

LIS590DCL Final Project

Jialu Wang

1 Initial assessment of the dataset

Microsoft excel and Rstudio is initially used to form an opinion on the initial assessment.

1.1 Structure

- There are 59 attributes and 8,665 records in the dataset

```
> dim(dataSet)
```

```
[1] 8665 59
```

- There are 185145 missing values in the dataset:

```
> sum(is.na(dataSet))
```

```
[1] 185145
```

- There are 3023 records that miss the values for baked goods to wild harvest.

1.2 Content

The attributes are:

```
> names(dataSet)
```

```
[1] "FMID" "MarketName" "Website"
```

```
[4] "Facebook" "Twitter" "Youtube"
```

```
[7] "OtherMedia" "street" "city"
```

```
[10] "County" "State" "zip"
```

```
[13] "Season1Date" "Season1Time" "Season2Date"
```

```
[16] "Season2Time" "Season3Date" "Season3Time"
```

```
[19] "Season4Date" "Season4Time" "x"
```

```
[22] "y" "Location" "Credit"
```

```
[25] "WIC" "WICcash" "SFMNP"
```

```
[28] "SNAP" "Organic" "Bakedgoods"
```

```
[31] "Cheese" "Crafts" "Flowers"
```

```
[34] "Eggs" "Seafood" "Herbs"
```

```
[37] "Vegetables" "Honey" "Jams"
```

```
[40] "Maple" "Meat" "Nursery"
```

```
[43] "Nuts" "Plants" "Poultry"
```

```
[46] "Prepared" "Soap" "Trees"
```

```
[49] "Wine" "Coffee" "Beans"
```

```
[52] "Fruits" "Grains" "Juices"
```

```
[55] "Mushrooms" "PetFood" "Tofu"
```

```
[58] "WildHarvested" "updateTime"
```

Attributes	Description	Type	Errors	Missing Values	Inconsistency	Note
FMID	uniquely identify a farmer market(FM)	number	N	0	N	range from 1000001 - 2000036
MarketName	full name of the market.	string	Not know	0	Y	
Website	Link to the farmer market.	URL/ string	Y	3458	Y	Can be vaildated
Facebook, Twitter, Youtube, Other Media	Link to / account name of medias	URL/string	Y	Y	Y	Can be vaildated
Street, city, county, state, zip	Location information	strings	Not know	Y	Y	Hard to test
Season Dates &Times	Open date & time in four seasons	Dates & times	Not know	Y	Y	8500+ missing except season 1
x, y	Longitude and latitude	number		29	N	Can be vaildated
location	Location type	Strings	Not know	5729	Y	No standard form
Credit, WIC, WICcash, SFMNP, SNAP	Payment Methods	Booleans	Not know	0	N	
Organic	Whether products are organic	Booleans	Not know	5070	N	
Bakedgoods-WildHarvested	Product types	Booleans	Not know	3023	N	
UpdatedTime	Latest updated time	Dates & times	Not know	0	Y	

1.3 data cleaning goals

- Free of error – clean wrong values
- Assure Completeness – clean empty values
- Ensure Consistency - standardization
- Provide Feedback for Improvements

1.4 Quality

- Accessibility: the data is accessible online

- Volume of Data: large data set
- Completeness: contain missing values
- Representation: easy to understand
- Consistency: data are not standardized
- Relevancy: attributes are relevant to their uses
- Timeliness: dataset is up to date and continuously updated
- Free of error: there are errors, for instance, some websites are invalid

