R實作
GIS空間資料

空間分析 2020.03.23

TA 杜承軒

Lab 3

1. **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。

2. 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。

隨堂練習參考

- **哪一區是大安區／金城鎮？**

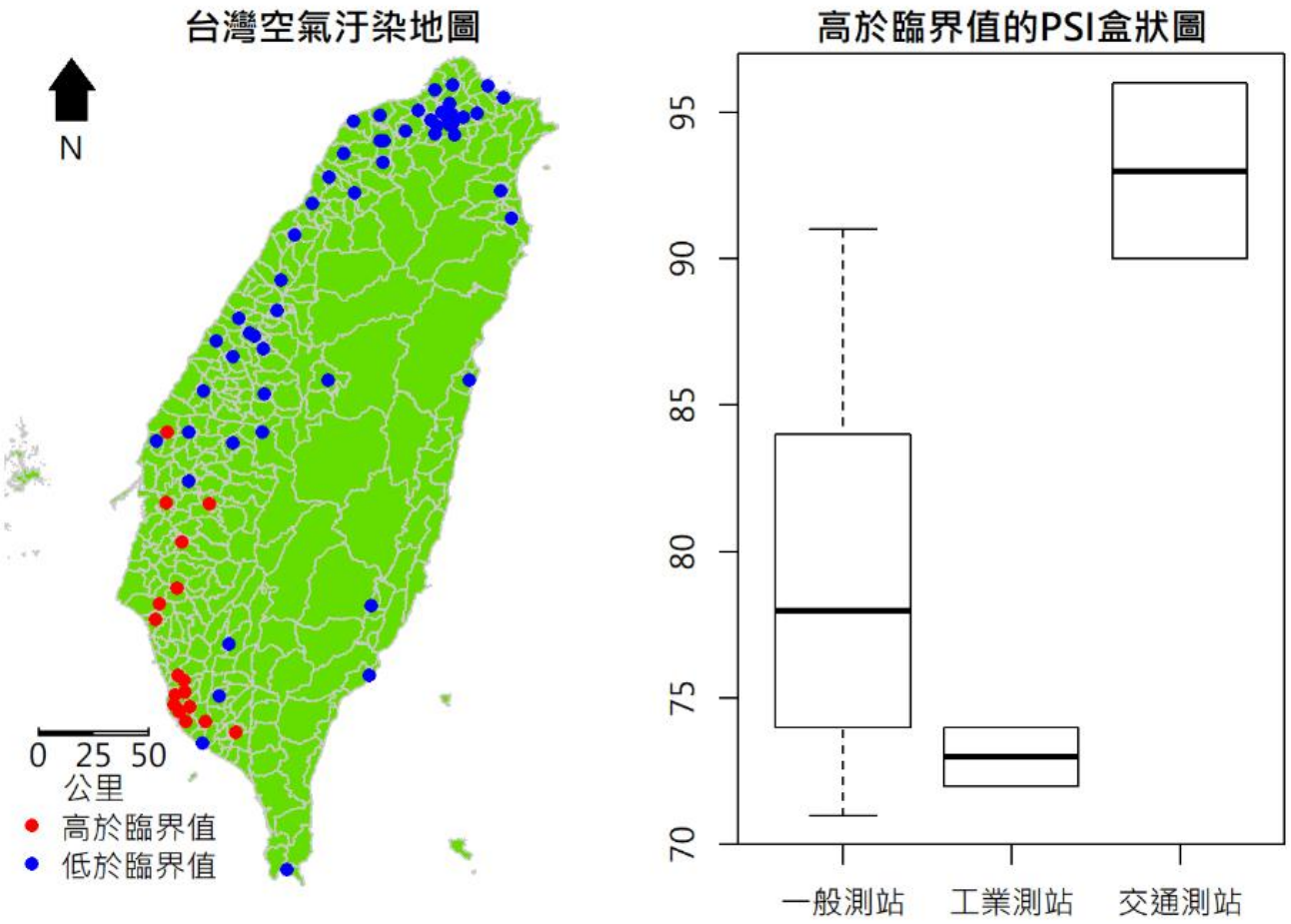
	TOWN_ID	TOWN	COUNTY_ID	COUNTY
0	09007010	南竿鄉	09007	連江縣
1	09007020	北竿鄉	09007	連江縣
2	09007030	莒光鄉	09007	連江縣
3	09007040	東引鄉	09007	連江縣
4	09020010	金城鎮	09020	金門縣
5	09020020	金沙鎮	09020	金門縣
6	09020030	金湖鎮	09020	金門縣

