Topographical Modeling of the Boston Housing Market

a study by Daren McCulley for CS 591B – Networks and Markets

Abstract

Modern computing provides a way to model elaborate structures and systems by reducing them into discrete nodes at fine enough granularity to capture the underlying complexity. Finite element modeling (FEM) has revolutionized many fields, like meteorology, allowing software to supersede more expensive or less reliable methods. This study applied the principles of FEM to an urban housing market to explore the hypothesis that like the weather, dense housing markets have fronts, currents, and patterns that propagate through an interconnected system, which can be forecasted using a similar technique.

Research on housing markets tends to introduce multiple datasets, variables, and controls in an effort to build a complex model suited to a complex problem[1,2,3]. By contrast, this study relied solely on property assessment data, which all major cities have a financial stake in producing accurately on a routine basis. Raw assessment data was distilled into a series of mesh-like graphs modeling the distribution of property value throughout the city of Boston in a given year. The weighted nodes and edges in each graph constituted the finite element model used to predict market behavior. While this study focused on Boston, the methods employed are easily applicable to any city for which assessment data is readily available.

Introduction

In the United States, cities are home to over 62% of the total population[4]. Along with the benefits of population density on that scale come a host of challenges facing the governments and organizations charged with managing it. This study attempts to shed light on one of those challenges; providing affordable housing options for residents of all income levels. Even as the market continues to place upward pressure on the value of urban housing, cities continue to rely on an economically diverse labor force in order to function. For every university president, there is a small army of adjunct professors; for every bank manager, scores of tellers; and for every surgeon, a team of lower-wage workers that keep the hospital running.

Cities have an interest in preventing the displacement of their lower-income workforce[5]. However, they have limited financial and political resources to check the forces making it unaffordable for many residents to stay. Applying FEM to predict localized changes in the housing market could help decide where to invest those scarce resources. The sections that follow describe the steps used to construct a discretized model from the raw data, the way interactions between nodes give rise to latent structure, and the results of applying this approach on the property assessment data for Boston from 1985 to 2016.

Data Sources

Boston publishes annual property assessments on its public data portal[6]. From the 2016 assessment records, I collected the parcel id (PID) and geographic coordinates, if available, for every assessed parcel. PID is a key attribute and the only information necessary to scrape the city's online assessment site[7] for each parcel's assessment history. Historical assessment records include property type, year, and

assessed value for every year the property was assessed starting in 1985. In addition to the historical assessments, each parcel's address and living area were also scraped.

Location is provided for roughly 60% of the parcels in the 2016 assessment records, but is required data in order to build an accurate model. The only other data source used in this study was Google's Geocoding API, which I used to translate an address into geographic coordinates for the 40% of records missing location data. Of the 169,199 parcels assessed in 2016 only 515 were set aside for lack of a latitude and longitude in Boston and the majority of these were non-residential.

Data Processing

All of the raw data was stored, indexed, and processed using MongoDB and the PyMongo library for Python. Only one, two, and three-family homes along with condominiums and four-to-six-unit apartment buildings were selected for this study. Early on, I decided to use value per square foot to compare the cost of housing on equal grounds regardless of the amount of living area available. For the sake of brevity, this paper uses value interchangeably value per square foot. Larger apartment complexes and multi-use commercial/residential parcels were rejected because assessed value in their case encompasses more than just the value of the residential space and the reported living area is not entirely dedicated to housing tenants.

For the approximately 120,000 parcels remaining, a value per square foot was calculated for each year of available historical assessment data and adjusted for inflation based on 2016 dollars. By using John Tukey's method[8] to detect outliers, I removed a relatively small number of parcels from the dataset based on abnormally high or low values. However, I saved those parcels in a separate collection to determine the impact this step has on the model and the validity of the outlier label using a less objective approach in the future.

Model Design

In order to approximate the distribution of housing value using discrete elements, the entire map of Boston was subdivided into tiles through a process known as tessellation. A hexagonal shape was chosen for the tile because six shared edges leads to a well-connected graph following the transformation described in the next section. Additionally, the centers of the hexagonal tiles are equidistant to the centers of each neighbor. A side length of 0.1 miles was chosen, somewhat arbitrarily. Obviously, the granularity of the model is dependent on this choice. One might consider what resolution is necessary to capture the characteristics of the specific city being modeled before choosing this parameter.

