1 Chapter 10: Clustering

1.1 Opening Story

Lorem ipsum dolor amet pug pop-up authentic seitan brunch. Health goth +1 wayfarers glossier blog knausgaard stumptown green juice neutra williamsburg jianbing ethical. Leggings blue bottle meditation, organic intelligentsia offal knausgaard craft beer tacos butcher cred shoreditch semiotics. Blue bottle deep v VHS skateboard, vaporware pug glossier. Cardigan pitchfork taiyaki plaid art party quinoa. Tattooed vice kickstarter, jianbing woke la croix humblebrag ethical disrupt 8-bit whatever vape lo-fi selvage.

Bushwick poutine offal, lyft roof party asymmetrical next level edison bulb cloud bread. Brunch deep v echo park raw denim synth XOXO disrupt migas. Health goth yuccie taxidermy, pitchfork artisan kogi af deep v wayfarers coloring book farm-to-table austin twee. Sriracha raclette hexagon, selvage scenester small batch intelligentsia retro salvia vice dreamcatcher poutine flexitarian health goth. Trust fund kogi tacos freegan. Keffiyeh chillwave subway tile selfies kombucha hot chicken swag jianbing blue bottle pop-up pork belly williamsburg chartreuse vegan la croix. Offal pickled DIY, normcore af bespoke activated charcoal health goth.

Asymmetrical slow-carb hell of lomo live-edge kinfolk jean shorts synth 3 wolf moon gochujang la croix tattooed pork belly. Prism enamel pin hella shabby chic meditation stumptown, actually artisan. Polaroid meggings small batch tote bag green juice waistcoat meh banh mi. Authentic flannel put a bird on it, dreamcatcher tousled vegan pabst.

Pitchfork pinterest occupy, PBR&B jianbing subway tile activated charcoal live-edge scenester fam church-key iceland disrupt etsy freegan. Asymmetrical drinking vinegar schlitz pop-up farm-to-table brooklyn live-edge tbh squid trust fund microdosing selvage gluten-free. Tacos semiotics palo santo activated charcoal pabst biodiesel kogi you probably haven't heard of them wolf tumeric woke kickstarter snackwave. Taiyaki mustache artisan etsy yr listicle viral umami. Synth blue bottle migas, echo park stumptown man braid hexagon enamel pin umami sartorial pitchfork.

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 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.[1]

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

[1] Risk analysis paper citation

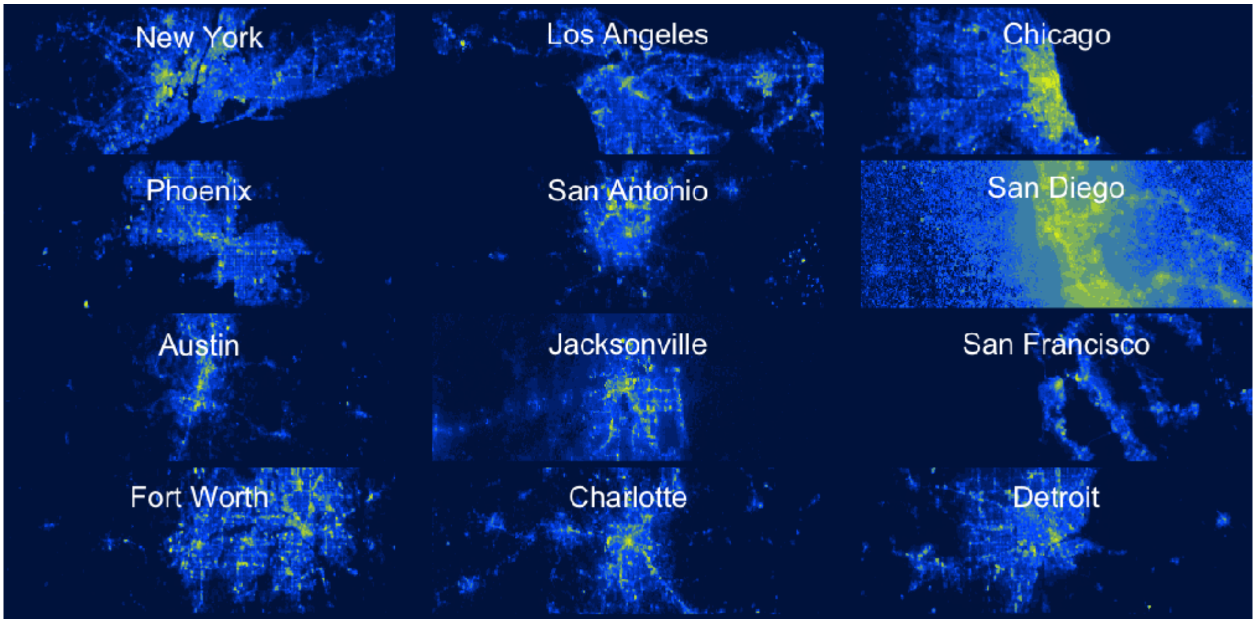

Figure 1: Night time imagery from the NOAA-NASA Suomi NPP Satellite's Visible Infrared Imaging Radiometer Suite (VIIRS)

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:

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

And 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.