陷阱：**ID (rownames) 從0開始**

```
> Popn.TWN[4,]$TOWN
[1] 東引鄉
> Popn.TWN[5,]$TOWN
[1] 金城鎮
> Popn.TWN['4',]$TOWN
[1] 金城鎮
```

Lab 2

```
Pollution_Map(0.3)
```




```
## [1] 68.12457
```

Lab 2 FAQ

- 篩選三種測站後，還是畫出六種？ → 先轉換成character

```
> high.sub=subset(high.STN,SiteType %in% c("一般測站","工業測站","交通測站"))
> high.sub$SiteType
[1] 一般測站 一般測站 一般測站 一般測站 一般測站 一般測站 一般測站 工業測站 工業測站
[10]一般測站 交通測站 一般測站一般測站 一般測站 一般測站 交通測站 一般測站
Levels: 一般測站 工業測站 公園測站 交通測站 其它測站 背景測站
```

- 圖中新增點：建議用points取代plot(……,add=T)

- 函數匯出多個東西：用list存取

```
ans=list(val=ind,map=map)
return(ans)                ↑ i.e.ggplot
A$map; A$val
```

地圖邊界調整

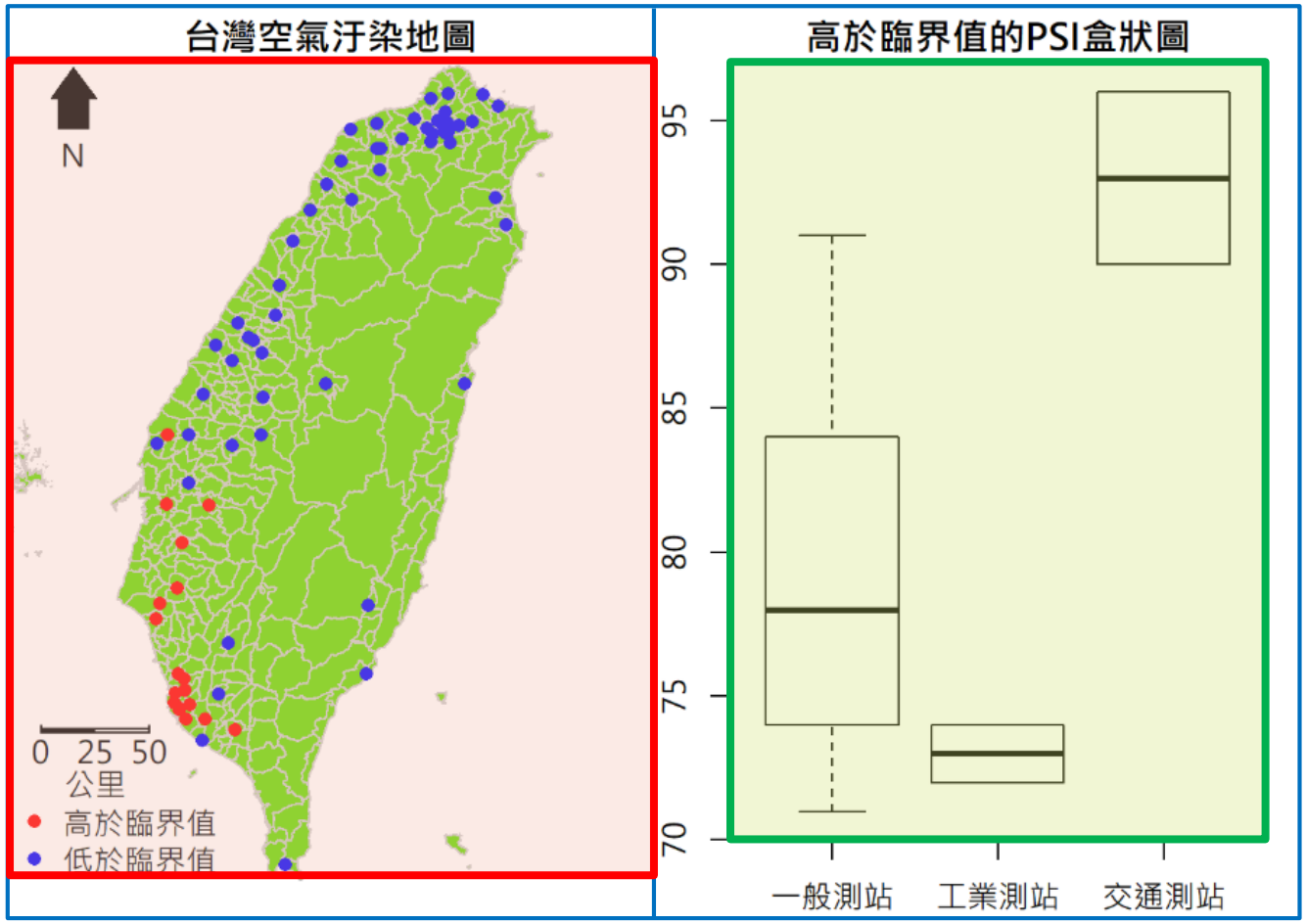
- 地圖放大目標區域

```
#1 xlim/ ylim
plot(Popn.TWN, xlim=c(148466,368986),ylim=c(2415399,2806277))
#2 先畫一層點座標來確認範圍大小
plot(EPA.STN,cex=0) 或 plot(EPA.STN, type="n")
```

