Most Popular Cuisines Across the US

Kevin Chan

6 May 2018

Purpose of This Project

The purpose of this project is to find out what types of cuisines are most popular among residents and tourists of the cities of Boston, Chicago, Dallas, and San Francisco. To do this, real-time data was obtained by tapping into the Foursquare API.

Loading the Libraries Required for this Project

```
# install.packages("rgeos")
library(rgeos)
```

```
## Warning: package 'rgeos' was built under R version 3.4.4
```

```
## rgeos version: 0.3-26, (SVN revision 560)
##  GEOS runtime version: 3.6.1-CAPI-1.10.1 r0
##  Linking to sp version: 1.2-7
##  Polygon checking: TRUE
```

```
# install.packages("rgdal")
library(rgdal)
```

```
## Warning: package 'rgdal' was built under R version 3.4.4
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 3.4.3
```

```
## rgdal: version: 1.2-18, (SVN revision 718)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: C:/Users/Kevin Chan/Documents/R/win-library/3.4/rgdal/gdal
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: C:/Users/Kevin Chan/Documents/R/win-library/3.4/rgdal/proj
##  Linking to sp version: 1.2-7
```

```
library(httr)
```

```
## Warning: package 'httr' was built under R version 3.4.3
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.4.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:rgeos':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(jsonlite)
```

```
## Warning: package 'jsonlite' was built under R version 3.4.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

The Values Needed to Tap into the Foursquare API

```
client_id="GWSECN2BPBIDYSBEZD35XYTLVDYYNIHFQ3FBE2YTMO3BFJWD"
client_secret="WAZDXRFGQO12IISOOG2KZSOH34BEVDWYDDGCM4AWOWEXRF3K"
v="20180504"
query="Restaurant"
radius=10000
```

Files Necessary to Get the Latitude and Longitude of Cities

Note: These files were downloaded from https://developers.google.com/adwords/api/docs/appendix/geotargeting?csw=1.

```
cities<-read.csv("files/cities_google.csv")
states<-read.csv("files/states_google.csv")

states<-states[,-4]
names(states)[1]<-"Parent.ID"
city.state<-merge(cities,states,by="Parent.ID",all=TRUE)
city.state<-city.state[,-c(9,10,11,12)]
names(city.state)<-c("Parent.ID","Criteria.ID","City","Canonical.Name","County.Code","Target", "Status"

abb<-read.csv("files/abbreviations.csv")

city_state<-merge(city.state,abb,by="State")
city_state[,10]<-as.character(city_state[,10])
city_state[,4]<-as.character(city_state[,4])
```

Functions to Obtain the Geocode for Cities

Note: The code is from http://stackoverflow.com/questions/27867846/quick-way-to-get-longitude-latitude-fr

```
geo_init <- function() {

  try({
    GET("http://www.mapcruzin.com/fcc-wireless-shapefiles/cities-towns.zip",
        write_disk("cities.zip"))
    unzip("cities.zip", exdir="cities") })

  shp <- readOGR("cities-towns/citiesx020.shp", "citiesx020")

  geo <-
    gCentroid(shp, byid=TRUE) %>%
    data.frame() %>%
    rename(lon=x, lat=y) %>%
    mutate(city=shp@data$NAME, state=shp@data$STATE)

}

geocode <- function(geo_db, city, state) {
  do.call(rbind.data.frame, mapply(function(x, y) {
    geo_db %>% filter(city==x, state==y)
  }, city, state, SIMPLIFY=FALSE))
}
geo_db <- geo_init()
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\Kevin Chan\Documents\BU Courses\MA415\MA415-Final-Project\cities-towns\citiesx020.s
## with 28706 features
## It has 11 fields
## Integer64 fields read as strings:  CITIESX020 POP
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.3
```

Using the Functions Above to Get the Latitude and Longitude for Boston, Chicago, Dallas, and San Francisco

```
# Boston's Latitude and Longitude
boston=geo_db %>% geocode(city="Boston",state="MA")
boston_ll=paste0(boston$lat,",",boston$lon)

# Chicago's Latitude and Longitude
chicago=geo_db %>% geocode(city="Chicago",state="IL")
chicago_ll=paste0(chicago$lat,",",chicago$lon)

