Travel Optimization R script

by Aaron White

Context

This script came out of a request for to analyze the possibility of having employees based in different locations across North Carolina attend meetings in the same place at the same time. Instead of just writing the code once, I decided to make it reproducible. This is the result.

The code does the following:

- Downloads necessary information from Google APIs for declared origins and destinations
- Maps the data
- Generates a plan with optmized routes

The resulting code could easily be adapted to other similar situations or applied at a larger scale. (Google APIs limit the number of elements per call on the API.)

Getting the data

Let's say I have I'm in Raleigh, NC, and I have two colleagues that live in Charlotte, NC and Wilmington, NC respectively. We need to visit Asheville, NC, Johnson City, TN, Eden, NC, Ahoskie, NC Clemson, SC, Fayetteville, NC, and Greensboro, NC.

```
origins <- c("raleigh, nc", "charlotte, nc", "wilmington, nc")
destinations <- c("asheville, nc", "johnson city, tn", "eden, nc",
                  "ahoskie, nc", "clemson, sc", "fayetteville, nc", "greensboro, nc")
```

Distance Matrix

The first important set of information is the distance matrix.

Download the matrix information

get_api_data(origins, destinations)

get_api_data accesses the Google Maps Distance Matrix API and downloads the JSON output. It works by reading the input origins and destinations and then formats them to a URL for Google. ***get_api_data*** then downloads the information and uses the library RJSONIO to create a list of lists. -This function is limited to *10 total locations* because the API restricts to 100 elements and this generates up to a 10x10 matrix of values between every single point.

```
rawData <- get_api_data(origins, destinations)
```

This information could also be modified to include more options within the maps API.

Geolocation information

Since I also want to map the information to help visualize the potential routes, I needed to get the latitude and longitude values for each location as well. For this, I'll the Google Maps Geo Coding API.

Download geo data

parse_geoLocation(locations)

parse_geoLocation performs a similar function to the combined ***get_api_data*** and also parses the data. The function polls the API for each location and stores the information in a new matrix.

```
geoData <- parse_geoLocation(c(origins, destinations))
print(geoData)
```

```
##                            Lat   Long
## Raleigh, NC, USA         35.78 -78.64
## Charlotte, NC, USA       35.23 -80.84
## Wilmington, NC, USA      34.23 -77.94
## Asheville, NC, USA       35.60 -82.55
## Johnson City, TN, USA    36.31 -82.35
## Eden, NC 27288, USA      36.49 -79.77
## Ahoskie, NC 27910, USA   36.29 -76.98
## Clemson, SC, USA         34.68 -82.84
## Fayetteville, NC, USA    35.05 -78.88
## Greensboro, NC, USA      36.07 -79.79
```

Generating the routes

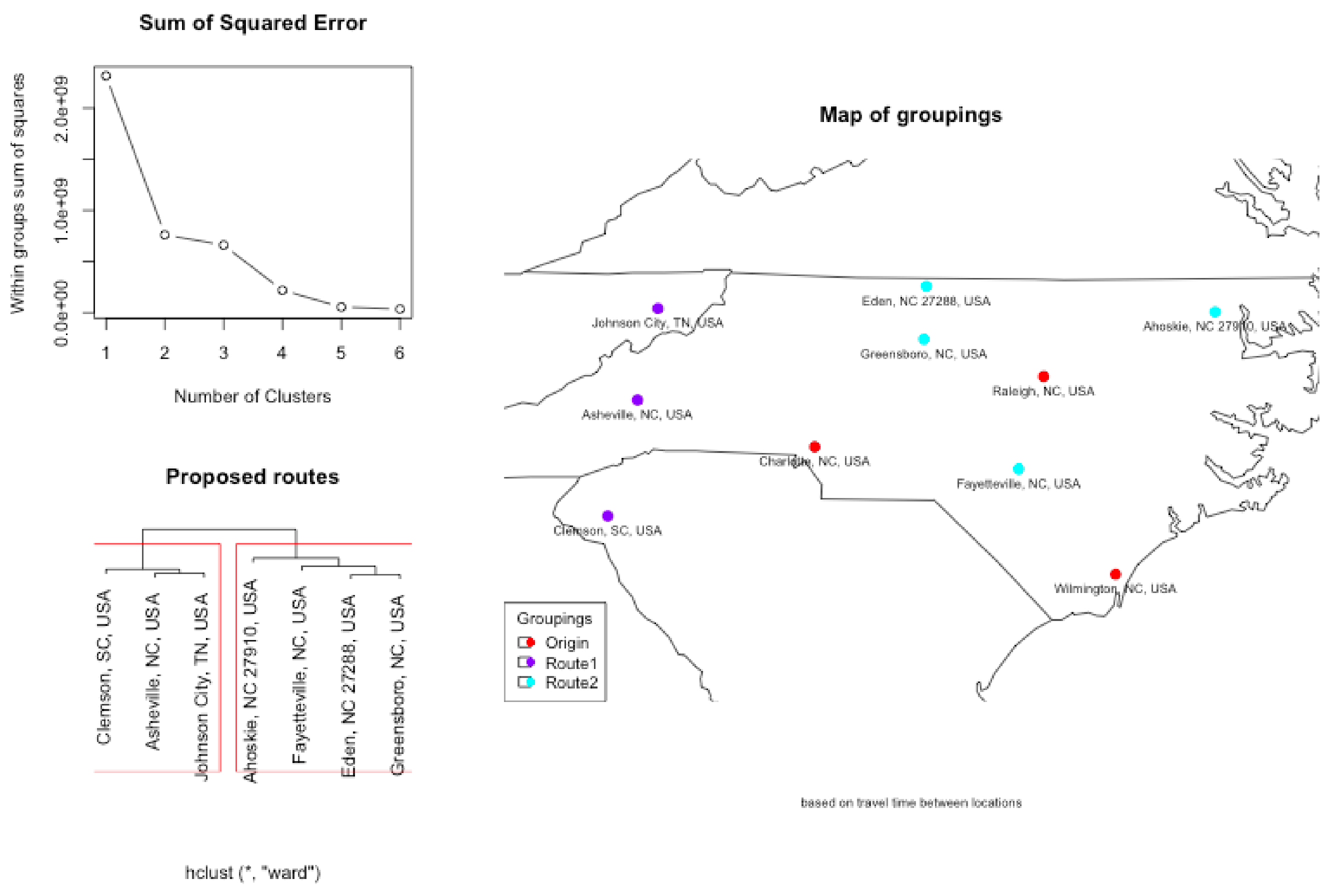
OK. I've got all the information I need. Let's do something with it.

