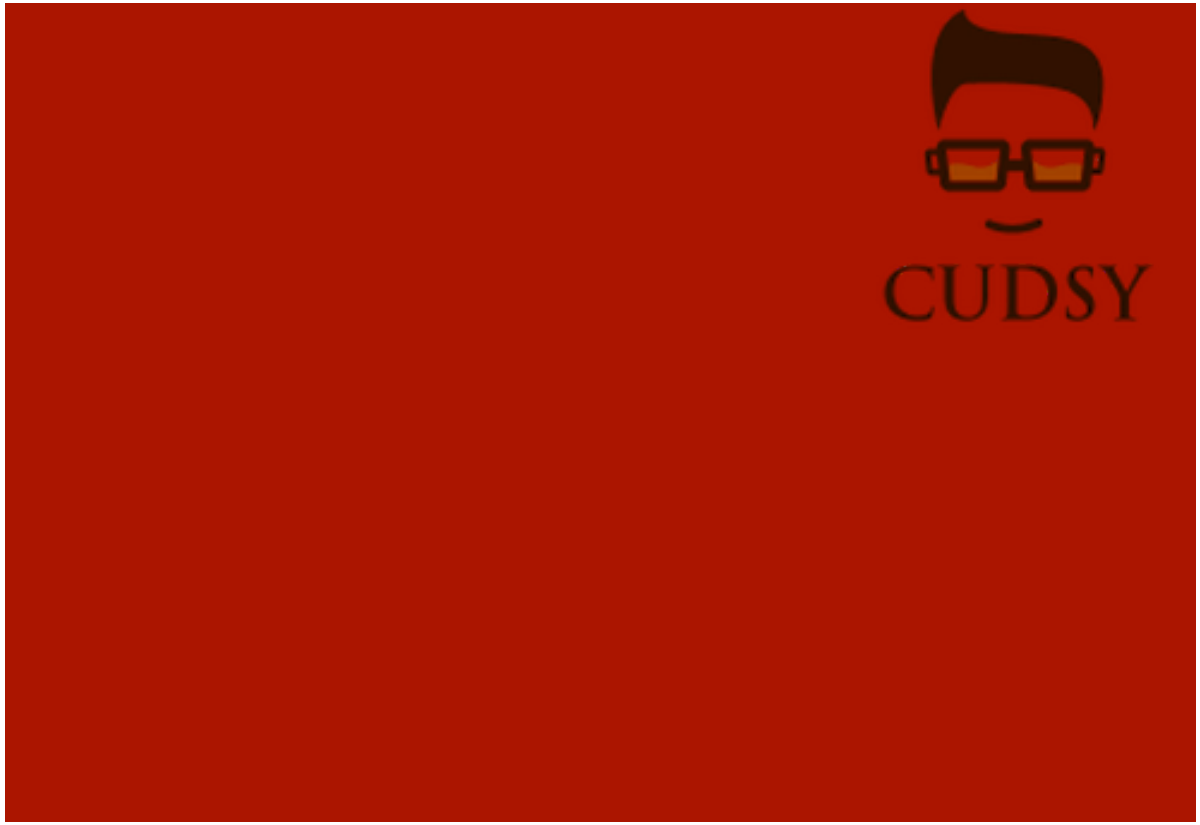


CUDSY

Columbia University Data Science Yelp

Introduction

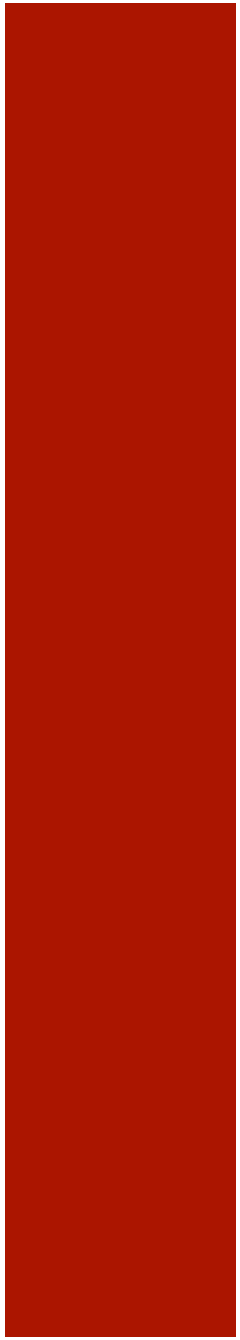
- What is the data?
  - Yelp restaurant data near Columbia
- Why analyze the Yelp data?
  - Find better way to view search results
  - Implement new features
- Our visualization concepts
  - 2D Search (User Preferences)
  - Food Map D3
    - Food Map Algorithm
  - Discovering Similar Restaurants

