Final Project Data retrieval

Mohsen Nabian

8/20/2015

Introduction:

This project intends to provide some scientific insight about the distribution of the restaurants, their prices, their customer satisfactions and their popularities based on the data extracted from "www.yelp.com"" the biggest repository for restaurants information.

To obtain the data, a web scrapter is written to automatically extract all the data. Moreover, to obtain the population and salary data associated with the zipcodes, we used the data provided by Population Study Center from university of michigan . This data is provided as xlsx data file in the following website: "<http://www.psc.isr.umich.edu/dis/census/Features/tract2zip/>"

By scraping Yelp website, 13474 restaurants are recorded in 15 major cities in united states with 546 unique zip codes associated with each restaurants.

The cities are:

"New York,NY","Chicago,IL","Boston,MA","Los Angeles,CA","Houston,TX","Philadelphia,PA","San Francisco,CA","Houston,TX","Washington ,DC","Phoenix,AZ","Seattle,WA","Baltimore,MD","Cleaveland,OH","Las Vegas,NV","Austin,TX"

It is aimed to perform regression analysis to find direct or indirect significant linear relationships between independent parameters like price , satisfaction , wealth of people, popularity of restaurants and so on.

STEP1) WEb Scraping

This code is scraping the yelp website through 15 major cities of united states and takes 900 restaurant from each city. By running this code, your ip might become blocked by Yelp forever, as it happened to me . However, before being blocked, I recieved very good amount of data for my analysis.

#################################################################################################  
  
  
  
cat("\014") #This clears the Consol  
rm(list=ls()) #This removes all the variables previously existed in Global Environment.  
require(RCurl) #Downloading Package  
require(XML)  
library(RCurl) #Using package in this program  
library(XML)  
library(stringr)  
#Set the working directory to your workspace  
setwd("C:/Users/monabiyan/SkyDrive/Summer 2015/Collect,retrieve data/Week14(Term-Project)")  
  
#YelpParse Function parses data in the YELP search pages. It provides Name,Type,Price, the number of reviews and the Tell number for each element.  
#The Elements could be restaurant or bars or coffee-shops or so many other places.  
#It is requiered to find the link in YELP website and put it in the YelpParse function here.  
#User is able to choose which information he/she wants as output by putting TRUE or FALSE in the correponding places for input.  
#TYPE,PRICE,REVIEW,TELL should be substituted by TRUE or FALSE based on User's need.  
YelpParse<-function(link)   
{  
   
 ########## To understand which information the user wants######  
 #choice=0;  
 #choice[1]=1  
 #choice[2]=as.numeric(TYPE)\*2  
 #choice[3]=as.numeric(PRICE)\*3  
 #choice[4]=as.numeric(REVIEW)\*4  
 #choice[5]=as.numeric(TELL)\*5  
 ################################################################  
 # This is the URL of the website we need scrape to get information on the   
   
 theurl <- link  
 theurl<-gsub(" ","",theurl); #No extra spaces  
 webpage <- getURL(theurl)  
 # convert the page into a line-by-line format rather than a single string  
 tc <- textConnection(webpage)  
 webpage <- readLines(tc) #webpage is now a vector of string each elament is a line of string  
 close(tc)  
 pagetree <- htmlTreeParse(webpage, useInternalNodes = TRUE) #pagetree is now in html format and parseable with xpath syntax.  
   
   
   
 ######################### NAME ###############################  
   
 restaurant.name<- unlist(xpathApply(pagetree,"//\*/span[@class='indexed-biz-name']/a[@\*][@\*][@\*]",xmlValue))  
 if(length(restaurant.name)==11) #Sometimes it gives 11 ellements and the first one is wrong"   
 {restaurant.name<-restaurant.name[2:11]}  
 restaurant.name<-as.character(restaurant.name)  
 restaurant.name<-gsub("<U+0092>","",restaurant.name)  
   
 ###### Removing <U+0092> #### I Found this in Internet  
 Encoding(restaurant.name) <- "latin1" # (just to make sure)  
 iconv(restaurant.name, "latin1", "ASCII", sub="")  
 #####  
   
   
 restaurant.name  
   
   
   
 ############################ REVIEW COUNT #############################  
   
 review.count<-unlist(xpathApply(pagetree,"//\*/span[@class='review-count rating-qualifier']",xmlValue))  
 review.count  
 review.count<-gsub("\n ", "",review.count) #Removing extra characters"  
 review.count<-gsub("\n ", "",review.count)  
 review.count<-gsub(" reviews", "",review.count)  
 if(length(review.count)==11)  
 {review.count<-review.count[2:11]}  
 review.count<-as.numeric(review.count)  
 review.count  
   
 ############################# PRICE #############################  
   
 restaurant.price<-unlist(xpathApply(pagetree,"//\*/span[@class='business-attribute price-range']",xmlValue))  
 print(restaurant.price)  
 for (i in 1:length(restaurant.price)) #Scaling price notations to 1,2,3,4 accordingly where 4 is very epensive.  
 {  
 if (restaurant.price[i]=="$") {restaurant.price[i]="1"; }  
 if (restaurant.price[i]=="$$") {restaurant.price[i]="2"; }  
 if (restaurant.price[i]=="$$$") {restaurant.price[i]="3"; }  
 if (restaurant.price[i]=="$$$$") {restaurant.price[i]="4"; }  
 }  
 restaurant.price  
 if(length(restaurant.price)==11)  
 {restaurant.price<-restaurant.price[2:11]}  
 restaurant.price<-as.numeric(restaurant.price)  
 restaurant.price  
   
   
 ######################### TYPE ############################  
   
 restaurant.type<-unlist(xpathApply(pagetree,"//\*/span[@class='category-str-list']/a[@\*][1]",xmlValue)) ##Some times there are several <a> tags. We need the first one.  
 if(length(restaurant.type)==11)  
 {restaurant.type<-restaurant.type[2:11]}  
 restaurant.type<-as.character(restaurant.type)  
 restaurant.type  
   
 ######################## star ###############################  
 restaurant.star<-unlist(xpathApply(pagetree,"//\*/div[@class='rating-large']/i",xmlAttrs))  
 restaurant.star<-as.character(restaurant.star)  
   
 restaurant.star<-gsub(" star rating", "",restaurant.star)  
 hh=0;  
 for (i in 1:(length(restaurant.star)/2))  
 {hh[i]<-restaurant.star[2\*i]}  
 restaurant.star<-hh  
 restaurant.star<-as.numeric(restaurant.star)  
   
   
   
 ######################## Neighborhood ###############################  
 restaurant.neighborhood<-unlist(xpathApply(pagetree,"//\*/span[@class='neighborhood-str-list']",xmlValue))  
   
 restaurant.neighborhood<-gsub("\n ", "",restaurant.neighborhood)  
 restaurant.neighborhood<-gsub(" ", "",restaurant.neighborhood)  
 if(length(restaurant.neighborhood)==11)  
 {restaurant.neighborhood<-restaurant.neighborhood[2:11]}  
 restaurant.neighborhood<-as.character(restaurant.neighborhood)  
 restaurant.neighborhood  
   
   
 ######################## ADDRESS ###############################  
 restaurant.address<-unlist(xpathApply(pagetree,"//\*/address",xmlValue))  
   
 restaurant.address<-gsub("\n ", "",restaurant.address)  
 restaurant.address<-gsub(" ", "",restaurant.address)  
 restaurant.address<-gsub("\n", "",restaurant.address)  
 restaurant.address<-gsub(city,paste(" ",city),restaurant.address)  
 if(length(restaurant.address)==11)  
 {restaurant.address<-restaurant.address[2:11]}  
 restaurant.address<-as.character(restaurant.address)  
 restaurant.address  
 restaurant.zip<-str\_sub(restaurant.address,-5,-1)  
 ############################ TELL ##############################  
   
 restaurant.tell<-unlist(xpathApply(pagetree,"//\*/div[@class='secondary-attributes']/span[@class='biz-phone']",xmlValue))  
 restaurant.tell<-gsub("\n ", "",restaurant.tell)  
 restaurant.tell<-gsub("\n ", "",restaurant.tell)  
 if(length(restaurant.tell)==11)  
 {restaurant.tell<-restaurant.tell[2:11]}  
 restaurant.tell<-as.character(restaurant.tell)  
 restaurant.tell  
   