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.

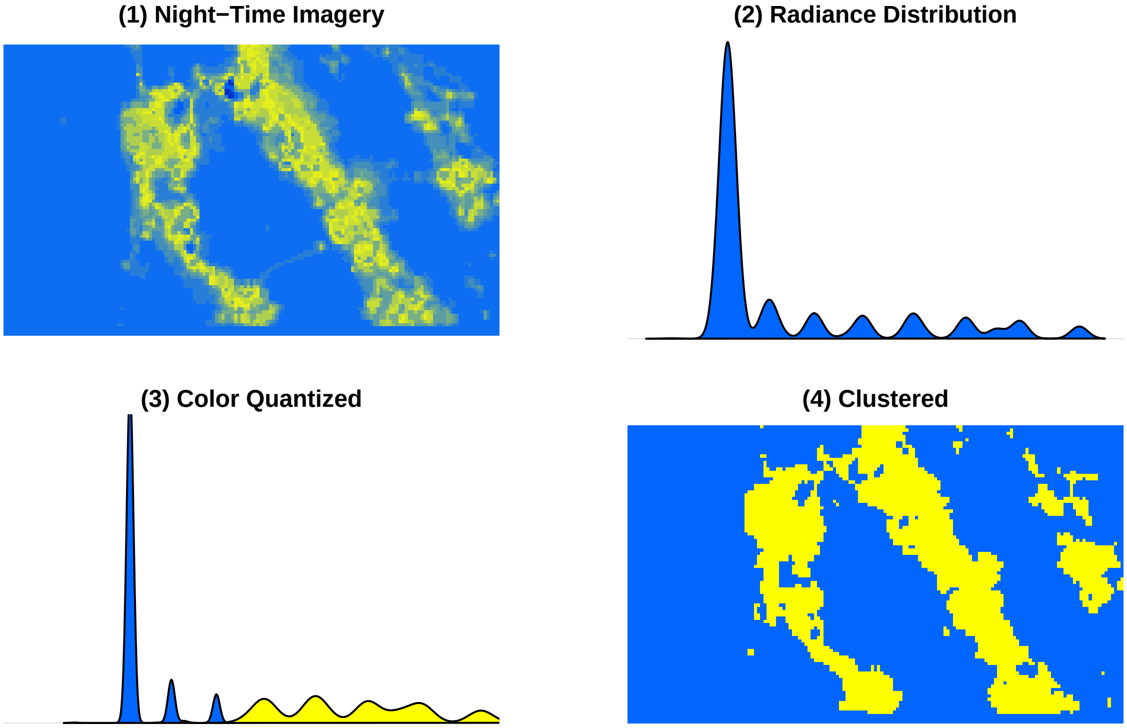

Figure 2: Applying clustering to extract likely inhabited areas.

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;
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.

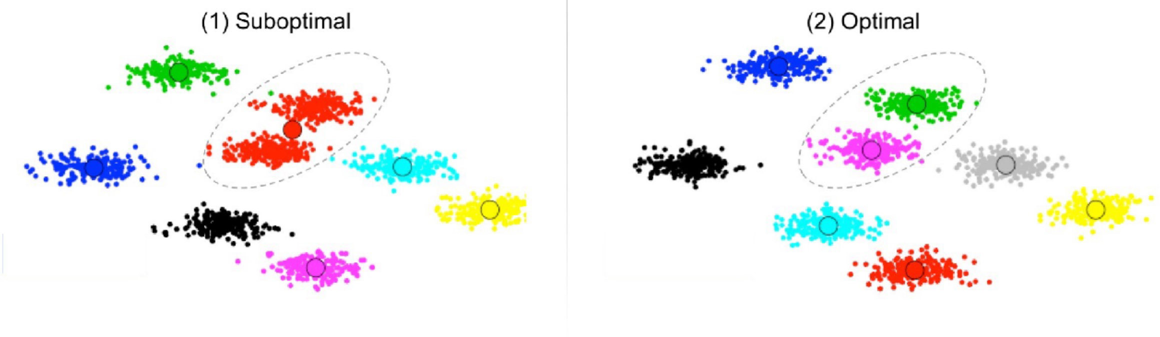

Figure 3: Comparison of a suboptimal result and optimal result

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. 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

