**Predicting Teacher Cross County Commute Times**

Addison Armstrong

William Gribbin

Gillian Garbus

Kyle Hollander

Theresta Desir

Tyler Allen

**I. Introduction**

Currently the length of the commute times between counties for teachers if they do not live in the same county they work in is not accounted for when determining funds for schools. This is an obstacle when attempting to distribute funds designated for hiring teachers that are determined by location in a manner that won’t cause teachers in low cost of living areas to commute further for a higher salary, while not wasting funds when there is no feasible way to commute to a higher paying school. We will be finding a solution to this problem in order to answer the greater question of how to determine the relationship between commute time and salary to determine how far people are willing to travel for additional pay.

**II. Literature Review**

Cross county commuting flows are nothing new to the econometrics world. There have been a number of research papers and studies on the causes and effects of wage difference and commuters within the United States. One study aims to provide additional information about county level commuting flows, which only makes up one component of the metropolitan and micropolitan statistical areas delineation process. During the 2006-2010 period, more than a quarter (27.4 percent) of U.S. workers traveled outside of their residence country for work during a typical week, compared to 26.7 percent in 2000. Transportation infrastructure such as highways and transit routes shape the spatial dimension of travel and commercial development, concentrating commuting flows within a relatively small number of origin and destination communities (Mckenzie, 2013).

In order to gain a better understanding of the subject at hand, we researched both how commute times affect salaries in general, and the factors that go into determining a teacher’s salary. Redmond and Mokhtarian (2001) learned that just over half of people believe they are commuting too far. They surveyed a number of people analyzed the gathered data using orbit and probit. They found that their ideal commute time was related positively to the commute time they are currently enduring. Additionally, just over half of people felt as though they were commuting too far.

We additionally found that teachers salaries are the highest expense for primary schools (Kirby, P., Holmes, C., Matthews, K., & Watt, A. 1993) as well as the fact that school salaries are incredibly sporadic, even within the same geographic location for the same amount of work (Wagner, G. A. 1999).

Understanding and using this data can be a challenge, and one that others have had issues with in the past. (Green, Kutzback, & Vilhuber 2017) compares ACS and LEHD source files, using identifying information and probabilistic matchmaking to join person and job records. Green et al. compared commuting statistics for job frames linked on person, employment status, employers and workplace and they identified person and job characteristics as well as design features of the data frames that explain aggregate differences. Their results suggest two broad issues that contribute to the difference in commute distance. The first one being a disagreement in workplace locations as reported in survey and administrative data. The second being the missing data problem associated with linking workers to their unique work locations in the administrative data.

Lastly, the 2013 report on commuting in america (AASHTO 2013) shows that the vast majority of commuters - 86% - commute within their metropolitan area. This means that only a small percentage of teachers are likely to commute across county lines to begin, which would make gathering consistent data difficult. Additionally, the article verifies the assertion that people who live in non metropolitan areas - or suburbs - have the longest commute time. The implications of this report coincide with our findings and serve as further support for our analysis.

**III. Data**

We retrieved our data from the USA IPUMS Census data website. The data sample used comes from the year 2015 through 2000, American Community Survey (ACS), which holds data from 2015 and all years through 2000. Both household and personal level data were included in the sample set for this analysis. We retrieved a number of variables that we believed to be pertinent to our research topic. We also created new variables using that data, for the sake of easing analysis.

**Year** the year

**Datanum** the data set number

**Serial** household serial number

**Hhwt** household weight

**Statefip** State FIPS code

**Pumaarea** land area of puma

**PUMA** Public use microdata area

**Perwt**  Person weight

**Sex** Person Gender

**Age** Person age

**Occ** Person Occupation

**Inctot** Total personal income from salary (includes stocks etc.)

**PWPUMA00** Place of work by puma after 2000

**Trantime** transportation time

**AvgTranTime** Average transit time between home and work

**Occ2** Occupation codes for educators (2200-2550)

**Educ** Education attainment

**Educd** Education attainment (detailed)

**Route** Generated variable label of grouping puma to pwpuma00

**Pop** Total population density by puma

**Area** pumaarea / 1000000

**Avgtrantime** average transit time between puma and pwpuma00

**Popden** pop / area

**Startpop** pop \* puma

**Endpop** pop \* pwpuma00

**State** grouping of (statefip)

A few of these variables (datanum, serial, and hhwt) were only pulled in order to gather other information, and are simply the parents of the data we needed. AvgTranTime is a variable we created during analysis, using trantime. Occ2 was another generated variable using the original occ codes provided in the extract; which groups educators into a single occ code of “100000”. The rest of the codes for the Occupations that were used are listed below.

OCC CODES | Occupation

21 Farmers and Ranchers

31 Food Service Managers

34 Lodging Managers

94 Tax Preparers

352 Opticians, Dispensing

384 Miscellaneous Law Enforcement Workers

430 First-Line Supervisors/Managers of Gaming Workers

432 First-Line Supervisors/Managers of Personal Service Workers

446 Funeral Service Workers

470 First-Line Supervisors/Managers of Retail Sales Workers

522 Court, Municipal, and License Clerks

534 New Accounts Clerks

552 Dispatchers

553 Meter readers, Utilities

555 Postal Service Mail Carriers

703 Avionics Technicians

862 Water and Liquid Waste Treatment Plant and System Operators

891 Etchers and Engravers

981 First-Line Enlisted Military Supervisors/Managers

983 Military, Rank Not Specified (Census only)

OCC2 CODES | Occupation

220 Postsecondary Teachers

230 Preschool and Kindergarten Teachers

231 Elementary and Middle School Teachers

232 Secondary School Teachers

233 Special Education Teachers

234 Other Teachers and Instructors

254 Teacher Assistants

243 Librarians

244 Library Technicians

255 Library Technicians

**IV. Analytical Method**

To better understand how our variables affected the distance that people will drive we incorporated each variable into the analysis. For this we decided to use an OLS Regression Model. This model was chosen since it allows us not only to look at variables in the easiest way but it also gave us a measure in which to see which of the models that we tested is the best. The model that we decided to work on and refine is as follows:

**CommuteTime = βX + ϒZ + ϴW**

Where β are the characteristics of the person, ϒ is the destination, and ϴ is the origin. This model was used to take a better look at our variables and after assigning then to their proper places the model was able to give us some answers to our main question. For the β part of the model the variables occupation, sex, age and income were assigned to it. For ϒ we assigned PWPUMA00, and for ϴW we assigned PUMA. When looking at the different regression models we noticed that some of the variables that were listed above were not necessary or could be done a better way. Along with using the PUMA and the PWPUMA, separate variables were made to indicate drive times between them as well as just a variable of all of the possible links between them. We also chose to take a look at the first part of our model and see what variables where statistically significant. We were able to see that all of them where except for the sex or gender variable so it was removed from our testing.

With this in mind we could find what is making people drive from one county or in this case PUMA to another. For this we needed to create a variable based on the data that we have. We needed to create a variable that holds the mean travel time for people driving from one PUMA to another and any combination of it that has happened in our data. This variable was specifically created in STATA since it gave us the best way to play with the variables. To do this, we wrote a statement that sorts all of the data into the combinations of different travel destinations and departures for work and averaged them together. The exact code that created this is below.

**bysort PUMA PWPUMA00 : egen averagetrantime=mean(trantime)**

Using the generated variables (above & specified in previous section), we were able to sort and rank the occupations by average income, average transit time, and education level. Once the data was sorted and ranked accordingly, the remaining nine occupation codes that most resembled educators were kept for further analysis. This group of variables then became what we called our “like teachers” occupations, or occupations that had similar levels of education, income and travel time to their place of employment. This is an important thing to have because not only are we trying to see what is making teachers drive further and further for work, but we can also possibly determine why any employee would drive farther for work allowing our findings to be used for more than just the education field.

With all of the main variables that we needed created and modified, we needed to take a look at the summary statistics (below). These statistics are the basics of the information. For example, they are going to tell us the number of observations, which number appears most and the average number of our variables. All of these are very important when it comes to understanding the data but not really when it comes to talking about the data.

Variable | Obs Mean Std. Dev. Min Max

-------------+--------------------------------------------------------

sex | 299608 1.619046 .4856222 1 2

age | 299608 41.34268 12.12265 16 93

puma | 299608 1749.747 1560.956 100 8106

pwpuma00 | 299608 1763.366 1558.357 1 8200

parea | 299608 9892.28 58558.06 3.458766 1329354

-------------+--------------------------------------------------------

pwparea | 299608 9892.522 58558.22 3.458766 1329354

pop | 299608 81927.62 75532.92 1693 228933

popdenpuma | 299608 96.14303 292.0293 .1555334 9140.289

popdenpwpuma | 299608 96.14245 292.0293 .1555334 9140.289

**V. Results**

After using our OLS Regression model, discussed in the previous section, as a guide; we began the creation of our regressions to best predict the reason for travel and how far they are willing to travel. At first, we began trying to figure out if there is any connection to a certain PUMA and a PWPUMA and if there is any main reason for someone to want to travel between them. This idea of seeing the connections is completed by using the following code inside our program of choice, STATA.

**Egen route = group (puma, pwpuma00), label**

What this code does is, it groups every known connection of the pumas to the pwpumas so that we have a basic understanding of the patterns to look at. However, before we could even do this there was a lot of data manipulation and cleaning that had to be done. This mainly was trying to find what occupations have a similar amount of education, travel time and income. This was done inside STATA by first doing a collapse command followed by a simple rank command. This then allowed us to see exactly which occupations have the similar traits as people in the teaching field so that we can go in and remove everyone else that does not. The code itself for this is available inside Appendix B. Another thing that was done was taking the natural log of our numerical data so that we get our results in terms of percentages not actual numbers, which is just preference to see the data in a certain way for analysis.

Now that we have a link variable for the groupings of pumas to pwpumas we can begin to think about what variables are going to be needed for us to create a regression model. Looking at our basic model above we can see that we need characteristics of the person, their home, and their place of work. For the individual level we believe that using the sex of the individuals, their age, their occupation, and their education level is a good starting place. For the puma or their home location we have the population density. We also have the same information for the pwpumas. The regression below has the attributes mentioned for the individual level characteristics.

**reg trantime sex age occ educd popdenpuma popdenpwpuma [pw=perwt], cluster(route)**

**(sum of wgt is 5.9342e+06)**

**Linear regression Number of obs = 299536**

**F( 6, 7068) = 40.47**

**Prob > F = 0.0000**

**R-squared = 0.0066**

**Root MSE = 20.399**

**(Std. Err. adjusted for 7069 clusters in route)**

**------------------------------------------------------------------------------**

**| Robust**

**trantime | Coef. Std. Err. t P>|t| [95% Conf. Interval]**

**-------------+----------------------------------------------------------------**

**sex | -2.368856 .1905584 -12.43 0.000 -2.742408 -1.995305**

**age | .0044738 .0049055 0.91 0.362 -.0051423 .01409**

**occ | .0034881 .0004446 7.85 0.000 .0026166 .0043596**

**educd | .0172579 .0029647 5.82 0.000 .0114462 .0230695**

**popdenpuma | -.10863 .065767 -1.65 0.099 -.237553 .020293**

**popdenpwpuma | .1116001 .0657904 1.70 0.090 -.0173687 .2405689**

**\_cons | 21.6547 .7560242 28.64 0.000 20.17267 23.13674**

**------------------------------------------------------------------------------**

As you can see, the regression above has what is called a cluster. This is where our route variable comes into play. It is allowing us to not only look at each variable and how it interacts with transit time, but also cluster that value based upon their start and end point. Looking at the regression itself and the p values and coefficients we believe that we can make some educated assumptions. The first assumption is which variables do not matter when trying to determine the commute times. We can, in a sense, throw out the gender or sex variable. The reason that this makes sense logically is that we have already collapsed the data down into a form that is only occupations like teachers in the united states. So we already took out most of the people that would not want to be a teacher or work in a position like a teacher. This usually will remove all biases towards the job, one of which usually being gender.