1.5 Uses


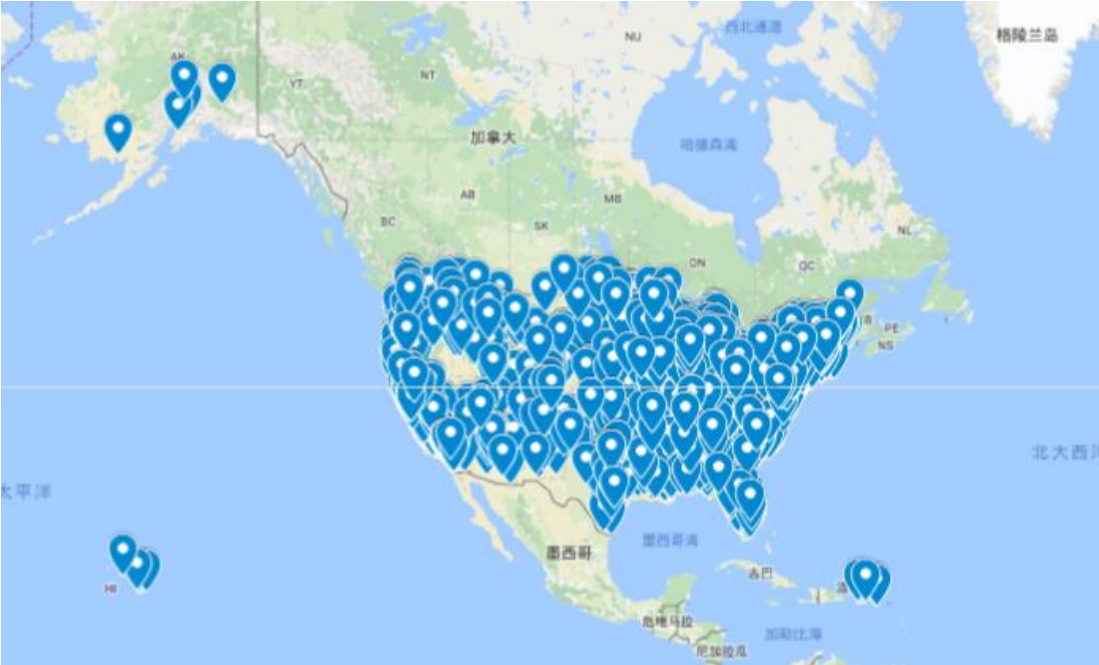
- Possible Uses:
 - a. Find nearby farmer markets (by longitude and latitude)
 - b. Find markets within a certain region level or the detailed address of a market (by city, county, state)
 - c. Search for which products are supported in a market or which markets sell certain product
 - d. Find which payment methods are supported in a market or which markets support certain payment methods
 - e. Find on which season/time a market is open or which markets open in a certain period
 - f. Find online media presentation of a certain market
- Uses without needs for cleaning
 - a. Looking for interesting social media contents of farmer markets
 - b. Looking for nearby farmer markets
 - c. Search for markets that contain certain products
- Uses can never be clean enough
 - a. Count how many markets that fits a certain condition
 - b. Study distributions of opening times

2 DC with OpenRefine (quantify narratives + supplemental info)

We deal with the dataset in the order of its attributes.

Attributes	Actions
FMID	Sort as numeric values
MarketName	Clustering
Website	Hard to refine
Facebook	Hard to refine
Twitter	Hard to refine
Youtube	Hard to refine
OtherMedia	Hard to refine
street	Reconcile to text

city	<p>Reconcile with cities in the United Stated, result is not good</p> <div><div><div>city: judgment</div><div>change invert reset</div></div><div>4 choices Sort by: name count</div><div>(blank) 39</div><div>matched 379</div><div>new 140</div><div>none 7292</div><div>exclude</div><div>Facet by choice counts</div></div> <div><div><div>city: best candidate's score</div><div>change reset</div></div><div></div><div>19.00 — 90.00</div><div><div><input checked="" type="checkbox"/> Numeric 5069</div><div><input type="checkbox"/> Non-numeric 0</div><div><input type="checkbox"/> Blank 0</div><div><input checked="" type="checkbox"/> Error 3038</div></div></div>
------	---

Season1Date Season1Time Season2Date Season2Time Season3Date Season3Time Season4Date Season4Time	Convert to date
X, y	<div>Scatterplot facet</div>  <div>Geo-mapped with google map</div> 
Location	Not really relevant

Credit WIC WICcash SFMNP SNAP	No need to clean
Organic	Clean null values and empty values
Bakedgoods, Cheese, Crafts, Flowers, Eggs, Seafood, Herbs, Vegetables, Honey, Jams, Maple, Meat, Nursery, Nuts, Plants, Poultry, Prepared, Soap, Trees, Wine, Coffee, Beans, Fruits, Grains, Juices, Mushrooms, PetFood, Tofu, WildHarvested	Remove empty values when analyzing product types.
updateTime	

3 Alternatives

3.1 R language

- Advantages:
 - Can remove/replace values with certain characteristics with a single command, without selecting every attribute once a time


```
3 df[df=="no" | df=="none" | df=="N/A" | df=="n/a" | df=="None" | df=="NA" | df=="-" ] <- NA
```
 - Can get statistics with single command


```
4 summary(df)
```
 - Can get street/city values from longitudes and latitudes

```

8 #find street values with longitudes and latitudes
9 which(is.na(df$street))
10 i=1
11 df$street[i] = stri_extract(revgeocode(c(df$x[i], df$y[i])), regex='[A,]*')
12 df$street[i]
13
8:50 (Top Level) R Script