Each hexagonal cell was instantiated with a unique id and the six coordinate pairs describing its location on the map. An initial block of cells was generated by choosing an origin south and west of Boston, iteratively performing geometric calculations and bounding the maximum latitude and longitude of the center points. Even for an area as small as Boston, it was important to consider the shrinking distance between lines of longitude at higher latitudes in order to keep the hexagons regular. MongoDB's geospatial indexing along with the geoWithin feature allowed me to quickly sort each parcel into its host cell. The historical assessment data for the parcels in each cell was aggregated to characterize that cell for each year between 1985 and 2016. The cell-level attributes stemming from the parcels contained were average value per square foot, total living area, and total number of parcels.

To prevent a cell with a small number of parcels or very little residential capacity from having an outsized influence on the model they were each screened for liveness. The threshold for liveness was set at 10 parcels and 30,000 square feet of total living area. Failure to meet either would disqualify a cell from a place in the FEM for not having large enough population of parcels or living space to provide a representative sample of that cell's characteristics. As in the case of choosing a specific size for each cell, the decision on where to draw this line was somewhat arbitrary and should be explored further.

Graph Transformation

The origin and bounds on the initial block of over 8,500 cells was purposefully chosen such that the live cells would fall somewhere in the middle of the block and never on the border. As a result, there exists a simple mathematical description for cell adjacency as a function of cell id. For every year between 1985 and 2016 the annual cell data is transformed into a graph through a series of steps described in the following paragraphs. I utilized the NetworkX graphing library for Python to test for connectedness or calculate shortest paths between nodes, which were both required to construct the FEM.

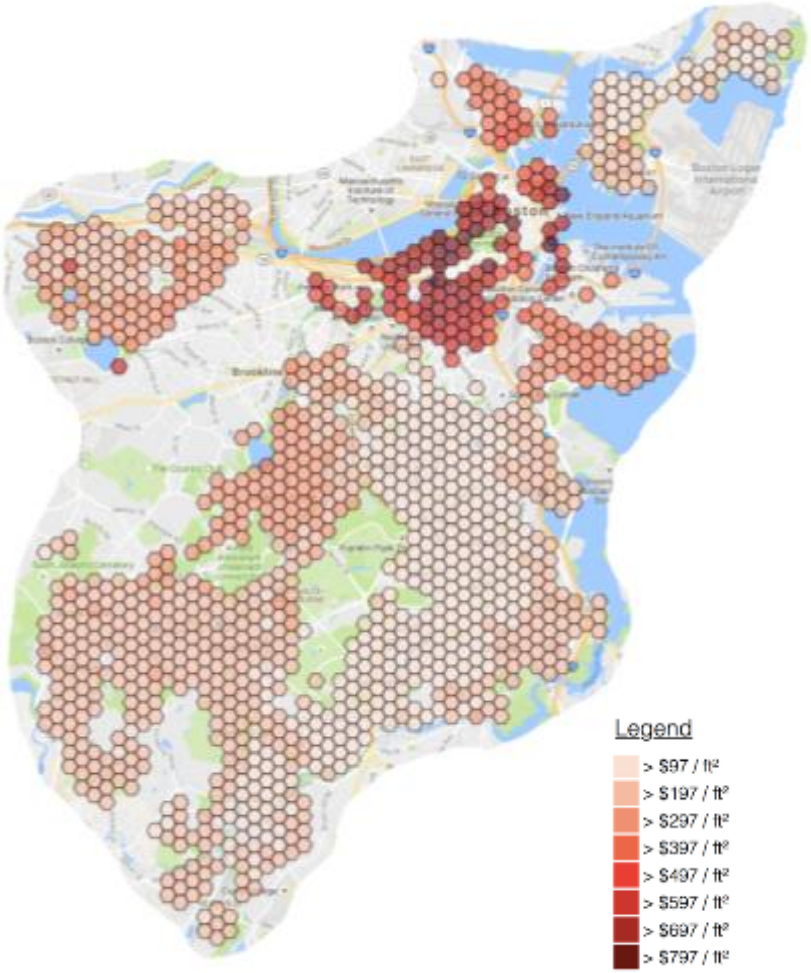
Each live cell corresponds to a node in the graph with weight equal to the average assessed value per square foot in that cell. Strong edges exist between two nodes if their cells are adjacent. The unique geography of Boston coupled with the threshold for liveness meant that none of the 32 graphs were completely connected using only strong edges. For example, Brookline and Boston University severed the nodes in Fenway from the nodes in Allston presenting a problem for modeling cell interaction.