The reason the gender variable is the only one that we are throwing away when even those too have above the normal significant value of .05, is because of what the variables mean by themselves. The popdensity variable for the puma and pwpuma level is a variable that repeats values multiple times for every instance of the puma or pwpuma. Knowing this about the popdensity variable, makes sense that the regression model does not believe that it is significant when it actually is. This is why only the education variable has been dropped for our regression listed below.

**reg trantime age occ educd popdenpuma popdenpwpuma [pw=perwt], cluster(route)**

**(sum of wgt is 5.9342e+06)**

**Linear regression Number of obs = 299536**

**F( 6, 7068) = 40.47**

**Prob > F = 0.0000**

**R-squared = 0.0066**

**Root MSE = 20.399**

**(Std. Err. adjusted for 7069 clusters in route)**

**------------------------------------------------------------------------------**

**| Robust**

**trantime | Coef. Std. Err. t P>|t| [95% Conf. Interval]**

**-------------+----------------------------------------------------------------**

**sex | -2.368856 .1905584 -12.43 0.000 -2.742408 -1.995305**

**age | .0044738 .0049055 0.91 0.362 -.0051423 .01409**

**occ | .0034881 .0004446 7.85 0.000 .0026166 .0043596**

**educd | .0172579 .0029647 5.82 0.000 .0114462 .0230695**

**popdenpuma | -.10863 .065767 -1.65 0.099 -.237553 .020293**

**popdenpwpuma | .1116001 .0657904 1.70 0.090 -.0173687 .2405689**

**\_cons | 21.6547 .7560242 28.64 0.000 20.17267 23.13674**

**------------------------------------------------------------------------------**

The only other thing that we believe might be worth looking at is the average transit times of every occupation that we created before. Once this average transit time for occupation or avgtrantime variable is placed inside the regression we receive the lowest root mean squared error out of all of the other regression. What this means is that this model has the most certainty out of all of our models to be accurate by the smallest margin of error. We can say that this model is going to have the smallest margin of error due to the r squared value that is shown in the regression. Not only does this model give us a lower root mean squared error and a higher r squared value but it is also a model that is giving us other variables that do not seem to be significant. From our newest model we are going to be able to remove the age variable as well, since it has a p value of .321 in this new regression. The modified regression model is located on the next page.

**reg trantime occ lnincwage educd lnpopdenpuma lnpopdenpwpuma lnavgtrantime [pw=perwt], cluster(route)**

**(sum of wgt is 5.9342e+06)**

**Linear regression Number of obs = 299536**

**F( 5, 7068) = 625.69**

**Prob > F = 0.0000**

**R-squared = 0.6648**

**Root MSE = 11.85**

**(Std. Err. adjusted for 7069 clusters in route)**

**--------------------------------------------------------------------------------**

**| Robust**

**trantime | Coef. Std. Err. t P>|t| [95% Conf. Interval]**

**---------------+----------------------------------------------------------------**

**occ | -.0006804 .0002229 -3.05 0.002 -.0011173 -.0002435**

**lnincwage | -.0749545 .0382231 -1.96 0.050 -.1498832 -.0000258**

**educd | -.0192125 .0013686 -14.04 0.000 -.0218955 -.0165296**

**lnpopdenpuma | -4.228752 3.415709 -1.24 0.216 -10.92457 2.467061**

**lnpopdenpwpuma | 3.931961 3.405997 1.15 0.248 -2.744814 10.60873**

**lnavgtrantime | 27.33043 .260216 105.03 0.000 26.82033 27.84053**

**\_cons | -29.4452 1.108951 -26.55 0.000 -31.61907 -27.27132**

**--------------------------------------------------------------------------------**

Analyzing the regression above we can see some interesting things. We expected most of them, but one was a surprise. To start lets go over what was expected. The occ code of the individual is significant but only very very slightly affects the transit time along with the education variable. This is due to the collapse that we did removing all of the non like teachers occ codes based on income education and transit time. This is also apparent in the incwage variable which has been added in for this regression. Now the lessapprents things that were found from this regression is that the average trantime variable seems to be affecting the transit time massively in a positive way. This means that occupation is one of the most important factors in an individual's commute time. The final thing that we are pulling from this regression model is that of the coefficients of the population density at the individuals starting and ending location. We can see that both are mostly significant, and that both of them affect the data at about the same rate. This rate however is inverse to one and other. This means that the starting location can lower commute time, whereas the occupation increases it.

**VI. Cross-County Commute Times Calculation Program**

During this time we also created a program that would take all counties in Florida, segment them into three sections (grade level of Elementary, Middle, and High School), and get the closest school between every possible county. This would allow us to compare the drive times from real data and the drive times supplied to us by IPUMS/PUMA data.

**i. Data Cleaning**

We obtained a large data sheet containing every school in Florida and including information on what kind of school it was (high, middle, or elementary) and other data like its physical location and what county it is within from the state of florida website. Next we brought the Data into a small instance of Microsoft SQL to edit and cull the data of things we deemed not usable. First, we removed all schools that were private academies, charter schools, or lab schools on university campuses and all but seven columns: District, SchoolNum, SchoolName, Street, City, State, Zip, and Type. We divided the data into three categories depending on which grades the school included and performed a distinct select on the school districts in each school grade. This generated another three datasets each holding one instance of a unique county/district name, enabling us to access the schools’ information by county.

