The Battle of Neighborhoods

Or

"*Knick-knack, data whack, find the geek a home*"

Coursera Capstone Project

Brian Bloom
Dec 30, 2020

Introduction

Background

While there are quite a number of fictional business problems I could solve, I have a more immediate personal one that is of interest to solve. I have just returned from living in the Netherlands for the last 3 years, and am staying at my father's house near Austin, Texas, as temporary lodging. As a soon-to-be-certified data science geek, I would like to leverage my newly found data skills to help inform where I should look for a place to live in Austin that best suits me and my family. A realtor can provide me with nearly unlimited options that are for sale or rent, but would benefit from knowing where to start, which we can do by limiting that search to a specific subset of zip codes or neighborhoods. So my plan is to "connect the dots" with relevant available location data to help with that search.

Problem

My family would like to find a place that maximizes our different needs. To that end, I will need to pull together several datasets to find an optimal area of Austin to look for housing in, described in more detail below. First and foremost, I need to constrain my search to areas we can afford, so rent prices will be an important variable. My daughter is a swimmer, currently on a collegiate swim team, so I am interested in places close to swimming pools for her. My wife prefers not to drive and instead use busses or other mass transit, so we want places with good public transit options. For all of us, we need to obviously only consider places that we can afford. And finally, as a family, we'd really like places that have two of our favorite foods nearby: sushi, and pizza, and will weight those places higher. Thus places that satisfy all of these criteria are of most interest to us.

Deliverable

Using the datasets specified, I will construct a "heatmap" of the places that are most optimal for us, as well as a "top 5" list of zip codes that we can give to a realtor to help us find a place to live that most optimally meets our needs.

Audience

While this research project primarily benefits my family in finding a new home, it also serves as valuable experience for me in data analysis using Python, and I have published my full notebook so that others can leverage the code or methods that I used. Thus anyone with an interest in aggregating disparate datasets of geographic data may find this useful.

Data Sources

Zip codes:

The starting point for this project needs to be a list of the zip codes for Austin, ideally with latitude and longitude of the centroid for each. Fortunately, such data are widely available, and I used a public file I found from https://simplemaps.com/data/us-zips.

This file contained CSV records for every zip code in the entire US, so to keep the file more manageable, I filtered it for "Austin", "TX" first, removed columns I would not need, and then saved that as a local file for reuse.

Example data

```
"zip","lat","lng","city","state_id","state_name","zcta","parent_zcta","popu
lation","density","county_fips","county_name","county_weights","county_name
s_all","county_fips_all","imprecise","military","timezone"
"78701","30.27049","-97.74235","Austin","TX","Texas","TRUE","","9427","1621
.8","48453","Travis","{'48453':100}","Travis","48453","FALSE","FALSE","Amer
ica/Chicago"
"78702","30.26327","-97.71432","Austin","TX","Texas","TRUE","","23389","164
8.1","48453","Travis","{'48453':100}","Travis","48453","FALSE","FALSE","Ame
rica/Chicago"
"78703","30.29409","-97.76571","Austin","TX","Texas","TRUE","","20890","136
6.4","48453","Travis","{'48453':100}","Travis","48453","FALSE","FALSE","Ame
rica/Chicago"
"78704","30.24315","-97.76537","Austin","TX","Texas","TRUE","","48486","187
1.5","48453","Travis","{'48453':100}","Travis","48453","FALSE","FALSE","Ame
rica/Chicago"
```

```
"78705","30.29437","-97.73855","Austin","TX","Texas","TRUE","","33948","551
1.2","48453","Travis","{'48453':100}","Travis","48453","FALSE","FALSE","Ame
rica/Chicago"
```


Pools:

My daughter is a swimmer, and needs to train often, so having a nearby swimming pool is of key importance. Thankfully, the **City of Austin** maintains and publishes a JSON file with that data:

https://data.austintexas.gov/Recreation-and-Culture/Pool-Map/jfqh-bqzu

This data file contains all 45 of the public pools and play areas for Austin. I will need to filter out the latter ones (pool_type = "Splashpad") and retain the ones where pool_type = "Neighborhood", "Community", or "Regional". Each pool has latitude and longitude coordinates so I can compute a distance from each to any location. Geopy can do this for me using the lat/long values:

https://geopy.readthedocs.io/en/stable/#module-geopy.distance

Example data


```
{
  "pool_type" : "Neighborhood",
  "weekend" : "Closed",
  "closure_days" : "Closed for Winter",
  "weekday" : "Closed",
  "status" : "Closed",
  "pool_name" : "Patterson",
  "website" : {
    "description" : "Patterson Pool",
    "url" : "http://www.austintexas.gov/department/patterson-pool"
  },
  "phone" : "512-974-9331",
  "location_1" : {
    "latitude" : "30.296592644000043",
    "human_address" : "{\"address\": \"4200 Brookview Dr\", \"city\":
\"Austin\", \"state\": \"TX\", \"zip\": \"\"}",
    "needs_recoding" : false,
    "longitude" : "-97.71400165799997"
  }
}
```

