Using R as a GIS

授課教師：溫在弘
E-mail: wenthung@ntu.edu.tw

Contents

- Using R as a GIS
  - 1. Spatial Join
  - 2. Buffering & Merging Spatial Features
  - 3. Data Join
  - 4. Point-in-Polygon and Area Calculations
  - 5. Distance Analysis

R Examples for Geo-processing

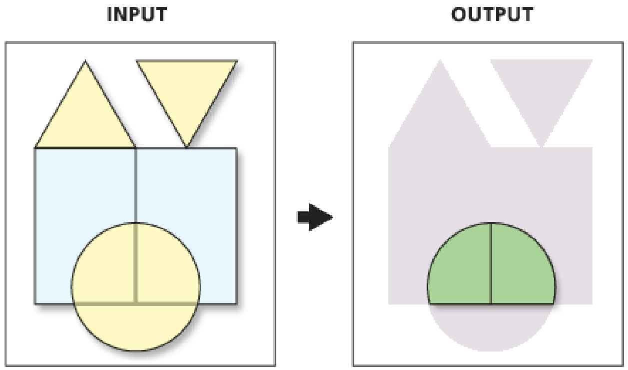
- Task 1.1: Tornado damage assessment (Spatial Join)
- Task 1.2: Creating a tornado density map (Spatial Join)
- Task 2: Creating service areas (Buffer Zone)
- Task 3.1: Identifying service areas (Distance Analysis)
- Task 3.2: Geographical accessibility (Distance Analysis)

Introducing **R functions** for spatial analysis

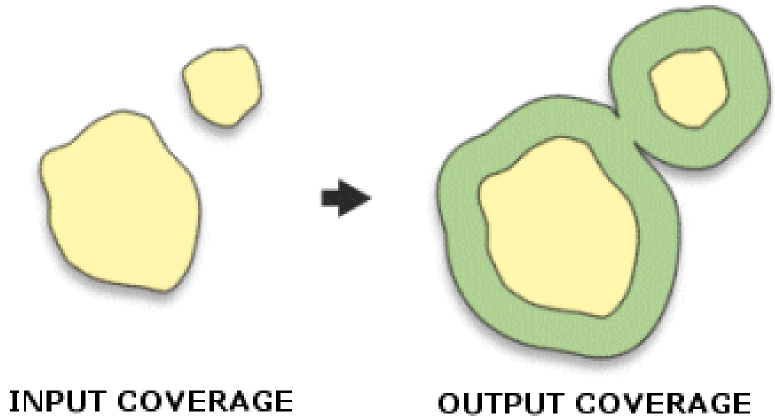
- Spatial Join (Point-In-Polygon): st_join()
- Buffering: st_buffer()
- Merging Spatial Features: st_union()
- Data Join: left_join() [dplyr套件]
- Area Calculation: st_area()
- Distance Matrix: st_distance() + st_is_within_distance ()

Spatial Operational Functions in GIS

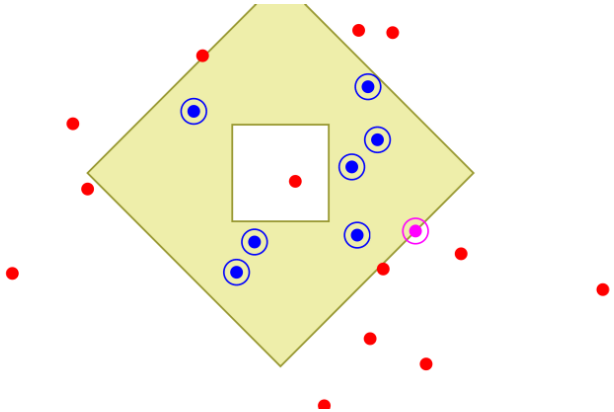
Spatial Intersection



Buffering

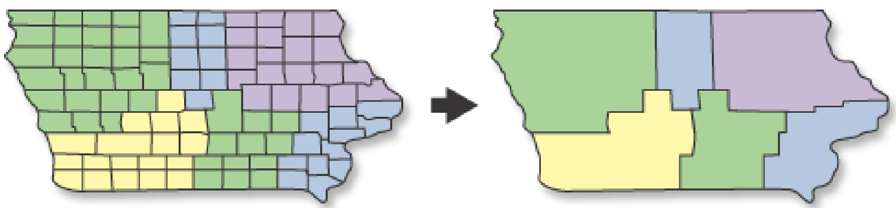


Spatial Join: Point-in-Polygon



Spatial Operational Functions in GIS

Merging Features (dissolve)



INPUT   OUTPUT

Data Join

GIS layer (shp)   Table (txt)

INPUT

OBJECT ID#	Landuse Code
1	2
2	0
3	1

← Join Fields →

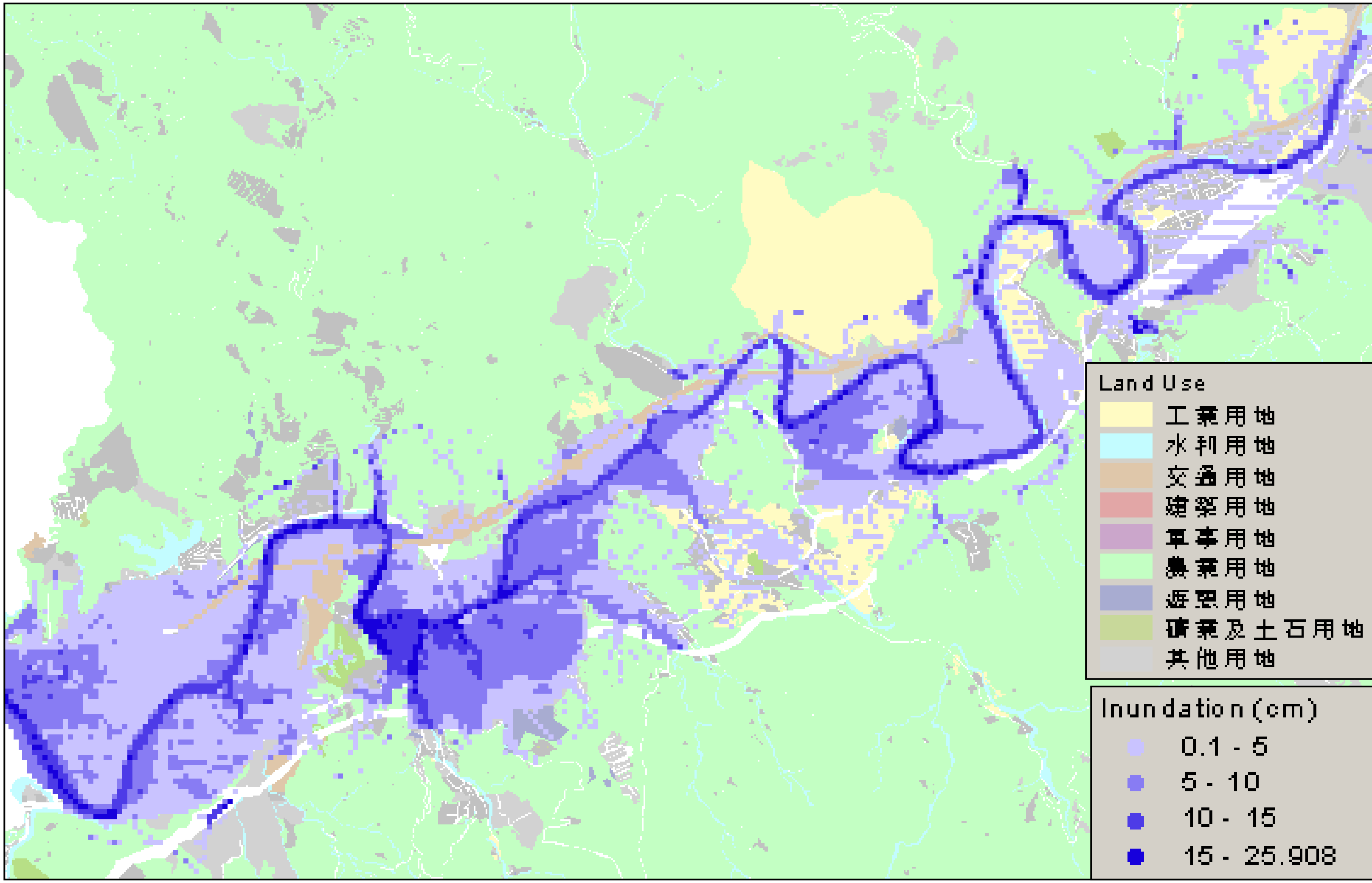
\+

Landuse Code	Landuse Type
0	Unclassified
1	shrub
2	water

OUTPUT

OBJECT ID#	Landuse Code	Join Table Landuse Code	Join Table Landuse Type
1	2	2	water
2	0	0	Unclassified
3	1	1	shrub

