

deriving_landmarks

Don Li

06/06/2020

Purpose

Derive landmark locations and use this to group the data better.

Approaches:

- Use Google Places API or something like that to determine what locations people are going to
- Use statistics

I will use the second one because I don't know how to make an API work.

Data and stuff

K-means

Use k-means clustering (Hartigan-Wong) on the start and end points.

```
load( "../dataset/all_SNG.RData" )

kmeans_map = function( data, centers, nstart, iter.max, trace = F ){
  # Data needs a column of latitudes and longitudes
  kmeans_result = kmeans( data, centers = centers,
    nstart = nstart, iter.max = iter.max, trace = trace )
  kmeans_result
}

predict_kmeans = function(object, newdata){
  centers = object$centers
  n_centers = nrow(centers)
  dist_mat = as.matrix(dist(rbind(centers, newdata)))
  dist_mat = dist_mat[-seq(n_centers), seq(n_centers)]
  max.col(~dist_mat)
}

start_end_data = all_data[ ,
  list(
    lat1 = rawlat[1], lng1 = rawlng[1],
    latN = rawlat[N], lngN = rawlng[N]
  ), by = "trj_id" ]

kmeans_data = start_end_data[ , {
  cbind(
    lats = c( lat1, latN),
    lngs = c(lng1, lngN )
  ) } ]
```

```

kmeans_result = kmeans_map( kmeans_data, centers = 300, nstart = 1000, iter.max = 1e9 )

save( kmeans_result, start_end_data, kmeans_data, file = "deriving_landmarks_kmeans.RData" )

```

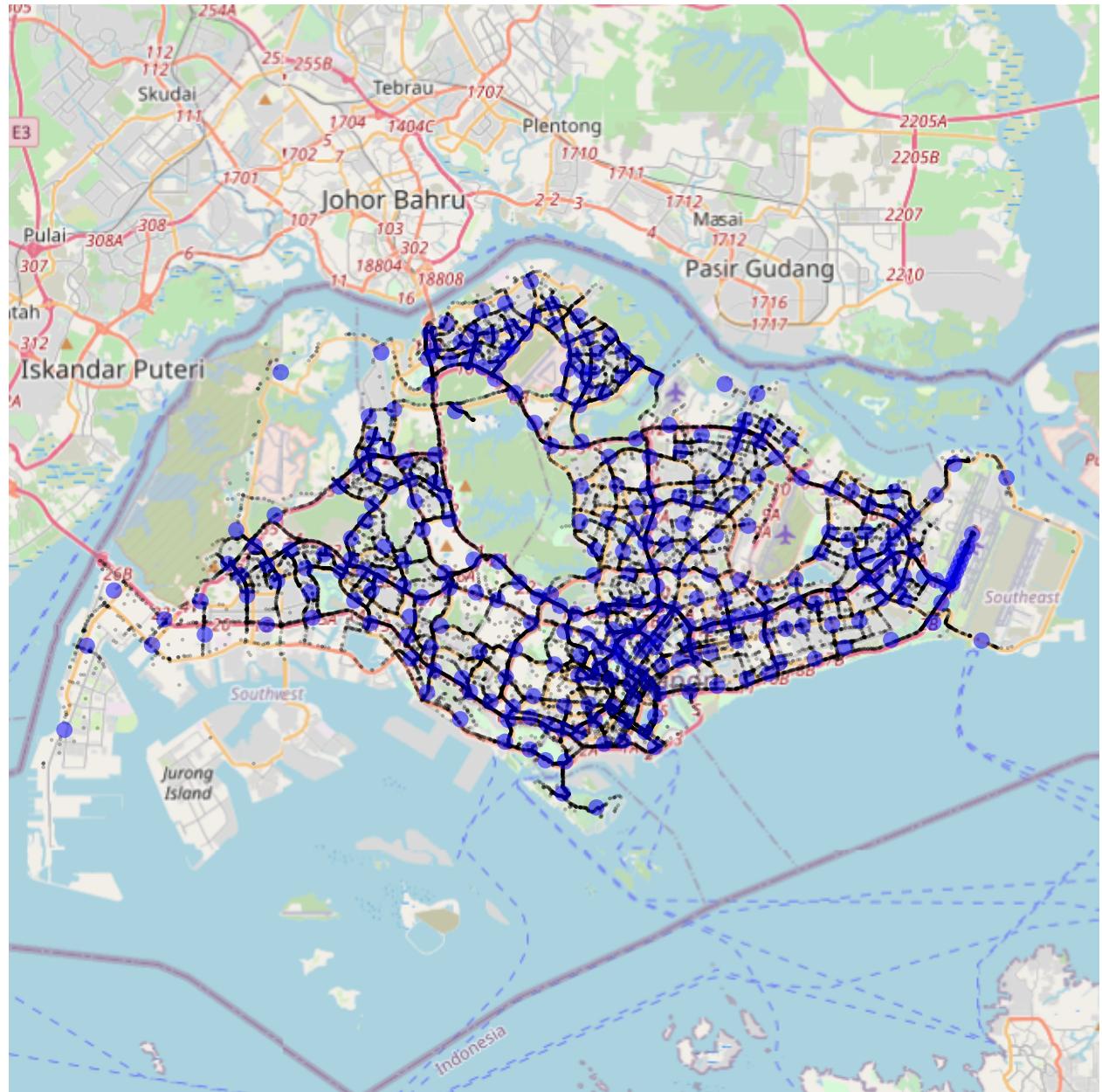
Put these points on a map

```

## sleptTotal= 0

## [1] "Caution: map type is OpenStreetMap. Until we find the correct projection algorithm, we treat la
## NULL
## NULL