Walkability and Transit:

My wife prefers to avoid driving (especially in traffic like Austin!) and prefers public transportation whenever possible. Thankfully, the site **WalkScore.com** offers metrics for this, and allows up to 5000 queries daily for free:

https://www.walkscore.com/professional/public-transit-api.php

Their service takes a latitude, a longitude, and a city, state and returns a JSON response with a 0-100 transit_score for each location. This can be used to find places that have the best transit scores for her ease of travel.

Example response data


```
{
 'transit_score': 72,
 'help_link': 'https://www.redfin.com/how-walk-score-works',
 'summary': '29 nearby routes: 28 bus, 1 rail, 0 other',
 'logo_url': 'https://cdn.walk.sc/images/transit-score-logo.png',
 'ws_link':
'https://www.walkscore.com/score/loc/lat=30.2705/lng=-97.7424/?utm_source=mooman.com&utm_medium=ts_api&utm_campaign=ts_api',
 'description': 'Excellent Transit'
}
```


Affordability:

Clearly there may be some neighborhoods that are simply out of my price range despite meeting my other criteria. So to keep my search limited to places I might actually be able to afford, I would like to incorporate rent data. Fortunately, Austin is one of the areas included in **federal HUD data**, which offers a spreadsheet of "Fair Market Rents" per zip code, so I can use that as a weighted criteria as well. https://www.huduser.gov/portal/datasets/fmr.html#2021_data

Their data is organized per zip code and offers the median rents for 2 bedroom, 3 bedroom, and 4 bedroom lodgings, along with a number of extraneous columns I can safely discard. Since these are medians, I will assume (hope) that a given zip code will have a few offerings that I could afford even if the median itself is over my budget. I will consider my ideal rent for a 3 bedroom to be 1800. The rental data from the federal HUD database lists the median rents, so even if the *median* cost in a zip is out of my range, it's likely that that zip code will contain some units at lower rates. So I'll use a 20% over my budget as the center point for what I hope to afford, or $2160.

Example data

```
"ZIP Code",HUD Area Code,HUD Metro Fair Market Rent Area Name,"SAFMR
0BR","SAFMR 1BR","SAFMR 2BR","SAFMR 3BR","SAFMR 4BR"
78701,METRO12420M12420,"Austin-Round Rock, TX
MSA","$1,590","$1,820","$2,150","$2,770","$3,310"
78702,METRO12420M12420,"Austin-Round Rock, TX
MSA","$1,030","$1,170","$1,390","$1,790","$2,140"
78703,METRO12420M12420,"Austin-Round Rock, TX
MSA","$1,160","$1,330","$1,570","$2,020","$2,420"
78704,METRO12420M12420,"Austin-Round Rock, TX
MSA","$1,240","$1,420","$1,680","$2,170","$2,590"
78705,METRO12420M12420,"Austin-Round Rock, TX
MSA","$1,400","$1,610","$1,900","$2,450","$2,920"
```

Bonus points: Food

All of my family is quite a fan of certain foods, in particular sushi and pizza. Having a home close to such places would be a distinct bonus. So I would like to query the **Foursquare** data for Austin specifically looking for locations that have pizza and/or sushi nearby. I originally intended to use a 2km search radius for each zip code centroid, but after determining that some of the zip codes are large residential areas, I expanded this to a 5km search radius.

For each query, I use the *'explore'* endpoint, and pass in the unique category ids for 'Sushi Restaurant' (categoryId = 4bf58dd8d48988d1d2941735) and 'Pizza Place' (categoryId = 4bf58dd8d48988d1ca941735) and get a count of how many are near each zip code centroid. This is scaled using feature scaling to produce 0-1 scores for each, and since both are important in our family, I then compute their harmonic mean for each zip code, which should promote the places with both, and penalize the places with only one.