   
 ############################### Putting ALL in a DATA FRAME ##############################  
 min\_length=min(length(restaurant.name),length(restaurant.type),length(restaurant.price),length(review.count),length(restaurant.tell),length(restaurant.address),length(restaurant.star))  
 restaurant.data<-data.frame(NAME=restaurant.name[1:min\_length],TYPE=restaurant.type[1:min\_length],PRICE= restaurant.price[1:min\_length],REVIEW\_COUNT=review.count[1:min\_length],STAR=restaurant.star[1:min\_length],TELL=restaurant.tell[1:min\_length],ADDRESS=restaurant.address[1:min\_length],ZIPCODE=restaurant.zip[1:min\_length])  
 return(restaurant.data)  
}  
   
 ##################Statistical and Searching Questions ###############  
   
 #print(restaurant.data[,choice])  
 #print(paste("The average price levels is",mean(restaurant.data$PRICE)))  
 #print(paste("The standard deviation of price levels is",sd(restaurant.data$PRICE)))  
 #print(paste("The average number of reviews for restaurant is",mean(restaurant.data$REVIEW\_COUNT) ))  
 #print(paste(restaurant.data$NAME[restaurant.data$PRICE==1]," is inexpensive. ENJOY!!"))  
   
   
  
  
  
  
  
#######################################  
AddSalary<-function(df)  
{  
 zip\_info<-read.csv("MedianZIP-3.csv",header=TRUE,stringsAsFactors=FALSE)  
   
   
 zip\_info$Zip<-as.character(zip\_info$Zip)  
 for (i in 1:length(zip\_info[,1]))  
 {  
 if(nchar(zip\_info$Zip[i])==4)  
 {  
 zip\_info$Zip[i]<-paste("0",zip\_info$Zip[i])  
 }  
 }  
 zip\_info$Zip<-gsub(" ", "",zip\_info$Zip)  
   
   
 zip\_info$Median<-as.character(zip\_info$Median)  
 zip\_info$Mean<-as.character(zip\_info$Mean)  
 zip\_info$Pop<-as.character(zip\_info$Pop)  
 df$ZIPCODE<-as.character(df$ZIPCODE)  
   
   
 MEDIAN\_SAL<-0;  
 MEAN\_SAL<-0;  
 POP<-0;  
   
   
   
 for(i in 1:length(df[,1]))  
 {  
 if (sum(df$ZIPCODE[i]==zip\_info$Zip)==0)  
 {  
 MEDIAN\_SAL[i]=0;  
 MEAN\_SAL[i]=0;  
 POP[i]=0;  
 }  
 else  
 {  
 MEDIAN\_SAL[i]<-zip\_info$Median[df$ZIPCODE[i]==zip\_info$Zip]  
 MEAN\_SAL[i]<-zip\_info$Mean[df$ZIPCODE[i]==zip\_info$Zip]  
 POP[i]<-zip\_info$Pop[df$ZIPCODE[i]==zip\_info$Zip]  
 }  
 }  
 df<-cbind(df,MEDIAN\_SAL,MEAN\_SAL,POP)  
 return(df)  
   
}  
  
  
  
  
  
##############################################################  
  
setwd("C:/Users/nabian.m/Desktop/cities")  
  
  
megacities<-c("New York,NY","Chicago,IL","Boston,MA","Los Angeles,CA","Houston,TX","Philadelphia,PA","San Francisco,CA","Houston,TX","Washington ,DC","Phoenix,AZ","Seattle,WA","Baltimore,MD","Cleaveland,OH","Las Vegas,NV","Austin,TX","Oklahama City,OK")  
  
n<-90  
  
for (location in megacities)  
{  
 print(location)  
 comma<-unlist(str\_locate\_all(pattern =',',location))[1]  
 comma<-as.numeric(comma)  
 city<-str\_sub(location,start=1,end=(comma-1))  
 state<-str\_sub(location,start=(comma+1),end=nchar(location))  
 print(city)  
 address<-paste("http://www.yelp.com/search?find\_desc=Restaurants&find\_loc=",city,"%2C+",state,"&start=",as.character(0),sep="")  
 all<-YelpParse(address)  
   
 print(all)  
   
 #ads <- all$ADDRESS  
 #locations <- ldply(ads, function(x) getLocation(x))  
 #names(locations) <- c("LATTITUDE", "LONGITUDE", "location\_type", "formatted")  
 #all<-cbind(all,locations)  
   
   
   
 for (i in 1:n)  
 {  
 print(i)  
 address<-paste("http://www.yelp.com/search?find\_desc=Restaurants&find\_loc=",city,"%2C+",state,"&start=",as.character(i\*10),sep="")  
 df<-YelpParse(address)  
   
 #ads <- df$ADDRESS  
 #locations <- ldply(address, function(x) getLocation(x))  
 #names(locations) <- c("LATTITUDE", "LONGITUDE", "location\_type", "formatted")  
   
 #hh<-cbind(df,locations)  
 #all<-rbind(all,hh)  
 all<-rbind(all,df)  
 }   
   
   
   
 all<-AddSalary(all) #Update with the salaries and populations  
 all<cbind(all,city)  
 all<-all[-which((all$POP==0)==TRUE),]  
 head(all)  
   
 write.csv(all,file=paste(city,"\_res.csv"),row.names = FALSE)  
  
}  
  
  
#########################################################################  
  
  
######################### Adding City as a new column ###############  
  
setwd("C:/Users/nabian.m/Desktop/cities")  
megacities<-c("New York,NY","Chicago,IL","Boston,MA","Los Angeles,CA","Houston,TX","Philadelphia,PA","San Francisco,CA","Houston,TX","Washington ,DC","Phoenix,AZ","Seattle,WA","Baltimore,MD","Cleaveland,OH","Las Vegas,NV","Austin,TX","Oklahama City,OK")  
  
for (location in megacity)  
{  
 print(location)  
 comma<-unlist(str\_locate\_all(pattern =',',location))[1]  
 comma<-as.numeric(comma)  
 city<-str\_sub(location,start=1,end=(comma-1))  
 state<-str\_sub(location,start=(comma+1),end=nchar(location))  
   
   
 all<-read.csv(file=paste(city,"\_res.csv"))  
 all<-cbind(all,city)  
 write.csv(all,file=paste(city,"\_res.csv"),row.names = FALSE)  
}  
  
###############################################################  
  
  
############ earasing comma "," from Salary and Population ####  
  
setwd("C:/Users/nabian.m/Desktop/cities")  
megacities<-c("New York,NY","Chicago,IL","Boston,MA","Los Angeles,CA","Houston,TX","Philadelphia,PA","San Francisco,CA","Houston,TX","Washington ,DC","Phoenix,AZ","Seattle,WA","Baltimore,MD","Cleaveland,OH","Las Vegas,NV","Austin,TX","Oklahama City,OK")  
  
for (location in megacities)  
{  
 print(location)  
 comma<-unlist(str\_locate\_all(pattern =',',location))[1]  
 comma<-as.numeric(comma)  
 city<-str\_sub(location,start=1,end=(comma-1))  
 state<-str\_sub(location,start=(comma+1),end=nchar(location))  
   
   
 all<-read.csv(file=paste(city,"\_res.csv"))  
   
 all$MEDIAN\_SAL<-gsub(",","",all$MEDIAN\_SAL)  
 all$MEAN\_SAL<-gsub(",","",all$MEAN\_SAL)  
 all$POP<-gsub(",","",all$POP)  
   
 write.csv(all,file=paste(city,"\_res.csv"),row.names = FALSE)  
}  
  
  
########################################################

So far all requiered data are collected and saved in csv files. Each city,900 restaurant in one file. 15 cities and 15 files.

Now lets start reading the data and putting them all together as 'all' data frame.

