster is of a different size, suggesting that some counties do indeed cluster together more than others. We should ask *Why five clusters? Why not two or 50?*

[12] For simplicity, we define online tech industries using NAICS codes 5182, 5112, 5179, 5415, 5417, and 454111 although we recognize this may exclude subindustries that are rapidly growing in importancein tech.

[13] https://www.census.gov/programs-surveys/cbp.html

[14] U.S. Census Bureau, Model-based Small Area Income & Poverty Estimates (SAIPE) - https://www.census.gov/programs-surveys/saipe.html

[15] U.S. Census Bureau, Model-based Small Area Income & Poverty Estimates (SAIPE) - https://www.census.gov/programs-surveys/saipe.html

```r
#Dry run
  set.seed(123)
  cl <- kmeans(inputs, centers = 5)
  table(cl$cluster)
```

```
##
##    1    2    3    4    5
## 1018  451 1138  452   78
```

We identify the optimal value of k by comparing mean silhouette widths as calculated using the `silhouette` function in the `cluster` library. It requires two inputs: the cluster assignment and a dissimilarity matrix that shows the distances between each observation. The former is an output of `kmeans` and the latter is obtained using the `dist` function applied to the scaled input variables. The silhouette function calculates the silhouette width for each observation in the `sil` object. To compute the mean silhouette width, we simply take the mean of the third column of `sil`. For this dry run example, we can infer that the points are relatively well matched to their cluster given the positive value. As the value is closer to zero than one, some points are at the borderline between two clusters, thus we should test other values of k to optimize.

```r
#Calculate dissimilarity matrix
  dis <- dist(inputs)

#Calculate silhouette widths
  sil <- silhouette(cl$cluster, dis)

#Calculate mean silhouette width
  mean(sil[,3])
```

```
## [1] 0.218798
```

Optimizing k. To optimize k-means, we compute the mean silhouette width for values of $k \in \{2, 30\}$. For good measure, we combine the kmeans and silhouette functions into a function `km` that returns diagnostics given values of inputs `x`, desired number of clusters `k`, and dissimilarity matrix `d`.

```r
  km <- function(x, k, d){
      cl <- kmeans(x, centers = k)
      sil <- silhouette(cl$cluster, d)
      return(data.frame(k = k, sil = mean(sil[,3])))
  }
```

Then, we loop through each value of `k` in hopes of finding the optimum k, and plot the result. Ideally, the resulting silhouette curve will have (1) a global maximum, and (2) is not monotonically increasing. If the curve is monotonically increasing, this signals that the range of the k parameter is not wide enough and requires further testing. If the mean silhouette width is largest at values of $k \approx n$, then we can reasonably conclude that the problem is not well-suited for clustering as no two points are well-matched.

```r
  opt <- data.frame()
  for(k in 2:30){
    opt <- rbind(opt, km(inputs, k, dis))
  }
```

By plotting the silhouette curve, we find the global maximum at $k = 2$, separating the country into two relatively distinct groups. Bivariate plots can be used to tease out how clusters were formed. We see that higher income, better educated, high tech employment, large employment centers are clustered into one smaller group of $n = 526$ with the remainder of the country in another cluster of $n - 2611$. This is not surprising given the trend towards urbanization. In effect, the k-means algorithm was able to develop a multivariate strategy to partition affluent and better resourced communties from ones that are less well-off.

Usage. The real question is how one can use these clusters. An economic developer in a more affluent area