Example response data

```
{'meta': {'code': 200, 'requestId': '5febb7d6d358eb7aca556826'},
 'notifications': [{'type': 'notificationTray', 'item': {'unreadCount':
0}}],
 'response': {'suggestedFilters': {'header': 'Tap to show:',
   'filters': [{'name': 'Open now', 'key': 'openNow'},
    {'name': '$-$$$$', 'key': 'price'}]},
  'headerLocation': 'Austin',
  'headerFullLocation': 'Austin',
  'headerLocationGranularity': 'city',
  'query': 'pizza',
  'totalResults': 51,
```

```
  'suggestedBounds': {'ne': {'lat': 30.447680045000045,
    'lng': -97.70897284566948},
   'sw': {'lat': 30.357679954999956, 'lng': -97.81312715433052}},
  'groups': [{'type': 'Recommended Places',
    'name': 'recommended',
    'items': [{'reasons': {'count': 0,
       'items': [{'summary': 'This spot is popular',
         'type': 'general',
         'reasonName': 'globalInteractionReason'}]},
      'venue': {'id': '523b35acbce69c7c875b38cc',
       'name': 'Pour House Pints & Pies',
       'location': {'address': '11835 Jollyville Road',
        'lat': 30.422826069375297,
        'lng': -97.7552983452688,
        'labeledLatLngs': [{'label': 'display',
          'lat': 30.422826069375297,
          'lng': -97.7552983452688},
         {'label': 'entrance', 'lat': 30.422861, 'lng': -97.755475}],
        'distance': 2309,
        'postalCode': '78759',
        'cc': 'US',
        'city': 'Austin',
        'state': 'TX',
        'country': 'United States',
        'formattedAddress': ['11835 Jollyville Road', 'Austin, TX 78759']},
       'categories': [{'id': '4bf58dd8d48988d1ca941735',
         'name': 'Pizza Place',
         'pluralName': 'Pizza Places',
         'shortName': 'Pizza',
         'icon': {'prefix':
'https://ss3.4sqi.net/img/categories_v2/food/pizza_',
          'suffix': '.png'},
         'primary': True}],
       'photos': {'count': 0, 'groups': []},
       'venuePage': {'id': '71979466'}},
      'referralId': 'e-0-523b35acbce69c7c875b38cc-0'}
```


Geojson:

After completing my exploratory analysis and computing the scoring metrics, I decided I wanted to also produce a visualization using a choropleth map, which necessitated finding geojson

representations for the Austin zip codes. Initially, I could only find a file for all of Texas which was an unmanageable 80 megabytes in size, but fortunately, I then found one specific to just Austin zip codes at:

https://openaustin.carto.com/u/oa-admin/tables/austin_area_zip_codes/public

This file was only about 1 MB in size and was able to be used for Folium mapping once I determined the proper key names in it.

Example data (excerpted)


```
{"type": "FeatureCollection", "features":
[{"type":"Feature","geometry":{"type":"MultiPolygon","coordinates":[[[[-97.
890016,30.209412],[-97.889286,30.208708],...<trimmed>...,
[-97.890016,30.209412]]]]},"properties":{"geodb_oid":3,"objectid":3,"zipcod
es_i":56,"zipcode":"78739","name":"AUSTIN","created_by":null,"created_da":n
ull,"modified_b":null,"modified_d":null,"shape_area":338340541.543711,"shap
e_len":97785.7933337692,"cartodb_id":3,"created_at":"2015-06-07T00:48:14Z",
"updated_at":"2015-06-07T00:48:14Z"}}
```


Methodology

Once the starting set of zip code data was seeded, each additional dataset was cleaned and merged into it following this logic:

The transit scores from WalkScore and the Foursquare restaurant data were API calls that are each rate-limited. So it made sense to reduce the number of queries needed by first eliminating as many ineligible areas as possible before making the calls. Thus the overall sequence of operations performed was:

1. Manually edit the initial zip code file to just be those in Austin proper, and not any of the neighboring towns (which would likely have more limited transit options into Austin).

2. Apply the rent data first to determine affordability. This would let me cull any locations that are cost-prohibitive and not worth exploring further. After merging the rent values for each zip code, I generated a normalized scoring metric for the "keepers" that prioritizes if they are particularly cheap and would save me money while meeting all my other needs.

   This metric uses the linear scaling formula $\mathbf{1 - (Rent_i - Rent_{min})/ (Rent_{max} - Rent_{min})}$ and produces a 0 to 1 score with '1' being the most affordable, and '0' being the most expensive.