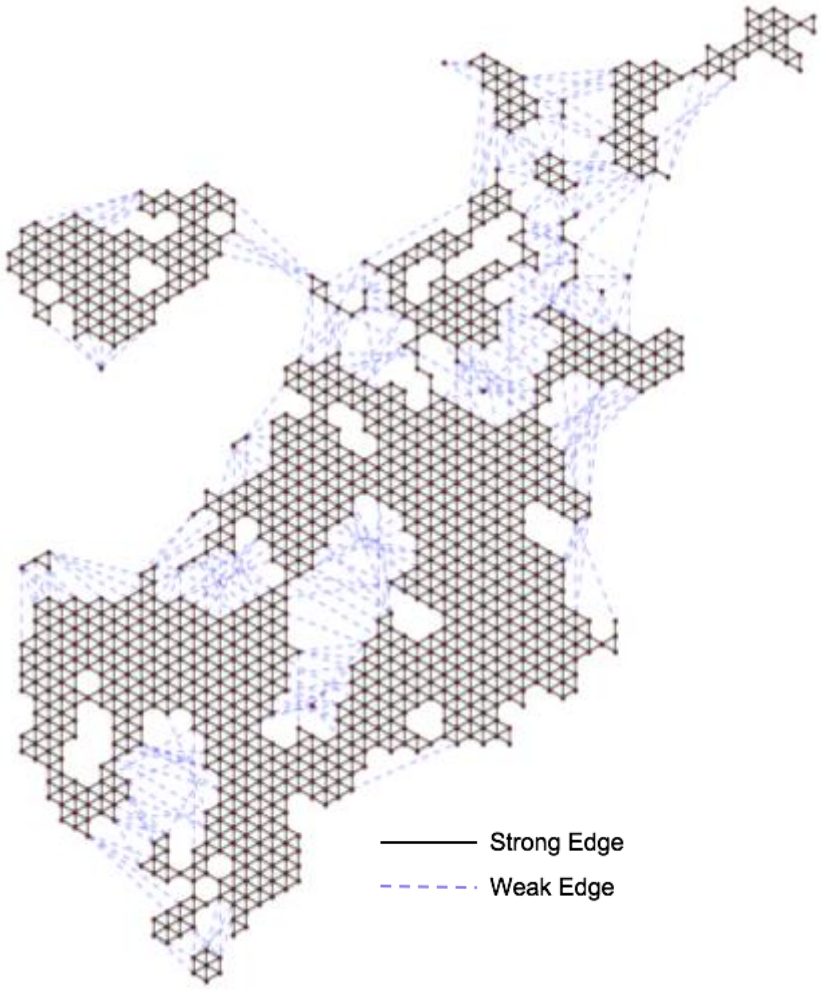
To address this issue, I introduced some new types of cells beginning with border cells. Any non-live cell that was adjacent to a live cell was labeled a border cell and added to the graph. This effectively grew the original group of live cells by one layer in all open directions. I repeated this process until the graph composed of all live and border cells was completely connected. In all iterations after the first, any non-live cell adjacent to a border cell became a new border cell. From the set of border cells, it was desirable to maintain only the cells necessary to connect the graph, which I labeled bridge cells. A bridge cell was any border cell which lied along the shortest path between any nearby pair of live cells, where nearby meant within a radius of eight.