# Dallas' Latitude and Longitude
dallas=geo_db %>% geocode(city="Dallas",state="TX")
dallas_ll=paste0(dallas$lat,",",dallas$lon)

# San Francisco's Latitude and Longitude
```

```
sf=geo_db %>% geocode(city="San Francisco",state="CA")
sf_ll=paste0(sf$lat,",",sf$lon)
```

The Foursquare Function that Taps into the API

```
foursquare<-function(client_id,client_secret,v,query,radius,near.df){
  foursquare.list<-list();
  require(jsonlite);
  for(i in 1:length(near.df)){
    near=near.df[i];
    fqs.query=paste0("https://api.foursquare.com/v2/venues/search?client_id=",client_id,
                     "&client_secret=",client_secret,
                     "&intent=browse",
                     "&v=",v,
                     "&radius=",radius,
                     "&ll=",near,
                     "&query=",query);
    fqs.request=readLines(fqs.query);
    foursquare.list[[i]]<-fromJSON(fqs.request,simplifyDataFrame=TRUE);
    Sys.sleep(20);
  }
  return(foursquare.list);
}
```

Calling the Foursquare API to obtain restaurant data for each of the cities

```
# Boston's Restaurant Data
boston_restaurants=foursquare(client_id,client_secret,v,query,radius,boston_ll)
```

```
## Warning in readLines(fqs.query): incomplete final line
## found on 'https://api.foursquare.com/v2/venues/search?
## client_id=GWSECN2BPBIDYSBEZD35XYTLVDYYNIHFQ3FBE2YTMO3BFJWD&client_secret=WAZDXRFGQO12IIS00G2KZS0H34B
```

```
# Chicago's Restaurant Data
chicago_restaurants=foursquare(client_id,client_secret,v,query,radius,chicago_ll)
```

```
## Warning in readLines(fqs.query): incomplete final line
## found on 'https://api.foursquare.com/v2/venues/search?
## client_id=GWSECN2BPBIDYSBEZD35XYTLVDYYNIHFQ3FBE2YTMO3BFJWD&client_secret=WAZDXRFGQO12IIS00G2KZS0H34B
```

```
# Dallas' Restaurant Data
dallas_restaurants=foursquare(client_id,client_secret,v,query,radius,dallas_ll)
```

```
## Warning in readLines(fqs.query): incomplete final line
## found on 'https://api.foursquare.com/v2/venues/search?
## client_id=GWSECN2BPBIDYSBEZD35XYTLVDYYNIHFQ3FBE2YTMO3BFJWD&client_secret=WAZDXRFGQO12IIS00G2KZS0H34B
```

```
# San Francisco's Restaurant Data
sf_restaurants=foursquare(client_id,client_secret,v,query,radius,sf_ll)
```

```
## Warning in readLines(fqs.query): incomplete final line
## found on 'https://api.foursquare.com/v2/venues/search?
## client_id=GWSECN2BPBIDYSBEZD35XYTLVDYYNIHFQ3FBE2YTMO3BFJWD&client_secret=WAZDXRFGQO12IIS00G2KZS0H34B
```

Function to Obtain List of Restaurants from Foursquare

```
get_business<-function(list){
  return(list$response$venues)
}
```

Boston's Restaurant Data

```
## Obtaining list of Boston's restaurants from Foursquare.
boston_food<-lapply(boston_restaurants,get_business)
boston_test<-boston_food[sapply(boston_food, length)>=1]
boston_test<-lapply(boston_test,flatten)

boston_foodplaces<-rbind.fill(lapply(boston_test,function(y){as.data.frame((y),stringsAsFactors=F)}))
boston_foodplaces<-boston_foodplaces[!duplicated(boston_foodplaces$id),]