########### Reading and Making One DataFarme of All Files ############################################  
  
  
library(stringr)  
setwd("C:/Users/nabian.m/Desktop/cities")  
megacity<-"New York,NY"  
  
for (location in megacity)  
{  
 print(location)  
 comma<-unlist(str\_locate\_all(pattern =',',location))[1]  
 comma<-as.numeric(comma)  
 city<-str\_sub(location,start=1,end=(comma-1))  
 state<-str\_sub(location,start=(comma+1),end=nchar(location))  
   
   
 all<-read.csv(file=paste(city,"\_res.csv"))  
}

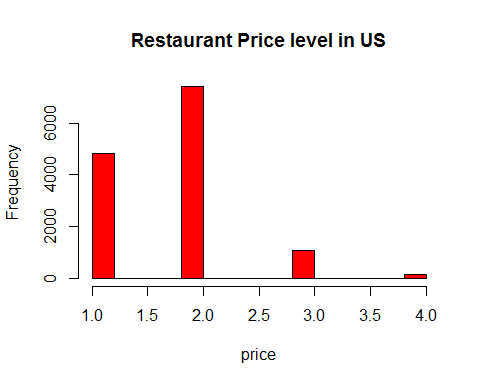
## [1] "New York,NY"

megacities<-c("Chicago,IL","Boston,MA","Los Angeles,CA","Houston,TX","Philadelphia,PA","San Francisco,CA","Houston,TX","Washington ,DC","Phoenix,AZ","Seattle,WA","Baltimore,MD","Cleaveland,OH","Las Vegas,NV","Austin,TX","Oklahama City,OK")  
  
  
for (location in megacities)  
{  
   
 comma<-unlist(str\_locate\_all(pattern =',',location))[1]  
 comma<-as.numeric(comma)  
 city<-str\_sub(location,start=1,end=(comma-1))  
 state<-str\_sub(location,start=(comma+1),end=nchar(location))  
 print(location)  
 df<-read.csv(file=paste(city,"\_res.csv"))  
 all<-rbind(df,all)  
}

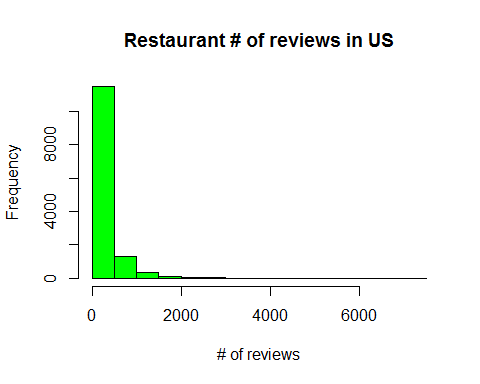
## [1] "Chicago,IL"  
## [1] "Boston,MA"  
## [1] "Los Angeles,CA"  
## [1] "Houston,TX"  
## [1] "Philadelphia,PA"  
## [1] "San Francisco,CA"  
## [1] "Houston,TX"  
## [1] "Washington ,DC"  
## [1] "Phoenix,AZ"  
## [1] "Seattle,WA"  
## [1] "Baltimore,MD"  
## [1] "Cleaveland,OH"  
## [1] "Las Vegas,NV"  
## [1] "Austin,TX"  
## [1] "Oklahama City,OK"

Here are some plots from 'all' data frame. That means all data (13474 restaurants) as a whole regardress of their locations are being ploted here:

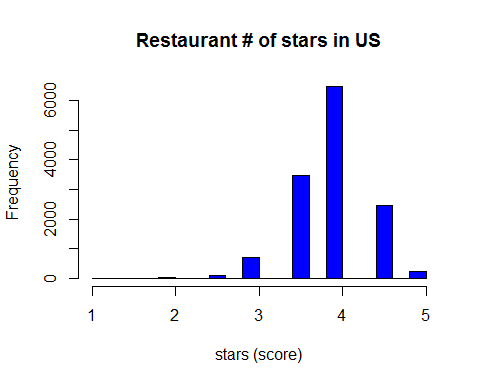
hist(all$PRICE,breaks=20, col="red",xlab="price",main="Restaurant Price level in US")



hist(all$REVIEW\_COUNT,breaks=20, col="green",xlab="# of reviews",main="Restaurant # of reviews in US")



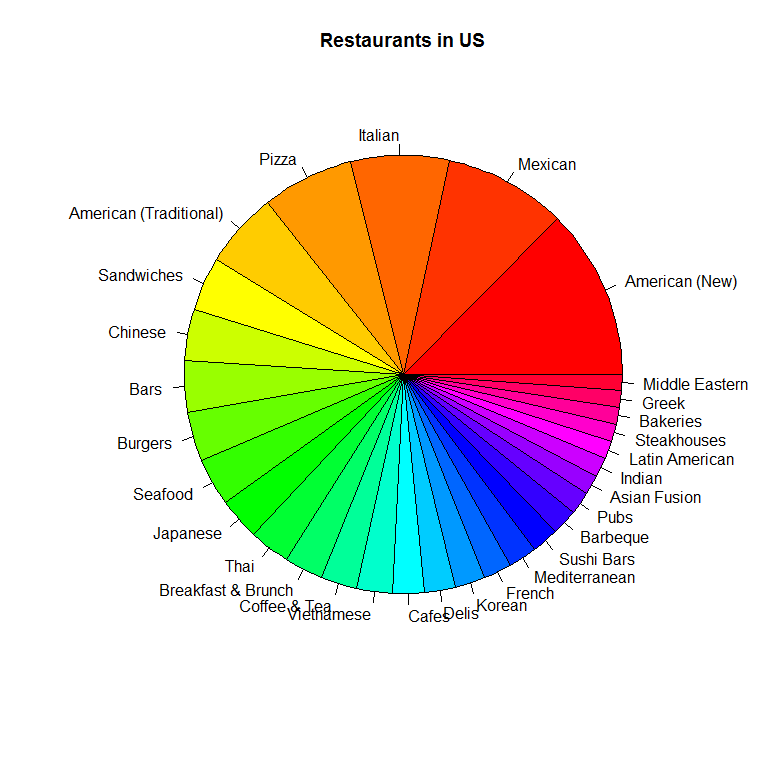
hist(all$STAR,breaks=20, col="blue",xlab="stars (score)",main="Restaurant # of stars in US")



#Restaurants Type  
tp<-as.data.frame(table(as.factor(all$TYPE)))  
tp<-tp[with(tp, order(-tp$Freq)), ] # sorting restaurant types based of frequency  
tp[1:20,]

## Var1 Freq  
## 2 American (New) 1305  
## 35 Mexican 938  
## 29 Italian 757  
## 37 Pizza 698  
## 3 American (Traditional) 582  
## 40 Sandwiches 414  
## 15 Chinese 398  
## 7 Bars 389  
## 11 Burgers 384  
## 41 Seafood 365  
## 30 Japanese 317  
## 48 Thai 313  
## 10 Breakfast & Brunch 297  
## 17 Coffee & Tea 282  
## 49 Vietnamese 271  
## 12 Cafes 241  
## 61 Delis 241  
## 76 Korean 225  
## 70 French 222  
## 34 Mediterranean 214

pie(tp$Freq[1:30], labels = tp$Var1[1:30], main="Restaurants in US",col=rainbow(30))



######## For Each city Medians and Means  
  
  
megacities<-c("New York,NY","Chicago,IL","Boston,MA","Los Angeles,CA","Houston,TX","Philadelphia,PA","San Francisco,CA","Houston,TX","Washington ,DC","Phoenix,AZ","Seattle,WA","Baltimore,MD","Cleaveland,OH","Las Vegas,NV","Austin,TX")  
  
i=1;  
median\_price=0;  
mean\_price=0;  
median\_reviews=0;  
mean\_reviews=0;  
median\_star=0;  
mean\_star=0;  
mean\_of\_median\_salary=0;  
population=0;  
city\_name=0;  
  
for (location in megacities)  
{  
 comma<-unlist(str\_locate\_all(pattern =',',location))[1]  
 comma<-as.numeric(comma)  
 city<-str\_sub(location,start=1,end=(comma-1))  
 print(city)  
 state<-str\_sub(location,start=(comma+1),end=nchar(location))  
   
   
 median\_price[i]<-median(as.numeric((all$PRICE[all$city==city])),na.rm = TRUE)  
   
 mean\_price[i]<-mean(as.numeric((all$PRICE[all$city==city])),na.rm = TRUE)  
   
 median\_reviews[i]<-median(as.numeric((all$REVIEW\_COUNT[all$city==city])),na.rm = TRUE)  
 mean\_reviews[i]<-mean(as.numeric((all$REVIEW\_COUNT[all$city==city])),na.rm = TRUE)  
   
 median\_star[i]<-median(as.numeric((all$STAR[all$city==city])),na.rm = TRUE)  
 mean\_star[i]<-mean(as.numeric((all$STAR[all$city==city])),na.rm = TRUE)  
   
 mean\_of\_median\_salary[i]<-median(as.numeric((all$MEDIAN\_SAL[all$city==city])),na.rm = TRUE)   
   
 population[i]<-sum(as.numeric((all$POP[all$city==city])),na.rm = TRUE)  
   
 city\_name[i]<-city  
 i=i+1  
}

## [1] "New York"  
## [1] "Chicago"  
## [1] "Boston"  
## [1] "Los Angeles"  
## [1] "Houston"  
## [1] "Philadelphia"  
## [1] "San Francisco"  
## [1] "Houston"  
## [1] "Washington "  
## [1] "Phoenix"  
## [1] "Seattle"  
## [1] "Baltimore"  
## [1] "Cleaveland"  
## [1] "Las Vegas"  
## [1] "Austin"

