

Causes for Eviction Notices in Assorted Neighborhoods of San Francisco

Aitken, Connor
`connor.aitken500@gmail.com`

Cheung, Leon
`leonjcheung@gmail.com`

May 20, 2015

Abstract

In recent years, the gentrification of San Francisco has become an increasingly controversial topic for those living in the Bay Area. We have set out to investigate whether the evictions we hear about in the news are morally justifiable or a result of landlords looking to generate greater revenues on their property, due to the great demand for housing. We pulled data from the city's open database initiative *dataSF* and used the C++ Programming Language to statistically analyze the data. We found that there is a veritably greater rate in evictions that we perceive to be motivated by greater potential profits, especially in neighborhoods which are exploding due to the rise of tech companies wishing to maintain offices in San Francisco. With this result in mind, we gain a greater perspective on the objective state of the state of housing in San Francisco.

1 Data Collection

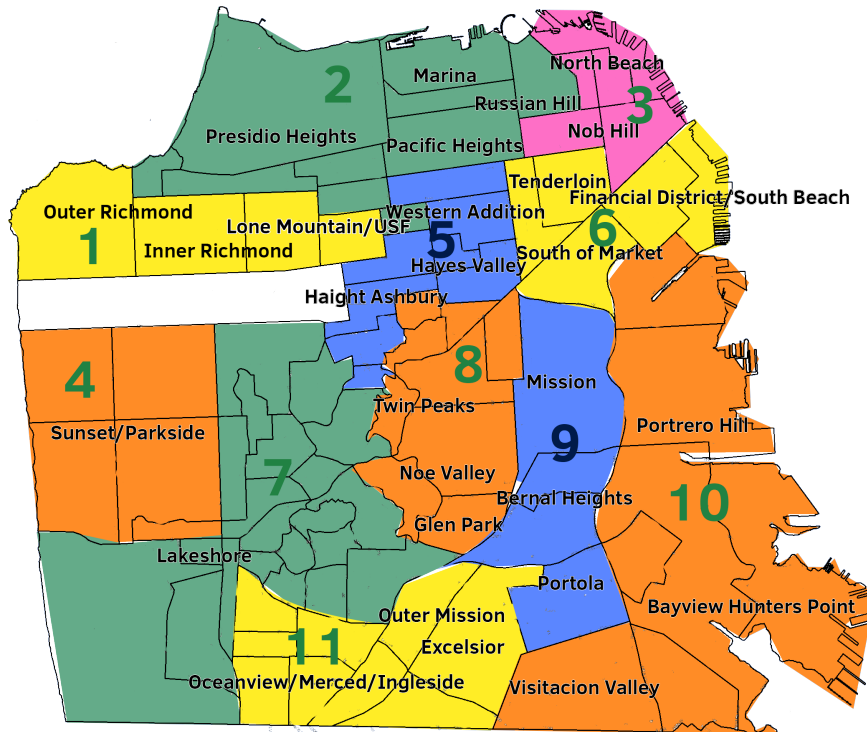
We downloaded data on eviction statistics from SF OpenData and counted each of the reasons for eviction with a program we designed and created using C++. For some of these expected counts, the data was less than five so we were unable to incorporate it in our testing. After separating the data into eleven districts based off location, we performed χ^2 Data Tests for Homogeneity using the valid counts.

2 Data Context

Not all neighborhoods of San Francisco are included in the provided data set because some neighborhoods do not contain any rented properties. Below are neighborhoods available to us in the data set, including the District groupings we assigned to them.

District	Neighborhoods
1	Outer Richmond, Inner Richmond, Lone Mountain/USF
2	Presidio Heights, Marina, Russian Hill, Pacific Heights
3	North Beach, Nob Hill
4	Sunset/Parkside
5	Haight Ashbury, Hayes Valley, Western Addition
6	Tenderloin, South of Market, Financial District/South Beach
7	Lakeshore
8	Noe Valley, Twin Peaks, Glen Park
9	Portola Heights, Bernal Heights, Mission
10	Visitacion Valley, Bayview Hunters Point, Portrero Hill
11	Oceanview/Merced/Ingleside, Outer Mission, Excelsior

Table 1: Available Neighbrohoods and Their Districts



Some neighborhoods and eviction reasons contained far too little data to produce anything meaningful. They are listed below.