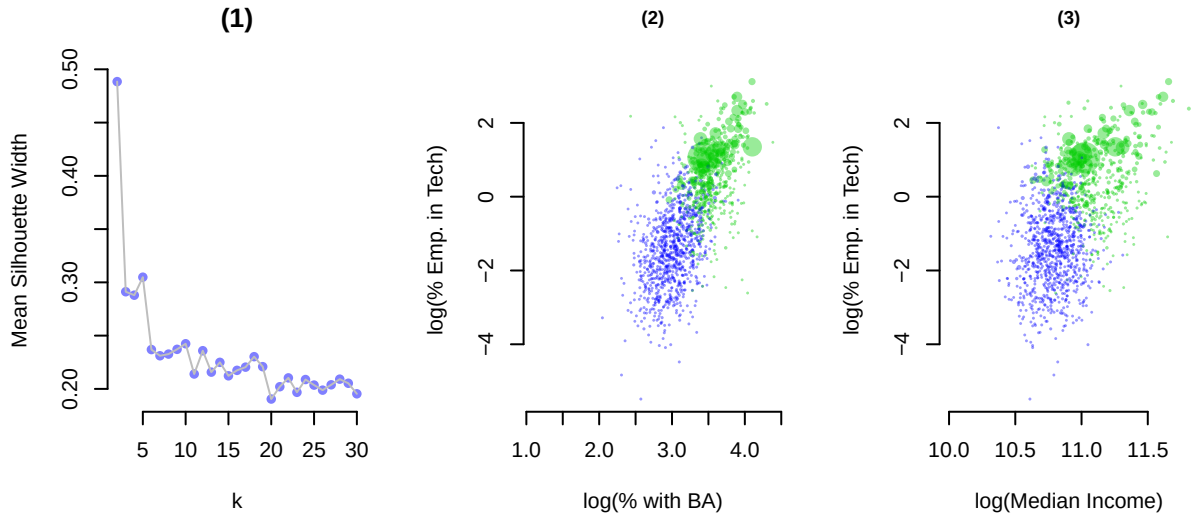


Figure 5: (1) Grid search for optimal k with respect to mean silhouette width, (2) and (3) Clusters for select input variables bivariate plots - scaled by total employment.

may be able to use the clusters to find comparable competitors. The smaller of the two clusters contains $n = 526$ counties, which is comprised of some of the nation's highest-tech counties such as San Francisco and Santa Clara in California as well as large cities such as New York City (New York City) and Seattle (King County, WA). But we also find less densely populated areas like Durham, NC and Arlington, VA – areas that in recent memory have been targeted by tech companies for new offices. By identifying similar competing areas, economic developers can research the competition and develop strategies to differentiate their offerings. Alternatively, an economic developer in the less developed cluster can use counties from the more affluent list as performance benchmarks to inform if policies are improving their economy.

Table 2: Counties with high tech concentrations and total employment greater than 100,000.

State	County	State	County
VA	Fairfax County	VA	Arlington County
CA	Santa Clara County	MA	Middlesex County
CA	San Mateo County	NJ	Middlesex County
MD	Howard County	CO	Boulder County
NC	Durham County	AL	Madison County

4 Hierarchical clustering

While k-means is an easy to use clustering algorithm, it can be improved upon using hierarchical clustering that captures relationships between observations in a tree-like structure. This has a number of notable advantages:

- The tree structure can be visualized in the form of *dendrograms* as well as provide context on how clusters are formed and if clusters are meaningful.
- Hierarchical clustering captures the interrelationships amongst observations in addition to cluster assignments.
- Hierarchical clustering is not initiated with random assignment, thus reducing the risk of inconsistent cluster assignments.

Why would k-means be used given these advantages? *Compute time.* As we will see later in this section, hierarchical clustering is not scalable as it requires a significantly longer run time and more memory storage to cluster even a moderate sized sample.[16]

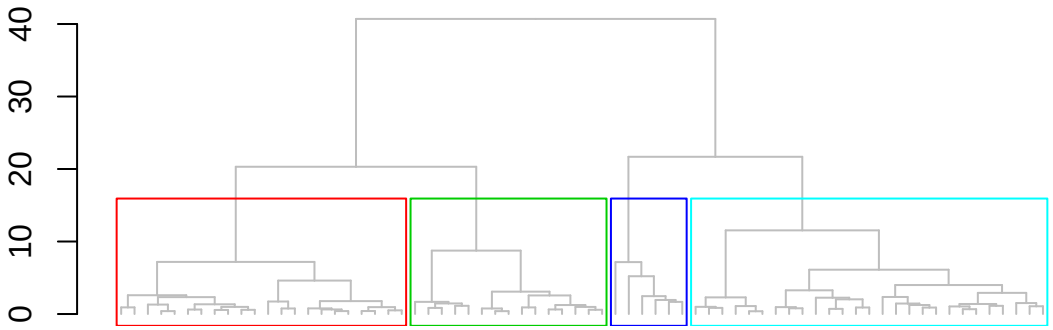


Figure 6: A Dendrogram.