Rather than allow bridge-cells to participate in the FEM as faux-live nodes existing for the sole purpose of allowing a distant interaction between disconnected components of the model I sought to replace them with weak edges between distant nodes. To control the number of weak edges and normalize the possible flow at each node I decided to cap the number possible edges at six; the number any interior live cell could possibly have due to the hexagonal tessellation. To construct weak edges, I labeled live cells adjacent to bridge cells as bridge-neighbors. If the shortest path between two nearby bridge-neighbors traversed only bridge cells, no short path through live cells existed, and each neighbor had the capacity for another edge, then a weak edge was added between that pair.

In summation, I artificially grew each graph to completely connect it and then removed the border nodes to leave only bridge nodes. Bridge nodes were replaced by weak edges where there was capacity for new edges and no short path through live nodes existed. In the end, I was left with a completely connected graph of only live nodes featuring strong and weak edges where each node had at most a degree of six. A picture of the cell map overlay for 2016 along with the corresponding graph this approach produced is on the following page.

2016 Cell Map and Corresponding Graph (FEM)





Flow Algorithm

A pseudo-code description of the flow/flush algorithm:

```
total_value = sum(value for each cell)

t = 0                              // t: epoch counter
while total_value > T:             // T: a fixed percentage of total_value

  // update flow vectors:
  for each cell i:
    for each strongly adjacent cell j to i:
      diff = i.value – j.value
      if diff > 0:
        i.strong_flow_vector.append(diff * strong_resistance)
      else:
        i.strong_flow_vector.append(0)

  for each cell i:
    for each weakly adjacent cell j to i:
      diff = i.value – j.value
      if diff > 0:
        i.weak_flow_vector.append(diff * weak_resistance)
      else:
        i.weak_flow_vector.append(0)

  // outflow
  for each cell i:
    k = 0                                      // k: flow_index
    for each strongly adjacent cell j to i:
      j.value = j.value + i.strong_flow_vector[k]
      j.flow_history[t] = j.flow_history[t] + i.strong_flow_vector[k]
      i.value = i.value - i.strong_flow_vector[k]
      i.flow_history[t] = i.flow_history[t] - i.strong_flow_vector[k]
      k++

    for each weakly adjacent cell j to i:
      j.value = j.value + i.weak_flow_vector[k]
      j.flow_history[t] = j.flow_history[t] + i.weak_flow_vector[k]
      i.value = i.value - i.weak_flow_vector[k]
      i.flow_history[t] = i.flow_history[t] - i.weak_flow_vector[k]
      k++

  // flush:
  for each cell i:
    if remaining value is greater than flush:
      i.value = i.value – flush
      total_value = total_value - flush
    else:
      i.value = 0
      total_value = total_value – i.value

t++
```

I divided the feature set used to train the classifier into two categories: static and dynamic. With the inclusion of weak and strong node degree the static features are completely known once the graph is built. While some of the static features are significant for their predictive value, the benefit of constructing the graph is that one can search for meaningful structure by employing dynamic analytical

algorithms, e.g. PageRank. The motivation for the algorithm that follows is a belief that when a person is displaced as a result of rising housing prices that person does not typically resettle further from their old home than necessary. Furthermore, like ripples in a pond the displacement of the middle-class by the upper-class may in turn displace the lower-class as the middle-class resettle in neighborhoods they once called home.

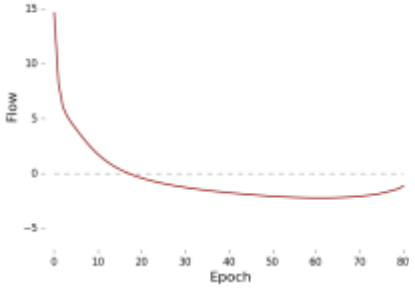
In contrast to the considerable time and energy spent obtaining the data and building the FEM, the algorithm is simple to describe and execute. It is an iterative method, with three phases for each epoch. Those phases in order are *update flow*, *outflow*, and *flush*. Flow vectors are updated as a linear function of the difference in value between adjacent nodes. The parameters strong (k_s) and weak resistance (k_w) moderate the flow along strong and weak edges respectively. Outflows are pushed all at once in a separate loop, meaning that changes in potentials resulting from flow occurring during this epoch are not realized until the beginning of the next update step.

