Spatial Data Handling

Textbook: Chapter 3

https://ceiba.ntu.edu.tw/1092Geog2017_

授課教師：溫在弘
E-mail: wenthung@ntu.edu.tw

Using R as GIS

- week 1: 3/8 (spatial data handling)
  - GIS data format in R
  - Mapping + attribute query + plots
- week 2: 3/15 (geo-processing)
  - Intersection + buffer zone
  - Distance analysis: Accessibility assessment
- week 3: 3/22 (warm-up exam, 10%)

Contents

- Chapter 3: Spatial data handling
  - GIS data format in R: *sf* data format
  - Mapping spatial objects and attributes
  - Attribute data query and manipulation
  - Statistical plots: Using ggplot2

Learning Objectives

- *sf* format and using R Package for mapping: tmap
- Compile maps based on multiple layers
- Set different shading schemes
- Plot spatial data with different parameters

tmap v3.3 Other versions

NaN Monthly downloads › 99.99th Percentile

by Martijn Tennekes   View Source   https://www.rdocumentation.org/packages/tmap Copy

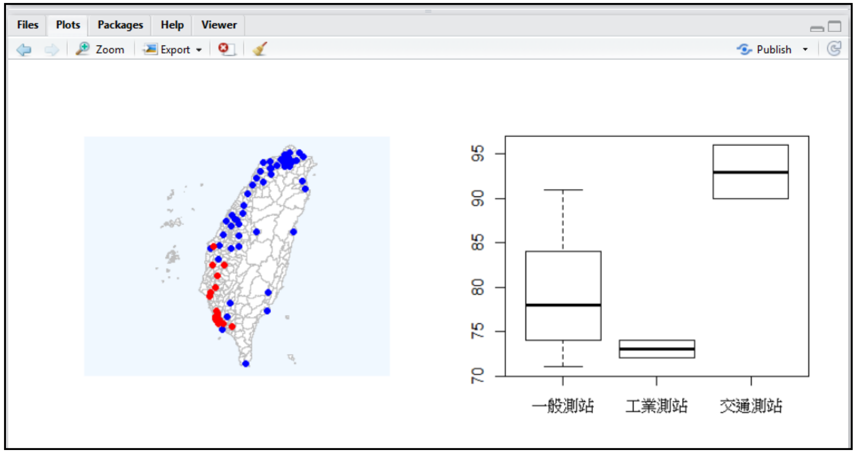
Thematic Maps

Thematic maps are geographical maps in which spatial data distributions are visualized. This package offers a flexible, layer-based, and easy to use approach to create thematic maps, such as choropleths and bubble maps.

學習要點

- 利用 R 相關套件，處理空間資料與繪製地圖。包括：
  - 幾何元件與屬性資料
  - 投影座標系統的設定
  - 圖資繪製與疊合
  - 繪製面量圖與相關設定
  - 繪製統計圖表

```
> Pollution_Map(0.3)
[1] 68.12457
```



Spatial Data in R

sf **v0.9-7** Other versions

NaN Monthly downloads › 99.99th Percentile

by Edzer Pebesma View Source https://www.rdocumentation.org/packages/sf Copy

Simple Features for R

Support for simple features, a standardized way to encode spatial vector data. Binds to 'GDAL' for reading and writing data, to 'GEOS' for geometrical operations, and to 'PROJ' for projection conversions and datum transformations. Optionally uses the 's2' package for spherical geometry operations on geographic coordinates.

Simple features or *simple feature access* refers to a formal standard (ISO 19125-1:2004) that describes how objects in the real world can be represented in computers, with emphasis on the *spatial* geometry of these objects. It also describes how such objects can be stored in and retrieved from databases, and which geometrical operations should be defined for them.

The standard is widely implemented in spatial databases (such as PostGIS), commercial GIS (e.g., ESRI ArcGIS) and forms the vector data basis for libraries such as GDAL. A subset of simple features forms the GeoJSON standard.

Simple Features for R

Features have a *geometry* describing *where* on Earth the feature is located, and they have attributes, which describe other properties. The geometry of a tree can be the delineation of its crown, of its stem, or the point indicating its center. Other properties may include its height, color, diameter at breast height at a particular date, and so on.

The standard says: "*A **simple feature** is defined by the OpenGIS Abstract specification to have both spatial and non-spatial attributes. Spatial attributes are geometry valued, and simple features are based on 2D geometry with linear interpolation between vertices.*"

Simple feature geometry types

type	description
`POINT`	zero-dimensional geometry containing a single point
`LINESTRING`	sequence of points connected by straight, non-self intersecting line pieces; one-dimensional geometry
`POLYGON`	geometry with a positive area (two-dimensional); sequence of points form a closed, non-self intersecting ring; the first ring denotes the exterior ring, zero or more subsequent rings denote holes in this exterior ring
`MULTIPOINT`	set of points; a MULTIPOINT is simple if no two Points in the MULTIPOINT are equal
`MULTILINESTRING`	set of linestrings
`MULTIPOLYGON`	set of polygons
`GEOMETRYCOLLECTION`	set of geometries of any type except GEOMETRYCOLLECTION

A *sf* object

```
## Simple feature collection with 100 features and 6 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## precision:      double (default; no precision model)
## First 3 features:
##   BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79                           geom
## 1  1091     1      10  1364     0      19 MULTIPOLYGON(((-81.47275543...
## 2   487     0      10   542     3      12 MULTIPOLYGON(((-81.23989105...
## 3  3188     5     208  3616     6     260 MULTIPOLYGON(((-80.45634460...
```

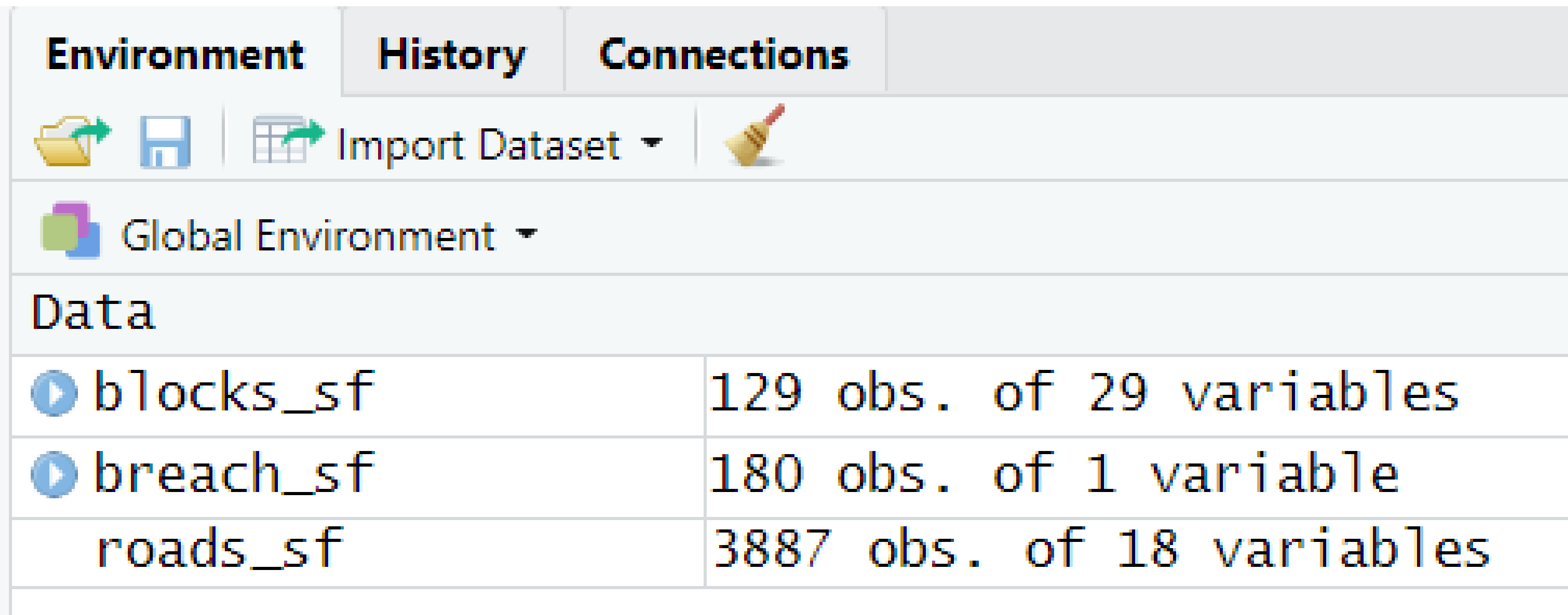
Simple feature

Simple feature geometry list-colum (sfc)

Simple feature geometry (sfg)

Loading Spatial Data from NTU CEIBA

```
setwd("C:/Wen_Files/SA_2021/Data")
load("Sample.RData")
```



0. Understanding *sf* format and coordinates

```
class(blocks_sf)
head(blocks_sf)

# extracing as a new layer
new_sf<-blocks_sf[,6]
new2_sf<-blocks_sf[1:3,]

# attribute table
blocks_df<- as.data.frame(blocks_sf)
class(blocks_df)

# coordinate system
st_crs(blocks_sf)
st_crs(roads_sf)
st_crs(roads_sf)<-st_crs(blocks_sf)

# export/import shapefiles
st_write(blocks_sf,"blocks.shp", delete_layer = TRUE)
blocks2_sf<- st_read("blocks.shp")
```

