E04

R語言畫地圖 (Maps)

吳漢銘

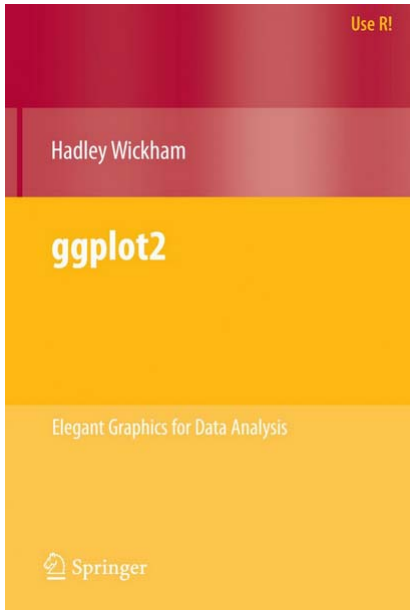
國立政治大學 統計學系

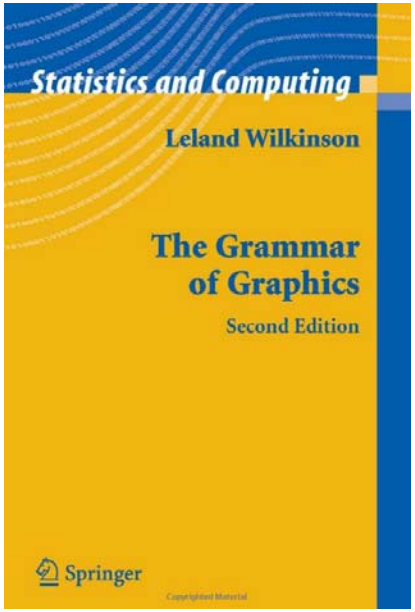
http://www.hmwu.idv.tw

本章大綱

- ggplot2簡介
- 使用Google Cloud Platform雲端服務
- **RgoogleMaps** : plotting on Google static maps in R
- **ggmap**: Spatial visualization with ggplot2
- 分級著色圖 /面量圖(Choropleth Maps)
- **choroplethr**: Simplify the creation of choropleth maps in R

What is ggplot2





- High-level graphics system developed by Hadley Wickham.
- Implements grammar of graphics from Leland Wilkinson.
- Streamlines many graphics workflows for complex plots.
- Syntax centered around main **`ggplot`** function.
- Simpler **`qplot`** function provides many shortcuts.

- The principle that a plot: **Plot = data + aesthetics + geometry**
  - **`data`**: a data frame (dataset).
  - **`aesthetics`**:
    - indicates **`x`** and **`y`** variables,
    - tells R how data are displayed in a plot.
      (e.g. color, size and shape of points etc.)
  - **`geometry`**: to the type of graphics
    (bar chart, histogram, box plot, line plot, density plot, dot plot etc.)

What is ggplot2

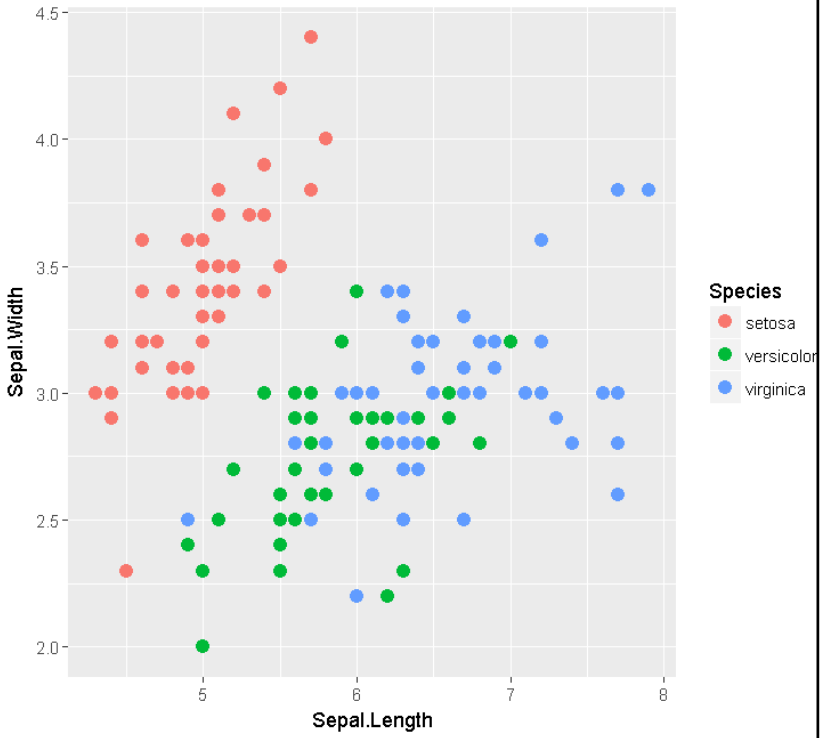
General ggplot syntax

```
ggplot(data, aes(...)) + geom() + ... + stat() + ...
```

```
> install.packages("ggplot2")
> library(ggplot2)
> ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
+ geom_point(size=3)
```

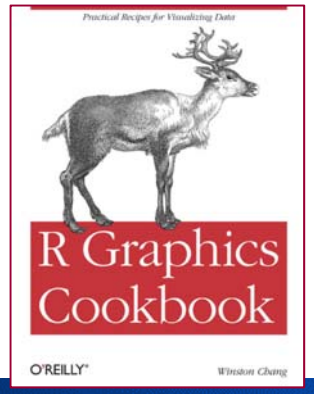
Other important parts of plot:

- **Faceting** implies the same type of graph can be applied to each subset of the data. (e.g, for variable gender, creating 2 graphs for male and female.)
- **Annotation** lets you to add text to the plot.
- **Summary Statistics** allows you to add descriptive statistics on a plot.
- **Scales** are used to control x and y axis limits.



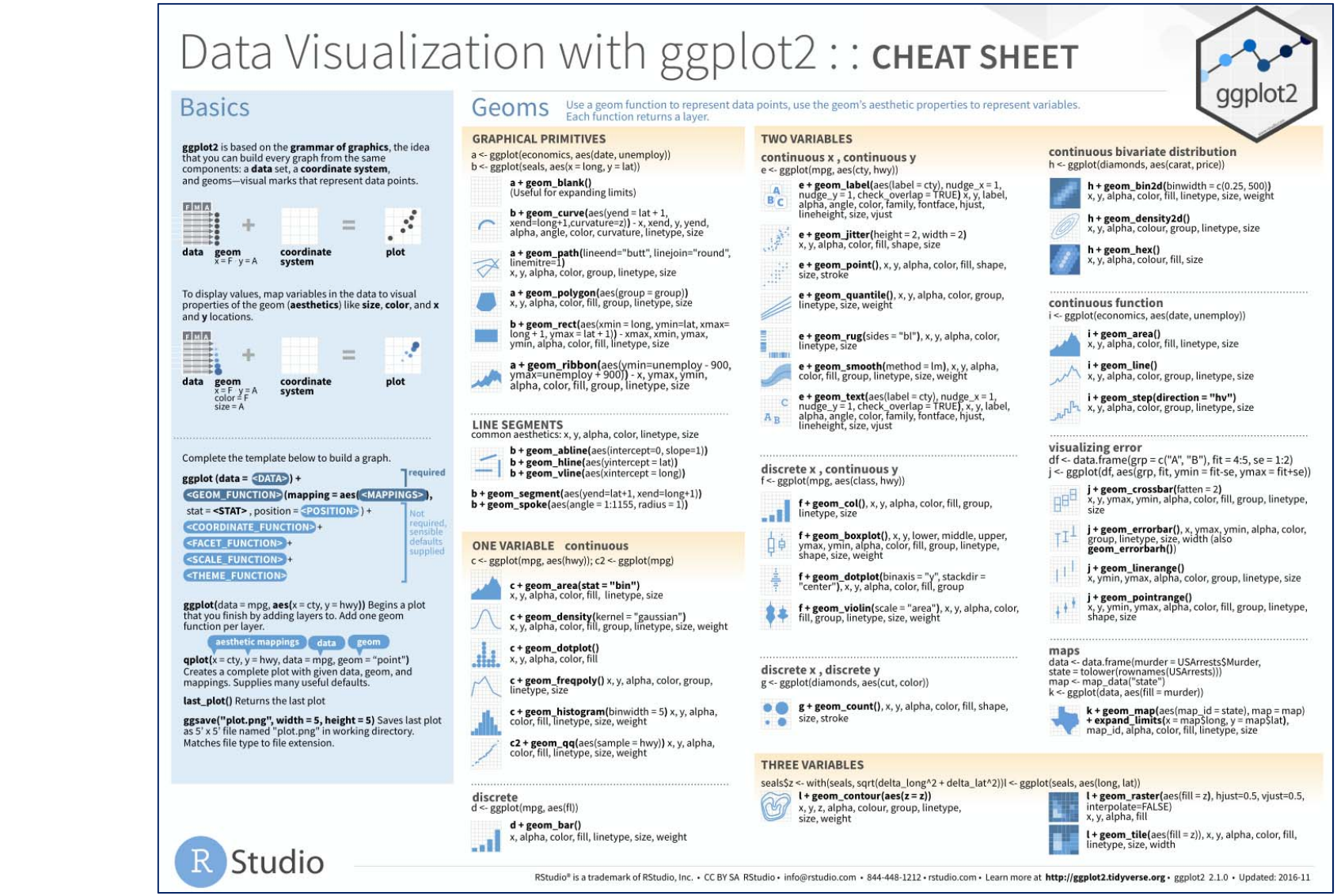
Why ggplot2?

- More elegant & compact code than base graphics.
- More aesthetically pleasing than base graphics.
- Very powerful for exploratory analysis.
- Supports a continuum of expertise.
- Easy to get started, plenty of power for complex figures.
- Publication-quality figures.
- Excellent themes can be created with a single command.
- Its colors are nicer and more pretty than the usual graphics.
- Easy to visualize data with multiple variables.
- Provides a platform to create simple graphs providing plethora of information.



- Manual: http://had.co.nz/ggplot2/
- Introduction: http://www.ling.upenn.edu/~joseff/rstudy/summer2010_ggplot2_intro.html
- Book: http://had.co.nz/ggplot2/book/
- R Graphics Cookbook: http://www.cookbook-r.com/Graphs/

Data Visualization with ggplot2



Data Visualization with ggplot2 : : CHEAT SHEET

ggplot2

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

data · geom (x = F, y = A) · coordinate system · plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data · geom (x = F, y = A, color = F, size = A) · coordinate system · plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
  stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, geom = "point**")** Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

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

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

b + geom_segment(aes(yend=lat+1, xend=long+1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)**)** x, y, alpha, color, fill, linetype, size, weight

discrete

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

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

TWO VARIABLES

continuous x , continuous y

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

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**, x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

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

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

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

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

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

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

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

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

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

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

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

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map**)**
+ expand_limits(x = map$long, y = map$lat**)**, map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

l + geom_tile(aes(fill = z)**)**, x, y, alpha, color, fill, linetype, size, width

RStudio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **http://ggplot2.tidyverse.org** • ggplot2 2.1.0 • Updated: 2016-11

https://www.rstudio.com/resources/cheatsheets/

Data Visualization with ggplot2

Stats An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).

Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

geom to use | stat function | geommappings

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

variable created by stat

c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1) **x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30) **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) **x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)) **x** | ..x.., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) **sample, x, y** | ..sample.., ..theoretical..

e + stat_sum() x, y, size | ..n.., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")

h + stat_summary_bin(fun.y = "mean", geom = "bar")

e + stat_unique()

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

(n <- d + geom_bar(aes(fill = fl)))

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

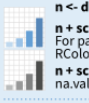
scale_*_continuous() - map cont' values to visual ones
scale_*_discrete() - map discrete values to visual ones
scale_*_identity() - use data values **as** visual ones
scale_*_manual(values = c()) - map discrete values to manually chosen visual ones
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks") - treat data values as dates.
scale_*_datetime() - treat data x values as date times. Use same arguments as scale_x_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)



n <- d + geom_bar(aes(fill = fl))
n + scale_fill_brewer(palette = "Blues")
For palette choices:
RColorBrewer::display.brewer.all()
n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = ..x..))
o + scale_fill_distiller(palette = "Blues")
o + scale_fill_gradient(low="red", high="yellow")
o + scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 25)
o + scale_fill_gradientn(colours=topo.colors(6))
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system

r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

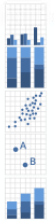
r + coord_polar(theta = "x", direction=1)
theta, start, direction
Polar coordinates

r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0))projection, orienztation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.



s <- ggplot(mpg, aes(fl, fill = drv))
s + geom_bar(position = "dodge")
Arrange elements side by side
s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height
e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting
e + geom_label(position = "nudge")
Nudge labels away from points
s + geom_bar(position = "stack")
Stack elements on top of one another

Each position adjustment can be recast as a function with manual **width** and **height** arguments
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines
r + theme_gray()
Grey background (default theme)
r + theme_dark()
dark for contrast

r + theme_classic()
r + theme_light()
r + theme_linedraw()
r + theme_minimal()
Minimal themes
r + theme_void()
Empty theme

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(. ~ fl)
facet into columns based on fl

t + facet_grid(year ~ .)
facet into rows based on year

t + facet_grid(year ~ fl)
facet into both rows and columns

t + facet_wrap(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(drv ~ fl, scales = "free")
x and y axis limits adjust to individual facets
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(. ~ fl, labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(fl ~ ., labeller = label_bquote(alpha ^ .(fl)))

α^{c}	α^{d}	α^{e}	α^{p}	α^{r}

t + facet_grid(. ~ fl, labeller = label_parsed)

c	d	e	p	r

Labels

t + labs(**x** = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
<AES> = "New <AES> legend title")

Use scale functions to update legend labels

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom to place | manual values for geom's aesthetics

Legends

n + theme(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"

n + guides(fill = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

n + scale_fill_discrete(name = "Title",
labels = c("A", "B", "C", "D", "E"))
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)
t + coord_cartesian(
xlim = c(0, 100), ylim = c(10, 20))

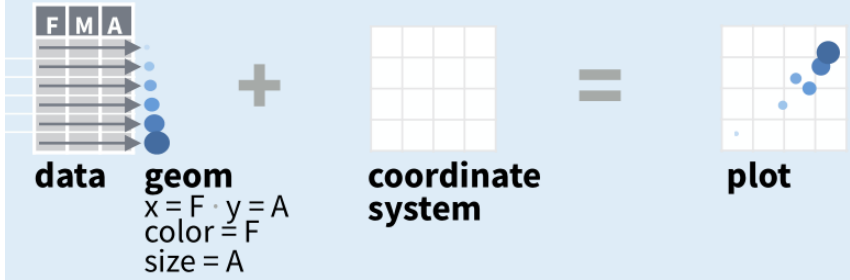
With clipping (removes unseen data points)
t + xlim(0, 100) **+ ylim**(10, 20)
t + scale_x_continuous(limits = c(0, 100)) **+**
scale_y_continuous(limits = c(0, 100))

RStudio

ggplot2 Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

ggplot (data = <DATA>**) +**
<GEOM_FUNCTION>**(mapping = aes(** <MAPPINGS>**),**
stat = **<STAT>** , position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

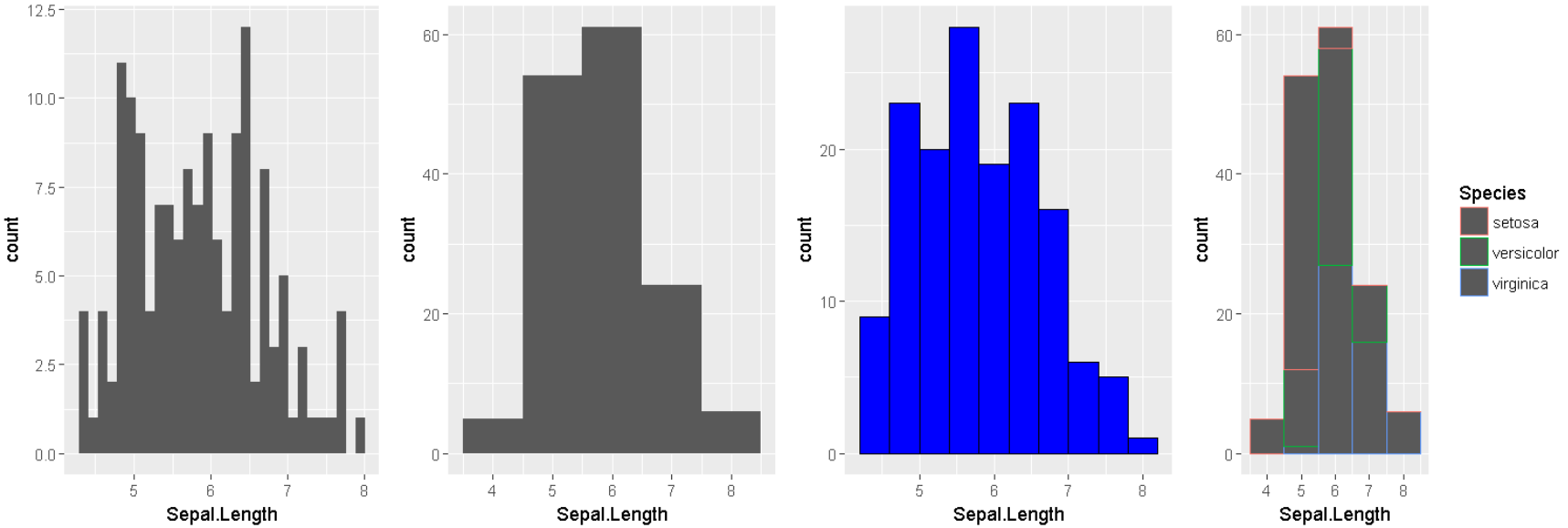
aesthetic mappings | data | geom

qplot(x = cty, y = hwy, data = mpg, geom = "point"**)** Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

geom_histogram()

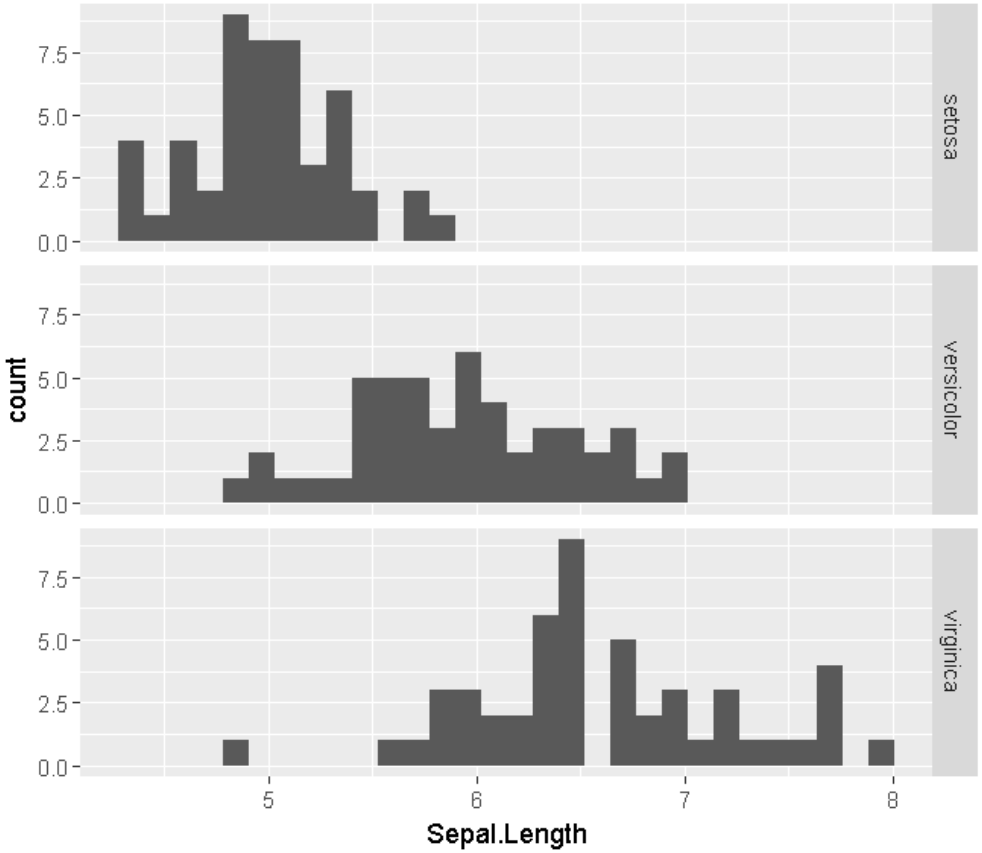
geoms: Use a geom function to represent data points, use the geom’ s aesthetic properties to represent variables.
Each function returns a layer.

```
> # install.packages("gridExtra")
> library(gridExtra)
> h1 <- ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram()
> h2 <- ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(binwidth=1)
> h3 <- ggplot(data=iris, aes(x=Sepal.Length)) +
        geom_histogram(color="black", fill="blue", bins = 10)
> h4 <- ggplot(data=iris, aes(x=Sepal.Length, color=Species)) + geom_histogram(binwidth = 1)
> grid.arrange(h1, h2, h3, h4, nrow=1, ncol=4)
```



geom_histogram()

```
> p <- ggplot(data=iris, aes(x=Sepal.Length))
> p <- p + geom_histogram()
> p + facet_grid(Species~.)   #row
```



mtcars {datasets}

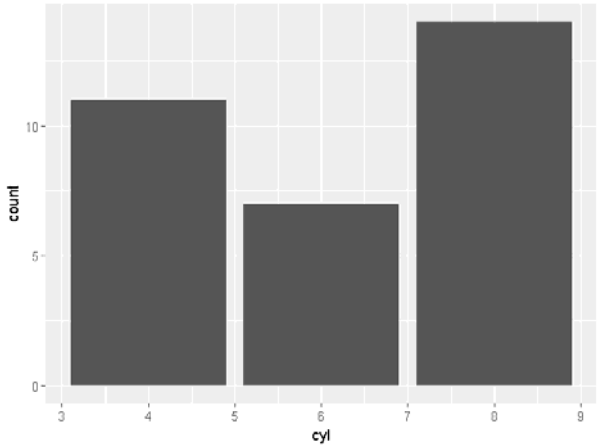
mtcars {datasets}: Motor Trend Car Road Tests, A data frame with 32 observations on 11 variables.