genRoutes(origins, data, geoData, num_routes, datatype, max_driving)

genRoutes takes performs a cluster analysis to determine the optimal grouping of the destinations based on the raw data I just downloaded.

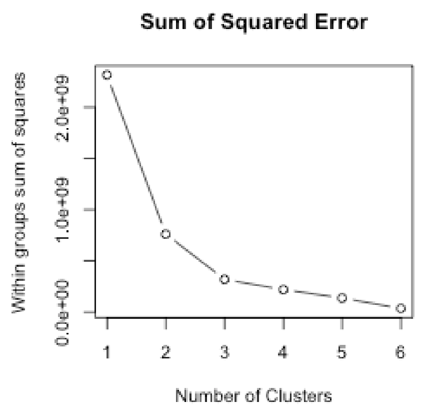
```
plan <- genRoutes(origins, rawData, geoData)
```

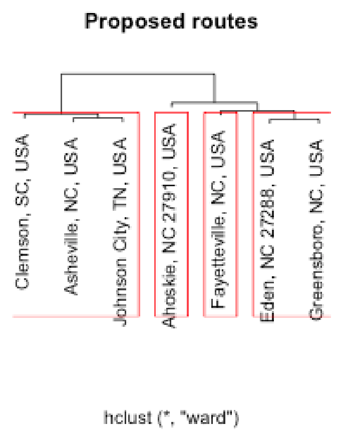
The function produces a plot with a map of the routes highlighed, a “SCREE” plot using kmeans clustering, and a visualization of the hierarchical clustering of the locations. The 'elbow' of the graph shows the optimal number of clusters.