Spatial Intersection: Flooding Risk Analysis



土地利用	面積(平方公尺)	面積百分比 %
工業用地	4,306,266	10.4
水利用地	2,777,436	6.7
交通用地	1,225,045	3.0
建築用地	4,073,363	9.9
軍事用地	197,126	0.5
農業用地	24,841,284	60.1
遊憩用地	575,589	1.4
礦業及土石用地	2,653	0.0
其他用地	3,317,054	8.0
200-Yrs洪氾區	**41,315,816**	**100.0**

	洪水淹沒面積	面積百分比 %
小於5 cm	10,312,488	66.50
5-10 cm	3,889,655	25.08
10-15 cm	1,189,358	7.67
15-26 cm	115,555	0.75
200-Yrs洪氾區	15,507,056	100.00

Spatial Intersection: Two-way table

土地利用 / 淹水深度	小於5 cm	5-10 cm	10-15 cm	15-26 cm	淹沒面積 (平方公尺)
工業用地	1,644,814	406,837	6,836	0	**2,058,487**
水利用地	440,763	956,183	841,913	85,017	**2,323,876**
交通用地	793,649	126,423	27,692	0	**947,764**
建築用地	2,323,457	797,133	71,197	5,990	**3,197,776**
軍事用地	77,154	25,713	0	0	**102,867**
農業用地	3,091,893	818,561	120,144	12,185	**4,042,783**
遊憩用地	287,854	185,023	32,250	3,366	**508,493**
礦業及土石用地	780	628	0	0	**1,408**
其他用地	1,652,125	573,154	89,326	8,997	**2,323,602**
淹沒面積 (平方公尺)	**10,312,488**	**3,889,655**	**1,189,358**	**115,555**	**15,507,056**

本週課程圖資

https://wenlab501.github.io/GEOG2017/DATA/

空間分析

課程圖資清單

課程提供之圖資僅供練習使用

Sample2	.Rdata	R空間繪圖範例資料2	配合課程【2】

```
rm(list = ls())

library(sf)
library(tmap)

load("./data/Sample2.RData")
```

Data	
blocks_sf	129 obs. of 29 variables
georgia2_sf	159 obs. of 15 variables
places_sf	9 obs. of 2 variables
torn_sf	46931 obs. of 24 variables
us_states_sf	49 obs. of 52 variables

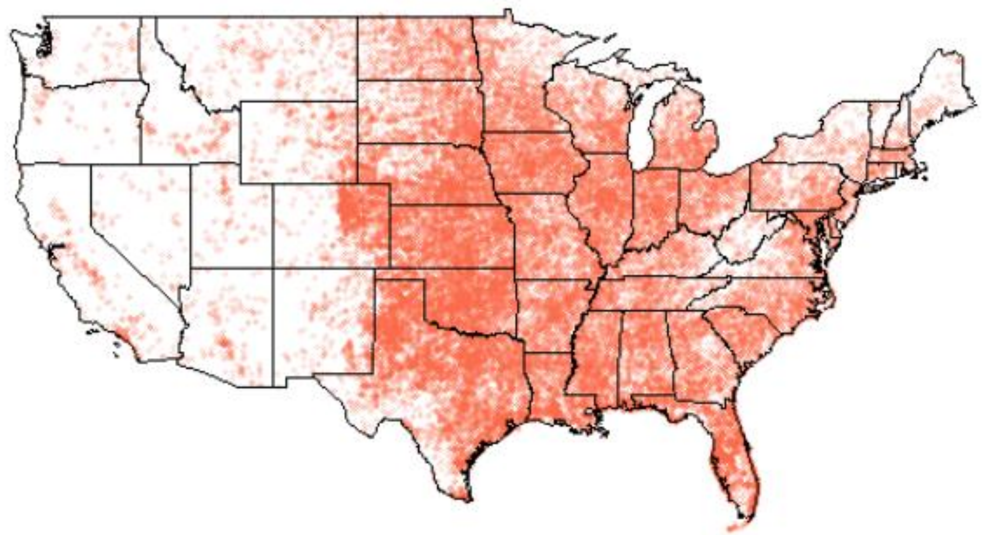
Spatial Join

Selecting Layer 1 in the Layer 2

Data	
blocks_sf	129 obs. of 29 variables
georgia2_sf	159 obs. of 15 variables
places_sf	9 obs. of 2 variables
torn_sf	46931 obs. of 24 variables
us_states_sf	49 obs. of 52 variables

Layer 1: tornados (torn_sf)

Layer 2: US States



Selecting tornados in the Area of Interest (Texas, New Mexico, Oklahoma, Arkansas)

Spatial Join

Layer 1: tornados (torn_sf) + Layer 2: US States

```
Simple feature collection with 6 features and 23 fields
geometry type:  POINT
dimension:      XY
bbox:           xmin: -94.37 ymin: 33.27 xmax: -91.83 ymax: 36.15
CRS:             +proj=longlat +ellps=WGS84
  TORNADX020 YEAR NUM STATE MONTH DAY      DATE TOR_NO NO_STS STATE_TOR SEGNO STLAT STLON SPLAT SPLON
2          3 1950   1    05     1  13 1/13/1950      4      1         1     1 34.40 94.37  0.00  0.00
3          4 1950   2    05     2  12 2/12/1950     19      1         1     1 34.48 92.40  0.00  0.00
4          5 1950   3    05     2  12 2/12/1950     23      1         1     1 33.27 92.95 33.35 92.95
5          6 1950   4    05     3  26 3/26/1950     33      1         1     1 34.12 93.07 34.32 92.88
6          7 1950   5    05     3  26 3/26/1950     34      1         1     1 36.15 91.83 36.20 91.75
7          8 1950   6    05     3  26 3/26/1950     35      1         1     1 34.70 92.35 34.80 92.22
  LGTH WIDTH FATAL INJ DAMAGE F_SCALE coords.x1 coords.x2                geometry
2    6     5     1   1      3       3    -94.37     34.40  POINT (-94.37 34.4)
3    1    30     0   0      3       2    -92.40     34.48  POINT (-92.4 34.48)
4   57    30     0   0      4       2    -92.95     33.27 POINT (-92.95 33.27)
5  174    45     0   3      4       2    -93.07     34.12 POINT (-93.07 34.12)
6   57    60     0   1      5       3    -91.83     36.15 POINT (-91.83 36.15)
7  104   180     0   7      5       2    -92.35     34.70  POINT (-92.35 34.7)
```

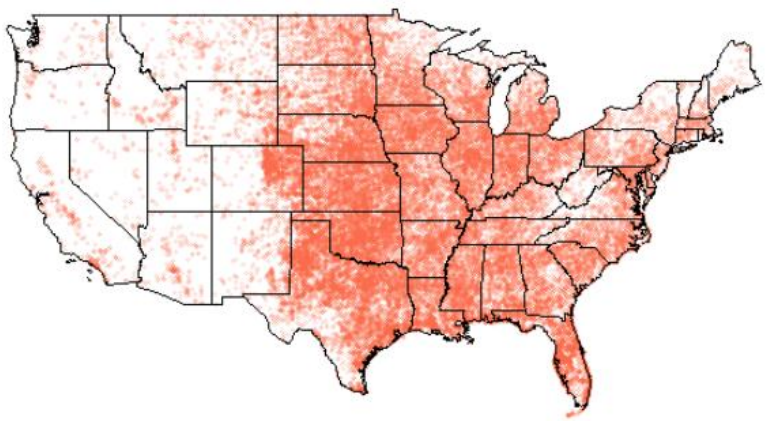
預期成果：Spatial Join

Task 1.1: Tornado damage assessment (creating a two-way table)

	Arkansas	New Mexico	Oklahoma	Texas
0	563	283	1280	4053
1	23	35	211	378
2	39	33	124	235
3	132	49	376	676
4	235	49	506	919
5	222	25	359	560
6	60	2	92	165
7	15	1	17	36

