San Francisco Restaurants

Process Book

Mathieu Clément <mclement2@usfca.edu>
Byron Han <zhan12@usfca.edu>

Project repository: https://github.com/tiktaktok/cs560-restaurants
Project website: https://tiktaktok.github.io/cs560-restaurants

CS 360/560 Data Visualization, University of San Francisco
May 16th, 2018

Overview and Motivation

If you have ever eaten out in San Francisco, chances are that you have seen a document from the San Francisco Health Department displayed prominently in the restaurant, reporting the results of their announced and unannounced inspections. City officials will assess whether there are violations regarding public health and safety, in order to reduce food borne illnesses.
Inspections receive a score from 0-100 where 100 means no violations have been observed.

While the health score and violations are interesting in themselves, the data is relatively dry. Not only do we want to visualize that data (we can easily plot it on a map), but it might be interesting to fuse this data with information fetched from Yelp, such as reviews, category of restaurant, prices, etc.

We have found a relevant dataset containing inspections including health scores from the SF Health Department on Kaggle (detailed below) and successfully used the Yelp API through Postman, which led us to believe that this project was feasible and that the information is available. We have also chosen this project because we were not able to find any insights regarding the subject matter, and we are interested in the subject. We hope to find correlations in the data, and allow the public to make smart decisions when picking up a place to eat.

Currently Yelp does not offer to filter results based on the health score. And we believe this is a shame.

Food safety and food ratings are two major concerns when people choose to eat everywhere. Luckily, we live in San Francisco. Yelp api and sf.gov provide very detailed and reliable information about safety and ratings.

Instead of analyze all categories, we have picked up the most interesting and large categories that people are interested and visit most often. Then we decide to look into the dataset, see if there are interesting patterns and see what the correlations are and see if there is any possible distribution and patterns or regression models to describe any of the score or ratings in particular in any category of interests.

Related Work

Related works which we can learn from:
*:interaction included
*bar chart: previous assignment
*bi/tri/multivariate scatter plot: previous assignments
*line chart: previous assignment
tree: for tree we can do regression tree or classification tree so that we can predict values
tree graph: https://cran.r-project.org/web/packages/data.tree/vignettes/data.tree.html
boxplot: https://www.statmethods.net/graphs/boxplot.html
*Geo map: http://bl.ocks.org/micahstubbs/8e15870eb432a21f0bc4d3d527b2d14f
Google map api: For pin point the restaurants
Test html and css: https://codepen.io/irinakramer/pen/jcLlp
12 different color for scatter plot:
http://colorbrewer2.org/#type=qualitative&scheme=Paired&n=12
Paper on inspection score https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3323064/

Questions

We would like to be able to answer the following questions:

- Are Korean restaurants cleaner than their Chinese counterpart(or any category in general)?
- Are Korean restaurants have better ratings than their Chinese counterpart(or any category in general)?
- Is there a correlation between: health score and ratings?
- What was the health score and which violations were observed at restaurant X?
- What is the distribution of health scores?
- What is the average for each neighborhood in San Francisco?
- Has the health score improved since the last inspection of a given restaurant X?
- What are the restaurants with High Risk, Moderate Risk, Low Risk?

We would also like users to be able to explore the restaurant in the city using a map, markers and tooltips/detail page.

Regular Map

Design Objectives:

1. Either D3 package or Google API or other tools to possibly enable user interaction
2. Able to filter based on risk level, we define certain inspection score for different level
3. Maybe color filtering added, mouse over interaction
4. Tooltips to display certain restaurant info by the side
5. Possible link to timeline of this restaurant's inspection score

User Objectives:

1. Able to see restaurants risk category
2. Show user more info in one page, like what violation

Timeline Graph

Design Objectives:

1. Inspection score vs. Time
2. Mouse over interaction

User Objective:

1. User can see how this restaurant is doing over time

Geo Map maybe

Design Objectives:

1. Geomap usually will affect visualization based on the area of the neighborhood
2. We normalize each neighborhood and assign average score as a color to the neighborhood

User Objectives:

1. Have user visualize the correlation of cleanness vs area

Trivariate/Multivariate scatter plot:

Design Objectives:

1. Depends on how many data we have, 4 variate if data from Yelp is large enough
2. Rating vs. Inspection Score vs. Restaurant Category
3. Primary choice: Rating vs Inspection score, category as color as menu to choose from.

User Objectives:

1. User can select which one to have on the graph and which one to be legend
2. Find out if there are any variable has clusters
3. See if any type of restaurant has cluster in cleanness, ratings etc.
4. This is the initial attempt to visualize difference and correlation

Box Plot

Design Objectives:

1. Inspection ratings and ratings

2. Use R code to analyze and find out percentile and mean of different category
3. Have a better understanding of the distribution and spread of the score and rating of different category

User Objectives:

1. Have user visual the correlation between price and cleanness
2. Better statistical understanding about the mean and variance and percentile

Inspection score distribution bar chart and fit a curve onto it

Design Objectives:

1. Based on the area and zip code we can see the distribution of inspection score
2. Fit a curve onto it to see the best distribution modeled

User Objective:

1. See if outlier is contributing to the overall score or not
2. Explore to see the distribution type

Overall, initially we wanted to look at the correlation between different categories and types and neighborhood etc. But because this is data visualization course, not a machine learning course, so we decided to focus on the visualization part instead of the prediction part.

The regression model would produce a model and do prediction, but there is not so much left to do for visualization.

Data

Inspections and violations

We had the idea of doing this project by browsing Kaggle, when we stumbled upon the following dataset: https://www.kaggle.com/datasf/sf-restaurant-inspection-scores

We later discovered that this data was several years old, and that the San Francisco Health Department updates it every week and publishes it here: https://data.sfgov.org/Health-and-Social-Services/Restaurant-Scores-LIVES-Standard/pyih-qa8i

The data is formatted using a simplified LIVES schema: https://docs.google.com/document/d/1eeO5T_It8QHGHMjpj6M9071y5OFOXcADFI4F7AIZ2iE/edit

Basically, it's a CSV file:

business_id	business_name	business_address	business_city	business_state	business_postal_code
19	Nrgize Lifestyle Cafe	1200 Van Ness Ave, 3rd Flo	San Francisco	CA	94109
19	Nrgize Lifestyle Cafe	1200 Van Ness Ave, 3rd Flo	San Francisco	CA	94109
19	Nrgize Lifestyle Cafe	1200 Van Ness Ave, 3rd Flo	San Francisco	CA	94109
19	Nrgize Lifestyle Cafe	1200 Van Ness Ave, 3rd Flo	San Francisco	CA	94109
19	Nrgize Lifestyle Cafe	1200 Van Ness Ave, 3rd Flo	San Francisco	CA	94109
24	OMNI S.F. Hotel - 2nd Floor Pantry	500 California St, 2nd Floo	San Francisco	CA	94104
24	OMNI S.F. Hotel - 2nd Floor Pantry	500 California St, 2nd Floo	San Francisco	CA	94104
24	OMNI S.F. Hotel - 2nd Floor Pantry	500 California St, 2nd Floo	San Francisco	CA	94104
24	OMNI S.F. Hotel - 2nd Floor Pantry	500 California St, 2nd Floo	San Francisco	CA	94104
31	Norman's Ice Cream and Freezes	2801 Leavenworth St	San Francisco	CA	94133
31	Norman's Ice Cream and Freezes	2801 Leavenworth St	San Francisco	CA	94133
31	Norman's Ice Cream and Freezes	2801 Leavenworth St	San Francisco	CA	94133
45	CHARLIE'S DELI CAFE	3202 FOLSOM St	San Francisco	CA	94110

inspection_id	inspection_date	inspection_score	inspection_type	violation_id	violation_description	risk_category
19_20171211	12/11/2017 12:00:00 AM	94	Routine - Unscheduled	19_20171211_103144	Unapproved or unmaintained equipment	Low Risk
19_20171211	12/11/2017 12:00:00 AM	94	Routine - Unscheduled	19_20171211_103116	Inadequate food safety knowledge or lac	Moderate Risk
19_20160513	05/13/2016 12:00:00 AM	94	Routine - Unscheduled	19_20160513_103144	Unapproved or unmaintained equipment	Low Risk
19_20160513	05/13/2016 12:00:00 AM	94	Routine - Unscheduled	19_20160513_103157	Food safety certificate or food handler ca	Low Risk
19_20160513	05/13/2016 12:00:00 AM	94	Routine - Unscheduled	19_20160513_103154	Unclean or degraded floors walls or ceili	Low Risk
24_20160311	03/11/2016 12:00:00 AM	96	Routine - Unscheduled	24_20160311_103154	Unclean or degraded floors walls or ceili	Low Risk
24_20160311	03/11/2016 12:00:00 AM	96	Routine - Unscheduled	24_20160311_103154	Unclean or degraded floors walls or ceili	Low Risk
24_20161005	10/05/2016 12:00:00 AM	98	Routine - Unscheduled			
24_20171101	11/01/2017 12:00:00 AM	98	Routine - Unscheduled	24_20171101_103139	Improper food storage	Low Risk
31_20160901	09/01/2016 12:00:00 AM		Reinspection/Followup			
31_20161020	10/20/2016 12:00:00 AM		Non-inspection site visit			
31_20151204	12/04/2015 12:00:00 AM	98	Routine - Unscheduled	31_20151204_103157	Food safety certificate or food handler ca	Low Risk
45_20160614	06/14/2016 12:00:00 AM	84	Routine - Unscheduled	45_20160614_103154	Unclean or degraded floors walls or ceili	Low Risk
45_20160614	06/14/2016 12:00:00 AM	84	Routine - Unscheduled	45_20160614_103144	Unapproved or unmaintained equipment	Low Risk

In fact, the up-to-date data can be viewed online at the following location: https://data.sfgov.org/Health-and-Social-Services/Restaurant-Scores-LIVES-Standard/pyih-qa8i/data

You can see a lot of duplicates. That's because the file basically contain a list of violations, that belong to inspections, that belong to restaurants. The rest of the data, that stays identical, is duplicated, such as the name of the restaurant, and the inspection date.

In other words, for each restaurant there will be multiple inspections.
Each inspection is a compilation of violations.

We first wrote a class ScoresReader that parses the CSV file in an object-oriented fashion:

```python
class Restaurant:
    def __init__(self, rest_id, name, street, zip_code, lat, lon):
        self.id = rest_id
        self.name = name
        self.street = street
        self.zip_code = zip_code
        self.lat = lat
        self.lon = lon
        self._inspections = []

    @property
    def inspections(self):
        return sorted(self._inspections)

    @property
    def score(self):
        return self.inspections[0].score # most recent inspection score

    def add_inspection(self, inspection):
        self._inspections.append(inspection)

    def __repr__(self):
        return 'Restaurant(%s, %s, San Francisco, CA %s, id=%s, inspections=%s' % \
        (self.name, self.street, self.zip_code, self.id, self.inspections)


class Inspection:
    def __init__(self, date, score):
        self.date = date
        self.score = score
        self._violations = []

    @property
    def violations(self):
        return sorted(self._violations)

    def add_violation(self, violation):
        self._violations.append(violation)

    def __repr__(self):
        return 'Inspection(date=%s, score=%s, violations=%s' % (self.date, self.score, self.violations)

    def __eq__(self, other):
        return self.date == other.date

    def __lt__(self, other):
        # most recent first
        return other.date < self.date


class Violation:
    def __init__(self, description, risk):
        self.description = description
        self.risk = risk

    def __repr__(self):
        return '(%s) %s' % (self.risk, self.description)

    def __eq__(self, other):
        return self.risk == other.risk and self.description == other.description

    def __lt__(self, other):
        if self.risk == other.risk:
            return self.description < other.description
        else:
            return other.risk_value < self.risk_value
        return self.description < other.description
```

Rows that have the following issues are excluded, a.k.a. **cleaning**:

- Missing inspection type or Scheduled, New entry, Transfer of ownership, etc. because they don't correspond to inspections.
- Inspection score is missing
- ZIP code is missing
- ZIP code is incorrect ("CA" is not a ZIP code…)

This work is done by the ScoresReader in scores.py.

Yelp

Once we have established the list of restaurants with valid inspections, we can fetch the corresponding information on Yelp.

We use the Business Search REST API endpoint to query the Yelp database:
https://www.yelp.com/developers/documentation/v3/business_search

There is a 5,000 requests/day limit, so we took the following measures to not waste our allowance:

1. Test/Debug only a few restaurants at a time
2. Make sure our code was well-tested before doing bulk requests
3. The API results are saved to a JSON file on disk, and will be later read by another program.
4. The Fetching part of the process is as small and simple as possible to avoid bugs and exceptions.
5. A piece of code lets us select (kind of like "brushing" in data visualization) only for example the 2,500th to 2,999th restaurant of our dataset.

A token (API secret) is required so that Yelp can make sure nobody does more than 5,000 queries per account.

To match an entry from the CSV file to the right web page on Yelp, we search the Yelp database using:

the business name
the street address (street and address)
sorting by distance

We want Yelp to return the restaurant that is closest to the address provided and matching the name provided. The distance criterion is important as certain chains such as Starbucks have multiple stores in a small area, sometimes right across the street.

When no result is found, the restaurant is not taken into consideration. This sometimes happens as not every food facility is recorded on Yelp as they might not be open to the public.

When Yelp only finds a restaurant that is too far away from the provided location, we exclude that restaurant. The JSON response from Yelp contains the distance in meters.

Finally, we will only consider that the entry on Yelp and in our dataset match if one of those conditions is met:

1. The Levenshtein distance (case-insensitive) between the business name from both sources is less than 10 and the geographical distance is less than 20 meters.
2. The street address is the same and the Levenshtein distance of the name is less than 5.