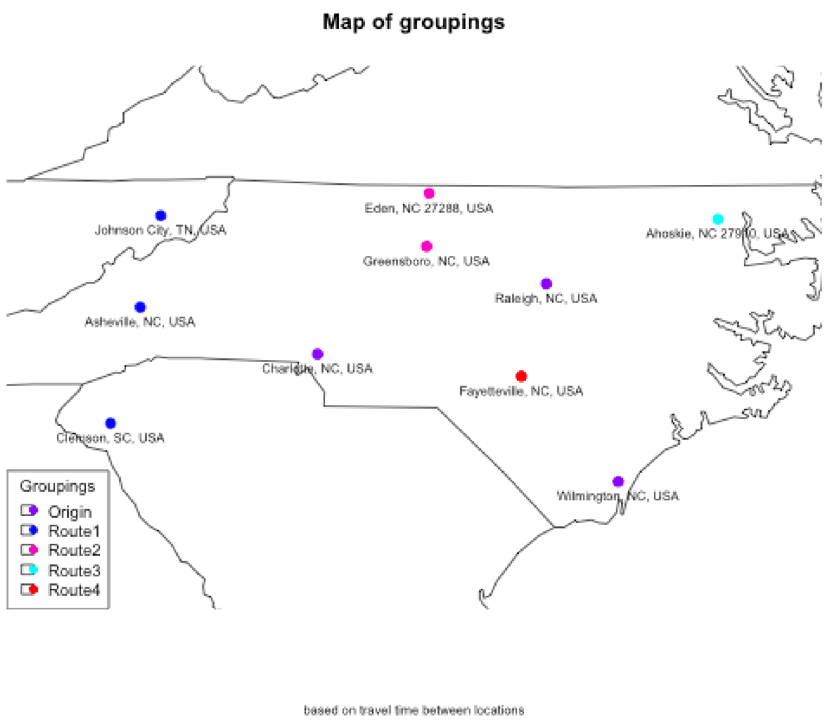


genRoutes automatically will estimate the optimal number of clusters, but the total clusters can be overridden by adding the num_routes argument.

```
plan <- genRoutes(origins, rawData, geoData, num_routes = 4)
```







It also returns a list of matrices with travel times to each destination and back to the origin for each origin. The function determines which origin has to travel the farthest distance and uses that as the first destination. It then calculates total travel time and an estimated days necessary to complete the route. Days are estimated based on the max_driving argument and rounded up. If nothing is passed, the default is four hours per day.

```
## $datatype
## [1] "duration"
##
## $max_driving
## [1] "4"
##
## $Route1
##                       Johnson City, TN, USA Asheville, NC, USA
## Raleigh, NC, USA                        4.2                  1
## Charlotte, NC, USA                      2.9                  1
## Wilmington, NC, USA                     6.1                  1
##                       Clemson, SC, USA Return Home Total Est.
Days
## Raleigh, NC, USA                   1.6         4.6  11.4
3
## Charlotte, NC, USA                 1.6         2.2   7.8
2
## Wilmington, NC, USA                1.6         5.0  13.7
4
##
## $Route2
##                       Eden, NC 27288, USA Greensboro, NC, USA
Return Home
## Raleigh, NC, USA                      1.7                 0.7
1.3
## Charlotte, NC, USA                    2.0                 0.7
1.4
## Wilmington, NC, USA                   3.5                 0.7
3.1
##                       Total Est. Days
## Raleigh, NC, USA        3.7         1
## Charlotte, NC, USA      4.2         2
## Wilmington, NC, USA     7.3         2
##
## $Route3
##                       Ahoskie, NC 27910, USA Return Home Total
Est. Days
## Raleigh, NC, USA                         2.0         2.0   3.9
1
## Charlotte, NC, USA                       4.4         4.4   8.8
3
## Wilmington, NC, USA                      3.1         3.1   6.2
2
##
## $Route4
##                       Fayetteville, NC, USA Return Home Total
Est. Days
## Raleigh, NC, USA                        1.0         1.0   2.1
1
## Charlotte, NC, USA                      2.6         2.6   5.3
2
## Wilmington, NC, USA                     1.6         1.6   3.1
1
```

The output can also be called based on route number to show a specific route plan:

```
plan$Route2
```

```
##                       Eden, NC 27288, USA Greensboro, NC, USA
Return Home
## Raleigh, NC, USA                      1.7                 0.7
1.3
## Charlotte, NC, USA                    2.0                 0.7
1.4
## Wilmington, NC, USA                   3.5                 0.7
3.1
##                       Total Est. Days
## Raleigh, NC, USA        3.7         1
## Charlotte, NC, USA      4.2         2
## Wilmington, NC, USA     7.3         2
```

Or a specific line within the route:

```
plan$Route1["Raleigh, NC, USA",]
```

```
## Johnson City, TN, USA    Asheville, NC, USA      Clemson,
SC, USA
##                   4.2                   1.0
1.6
##           Return Home                 Total             Est.
Days
##                   4.6                  11.4
3.0
```

genRoutes also supports optional arguments of datatype and max_driving.

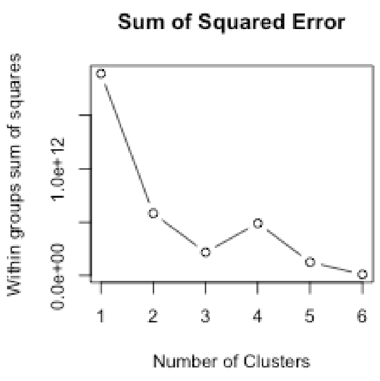
- *datatype* determines how the raw values are converted. If datatype = “distance”, values will be converted to miles. By default *datatype* is set to “duration”.
- *max_driving* determines how many estimated days the plan will take. ***genRoutes*** calculates an estimated number of days based on the total travel time. Passing the *max_driving* argument will override the default value of 4 if *datatype == “duration”* or 240 if *datatype == “distance”*.