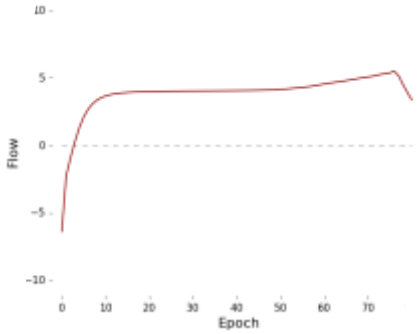
The flush step is a loss of value incurred by each node during each epoch, which is proportional to the total sq-ft (i.e. capacity) in that node. The flush constant k_f, which is multiplied by the total living area in each cell to determine the amount flushed at each epoch, is adjusted for the total value each year to keep the number of epochs fairly consistent from year to year. The model is very sensitive to variance in k_f, in part because there is such a wide variance in total living area, which quickly begins to dominate flow behavior when the flush constant is too high. Flush is akin to time-to-live in a routing protocol. It drives the algorithm to termination, but also places a limit on how far a dollar per sq-ft of value can travel in the graph before being consumed. Adding the flush step dramatically increased the predictive power of the model, which was originally allowed to run to equilibrium before termination.

This model is analogous to an electric circuit, where each cell has a battery whose voltage corresponds to the cell's initial value. Cells are wired to one another across two types of resistors corresponding to the strong and weak edges in the graph. Additionally, each cell has a capacitor with capacitance proportional to the living area in that cell, which is allowed to draw current from that cell's battery and eventually discharge to ground before charging again by draining the battery according to some imposed timing scheme. Finally, an oscilloscope is attached to each cell to monitor every electron that passes in or out of the cell and into the capacitor to paint a picture over time of what happens as all of the stored energy in the batteries eventually flushes to ground.

Feature Selection and Machine Learning

The flow history for two of the cells in the 2016 FEM is shown in the plots below.





Initially, I only considered net flow, i.e. the sum of all instantaneous flows for all epochs. However, that only described the integral flow and reported nothing about how a cell may have arrived at a certain net flow. Over time I developed the dynamic feature set to enrich the static features shown in the following table. I used cross-validation and some manual tuning to select relevant features for training an off-the-shelf support vector machine (SVM) classifier that ships with Python's SciKit Learn ML library.

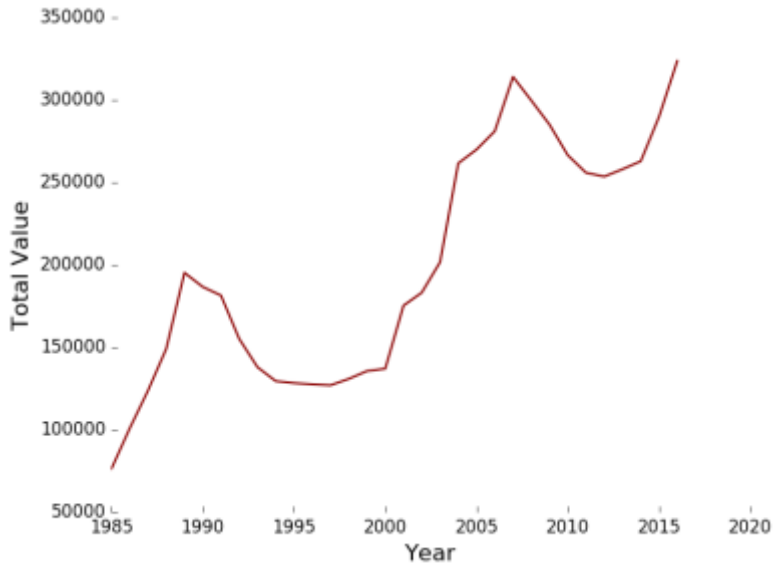
Static Cell Features	
parcel count	not selected
total living area	not selected
average assessed value	not selected
strong edge count	not selected
weak edge count	selected
rank	selected
Dynamic Cell Features	
max flow	selected
min flow	not selected
initial flow	selected
early flow	selected
middle flow	not selected
late flow	selected
net flow	not selected
outflow	selected
inflow	selected
absolute flow	not selected
total flushed	selected
flush ratio (total flushed / avg assessed value)	not selected

Classification requires separating observations into discrete categories. Since the objective was to predict precipitous rises home valuation, a cell's rank with respect to its value was chosen as the discriminating metric between a positive and negative observation. Specifically, cells which exhibited at least a 25-percentile jump in rank from the modeled year to the observation year were considered positive observations. For clarity, modeled year denotes which year an FEM was used to generate the feature set parsed by the trained SVM classifier into predictions about the change in rank observed during the observation year.