Category	Removed
Reasons	Other, lead remediation, good samaritan
Neighborhoods	Presidio, Seacliff, Mission Bay, Golden Gate Park
Time period	All eviction notices before January 2005

Table 2: Available Neighbrohoods and Their Districts

Instead of running one χ^2 test on all of the eviction reasons, we decided to group the reasons into four categories.

Tenant Action	Non Payment, Breach of Contract, Nuisance, Illegal, Unapproved Subtenant, Late Payment
Landlord Action	Fail to Sign Renew, Owner Move-in, Capital Improvement, substantial rehab, roommate same unit
Development	Demolition, Ellis Act Withdrawal, Condo Conversion
Just Cause	Non Payment, Late Payment, Breach of Contract, Owner Move-in, Capital Improvement, Ellis Act Withdrawal, Nuisance, Illegal, Demolition

Table 3: Eviction Reason Categories

3 Inferential Procedures

3.1 District 1

Neighborhoods	Inner Richmond, Lone Mountain/USF, Outer Richmond
---------------	---

Reason	χ^2	df	p-Value
Tenant Action	15.1570	10	0.1264
Landlord Action	10.4669	6	0.1062
Development	0.9854	2	0.6110
Just Cause Removal	39.9610	16	0.0008

For District 1, there was homogeneity shown in all areas besides "Just Cause Removal". This may be because of Lone Mountain's low nonpay component of 0.4827 or Inner Richmond's high ownermovein component of 4.1747.

3.2 District 2

Neighborhoods | Presidio Heights, Marina, Russian Hill, Pacific Heights

Reason	χ^2	df	p-Value
Tenant Action	45.4120	15	0.0001
Landlord Action	11.8037	6	0.0665
Development	0.3398	6	0.9993
Just Cause Removal	127.1146	24	0.0000

For District 2, homogeneity was only shown in "Just Cause Removal" and "Tenant Action." Pacific Heights' high nonpayment component of 11.6593 likely impacts both of these sections.

3.3 District 3

Neighborhoods | North Beach, Nob Hill

Reason	χ^2	df	p-Value
Tenant Action	8.0298	5	0.1955
Landlord Action	2.2699	3	0.7037
Development	0.0000	1	1.0000
Just Cause Removal	19.6549	8	0.0117

For District 4, Homogeneity was shown in all categories but "Just Cause Removal." This may be because of North Beach's almost double nonpayment and ellisactwithdrawal compared to Nob Hill's components.

3.4 District 5

Neighborhoods | Haight Ashbury, Hayes Valley, Western Addition

Reason	χ^2	df	p-Value
Tenant Action	32.4910	10	0.0003
Landlord Action	10.6710	6	0.0991
Development	0.9271	4	0.9206
Just Cause Removal	111.3054	16	0.0000

Homogeneity is shown in all categories besides "Tenant Action" and "Just Cause Removal." In regards to "Tenant Action," Hayes Valley high nonpayment component of 6.8882 and Western Addition's high nuisance component of 6.3202

likely contribute to the lack of homogeneity, and in "Just Cause Removal," Western Addition's extremely low capital improvement component of 0.1674 does not correlate with the next lowest in Western Addison of 14.6997.

3.5 District 6

Neighborhoods | Tenderloin, South of Market, Financial District/South Beach

Reason	χ^2	df	p-Value
Tenant Action	46.9304	4	0.0000
Landlord Action	0.0000	-2	1.0000
Development	0.0000	-2	1.0000
Just Cause Removal	63.8182	4	0.0000

Homogeneity is shown in the categories of "Tenant Action" and "Just Cause Removal." This may be because of Financial District/South Beach's high nuisance component of 8.1637 and its low nonpayment component of 6.3031.

3.6 District 8

Neighborhoods | Noe Valley, Glen Park, Twin Peaks

Reason	χ^2	df	p-Value
Tenant Action	10.5992	2	0.0050
Landlord Action	1.1193	0	1.0000
Development	1.1314	0	1.0000
Just Cause Removal	37.6297	6	0.0000

Homogeneity is shown in the categories of "Tenant Action" and "Just Cause Removal." The reasons for this in "Tenant Action" may be Glen Park's low nuisance component of 0.2598, and in "Just Cause Removal" it may be because of Glen Park's high breach value of 5.9563 and low ellisactwithdrawl of 0.0003.