## Cleaning list of Boston's restaurants.
boston_temp<-data.frame(t(sapply(boston_foodplaces$categories,c)))

clean_list_data_frame<-function(data){
  for(i in 1:ncol(data)){
    data[,i]<-as.character(data[,i])
  }
  return(data)
}

boston_temp<-clean_list_data_frame(boston_temp)
boston_foodplaces<-within(boston_foodplaces,rm(categories,venueChains,location.formattedAddress,hereNow
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'venueChains' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'hereNow.groups' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'specials.items' not found
```

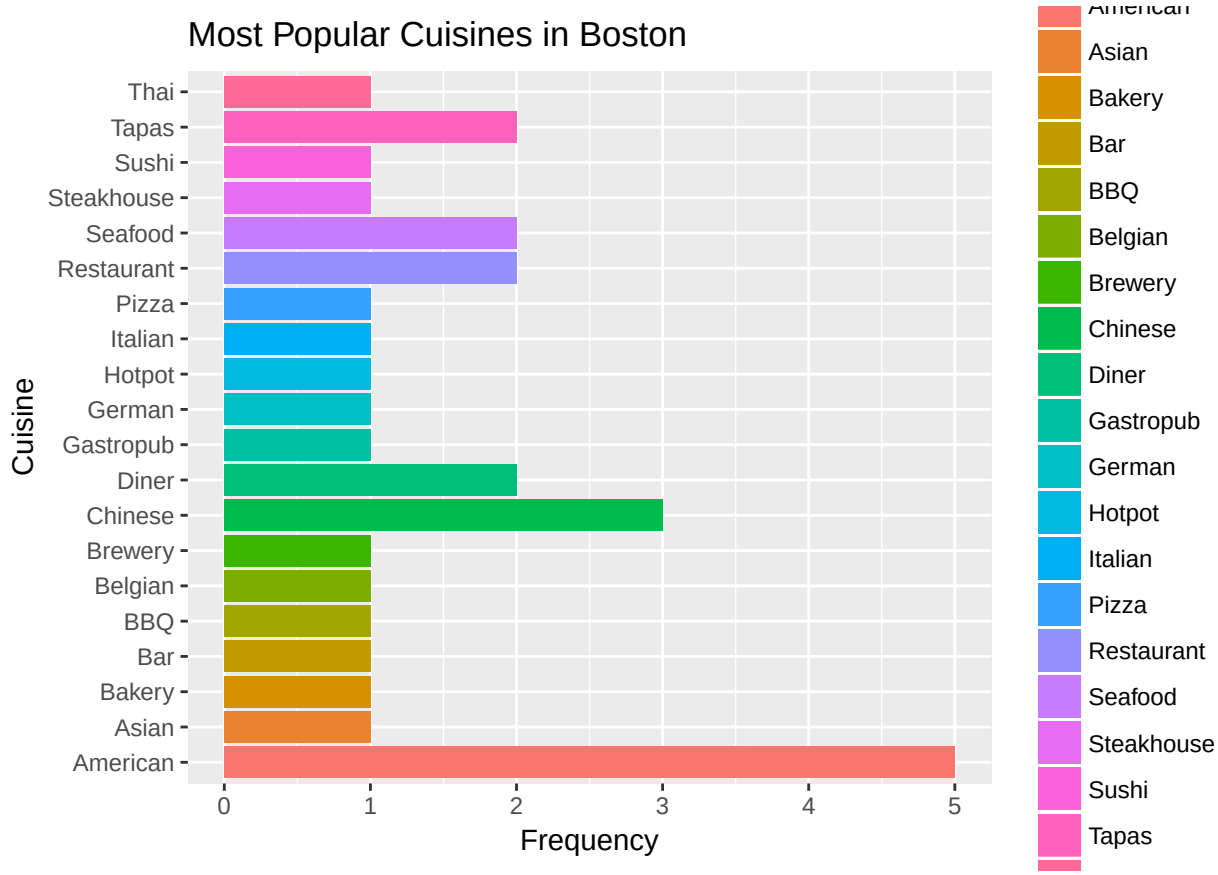
```
boston_foodplaces<-cbind(boston_foodplaces,boston_temp)
boston_foodplaces <- boston_foodplaces[,!duplicated(colnames(boston_foodplaces))]
```

Graph of Boston's Most Popular Cuisines

```
detach("package:plyr",unload=T)
```

```
## Warning: 'plyr' namespace cannot be unloaded:
##   namespace 'plyr' is imported by 'scales', 'ggplot2' so cannot be unloaded
```

