1 Chapter 10: Clustering

1.1 Opening Story

predictive market segmentation

demographics – play a huge role surveys would help

'Flag and Family Republicans' might receive literature on a flag-burning amendment from its sponsor, while 'Tax and Terrorism Moderates' get an automated call from [former New York mayor] Rudy Giuliani talking about the war on terror, even if they lived right next door to one another," Alex Lundry, the senior research director of TargetPoint – the firm Gage founded in 2003 – wrote recently in Winning Campaigns magazine."[1]

"For the general public, there was no way to know that the idea for the Parker contest had come from a data-mining discovery about some supporters: affection for contests, small dinners and celebrity. But from the beginning, campaign manager Jim Messina had promised a totally different, metric-driven kind of campaign in which politics was the goal but political instincts might not be the means. "We are going to measure every single thing in this campaign," he said after taking the job. He hired an analytics department five times as large as that of the 2008 operation, with an official "chief scientist" for the Chicago headquarters named Rayid Ghani, who in a previous life crunched huge data sets to, among other things, maximize the efficiency of supermarket sales promotions."[2]

Question is how to find people on the political spectrum and deliver ads to them to open their wallets. tiny little clusters of targeted voters and then sending highly customized ads to them https://fivethirtyeight.com/features/a-history-of-data-in-american-politics-part-2-obama-2008-to-the-present/

It turns out their brand preferences matter. [3]

"Parties search through voter file with voting history, consumer data (car owner or not, home owner or renter, gun owner, Sierra Club member, etc) to identify likely target voters. Parties conduct large sample survey of its universe of targeted voters to see what issues are bothering them, motivating them, angering them. Based on this survey, campaigns "cluster" sub-groups of voters in terms of profile of interests and concerns: middle-income Democratic blue-collar gun owners in rural Michigan, for example."[4]

"The more information he has, the better he can group people into"target clusters" with names such as "Flag and Family Republicans" or "Tax and Terrorism Moderates." Once a person is defined, finding the right message from the campaign becomes fairly simple.

Goal: Find new patterns via clustering and latent features Data: Only input features Eval: Many evaluation measures or techniques, but none that are standard Complexity: Statistical techniques tend to be simple and iterative

1.2 Clustering is Natural

Perhaps the most powerful pattern recognition machine is the human brain, using the signal collected by the eye. Without knowing context, this dynamic duo can pick out patterns that in turn form the basis of knowledge and inference. And on their own, they are remarkable instruments to help us humans navigate our world. But, as we all experience, it is not easy to clone oneself to do human tasks at scale.

There is an abundance sociodemographic data published by statistical agencies around the world, but for the data to be more locally applicable and actionable, it needs to be more geographically granular. There are restrictions placed on governments publishing data in order to protect the privacy of individuals, thus resulting in coarse units of reporting. There are, for example, plenty of uses of where are people located

[1] http://www.washingtonpost.com/wp-dyn/content/article/2007/07/04/AR2007070401423.html
[2] http://swampland.time.com/2012/11/07/inside-the-secret-world-of-quants-and-data-crunchers-who-helped-obama-win/
[3] https://www.bloomberg.com/news/articles/2016-03-24/-wheel-of-fortune-cashes-in-on-2016-political-ads
[4] http://www.nbcnews.com/id/15292903/ns/politics-tom_curry/t/mechanics-micro-targeting/#.XANCoZNKgxc