Note: The Yelp dataset is from 2012

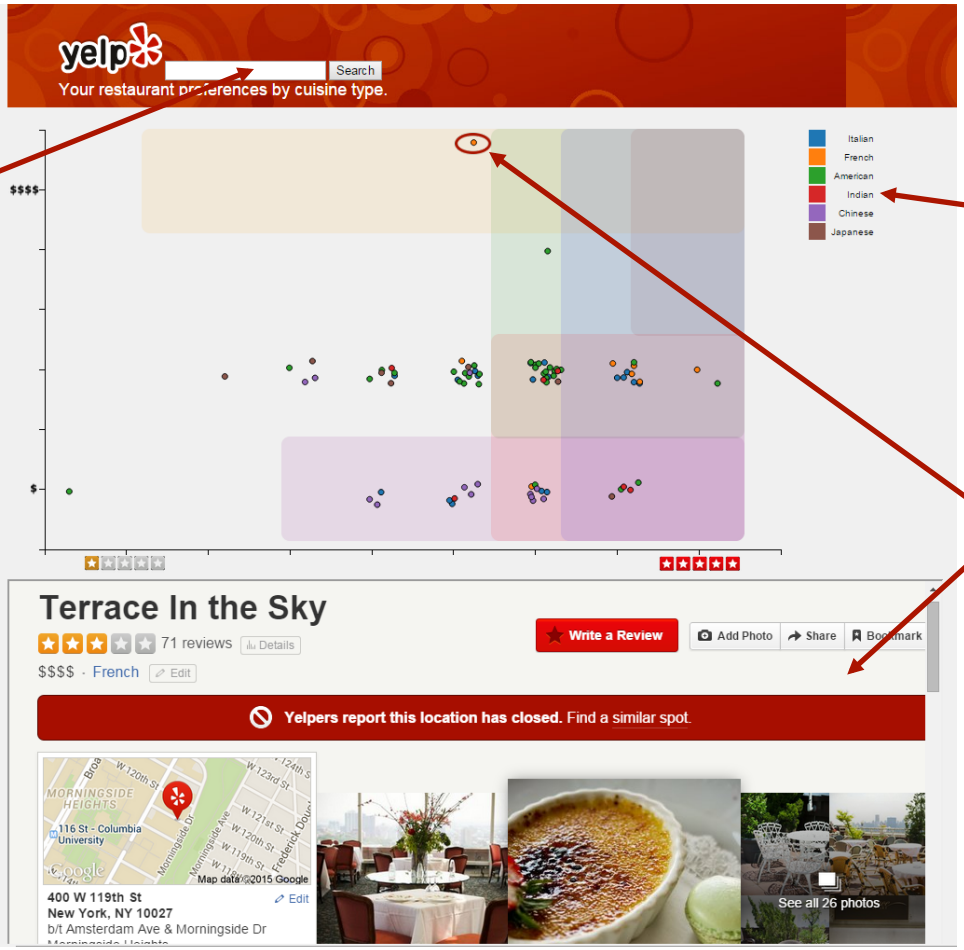
Why analyze the Yelp data?

- There are missing Yelp features that we want to see implemented
  - Given a restaurant, what other restaurants might users like (similar to Netflix recommendation system)?
  - What dishes are most popular at a particular restaurant?
  - Given user price and star preferences by cuisine, which restaurants should be recommended?

2D Search

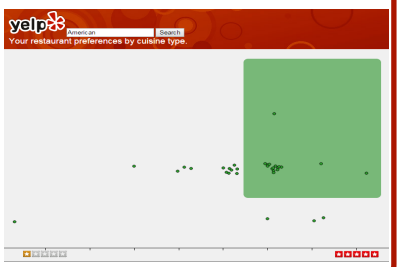


2D Search (User Preferences)



1 Users top cuisine choices

2 Filter to tonight's choice



3 One click to see details

2D Search

2D Search (R Code)

3 Jitter Samples a bit

1 Read Data into memory

2 Sqldf to clean and filter

4 Output clean .tsv for D3

```r
#function to jitter the samples
jitterSamples <- function(sample){
  for (i in 0: nrow(sample) - 1 )
  {
    sample[i,1] = sample[i,1] + runif(1, -.1, .1)
    sample[i,2] = sample[i,2] + runif(1, -.1, .1)
  }
  return(sample)
}

#Load the yelp dataset
ColBusPrices <- read.csv("ColumbiaDataJoin.csv", header = TRUE, sep = ",", quote = "\"")

#combine using sqldf into a clean set with clean categories
sqlStr <- "Select PriceRange, Stars, 'Italian' as catagory, Name, url from ColBusPrices where categories like('%Itali
           Select PriceRange, Stars, 'French' as catagory, Name, url from ColBusPrices where categories like('%French
           Select PriceRange, Stars, 'American' as catagory, Name, url from ColBusPrices where categories like('%Amer
           Select PriceRange, Stars, 'Indian' as catagory, Name, url from ColBusPrices where categories like('%Indian
           Select PriceRange, Stars, 'Chinese' as catagory, Name, url from ColBusPrices where categories like('%Chine
           Select PriceRange, Stars, 'Japanese' as catagory, Name, url from ColBusPrices where categories like('%Japa

#return clean dataset and convert factor colums to name
sample <- sqldf(sqlStr)
sample$Name = as.character(sample$Name)
sample$url = as.character(sample$url)
sample <- jitterSamples(sample)

#export to table for D3
write.table(sample, file = "C:/Users/Sam/Documents/HomeSVN/Python/EDAV/EDAV/yelpR/yelpR/output.tsv", append = FALSE,
            eol = "\n", na = "NA", dec = ".", row.names = FALSE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")
```

R Code in Markdown

2D Search (D3 Code Highlights)

Set boarders based on $ and Rating

```
function setRectangle(StarsMin, StarsMax, PriceMin, PriceMax, Color)
{
  var yLeft =  110*(4 - PriceMax) + 0;
  var ySize = 110*(PriceMax - PriceMin + 1);
  var xTop = 150*(StarsMin - 1) + 105;
  var xSize = 150*(StarsMax - StarsMin + 1) - 105 ;

  //user selection rectangle
  svg.select(".prefBorder")
   .attr("x", xTop)
   .attr("y", yLeft)
   .attr("width", xSize)
   .attr("height", ySize)
   .style("fill", Color)
   .style("opacity", .5);
   return;
}
```

Update view from search

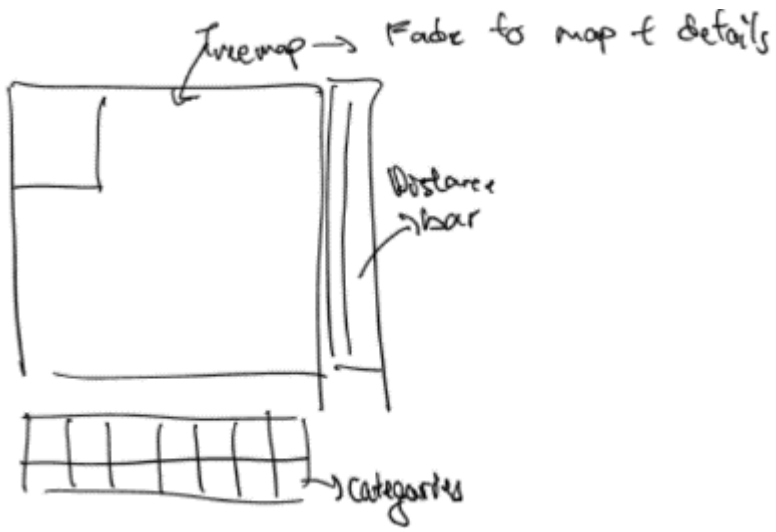
```
// ** Update data section (Called from the onclick)
function updateData()
{
  var selected = document.getElementById("fname").value;

  displayOthers = this.checked ? "inline" : "none";
  display = this.checked ? "none" : "inline";

  //hide all borders to start
  svg.select(".prefBorderAmerican").style("opacity", 0);
  svg.select(".prefBorderJapanese").style("opacity", 0);
  svg.select(".prefBorderChinese").style("opacity", 0);
  svg.select(".prefBorderItalian").style("opacity", 0);
  svg.select(".prefBorderFrench").style("opacity", 0);
  svg.select(".prefBorderIndian").style("opacity", 0);

  //set the  visible borders based on what was found in search filter
  //StarsMin, StarsMax, PriceMin, PriceMax
  switch (selected){
    case "American":
      svg.select(".prefBorderAmerican").style("opacity", .5);
      break;
    case "Japanese":
    ...
  else
  {
    svg.selectAll(".dot") // d.catagory.indexOf(selected) <      selected != d.catagory;
        .filter(function(d) {return selected != d.catagory;})
        .attr("display", displayOthers);

    svg.selectAll(".dot")
        .filter(function(d) {return selected == d.catagory;}) //d.catagory.indexOf(select
        .attr("display", display);
  }
}
```

D3 Code in Markdown

Food Map

Food Map D3

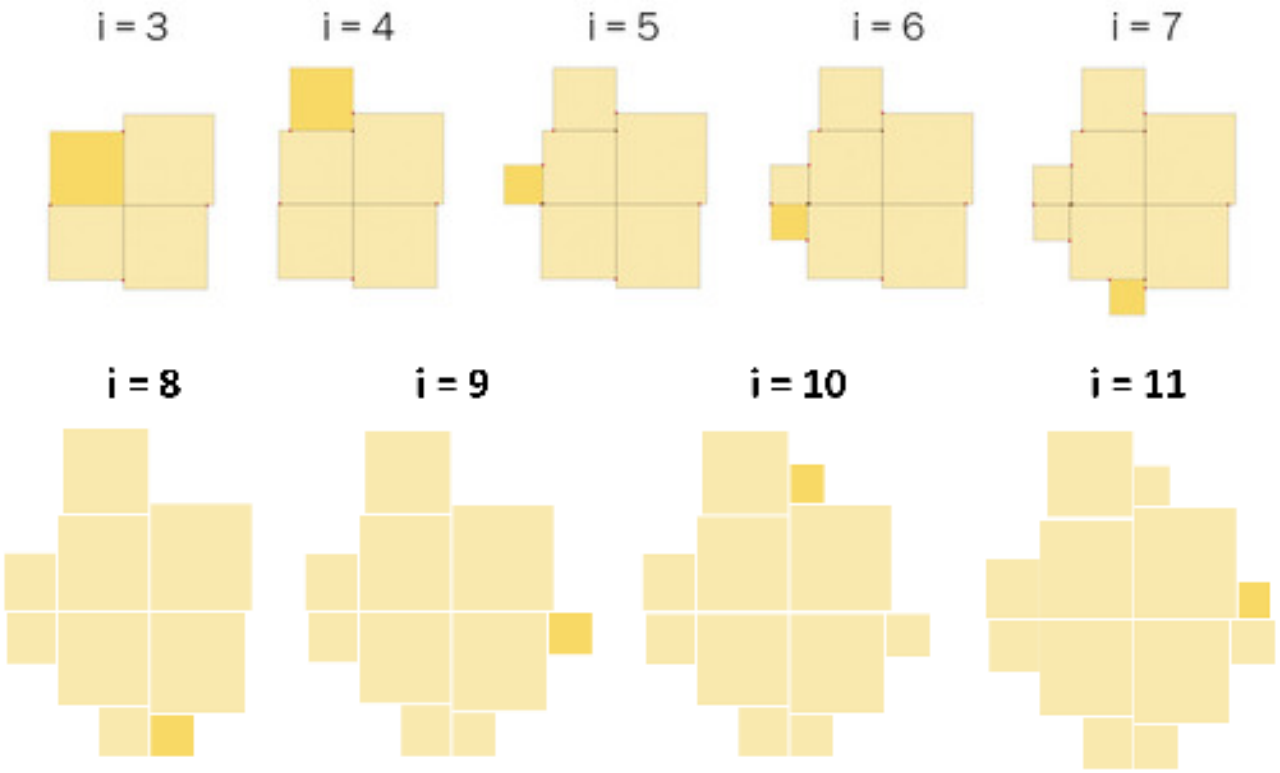
- Food, not restaurants
- Pictures, not names
- Visual, not text
- Squares, not rectangles
- Spiral, not sequential
  - Washington Post 2014 Presidential budget visualization
    http://www.washingtonpost.com/wp-srv/special/politics/presidential-budget-2014/



Algorithm for drawing squares

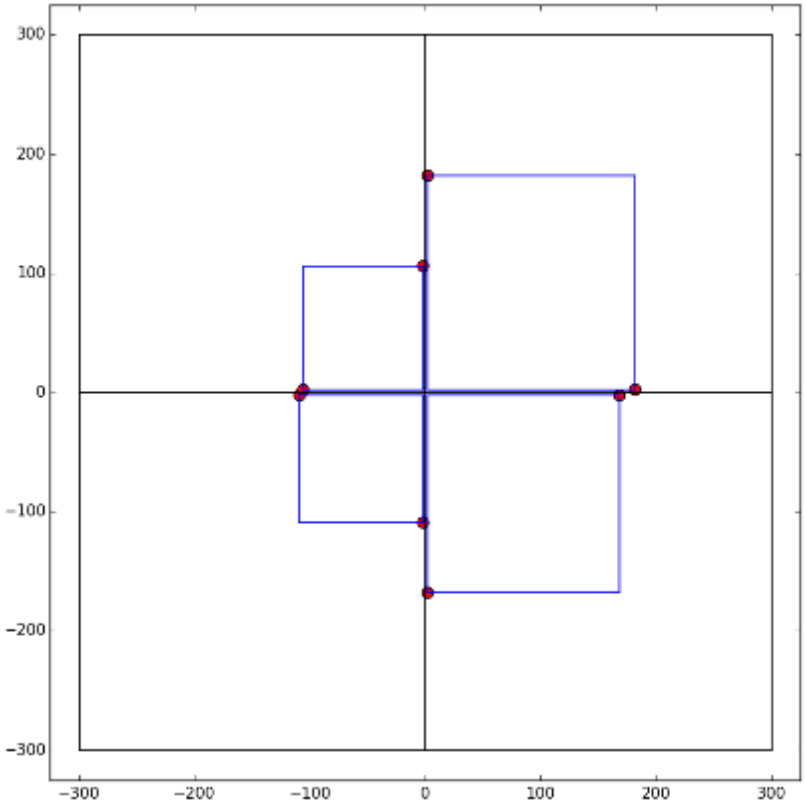
- To draw a square in D3, we need:
  1. Upper left coordinate of the square
  2. Edge length of square
- Algorithm should:
  - Input: an array of edge lengths
  - Output: coordinates and scaled lengths of squares
  - Minimize: total width and length used
  - Written in javascript so compatible with d3

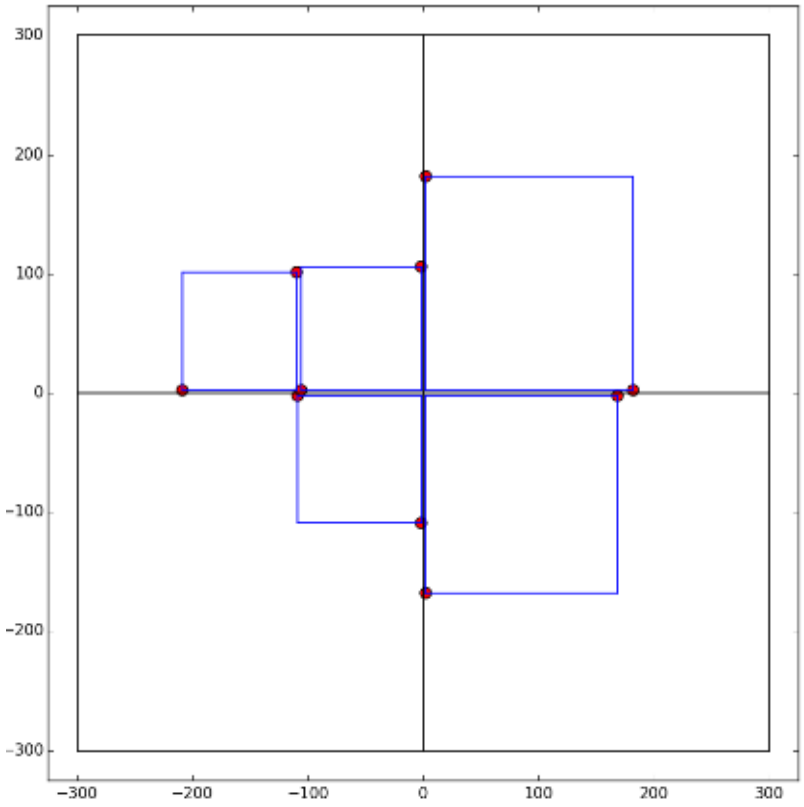
Food Map Algorithm



https://sites.google.com/site/e90e50fx/home/square-chart-and-treemap-in-excel

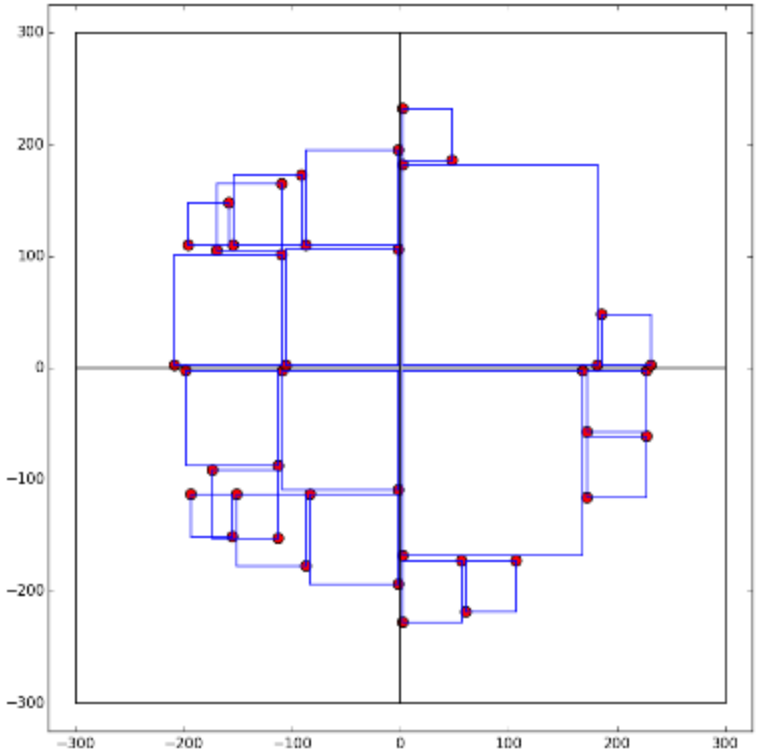
Prototyping in Python

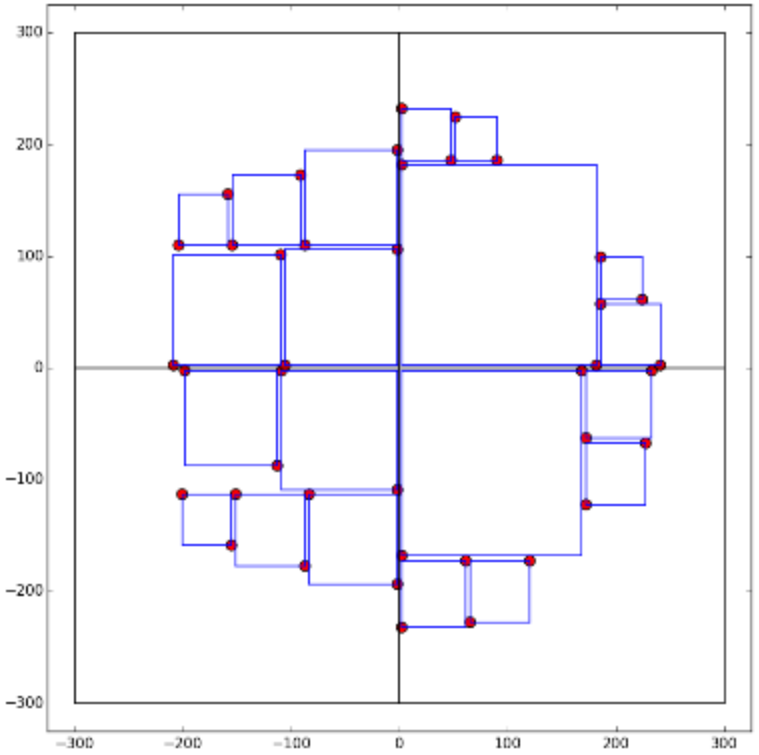




Red points in the figure represent allowable locations for placing squares

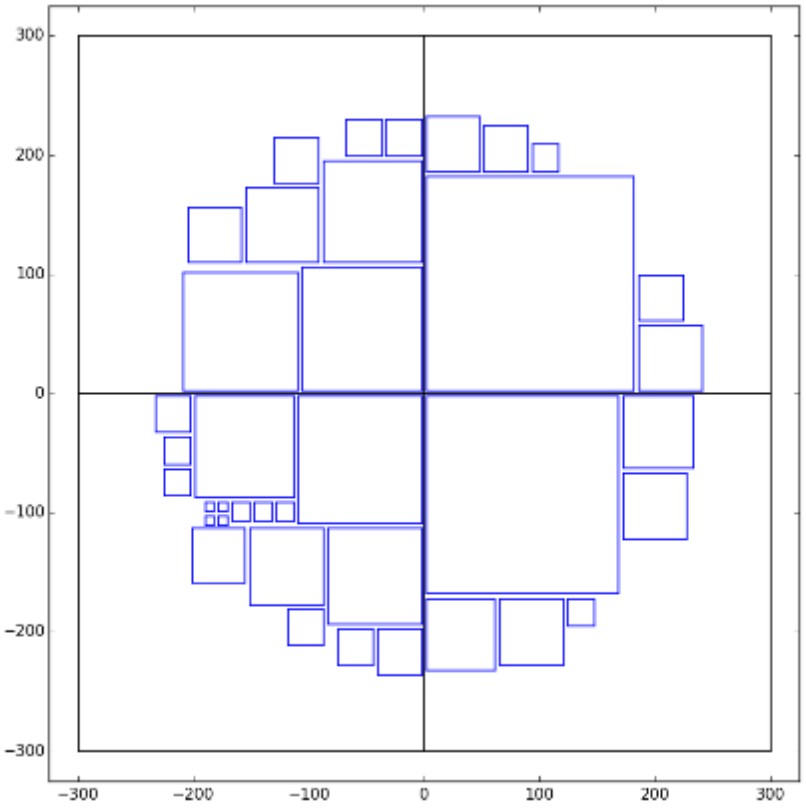
Overlapping issue





For each new optimal point, check if adding a square there would result in overlapping

Still room for improvements…



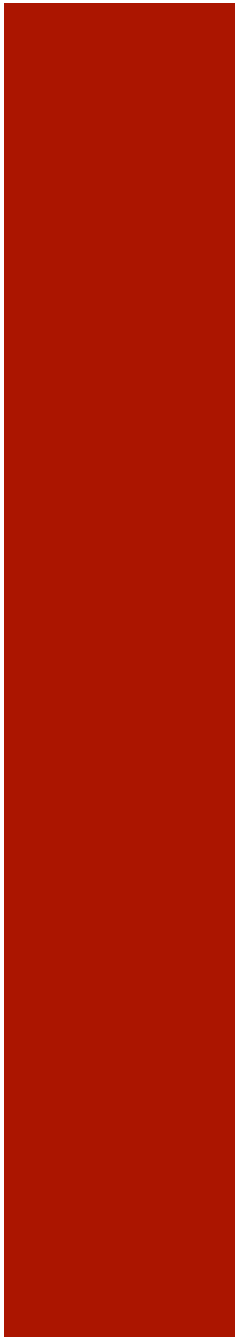
- Filling squares in empty spaces between squares (back filling)
- Scaling should be applied to all the squares
- Greedy algorithm: not guaranteed to be ideal for all squares

Food Map Data Scraping

- 196 Restaurants Listed on 2012 Data
- 93 Restaurant Menus Available
- 5801 Food Items
- 383 Food Items with Photos
- 335 Food Items with Photos and Reviews
- 54 Restaurants Remaining

- Photos are tagged to reviews
- Reviews tagged to food on Yelp are based on basic string matching
- Reviews can be tagged to more than one food

Discovering Similar Restaurants

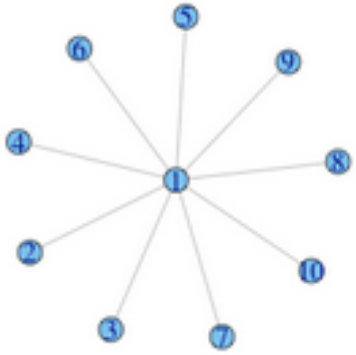


Discovering Similar Restaurants

- Feature-based recommendation
- User-based recommendation
  - Collaborative filtering
    - Jaccard Index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

  - Matrix Factorization



Discovering Similar Restaurants

- Collaborative filtering and matrix factorization to identify similar restaurants
- Leaflet with bubbles sized according to rank of similarity to visualize restaurants on map
- Shiny interface to allow:
  - Searching of restaurants
  - Sortable data table
  - Restaurant information

Discovering Similar Restaurants

Creating the Leaflet map and data table, and passing it to the Shiny output

```
m = leaflet() %>% addTiles() %>%
  addMarkers(loc$lon1[1], loc$lat1[1], icon = JS(JSlink), popup=loc$att[1]) %>%
  addCircleMarkers(data = loc, lat = lat2, lng = lon2, popup=loc$att, color = c('red',rep_len('green',10)), radius= ~ 15-rank, fillOpacity = 0.5)
data_shiny <- data.frame(name1, stars1, price1, rank1, link1)
colnames(data_shiny) <- c('Restaurant', 'Stars', 'Price', 'Similarity', 'Link')
newList=list("map"=m,"data"=data_shiny,"photo"=photo)
return(newList)
```

Shiny Output – Integrating Leaflet, data tables, and Passing restaurant name to FoodMap

```
  # Output map, restaurant information, and table of similar restaurants
  output$YelpMap = renderLeaflet(react_output()$map)
  output$table1 <- renderUI({
    data1 <- react_output()$data[1,]
    photo <- react_output()$photo
    photo <- paste0('<img src="',photo)
    photo <- paste0(photo,'" />')
    name <- paste0('<b>Restaurant: </b>', as.character(data1$Restaurant))
    stars <- paste0('<b>Stars: </b>', as.character(data1$Stars))
    price <- paste0('<b>Price: </b>', as.character(data1$Price))
    link <- as.character(data1$Link)
    output <- HTML(c(photo, '<br>', name,'<br>',stars,'<br>',price,'<br>',link))
    output
  })
  output$table2 <- renderDataTable({
    data2 <- react_output()$data[2:11,]
    data2
  }, options = list(pageLength = -1, searching = FALSE), escape = FALSE)
  # An observer is used to send messages to the client. The message is converted to JSON
  observe({ session$sendCustomMessage(type = 'FoodMap', message = list(restname = input$restname)) })
}
```

Discovering Similar Restaurants

Shiny UI – Integrating Leaflet, similar restaurants, and FoodMap

```
# Run Shiny UI - specifying title, leaflet map, restaurant information, and similar restaurant table
shinyUI = fluidPage(fluidRow(column(2,img(src='header_logo.png', width=150, height=75)),
                             column(10,titlePanel('Recommendation System'), style = "color: white"),
                             style = "background-image: url('background-red.jpg')",
  mainPanel('Once you choose a restaurant, bubble sizes indicate similarity', style = "color: white")),
  fluidRow(column(8,leafletOutput('YelpMap',width = "100%", height = 600)),
                column(4,wellPanel(span(htmlOutput("table1"))),
                         wellPanel('Most popular food items:', HTML(FoodMap), includeCSS("foodMap.css")))),
      fluidRow(
        wellPanel(
          selectizeInput('restname', "Pick a Restaurant:", choices=main_names, multiple = FALSE, options = list(placeholder = 'Type in restaurant name')),
          singleton( tags$head(tags$script(src = "d3.v3.min.js"))),
          singleton( tags$head(tags$script(src = "d3.tip.min.js"))),
          singleton( tags$head(tags$script(src = "foodMap.js")))),
      fluidRow(dataTableOutput(outputId="table2")),
      fluidRow(HTML('<a href="http://run.plnkr.co/plunks/jRIGKXMZLDn5xrMJgqTl/">Switch to Yelp User Preference Search</a>')))
```

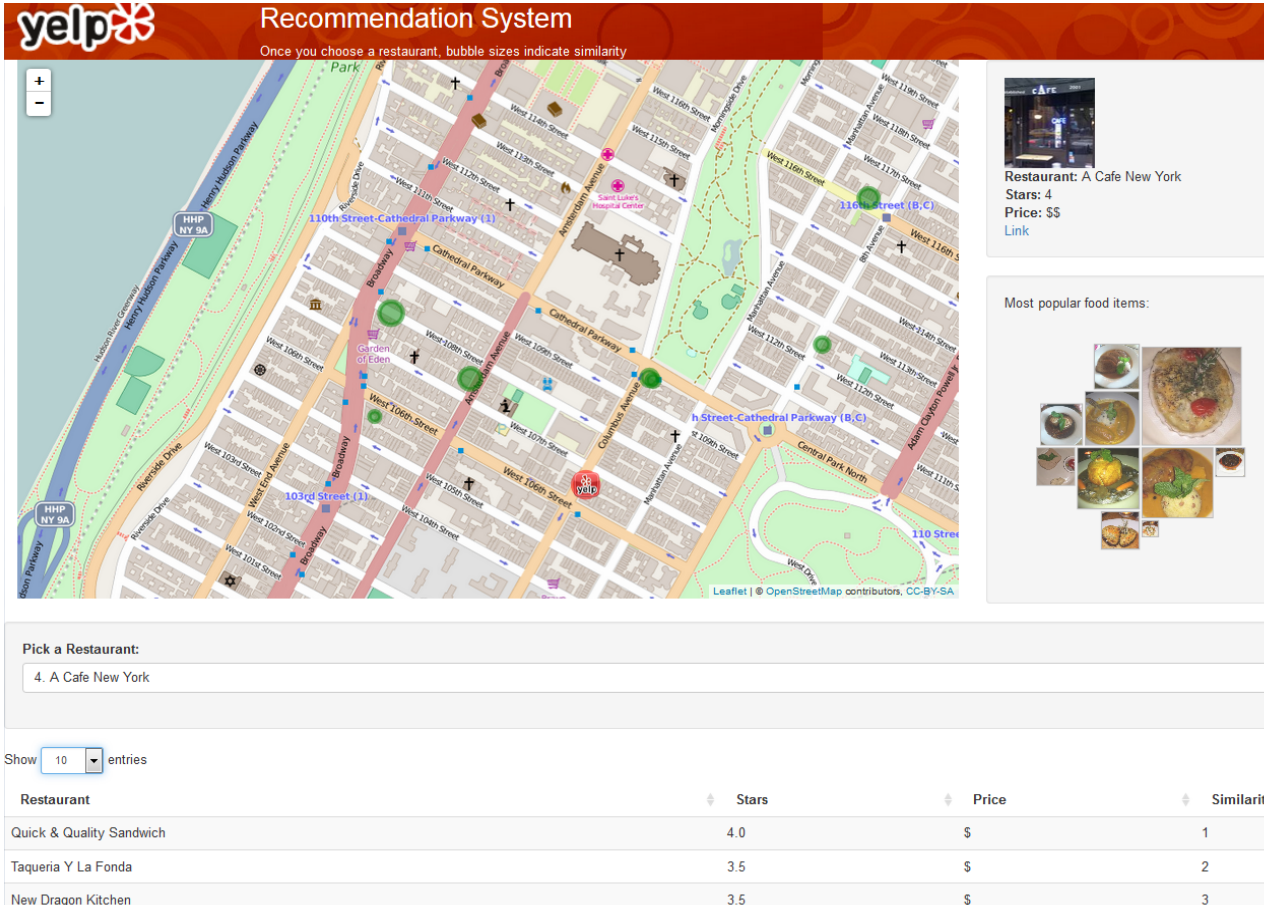
- What is a Singleton?
  - Makes sure page loads JS file in the HTML head. Call to singleton ensures it is only included once.
  - Design pattern with private constructor
  - Guarantees that only one instance of a class is created

Integration and Conclusion

Integration

- Merging FoodMap (built in D3) with Shiny/Leaflet app
  - Passing input restaurant in Shiny as variable to FoodMap.js file
  - Placing FoodMap in div tag within Shiny app
- Merging FoodMap with user preference 2D search (both built in D3)

Integrated Designs



Similarity with Food Map

2D Search With Food Map

Lessons Learned and Next Steps

- One letter global variables should be avoided (thanks, Leaflet and Steve)
- Implement Leaflet in JavaScript instead of an R Leaflet package to allow enhanced functionality
  - Clicks on map can be tracked and passed back to script

Next Steps – Food Map