Find the closest county

Web application to display counties to a certain county based on Eucledian distance

Riin Aljas†

Philip Merril College of Journalism, University of Maryland, College Park, Maryland, USA raljas@umd.edu

1. Introduction

Every journalist, no matter their data skills, comes across basic data every day. One of the most common data they work with is demographic data related to their subject. For example, education reporter in Baltimore has had to look up numbers about population, median income, education level, racial breakdown etc in Baltimore area. Demographic data helps the reporter to contextualize their findings and offer background about the topic they’re dealing with.

Furthermore, by comparing similar locations, demographic data helps to understand whether reporter’s findings are anomaly or a common phenomenon in places with certain characteristics. In Baltimore, the reporter would want to find a city or county with similar population and racial breakdown. Doing this manually is time consuming, as U.S has 3220 counties.

The goal of this project was to create an easy and ready to use web application, where a reporter with no data analysis skills could easily look up similar counties to their county of choosing. The prototype uses basic clustering algorithm with Euclidian distance and is built with R programming language using U.S. census data with set parameters of median income, median size and population size. As a concept, however, the model will work with any parameters and datasets.

U.S. census data has been imported through census API. The author contributed with writing a script for clustering counties with Euclidian distance and building a web application interface using R Shiny.

1. Literature review

Clusters are geographic concentrations of industries related by knowledge, skills, inputs, demand, and/or other linkages (Delgado, et al 2016). Clustering Census or any demographic data is a basic concept in statistical analysis, but overall, clustering is used widely in several fields. For example, in business.

According to (Lu et. al, 2018), the volume of research about clusters has increased dramatically over the last two decades or so: “Many paradigms have appeared, such as Porter's (1990) diamond model, Krugman's (1991) core‐periphery model, and the innovation argument of the Nordic scholars (e.g., Cooke, 1992; Asheim et al., 2011; Cooke, 2012).” The reason for that is mainly that there are economical and academical benefits to gain from studying clusters.

Clusters are created by considering indicators (such as population figures, median income etc.) and calculating similarity between different groups by using similarity measures. Distance (dissimilarity) and similarity are the basis for clustering analysis. Similarity measure helps to understand what the ‘distance’ between two groups is. The main measures used in quantitative analysis are: Minkowski distance, Euclidean distance, Cosine distance, Pearson correlation distance and Mahalanobis distance (Xu, 2015).

Due to the simplicity of the end goal – to create a similarity list of single counties based on pre-chosen parameters -, we use ‘eucledian distance’.

In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space. Euclidean distance allows to measure two counties based on more than one parameter which are independent and incomparable (such as population vs income). In recent years, progress has been made by offering several sites and solutions for comparing US counties.

The closest to the end goal of this project is ‘Clustermapping U.S’ project by Harvard Business School and U.S. Economic Development Administration. There, researchers from Harvard Business School, MIT Sloan School of Management, and Temple University's Fox School of Business generated cluster definitions based on a novel algorithm that allows for the systematic generation and comparison of clusters across the United States.

They offer several regional clusters with the possibility to compare two counties. Yet creating a concrete list of chosen parameters based on one county, is not yet an option. Also, a registered user is required, making the process more complicated.

This project, however, aims to fill the gap, by creating a simple algorithm for creating county lists. The code itself will work for any dataset, but the interactive solution is currently for U.S. census data only.

1. Methodology

This application was entirely developed using R programming language. Data used in the prototype is deriving from American Community Survey’s (ACS) 2017 five-year estimates. The data could be downloaded manually or written into the script by using the Census bureau’s API and R ‘tidycensus’ package. The latter approach was used in this project. All data is broken down by U.S. counties.

The first version of application is using three parameters: median income, population size, and median income. The script can be adapted by adding additional U.S. ACS variables.

Further testing is necessary to determine the optimal number of parameters against dataset size, so that the usability wouldn’t decrease due to the time it takes to run the algorithm.