I also created an "IsAffordable" boolean flag for all let me skip those in the API calls. I set the cutoff to any rents that exceeded $250 over my budget.

3. Next I applied the swimming pool data, since it a single downloaded JSON file, I was able to compute the distances for each pool to each zip code using a haversine formula and the latitude and longitude of each. Using these distances, I created another boolean flag "nearPool" and set that to False for any zip codes that were more than 10 miles to the nearest pool. After excluding those, I generated another normalized scoring metric with '1' having the closest pools and '0' having the farthest.

4. Since the transit score is a simple single score per zip code searched, it was performed by sending the lat/long of each zip code (where IsAffordable and nearPool were both true) and parsing the transit_score value from the JSON response. These transit scores were also mapped to a normalized scoring metric with '1' having the highest transit score (which turned out to be only a 60, much lower than either Rotterdam or Portland, the two places I lived most recently).

5. Then I queried the Foursquare API for each zip code, filtering out again based on IsAffordable and nearPool, and having it return a list of all of the sushi and pizza restaurants in a 5k radius with those two categoryIds. I was interested just in how many options there were, so I used a simple regular expression to count the matches for those categoryIds and added columns for the numbers of pizza and sushi restaurants found.

6. To compute a score for this, however, I reasoned that a place having 30 nearby pizza options was probably not much worse than a place with 60, since I would likely find the best couple of restaurants and stick to those. But a zip code with only 3 nearby would clearly be limiting. So I decided I needed to rescale the numbers so that 30 was more like 60 than like 3. This led me to applying a natural log function, which scaled those numbers to 3.4, 4.1, and 1.4, respectively, which accomplished the desired non-linear "compression". I then computed a matching normalized scoring metric using these logarithmic numbers for each pizza and sushi. And as mentioned earlier, since we want both scores to be high, I computed a harmonic mean of them using the formula:

   mean = (2 * pizzascore * sushiscore) / (pizzascore + sushiscore).

7. Finally, having standardized scores for affordability, pool convenience, transit, and restaurant availability, I could produce an overall composite score for each zip code. I decided that affordability was a much more important criteria (since if a neighborhood strained my budget it would limit everything else), and food selection was less important (because it would not be very hard to simply travel a little farther to good eats). The needs of my daughter and wife fell between both of these bounds. So for an overall score, I calculated a weighted average as follows:

   overall score = (**3***affordability* + **2***pool distance* + **2***transit score* + **1***food options*) / **8**

This final score for each zip code could then be used for determining the ranking for the best zip codes to begin our search in, and allowed me to produce a map of those areas.

Results

Since the process was piecemeal of applying one dataset at a time to the zip codes of Austin, I can share the incremental results before presenting the final findings of the research.

After loading the initial zip code data, I generated a simple map showing the center of each, with the circle scaled based on the population of that zip code, to first ensure it covered all the areas I needed to explore in Austin and no areas were accidentally omitted.

