Map Reconstruction

Lawson Busch Amanda Doan Raven McKnight Joseph Wriedt

April 30, 2018

Abstract

In this report, we reconstruct a map of the United States given distances between major cities. We use singular value decomposition to do so. We construct a matrix CC^T based on an initial matrix A of distances between cities. The singular value decomposition of CC^T allows us to plot cities in the plan based on the given distances between cities. This report outlines our methodology, illustrates our results, and concludes with possible applications of this map reconstruction technique.

1 Introduction

In this technical report, we recreate the work of Richard Pulskamp to create a "map" of the United States using singular values. We are given a matrix of distances between major cities. We transform the matrix as we describe in Section 2. Using the singular values and eigenvectors of this transformed matrix, we are able to plot a representation of the country based on distance between cities alone.

To test the accuracy of our "map", we chose to add an "outlier" city: Anchorage, Alaska. The addition of this city allows us to examine how our mapmaking technique reacts to cities outside of the continental United States.

In Section 2, we discuss our methodology and discuss why this method creates an accurate map. Section 3 contains our resulting map and Section 4 reflects on this project and introduces possible applications for this map reconstruction technique.

2 Methodology

2.1 Introduction

For this project, we implement Richard Pulskamp's methodology. We start with a matrix of distances between cities (Appendix 1). The entries in this matrix are in hundreds of miles. While this choice of unit requires significant rounding, these measurements are sufficient for our exploratory purposes. We adopted Pulskamp's original 15×15 matrix and augmented it with the final row, the distances between Anchorage, Alaska, and the previous 15 cities.

Our goal is to reconstruct a map of the United States such that the positions of cities in the plane reflect the entries in our matrix. Unfortunately, this cannot be accomplished by starting with one city and building the map up based on distances, as we might hope. This is due to the rounding of the distances between our cities. As we add more cities to the plane, the distances between the points begins to vary away from our initial stated distances. As a result, adding more points causes the map to become more inaccurate.

Thus, Pulskamp introduces a methodology which allows us to find the coordinates of each city and plot them simultaneously using linear algebra. Instead of calculating the distance between each city one at a time and plotting them, Pulskamp suggests constructing a matrix of relative distances between every city. From this matrix, we can use the SVD decomposition to find the two dominant eigenvectors of our matrix. These eigenvectors will correspond to the x position and y position of our cities respectively, as they are the projection of each city's position into two dimensional space. This simplifies the process of map reconstruction significantly. Additionally, this methodology can be applied to R^k where $k > 2$, by using the first k eigenvectors. However, it often does not provide a meaningful mapping in dimensions larger than 2.

It has to be kept in mind that this method plots points in relation to each other because they are specified by relative distance between each other and nothing else. Consequently, the resulting map is not unique, as the points may be rotated or flipped in different directions. Thus, one potential hindrance is that the resulting map may be correct but not resemble the expected output map. However, this can be fixed by simple multiplying a rotation matrix with the resulting matrix of (x, y) city positions, making Pulskamp's method a highly effective method of map reconstruction.

2.2 Matrix Construction

Consider a matrix C which contains our desired x and y coordinates for each city in the first and second columns, respectively. This matrix is 16×2 in our example. Unfortunately, we cannot construct C based on distances between cities as these measurements give us no insight into our desired planar embedding. Thus we will use Pulskamp's method to create a matrix of relative distances and then use its SVD decomposition to solve for C.

In order to solve for the entries in C, we will first construct a matrix $CC^\top$. This matrix holds the relative distances between all of the cities in our initial matrix A. However, because we still do not have the matrix C, we must construct $CC^\top$ in a clever manner. As illustrated in Section 2.2.1, we must construct known matrices and work "backwards" to find the ideal coordinates for our map.

