Randall Shane, PhD
Stanford University – SCPD
STATS290
15 March 2011

Address Matching Package in R
Package Name: geoPlot

PACKAGE DESCTIPTION:

This package simplifies the process of performing address matching without the necessity of excessive parsing of all the permutations of address data. This is accomplished through two algorithms. The first is geocoding the address and applying the Haversine formula to determine global distance. The second is a parsing algorithm that utilizes the numerical portion of the address along with unique locale (zip or postal code) to determine similarities. There are 2 dependencies for this package. They are the XML library if you intend to use XML data and the RgoogleMaps library which is necessary only for the map visualization, not for the geocoding.

There are two main functions to be applied and both are predicated on the assumption that there is a base list of addresses to match against and an evaluate list of addresses to evaluate against the base list. Functions were written import delimited files and XML data containing a specific structure of address data. Both types must contain the fields; id, address, city, state, zip.

GEOCODING FUNCTION:

The first function geocodes the address on both lists thereby obtaining the GPS location. The Google Maps API was used in performing the geocoding and it should be noted there are other options available but due to time constraints, only the Google Maps API was implemented. The data is subsequently compared using the Haversine formula to obtain the distances and after applying a filter. The filter that was applied concatenated the degree number without decimals from the latitude and longitude and performs a many to many join from the match list to the base list. The result here is all possible locations that are within one degree of latitude and longitude of each other. In terms of distance, this yields all locations within 69 miles or 110 kilometers proximity. The haversine formula is then used to determine the exact distance and the data provided in both miles and kilometers in both a data frame and a delimited file. A decision was made to provide all of the data units generated during the process. This includes a URL string that is rendered for the API, raw data returned and also the generated calculations. For those entries that are not able to be resolved, a latitude and longitude of zero is returned and a column named 'geoValidate' was generated with a logical value containing true indicating the geocoding was successful and false if it was not.

Two graphical outputs are available which are additions to the proposed package. The first uses the RgoogleMaps library, locates a center point of the geocoded range of addresses, pulls the map (satellite or mobile) and then plots a red circle indicating the data from the base list and the match list overlays the map visually with a blue triangle. During the course of development and specifically as this development finished, versions of R migrated 2.12.2. There is a bug in the 'rgdal' support package for RgoogleMaps for 2.12.2 where this visualization will only function on Windows. This is a known existing issue and is not yet solved for OSX or UNIX. The second visualization is a basic plot of the valid geocoded data generated through the plot function with latitude on the x axis and longitude on the y axis. The same convention of a red circle from the base and a blue triangle from the match list is kept. The overlap is apparent to the user visually.

ADDRESS PARSING FUNCTION:

The second function performs matching without an external data requirement by isolating and concatenating the postal code and the numerical portion of the address into a match field. The match field is compared between the base list and the evaluate list to provide like data units. The output contains address data from both lists organized into a data frame and also as a delimited file.

CHANGES FROM ORIGINAL PROPOSAL:

There are 2 changes from the original proposal.

The first change is the addition of the visualization portion of the geocoding algorithm including the basicPlot and geoPlot functions. The second change is an omission of a percentage from the parsed address algorithm. Several attempts were made to yield an intelligent result and although best efforts (like soundex) were implemented, the results were not reliable enough to be deemed useful.

TEST SCRIPT:

A test script named 'Test_Script.R' is included with this package under the 'tests' sub directory along with 2 data files, 'EPA_List.csv' and 'Vendors.xml'. This is a test of the functionality of this package by comparing the addresses of proposed vendors to a subset of the Environmental Protection Agency's list of known potential contamination sites which can be found at http://www.epa.gov/tri/tridata/indexdata.htm.

A narrative of the test script follows.

Environment Section: Start by removing all objects, setting the working directory and loading the packages and libraries for execution.

```
rm(list=ls(all=TRUE))
#setwd("~/R_Working")  # Windows
#setwd("~/R_Working")  # Mac
install.packages("XML", dependencies=TRUE)
install.packages("RgoogleMaps", dependencies=TRUE)
install.packages("geoPlot", dependencies=TRUE)  #This is this package
library("geoPlot")
```

Import Section: This section imports the delimited 'EPA_List.csv' file as the base list and the 'Vendors.xml' file as the match list. Please note the XML library is needed for the xml load. The EPA list is approximately 1146 entries and the vendors file is 35 entries.

```
# Import Testing
baseData <- csvImport("EPA List.csv")
library("XML")
matchData <- xmlImport("Vendors.xml")
```

The following section is commented out but saves the loaded files as *.rda files which are found in the data subdirectory of this package. The load functions exist here as well.

```
#save(baseData, file="baseData.rda")
#save(matchData, file="matchData.rda")
#load("baseData.rda")
#load("matchData.rda")
```