CRS: Coordinate Reference System

```
> st_crs(blocks_st)
Coordinate Reference System:
  User input:  +proj=lcc +datum=NAD27 +lon_0=-72d45 +lat_1=41d52 +lat_2=41d12 +lat_0=40d50 +x_0=182880.3657
607315 +y_0=0 +units=us-ft +no_defs +ellps=clrk66 +nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat
```

Lambert Conformal Conic projection (LCC)

Coordinate Reference System:

User input: +proj= lcc

+datum=NAD27

+lon_0=-72d45 +lat_1=41d52 +lat_2=41d12 +lat_0=40d50

+x_0=182880.3657607315

+y_0=0

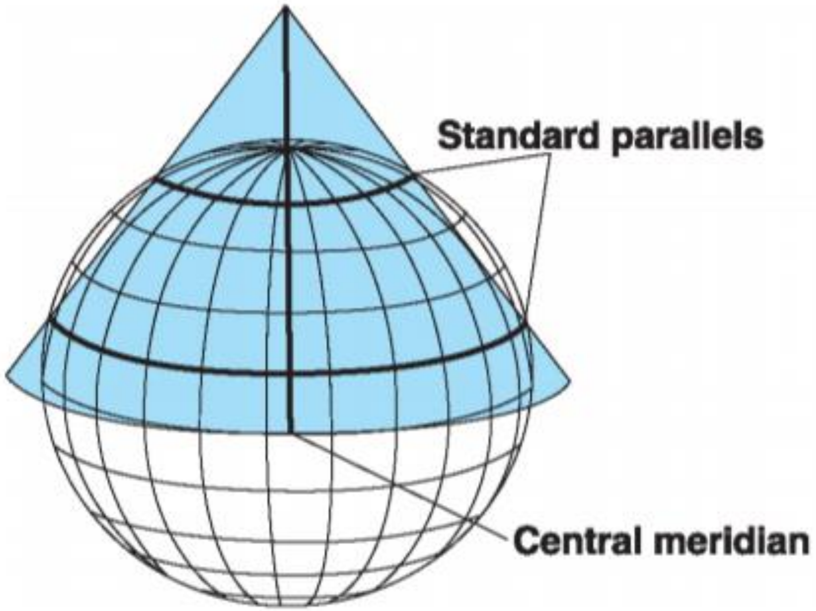
+units=us-ft

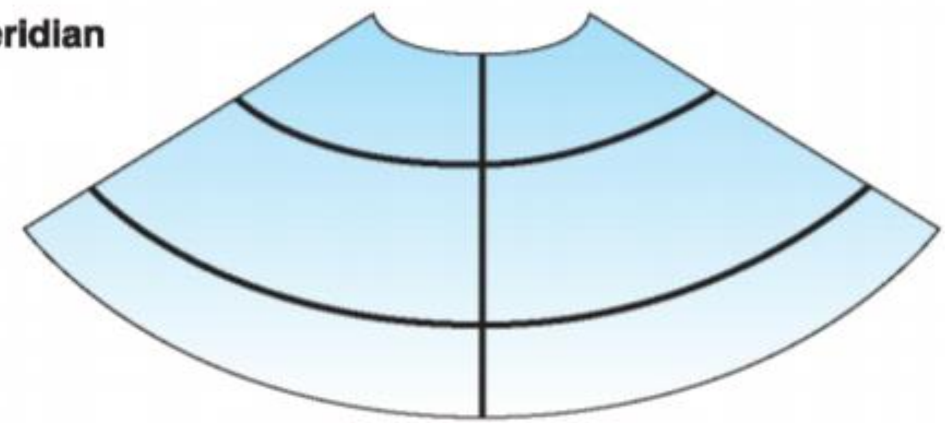
+no_defs

+ellps=clrk66

+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat

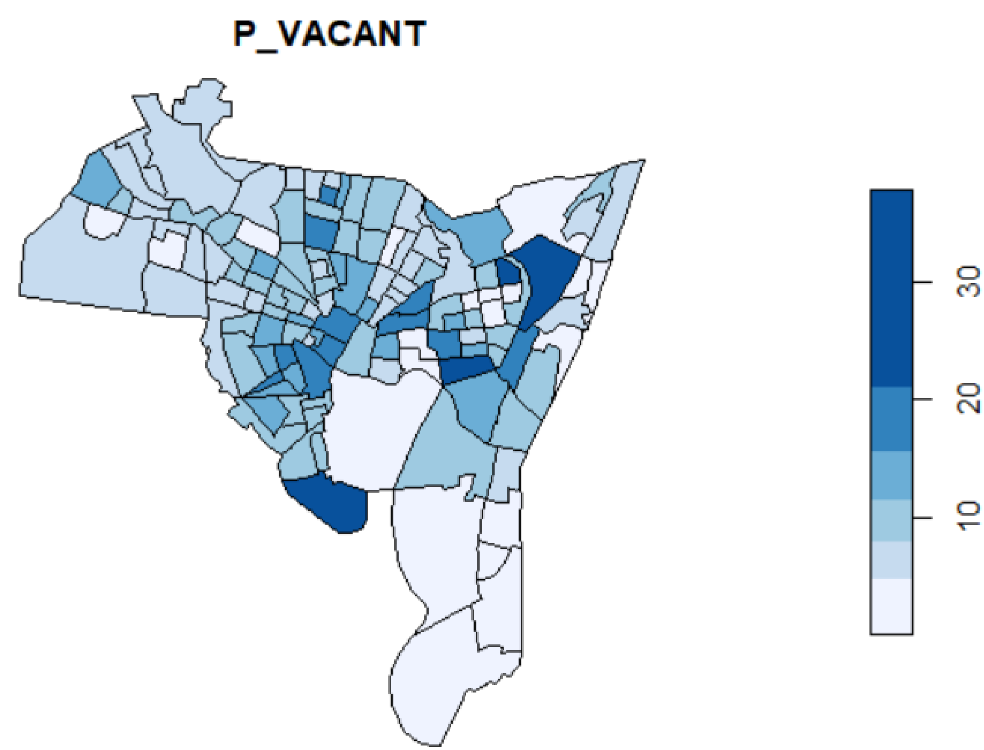
Lambert Conformal Conic projection





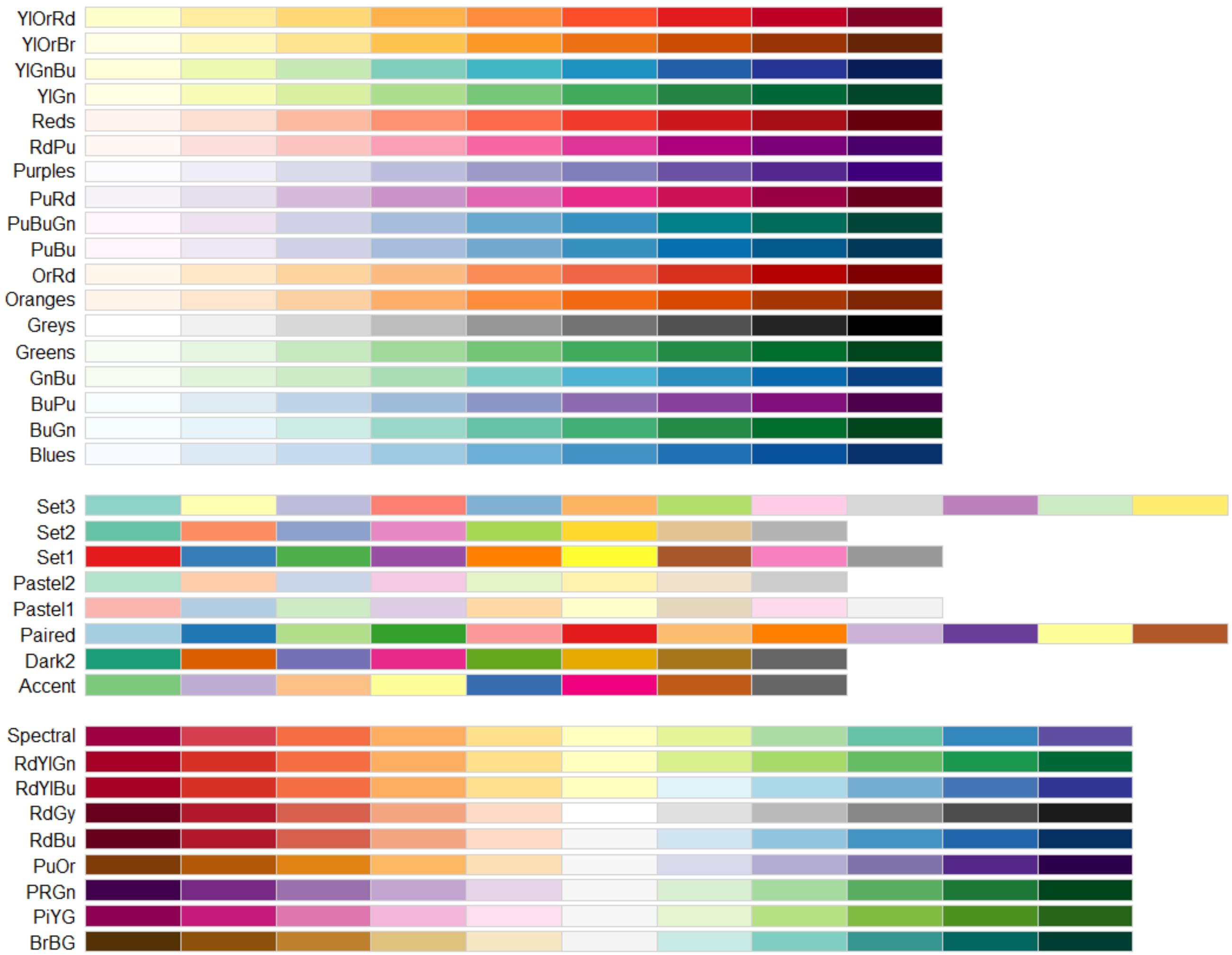
1. Mapping Spatial Objects
1.1 Using plot()

```
brewer.blues(6)
plot(blocks_sf["P_VACANT"], breaks = "jenks", nbreaks = 6, pal=brewer.blues(6))
```