**ii. Program Construction**

The program is broken up into three distinct sections: data importing, latitude/longitude distance comparison, and Google Maps Distance Matrix Analysis. This program uses the Google Maps Distance Matrix Api to return the time it takes to drive from one location to another based on two geocoded locations (latitude and longitude.) The first section of the program can be seen below in Appendix Section C, it will be labeled “*def CSVGenerator*” and this handles the first section and part of the second section of this program. It takes a input supplied upon startup which contains the first county and second county being compared, a variable labeled timer that assists with writing CSV without overwriting prewritten data, and the name of the file to be exported as a csv. The program will then go through the provided csv and pull all of the counties supplied in the first county variable and load them into a temporary csv, then it will take the second county supplied and collect all the schools from that county and dump their data into a second temporary file. Then it will load these temporary files into their own lists and begin the analysis of each school. First it will take the first school in List A then compare it with every school in List B then move to the next school in List A and repeat this process until it has analyzed every possible combination of schools in the lists. When completed it will generate a new list and call the next major function of the program which is the distance calculator. The distance calculator can be seen below in Index C under “*def disCalc*”. This is a recallable function that takes two geocodes and gets the distance between each other. Then it returns this distance to be added to the third list we mentioned above. Finally the “*def* *CSVGenerator*” function will take the completed list containing all the schools that have been compared and their distances and reorganize them into lowest distance to highest distance. This will be fed into the Google Maps Distance Matrix API. It will then take the lowest data set in the list which will contain the shortest distance and sends it to the Google Maps API. It does this by declaring our third important function which is seen below as well and labeled “*def gmapcalc*”, this function takes in three variables (first location geocode, second location geocode). This takes both of the geocodes of the schools and submits them to the google api. Then it formats the returning *JavaQuery (jquery)* that the Google API returns. This *Jquery* contains a lot of information that needs to be culled and removed for optimal time efficiency. This is all handled by the *gmapcalc* function itself and returns the drive time to be brought into the data. Then the *def CSVGenerator* writes the single school pair to the csv file chosen in the beginning.

**iii. Increasing Efficiency**

To further facilitate automation of this process we created four helper functions to parse through all the counties and make sure it is handled automatically. The first three are for each grade level (elementary, middle, and high school) with the final one being a function that simply calls the previous three functions mentioned, handles errors caused by the program, and finally provide output that the user can see to notify what is running and when a dataset is completed. Now, the three functions based on each grade level perform the same action just on a different data set. They are labeled *def Elementary, def Middle,*  and *def High*. They take their corresponding driverdata file (the file containing all unique counties) and generates two identical lists. It steps through the first list one item at a time, when starting a item then going through every item in the second list similar to the comparison program outlined above. This will send every possible county combination to the CSVGenerator creating a full csv file containing the shortest distance between two schools in each county. After running all three functions, csv’s containing every cross county drive times for the closest possibles schools will have been created, giving us the data for comparison with the IPUMS/PUMA data.

**VII. Conclusion**

According to our analysis, most individuals who commute between Pumas do so for a combination of what occupation they are in, their starting and ending location, and how much they earn per year with their company. People are commuting into population dense areas for better pay while living within a reasonable commute in a lower cost of living area; meaning, individuals commuting long distance tend to live in suburbs and commute to the cities. There does not seem to be one standalone factor that affects their commute time day to day, but a combination of all of them. This is seen with the different regression models being executed with different variables and ultimately receiving different levels of significance for each.

**VIII. Appendix A (Do File)**

set more off

set matsize 10000

clear

cd "C:\Users\kholl\Desktop\DR"

log using "log.log",replace

import delimited "finaldata.csv"

//data manipulation

drop if occ == 0

drop if trantime == 0

drop if incwage == 0

drop if pwpumaland == 0

drop if pumaarea == 0

drop if missing(trantime)

drop if missing(incwage)

drop if missing(occ)

drop if missing(puma)

drop if missing(pwpuma00)

drop if missing(pumaarea)

drop if missing(pwpumaland)

gen lnincwage = ln(incwage)

gen lntrantime = ln(trantime)

save "DRresearchDo.dta", replace

/\*

//a collapse to find positions like teachers

//teacher occ codes 4 31 57 93 10 11 12 1 14 15 16 17 18 19 23 24 25 26

gen occ2 = 0

replace occ2 = 100000 if occ == 220 | occ == 230 | occ == 231 | occ == 232 | occ == 233 | occ == 234 | occ == 254 | occ== 243 | occ == 244| occ == 255

replace occ2 = occ if occ2 == 0

//collapse to find occ like teach

collapse edu inctot trantime [pw=perwt], by(occ2)

egen incrank = rank(inctot)

egen edurank = rank(edu)

egen tranrank = rank(trantime)

summ incrank if occ2 == 100000

gen teachinc = 1 if abs(incrank - r(mean)) < 100

summ tranrank if occ2 == 100000

gen teachtran = 1 if abs(tranrank - r(mean)) < 100

summ edurank if occ2 == 100000

gen teachedu = 1 if abs(edurank - r(mean)) < 100

drop if missing(teachedu)

drop if missing(teachtran)

drop if missing(teachinc)

save "DRCollapse.dta",replace

//going back to before the collapse to do things.

clear

use "DRresearchDo.dta"

\*/

// We gotta drop everything but these + the eductaors Occ codes

egen like = anymatch(occ), value(21 31 34 94 352 384 430 432 446 470 522 534 552 553 555 703 862 891 981 983 220 230 231 232 233 234 254 243 244 255)

keep if like

//create a puma to pwpuma link variable for clustering

egen route = group(puma pwpuma00), label

save "DRresearchDo.dta", replace

//create population density

egen pop = total(perwt), by (puma)

gen parea = pumaarea / 1000000 // turn sq meters into sq kilometers

gen pwparea = pwpumaland / 1000000

gen popdenpuma = pop/parea

gen popdenpwpuma = pop/pwparea

gen lnpopdenpwpuma = ln(popdenpwpuma)

gen lnpopdenpuma = ln(popdenpuma)

// creating avg tran time (locations)

bysort puma pwpuma00 : egen avgtrantime = mean(trantime)

gen lnavgtrantime = ln(avgtrantime)