圖框
邊界調整

```
par(mfrow=c(1,2),family ="JH")
par(mar = c(1,0,1.5,0)) #邊界：下,左,上,右
plot(EPA.STN,cex=0)
plot(Popn.TWN,add=T)
points(high.STN,col='red')
points(low.STN,col='blue')
map.scale(…)
north.arrow(…)
legend("bottomleft",legend=c("高","低"),
       col=c("red", "blue"))

par(mar = c(2.5,2,1.5,1))
attach(high.STN@data)
boxplot(PSI[SiteType=="一般測站"],…,
    names=c("一般測站","工業測站","交通測站"))
detach(high.STN@data) #解除連結好習慣
par(mfrow=c(1,1)) #還原繪圖區好習慣
```



ggplot 座標轉換

- EPA (points): TWD97/TM2
- TW (polygons): WGS84

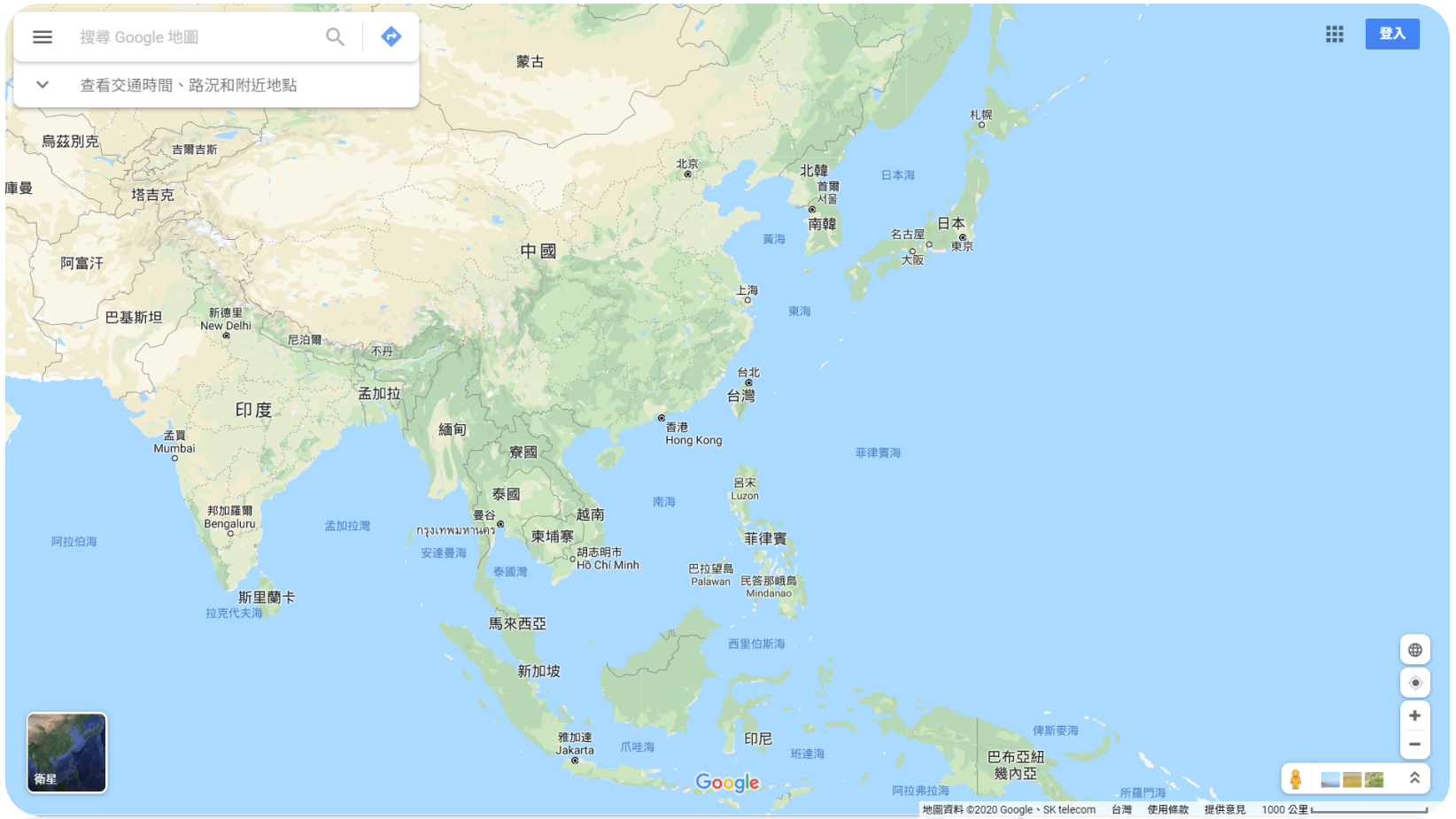
空間資料＋屬性資料

```
#轉換CRS
TW=spTransform(TW, EPA@proj4string)


#畫底圖(多邊形)
#SpatialPolygon畫ggplot需要先fortify
TW.f = fortify(TW,region = 'TOWN')