Here are the the independent variables for each city:

median\_price

## [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1

mean\_price

## [1] 1.980877 1.828194 1.743274 1.744469 1.738069 1.739421 1.813877  
## [8] 1.738069 1.874580 1.547991 1.791160 1.684770 1.585642 1.824834  
## [15] 1.537264

median\_reviews

## [1] 247.0 222.0 126.0 291.0 124.0 134.5 412.0 124.0 178.0 88.0 183.0  
## [12] 41.0 27.0 186.0 170.0

mean\_reviews

## [1] 451.16535 377.31608 211.02130 497.59735 177.42064 211.16147 646.31057  
## [8] 177.42064 270.64054 145.73884 296.07845 85.46368 55.85073 326.19734  
## [15] 237.16352

median\_star

## [1] 4.0 4.0 3.5 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0

mean\_star

## [1] 4.120360 4.024780 3.721973 4.046460 3.824084 3.912584 3.996696  
## [8] 3.824084 3.768197 3.917411 3.931492 3.793388 3.764484 4.031042  
## [15] 3.980534

mean\_of\_median\_salary

## [1] 73075 68426 61807 40973 64545 41800 75106 64545 68051 44596 57971  
## [12] 48519 30283 42954 47497

population

## [1] 39070915 45725683 13703325 33130490 47196306 23527972 33151152  
## [8] 47196306 23817566 26551511 22966908 22517160 22586851 27006361  
## [15] 24191203

city\_name

## [1] "New York" "Chicago" "Boston" "Los Angeles"   
## [5] "Houston" "Philadelphia" "San Francisco" "Houston"   
## [9] "Washington " "Phoenix" "Seattle" "Baltimore"   
## [13] "Cleaveland" "Las Vegas" "Austin"

saving data:

city\_info<-data.frame(city\_name,mean\_price,median\_price,median\_reviews,mean\_reviews,median\_star,mean\_star,  
 mean\_of\_median\_salary,population)  
   
city\_info

## city\_name mean\_price median\_price median\_reviews mean\_reviews  
## 1 New York 1.980877 2 247.0 451.16535  
## 2 Chicago 1.828194 2 222.0 377.31608  
## 3 Boston 1.743274 2 126.0 211.02130  
## 4 Los Angeles 1.744469 2 291.0 497.59735  
## 5 Houston 1.738069 2 124.0 177.42064  
## 6 Philadelphia 1.739421 2 134.5 211.16147  
## 7 San Francisco 1.813877 2 412.0 646.31057  
## 8 Houston 1.738069 2 124.0 177.42064  
## 9 Washington 1.874580 2 178.0 270.64054  
## 10 Phoenix 1.547991 2 88.0 145.73884  
## 11 Seattle 1.791160 2 183.0 296.07845  
## 12 Baltimore 1.684770 2 41.0 85.46368  
## 13 Cleaveland 1.585642 2 27.0 55.85073  
## 14 Las Vegas 1.824834 2 186.0 326.19734  
## 15 Austin 1.537264 1 170.0 237.16352  
## median\_star mean\_star mean\_of\_median\_salary population  
## 1 4.0 4.120360 73075 39070915  
## 2 4.0 4.024780 68426 45725683  
## 3 3.5 3.721973 61807 13703325  
## 4 4.0 4.046460 40973 33130490  
## 5 4.0 3.824084 64545 47196306  
## 6 4.0 3.912584 41800 23527972  
## 7 4.0 3.996696 75106 33151152  
## 8 4.0 3.824084 64545 47196306  
## 9 4.0 3.768197 68051 23817566  
## 10 4.0 3.917411 44596 26551511  
## 11 4.0 3.931492 57971 22966908  
## 12 4.0 3.793388 48519 22517160  
## 13 4.0 3.764484 30283 22586851  
## 14 4.0 4.031042 42954 27006361  
## 15 4.0 3.980534 47497 24191203

Now it is time to do analysis based on zip codes.

######## For Each zip Code Medians and Means  
  
  
i=1;  
median\_price=0;  
mean\_price=0;   
median\_reviews=0;  
mean\_reviews=0;  
median\_star=0;  
mean\_star=0;  
median\_salary=0;  
population=0;  
city\_name=0;  
zip\_code=0;  
zip\_count=0;  
for (zip in unique(all$ZIPCODE))  
{  
   
 median\_price[i]<-median(as.numeric((all$PRICE[all$ZIPCODE==zip])),na.rm = TRUE)  
   
 mean\_price[i]<-mean(as.numeric((all$PRICE[all$ZIPCODE==zip])),na.rm = TRUE)  
   
 median\_reviews[i]<-median(as.numeric((all$REVIEW\_COUNT[all$ZIPCODE==zip])),na.rm = TRUE)  
 mean\_reviews[i]<-mean(as.numeric((all$REVIEW\_COUNT[all$ZIPCODE==zip])),na.rm = TRUE)  
   
 median\_star[i]<-median(as.numeric((all$STAR[all$ZIPCODE==zip])),na.rm = TRUE)  
 mean\_star[i]<-mean(as.numeric((all$STAR[all$ZIPCODE==zip])),na.rm = TRUE)  
   
 median\_salary[i]<-as.numeric((all$MEDIAN\_SAL[all$ZIPCODE==zip]))  
   
 median\_salary[i]<-as.numeric((all$MEDIAN\_SAL[all$ZIPCODE==zip]))  
   
 population[i]<-as.numeric((all$POP[all$ZIPCODE==zip]))  
   
 city\_name[i]<-as.character(all$city[all$ZIPCODE==zip])  
   
 zip\_code[i]<-zip  
   
 zip\_count[i]<-sum(all$ZIPCODE==zip)  
 i=i+1  
   
}  
  
head(mean\_price)

## [1] 1.863636 2.100000 1.500000 1.692308 2.000000 1.500000

head(median\_reviews)

## [1] 87.5 67.5 69.0 81.0 72.0 49.5

head(mean\_reviews)

## [1] 96.50000 84.30000 148.00000 107.84615 99.64706 84.16667

head(median\_star)

## [1] 4.00 4.00 4.25 4.00 4.00 4.00

head(mean\_star)

## [1] 3.886364 3.900000 4.375000 3.923077 4.088235 4.000000

head(median\_salary)

## [1] 48692 66665 25482 33739 35294 37011

head(population)

## [1] 4345 10049 15181 13850 3362 24959

head(city\_name)

## [1] "Oklahama City" "Oklahama City" "Oklahama City" "Oklahama City"  
## [5] "Oklahama City" "Oklahama City"

head(zip\_code)

## [1] 73103 73116 73108 73106 73102 73107

head(zip\_count)

## [1] 22 10 4 13 17 6

Regression Analysis:

Since our analysis is based on zip codes, we only take zipcodes with more than 30 restaurants captured in that zipcode.

zip\_info<-data.frame(zip\_code,city\_name,mean\_price,median\_price,median\_reviews,mean\_reviews,median\_star,mean\_star,median\_salary,population,zip\_count)  
  
zip\_credit<-zip\_info[zip\_info$zip\_count>30,] # Zip codes that have more than 50 restaurants  
  
head(zip\_credit)