3.7 District 9

Neighborhoods	Portola, Bernal Heights, Mission		
Reason	χ^2	df	p-Values
Tenant Action	12.4686	6	0.0523
Landlord Action	1.3052	2	0.5207
Development	19.1170	0	1.0000
Just Cause Removal	133.8546	12	0.0000

Homogeneity is shown in all areas besides "Just Cause Removal." This is most likely because Portola has an extremely high χ^2 value of 20.7812. The next highest is only 4.3994 in Mission.

3.8 District 10

Neighborhoods	Visitacion Valley, Bayview Hunters Point, Portrero Hill			
Reason	χ^2	df	p-Value	
Tenant Action	3.8906	4	0.4210	
Landlord Action	1.5766	1	0.3173	
Development	0.0000	-1	1.0000	
Just Cause Removal	8.6997	5	0.1640	

District 10 has evidence of homogeneity. Each of the reason's p-Value is above 0.05.

3.9 District 11

Neighborhoods	Oceanview/Merced/Ingleside, Outer Mission, Excelsior			
Reason	χ^2	df	p-Value	
Tenant Action	23.1291	8	0.0032	
Landlord Action	2.2277	2	0.3283	
Development	3.8359	2	0.1469	
Just Cause Removal	30.6304	14	0.0062	

Only "Tenant Action" and "Just Cause Removal" do not show evidence of homogeneity in District 11. For "Tenant Action," this may be because of Outer Mission's high breach component of 5.7766, and in "Just Cause Removal" it could be because of the Oceanview/Merced/Ingleside's nuisance component of 0.0220.

3.10 District 3 vs. District 6

Neighborhoods	North Beach, Nob Hill, Tenderloin, South of Market, Financial District/South Beach
---------------	---

Reason	χ^2	df	p-Value
Tenant Action	50.260	8	0.0000
Landlord Action	na	na	na
Development	na	na	na
Just Cause Removal	341.925	12	0.0000

The Landlord Action and Development categories yielded no results because we were unable to perform the test, due to the expected counts condition. We were successful in running a test that produced clear results for the Tenant Action and Just Cause Removal categories, however. The greatest component of the Tenant Action category was the Non Payment reason for District 6, which we think is likely due to rising rents in the Tenderloin neighborhood.

We saw clear non-homogeneity in the Just Cause Removal test, where the p-Value approached zero. Here, the largest components for both District 3 and District 6 were due to the Ellis Act Withdrawal eviction reason, with components of 144.9395 and 81.4898, respectively. This indicates that an immense amount of landlords are removing their property from the rental market for other uses.

3.11 District 1 vs. District 4

Neighborhoods	Outer Richmond, Inner Richmond, Lone Mountain/USF, Sunset/Parkside
---------------	---

Reason	χ^2	df	p-Value
Tenant Action	13.3248	12	0.3459
Landlord Action	8.6267	6	0.1957
Development	78.7158	3	0.0000
Just Cause Removal	106.561	24	0.0000

Between these two districts, we see that the evictions due to renter and landlord action occur at relatively the same rate relative to the size of each district. It may be interesting to note that for the Landlord Action category, much of the statistic was made of one component: the Capital Improvement reason for District 4 at 5.2804, which suggests that in the Sunset/Parkside neighborhood landlords are more likely to make significant improvements on their apartments,

which temporarily evict tenants from their rooms.

For reasons which we grouped under Development, these produced components with large values. The greatest contributor to the Development reason was from the Demolition reason from District 4 at 39.8452. This altogether may not be too unexpected, because, from anecdotal experience, there are many buildings which have fallen into disarray or may not be within building code in the first place.

When we study the Just Cause Removal test, we find more clear differences between these two districts. Both of the largest components of the test statistic resulted from District 4, in the Demolition and Ellis Act Withdrawal reasons, at 47.078 and 15.015 respectively.

3.12 District 3 vs. District 5

Neighborhoods	North Beach, Nob Hill, Haight Ashbury, Hayes Valley, Western Addition
---------------	--

Reason	χ^2	df	p-Value
Tenant Action	9.7614	16	0.8788
Landlord Action	0.9936	8	0.9983
Development	0.5791	4	0.9654
Just Cause Removal	117.5088	28	0.0000

Upon examining the p-Values for each of these tests, it is immediately apparent that these two districts are quite homogenous to each other, at p-Values above 0.85 for Tenant Action, Landlord Action, and Development. However, we see a departure from homogeneity in the Just Cause Removal test, which we could see as a unification of all three of these tests.

