Zillow research

<http://blog.revolutionanalytics.com/2015/08/how-r-is-used-at-zillow-to-estimate-housing-values.html>

# Exploring census data, home value and school rating data with R

I wanted to use R to explore how to access and visualize census data, home value data and school rating data using Indianapolis metro area as an example. My learning objectives here are to learn how to use R to:

* Make choropleth maps at zip code level
* Process data in XML or JSON format using API provided by sites (here I chose [Zillow](http://www.zillow.com/howto/api/GetDemographics.htm) and [education.com](http://www.education.com/schoolfinder/tools/webservice/) as an example)

## Getting zip code data and shapefiles

# set working dir

setwd("~/notesofdabbler/Rspace/realEstate")

# load libraries

library(XML)

library(RCurl)

## Loading required package: bitops

library(stringr)

library(ggplot2)

library(maptools)

## Loading required package: sp

## Checking rgeos availability: TRUE

library(rgeos)

## rgeos version: 0.3-2, (SVN revision 413M)

## GEOS runtime version: 3.4.2-CAPI-1.8.2 r3921

## Polygon checking: TRUE

library(ggmap)

library(plyr)

library(RJSONIO)

I first extracted the zipcodes in the Indianapolis area by filtering the zip-county mapping for US using [counties](http://geology.com/county-map/indiana.shtml) in the Indianapolis area.

The dataset with zip-county mapping is created using another [code](http://notesofdabbler.bitbucket.org/2013_12_censusBlog/USgeodata_wdesc.html)

# get zip-city-county-state map, this is loaded into dataset zipMap2

load("zipCityCountyStateMap.Rda")

head(zipMap2)

## STATE COUNTY ZCTA5 city ctyname stname

## 1 01 001 36006 Billingsley Autauga County, AL AL

## 2 01 001 36749 Jones Autauga County, AL AL

## 3 01 001 36003 Autaugaville Autauga County, AL AL

## 4 01 001 36067 Prattville Autauga County, AL AL

## 5 01 001 36051 Marbury Autauga County, AL AL

## 6 01 001 36066 Prattville Autauga County, AL AL

# list of counties in the greater Indianapolis area

cntyList=c("Marion County, IN",

"Hamilton County, IN",

"Boone County, IN",

"Hendricks County, IN",

"Johnson County, IN",

"Shelby County, IN",

"Hancock County, IN")

# get list of zips in the greater Indianapolis area

zipIndy=zipMap2[zipMap2$ctyname %in% cntyList,]

# remove duplicates (due to a zip in multiple counties)

zipIndy2=zipIndy[!duplicated(zipIndy$ZCTA5),]

head(zipIndy2)

## STATE COUNTY ZCTA5 city ctyname stname

## 10309 18 011 46069 Sheridan Boone County, IN IN

## 10310 18 011 46035 Colfax Boone County, IN IN

## 10311 18 011 46071 Thorntown Boone County, IN IN

## 10312 18 011 47968 New Ross Boone County, IN IN

## 10313 18 011 46268 Indianapolis Boone County, IN IN

## 10314 18 011 46052 Lebanon Boone County, IN IN

I wanted to get zip code level shape files to be able to make choropleth maps. The census site has [zip code level shape files](http://www.census.gov/cgi-bin/geo/shapefiles2010/layers.cgi) for different states. I found the following [tutorial](http://rpubs.com/m_dev/Intro-to-Spatial-Data-and-ggplot2) on spatial data and ggplot2 very helpful in processing and mapping shape files.

# get shape file of zips in Indiana

# http://www.census.gov/cgi-bin/geo/shapefiles2010/layers.cgi

zipShp=readShapePoly("tl\_2010\_18\_zcta510/tl\_2010\_18\_zcta510.shp")

zipShp2=fortify(zipShp,region="ZCTA5CE10")

# retain zips in Indy area

zipShp3=zipShp2[zipShp2$id %in% zipIndy$ZCTA5,]

The plot below shows the zip codes overlaid on static google map of Indianapolis area obtained using [ggmap](http://cran.r-project.org/web/packages/ggmap/index.html) package

x=get\_googlemap(center="indianapolis",maptype=c("roadmap"))

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=indianapolis&zoom=10&size=%20640x640&maptype=roadmap&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=indianapolis&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

p=ggmap(x)

p=p+geom\_polygon(data=zipShp3,aes(x=long,y=lat,group=id),fill="blue",color="black",alpha=0.2)

print(p)

## Getting median house price data from Zillow zlogo

Disclaimer: This is a one off use of Zillow API for exploratory purposes. I believe I am in compliance with their [*terms of use*](http://www.zillow.com/howto/api/APITerms.htm). However, if somebody notices a violation of their terms of use, please let me know and I will remove this material.