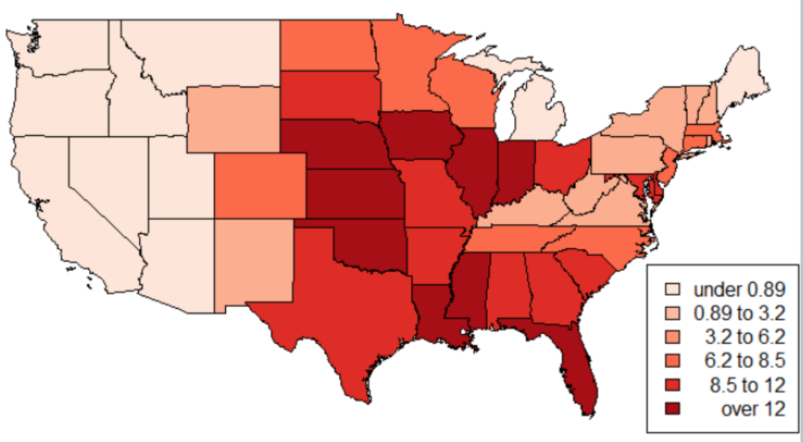
預期成果：Spatial Join

Task 1.2: Creating a tornado density map



Locations of tornados: point event

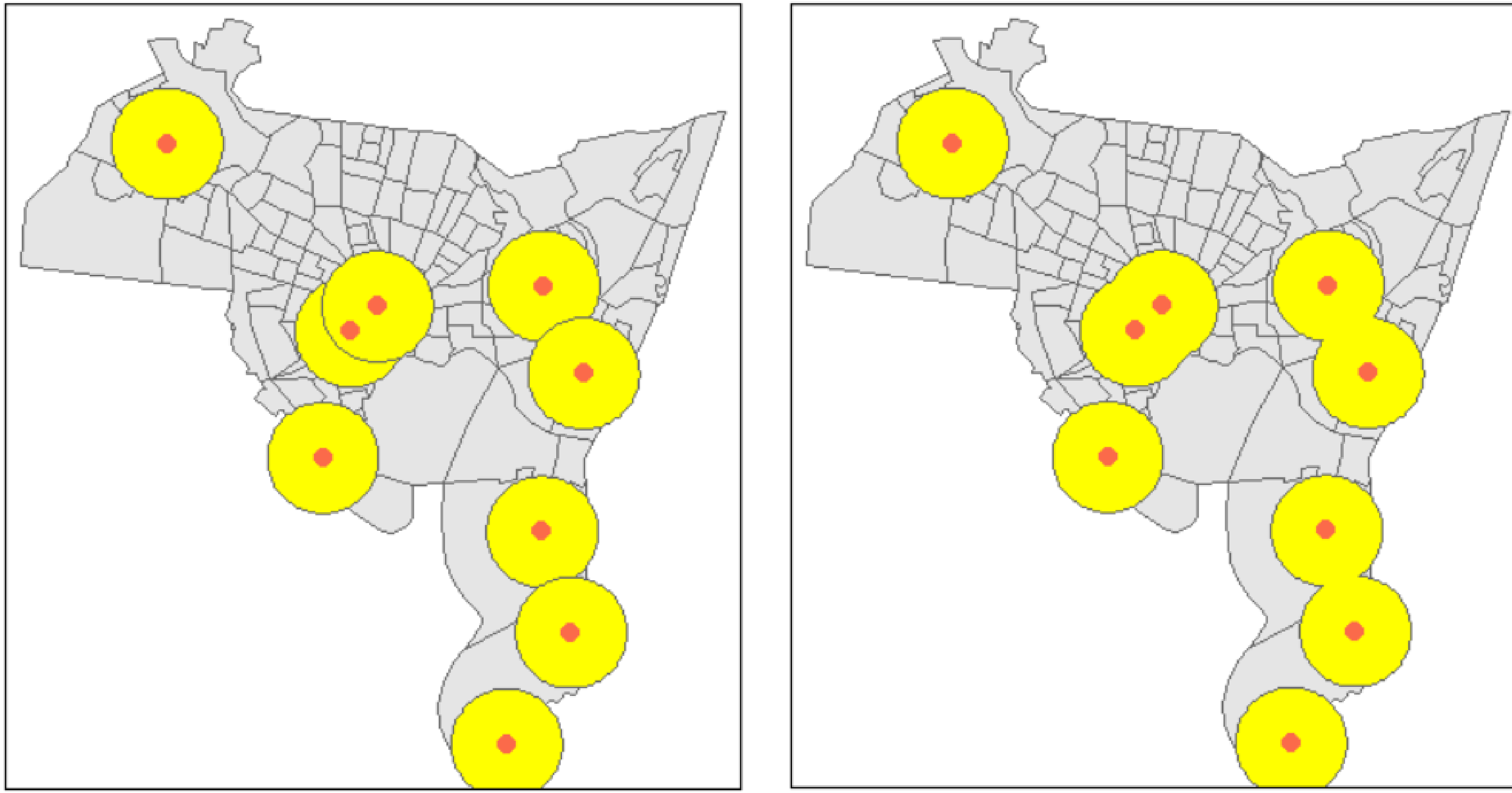
Density of tornados in each state



預期成果：Buffer Zone

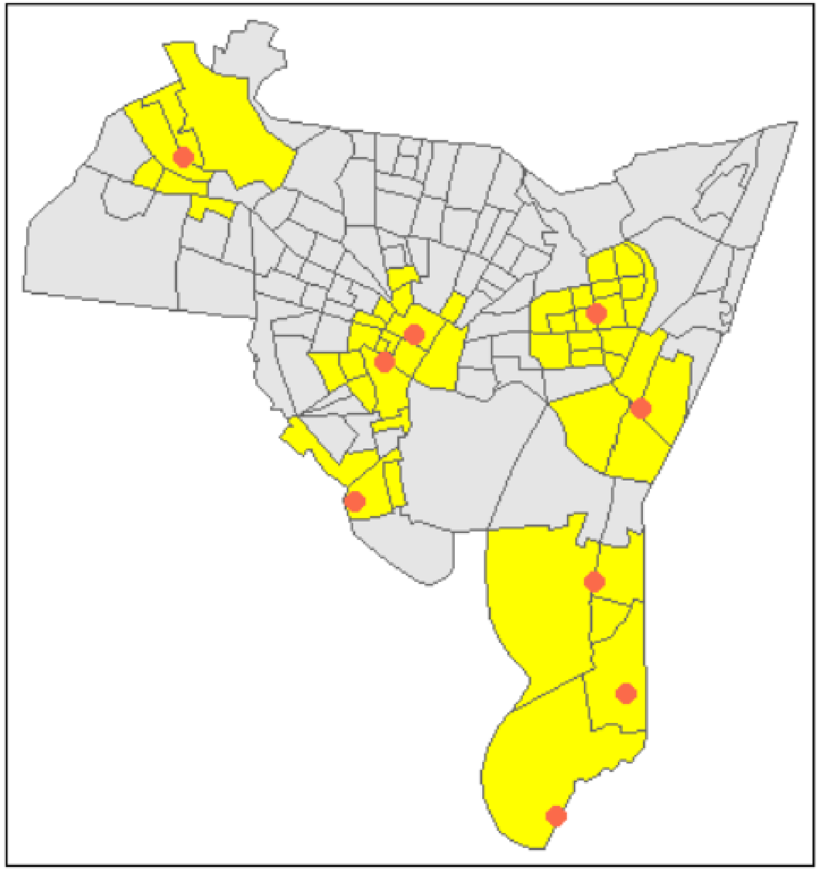
Task 2: Creating service areas

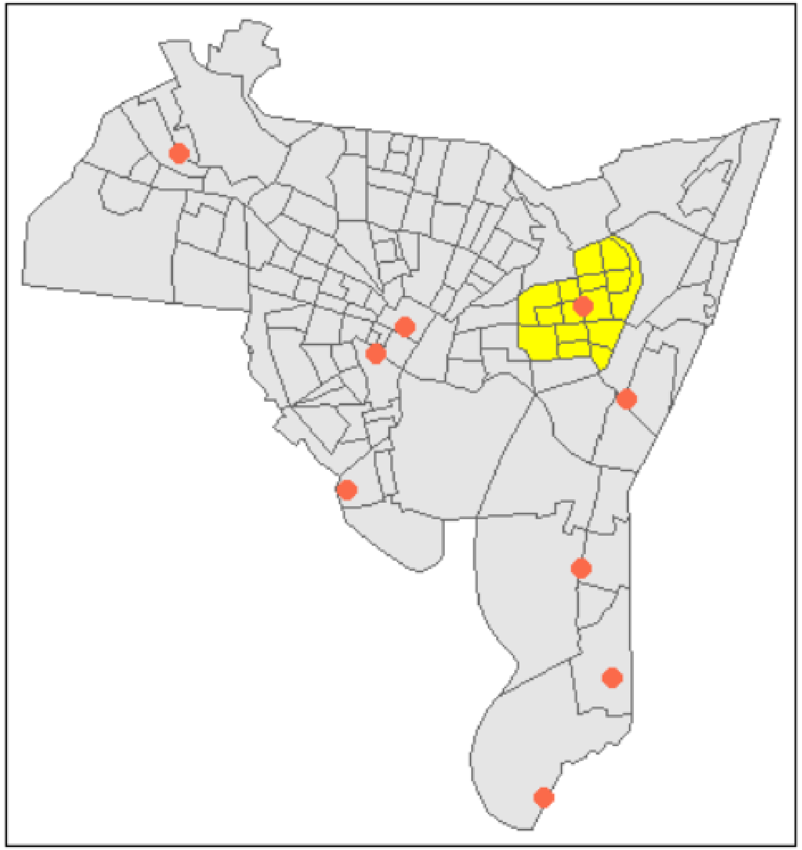
Data	
blocks_sf	129 obs. of 29 variables
georgia2_sf	159 obs. of 15 variables
places_sf	9 obs. of 2 variables
torn_sf	46931 obs. of 24 variables
us_states_sf	49 obs. of 52 variables