```



K-means with outliers removed

I want to automatically detect high transit destinations. In order to do this, I will filter the points that have no neighbours within 500m.

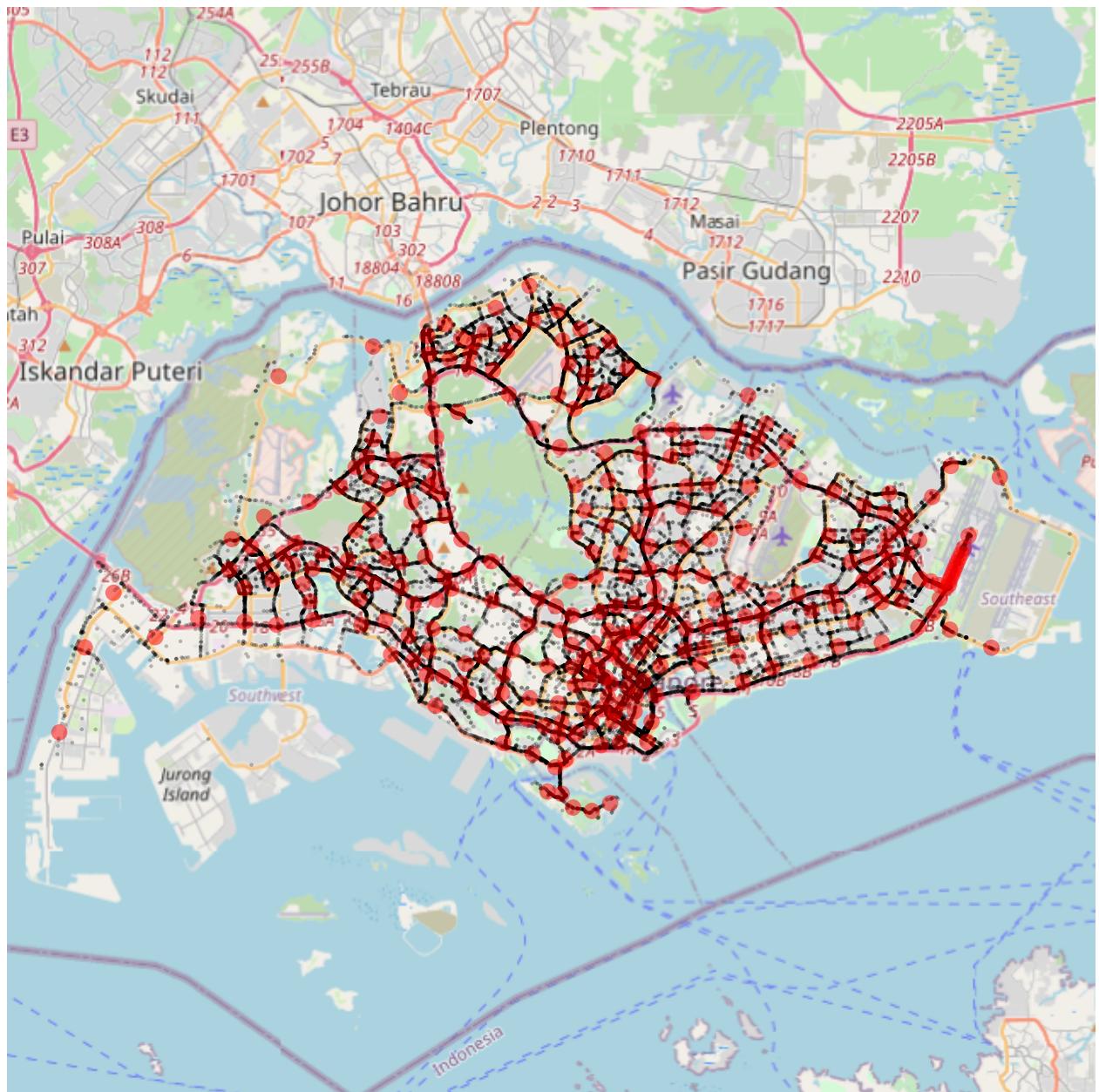
```
min_dist_km = 0.5
isolated_pts = rep( F, nrow( kmeans_data ) )
for ( i in 1:length(isolated_pts) ){
  if ( i %% 100 == 0 ) print( i )
  all_haversines = haversine2( kmeans_data[i,,drop = F], kmeans_data )
  isolated = (sum(all_haversines < min_dist_km) - 1) == 0
  if ( isolated ){
    isolated_pts[i] = T
  }
}

new_keans_data = kmeans_data[ !isolated_pts, ]

set.seed(312)
kmeans_result2 = kmeans_map( new_keans_data, centers = 300,
  nstart = 1000, iter.max = 1e9 )
save( new_keans_data, kmeans_result2, file = "deriving_landmarks_kmeans2.RData" )
```