Set different shading schemes: colors

```
display.brewer.all()
```



1.2 Using tmap package

https://cran.r-project.org/web/packages/tmap/vignettes/tmap-getstarted.html

tmap: get started!

With the tmap package, thematic maps can be generated with great flexibility. The syntax for creating plots is similar to that of `ggplot2`, but tailored to maps. This vignette is for those who want to get started with tmap within a couple of minutes. A more detailed description of tmap can be found in an article published in the Journal of Statistical Software (JSS). However, that article describes tmap version 1.11-2, which is out-of-date. Some major changes have been made since then, which are described in `vignette("tmap-changes")`.

For more context on R's geographic capabilities we recommend the online version of the book Geocomputation with R. The Making maps with R chapter of the book provides many more context and abundant code examples of map making with `tmap` and other packages. Other good resources are the vignettes of the `sf` package, and the website rspatial.org .

Using qtm() in tmap package

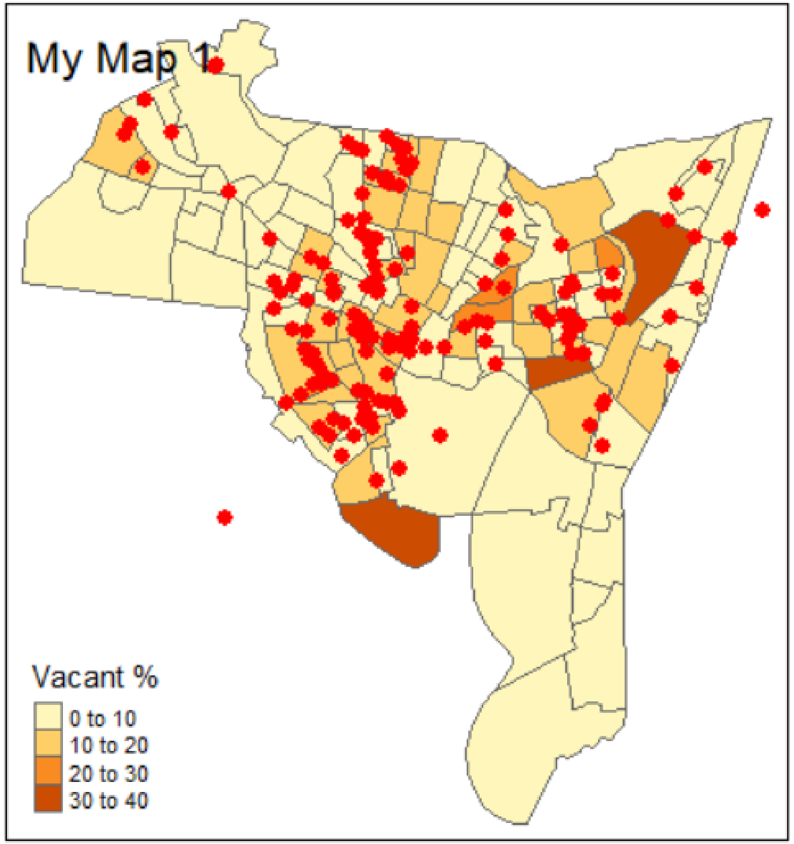
```
qtm(blocks_sf, fill="red", style="natural")

qtm(blocks_sf, fill="P_VACANT",
      fill.title="Vacant %", title="My Map 1")
```

Mapping Spatial Objects

```
# choropleth
lyr1<- qtm(blocks_sf, fill="P_VACANT",
                fill.title="Vacant %", title="My Map 1")

# bubble map
lyr2<- qtm(blocks_sf, symbols.size="P_VACANT",
symbols.title.size="Vacant %", title="My Bubble Map")

# lines
lyr_road <- tm_shape(roads_sf)+tm_lines(col="orange")

# points
lyr_crimes <- tm_shape(breach_sf)+
                tm_dots(col="red", size= 0.3)
```

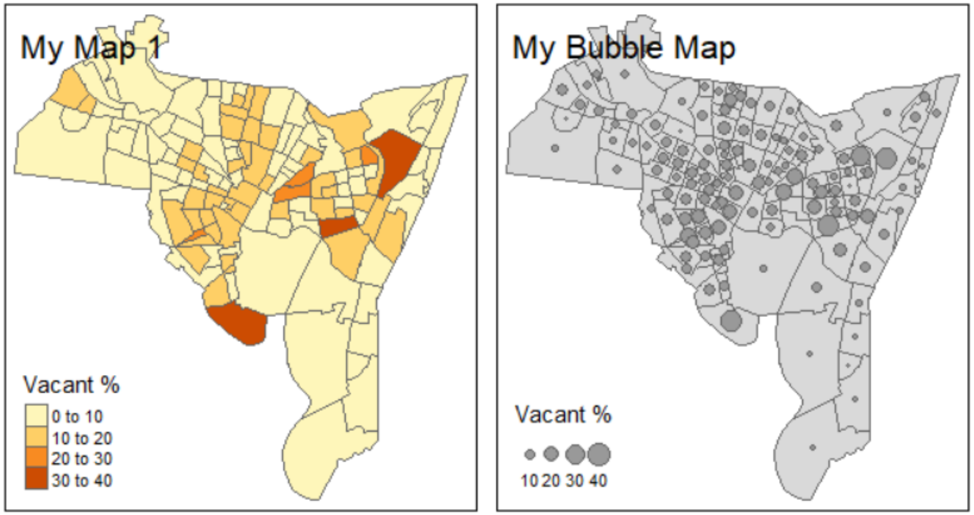
Plotting multiple layers

```
# overlay multiple plots
lyr1+lyr_crimes

st_crs(breach_sf)
st_crs(blocks_sf)
```



Plotting multiple layers (cont'd)

```
# showing multiple plots

library(grid)
# open a new plot page
grid.newpage()
# set up the layout
pushViewport(viewport(layout=grid.layout(1,2)))
# plot using the print command
print(lyr1, vp=viewport(layout.pos.col = 1))
print(lyr2, vp=viewport(layout.pos.col = 2))

dev.off() # reset
```



2. Attribute Query & Selection

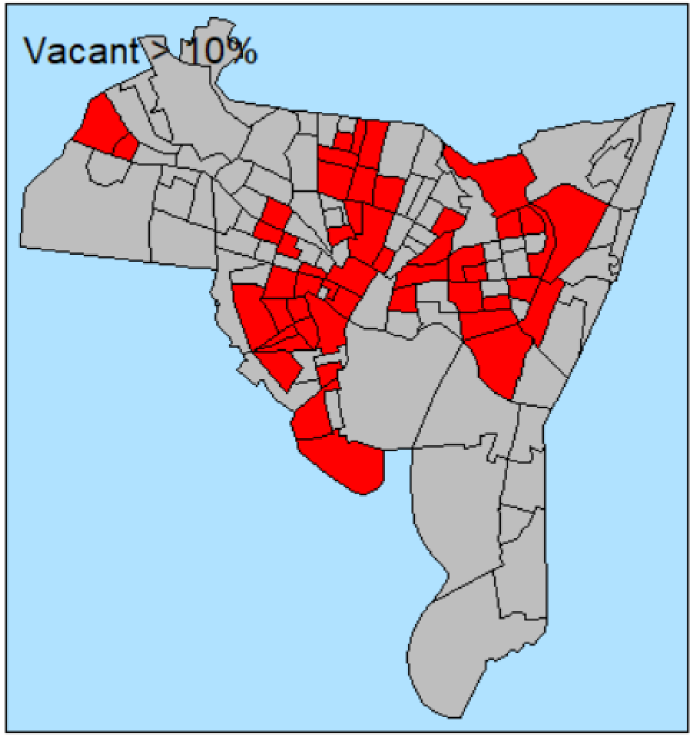
The code used above includes logical operators and illustrates how they can be used to select elements that satisfy some condition. These can be used singularly or in combination to select in the following way:

```
data <- c(3, 6, 9, 99, 54, 32, -102)
index <- (data == 32 | data <= 6)
data[index]

## [1]   3  6  32  -102
```

These are described in greater detail in Chapter 4.