map = ggplot()+geom_polygon(TW.f,aes(x=long,y=lat,group=group))
#固定的寫法，long,lat這兩欄由fortify生成，真正代表是的是你設定投影座標的x和y


#畫測站(點)
EPA$new_X=EPA@coords[,1]; EPA$new_Y=EPA@coords[,2] #將真正的座標放進屬性表
map + geom_point(data=EPA@data, aes(x=new_X, y=new_Y))
```

osm
座標轉換



地理座標系統 vs 投影座標系統

- **TWD97/TM2** *EPSG:3826* (250000, 2765778)
- **WGS84** *EPSG:4326* (121,25)
- **WGS84/Pseudo-Mercator** *EPSG:3857* (13469658,2875745) ← osm()是這個

osm 座標轉換

```
map=openmap(c(21.8,120),c(25.5,122),zoom = 7,type="osm")
```

*經緯度(先緯度,再經度) P.S. 21.8/25.5可交換順序、120/122可交換順序

plot(map)

1° 轉成 WGS84 → 求經緯度
```
TW = spTransform(TW,CRS("+init=epsg:4326"))
```

2° 找到經緯度邊界
```
ul = c(TW@bbox[2,2],TW@bbox[1,1])
lr = c(TW@bbox[2,1],TW@bbox[1,2])
```