Unfortunately the SF Health Department does not use standardized addresses (as per the United States Postal Service guidelines).
For example,
2130 Fulton Street Building 50 Room 4 C, San Francisco California 94117
becomes
2130 Fulton St Bdg 50 Rm 4C, San Francisco, CA 94117

We apply the following transformations:

1. Leading zeros are removed from house numbers (033 => 33)
2. Dots are removed (St. => St)

3. Abbreviations are expanded as following:

```
'ave': 'avenue',
'ct': 'court',
'st': 'street',
'blvd': 'boulevard',
'dr': 'drive',
'wy': 'way',
'ln': 'lane',
'rd': 'road',
'terr': 'terrace',
'pl': 'place',
'pkwy': 'parkway',
'hwy': 'highway',
'cl': 'close'
```

Data Fusion

The Yelp data (saved to yelp.json) needs to be combined with the restaurant scores and inspections (from a CSV file) to a JSON file (combined.json).

We basically start with the JSON from Yelp, that looks like this:

```
{
  "rating": 4,
  "price": "$",
  "phone": "+14152520800",
  "id": "E8RJkjfdcwgtyoPMjQ_Olg",
  "alias": "four-barrel-coffee-san-francisco",
  "is_closed": false,
  "categories": [
    {
      "alias": "coffee",
      "title": "Coffee & Tea"
    }
  ],
  "review_count": 1738,
  "name": "Four Barrel Coffee",
  "url": "https://www.yelp.com/biz/four-barrel-coffee-san-francisco",
  "coordinates": {
    "latitude": 37.7670169511878,
    "longitude": -122.42184275
  },
  "image_url": "http://s3-media2.fl.yelpcdn.com/bphoto/MmgtASP3l_t4tPCL1iAsCg/
  "location": {
    "city": "San Francisco",
    "country": "US",
    "address2": "",
    "address3": "",
    "state": "CA",
    "address1": "375 Valencia St",
    "zip_code": "94103"
  },
  "distance": 1604.23,
  "transactions": ["pickup", "delivery"]
},
```


And we add our search criteria, so we can tell if our matching algorithm made a mistake, plus we convert our Python object containing the list of inspections, containing the list of violations into a JSON representation.

The “combined.json” file is actually a JSON dictionary, where the value is what we’ve mentioned, and the key is the ID of the restaurant as per the inspection dataset.

The neighborhood is also added to that file after we go to the process mentioned in the next section.

```
{
  "91157": {
    "id": "78mfppJEnc-2NBrxtS9u7A",
    "alias": "woodlands-market-san-francisco-4",
    "name": "Woodlands Market",
    "image_url": "https:\/\/s3-media3.fl.yelpcdn.
    "is_closed": false,
    "url": "https:\/\/www.yelp.com\/biz\/woodland
    "review_count": 66,
    "categories": [
      {
        "alias": "grocery",
        "title": "Grocery"
      },
      {
        "alias": "beer_and_wine",
        "title": "Beer, Wine & Spirits"
      },
      {
        "alias": "bakeries",
        "title": "Bakeries"
      }
    ],
    "rating": 3.5,
    "coordinates": {
      "latitude": 37.789041,
      "longitude": -122.392012
    },
    "transactions": [

    ],
    "price": "$$",
    "location": {
      "address1": "203 Folsom St",
      "address2": "",
      "address3": null,
      "city": "San Francisco",
      "zip_code": "94105",
      "country": "US",
      "state": "CA",
      "display_address": [
        "203 Folsom St",
        "San Francisco, CA 94105"
      ]
    },
    "phone": "+14153564000",
    "display_phone": "(415) 356-4000",
    "distance": 2.6287343441307,
    "search": {
      "name": "Woodlands Market",
      "street": "203 Folsom St.",
      "zip_code": 94105
    },
```

```
    "inspections": [
      {
        "date": "2017-10-26",
        "score": 89,
        "violations": [
          {
            "description": "Unclean or unsanitary food contact surfaces",
            "risk": "High Risk"
          },
          {
            "description": "Foods not protected from contamination",
            "risk": "Moderate Risk"
          }
        ]
      }
    ],
    "neighborhood": "Financial District\/South Beach"
  }
}
```


The “id” that you can see in the value part is the ID as reported by Yelp, whereas the ID that you see first (“91157”) is the ID in the health score dataset.

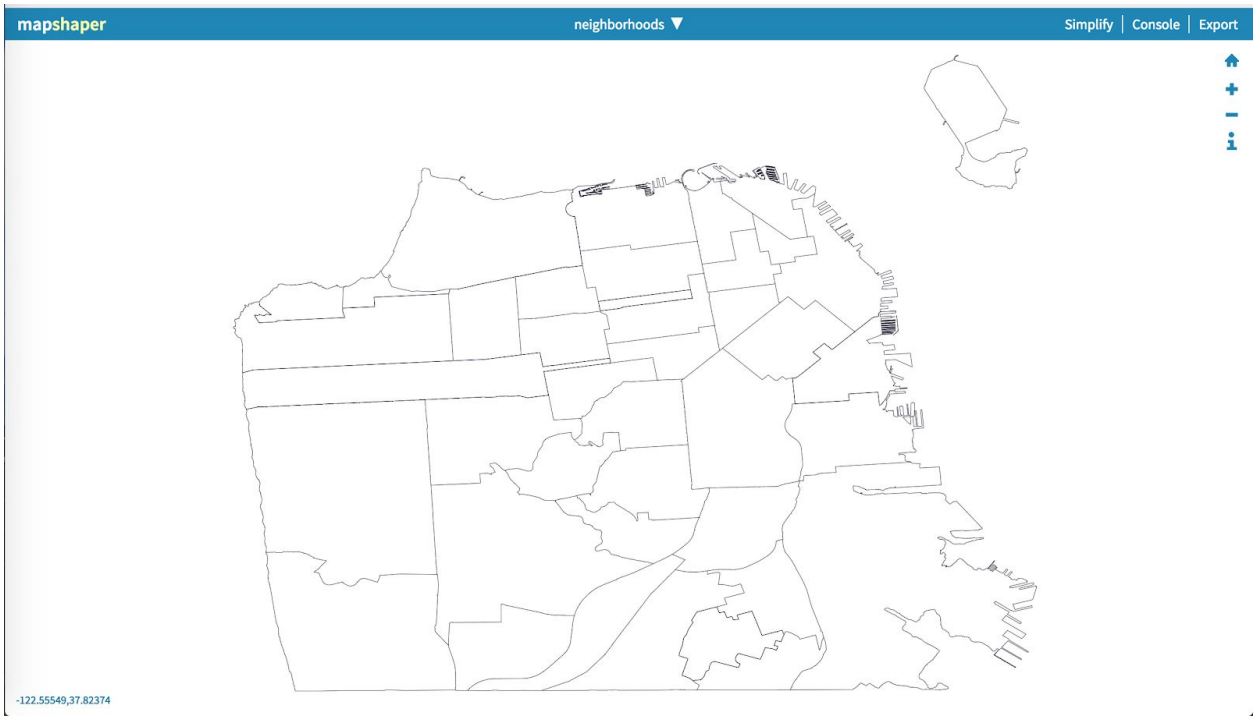
Neighborhood map

We want to create a map of San Francisco that shows the districts of the city, such as South of Market, the Presidio, or the Marina.

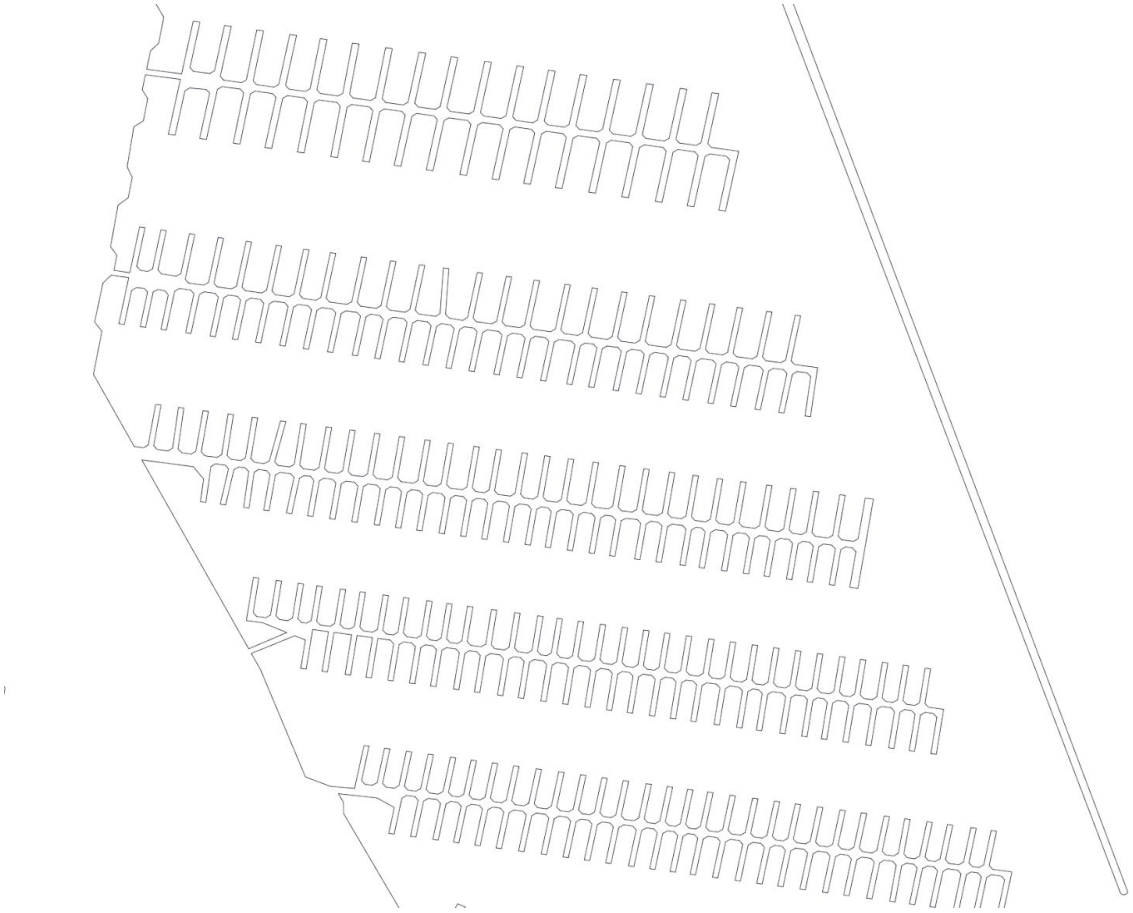
The dataset that we found on Kaggle, while not up-to-date with the rest of the inspection data does provide shapefiles (.sbp, .shx, etc.) that we loaded into MapShaper, simplified the outlines, and converted to GeoJSON so we could use it in D3.

The shapefile come with a description of the polygons (i.e. the districts), including a “nhood” (neighborhood) attribute that gives us the name of the district.

Here is the data loaded into MapShaper:

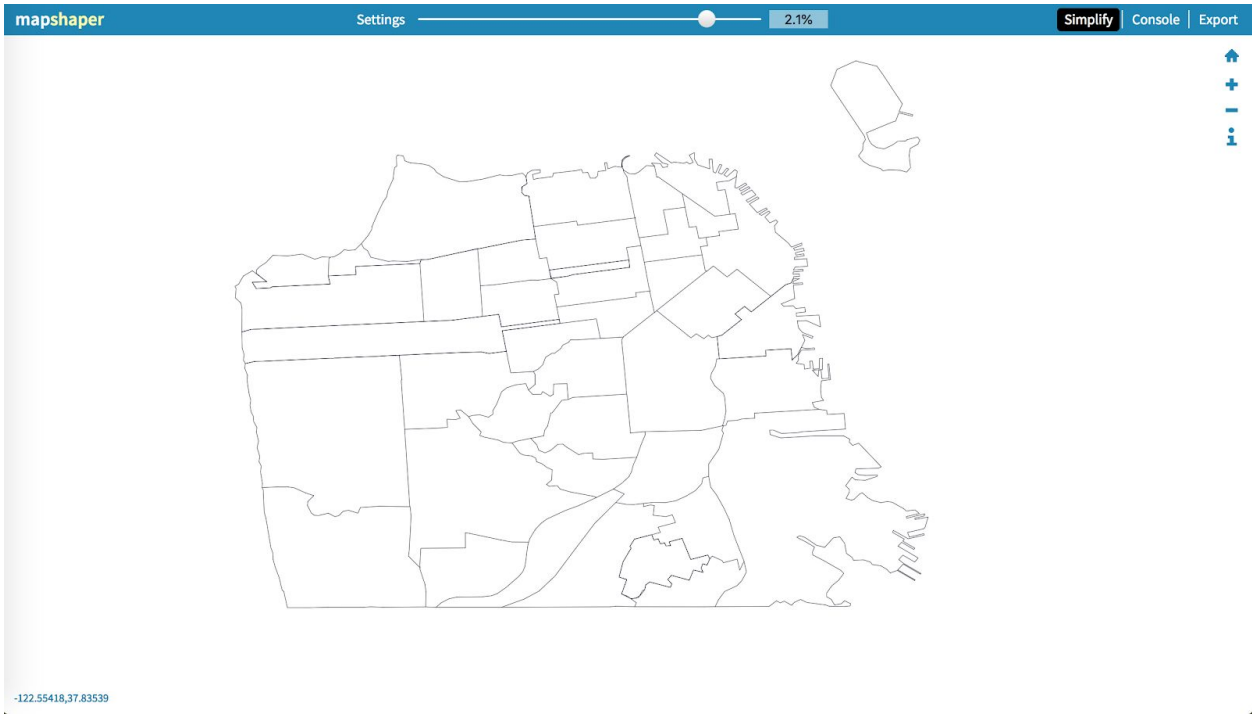


Towards the top of the map, you can see there a lot of details, especially in the Marina. All the docks have been drawn with fine details, as can be seen on a zoomed in version:



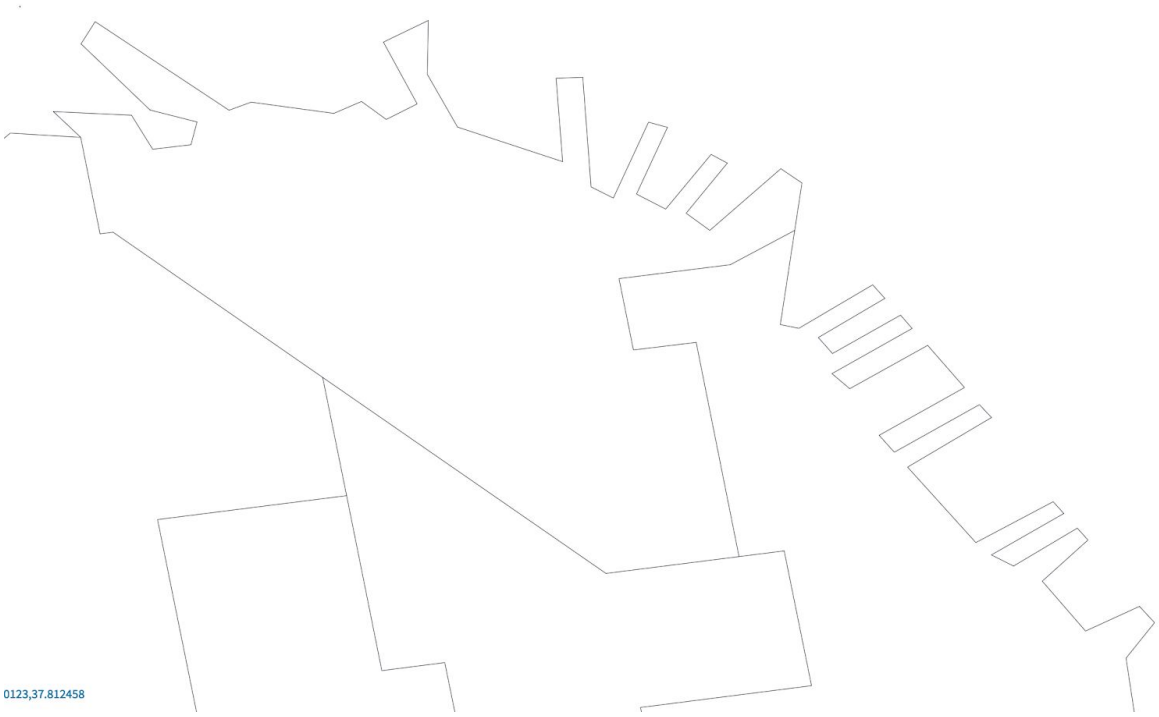
None of this can be seen when zoomed out!

MapShaper fortunately allows us to get rid of these fine details, and significantly reduce the size of the data and resulting GeoJSON file. Here is what it looks like when retaining 2 % of the original information:

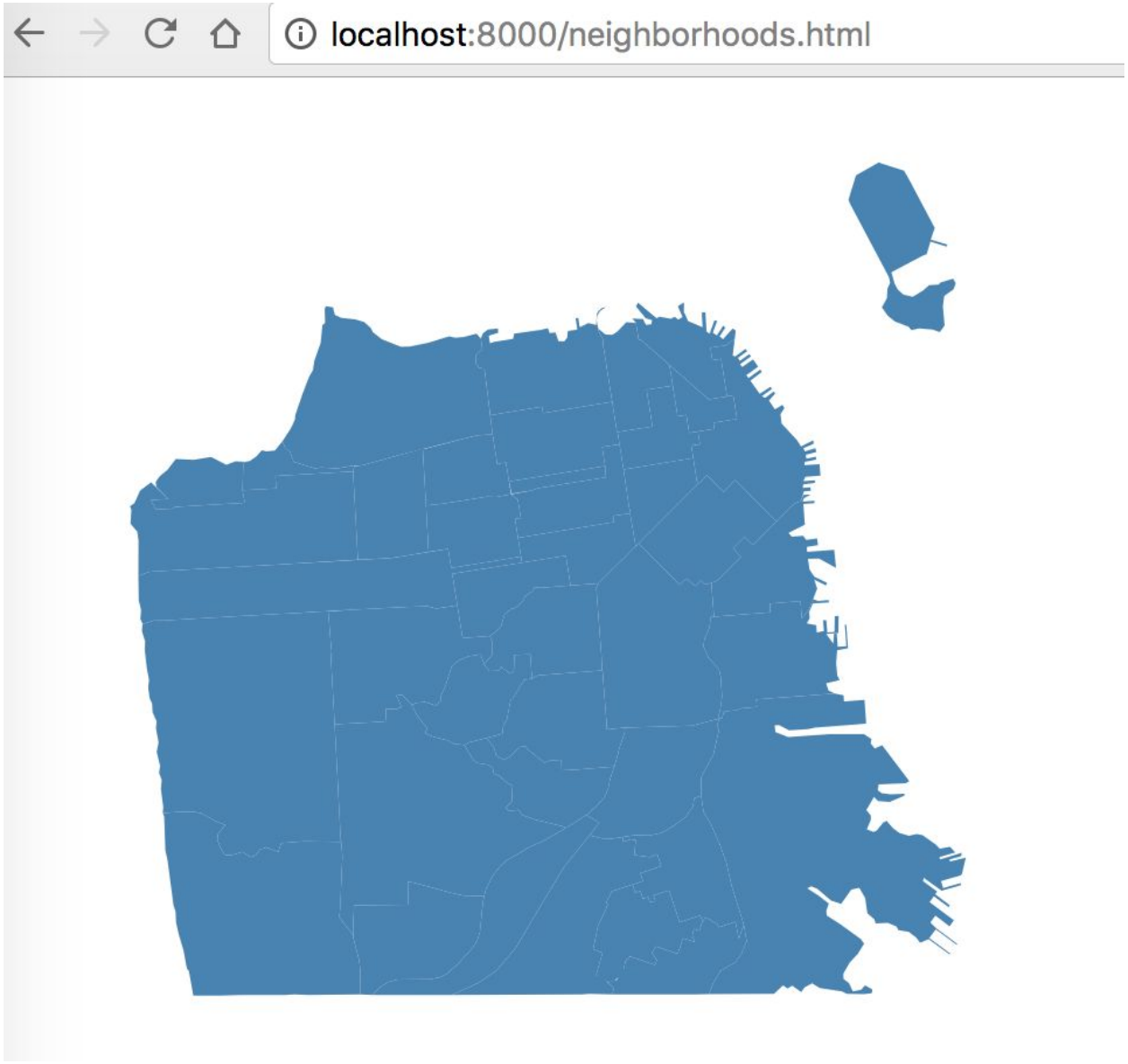


It’s amazing that at 2 % we still have so much information. This tells you how much fine detail was buried into the shapefiles…

The main piers are still visible, that’s fine:



We then used the knowledge from the course to create this simple map based on the GeoJSON file produced by MapShaper:



We now want to show the average score for each neighborhood, and color the map accordingly proportionally to that score (**choropleth** map).

The GeoJSON file also contains the name of the neighborhood as a property.

Neighborhood_average_scores.py is a Python script that we wrote, that:

1. Reads polygons (i.e. districts) from the GeoJSON file.
2. Parses the combined.json file to find all restaurants that belong to a certain neighborhood. In geographical terms, we want to find which polygon contains the coordinates X,Y of the restaurant.
   The combined.json file is modified to reflect the neighborhood to which restaurants belong.

3. Calculates the mean for each neighborhood.

This is the generated file:


```
{
  "Bayview Hunters Point": 90.073529411765,
  "Bernal Heights": 91.142857142857,
  "Castro\/Upper Market": 91.311688311688,
  "Chinatown": 87.203592814371,
  "Excelsior": 88,
  "Financial District\/South Beach": 90.53995157385,
  "Glen Park": 92.631578947368,
  "Golden Gate Park": 90.5,
  "Haight Ashbury": 90.392156862745,
  "Hayes Valley": 90.081395348837,
  "Inner Richmond": 86.064102564103,
  "Inner Sunset": 91.604651162791,
  "Japantown": 89.387096774194,
  "Lakeshore": 92.326086956522,
  "Lincoln Park": 82,
  "Lone Mountain\/USF": 93.083333333333,
  "Marina": 90.285714285714,
  "McLaren Park": 97,
  "Mission": 88.084459459459,
  "Mission Bay": 91,
  "Nob Hill": 92.533333333333,
  "Noe Valley": 90.045454545455,
  "North Beach": 92.483870967742,
  "Oceanview\/Merced\/Ingleside": 93.375,
  "Outer Mission": 88.085714285714,
  "Outer Richmond": 86.826923076923,
  "Pacific Heights": 90.74025974026,
  "Portola": 85.148148148148,
  "Potrero Hill": 91.380952380952,
  "Presidio": 90,
  "Presidio Heights": 91.44,
  "Russian Hill": 91.712121212121,
  "Seacliff": 100,
  "South of Market": 90.134969325153,
  "Sunset\/Parkside": 90.434210526316,
  "Tenderloin": 90.550561797753,
  "Treasure Island": 87,
  "Twin Peaks": 96,
  "Visitacion Valley": 86.833333333333,
  "West of Twin Peaks": 91.465517241379,
  "Western Addition": 91.137254901961
}
```


The D3 code needs to load both neighborhoods.json (a GeoJSON containing polygon and neighborhood name) and neighborhood_average_scores.json presented just above to show the complete graph.

We use the same technique as presented in the D3 book to fuse these two files, by adding the average as a property of each polygon in the GeoJSON file.

The average score is then used to color the polygon according to the following color scale:

90.0 - 100.0

86.0 - 89.9

71.0 - 85.9

50.0 - 70.9

The number represent the inspection score.

This matches the specification from the SF Health Department:

Food Safety Score Categories and Interpretation

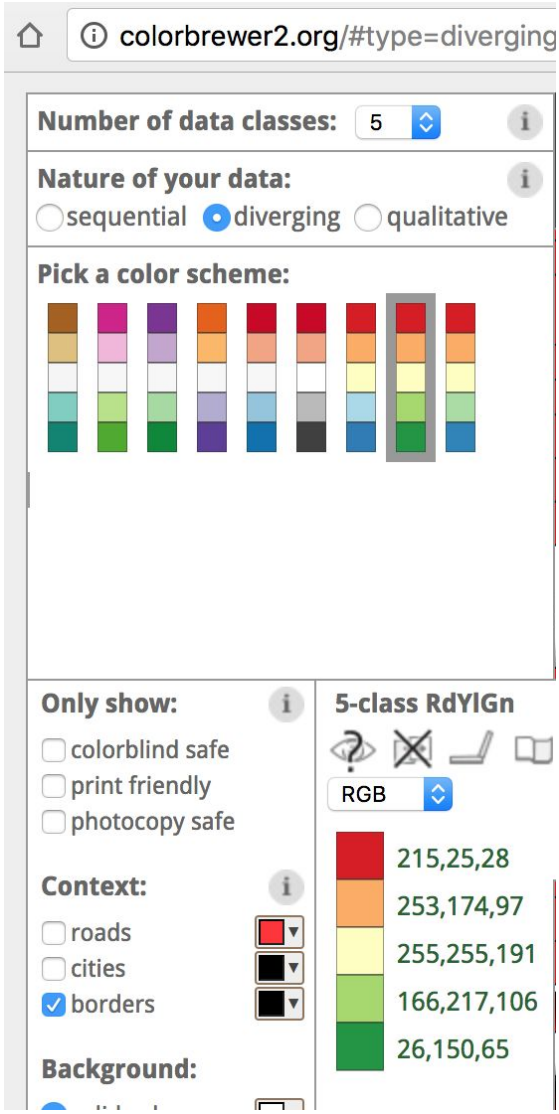
Score	Operating Condition Category	Inspection Findings
>90	Good	• Typically, only lower-risk health and safety violations observed • May have high-risk violations
86-90	Adequate	• Several violations observed • May have high-risk violations
71-85	Needs Improvement	• Multiple violations observed • Typically, several high-risk violations
Less than or equal to 70	Poor	• Multiple violations observed • Typically, several high-risk violations