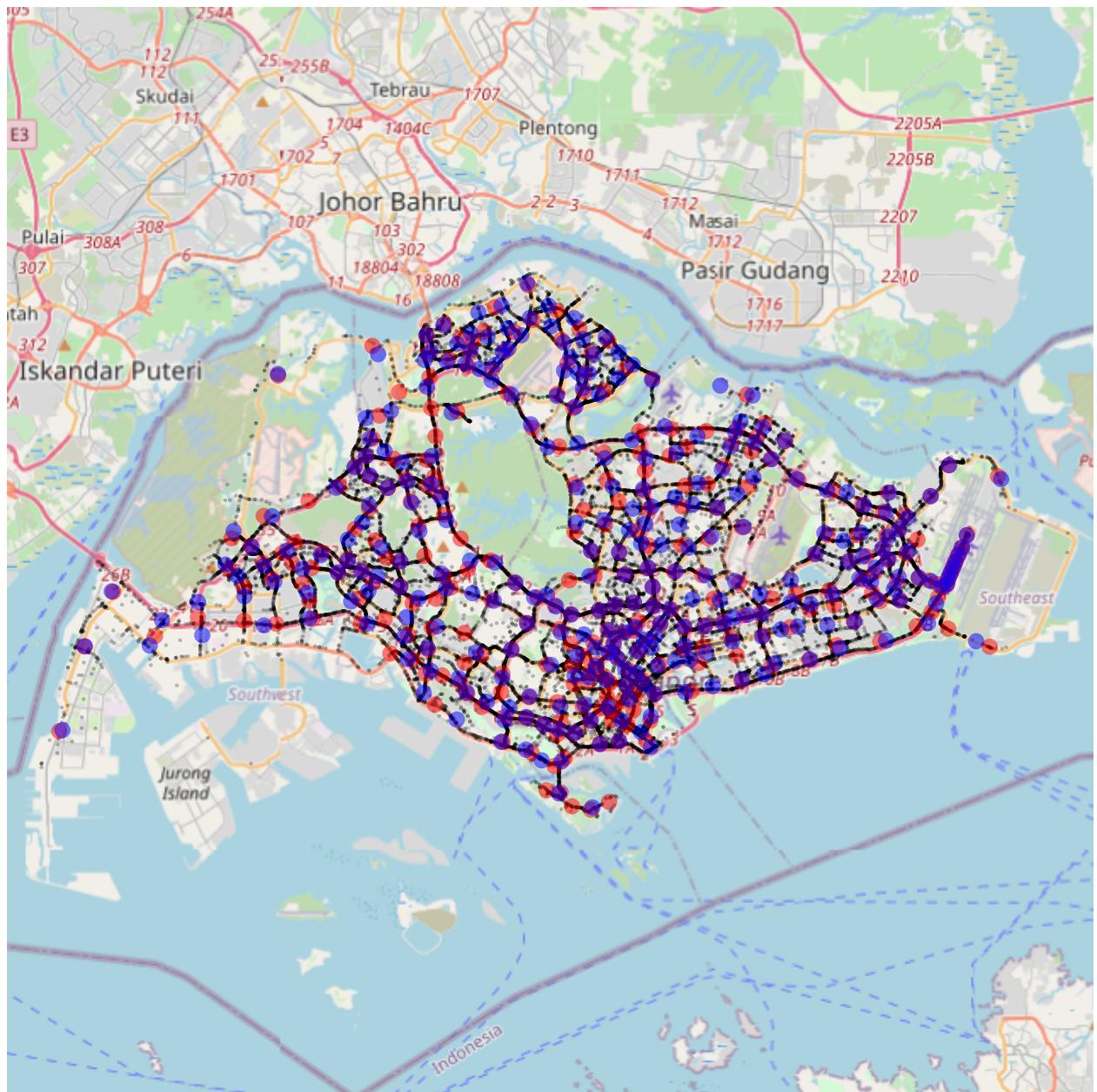
In my opinion, the clustering is nicer, especially around Jurong Island.

```
## [1] "Caution: map type is OpenStreetMap. Until we find the correct projection algorithm, we treat la
## NULL
## NULL
```



Overlay the two clusters to see how they differ. Picture below. Red is the clustering with the outliers removed. Blue is the clustering with the outliers in (first clustering). Expect some randomness due to the k-means clustering. But, the outliers removed tends to make a lot of random stuff. For example, down the road from Pasir Gudang, the outlier removed (red) makes one cluster, while the outlier kept (red) makes two.

```
## [1] "Caution: map type is OpenStreetMap. Until we find the correct projection algorithm, we treat la
## NULL
## NULL
```



Stacking clustering algorithms

Using the results from K-means (outliers removed), we can link clusters together using DBSCAN. This is useful for points like the airport, where there is a whole collection of points in a row.

