Using R as a GIS

Textbook: Chapter 5

https://ceiba.ntu.edu.tw/1072_Geog2017

授課教師：溫在弘
E-mail: wenthung@ntu.edu.tw

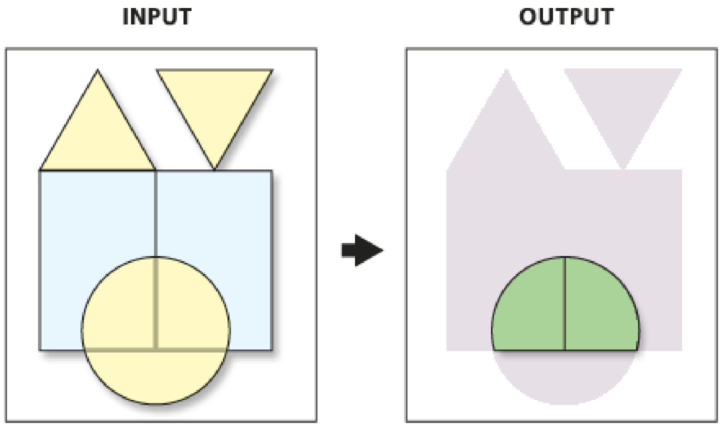
Contents

Introducing **R functions** for spatial analysis

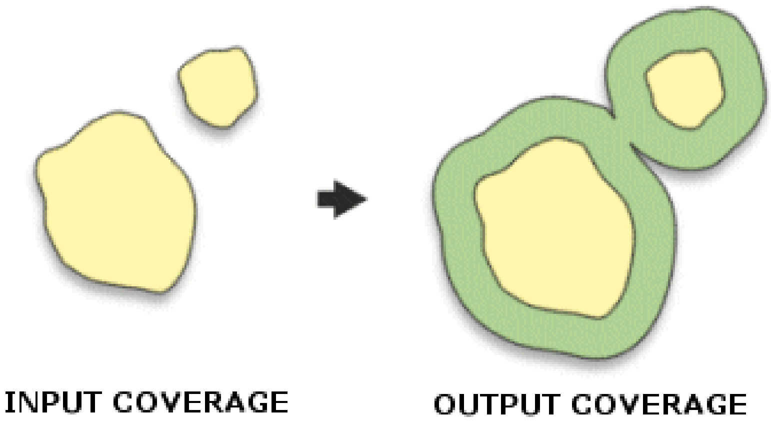
- Spatial Intersection: gIntersection()
- Buffering: gBuffer()
- Merging Spatial Features: gUnaryUnion()
- Point-in-Polygon: poly.counts()
- Data Join: left_join()
- Area Calculation: poly.areas()
- Distance Matrix: gDistance() and gWithinDistance()

Spatial Operational Functions in GIS

Spatial Intersection

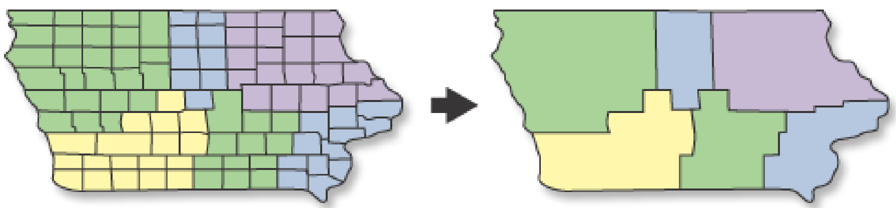


Buffering



Spatial Operational Functions in GIS

Merging Features (dissolve)



INPUT OUTPUT

Data Join

GIS layer (shp) Table (xls)