```

```

Console -/
> which(is.na(df$street))
[1] 1 48 72 226 357 419 434 453 455 468 554 561 565 577 604 677 798 821
[19] 826 834 884 889 941 988 1110 1139 1234 1239 1243 1283 1328 1346 1360 1389 1400 1418
[37] 1438 1474 1512 1534 1543 1554 1563 1591 1594 1614 1653 1656 1673 1720 1722 1753 1899 1921
[55] 1982 2007 2009 2045 2078 2079 2095 2101 2109 2134 2222 2264 2266 2267 2268 2269 2270 2271
[73] 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2286 2287 2288 2289 2290 2291
[91] 2292 2293 2294 2295 2296 2297 2298 2299 2462 2497 2500 2505 2508 2524 2566 2618 2667 2677
[109] 2720 2741 2815 2862 2884 2906 3042 3073 3075 3122 3181 3185 3217 3223 3244 3260 3281 3321
[127] 3345 3346 3375 3415 3473 3492 3545 3575 3582 3637 3684 3731 3748 3758 3781 3801 3830 3852
[145] 3942 3953 4036 4062 4161 4182 4192 4288 4293 4297 4326 4345 4355 4380 4388 4463 4622 4623
[163] 4629 4688 4697 4702 4734 4786 4796 4807 4838 4904 4912 4920 4922 4957 4969 4977 4980 4981
[181] 5080 5200 5263 5293 5353 5354 5389 5391 5408 5410 5423 5483 5636 5684 5692 5704 5758 5803
[199] 5807 5833 5863 5891 5916 5943 5967 5976 5992 6085 6227 6233 6323 6333 6337 6359 6381 6382
[217] 6405 6431 6456 6473 6525 6531 6558 6559 6622 6689 6699 6715 6720 6901 6956 6996 7047 7062
[235] 7092 7094 7100 7122 7177 7178 7181 7184 7205 7257 7264 7323 7327 7370 7391 7464 7538 7566
[253] 7616 7671 7727 7728 7748 7755 7770 7810 7811 7816 7821 7859 7868 7882 7886 7919 7948 8044
[271] 8045 8076 8263 8275 8393 8412 8416 8444 8509 8605 8619 8631 8633 8636 8644
> i=1
> df$street[i] = stri_extract(revgeocode(c(df$x[i], df$y[i])), regex='[A,]*')
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?latlng=44.411013,-72.140305&sensor=false
> df$street[i]
[1] "124 Park St"

```

- Cleaning of URLs:

```

6 # check if URL exists
7 library(RCurl)
8 for (i in length(df$Website)){
9   if(url.exists(df$Website[i])== FALSE){
10     df$Website[i]<-NA}}
11 df$Website
12
13 for (i in length(df$Facebook)){
14   if(url.exists(df$Facebook[i])== FALSE){
15     df$Facebook[i]<-NA}}
16 summary(df$Facebook)

5 library(RCurl)
6 a={}
7 for i in df$MarketName:
8   if url.exists(df[i,3]) is TRUE:
9     df[i,3]=df[i,3]
10   if url.exists(df[i,3]) is FALSE:
11     df[i,3]=NA
12   print(df[,3])

```

- Seeing the geographic distance matrix between long/lat points (R explode =o=):

```

7 deg2rad <- function(deg) return(deg*pi/180)
8
9 gcd.hf <- function(long1, lat1, long2, lat2) {
10   R <- 6371 # Earth mean radius [km]
11   delta.long <- (long2 - long1)
12   delta.lat <- (lat2 - lat1)
13   a <- sin(delta.lat/2)^2 + cos(lat1) * cos(lat2) * sin(delta.long/2)^2
14   c <- 2 * asin(min(1,sqrt(a)))
15   d = (R * c)*1000
16   return(d) }
17
18 CalcDists <- function(longlats) {
19   name <- list(rownames(longlats), rownames(longlats))
20   n <- nrow(longlats)
21   z <- matrix(0, n, n, dimnames = name)
22   for (i in 1:n) {
23     for (j in 1:n) z[i, j] <- gcd.hf(long1 = deg2rad(longlats[i, 2]),
24                                     lat1 = deg2rad(longlats[i, 1]), long2 =
25                                     lat2 = deg2rad(longlats[j, 1]))
26   }
27   z <- as.dist(z)
28   return(z)
29 }
30 # E.g.
31 longlats <- data.frame(long = df$x, lat = df$y)
32 dists <- CalcDists(longlats)
33