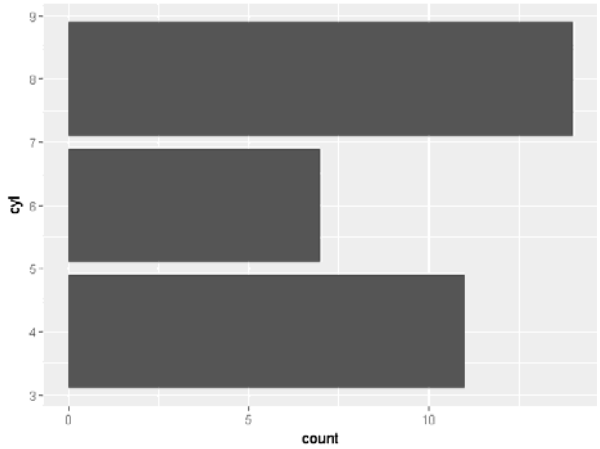
- mpg: Miles/(US) gallon (油耗)
- cyl: Number of cylinders (氣缸)
- disp: Displacement (cu.in.) (單汽缸排氣量)
- hp: Gross horsepower (馬力)
- drat: Rear axle ratio (後軸比)
- wt: Weight (1000 lbs)
- qsec: 1/4 mile time
- vs: V/S
- am: Transmission (0 = automatic, 1 = manual) (自手排)
- gear: Number of forward gears (前進檔的數量)
- carb: Number of carburetors (化油器的數量)

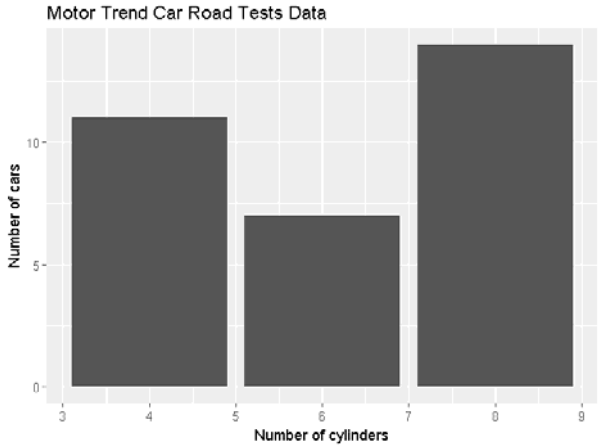
```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

geom_bar()

```
> p <- ggplot(mtcars, aes(x= cyl)) + geom_bar()
> p
> ggplot(mtcars, aes(x= cyl)) + geom_bar() + coord_flip()
> p + labs(title = "Motor Trend Car Road Tests Data",
         x = "Number of cylinders", y = "Number of cars")
```



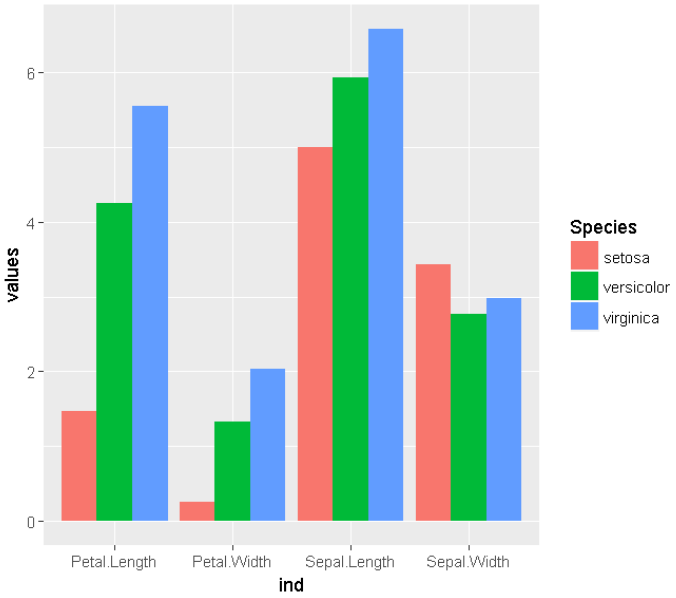




ggtitle("Main title"): Adds a main title above the plot
xlab("X axis label"): Changes the X axis label
ylab("Y axis label"): Changes the Y axis label
labs(title = "Main title", x = "X axis label", y = "Y axis label"): Changes main title and axis labels

geom_bar()

```
> iris.mean <- aggregate(iris[,1:4], by=list(Species=iris$Species), FUN=mean)
> iris.mean
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
> mydata <- cbind(stack(iris.mean[,-1]), Species = iris.mean$Species)
> mydata
   values          ind    Species
1   5.006 Sepal.Length     setosa
2   5.936 Sepal.Length versicolor
3   6.588 Sepal.Length  virginica
4   3.428  Sepal.Width     setosa
5   2.770  Sepal.Width versicolor
6   2.974  Sepal.Width  virginica
7   1.462 Petal.Length     setosa
8   4.260 Petal.Length versicolor
9   5.552 Petal.Length  virginica
10  0.246  Petal.Width     setosa
11  1.326  Petal.Width versicolor
12  2.026  Petal.Width  virginica
> ggplot(mydata, aes(x=ind, y=values, fill = Species)) +
+   geom_bar(position="dodge", stat="identity")
```



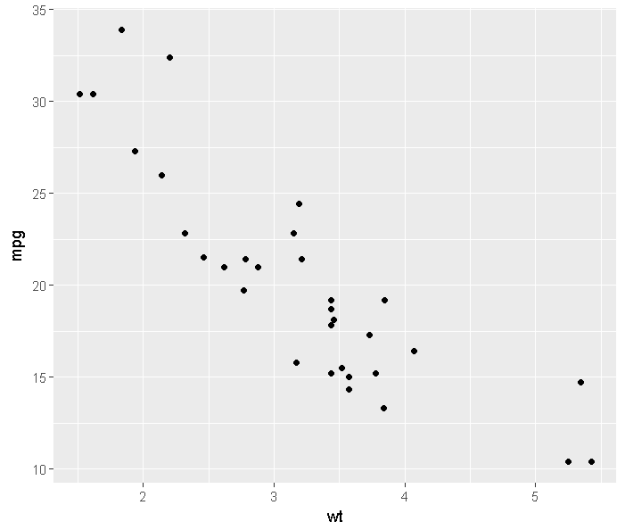
position_dodge for creating side-by-side barcharts.

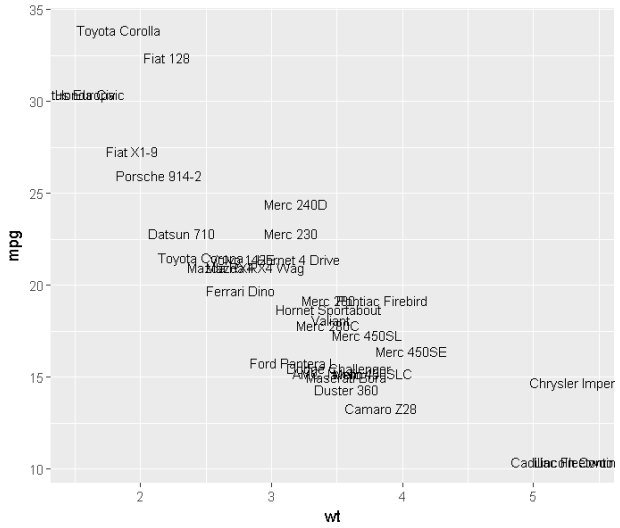
Other position adjustments: **position_identity, position_jitterdodge, position_jitter, position_nudge, position_stack**

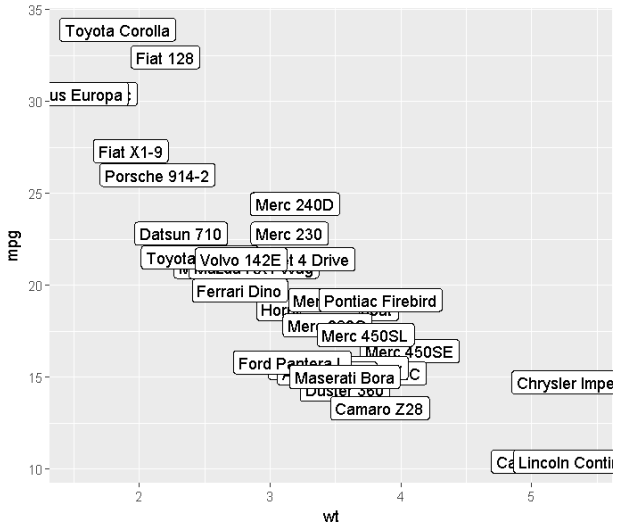
geom_text

```
> p <- ggplot(data=mtcars, aes(x=wt, y=mpg, label = rownames(mtcars)))
> p + geom_point()
> p + geom_text(size=3)
> p + geom_label()
```

- mpg: Miles/(US) gallon (油耗)
- wt: Weight (1000 lbs)



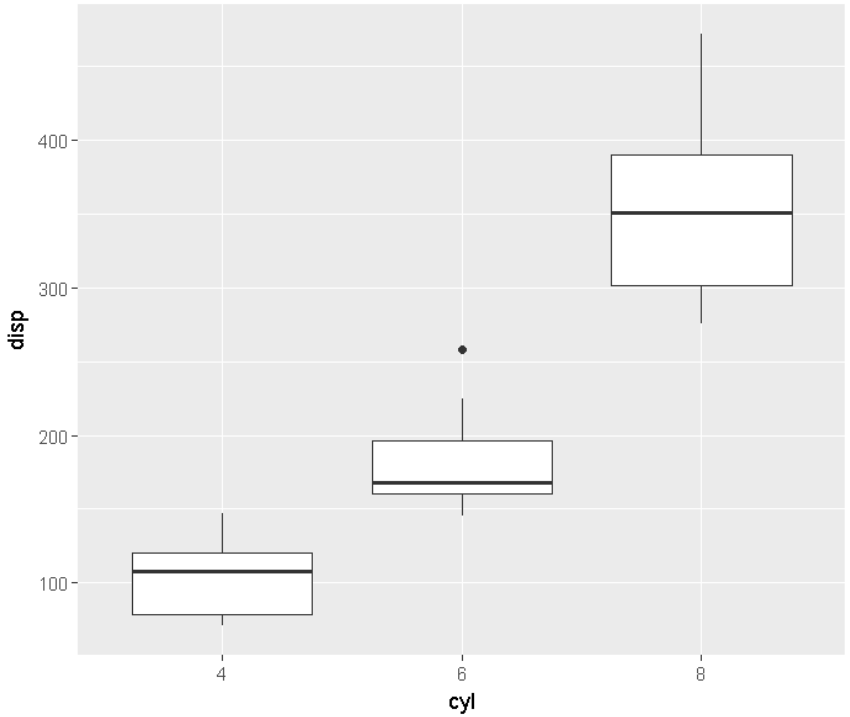




geom_text understands the following aesthetics (required aesthetics are in bold):
x, y, label, alpha, angle, colour, family, fontface, group, hjust, lineheight, size, vjust

geom_boxplot()

```
> mtcars$cyl <- factor(mtcars$cyl)
> ggplot(data=mtcars, aes(x=cyl, y=disp)) + geom_boxplot()
```



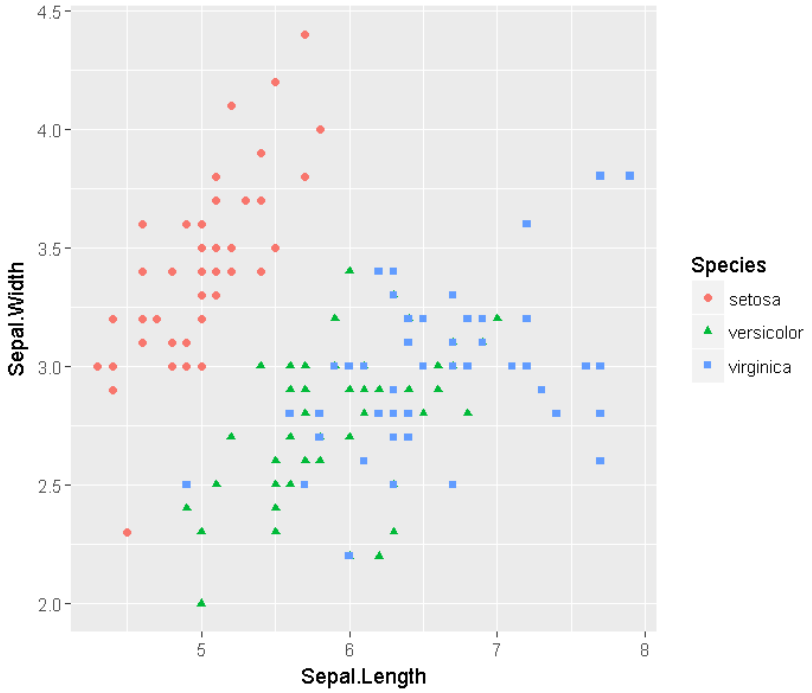
- cyl: Number of cylinders (氣缸)
- disp: Displacement (cu.in.) (單汽缸排氣量)

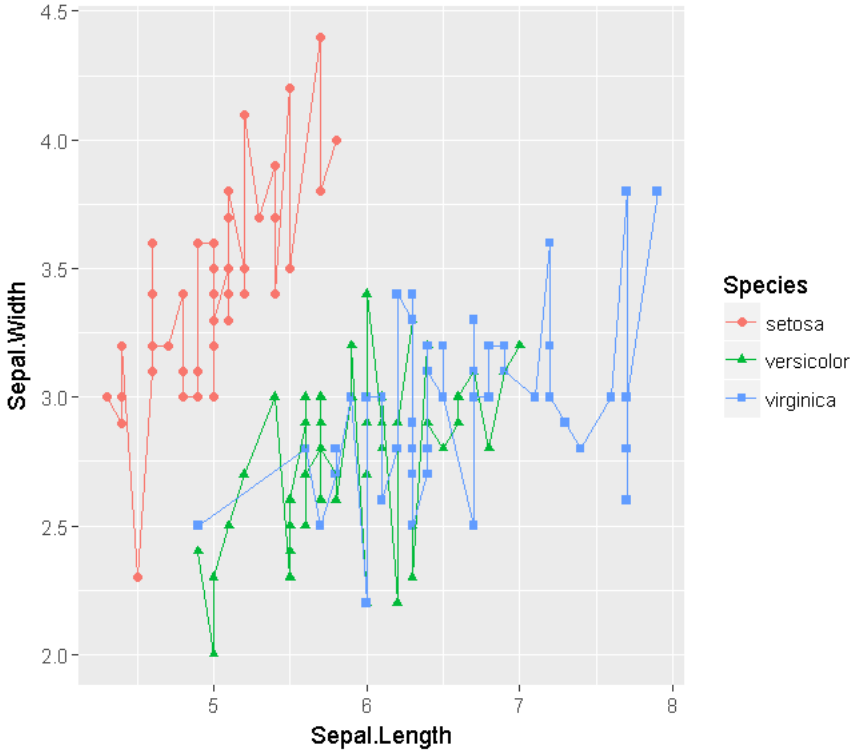
geom_boxplot understands the following aesthetics (required aesthetics are in bold):
x, lower, upper, middle, ymin, ymax, alpha, colour, fill, group, linetype, shape, size, weight

geom_point()

```
> ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, shape=Species, color=Species)) +
    geom_point()
```