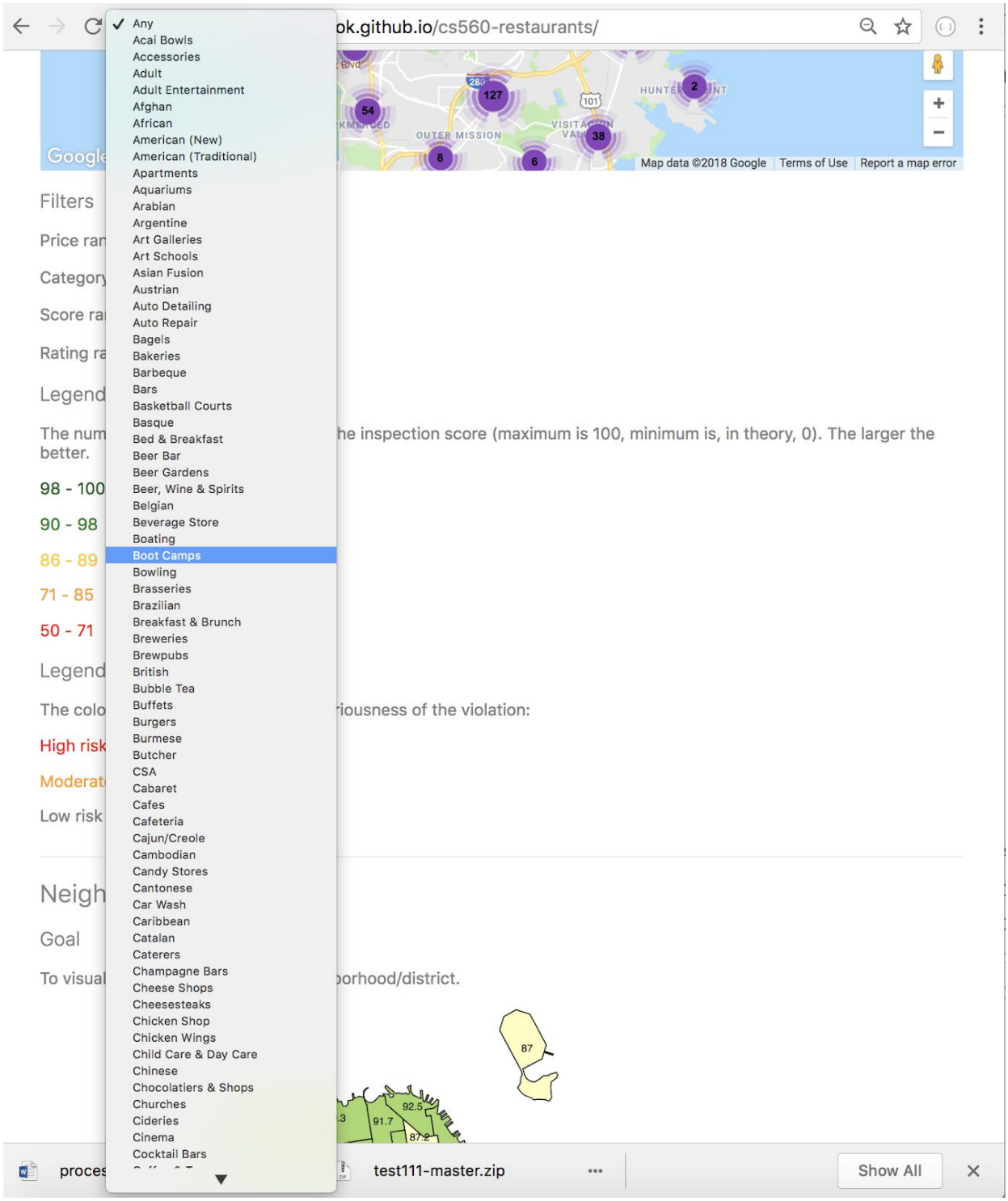
We have observed that the dataset doesn't not contain any entry below the score of 50.

By the way, they define the severity of the violations as following:

- **High Risk:** Violations that directly relate to the transmission of food borne illnesses, the adulteration of food products and the contamination of food-contact surfaces.
- **Moderate Risk:** Violations that are of a moderate risk to the public health and safety.
- **Low Risk:** Violations that are low risk or have no immediate risk to the public health and safety.

The colors are extracted from ColorBrewer:


colorbrewer2.org/#type=diverging
Number of data classes: 5
Nature of your data:
sequential diverging qualitative
Pick a color scheme:
Only show:
colorblind safe
print friendly
photocopy safe
Context:
roads
cities
borders
Background:
5-class RdYlGn
RGB
215,25,28
253,174,97
255,255,191
166,217,106
26,150,65



So the final data set contains categories of more than 40 different kinds, and a lot of them can be aggregated into a some single ones, so we used some code and combined them together. And only look at the most dominant few categories.

```
// var data[key].categories[0].title);
for (var i in data[key].categories){
    var title = data[key].categories[i].title;
    if (title=="American (Traditional)"|| title=="American (New)"){
        tit = "american";
    }
    if (title=="Cantonese"|| title=="Chinese"||title=="Dim Sum"||title=="Shanghainese"||title=="Taiwanese"||title=="Szechuan"){
        tit = "chinese"
    }
    if (title=="French"){
        tit = "french";
    }
    if (title=="Indian"){
        tit = "indian";
    }
    if (title=="Italian"){
        tit = "italian";
    }
    if (title=="Japanese"){
        tit = "japanese";
    }
    if (title=="Korean"){
        tit = "korean";
    }
    if (title=="Latin American"||title=="Mexican"){
        tit = "latin";
    }
    if (title=="Vietnamese"){
        tit = "viet";
    }
    if (title=="Thai"){
        tit = "thai";
    }
    if (title=="Mediterranean"){
        tit = "medi";
    }
}
```

So we choose these categories from the dataset and analyze the ones from this type.

We also filtered out those with undefined/null/empty entries, this left us with roughly 3000 entries.

The following code is the one we used for extracting the csv file from the combined.json It is in the scatterdf.html

```
function convertArrayOfObjectsToCSV(args) {
  var result, ctr, keys, columnDelimiter, lineDelimiter, data

  // data = args.data || null;
  // if (data == null || !data.length) {
  //     return null;
  // }

  columnDelimiter = args.columnDelimiter || ',';
  lineDelimiter = args.lineDelimiter || '\n';

  keys = Object.keys(dataset[0]);
  console.log(keys);

  result = '';
  result += keys.join(columnDelimiter);
  result += lineDelimiter;

  dataset.forEach(function(item) {
      ctr = 0;
      keys.forEach(function(key) {
          if (ctr > 0) result += columnDelimiter;

          result += item[key];
          ctr++;
      });
      result += lineDelimiter;
  });

  return result;
}

function downloadCSV(args) {
  var data, filename, link;
  var csv = convertArrayOfObjectsToCSV({
      data: dataset
  });
  if (csv == null) return;

  filename = args.filename || 'export.csv';

  if (!csv.match(/^data:text\/csv/i)) {
      csv = 'data:text/csv;charset=utf-8,' + csv;
  }
  data = encodeURI(csv);

  link = document.createElement('a');
  link.setAttribute('href', data);
  link.setAttribute('download', filename);
  link.click();
}
```

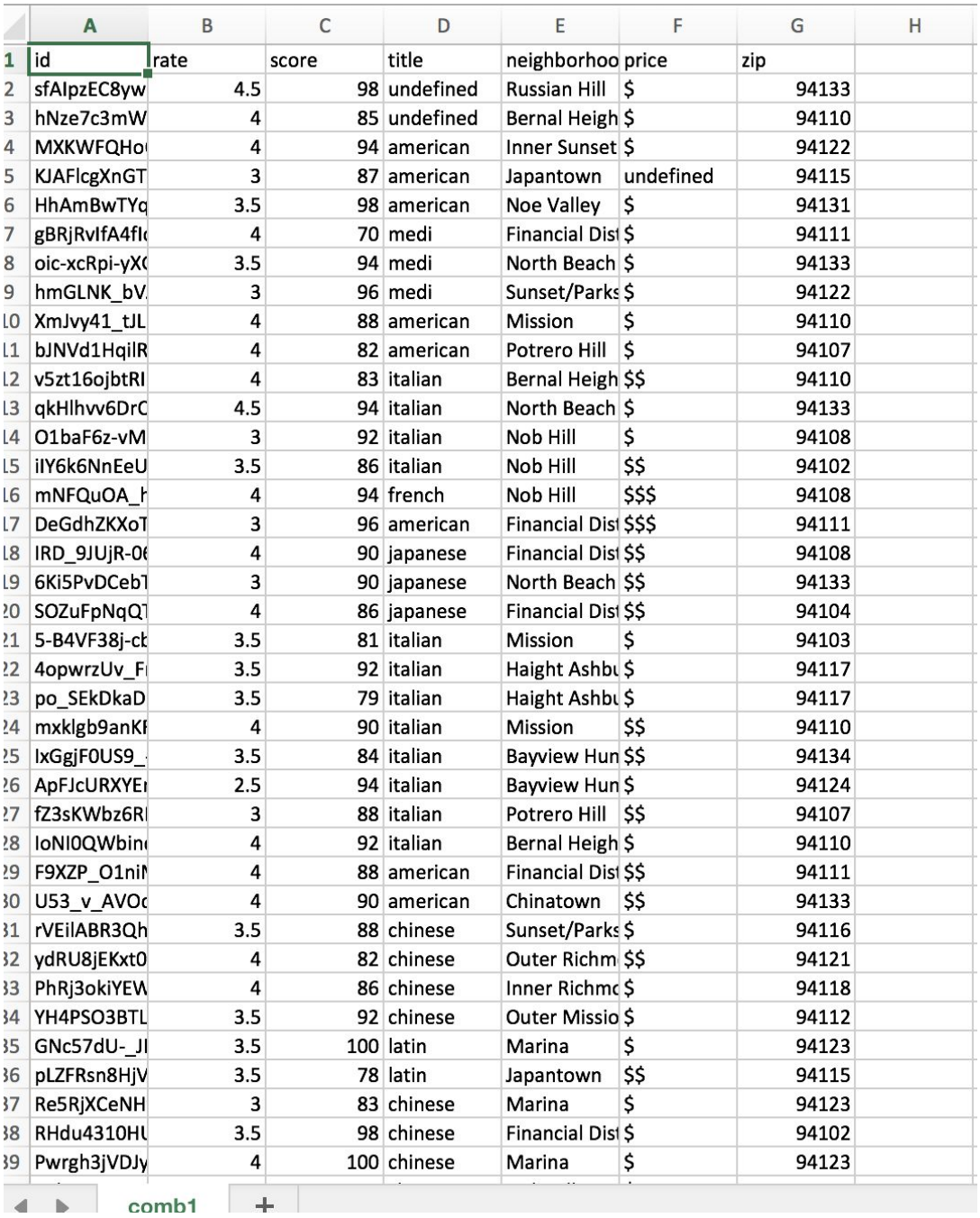
Exploratory Data Analysis

We use google API to see the overall spread of the restaurants in different areas, also the neighborhood map, we use this to get a general overview of the data and all the scores in San Francisco.

We realize that there are more than 20 categories , so we decides to choose some major categories like American, Chinese and Korean etc.

As we have done in the previous section, we combined some categories together and explore the patterns in this csv file.

Initially,we tried to do it with d3 straight away, it didn't work very well, so we find some website and consult on some of them, then we extract the partial code and extract csv file from the combined JSON. now it is much easier to do visualizations and perform R code and do analysis on it.
So the result csv file looks like this.

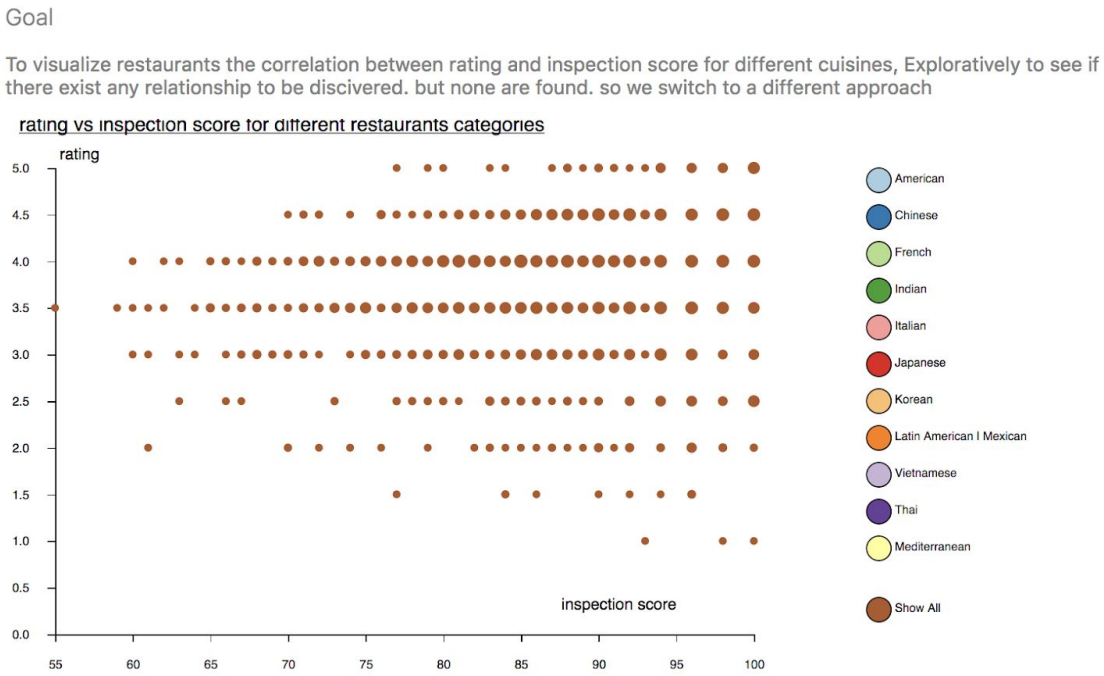


	A	B	C	D	E	F	G	H
1	id	rate	score	title	neighborhoo	price	zip	
2	sfAIpzEC8yw	4.5	98	undefined	Russian Hill	$	94133	
3	hNze7c3mW	4	85	undefined	Bernal Heigh	$	94110	
4	MXKWFQHo	4	94	american	Inner Sunset	$	94122	
5	KJAFlcgXnGT	3	87	american	Japantown	undefined	94115	
6	HhAmBwTYq	3.5	98	american	Noe Valley	$	94131	
7	gBRjRvIfA4fI	4	70	medi	Financial Dist	$	94111	
8	oic-xcRpi-yX	3.5	94	medi	North Beach	$	94133	
9	hmGLNK_bV	3	96	medi	Sunset/Parks	$	94122	
10	XmJvy41_tJL	4	88	american	Mission	$	94110	
11	bJNVd1HqilR	4	82	american	Potrero Hill	$	94107	
12	v5zt16ojbtRI	4	83	italian	Bernal Heigh	$$	94110	
13	qkHlhvv6DrC	4.5	94	italian	North Beach	$	94133	
14	O1baF6z-vM	3	92	italian	Nob Hill	$	94108	
15	iIY6k6NnEeU	3.5	86	italian	Nob Hill	$$	94102	
16	mNFQuOA_h	4	94	french	Nob Hill	$$$	94108	
17	DeGdhZKXoT	3	96	american	Financial Dist	$$$	94111	
18	IRD_9JUjR-0	4	90	japanese	Financial Dist	$$	94108	
19	6Ki5PvDCebT	3	90	japanese	North Beach	$$	94133	
20	SOZuFpNqQT	4	86	japanese	Financial Dist	$$	94104	
21	5-B4VF38j-cb	3.5	81	italian	Mission	$	94103	
22	4opwrzUv_F	3.5	92	italian	Haight Ashbu	$	94117	
23	po_SEkDkaD	3.5	79	italian	Haight Ashbu	$	94117	
24	mxklgb9anKF	4	90	italian	Mission	$$	94110	
25	lxGgjF0US9_	3.5	84	italian	Bayview Hun	$$	94134	
26	ApFJcURXYE	2.5	94	italian	Bayview Hun	$	94124	
27	fZ3sKWbz6R	3	88	italian	Potrero Hill	$$	94107	
28	loNI0QWbin	4	92	italian	Bernal Heigh	$	94110	
29	F9XZP_O1niN	4	88	american	Financial Dist	$$	94111	
30	U53_v_AVOc	4	90	american	Chinatown	$$	94133	
31	rVEilABR3Qh	3.5	88	chinese	Sunset/Parks	$	94116	
32	ydRU8jEKxt0	4	82	chinese	Outer Richm	$$	94121	
33	PhRj3okiYEW	4	86	chinese	Inner Richmc	$	94118	
34	YH4PSO3BTL	3.5	92	chinese	Outer Missio	$	94112	
35	GNc57dU-_JI	3.5	100	latin	Marina	$	94123	
36	pLZFRsn8HjV	3.5	78	latin	Japantown	$$	94115	
37	Re5RjXCeNH	3	83	chinese	Marina	$	94123	
38	RHdu4310HU	3.5	98	chinese	Financial Dist	$	94102	
39	Pwrgh3jVDJy	4	100	chinese	Marina	$	94123	