1. Calculate a similarity measure s between all observations using a linkage method m. Each observation is initially its own cluster known as a *singleton*.

2. Do the following until there is only one cluster:

- Find the closest pair of clusters.
- Merge the pair into a single cluster.
- Recalculate distances from new cluster to all other clusters.

We can see that the process is quite computationally intensive. A sample of n observations requires $n-1$ iterations in which similarities s are recalculated between all clusters then grouping only the closest pair of clusters. The newly formed subclusters from each iteration are recorded, serving as a sort of paper trail of how all points are related to one another. When plotted, the paper trail resembles an inverted tree known as a dendrogram and has a striking similarity to decision trees as covered in the previous chapter. Each node in the tree is a distinct set of observations that are more similar to one another. The vertical axis indicates how similar clusters are to one another – if two or more nodes join together near the bottom of the chart, those clusters are more similar than clusters that merge closer to the top.

What if we would like to find $k = 4$ (or any number) clusters? Whereas k-means requires a value of k before executing the algorithm, HAC allows for selection after the algorithm converges. We identify clusters by finding the similarity value s that yields $k = 4$ – or drawing a horizontal line across each value of s that intersects exactly four edges in the dendrogram. Similar to k-means, the optimal value of k can be selected using the Elbow method or optimizing the mean silhouette width.

Linkage Methods. How is similarity measured between two potential clusters? Should a pair of points be used to to measure distance or should there be a group of points? The linkage method defines how the similarity measure s is calculated, choosing which points to use include in the calculation. The choice of linkage method not only changes the composition of clusters, but has a direct influence on the processing time.

The simplest is the *single linkage* or nearest neighbor linkage in which the closest pair of points, X_i and X_j, from two different clusters are merged together.

$$s_{ik} = min(d(X_i, X_j))$$

A close cousin is *complete linkage*, which measures distance as the two farthest points in two different clusters.

$$s_{ik} = max(d(X_i, X_j))$$

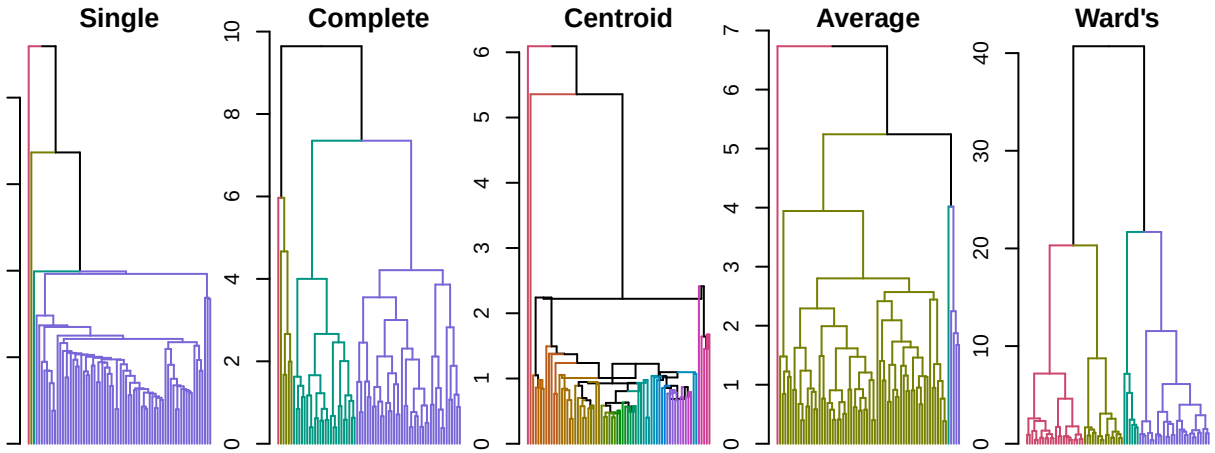


Figure 7: Effect of Linkage Methods on Clustering Results. Number of clusters set to k = 4.