3° 填入openmap()
```
map = openmap(ul, lr, 7, "osm")
plot(map) *此時圖上的座標是 WGS84/Pseudo-Mercator
```

4° 轉成 WGS84/P-M → 疊圖
```
plot(spTransform(TW,osm()), add=T)
```

- 中間兩步驟可以直接合併為：

```
map = openmap(rev(TW@bbox[,1]), rev(TW@bbox[,2]), 7, "osm")
```

- openproj()：更改底圖的座標系統

```
map = openproj(map,CRS("+init=epsg:4326"))
```

```
> TW@bbox
     min  max
x    120  122
y   21.8 25.5
```

apply

- lapply(LIST,FUN)
- mapply(FUN,arg1,arg2,…..)
- apply (X, MARGIN, FUN)

MARGIN:
1 by row
2 by column

```
> M
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> apply(M,1,sum)
[1] 16 20
> apply(M,2,sum)
[1]  3  7 11 15
```

input format	function	output format
list data.frame	lapply	list
parameters	mapply	vector matrix
list data.frame array	apply	vector matrix

unlist() → vector

polygons

poly.areas(polygon)

poly.counts(points, polygons) → 可以轉成vector

left_join　　left_join(x, y, by = c("name.x" = "name.y"))

1. 確認兩欄的格式要一樣（事先型別轉換）
2. 配對的兩欄名稱不同 → 用by連接

```
> x
  id name
1  1   甲
2  2   乙
3  3   丙
> y
  id2 name2
1   1     A
2   2     B
3   4     D
```

```
> left_join(x,y,by=c('id'='id2'))
  id name name2
1  1   甲     A
2  2   乙     B
3  3   丙  <NA>
> full_join(x,y,by=c('id'='id2'))
  id name name2
1  1   甲     A
2  2   乙     B
3  3   丙  <NA>
4  4 <NA>     D
```

left_join：
- x 被保留
- 只在x→NA值
- 只在y→消失

full_join：
- x, y 都被保留
- 只在其中一者→NA值

GISTools

Buffer

Dissolve

Centroid

Distance

```
gBuffer(sp, width, byid=T)
gUnaryUnion(sp, id=group)
gCentroid(sp, byid=T)
gDistance(sp, sp2=NULL, byid=T)
gWithinDistance(sp, sp2=NULL, dist, byid=T)
gWithin(pts,poly,byid=T)
gContain(poly,pts,byid=T)
```

A.B.dist=**gDistance**(B,A,byid = T)

B

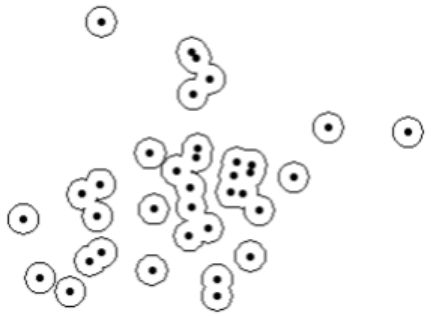
A	0	1	2
1	128517.49	126907.54	120264.35
2	123956.87	122449.94	116278.79
3	29738.99	38177.79	46334.56
4	243555.73	239797.71	224000.15
5	47813.73	55654.62	63906.59

gDistance

	0	1	2
1	FALSE	FALSE	FALSE
2	FALSE	FALSE	FALSE
3	TRUE	TRUE	TRUE
4	FALSE	FALSE	FALSE
5	TRUE	TRUE	TRUE

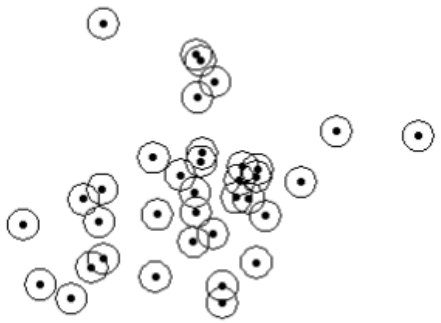
gWithinDistance

gBuffer(sp,width=1000)