I used [Zillow getDemographics API](http://www.zillow.com/howto/api/GetDemographics.htm) to extract median household prices by zipcode. To use this API, you need to get a API key. The following [overview](http://www.zillow.com/howto/api/APIOverview.htm) of the Zillow API also has a section on how to get a API key.

To extract data for a zip code, the url that needs to be used is of the form:

http://www.zillow.com/webservice/GetDemographics.htm?zws-id=<ZWSID>&zip=<zipcode>

The result is in XML format and a sample is listed [here](http://www.zillow.com/howto/api/GetDemographics.htm). I am only extracting the median listing price and median value per square feet. The code below uses XML package for processing the API data so that it can be visualized.

zwsid="yourAPIkey"

# Get demographic data for each zipcode in Indianapolis area with the API and process it to plot it on a map

# List of zips for which XML file with Zillow demographic data is to be extracted

zipList=zipIndy2$ZCTA5

zdemodata=list(zip=character(),medListPrice=numeric(),medValSqFt=numeric())

for (i in 1:length(zipList)) {

url=paste("http://www.zillow.com/webservice/GetDemographics.htm?zws-id=",zwsid,"&zip=",zipList[i],sep="")

x=xmlInternalTreeParse(url)

zdemodata$zip[i]=zipList[i]

x2=xpathApply(x,"//table[name = 'Affordability Data']/data/attribute[name = 'Median List Price']/values/zip/value",xmlValue)

zdemodata$medListPrice[i]=x2[[1]][1]

x3=xpathApply(x,"//table[name = 'Affordability Data']/data/attribute[name = 'Median Value Per Sq Ft']/values/zip/value",xmlValue)

zdemodata$medValSqFt[i]=x3[[1]][1]

}

zdemodata2=data.frame(zdemodata,stringsAsFactors=FALSE)

zdemodata2$medListPrice=as.numeric(zdemodata2$medListPrice)

zdemodata2$medValSqFt=as.numeric(zdemodata2$medValSqFt)

head(zdemodata2)

## zip medListPrice medValSqFt

## 1 46069 119900 73

## 2 46035 59000 43

## 3 46071 129900 78

## 4 47968 69900 73

## 5 46268 104900 68

## 6 46052 119900 84

The values of median price and median value per square feet are examined to choose appropriate buckets.

summary(zdemodata2)

## zip medListPrice medValSqFt

## Length:104 Min. : 0 Min. : 0.0

## Class :character 1st Qu.: 98725 1st Qu.: 65.0

## Mode :character Median :127100 Median : 77.0

## Mean :136116 Mean : 73.8

## 3rd Qu.:174000 3rd Qu.: 88.2

## Max. :365000 Max. :193.0

zdemodata2$medListPriceLvl=cut(zdemodata2$medListPrice,

breaks=c(0,100000,200000,300000,350000,400000),

labels=c("<100K","100-200K","200-300K","300-350K",">350K"))

zdemodata2$medValSqFtLvl=cut(zdemodata2$medValSqFt,

breaks=c(0,50,75,100,125,200),

labels=c("<50","50-75","75-100","100-125",">125"))

#qc

ddply(zdemodata2,c("medListPriceLvl"),summarize,minval=min(medListPrice),maxval=max(medListPrice))

## medListPriceLvl minval maxval

## 1 <100K 31900 99900

## 2 100-200K 101500 199700

## 3 200-300K 205000 269900

## 4 300-350K 325000 325000

## 5 >350K 355000 365000

## 6 <NA> 0 0

ddply(zdemodata2,c("medValSqFtLvl"),summarize,minval=min(medValSqFt),

maxval=max(medValSqFt))