Single and complete linkage place a greater weight on two observations and at times yield dendrograms that are not informative – the results might not capture the spirit of each cluster. In contrast, *Average Linkage* relies on more observations and comes in three varieties. Most commonly, the term average linkage refers to calculating similarity based on the average distances between all points in a pair of clusters.

$$s_{ik} = \sum_{i}^{k} \sum_{j}^{l} d(X_i, X_j)$$

A slight modification to the calculation results in *centroid linkage*, in which a centroid is defined as the set of means of each variable within a cluster. Distance is then calculated from using the centroid coordinates, $\bar{x_i}$ and $\bar{x_i}$, rather than directly from individual observations.

$$s_{ik} = d(\bar{x_i}, \bar{x_j})$$

Arguably, the most sophisticated and statistically-grounded linkage is *Ward's Method*. Rather than using direct measures of distance, Ward's approaches clustering from the lens of an analysis of variance (ANOVA). Two clusters are merged if it minimizes the increase in the sum of squares when compared to alternative candidate merges. This *merge cost* is concisely summarized as:

$$\Delta(x_i, x_j) = \frac{n_i n_j}{n_i + n_j} ||\bar{x_i} - \bar{x_j}||^2$$

where n_i and n_j are the sample sizes of a pair of clusters i and j. $\bar{x_i}$ and $\bar{x_j}$ are the centers.

Each of these linkage strategies have different effects on the shape of the dendrograms – some with clusters with more equal number of observations while others with one large cluster and many smaller clusters. Using the economic development data, we plot the dendrogram for each type of linkage. Single, centroid and average linkages are unable to separate observations into clean clusters when $k = 4$ – one cluster contains the majority of observations. In contrast, Ward's method and complete linkage are able to break the sample into more equal-sized, more organic clusters.

Ultimately, the choice of linkage method is a matter of trial and error, but Ward's generally is a sound choice with a more robust statistically-grounded solution.

4.2 DIY: Clustering of 311 Service Requests

To see the benefit of HAC, we apply it to data from NYC's 311 service – a public non-emergency hotline service that is offered in dozens of United States cities, allowing for residents to file complaints, report issues, and request information from their local government.[18] Originally an experiment implemented by the City of Baltimore in 1996[19], the service has evolved over time to meet the needs of citizens, scaling up via web-based versions of the service and in some cases scaling down depending on demand for services.

The multitude of 311 systems are a rich source of data that enable detailed analyses of sociodemographics of neighborhoods.[20] In New York City, for example, the 311 system has received over 200 million calls by 2015. A proportion of these calls result in a formal service request, giving a glimpse into what local residents experience as published via open data portals.[21] It is easy to imagine that residents in tree-lined suburban neighborhoods are concerned with different issues than those who live in high rise apartments. Using NYC's 311 data, we can cluster the population into segments of similar concern, which in turn can form the basis of how to communicate on key issues to local constituencies as well as targeting issues that may matter most. In addition, by establishing a baseline of concerns, we can track if certain types of resident concerns are trending or evolving.

Prepping the data. We begin by loading in NYC 311 service request data for 2016 that has been processed into 0.005 degree grid cells. In total, there are $n = 3770$ grid cells and 31 complaint types. Each variable contains the number of service requests logged in each grid cell. Note that we have omitted any complaint type with 1000 service requests in a 2016 and have consolidated similar complaint types. For example, *dead tree* and *fallen tree* have been rolled into an aggregate *tree* category.

```
#Load in pre-processed data
  load("data/nyc311.Rda")

#Check what's in the data
  dim(nyc311)
```

```
## [1] 3270   73
```

```
#Check column names
  colnames(nyc311)[1:10]
```

```
##  [1] "lat"               "lon"               "air.quality"
##  [4] "animal.abuse"      "appliance"         "asbestos"
##  [7] "blocked.driveway"  "boilers"           "broken.muni.meter"
## [10] "building.use"
```