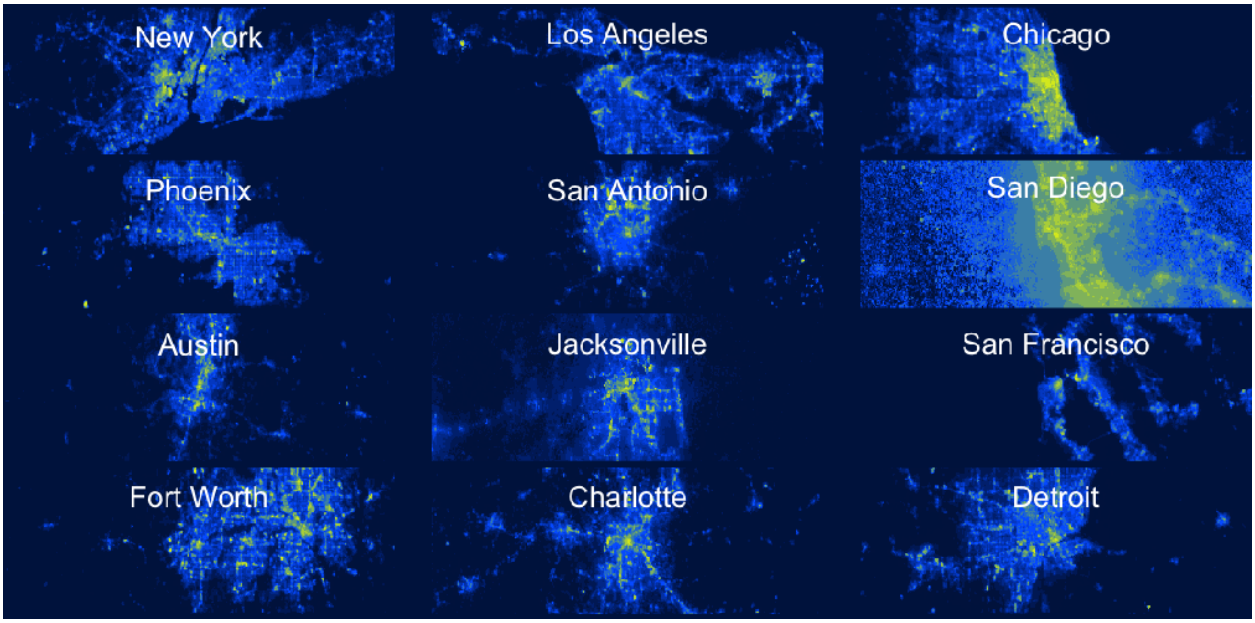


Figure 1: Night time imagery from the NOAA-NASA Suomi NPP Satellite's Visible Infrared Imaging Radiometer Suite (VIIRS)

within a zip code or county. Urban planners could better zone. Emergency responders can have a better sense of building assets and density of people in order to better plan respond to a natural disaster.[5]

Using nighttime satellite imagery from the Suomi NPP satellite, one could simply trace the outline of each city using our eyes as a guide, then apply the outline as a mask to refine the geographic shape of publicly available data. But manually doing this task is simply not scalable to billions of pixels of imagery. Like all things data science, we could apply clustering techniques at scale, extracting the areas in satellite imagery that emit a distinguishable level of light.

This raises the question: *what distinguishes a light pixel from a dark one?* This is a strikingly similar idea to separability, a measure of dissimilarity. If we think of each pixel as an element in a vector of light intensity, we could treat satellite imagery as data. This data in turn has statistical properties that can be exploited to scale our problem. Let's take the case of San Francisco, which is a mass of human activity on a peninsula on the Pacific Coast. When we plot a histogram of logarithm transformed radiances from the satellite imagery, we find a bimodal distribution, which implies that there are at least two types of activity that reflected in this data – a mixture of distributions. A simple way to separate one distribution from the other is to define each activity by its mean, located at the peaks.

To classify each pixel as either light or dark, we can use a simple dissimilarity measure. For continuous values, dissimilarity can be as Euclidean Distance:

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

Alternatively, Manhattan or binary distances could be used for discrete variables or Gower distances for mixed variable types. Regardless of distance measure, we can then calculate distance between each point and each peak, assigning each pixel to the closest peak. While this simplistic example is effective, it can be automated and scaled using one of the many commonly used clustering algorithms.