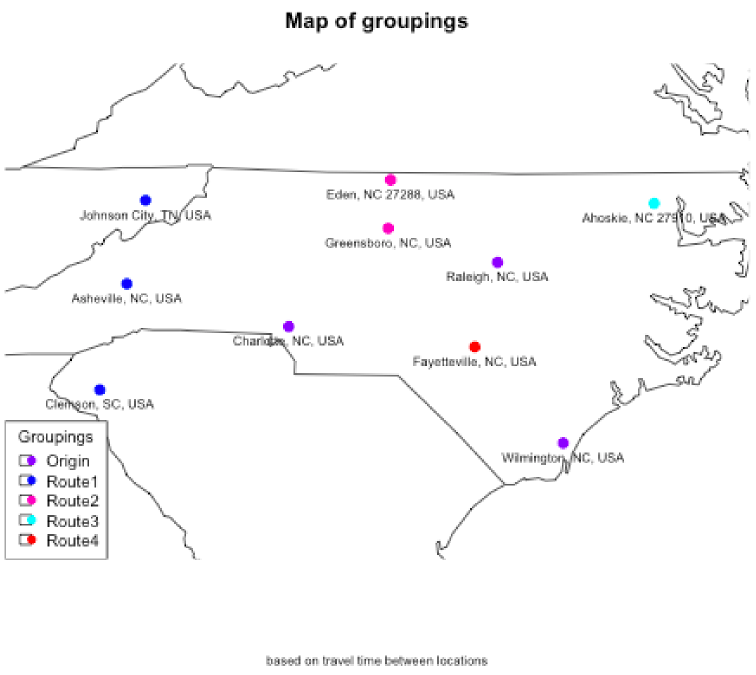
Maybe I want to know the grouping based on distance, and I'm willing to drive 500 miles a day.

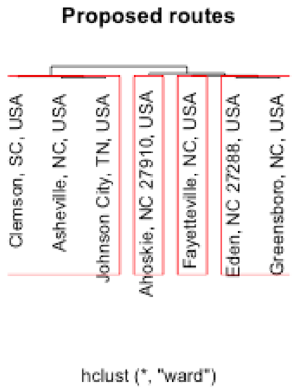
```
plan <- genRoutes(origins, rawData, geoData, 4,
datatype="distance", max_driving=500)
```

```
## [1] "Estimated number of routes: 2"
```

```
## [1] "Final number of routes: 4"
## [1] "===== Route1 ====="
## [1] "Johnson City, TN, USA is the farthest from Wilmington,
NC, USA"
## [1] "Optimizing route starting with Wilmington, NC, USA"
##                      Johnson City, TN, USA Asheville, NC, USA
## Raleigh, NC, USA                     243.0               60.4
## Charlotte, NC, USA                   147.4               60.4
## Wilmington, NC, USA                  375.0               60.4
##                      Clemson, SC, USA Return Home Total Est.
Days
## Raleigh, NC, USA                 89.1       295.8 688.3
2
## Charlotte, NC, USA               89.1       132.8 429.6
1
## Wilmington, NC, USA              89.1       333.4 857.8
2
## [1] "===== Route2 ====="
## [1] "Eden, NC 27288, USA is the farthest from Wilmington,
NC, USA"
## [1] "Optimizing route starting with Wilmington, NC, USA"
##                      Eden, NC 27288, USA Greensboro, NC, USA
Return Home
## Raleigh, NC, USA                    97.2                36.7
76.7
## Charlotte, NC, USA                 126.3                36.7
90.6
## Wilmington, NC, USA                229.1                36.7
208.7
##                      Total Est. Days
## Raleigh, NC, USA    210.6         1
## Charlotte, NC, USA  253.5         1
## Wilmington, NC, USA 474.5         1
## [1] "===== Route3 ====="
##                      Ahoskie, NC 27910, USA Return Home Total
Est. Days
## Raleigh, NC, USA                      122.0       122.0 243.9
1
## Charlotte, NC, USA                    284.9       284.9 569.9
2
## Wilmington, NC, USA                   174.3       174.3 348.5
1
## [1] "===== Route4 ====="
##                      Fayetteville, NC, USA Return Home Total
Est. Days
## Raleigh, NC, USA                      65.2        65.2 130.4
1
## Charlotte, NC, USA                   132.5       132.5 265.1
1
## Wilmington, NC, USA                   89.0        89.0 178.0
1
## [1] "datatype = distance"
## [1] "max_driving = 500"
```







The output also stores the datatype and max_duration values that either were input by the user, or the automatic defaults. Maybe I forgot what type of data I was using as I generated the results…

```
plan$datatype
```

```
## [1] "distance"
```

```
plan$max_driving
```

```
## [1] "500"
```