The results below show the linkage of the K-means centers using DBSCAN. In particular, look at the orange points at the airport (labeled “2”), also at some points in the city center. We can use this cluster information to classify trips based on start or end.

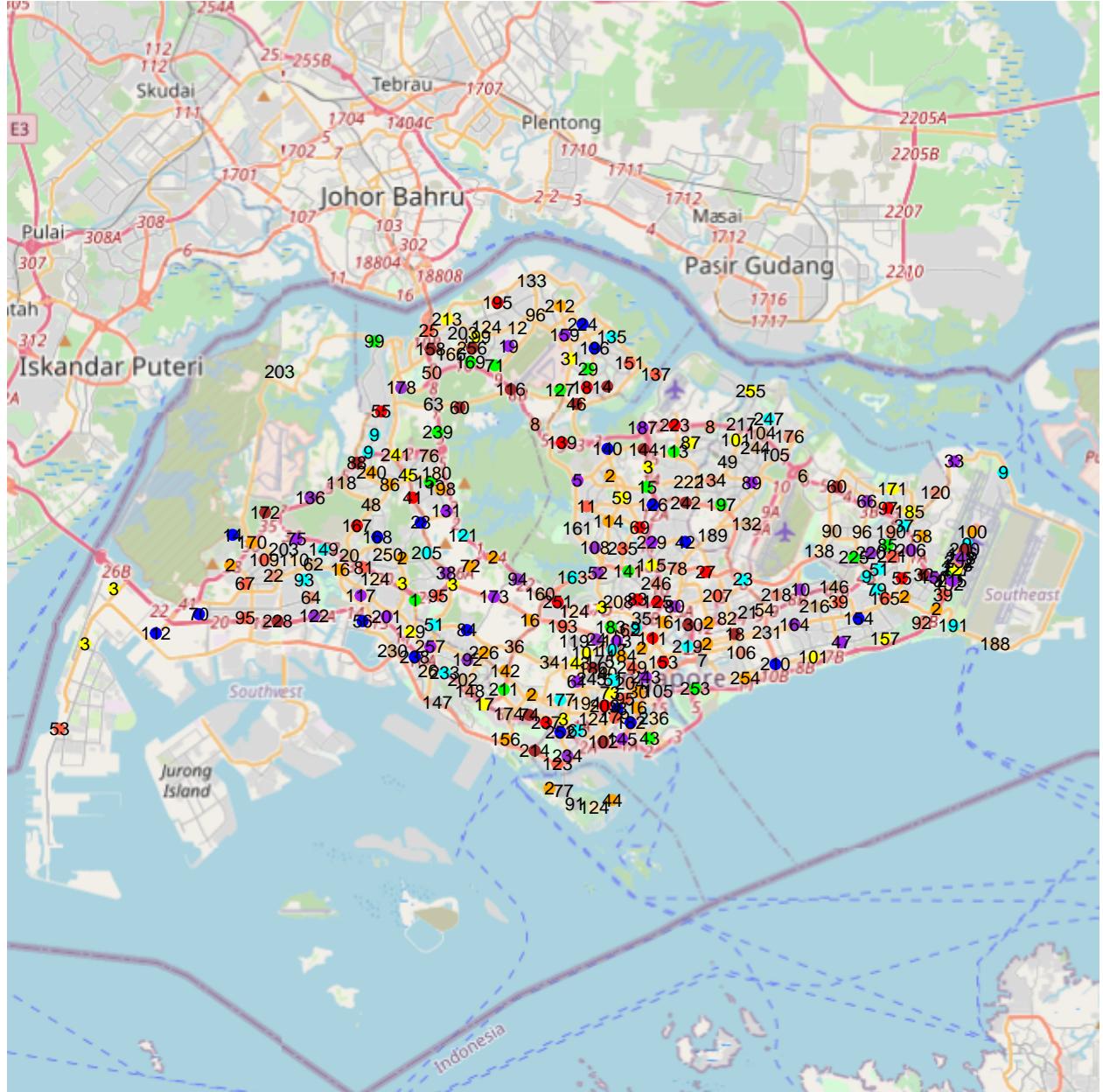
```
library( dbSCAN )
set.seed(42323)
dbSCAN_results = dbSCAN( kmeans_result2$centers, minPts = 1, eps = 0.0055 )
k2_centers = data.table(
```

```

  cbind( kmeans_result2$centers, size = kmeans_result2$size,
         cluster = dbscan_results$cluster )
)
k2_centers[ , total_size := sum(size), by = "cluster" ]
big_clusters = k2_centers[ total_size > 500 ]
save( dbscan_results, big_clusters, file = "deriving_landmarks_dbscan.RData" )

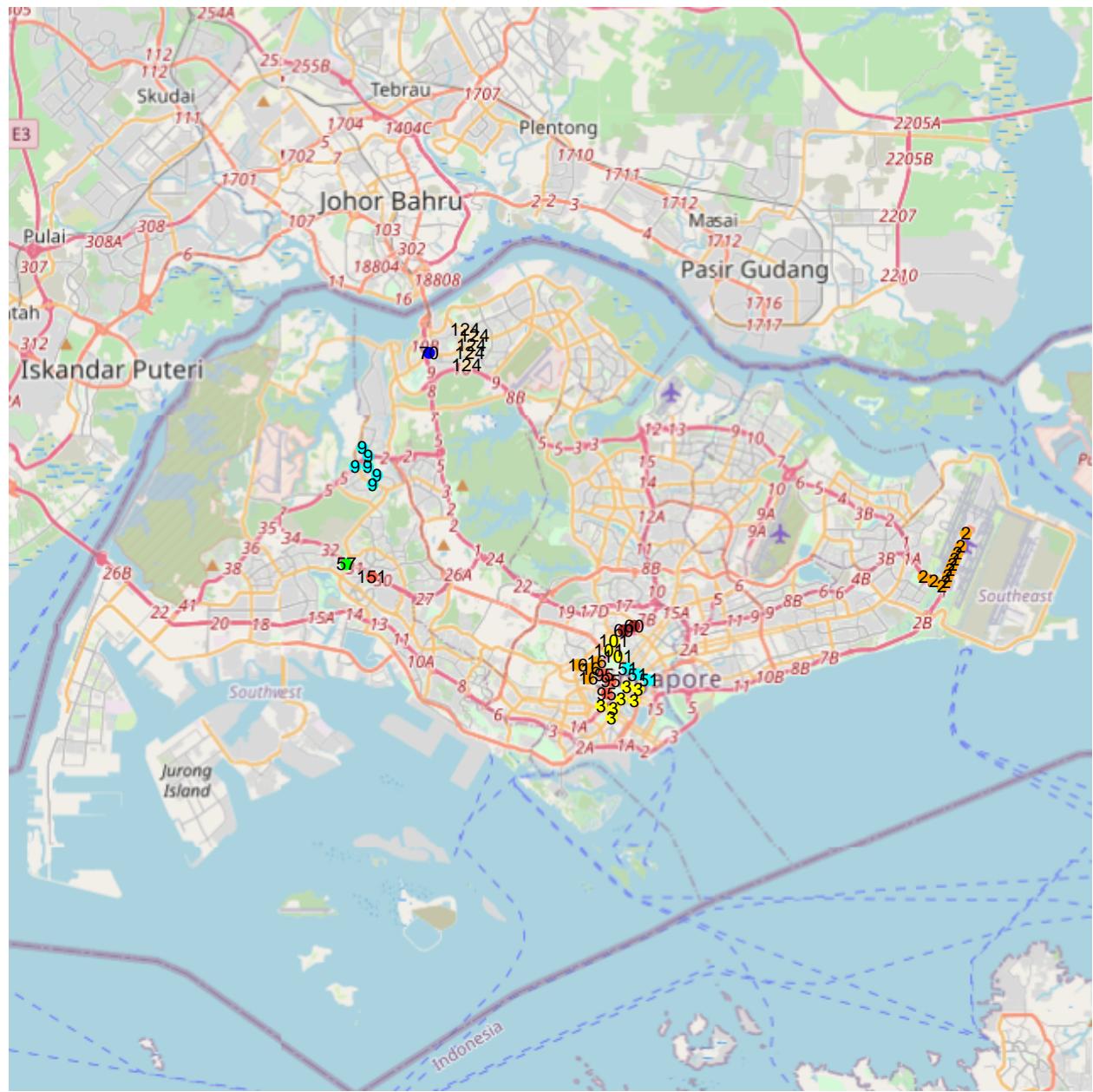
## [1] "Caution: map type is OpenStreetMap. Until we find the correct projection algorithm, we treat la