We scale the complaint types to be mean-centered with unit variance, excluding the latitude and longitude fields. The resulting matrix `input311` is then converted into dissimilarity matrix using the `dist` function that maps the *Euclidean* distance between all points. This step is critical as it is the core ingredients on which linkage methods are applied. Note that hierarchical clustering can also be run on different types of distances such as *binary* that is more common when working with discrete variables.

```
#Extract necessary variables and scale them
  input311 <- scale(nyc311[,c(3:ncol(nyc311))])

#Calculate Euclidean distance matrix
  d <- dist(input311, method = "euclidean")
```

Clustering. The HAC procedure is neatly packaged into the `hclust` function, which is flexible in accommodating different linkage methods. The `hclust` function requires at minimum two arguments:

[18] http://www.govtech.com/dc/What-is-311.html
[19] https://www.citylab.com/city-makers-connections/311/
[20] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5645100/
[21] https://nycopendata.socrata.com/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

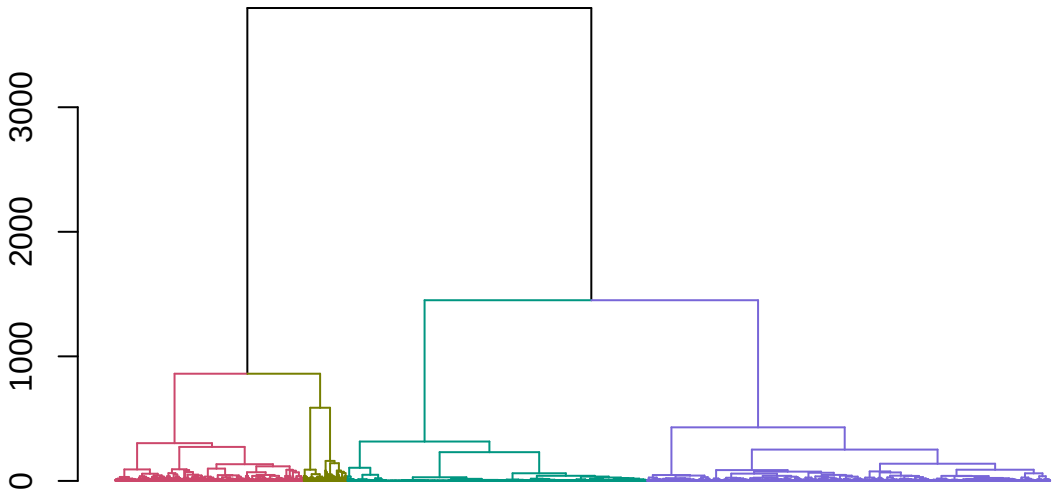


Figure 8: Dendrogram of 311 service requests by grid cell.

- d is a dissimilarity matrix from the `dist` function
- *method* is the type of linkage method that guides agglomeration, such as "single", "complete", "average", "centroid", "ward.D", among others. In this example, we apply the *ward.D* method.

The resulting tree and all its intricate relationships are stored in the object `hc` on which we rely quite a bit.

```
hc <- hclust(d, method = "ward.D")
```

With the tree grown, we can now visualize the dendrogram. Normally, the HCA object can be directly plotted using `plot(hc)`, but given the large number of singletons, it would be prudent to tidy the dendrogram. First, we convert `hc` into a dendrogram object `dend` and remove each singleton's labels, which would otherwise densely occupy the horizontal axis. Using the `dendextend` package, we can stylize the dendrogram so that an arbitrary group of $k = 4$ clusters are distinctly highlighted. The resulting `dend` object is then plotted.