comb1

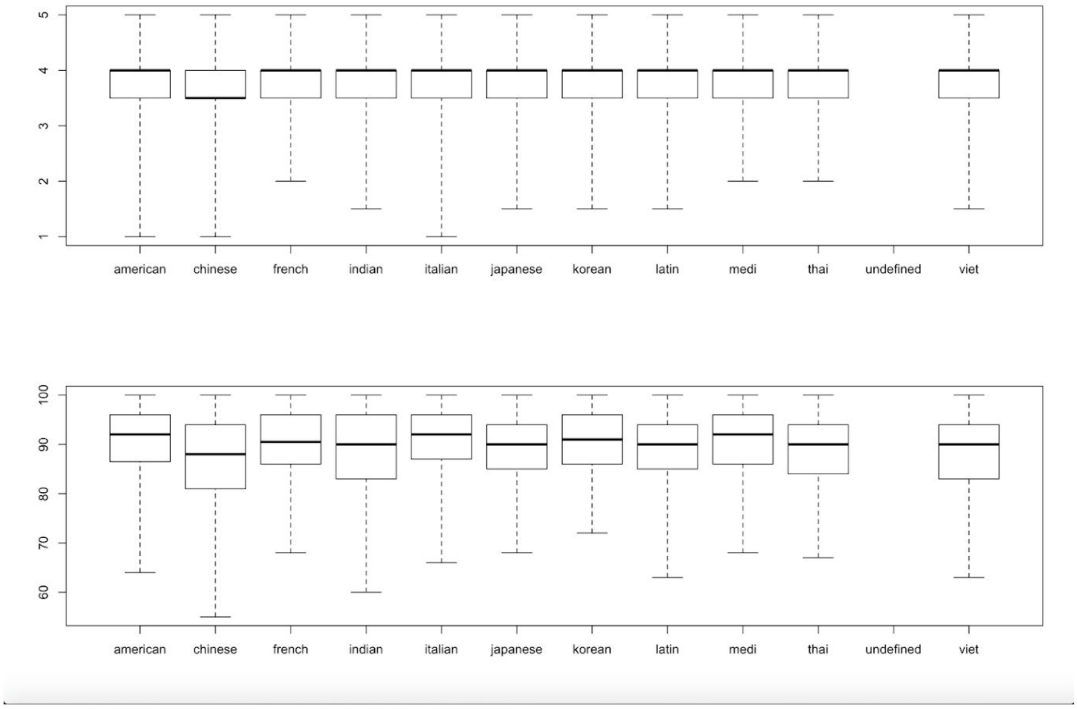
In short, we wanted to explore the distribution of different categories in san francisco. So we took the rate column and score columns, and see if there is any correlation between two of those.

We used scatter plot to explore the relationship in each category, but we didn't observe any significant trend.



So we decided to look into the boxplot which we could explore the distribution of different category and compare the median and extreme values of different categories.

At the end, we didn't see any specific trends or values that can differentiate them, they all seems to cluster together and the median, variance are indistinguishable.



So we looked at the more specific mathematical tools to analyze this and hope it would give a better result.

So we tried to look for papers online which discuss similar topics.

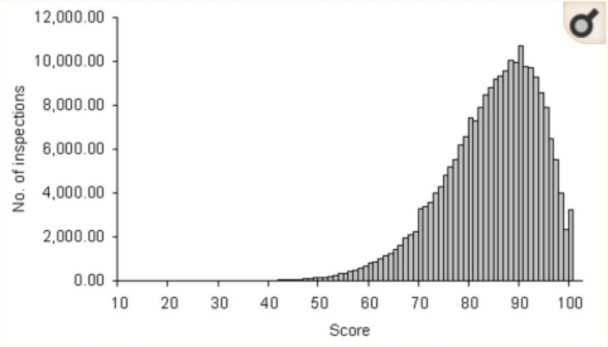


Figure 1

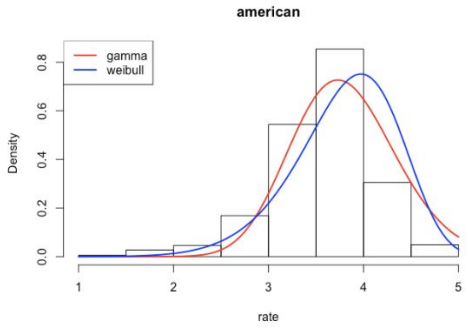
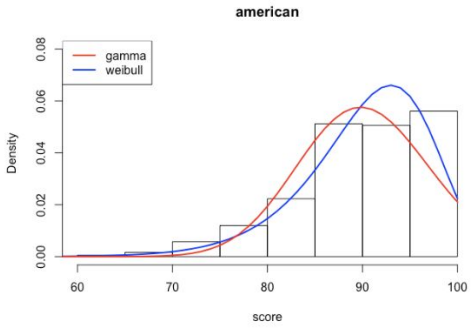
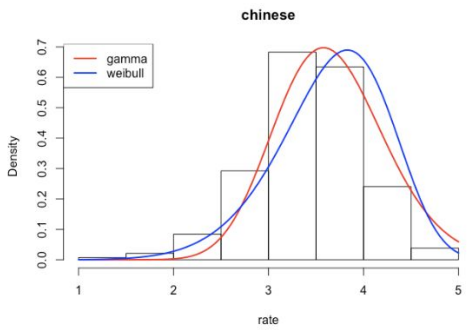
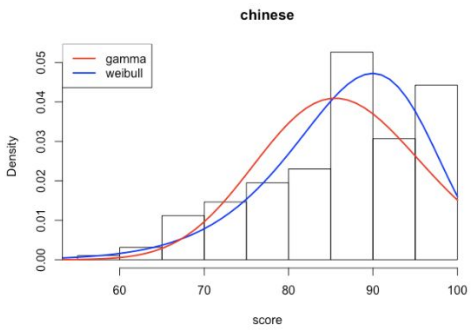
Distribution of scores of restaurants inspected statewide from July 1993 to June 2000, based on a standardized inspection with 44 scored items and a maximum score of 100.

The paper mentioned specifically that the distribution of inspection score is right skewed and has the mean deviated to the right.

It didn't mention the specific distribution type of this inspections score. So we used the most common ones in special distributions.

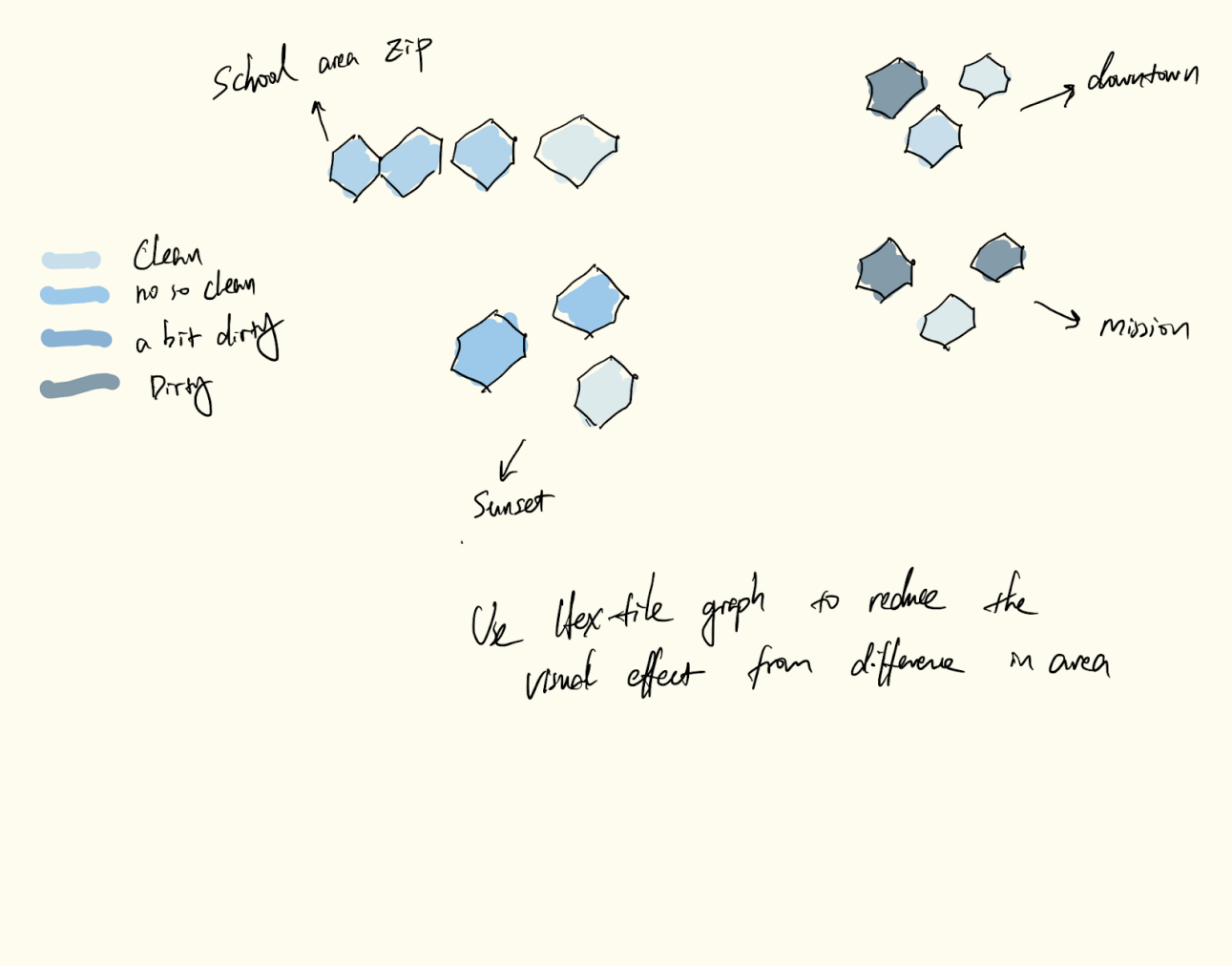
Gamma and Weibull,

This would give a better understanding of all distribution of datas

	Rating	Score
American		
	Gam: 3.82, Wei: 3.97	Gam: 89.89, Wei: 93.01
Chinese		
	Gam 3.61, Wei: 3.87	Gam: 85.41, Wei: 90.19

Design Evolution

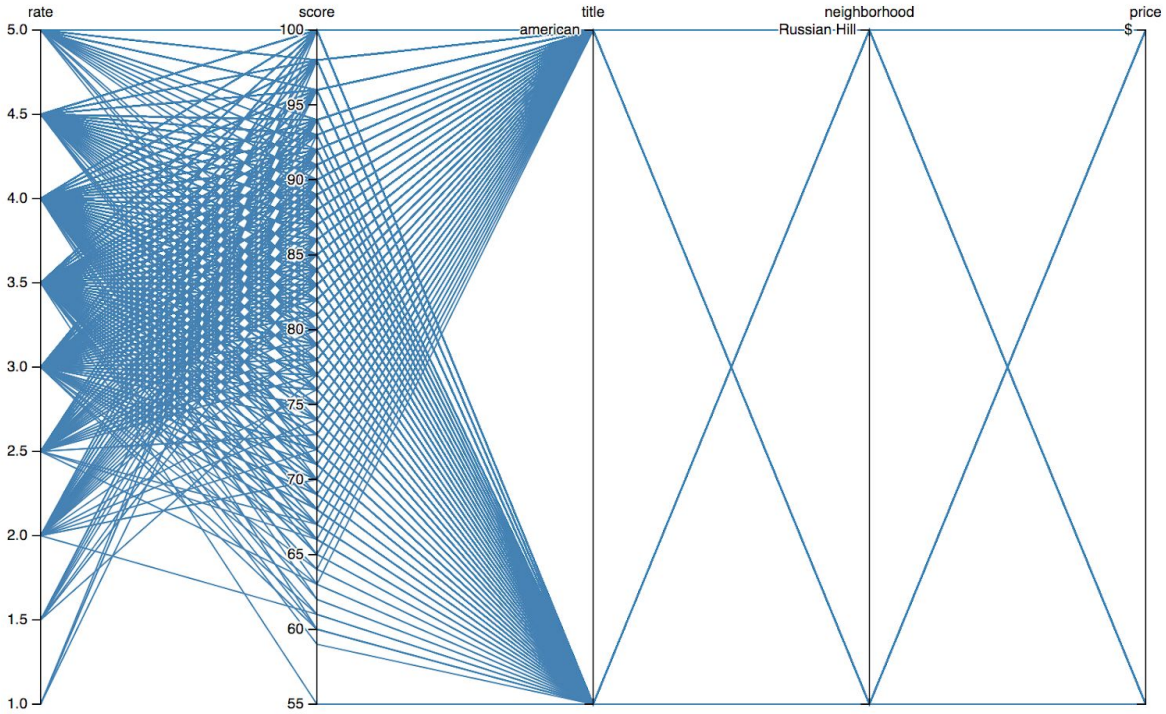
We have considered Hex map but we lack the the topo json file for san francisco, also more importantly we all know san francisco very well so we decided to abandons the idea and use choropleth map, this gives everyone a more direct view of the score of different restaurants in san francisco.