As we parse the components of the Just Cause Removal test, we find that three key components make up the bulk of the test statistic. For District 3, it is the Ellis Act Withdrawal reason, at a value of 39.0179, and for District 5, it is the Owner Move In and Capital Improvement reasons at 30.4084 and 21.7573 respectively that contribute the most. Perhaps District 5 is an attractive place for landlords to find to live in their apartment property, and it is possible that in District 3, landlords are liquidating their property in anticipation of higher profits outside of the rental market.

3.13 District 5 vs. District 6

For Landlord Action and Development, we could not run the tests because we had expected counts less than 5.

Neighborhoods	Haight Ashbury, Hayes Valley, Western Addition Tenderloin, South of Market
---------------	---

Reason	χ^2	df	p-Value
Tenant Action	45.3672	10	0.0000
Landlord Action	na	na	na
Development	na	na	na
Just Cause Removal	407.8489	20	0.0000

In the Tenant Action test, we find a clear departure from homogeneity between District 5 and District 6, suggesting a difference in the type of renters between these two areas. Indeed, we can see that District 6 contributes much to the final test statistic, with the two reasons Non Payment and Nuisance at 16.0512 and 21.4209 respectively, suggesting that it may not be advantageous for landlords to hold a property in this area relative to District 5.

In the Just Cause Removal test, the largest departures from homogeneity are revealed in the components Owner Move In of District 5, and Nuisance and Owner Move In of District 6, at 133.2342, 68.6161, and 61.3065.

4 Conclusion

Through our multiple studies, we found that most districts were homogenous except for when we tested for the Tenant Action and Just Cause Removal categories, which suggests that there were a higher amount of renter evictions due to the fault of the renter, and for reasons that the City of San Francisco defines as "just." In our inter-district analyses, we found that there were significant differences between the relative frequencies of each eviction reason, most prominently the Ellis Act Withdrawal was invoked in some neighborhoods far more than others. We can take the information we have learned and apply it in context with the news articles that we read about the housing crisis in San Francisco.

A Source code

```
1  #include <EvictionNotice.hpp>
2  #include <algorithm>
3  #include <json/json.h>
4  #include <iostream>
5  #include <fstream>
6  #include <streambuf>
7  #include <map>
8  #include <vector>
9  #include <StatFunctions.hpp>
10
11 std::string getJSON(const std::string& filename)
12 {
13     std::ifstream file(filename);
14
15     std::string str;
16     file.seekg(0, std::ios::end);
17     str.reserve(file.tellg());
18     file.seekg(0, std::ios::beg);
19
20     str.assign((std::istreambuf_iterator<char>(file)),
21              std::istreambuf_iterator<char>());
22
23     return str;
24 }
25
26
27 std::vector<NeighborhoodCounts> selectColumns(
28     const std::vector<Indices>& columns
29     , std::vector<NeighborhoodCounts> counts)
30 {
31     std::for_each (counts.begin(), counts.end(),
32                   [&] (NeighborhoodCounts& row) {
33         for (int i = 0; i < row.counts.size(); i++)
34         {
35             bool selected = false;
36             for (int j = 0; j < columns.size(); j++)
37             {
38                 if (i == columns[j])
39                 {
40                     selected = true;
41                 }
42             }
43             if (!selected)
44                 row.counts[i] = 0;
45         }
46     }
47 );
```

```

48     return counts;
49 }
50
51 std::vector<std::vector<Json::Value>> generateRowNotices(const
    Json::Value& data)
52 {
53     std::vector<std::vector<Json::Value>> rows;
54     for (unsigned int i = 0; i < data.size(); i++)
55     {
56         rows.push_back(std::vector<Json::Value>());
57         for (auto it = data[i].begin(); it != data[i].end(); ++it)
58         {
59             rows[i].push_back(*it);
60         }
61     }
62     return rows;
63 }
64
65 bool contains(const std::vector<std::string>& vec, const std::string str)
66 {
67     for (auto &e : vec)
68         if (e.compare(str) == 0) return true;
69
70     return false;
71 }
72
73 std::map<std::string, std::vector<EvictionNotice>> convertRawToBools(
74     std::vector<std::vector<Json::Value>> rows)
75 {
76     enum Columns
77     {
78         ADDRESS = 9, CITY, STATE, ZIP,
79         DATE,
80         NON_PAYMENT,
81         BREACH,
82         NUISANCE,
83         ILLEGAL,
84         FAIL_SIGN_RENEW,
85         ACCESS_DENIAL,
86         UNAPPROVED_SUBTENANT,
87         OWNER_MOVE_IN,
88         DEMOLITION,
89         CAPITAL_IMPROVEMENT,
90         SUBSTANTIAL_REHAB,
91         ELLIS_ACT_WITHDRAWAL,
92         CONDO_CONVERSION,
93         ROOMMATE_SAME_UNIT,
94         OTHER,
95         LATE_PAY,
96         LEAD_REMEDIATION,