For calculating Euclidean distance, R package ‘philentropy’ was used. No data cleaning besides changing column names for joining was necessary, as the U.S. census data had no missing values and dataset itself is reputable and trustworthy. Further versions would develop a basic cleaning script for custom dataset.

1. Solution Framework

First part of the script deals with reading in and joining the census data with necessary parameters. As the first version of the solution deals with U.S. data from 2018 only, we can save the dataset so we can save time by not loading it every time or calculating Euclidean distance, R package ‘philentropy’ was used.

ALGORITHM 1: Getting the data

countyincome <- get\_acs(geography = "county", variables = c(medincome="B19013\_001")) %>%

select(-variable, -moe)

population <- get\_acs(geography = "county", variables = c(pop= "B01003\_001E")) %>%

select(-variable, -moe)

age <- get\_acs(geography = "county", variables = c(age = "B01002\_001E")) %>%

select(-variable, -moe)

#rename so that we wouldn't have same variables

 population <- population %>%

rename(pop = estimate)

age <- age %>%

rename(age = estimate)

data <- left\_join(countyincome, population) %>%

left\_join(age)

#be sure there's no NA-s

data <- na.omit(data)

# create a second data variable to remove, so that the code would only deal with numeric variables

data2 <- data %>%

column\_to\_rownames(var="NAME") %>%

select(-GEOID)

# calculate eucledian distance

 eucdata <- as.data.frame(distance(data2, method = "euclidean")) %>%

rownames\_to\_column()

data <- bind\_cols(data, eucdata) %>%

column\_to\_rownames(var = "rowname")

Secondly, we calculate the Euclidean distance which will become our metric for comparing the counties.

By testing the solution with three different machines with different processing power, the average time for loading the results for Euclidean distance was about 30-60 seconds.

 As the first version of the application is based on static dataset, we don’t have to write the Euclidian distance calculation into the interactive application script and can save Euclidean values as a pre-existing dataset within the project.

 To get the user input, we use RStudio’s interactive environment Shiny. Shiny is an R package that makes it easy to build interactive web apps straight from R.

 First, user input is asked by letting them choose the county. Based on chosen county, the application will update itself by displaying counties most like the county chosen.

 Every time the user changes the county, updated version will display.

 ALGORITHM 2: Interactive solution for user input

data <- readRDS(“sampledata.RDS”)

vchoices <- 1:3220

names(vchoices) <- mydata$NAME

mylist <- as.data.frame(mydata$NAME)

ui = fluidPage(

    sidebarLayout(

        sidebarPanel(

            helpText("Order US counties based on the closest county to it

                     based on eucledian distance."),

            checkboxGroupInput("parameters", "Select parameters", choices = c("Population",

                                                                       "Median age",

                                                                       "Median income")

                        ),

selectInput("columns","Select Counties",choices=vchoices)),

        mainPanel(

dataTableOutput('mytable')

)))

server = function(input, output) {

observeEvent(

      input$columns, {

        cols <- as.numeric(input$columns)

        if(c(length(input$columns) == 1, input$parameters == "Median income")) {

           mydata <- readRDS("sampledataincome.RDS")

            df <- data.frame(mydata[,cols])

            df <- bind\_cols(df, mylist)  %>%

                           rename(County = `mydata$NAME`,

                       euc\_dist = 1)

            output$mytable = renderDataTable(df)

        }else{

            output$mytable = renderDataTable(mydata[,cols])

        }

    })

}

shinyApp(ui = ui, server = server)

