R Notebook

Code

Calculating distances - and the nearest point

We have a dataset of over 7,000 locations. For each of those we want to find the nearest *other* point in the dataset.

This is not simple. Each location has a latitude and longitude and we are dealing with distances across a sphere.

The solution is the Haversine equation (). In R this can be calculated using the `haversine` function (https://www.rdocumentation.org/packages/pracma/versions/1.9.9/topics/haversine) which is part of the `pracma` package (an alternative would be the `distHaversine` function (https://www.rdocumentation.org/packages/geosphere/versions/1.5-5/topics/distHaversine) from the geosphere package (https://www.rdocumentation.org/packages/geosphere/versions/1.5-5).

Hide

```
pracma::haversine(c(0,0),c(90,90))
```

```
[1] 10007.54
```

Looping through the coordinates

We need to find out, given one set of lat-long coordinates, what the closest pair of coordinates is from a list of 7000 - in other words the smallest distance.

First, a test where we calculate the distance between row 1's lat-long and row 2's lat-long:

Hide

```
#I've added new lines to make this easier to break down
pracma::haversine(
  c(locations$Latitude[[1]],locations$Longitude[[1]]),
  c(locations$Latitude[[1]],locations$Longitude[[2]])
  )
```

```
[1] 0.5233688
```

We need to replace those numbers with a variable.

In this code we loop through all the numbers from 2 to 7052 (the length of the data), calculate a distance, between our first row lat-long and the lat-long on that row, and then check if it's less than our current 'minimum distance' (set arbitrarily at 100 before the loop begins):

Hide

```
#Set an arbitrary value to compare against first
closestdistance <- 100
#Loop through all the numbers from 2 to the length of the Latitude column (7052)
for(i in seq(2,length(locations$Latitude))){
  #Store the distance between the first lat-long and the lat-long in that row index
  havdist <- pracma::haversine(
    c(locations$Latitude[[1]],locations$Longitude[[1]]),
    c(locations$Latitude[[i]],locations$Longitude[[i]])
  )
  #If that distance is less than our current lowest value
  if(havdist < closestdistance){
    #replace the lowest value with that one
    closestdistance <- havdist
    #And store the index as well, which we'll need to retrive the location
    closestindex <- i
  }
}
```

We need to do this for each, rather than just the first one. But we want to make sure that it avoids comparing any row with itself (which would result in a distance of 0, always the closest!).

We can simply say that the match cannot be 0:

Hide

```
#Set an arbitrary value to compare against first
closestdistance <- 1000
#Loop through all the numbers from 1 to the length of the Latitude column (7052)
for(i in seq(1,length(locations$Latitude))){
  #Store the distance between the first lat-long and the lat-long in that row index
  havdist <- pracma::haversine(
    c(locations$Latitude[[1]],locations$Longitude[[1]]),
    c(locations$Latitude[[i]],locations$Longitude[[i]])
  )
  #If that distance is less than our current lowest value - but NOT 0
  if(havdist < closestdistance & havdist != 0){
    #replace the lowest value with that one
    closestdistance <- havdist
    #And store the index as well, which we'll need to retrive the location
    closestindex <- i
  }
}
```

That works. Now to do this for more than just the first row. For that we need to nest this whole loop inside another loop, so that *for* each item in the 7052 coordinates, it compares it *for* each item in the 7052 coordinates.

We also need to find a way to store all 7052 results - the closest matches. For that we create some empty vectors and then keep adding to those as we go.

It might also help if we separate the code for finding the closest distance into its own function.

Hide

```
getnearest <- function(coordindex){
    for(secondcoords in seq(1,length(locations$Latitude))){
    #Store the distance between the first lat-long and the lat-long in that row index
    havdist <- pracma::haversine(
      c(locations$Latitude[[coordindex]],locations$Longitude[[coordindex]]),
      c(locations$Latitude[[secondcoords]],locations$Longitude[[secondcoords]])
    )
    #If that distance is less than our current lowest value - but NOT 0
    if(havdist < closestdistance & havdist != 0){
      #replace the lowest value with that one
      closestdistance <- havdist
      #And store the index as well, which we'll need to retrive the location
      closestindex <- secondcoords
    }
  }
  #after the loop has finished, return the distance and index that was left as the sm
allest
  return(c(closestdistance,closestindex))
}
```

Now we use that function as we loop through some of the coordinates looking for matches. 7052 is too many to do all at once - multiplied by 7052 comparisons each it means over 49million comparisons!