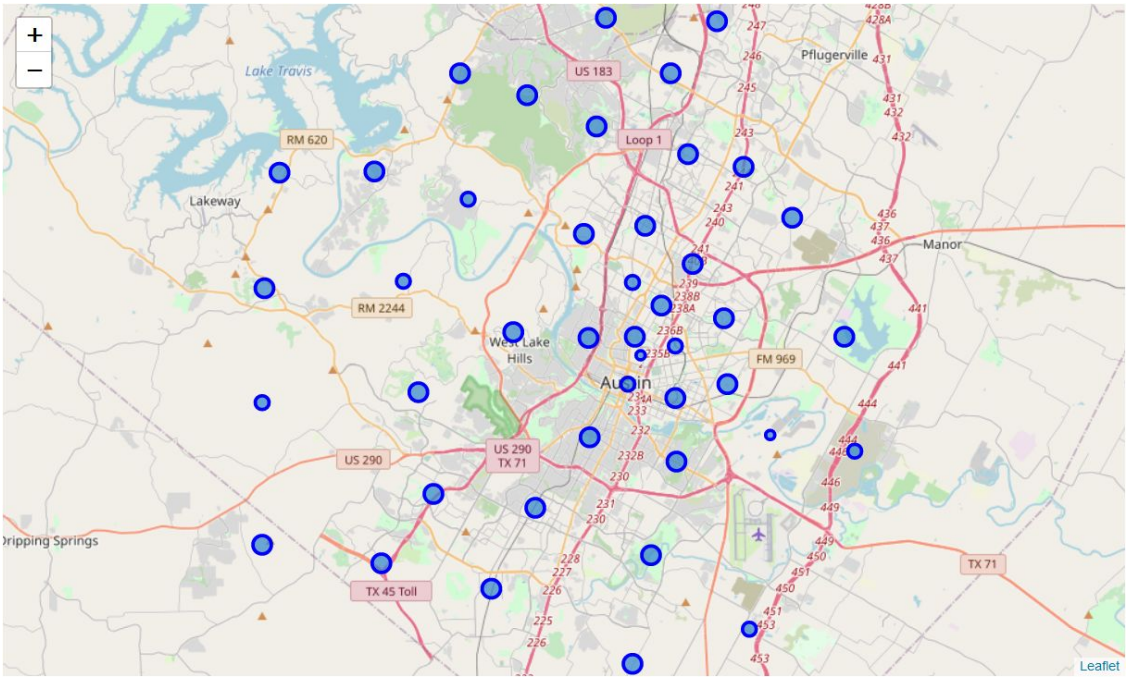

Image 1 *Starting zip code file for Austin, showing populations*

This looked like it covered the full area, and would be a good starting point for the other data sources.

Once the rent data was added, it shows pretty clearly that downtown and center city is too expensive, as are the lakeside areas to the west of town (those had a False for 'IsAffordable' and are coded in red). It looks like the eastern and northern sides of Austin are more affordable.

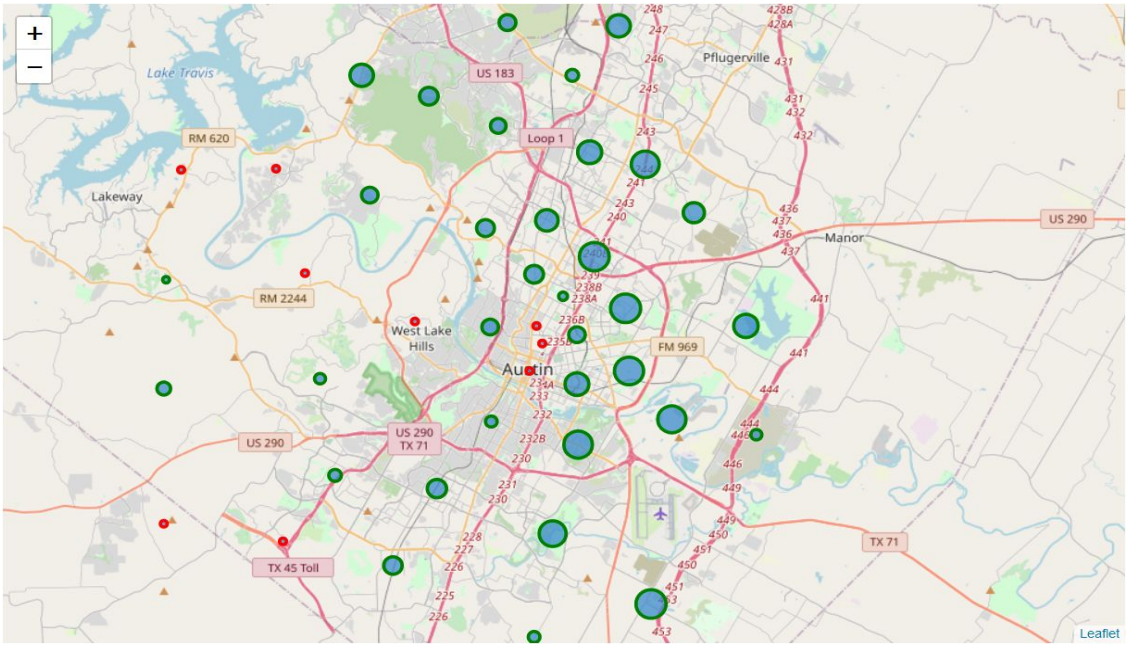

Image 2 *Map of 3 bedroom rents in Austin, larger disc = more affordable, red = prohibitive*

Then the pool data file was merged in, and the distances computed to the nearest public pool from each zip code were added and normalized. Anything further than 10 miles away was flagged as too far and is shown in red in the following map, Image 3. Fortunately Austin has plenty of swimming options so other than the hilly area west of town (which is already too expensive for us), it looks like we're well covered.