預期成果：Distance Analysis

Task 3.1: Identifying service areas

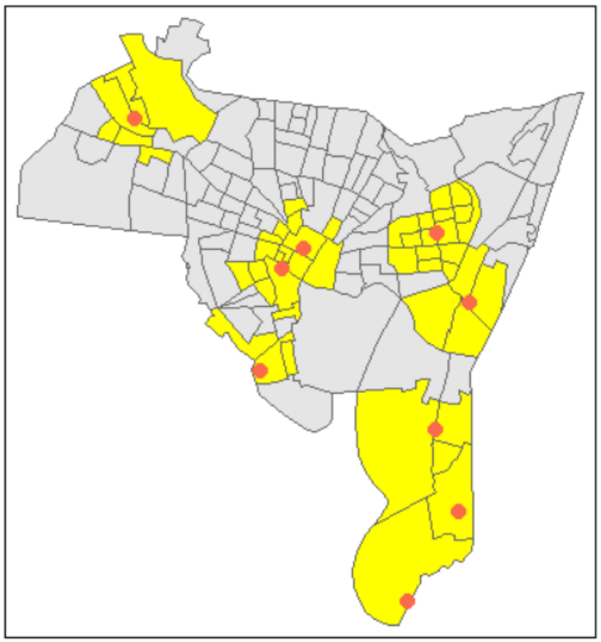


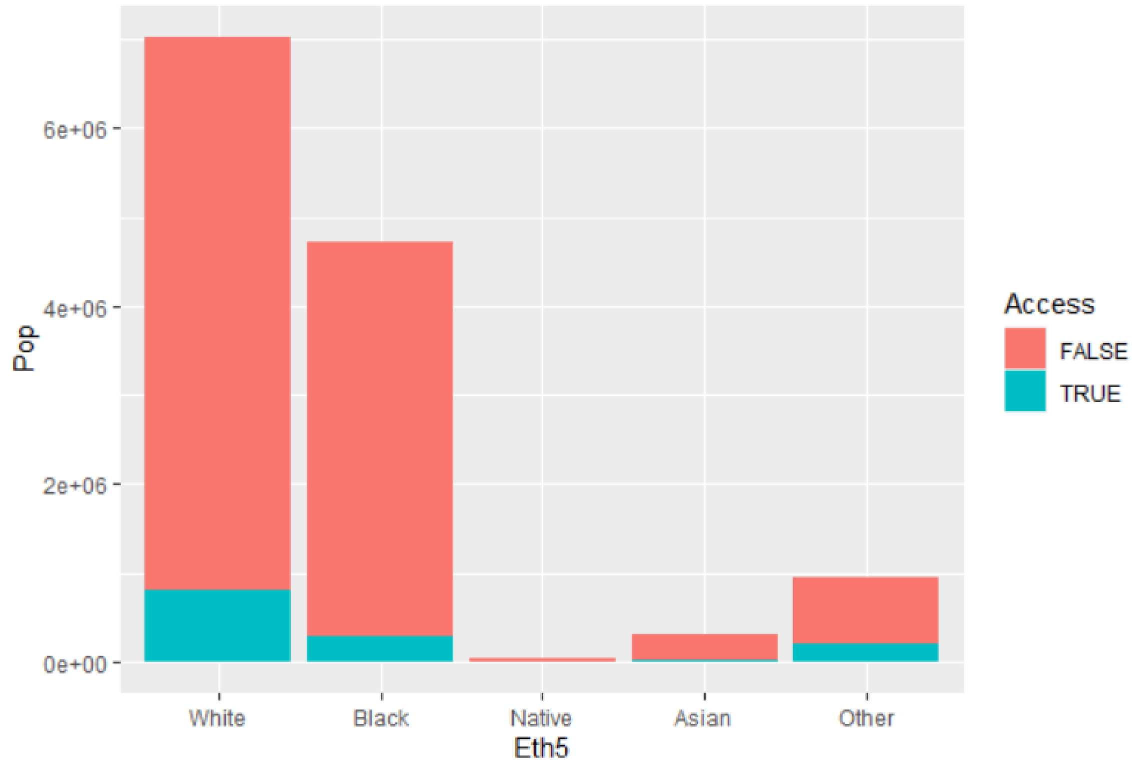


TOTAL_POP	13081
TOTAL_WHITE	8028.00001902

預期成果：Distance Analysis

Task 3.2: Geographical accessibility

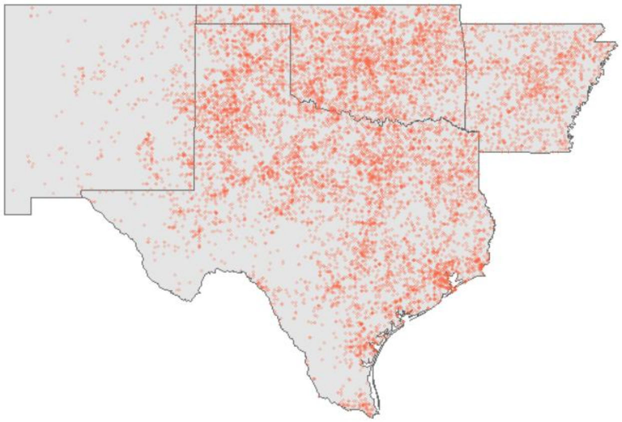




Task 1.1: Tornado damage assessment

(creating a two-way table)

1. Mapping
2. Selecting area + clip
3. Spatial join
4. Creating new data frame
5. Crosstab analysis



```
> head(newdf)
      Name Damage
1 Arkansas      3
2 Arkansas      3
3 Arkansas      4
4 Arkansas      4
5 Arkansas      5
6 Arkansas      5
```

Task 1.1: Tornado damage assessment

2. Selecting area+ clip

```
index <- us_states_sf$STATE_NAME == "Texas" |
         us_states_sf$STATE_NAME == "New Mexico" |
         us_states_sf$STATE_NAME == "Oklahoma" |
         us_states_sf$STATE_NAME == "Arkansas"

AoI_sf <- us_states_sf [index,]
AoI_bg <- tm_shape(AoI_sf) + tm_polygons("grey90")

torn_clip_sf <- torn_sf [AoI_sf, ]

torn_clip <- tm_shape(torn_clip_sf) + tm_dots(fill = "red", size = 0.04, fill_alpha = 0.5)
AoI_bg + torn_clip
```

Task 1.1: Tornado damage assessment

3. Spatial join + 4. Creating a new data frame

```
AoI_torn_sf <- st_join(torn_clip_sf, AoI_sf, join = st_within)

AoI_torn_df <- as.data.frame(AoI_torn_sf)

AoI_torn_df$STATE_NAME <- droplevels(AoI_torn_df$STATE_NAME)
newdf<- data.frame(Name = AoI_torn_df$STATE_NAME,
                   Damage= AoI_torn_sf$DAMAGE)
```

Task 1.1: Tornado damage assessment

5. Crosstab analysis

```
> head(newdf)
      Name Damage
1 Arkansas      3
2 Arkansas      3
3 Arkansas      4
4 Arkansas      4
5 Arkansas      5
6 Arkansas      5
```

Count <- table (newdf$Damage, newdf$Name)

Count2 <- xtabs (~ newdf$Damage + newdf$Nam)

複習 1.1：關鍵程式碼

```
index <- us_states_sf$STATE_NAME == "Texas" |
         us_states_sf$STATE_NAME == "New Mexico" |
         us_states_sf$STATE_NAME == "Oklahoma" |
         us_states_sf$STATE_NAME == "Arkansas"

AoI_sf <- us_states_sf[index,]
torn_clip_sf <- torn_sf[AoI_sf,]

AoI_torn_sf <- st_join(torn_clip_sf, AoI_sf, join = st_within)

AoI_torn_df$STATE_NAME <- droplevels(AoI_torn_df$STATE_NAME)
newdf<- data.frame(Name = AoI_torn_df$STATE_NAME,
                   Damage= AoI_torn_sf$DAMAGE)
newdf$Damage <- as.integer(newdf$Damage)

table(newdf$Damage, newdf$Name)
```