Geocoding Section: The following section performs the geocoding and creates the baseGeoCoded, matchGeoCoded and geoComplete data frames. Please note that the geoComplete data frame is also saved in the working directory as the result file named 'geoOutput.csv'. The geocoding process consistently takes approximately 3 minutes for the base list of 1146 entries and under a minute for the match list of 35 entries. This can be connection dependent.

```
# Geocoding Section
matchGeoCoded <- geoCode(matchData)
baseGeoCoded <- geoCode(baseData)
geoComplete <- geoCompare(matchGeoCoded, baseGeoCoded)
```

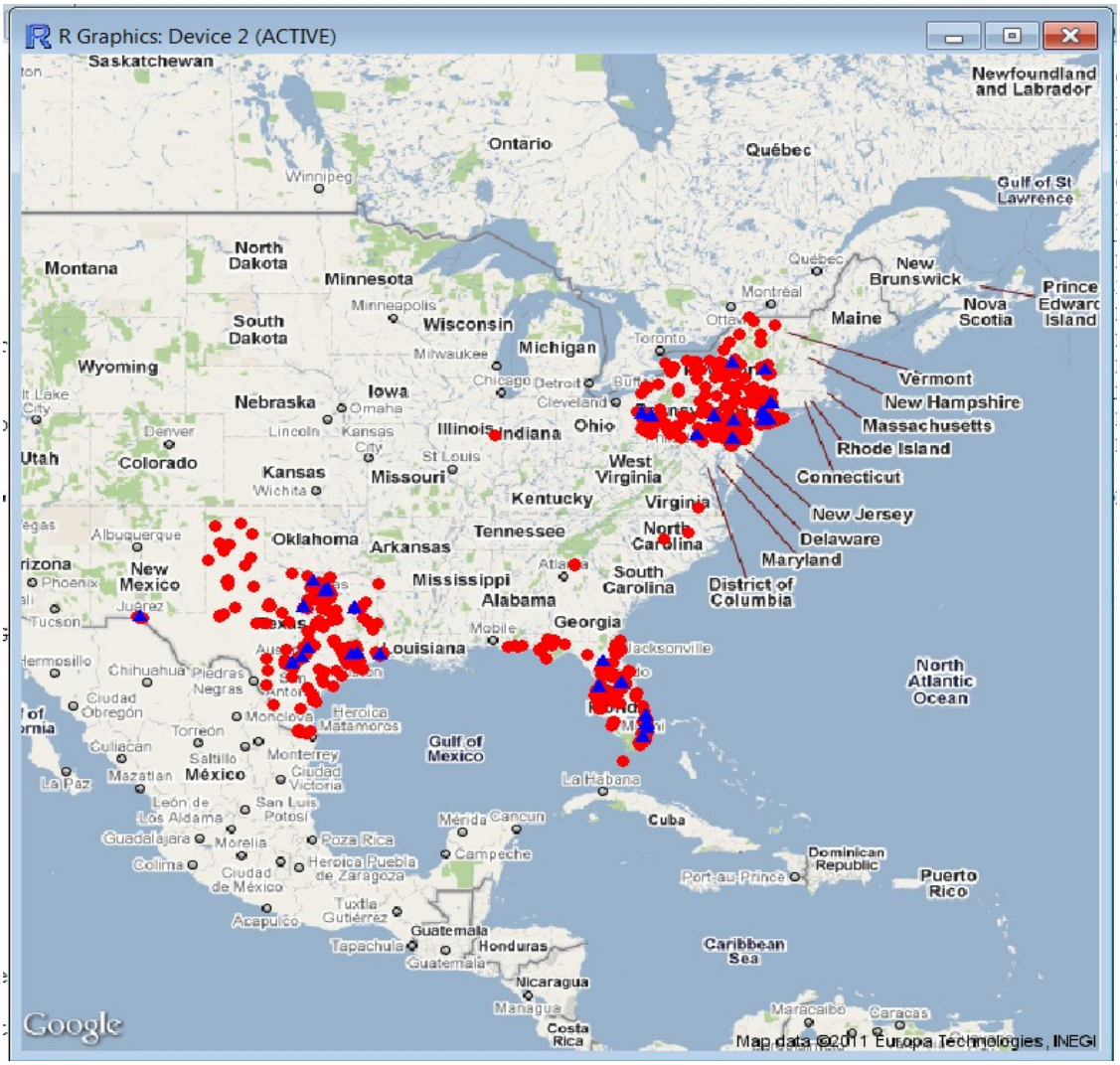
The following section is commented out but saves the loaded files as *.rda files which are found in the data subdirectory of this package. The load functions exist here as well.

```
#save(baseGeoCoded, file="baseGeoCoded.rda")
#save(matchGeoCoded, file="matchGeoCoded.rda")
#save(geoComplete, file="geoComplete.rda")
#load("baseGeoCoded.rda")
#load("matchGeoCoded.rda")
#load("geoComplete.rda")
```