INPUT

OBJECT ID#	Landuse Code
1	2
2	0
3	1

Join Fields

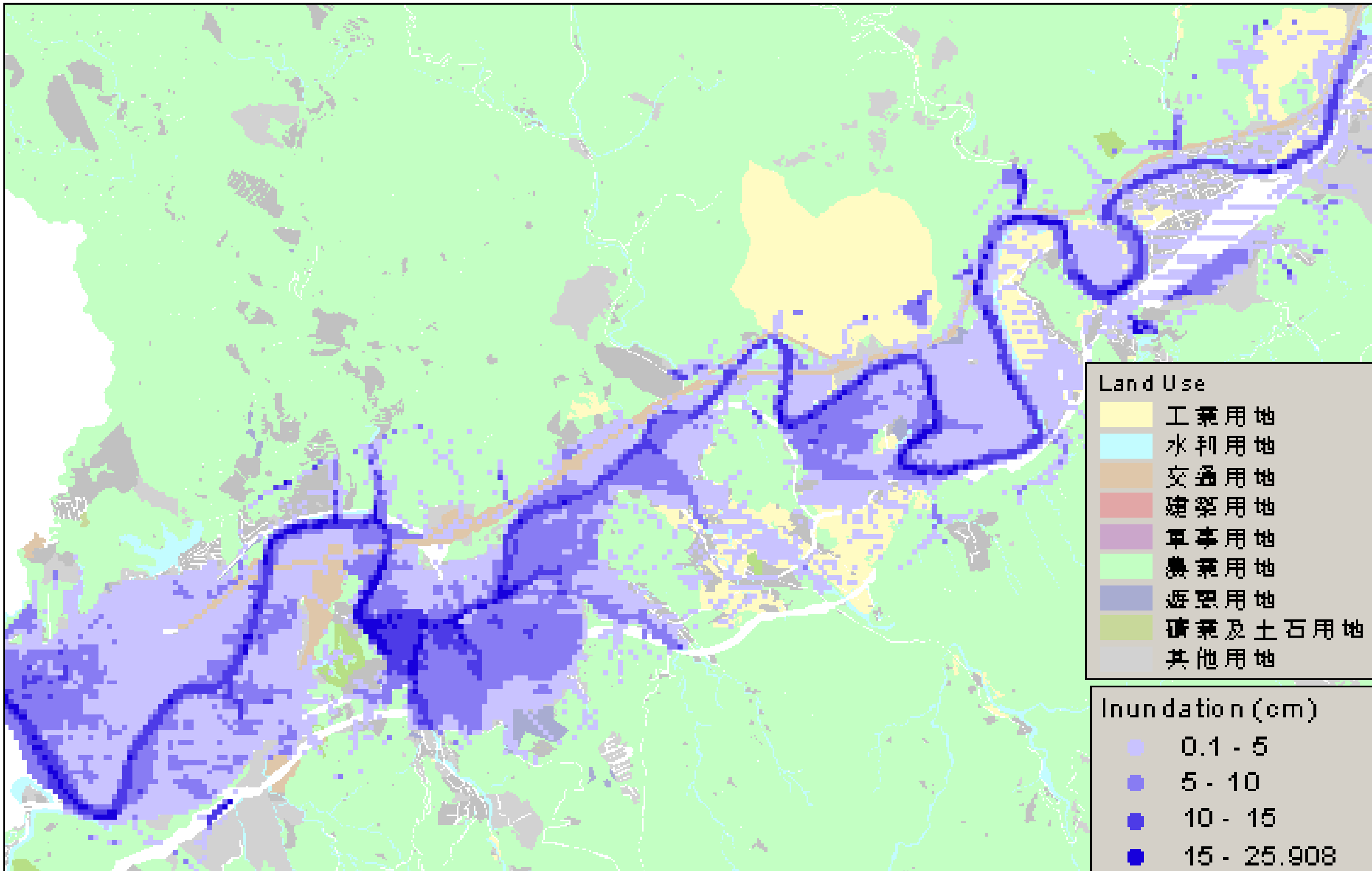
+

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

1. 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: 2-way table

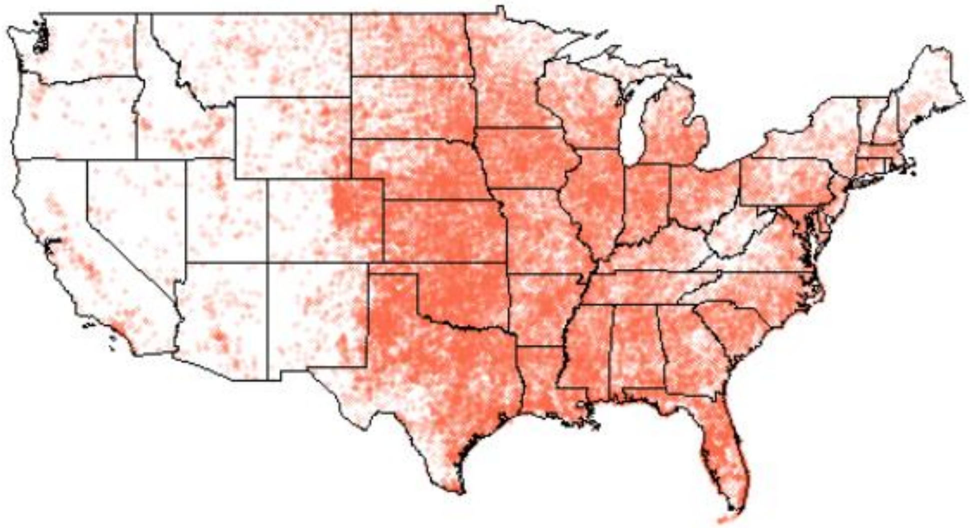
土地利用 / 淹水深度	小於5 cm	5-10 cm	10-15 cm	15-26 cm	淹沒面積 (平方公尺)
工業用地	1,644,814	406,837	6,836	0	**2,058,487**
水利用地	440,763	956,183	841,913	85,017	**2,323,876**
交通用地	793,649	126,423	27,692	0	**947,764**
建築用地	2,323,457	797,133	71,197	5,990	**3,197,776**
軍事用地	77,154	25,713	0	0	**102,867**
農業用地	3,091,893	818,561	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**

Spatial Intersection: R Example

Selecting Layer 1 in the Layer 2

Layer 1: tornados (torn)

Layer 2: US States



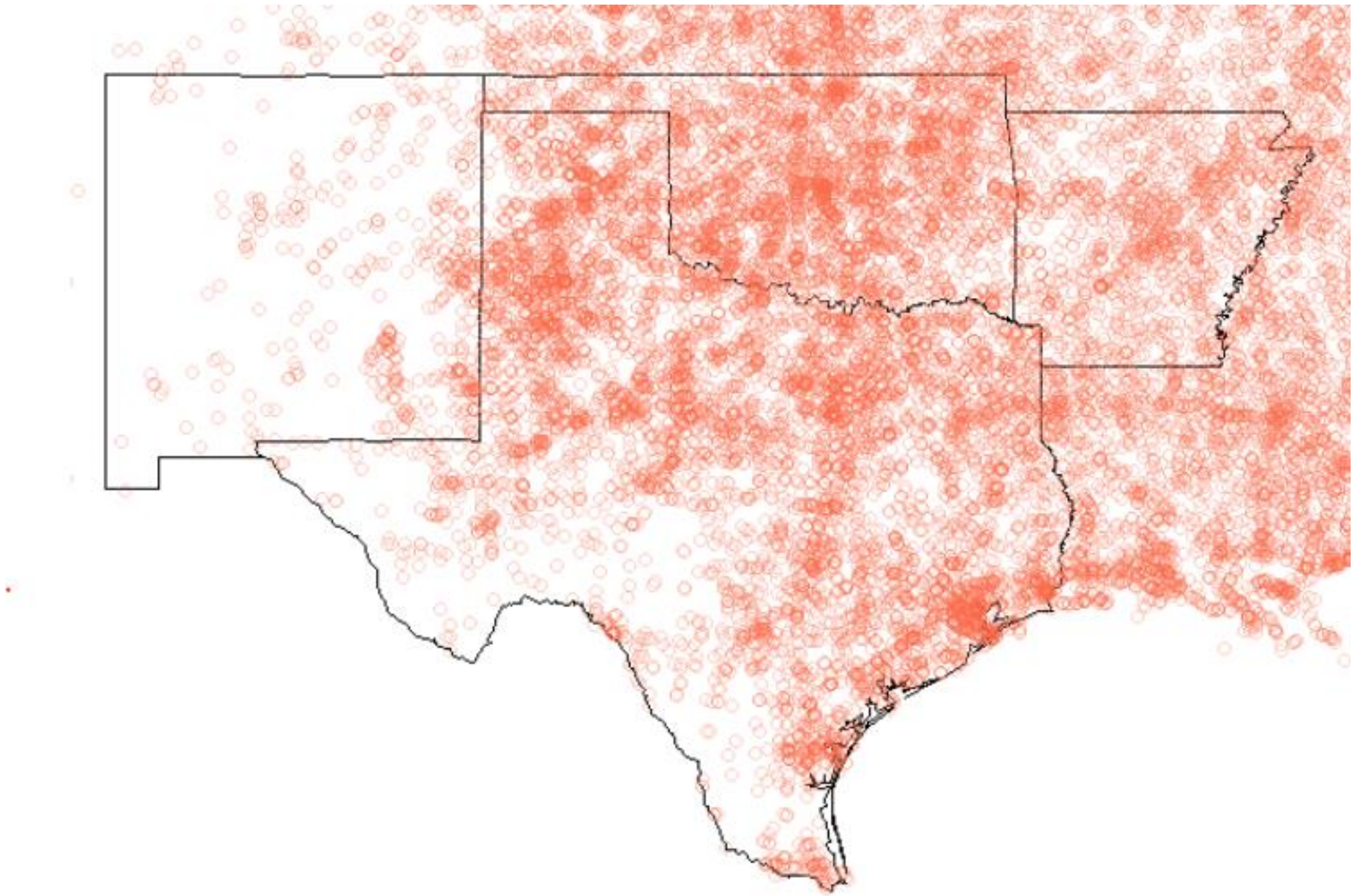
Selecting tornados in the Area of Interest (Texas, New Mexico, Oklahoma, Arkansas)