2.2.1 Creating $CC^\top$

Let there be matrices J and $\mathscr{D}$ where J is the matrix where every entry is $j = \frac{1}{n}$ and $\mathscr{D}$ where every entry is d_{ij}^2 where d_{ij} is the distance between city i and city j.

$$A = -\frac{1}{2}(I - J)\mathscr{D}(I - J) = -\frac{1}{2}(I - J)(\mathscr{D} - \mathscr{D}J) = -\frac{1}{2}(\mathscr{D} - J\mathscr{D} - \mathscr{D}J - J\mathscr{D}J)$$

Each entry in A, $a_{ij} = -\frac{1}{2}(d_{ij}^2 - c_j - r_i + t)$ where

1. $r_i = \frac{1}{n}\sum_{j=1}^{n} d_{ij}^2$
2. $c_j = \frac{1}{n}\sum_{i=1}^{n} d_{ij}^2$
3. $t = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2$

Note that (1) is the mean squared difference from city i to all other cities. Similarly, (2) is the mean squared difference from all cities to city j and (3) is the mean squared distance between all cities. These mean squared distances allow us to establish a measure of relative distance between individual cities.

This is true because every element in row i of $\mathscr{D}J$ is r_i, every element in column j of $J\mathscr{D}$ is c_j and each element in $J\mathscr{D}J$ is t. Now we have to show that element $a_{ij} = (x_i x_j + y_i y_j) = c_{ij}$. We have to expand d_{ij}^2, r_i, c_j and t:

1. $d_{ij}^2 = x_i^2 + y_i^2 + x_j^2 + y_j^2 - 2x_i x_j - 2y_i y_j$
2. $r_i = x_i^2 + \frac{1}{n}\sum_j x_j^2 + y_i^2 + \frac{1}{n}\sum_j y_j^2$
3. $c_j = x_j^2 + y_j^2 + \frac{1}{n}\sum_i x_i^2 + \frac{1}{n}\sum_i y_i^2$
4. $t = \frac{2}{n}\sum_i x_i^2 + \frac{2}{n}\sum_i y_i^2$

Therefore expanding the first two elements:

$$a_{ij} = -\frac{1}{2}(x_i^2 + y_i^2 + x_j^2 + y_j^2 - 2x_i x_j - 2y_i y_j - (x_j^2 + y_j^2 + \frac{1}{n}\sum_i x_i^2 + \frac{1}{n}\sum_i y_i^2) - r_i + t)$$

$$a_{ij} = -\frac{1}{2}(x_i^2 + y_i^2 - 2(x_i x_j + y_i y_j) - \frac{1}{n}\sum_i x_i^2 - \frac{1}{n}\sum_i y_i^2 - r_i + t)$$

Now expand r_i:

$$a_{ij} = -\frac{1}{2}(x_i^2 + y_i^2 - 2(x_i x_j + y_i y_j) - \frac{1}{n}\sum_i x_i^2 - \frac{1}{n}\sum_i y_i^2 - (x_i^2 + \frac{1}{n}\sum_j x_j^2 + y_i^2 + \frac{1}{n}\sum_j y_j^2) + t)$$

$$a_{ij} = -\frac{1}{2}(-2(x_i x_j + y_i y_j) - \frac{1}{n}\sum_i x_i^2 - \frac{1}{n}\sum_i y_i^2 - \frac{1}{n}\sum_j x_j^2 - \frac{1}{n}\sum_j y_j^2 + t)$$

Now finally expand t:

$$a_{ij} = -\frac{1}{2}(-2(x_i x_j + y_i y_j) - \frac{1}{n}\sum_i x_i^2 - \frac{1}{n}\sum_i y_i^2 - \frac{1}{n}\sum_j x_j^2 - \frac{1}{n}\sum_j y_j^2 + \frac{2}{n}\sum_i x_i^2 + \frac{2}{n}\sum_i y_i^2)$$

$$a_{ij} = -\frac{1}{2}(-2(x_i x_j + y_i y_j))$$
$$a_{ij} = x_i x_j + y_i y_j$$

Now notice that every element in $CC^\top$ is $c_{ij} = x_i x_j + y_i y_j$. Therefore matrix A is another way to express $CC^\top$. This method of creating $CC^\top$ is the most preferable because matrices I and J are easy to create. Matrix $\mathscr{D}$ is a bit harder as you have to pull data to create the entries. We will forgo illustrating the construction of $\mathscr{D}$ because it is conceptually straightforward.