## medValSqFtLvl minval maxval

## 1 <50 30 50

## 2 50-75 54 75

## 3 75-100 76 99

## 4 100-125 102 124

## 5 >125 127 193

## 6 <NA> 0 0

# keep only zips for which median list price data is available and remove duplicated zips

zdemodata2=zdemodata2[zdemodata2$medListPrice != 0,]

zdemodata2=zdemodata2[!duplicated(zdemodata2$zip),]

head(zdemodata2)

## zip medListPrice medValSqFt medListPriceLvl medValSqFtLvl

## 1 46069 119900 73 100-200K 50-75

## 2 46035 59000 43 <100K <50

## 3 46071 129900 78 100-200K 75-100

## 4 47968 69900 73 <100K 50-75

## 5 46268 104900 68 100-200K 50-75

## 6 46052 119900 84 100-200K 75-100

## Get US census data on median income and age

US census data can be extracted using a API. For this also you need a API key which can be obtained [here](http://www.census.gov/developers/tos/key_request.html).

# API key for census data

APIkey="yourAPIkey"

The url form to retrieve data and the documentation about data is listed in the US census site [here](http://www.census.gov/developers/data/). The result is returned in JSON format and processed using RJSONIO package. The median age is extracted from US census data and median household income is extracted from ACS (American Community Survey) data (that is part of US census site). There are also two packages [UScensus2010](http://cran.r-project.org/web/packages/UScensus2010/) and [acs](http://cran.r-project.org/web/packages/acs/). I still tried to directly access data in this instance to learn how to extract using API and parse JSON files.

Extract median age data

# state code (indiana)

state=18

# function to retrieve data from 2010 US census data

getCensusData=function(APIkey,state,fieldnm){

resURL=paste("http://api.census.gov/data/2010/sf1?get=",fieldnm,

"&for=zip+code+tabulation+area:\*&in=state:",state,"&key=",

APIkey,sep="")

dfJSON=fromJSON(resURL)

dfJSON=dfJSON[2:length(dfJSON)]

dfJSON\_zip=sapply(dfJSON,function(x) x[3])

dfJSON\_val=sapply(dfJSON,function(x) x[1])

df=data.frame(dfJSON\_zip,as.numeric(dfJSON\_val))

names(df)=c("zip","val")

return(df)

}

# get median age from US census 2010 for a state

fieldnm="P0130001"

dfage=getCensusData(APIkey,state,fieldnm)

names(dfage)=c("zip","medAge")

head(dfage)

## zip medAge

## 1 45053 44.5

## 2 46001 42.2

## 3 46011 43.9

## 4 46012 40.2

## 5 46013 41.4

## 6 46016 34.2

Extract median income data

# Get median income by ZCTA from 2011 ACS data (this is for all states)

fieldnm="B19013\_001E"

resURL=paste("http://api.census.gov/data/2011/acs5?get=",fieldnm,"&for=zip+code+tabulation+area:\*&key=",

APIkey,sep="")

dfInc=fromJSON(resURL)

dfInc=dfInc[2:length(dfInc)]

dfInc\_zip=as.character(sapply(dfInc,function(x) x[2]))

dfInc\_medinc=as.character(sapply(dfInc,function(x) x[1]))

dfInc2=data.frame(dfInc\_zip,as.numeric(dfInc\_medinc))

## Warning: NAs introduced by coercion

names(dfInc2)=c("zip","medInc")

dfInc2=dfInc2[!is.na(dfInc2$medInc),]

head(dfInc2)

## zip medInc

## 1 01001 59453

## 2 01002 54395

## 4 01005 74167

## 5 01007 75502

## 6 01008 78563

## 7 01009 46625

Next I merge the zillow data extracted previously with US census data

zdata=merge(zdemodata2,dfage,by=c("zip"),all.x=TRUE)

zdata=merge(zdata,dfInc2,by=c("zip"),all.x=TRUE)

zdata$medAgeLvl=cut(zdata$medAge,

breaks=c(0,30,35,40,45,50),

labels=c("<30","30-35","35-40","40-45",">45"))

zdata$medIncLvl=cut(zdata$medInc,

breaks=c(0,50000,75000,100000,120000),

labels=c("<50K","50-75K","75-100K",">100K"))

head(zdata)