```

- Disadvantages:
 - Can be slow with large data
 - Need coding technique (not new-user friendly)
 - Visualization not good

3.2 Python (only used to find near points because it is super slow in R)

```

from math import radians, cos, sin, asin, sqrt
import csv
import pandas
def main():
    data = pandas.read_csv("farmers-markets.csv", usecols=[0,20,21])
    sim_list = []
    for i in range(len(data.x)):
        for j in range(i, len(data.y)):
            if data.x[i]:
                temp_dis = haversine(data.x[i], data.y[i], data.x[j], data.y[j])
                if temp_dis < 0.1 and i != j:
                    sim_list.append((data.FMID[i], data.FMID[j]))
    print(sim_list)
    sim_data = pandas.DataFrame(sim_list)
    sim_data.to_csv('sim.csv')
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6367 * c
    return km
if __name__ == "__main__":
    main()

```

	A	B
286	1003445	1005574
287	1007693	1011997
288	1010236	1006120
289	1007771	1009450
290	1003881	1009848
291	1009910	1006324
292	1011764	1002246
293	1008429	1012771
294	1011499	1006550
295	1012325	1012329
296	1010478	1010339
297	1003769	1003768
298	1012016	1003473
299	1001794	1001796
300	1001794	1005795
301	1001796	1005795
302	1009270	1009074
303	1006893	1008931
304	1005969	1011301
305	1006422	1006423

There are 305 pairs of results.

- 3.3 Excel (used to view the CSV file and find values such as "no" "none" "N/A" "n/a" "None" "NA" "-")

4 Integrity constraints with SQLite

4.1 [column name] cannot be null

Enter SQL Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

```
SELECT FMID, x, y
FROM farmersmarkets
WHERE x IS NULL
OR y IS NULL;
```

Last Error: not an error

FMID	x	y
2000016		
2000017		
2000019		
2000020		
2000021		
2000022		
2000026		
2000028		
2000030		
2000032		
2000033		
2000034		
2000035		

Number of Rows Returned: 29 ET: 13 ms

4.2 X, y within US boundaries (outliers are from Hawaii and Alaska and Virgin Islands)

Structure | Browse & Search | Execute SQL | DB Settings | Export Wizard

Enter SQL Select | Data Manipulation | Create/Alter | Drop | ReIndex | PRAGMA

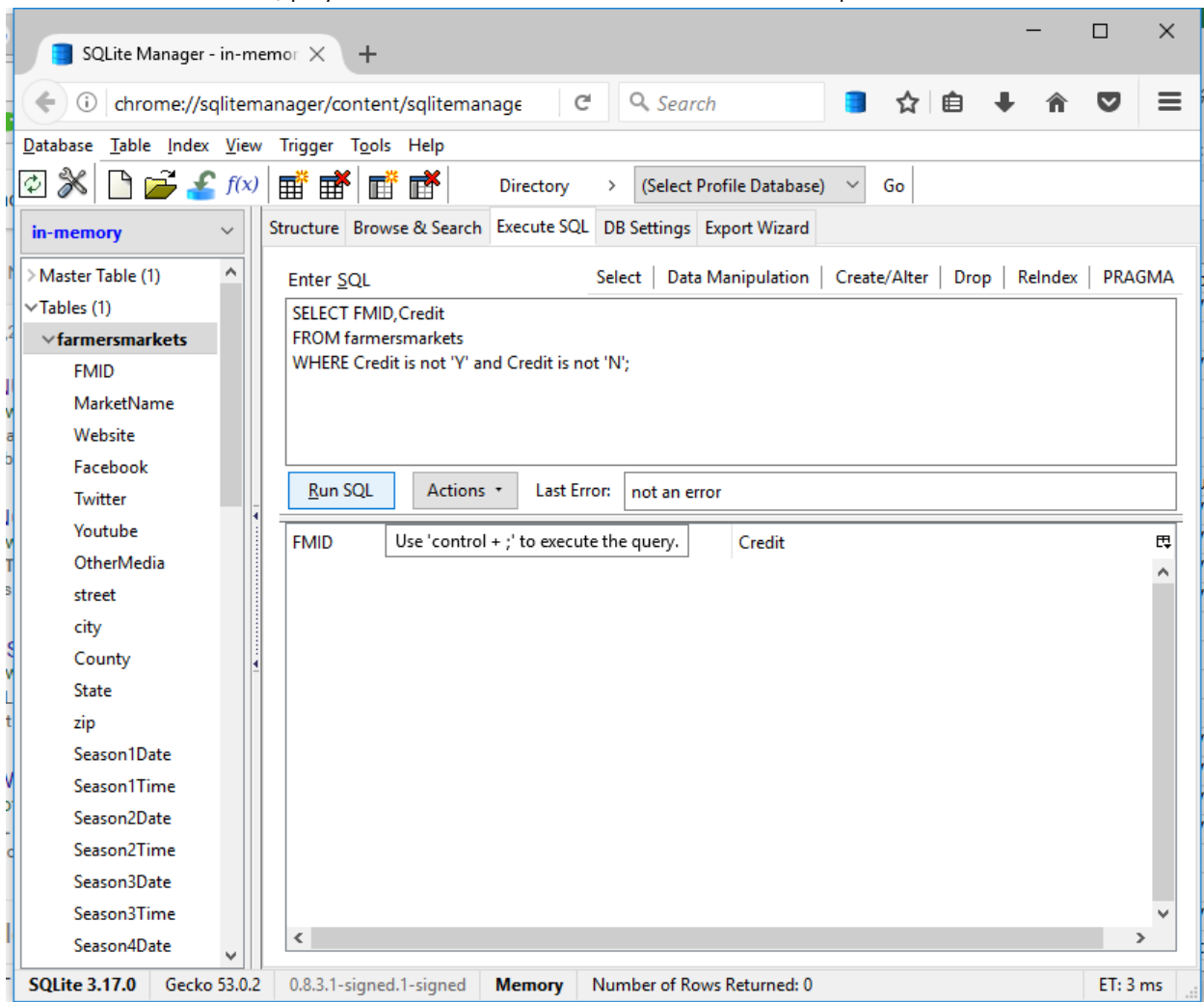
```
SELECT FMID, x, y, State
FROM farmersmarkets
WHERE x <= -124.7844079
OR x >= -66.9513812
OR y <= 24.7433195
OR y >= 49.3457868;
```