2.3 Singular Value Decomposition

With our newly constructed matrix of relative distance $CC^\top$, we can use singular value decomposition to solve for our matrix of positions C. To do this, we first find the two dominant eigenvectors of $CC^\top$ by taking the first two columns of the U matrix in the SVD decomposition. These eigenvectors represent the two most principal directions of the change-of-basis transformation and allow us to plot our 16-dimensional vectors of distances in R^2. We scale these eigenvalues by their corresponding eigenvalues λ because λ represents the "stretch" caused by the change-of-basis transformation. In this application, it gives a sense of scale the map.

The two dominant eigenvalues of $CC^\top$ are below.

```
Vector 1 (x): Vector 2 (y):
-0.24009072  -0.323959814
-0.06821065  -0.157897472
-0.13033289   -0.041817697
0.09632898  0.094662254
-0.06095360 0.356084457
0.26199212  0.375148231
-0.29126246  0.193172007
-0.12940237  0.274442650
-0.22720588  -0.232150695
-0.21495514 -0.183389801
0.36397169   -0.004805121
0.32822893   0.221923284
0.36304606  -0.081298775
-0.25038013  0.200810397
-0.20463013 -0.176907493
0.40385619 -0.514016412
```

These will act as the x and y-coordinates of our mapping – in other words these are our approximations of the columns of C we sought out to find initially.

3 Results

Our results using the methodology in Section 2 are shown below (Figure 1).

Intuitively speaking, this plot is a decent representation of the United States. We worried that adding an outlier such as Anchorage may harm our resulting map. However, the scale aside, this technique has placed Anchorage approximately where we would expect it to. This speaks the the method's ability to convert simple distances into meaningful spatial relationships.

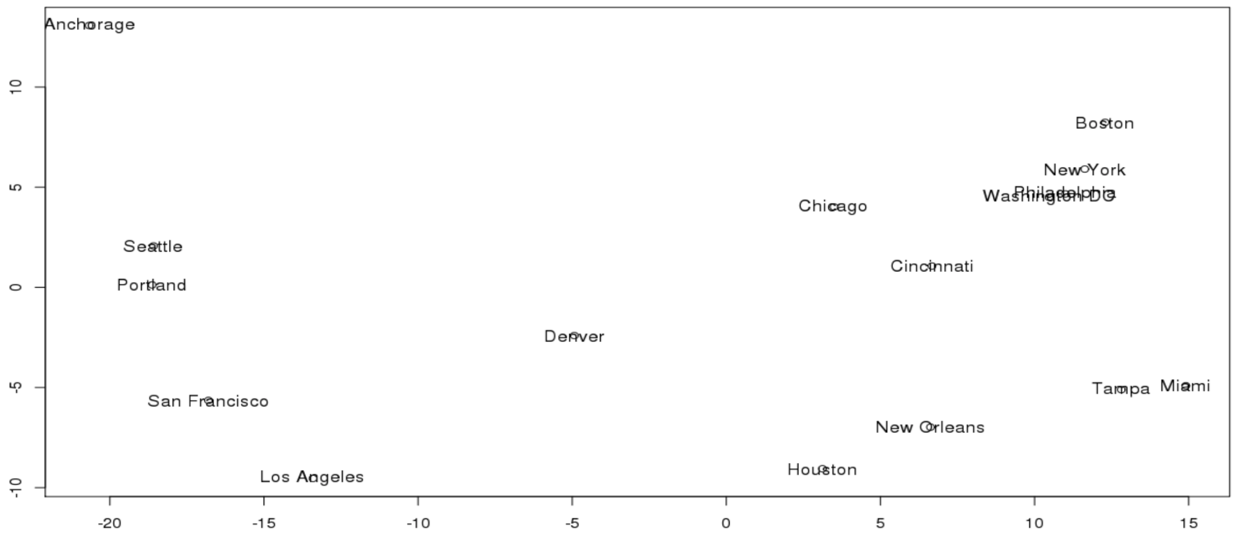

Figure 1: This figure depicts a labeled outline of the United States using the relative distances of the cities between each other.

One could argue for or against the accuracy of this map. Certainly we would expect Miami to be further "south" than Houston, or Anchorage and the west coast to be significantly further "north" relative to Boston. However, we can also see a decent outline of the country in this map. Some of the error in scale and relative distance could be due to the rounding which occurred in constructing our original matrix of distances.