Spatial Intersection: R Example (2-way table)

new_damage \ new_statename	Arkansas	New Mexico	Oklahoma	Texas
0	276	129	698	1679
1	82	32	228	553
2	44	25	174	433
3	260	144	746	1484
4	424	83	775	1859
5	177	62	304	822
6	22	1	50	189
7	6	1	1	21

Spatial Intersection: R Example
Step 1: extract the Area of Interest (AOI)

```
library(GISTools)
data(tornados)

# set plot parameters and initial plot for map extent
par(mar=c(0,0,0,0))
plot(us_states)
plot(torn, add = T, pch = 1, col = "#FB6A4A4C", cex = 0.4)
plot(us_states, add = T)

head(data.frame(torn))
USstate_attr<- data.frame(us_states)

#AoI
index <- us_states$STATE_NAME == "Texas" | us_states$STATE_NAME == "New Mexico" |
      us_states$STATE_NAME == "Oklahoma" | us_states$STATE_NAME == "Arkansas"

AoI <- us_states[index,]
head(data.frame(AoI))
plot(AoI)
plot(torn, add = T, pch = 1, col = "#FB6A4A4C")
```



Spatial Intersection: R Example
Step 2: intersect using gIntersection()

```
AoI.torn <- gIntersection(AoI, torn, byid = TRUE)
par(mar=c(0,0,0,0))
plot(AoI)
plot(AoI.torn, add = T, pch = 1, col = "#FB6A4A4C")

head(data.frame(AoI.torn))
head(rownames(data.frame(AoI.torn)))
tail(rownames(data.frame(AoI.torn)))

rownames(data.frame(us_states[index,]))
us_states$STATE_NAME[index]
```

AoI.torn	Large SpatialPoints (11784 elements, 901.7 Kb)

```
> AoI.torn
SpatialPoints:
             x     y
37 139  -97.60 35.55
37 140  -95.75 34.85
37 141  -97.02 35.82
37 142  -95.83 36.13
37 143  -99.28 34.88
37 144  -96.40 35.08
37 145  -96.20 34.55
37 146  -99.55 35.25
```

row-names (us.states-index + torn-index)

Spatial Intersection: R Example
Step 3: attach attributes

預期結果

from **us_states** from **torn**

```
> dfnew
       new_tornid new_stateid new_statename new_damage
1             139          37      Oklahoma          6
2             140          37      Oklahoma          4
3             141          37      Oklahoma          3
4             142          37      Oklahoma          4
5             143          37      Oklahoma          3
6             144          37      Oklahoma          5
7             145          37      Oklahoma          5
8             146          37      Oklahoma          4
9             147          37      Oklahoma          4
10            148          37      Oklahoma          3
11            149          37      Oklahoma          5
12            150          37      Oklahoma          0
```

Spatial Intersection: R Example
Step 3: attach attributes

```
tmp <- rownames(data.frame(AoI.torn))
n<-nrow(data.frame(AoI.torn))
new_stateid<-c(1:n); new_tornid<-c(1:n)
new_statename<-c(1:n); new_damage<-c(1:n)
```

```
for (i in 1:n) {
  new_stateid[i]<-substring(tmp[i], 1,2)
  new_tornid[i]<-substring(tmp[i], 4,7)
  new_statename[i]<-as.character(us_states$STATE_NAME[as.numeric(new_stateid[i])])
  new_damage[i]<-as.character(torn$DAMAGE[as.numeric(new_tornid[i])])
  }
```

```
dfnew=cbind(new_tornid, new_stateid, new_statename,new_damage)
names(dfnew) <- c("torn_id","state_id","state_name", "torn_damage")
dfnew=data.frame(dfnew)
```

Spatial Intersection: R Example

Final Step: crosstab analysis

```
AoI.torn_new <- SpatialPointsDataFrame(AoI.torn, data = dfnew)
head(data.frame(AoI.torn_new))
```

```
attach(data.frame(AoI.torn_new))
count<-table(new_damage, new_statename)
count

detach(data.frame(AoI.torn_new))
```

	new_statename			
new_damage	Arkansas	New Mexico	Oklahoma	Texas
0	276	129	698	1679
1	82	32	228	553
2	44	25	174	433
3	260	144	746	1484
4	424	83	775	1859
5	177	62	304	822
6	22	1	50	189
7	6	1	1	21