//get a general idea of trantimes through graphing bc its fun

graph twoway bar lnavgtrantime occ

//used regression models

reg trantime sex age occ educd popdenpuma popdenpwpuma [pw=perwt], cluster(route)

reg trantime age occ educd popdenpuma popdenpwpuma [pw=perwt], cluster(route)

reg trantime occ lnincwage educd lnpopdenpuma lnpopdenpwpuma lnavgtrantime [pw=perwt], cluster(route)

//summ stats

summ sex age puma pwpuma00 parea pwarea pop popdenpuma popdenpwpuma

log close

**IX. Appendix B (Log File)**

**--------------------------------------------------------------------------------------**

**name: <unnamed>**

**log: C:\Users\kholl\Desktop\DR\log.log**

**log type: text**

**opened on: 28 Apr 2018, 20:07:31**

**. import delimited "finaldata.csv"**

**(18 vars, 8685922 obs)**

**.**

**. //data manipulation**

**. drop if occ == 0**

**(0 observations deleted)**

**. drop if trantime == 0**

**(0 observations deleted)**

**. drop if incwage == 0**

**(0 observations deleted)**

**. drop if pwpumaland == 0**

**(5670534 observations deleted)**

**. drop if pumaarea == 0**

**(0 observations deleted)**

**. drop if missing(trantime)**

**(0 observations deleted)**

**. drop if missing(incwage)**

**(0 observations deleted)**

**. drop if missing(occ)**

**(0 observations deleted)**

**. drop if missing(puma)**

**(0 observations deleted)**

**. drop if missing(pwpuma00)**

**(0 observations deleted)**

**. drop if missing(pumaarea)**

**(0 observations deleted)**

**. drop if missing(pwpumaland)**

**(0 observations deleted)**

**. gen lnincwage = ln(incwage)**

**. gen lntrantime = ln(trantime)**

**. save "DRresearchDo.dta", replace**

**file DRresearchDo.dta saved**

**. /\***

**> //a collapse to find positions like teachers**

**> //teacher occ codes 4 31 57 93 10 11 12 1 14 15 16 17 18 19 23 24 25 26**

**> gen occ2 = 0**

**> replace occ2 = 100000 if occ == 220 | occ == 230 | occ == 231 | occ == 232 | occ == 233 | occ == 234 | occ == 254 | occ== 243 | occ == 244| occ == 255**

**> replace occ2 = occ if occ2 == 0**

**>**

**> //collapse to find occ like teach**

**> collapse edu inctot trantime [pw=perwt], by(occ2)**

**>**

**> egen incrank = rank(inctot)**

**> egen edurank = rank(edu)**

**> egen tranrank = rank(trantime)**

**>**

**> summ incrank if occ2 == 100000**

**> gen teachinc = 1 if abs(incrank - r(mean)) < 100**

**> summ tranrank if occ2 == 100000**

**> gen teachtran = 1 if abs(tranrank - r(mean)) < 100**

**> summ edurank if occ2 == 100000**

**> gen teachedu = 1 if abs(edurank - r(mean)) < 100**

**>**

**> drop if missing(teachedu)**

**> drop if missing(teachtran)**

**> drop if missing(teachinc)**

**>**

**> save "DRCollapse.dta",replace**

**>**

**> //going back to before the collapse to do things.**

**> clear**

**> use "DRresearchDo.dta"**

**> \*/**

**. // We gotta drop everything but these + the eductaors Occ codes**

**. egen like = anymatch(occ), value(21 31 34 94 352 384 430 432 446 470 522 534 552 553 555 703 862 891 981 983 220 230 231 232 233 234 254 243 244 255)**