So we try it on the first 10 coordinates first:

Hide

```
#Set an arbitrary value to compare against first
closestdistance <- 100000
#Create an empty vector to hold our 7052 distances
nearestdistances <- c()
#Create an empty vector to hold our 7052 indexes
closestplaces <- c()
#test it works
distandplace <- getnearest(1)

#Loop through all the numbers from 1 to 10
for(firstcoords in seq(1, 10)){
  #Call the function we created earlier -
  #this will return a vector with the nearest distance and place for a given point
  #the vector is stored as distandplace
  distandplace <- getnearest(firstcoords)
  #We add the first item (distance) in that to our ongoing vector
  nearestdistances <- c(nearestdistances, distandplace[1])
  #And the second (place index) to the other vector
  closestplaces <- c(closestplaces, distandplace[2])
  }
```

This works - we can see that location 1 and 2 are the nearest to each other, but 3 is nearest to location 20. Locations 6 and 7 are also nearest to each other.

The shortest and longest distances between points in our sample are…

Hide

```
min(nearestdistances)
```

```
[1] 0.1926167
```

Hide

```
max(nearestdistances)
```

```
[1] 1.916535
```

Let’s see if we can do another 100…

Hide

```
#Loop through all the numbers from 11 to 100
for(firstcoords in seq(11, 100)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 0.01430213
```

Hide

```
max(nearestdistances)
```

```
[1] 16.22489
```

And another 900…

Hide

```
#Loop through all the numbers from 101 to 1000
for(firstcoords in seq(101, 1000)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 5.865001e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 43.50506
```

Keep going…

Hide

```
#Loop through all the numbers from 101 to 1000
for(firstcoords in seq(1001, 2000)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 5.865001e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 43.50506
```

Hide

```
#Loop through all the numbers from 101 to 1000
for(firstcoords in seq(2001, 3000)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 5.865001e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 43.50506
```

Hide

```
#Loop through all the numbers from 101 to 1000
for(firstcoords in seq(3001, 4000)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 5.865001e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 5473.594
```

Hide

```
#Loop through all the numbers from 101 to 1000
for(firstcoords in seq(4001, 5000)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 5.865001e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 5473.594
```

Hide

```
min(nearestdistances)
```

```
[1] 5.865001e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 5473.594
```

Now for the final bunch:

Hide

```
#Loop through all the numbers from...
for(firstcoords in seq(6001, 7052)){
  #Loop through all the numbers from 1 to the length of the Latitude column (7052)
  #Add to the vectors - this happens outside the loop so it only happens once for eac
h of the 7052 firstcoordinates
  distandplace <- getnearest(firstcoords)
  nearestdistances <- c(nearestdistances, distandplace[1])
  closestplaces <- c(closestplaces, distandplace[2])
}
min(nearestdistances)
```

```
[1] 3.635439e-05
```

Hide

```
max(nearestdistances)
```

```
[1] 5473.594
```

Adding to the original data

We should now have 2 vectors with 7052 values. This is the right length to add it to the original data:

Hide

```
#Add the nearest distances as a column
locations$nearestdistance <- nearestdistances
#These indexes will be useful because the exported file will include an index column
locations$nearestplaceindex <- closestplaces
```

The vector of indexes isn't so useful on its own, but it can be used to create a vector of location names like so:

Hide

```
#Create an empty vector to fill for each column
closestplaceids <- c()
closestplacenames <- c()
closestlats <- c()
closestlngs <- c()
#Loop through the indexes
for (i in closestplaces){
  #Grab the id number at that index
  closestplaceids <- c(closestplaceids,locations$AddressInfo...ID[i])
  #And other details
  closestplacenames <- c(closestplacenames,locations$Location.of.Charging.Point[i])
  closestlats <- c(closestlats,locations$Latitude[i])
  closestlngs <- c(closestlngs,locations$Longitude[i])
}
```

Then we can add those:

Hide

```
#Add each vector as a new column
locations$nearestid <- closestplaceids
locations$nearestname <- closestplacenames
locations$closestlat <- closestlats
locations$closestlng <- closestlngs
```

Export the results!

The ID codes can be used with VLOOKUP in Excel anyway, so let's export it:

Hide

```
write.csv(locations,"locationanalysis.csv")
```