```
> p <- ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, shape=Species, color=Species))
> p <- p + geom_point()
> p
> p + geom_line(aes(y=Sepal.Width))
```





使用Google Cloud Platform雲端服務

 # step1: Get API Key

https://cloud.google.com/maps-platform/

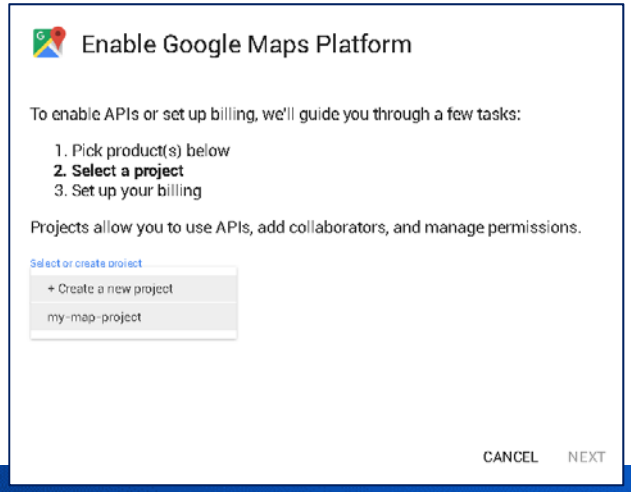
https://developers.google.com/maps/documentation/maps-static/get-api-key

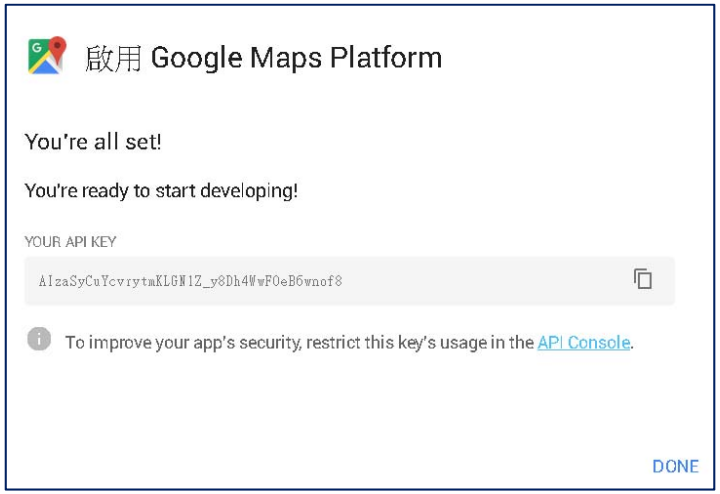
As of mid-2018, the Google Maps Platform requires a registered API key.

 # Step2: Enable Billing

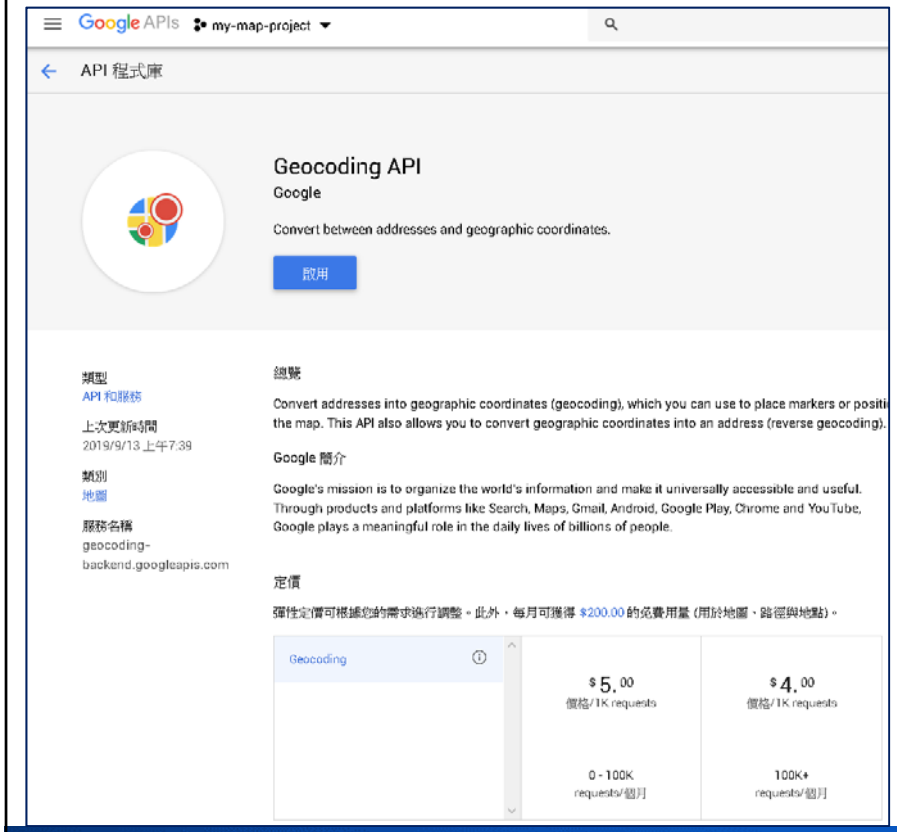
https://console.cloud.google.com/project/_/billing/enable

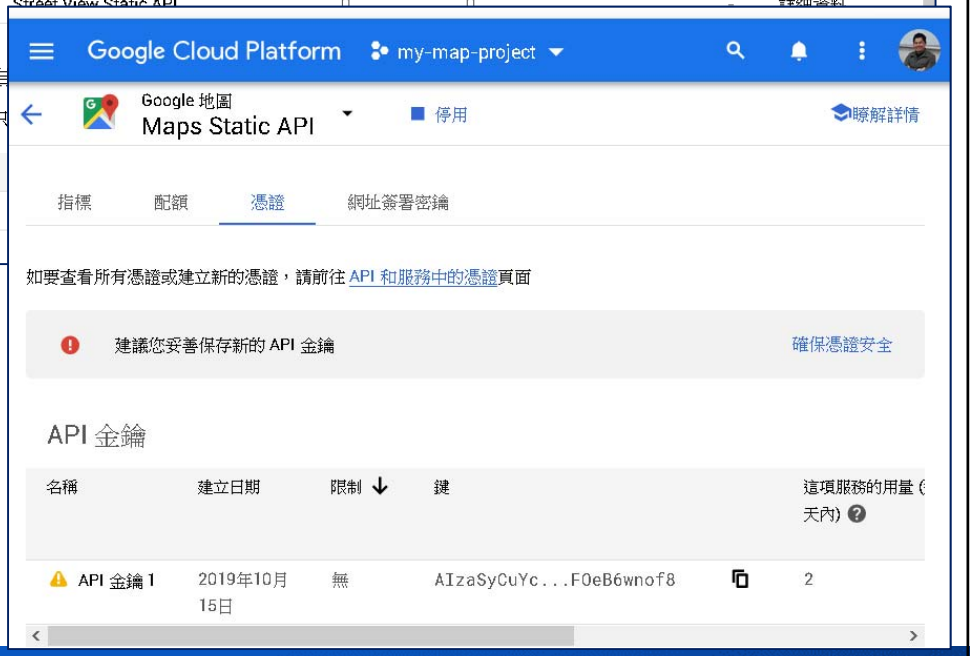
Google Cloud Platform





使用Google Cloud Platform雲端服務





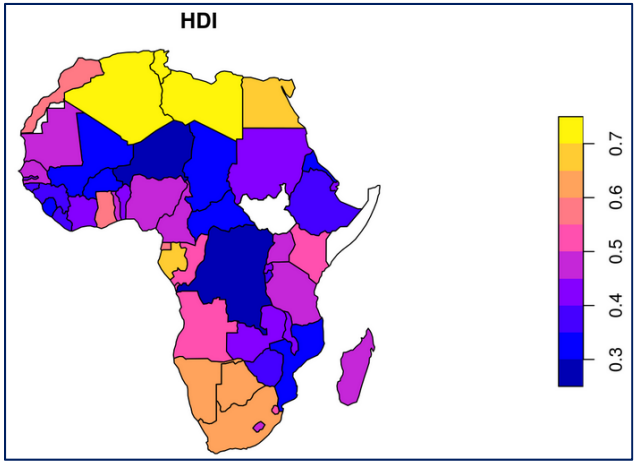
從RStudio測試是否已註冊GCP成功

```
> library(ggmap)
> register_google(key = "AIzaSyCuYcvrytmKLGNxxxxxxxx", write = TRUE)
Replacing old key (AIzaSyCuYcvrytmKLGNxxxxxx) with new key in
C:/Users/userpc/Documents/.Renviron
> tw.xy <- geocode("Taiwan")
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Taiwan&key=xxx
> tw.xy
# A tibble: 1 x 2
    lon   lat
  <dbl> <dbl>
1  121.  23.7
> tw.map <- get_map(location = 'Taiwan', zoom = 7, language = "zh-TW")
Source :
https://maps.googleapis.com/maps/api/staticmap?center=Taiwan&zoom=7&size=640x640&
scale=2&maptype=terrain&language=zh-TW&key=xxx
Source : https://maps.googleapis.com/maps/api/geocode/json?address=Taiwan&key=xxx
>

> has_google_key()
[1] TRUE
> google_key()
[1] "AIzaSyCuYcvrytmKLGNxxxxxx"
```

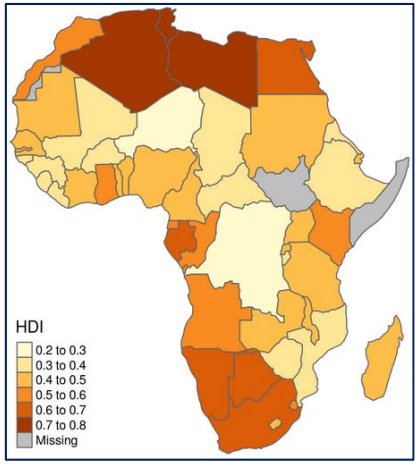
```
> tw.map <- get_map(location = 'Taiwan', zoom = 7, language = "zh-TW")
Error: Google now requires an API key.
       See ?register_google for details.
```

Making maps with R packages

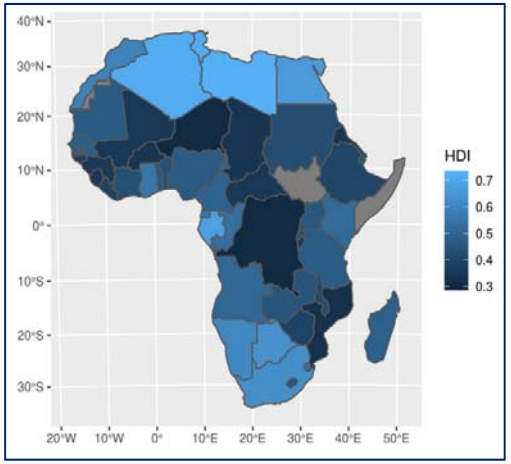
plot



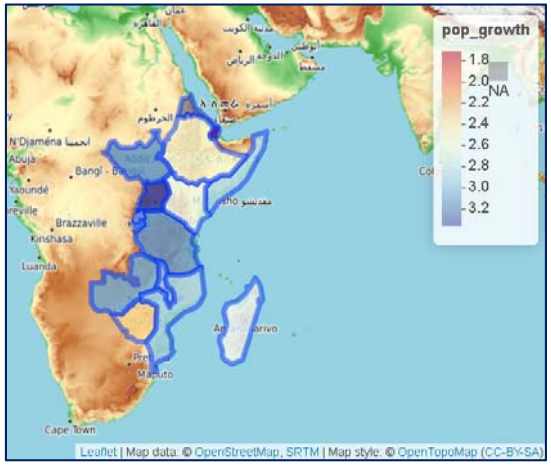
tmap



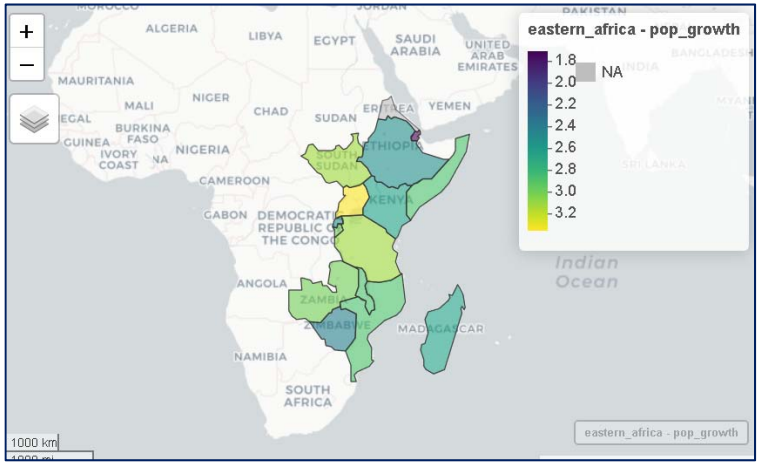
ggplot2



leaflet



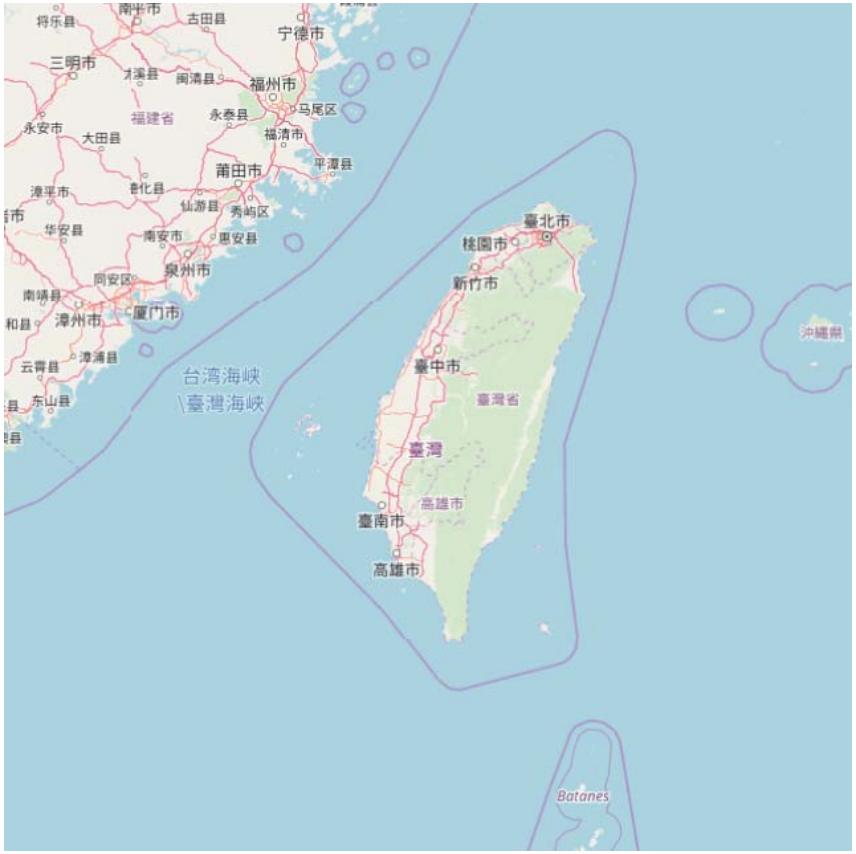
mapview



Chapter 8: Making maps with R, Robin Lovelace, Jakub Nowosad, Jannes Muenchow, 2019-08-30
https://geocompr.github.io/geocompkg/articles/solutions08.html

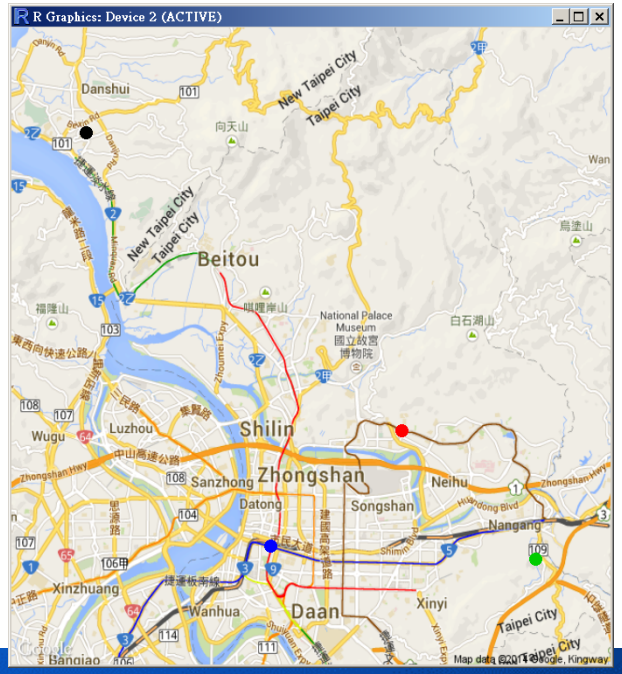
Plotting on Google Static Maps in R: RgoogleMaps

```
library(RgoogleMaps)
TaiwanMap <- GetMap(center=c(lat = 23.58, lon =120.58), zoom =7, destfile = "Taiwan1.png")
TaiwanMap <- GetMap(center=c(lat = 23.58, lon =120.58), zoom = 10, destfile = "Taiwan2.png",
maptype = "terrain")
```



於地圖上標記

```
> my.lat <- c(25.175339, 25.082288, 25.042185, 25.046254)
> my.lon <- c(121.450003, 121.565481, 121.614548, 121.517532)
> bb = qbbox(my.lat, my.lon)
> print(bb)
$latR
[1] 25.04152 25.17600

$lonR
[1] 121.4492 121.6154
> MyMap <- GetMap.bbox(bb$lonR, bb$latR, destfile = "my.png", maptype = "roadmap")
> My.markers <- cbind.data.frame(lat = my.lat, lon = my.lon)
> tmp <-  PlotOnStaticMap(MyMap, lat = My.markers[,"lat"], lon = My.markers[,"lon"], destfile
= "my.png", cex=2.5, pch=20, col=1:4, add=F)
```

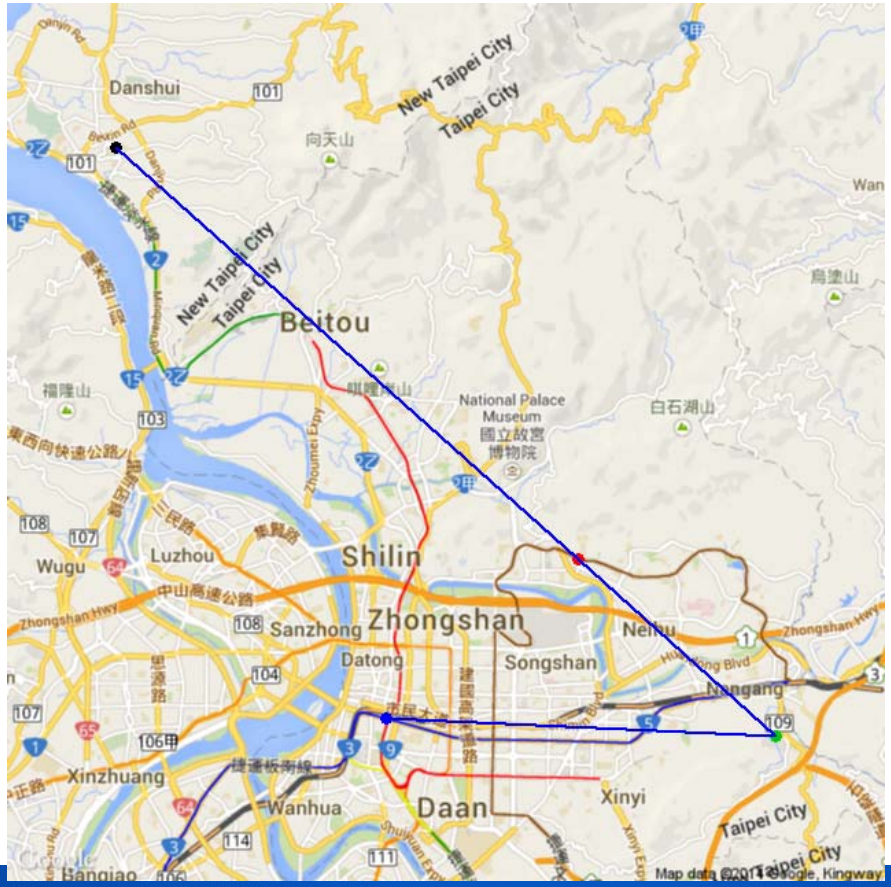


查詢經緯度
http://card.url.com.tw/realads/map_latlng.php

- 淡江大學 25.175339, 121.450003
- 台北市的地理中心位置: 內湖區環山路和內湖路一段跟基湖路口: 25.082288, 121.565481
- 中研院 25.042185, 121.614548
- 捷運台北站: 25.046254,121.517532

於地圖上標記

```
png("my2.png", 640, 640)
tmp <-  PlotOnStaticMap(MyMap, lat = My.markers[,"lat"], lon = My.markers[,"lon"], cex=2.5,
pch=20, col=1:4, add=F)
tmp <-  PlotOnStaticMap(MyMap, lat = My.markers[,"lat"], lon = My.markers[,"lon"], col="blue",
add=T, FUN = lines, lwd = 2)
dev.off()
```



於地圖上加文字，加圖片

```
> library(RgoogleMaps)
> my.lat <- c(25.175339, 25.14362, 24.942605)
> my.lon <- c(121.450003, 121.501768, 121.368381)
>
> bb = qbbox(my.lat, my.lon)
> print(bb)
$latR
[1] 24.94144 25.17650