```
boston_cuisine <- boston_foodplaces %>% group_by(shortName) %>% summarise(count = n())
ggplot(boston_cuisine, aes(x = shortName, y = count, fill = shortName)) + geom_bar(stat = "identity") +
      labs(title = "Most Popular Cuisines in Boston", x = "Cuisine", y="Frequency")
```



Conclusion: The most popular cuisine in Boston is American cuisine, followed by Chinese cuisine. Some notes to consider are that the API collects real-time data and that it only collects a small sample of all the restaurants, so the data collected and graph are subject to change.

Chicago's Restaurant Data

```
## Obtaining list of Chicago's restaurants from Foursquare.
chicago_food<-lapply(chicago_restaurants,get_business)
chicago_test<-chicago_food[sapply(chicago_food, length)>=1]
chicago_test<-lapply(chicago_test,flatten)

library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.4.3
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
chicago_foodplaces<-rbind.fill(lapply(chicago_test,function(y){as.data.frame((y),stringsAsFactors=F)}))
chicago_foodplaces<-chicago_foodplaces[!duplicated(chicago_foodplaces$id),]

## Cleaning list of Chicago's restaurants.
chicago_temp<-data.frame(t(sapply(chicago_foodplaces$categories,c)))

chicago_temp<-clean_list_data_frame(chicago_temp)
chicago_foodplaces<-within(chicago_foodplaces,rm(categories,venueChains,location.formattedAddress,hereN
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'venueChains' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'hereNow.groups' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'specials.items' not found
```

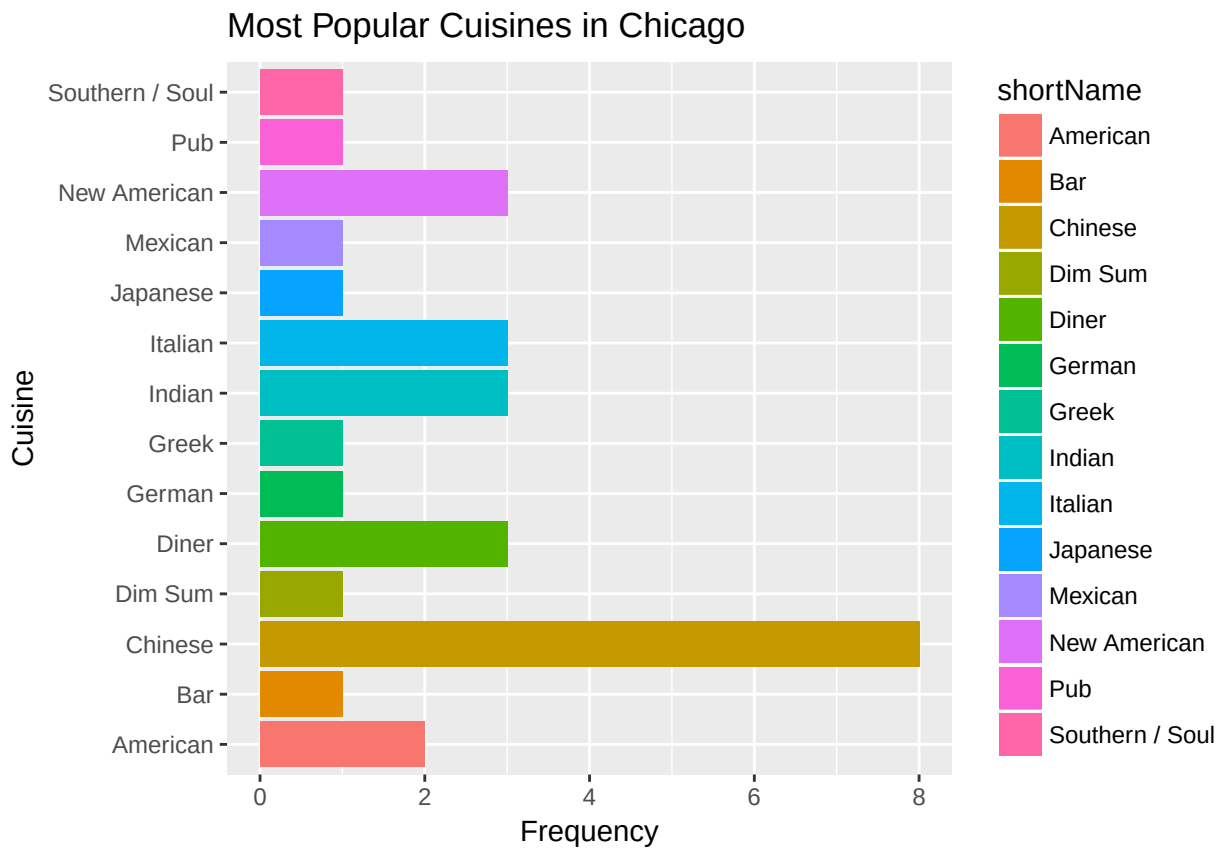
```
chicago_foodplaces<-cbind(chicago_foodplaces,chicago_temp)
chicago_foodplaces <- chicago_foodplaces[,!duplicated(colnames(chicago_foodplaces))]
```

Graph of Chicago's Most Popular Cuisines