10 min 自行練習 #1

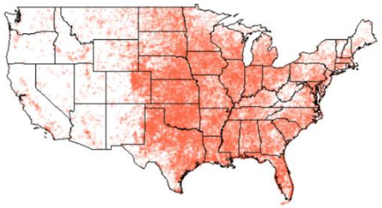
Fast_Food (points)：台北市速食店分布

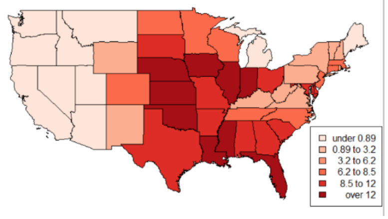
Popn_TWN2 (polygons)：台灣行政區人口數

- 利用 st_join 計算「台北市大安區」的麥當勞與肯德基的店家數

Task 1.2: Creating a tornado density map

(Counting points in polygons)






```
points_sf_joined <- st_join(torn_sf, us_states_sf, join = st_within)

table1 <- table (points_sf_joined $STATE_NAME)
count_sf <-as.data.frame(table1)
colnames(count_sf ) <- c("STATE_NAME","Counts")

us_states_sf <- left_join (us_states_sf, df3) # dplyr package
area1 <- st_area(us_states_sf)                # unit: foot

area1 <- set_units(area1, km^2)               # units package
us_states_sf$AREA1 <- area1
us_states_sf$density <- us_states_sf$Counts / us_states_sf$AREA1

plot(us_states_sf[“density"], breaks = "jenks", nbreaks = 6, pal=brewer.reds(6))
```

dplyr::left_join()

left_join (us_states_sf, df3)

sf format

data.frame

```
> head(us_states_sf)
Simple feature collection with 6 features and 51 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -124.7314 ymin: 40.9943 xmax: -66.96
CRS:             +proj=longlat +ellps=WGS84
       AREA   STATE_NAME STATE_FIPS SUB_REGION STATE_ABBR
1  67286.88   Washington         53    Pacific         WA
2 147236.03      Montana         30        Mtn         MT
3  32161.66        Maine         23      N Eng         ME
4  70810.15 North Dakota         38    W N Cen         ND
5  77193.62 South Dakota         46    W N Cen         SD
6  97799.49      Wyoming         56        Mtn         WY
```

```
> head(df3)
   STATE_NAME Counts
1     Alabama   1266
2      Alaska      0
3     Arizona    196
4    Arkansas   1291
5  California    301
6    Colorado   1631
```

複習 1.2：關鍵程式碼

```
points_sf_joined <- st_join(torn_sf, us_states_sf, join = st_within)

table1 <- table(points_sf_joined$STATE_NAME)

count_sf <- as.data.frame(table1)

colnames(count_sf) <- c("STATE_NAME","Counts")

us_states_sf<- left_join(us_states_sf, count_sf)

area1<-st_area(us_states_sf)                          # unit: foot
area1<-set_units(area1, km^2)                         # units package
us_states_sf$AREA1 <- area1
us_states_sf$Density <- us_states_sf$Counts / us_states_sf$AREA1

plot(us_states_sf["Density"], breaks = "jenks",
     nbreaks = 6, pal=brewer.reds(6))
```

10 min 自行練習 #2

Fast_Food (points)：台北市速食店分布

Popn_TWN2 (polygons)：台灣行政區人口數

- 列出台北市各行政區(名稱)的麥當勞店家總數

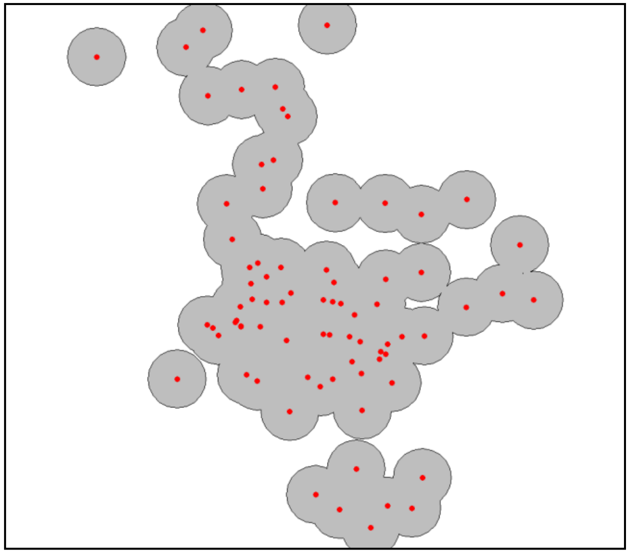
Task 2: Creating service areas

```
st_crs(places_sf)
st_crs(places_sf)<- st_crs(blocks_sf)

block_bg <- tm_shape(blocks_sf) + tm_polygons("grey90")
places_pts <- tm_shape(places_sf) + tm_dots(col = "#FB6A4A", size = 0.5)
block_bg + places_pts

places_buf_sf <- st_buffer (places_sf, dist = 3000)
places_buf <- tm_shape(places_buf_sf) + tm_polygons("yellow")
block_bg + places_buf + places_pts

places_bufU_sf <- st_union(places_buf_sf)

places_bufU <- tm_shape(places_bufU_sf) + tm_polygons("yellow")
block_bg + places_bufU + places_pts
```

10 min 自行練習 #3

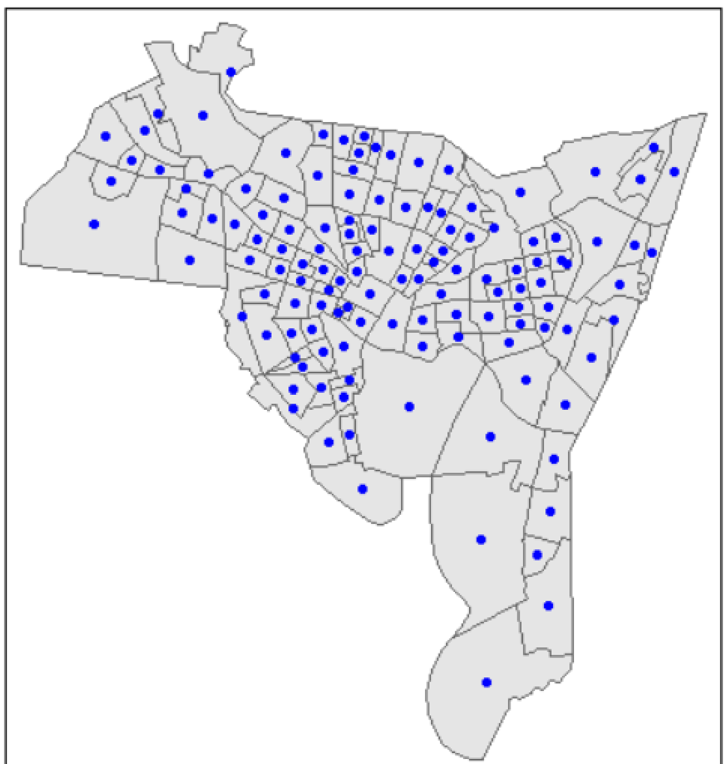
Fast_Food (points)：台北市速食店分布

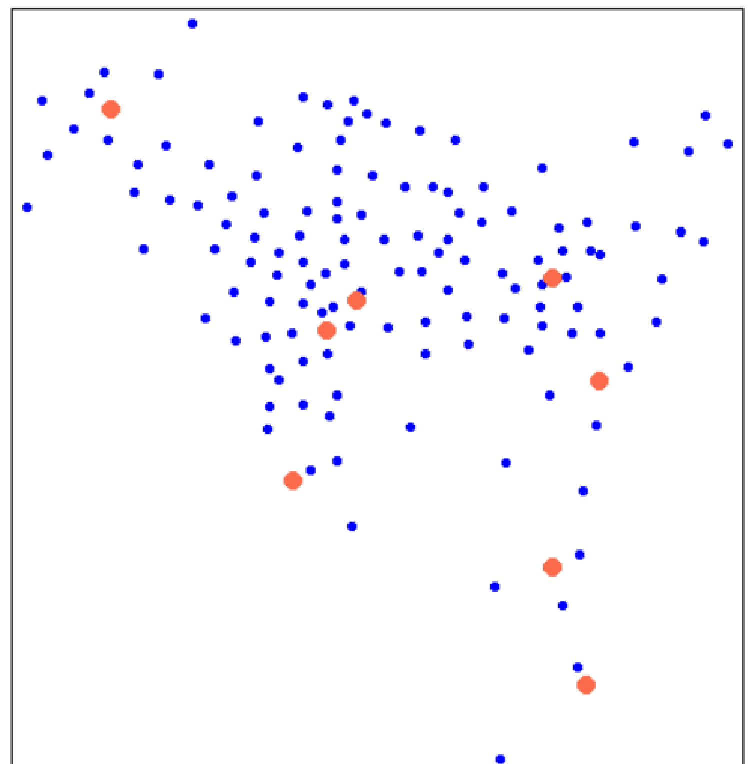
- 利用 st_buffer 建立服務範圍地圖
  - 麥當勞店家位置 + 合併的1 km 服務範圍