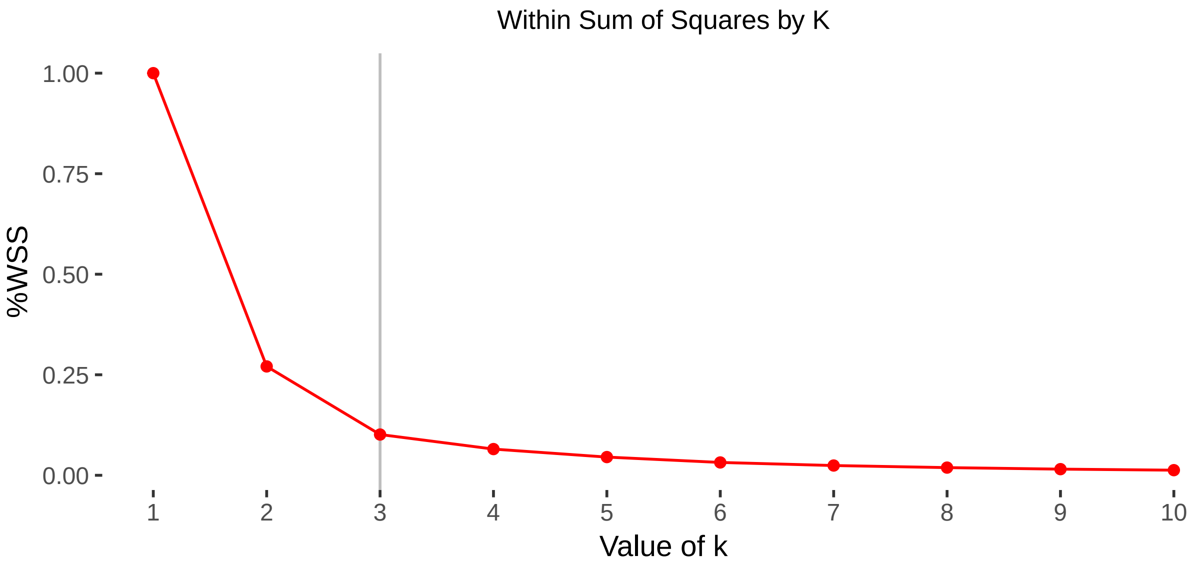

Figure 4: Elbow method: Choose k at the inflection point

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). Often times, a smaller sample should be used to enable the use of this method.

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.[2] The global manufacturer Foxxconn announced it will bring high tech manufacturing to Racine, WS.[3] 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.[4]

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 CEO in Santa Clara or Alameda in the Bay Area in California wanted to find another county with similar 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?*

It 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

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

comparable economic activity, focusing on finding clusters associated with online tech industries.[5]

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.

```r
  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[6]
- `pct.tech`: percent of the employed population in tech industry
- `est`: percent of company establishments in that industry
- `pov`: the poverty rate[7]
- `inc`: median household income[8]
- `ba`: percent that is college educated

```r
  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`.

```r
#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. 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

[5] 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.

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

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

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

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])))
  }

#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.

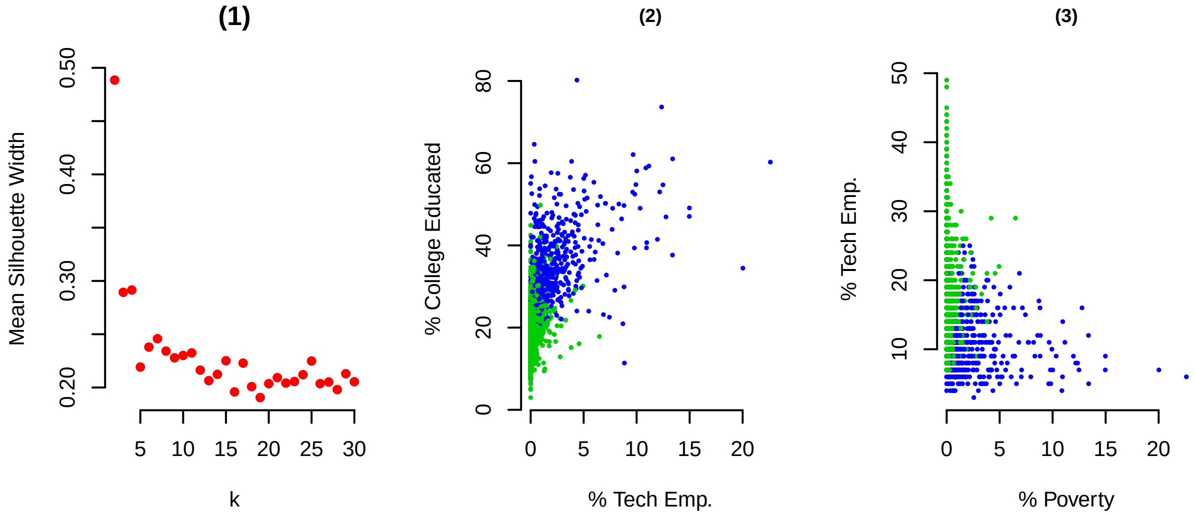

Figure 5: (1) Grid search for optimal k with respect to Mean Silhouette Width, (2) and (3) Clusters for select input variables bivariate plots.

Upon plotting bivariate plots of each input variable, we can see that there are not natural occurring clusters, but rather the kmeans algorithm provided an efficient means 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 tool that helps inform initial hypotheses when targets are not yet available.

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

Whereas k-means initializes on random centroids, hierarchical clustering take a more computationally costly ground-up approach:

```
Calculate distance d between all points

All points are start as their own clusters (singletons)
Do until there is only one cluster:
  Find the closest pair of clusters in terms of linkage distance
  Merge into a single cluster
  Recalculate distances from new cluster to all other clusters
Stop when all points are in one cluster
```