Phew! Good thing I used `plan$duration`. That's what I wanted! But I want to change my *max_driving* value to something different from `plan$max_driving`.

```
plan <- genRoutes(origins, rawData, geoData, 4,
datatype="distance", max_driving=200)
```

```
## [1] "Estimated number of routes: 2"
```

```
## [1] "Final number of routes: 4"
## [1] "===== Route1 ====="
## [1] "Johnson City, TN, USA is the farthest from Wilmington,
NC, USA"
## [1] "Optimizing route starting with Wilmington, NC, USA"
##                      Johnson City, TN, USA Asheville, NC, USA
## Raleigh, NC, USA                     243.0               60.4
## Charlotte, NC, USA                   147.4               60.4
## Wilmington, NC, USA                  375.0               60.4
##                      Clemson, SC, USA Return Home Total Est.
Days
## Raleigh, NC, USA                 89.1       295.8 688.3
4
## Charlotte, NC, USA               89.1       132.8 429.6
3
## Wilmington, NC, USA              89.1       333.4 857.8
5
## [1] "===== Route2 ====="
## [1] "Eden, NC 27288, USA is the farthest from Wilmington,
NC, USA"
## [1] "Optimizing route starting with Wilmington, NC, USA"
##                      Eden, NC 27288, USA Greensboro, NC, USA
Return Home
## Raleigh, NC, USA                    97.2                36.7
76.7
## Charlotte, NC, USA                 126.3                36.7
90.6
## Wilmington, NC, USA                229.1                36.7
208.7
##                      Total Est. Days
## Raleigh, NC, USA    210.6         2
## Charlotte, NC, USA  253.5         2
## Wilmington, NC, USA 474.5         3
## [1] "===== Route3 ====="
##                      Ahoskie, NC 27910, USA Return Home Total
Est. Days
## Raleigh, NC, USA                      122.0       122.0 243.9
2
## Charlotte, NC, USA                    284.9       284.9 569.9
3
## Wilmington, NC, USA                   174.3       174.3 348.5
2
## [1] "===== Route4 ====="
##                      Fayetteville, NC, USA Return Home Total
Est. Days
## Raleigh, NC, USA                      65.2        65.2 130.4
1
## Charlotte, NC, USA                   132.5       132.5 265.1
2
## Wilmington, NC, USA                   89.0        89.0 178.0
1
## [1] "datatype = distance"
## [1] "max_driving = 200"
```

Now I have four travel plans to help divide up the amount of time my colleagues and I are traveling. I probably could have done this visually, but this application is now scaleable.

One last thing. I also want the whole travel matrix in case anything changes, and I need to make another stop on the fly.

parse_data(rawData, datatype, pretty)

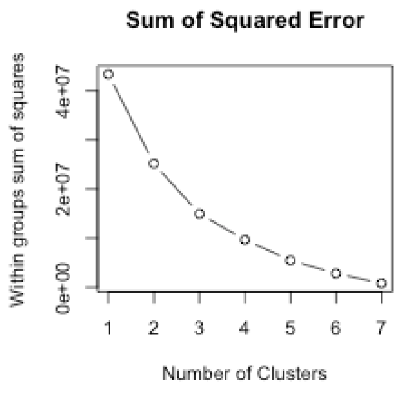
I'll use the *parse_data* function to take care of that. The values that Google provides are in seconds and meters, but the output also gives me the data in a "human readable" format. I'll pull that information.

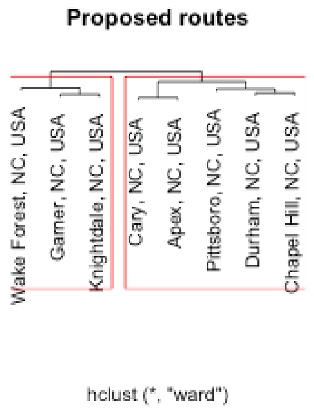
```
travelMatrix <- parse_data(rawData, "distance", pretty=TRUE)
print(travelMatrix)
```