[5]Risk analysis paper citation

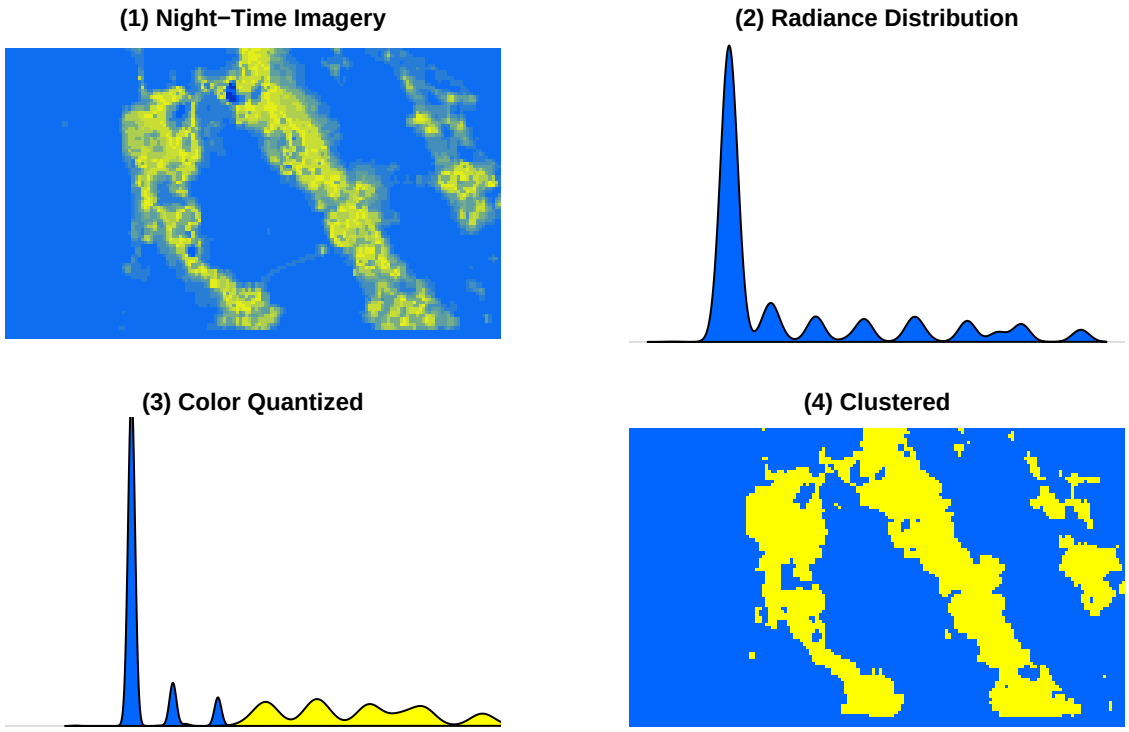


Figure 2: Applying clustering to extract likely inhabited areas.

In this chapter, 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.

1.3 K-Means

1.3.1 How It Works

K-means clustering identifies observations that belong to a latent group by treating variable sets as coordinates in n-dimensional space. As the true value of k is not known, the user defines the parameter that controls the number of clusters that will be returned upon running the algorithm.

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 all input features x and the j^{th} cluster centroid μ.

To identify clusters, algorithm takes an iterative, computationally intensive, but straightforward set of steps:

1. Randomly assign k-centroids;
2. Assign each point i to the closest cluster C;

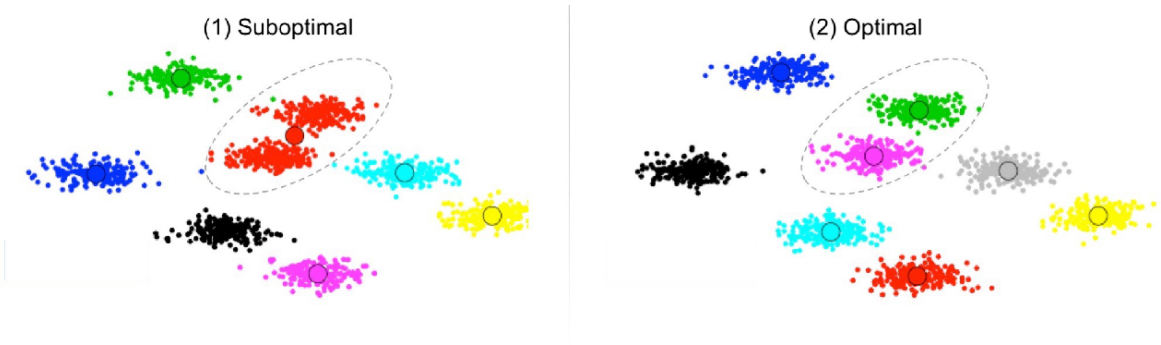


Figure 3: Comparison of a suboptimal result and optimal result

3. Recalculate the centroid coordinates for each C;
4. Repeat steps 2 and 3 until point assignments no longer change

The first step involves selecting k-number of random centroids from the feature space and assigning each centroid a label. For each observation in the data, calculate the Euclidean distance from each point to each centroid, then assigning observations 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 likely suboptimal, thus the algorithm will *update* the centroid coordinates by calculating the mean value of each feature within each cluster. Upon doing so, this assignment-update procedure is iteratively repeated until the centroid coordinates no longer change between iterations.

1.3.2 Guiding 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 variables as coordinates. Thus, each feature is assumed to have equal importance, which in turn means that results may be inadvertently biased simply by the scale and variances of underlying features. To remedy this problem, input features should be mean-centered standardized ($\frac{x_i - \mu}{\sigma}$) or otherwise transformed to reduce scaling effects. Note, however, that the influence of scaling may not always be removed. For example, a data set containing both continuous and binary features would likely perform quite poorly as Euclidean distances are not well-suited for binary. Thus, where possible, apply k-means when the formats are homogeneous, doing so using Euclidean L2-distances for continuous and binary distances for matrices of discrete features.
- *Missing Values.* K-Means do not handle missing values as each data point is essentially a coordinate. 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 lead to suboptimal and unstable clusters. The instability means that two sequential runs of the same algorithm with the same data may yield different results! 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 (see the following bullet point for further discussion). Second, as all variables have equal weight, highly dimensional training sets can have many local optima – there may simply be more nooks and cranies on the optimization surface into which the model may fall. Applying dimensionality reduction techniques (e.g. PCA) can improve stability.
- *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. One method known as the *Elbow method* chooses k at the inflection point where an additional cluster does not significantly reduce the variance explained or reduction of error.