Mapping Selected Data

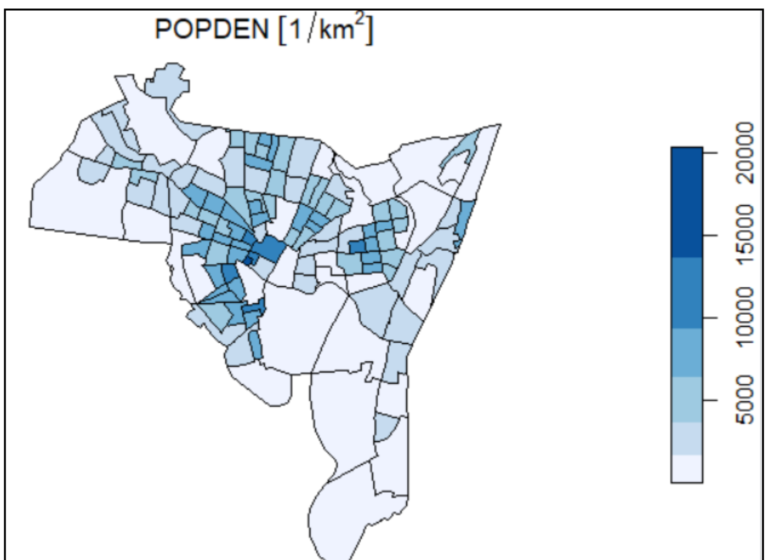
```
index <- (blocks_sf$P_VACANT > 10)
newblocks_sf <- blocks_sf[index,]
lyr3<- qtm(newblocks_sf, fill="red", title="Vacant > 10%", style="natural")
lyr_bg<- qtm(blocks_sf, fill="grey")
lyr_bg+lyr3
```

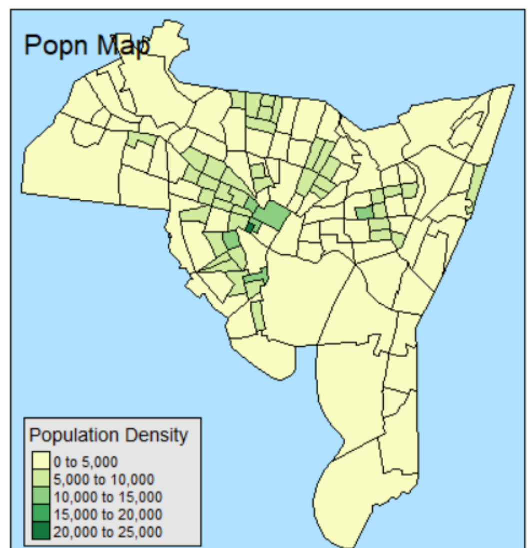


3. Calculating Fields

st_area()

```
# add a new AREA field
x<-st_area(blocks_sf) # unit: foot

library(units)
x2<-set_units(x, km^2)

blocks_sf$AREA1 <- x2

# remove a field
# blocks_sf <- subset(blocks_sf, select = -c(AREA1))

head(blocks_sf)

blocks_sf$POPDEN <- blocks_sf$POP1990 / blocks_sf$AREA1

plot(blocks_sf["POPDEN"], breaks = "jenks", nbreaks = 6, pal= brewer.blues(6))
qtm(blocks_sf, fill="POPDEN", fill.title="Population Density", title="Popn Map", style="natural")
```





4. Using tmap: detailed settings for mapping

tm_shape(檔名)+tm_polygon(欄位設定)

+tm_scale_bar()

+tm_compass()

+tm_layout()

↓

tm_lines()

tm_dots()

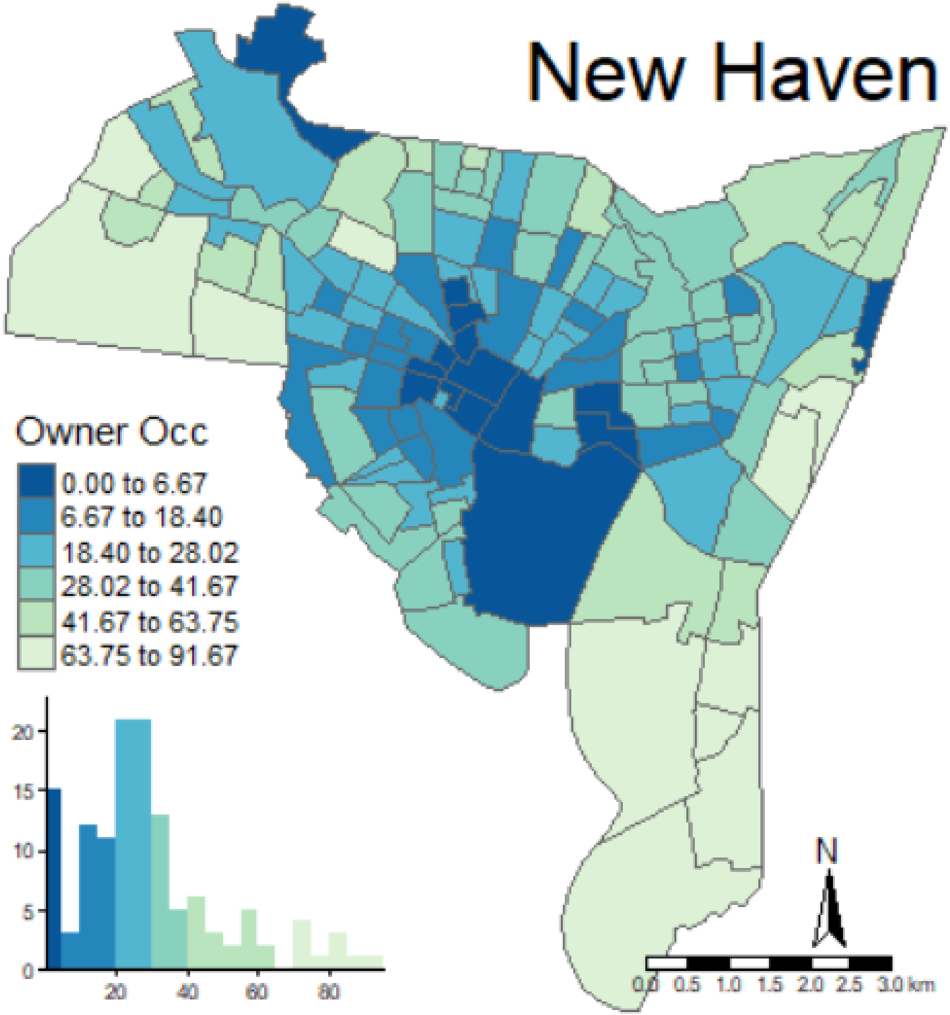
Detailed settings for mapping

```
tm_shape(blocks_sf) +
  tm_polygons("P_OWNEROCC", title = "Owner Occ", palette = "-GnBu",
              breaks = c(breakv),
              legend.hist = T) +
  tm_scale_bar(width = 0.22) +
  tm_compass(position = c(0.8, 0.08)) +
  tm_layout(frame = F, title = "New Haven",
            title.size = 2, title.position = c(0.55, "top"),
            legend.hist.size = 0.5)
```

```
breakv<- getBreaks( v = blocks_sf$P_OWNEROCC,
          nclass = 6, method = "jenks“ )
```

classification method: "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", "jenks", "dpih", "q6", "geom", "arith", "em", "msd"