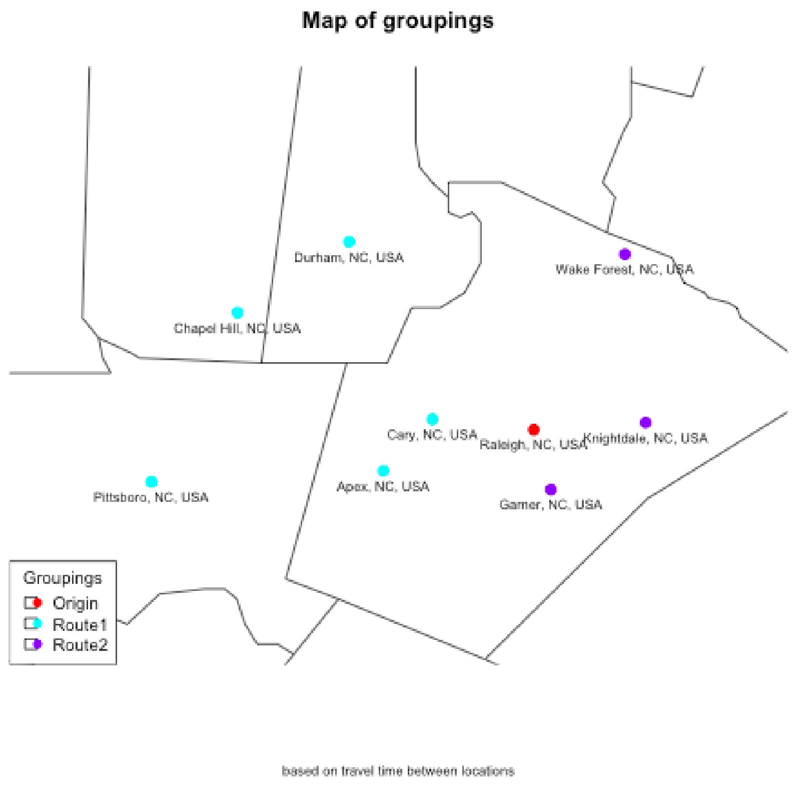
```
##                          Raleigh, NC, USA Charlotte, NC, USA
## Raleigh, NC, USA         "1 ft"           "166 mi"
## Charlotte, NC, USA       "164 mi"         "1 ft"
## Wilmington, NC, USA      "131 mi"         "198 mi"
## Asheville, NC, USA       "247 mi"         "125 mi"
## Johnson City, TN, USA    "243 mi"         "148 mi"
## Eden, NC 27288, USA      "98.8 mi"        "127 mi"
## Ahoskie, NC 27910, USA   "120 mi"         "286 mi"
## Clemson, SC, USA         "294 mi"         "134 mi"
## Fayetteville, NC, USA    "65.4 mi"        "133 mi"
## Greensboro, NC, USA      "76.6 mi"        "90.5 mi"
##                          Wilmington, NC, USA Asheville, NC,
USA
## Raleigh, NC, USA         "133 mi"            "247 mi"
## Charlotte, NC, USA       "198 mi"            "124 mi"
## Wilmington, NC, USA      "1 ft"              "359 mi"
## Asheville, NC, USA       "359 mi"            "1 ft"
## Johnson City, TN, USA    "367 mi"            "60.4 mi"
## Eden, NC 27288, USA      "232 mi"            "188 mi"
## Ahoskie, NC 27910, USA   "176 mi"            "367 mi"
## Clemson, SC, USA         "332 mi"            "88.9 mi"
## Fayetteville, NC, USA    "88.9 mi"           "262 mi"
## Greensboro, NC, USA      "210 mi"            "173 mi"
##                          Johnson City, TN, USA Eden, NC 27288,
USA
## Raleigh, NC, USA         "243 mi"              "97.2 mi"
## Charlotte, NC, USA       "147 mi"              "126 mi"
## Wilmington, NC, USA      "375 mi"              "229 mi"
## Asheville, NC, USA       "60.7 mi"             "188 mi"
## Johnson City, TN, USA    "1 ft"                "183 mi"
## Eden, NC 27288, USA      "183 mi"              "1 ft"
## Ahoskie, NC 27910, USA   "363 mi"              "185 mi"
## Clemson, SC, USA         "148 mi"              "257 mi"
## Fayetteville, NC, USA    "258 mi"              "127 mi"
## Greensboro, NC, USA      "168 mi"              "36.4 mi"
##                          Ahoskie, NC 27910, USA Clemson, SC,
USA
## Raleigh, NC, USA         "122 mi"               "296 mi"
## Charlotte, NC, USA       "285 mi"               "133 mi"
## Wilmington, NC, USA      "174 mi"               "333 mi"
## Asheville, NC, USA       "368 mi"               "89.1 mi"
## Johnson City, TN, USA    "364 mi"               "147 mi"
## Eden, NC 27288, USA    "184 mi"                 "257 mi"
## Ahoskie, NC 27910, USA "1 ft"                   "416 mi"
## Clemson, SC, USA       "415 mi"                 "1 ft"
## Fayetteville, NC, USA  "162 mi"                 "291 mi"
## Greensboro, NC, USA    "197 mi"                 "221 mi"
##                        Fayetteville, NC, USA Greensboro, NC,
USA
## Raleigh, NC, USA       "65.2 mi"               "76.7 mi"
## Charlotte, NC, USA     "133 mi"                "90.6 mi"
## Wilmington, NC, USA    "89.0 mi"               "209 mi"
## Asheville, NC, USA     "266 mi"                "173 mi"
## Johnson City, TN, USA  "261 mi"                "169 mi"
## Eden, NC 27288, USA    "131 mi"                "36.7 mi"
## Ahoskie, NC 27910, USA "162 mi"                "197 mi"
## Clemson, SC, USA       "290 mi"                "221 mi"
## Fayetteville, NC, USA  "1 ft"                  "92.4 mi"
## Greensboro, NC, USA    "95.8 mi"               "1 ft"
```

Another example

genRoutes can map most situations. For this, I will just use one origin starting in Raleigh.