$lonR
[1] 121.3677 121.5024
> MyMap <- GetMap.bbox(bb$lonR, bb$latR,
+          destfile = "my.png", maptype = "roadmap")
>
> My.markers <- cbind.data.frame(lat = my.lat, lon = my.lon)
> tmp <- PlotOnStaticMap(MyMap, lat = My.markers[,"lat"],
+                                lon = My.markers[,"lon"],
+    destfile = "my.png", cex=2.5, pch=18:10, col=1:3, add=F)
> TextOnStaticMap(MyMap, lat = My.markers[,"lat"]+0.01,
+                        lon = My.markers[,"lon"],
+  labels=c("我家", "復興高中", "國立臺北大學三峽校區"),  add=T)
```

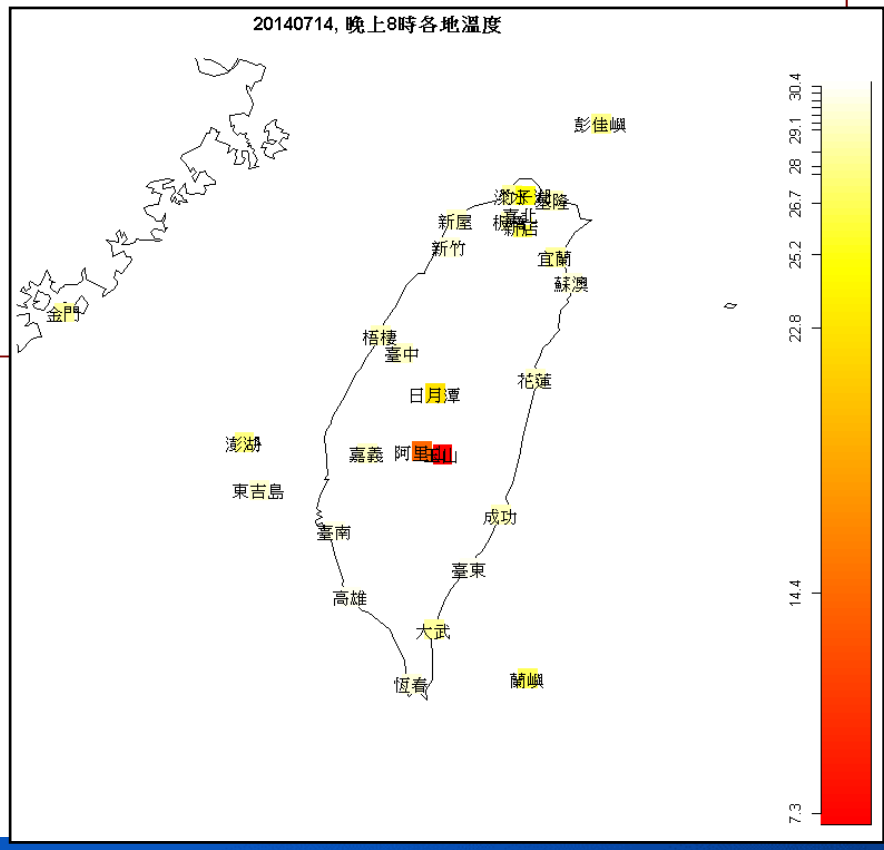

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("EBImage")
```

```
> library(EBImage)
> ntpu <- readImage("NTPUcolorlogo.jpg")
> loc <- LatLon2XY.centered(MyMap, lat=My.markers[3, 1], lon=My.markers[3, 2])
> rasterImage(ntpu, loc[[1]], loc[[2]]+30, loc[[1]]+50, loc[[2]]+80)
> Fuxing <- readImage("Fuxinglogo.jpg")
> loc <- LatLon2XY.centered(MyMap, lat=My.markers[2, 1], lon=My.markers[2, 2])
> rasterImage(Fuxing, loc[[1]], loc[[2]]+30, loc[[1]]+50, loc[[2]]+80)
```

Package: maps, mapdata

氣象局開放資料平台 http://opendata.cwb.gov.tw/

```
library(maps); library(maptools); library(mapdata); library(mapproj)
layout(matrix(c(1,1,1,0,2,0), ncol=2), widths=c(10, 1), heights=c(1, 10, 1))
map("world2Hires", xlim=c(118, 123), ylim=c(21, 26))
data <- read.table("20140714-weather.txt", sep="\t", header=TRUE, row.names=1)
x <- data$TEMP
tm <- floor((100-1)/(max(x)-min(x))*(x-min(x)) + 1)
used.col <- heat.colors(100)[tm]
points(data$lon, data$lat, pch=15, col=used.col)
text(data$lon, data$lat, labels=row.names(data))
title("20140714, 晚上8時各地溫度")
par(mar=c(1,1,1,1))
image(t(matrix(c(1:100), ncol=1)),
      col=heat.colors(100), xaxt="n", yaxt="n")
axis(LEFT <- 2, at=tm/100,
     labels=as.character(x), cex.axis=1)
```



See also:

Spatial data in R: Using R as a GIS

http://pakillo.github.io/R-GIS-tutorial/

ggmap: Spatial Visualization with ggplot2

ggmap quickstart

For more functionality, see ggmap documentation and https://dl.dropboxusercontent.com/u/24648660/ggmap%20useR%202012.pdf

There are 2 basic steps to making a map using ggmap:

Part 1: Download map raster → Part 2: Plot raster and overlay data

Part 1: Downloading the map raster

Start by loading the package: library(ggmap)

1. Define location: 3 ways

- location/address
  myLocation <- "University of Washington"
- lat/long
  myLocation <- c(lon = -95.3632715, lat = 29.7632836)
- bounding box lowerleftlon, lowerleftlat, upperrightlon, upperrightlat (a little glitchy for google maps)
  myLocation <- c(-130, 30, -105, 50)

Convert location/address its lat/long coordinates:
geocode("University of Washington")

2. Define map source, type, and color

The get_map function provides a general approach for quickly obtaining maps from multiple sources. I like this option for exploring different styles of maps.

There are 4 map "sources" to obtain a map raster, and each of these sources has multiple "map types" (displayed on right).

- stamen: maptype = c("terrain", "toner", "watercolor")
- google: maptype = c("roadmap", "terrain", "satellite", "hybrid")
- osm: open street map
- cloudmade: 1000s of maps, but an api key must be obtained from http://cloudmade.com

```
myMap <- get_map(location=myLocation,
source="stamen", maptype="watercolor", crop=FALSE)
ggmap(myMap)
```

This will produce a map that looks something like this

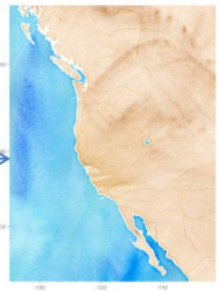
NOTE: crop = FALSE because otherwise, with stamen plots, the map is slightly shifted when I overlay data.

3. Fine tune the scale of the map using zoom

The get_map function takes a guess at the zoom level, but you can alter it:
zoom = integer from 3-21
3 = continent, 10=city, 21=building
(openstreetmap limit of 18)

The following maps show different map source/type options (except cloudmade)
The appearance of these maps may be very different depending on zoom/scale

stamen: watercolor



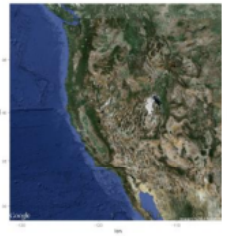
stamen: toner

stamen: terrain

If you can't get the map you want by adjusting the location/zoom variables, the functions designed for the different map sources provide more options:
get_googlemap,
get_openstreetmap,
get_stamenmap,
get_cloudmademap

google: terrain

google: satellite



google: roadmap

google: hybrid

osm*

*Open street maps may return a '503 Service Unavailable' error. This means their server is unavailable, and you must wait for it to become available again.

All maps can be displayed in black and white
color = "bw"

myMap <-
get_map(location=myLocation,
source="osm", color="bw"))

- If you use Rstudio: Sometimes a plot will not display. Increasing the size of the plot window may help. dev.off() prior to plotting may also help.
- The urlonly = TRUE will return a url that will display your map in a web browser. Which is pretty cool and may be handy!
- legend="topleft" will inset the legend on the top left of the map is data is overlayed (page 2).

Page 1 ggmap, Melanie Frazier

ggmap: Spatial Visualization with ggplot2

ggmap quickstart

Part 2: Plotting the maps and data

1. Plot the raster:

```
ggmap(myMap)
```

2. Add points with latitude/longitude coordinates:

```
ggmap(myMap)+
geom_point(aes(x = Longitude, y = Latitude), data = data,
        alpha = .5, color="darkred", size = 3)
```

alpha = transparency
color = color
size = size of points

The size, color, alpha, and shape of the points can be scaled relative to another variable (in this case estArea) within the aes function:

```
ggmap(myMap)+
geom_point(aes(x = Longitude, y = Latitude, size=sqrt(estArea)),
data = data, alpha = .5, color="darkred")
```

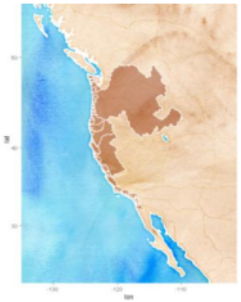
Additional functions can be added to control scaling, e.g.:

```
ggmap(myMap)+
  geom_point(aes(x = Longitude, y = Latitude, size=sqrt(estArea)),
  data = data, alpha = .5, color="darkred")+
  scale_size(range=c(3,20))
```

3. Add polygons from shp file

The shp file is imported into R using the rgdal package, and must be transformed to geographic coordinates (latitude/longitude) on the World Geodetic System of 1984 (WGS84) datum using the rgdal package:

```
library(rgdal)
shpData <- readOGR(dsn="C:\\Documents and Settings\\Watershed", layer="WS")
proj4string(shpData)    # describes data's current coordinate reference system
# to change to correct projection:
shpData <- spTransform(shpData,
CRS("+proj=longlat +datum=WGS84"))
```



To plot the data:

```
geom_polygon(aes(x = long, y = lat, group=id),
data = shpData, color ="white", fill ="orangered4",
alpha = .4, size = .2)
```

color= outline color alpha = transparency
fill = polygon color size = outline size

4. Annotate figure

```
baylor <- get_map('baylor university', zoom = 15, maptype = 'satellite')
ggmap(baylor) +
 annotate('rect', xmin=-97.11, ymin=31.54, xmax=-97.12, ymax=31.55, col="red", fill="white")+
 annotate('text', x=-97.12, y=31.54, label = 'Statistical Sciences', colour = I('red'), size = 8)+
 annotate('segment', x=-97.12, xend=-97.12, y=31.55, yend=31.55,
       colour=I('red'), arrow = arrow(length=unit(0.3,"cm")), size = 1.5) +
 labs(x = 'Longitude', y = 'Latitude') + ggtitle('Baylor University')
```

Controlling size and color

size
- scale_size(range = c(3, 20))
- But, the following is better for display because it is based on area (rather than radius)
- scale_area(range=c(3,20))

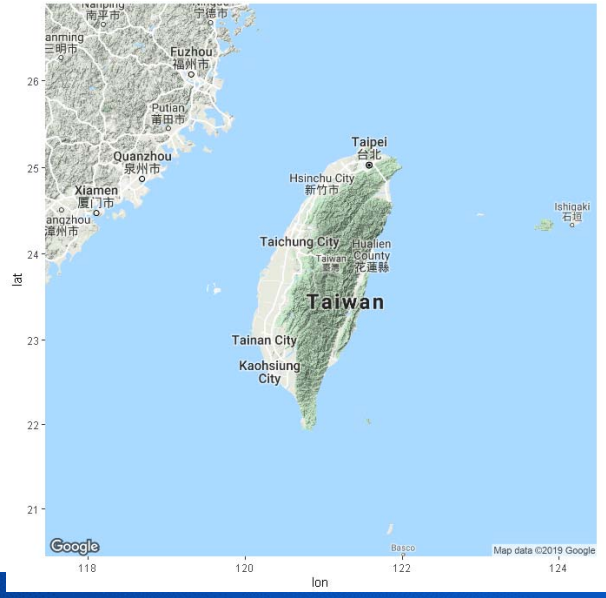
color
- Continuous variables: color gradient between n colors, e.g.:
- scale_colour_gradientn(colours = rainbow_hcl(7))
- Discrete variables, e.g.:
- scale_colour_manual(values=rainbow_hcl(7))
- scale_colour_manual(values=c("8" = "red", "4" = "blue","6" = "green")
- *Use colorspace and RColorBrewer to choose color combinations

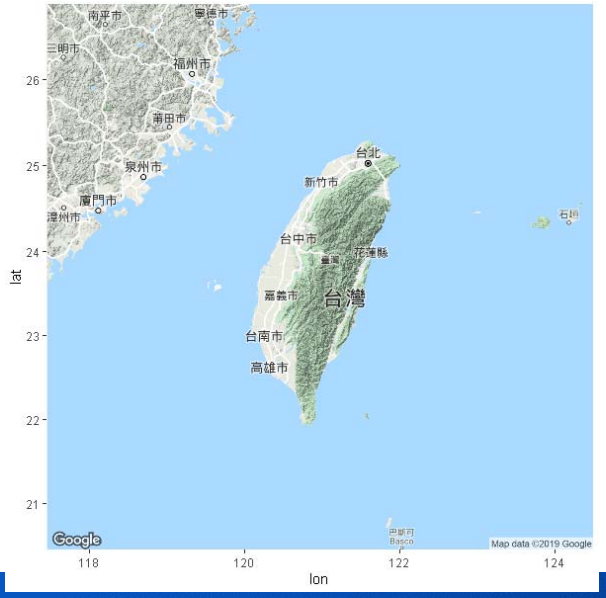
Page 2 ggmap, Melanie Frazier

以ggmap獲取google提供的台灣地圖

ggmap: A collection of functions to visualize spatial data and models on top of static maps from various online sources (e.g Google Maps and Stamen Maps). It includes tools common to those tasks, including functions for geolocation and routing.

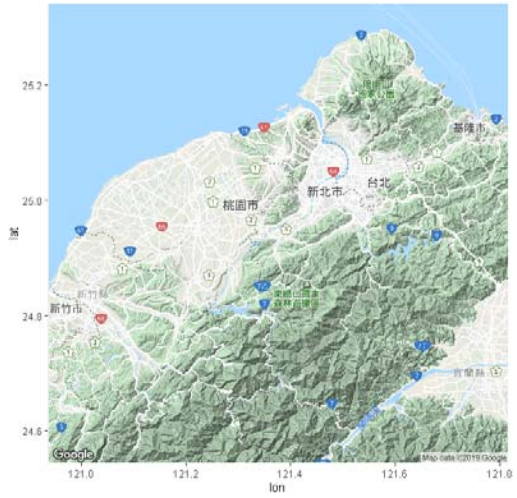
```
> library(ggmap)
> library(mapproj)
> tw.map <- get_map(location = 'Taiwan', zoom = 7)
Source :
https://maps.googleapis.com/maps/api/staticmap?center=Taiwan&zoom=7&size=640x640&scale=2&map
type=terrain&language=en-EN&key=AIzaSyCuYcvrytmKLGN
Source :
https://maps.googleapis.com/maps/api/geocode/json?address=Taiwan&key=AIzaSyCuYcvrytmKLGN
> ggmap(tw.map)
> tw.map.zh <- get_map(location = 'Taiwan', zoom = 7, language = "zh-TW")
> ggmap(tw.map.zh)
```

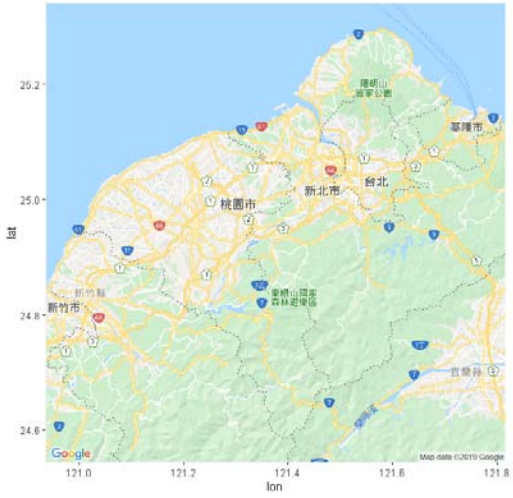




不同的地圖呈現方式

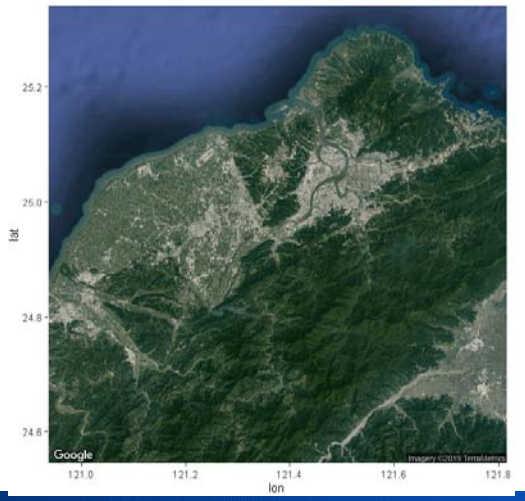
```
> tw.map.ntpu <- get_map(location = c(lon = 121.374925, lat = 24.943403),
                         zoom = 10, language = "zh-TW" , maptype = "terrain")
> ggmap(tw.map.ntpu)
```

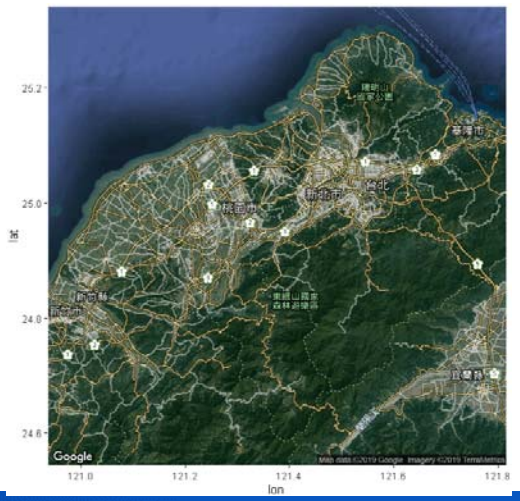


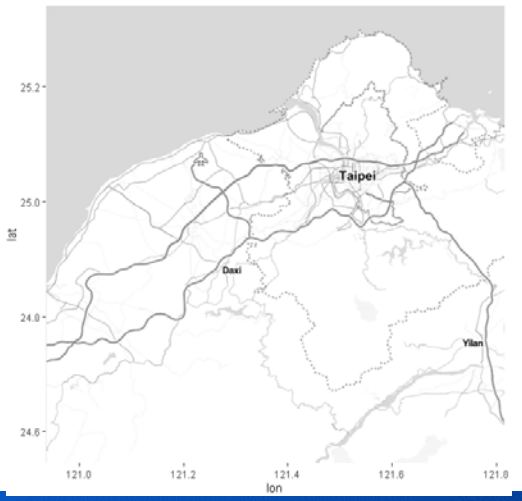


roadmap
satellite
hybrid
toner-lite

國立臺北大學三峽校區.
新北市三峽區大學路151號.
經緯度：(121.374925,24.943403)







