Generating Census Tract

Research using residential address needs more precise segments than zip code

Census tract data can be used once we convert address to latitude, longitude

Desire open source solution, minimal user effort, short run time, accuracy

Alternatives Considered

- Web sites generate lat/lng from CSV files
  - Questionable procedures (violate Google API)
  - Quotas on batch jobs
  - Too much effort, not very fast
- ArcMap spatial joining
  - No Linux solution
  - Several steps, must learn macros
  - Vanderbilt license and expertise available

Process

- Grab data from MySql table
- Create Sqlite file with Census data
- Generate lat/lng using Ruby script
- Build census tract polygons and create spatial join with R script
- Return data back to MySql

MySql interface

- Ruby script
  - sequel gem
    - http://sequel.rubyforge.org/
  - Config file
    - Credentials
    - Required columns (id, resAddress)
    - Selection criteria

Census data

- 2011 Tennessee data
  - ftp://ftp2.census.gov/geo/tiger/TIGER2011/
  - ADDR, EDGES, FEATNAMES
- Sqlite database
  - Ruby geocoder gem
    - https://github.com/geocommons/geocoder

```
./tiger_import /opt/tiger/tenn11.db /opt/tiger/11_TENN
../bin/rebuild_metaphones /opt/tiger/tenn11.db
./build_indexes /opt/tiger/tenn11.db
```

Generate latitude & longitude

- Ruby script
  - Geocoder/US gem

```
db = Geocoder::US::Database.new("/opt/geocoder/tenn11.db")
db2 = Geocoder::US::Database.new("/opt/geocoder/nontn11.db")
coded = db.geocode(line.join(', '))
if score >= 0.8
  fh1.write '"'+output.join('", "')+'"'+"\n"
else
  fh2.write '"'+output.join('", "')+'"'+"\n"
end
```

- Example

```
db.geocode("1420 State Highway 96 N, Fairview, TN, 37062")
=> [{:lat=>36.006971, :score=>0.805, :zip=>"37062", :city=>"Fairview", :number=>"1074",
:street=>"State Hwy 96 N", :prenum=>"", :state=>"TN", :fips_county=>"47059", :lon=>-
87.141294, :precision=>:street}]
```

Census Tract Polygons

- R script
  - maptools library
    - http://cran.r-project.org/web/packages/maptools/index.html
  - rgeos library
    - http://cran.r-project.org/web/packages/rgeos/index.html
    - Requires GEOS engine
      - http://trac.osgeo.org/geos/
  - Census tract data
    - ftp://ftp2.census.gov/geo/tiger/TIGER2011/TRACT/tl_2011_47_tract.zip

Census Tract Polygons

- Example

```
tt <- readShapePoly(file.path("", "opt", "geocoder", "tl_2011_47_tract.shp"))
ct <- numeric(length(tt@polygons))
for(i in seq_along(ct)) {
  coord <- tt@polygons[i][[1]]@Polygons[[1]]@coords
  ct[i] <- point.in.polygon(-87.14129, 36.00697, coord[,1], coord[,2])
}
which(ct==1)
[1] 1239
tt@data$GEOID[ct==1]
[1] 47187050503
```

Results

- Open source
  - Ruby, R, MySql, Sqlite, GEOS, U.S. Census
- User effort
  - Modify simple config file
- Speed
  - Query with 1637 results: < 90 seconds
- Accuracy
  - 1230 with lat/lng (407 without)
  - 1160 with census tract (70 without)
  - Many things to investigate (year, state, multiple polygons)

```
# R code
# Cole Beck, Vanderbilt University, Dept of Biostatistics
# May 10, 2012
library(maptools) #rgeos

# let ruby generate lat/lon -- returns CSV file name
infile <- system("env -i /usr/bin/ruby geocode_addr.rb", intern=TRUE)

addr <- read.csv(infile, stringsAsFactors=FALSE, header=FALSE)
names(addr) <- c('id','number','street','city','state','zip','lat','lon','score')
lat.points <- addr$lat
lon.points <- addr$lon

# TN census tract shape file
tt <- readShapePoly(file.path("", "opt", "geocoder", "tl_2011_47_tract.shp"))

n <- length(tt@polygons)
addrPoints <- matrix(0, nrow=n, ncol=length(lat.points))
for(i in seq(n)) {
  coord <- tt@polygons[i][[1]]@Polygons[[1]]@coords
  addrPoints[i,] <- point.in.polygon(lon.points, lat.points, coord[,1], coord[,2])
}

# colSums(addrPoints)

gids <- as.character(tt@data$GEOID[apply(addrPoints, MARGIN=2, FUN=function(i) { which(i==1)[1] })])

addr$geoid <- gids
addr$TRACTCE <- as.numeric(substr(gids, 6, 11))
write.csv(addr, sprintf("out_%s", basename(infile)), row.names=FALSE)
```