## zip medListPrice medValSqFt medListPriceLvl medValSqFtLvl medAge

## 1 46030 124000 60 100-200K 50-75 39.2

## 2 46031 139900 84 100-200K 75-100 40.0

## 3 46032 325000 119 300-350K 100-125 39.1

## 4 46033 365000 111 >350K 100-125 40.5

## 5 46034 249900 104 200-300K 100-125 41.0

## 6 46035 59000 43 <100K <50 40.7

## medInc medAgeLvl medIncLvl

## 1 58648 35-40 50-75K

## 2 59345 35-40 50-75K

## 3 93835 35-40 75-100K

## 4 114941 40-45 >100K

## 5 61808 40-45 50-75K

## 6 63508 40-45 50-75K

#qc

ddply(zdata,c("medAgeLvl"),summarize,minval=min(medAge),maxval=max(medAge))

## medAgeLvl minval maxval

## 1 <30 28.0 28.0

## 2 30-35 30.4 34.7

## 3 35-40 35.1 40.0

## 4 40-45 40.1 43.9

## 5 >45 45.5 48.2

ddply(zdata,c("medIncLvl"),summarize,minval=min(medInc),maxval=max(medInc))

## medIncLvl minval maxval

## 1 <50K 24486 49583

## 2 50-75K 50071 74930

## 3 75-100K 75697 93835

## 4 >100K 105040 114941

Combine the dataset containing census data and zillow data with shape file and plot using ggmap/ggplot2

zipShp3$rnum=seq(1,nrow(zipShp3))

zipPlt=merge(zipShp3,zdata,by.x=c("id"),by.y=c("zip"))

zipPlt=zipPlt[order(zipPlt$rnum),]

# choropleth map of median Age

x=get\_googlemap(center="indianapolis",maptype=c("roadmap"))

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=indianapolis&zoom=10&size=%20640x640&maptype=roadmap&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=indianapolis&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

p1=ggmap(x)

p1=p1+geom\_polygon(data=zipPlt,aes(x=long,y=lat,group=id,fill=medAgeLvl),color="black",alpha=0.2)

p1=p1+scale\_fill\_manual(values=rainbow(20)[c(4,8,12,16,20)])

p1=p1+labs(title="Median Age by Zip Code (US Census data)")

p1=p1+theme(legend.title=element\_blank(),plot.title=element\_text(face="bold"))

print(p1)

# choropleth map of median Income

x=get\_googlemap(center="indianapolis",maptype=c("roadmap"))

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=indianapolis&zoom=10&size=%20640x640&maptype=roadmap&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=indianapolis&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

p2=ggmap(x)

p2=p2+geom\_polygon(data=zipPlt,aes(x=long,y=lat,group=id,fill=medIncLvl),color="black",alpha=0.2)

p2=p2+scale\_fill\_manual(values=rainbow(20)[c(4,8,12,16,20)])

p2=p2+labs(title="Median Income by Zip Code (US census - ACS data)")

p2=p2+theme(legend.title=element\_blank(),plot.title=element\_text(face="bold"))

print(p2)

# choropleth map of median Listing price

x=get\_googlemap(center="indianapolis",maptype=c("roadmap"))

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=indianapolis&zoom=10&size=%20640x640&maptype=roadmap&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=indianapolis&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

p3=ggmap(x)

p3=p3+geom\_polygon(data=zipPlt,aes(x=long,y=lat,group=id,fill=medListPriceLvl),color="black",alpha=0.2)

p3=p3+scale\_fill\_manual(values=rainbow(20)[c(4,8,12,16,20)])

p3=p3+labs(title="Median Listing Price (Zillow API data)")

p3=p3+theme(legend.title=element\_blank(),plot.title=element\_text(face="bold"))

print(p3)

# choropleth map of median value per square feet

x=get\_googlemap(center="indianapolis",maptype=c("roadmap"))

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=indianapolis&zoom=10&size=%20640x640&maptype=roadmap&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=indianapolis&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

p4=ggmap(x)

p4=p4+geom\_polygon(data=zipPlt,aes(x=long,y=lat,group=id,fill=medValSqFtLvl),color="black",alpha=0.2)

p4=p4+scale\_fill\_manual(values=rainbow(20)[c(4,8,12,16,20)])

p4=p4+labs(title="Median Value per Square Feet (Zillow API data)")

p4=p4+theme(legend.title=element\_blank(),plot.title=element\_text(face="bold"))

print(p4)

## Get school ratings data logo

Disclaimer: This is a one off use of education.com API for exploratory purposes. I believe I am in compliance with their [*terms of use*](http://www.education.com/webservice/documentation/?section=terms). However, if somebody notices a violation of their terms of use, please let me know and I will remove this material.