Contexualizing clusters. As each observation is linked to a specific location in New York City, we can contexualize and better understand what complaints are captured within each cluster. The first step involves obtaining cluster assignments using the `cutree` function that relies on the HCA object `hc` and a desired number of clusters $k = 4$. The function returns a vector of cluster assignments.

```
  nyc311$groups <- cutree(hc, k = 4)
```

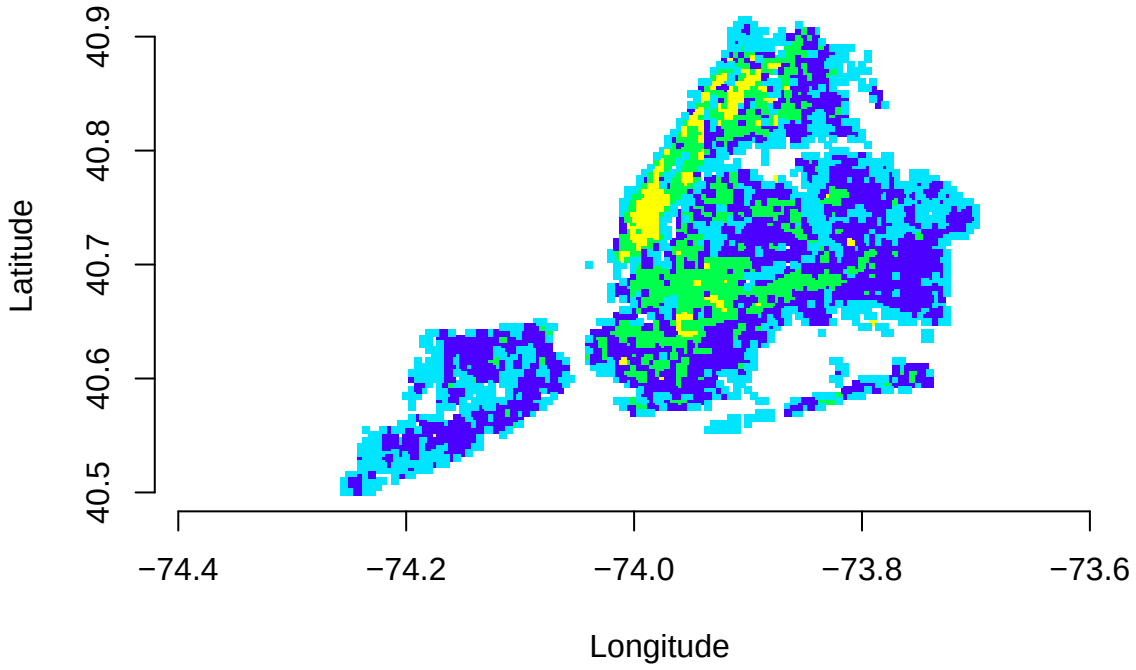
In order to geographically visualize the clusters, we will need to convert each cluster assignment in `groups` into a color. Fortunately, R comes standard witha number of readily available aesthetically color palettes such as `raindow()` that produces a divergent color spectrum, `heat.colors()` that produces a red-blue spectrum, among others. We rely on a togographic color spectrum `topo.colors` and produce a vector of colors with the same length as `groups` and `nyc311`.

When wondering the map, we find that the types of service requests are also spatially clustered: each type of neighborhood has different concerns: Yellow areas are high rise areas zones, green generally aligns with multistory apartments, dark blue are more suburban areas, and light blue correspond to coastal and parkland areas.

```
#Color code each assignment
  col.vector <- topo.colors(4)[nyc311$groups]

#Plot grid cells
  plot(nyc311$lon, nyc311$lat,
       col = col.vector,
       pch = 15, cex = 0.5,
```

```
       ylab = "Latitude", xlab = "Longitude",
       frame.plot = FALSE, asp = 1)
```