```
detach("package:plyr",unload=T)
```

```
## Warning: 'plyr' namespace cannot be unloaded:
##   namespace 'plyr' is imported by 'scales', 'ggplot2' so cannot be unloaded
```

```
chicago_cuisine <- chicago_foodplaces %>% group_by(shortName) %>% summarise(count = n())
ggplot(chicago_cuisine, aes(x = shortName, y = count, fill = shortName)) + geom_bar(stat = "identity") +
  labs(title = "Most Popular Cuisines in Chicago", x = "Cuisine", y="Frequency")
```



Conclusion: The most popular cuisine in Chicago is Chinese cuisine, followed by Italian and Indian cuisine as well as New American and Diner-style establishments. Some notes to consider are that the API collects real-time data and that it only collects a small sample of all the restaurants, so the data collected and graph are subject to change.

Dallas' Restaurant Data

```
## Obtaining list of Dallas' restaurants from Foursquare.
dallas_food<-lapply(dallas_restaurants,get_business)
dallas_test<-dallas_food[sapply(dallas_food, length)>=1]
dallas_test<-lapply(dallas_test,flatten)

library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.4.3
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
dallas_foodplaces<-rbind.fill(lapply(dallas_test,function(y){as.data.frame((y),stringsAsFactors=F)}))
dallas_foodplaces<-dallas_foodplaces[!duplicated(dallas_foodplaces$id),]
## Cleaning list of Dallas' restaurants.
dallas_temp<-data.frame(t(sapply(dallas_foodplaces$categories,c)))