```



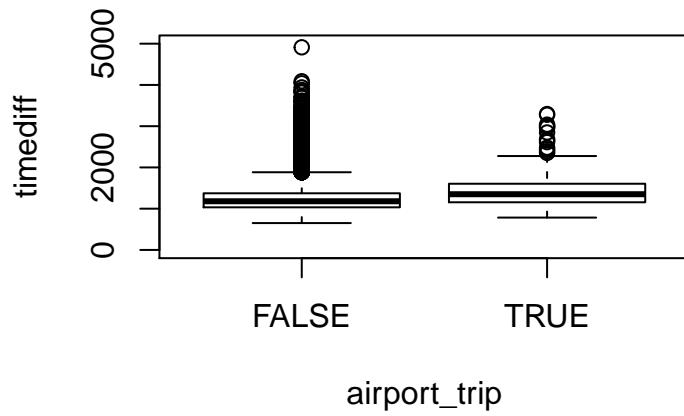
Count up the size of each cluster from K-means, summed over the cluster membership from DBSCAN and plot only the clusters with more than 100 total.

```
## [1] "Caution: map type is OpenStreetMap. Until we find the correct projection algorithm, we treat la
```



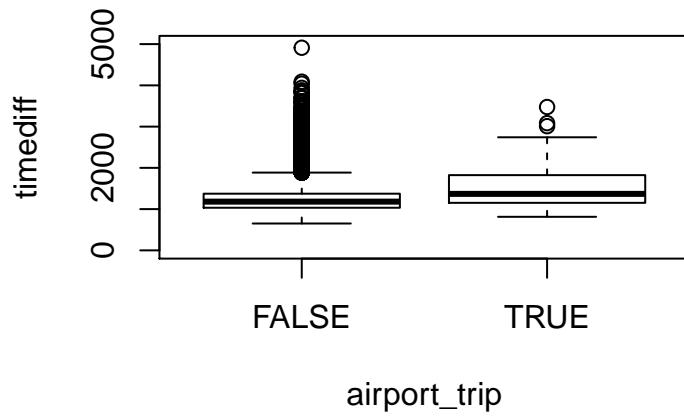
Proof of concept

To show that this was not for nothing, we can look at trips to specific places. We can see below that a trip to the airport does take longer than other trips.