Some additional data manipulation is required to extract the most frequently occurring service requests for each group. First, the `aggregate` function summarizes the average number of requests per service request type by cluster group and record. The output `nyc.sum` is a 4×72 matrix. To narrow down the results to the top 10 requests per cluster, we need to rank requests in each cluster. To obtain a ranks matrix, we transpose `nyc.sum` and use the `rank` function in descending order (`-x`), doing so by column using the `apply` function.

```
#Get average volume of calls
  nyc.sum <- aggregate(. ~ groups,
                       data = nyc311[, -c(1:2)],
                       FUN = mean)

#Transpose so that each column is a cluster
  nyc.sum <- t(nyc.sum)

#Obtain descending rank for each column
  nyc.ranks <- apply(nyc.sum, 2,
                     function(x){
                       rank(-x, ties.method = "first")
                       })
```

We bring together the two tables in order to construct a 10×4 table: each column contains the top 10 service requests in rank sorted order with a label. The priorities of each group are similar, but the relative share and quantities of each service requests are different. Whereas suburban and coastal areas have relatively more tree-related complaints, multi-story and high-rise areas have far more noise and heat/hot water issues. From a policy perspective, cluster mapping is a simple way of segmenting constituencies and determine which city operations issues matter most for each community.

```
#Compile the top 10 ranks
  out <- lapply(1:4, FUN = function(x){
     top <- nyc.ranks[,x] <= 10
     res <- sort(nyc.sum[top,x], decreasing =  TRUE)
```

```
    res <- paste0(names(res), ": ", round(res))
    return(res)
  })

#Convert into table
  out <- do.call(cbind, out)
```

Table 3: Top 10 Service Requests by Cluster

1 - Suburban	2 - Coastal	3 - Multi-story	4 - High-rise
tree: 83	tree: 24	noise: 257	noise: 600
noise: 58	noise: 14	heat.hot.water: 172	heat.hot.water: 463
illegal.parking: 38	illegal.parking: 7	tree: 151	tree: 261
blocked.driveway: 37	heat.hot.water: 5	blocked.driveway: 82	homeless.person: 202
heat.hot.water: 27	blocked.driveway: 5	illegal.parking: 71	unsanitary.condition: 157
derelict.vehicle: 24	water.system: 5	unsanitary.condition: 62	paint.plaster: 133
water.system: 18	derelict.vehicle: 4	paint.plaster: 44	plumbing: 116
unsanitary.condition: 12	sewer: 3	plumbing: 41	water.system: 91
sewer: 12	sanitation.condition: 2	water.system: 41	illegal.parking: 81
missed.collection: 11	missed.collection: 2	door.window: 28	door.window: 80

5 Best Practices

In this chapter, we illustrated how two clustering techniques can be used to structure otherwise amorphous data problems. K-means and hierarchical clustering are simple techniques, but ultimately, they are best suited for exploratory problems – ones in which we need to data to speak for themselves. Unliked supervised learning problems in which a target is known, the insights drawn from clustering can be rather fickle and prone to manipulation due to the absense of a target. Nonetheless, these techniques are popular for informing hypotheses and narratives from microtargeting to agriculture.

We conclude this chapter by highlighting best practices that allow clustering projects to play an active and operational role in policy problems. First, be clear about the use case. Crafting a narrative about market segments and trends is a different focus than micro-targeting subgroups. The use case, in turn, has different implications for modeling choices. For example, the choice of k is dependent on the use case. Lower value of k lend themselves to more concise narratives. Optimized values of k through the Elbow method or Silhouette method are helpful for establishing baselines for more precise market segments. Second, as clustering treats input variables with equal weight, each variable should be standardized though we should note that there are varieties of clustering that allow inputs to have different weights. But perhaps more importantly, be cognizant of the variable types as those will dictate the type of similarity measure that can be employed. Third, while we do not cover the following, it is a logical extension to the classification chapter: clusters can support monitoring and contextualizing trends. Cluster assignments can serve as a baseline against which new data can be mapped using *1-nearest neighbor* classification. The implications are many. For political campaigns, clusters can be defined in one election cycle, then new data for subsequent election cycles can be contexualized relative to the baseline. This allows for tracking how trends evolve over time. For finance and economics, clustering of series based on their seasonal patterns can establish a baseline of consumer behavior. New information can be mapped to clusters to find similarities in behavior.