10 min 隨堂練習 #1

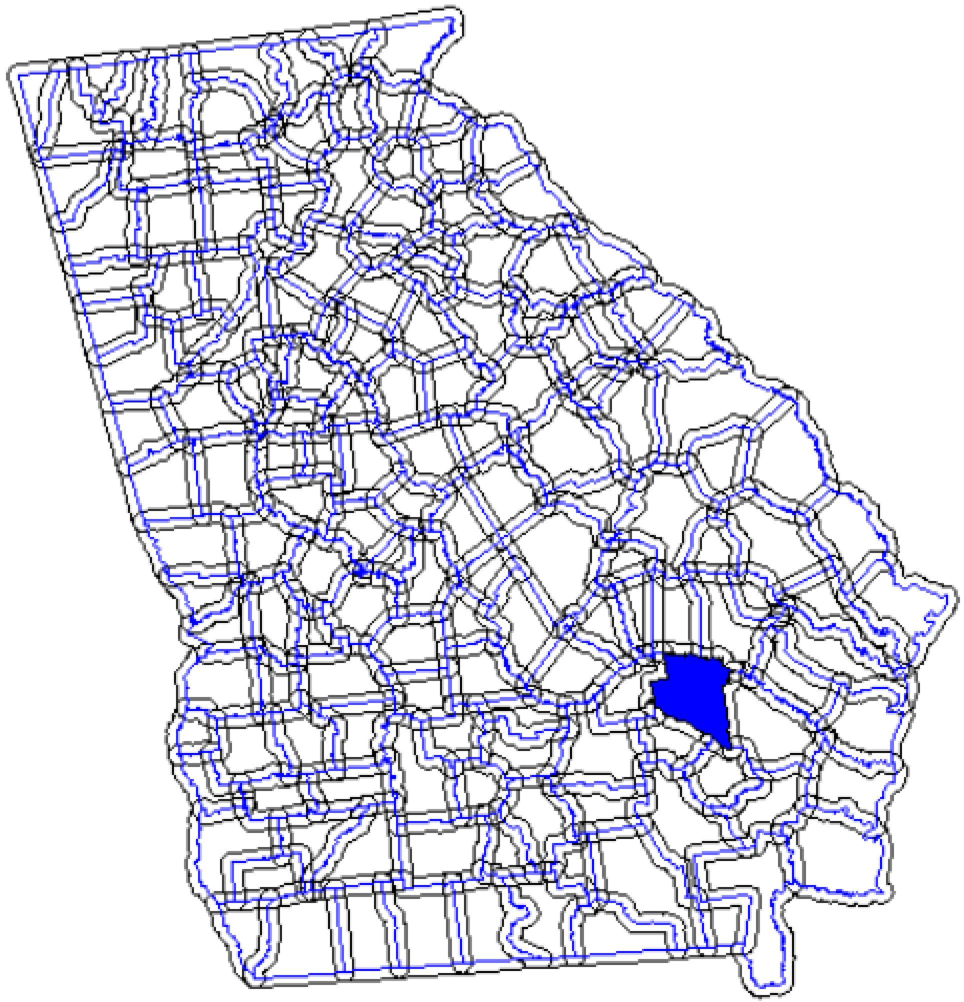
Fast_Food (points)台北市速食店分布

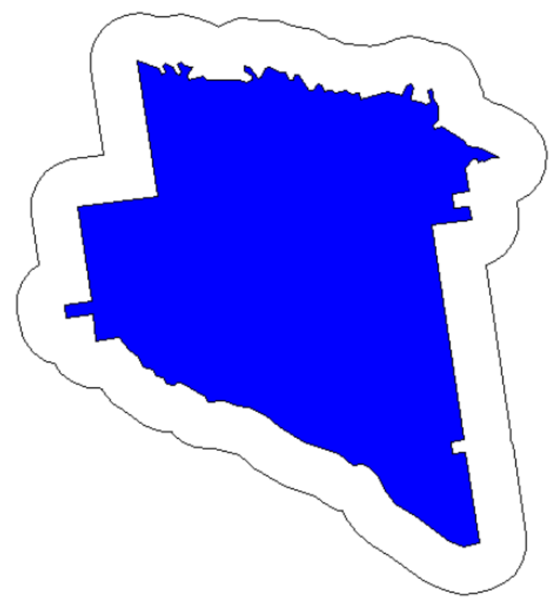
Popn_TWN2 (polygons)台灣行政區人口數

- 利用 gIntersection() 計算台北市大安區的麥當勞與肯德基的店家數

2. **Buffering**: using gBuffer()

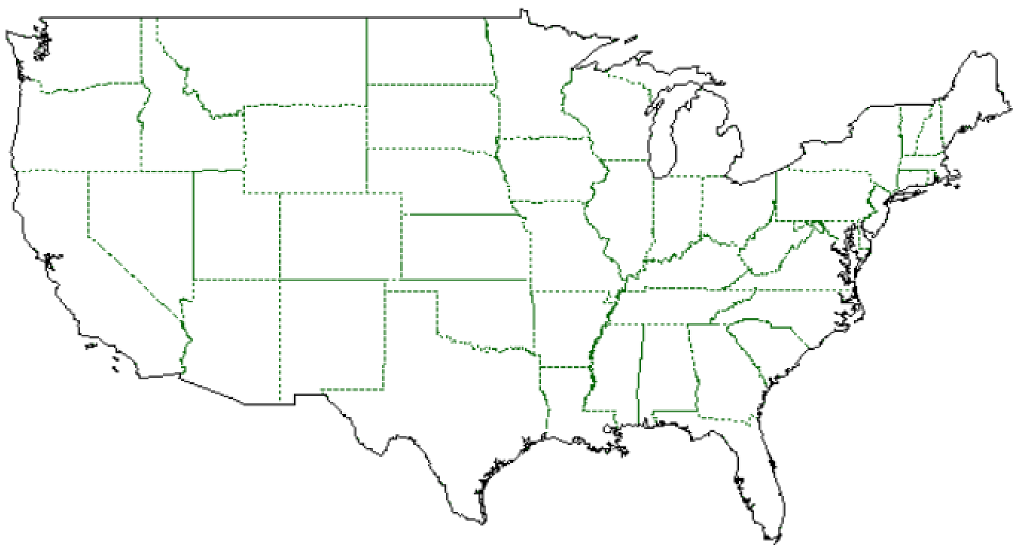
```
# Example 2.1
# select an Area of Interest and apply a buffer
AoI <- us_states2[us_states2$STATE_NAME == "Texas",]
AoI.buf <- gBuffer(AoI, width = 25000)
plot(AoI.buf)
plot(AoI, add = T, border = "blue")

# Example 2.2
data(georgia)
# apply a buffer to each object
buf.t <- gBuffer(georgia2, width = 5000, byid = T, id = georgia2$Name)

plot(buf.t)
plot(georgia2, add = T, border = "blue")

plot(buf.t[1,])
plot(georgia2[1,], add = T, col = "blue")
```





3. **Merging**: using gUnaryUnion()

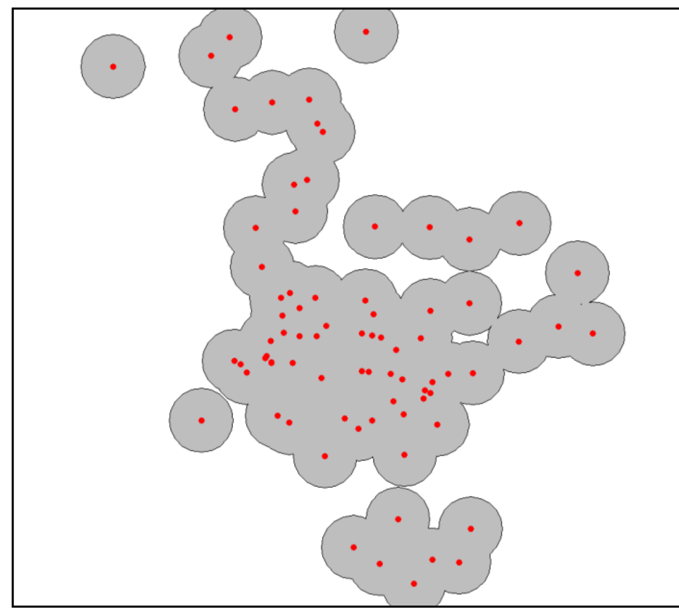
```
AoI.merge <- gUnaryUnion(us_states)
# now plot
par(mar=c(0,0,0,0))
plot(us_states, border = "darkgreen", lty = 3)
plot(AoI.merge, add = T, lwd = 1.5)
```