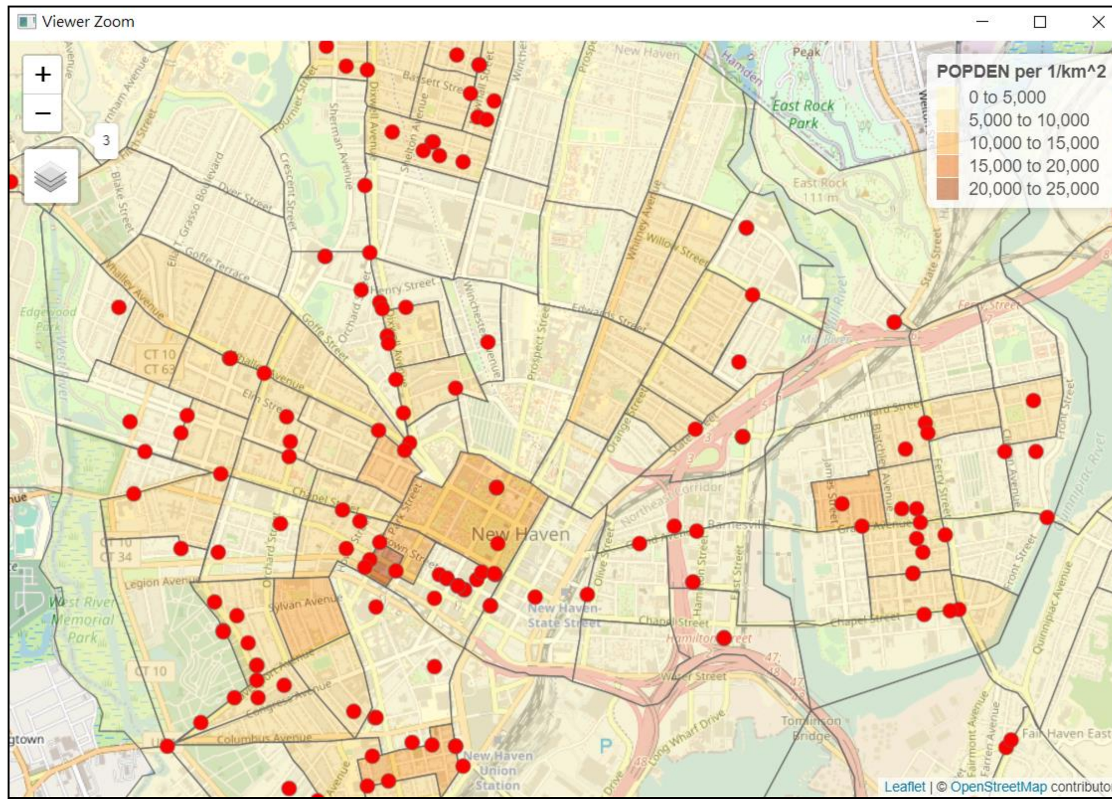
Detailed settings for mapping



5. Interactive Mapping

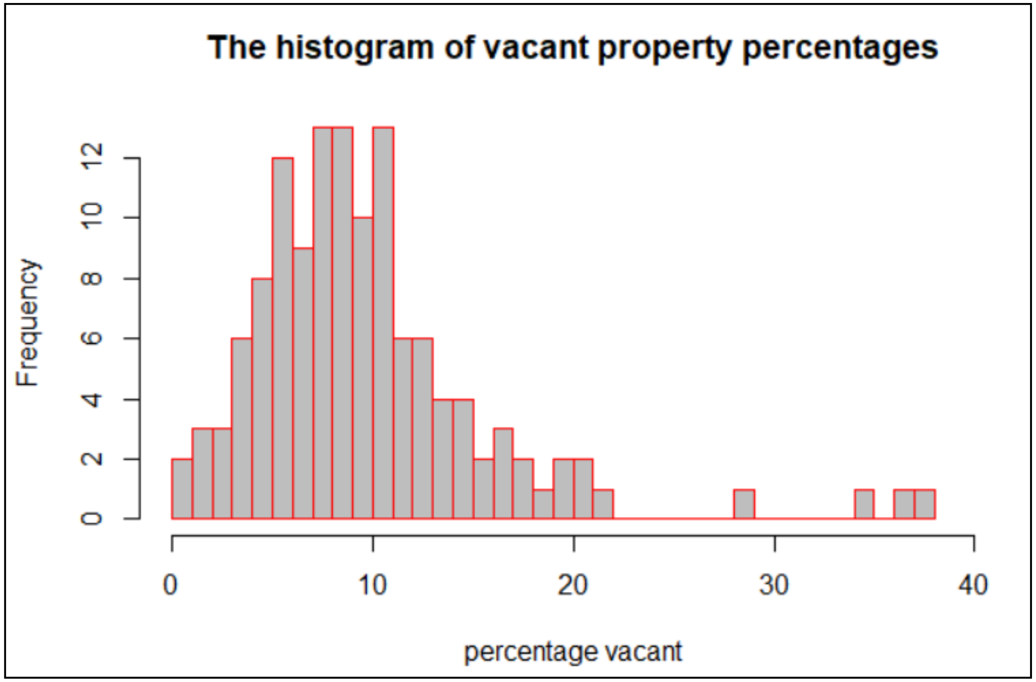
transparency number

```
tmap_mode("view")
lyr5<- tm_shape(blocks_sf)+tm_polygons("POPDEN", alpha=0.5)
lyr5+lyr_crimes

ttm()
```



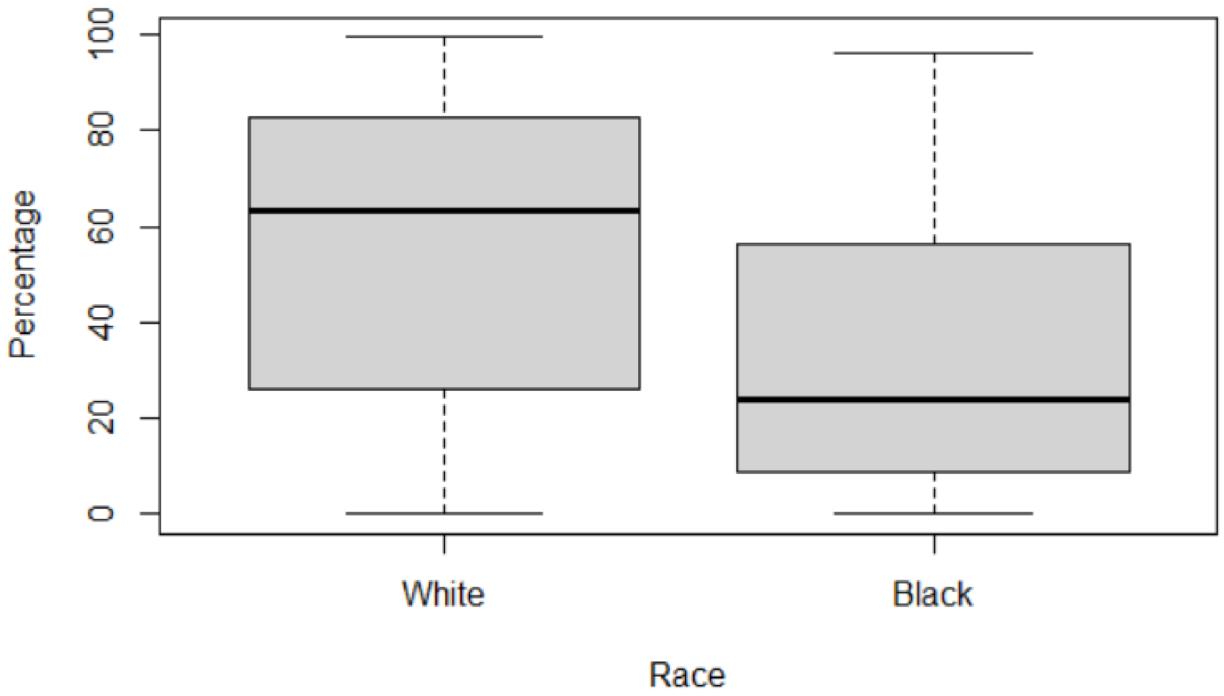
6. Statistical Plots：histogram

```
hist(blocks_sf$P_VACANT, breaks = 40, col = "grey",
     border = "red",
     main = "The histogram of vacant property percentages",
     xlab = "percentage vacant", xlim = c(0,40))
```



6. Statistical Plots：box plot

```
boxplot(blocks_sf$P_WHIT, blocks_sf$P_BLACK, names=c("White", "Black"),
        xlab="Race", ylab="Percentage")
```