**. keep if like**

**(2715780 observations deleted)**

**.**

**. //create a puma to pwpuma link variable for clustering**

**. egen route = group(puma pwpuma00), label**

**.**

**. save "DRresearchDo.dta", replace**

**file DRresearchDo.dta saved**

**.**

**. //create population density**

**. egen pop = total(perwt), by (puma)**

**. gen parea = pumaarea / 1000000 // turn sq meters into sq kilometers**

**. gen pwparea = pwpumaland / 1000000**

**. gen popdenpuma = pop/parea**

**. gen popdenpwpuma = pop/pwparea**

**. gen lnpopdenpwpuma = ln(popdenpwpuma)**

**. gen lnpopdenpuma = ln(popdenpuma)**

**.**

**. // creating avg tran time (locations)**

**. bysort puma pwpuma00 : egen avgtrantime = mean(trantime)**

**. gen lnavgtrantime = ln(avgtrantime)**

**.**

**. //get a general idea of trantimes through graphing bc its fun**

**. graph twoway bar lnavgtrantime occ**

**.**

**. //used regression models**

**. reg trantime sex age occ educd popdenpuma popdenpwpuma [pw=perwt], cluster(route)**

**(sum of wgt is 5.9342e+06)**

**Linear regression Number of obs = 299536**

**F( 6, 7068) = 40.47**

**Prob > F = 0.0000**

**R-squared = 0.0066**

**Root MSE = 20.399**

**(Std. Err. adjusted for 7069 clusters in route)**

**------------------------------------------------------------------------------**

**| Robust**

**trantime | Coef. Std. Err. t P>|t| [95% Conf. Interval]**

**-------------+----------------------------------------------------------------**

**sex | -2.368856 .1905584 -12.43 0.000 -2.742408 -1.995305**

**age | .0044738 .0049055 0.91 0.362 -.0051423 .01409**

**occ | .0034881 .0004446 7.85 0.000 .0026166 .0043596**

**educd | .0172579 .0029647 5.82 0.000 .0114462 .0230695**

**popdenpuma | -.10863 .0657669 -1.65 0.099 -.2375527 .0202928**

**popdenpwpuma | .1116001 .0657902 1.70 0.090 -.0173684 .2405686**

**\_cons | 21.6547 .7560242 28.64 0.000 20.17267 23.13674**

**------------------------------------------------------------------------------**

**. reg trantime age occ educd popdenpuma popdenpwpuma [pw=perwt], cluster(route)**

**(sum of wgt is 5.9342e+06)**

**Linear regression Number of obs = 299536**

**F( 5, 7068) = 26.56**

**Prob > F = 0.0000**

**R-squared = 0.0036**

**Root MSE = 20.43**

**(Std. Err. adjusted for 7069 clusters in route)**

**------------------------------------------------------------------------------**

**| Robust**

**trantime | Coef. Std. Err. t P>|t| [95% Conf. Interval]**

**-------------+----------------------------------------------------------------**

**age | .0063613 .004876 1.30 0.192 -.0031971 .0159197**

**occ | .0051314 .0004603 11.15 0.000 .004229 .0060339**

**educd | .0174615 .0029051 6.01 0.000 .0117667 .0231563**

**popdenpuma | -.1210768 .0651126 -1.86 0.063 -.248717 .0065635**

**popdenpwpuma | .1241061 .0651366 1.91 0.057 -.0035811 .2517932**

**\_cons | 17.22728 .4975492 34.62 0.000 16.25193 18.20262**

**------------------------------------------------------------------------------**

**. reg trantime occ lnincwage educd lnpopdenpuma lnpopdenpwpuma lnavgtrantime [pw=perwt], cluster(route)**

**(sum of wgt is 5.9342e+06)**

**Linear regression Number of obs = 299536**

**F( 6, 7068) = 2313.63**

**Prob > F = 0.0000**

**R-squared = 0.2347**

**Root MSE = 17.905**

**(Std. Err. adjusted for 7069 clusters in route)**

**--------------------------------------------------------------------------------**

**| Robust**

**trantime | Coef. Std. Err. t P>|t| [95% Conf. Interval]**

**---------------+----------------------------------------------------------------**

**occ | .0015068 .0003413 4.42 0.000 .0008378 .0021758**

**lnincwage | .6106189 .0524885 11.63 0.000 .5077256 .7135121**

**educd | -.0079505 .0021708 -3.66 0.000 -.012206 -.0036951**

**lnpopdenpuma | 2.220336 4.204894 0.53 0.597 -6.022517 10.46319**

**lnpopdenpwpuma | -2.375552 4.19557 -0.57 0.571 -10.60013 5.849021**

**lnavgtrantime | 27.33043 .260216 105.03 0.000 26.82033 27.84053**

**\_cons | -65.63528 1.010829 -64.93 0.000 -67.61681 -63.65376**

**--------------------------------------------------------------------------------**

**. summ sex age puma pwpuma00 parea pwarea pop popdenpuma popdenpwpuma**

**Variable | Obs Mean Std. Dev. Min Max**

**-------------+--------------------------------------------------------**

**sex | 299608 1.619046 .4856222 1 2**

**age | 299608 41.34268 12.12265 16 93**

**puma | 299608 1749.747 1560.956 100 8106**

**pwpuma00 | 299608 1763.366 1558.357 1 8200**

**parea | 299608 9892.28 58558.06 3.458766 1329354**

**-------------+--------------------------------------------------------**

**pwparea | 299608 9892.522 58558.22 3.458766 1329354**

**pop | 299608 81927.62 75532.92 1693 228933**

**popdenpuma | 299608 96.14303 292.0293 .1555334 9140.289**

**popdenpwpuma | 299608 96.14245 292.0293 .1555334 9140.289**

**.**

**. log close**

**name: <unnamed>**

**log: C:\Users\kholl\Desktop\DR\log.log**

**log type: text**

**closed on: 28 Apr 2018, 20:10:39**

**--------------------------------------------------------------------------------------**

**X. Appendix C (Cross County Commute Times Program)**

*#Written By: William Gribbin in association with Gillian Garbus, Kyle Hollander, Addison Armstrong, Theresta Desir, and Tyler Allern*

*#The purpose of this program is generate data files showing the shortest drive times between schools in florida counties*

*#the extra extensions imported to make this program funciton*

**import** sys

**import** time

**import** csv

**from** math **import** sin, cos, sqrt, atan2

**from** operator **import** itemgetter

**from** googlemaps.client **import** Client

**from** googlemaps.distance\_matrix **import** distance\_matrix

*#helps give how long the program runs after execution*

start\_time = time.time()

apiusecounter = 0

*#A callable function that gives the distance between two geocodes*

**def** disCalc(latty1,longi1 , latty2, longi2):

*# approximate radius of earth in km*

R = 6373.0

*# 'converts the str into INT because thats how python grabs data from a csv*

lat1 = float(latty1)

lon1 = float(longi1)

lat2 = float(latty2)

lon2 = float(longi2)

*# 'The math that calculates the distance between 2 geopoints on the curve of the earth*

dlon = lon2 - lon1

dlat = lat2 - lat1

a = sin(dlat / 2) \*\* 2 + cos(lat1) \* cos(lat2) \* sin(dlon / 2) \*\* 2

c = 2 \* atan2(sqrt(a), sqrt(1 - a))

distance = R \* c

**return** distance

*#A callable function that compares two counties schools and limits it down to one school dependent on discalc*

*#and gmapcalc*

**def** CSVGenerator(county1, county2, schoolLevel, timer, outfile):

schoolLevelCSV = schoolLevel + **'.csv'**

*#brings in the first county data collected from user and moves it to county1.csv to allow easier use*

**with** open(schoolLevelCSV, **'r'**) **as** csv\_file:

csv\_reader = csv.DictReader(csv\_file)

**with** open(**'county1.csv'**, **'w'**) **as** new\_file:

fieldnames = [**'Driver'**,**'SchoolNum'**,**'SchoolName'**,**'Street'**,**'County'**,**'State'**,**'Zip'**,**'Type'**,**'Lat'**,**'Lon'**]

csv\_writer = csv.DictWriter(new\_file, fieldnames=fieldnames)

csv\_writer.writeheader()

