MongoDb Practice

Notes

In this section we will carry out practice working with MongoDb, using as a database the "Consumer Expenditure Survey" public-use data, available here[1]. In particular we have downloaded the data for 2015 from here[2]. We will look at only a small part of the massive datasets available there. You are encouranged to explore this data further.

The datasets we will process contain coded information. Detailed info on the files and how to use them are provided here[3]. In particular we will make use of the information provided on this file[4].

The first task would be to get the various data inserted into mongodb. First, we should clean up the database a bit. Log in to your database, and drop any collections you no longer need. After that, do:

```
db.repairDatabase()
```

Then log back out to do the following import.

We will download the data via curl, then use mongoimport (as usual you need to change the mongoimport line). We would ideally be able to download all quarters, but the 500MB storage limit does not allow us to.

```
curl https://raw.githubusercontent.com/skiadas/DataWranglingCourse/gh-pages/labs/fmli151.csv
mongoimport -h ds011168.mlab.com:11168 -d wranging -c fmli -u haris -p haris --file fmli151.c
```

We will later restrict to only the following variables, from the hundreds of variables included by default. But there is no easy way to do this in MongoDb directly, we will do it in Python when we later discuss Python's MongoDb interface.

For now the data we will look at is over a single quarter and it measures a large number of families.

Here is a list of the fields we will be using. You can look for their meaning in this document[5].

NEWID, HH_CU_Q, CUID, BLS_URBN, CUTENURE, FAM_SIZE, FAM_TYPE, NO_EARNR, NUM_AUTO, PERSLT18
PERSOT64, REGION, INCLASS, AGE_REF, EDUC_REF, MARITAL1, REF_RACE, SEX_REF, FINCATXM, FINCBTXM
INC_HRS1, INC_HRS2, INCWEEK1, INCWEEK2, OCCUCOD1, OCCUCOD2, OTHRINCM, WELFAREM, ROYESTXM
CREDITB, CREDYRB, IRAB, LIQUDYRB, LIQUIDB, OTHASTB, OTHLONB, STDNTYRB, STOCKB, STOCKYRB
STUDNTB, INT_HOME, TOTEXPPQ, FOODPQ, FDHOMEPQ, FDAWAYPQ, FDMAPPQ, ALCBEVPQ, HOUSPQ, SHELTPQ
OWNDWEPQ, MRTINTPQ, PROPTXPQ, MRPINSPQ, RENDWEPQ, UTILPQ, NTLGASPQ, ELCTRCPQ, ALLFULPQ
TELEPHPQ, WATRPSPQ, DOMSRVPQ, BBYDAYPQ, OTHHEXPQ, HOUSEQPQ, APPARPQ, TRANSPQ, HEALTHPQ
HLTHINPQ, MEDSRVPQ, PREDRGPQ, MEDSUPPQ, ENTERTPQ, PERSCAPQ, READPQ, EDUCAPQ, TOBACCPQ, TTOTALP

We will start with a simple question. The variable CUID contains a unique identifier for each "consumer unit". Ideally, since we have loaded data for only one quarter, each

[1] http://www.bls.gov/cex/pumd.htm
[2] http://www.bls.gov/cex/pumd_data.htm#csv
[3] http://www.bls.gov/cex/pumd_doc.htm#2015
[4] http://www.bls.gov/cex/2015/csxintvwdata.pdf
[5] http://www.bls.gov/cex/2015/csxintvwdata.pdf

document we have should correspond to a distinct consumer unit. We will count how many times each consumer unit appears and see if any of them appears more than once. We will do it via two aggregate methods. mapReduce is not suitable for this task.

```
db.fmli.count()          // How many documents total
// The following tries to find any CUIDs that appear more than once.
// It should result in no matches
db.fmli.aggregate([
  { $group: {
    _id: "$CUID",
    count: { $sum: 1 }
  }},
  { $match: { count: { $gt: 1 }}}
])
// The following adds all the CUIDs, counting each only once.
// It should return the same answer as db.fmli.count()
db.fmli.aggregate([
  { $group: {
    _id: "$CUID"
  }},
  { $group: {
    _id: "",
    count: { $sum: 1 }
  }}
])
```

Next we will look into the education levels of the primary contact on each unit. Here's what the key numbers for that variable mean:

```
00 Never attended school
10 First through eighth grade
11 Ninth through twelfth grade (no H.S. diploma)
12 High school graduate
13 Some college, less than college graduate
14 Associate's degree (occupational/vocational or academic) 15 Bachelor's degree
16 Master's degree, (professional/Doctorate degree)*
```

And here's the code in aggregate and map-reduce form. we will compute counts only in the aggregate form, and both.

```
db.fmli.aggregate([
  { $group: {
    _id: "$EDUC_REF",
    count: { $sum: 1 }
  }}
])
// The mapReduce way
// First we compute total number of documents that have an 'EDUC_REF' entry.
var total = db.fmli.count({ EDUC_REF: { $gte: 0 }})
db.fmli.mapReduce(
  function() {
    emit(this.EDUC_REF, 1);
  },
  function(key, values) {
    return Array.sum(values);
  },
```

```
  {
    out: { inline: 1 },
    finalize: function(key, reducedValue) {
      return { count: reducedValue, percent: reducedValue/total };
    },
    scope: { total: total }
  }
).results
```

Looking at the results, make some statements about the education level of the respondents.

Practice: Try the same to look at the racial breakdown of the main respondents, which is given by the variable REF_RACE.

Practice: Find out what happened to the hispanic population. It does not really appear in the race data.

Next we will look at some numerical data, like the total income before taxes, coded in the field FINCBTXM. We start with computing the average income. Here's how that might work:

```
db.fmli.aggregate([
  { $group: {
    _id: "",
    income: { $avg: "$FINCBTXM" }
  }}
])
```

Now let's find the average income per education level. Compare with what we expect, that the higher the education level the higher the income, on average.

```
db.fmli.aggregate([
  { $group: {
    _id: "$EDUC_REF",
    income: { $avg: "$FINCBTXM" }
  }},
  { $sort: { _id: 1 }}
])
```

Practice: We would like to compare the average incomes per education level, but now separately for each of the four regions given by the variable REGION (Northeast, Midwest, South, West). You will need to use a more complicated _id for the $group. Sort the results by education level then by region, so that you can look at the results of the same level close to each other. Ansewr the following question: Are there instances that within the same education level, the income varies quite a lot between regions?

Practice: Continuing from the previous problem, add alongside the average income values a count of the number of cases in each group.

We will now try to compare the education levels between men and women. We will use the SEX_REF variable for that. We start with counting within each group:

```
db.fmli.aggregate([
  { $group: {
    _id: { level: "$EDUC_REF", sex: "$SEX_REF" },
```

```
    count: { $sum: 1 }
  }},
  { $sort: { "_id.level": 1, "_id.sex": 1 }}
])
```

Question: Which education-level categories have more women respondents than men respondents?