Statistical Plots: Using ggplot2 package

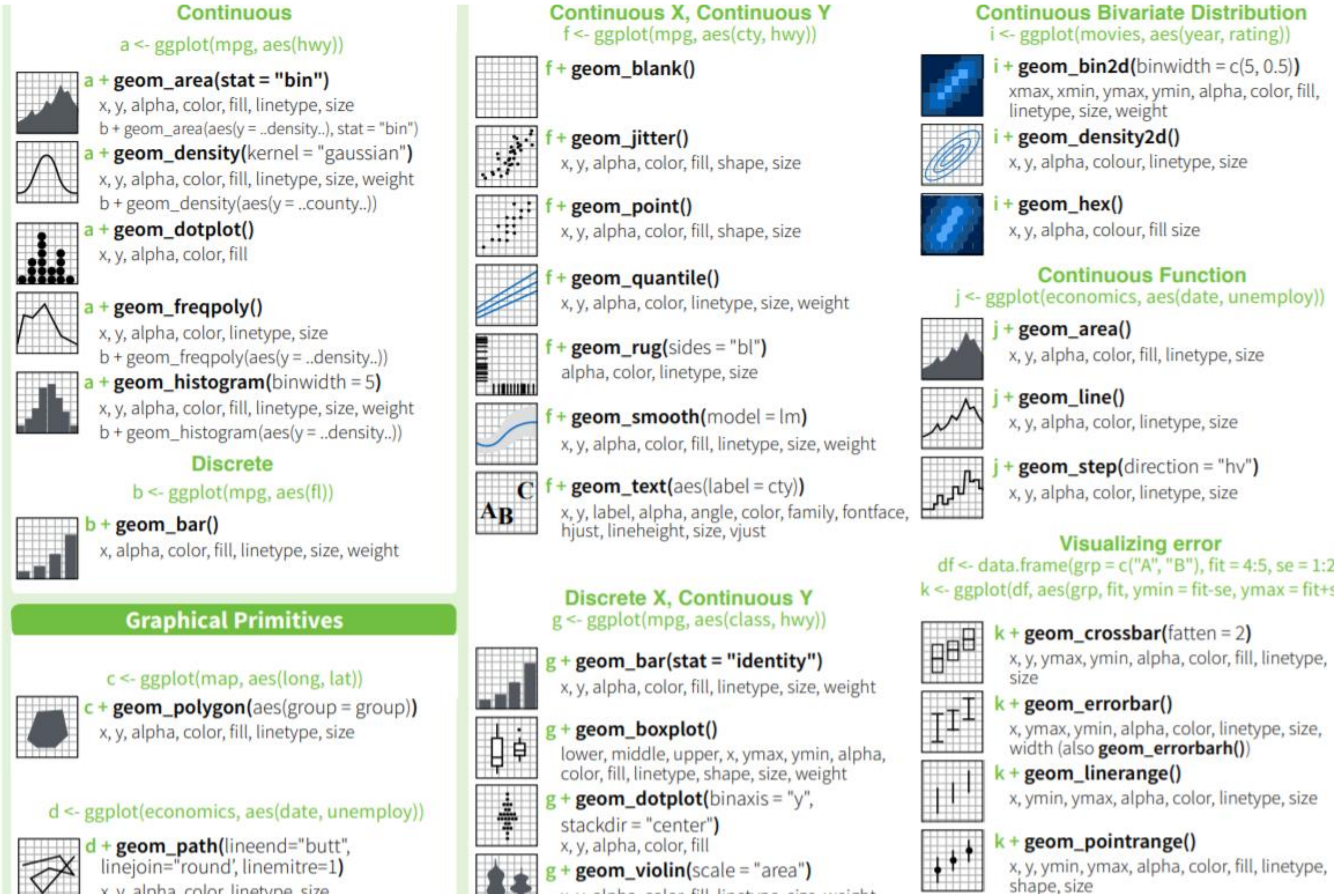
ggplot(檔名) +aes(欄位設定)

+ geometric objects (geom_) 設定圖表格式

（例如：geom_histogram(),

geom_boxplot()...）

geometric objects (geom_)



Continuous

a <- ggplot(mpg, aes(hwy))

a + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path(**lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, linetype, size

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug(**sides = "bl"**)**
alpha, color, linetype, size

f + **geom_smooth(**model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text(**aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + **geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d(**binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step(**direction = "hv"**)**
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+s

k + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, linetype, size

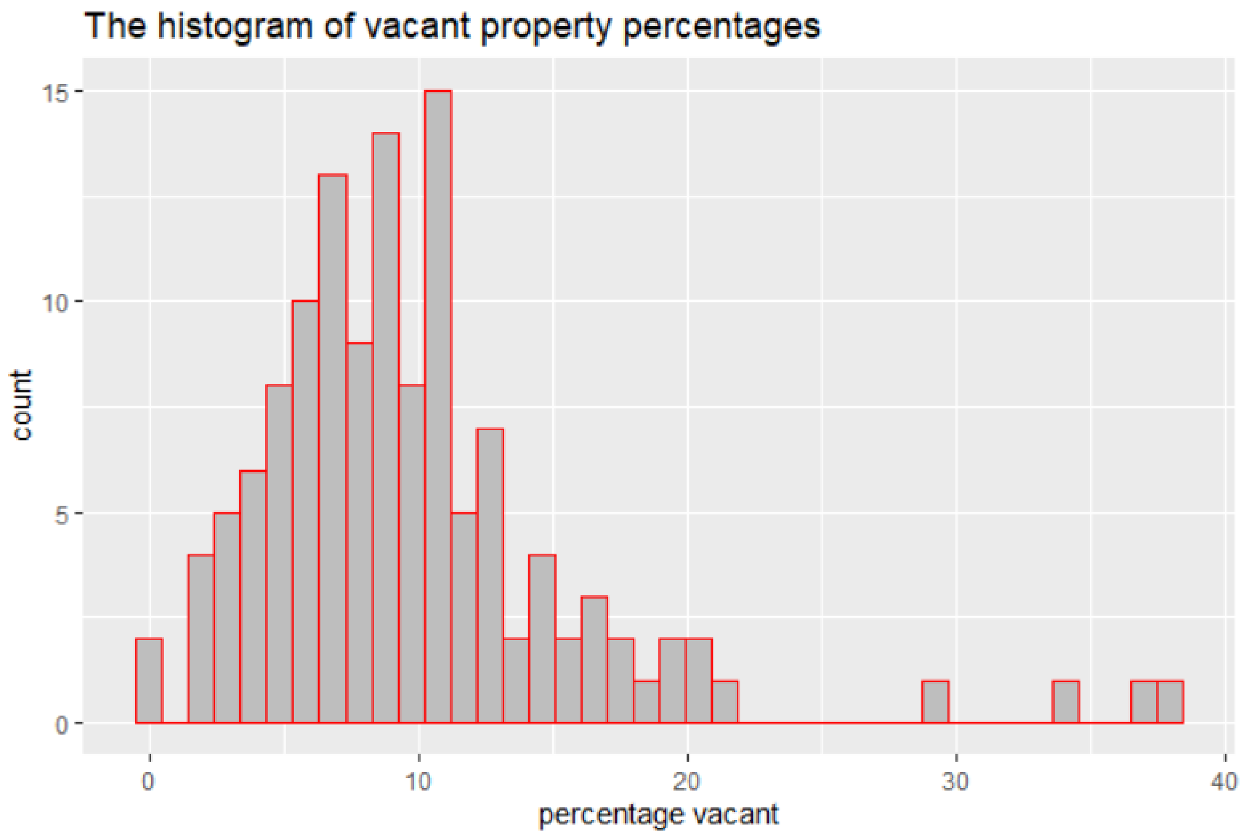
k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Using ggplot2

```
plot1<- ggplot(blocks_df) + aes(P_VACANT) +
          geom_histogram(col = "red", fill = "grey", bins = 40) +
          xlab("percentage vacant") +
          labs(title = "The histogram of vacant property percentages")
```



Using ggplot2: Boxplot

Our dataset (n=129)

	POP1990	P_MALES	P_FEMALES	P_WHITE	P_BLACK	P_AMERI_ES
0	2396	40.02504	59.97496	7.095159	87.020033	0.584307
1	3071	39.07522	60.92478	87.105177	10.452621	0.195376
2	996	47.38956	52.61044	32.931727	66.265060	0.100402
3	1336	42.66467	57.33533	11.452096	85.553892	0.523952
4	915	46.22951	53.77049	73.442623	24.371585	0.327869
5	1318	50.91047	49.08953	87.784522	7.435508	0.758725

What we need (n=387)

race percent

	variable	value
123	P_WHITE	96.545769
124	P_WHITE	84.200743
125	P_WHITE	99.135135
126	P_WHITE	98.731884
127	P_WHITE	98.068966
128	P_WHITE	99.417098
129	P_WHITE	98.895706
130	P_BLACK	87.020033
131	P_BLACK	10.452621
132	P_BLACK	66.265060
133	P_BLACK	85.553892
134	P_BLACK	24.371585
135	P_BLACK	7.435508
136	P_BLACK	30.931796

Introducing wide vs. long tables

From wide…

Weekday	Q1	Q2	Q3	Q4
Mon	9.9	5.4	8.8	6.9
Tues	4.9	9.7	7.9	5.0
Wed	8.8	11.1	10.2	9.3
Thurs	12.2	10.2	9.2	9.7
Fri	12.2	8.1	7.9	5.6

… to long

Weekday	Quarter	Delay
Mon	Q1	9.9
Tues	Q1	4.9
Wed	Q1	8.8
Thurs	Q1	12.2
Fri	Q1	12.2
Mon	Q2	5.4
Tues	Q2	9.7
Wed	Q2	11.1
Thurs	Q2	10.2
Fri	Q2	8.1
Mon	Q3	8.8
Tues	Q3	7.9
Wed	Q3	10.2
Thurs	Q3	9.2
Fri	Q3	7.9
Mon	Q4	6.9
Tues	Q4	5.0
Wed	Q4	9.3
Thurs	Q4	9.7
Fri	Q4	5.6