Next, I used the API from [education.com](http://www.education.com/) to get school rating data to plot on a map. You can request the API key [here](http://www.education.com/schoolfinder/tools/webservice/). With this API, you need to give a IP address. I had some issues with this since the public IP address seems to change from time to time. So I had to reregister for the API key if the public IP address was different.

I used the [schoolSearch](http://www.education.com/webservice/documentation/?section=schoolsearch) API and used the result returned in JSON format. Here the form of URL I used is to get schools in a radius of 50 miles around Indianapolis area (latitude: 39.7910, longitude: -86.1480)

eduAPIkey="yourAPIkey"

#------------------Get school ratings data-------------------------------------------------

lat=39.7910

lon=-86.1480

radius=50

resURL=paste("http://api.education.com/service/service.php?f=schoolSearch&key=",eduAPIkey,

"&sn=sf&v=4&latitude=",lat,"&longitude=",lon,"&distance=",radius,"&resf=json",

sep="")

schdata=fromJSON(resURL)

fieldList=c("nces\_id","schoolname","districleaid",

"zip","city","state","latitude",

"longitude","schooltype","testrating\_text","gradelevel","studentteacherratio")

# convert list into a dataframe

x=sapply(schdata,function(x) unlist(x$school[fieldList]))

schdata2=rbind.fill(lapply(x,function(y) as.data.frame(t(y),stringsAsFactors=FALSE)))

schdata2$latitude=as.numeric(schdata2$latitude)

schdata2$longitude=as.numeric(schdata2$longitude)

# extract numeric school test rating from testrating\_text field

schdata2$testrating=as.numeric(gsub("[^0-9]","",schdata2$testrating\_text))

head(schdata2)

## nces\_id schoolname zip city state

## 1 180477000810 Charity Dye School 27 46202 Indianapolis IN

## 2 180005802315 Herron High School 46202 Indianapolis IN

## 3 A0301560 The Oaks Academy 46205 Indianapolis IN

## 4 A9102688 Indianapolis Christian School 46202 Indianapolis IN

## 5 180006802435 The Indianapolis Project School 46202 Indianapolis IN

## 6 180477001440 Center for Inquiry 46202 Indianapolis IN

## latitude longitude schooltype testrating\_text

## 1 39.79 -86.15 Public Education.com TestRating 1

## 2 39.79 -86.16 Charter Education.com TestRating 8

## 3 39.80 -86.15 Private

## 4 39.78 -86.15 Private

## 5 39.80 -86.14 Charter Education.com TestRating 1

## 6 39.78 -86.15 Magnet Education.com TestRating 7

## gradelevel studentteacherratio.total testrating

## 1 Elementary 14 1

## 2 High 14 8

## 3 Preschool,Middle,Elementary 11 NA

## 4 Preschool,High,Elementary,Middle 4 NA

## 5 Elementary,Middle 15 1

## 6 Elementary,Middle 14 7

#QC

ddply(schdata2,c("testrating\_text","testrating"),summarize,count=length(testrating))

## testrating\_text testrating count

## 1 NA 238

## 2 Education.com TestRating 1 1 38

## 3 Education.com TestRating 10 10 43

## 4 Education.com TestRating 2 2 35

## 5 Education.com TestRating 3 3 42

## 6 Education.com TestRating 4 4 47

## 7 Education.com TestRating 5 5 54

## 8 Education.com TestRating 6 6 70

## 9 Education.com TestRating 7 7 60

## 10 Education.com TestRating 8 8 64

## 11 Education.com TestRating 9 9 48

# keep only public schools

schdata3=schdata2[schdata2$schooltype=="Public",]