Task 3.1: Identifying service areas

```
blocks_center_sf <- st_centroid (blocks_sf)
blocks_center <- tm_shape(blocks_center_sf) + tm_dots(fill = "blue")
blocks_center + places_pts

st_distance (blocks_center_sf[1,], places_sf[2,])
```





Distance matrix

```
distance.matrix        Units: [US_survey_foot] num [1:129, 1:9]
```

places (e.g. hospital)

centroids

```
> distance.matrix <- st_distance(blocks_center_sf, places_sf)
> distance.matrix
Units: [US_survey_foot]
            [,1]      [,2]       [,3]       [,4]      [,5]       [,6]      [,7]       [,8]      [,9]
 [1,] 15368.9406  5448.429 20178.0134 14801.6400 24781.139 21397.9220 29805.861 35214.1327 39097.818
 [2,] 14079.7761  2702.521 20341.0506 13859.7754 24610.072 19585.3929 28865.697 34127.9939 37608.631
 [3,] 15649.5754  1755.740 22616.3985 15673.7962 26737.346 20592.6104 30545.088 35681.6991 38818.176
 [4,] 15440.8047  1298.718 22890.8965 15611.5919 26862.796 20072.1415 30345.024 35399.5893 38351.857
 [5,] 16765.4483  3179.382 24752.1461 17133.6300 28561.889 20832.3188 31607.456 36521.2749 39144.329
 [6,] 20230.5412 28215.109 10033.9304 18353.2333 12308.029 25092.8183 20988.075 25634.8290 32375.471
 [7,] 10144.1070  6767.446 15283.5915  9452.4505 19641.656 16527.2210 24443.114 29861.8132 33875.331
 [8,] 19903.0765 27110.731 10150.5028 17975.3860 13073.630 25127.2629 21861.148 26687.5699 33345.272
 [9,] 10797.2070  8809.865 14126.5185  9703.2707 18796.924 17543.1673 24344.882 29873.2287 34284.147
[10,] 10391.0545  9925.199 13007.6534  9124.2215 17742.623 17255.9078 23519.356 29078.8075 33647.313
[11,] 10693.2968 11074.266 12277.0756  9256.7941 17136.687 17628.4146 23261.811 28853.7237 33612.564
[12,] 16498.5326 23953.083  7200.0902 14557.3282 11017.562 21920.2886 19808.194 24955.9150 31385.416
```

distance.matrix <- set_units (distance.matrix, km)

Using **apply()** function

near_dist<- apply (distance.matrix, 1, mean)

near_dist [1]

```
Units: [km]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]       [,8]       [,9]
 [1,] 4.6844625 1.6606846 6.1502708 4.5115489 7.5533061 6.5220997 9.0848447 10.7332891 11.9170387
 [2,] 4.2915243 0.8237301 6.1999646 4.2244680 7.5011651 5.9696397 8.7982820 10.4022333 11.4631336
 [3,] 4.7700001 0.5351507 6.8934920 4.7773826 8.1495594 6.2766402 9.3101613 10.8758037 11.8318036
 [4,] 4.7063667 0.3958501 6.9771592 4.7584227 8.1877967 6.1180010 9.2491818 10.7898164 11.6896694
 [5,] 5.1101189 0.9690774 7.5444692 5.2223409 8.7056813 6.3497035 9.6339718 11.1317069 11.9312154
 [6,] 6.1662813 8.5999825 3.0583481 5.5940767 3.7514949 7.6483063 6.3971782  7.8135115  9.8680632
 [7,] 3.0919300 2.0627218 4.6584480 2.8811127 5.9867886 5.0375070 7.4502760  9.1018989 10.3252214
 [8,] 6.0664698 8.2633673 3.0938794 5.4789086 3.9848505 7.6588051 6.6632911  8.1343876 10.1636593
 [9,] 3.2909953 2.6852522 4.3057715 2.9575628 5.7293138 5.3471681 7.4203348  9.1053783 10.4498289
[10,] 3.1671998 3.0252066 3.9647407 2.7810683 5.4079623 5.2596112 7.1687139  8.8632382 10.2557214
[11,] 3.2593234 3.3754430 3.7420601 2.8214765 5.2232725 5.3731515 7.0902143  8.7946326 10.2451301
```

檢索與查詢

```
> near_dist
  [1] 6.979727 6.630460 7.046666 6.985807 7.399809 6.544138 5.621767 6.611958 5.699067 5.543718
 [11] 5.547189 5.801250 5.362223 5.235401 5.144283 5.070403 5.327462 6.851863 5.181627 6.907600
 [21] 4.444680 6.132781 6.436491 4.999766 5.132661 5.929551 6.952638 5.497788 4.837457 5.121482
 [31] 4.685944 6.013989 5.524350 4.552128 4.615325 4.557320 4.505207 5.273496 5.864437 5.031492
 [41] 5.101440 4.535758 4.510927 4.320297 4.767403 4.328288 4.066674 4.516341 5.385336 4.698343
 [51] 4.263219 4.937443 4.365297 5.520851 5.492109 4.490339 4.371281 4.073150 4.647792 4.065616
 [61] 4.146566 4.858720 4.351617 3.914714 4.492461 3.945327 4.289878 4.140602 4.500801 3.934279
 [71] 4.091489 4.882253 3.887926 3.778863 3.777889 4.615707 4.106746 3.929693 4.226380 3.654365
 [81] 3.683979 3.923749 3.933471 3.823874 4.473364 4.132668 3.456322 3.868349 3.997306 4.020446
 [91] 4.579319 3.795259 3.659066 3.563309 3.633051 4.270348 3.486571 3.681511 3.510291 3.531218
[101] 3.720148 4.121290 3.839815 3.820938 4.008507 3.453293 3.578081 3.380022 3.669181 3.321075
[111] 3.551878 3.914578 3.833612 3.789689 3.415499 3.479151 3.657364 3.895911 3.913929 3.516730
[121] 3.664736 3.540499 3.763959 3.722855 3.764110 3.867976 4.006165 4.452278 5.089665
>
```

xid<- which.min (near_dist) 到醫院的平均距離最短的 block id = “110”

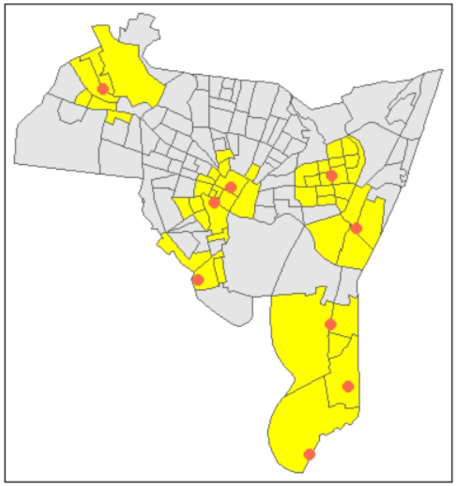
near_dist [xid] block id = “110”的就醫平均距離 = 3.32017

blocks_sf [xid,] 挑選出該 block

blocks_sf$POP1990 [xid] 該 block的人口數(POP1990)

Identifying service areas
挑選最近就醫距離小於1公里的地區

```
min_dist<- apply(distance.matrix, 1, min)
sel_blocks<- min_dist < 1000
sel_sf <- blocks_sf [sel_blocks, ]
sel_map <- tm_shape(sel_sf) + tm_polygons("yellow")
block_bg + sel_map + places_pts
```