## zip\_code city\_name mean\_price median\_price median\_reviews mean\_reviews  
## 25 78705 Austin 1.426230 1 151.0 213.5574  
## 26 78704 Austin 1.504065 1 218.0 342.4553  
## 29 78757 Austin 1.541667 1 167.0 245.4583  
## 30 78702 Austin 1.552632 1 148.5 221.1053  
## 31 78701 Austin 1.712000 2 174.0 270.4960  
## 32 78759 Austin 1.694444 2 215.0 261.0833  
## median\_star mean\_star median\_salary population zip\_count  
## 25 4 3.934426 14590 31340 61  
## 26 4 3.983740 47497 42117 123  
## 29 4 3.947917 53358 21310 48  
## 30 4 4.125000 31715 21334 76  
## 31 4 4.016000 65353 6841 125  
## 32 4 3.861111 73831 37767 36

sum(zip\_credit$zip\_count) #9863 restaurants from 154 zipcodes in which 30 restaurants are there

## [1] 9863

#################################################

So here are a great information about the zip codes :

############ Highest Score average score for zip codes ######  
zip\_credit$zip\_code[sort(zip\_credit$mean\_star,decreasing = TRUE)==zip\_credit$mean\_star]

## [1] 85020

zip\_mean\_star\_ordered<-zip\_credit[order(-zip\_credit$mean\_star),]

Highest satisfaction (highest star score) :

head(zip\_mean\_star\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 485 10013 New York 1.958333 2 358.5  
## 481 11211 New York 2.021739 2 242.0  
## 65 89101 Las Vegas 1.760000 2 218.0  
## 482 10002 New York 1.576271 2 192.0  
## 30 78702 Austin 1.552632 1 148.5  
## 365 90026 Los Angeles 1.750000 2 230.0  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 485 475.1250 4 4.187500 69836 24723 48  
## 481 383.5870 4 4.184783 37632 84434 46  
## 65 270.5000 4 4.140000 29139 50493 50  
## 482 642.1525 4 4.135593 30844 70878 59  
## 30 221.1053 4 4.125000 31715 21334 76  
## 365 411.6875 4 4.125000 43918 67869 32

Lowest satisfaction (lowest star score) :

tail(zip\_mean\_star\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 232 20037 Washington 2.000000 2 78.0  
## 427 2128 Boston 1.618182 2 53.0  
## 429 2115 Boston 1.528571 2 95.5  
## 422 2215 Boston 1.443038 1 136.0  
## 423 2114 Boston 1.750000 2 166.0  
## 100 44115 Cleaveland 1.698630 2 34.0  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 232 226.06383 3.5 3.691489 60051 14636 47  
## 427 83.07273 3.5 3.672727 42613 38185 55  
## 429 196.82857 3.5 3.671429 30282 25637 70  
## 422 236.68354 3.5 3.620253 30726 21896 79  
## 423 169.77273 3.5 3.613636 69809 11170 44  
## 100 67.54688 3.5 3.595890 13316 7434 73

unfortunaltely Boston has a big share in unsatisfactory restaurants.

Here are the zipcodes with most expensive restaurants:

zip\_mean\_price\_ordered<-zip\_credit[order(-zip\_credit$mean\_price),]  
head(zip\_mean\_price\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews mean\_reviews  
## 450 60654 Chicago 2.524590 3 432 516.6557  
## 491 10014 New York 2.396226 2 335 542.7925  
## 63 89109 Las Vegas 2.371287 2 384 646.9653  
## 484 10011 New York 2.303030 2 544 828.9091  
## 250 77019 Houston 2.294118 2 170 237.3235  
## 264 77056 Houston 2.270270 2 165 212.6216  
## median\_star mean\_star median\_salary population zip\_count  
## 450 4 4.024590 93396 14875 61  
## 491 4 4.084906 98450 32867 53  
## 63 4 4.089109 37456 9490 202  
## 484 4 4.106061 92359 45899 33  
## 250 4 3.823529 94223 18944 68  
## 264 4 3.851351 95008 18673 74

Here are the zipcodes with least expensive restaurants:

tail(zip\_mean\_price\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 25 78705 Austin 1.426230 1 151.0  
## 341 19145 Philadelphia 1.384615 1 64.0  
## 286 77036 Houston 1.344828 1 89.0  
## 208 85013 Phoenix 1.343750 1 80.5  
## 38 78753 Austin 1.257143 1 121.0  
## 425 2127 Boston 1.208333 1 40.0  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 25 213.55738 4 3.934426 14590 31340 61  
## 341 89.79487 4 3.961538 34999 45646 39  
## 286 174.72414 4 3.793103 28529 71969 58  
## 208 95.00000 4 3.859375 45862 19314 32  
## 38 161.14286 4 4.057143 39658 49301 35  
## 425 76.85417 4 3.791667 64285 29457 48

So if you are a food lover and have not so much money go to Austin 78705.

here we do the same procedure but instead of average price we worked on the median price:

zip\_median\_price\_ordered<-zip\_credit[order(-zip\_credit$median\_price),]  
head(zip\_median\_price\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews mean\_reviews  
## 450 60654 Chicago 2.524590 3 432.0 516.6557  
## 31 78701 Austin 1.712000 2 174.0 270.4960  
## 32 78759 Austin 1.694444 2 215.0 261.0833  
## 34 78703 Austin 1.815789 2 196.5 243.4211  
## 37 78758 Austin 1.562500 2 200.5 228.8333  
## 42 78751 Austin 1.594595 2 227.0 333.7297  
## median\_star mean\_star median\_salary population zip\_count  
## 450 4 4.024590 93396 14875 61  
## 31 4 4.016000 65353 6841 125  
## 32 4 3.861111 73831 37767 36  
## 34 4 4.039474 80708 19307 38  
## 37 4 3.927083 42886 44072 48  
## 42 4 3.932432 37072 14385 37

tail(zip\_median\_price\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 376 90012 Los Angeles 1.529412 1 530  
## 419 2108 Boston 1.600000 1 129  
## 422 2215 Boston 1.443038 1 136  
## 425 2127 Boston 1.208333 1 40  
## 428 2110 Boston 1.603175 1 70  
## 444 60618 Chicago 1.488889 1 169  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 376 838.64706 4.0 4.009804 39775 27522 51  
## 419 271.18182 3.5 3.745455 96033 3324 55  
## 422 236.68354 3.5 3.620253 30726 21896 79  
## 425 76.85417 4.0 3.791667 64285 29457 48  
## 428 96.66667 4.0 3.761905 102972 1606 63  
## 444 446.02222 4.0 4.033333 58006 92084 45

Zipcodes with the highest avergare in popularity (written reviews ) are:

zip\_mean\_number\_review\_ordered<-zip\_credit[order(-zip\_credit$mean\_reviews),]  
head(zip\_mean\_number\_review\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 367 90013 Los Angeles 2.058824 2 723.0  
## 321 94122 San Francisco 1.615385 2 675.0  
## 309 94114 San Francisco 1.951220 2 658.0  
## 319 94111 San Francisco 1.843750 2 428.0  
## 376 90012 Los Angeles 1.529412 1 530.0  
## 306 94133 San Francisco 1.875000 2 766.5  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 367 1215.1471 4 4.000000 17628 11772 34  
## 321 958.6923 4 4.089744 77483 56023 39  
## 309 938.6585 4 4.048780 103206 31124 41  
## 319 914.4062 4 3.890625 93790 3712 32  
## 376 838.6471 4 4.009804 39775 27522 51  
## 306 834.4688 4 3.992188 45203 22499 64

So losangeles 90013 has the highest average popularity restaurants. This is downtown Los Angeles Sanfransico 94122 is the next.

Least popular restaurants

tail(zip\_mean\_number\_review\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 102 44102 Cleaveland 1.477273 1 46  
## 98 44114 Cleaveland 1.541176 2 23  
## 101 44106 Cleaveland 1.709091 2 23  
## 137 21218 Baltimore 1.547170 1 33  
## 144 21212 Baltimore 1.645161 2 25  
## 104 44111 Cleaveland 1.475000 1 19  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 102 57.97436 4.00 3.875000 25616 52158 44  
## 98 53.41975 4.00 3.805882 22793 4081 85  
## 101 46.54717 4.00 3.763636 28835 32154 55  
## 137 43.76000 4.00 3.773585 41265 53777 53  
## 144 33.23333 4.00 3.758065 72503 34073 31  
## 104 26.22500 3.75 3.825000 42117 42810 40

Mostly in Cleavland and Baltimore.