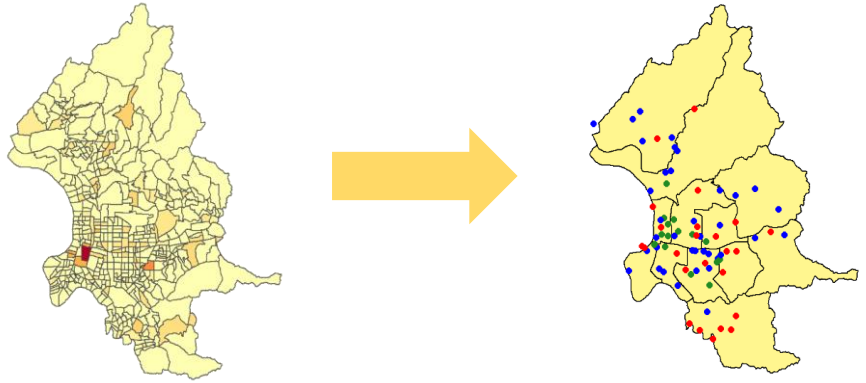
n points→only 1 polygons

gBuffer(sp,width=1000,byid=T)



n points→n polygons

Town = **gUnaryUnion**(**Vill**,Vill$TOWN)



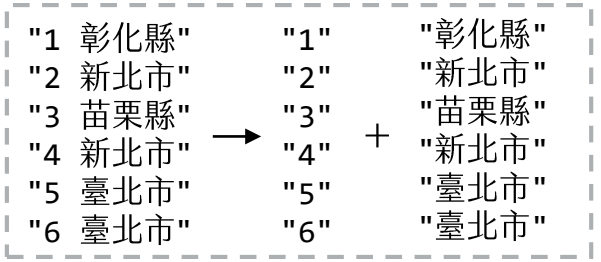
GISTools

Intersect
點×面→點

```
gIntersection(sp1,sp2,byid=T)

Inter=gIntersection(pts,poly,byid=T)
I.name=rownames(data.frame(Inter))
```

```
> I.name
"1 彰化縣"  "2 新北市"
"3 苗栗縣"  "4 新北市"
"5 臺北市"  "6 臺北市"
```



1° 將字串拆開：strsplit()

```
I.split=strsplit(I.name," ")
```

得到N個list，每個list都有兩個id

```
> strsplit(I.name," ")
[[1]]
[1] "1"      "彰化縣"

[[2]]
[1] "2"      "新北市"
```

2° 分別存入X_id、Y_id欄位

way 1.

```
get.X=lapply(I.split, function(x) x[1])
X.id=unlist(get.X)
```

way 2.

```
Y.id=c()
for(i in I.split){Y.id=c(Y.id, i[2])}
```

3° 建立XY.Inter的資料表*(optional.)*

```
I.data = data.frame(X.id, Y.id, row.names=I.name)
Inter = SpatialPointsDataFrame(Inter, I.data , match.ID = F)
```

*兩行斜線選一個寫

※ 善用xtabs()、left_join()來處理資料

Intersect
面×面→面

```
gIntersection(sp1,sp2,byid=T)

Inter=gIntersection(poly1,poly2,byid=T)
I.name=names(Inter)
```

1° 將字串拆開：strsplit()

```
I.split=strsplit(I.name," ")
```

```
> strsplit(I.name," ")
[[1]]
[1] "1"      "彰化縣"

[[2]]
[1] "2"      "新北市"
```

```
> unlist(I.split)
  [1] "1"  "彰化縣" "2" "新北市"
      "3"  "苗栗縣"
```

2° 分別存入X_id、Y_id欄位

way 1.

```
get.X=lapply(I.split, function(x) x[1])
X.id=unlist(get.X)
```

way 2.