As a first foray into map reconstruction, we consider this a successful project. There is certainly room for improvement and further research

4 Conclusion

Pulskamp's methodology introduces a viable option for map reconstruction given nothing but intercity distances. In this report, we emulated Pulskamp's work with the addition of an "outlier" city. We found that the two dominant eigenvalues of $CC^\top$ produce a relatively accurate set of coordinates for plotting our cities in R^2. While we did not discuss multidimensional applications in this report, it is worth noting that Pulskamp's methodology allows us to create 3-dimensional maps and provides a way to think about spatial relationships between objects in any R^k.

This technique could be used to reconstruct maps of historical societies. Suppose you have records of travel between long-gone cities but no map: this application of singular value decomposition could allow historians to better understand the geographic layout of ancient societies.

Pulskamp's methodology is one of many applications of eigenvectors and singular value decomposition. It allows us to create a map of objects (in this case, cities) based on their similarities (in this case, intercity distances). Similar logic and processes can be applied to many other sorts of objects (such as cereals grouped by nutritional information, for example). In this regard, Pulskamp's method and eigenvalues in general are incredibly useful and applicable to many real world problems.

5 Works Cited

1. R.J. Pulskamp, *Constructing a Map from a Table of Intercity Distances.* The College Mathematics Journal, Vol 19 No 2, pp. 154-163, March 1988.

6 Appendix

The following code is used to recreate the data that the research uses, with our additional data for Anchorage, Alaska:

```
n = 16
# lower triangle for symmetric matrix
c1 = c(0,10,9,20,20,31,15,16,2,3,32,32,32,14,4)
c2 = c(0,0,3,10,11,22,14,9,8,8,23,22,22,12,7)
c3 = c(0,0,0,12,11,23,11,8,6,6,25,24,25,10,5)
c4 = c(0,0,0,0,10,12,21,13,19,18,13,13,14,19,17)
c5 = c(0,0,0,0,0,16,13,4,17,16,24,20,24,10,15)
c6 = c(0,0,0,0,0,0,29,19,29,28,10,4,12,26,28)
c7 = c(0,0,0,0,0,0,0,9,13,12,34,32,35,3,11)
c8 = c(0,0,0,0,0,0,0,0,14,13,27,23,27,6,12)
c9 = c(0,0,0,0,0,0,0,0,0,1,31,31,30,12,2)
c10 = c(0,0,0,0,0,0,0,0,0,0,30,30,29,11,1)
c11 = c(0,0,0,0,0,0,0,0,0,0,0,6,2,32,29)
c12 = c(0,0,0,0,0,0,0,0,0,0,0,0,8,30,29)
c13 = c(0,0,0,0,0,0,0,0,0,0,0,0,0,33,29)
c14 = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,10)
c15 = rep(0,15) # these columns can be used to create the matrix for only 15 cities

c.anc = c(34, 25, 31, 24, 33, 24, 40, 34, 34, 34, 15, 20, 14, 38, 33)
labels = c("Boston", "Chicago", "Cincinnati", "Denver", "Houston", "Los Angeles", "Miami", "
    New Orleans", "New York", "Philadelphia", "Portland", "San Francisco", "Seattle", "Tampa
    ", "Washington DC", "Anchorage")
A = cbind(c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15)
A = rbind(A, c.anc)
A = cbind(A, rep(0,16))
# cloning lower matrix to upper matrix to create full symmetric matrix
for (i in 1:(n-1)){
  for (j in (i+1):(n)){
    A[i,j] = A[j,i]
  }
}
```

The following code converts the matrix according to the above methodology and renders a plot with labels:

```
require(Matrix)
D = Matrix(0,n,n, sparse = TRUE)
for (i in 1:n){
  for (j in 1:n){
    if (i == j){
      D[j,i]=0
    }

    D[j,i] = D[i,j] = (A[j,i])^2
  }
}
J = matrix(1/n, n, n)
CCt = -1/2*(diag(n)-J)%*%D%*%(diag(n)-J)

out = svd(CCt)
C = cbind(sqrt(out$d[1])*out$u[,1], sqrt(out$d[2])*out$u[,2])

plot(C[,1],C[,2])

C.new = -1*C #rotates the point for viewer to recognize the geographic perspective

```

```
plot(C.new[,1],C.new[,2])
text(C.new[,1],C.new[,2], labels, cex = 0.75)
```