The same analysis but with the zip codes wityh highest and lowest median popularity restaurants.

zip\_median\_number\_review\_ordered<-zip\_credit[order(-zip\_credit$median\_reviews),]  
head(zip\_median\_number\_review\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 306 94133 San Francisco 1.875000 2 766.5  
## 367 90013 Los Angeles 2.058824 2 723.0  
## 321 94122 San Francisco 1.615385 2 675.0  
## 309 94114 San Francisco 1.951220 2 658.0  
## 484 10011 New York 2.303030 2 544.0  
## 376 90012 Los Angeles 1.529412 1 530.0  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 306 834.4688 4 3.992188 45203 22499 64  
## 367 1215.1471 4 4.000000 17628 11772 34  
## 321 958.6923 4 4.089744 77483 56023 39  
## 309 938.6585 4 4.048780 103206 31124 41  
## 484 828.9091 4 4.106061 92359 45899 33  
## 376 838.6471 4 4.009804 39775 27522 51

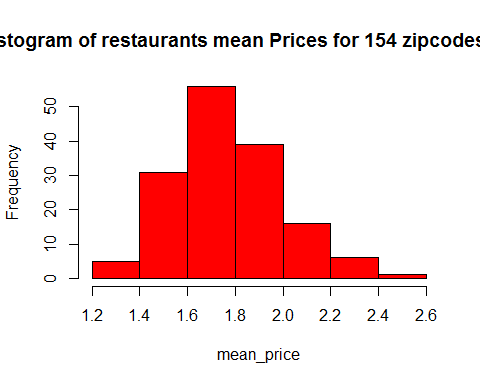
tail(zip\_median\_number\_review\_ordered)

## zip\_code city\_name mean\_price median\_price median\_reviews  
## 100 44115 Cleaveland 1.698630 2 34  
## 137 21218 Baltimore 1.547170 1 33  
## 144 21212 Baltimore 1.645161 2 25  
## 98 44114 Cleaveland 1.541176 2 23  
## 101 44106 Cleaveland 1.709091 2 23  
## 104 44111 Cleaveland 1.475000 1 19  
## mean\_reviews median\_star mean\_star median\_salary population zip\_count  
## 100 67.54688 3.50 3.595890 13316 7434 73  
## 137 43.76000 4.00 3.773585 41265 53777 53  
## 144 33.23333 4.00 3.758065 72503 34073 31  
## 98 53.41975 4.00 3.805882 22793 4081 85  
## 101 46.54717 4.00 3.763636 28835 32154 55  
## 104 26.22500 3.75 3.825000 42117 42810 40

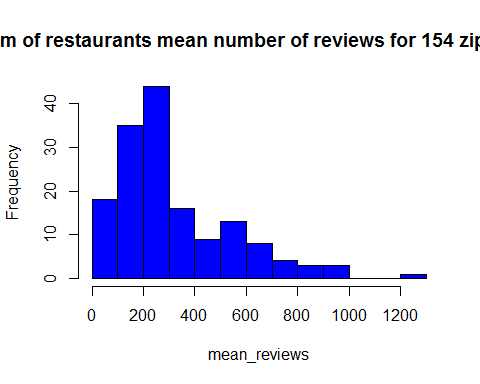
###############################################################

Some histogram visualization: (self explanatory)

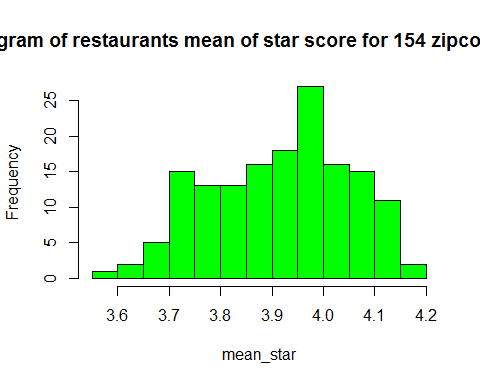
hist(zip\_credit$mean\_price,xlab="mean\_price",col = "red",main="Histogram of restaurants mean Prices for 154 zipcodes in US")



hist(zip\_credit$mean\_reviews,xlab="mean\_reviews",col = "blue",main="Histogram of restaurants mean number of reviews for 154 zipcodes in US")



hist(zip\_credit$mean\_star,xlab="mean\_star",col = "green",main="Histogram of restaurants mean of star score for 154 zipcodes in US")



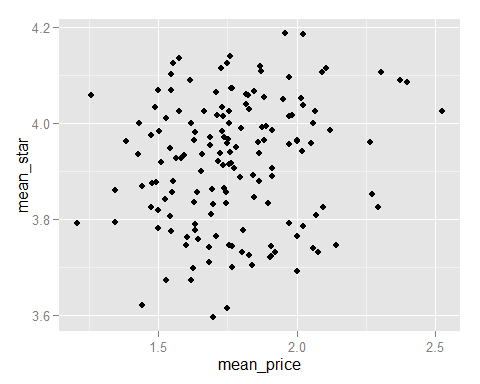
Regression Analysis:

In this section, we perform statistical and regression analysis to discover underlying significant linear relationhips between the independent variables:

library(ggplot2)  
########## plot star vs price ######################  
lm(mean\_star~mean\_price,data=zip\_credit) #Not significant

##   
## Call:  
## lm(formula = mean\_star ~ mean\_price, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) mean\_price   
## 3.76119 0.08786

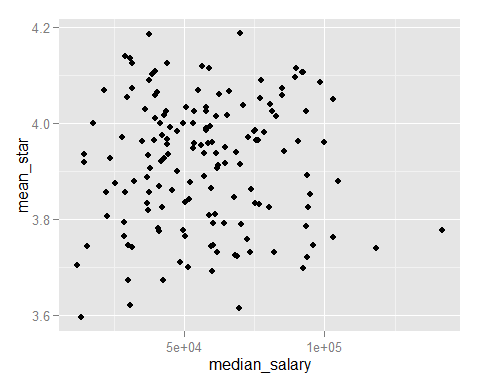
ggplot(zip\_credit, aes(x=mean\_price, y=mean\_star)) + geom\_point()



lm(mean\_star~median\_salary,data=zip\_credit) # Not significant

##   
## Call:  
## lm(formula = mean\_star ~ median\_salary, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) median\_salary   
## 3.912e+00 8.865e-08

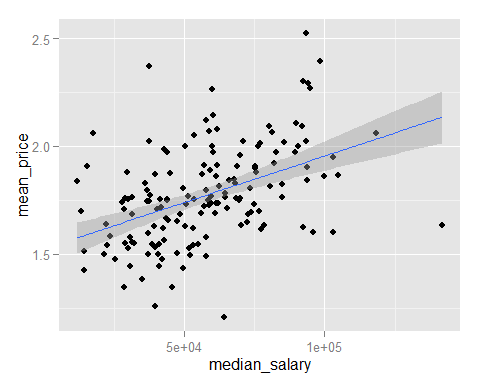
ggplot(zip\_credit, aes(x=median\_salary, y=mean\_star)) + geom\_point()



summary(lm(mean\_price~median\_salary,data=zip\_credit)) #linear relation (positive slope)

##   
## Call:  
## lm(formula = mean\_price ~ median\_salary, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.59297 -0.12427 -0.02076 0.10952 0.68549   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.525e+00 4.354e-02 35.018 < 2e-16 \*\*\*  
## median\_salary 4.305e-06 7.014e-07 6.138 6.96e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2057 on 152 degrees of freedom  
## Multiple R-squared: 0.1986, Adjusted R-squared: 0.1933   
## F-statistic: 37.67 on 1 and 152 DF, p-value: 6.961e-09

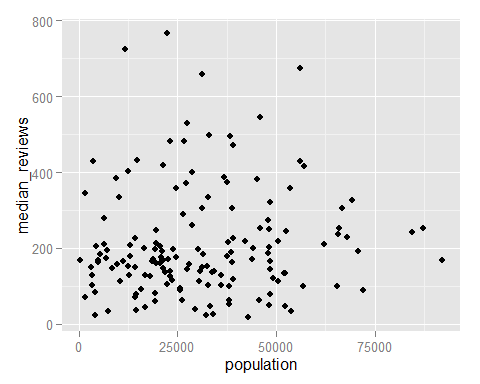
ggplot(zip\_credit, aes(x=median\_salary, y=mean\_price)) + geom\_point() + geom\_smooth(method=lm)



lm(median\_reviews~population,data=zip\_credit)# Not significant

##   
## Call:  
## lm(formula = median\_reviews ~ population, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) population   
## 1.931e+02 5.451e-04