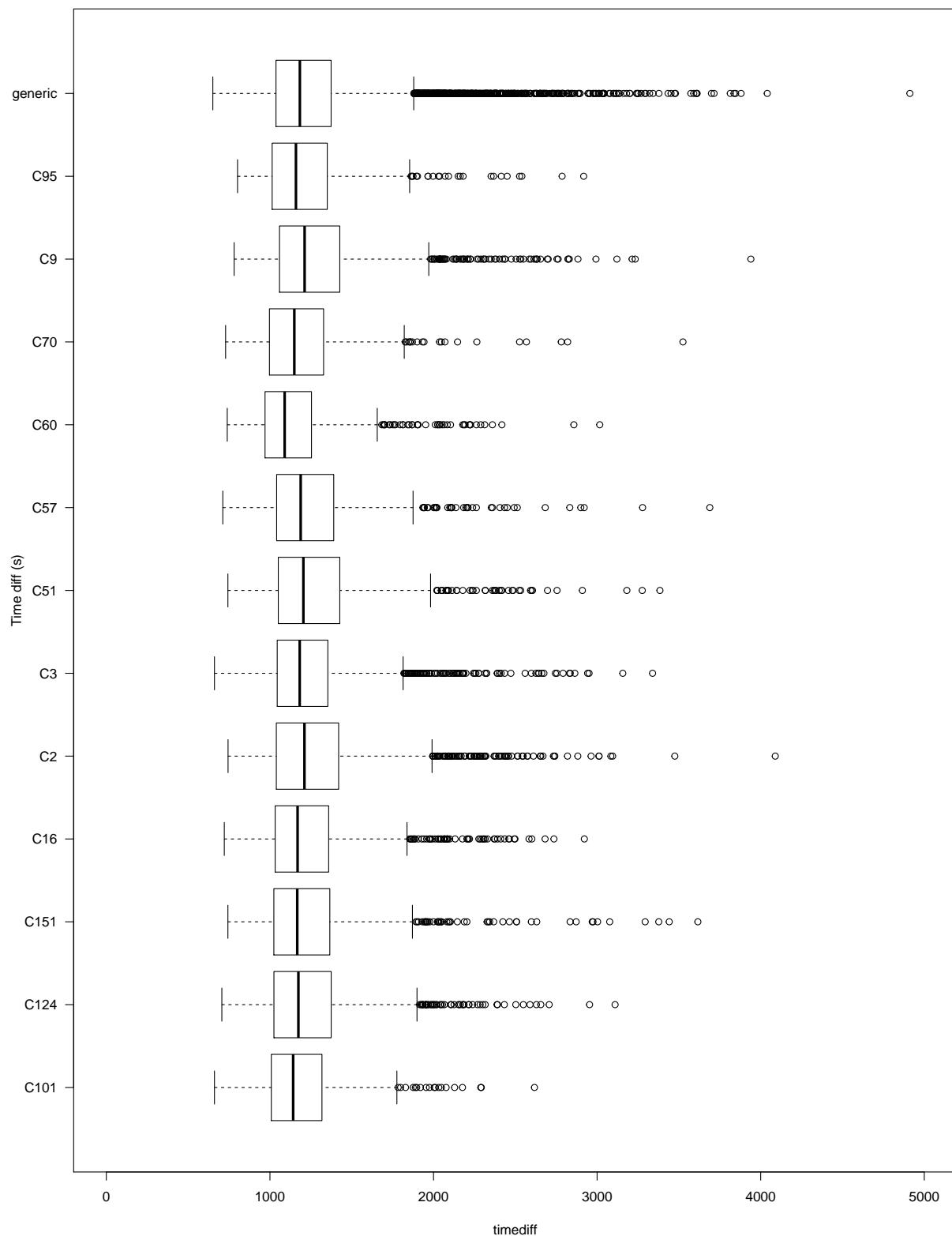
```
## NULL
```

Similarly, trips to the airport also take longer.