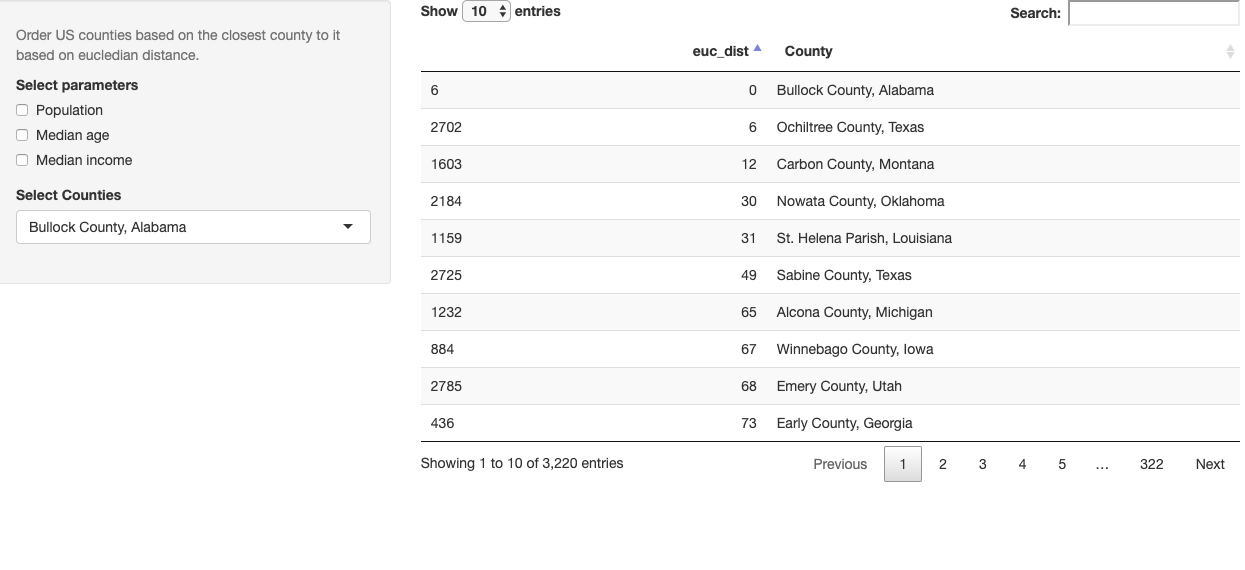


Figure 1: Screenshort of the Shiny application (December 2019)

1. Limitations and future work

There are several limitations to the existing solution. Parameters aren’t scaled which might create a biases towards certain parameters. Interface needs fixes, so that user input would work perfectly.

Additionally, the next version would include the following additions:

* Basic data cleaning algorithm
* Code needs to be developed further so that parameters for Euclidian distance would be chosen by function
* Tests to understand the optimal data size and parameters
* Add visualizations

REFERENCES

1. Porter, M. 2003. The Economic Performance of Regions. Regional Studies, Vol. 37.6&7, pp. 549–578*,* [*h*ttps://clustermapping.us/sites/default/files/files/resource/The\_Economic\_Performance\_of\_Regions.pdf](https://clustermapping.us/sites/default/files/files/resource/The_Economic_Performance_of_Regions.pdf)
2. Delgado M., Porter, M., Stern, S. 2014. Defining Clusters of Related Industries. NBER Working paper series, 20375. [*https://www.clustermapping.us/sites/default/files/files/resource/Defining%20Clusters%20of%20Related%20Industries%20-%20NBER%20Working%20Paper%2020375.pdf*](https://www.clustermapping.us/sites/default/files/files/resource/Defining%20Clusters%20of%20Related%20Industries%20-%20NBER%20Working%20Paper%2020375.pdf)
3. Dongkuan, Xu, Yingjie Tian 2018. Comprehensive Survey of Clustering Algorithms. *Ann. Data. Sci. 2(2):165–193 https://link.springer.com/article/10.1007/s40745-015-0040-1*
4. Lu, R., Reve, T., Huang, J., Jian, Z. and Chen, M. (2018), A Literature Review of Cluster Theory: Are Relations Among Clusters Important? Journal of Economic Surveys, 32: 1201-1220. doi:[10.1111/joes.12255](https://doi.org/10.1111/joes.12255)
5. <https://clustermapping.us/>
6. <https://clustermapping.us/>