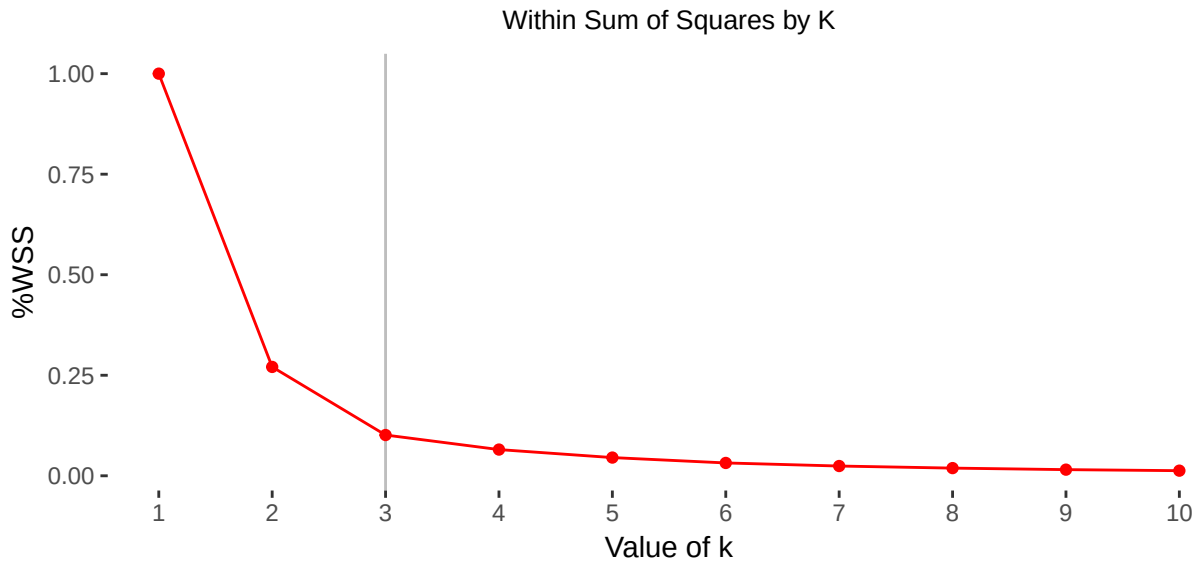


Figure 4: Elbow method: Choose k at the inflection point

The simplest method of identifying the inflection point can be seen by plotting the percent WSS over all values of k that were tested. This approach is deceptively simple as the inflection point might not manifest itself in some data sets.

An alternative but far more computationally intensive approach involves calculating the *silhouette*, 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 to 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. Note that silhouettes do not scale well with very large data sets as a $n \times n$ similarity matrix (e.g. distance between all points to all points).

(TO DO)

- K-mediods can be a useful alternative to k-means if input variables are of mixed type
- Also helpful to know that k-means yields "flat" clusters: We only know which observations fall into which cluster, but we do not know which observations are more related than others.

1.3.3 DIY: Clustering for Economic Development

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 is a fierce affair, sometimes pitting one community against another in bidding wars on tax benefits. In recent memory, Amazon.com announced new secondary headquarters in New York City and Arlington, VA after an exhaustive 20 city search.[6] The global manufacturer Foxxconn announced it will bring high tech manufacturing to Racine, WS.[7] And a long-standing 'border war' between Kansas City, MO and Kansas City, KS has seen a

[6] https://blog.aboutamazon.com/company-news/amazon-selects-new-york-city-and-northern-virginia-for-new-headquarters
[7] https://www.jsonline.com/story/money/business/2018/10/02/foxconn-develop-downtown-racine-site/1499783002/

number of high profile companies like AMC Theaters move headquarters a mere miles, chasing economic benefits.[8]

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 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.[9]

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[10]
- `pct.tech`: percent of the employed population in tech industry
- `est`: percent of company establishments in that industry
- `pov`: the poverty rate[11]
- `inc`: median household income[12]
- `ba`: percent that is college educated

```
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
06029	CA	Kern County	15.665	190503	1.123	0.220	22	49812
54025	WV	Greenbrier County	19.582	10905	0.064	0.437	16	38784

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`.

```
#Scale input variables
  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 centroids, and setting the seed to a constant so the analysis is replicable.

[8] https://www.economist.com/united-states/2014/03/22/the-new-border-war

[9] 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.