Using Reshape package

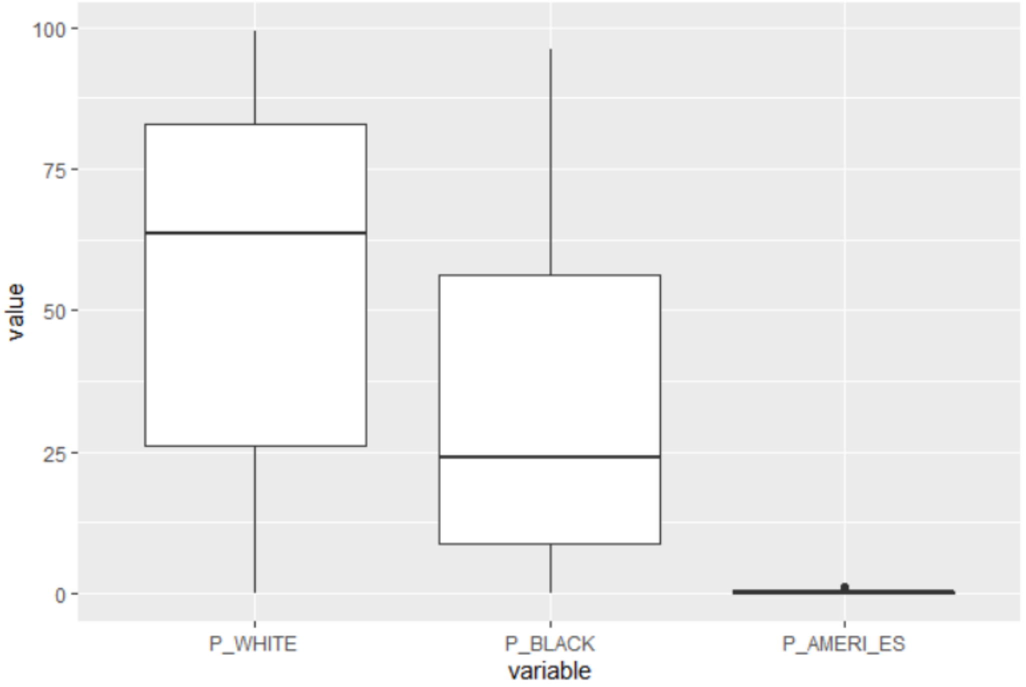
```
library(reshape2)

blocks2_df<- melt(blocks_df[, c("P_WHITE", "P_BLACK", "P_AMERI_ES")])
head(blocks2_df)
```

```
plot2<- ggplot(blocks2_df) +
        aes(variable, value) +
        geom_boxplot()
```

	variable	value
123	P_WHITE	96.545769
124	P_WHITE	84.200743
125	P_WHITE	99.135135
126	P_WHITE	98.731884
127	P_WHITE	98.068966
128	P_WHITE	99.417098
129	P_WHITE	98.895706
130	P_BLACK	87.020033
131	P_BLACK	10.452621
132	P_BLACK	66.265060
133	P_BLACK	85.553892
134	P_BLACK	24.371585
135	P_BLACK	7.435508
136	P_BLACK	30.931796

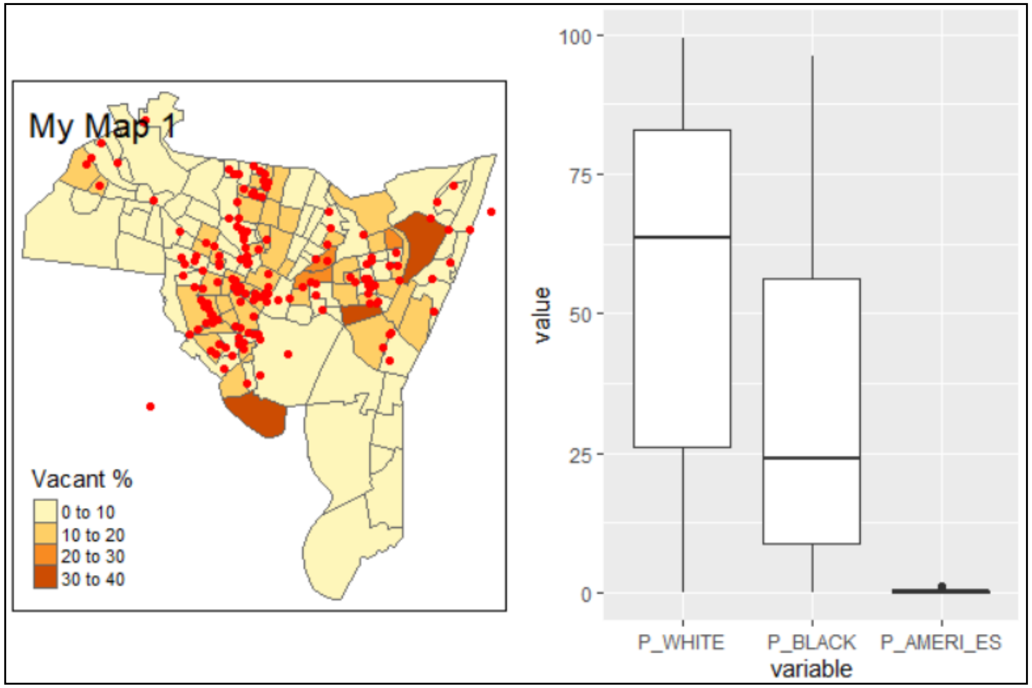
Using ggplot2: Boxplot

```
plot2<- ggplot(blocks2_df) +
            aes(variable, value) +
            geom_boxplot()
```



Displaying multiple maps and plots

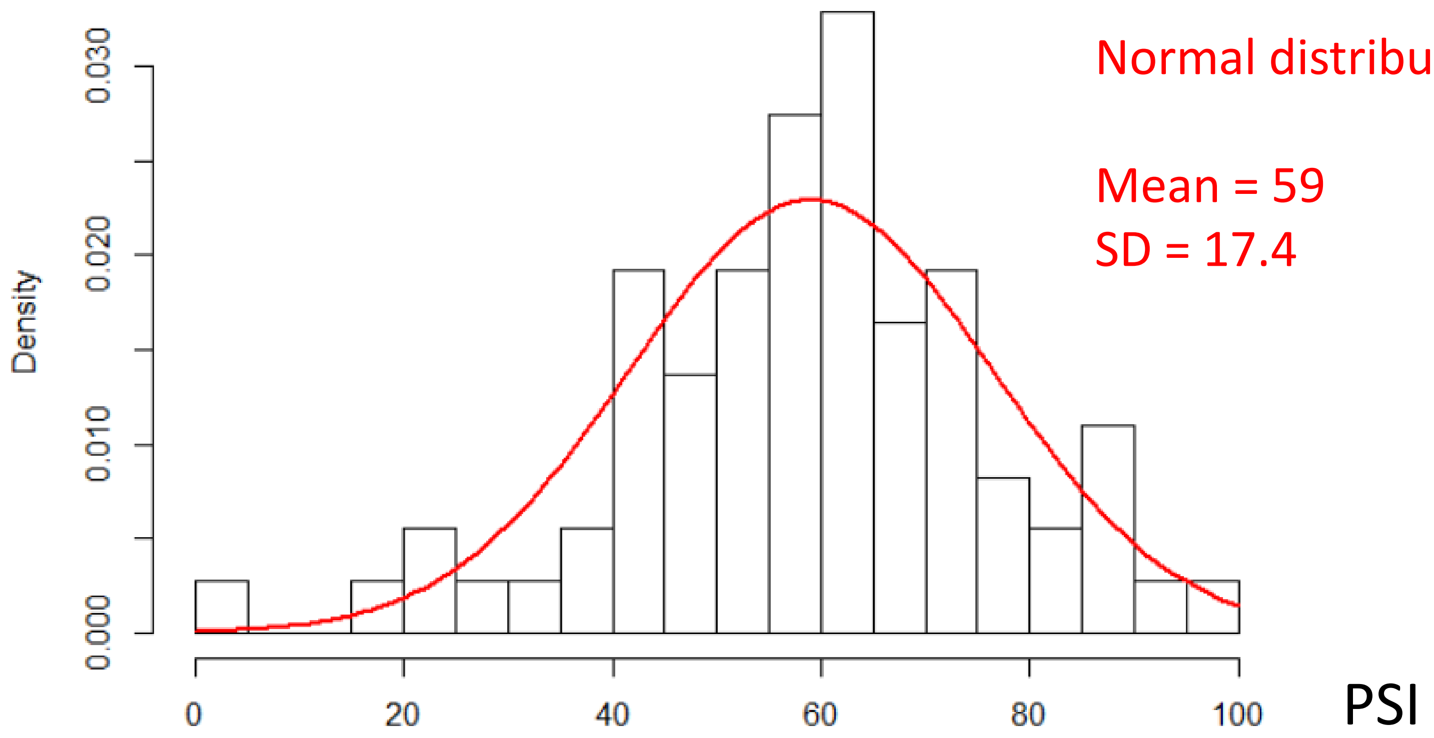
```
grid.newpage()
pushViewport(viewport(layout=grid.layout(1,2)))
print(lyr1+lyr_crimes, vp=viewport(layout.pos.col = 1))
print(plot2, vp=viewport(layout.pos.col = 2))
```



實習:建立特定超越機率的空汙地圖

EPA_STN1.shp

PSI is a type of air quality index



Normal distribution:

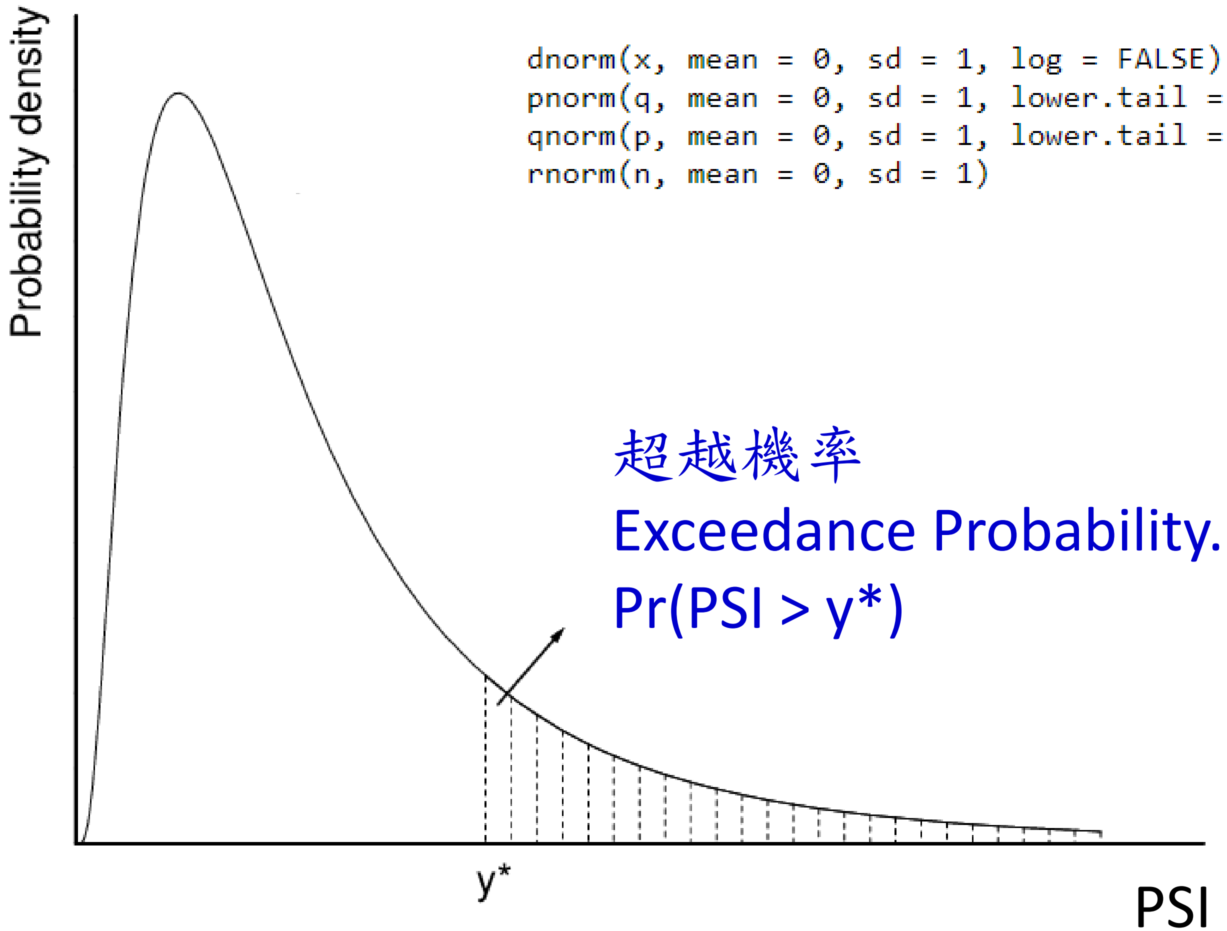
Mean = 59
SD = 17.4

實習:超越機率的概念

PSI is a type of air quality index

複習R的機率函數使用

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```



實習:建立特定超越機率的空汙地圖

- 建立繪製地圖的函數：Pollution_Map (agr1)

  引數agr1 是可自行設定的超越機率 (e.g. 0.2)

  - (1) 該函數會回傳該超越機率所對應的PSI值。
  - (2) 以此數值為臨界值，繪製空氣污染地圖，

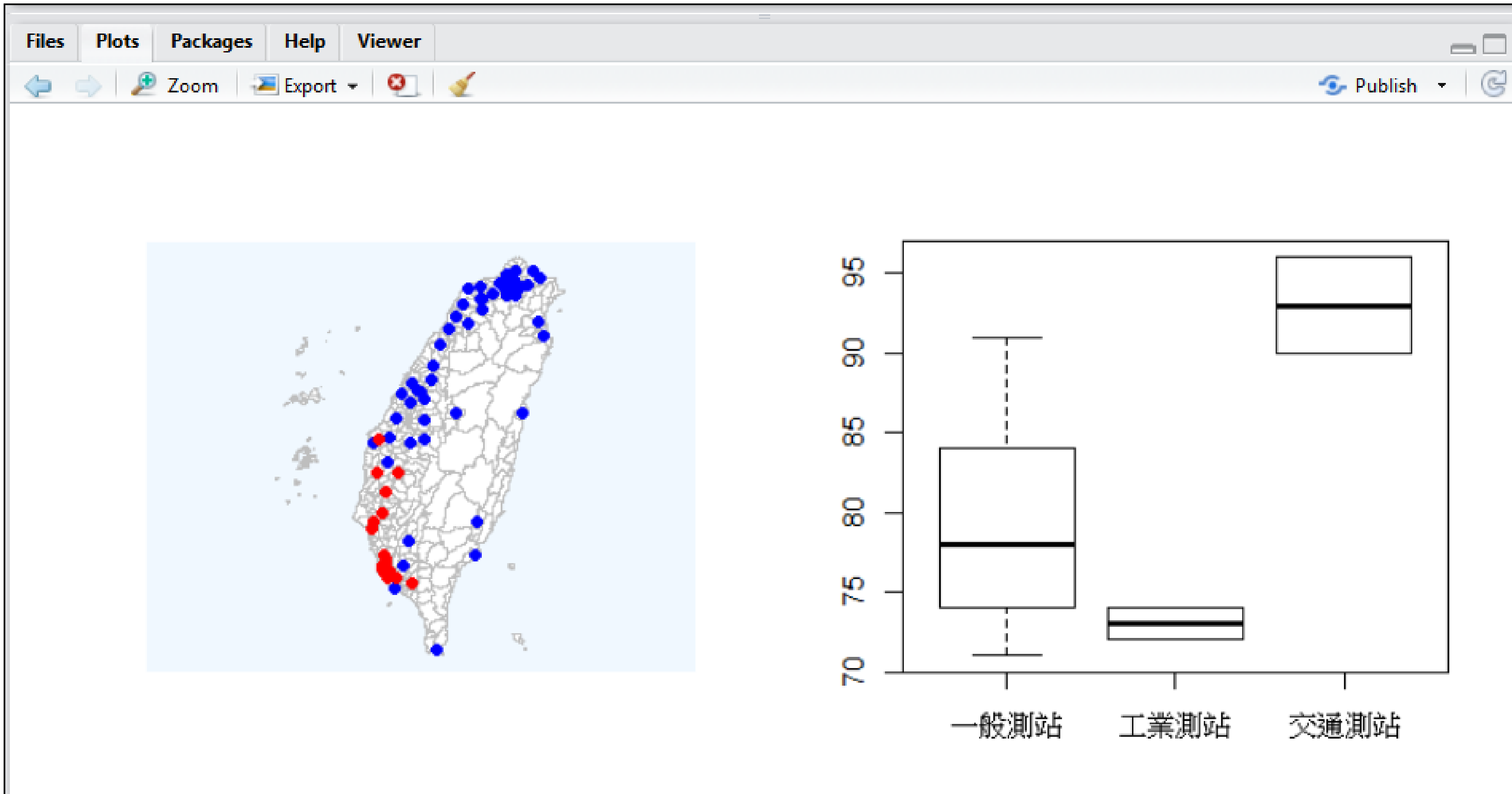
    超過該數值的測站，表示紅色，其餘為藍色。
  - (3) 以此數值為臨界值，針對超過該數值的測站，

    按照測站類別(SiteType)，依照「一般測站、工業測站、交通測站」這三類，以box plot呈現PSI分布。

實習的預期結果

* 執行Pollution_Map(0.3)與Pollution_Map(0.5)來檢核函數結果

```
> Pollution_Map(0.3)
[1] 68.12457
```



作業：繪製人口老化地圖與統計圖表

Data: Popn_TWN2.shp

- 1: 台灣人口密度地圖
- 2: 大台北人口老化地圖
- 3: Boxplot: 比較各地區的老年人口分布以及不同年齡結構的人口分布

作業成果的詳細說明

- [1]繪製**台灣**鄉鎮人口密度的面量圖 (Popn/Area)

  [按照Quantile 分成6級，含圖例、比例尺、圖名和指北針]
- [2]在大台北地區 (含台北、新北、基隆、桃園、宜蘭等)範圍內，以紅色標示老年人口比例 (Age_L65/Popn)在**top20%**的鄉鎮市區，繪製**大台北地區**的人口老化地圖 。
- [3-1] 繪製boxplot。比較台灣的高密度(鄉鎮人口密度 > 10,000/km2) vs. 低密度(鄉鎮人口密度 < 2,000/km2) 的老年人口比例的分布。
- [3-2] 繪製boxplot。比較台灣老/中/青年群族的鄉鎮人口數分布。

  老人：年齡 >= 65

  中年：年齡 21-64

  青年：年齡 <= 20