dallas_temp<-clean_list_data_frame(dallas_temp)
dallas_foodplaces<-within(dallas_foodplaces,rm(categories,venueChains,location.formattedAddress,hereNow
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'venueChains' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'hereNow.groups' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'specials.items' not found
```

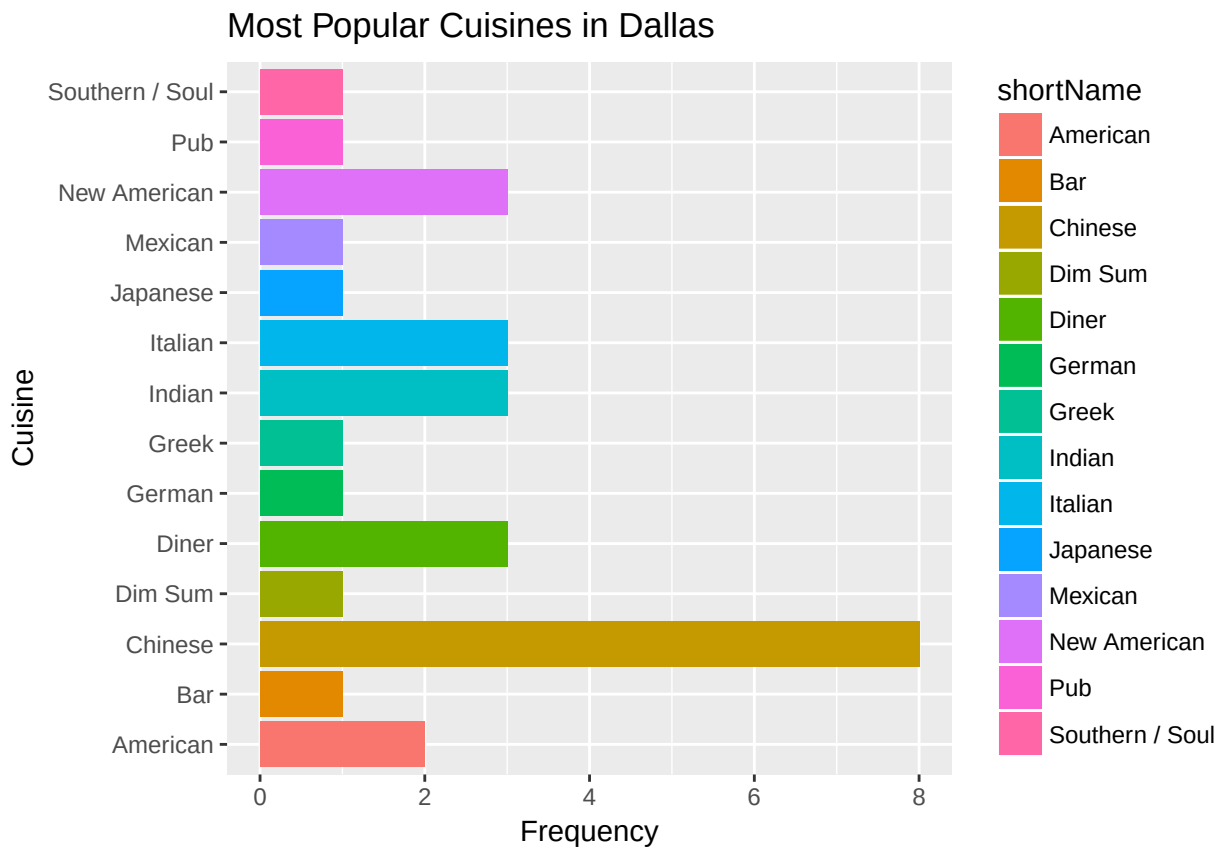
```
dallas_foodplaces<-cbind(dallas_foodplaces,chicago_temp)
dallas_foodplaces <- dallas_foodplaces[,!duplicated(colnames(dallas_foodplaces))]
```

Graph of Dallas' Most Popular Cuisines

```
detach("package:plyr", unload=T)
```

```
## Warning: 'plyr' namespace cannot be unloaded:
##   namespace 'plyr' is imported by 'scales', 'ggplot2' so cannot be unloaded
```

```
dallas_cuisine <- dallas_foodplaces %>% group_by(shortName) %>% summarise(count = n())
ggplot(dallas_cuisine, aes(x = shortName, y = count, fill = shortName)) + geom_bar(stat = "identity") +
  labs(title = "Most Popular Cuisines in Dallas", x = "Cuisine", y="Frequency")
```



Conclusion: The most popular cuisine in Dallas is Chinese cuisine, followed by Italian and Indian cuisine as well as New American and Diner-style establishments. Some notes to consider are that the API collects real-time data and that it only collects a small sample of all the restaurants, so the data collected and graph are subject to change.

San Francisco's Restaurant Data

```
## Obtaining list of San Francisco's restaurants from Foursquare.
sf_food<-lapply(sf_restaurants,get_business)
sf_test<-sf_food[sapply(sf_food, length)>=1]
sf_test<-lapply(sf_test,flatten)

library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.4.3
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
sf_foodplaces<-rbind.fill(lapply(sf_test,function(y){as.data.frame((y),stringsAsFactors=F)}))
sf_foodplaces<-sf_foodplaces[!duplicated(sf_foodplaces$id),]

## Cleaning list of San Francisco's restaurants.
sf_temp<-data.frame(t(sapply(sf_foodplaces$categories,c)))

sf_temp<-clean_list_data_frame(sf_temp)
sf_foodplaces<-within(sf_foodplaces,rm(categories,venueChains,location.formattedAddress,hereNow.groups,s
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'venueChains' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'hereNow.groups' not found
```

```
## Warning in rm(categories, venueChains, location.formattedAddress,
## hereNow.groups, : object 'specials.items' not found
```

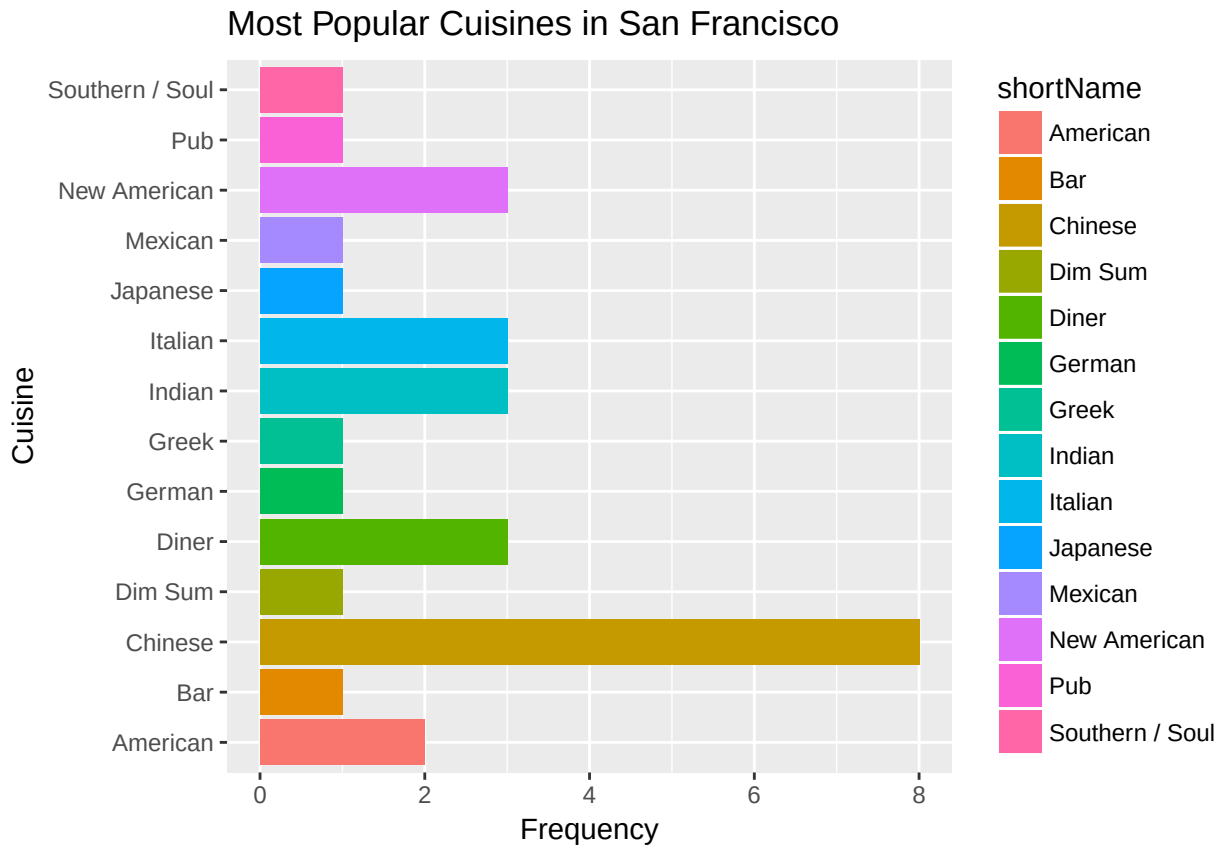
```
sf_foodplaces<-cbind(sf_foodplaces,chicago_temp)
sf_foodplaces <- sf_foodplaces[,!duplicated(colnames(sf_foodplaces))]
```

Graph of San Francisco's Most Popular Cuisines

```
detach("package:plyr",unload=T)
```

```
## Warning: 'plyr' namespace cannot be unloaded:
##   namespace 'plyr' is imported by 'scales', 'ggplot2' so cannot be unloaded
```

```
sf_cuisine <- sf_foodplaces %>% group_by(shortName) %>% summarise(count = n())
ggplot(sf_cuisine, aes(x = shortName, y = count, fill = shortName)) + geom_bar(stat = "identity") + coo
  labs(title = "Most Popular Cuisines in San Francisco", x = "Cuisine", y="Frequency")
```



Conclusion: The most popular cuisine in San Francisco is Chinese cuisine, followed by Italian and Indian cuisine as well as New American and Diner-style establishments. Some notes to consider are that the API collects real-time data and that it only collects a small sample of all the restaurants, so the data collected and graph are subject to change.