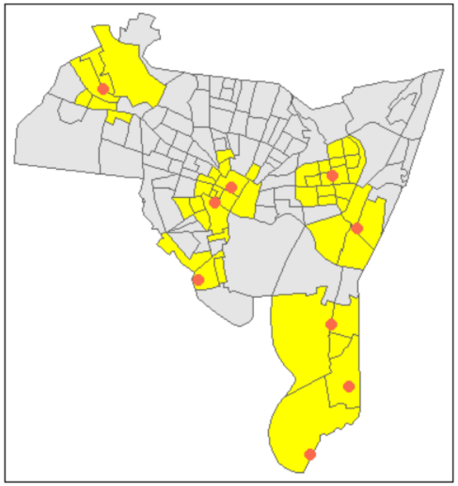


使用另一種函數語法：st_is_within_distance()

```
d1<- set_units(1, km)
sel_blocks = st_is_within_distance (blocks_center_sf, places_sf, dist = d1)
sel_blocks <- lengths(sel_blocks) > 0
sel_sf <- blocks_sf [sel_blocks,]
sel_map <- tm_shape(sel_sf) + tm_polygons("yellow")
block_bg + sel_map + places_pts
```

```
> sel_blocks
Sparse geometry binary predicate list of length 129, where the predicate was `is_within_distance'
first 10 elements:
 1: (empty)
 2: 2
 3: 2
 4: 2
 5: 2
 6: (empty)
 7: (empty)
 8: (empty)
 9: (empty)
 10: (empty)
```

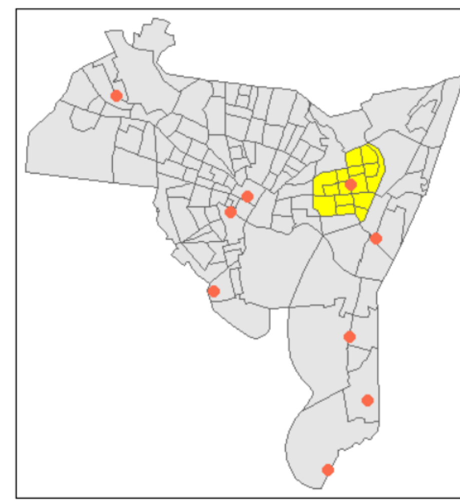
i.d. of place_sf



st_is_within_distance() -- 續

（挑選特定一間醫院的1公里服務地區）

```
sel_blocks = st_is_within_distance (blocks_center_sf, places_sf[3,] , dist = d1)

sel_blocks <- lengths (sel_blocks) > 0

sel_sf <- blocks_sf [sel_blocks,]

sel_map <- tm_shape(sel_sf) + tm_polygons("yellow")

block_bg + sel_map + places_pts
```



```
TOTAL_POP <- sum(sel_sf$POP1990)

TOTAL_WHITE <- sum(sel_sf$POP1990 * (sel_sf$P_WHITE/100) )
```

Task 3.2: Geographical accessibility

```
> eth
        White        Black      Native      Asian      Other
0   17000.001 208499.99907 1399.99957   99.99946 12600.0009
1  267499.999  32099.99909  599.99970 1600.00021  5299.9994
2   32800.000  65999.99976  100.00039  199.99979   500.0000
3   15300.000 114299.99971  699.99987  699.99987  2600.0003
```

```
> eth.access

sel_blocks   White   Black Native  Asian  Other
     FALSE 3526300 2957100  21600 172800 333300
     TRUE  3502000 1763700  18600 141800 617800
```

eth<- as.data.frame(blocks_center_sf[,14:18])

eth<- as.matrix(eth[,1:5])

eth<- apply(eth, 2, function(x) (x*blocks_center_sf$POP1990))

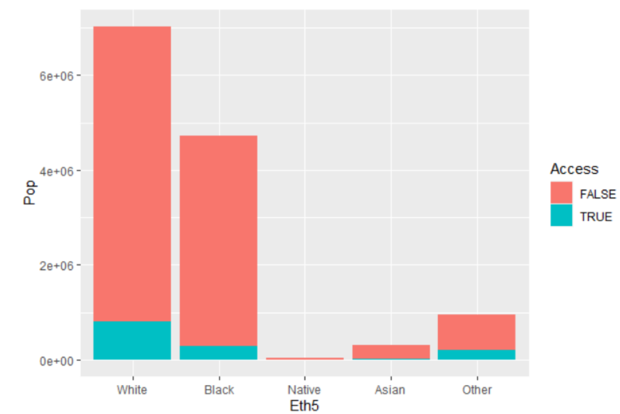
colnames(eth)<- c("White","Black","Native","Asian","Other")

eth.access <- xtabs (eth~sel_blocks)

Task 3.2: Geographical accessibility (cont'd)

```
> eth.access

sel_blocks   White   Black  Native   Asian   Other
     FALSE 3526300 2957100   21600  172800  333300
     TRUE  3502000 1763700   18600  141800  617800
```

```
> eth.access
  sel_blocks   Var2    Freq
1      FALSE  White 3526300
2       TRUE  White 3502000
3      FALSE  Black 2957100
4       TRUE  Black 1763700
5      FALSE Native   21600
```




```
eth.access <- as.data.frame(eth.access)
colnames(eth.access) <- c("Access","Eth5","Pop")

library(ggplot2)
ggplot(eth.access) + aes(fill=Access, y=Pop, x=Eth5) +
  geom_bar(position="stack", stat="identity")
```

Review: R functions for spatial analysis

- Spatial Join: st_join()
- Buffering: st_buffer()
- Merging Spatial Features: st_union()
- Point-in-Polygon: crosstab analysis
- Data Join: left_join()
- Area Calculation: st_area()
- Distance Matrix: st_distance() and
  st_is_within_distance()

本週實習

Fast_Food (points) 台北市速食店位置

Taipei_Vill (polygons) 台北市各里人口數

擷取麥當勞店家位置；

- 以台北市為範圍，麥當勞 1 km為服務範圍內所涵蓋的麥當勞分店數，定義為該家麥當勞店家的連鎖密度，**請問哪一家麥當勞的連鎖密度最高?** 繪製在地圖上，並標示該店家名稱。
- 以台北市為範圍，麥當勞 1 km為服務範圍。以台北市各里中心點是否在涵蓋該麥當勞的服務範圍，作為判斷該麥當勞是否能服務到該里的標準。**請問哪個里可被麥當勞服務的家數最多?** 繪製在地圖上，並標示該里的位置及可及的麥當勞店家。

實習：延伸應用實作

Q1.計算「台北市大安區」麥當勞與肯德基的店家數（利用st_intersection）

Q2.列出台北市各行政區(名稱)的麥當勞店家總數

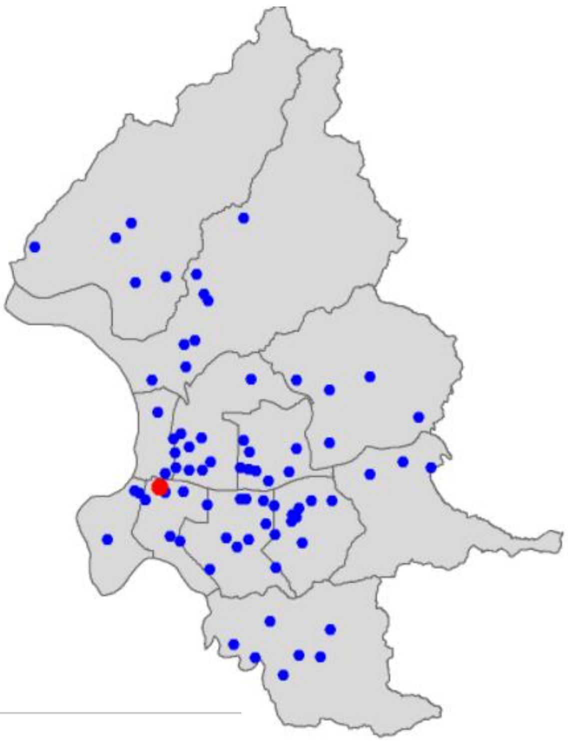
Q3.利用st_buffer 建立服務範圍地圖（麥當勞店家位置+ 合併的1 km 服務範圍）

https://wenlab501.github.io/GEOG2017/

實習參考解答

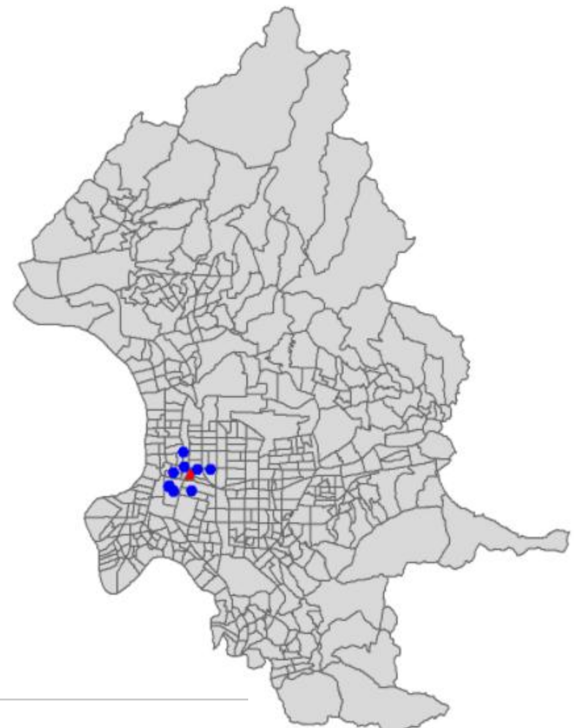
https://wenlab501.github.io/GEOG2017/LAB/Lab2.html

哪一家麥當勞的連鎖密度最高?



```
## [1] "麥當勞(台北館前)"
```

哪個里可被麥當勞服務的家數最多?