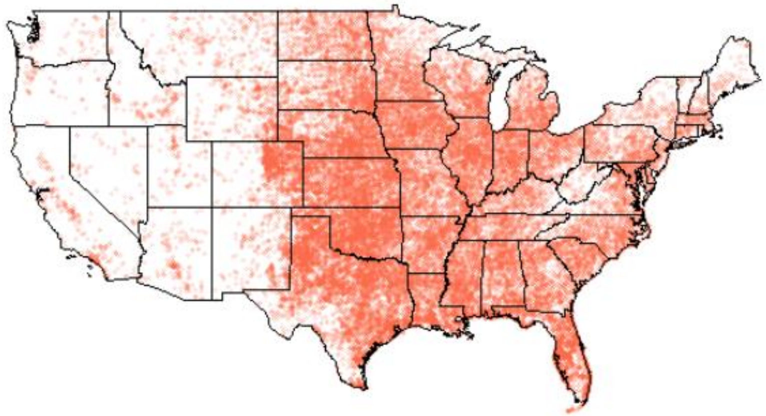
10 min 隨堂練習 #2

Fast_Food (points)台北市速食店分布

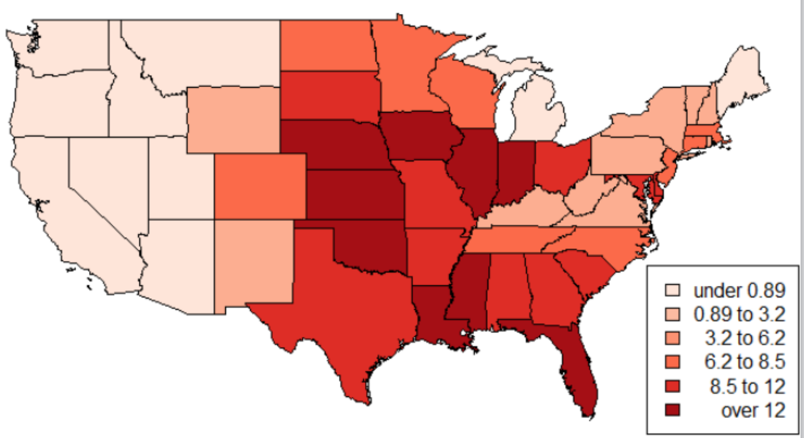
- 利用 gBuffer () 建立服務範圍地圖
  - 麥當勞店家位置 + 合併的1 km 服務範圍



4. Point-in-Polygon, Data Join and Area Calculations: Using poly.counts(); left_join(); poly.areas()



Locations of tornados: point event



Density of tornados in each state

Step 1: Point-in-Polygon using poly.counts()

```
> torn.count <- poly.counts(torn, us_states)
> torn.count
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24 
  79  341   87 1121 1445  549 1015  168   36 1275   86   77 1901  138 2306  325  628   77    7  121 1087   71  106  301 
  25   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48 
 780 1771    2   56  108  243 1631  579 2893  468 1540  196 2976  851  774 7040  477 1266 1399 1110  704 1291 1402 2340 
  50 
 848 
```

us_states index	1	2	3	4	5	6
counts of torn	79	341	87	1121	1445	549

Step 2-1: create a new table

```r
stateid<-names(torn.count)

n<-49
new_stateid<-c(1:n); new_statename<-c(1:n)

for (i in 1:n) {
  new_stateid[i]<-stateid[i]
  new_statename[i]<-as.character(us_states$STATE_NAME[as.numeric(stateid[i])])
}

# create new table
dfnew=data.frame(new_stateid, STATE_NAME=new_statename, torn.count)
```

```
> dfnew
  new_stateid   STATE_NAME torn.count
1           1   Washington         79
2           2      Montana        341
3           3        Maine         87
4           4 North Dakota       1121
5           5 South Dakota       1445
6           6      Wyoming        549
```

Step 2-2: create us_state attribute: using left_join()

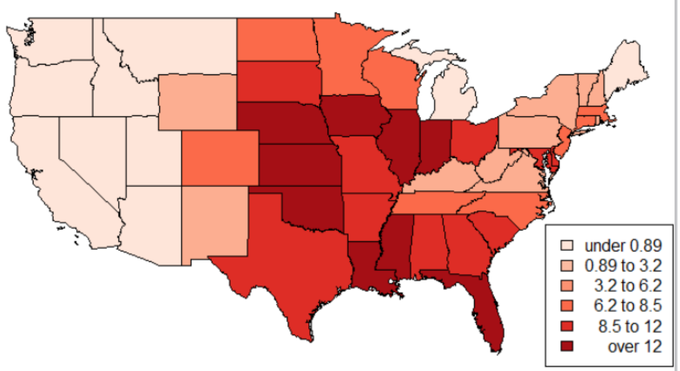
```
# create us_state attribute
library(dplyr)
us_states@data<- left_join(us_states@data, dfnew)
new_us.attr<-us_states@data

for (i in 1:n) {
if( is.na(us_states$torn.count[i]) ) {us_states$torn.count[i]=0}
}
```

```
> dfnew
  new_stateid    STATE_NAME torn.count
1           1    Washington         79
2           2       Montana        341
3           3         Maine         87
4           4  North Dakota       1121
5           5  South Dakota       1445
6           6       Wyoming        549
```

```
> us_states@data
        AREA    STATE_NAME STATE_FIPS SUB_REGION STATE_ABBR  POP1990  POP1997 POP90_SQMI
1  67286.878    Washington         53    Pacific         WA  4866692  5604260         72
2 147236.028       Montana         30        Mtn         MT   799065   888723          5
3  32161.664         Maine         23      N Eng         ME  1227928  1244828         38
4  70810.153  North Dakota         38    W N Cen         ND   638800   644782          9
5  77193.624  South Dakota         46    W N Cen         SD   696004   736549          9
6  97799.492       Wyoming         56        Mtn         WY   453588   484529          5
```

Step 3: calculating area and density: using poly.areas()

```
proj4string(us_states2)
us_states$AREA.KM2<-poly.areas(us_states2) / (1000 * 1000)

attach(us_states@data)
us_states$torn.density<-torn.count*1000/AREA.KM2
vacant.shades = auto.shading(us_states$torn.density,n=6)
choropleth(us_states,us_states$torn.density)
choro.legend(-76.23261,34.20205,vacant.shades)
```



