w205 Final Project
Austin's Gentrification and Its Impact on Neighborhood

Dave Huber, Keri Wheatley, John Sullivan, Yubo Zhang

Abstract: Austin, Texas, is one of the fastest growing cities in the US. In this project, we will use available data provided by Austin's open government project to evaluate its gentrification progress in recent years and how it impacts Austin neighborhood and local services. The Austin datasets are made available through the Socrata Open Data API (SODA). Together with data from Zillow.com, we can determine current status and comparisons over time of the various measures to determine the overall effect of gentrification on each neighborhood and the city of Austin as a whole.

Introduction: Austin, Texas, is one of the fastest growing cities in the United States. In recent years, the city has received attention from investors and companies interested in its location and unique culture aptly summarized by the city's slogan "Keep Austin Weird". This sleepy town is poised to become one of the United States' leading technology hubs in the next few decades. Like other fast-growing cities, Austin has been experiencing gentrification in many of its oldest neighborhoods. Gentrification is easy to define and to identify, however, it is hard to determine the intended and unintended consequences within the neighborhoods experiencing gentrification. How do city services respond to neighborhoods becoming more or less wealthy? Are more potholes getting fixed in richer neighborhoods? Are gentrifying areas experiencing more road closures and resident complaints? Using data provided by Austin's open government initiative, we want to find the answers to some of these difficult questions. Our project will consist of two parts. First, we will create a metric to identify gentrifying neighborhoods within Austin. Second, we will analyze various city data streams to understand the effects of gentrification on the neighborhood infrastructures and residents.

Objectives of this project:

1. How do we determine gentrification?

With real estate valuation data from Zillow, we will be able to see how the wealth of neighborhoods changes over time both in absolute and relative terms. There is no generally accepted metric for gentrification so we will have to make our own based on the information we have. Our initial approach will be to identify areas with houses increasing in property value that outpace the increases in the city as a whole.

2. How does gentrification affect neighborhoods?

We are interested in how the influx of more affluent residents and gentrification affect neighborhoods in the following perspectives:

- ➢ Does gentrification affect restaurant food quality and sanitation practices (sanitation, food safety, proper maintenance)?
- ➢ Could gentrification lead to better services from the city (311, potholes)?
- ➢ What happens to code complaints? Does the number of complaints go up in gentrifying areas?
- ➢ Does overall consumption in the area increase in gentrifying areas (water, garbage and electricity)?
- ➢ Does racial profiling increase or decrease with gentrification?

Architecture overview:

This data storage and retrieval system periodically downloads new data at a set interval, loads it into a Postgres database, transforms it, and computes aggregate information that can be visualized with Tableau.

The architecture consists of a main control script written in Python that is driven by a configuration file and contains three main parts: load, transform, and aggregate. The data loading and aggregation functions were written abstractly and their behavior is controlled primarily by configuration. The middle transformation layer allows for customization of data flow per source, if needed. This architecture allows for new data sources, aggregations, and visualizations to be added in the future with little to no change in the codebase.