```

```

97         DEVELOPMENT,
98         GOOD_SAMARITAN,
99         CONSTRAINTS,
100        CONSTRAINTS_DATE, SUPERVISOR,
101        NEIGHBORHOOD,
102        COORDINATES
103    };
104
105    std::map<std::string, std::vector<EvictionNotice>>
        neighborhoodNotices;
106
107    //parse out all entries not wanted and columns and convert to bool
108    for (const std::vector<Json::Value> &columns : rows)
109    {
110        if (columns[DATE].asString().compare("2005-01-01T00:00:00") < 0)
111            //only entries recent 10 years
112            continue;
113        std::string neighName =
114            columns[Columns::NEIGHBORHOOD].asString();
115
116        EvictionNotice notice;
117
118        for (int i = 0; i < notice.reasons.size(); i++)
119        {
120            notice.reasons[i] = columns[i + 14].asBool();
121        }
122        neighborhoodNotices[columns[Columns::NEIGHBORHOOD].asString()].push_back(notice);
123    }
124
125    return neighborhoodNotices;
126 }
127
128 Indices getIndex(const std::string& reason)
129 {
130     if (reason.compare("NON_PAYMENT") == 0) return NON_PAYMENT;
131     if (reason.compare("BREACH") == 0) return BREACH;
132     if (reason.compare("NUISANCE") == 0) return NUISANCE;
133     if (reason.compare("ILLEGAL") == 0) return ILLEGAL;
134     if (reason.compare("FAIL_SIGN_RENEW") == 0) return FAIL_SIGN_RENEW;
135     if (reason.compare("ACCESS_DENIAL") == 0) return ACCESS_DENIAL;
136     if (reason.compare("UNAPPROVED_SUBTENANT") == 0) return
137         UNAPPROVED_SUBTENANT;
138     if (reason.compare("OWNER_MOVE_IN") == 0) return OWNER_MOVE_IN;
139     if (reason.compare("DEMOLITION") == 0) return DEMOLITION;
140     if (reason.compare("CAPITAL_IMPROVEMENT") == 0) return
141         CAPITAL_IMPROVEMENT;
142     if (reason.compare("SUBSTANTIAL_REHAB") == 0) return
143         SUBSTANTIAL_REHAB;
144     if (reason.compare("ELLIS_ACT_WITHDRAWAL") == 0) return
145         ELLIS_ACT_WITHDRAWAL;

```

```

140     if (reason.compare("CONDO_CONVERSION") == 0) return CONDO_CONVERSION;
141     if (reason.compare("ROOMMATE_SAME_UNIT") == 0) return
        ROOMMATE_SAME_UNIT;
142     if (reason.compare("OTHER") == 0) return OTHER;
143     if (reason.compare("LATE_PAY") == 0) return LATE_PAY;
144     if (reason.compare("LEAD_REMEDIATION") == 0) return LEAD_REMEDIATION;
145     if (reason.compare("DEVELOPMENT") == 0) return DEVELOPMENT;
146     if (reason.compare("GOOD_SAMARITAN") == 0) return GOOD_SAMARITAN;
147 }
148
149 bool neighborhoodMode = false;
150 bool columnMode = false;
151 bool districtMode = false;
152
153 void parseArgs(std::string& filename
154               , std::vector<std::string>& neighs
155               , std::vector<Indices>& cols
156               , std::vector<std::string>& dists
157               , int argc
158               , const char* argv[])
159 {
160     const std::string NEIGH_SELECTOR = "-n";
161     const std::string COL_SELECTOR = "-c";
162     const std::string DISTR_SELECTOR = "-d";
163
164     if (argc > 1)
165     {
166         filename = argv[1];
167         for (int i = 2; i < argc; i++)
168         {
169             std::string arg = argv[i];
170
171             if (arg.compare(NEIGH_SELECTOR) == 0)
172             {
173                 neighborhoodMode = true;
174                 columnMode = false;
175                 districtMode = false;
176             }
177             else if (arg.compare(COL_SELECTOR) == 0)
178             {
179                 neighborhoodMode = false;
180                 columnMode = true;
181                 districtMode = false;
182             }
183             else if (arg.compare(DISTR_SELECTOR) == 0)
184             {
185                 neighborhoodMode = false;
186                 columnMode = false;
187                 districtMode = true;
188             }