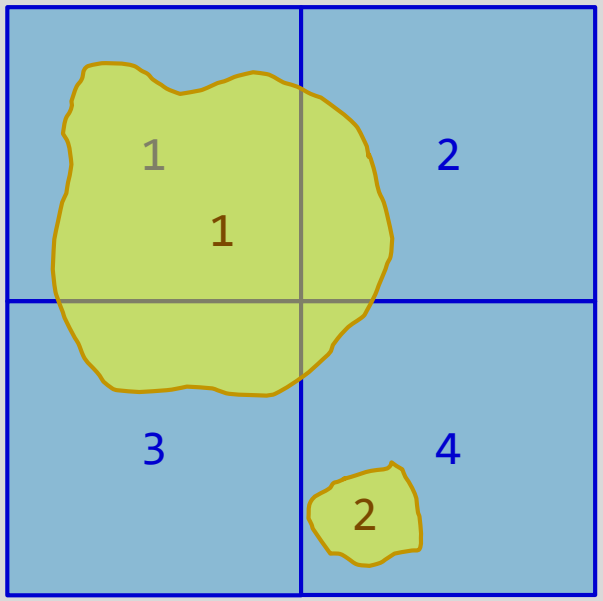
```
X.id=Y.id=c()
for(i in I.split){X.id=c(X.id, i[1]);Y.id=c(Y.id, i[2])}
```

way 3.

```
ans=unlist(I.split)
X.id=Y.id=c()
x=1
for (i in seq(1,length(ans),2)) {
 X.id[x] = ans[i]
 Y.id[x] = ans[i+1]
 x=x+1
}
```

```
N=length(ans)
for (i in 1:N) {
 X.id[i] = ans[2*i-1]
 Y.id[i] = ans[2*i] }
```

Intersect




```
XY.Inter = gIntersection(X,Y,byid=T)
```

將截切後的名字分隔，分別儲存各自的**id**

```
XY.name=strsplit(names(XY.Inter), " ")

X.id=unlist(lapply(XY.name,function(x) x[1]))
Y.id=unlist(lapply(XY.name,function(x) x[2]))
```

XY.Inter	
"1"	"1"
"2"	"1"
"3"	"1"
"4"	"1"
"4"	"2"

X.id	Y.id
1	1
2	1
3	1
4	1
4	2

建立XY.Inter的資料表

兩行斜線選一個寫

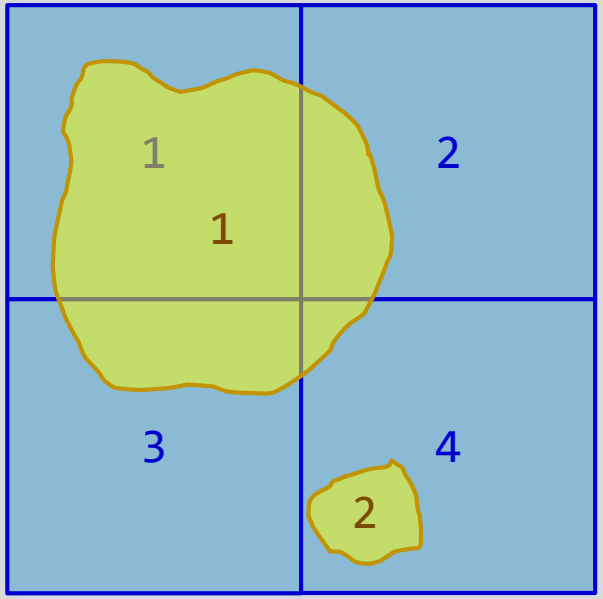
```
XY.data = data.frame( X.id, Y.id, row.names=names(XY.Inter) )

XY.Inter = SpatialPolygonsDataFrame(XY.Inter, XY.data, match.ID = F)
```

	X.id	Y.id
1 1	1	1
2 1	2	1
3 1	3	1
4 1	4	1
4 2	4	2

善用xtabs()、left_join()

Intersect



1. 透過新圖形的特徵做運算

   i.e.各個截切小區的面積：poly.areas(XY.Inter)

2. 透過原本的表格做運算

   i.e.抓取原本的人口：X$pop[X.id] **※注意ID是否從1開始**

   ※ **從0開始**

   X$pop[X.id+1]

   ※ **id不是數字格式**

   或資料不連續

   X[X.id,]$pop

id可先轉換成數字方便後續使用
（如果**id**是數字的話）

```
X.id=as.numeric(X.id)
Y.id=as.numeric(Y.id)
```