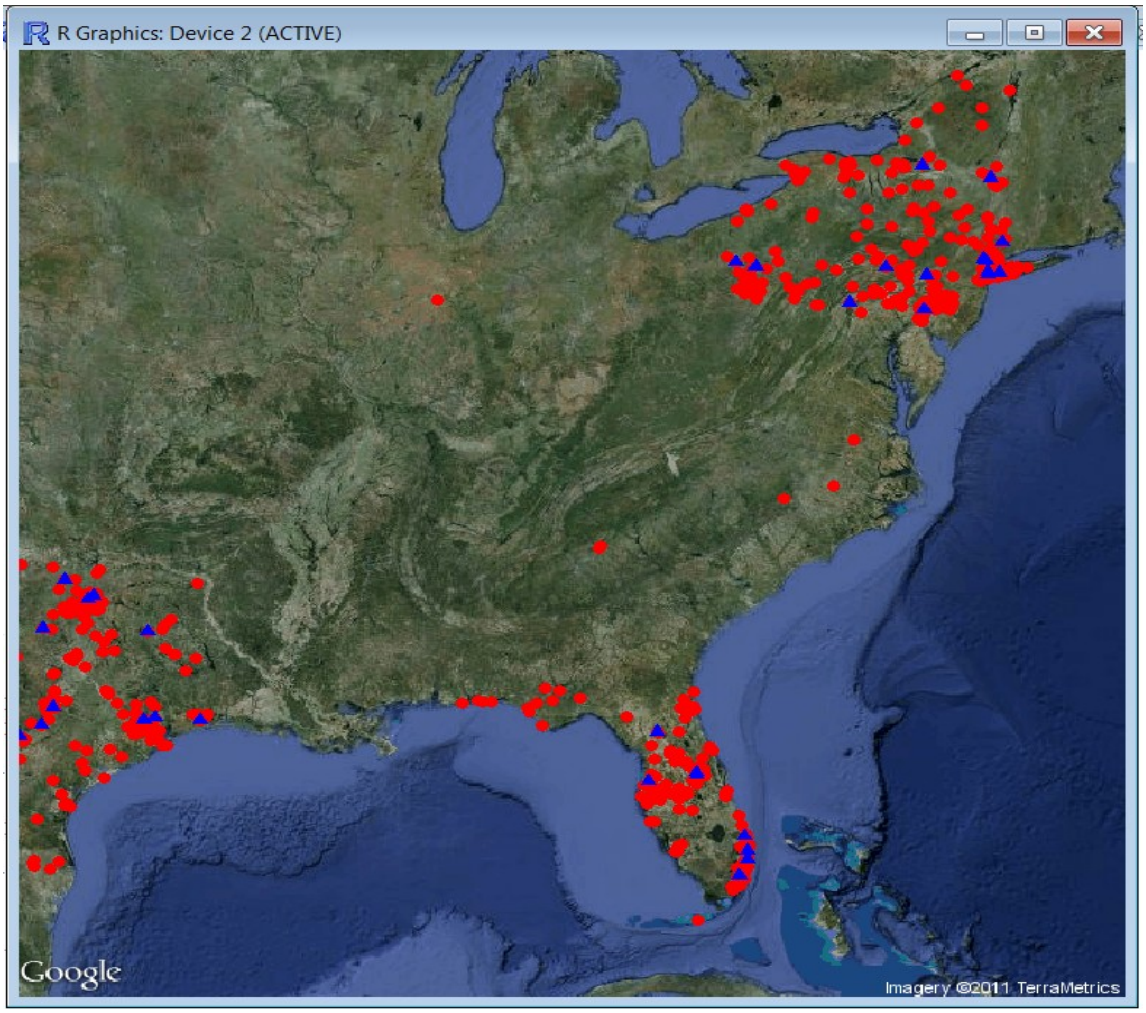
Visualization Section: There are two functions with three possible graphics. The first uses the RgoogleMaps package and will produce plots on a mobile or a satellite map. Both are below and the third is the output of the basicPlot function. In all three, the red circle indicates the base list and the blue triangle indicates the match list.

```
# Visualization Section
library("RgoogleMaps")
```

