cycleparking

Yuhei Ito

2020/10/7

Read me

input data

census.rda: A census data of 2011 subsetting by Greater London. route.rda: route data obtained from PCT. parking.rda: parking data merged from CID and OSM. lsoa_msoa.rda: LSOA and MSOA link.

Load data

Join MSOA code to census

```
## [1] FALSE
```

```
## [1] FALSE
```

Join MSOA code to parking

```
## [1] FALSE
```

Aggregate census' all and bicycle by MSOA.

```
## 'summarise()' regrouping output by 'omsoa' (override with '.groups' argument)
```

```
## [1] TRUE
```

Add distance and hilliness to census2.

```
## [1] FALSE
```

```
## [1] 308782
```

```
## [1] 308782
```

```
## [1] 617564
```

Join distance and hilliness to census2.

```
## [1] TRUE
```

```
## # A tibble: 219,452 x 6
## # Groups:   omsoa [7,147]
##    omsoa      dmsoa        all bicycle rf_dist_km rf_avslope_perc
##    <chr>      <chr>      <int>   <int>      <dbl>           <dbl>
##  1 E02000001 E02000509      1       0         NA              NA
##  2 E02000002 E02000053      2       0         NA              NA
##  3 E02000002 E02000093      3       0         NA              NA
##  4 E02000002 E02000106      2       0         NA              NA
##  5 E02000002 E02000116      1       0         NA              NA
##  6 E02000002 E02000119      2       0         NA              NA
##  7 E02000002 E02000150      1       0         NA              NA
##  8 E02000002 E02000220      1       0         NA              NA
##  9 E02000002 E02000225      1       0         NA              NA
## 10 E02000002 E02000244      1       0         NA              NA
## # ... with 219,442 more rows
```

Check census data.

```
## [1] 601902
```

```
## # A tibble: 601,902 x 7
## # Groups:   omsoa [7,189]
##    omsoa      dmsoa        all bicycle rf_dist_km rf_avslope_perc First3D
##    <chr>      <chr>      <int>   <int>      <dbl>           <dbl> <chr>
##  1 E02000001 E02000001   1506      33       1.05            1.64 E02
##  2 E02000001 E02000014      2       0       19.1            0.77 E02
##  3 E02000001 E02000016      3       0       13.9            0.87 E02
##  4 E02000001 E02000025      1       0       19.4            1.66 E02
##  5 E02000001 E02000028      1       0       18.5            1.58 E02
##  6 E02000001 E02000051      1       0       13              1.87 E02
##  7 E02000001 E02000053      2       0       15.8            2.34 E02
##  8 E02000001 E02000057      1       0       12.8            2.28 E02
##  9 E02000001 E02000058      1       0       12              2.31 E02
## 10 E02000001 E02000059      1       0       14.8            1.6  E02
## # ... with 601,892 more rows
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 4
##   First3O `n()` `sum(all)` `sum(bicycle)`
##   <chr>   <int>      <int>          <int>
## 1 E02    598276    3711412         147102
## 2 W02      3626       4570             97
```

```
## [1] 983
```

Check how many trips are excluded and how many are intra/inter zonal trips. (Table 6. Number of Commuters in/to Greater London)

```
## [1] 3715982
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
## # A tibble: 2 x 4
##   intra 'n()' 'sum(all)' 'sum(bicycle)'
##   <lgl> <int>     <int>          <int>
## 1 FALSE 600919   3589916         143899
## 2 TRUE     983    126066           3300
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
## # A tibble: 2 x 4
##   excluded 'n()' 'sum(all)' 'sum(bicycle)'
##   <lgl>    <int>     <int>          <int>
## 1 FALSE   382450   3145728         143182
## 2 TRUE    219452    570254           4017
```

```
## 'summarise()' regrouping output by 'excluded' (override with '.groups' argument)
```

```
## # A tibble: 3 x 5
## # Groups:   excluded [2]
##   excluded intra 'n()' 'sum(all)' 'sum(bicycle)'
##   <lgl>    <lgl> <int>     <int>          <int>
## 1 FALSE    FALSE 381467   3019662         139882
## 2 FALSE    TRUE     983    126066           3300
## 3 TRUE     FALSE 219452    570254           4017
```

```
## [1] 36.45976
```

```
## [1] 15.34598
```

```
## [1] 2.728959
```

BCG construction Step 1: Eliminating >30km od and CalculateGoDutch propensity

BCG construction Step 2: Calculate GoDutch number of cycle commuters for each OD pair.

BCG construction Step 3: Aggregate by destination MSOA.

BCG construction Step 4: Construct BCG by mean values

Parking 1: Aggregate parking space for each MSOA

Parking 2: Join parking space to census

Parking Occupancy Rate: por

4 Categories: Surplus : Less than 50% occupancy rate Ideal : 50% and more and 80% and less occupancy rate Threshold: More than 80% and 100% and less occupancy rate Shortage : More than 100% occupancy rate

Examples: Surplus : bicycle= 30, parking=100 - Less than 50% occupancy rate Ideal : bicycle= 70, parking=100 - 50% and more and 80% and less occupancy rate Threshold: bicycle= 90, parking=100 - More than 80% and 100% and less occupancy rate Shortage : bicycle=110, parking=100 - More than 100% occupancy rate

Exceptions: Case 1 : bicycle= 0, parking=100 -> Surplus (n = 22 current) Case 2 : bicycle=10, parking=0 -> Shortage (n = 1 current) Case 3 : bicycle= 0, parking=0 -> Shortage (n = 0 observation)

Current Occupancy Rate and Category

Future Occupancy Rate and Category

por checking

Reorder factor of por categorisation

Figures