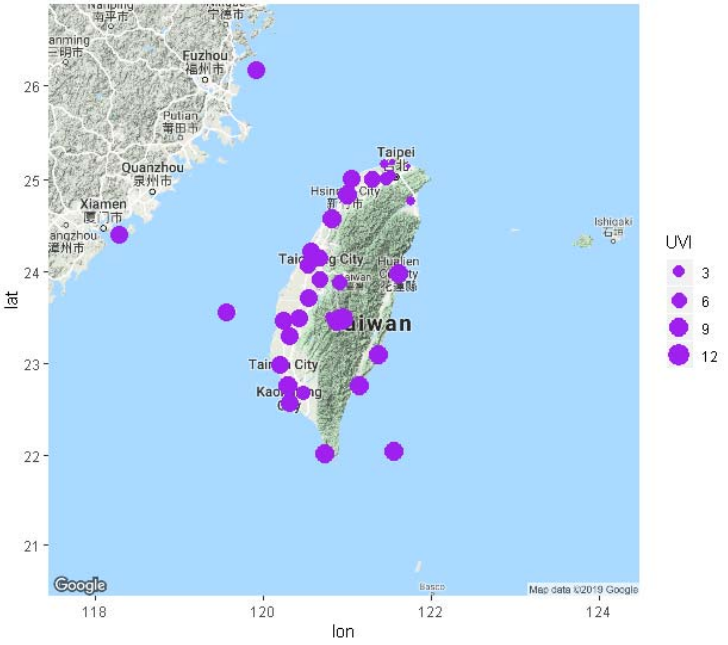
下載台灣各地紫外線即時監測資料

```
uv <- read.csv("data/UV_20191016130309.csv")
head(uv)

# 經緯度(度分秒) => 度數
lon.deg <- sapply((strsplit(as.character(uv$WGS84Lon), ",")), as.numeric)
lon.deg
uv$lon <- lon.deg[1, ] + lon.deg[2, ]/60 + lon.deg[3, ]/3600
lat.deg <- sapply((strsplit(as.character(uv$WGS84Lat), ",")), as.numeric)
uv$lat <- lat.deg[1, ] + lat.deg[2, ]/60 + lat.deg[3, ]/3600

tw.map <- get_map(location = 'Taiwan', zoom = 7)
ggmap(tw.map) +
  geom_point(data = uv, aes(x = lon, y = lat, size = UVI), color="purple")
```

https://data.gov.tw/dataset/6076

```
> head(uv)
  County PublishAgency      PublishTime SiteName   UVI WGS84Lat  WGS84Lon
1 花蓮縣     中央氣象局 2019-10-16 12:00     花蓮  9.34 23,58,30 121,36,48 121.61
2 連江縣     中央氣象局 2019-10-16 12:00     馬祖  8.14 26,10,09 119,55,24 119.92
3 高雄市     中央氣象局 2019-10-16 12:00     高雄  7.77 22,33,58 120,18,57 120.31
4 南投縣     中央氣象局 2019-10-16 12:00     玉山 12.11 23,29,15 120,57,34 120.95
5 臺南市     中央氣象局 2019-10-16 12:00     臺南  8.00 22,59,36 120,12,17 120.20
6 新竹縣     中央氣象局 2019-10-16 12:00     新竹  8.74 24,49,40 121,00,51 121.01
```

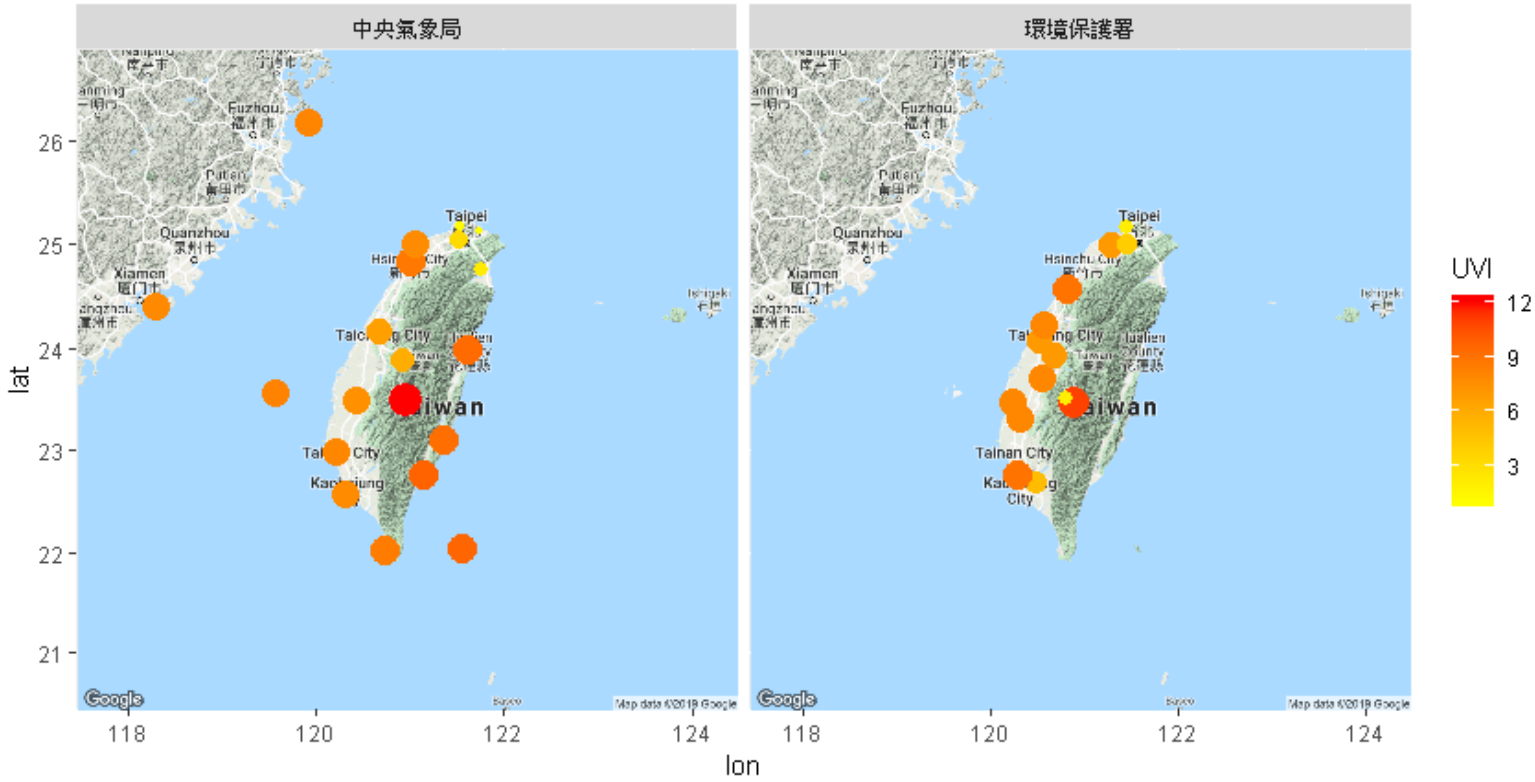
```
> lon.deg
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]  121  119  120  120  120  121  121  120  121   121   121   121
[2,]   36   55   18   57   12    0   31   44   30    22    44     2
[3,]   48   24   57   34   17   51   47   47   53    24    26    51
      [,21]  [,22] [,23]  [,24]  [,25]  [,26]  [,27] [,28]  [,29]  [
[1,] 120.00 120.00 120.0 120.00 120.00 120.00 120.00 120.0 120.00 12
[2,]  29.00  18.00  19.0  14.00  52.00  48.00  32.00  41.0  32.00  3
[3,]  16.92  20.48   2.1  52.12  50.06   5.02  41.98   7.1  29.47
```



參考: https://blog.gtwang.org/r/r-ggmap-package-spatial-data-visualization/2/

繪製台灣各地紫外線指數地圖

```
ggmap(tw.map) +
  geom_point(data = uv, aes(x = lon, y = lat, size = UVI, color = UVI)) +
  scale_color_continuous(low = "yellow", high = "red") +
  facet_grid(~PublishAgency) +
  guides(size = FALSE)
```

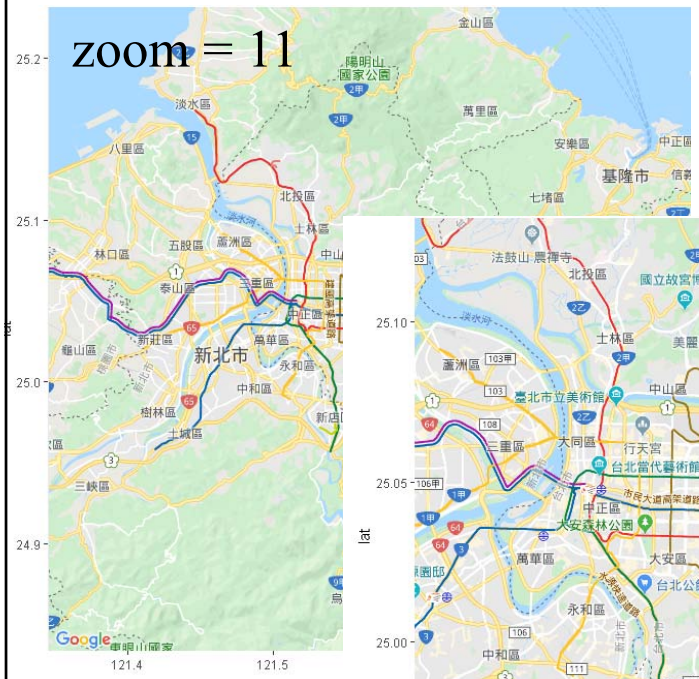


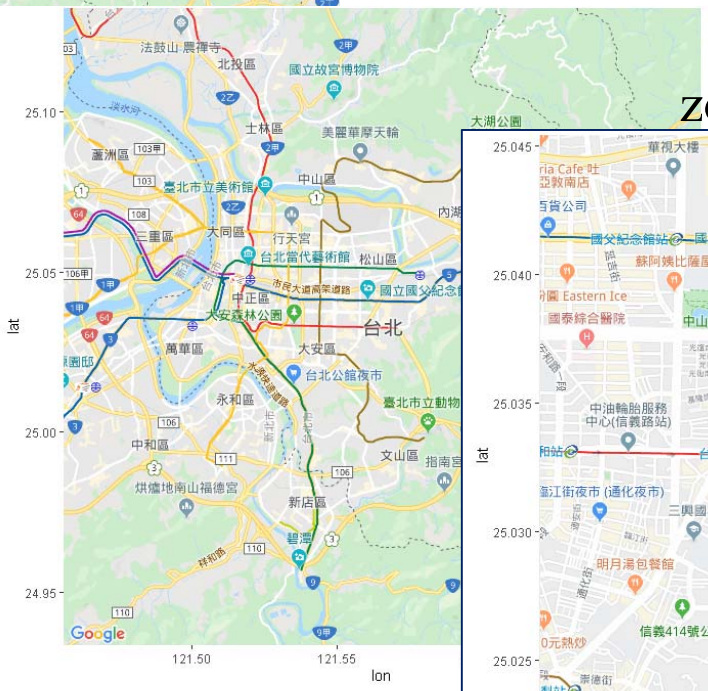
讀取中文檔問題:

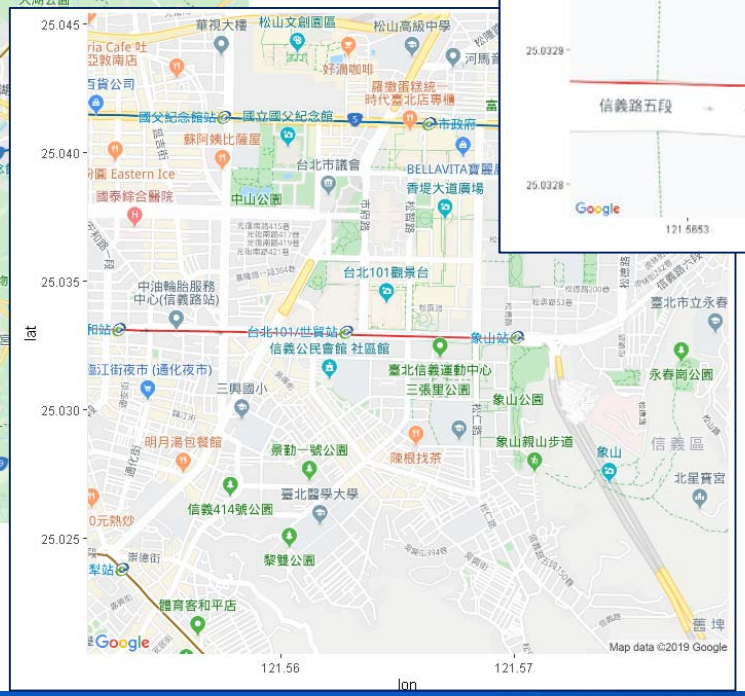
```
Sys.setlocale(category = "LC_ALL", locale = "zh_TW.big5")
```

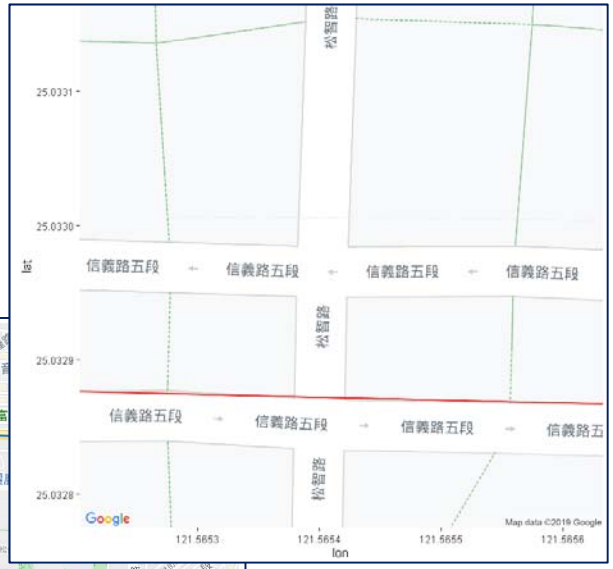
不同放大比例(zoom)的台北市地圖

```
tpe.map.zh11 <- get_map(location = 'Taipei', zoom = 11, maptype = "roadmap",
  language = "zh-TW")
ggmap(tpe.map.zh11)
```









map zoom: an integer from 3 (continent) to 21 (building), default value 10 (city).

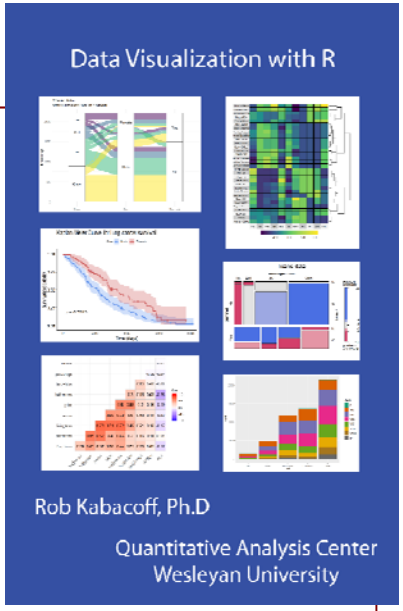
Dot Density Maps

```
> head(crime)
                  time     date hour premise           offense  beat     block    street type suffix number   month
82729 2010-01-01 14:00:00 1/1/2010    0     18A           murder 15E30 9600-9699   marlive   ln       -      1 january
82730 2010-01-01 14:00:00 1/1/2010    0     13R           robbery 13D10 4700-4799 telephone   rd       -      1 january
82731 2010-01-01 14:00:00 1/1/2010    0     20R aggravated assault 16E20 5000-5099  wickview   ln       -      1 january
82732 2010-01-01 14:00:00 1/1/2010    0     20R aggravated assault  2A30 1000-1099   ashland   st       -      1 january
82733 2010-01-01 14:00:00 1/1/2010    0     20A aggravated assault 14D20 8300-8399    canyon             -      1 january
82734 2010-01-01 14:00:00 1/1/2010    0     20R           burglary 18F60 9300-9399     rowan   ln       -      1 january
        day                  location           address       lon      lat
82729 friday    apartment parking lot   9650 marlive ln -95.43739 29.67790
82730 friday road / street / sidewalk 4750 telephone rd -95.29888 29.69171
82731 friday        residence / house  5050 wickview ln -95.45586 29.59922
82732 friday        residence / house   1050 ashland st -95.40334 29.79024
82733 friday                apartment       8350 canyon -95.37791 29.67063
82734 friday        residence / house     9350 rowan ln -95.54830 29.70223
```

```
> # install.packages("ggmap")
> library(ggmap)
> head(crime)
> dim(crime)
[1] 86314    17
```

The crime dataset from the **ggmap** package, contains the time, date, and location of six types of crimes in Houston, Texas between January 2010 and August 2010.

```
> summary(crime$offense)
aggravated assault  auto theft    burglary   murder  rape  robbery theft
              7177        7946       17802      157     378     6298    46556
>
> library(dplyr)
> rapes <- filter(crime, offense == "rape") %>%
+   select(date, offense, address, lon, lat)
> head(rapes)
      date offense             address       lon      lat
1 1/1/2010    rape   5950 glenmont dr -95.48498 29.72007
2 1/1/2010    rape    2350 sperber ln -95.34817 29.75505
...
6 1/2/2010    rape   1150 fidelity st -95.25535 29.74147
```

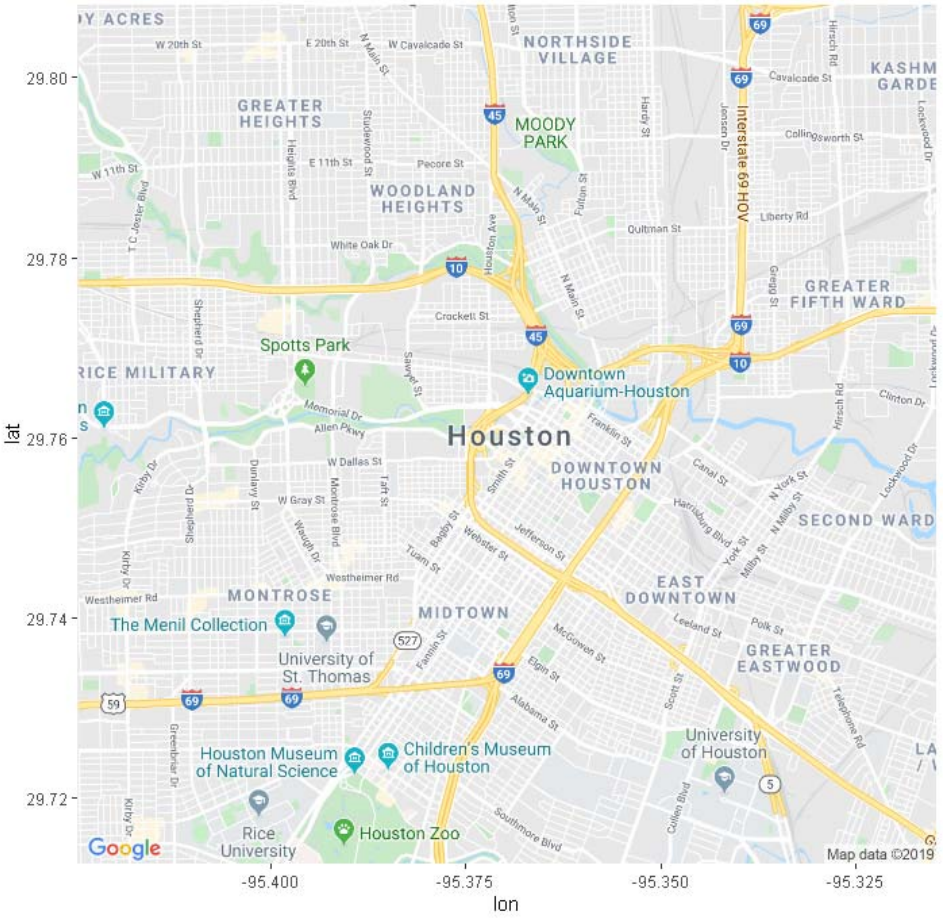


Get the background map image

```
> # (1) Find the center coordinates for Houston, TX
> houston_center <- geocode("Houston, TX")
Source :
https://maps.googleapis.com/maps/api/geocode/json?address=Houston,+TX&key=AIzaSyCuYcvrytmKLGN
> register_google(key = "AIzaSyCuYcvrytmKLGN1Z_y8Dh4WwFOeB6wnof8", write = TRUE)
Replacing old key (AIzaSyCuYcvrytmKLGN) with new key in C:/Users/userpc/Documents/.Renviron
> houston_center
# A tibble: 1 x 2
    lon   lat
  <dbl> <dbl>
1 -95.4  29.8
```

```
> has_google_key()
[1] TRUE
> google_key()
[1] "AIzaSyCuYcvrytmKLGN
```