Results and Conclusions

At first, my goal was to train a model to forecast values five years in the future, because on average project timelines from conception to completion posted on the Boston Planning and Development Agency's website[8] were about that long. Despite some observable patterns emerging from looking at five year models, the conclusion I reached was that knowing which direction the overall market was headed was critical to making accurate predictions in those cases. The lower ranked cells significantly lagged the higher ranked cells when the seasons changed in the housing market, but their winters were colder and their summers hotter. Lower ranked cells are the cells with the best opportunity to gentrify or in the language of this study make a significant jump in the rankings. Their volatility in the face of overall market trends makes predictions on a short timeline very difficult without taking a stance on the overall trend.

The plot below shows the change in total value over time for the properties considered in this study and lends some credence to my claim that a five-year predication is a difficult goal.



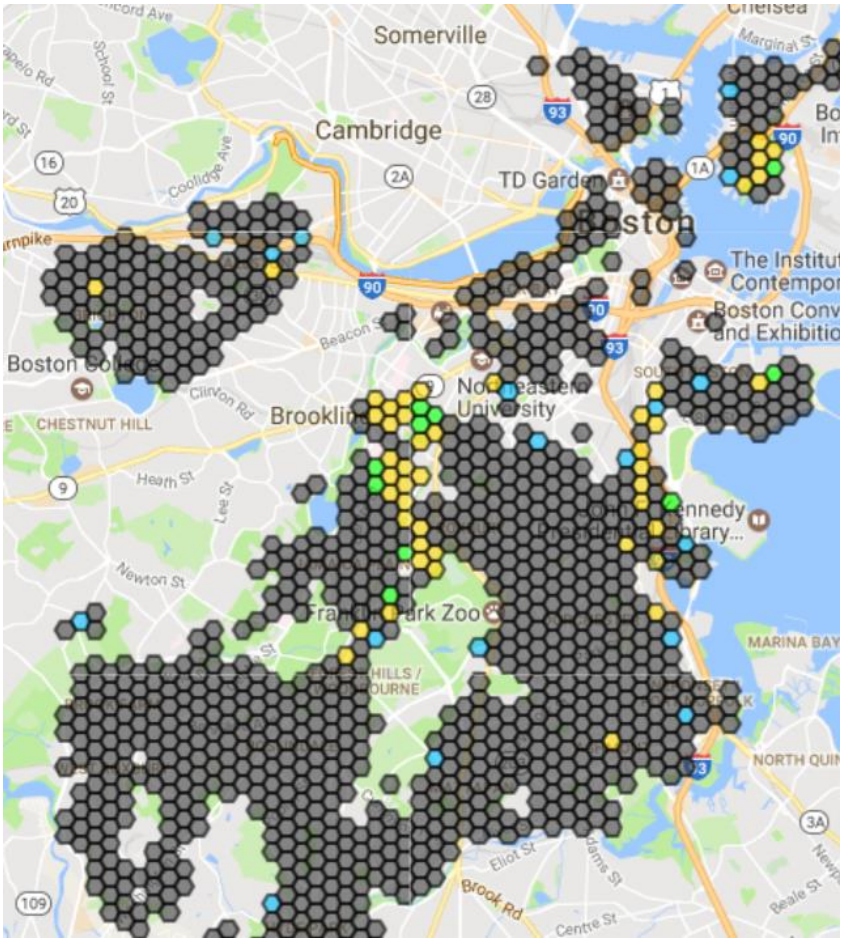
I abandoned five-year predictions in favor of something longer term and less sensitive to which period of time I considered. With over 30 years of data, I opted to train the SVM using the 1986 FEM and the observations in 2001. Then I tested this classifier by providing the 2001 FEM data and asking for predictions in 2016. One thing worth noting is that of the 36 positive observations in 2001, only 2 of those cells were still positive observations in 2016, which had a total of 54.