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

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

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

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 centroids? Why not two or 50?*

One common issue with clustering is that the value of k can be arbitrarily chosen, especially as it is an exploratory tool. In some scenarios, a low value of k is chosen for convenience as a way to articulate high-level patterns – essentially creating profiles that are empirically inspired. Greater values of k may minimize overall fitness tatistics, but may also result in a large number of small clusters that contain very similar information compared with other clusters. Two or more clusters could be approximately the same, but are arbitrarily split. This is an artifact of the choice of k – the k-means will force a solution for k clusters even if some are statistically similar.

Both strategies are valid when the objective is to surface insights. We recommend thus recommend choosing an arbitrarily low value of k, then comparing that value with one that maximizes a fitness statistic like the mean silhouette width.

```
#Dry run
  set.seed(999)
  cl <- kmeans(inputs, centers = 5)
  table(cl$cluster)
```

```
##
##    1    2    3    4    5
##  460 1491  907   82  197
```

Silhouette widths can be easily 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 `daisy` function applied to the scaled input variables. On its own, the results of the silhouette are not all that informative without a comparison to other values of k.

Note that the resulting object from the silhouette function computes the Silhouette width for each observation, recorded in the third column. For each value of k, we will compute the mean silhouette on this column.

```
#Calculate dissimilarity matrix
  dis <- daisy(inputs)

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

```
##  silhouette [1:3137, 1:3] 2 2 3 3 2 3 3 2 3 3 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "cluster" "neighbor" "sil_width"
##  - attr(*, "Ordered")= logi FALSE
##  - attr(*, "call")= language silhouette.default(x = cl$cluster, dist = dis)
```

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`, number of centroids `k` and dissimilarity matrix `d`. Then, we loop through each value of `k` in hopes of finding the optimum k, and plot the result.

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

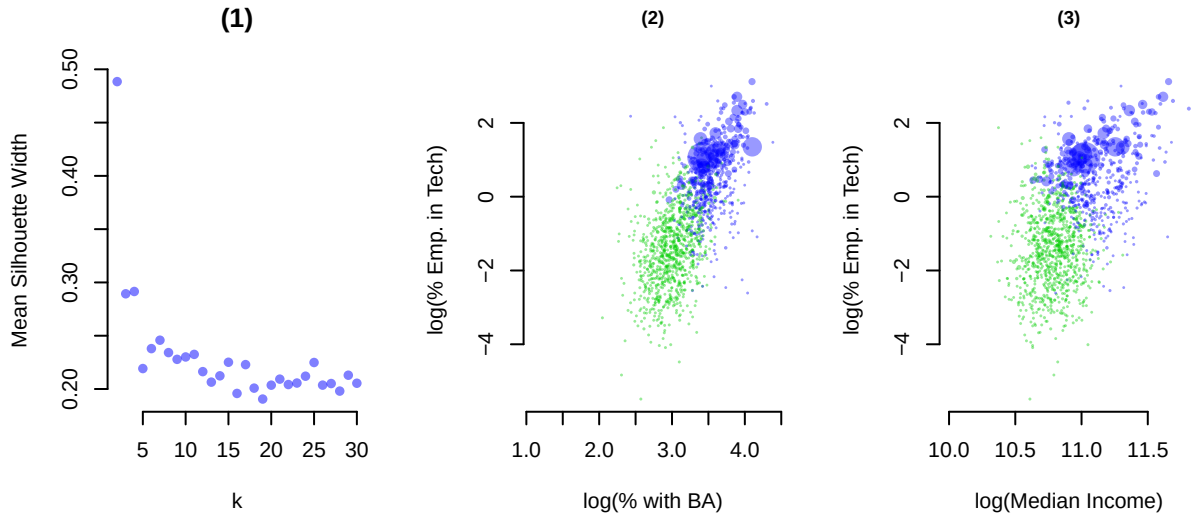


Figure 5: (1) Grid search for optimal k with respect to Mean Silhouette Width, (2) and (3) Clusters for select input variables bivariate plots - scaled by total employment.

```
  }

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

A couple points of practice. First, the optimal break is $k = 2$ – indicating that the data most naturally can be broken into pieces if need be. But this does not mean the clusters are optimal. Second, the peak mean silhouette should be somewhere to the left with declining values as k increases. This is an indication that clusters are naturally occurring in this data set. If the average silhouette grows with k, we would want to test k until a maximum is found. But there is always the chance that the maximum is reached at $k = n$ in which case the problem may not lend itself to a meaningful cluster analysis.

Upon plotting bivariate plots of each input variable, we can see that some characteristics have more power in separating counties than others. Larger employment centers tend also to have some tech and greater incomes as well – not surprising given the trend towards urbanization. In effect, the k-means algorithm provided an efficient strategy to partition affluent and better resourced communties from ones that are less well-off.