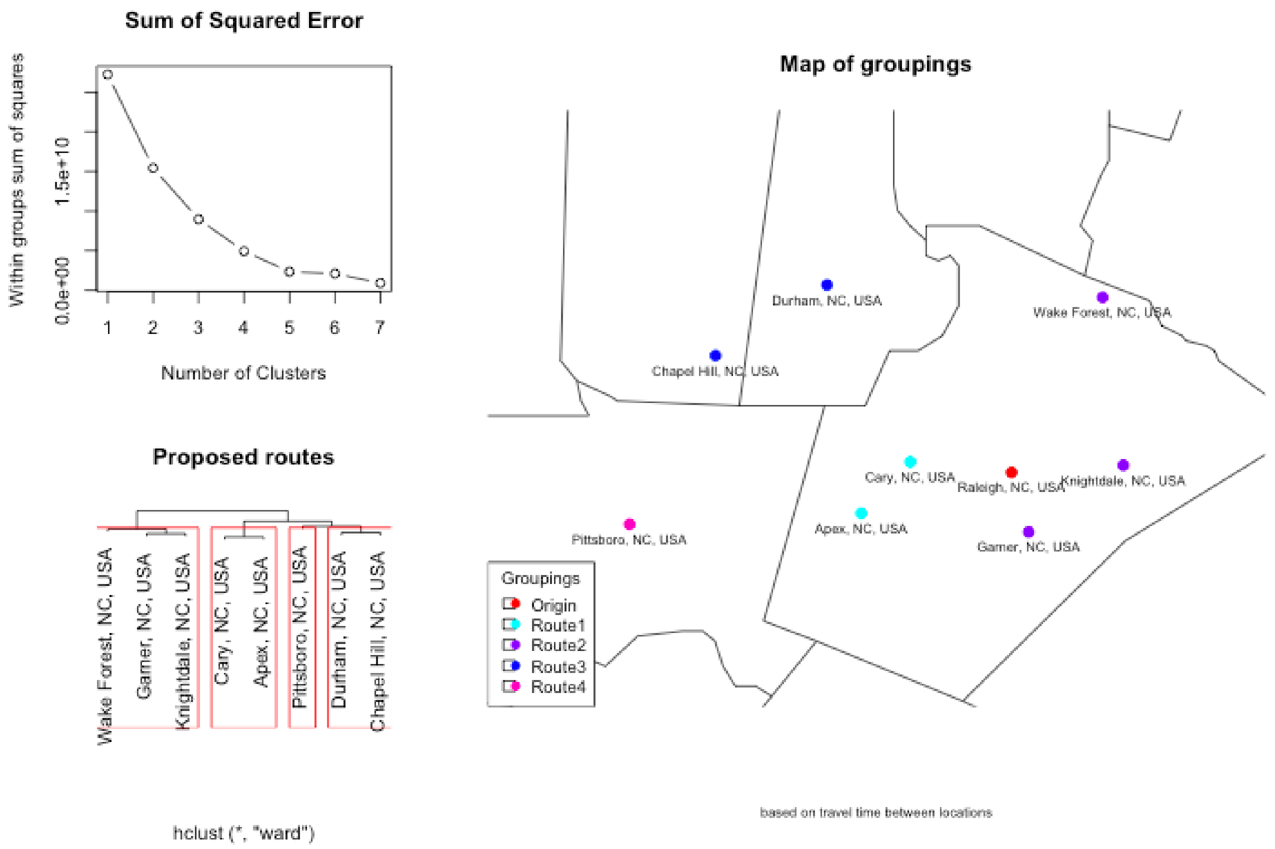




```
originList <- "Raleigh, NC"
destinationList <- c("Cary, NC","Wake forest, nc","durham, nc",
"apex, nc", "garner, nc", "knightdale, nc", "pittsboro, nc",
"chapel hill, nc")
rawData1 <- get_api_data(originList, destinationList)
geoData1 <- parse_geoLocation(originList, destinationList)
plan <- genRoutes(originList, rawData1, geoData1)
```

Or if I look at distance and only want to drive 30 miles a day to keep the time I spend in the car to a minimum?

```
plan <- genRoutes(originList, rawData1, geoData1,
datatype="distance", max_driving=30)
```