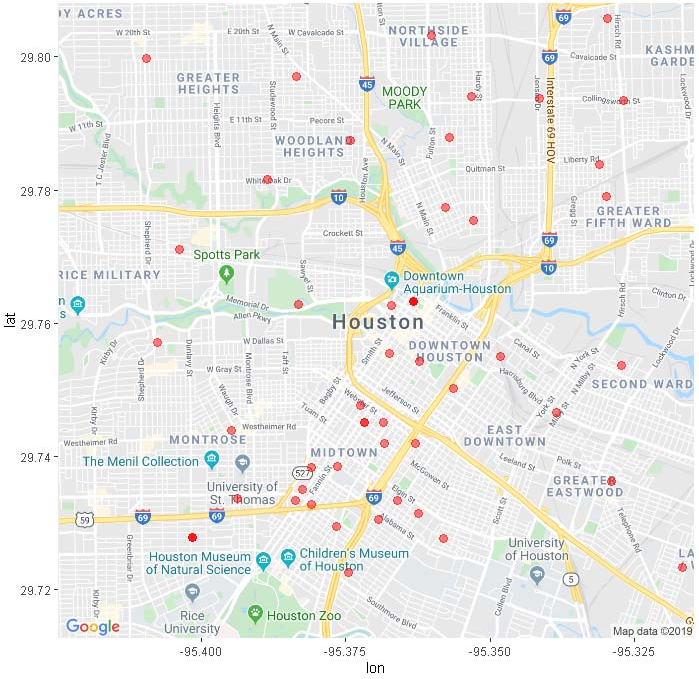
```
houston_map <- get_map(houston_center,
                       zoom = 13,
                       maptype = "roadmap")
ggmap(houston_map)
```

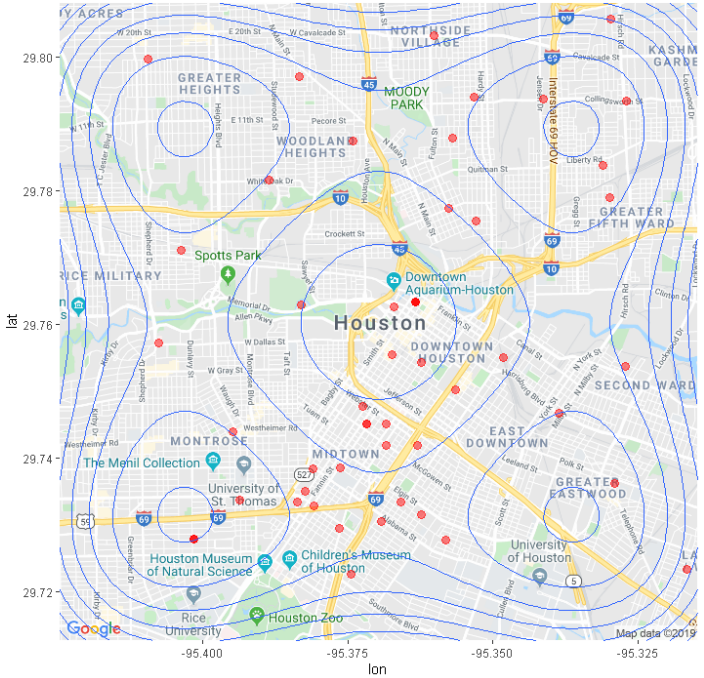


Add crime locations to the map

```
ggmap(houston_map,
      base_layer = ggplot(data = rapes, aes(x=lon, y = lat))) +
  geom_point(color = "red", size = 3, alpha = 0.5)
```

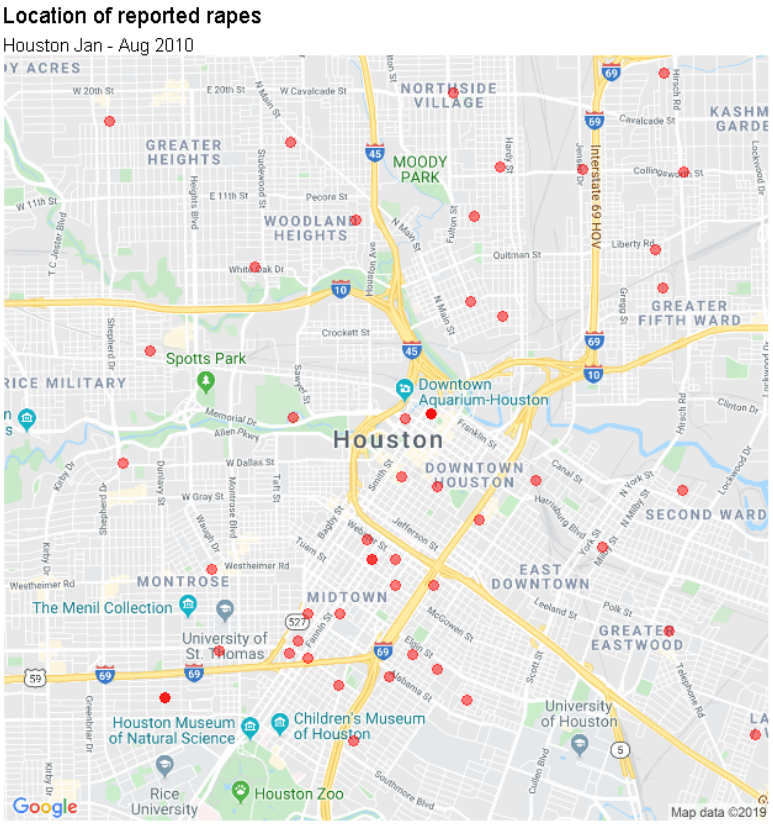
```
ggmap(houston_map) +
  geom_point(data = rapes, aes(x=lon, y = lat),
             color = "red", size = 3, alpha = 0.5) +
  geom_density2d(size = 0.3)
```





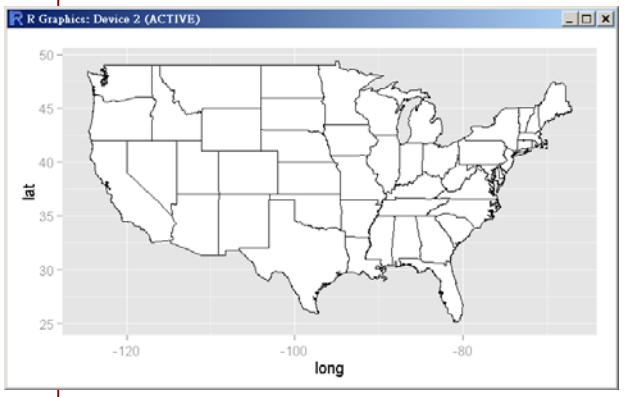
Remove long and lat numbers and add titles

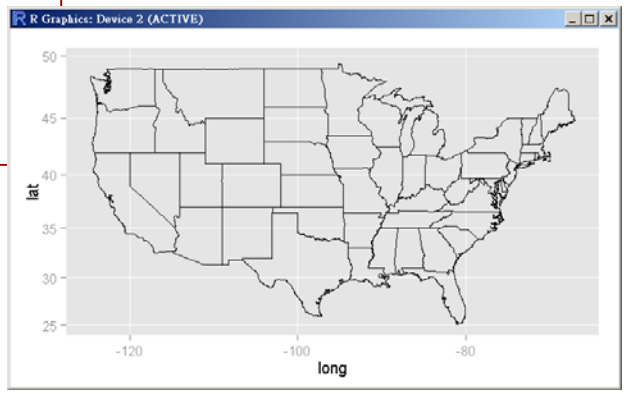
```
ggmap(houston_map,
      base_layer = ggplot(data = rapes, aes(x=lon, y = lat))) +
  geom_point(color = "red", size = 3, alpha = 0.5) +
  theme_void() +
  labs(title = "Location of reported rapes",
       subtitle = "Houston Jan - Aug 2010",
       caption = "source: http://www.houstontx.gov/police/cs/")
```



Creating a Map

```
> library(ggplot2)
> library(maps)
> library(mapproj)
> states.map <- map_data("state")
> head(states.map, 3)
       long      lat group order  region subregion
1 -87.46201 30.38968     1     1 alabama      <NA>
2 -87.48493 30.37249     1     2 alabama      <NA>
3 -87.52503 30.37249     1     3 alabama      <NA>
> tail(states.map, 3)
            long      lat group order  region subregion
15597 -107.9223 41.01805    63 15597 wyoming      <NA>
15598 -109.0568 40.98940    63 15598 wyoming      <NA>
15599 -109.0511 40.99513    63 15599 wyoming      <NA>

> ggplot(states.map, aes(x=long, y=lat, group=group)) +
    geom_polygon(fill="white", colour="black")

> ggplot(states.map, aes(x=long, y=lat, group=group)) +
    geom_path() + coord_map("mercator")
```





mercator: equally spaced straight meridians, conformal, straight compass courses

Source: 13.17. Creating a Map, R Graphics Cookbook 2nd

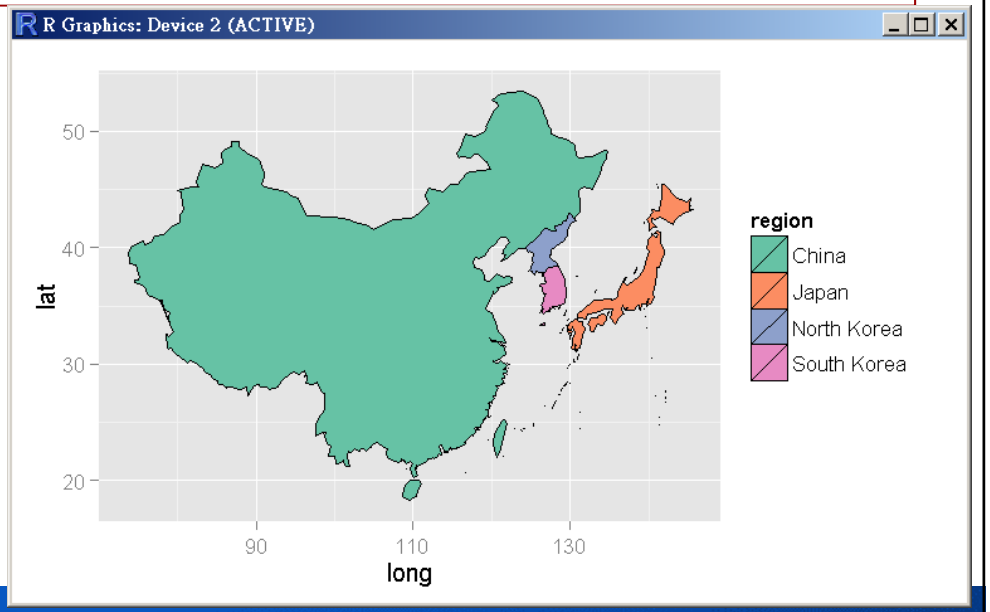
Creating a Map

```
> # map_data: county, france, italy, nz, state, usa, world, world2.
> # region names
> world.map <- map_data("world")
> sort(unique(world.map$region))
  [1] "Afghanistan"                "Albania"
  [3] "Algeria"                    "American Samoa"
  [5] "Andaman Islands"            "Andorra"
   ...
> east.asia <- map_data("world", region=c("Japan", "China", "North Korea",
"South Korea"))
> ggplot(east.asia, aes(x=long, y=lat, group=group, fill=region)) +
    geom_polygon(colour="black") +
    scale_fill_brewer(palette="Set2")
```

NOTE:

See the **mapdata** package for more map data sets. It includes maps of China and Japan, as well as a high-resolution world map, **worldHires**.

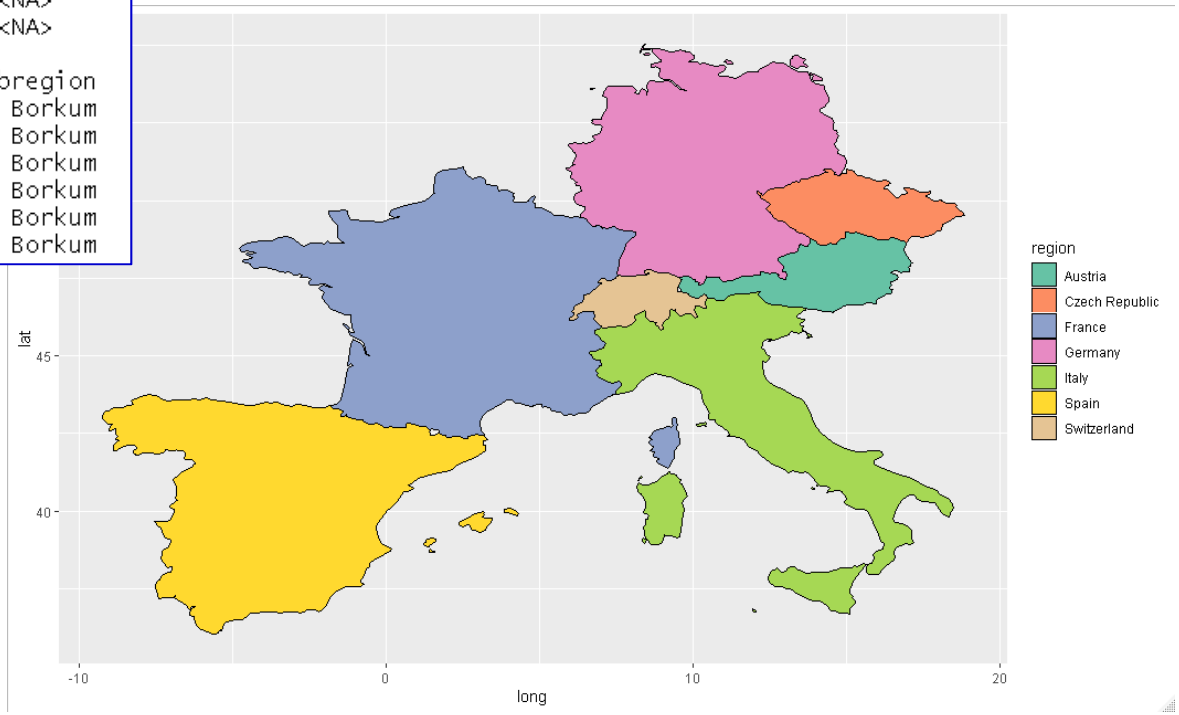
See Also: **mapproject, map**



分級著色圖 /面量圖(Choropleth Maps)

```
> country <- c("France", "Austria", "Italy", "Switzerland", "Germany", "Spain", "Czech
Republic")
> mymapdata <- map_data("world", region = country)
> ggplot(mymapdata, aes(x = long, y = lat, group = group, fill = region)) +
   geom_polygon(colour = "black") +
   scale_fill_brewer(palette = "Set2")
```

```
> head(mymapdata)
      long      lat group order  region subregion
1 16.95312 48.59883     1     1 Austria      <NA>
2 16.94883 48.58858     1     2 Austria      <NA>
3 16.94336 48.55093     1     3 Austria      <NA>
4 16.90449 48.50352     1     4 Austria      <NA>
5 16.86270 48.44141     1     5 Austria      <NA>
6 16.86543 48.38691     1     6 Austria      <NA>
> tail(mymapdata)
         long      lat group order  region subregion
2941 6.734766 53.58252    26  2941 Germany    Borkum
2942 6.642090 53.57920    26  2942 Germany    Borkum
2943 6.668555 53.60566    26  2943 Germany    Borkum
2944 6.754590 53.62549    26  2944 Germany    Borkum
2945 6.800879 53.62549    26  2945 Germany    Borkum
2946 6.734766 53.58252    26  2946 Germany    Borkum
```



Violent Crime Rates by US State (USArrests)

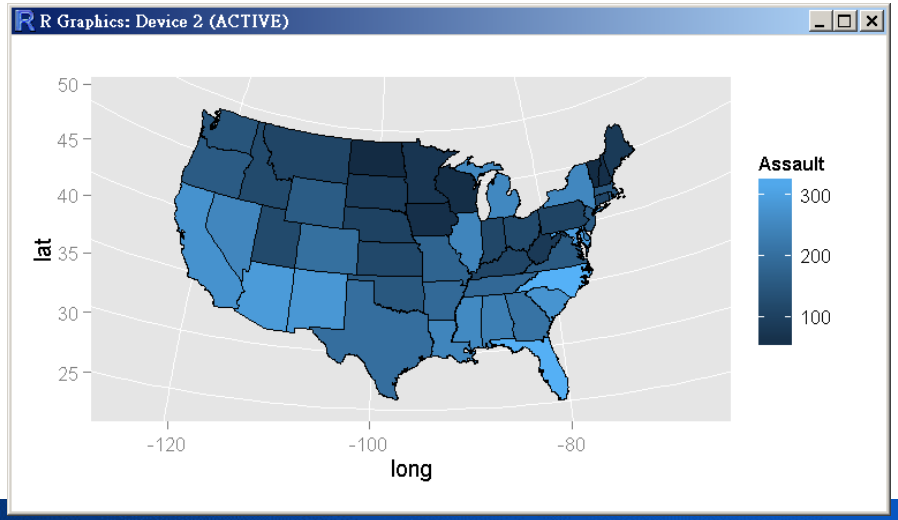
Violent Crime Rates by US State (USArrests): the data set contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

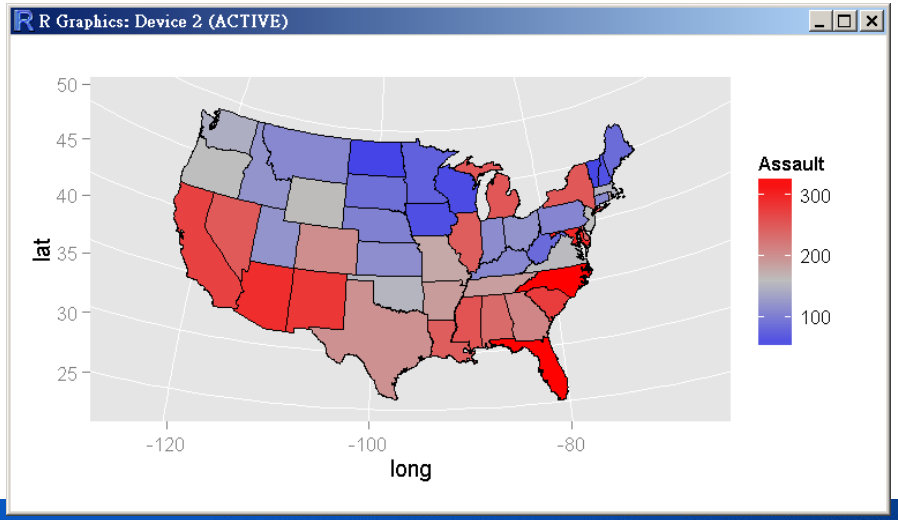
```
> head(USArrests, 3)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
> crimes <- data.frame(state = tolower(rownames(USArrests)), USArrests)
> head(crimes, 3)
               state Murder Assault UrbanPop Rape
Alabama      alabama   13.2     236       58 21.2
Alaska        alaska   10.0     263       48 44.5
Arizona      arizona    8.1     294       80 31.0

> library(maps); library(ggmap)
> states.map <- map_data("state")
> head(states.map, 3)
       long      lat group order  region subregion
1 -87.46201 30.38968     1     1 alabama      <NA>
2 -87.48493 30.37249     1     2 alabama      <NA>
3 -87.52503 30.37249     1     3 alabama      <NA>
> crime.map <- merge(states.map, crimes, by.x="region", by.y="state")
> head(crime.map, 3)
   region      long      lat group order subregion Murder Assault UrbanPop Rape
1 alabama -87.46201 30.38968     1     1      <NA>   13.2     236       58 21.2
2 alabama -87.48493 30.37249     1     2      <NA>   13.2     236       58 21.2
3 alabama -87.95475 30.24644     1    13      <NA>   13.2     236       58 21.2
```