Last Error: not an error

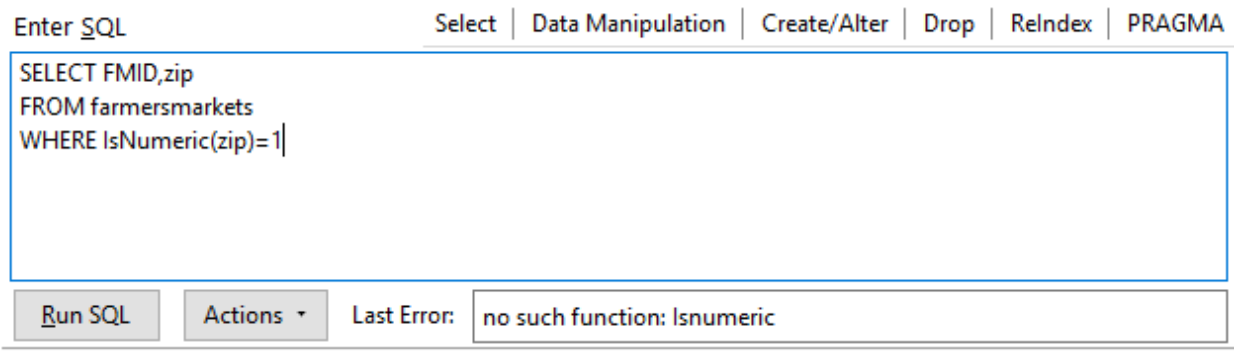
FMID	x	y	State
1000189	-148.01	64.8459	Alaska
1000199	-147.719593	64.844414	Alaska
1000636	-156.657	20.8491	Hawaii
1000982	-145.444	62.0879	Alaska
1001032	-159.527	21.9251	Hawaii
1001036	-149.868	61.1881	Alaska
1001127	-159.465	21.9081	Hawaii
1001265	-158.126	21.574	Hawaii
1001353	-157.862	21.3093	Hawaii
1001492	-151.261	60.555	Alaska
1001671	-155.88	19.4139	Hawaii

0.8.3.1-signed.1-signed **Memory** Number of Rows Returned: 182 ET: 7 ms

4.3 Product values/payment methods cannot be values except Y or N



4.4 Zip cannot be non-numeric values (have error)



5 Workflow model

5.1 what are the key inputs and outputs of your workflow?

Step 1

Input: database

Output: summary result

Step 2

Input: original data

Output: cleaned data (half cleaned)

Step 3

Input: integrity constraints

Output: result of check of ICs