The smaller of the two clusters contains $n = 404$ counties, which contains most of the nation's high-tech counties including 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 are able to see that less densely populated areas like Durham, NC and Arlington, VA also make the list of comparables. By identifying these likely competing areas, economic developers can better do their research on the competition and further differentiate their offerings. The same exercise could likewise be applied to understand customer behavior, by clustering their behavior in order to produce profiles. In either case, k-means is very much an exploratory technique that helps inform initial hypotheses. In policy setting, the choice of input variables may be more a matter of the objective of the priority than the math itself. Nonetheless, treat the data gently.

clusters	fips	state	name
2	51059	VA	Fairfax County
2	06085	CA	Santa Clara County
2	06081	CA	San Mateo County

clusters	fips	state	name
2	24027	MD	Howard County
2	37063	NC	Durham County
2	51013	VA	Arlington County
2	25017	MA	Middlesex County
2	34023	NJ	Middlesex County
2	08013	CO	Boulder County
2	01089	AL	Madison County

1.4 Hierarchical clustering

An alternative to k-means is hierarchical clustering, which captures cluster relationships in a tree-like structure. Unlike k-means, hierarchical clustering is maps the relationship between all observations, which has a number of notable technical benefits:

- Visual analysis of the tree structure through *dendrograms* can help build an understanding of how clusters are formed and if clusters are meaningful.
- Unlike k-means, hierarchical clustering provides context of the interrelationships amongst observations and allows cluster selection a more flexible process.
- Lastly, hierarchical clusters do not suffer from the inconsistent cluster assignments that are common among k-means clusters.

Why would k-means be used given these advantages? *Time*. As we will see later in this section, hierarchical clustering is so computationally intensive that it requires a significantly longer run time and greater memory storage to cluster even a moderate sized sample.[13]

```
## Loading required package: dendextend
```

```
## Warning: package 'dendextend' was built under R version 3.4.4
```

```
##
## ---------------------
## Welcome to dendextend version 1.9.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:raster':
##
##     rotate
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

[13] ref required

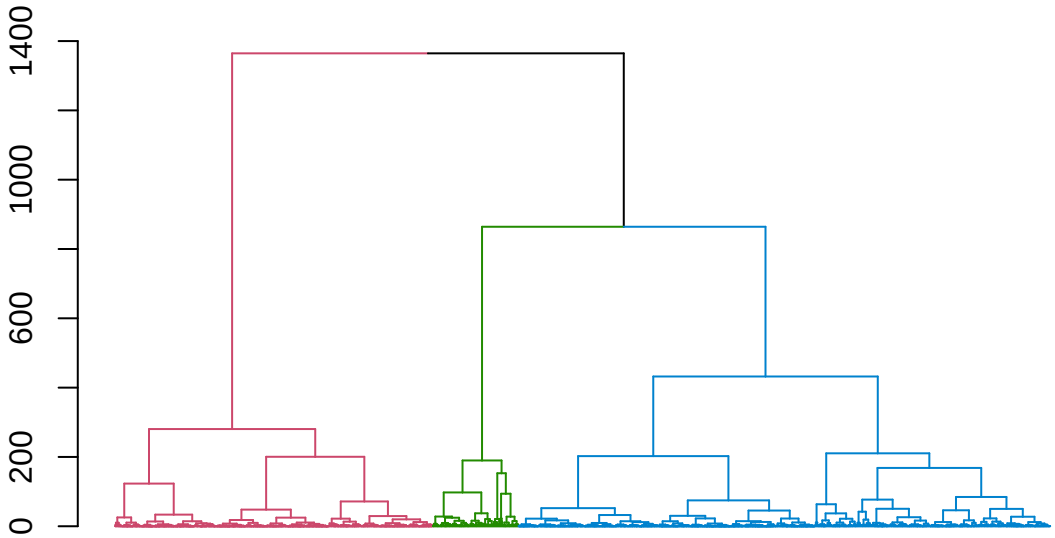


Figure 6: An example dendrogram