This would nicer only if we don't know about san francisco that directly.

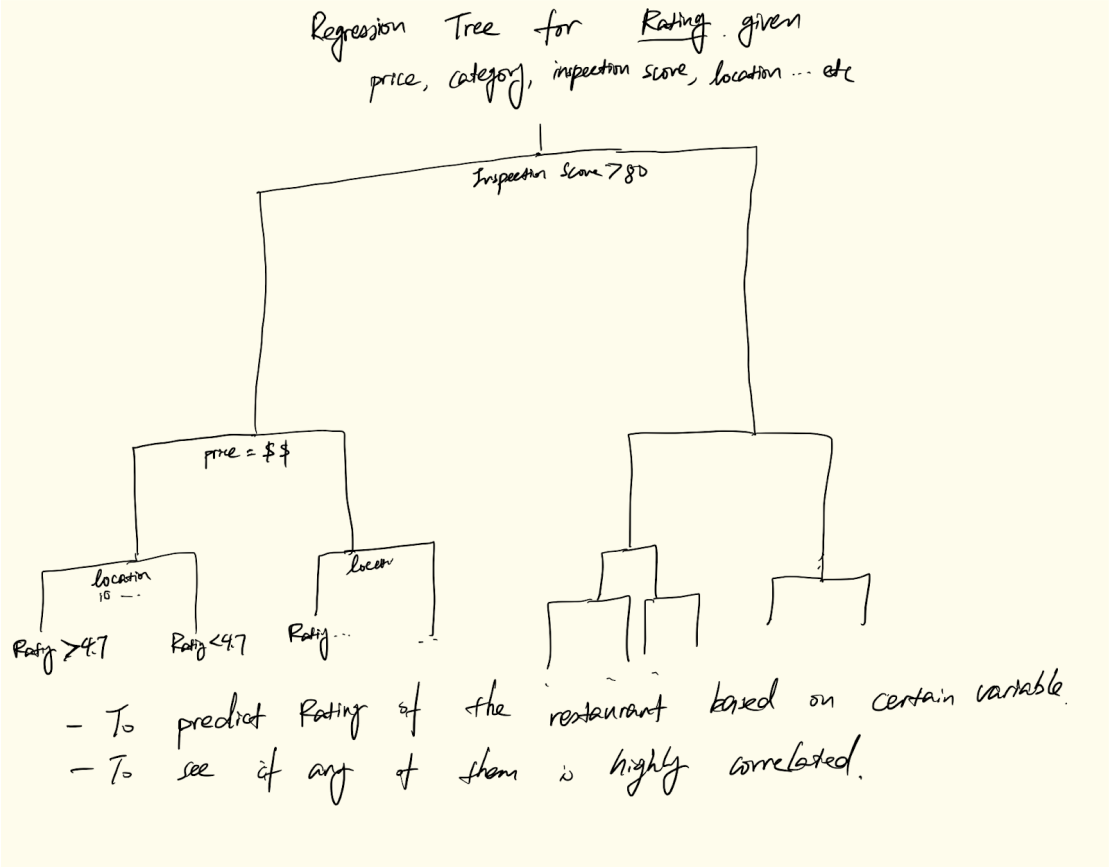
Also we have thought about parallel coordinates. We are half done with the parallel coordinates but we have already seen the pattern where the link are very evenly

distributed among different category, there is not much to be explore and discover in term of correlations. So we abandoned the idea too.

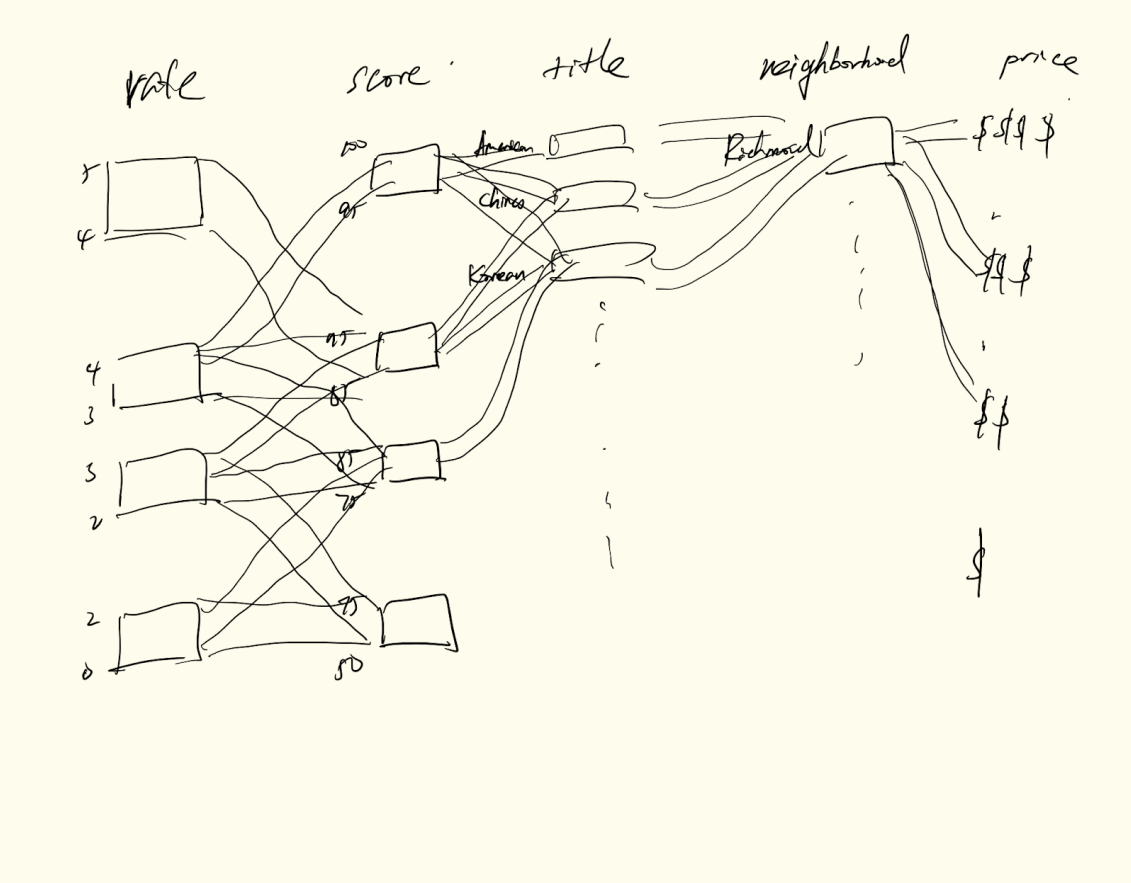


Another one we looked into is the tree regression diagram in R, producing the viz relatively meaningless if the user is unaware of the meaning about the random forest

and tree height etc. So we abandoned it too.



Another one we looked at the Sankey diagram, but for the same reason as the sankey diagram. We think it would be unnecessary if all channels are evenly distributed. There would not be much information to say except that they are all distributed among all channels.



Therefore, after all these concerns we decided to use other mathematical tools to analyze the data set and use them to analyze the data.

Also it would be very difficult to implement this graph, so we didn't make it eventually.

In principle, we didn't get deviated from the original goal.

The google map visualization we create view the locations and violations very directly and intuitively, it provide the most intuitive interaction and user experience. Also we choose to use simpler distribution diagram so that everyone has basic statistical background can understand what is going on and understand what we are trying to visualize in all senses.

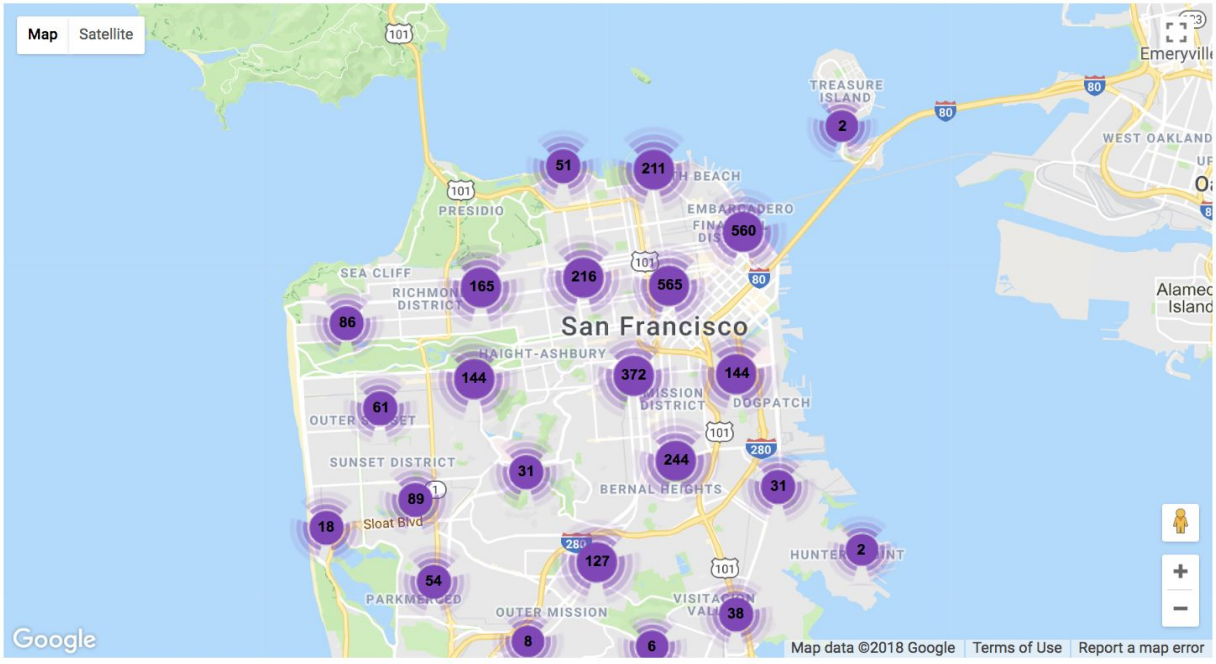
The biggest lesson we learned from this course is about storytelling, in this part, we believe we have done a fantastic job. We started by looking into all categories of data and have a intuitive general idea of what each one and category could be.

Then we look into the visualization and manage to select those ones is the most dominant and explore the correlation and distribution among those columns and rows.

Eventually we use some mathematical tools to analyze them and draw conclusions. Overall, we didn't get deviated from the original task and it is a fantastic job in doing this.

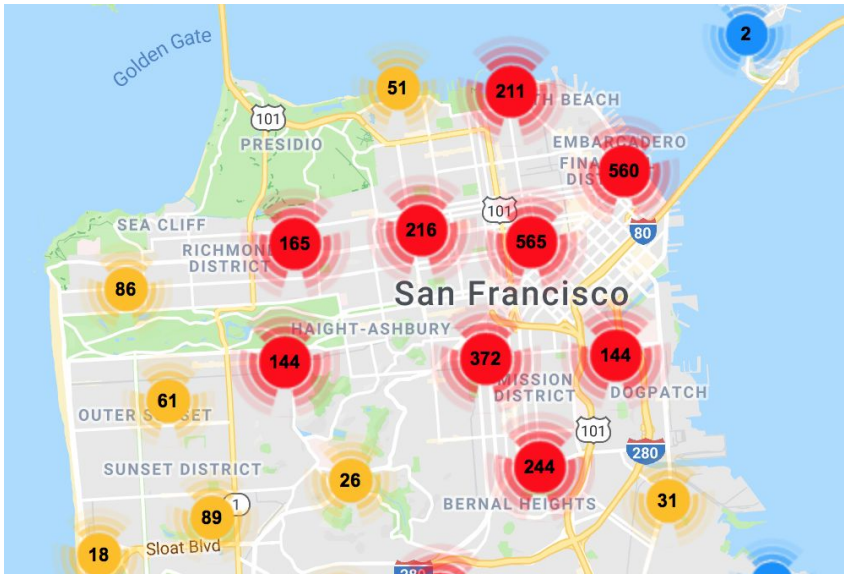
Implementation

So there are four visualizations we worked on mostly, the first one is the google map one.



It worked exactly as google map, you can see the clusters of the restaurants when you open the main page, then you would have a general idea about the where the clusters are based on the number.