10 min 隨堂練習 #3

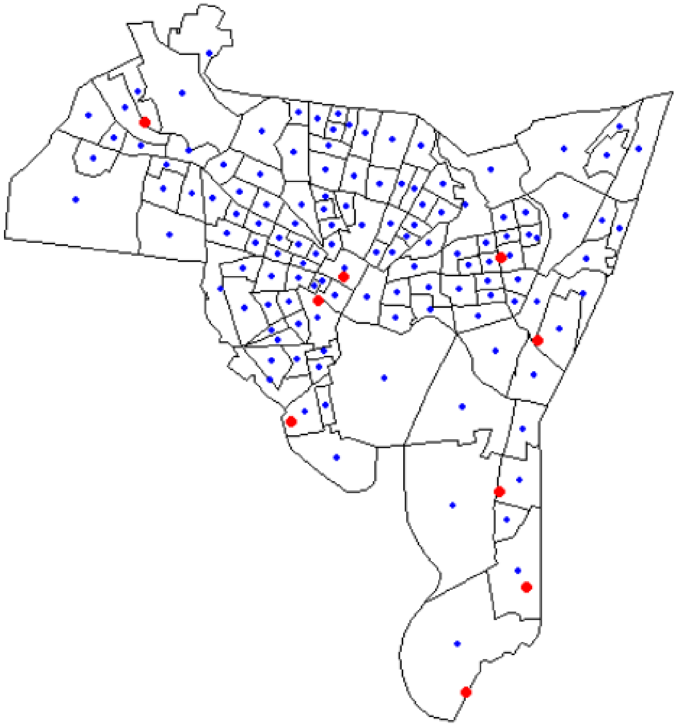
Fast_Food (points)台北市速食店分布

Popn_TWN2 (polygons)台灣行政區人口數

- 利用 poly.counts() 列出台北市各行政區(名稱)的麥當勞店家總數

5. Distance Analysis
using gDistance() and gWithinDistance()

```
data(newhaven)
proj4string(places) <- CRS(proj4string(blocks))
plot(blocks)
points(places, col="red", pch=16)

centroids <- gCentroid(blocks, byid = T, id = rownames(blocks))
points(centroids, col="blue", pch=16, cex=0.6)
```



Distance Matrix: using gDistance()

```
distances <- ft2miles(gDistance(places, centroids, byid = T))
```

places (e.g. hospital)

centroids

```
> distances
```

	1	2	3	4	5	6	7	8	9
0	2.9107842	1.0318995	3.8215934	2.80334091	4.6933975	4.0526367	5.6450495	6.6693433	7.404890
1	2.6666243	0.5118412	3.8524717	2.62495747	4.6609986	3.7093547	5.4669880	6.4636352	7.122847
2	2.9639347	0.3325266	4.2834088	2.96852201	5.0638913	3.9001156	5.7850545	6.7578976	7.351927
3	2.9243948	0.2459693	4.3353971	2.95674088	5.0876509	3.8015420	5.7471636	6.7044677	7.263609
4	3.1752743	0.6021556	4.6879065	3.24500567	5.4094488	3.9455149	5.9862606	6.9169081	7.413699
5	3.8315419	5.3437707	1.9003656	3.47599116	2.3310662	4.7524277	3.9750143	4.8550813	6.131718
6	1.9212324	1.2817133	2.8946196	1.79023684	3.7200105	3.1301555	4.6293776	5.6556464	6.415782
7	3.7695221	5.1346081	1.9224437	3.40442917	2.4760664	4.7589513	4.1403689	5.0544640	6.315392
8	2.0449256	1.6685350	2.6754770	1.83774067	3.5600234	3.3225696	4.6107730	5.6578085	6.493210
9	1.9680028	1.8797725	2.4635707	1.72807225	3.3603453	3.2681644	4.4544234	5.5073499	6.372597
10	2.0252456	2.0973989	2.3252037	1.75318069	3.2455846	3.3387149	4.4056461	5.4647204	6.366016
11	3.1247221	4.5365688	1.3636535	2.75706973	2.0866594	4.1515698	3.7515520	4.7264991	5.944208
12	1.9203705	2.2224716	2.1474532	1.62960662	3.0680617	3.2378476	4.2416206	5.3023044	6.217584
13	1.8759965	2.3824654	1.9753006	1.56336847	2.9031991	3.1925558	4.1076221	5.1705303	6.107048
14	1.9205998	2.6810839	1.7228084	1.57619579	2.6729924	3.2202756	3.9543566	5.0198053	6.000408
15	2.0048806	2.9993767	1.4640825	1.63886693	2.4339735	3.2670088	3.7975253	4.8616261	5.888023

Using **apply()** function: Find the block where the average distance to a hospital is the shortest.

```r
nearest <- apply(distances,1, mean)
```

places (e.g. hospital)

centroids

```
> distances
```

	1	2	3	4	5	6	7	8	9
0	2.9107842	1.0318995	3.8215934	2.80334091	4.6933975	4.0526367	5.6450495	6.6693433	7.404890
1	2.6666243	0.5118412	3.8524717	2.62495747	4.6609986	3.7093547	5.4669880	6.4636352	7.122847
2	2.9639347	0.3325266	4.2834088	2.96852201	5.0638913	3.9001156	5.7850545	6.7578976	7.351927
3	2.9243948	0.2459693	4.3353971	2.95674088	5.0876509	3.8015420	5.7471636	6.7044677	7.263609
4	3.1752743	0.6021556	4.6879065	3.24500567	5.4094488	3.9455149	5.9862606	6.9169081	7.413699
5	3.8315419	5.3437707	1.9003656	3.47599116	2.3310662	4.7524277	3.9750143	4.8550813	6.131718
6	1.9212324	1.2817133	2.8946196	1.79023684	3.7200105	3.1301555	4.6293776	5.6556464	6.415782
7	3.7695221	5.1346081	1.9224437	3.40442917	2.4760664	4.7589513	4.1403689	5.0544640	6.315392
8	2.0449256	1.6685350	2.6754770	1.83774067	3.5600234	3.3225696	4.6107730	5.6578085	6.493210
9	1.9680028	1.8797725	2.4635707	1.72807225	3.3603453	3.2681644	4.4544234	5.5073499	6.372597
10	2.0252456	2.0973989	2.3252037	1.75318069	3.2455846	3.3387149	4.4056461	5.4647204	6.366016
11	3.1247221	4.5365688	1.3636535	2.75706973	2.0866594	4.1515698	3.7515520	4.7264991	5.944208
12	1.9203705	2.2224716	2.1474532	1.62960662	3.0680617	3.2378476	4.2416206	5.3023044	6.217584
13	1.8759965	2.3824654	1.9753006	1.56336847	2.9031991	3.1925558	4.1076221	5.1705303	6.107048
14	1.9205998	2.6810839	1.7228084	1.57619579	2.6729924	3.2202756	3.9543566	5.0198053	6.000408
15	2.0048806	2.9993767	1.4640825	1.63886693	2.4339735	3.2670088	3.7975253	4.8616261	5.888023