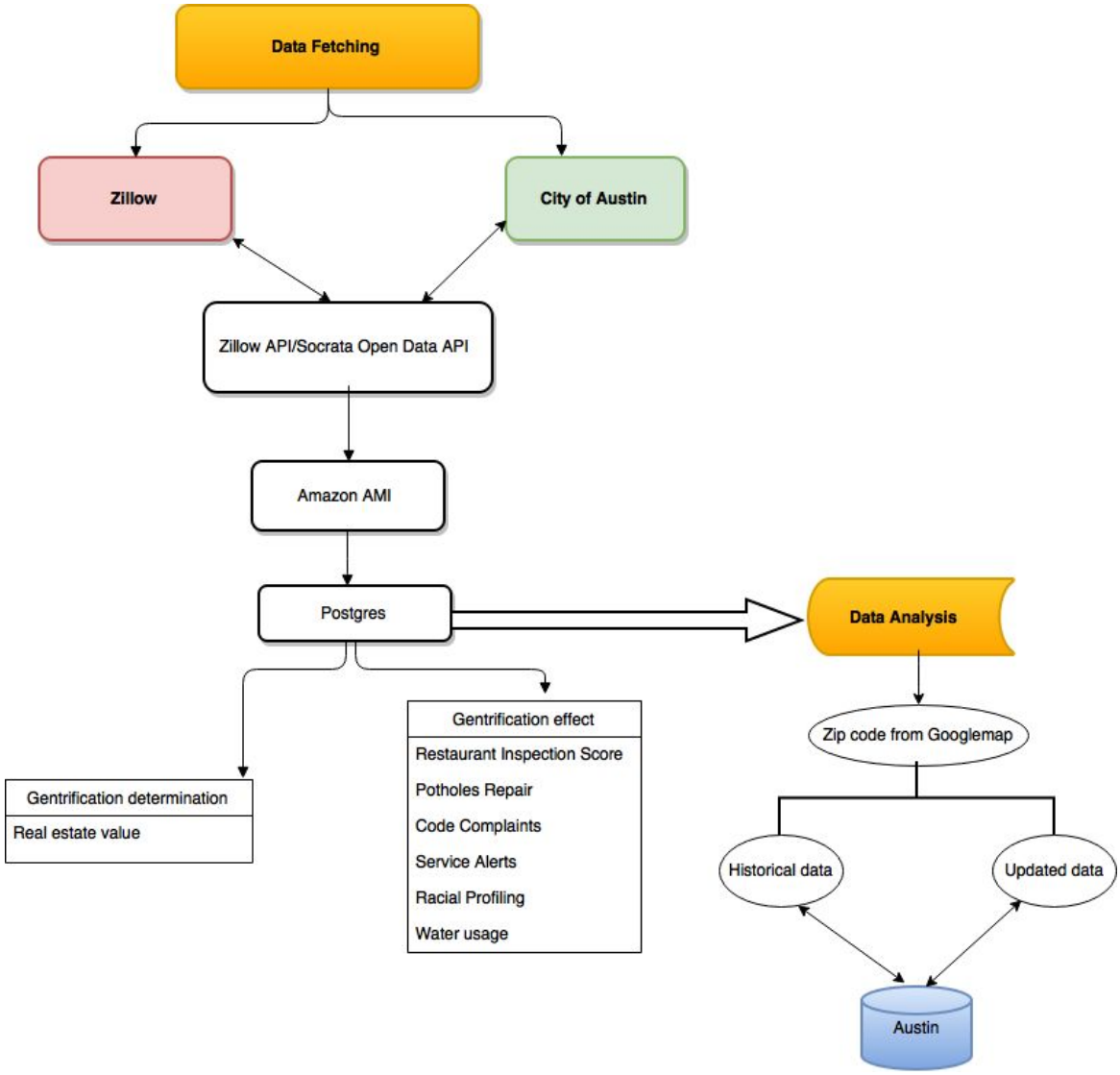
After the initial load of data, information about the last loaded record from each source table is stored in that table's section of the configuration file. This information is then referenced next time the program runs to determine if there is new information to download. Updating the data incrementally in this fashion keeps network transfer to a minimum. Currently, network transfer time is not a concern, but if the size of the system or the velocity of incoming data grows, this feature is expected to be an asset.

The data sets that will be leveraged are updated at varying rates, from monthly to hourly, and are accessed via direct download from Zillow and the Socrata Open Data API (SODA). There is a clear way to add new source download types in the event that different sources are added. The currently supported transfer methods do not provide a straightforward way to query about recently added data, so instead, the data update script is scheduled with the Linux task scheduler, Cron, to run once per day and the method described above is implemented to determine if new data is available.

As incoming data is stored in Postgres, and as this data flows in, aggregate measures (restaurant ratings, city services, garbage collection, etc.) per zip code will be computed and

updated. After the initial batch load of historical data completes, the database provides current status and comparisons over time of the various measures to determine the overall effect of gentrification on each zip code and the city of Austin as a whole.

The data is then served to end users through Tableau. Most of the aggregate tables are loaded directly into Tableau but some require custom SQL to be run in Tableau in order to format them in the way that is needed for reporting. Tableau can either live connect to Postgres or use cached extracts downloaded from Postgres. Currently we are using extracts for both performance and cost reasons. Tableau can query extracts faster than it can query Postgres and extracts also mean that we do not have to have the AMI constantly running on AWS in order to interact with the report. The downside to extracts is that they must be manually refreshed by the user if they want to update the data. This process just involves telling Tableau to refresh all extracts and takes about 5 minutes. In the future if a Tableau server is setup for the report periodic extract refreshes can be scheduled.



Required packages: Amazon EC2, Postgres, Tableau, Python 2.7 with psycopg2, googlemaps, pandas, json, datetime, requests, sqlalchemy

Data source:

Data Description	Link	How can we use it?
Home value and rent	http://files.zillowstatic.com/research/public/Zip/Zip_MedianValuePerSqft_AllHomes.csv http://files.zillowstatic.com/research/public/Zip/Zip_Zhvi_AllHomes.csv http://files.zillowstatic.com/research/public/Zip/Zip_Zri_AllHomesPlusMultifamily.csv http://files.zillowstatic.com/research/public/Zip/Zip_ZriPerSqft_AllHomes.csv	Determine changes in home values and rental price over time
City Construction Permits	https://data.austintexas.gov/Permitting/Issued-Construction-Permits/3syk-w9eu	Determine changing total values of construction permits over time
Restaurant Inspection Score	https://data.austintexas.gov/dataset/Restaurant-Inspection-Scores/ecmv-9xxi	Determine if wealthier neighborhoods have higher restaurant safety scores
Potholes Repair	https://data.austintexas.gov/Government/Pothole-Repair/fmm2-ytyt	Determine if wealthier neighborhoods receive faster city services
Code Complaints	https://data.austintexas.gov/Government/Austin-Code-Complaint-Cases/6wtj-zbtb	To determine if gentrifying neighborhoods receive more complaints (land use violation, property abatement)
Service Alerts	https://data.texas.gov/dataset/Service-Alerts/avj9-39zb	To determine increased or decreased disruption in these areas (ex. road closures)
Racial Profiling arresting and citation	https://data.austintexas.gov/Public-Safety/2014-Racial-Profiling-Dataset-Arrests/rnv4-98ze https://data.austintexas.gov/Public-Safety/Racial-Profiling-Dataset-2015-Arrests/vykk-upaj https://data.austintexas.gov/Public-Safety/2016-Racial-Profiling-Dataset-Arrests/834s-nvqn https://data.austintexas.gov/dataset/2014-Racial-Profiling-Dataset-Citations/mw6q-k5qy https://data.austintexas.gov/Pu	To find correlations between gentrification and racial profiling

	blic-Safety/Racial-Profiling-Dataset-2015-Citations/sc6h-qr9f https://data.austintexas.gov/Public-Safety/2016-Racial-Profiling-Dataset-Citations/gcpe-gehi	
Residential and Commercial Water Usage	https://data.austintexas.gov/Utility/Austin-Water-Commercial-Water-Consumption/5h9c-wmds https://data.austintexas.gov/Utility/Austin-Water-Residential-Water-Consumption/sxk7-7k6z	To determine if gentrification would impact residential and commercial water usage

Github repo: https://github.com/keriwheatley/w205_final_project.git

Directory in github repo:

File name	Location	File description
data_dashboards.twbx	w205_final_project/analysis/data_dashboards.twbx	Tableau program containing pre-loaded dashboards
__init__.py	w205_final_project/load/__init__.py	Load python files
aggregate_data.py	w205_final_project/load/aggregate_data.py	Function to aggregate data into final state after it is extracted and transformed
config.ini	w205_final_project/load/config.ini	Configuration file for running different extract, transform, and aggregate commands
config_clean.ini	w205_final_project/load/config_clean.ini	Backup version of config.ini for future reference
custom_map_data.py	w205_final_project/load/custom_map_data.py	Applying custom transformations to data between extract and aggregate commands (ex. find zip code using street address)
extract_data.py	w205_final_project/load/extract_data.py	Provide data extraction functions
update_tables.py	w205_final_project/load/update_tables.py	Parses config file to load all data and build updated aggregates

screenshots	w205_final_project/screenshots/	Screenshots of activated running applications
data_dictionary.csv	w205_final_project/setup/data_dictionary.csv	Data dictionary for status codes from data sources
load_zip_code_map.py	w205_final_project/setup/load_zip_code_map.py	Setup and create zip code map
table_setup.sh	w205_final_project/setup/table_setup.sh	Create tables
runApp.sh	w205_final_project/setup/runApp.sh	Update update_tables.py daily
transform_map.csv	w205_final_project/setup/transform_map.csv	Mapping logic for extract to aggregate tables
zip_code_map.csv	w205_final_project/setup/zip_code_map.csv	Raw zip code data
readme.txt	w205_final_project/readme.txt	Instruction on setting up and running the application

Future direction and potential improvement:

- For this project, we used Postgres and Tableau for storing and graphing incoming data. In the future Postgres and Tableau will still be able to handle incoming data that is many times as large as what is currently being hosted. However, when future users grow exponentially, scaling up would be needed. In this case, we would need to have our own server instead of Postgres or Tableau server if user number goes up.
- In the future, when we plan to include more major cities for this project, the current architecture should be still reliable to handle more data, however, we may need to purchase additional storage from AWS.
- One of the limitations we encounter is due to the fact that there is no automatic way to get zip code data. We used Google map for this exercise, however, Google map is not free to use. In the future, a more affordable and automatic way to get zip code data would be needed.
- Zillow provides data for free across a range of different metrics but the completeness of their data is inconsistent. Many of their metrics are missing a non trivial number of ZIP codes. This is why we decided to use rental data instead of home value data; the rental data was available for a much greater number of ZIP codes. There are superior sources of real estate transaction data but they must be purchased. Should this project expand

and get funded an investment in improved real estate data would be a good way to improve the analysis.

- One of the challenges we faced in this project is the different frequency of incoming data from Austin government website. For example, racial profiling and pothole repair have different update frequency. For the future direction and long term goal, it is important to figure out when it is the best frequency for the data to be updated.
- Architecturally, this design will hold for a much larger system, but configuration would become increasingly difficult. Configuration information would soon need to be moved to database storage for ease of access. A user interface, most likely a simple thin client, to modify configuration would also be required.