**for** line **in** csv\_reader:

**if** line[**'Driver'**] == county1:

csv\_writer.writerow(line)

*# 'brings the second county data collected from the user and moves it to county2.csv*

*# 'this allows easier use for testing and checking when the data is imported because it is saved*

**with** open(schoolLevelCSV, **'r'**) **as** csv\_file:

csv\_reader = csv.DictReader(csv\_file)

**with** open(**'county2.csv'**, **'w'**) **as** new\_file:

fieldnames = [**'Driver'**,**'SchoolNum'**,**'SchoolName'**,**'Street'**,**'County'**,**'State'**,**'Zip'**,**'Type'**,**'Lat'**,**'Lon'**]

csv\_writer = csv.DictWriter(new\_file, fieldnames=fieldnames)

csv\_writer.writeheader()

**for** line **in** csv\_reader:

**if** line[**'Driver'**] == county2:

csv\_writer.writerow(line)

*# 'Generates 2 Lists from my saved CSV data*

list1 = []

list2 = []

**with** open(**'county1.csv'**, **'r'**) **as** readfile:

csvread = csv.reader(readfile)

**with** open(**'county2.csv'**, **'r'**) **as** readfile2:

csvread2 = csv.reader(readfile2)

**for** row **in** csvread:

list1.append(row)

**for** row2 **in** csvread2:

list2.append(row2)

*# 'fixes an issue with extra [] in my lists from strait ripping the csv and removes headers*

list1clean = [x **for** x **in** list1 **if** x != []]

*#Imports the county 2 data into its own section*

lst1name = [item[2] **for** item **in** list1clean]

lst1name.pop(0)

lst1lat = [item[8] **for** item **in** list1clean]

lst1lat.pop(0)

lst1lon = [item[9] **for** item **in** list1clean]

lst1lon.pop(0)

*#Imports The Address For GMAP County 1*

lst1street = [item[3] **for** item **in** list1clean]

lst1street.pop(0)

lst1city = [item[4] **for** item **in** list1clean]

lst1city.pop(0)

lst1state = [item[5] **for** item **in** list1clean]

lst1state.pop(0)

lst1zip = [item[6] **for** item **in** list1clean]

lst1zip.pop(0)

*#Imports the county 2 data into its own section*

list2clean = [x **for** x **in** list2 **if** x != []]

lst2name = [item[2] **for** item **in** list2clean]

lst2name.pop(0)

lst2lat = [item[8] **for** item **in** list2clean]

lst2lat.pop(0)

lst2lon = [item[9] **for** item **in** list2clean]

lst2lon.pop(0)

*# Imports The Address For GMAP County 2*

lst2street = [item[3] **for** item **in** list2clean]

lst2street.pop(0)

lst2city = [item[4] **for** item **in** list2clean]

lst2city.pop(0)

lst2state = [item[5] **for** item **in** list2clean]

lst2state.pop(0)

lst2zip = [item[6] **for** item **in** list2clean]

lst2zip.pop(0)

i = 0

x = 0

AnaList = []

**while** i < len(lst1name):

**while** x < len(lst2name):

address1 = lst1lat[i] + **','** + lst1lon[i]

address2 = lst2lat[x] + **','** + lst2lon[x]

disty = (disCalc(lst1lat[i], lst1lon[i], lst2lat[x], lst2lon[x]))

AnaList.append([lst1name[i], lst2name[x], disty, address1, address2])

x = x + 1

x = 0

i = i + 1

sortlist = sorted(AnaList, key=itemgetter(2))

bestresult=sortlist[0]

bestresult.append(county1)

bestresult.append(county2)

*#gets the starting and ending address for gmap drive times*

startaddress = bestresult[3]

endaddress = bestresult[4]

*#removes the spaces from the drive time for url ease of use*

startaddressmod = startaddress.replace(**" "**, **"%"**)

endaddressmod = endaddress.replace(**" "**, **"%"**)

*#adds the drive time from google maps drive time api*

bestresult.append(gmapcalc(startaddressmod, endaddressmod, apiusecounter))

print(bestresult)

print(bestresult)

*#writes to csv*

**if** timer == 0:

**with** open(outfile, **'w'**) **as** new\_file:

*# configure writer to write standard csv file*

writer = csv.writer(new\_file, delimiter=**','**, quotechar=**'|'**, quoting=csv.QUOTE\_MINIMAL, lineterminator=**'\n'**)

*# Creates the header for the csv*

writer.writerow([**'School1'**, **'School2'**, **'DistanceApart'**, **'School1Lat'**, **'School1Lon'**, **'School2Lat'**, **'School2Lon'**, **'County1'**, **'County2'**, **'DriveTime'**])

writer.writerow(bestresult)

**else**:

**with** open(outfile, **'a'**) **as** new\_file:

*# configure writer to write standard csv file*

writer = csv.writer(new\_file, delimiter=**','**, quotechar=**'|'**, quoting=csv.QUOTE\_MINIMAL, lineterminator=**'\n'**)

writer.writerow(bestresult)

*#a callable function that returns the drive location of 2 gps coordinates*

**def** gmapcalc(location1, location2, apiuse):

api\_key = **'AIzaSyAn6U1F6NJIJx226L8sK5my\_ECvHm7k18o'**

gmaps = Client(api\_key)

data = distance\_matrix(gmaps, location1, location2)

timeaway = data[**'rows'**][0][**'elements'**][0][**'duration'**][**'text'**]

**global** apiusecounter

apiusecounter += 1

**return**(timeaway)

*#gives the formatting for running each schooling level in its own function*

**def** Elementary():

ElemList = open(**'DRIVERELEMENTARY.csv'**).read().splitlines()

i = 0

x = 0

timer = 0

**while** i < len(ElemList):

**while** x < len(ElemList):

Elist1 = ElemList[i]