Our first version used different colors depending on the density of the cluster:



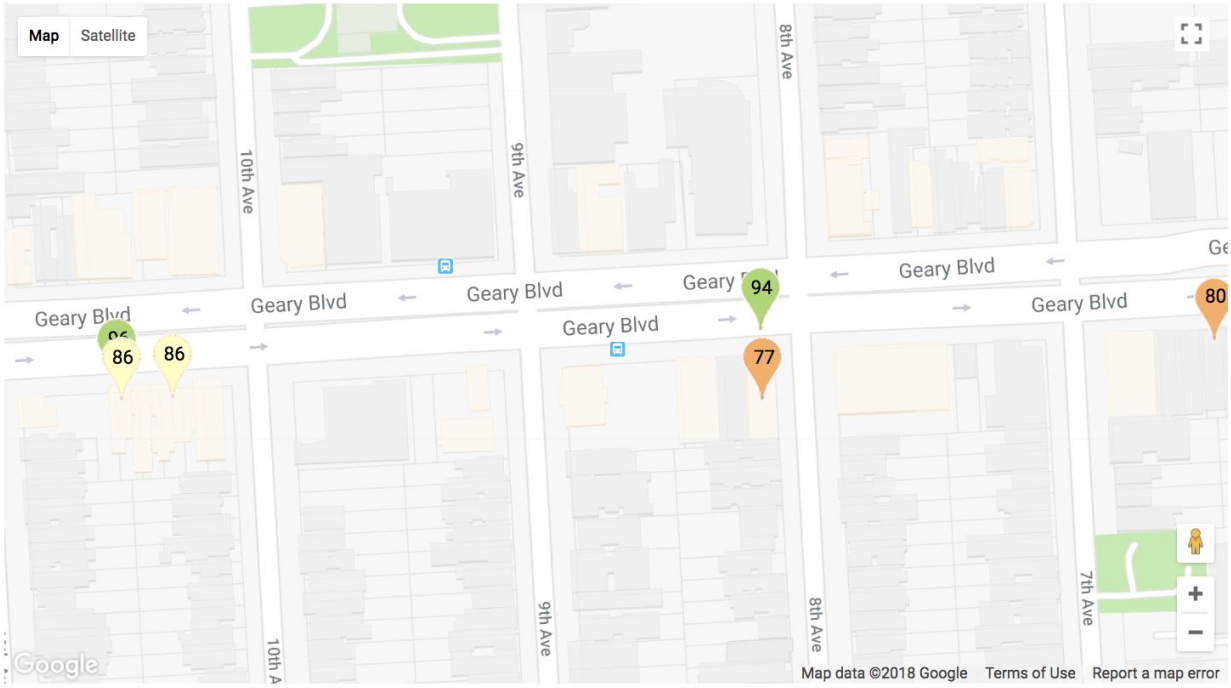
with the following meaning:

Blue:	2 - 9
Yellow:	10 - 99
Red:	100 - 999
Purple:	1000+

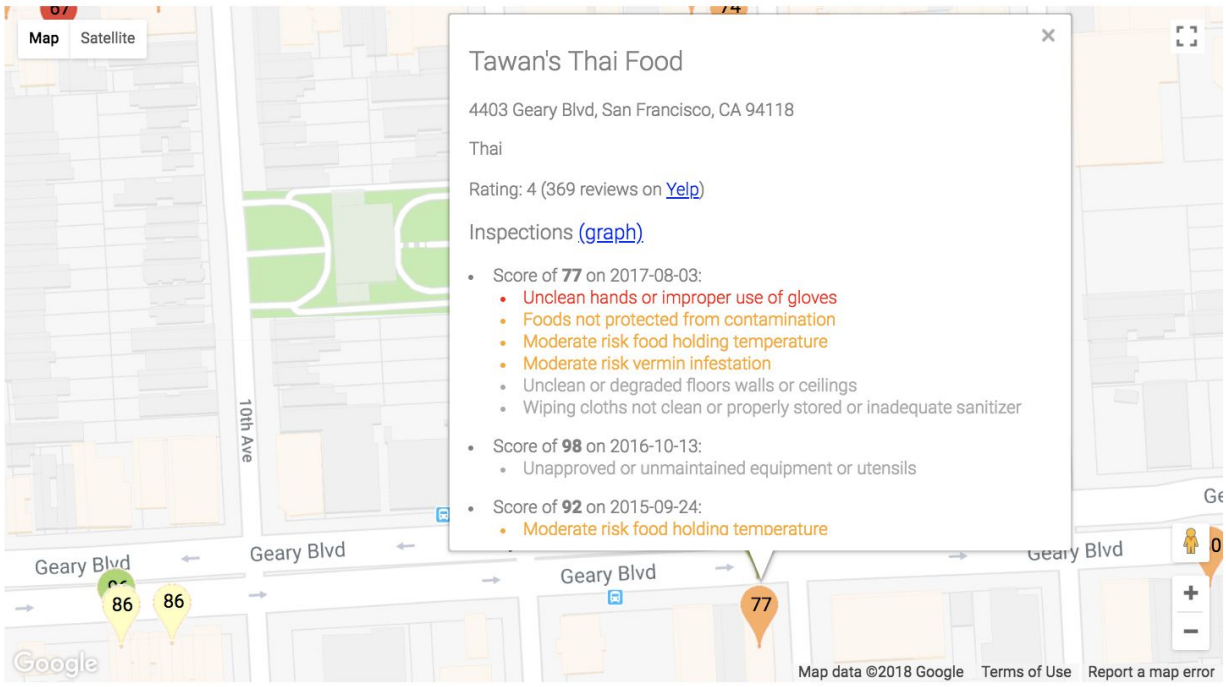
where the number on the right is the number of establishments within that cluster.

However, during our presentation in class, some people have reported that they were confused by the use of similar colors between the pinpoints where color is used for the latest inspection score (see below) and the clusters. Since the ability to distinguish clusters by size was very minor for what we tried to achieve, we simply removed the color. However the proportional size and beacon effect has been kept.

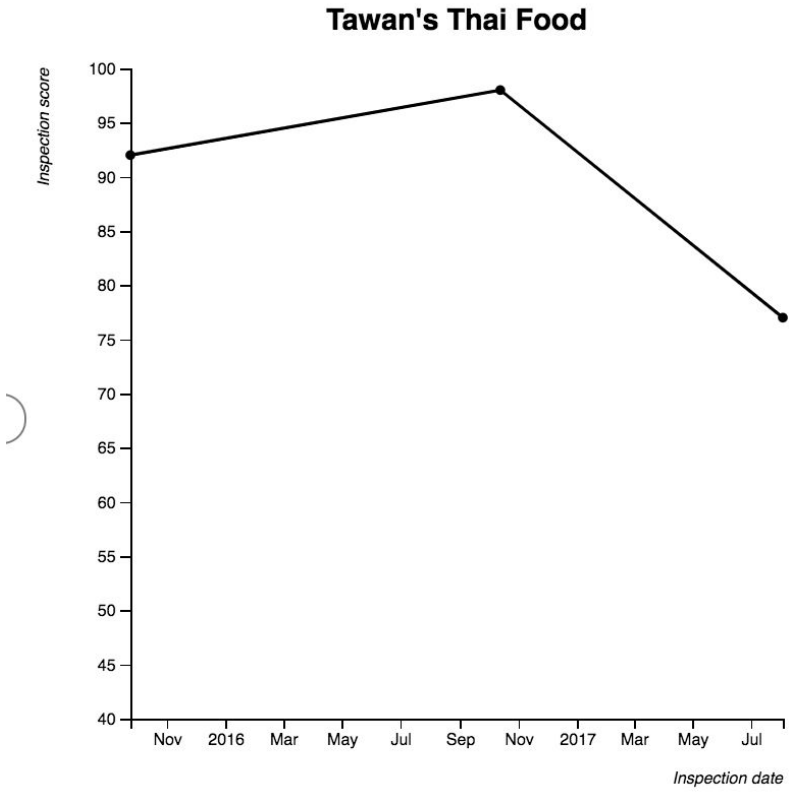
Then when you zoom in you can see the individual restaurants and be able to interact with them and see if any of them are any score in particular.



The when you click onto one of them you can see if they have one violations or multiple violations and see if any of right the bell. The color is also labeled.

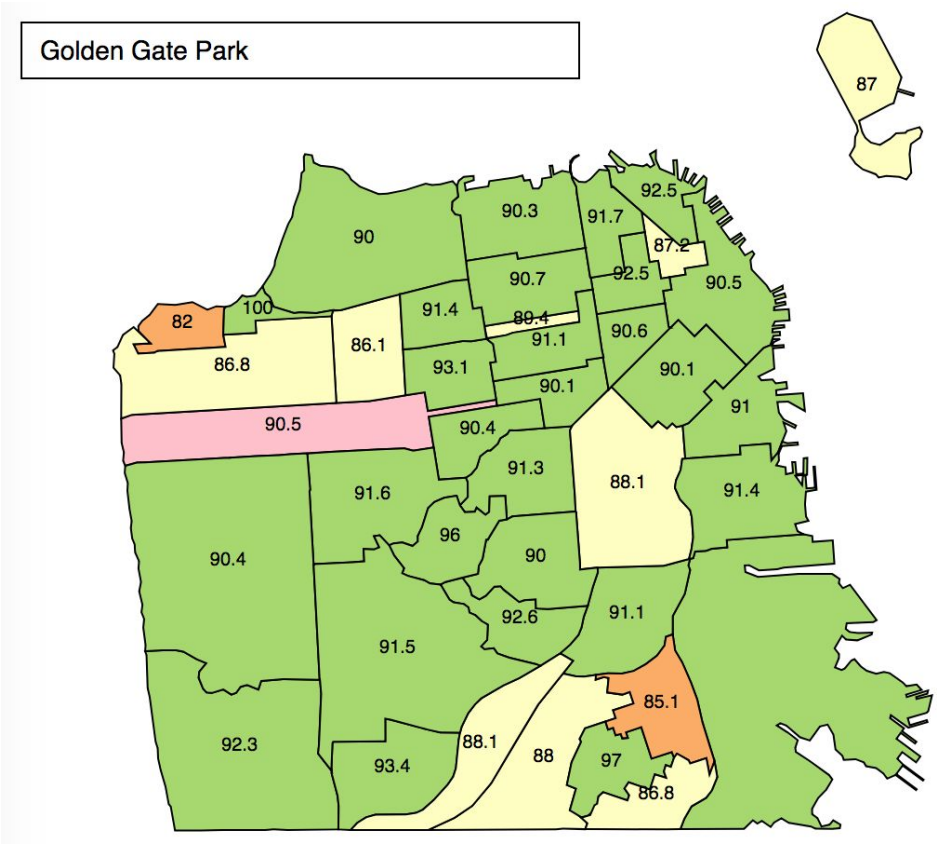


Then you can click onto the graph and see a line chart of the change and see any change in inspection score and see how it evolve through time.

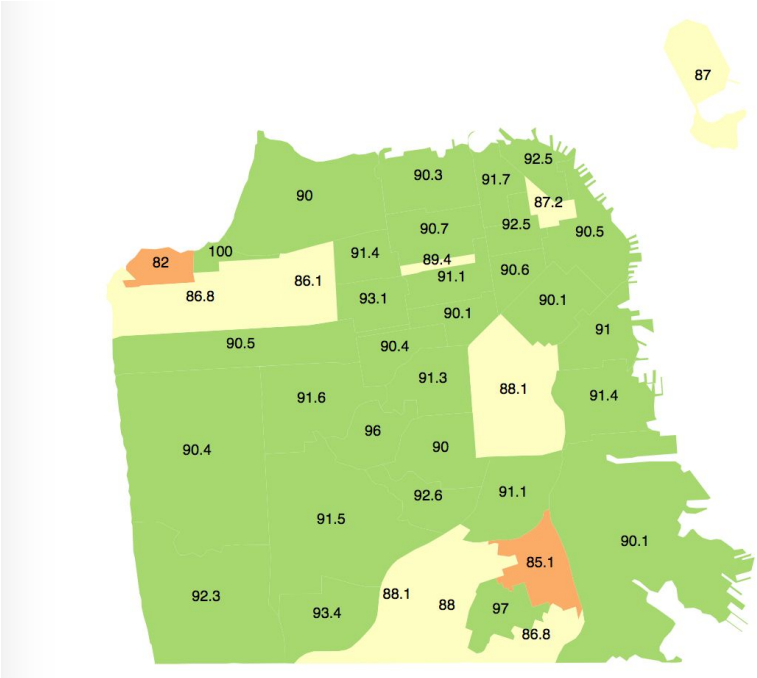


Then we can see the visualization of a geo map where we can see how different neighborhood varies in terms of inspection scores.

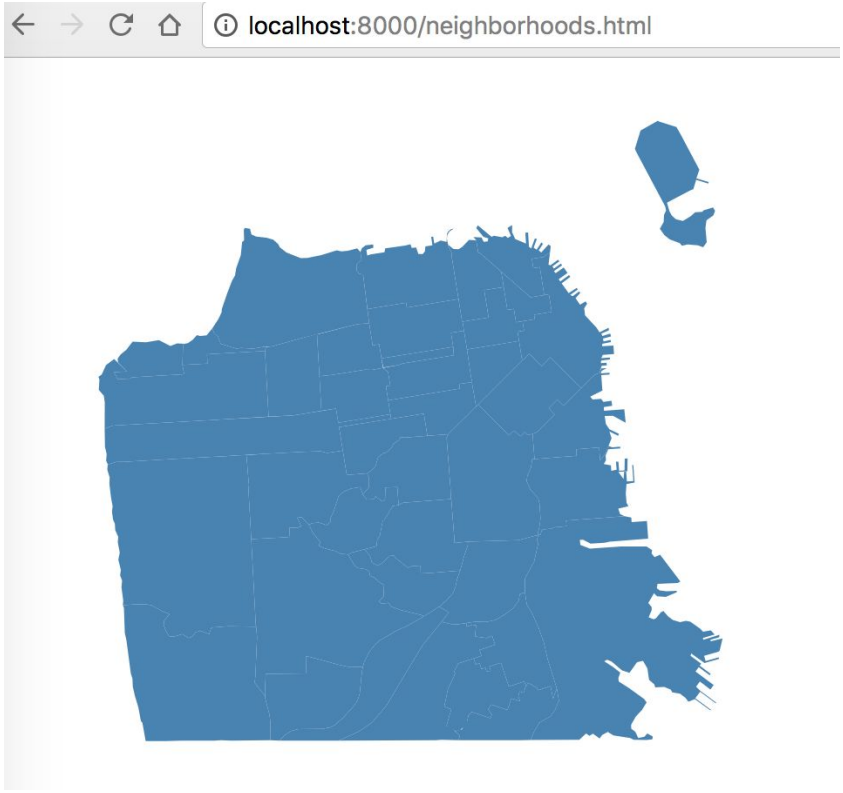
When you hover your mouse onto them you will see the color changes from specific color to pink.