Data Query

```
nearest <- apply(distances,1, mean)
```
每個里中心點到醫院的平均距離

```
nearest[1]
nearest <- unname(nearest)

nb <- which.min(nearest)
```
到醫院平均距離最短的里 (index)

```
nearest[nb]
```
該里到醫院的平均距離是 ??

```
blocks@data[nb,]
```
該里的屬性資料

```
> nb <- which.min(nearest)
> nb
[1] 110
> nearest[nb]
[1] 2.063616
>
> blocks@data[nb,]
    NEWH075H_ NEWH075H_I HSE_UNITS OCCUPIED VACANT P_VACANT P_
109       111         26       807      774     33 4.089219
     P_BLACK P_AMERI_ES P_ASIAN_PI  P_OTHER P_UNDER5    P_5_13
109 63.90215   0.238663   0.059666 9.725537 8.412888 15.99045
```

Distance Matrix 2: using gWithinDistance()

```
distances_2 <- gWithinDistance(places, blocks, byid = T, dist = miles2ft(1.2))
```

places

centroids

```
> distances_2
       1     2     3     4     5     6     7     8     9
0  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
1  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
2  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
3  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
4  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
6  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
11 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
12 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
13 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
14 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
15 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
16 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Mapping Serviced Areas of Hospital #1

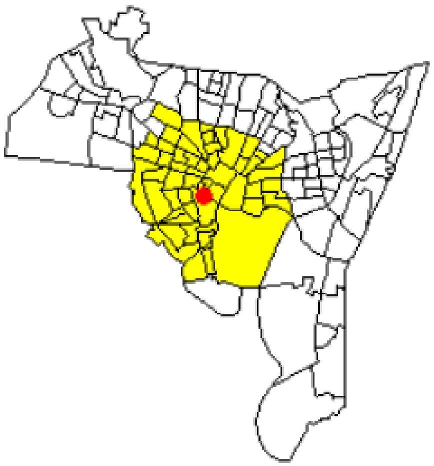
```
Hosp1 <- distances_2[,1]
plot(blocks)
plot(blocks[Hosp1,], col="yellow", add=TRUE)
points(places[1,], col="red", pch=16, cex=1.2)
```

Places (e.g. hosptial)

centroids

```
> distances_2
       1     2     3     4     5     6     7     8     9
0  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
1  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
2  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
3  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
4  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
5  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
6  FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
7  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
8  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
9  FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
10 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
11 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
12 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
13 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
14 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
15 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
16 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Analyzing the Service Areas



```
> length(blocks)
[1] 129
> length(blocks[Hosp1,])
[1] 53
> sum(blocks$POP1990[Hosp1])
[1] 54913
> sum(blocks$POP1990[Hosp1] * blocks$P_WHITE[Hosp1]/100 )
[1] 21223
```

研究區內村里的總數

研究區內hospital 1服務村里的總數

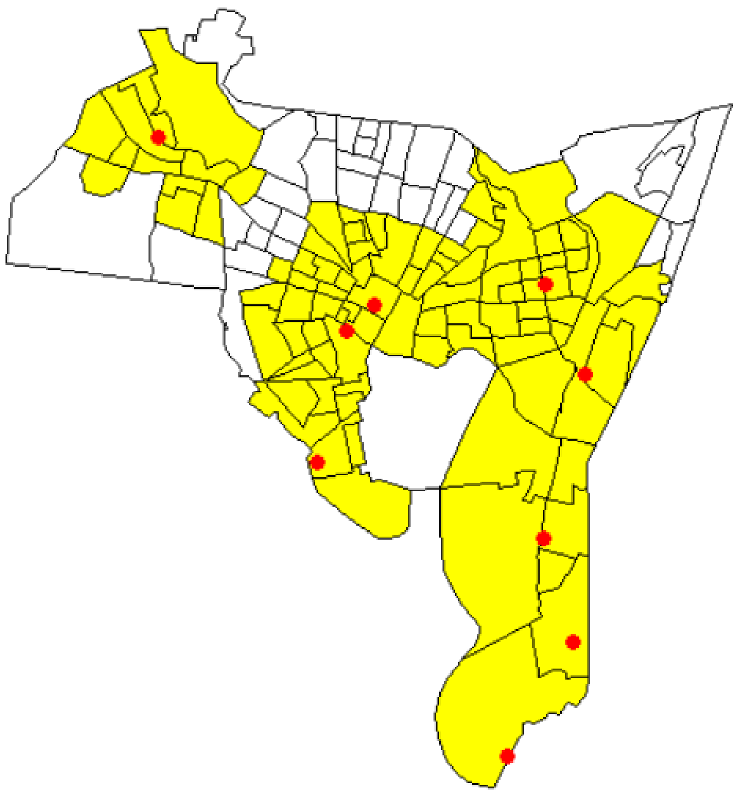
研究區內hospital 1服務村里的總人數

研究區內hospital 1服務村里的白人總數

Example 1: Accessibility Analysis

* spatial selection/query 範例程式 *

```
min.dist <- apply(distances,1, min)
access <- min.dist < 1
plot(blocks)
plot(blocks[access,], col="yellow", add=TRUE)
points(places, col="red", pch=16, cex=1.2)
```



apply() 函數的運用

Example 2: Extract the ethnicity data from the blocks variable

```
ethnicity <- as.matrix(data.frame(blocks[,14:18])/100)
ethnicity <- apply(ethnicity, 2, function(x) (x * blocks$POP1990))
ethnicity <- matrix(as.integer(ethnicity), ncol = 5)
colnames(ethnicity) <- c("White", "Black", "Native American", "Asian", "Other")
```

```
> ethnicity
     White Black Native American Asian Other
[1,]   170  2084              13     0   126
[2,]  2674   320               5    16    52
[3,]   328   659               1     1     4
[4,]   153  1142               6     6    26
[5,]   672   223               3    12     3
[6,]  1156    97               9    11    41
[7,]   690   321               0    16    10
```