分級著色圖 /面量圖(Choropleth Maps)

```
> library(plyr)
> crime.map <- arrange(crime.map, group, order)
> head(crime.map, 3)
   region      long      lat group order subregion Murder Assault UrbanPop Rape
1 alabama -87.46201 30.38968     1     1      <NA>   13.2     236       58 21.2
2 alabama -87.48493 30.37249     1     2      <NA>   13.2     236       58 21.2
3 alabama -87.52503 30.37249     1     3      <NA>   13.2     236       58 21.2
> ggplot(crime.map, aes(x=long, y=lat, group=group, fill=Assault)) +
        geom_polygon(colour="black") +
        coord_map("polyconic")
```

```
ggplot(crime.map, aes(x=long, y=lat, group=group, fill=Assault)) +
    geom_polygon(colour="black") +
    coord_map("polyconic") +
    scale_fill_gradient2(low="blue", mid="grey", high="red",
                         midpoint=median(crimes$Assault))
```



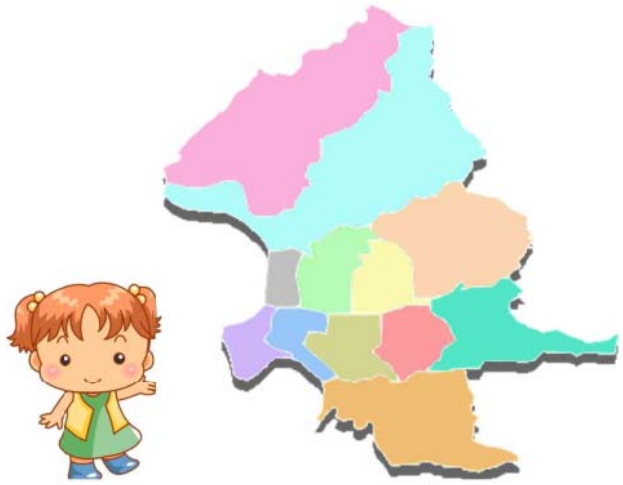


統計資訊地圖應用實例

美麗臺北 Beautiful Taipei
TAIPEI
花園 生態 友善 環保

臺北市各行政區
重要統計指標

中華民國 107 年



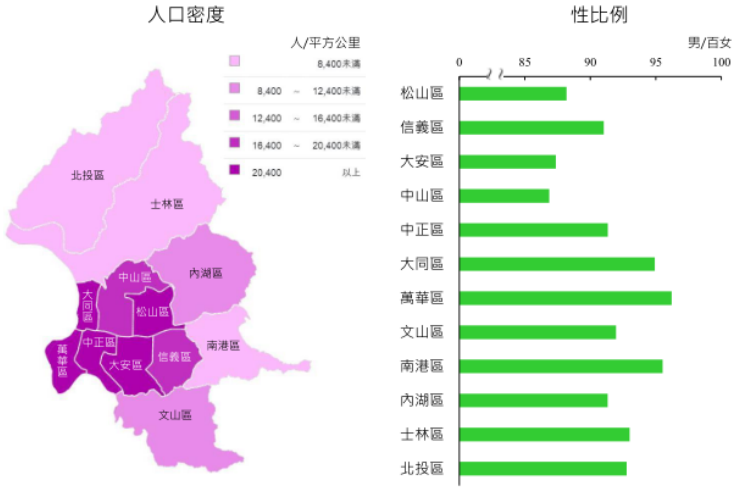
臺北市政府主計處編印

中華民國 108 年 6 月出版

3.人口概況

臺北市 107 年底戶籍登記人口密度以大安區最高，北投區最低；性比例則以萬華區最高，中山區最低，皆為女性多於男性。

人 口 概 況
107 年底



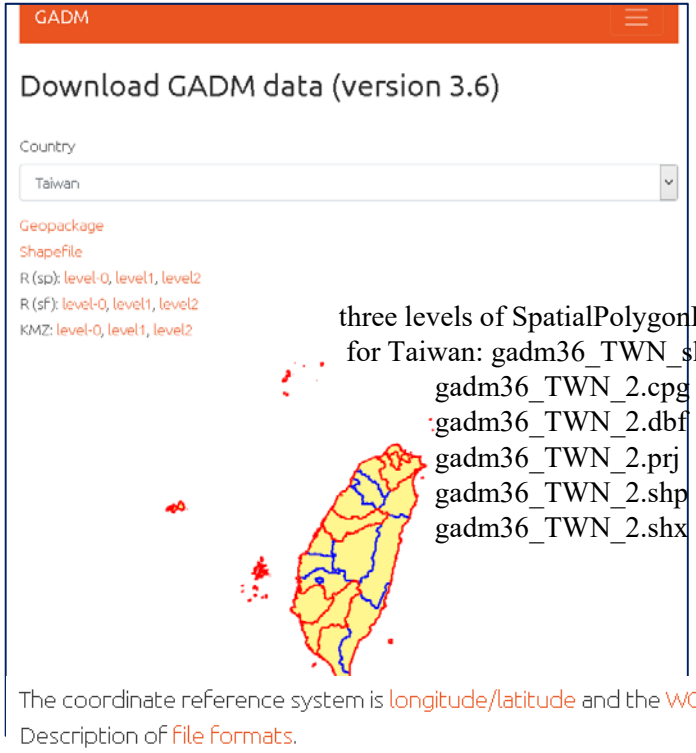
項目	人口數			戶數	戶量	人口密度	性比例
		男	女				
單位	人	人	人	戶	人/戶	人/平方公里	男/百女
107年底	2,668,572	1,273,375	1,395,197	1,056,233	2.53	9,818	91.27
松山區	205,702	96,393	109,309	81,128	2.54	22,148	88.18
信義區	223,406	106,457	116,949	89,574	2.49	19,933	91.03
大安區	高308,843	高144,007	高164,836	高121,344	2.55	高27,184	87.36
中山區	229,456	106,664	122,792	100,494	低2.28	16,771	低86.87
中正區	159,000	75,898	83,102	65,474	2.43	20,902	91.33
大同區	127,625	62,148	65,477	51,781	2.46	22,463	94.92
萬華區	189,603	92,969	96,634	78,770	2.41	21,419	高96.21
文山區	273,762	131,145	142,617	106,640	2.57	8,688	91.96
南港區	低121,670	低59,439	低62,231	低47,386	2.57	5,570	95.51
內湖區	287,429	137,196	150,233	108,739	2.64	9,102	91.32
士林區	286,544	138,079	148,465	107,521	高2.67	4,594	93.00
北投區	255,532	122,980	132,552	97,382	2.62	低4,497	92.78
計算方式 分子	人口數	男性人口數	女性人口數	戶數	人口數	人口數	男性人口數 *100
計算方式 分母					戶數	土地面積	女性人口數
資料來源	本府民政局						

說明：人口資料係戶籍登記數。

Shapefile圖形格式

ESRI Shapefile (shp)，或簡稱shapefile，是美國環境系統研究所公司（ESRI）開發的空間資料開放格式 (https://zh.wikipedia.org/wiki/Shapefile)

GADM.org is a spatial database of the location of the world's administrative areas (or adminstrative boundaries) for use in GIS and similar software.
https://gadm.org/download_country_v3.html



地圖Shape資料:

- shp: 用於儲存地圖元素的幾何資料。
- shx: 幾何資料索引，記錄每一shp檔案之中的位置。
- dbf: 以dBase IV的資料表格式儲存每個幾何形狀的屬性資料。
- prj (選項): shp檔中幾何資料使用的經緯度座標系統。

政府資料開放平台

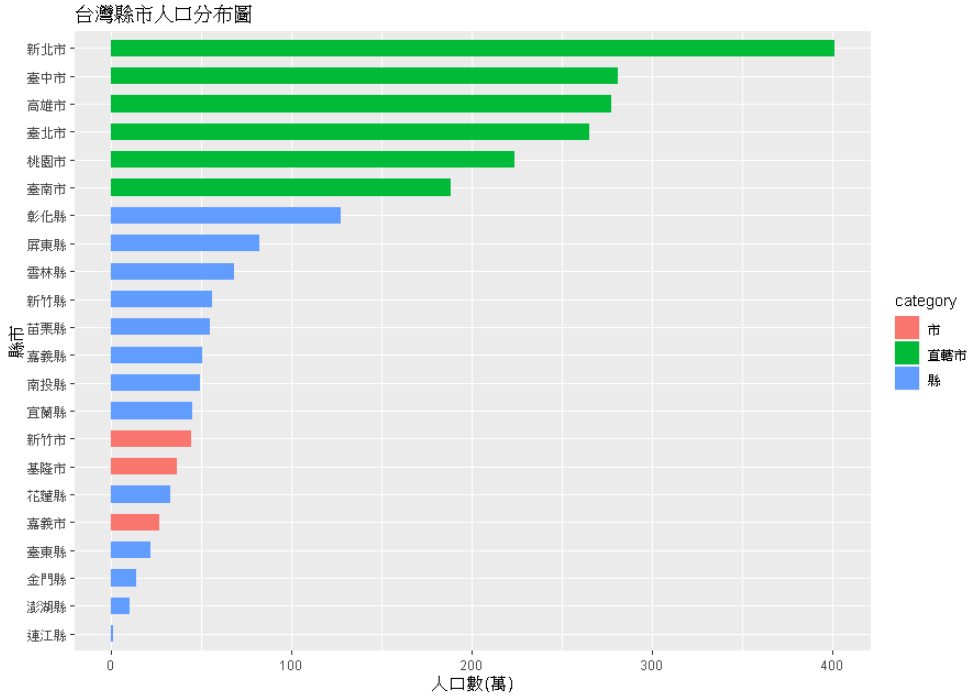
鄉鎮市區界線(TWD97經緯度)
https://data.gov.tw/dataset/7441
直轄市、縣市界線(TWD97經緯度)
https://data.gov.tw/dataset/7442

mapdata201907310953.zip
Metadata.xml
TOWN_MOI_1080726.dbf
TOWN_MOI_1080726.prj
TOWN_MOI_1080726.shp
TOWN_MOI_1080726.shx
TW-07-301000100G-614001.xml

零時政府: 台灣行政區域圖(Country, Town, Village)
https://github.com/g0v/twgeojson/tree/master/json

台灣各縣市人口分布長條圖

```
TW.Pop2019 <- read.csv("TW_Population2019.csv")
head(TW.Pop2019)
colnames(TW.Pop2019) <- c("rank", "city", "category", "population")
ggplot(TW.Pop2019, aes(x = reorder(city, population), y = population/10000,
                       fill = category)) +
  geom_bar(stat="identity", width=0.6) +
  coord_flip() +
  labs(title = "台灣縣市人口分布圖", x = "縣市", y = "人口數(萬)")
```



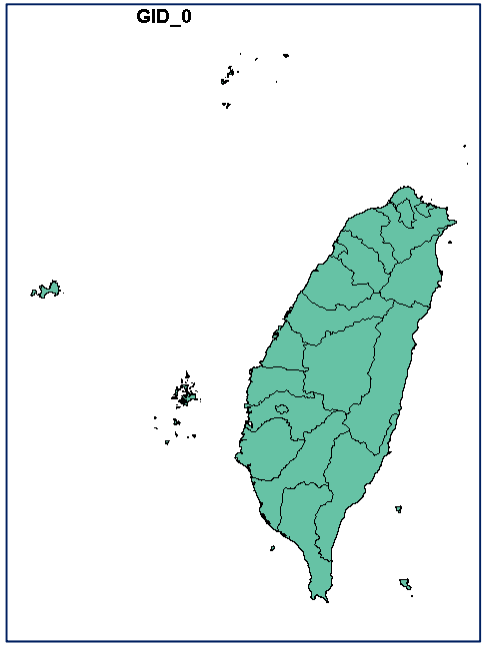
參考: How to plot a Taiwan Map colored with Population by ggplot2, Ozu Shi, 2018/01/13
http://www.rpubs.com/OzuShi/348822

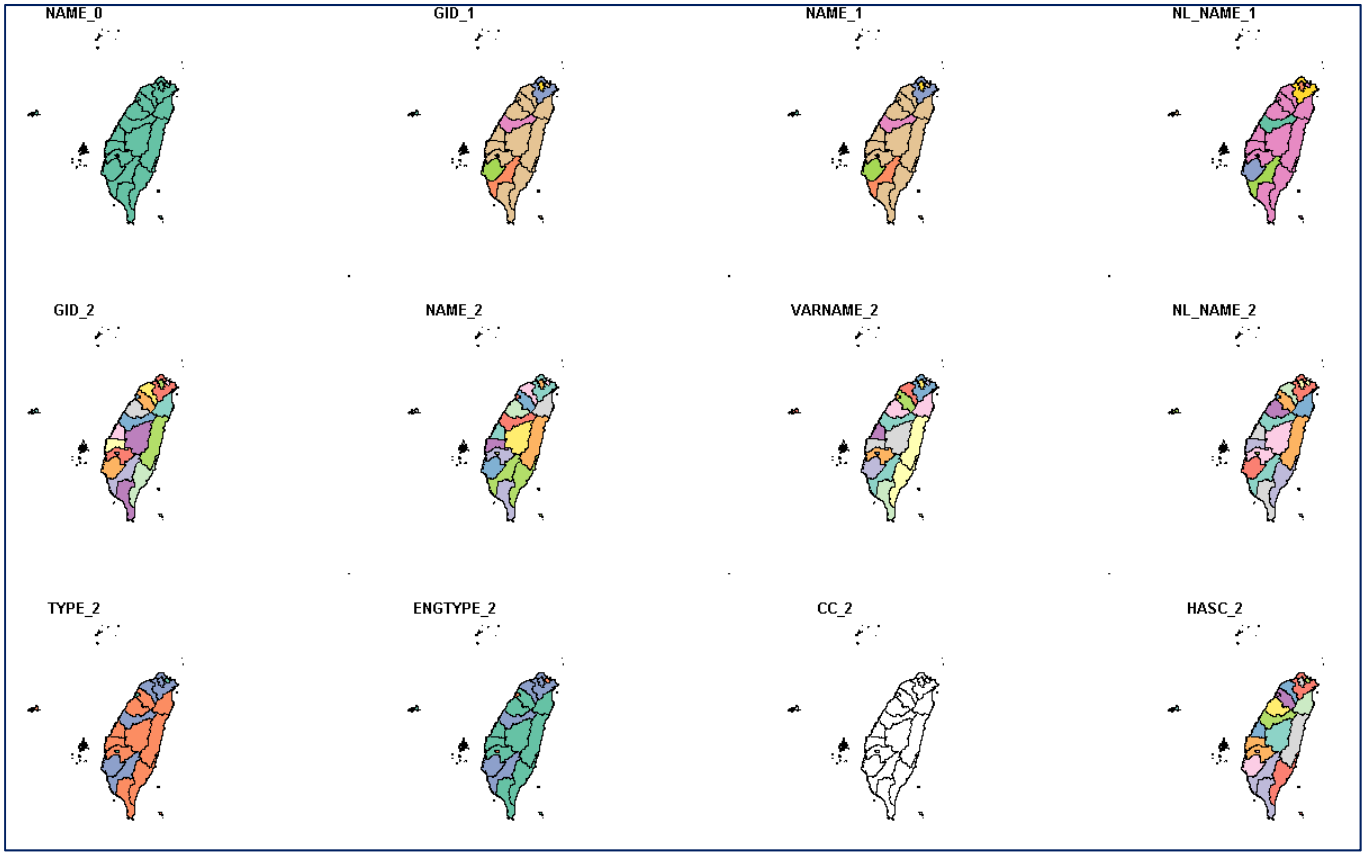
讀取台灣地圖Shapefile檔

```
> library(sf)
> taiwan.map <- st_read("data/gadm36_TWN_shp/gadm36_TWN_2.shp")
Reading layer `gadm36_TWN_2' from data source
`E:\TaipeiCityMap\data\gadm36_TWN_shp\gadm36_TWN_2.shp' using driver `ESRI Shapefile'
Simple feature collection with 22 features and 13 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 116.71 ymin: 20.6975 xmax: 122.1085 ymax: 26.38542
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
> head(taiwan.map)
Simple feature collection with 6 features and 13 fields
...
proj4string:    +proj=longlat +datum=WGS84 +no_defs
  GID_0 NAME_0   GID_1     NAME_1 NL_NAME_1     GID_2     NAME_2
VARNAME_2 NL_NAME_2
1   TWN Taiwan TWN.1_1     Fujian      福建 TWN.1.1_1     Kinmen
J<U+012B>nmen Xian    金門縣
2   TWN Taiwan TWN.1_1     Fujian      福建 TWN.1.2_1  Lienkiang M<U+01CE>z<U+01D4>
Lied<U+01CE>o|Matsu Islands  馬祖列島
...
6   TWN Taiwan TWN.5_1     Tainan      台南 TWN.5.1_1     Tainan
Tainan City      台南
     TYPE_2            ENGTYPE_2 CC_2   HASC_2                        geometry
1      Xian               County <NA> TW.FK.KM MULTIPOLYGON (((118.3999 24...
2      Xian               County <NA> TW.FK.LK MULTIPOLYGON (((120.086 26....
...
6 Zhixiashi Special Municipality <NA> TW.TN.TI MULTIPOLYGON (((120.1387 22...
> dim(taiwan.map)
[1] 22 14
> print(taiwan.map, n = 22)
```

直接繪製Shapefile地圖檔

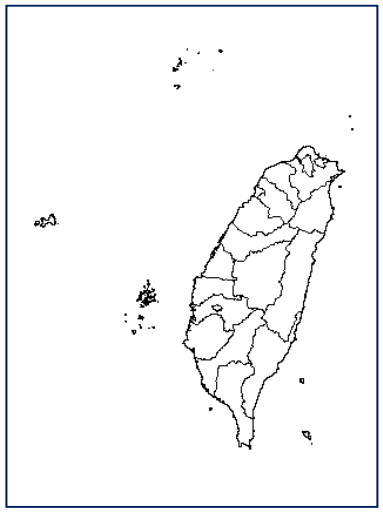
```
> plot(taiwan.map[1])
> plot(taiwan.map[2:13], max.plot = 12)
```