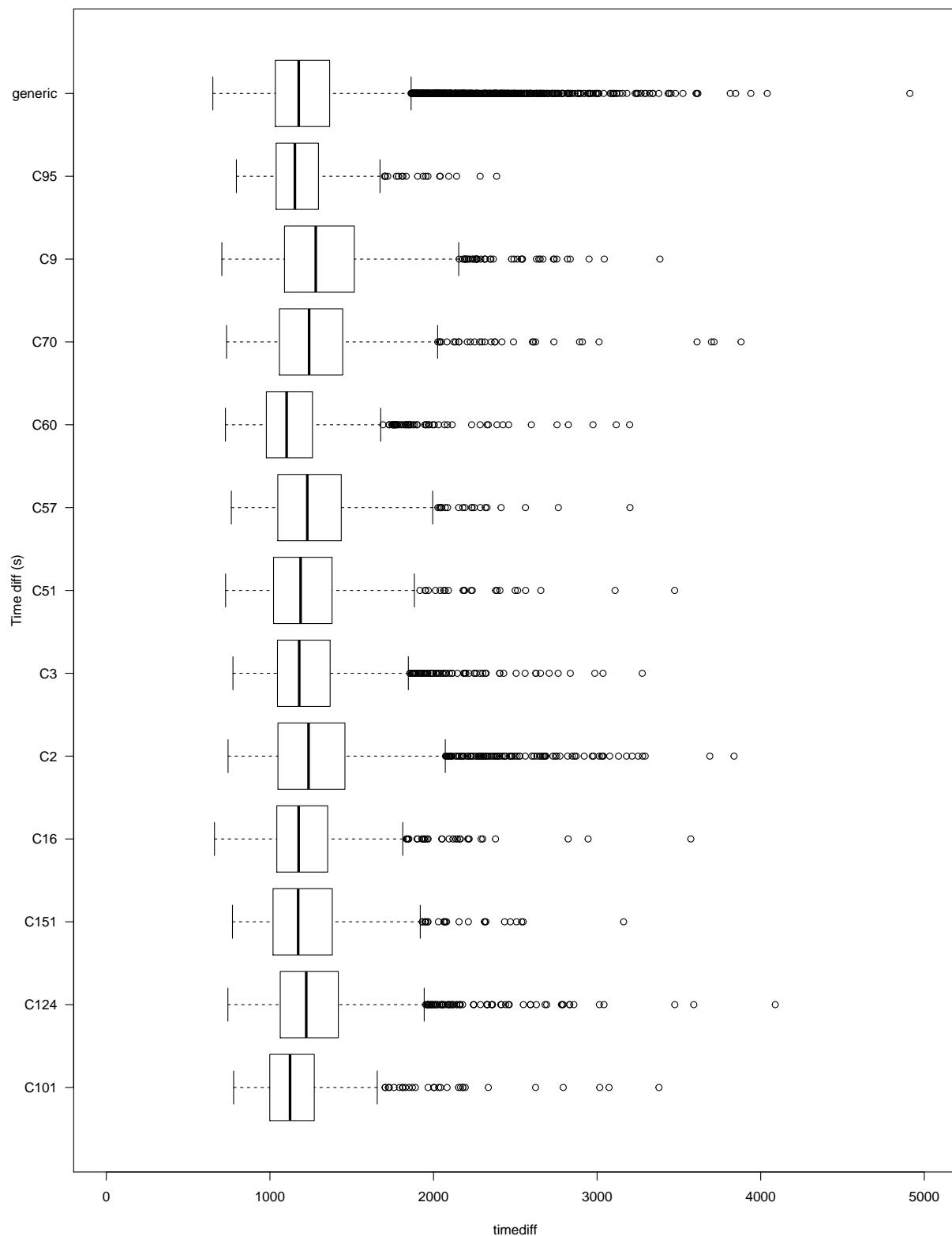


```
## NULL
```

We can do things more algorithmically:



```
## NULL
```



```
## NULL
```

```

##
## Call:
## lm(formula = tdiff2 ~ relevel(trip_Factor, ref = "generic") +
##      relevel(trip_Factor2, ref = "generic"), data = summary_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -659.2 -211.6 -65.6 123.0 15042.0 
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                 -1.9243   3.2851 -0.586
## relevel(trip_Factor, ref = "generic")C101 -79.8676 16.2730 -4.908
## relevel(trip_Factor, ref = "generic")C124  3.8244  9.7447  0.392
## relevel(trip_Factor, ref = "generic")C151  0.5945 11.1524  0.053
## relevel(trip_Factor, ref = "generic")C16 -30.0314 11.3295 -2.651
## relevel(trip_Factor, ref = "generic")C2   36.6564  7.6298  4.804
## relevel(trip_Factor, ref = "generic")C3  -27.3172  8.8877 -3.074
## relevel(trip_Factor, ref = "generic")C51  13.8851 12.6782  1.095
## relevel(trip_Factor, ref = "generic")C57  39.8445 12.6494  3.150
## relevel(trip_Factor, ref = "generic")C60 -124.2183 11.4901 -10.811
## relevel(trip_Factor, ref = "generic")C70 -43.0102 13.7513 -3.128
## relevel(trip_Factor, ref = "generic")C9   55.8726  9.2914  6.013
## relevel(trip_Factor, ref = "generic")C95 -21.5970 18.8175 -1.148
## relevel(trip_Factor2, ref = "generic")C101 -72.3761 13.9855 -5.175
## relevel(trip_Factor2, ref = "generic")C124  56.0451 10.1427  5.526
## relevel(trip_Factor2, ref = "generic")C151  3.3186 13.2976  0.250
## relevel(trip_Factor2, ref = "generic")C16 -19.8833 12.3205 -1.614
## relevel(trip_Factor2, ref = "generic")C2   79.1383  7.6828 10.301
## relevel(trip_Factor2, ref = "generic")C3  -13.6656  9.0560 -1.509
## relevel(trip_Factor2, ref = "generic")C51 -17.2659 13.7976 -1.251
## relevel(trip_Factor2, ref = "generic")C57  76.4836 14.5846  5.244
## relevel(trip_Factor2, ref = "generic")C60 -102.0410  9.1663 -11.132