Example 2 (cont'd)

```
access <- min.dist < 1
```

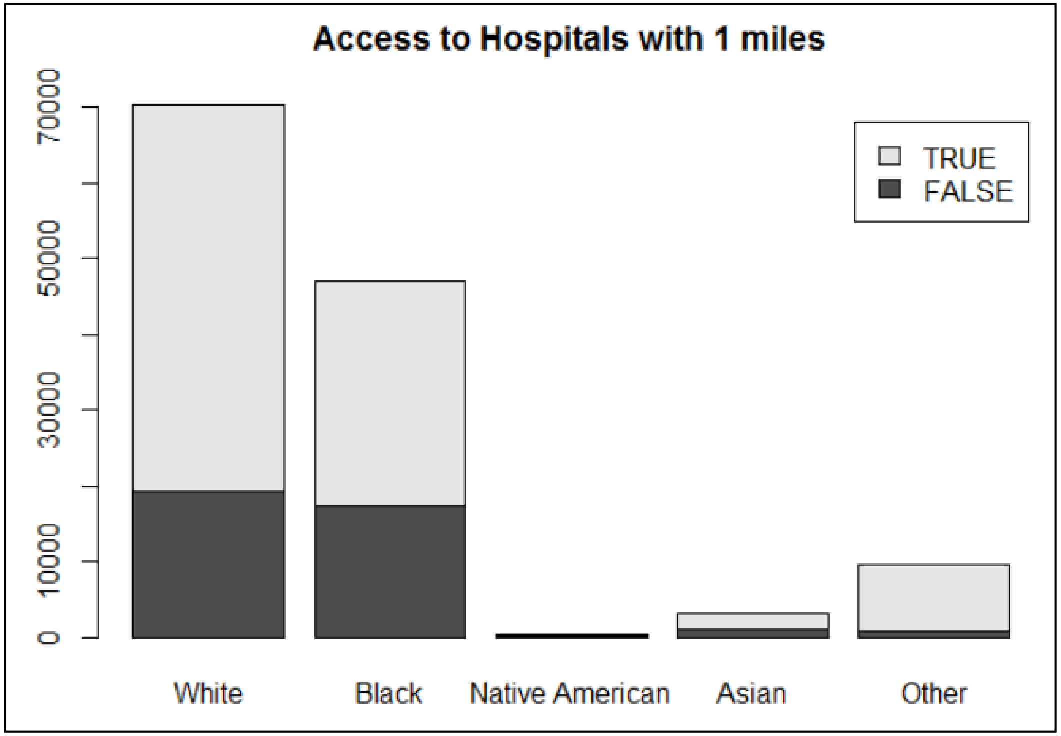
```
access.eth<-xtabs(ethnicity~access)
```

```
#Stacked Bar Plot
barplot(access.eth, names.arg=colnames(ethnicity), legend = rownames(access.eth),
        main="Access to Hospitals with 1 miles")
```

```
> access.eth

access   White Black Native American Asian Other
  FALSE 19161 17266             105  1077   900
  TRUE  51065 29875             247  2014  8545
```

Example 2 (cont'd)

```
access.eth<-xtabs(ethnicity~access)

#Stacked Bar Plot
barplot(access.eth, names.arg=colnames(ethnicity), legend = rownames(access.eth),
        main="Access to Hospitals with 1 miles")
```



Review: R functions for spatial analysis

- Spatial Intersection: gIntersection()
- Buffering: gBuffer()
- Merging Spatial Features: gUnaryUnion()
- Point-in-Polygon: poly.counts()
- Data Join: left_join()
- Area Calculation: poly.areas()
- Distance Matrix: gDistance() and gWithinDistance()

實習

Fast_Food (points) 台北市速食店位置

Taipei_Vill (polygons) 台北市村里人口數

擷取麥當勞店家位置；

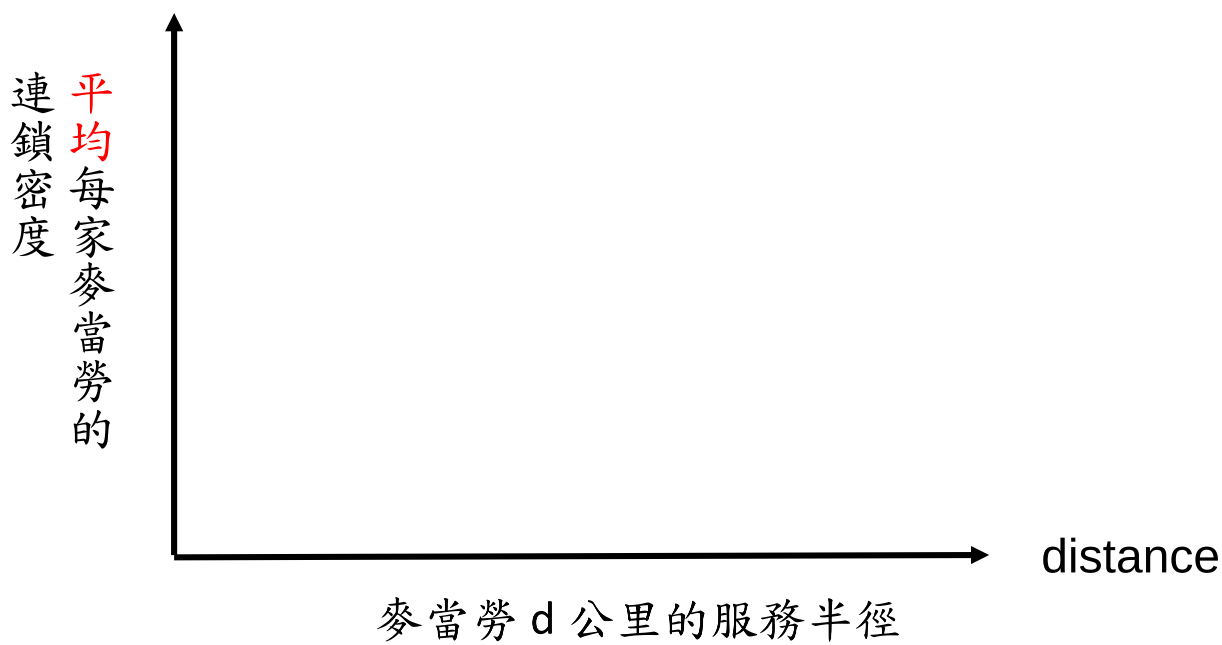
- 以台北市為範圍，麥當勞 1 km為服務範圍內所涵蓋的麥當勞分店數，定義為該家麥當勞店家的連鎖密度，**請問哪一家麥當勞的連鎖密度最高?** 繪製在地圖上，並標示該店家名稱。
- 以台北市為範圍，麥當勞 1 km為服務範圍。以台北市各里中心點是否在涵蓋該麥當勞的服務範圍，作為判斷該麥當勞是否能服務到該里的標準。**請問哪個里可被麥當勞服務的家數最多?** 繪製在地圖上，並標示該里的位置及可及的麥當勞店家。

作業 1

Fast_Food (points) 台北市速食店位置

Taipei_Vill (polygons) 台北市村里人口數

- 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。



作業 2

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)。

補充研讀教材：Reading_Statistical.Significance.pdf
(不需繳交研讀心得，但內容列入期中考範圍)