```
## Warning in `labels<-.dendrogram`(`*tmp*`, value = ""): The lengths of the
## new labels is shorter than the number of leaves in the dendrogram - labels
## are recycled.
```

1.4.1 How It Works

Hierarchical clustering is comprised of two forms: divisive and agglomerative. Hierarchical Divisive Clustering takes a top-down approach and is reminiscent of decision tree learning. It identifies clusters by recursively splitting a sample into smaller partitions.[14] Hierarchical Agglomerative Clustering (HAC) in contrast takes a bottom-up approach, clustering individual observations by distance until all observations are part of one mega cluster. In this section, we focus on HAC given its popularity in the R programming language.

Process. How HAC clusters observations is simple, but computationally intensive:

1. Calculate distance d between all points using a linkage method m. Each point is its own cluster known as a *singleton*.
2. Do 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.

At each step in the clustering algorithm, the similarity measure between each pair of clusters is recorded. This nifty feature means that an arbitrary number of k clusters can be identified by simply finding the value of the similarity measure that horizontally cuts across k edges in the dendrogram.

Linkage Methods. A key input into this process is the linkage method m, which dictates which observations are used in calculating distance between clusters. The choice of linkage method not only changes the composition of clusters, but has a direct influence on the processing time – more exhaustive the linkage, longer it will require to complete.

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.

[14]

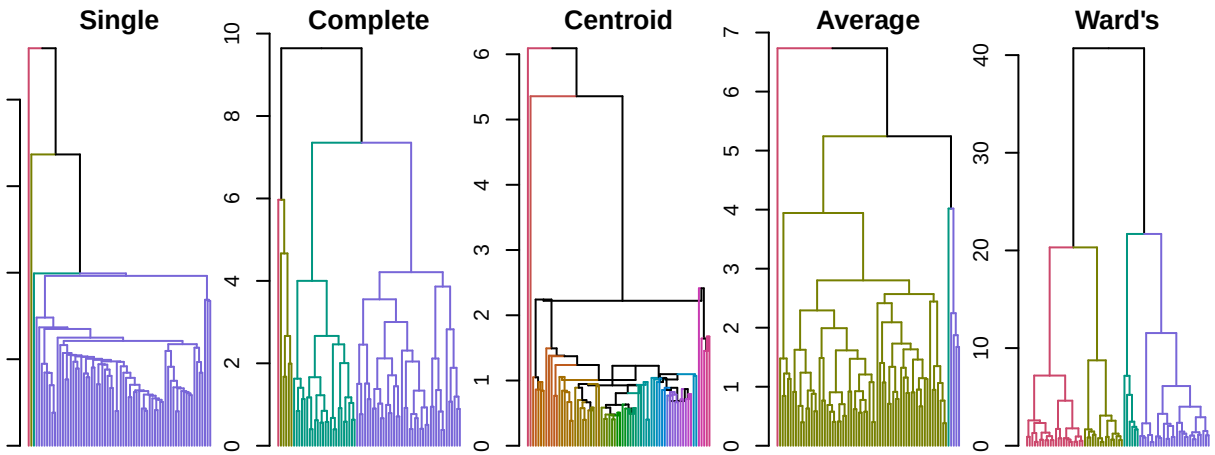


Figure 7: Effect of Linkage Methods on Clustering Results. Number of clusters set to k = 4.

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

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

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

Average Linkage is implemented in three different varieties. Most commonly, average linkage refers to calculating similar as the mean of distances between all points in two clusters.

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

A slight modification to the calculation can result in *centroid linkage*, in which a centroid is defined as the coordinate set that results from taking the mean of each variable within a cluster. The resulting centroids, $\bar{x_i}$ and $\bar{x_i}$, are used as the basis of calculating distance rather than directly 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). From among all the prospective merges, two clusters are merged if it minimizes the increase in the sum of squares. This *merge cost* can 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 resulting more distinct clusters than others. Using a random subset of the economic development data, we find that single, centroid and average linkages do a relatively inadequate job in separating observations into clean clusters at $k = 4$ – one cluster contains the majority of observations. In contrast Ward's method and complete linkage are able to break the sample into more equally sized clusters. Ultimately, the choice of linkage is a matter of trial and error, but Ward's will generally provide the more robust solution.

1.4.2 DIY: Clustering of 311 Service Requests

311 is 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.[15] Originally an experiment implemented by the City of Baltimore in 1996[16], 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 also are a rich source of data that can be used to analyze sociodemographics of neighborhoods.[17] 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.[18] 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.

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 *tree*.

```
#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"
```

Per usual, 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")
```

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:

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

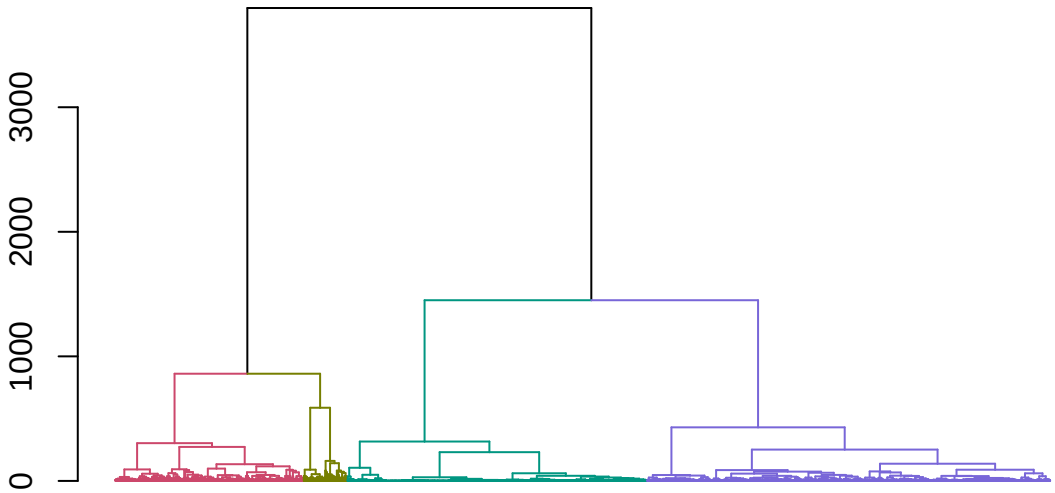


Figure 8: Dendrogram of 311 service requests by grid cell.

- 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, we will take the liberty to tidy up the dendrogram. First, we convert `hc` into a dendrogram object `dend` and remove each singleton's labels, which would otherwise densely line the x-axis. Using the `dendextend` package, we can stylize the dendrogram so that an arbitrary group of $k = 4$ tree branches can be neatly identified and visualized. The resulting `dend` object is then plotted.

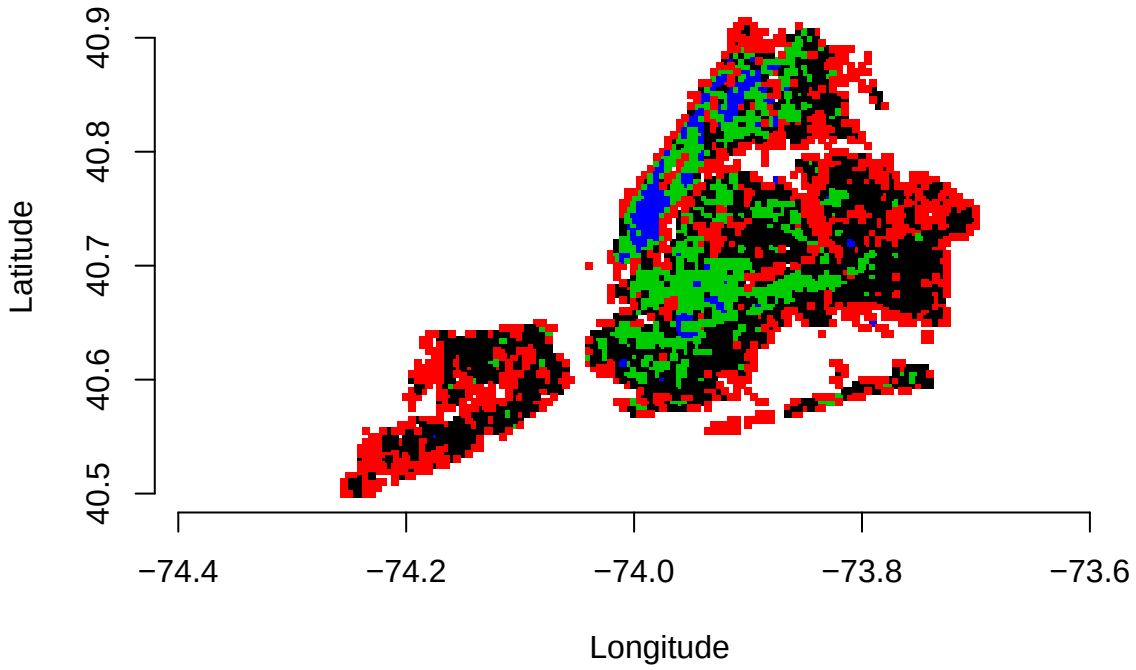
The number of clusters k can be easily optimized using the sihouette method described in the k-means section. For brevity, we arbitrarily select a value of $k = 4$ and dissect how each cluster differs.

Dissect top 10 request types

```
#Label groups
  groups <- cutree(hc, k = 4)
```

It turns out that HAC does a reasonable job at identifying distinct patterns in the NYC service request landscape: Purple are high rise areas in Manhattan, Light Orange indicates multistory apartment areas, Light Blue are more suburban and Green correspond to coastal and parkland areas.

```
  plot(nyc311$lon, nyc311$lat,
       col = groups,
       pch = 15, cex = 0.5,
       ylab = "Latitude", xlab = "Longitude",
       frame.plot = FALSE, asp = 1)
```



Commentary on how this could be used

- finding ways to be more proactive if certain types of services happen jointly
-