```

```

189
190     if (neighborhoodMode)
191     {
192         if (arg.compare(NEIGH_SELECTOR) != 0)
193             neighs.push_back(arg);
194     }
195     else if (columnMode)
196     {
197         if (arg.compare(COL_SELECTOR) != 0)
198             cols.push_back(getIndex(arg));
199     }
200     else if (districtMode)
201     {
202         if (arg.compare(DISTR_SELECTOR) != 0)
203             dists.push_back(arg);
204     }
205 }
206 }
207 }
208
209 std::map<std::string, NeighborhoodCounts> createCounts(
210     std::map<std::string, std::vector<EvictionNotice>> notices
211     , std::vector<std::string> selectedNeighborhoods)
212 {
213     std::map<std::string, NeighborhoodCounts> neighborhoodsCounts;
214
215     auto it = notices.begin();
216     auto itend = notices.end();
217     for (; it != itend; ++it)
218     {
219         std::string name = it->first;
220
221         if (!contains(selectedNeighborhoods, name)) continue;
222
223         neighborhoodsCounts[name].neighborhoodName = name; //for
224         converting map to vec later
225         for (auto jt = it->second.begin(); jt != it->second.end(); ++jt)
226         {
227             for (int i = 0; i < jt->reasons.size(); i++)
228             {
229                 if (jt->reasons[i])
230                 {
231                     neighborhoodsCounts[name].counts[i]++;
232                 }
233             }
234         }
235     }
236
237     return neighborhoodsCounts;

```

```

238 }
239
240 std::vector<NeighborhoodCounts> mapToVec(
241     const std::map<std::string, NeighborhoodCounts>& other)
242 {
243     std::vector<NeighborhoodCounts> finalCounts;
244     for (auto &e : other)
245     {
246         finalCounts.push_back(e.second);
247     }
248     return finalCounts;
249 }
250
251 int main(int argc, const char* argv[])
252 {
253     std::string evictionFilename = "eviction-notices.json";
254     std::vector<std::string> selectedNeighborhoods;
255     std::vector<Indices> selectedColumns;
256     std::vector<std::string> selectedDistricts;
257
258     parseArgs(evictionFilename, selectedNeighborhoods, selectedColumns,
259               selectedDistricts, argc, argv);
260
261     std::string evictions = getJSON(evictionFilename);
262
263     Json::Value root;
264     Json::Reader reader;
265
266     reader.parse(evictions, root);
267
268     Json::Value data = root.get("data", "error");
269
270     std::vector<std::vector<Json::Value>> rawNotices =
271         generateRawNotices(data);
272     std::map<std::string, std::vector<EvictionNotice>> parsedNotices =
273         convertRawToBools(rawNotices);
274
275     //TODO iterate here and generate all of our chis
276     std::map<std::string, NeighborhoodCounts> mappedCounts =
277         createCounts(parsedNotices, selectedNeighborhoods);
278     std::vector<NeighborhoodCounts> counts = mapToVec(mappedCounts);
279
280     auto selectedColumnCounts = selectColumns(selectedColumns, counts);
281
282     int df = 0;
283     double chi = chiSquareStatistic(selectedColumnCounts, df,
284                                     selectedDistricts);
285     double pVal = chiAreaRight(chi, df);
286     std::cout << "chi," << chi
287               << ",df," << df

```

```
283         << ",pVal," << pVal
284         << std::endl;
285     }
```