## relevel(trip_Factor2, ref = "generic")C70  80.8403 14.6067  5.534
## relevel(trip_Factor2, ref = "generic")C9   114.7209 10.8139 10.609
## relevel(trip_Factor2, ref = "generic")C95 -60.8712 16.9118 -3.599
## Pr(>|t|)
## (Intercept)          0.55803
## relevel(trip_Factor, ref = "generic")C101 9.25e-07 ***
## relevel(trip_Factor, ref = "generic")C124  0.69472
## relevel(trip_Factor, ref = "generic")C151  0.95749
## relevel(trip_Factor, ref = "generic")C16   0.00804 **
## relevel(trip_Factor, ref = "generic")C2   1.56e-06 ***
## relevel(trip_Factor, ref = "generic")C3   0.00212 **
## relevel(trip_Factor, ref = "generic")C51  0.27344
## relevel(trip_Factor, ref = "generic")C57  0.00163 **
## relevel(trip_Factor, ref = "generic")C60 < 2e-16 ***
## relevel(trip_Factor, ref = "generic")C70  0.00176 **
## relevel(trip_Factor, ref = "generic")C9   1.84e-09 ***
## relevel(trip_Factor, ref = "generic")C95  0.25110
## relevel(trip_Factor2, ref = "generic")C101 2.29e-07 ***
## relevel(trip_Factor2, ref = "generic")C124 3.31e-08 ***
## relevel(trip_Factor2, ref = "generic")C151  0.80293
## relevel(trip_Factor2, ref = "generic")C16  0.10657

```

```
## relevel(trip_Factor2, ref = "generic")C2      < 2e-16 ***
## relevel(trip_Factor2, ref = "generic")C3      0.13131
## relevel(trip_Factor2, ref = "generic")C51     0.21081
## relevel(trip_Factor2, ref = "generic")C57     1.58e-07 ***
## relevel(trip_Factor2, ref = "generic")C60     < 2e-16 ***
## relevel(trip_Factor2, ref = "generic")C70     3.15e-08 ***
## relevel(trip_Factor2, ref = "generic")C9      < 2e-16 ***
## relevel(trip_Factor2, ref = "generic")C95    0.00032 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 340 on 27975 degrees of freedom
## Multiple R-squared:  0.02282,   Adjusted R-squared:  0.02198
## F-statistic: 27.22 on 24 and 27975 DF,  p-value: < 2.2e-16
```

Conclusion

We can see that classifying trips could be a useful covariate.