What does my plan for route 2 look like and what kind of data do I have again?

```
plan$datatype
```

```
## [1] "distance"
```

```
plan$max_driving
```

```
## [1] "30"
```

```
plan$Route2
```

```
##                     Wake Forest, NC, USA Knightdale, NC, USA
Garner, NC, USA
## Raleigh, NC, USA                     18.4                20.1
12.9
##                     Return Home Total Est. Days
## Raleigh, NC, USA            5.9  57.3         2
```

Great! I'm ready to print the routes out and start traveling!

```
print(plan)
```

```
## $datatype
## [1] "distance"
##
## $max_driving
## [1] "30"
##
## $Route1
##                     Apex, NC, USA Cary, NC, USA Return Home
Total Est. Days
## Raleigh, NC, USA             15.6           6.6        12.1
34.2         2
##
## $Route2
##                     Wake Forest, NC, USA Knightdale, NC, USA
Garner, NC, USA
## Raleigh, NC, USA                     18.4                20.1
12.9
##                     Return Home Total Est. Days
## Raleigh, NC, USA            5.9  57.3         2
##
## $Route3
##                     Chapel Hill, NC, USA Durham, NC, USA Return
Home Total
## Raleigh, NC, USA                     28.4            11.1
24.5    64
##                     Est. Days
## Raleigh, NC, USA            3
##
## $Route4
##       Pittsboro, NC, USA Return Home Total Est. Days
## [1,]                  34          34    68         3
```

And my travel matrix:

```
travelMatrix <- parse_data(rawData1, "distance", pretty=TRUE)
print(travelMatrix)
```

```
##                         Raleigh, NC, USA Cary, NC, USA Wake
Forest, NC, USA
## Raleigh, NC, USA     "1 ft"           "12.1 mi"     "18.4
mi"
## Cary, NC, USA        "12.1 mi"        "1 ft"        "28.2
mi"
## Wake Forest, NC, USA "18.5 mi"        "27.0 mi"     "1 ft"
## Durham, NC, USA      "25.0 mi"        "19.0 mi"     "22.9
mi"
## Apex, NC, USA        "15.3 mi"        "6.6 mi"      "32.8
mi"
## Garner, NC, USA      "5.5 mi"         "14.3 mi"     "25.3
mi"
## Knightdale, NC, USA  "10.6 mi"        "23.8 mi"     "19.3
mi"
## Pittsboro, NC, USA   "33.7 mi"        "24.1 mi"     "50.4
mi"
## Chapel Hill, NC, USA "28.1 mi"        "22.2 mi"     "37.2
mi"
##                      Durham, NC, USA Apex, NC, USA Garner,
NC, USA
## Raleigh, NC, USA     "24.5 mi"       "15.6 mi"     "5.9 mi"
## Cary, NC, USA        "18.9 mi"       "6.7 mi"      "14.1 mi"
## Wake Forest, NC, USA "22.9 mi"       "32.7 mi"     "25.9 mi"
## Durham, NC, USA      "1 ft"          "23.6 mi"     "31.4 mi"
## Apex, NC, USA        "23.3 mi"       "1 ft"        "17.3 mi"
## Garner, NC, USA      "31.0 mi"       "17.8 mi"     "1 ft"
## Knightdale, NC, USA  "33.2 mi"       "27.3 mi"     "12.9 mi"
## Pittsboro, NC, USA   "36.6 mi"       "19.9 mi"     "35.8 mi"
## Chapel Hill, NC, USA "11.1 mi"       "23.5 mi"     "34.5 mi"
##                      Knightdale, NC, USA Pittsboro, NC, USA
## Raleigh, NC, USA     "10.6 mi"           "34.0 mi"
## Cary, NC, USA        "23.6 mi"           "24.2 mi"
## Wake Forest, NC, USA "20.1 mi"           "49.9 mi"
## Durham, NC, USA      "32.4 mi"           "36.2 mi"
## Apex, NC, USA        "26.8 mi"           "20.1 mi"
## Garner, NC, USA      "12.9 mi"           "36.3 mi"
## Knightdale, NC, USA  "1 ft"              "45.8 mi"
## Pittsboro, NC, USA   "45.3 mi"           "1 ft"
## Chapel Hill, NC, USA "40.8 mi"           "16.9 mi"
##                      Chapel Hill, NC, USA
## Raleigh, NC, USA     "28.4 mi"
## Cary, NC, USA        "22.8 mi"
## Wake Forest, NC, USA "37.3 mi"
## Durham, NC, USA      "11.2 mi"
## Apex, NC, USA        "23.5 mi"
## Garner, NC, USA      "34.9 mi"
## Knightdale, NC, USA  "40.6 mi"
## Pittsboro, NC, USA   "16.9 mi"
## Chapel Hill, NC, USA "1 ft"
```