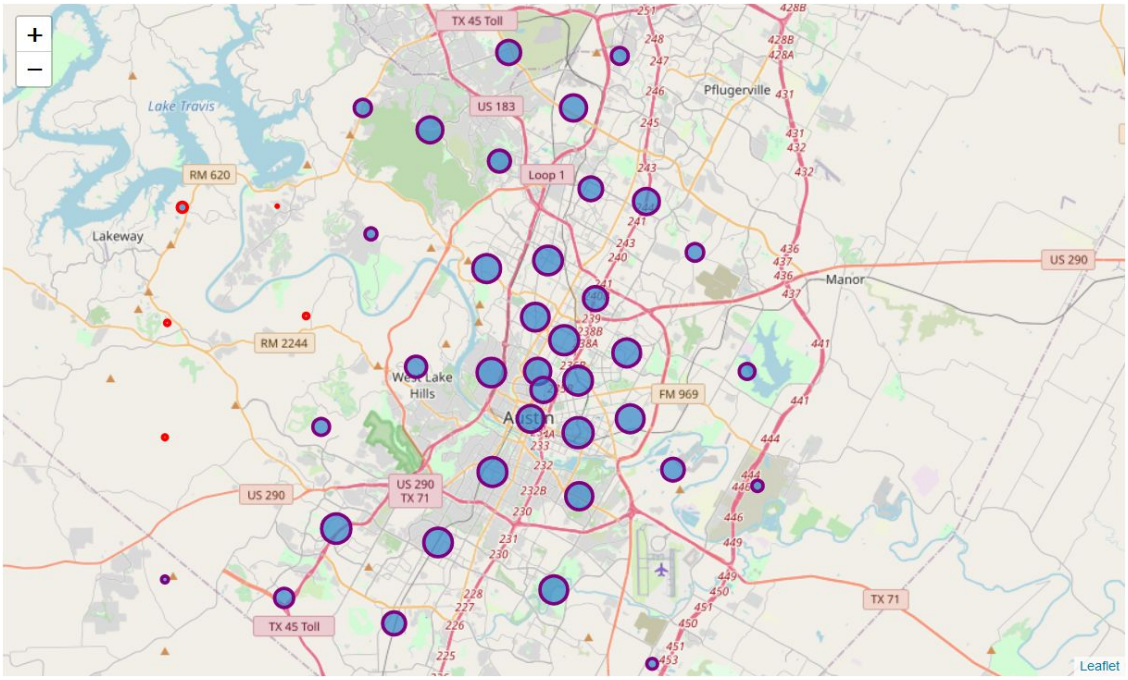

Image 3 *Map of proximity to public pools. Larger disc = nearby pool, red = none within 10 miles*

Following the pool data, I queried the transit score of each zip code and merged that data in. It's surprising to see that there are some gaps in areas that had previously been viable contenders. Downtown is of course well served, although the areas marked in red are those disqualified for affordability. The transit map in fact looks a lot like the affordability map which might have to do with the correlation of transit being used much more by lower class riders than middle and upper class. Image 4 shows the map of the results.