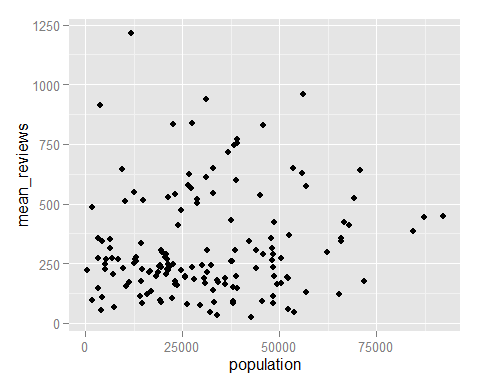
ggplot(zip\_credit, aes(x=population, y=median\_reviews)) + geom\_point()



lm(mean\_reviews~population,data=zip\_credit) #Not significant

##   
## Call:  
## lm(formula = mean\_reviews ~ population, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) population   
## 2.773e+02 1.257e-03

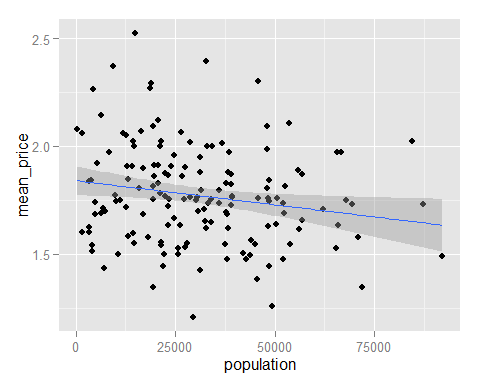
ggplot(zip\_credit, aes(x=population, y=mean\_reviews)) + geom\_point()



summary(lm(mean\_price~population,data=zip\_credit)) #linear relationship ( negetive slope)

##   
## Call:  
## lm(formula = mean\_price ~ population, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.5667 -0.1478 -0.0118 0.1205 0.7170   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.841e+00 3.466e-02 53.104 <2e-16 \*\*\*  
## population -2.235e-06 9.530e-07 -2.346 0.0203 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2257 on 152 degrees of freedom  
## Multiple R-squared: 0.03493, Adjusted R-squared: 0.02858   
## F-statistic: 5.502 on 1 and 152 DF, p-value: 0.02029

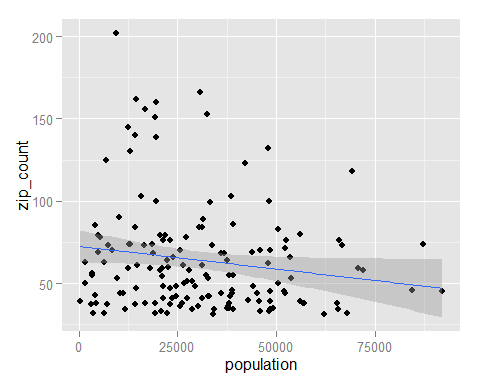
ggplot(zip\_credit, aes(x=population, y=mean\_price)) + geom\_point() + geom\_smooth(method=lm)



summary(lm(zip\_count~population,data=zip\_credit)) #linear relationship ( negetive slope) !!!

##   
## Call:  
## lm(formula = zip\_count ~ population, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -39.600 -22.796 -8.674 10.020 132.002   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 72.6286405 5.0380164 14.416 <2e-16 \*\*\*  
## population -0.0002772 0.0001385 -2.001 0.0471 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 32.81 on 152 degrees of freedom  
## Multiple R-squared: 0.02568, Adjusted R-squared: 0.01927   
## F-statistic: 4.006 on 1 and 152 DF, p-value: 0.04713

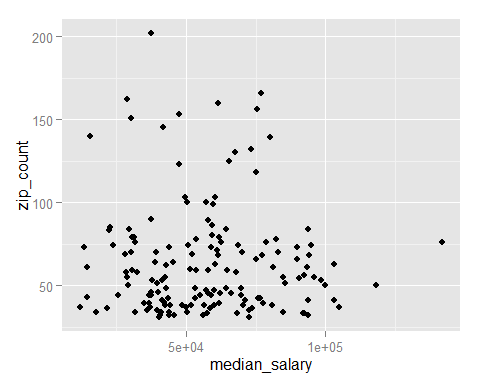
ggplot(zip\_credit, aes(x=population, y=zip\_count)) + geom\_point() + geom\_smooth(method=lm)



lm(zip\_count~median\_salary,data=zip\_credit) #Not significant

##   
## Call:  
## lm(formula = zip\_count ~ median\_salary, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) median\_salary   
## 6.724e+01 -5.573e-05

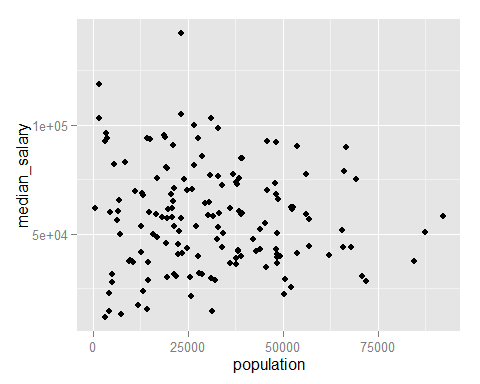
ggplot(zip\_credit, aes(x=median\_salary, y=zip\_count)) + geom\_point()



lm(median\_salary~population,data=zip\_credit) #Not significant

##   
## Call:  
## lm(formula = median\_salary ~ population, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) population   
## 6.012e+04 -8.813e-02

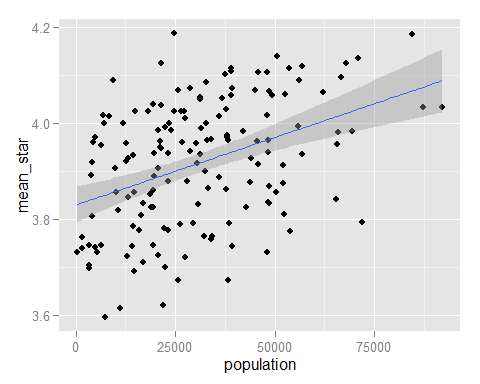
ggplot(zip\_credit, aes(x=population, y=median\_salary)) + geom\_point()



summary(lm(mean\_star~population,data=zip\_credit)) # linear relationship (positive slope)

##   
## Call:  
## lm(formula = mean\_star ~ population, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.271099 -0.098320 0.005864 0.102580 0.288198   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.830e+00 1.874e-02 204.337 < 2e-16 \*\*\*  
## population 2.812e-06 5.152e-07 5.458 1.92e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1221 on 152 degrees of freedom  
## Multiple R-squared: 0.1638, Adjusted R-squared: 0.1583   
## F-statistic: 29.79 on 1 and 152 DF, p-value: 1.92e-07

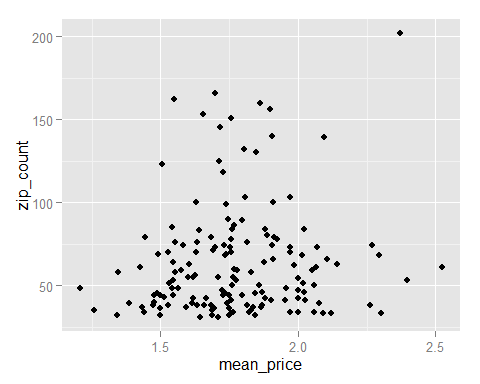
ggplot(zip\_credit, aes(x=population, y=mean\_star)) + geom\_point() + geom\_smooth(method=lm)



lm(zip\_count~mean\_price,data=zip\_credit) # Not significant

##   
## Call:  
## lm(formula = zip\_count ~ mean\_price, data = zip\_credit)  
##   
## Coefficients:  
## (Intercept) mean\_price   
## 34.60 16.62