I liken the lack of overlap between the two sets of positive observations to the passing of fronts in describing the weather. Those cells have climbed the ladder and are done climbing, at least quite as rapidly as they were. However, what was true for those cells in 1986 is now true for some other set of cells in 2001, and in 2016 those cells will be in the vicinity of the highest performing areas on the map. The results of the 2001-2016 predictions are tabulated below along with a comparison to some simple heuristics.

Approach	# Selected	True Positives (54 possible)	Avg Δ-Rank of Selected
Topographical	28	10	193.18
Positive in ’01	32	2	141.66
Static Only	10	2	146.81

Topographical – this study’s approach, leveraging flow characteristics
Positive ’01 – sticking with the best performing cells in the training data (’86 - ’01)
Static Only – training the SVM using features available without running the algorithm (e.g. rank)

None of these methods do a particularly good job of picking out cells which will gain 25 percentiles or roughly 278 ranks between 2001 and 2016, but all three do a decent job predicting high performing cells. The average 193.18 increase in rank exhibited in the topographical model’s 28 selected cells was high enough to make me wonder how close to the actual 54 true positives these 28 were on the map. The following is a picture of the 2001 cell map with the 44 non-predicted positive results in gold, correct predictions in green, and incorrect predictions in blue.



There are certainly some errant blue hexagons, which I attribute mostly to including weak edges where perhaps there should not be. Correctly weighting edges, particularly weak edges, is the next step towards improving this model. Despite the errant blue hexagons, i.e. false-positives, there were very few gold hexagons more than two hops away from a blue or green hexagon. Considering that over 100 live cells were added between 1986 and 2016, which in part accounts for the larger number of positive observations, this is a promising preliminary result for using this approach to model how Denver or Austin might evolve into the next San Francisco or Boston.

I expanded the ’01 heuristic and my approach by a radius of both one and two cells to measure what a wider net would catch. I also determined the best range of contiguously ranked cells to do a one-to-one comparison on a 255-cell set against the two-cell expansion of my topographical approach. While a 255-cell range is enough to ensnare 40 out of 54 of the positive results the average rank increase is abysmal by comparison, meaning there is much more noise in that set than signal.

Approach	# Selected	True Positives (54 possible)	Avg Δ-Rank of Selected
Topographical-1	131	32	140.91
Positive ’01-1	90	10	124.08
Topographical-2	255	48	109.57
Positive ’01-2	141	17	112.01
Best Range 255	255	40	31.19

If it were possible to reduce the number of false-positives particularly the blue hexagons not in the vicinity of any gold ones in the figure above by refining the model to more accurately capture the paths along which displaced people will resettle this approach has merit as a long-term predictive tool.

Two final criticisms I have as the author of this study are the fact that bordering municipalities are treated as dead cells in the same way as the industrial zones between South End and Dorchester. However, this cannot possibly capture the impact that Brookline or Cambridge has on where people choose to live in the greater Boston area. The second is that I entered into this without a very good understanding of what urban planners or people in the real estate business already accept as common knowledge. It could very well be the case that Mission Hill, Jamaica Plain, and Andrew Square were considered up-and-coming in 2001 and it was a foregone conclusion that those would be the highest performing neighborhoods in the near future. That said, having a data-driven model to back up or refute those claims or even accomplish what used to require an expert opinion is certainly a worthwhile goal.

References

1. https://www.geog.uni-heidelberg.de/md/chemgeo/geog/lehrstuehle/gis/helbich_etal_2012.pdf
2. http://www.governing.com/gov-data/boston-gentrification-maps-demographic-data.html
3. http://link.springer.com/article/10.1007/s101090200086
4. http://www.census.gov/newsroom/press-releases/2015/cb15-33.html
5. http://web.williams.edu/Economics/ArtsEcon/library/pdfs/WhyIsGentrificationAProbREFORM.pdf
6. https://data.cityofboston.gov/
7. http://www.cityofboston.gov/assessing/search/
8. http://www.bostonplans.org/