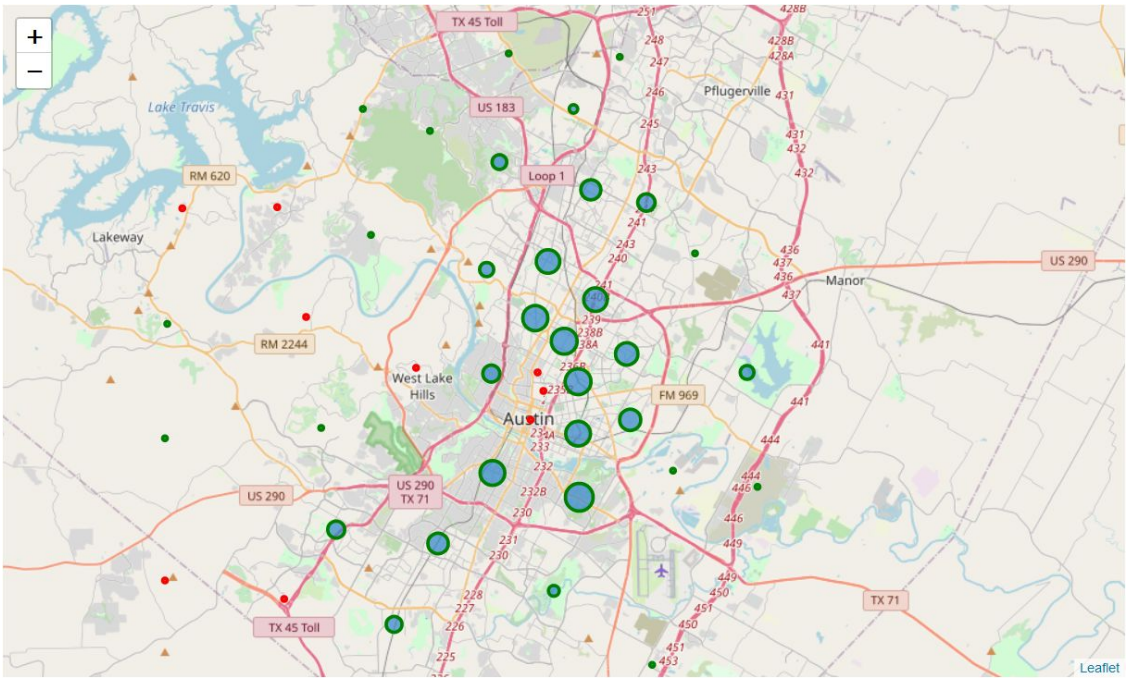

Image 4 *Map of transit scores. Larger disc = better transit options, red = too expensive.*

Since the food scores were not an essential component, but intended more as a "tie breaker" for similar places up to this point, no visualization was made specifically for those. But once all the scores were available at this point, I was able to compute an overall ranking of zip codes based on this score, listed in **Table 1** below. I have also added a choropleth map with the final scores coded from green to red based on fit.

	zip	city	rent3br	rentscore	nearestPool	poolscore	transitscore	foodscore	overallscore	finalrank
27	78741	Austin	1640	0.931507	Montopolis	0.866505	1.000000	0.882824	0.926294	1
10	78723	Austin	1590	1.000000	Bartholomew	0.898224	0.766667	0.712105	0.880236	2
37	78752	Austin	1600	0.986301	Brentwood	0.737103	0.816667	0.908961	0.871926	3
1	78702	Austin	1790	0.726027	Parque Zaragoza	1.000000	0.866667	0.968812	0.860028	4
8	78721	Austin	1620	0.958904	Givens	0.898096	0.716667	0.549722	0.831995	5
41	78757	Austin	1830	0.671233	Northwest	0.936935	0.816667	0.956283	0.809648	6
38	78753	Austin	1660	0.904110	Walnut Creek	0.818021	0.550000	0.666122	0.764312	7
40	78756	Austin	1950	0.506849	Ramsey	0.891479	0.883333	0.968812	0.754873	8
9	78722	Austin	2020	0.410959	Patterson	0.940369	0.916667	0.967125	0.739259	9
42	78758	Austin	1780	0.739726	Walnut Creek	0.717830	0.666667	0.901141	0.736164	10

Table 1 Final results showing top 10 overall ranked zip codes based on my search criteria.