Elist2 = str(ElemList[x])

**if** Elist1 != Elist2:

CSVGenerator(Elist1, Elist2, **'ELEMENTARY'**, timer, **'elementaryout.csv'**)

timer = timer + 1

x = x + 1

x = 0

i = i + 1

**def** Middle():

MiddleList = open(**'DRIVERMIDDLE.csv'**).read().splitlines()

i = 0

x = 0

timer = 0

**while** i < len(MiddleList):

**while** x < len(MiddleList):

Elist1 = MiddleList[i]

Elist2 = str(MiddleList[x])

**if** Elist1 != Elist2:

CSVGenerator(Elist1, Elist2, **'MIDDLE'**, timer, **'middleout.csv'**)

timer = timer + 1

x = x + 1

x = 0

i = i + 1

**def** High():

HighList = open(**'DRIVERHIGH.csv'**).read().splitlines()

i = 0

x = 0

timer = 0

**while** i < len(HighList):

**while** x < len(HighList):

Elist1 = HighList[i]

Elist2 = str(HighList[x])

**if** Elist1 != Elist2:

CSVGenerator(Elist1, Elist2, **'HIGH'**, timer, **'highout.csv'**)

timer = timer + 1

x = x + 1

x = 0

i = i + 1

*# '----------------------------------------------------------------*

*# Creates a final function that runs each grade level and notifies user of error*

**def** final():

**try**:

Elementary()

print(**'Elementary Completed!'**)

print(apiusecounter)

**except**: print(**'Elementary Scan Failed. Check That There are Matching Counties in your data have a matching driver county that matches the word EXACTLY'**, sys.exc\_info()[0])

**try**:

Middle()

print(**'Middle Completed!'**)

print(apiusecounter)

**except**: print(**'Middle Scan Failed. Check That There are Matching Counties in your data have a matching driver county that matches the word EXACTLY'**, sys.exc\_info()[0])

**try**:

High()

print(**'High Completed!'**)

print(apiusecounter)

**except**: print(**'High Scan Failed. Check That There are Matching Counties in your data have a matching driver county that matches the word EXACTLY'**, sys.exc\_info()[0])

*#------------------------------------------------------------------*

*#runs the final function*

final()

*#notifies you of how long it took to run this entire program*

print(**"This Analysis Took "**, ((time.time() - start\_time)/ 60), **"minutes to run"**)

*#stops script file from autoclosing for 10 min.*

time.sleep(1200)

**XI. Appendix D (Data Manipulation for PwPuma00 land area)**

**--------------------------------------------------------------------------------------**

**Description:** Microsoft SQL Code to find the pwpuma00 land area by summing the pumaland area for the pumas that create it

**Created by:** Kyle Hollander, Gillian Garbus, Jeff Garbus

**--------------------------------------------------------------------------------------**

SELECT

fc.[year], fc.[datanum], fc.[serial], fc.[hhwt], fc.[puma], fc.[pumaland] ,fc.[gq], fc.[perwt], fc.[occ], fc.[inctot], fc.[incwage], pp.[pwpuma00], fc.[trantime], fc.[statefip], fc.[age], fc.[sex], sum(convert(bigint,fc.pumaland)) as pwpumaland

INTO

datafile

FROM

finalcommute fc join [puma to Pwpuma00] pp

on fc.puma = pp.puma and fc.statefip= pp.statefip

GROUP BY

fc.[year], fc.[datanum], fc.[serial], fc.[hhwt], fc.[puma], fc.[pumaland] ,fc.[gq], fc.[perwt], fc.[occ], fc.[inctot], fc.[incwage], pp.[pwpuma00], fc.[trantime], fc.[statefip], fc.[age], fc.[sex], sum(convert(bigint,fc.pumaland)) as pwpumaland

**XII. Appendix E (Works Cited)**

AASHTO. Commuting in America 2013 - Brief 15: Commuting flow patterns. Technical report,

American Association of State Highway and Transportation Officials, 2015. from

http://traveltrends.transportation.org/Documents/B15 Commuting%20Flow% 20Patterns CA15-4 web.pdf.

Dewey, J. (2015, December 30). 2015 Florida Price Level Index. Retrieved January 26, 2018, from <https://floridapoly.edu/wp-content/uploads/2015fpli.pdf>

Green, A., Kutzbach, M., & Vilhuber, L (2017, April). Two Perspectives on Commuting: A Comparison of Home to Work Flows Across Job-Linked Survey and Administrative Files. *Center for Economic Studies, CES 17-34.* Retrieved from <https://ecommons.cornell.edu/bitstream/handle/1813/50976/CES-WP-17-34.pdf?sequence=2>

Mayock, T. (2015, July 01). Wages, Housing Prices and Commutes. Retrieved February 11, 2018, from <http://onlinelibrary.wiley.com/doi/10.1111/1540-6229.12097/full?refreshCitedByCounter=true>

McKenzie, B. (2015, August). Who drives to work? Commuting by automobile 2013(Rep.). Retrieved from <https://www.census.gov/content/dam/Census/library/publications/2015/acs/acs-32.pdf>

Mortellaro, A., & Sandoval, H. H. (2016, August 17). Why Do People Commute To Other Counties To Work? Retrieved from <https://www.bebr.ufl.edu/economics/website-article/why-do-people-commute-other-counties-work>

Kneebone, Elizabeth, and Natalie Holmes. (2015, March). “The Growing Distance between People and Jobs in Metropolitan America.” Retrieved from  
<https://www.brookings.edu/wp-content/uploads/2016/07/Srvy_JobsProximity.pdf>

Kopf, D. (2016, February 23). Which professions have the longest commute? Retrieved from <https://priceonomics.com/which-professions-have-the-longest-commutes/>

McKenzie, B. (2013, Jan). County-to-County Commuting Flows: 2006-10. Retrieved from

<https://www.census.gov/content/dam/Census/library/working-papers/2013/acs/2013-McKenzie.pdf>