This is our final version as before that we had this:

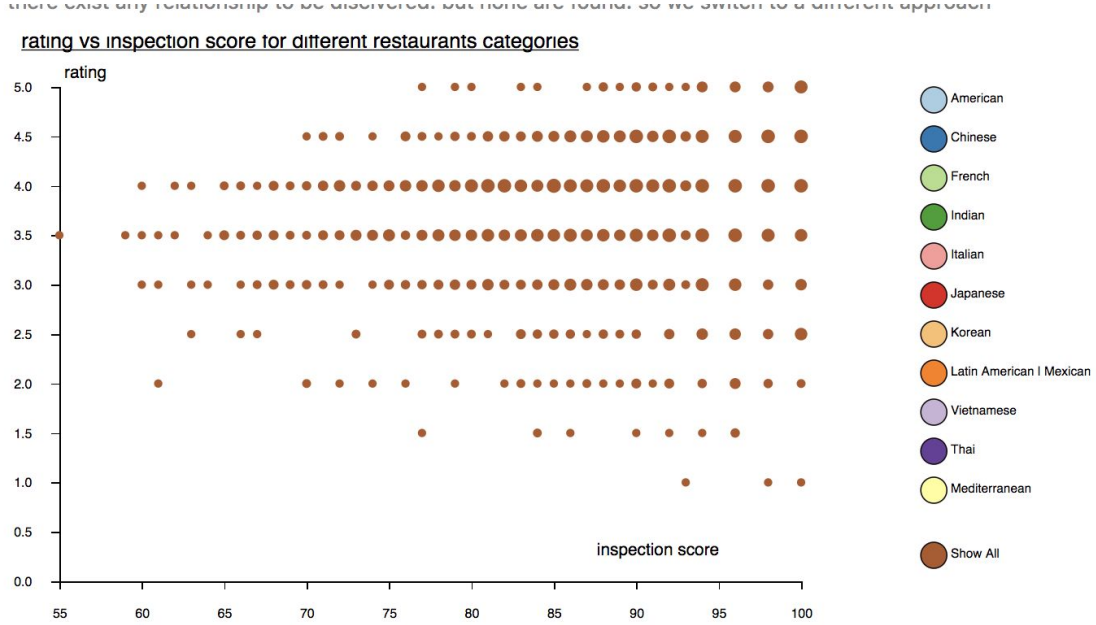


But the outlines of neighborhoods were hard to see.
And as already mentioned, we started with a non-colored map (not choropleth):



Now it is the turn to look at the scatter plot.

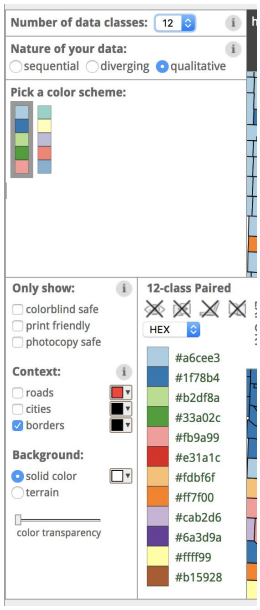
The radius of the circle is proportional to the number of overlaps. And if you move onto any of the legend and click on it then you will see the datas on that specific category of restaurant.



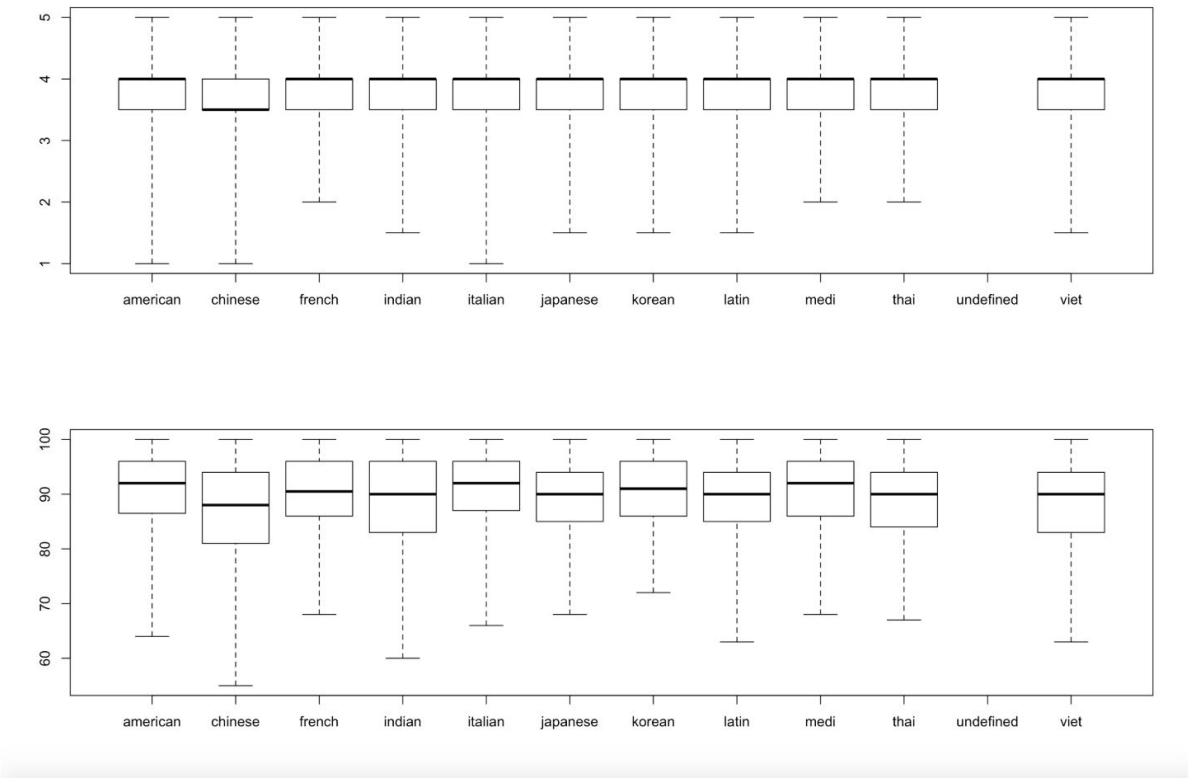
You can see the correlation between ratings and scores of each category.

So a natural question to ask is if there are not any correlation between them, what about the distribution among them.

We used this color-brewer to record the color.



A quicker way is to look at the distribution is to use boxplot, so it is the quick way to get a general idea of the distribution.



There seems like no strong and significant difference so we decide to use more complicated statistical tools.
So we defined 4 functions which are the negative mle score of the the gamma distribution and the weibull distribution. We will minimize this in order to find the maximum of the mle estimator given the data set.

```r
chinese.rate <- as.numeric(comb$rate[which(comb$title=="chinese")])
chinese.score <- as.numeric(comb$score[which(comb$title=="chinese")])
# hist(chinese.rate,freq=FALSE, main = "chinese" , xlab = "rating")
# hist(chinese.score,freq=FALSE, main = "chinese" , xlab = "score")

minuslogl.gam.rate <- function(theta, kappa) {
  y <- dgamma(chinese.rate, shape = kappa, scale = theta)
  # return -sum ln(y)
  nLL <- -sum(log(y))
  return(nLL)
}

minuslogl.gam.score <- function(theta, kappa) {
  y <- dgamma(chinese.score, shape = kappa, scale = theta)
  # return -sum ln(y)
  nLL <- -sum(log(y))
  return(nLL)
}

minuslogl.wei.rate <- function(yita, beta) {
  y <- suppressWarnings(dweibull(chinese.rate, shape = yita, scale = beta))
  # return -sum ln(y)
  nLL <- -sum(log(y))
  return(nLL)
}

minuslogl.wei.score <- function(yita, beta) {
  y <- suppressWarnings(dweibull(chinese.score, shape = yita, scale = beta))
  # return -sum ln(y)
  nLL <- -sum(log(y))
  return(nLL)
}
```

In this case, we use method of moment as the initialization point, and use built in function to iteratively find the minimum point so this would give me the optimum mle estimator.

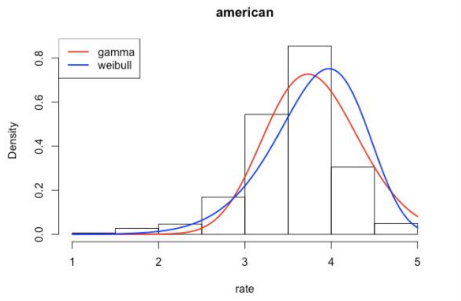
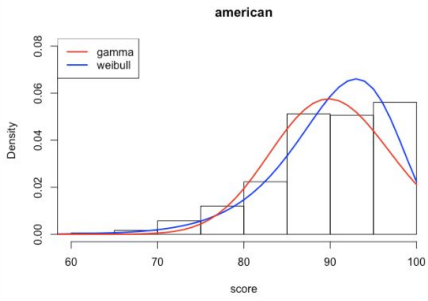
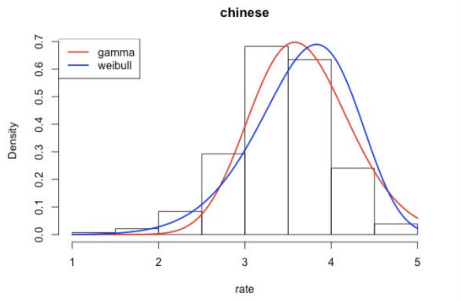
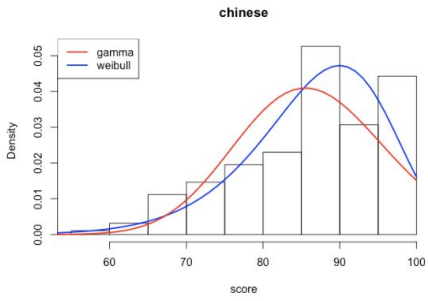
The we would use these estimator as the parameters for my distribution function. Then i just them on top of each other.

```r
c <- chinese.score
n <- length(c)
kappa.mme <- (n*mean(c)^2) / ((n-1)*var(c))
theta.mme <- ((n-1)*var(c)) / (n*mean(c))
fit.gam.score <- mle(minuslogl.gam.score, start = list(theta = theta.mme, kappa = kappa.mme))
summary(fit.gam.score)
theta.mle <- fit.gam.score@coef[1]
kappa.mle <- fit.gam.score@coef[2]
print(theta.mle*kappa.mle)

fit.wei.score <- mle(minuslogl.wei.score, start = list(yita = 1, beta = 1))
summary(fit.wei.score)
yita.mle <- fit.wei.score@coef[1]
beta.mle <- fit.wei.score@coef[2]
hist(c,freq=FALSE, main = "chinese" , xlab = "score")
curve(dweibull(x,shape=yita.mle,scale =beta.mle),from=0,to=100,col="blue",lwd=2,add=TRUE)
curve(dgamma(x,shape=kappa.mle,scale = theta.mle),from=0,to=100,col="red",lwd=2,add=TRUE)
legend("topleft",legend=c("gamma","weibull"),col=c("red","blue"),lwd=2)

c <- chinese.rate
n <- length(c)
kappa.mme <- (n*mean(c)^2) / ((n-1)*var(c))
theta.mme <- ((n-1)*var(c)) / (n*mean(c))
fit.gam.rate <- mle(minuslogl.gam.rate, start = list(theta = theta.mme, kappa = kappa.mme))
summary(fit.gam.score)
theta.mle <- fit.gam.rate@coef[1]
kappa.mle <- fit.gam.rate@coef[2]
print(theta.mle*kappa.mle)

fit.wei.rate <- mle(minuslogl.wei.rate, start = list(yita = 1, beta = 1))
summary(fit.wei.rate)
yita.mle <- fit.wei.rate@coef[1]
beta.mle <- fit.wei.rate@coef[2]
hist(c,freq=FALSE, main = "chinese" , xlab = "rate")
curve(dgamma(x,shape=kappa.mle,scale = theta.mle),from=1,to=5,col="red",lwd=2,add=TRUE)
curve(dweibull(x,shape=yita.mle,scale =beta.mle),from=1,to=5,col="blue",lwd=2,add=TRUE)
legend("topleft",legend=c("gamma","weibull"),col=c("red","blue"),lwd=2)
```

American		
	Gam: 3.82, Wei: 3.97	Gam: 89.89, Wei: 93.01
Chinese		
	Gam 3.61, Wei: 3.87	Gam: 85.41, Wei: 90.19

Eventually you can draw distribution curves on the bar charts and see if the mean is higher. This would better compare over simplified version of distribution curves.

Evaluation

We learned the biggest lesson from Data visualization is that storytelling is the most important one. It is important to have fancy diagrams and do a lot of fantastics thing, but the most important thing is to explore the data and draw conclusion from it.

We started by looking at the messy data with more than 20 entries per row and over 30000 rows, then we decided to trim it down and select the most significant columns so that only the most important features that we want to explore can be discovered and seesn.

We used some statistical tools to compare difference and draw conclusion from it. Eventually we saw the distribution and compare categorical distance from different category and see the eventual result.

Further improve is that we could look deeper and see other, explore other interesting patterns in this document and see if we can extract any useful information from it.

But so far we have answered our questions successfully, we have found out about the how the inspection score are distributed and how rating are distributed.

We believe we did a great job in this visualization task.