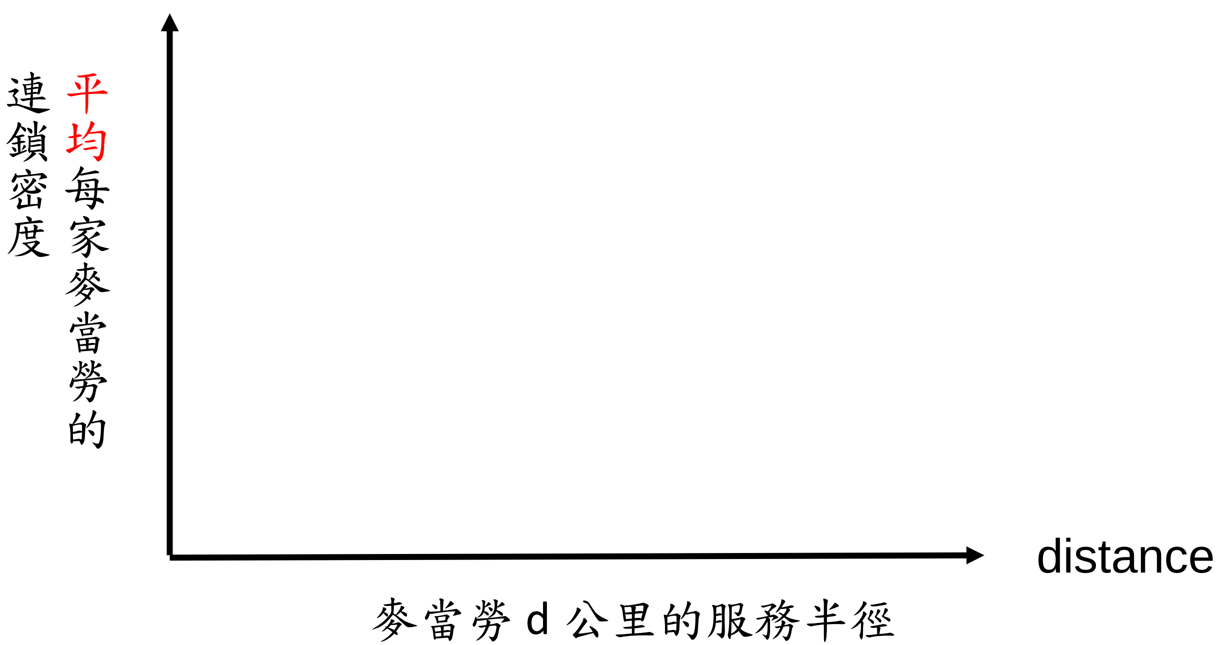
```
## [1] "正得里"
```

本週作業 #Q1

Fast_Food (points) 台北市速食店位置

Taipei_Vill (polygons) 台北市各里人口數

- 將實習所定義麥當勞的連鎖密度，建立 chainstore(d)的自訂函數，可繪製服務半徑(d) vs.麥當勞的關係圖表。



本週作業 #Q2

Fast_Food (points) 台北市速食店位置

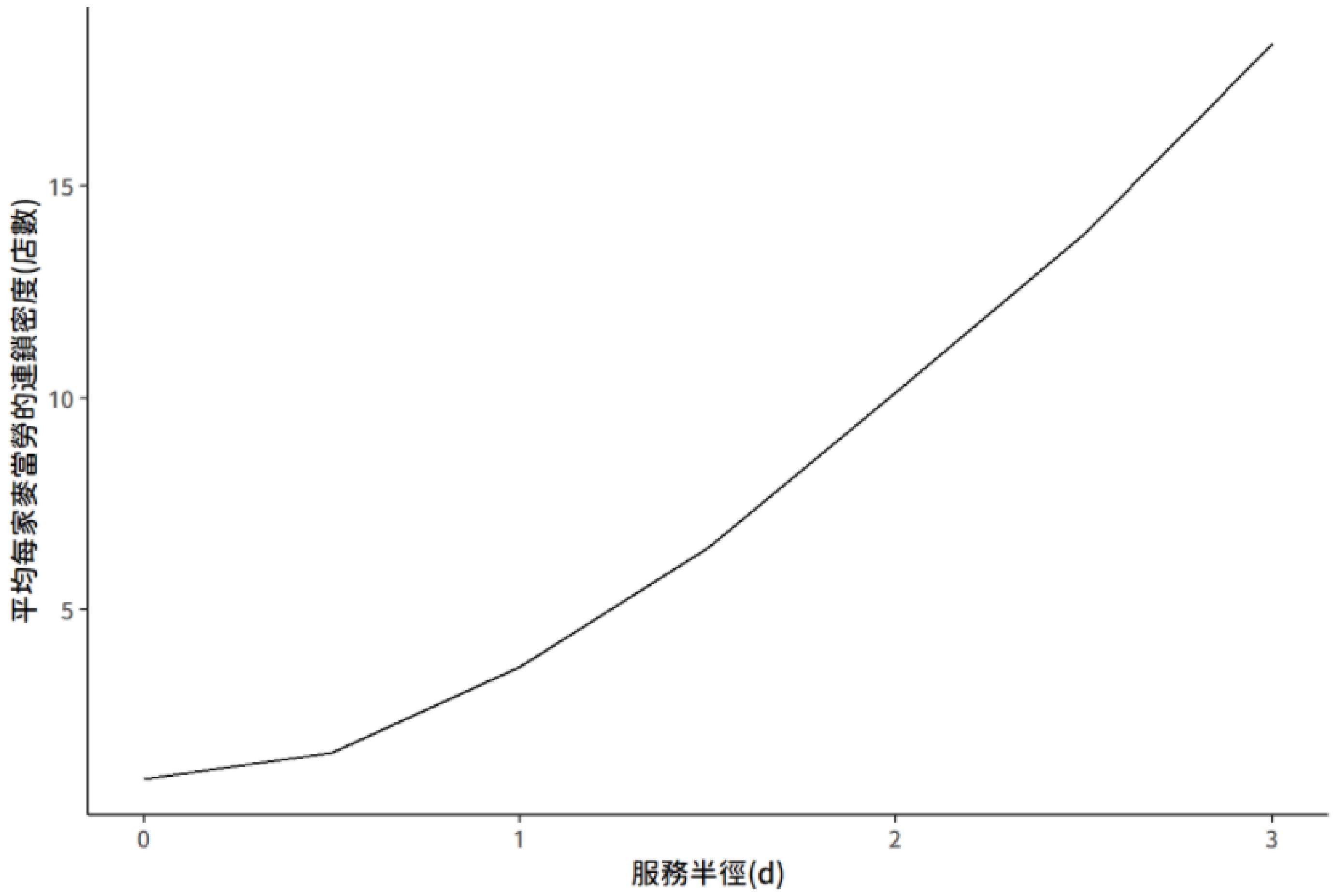
Taipei_Vill (polygons) 台北市村里人口數

- 比較 A區(文山+大安+中正)與 B區(信義+南港+松山) 的麥當勞連鎖密度：

  利用統計檢定方法，評估 A區的平均每家麥當勞連鎖密度是否顯著高於 B區。(服務半徑(d) = 1.5 km)

  (需列出虛無假設與對立假設，並說明檢定的顯著水準)。

參考解答 #1



參考解答 #2

```
# Step 1: Determine the null and alternative hypotheses
# H0:A區差異沒有顯著高於B區, A區的連鎖店家密度在1.5公里處的平均<=B區的
# H1:A區差異顯著高於B區, A區的連鎖店家密度在1.5公里處的平均>B區的
# 單尾檢定
#
# Step 2: Verify necessary data “conditions”, and if met,summarize
```

```
# Step 3: Assuming the null hypothesis is true, find the p-value
p_value=0.729

# Step 4: Decide whether or not the result is statistically significant based on the p-value.
alpha22 = 0.05
if (p_value < alpha22){
  print("拒絕虛無假設")
}else{ print("無法拒絕虛無假設")}
```

```
# Step 5: Decide whether or not the result is statistically significant based on the p-value
# 因為p_value大於0.05，無法拒絕虛無假設，故推斷A區差異沒有顯著高於B區,A區的連鎖店家密度在1.5公里處的平均<=B區的
```