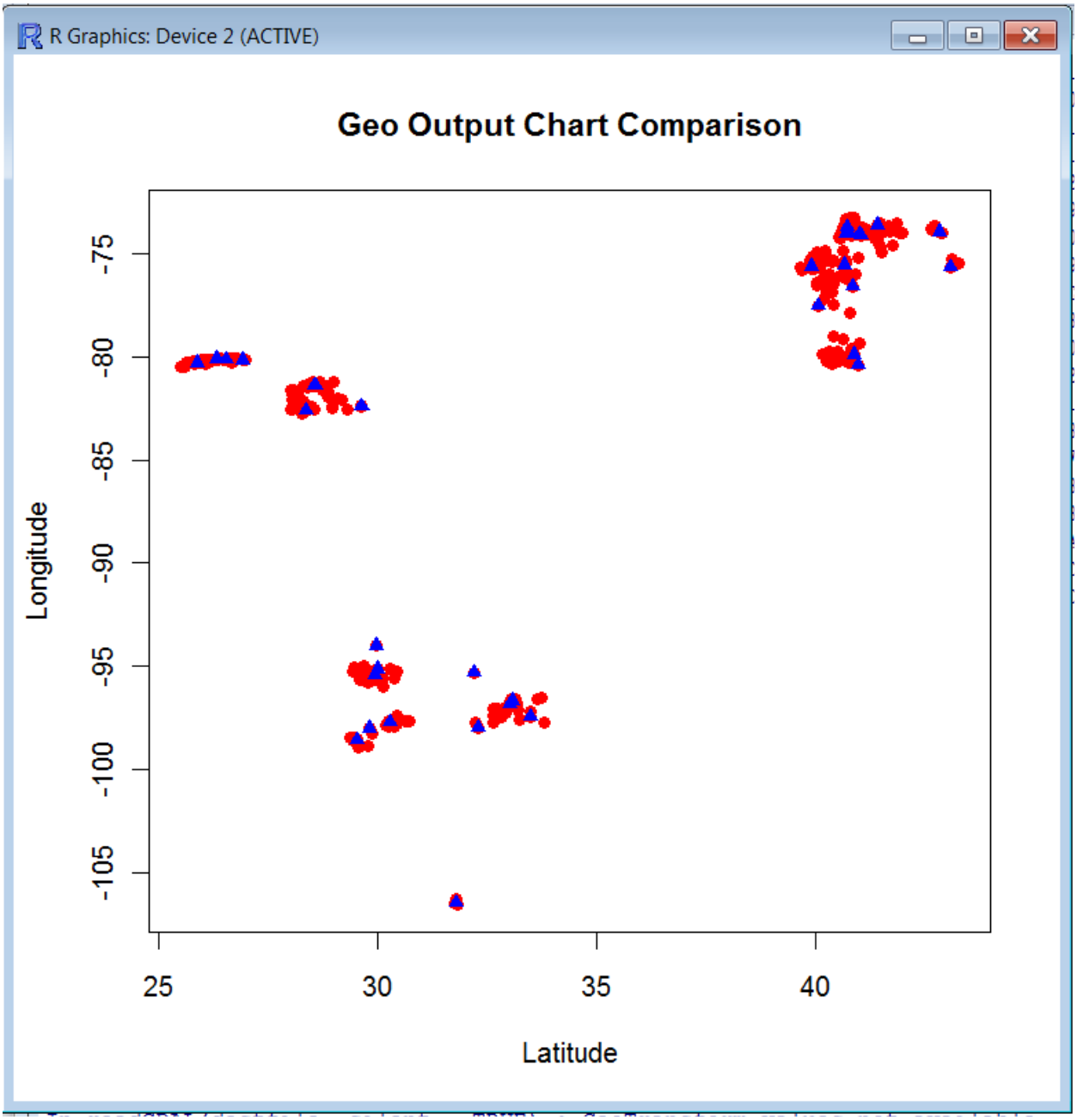
```
geoPlot(baseGeoCoded, matchGeoCoded, zoom=4, maptype="mobile")
```



```
geoPlot(baseGeoCoded, matchGeoCoded, zoom=5, maptype="satellite")
```



basicPlot(geoComplete)



Address Parsing Section: This section creates parses the address fields, creates the match field and performs the comparison. Please note that the result data frame is also saved as a delimited file named parsedMatches.csv in the working directory.

```
# Address Parsing Section
baseParsed <- addressParse(baseData)
matchParsed <- addressParse(matchData)
parsedMatches <- parseMatch(matchParsed, baseParsed)
```

The following section is commented out but saves the loaded files as *.rda files which are found in the data subdirectory of this package. The load functions exist here as well.

```
#save(baseParsed, file="baseParsed.rda")
#save(matchParsed, file="matchParsed.rda")
#load("baseParsed.rda")
#load("matchParsed.rda")
```

That is the end of the execution run. This package was the product of many hours of work and is functional and will be used in its current format. Future additions are already planned including another geocoding API and further visualization. If I had more time, there would be more content!

Please feel free to forward any additional questions to RShane@BaseXVI.com. Thank you.

Randall Shane, PhD