# group test ratings into buckets (<7, 7-8, 9-10)

schdata3$testratingLvl=cut(schdata3$testrating,breaks=c(0,6,8,10),

labels=c("< 7","7-8","9-10"))

#QC

ddply(schdata3,c("testratingLvl"),summarize,min=min(testrating),max=max(testrating),count=length(testrating))

## testratingLvl min max count

## 1 < 7 1 6 251

## 2 7-8 7 8 120

## 3 9-10 9 10 90

## 4 <NA> NA NA 27

head(schdata3)

## nces\_id schoolname zip city

## 1 180477000810 Charity Dye School 27 46202 Indianapolis

## 9 180477000844 H L Harshman Middle School 46201 Indianapolis

## 10 180477000893 Theodore Potter School 74 46201 Indianapolis

## 11 180477000801 Arsenal Technical High School 46201 Indianapolis

## 12 180477000815 Crispus Attucks Medical Magnet School 46202 Indianapolis

## 15 180477000897 Washington Irving School 14 46202 Indianapolis

## state latitude longitude schooltype testrating\_text

## 1 IN 39.79 -86.15 Public Education.com TestRating 1

## 9 IN 39.78 -86.13 Public Education.com TestRating 2

## 10 IN 39.78 -86.13 Public Education.com TestRating 4

## 11 IN 39.78 -86.13 Public Education.com TestRating 2

## 12 IN 39.78 -86.17 Public Education.com TestRating 3

## 15 IN 39.77 -86.14 Public Education.com TestRating 2

## gradelevel studentteacherratio.total testrating testratingLvl

## 1 Elementary 14 1 < 7

## 9 Middle 16 2 < 7

## 10 Elementary 12 4 < 7

## 11 High 15 2 < 7

## 12 Middle,High 17 3 < 7

## 15 Elementary 18 2 < 7

Plot school rating data overlaid on Indianapolis map.

x=get\_googlemap(center="Indianapolis",maptype=c("roadmap"))

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Indianapolis&zoom=10&size=%20640x640&maptype=roadmap&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Indianapolis&sensor=false

## Google Maps API Terms of Service : http://developers.google.com/maps/terms

p=ggmap(x)

p=p+geom\_point(data=schdata3,aes(x=longitude,y=latitude,color=testratingLvl),size=4)

p=p+scale\_color\_manual(values=c("red","#339900","blue"))

p=p+labs(title="School Ratings (Education.com API data)")

p=p+theme(legend.title=element\_blank(),plot.title=element\_text(face="bold"))

print(p)

## Warning: Removed 138 rows containing missing values (geom\_point).

## Session Info

The IDE used here is RStudio 0.98.490

sessionInfo()

## R version 3.0.2 (2013-09-25)

## Platform: x86\_64-w64-mingw32/x64 (64-bit)

##

## locale:

## [1] LC\_COLLATE=English\_United States.1252

## [2] LC\_CTYPE=English\_United States.1252

## [3] LC\_MONETARY=English\_United States.1252

## [4] LC\_NUMERIC=C

## [5] LC\_TIME=English\_United States.1252

##

## attached base packages:

## [1] stats graphics grDevices utils datasets methods base

##

## other attached packages:

## [1] mapproj\_1.2-1 maps\_2.3-6 RJSONIO\_1.0-3 plyr\_1.8

## [5] ggmap\_2.3 rgeos\_0.3-2 maptools\_0.8-27 sp\_1.0-14

## [9] ggplot2\_0.9.3.1 stringr\_0.6.2 RCurl\_1.95-4.1 bitops\_1.0-6

## [13] XML\_3.98-1.1 knitr\_1.5

##

## loaded via a namespace (and not attached):

## [1] colorspace\_1.2-4 dichromat\_2.0-0 digest\_0.6.4

## [4] evaluate\_0.5.1 foreign\_0.8-55 formatR\_0.10

## [7] grid\_3.0.2 gtable\_0.1.2 labeling\_0.2

## [10] lattice\_0.20-23 MASS\_7.3-29 munsell\_0.4.2

## [13] png\_0.1-7 proto\_0.3-10 RColorBrewer\_1.0-5

## [16] reshape2\_1.2.2 RgoogleMaps\_1.2.0.5 rjson\_0.2.13

## [19] scales\_0.2.3 tools\_3.0.2