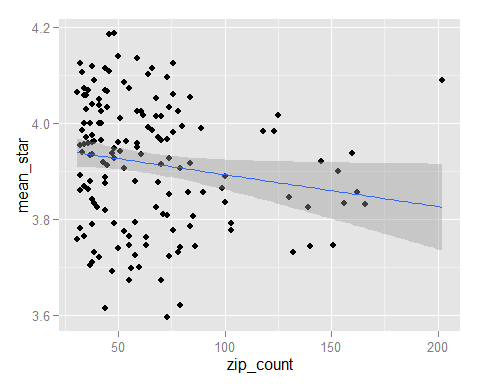
ggplot(zip\_credit, aes(x=mean\_price, y=zip\_count)) + geom\_point()



summary(lm(mean\_star~zip\_count,data=zip\_credit)) #linear relationship (negetive slope)

##   
## Call:  
## lm(formula = mean\_star ~ zip\_count, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.3166 -0.1018 0.0155 0.1034 0.2642   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.9595407 0.0231456 171.071 <2e-16 \*\*\*  
## zip\_count -0.0006665 0.0003212 -2.075 0.0397 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1316 on 152 degrees of freedom  
## Multiple R-squared: 0.02755, Adjusted R-squared: 0.02115   
## F-statistic: 4.306 on 1 and 152 DF, p-value: 0.03967

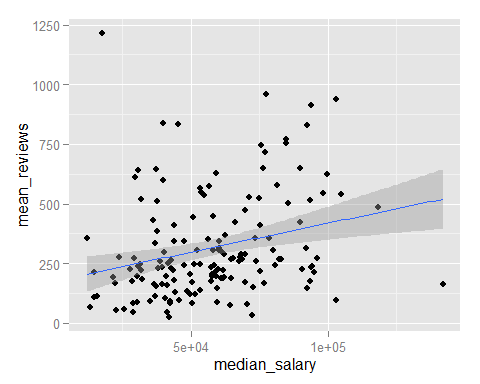
ggplot(zip\_credit, aes(x=zip\_count, y=mean\_star)) + geom\_point() + geom\_smooth(method=lm)



summary(lm(mean\_reviews~median\_salary,data=zip\_credit)) #linear relationship (positive slope)

##   
## Call:  
## lm(formula = mean\_reviews ~ median\_salary, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -356.2 -138.9 -58.3 104.5 995.1   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.774e+02 4.478e+01 3.961 0.000114 \*\*\*  
## median\_salary 2.420e-03 7.214e-04 3.354 0.001005 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 211.6 on 152 degrees of freedom  
## Multiple R-squared: 0.06891, Adjusted R-squared: 0.06279   
## F-statistic: 11.25 on 1 and 152 DF, p-value: 0.001005

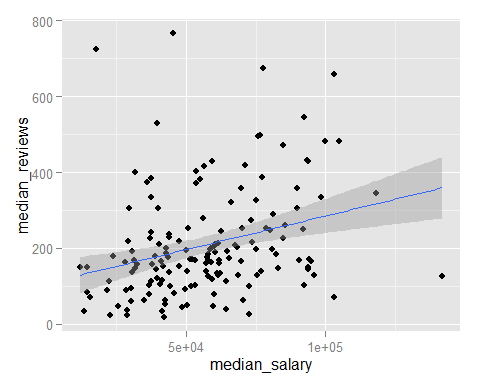
ggplot(zip\_credit, aes(x=median\_salary, y=mean\_reviews)) + geom\_point() + geom\_smooth(method=lm)



summary(lm(median\_reviews~median\_salary,data=zip\_credit)) #linear relationship (positive slope)

##   
## Call:  
## lm(formula = median\_reviews ~ median\_salary, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -234.53 -93.37 -28.33 38.29 583.40   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.084e+02 2.909e+01 3.727 0.000272 \*\*\*  
## median\_salary 1.769e-03 4.686e-04 3.776 0.000228 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 137.4 on 152 degrees of freedom  
## Multiple R-squared: 0.08574, Adjusted R-squared: 0.07973   
## F-statistic: 14.25 on 1 and 152 DF, p-value: 0.0002285

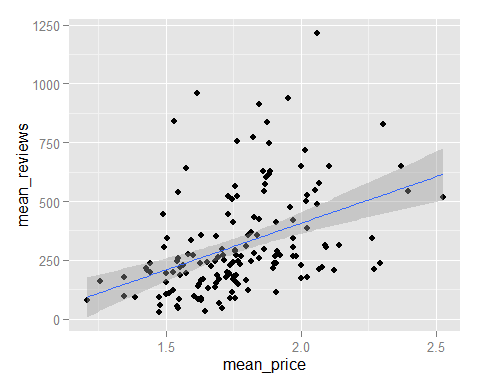
ggplot(zip\_credit, aes(x=median\_salary, y=median\_reviews)) + geom\_point() + geom\_smooth(method=lm)



summary(lm(mean\_reviews~mean\_price,data=zip\_credit)) #linear relationship (positive slope)

##   
## Call:  
## lm(formula = mean\_reviews ~ mean\_price, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -301.68 -133.57 -47.67 83.00 784.83   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -387.46 125.70 -3.083 0.00244 \*\*   
## mean\_price 397.21 70.37 5.645 7.89e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 199.4 on 152 degrees of freedom  
## Multiple R-squared: 0.1733, Adjusted R-squared: 0.1679   
## F-statistic: 31.86 on 1 and 152 DF, p-value: 7.887e-08

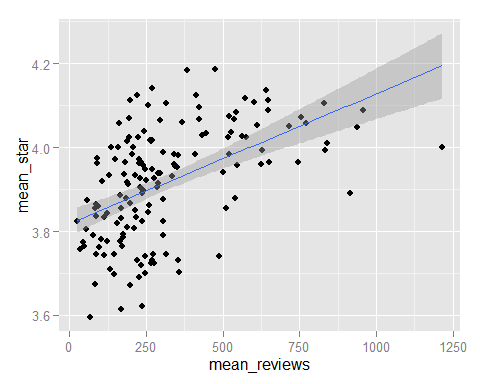
ggplot(zip\_credit, aes(x=mean\_price, y=mean\_reviews)) + geom\_point() + geom\_smooth(method=lm)



summary(lm(mean\_star~mean\_reviews,data=zip\_credit)) #linear relationship (positive slope)

##   
## Call:  
## lm(formula = mean\_star ~ mean\_reviews, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.27200 -0.07100 0.01034 0.07619 0.24710   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.819e+00 1.633e-02 233.82 < 2e-16 \*\*\*  
## mean\_reviews 3.092e-04 4.253e-05 7.27 1.77e-11 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.115 on 152 degrees of freedom  
## Multiple R-squared: 0.258, Adjusted R-squared: 0.2531   
## F-statistic: 52.85 on 1 and 152 DF, p-value: 1.771e-11

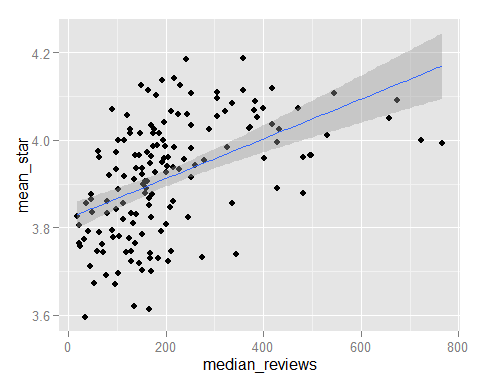
ggplot(zip\_credit, aes(x=mean\_reviews, y=mean\_star)) + geom\_point() + geom\_smooth(method=lm)



summary(lm(mean\_star~median\_reviews,data=zip\_credit)) #linear relationship (positive slope)

##   
## Call:  
## lm(formula = mean\_star ~ median\_reviews, data = zip\_credit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.28327 -0.08167 0.01140 0.08339 0.25339   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3.822e+00 1.669e-02 229.006 < 2e-16 \*\*\*  
## median\_reviews 4.537e-04 6.572e-05 6.904 1.29e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1165 on 152 degrees of freedom  
## Multiple R-squared: 0.2387, Adjusted R-squared: 0.2337   
## F-statistic: 47.66 on 1 and 152 DF, p-value: 1.29e-10

ggplot(zip\_credit, aes(x=median\_reviews, y=mean\_star)) + geom\_point() + geom\_smooth(method=lm)



###################################################

We can summerize these relationships in the following figure:

Next Page:

