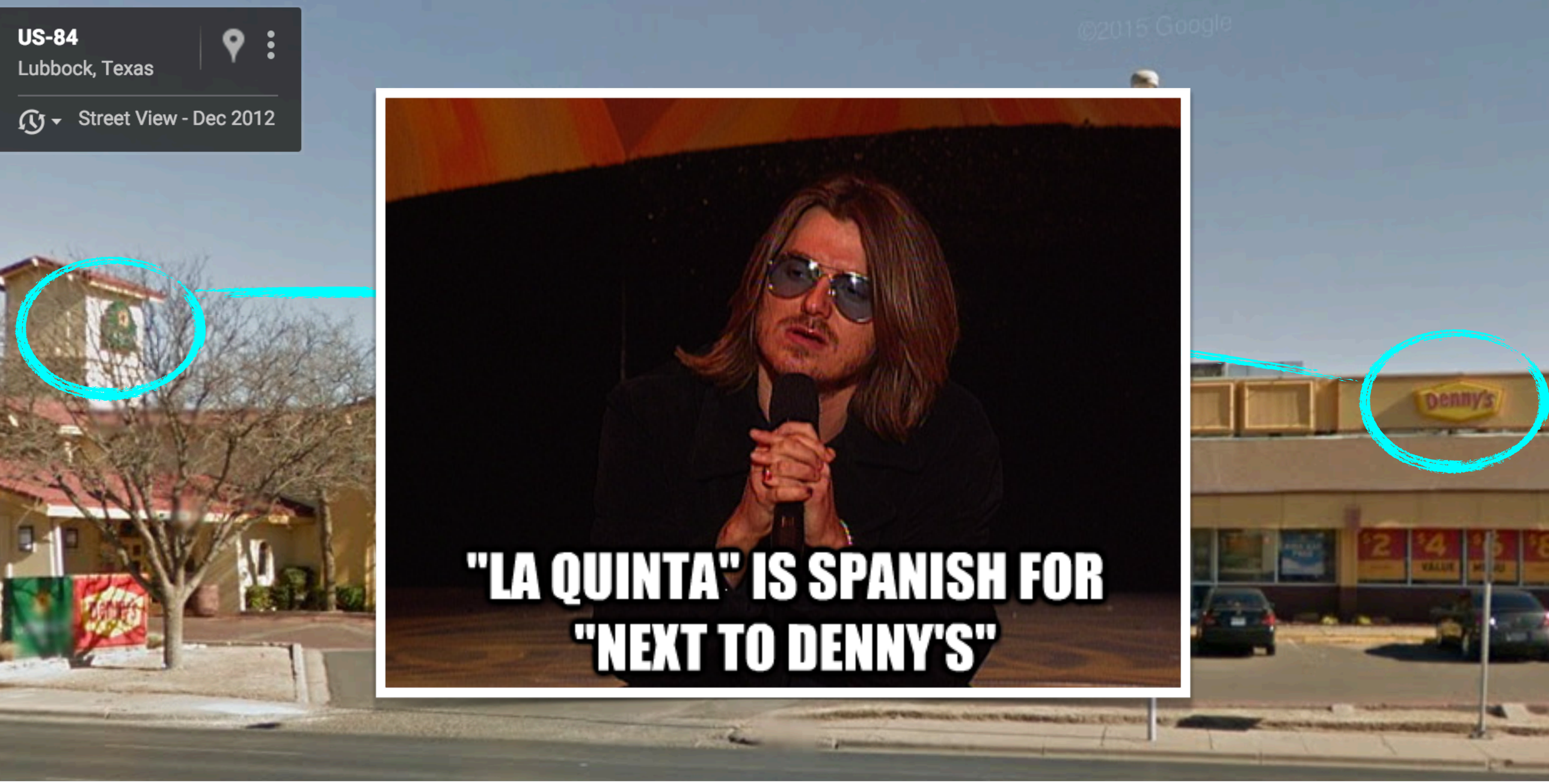
Data Science Case Study: La Quinta is Spanish for next to Denny's

Mine Çetinkaya-Rundel

RStudio®


US-84
Lubbock, Texas
Street View - Dec 2012
©2015 Google
"LA QUINTA" IS SPANISH FOR
"NEXT TO DENNY'S"
Denny's

Acknowledgement

J Reiser. (2014, Jan 30). new jersey geographer: Mitch Hedberg and GIS.

http://njgeo.org/2014/01/30/mitch-hedberg-and-gis/

Mitch Hedberg and GIS

Posted on January 30, 2014 by John Reiser

So, a recent post on Reddit highlighted a Mitch Hedberg joke.

> *"La Quinta" is Spanish for "next to Denny's."*

Thinking about this, I realized we could use GIS to find the number of La Quintas that are next to Denny's. Last night after the kids were asleep, I sat down with a beer (Sierra Nevada) and figured out how I could get the data and perform the calculation.

First, I visited La Quinta's website and their interactive map of hotel locations. Using Firebug, I found "hotelMarkers.js" which contains the locations of the chain's hotels in JSON. Using a regular expression, I converted the hotel data into CSV.

```
{"n":"Birmingham","i":"inns_suites","p":[33.353175,-86.783656],"s":"AL","c":"1"},
{"n":"Birmingham","i":"inns","p":[33.430991,-86.708318],"s":"AL","c":"1"},
{"n":"Homewood","i":"inns_suites","p":[33.455237,-86.81964],"s":"AL","c":"1"},
{"n":"Huntsville","i":"inns_suites","p":[34.73842,-86.656945],"s":"AL","c":"1"},

                    becomes

Birmingham,inns_suites,33.353175,-86.783656,AL
Birmingham,inns,33.430991,-86.708318,AL
Homewood,inns_suites,33.455237,-86.81964,AL
Huntsville,inns_suites,34.73842,-86.656945,AL

                    using

s/^.*"n":"(.+)","i":"(.+)","p":\[([\d\.\-]+),([\d\.\-]+)],"s":"(\w+)","c":"(.+)".*$/\1,\2,\3,\4,\5/;
```

Sample of the data. Click for full size.

Target audience

Those interested in

- data scraping using tidy tools,
- visualization of geospatial data, or
- a case study for a data science course.

Scraping the web: what? why?

- Increasing amount of data is available on the web, provided in an unstructured format: you can always copy&paste, but it's time-consuming and prone to errors.
- Web scraping is the process of extracting this information automatically and transform it into a structured dataset.
- Two different scenarios:
  - Screen scraping: extract data from source code of website, with html parser (straightforward) or regular expression matching (less straightforward).
  - Web APIs (application programming interface): website offers a set of structured http requests that return JSON or XML files.
- Why R? It includes all tools necessary to do web scraping, familiarity, direct analysis of data... But python, perl, java are also efficient tools.

Materials can be found at

http://bit.ly/ds-case-study

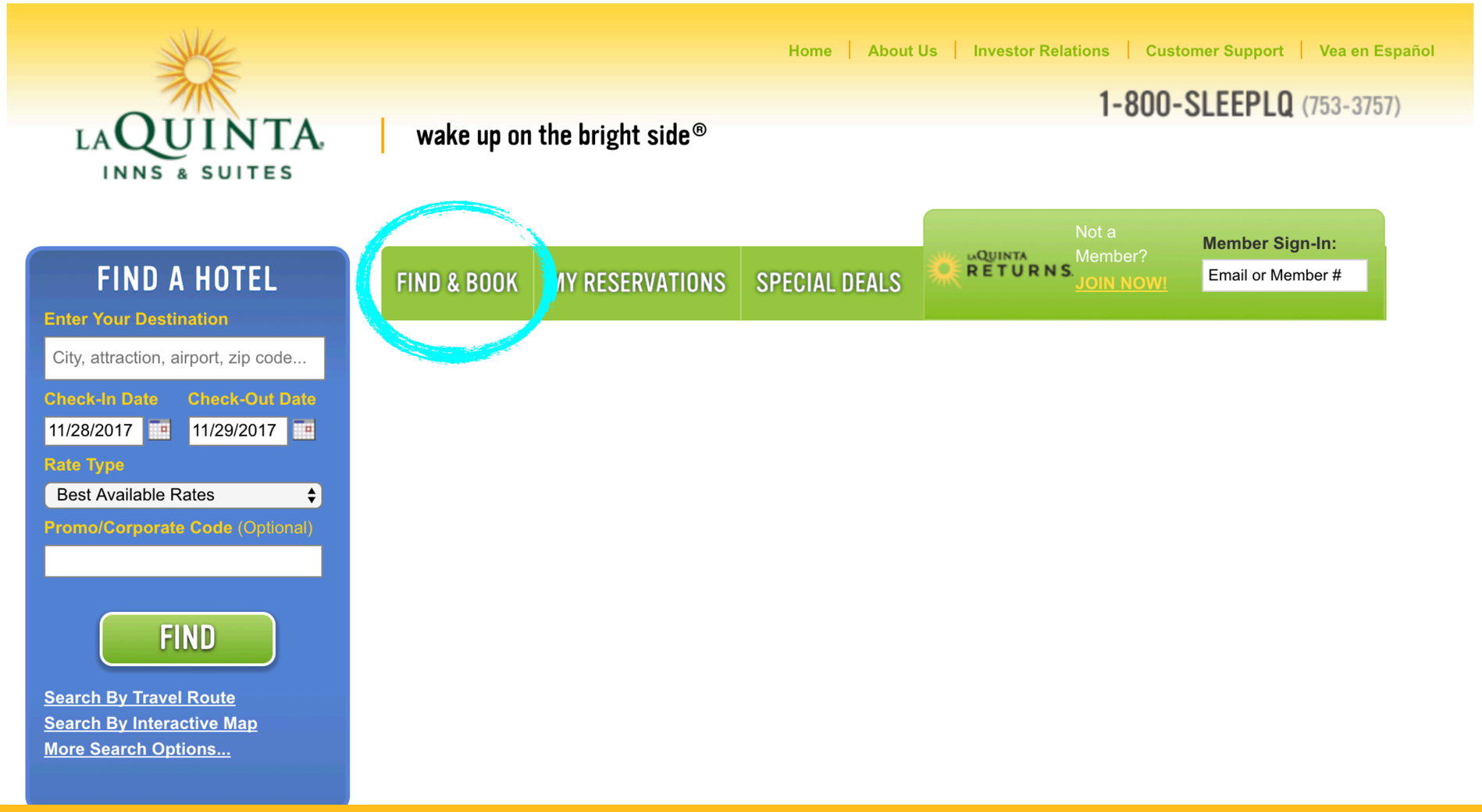
Outline

- Scraping La Quinta Inns & Suites locations
- Pointers for scraping Denny's locations
- Spatial analysis
  - Identifying nearest neighbors
  - Exploring the distance distributions
- Resources

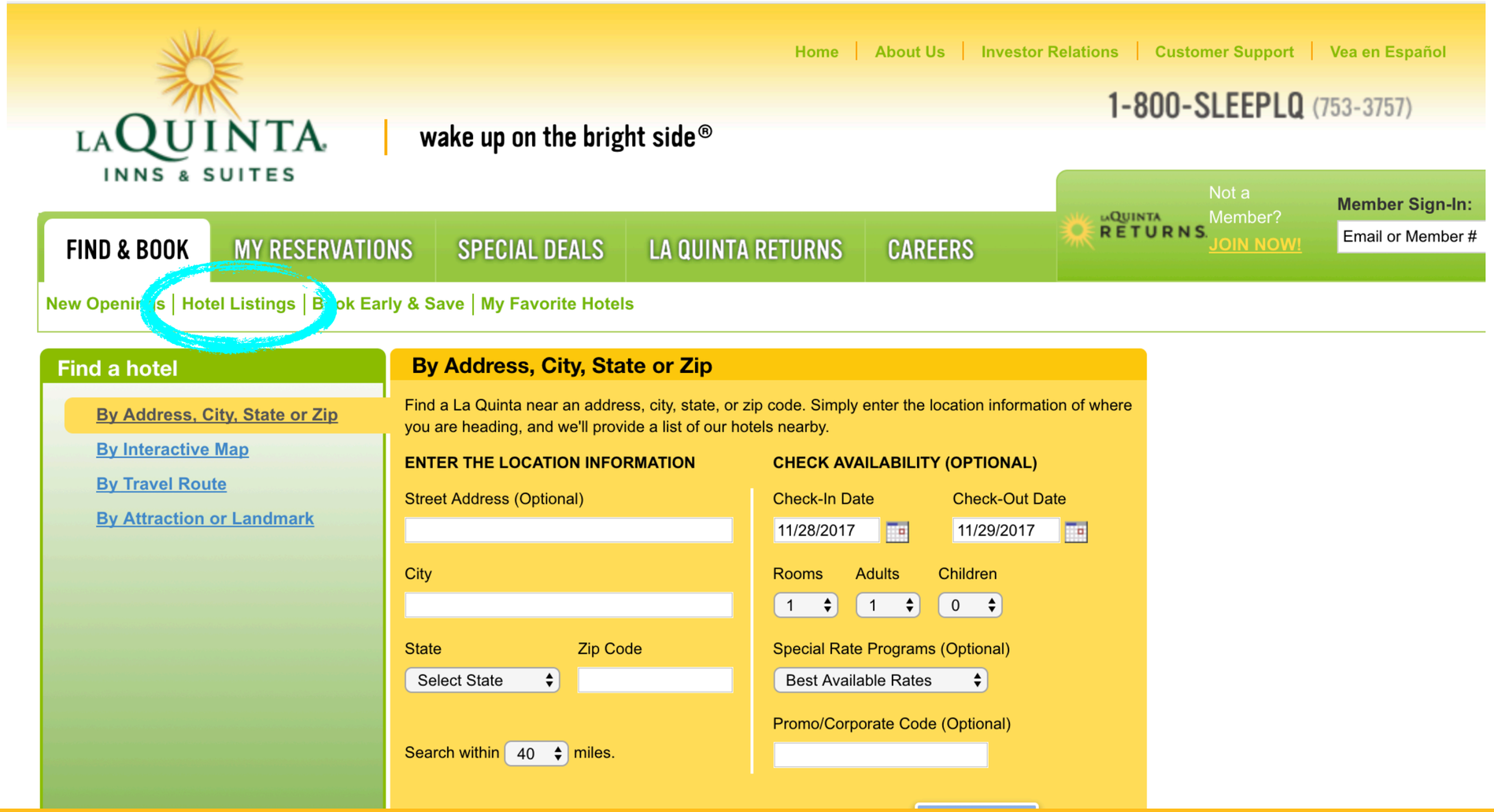
Scraping  locations

bit.ly/ds-case-study

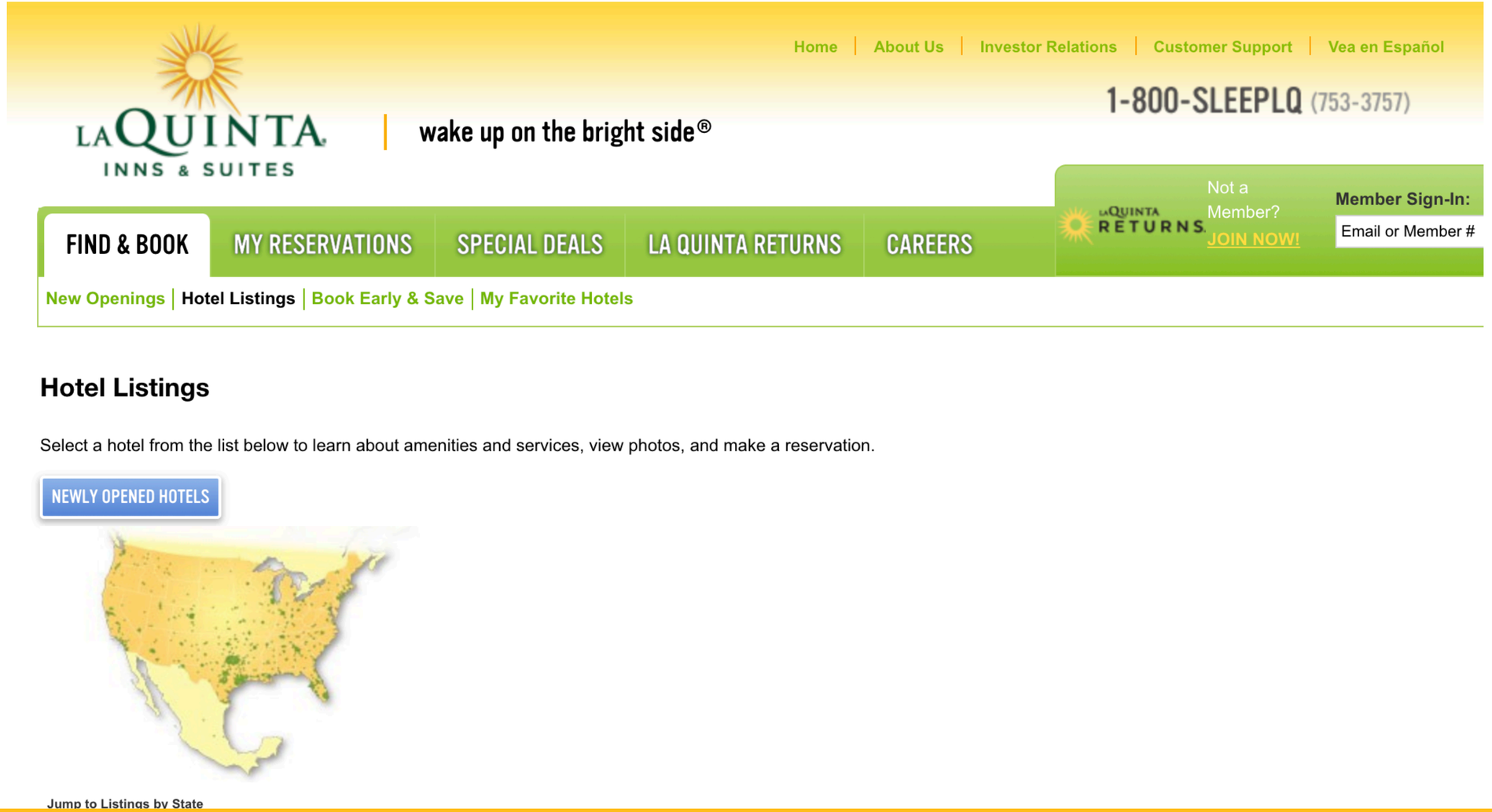
RStudio®



http://www.lq.com/en.html



http://www.lq.com/en/findandbook.html



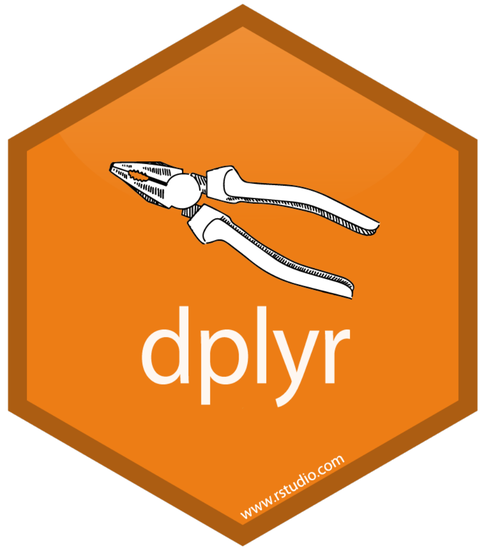
http://www.lq.com/en/findandbook/hotel-listings.html

Packages

+

+

+



+

+ `robotstxt`

rvest

- Create an html document from a url, a file on disk or a string containing html with `read_html()`.
- Select parts of a document using css selectors: `html_nodes(doc, "table td")`
- Extract components with `html_tag()` (the name of the tag), html_text() (all text inside the tag), `html_attr()` (contents of a single attribute) and `html_attrs()` (all attributes).
- Parse tables into data frames with `html_table()`.
- Extract, modify and submit forms with `html_form()`, `set_values()`, and `submit_form()`.
- Detect and repair encoding problems with `guess_encoding()` and `repair_encoding()`.

Selector gadget

- Open source tool that makes CSS selector generation and discovery on complicated sites a breeze: http://selectorgadget.com/ (add the Chrome extension)

- How does it work? Click on the selector gadget tool.
  - A box will open in the bottom right of the website.
  - Click on a page element that you would like your selector to match (it will turn **green**).
  - SelectorGadget will then generate a minimal CSS selector for that element, and will highlight (**yellow**) everything that is matched by the selector.
  - Click on a highlighted element to remove it from the selector (**red**), or click on an unhighlighted element to add it to the selector.
  - Through this process of selection and rejection, you can come up with the CSS selector for your need.

Selector gadget - learn more

Read the vignette at one of the following:

- `vignette("selectorgadget")`
- https://cran.r-project.org/web/packages/rvest/vignettes/selectorgadget.html

Follow along at

`code/01-scrape-la-quinta.R`

Scraping La Quinta locations

Step 0: Check bot permission

```
paths_allowed("http://www.lq.com/en/findandbook/hotel-listings.html")
```

Scraping La Quinta locations

Step 1: Create list of links to hotels

bit.ly/ds-case-study

RStudio®

```
base_url <- "http://www.lq.com"

pages <- read_html(file.path(base_url,"en/findandbook/hotel-listings.html")) %>%
  html_nodes("#hotelListing .col-sm-12 a") %>%
  html_attr("href") %>%
  discard(is.na) %>%
  discard(str_detect, "hotel-details\\.null\\.html") %>%
  file.path(base_url, .)
```

Scraping La Quinta locations

Step 2: Save hotels pages locally

```
# Create a directory to store downloaded hotel pages
data_dir <- "data/"
dir.create(data_dir, showWarnings = FALSE)

# Create a progress bar
p <- progress_estimated(length(pages))

# Download each hotel page
walk(pages, function(url){
  download.file(url, destfile = file.path(data_dir,basename(url)), quiet = TRUE)
  p$tick()$print()
})
```

Scraping La Quinta locations

Step 3: Process all hotel info into a data frame

Process all hotel info into a data frame

- Information on each hotel in an individual html file
- Tasks:
  - Fetch attributes on each hotel: name, address, phone, fax, longitude, latitude
  - Clean data, if need be
- This is a repetitive task — needs to be done for each hotel — hence we can write a function to do the tasks and apply that function to each hotel's html file

```
# Create a character vector of names of all files in directory
files <- dir(data_dir, full.names = TRUE)

# Function: get_hotel_details, to be applied to each hotel page
get_hotel_details <- function(file) {

  page <- read_html(file)

  details <- page %>%
    html_nodes(".hotelDetailsBasicInfoTitle p") %>%
    html_text() %>%
    str_replace_all("\\s{2,}", "\n")    # turn 2+ white spaces into a new line

  map_src <- page %>% html_nodes(".minimap") %>% html_attr("src")

  lat_long <- map_src %>%
    str_match("\\|([0-9.-]{3,}),([0-9.-]{3,})&") %>%
    .[,-1] %>%        # omit the first column of the matrix
    as.numeric()      # turn lat/lon to numeric

  data_frame(
    Name = page %>% html_node("h1") %>% html_text(),
    Address = details %>% str_replace("(?s)Phone:.*","") %>% str_trim(),
    Phone = details %>% str_match("Phone: ([0-9-]{6,})") %>% .[,-1],
    Fax = details %>% str_match("Fax: ([0-9-]{6,})") %>% .[,-1],
    Longitude = lat_long[2],
    Latitude = lat_long[1]
  )
}

# Apply the get_hotel_details function to each element of files
lq <- map_df(files, get_hotel_details)
```

Pointers for scraping  locations

bit.ly/ds-case-study

RStudio®

Scraping Denny's locations

- All locations listed at https://locations.dennys.com/
- Check that it's ok to scrape location data:

```
> paths_allowed("https://locations.dennys.com/")
 locations.dennys.com

[1] TRUE
```

- Two types of pages: California -> Adelanto vs. Anaheim
  - One restaurant per location
  - Multiple restaurants per location

Spatial analysis

bit.ly/ds-case-study

RStudio®

Spatial analysis

Identifying nearest neighbours

bit.ly/ds-case-study

RStudio®

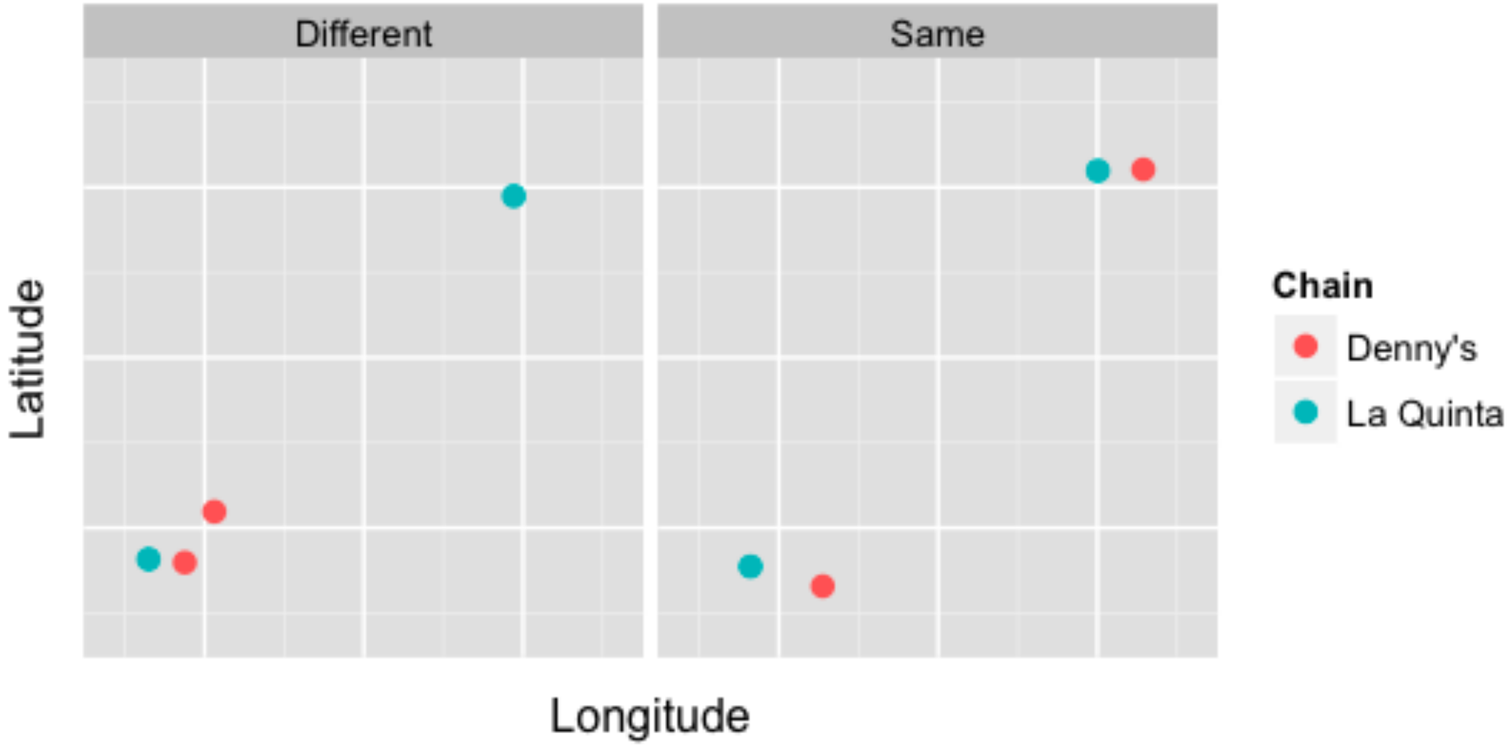
Identifying nearest neighbors

- In order to perform an analysis on the relationship between Denny's and La Quinta location, it is necessary to first identify all possible Denny's / La Quinta pairs and their distances from one another.
- However the analysis will focus only on the pairs that are the nearest neighbors.
- It is important to note that this subset is made up of two sets of pairs that are similar but not identical:

1. The set of pairs created by finding the La Quinta that is closest to any given Denny's location.
2. The set of pairs created by finding the Denny's closes to any given La Quinta location.

Left: Denny's-La Quinta pairs will have short distances (both Denny's are nearest to the same La Quinta) while the La Quinta-Denny's pairs will have one short and one long distance.

Right: Both the Denny's-La Quinta pairs and the La Quinta-Denny's pairs will have the identical distances.



Identifying nearest neighbors

- Construct a pairwise distance matrix between the Denny's and La Quinta locations.
- Nested iterations through each Denny's location and La Quinta location producing 1643 x 909 unique location pairs. Then, for each of these pairs, calculate a distance for and store the results in a matrix, with 1643 rows and 909 columns.
- Euclidean distance is not a good choice since it ignores the fact that latitude/longitude are coordinates for points on the earth's surface, which is a sphere and not a plane.

Finding the distance between two points

- Two options:
  - Adopting a distance function that is appropriate for locations of a sphere
  - Reproject locations into a new coordinate system where Euclidean distance is appropriate.
- For the sake of simplicity, choose former, and use haversine distance formula, which is a simple but reliable way of calculating the great circle distance.

$$d = R \times 2 \arcsin \sqrt{\sin^2\left(\frac{t_2 - t_1}{2}\right) + \cos(g_1)\cos(g_2)\sin^2\left(\frac{g_2 - g_1}{2}\right)}$$

```r
haversine <- function(long1, lat1, long2, lat2, round=3)
{
  # convert to radians
  long1 = long1 * pi / 180
  lat1  = lat1  * pi / 180
  long2 = long2 * pi / 180
  lat2  = lat2  * pi / 180

  R = 6371 # Earth mean radius in km

  a = sin((lat2 - lat1)/2)^2 + cos(lat1) * cos(lat2) * sin((long2 - long1)/2)^2
  d = R * 2 * asin(sqrt(a))

  return( round(d,round) ) # distance in km
}
```

Calculating distances

Two options:

- Nested loops
- Vectorization (*choose this)

Both applying the `haversine` function to pairs of La Quinta - Denny's locations

```
# Vectorized solution
dist <- matrix(NA, nrow(dn), nrow(lq))

for(i in 1:nrow(dn))
{
  dist[i,] = haversine(dn$Longitude[i], dn$Latitude[i],
                       lq$Longitude,    lq$Latitude)
}
```

Identify the nearests

	dist	dn.Address	dn.City	dn.State	lq.Address
1	0.016	160 N 10th Street	Fowler	CA	190 N. 10th Street, Fowler, CA 93625
2	0.016	607 Avenue Q	Lubbock	TX	601 Ave Q, Lubbock, TX 79401-2613
3	0.017	3321 Milan Road	Sandusky	OH	3304 Milan Rd, Sandusky, OH 44870
4	0.02	5910 Veterans	Metairie	LA	5900 Veterans Memorial Blvd, Metairie, LA 70003-1738
5	0.021	2801 No Black Canyon	Phoenix	AZ	2725 North Black Canyon Hwy, Phoenix, AZ 85009-1897
6	0.024	3026 Washington Rd	Augusta	GA	3020 Washington Rd, Augusta, GA 30907
7	0.026	1410 Seawall Blvd	Galveston	TX	1402 Seawall Blvd, Galveston, TX 77550-8199

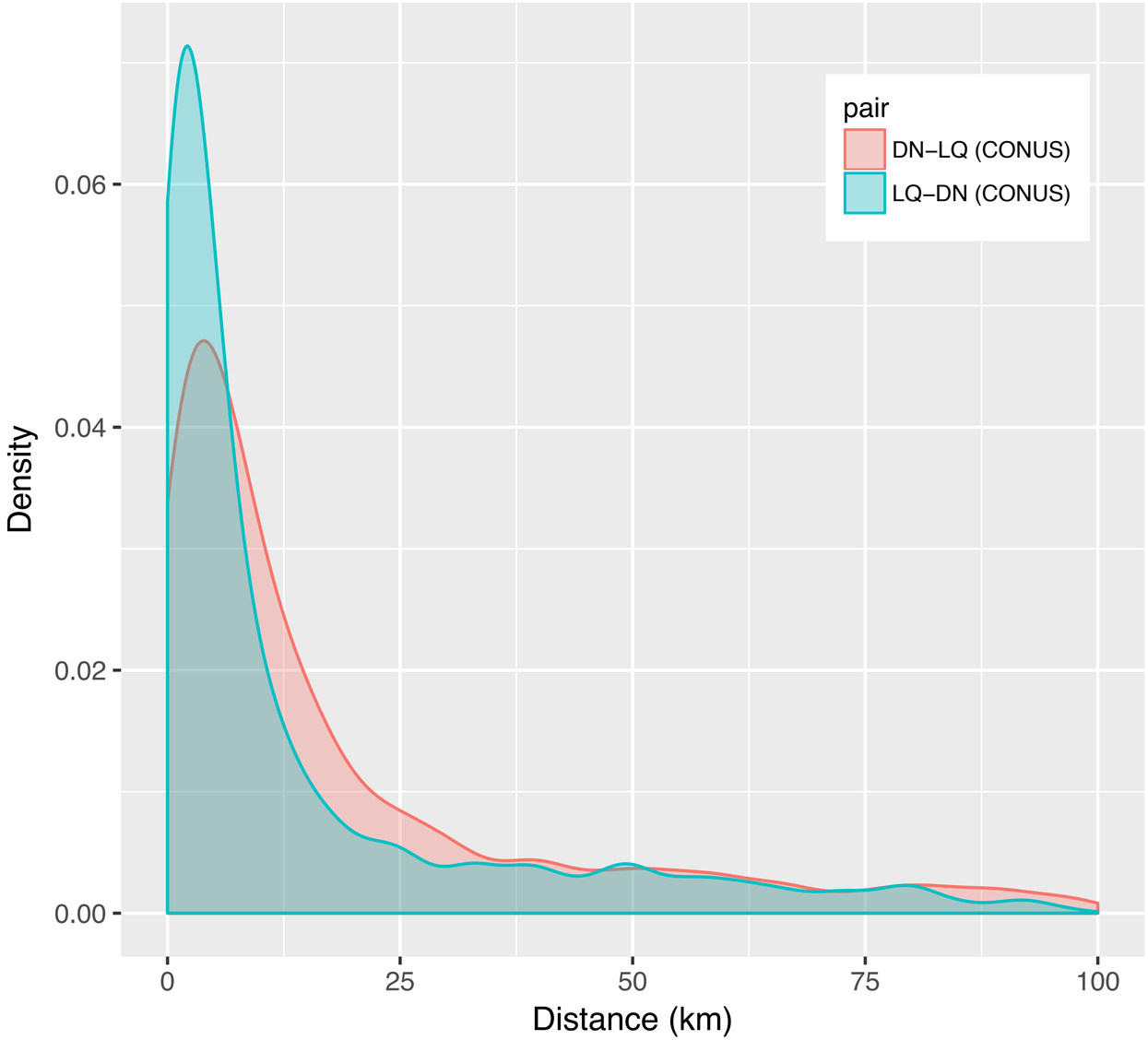

US-84
Lubbock, Texas
Street View - Dec 2012
©2015 Google
©2015 Google
Denny's

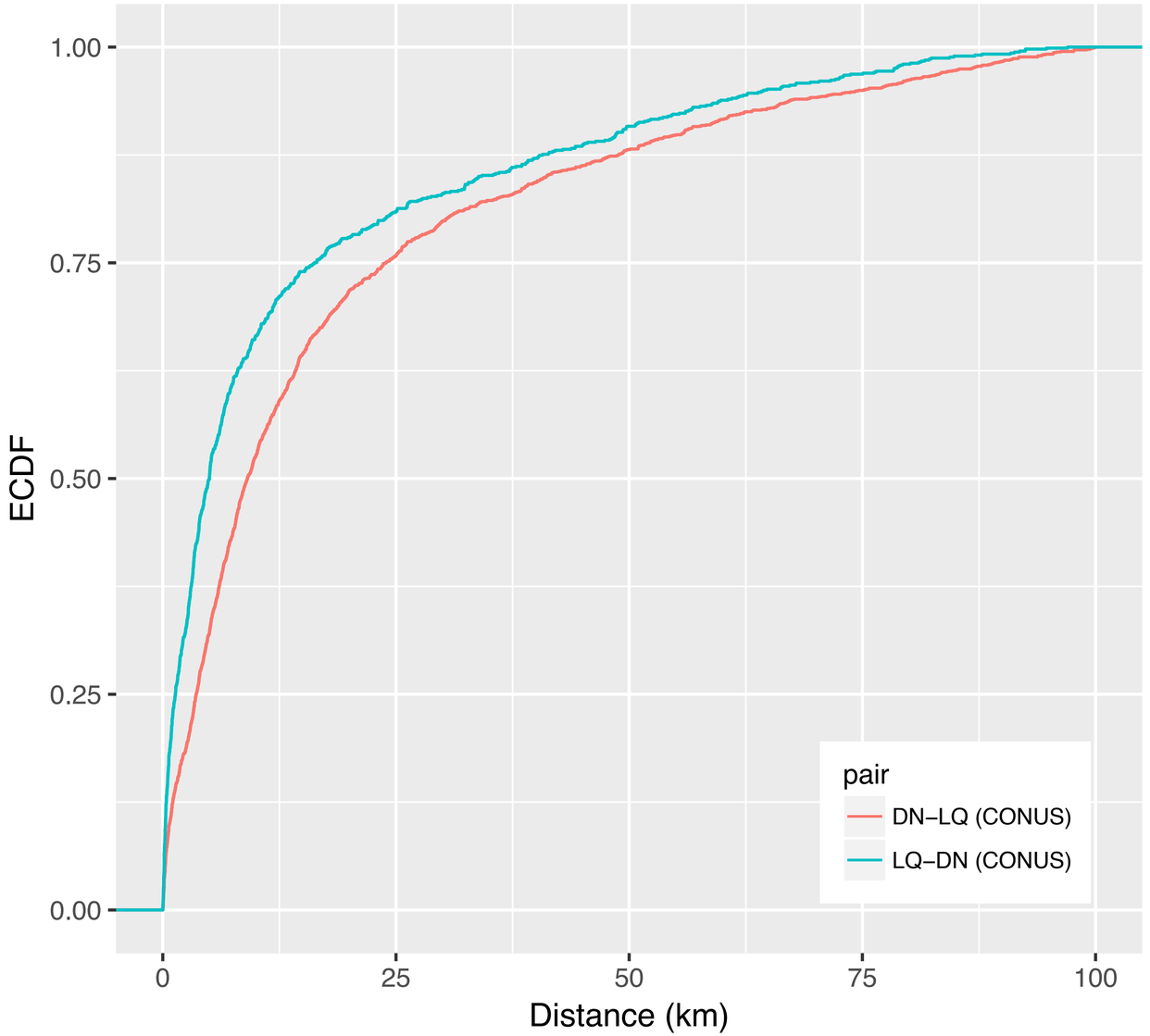
Spatial analysis

Exploring the distance distributions

bit.ly/ds-case-study

RStudio®


pair
DN–LQ (CONUS)
LQ–DN (CONUS)
Density
0.06
0.04
0.02
0.00
0
25
50
75
100
Distance (km)


ECDF
1.00
0.75
0.50
0.25
0.00
0
25
50
75
100
Distance (km)
pair
DN–LQ (CONUS)
LQ–DN (CONUS)

Resources

bit.ly/ds-case-study

RStudio®

CHANCE article http://chance.amstat.org/2016/04/la-quinta/ .

Read at https://github.com/rundel/Dennys_LaQuinta_Chance.