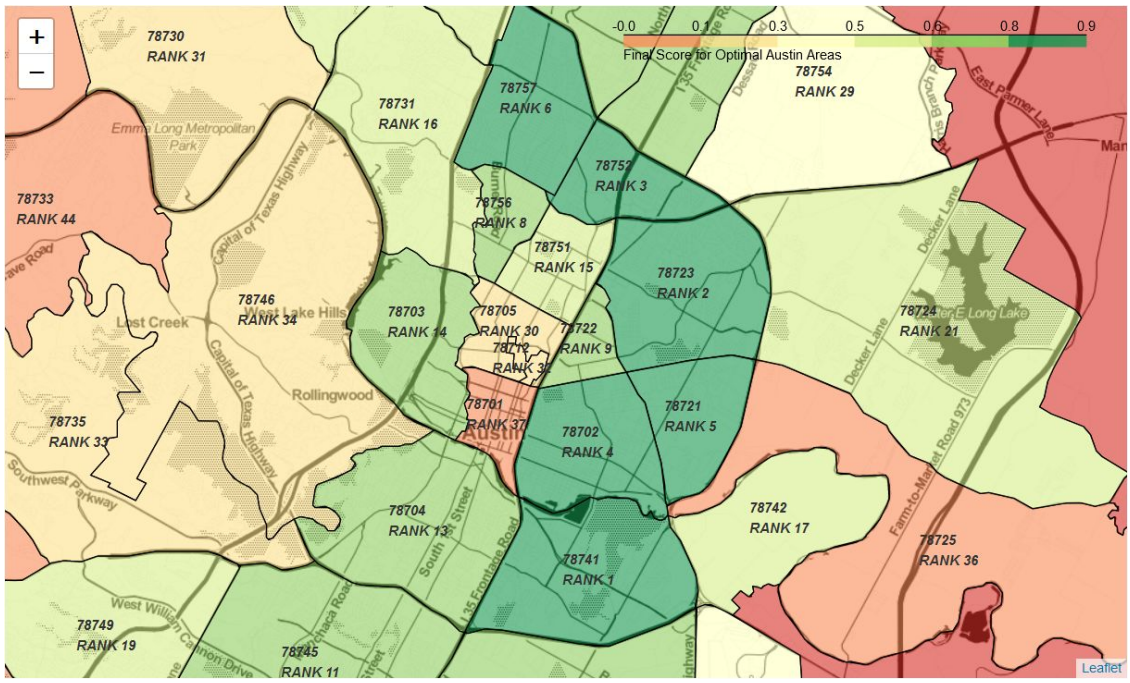

Image 5 *Choropleth showing the highest ranked Austin zip codes using my weighted averages.*

Discussion

It appears that the top contender is zip code 78741, with excellent scores in all categories. Some of these have higher rents, but have excellent food options (78722, overall 9th in the rankings), while others score highly on rent, but have lower transit and food scores (78723 and 78721 for example). In general, the data suggests that the eastern side of Austin, and north of center city, offer the best combination of affordable rents, pool access, transit, and food options, and I can furnish these top options to a realtor to begin my search for a new home for my family.

There are certainly other variables that could be added to this exercise to further optimize the results, but the real-world constraints are that at least for informing a starting point in my housing search, this is probably a sufficient level of accuracy and detail for the problem I'm trying to solve. We also have the ability to refine our search after consulting with a realtor based on other factors like crime, traffic, and other social parameters. I also realize that we may also have strong subjective feelings after visiting the areas in question.

Overall, this method can be arbitrarily extended as needed to embrace other variables. No machine learning or classification was used in this case, as my goal is simply to identify places that are highly rated in a number of attributes, and not to segment them based on similarity. This a computed metric seemed like the "right tool for the job".

Many of the decisions I made for thresholds (like my rent budget, how far is "too far" for a pool, and search radius for food) are somewhat arbitrary and based on rather subjective decisions after seeing the data. Similarly, using a logarithm to scale the restaurant densities, the calculations in sizing the markers in the maps, and of course the weightings I used for my overall score are also subject to debate, but this is an area that lacks any standards, and the cost of biased outcomes is low, so I felt safe in making those adjustments. For the research that I have done in my psychology studies, I would absolutely adhere to an academic rigor where any of those decisions would be made either prior to seeing the data or in consultation of others to ensure I am complying with data processing norms.

From a data quality perspective, the rental data from the HUD office may be the most questionable. The "fair market rent" values they calculate are based on federal housing assistance payments and may not accurately reflect the current rental landscape, and certainly don't take into account availability. So I am prepared to discover that those numbers are not as useful when it comes to what is open and available in Austin at this point in time.

The largest issues I ran into involved challenges with the Python syntax and variable types more than anything else. Referencing and maneuvering within JSON files, dataframes, dataframe slices, nested dictionaries, dictionaries in dataframes, numpy Series, lists, and strings was the greatest hurdle for this project. Thankfully the Jupyter notebooks allow for repeated attempts at debugging and the ease of making Folium map visualizations is very helpful.

Conclusion

Combining results from 4 publicly available data files and 2 API services, I was able to find an informed answer to the problem of "where should I look for a new home for my family?". My weighted metric points to certain neighborhoods in east and north Austin as the optimal places to consider. Because this was all performed in a Jupyter Notebook I can update my findings as needed if new data become available or I decide to add other criteria.

In fact, I often apply weighted metrics like the one I used here in order to help in my decision-making in many areas of my life. The university I just attended in the Netherlands was selected using a very similar methodology, albeit using spreadsheets rather than Python and JSON code. I am just delighted to have learned a new set of tools that I can use for both personal and professional challenges.

The full notebook used for this project is available at:
File:
https://github.com/brianbloom/Coursera_Capstone/blob/master/Brians_Capstone_Final.ipynb
Viewer:
https://nbviewer.jupyter.org/github/brianbloom/Coursera_Capstone/blob/master/Brians_Capstone_Final.ipynb