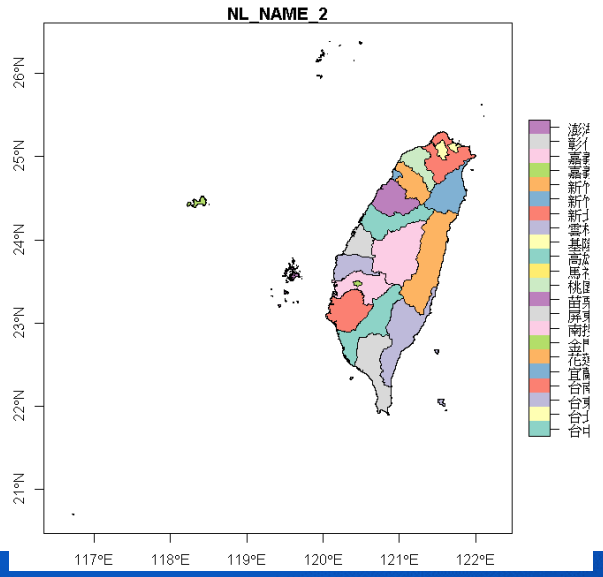




從Shapefile檔中某一欄位繪製地圖 (plot)

```
> plot(st_geometry(taiwan.map))
> plot(taiwan.map["NL_NAME_2"], axes = TRUE)
> st_geometry(taiwan.map)
Geometry set for 22 features
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 116.71 ymin: 20.6975 xmax: 122.1085 ymax: 26.38542
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
First 5 geometries:
MULTIPOLYGON (((118.3999 24.43569, 118.3979 24....
MULTIPOLYGON (((120.086 26.38403, 120.086 26.38...
MULTIPOLYGON (((120.3326 22.5532, 120.3326 22.5...
MULTIPOLYGON (((122.0788 25.63431, 122.0788 25....
MULTIPOLYGON (((120.5007 24.26875, 120.5007 24....
```

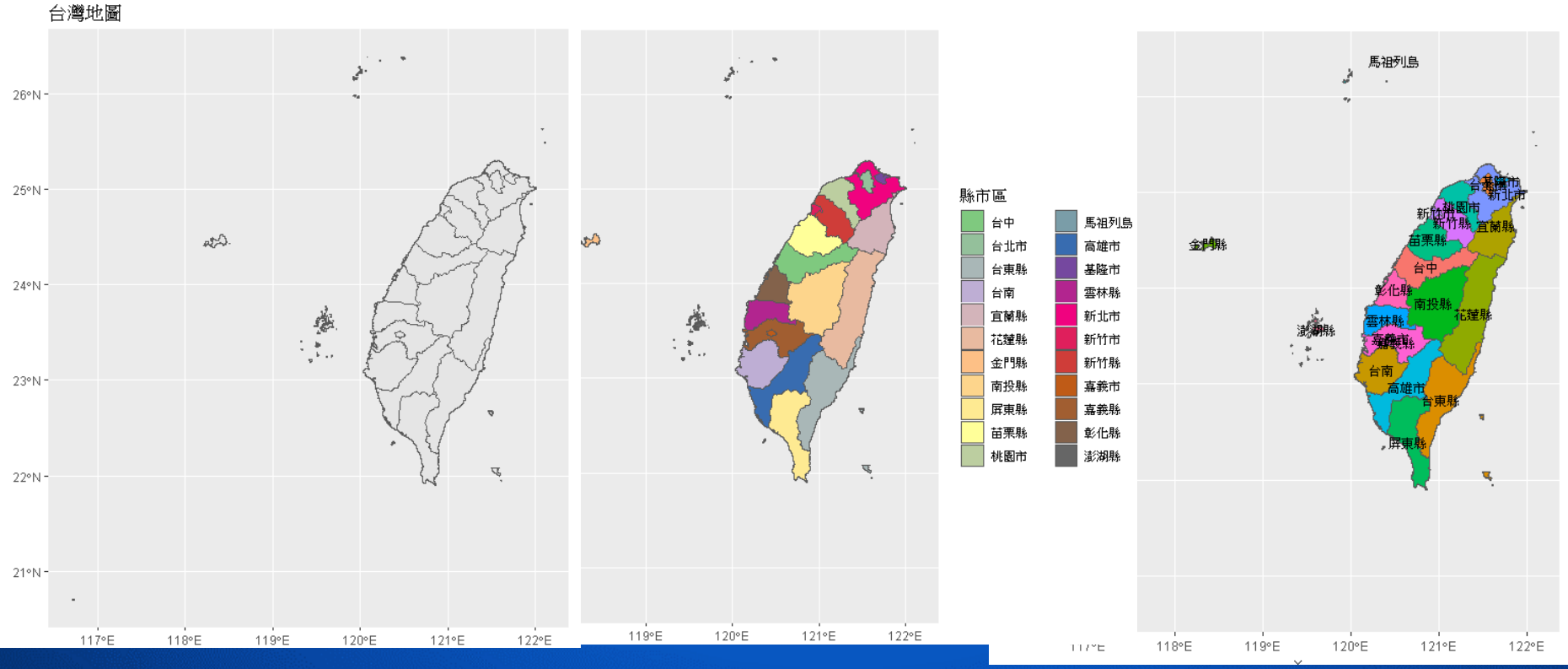




以ggplot繪製Shapefile地圖檔

```
ggplot(data = taiwan.map) +
  geom_sf() +
  labs(title = "台灣地圖")

library(RColorBrewer)
ggplot(data = taiwan.map) +
  geom_sf(aes(fill = NL_NAME_2)) +
  scale_fill_manual(name = "縣市區",
                    values = colorRampPalette(brewer.pal(8, "Accent"))(22)) +
  labs(title = "台灣地圖")
```

```
ggplot(data = taiwan.map) +
  geom_sf(aes(fill = NL_NAME_2), show.legend= F) +
  geom_sf_text(aes(label = NL_NAME_2), size = 3) +
  labs(title = "台灣地圖")
```



讀取台灣各縣市之人口統計資料檔

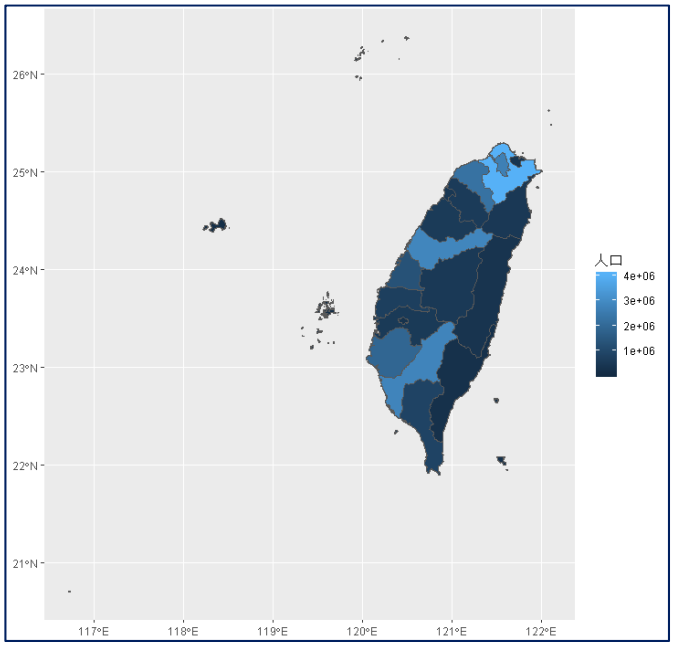
```
> TW.Population2019 <- read.csv("data/TW_Population2019.csv")
> head(TW.Population2019)
  排名   縣市   類別    人口
1    1 新北市 直轄市 4010657
2    2 臺中市 直轄市 2811729
3    3 高雄市 直轄市 2773786
4    4 臺北市 直轄市 2650154
5    5 桃園市 直轄市 2240328
6    6 臺南市 直轄市 1881730
> TW.Population2019$縣市
 [1] 新北市 臺中市 高雄市 臺北市 桃園市 臺南市 彰化縣 屏東縣 雲林縣 新竹縣 苗栗縣 嘉義縣
[13] 南投縣 宜蘭縣 新竹市 基隆市 花蓮縣 嘉義市 臺東縣 金門縣 澎湖縣 連江縣
22 Levels: 宜蘭縣 花蓮縣 金門縣 南投縣 屏東縣 苗栗縣 桃園市 高雄市 基隆市 連江縣 ... 澎湖縣
> TW.Population2019$縣市 <- as.character(TW.Population2019$縣市)
>
> taiwan.map$NL_NAME_2
 [1] 金門縣   馬祖列島 高雄市   新北市   台中     台南     台北市   彰化縣   嘉義市   嘉義縣
[11] 新竹市   新竹縣   花蓮縣   基隆市   苗栗縣   南投縣   澎湖縣   屏東縣   台東縣   桃園市
[21] 宜蘭縣   雲林縣
22 Levels: 台中 台北市 台東縣 台南 宜蘭縣 花蓮縣 金門縣 南投縣 屏東縣 苗栗縣 ... 澎湖縣
> levels(taiwan.map$NL_NAME_2)[c(1:4, 12)] <-
+   c("臺中市", "臺北市", "臺東縣", "臺南市", "連江縣")
> taiwan.map$NL_NAME_2
 [1] 金門縣 連江縣 高雄市 新北市 臺中市 臺南市 臺北市 彰化縣 嘉義市 嘉義縣 新竹市 新竹縣
[13] 花蓮縣 基隆市 苗栗縣 南投縣 澎湖縣 屏東縣 臺東縣 桃園市 宜蘭縣 雲林縣
22 Levels: 臺中市 臺北市 臺東縣 臺南市 宜蘭縣 花蓮縣 金門縣 南投縣 屏東縣 苗栗縣 ... 澎湖縣
```

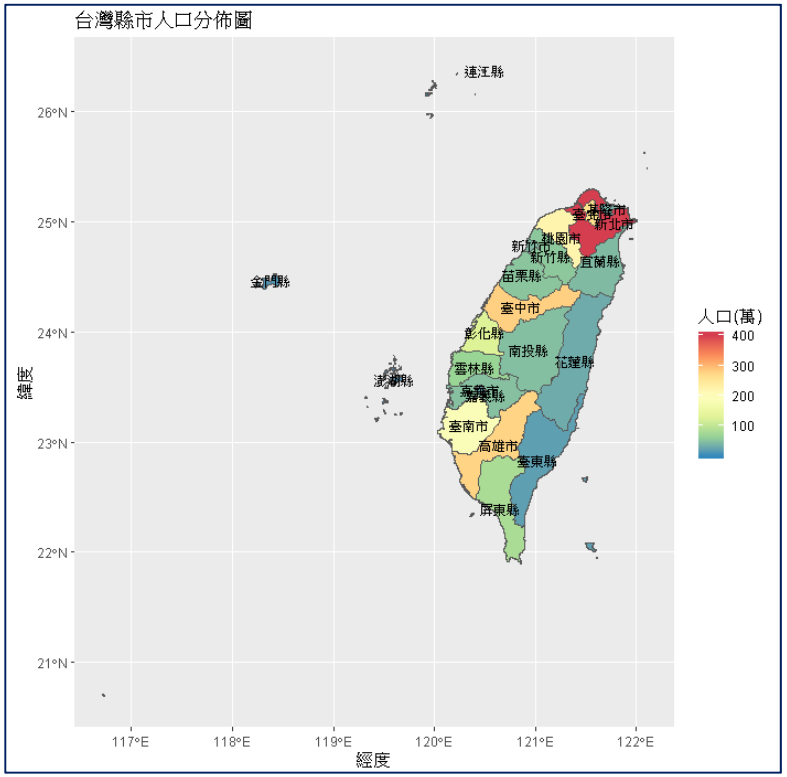
將統計資料與地圖資料結合

```
> my.taiwan.map <- taiwan.map[c("NL_NAME_2", "geometry")]
> my.taiwan.map$NL_NAME_2 <- as.character(my.taiwan.map$NL_NAME_2)
> head(my.taiwan.map)
Simple feature collection with 6 features and 1 field
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 118.2113 ymin: 22.47569 xmax: 122.1085 ymax: 26.38542
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
  NL_NAME_2                       geometry
1    金門縣 MULTIPOLYGON (((118.3999 24...
2    連江縣 MULTIPOLYGON (((120.086 26....
...
6    臺南市 MULTIPOLYGON (((120.1387 22...
>
> library(dplyr)
> my.taiwan.map.data <- left_join(my.taiwan.map, TW.Population2019,
+                                 by= c("NL_NAME_2" = "縣市"))
> head(my.taiwan.map.data)
Simple feature collection with 6 features and 4 fields
...
proj4string:    +proj=longlat +datum=WGS84 +no_defs
  NL_NAME_2 排名   類別    人口                        geometry
1    金門縣   20     縣  139319 MULTIPOLYGON (((118.3999 24...
2    連江縣   22     縣   13073 MULTIPOLYGON (((120.086 26....
...
6    臺南市    6 直轄市 1881730 MULTIPOLYGON (((120.1387 22...
> dim(my.taiwan.map.data)
[1] 22  5
```

繪製統計資訊地圖 (1)

```
ggplot(data = my.taiwan.map.data) +
  geom_sf(aes(fill = 人口))
```

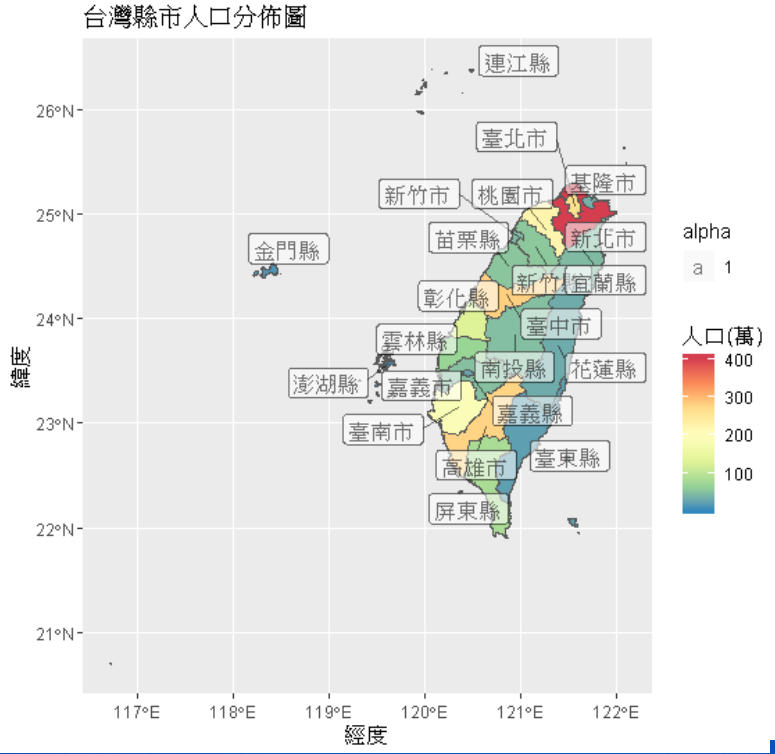





```
ggplot(data = my.taiwan.map.data) +
  geom_sf(aes(fill = 人口/10000)) +
  geom_sf_text(aes(label = NL_NAME_2), size = 3) +
  scale_fill_distiller(palette = "Spectral", name = "人口(萬)") +
  #scale_fill_gradientn(colours = tim.colors(22), name = "人口(萬)") +
  #scale_fill_viridis(name = "人口(萬)") +
  #scale_fill_distiller(palette = "YlOrRd", name = "人口(萬)") +
  #scale_fill_distiller(palette = "YlOrRd", direction = -1, name = "人口(萬)") +
  labs(title="台灣縣市人口分佈圖", x ="經度", y = "緯度")
```

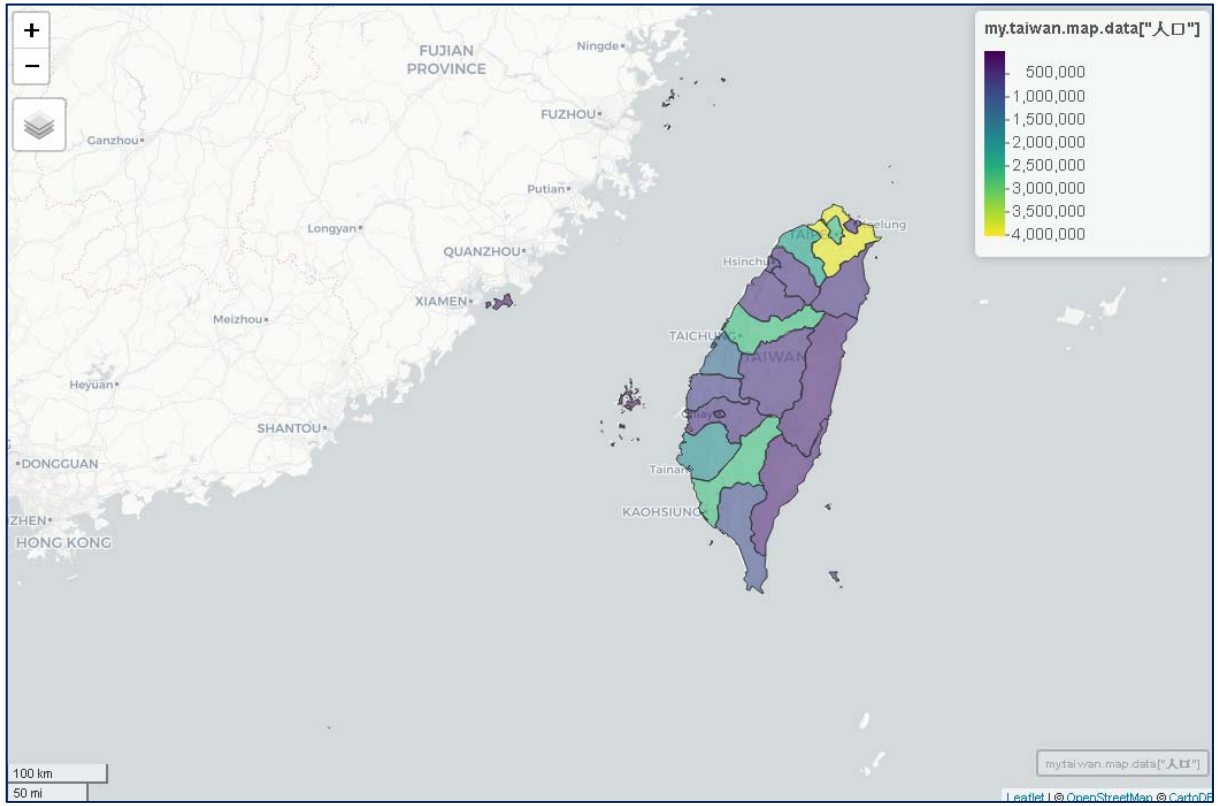
繪製統計資訊地圖 (2)

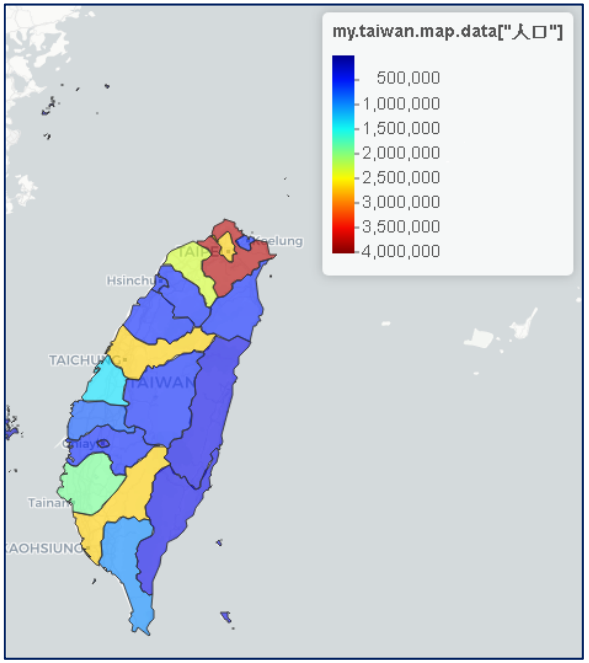
```
library(devtools)
devtools::install_github("yutannihilation/ggsflabel")
library(ggsflabel)
ggplot(data = my.taiwan.map.data) +
  geom_sf(aes(fill = 人口/10000)) +
  scale_fill_distiller(palette = "Spectral", name = "人口(萬)") +
  geom_sf_label_repel(aes(label = NL_NAME_2, alpha = 1)) +
    labs(title="台灣縣市人口分佈圖", x ="經度", y = "緯度")
```



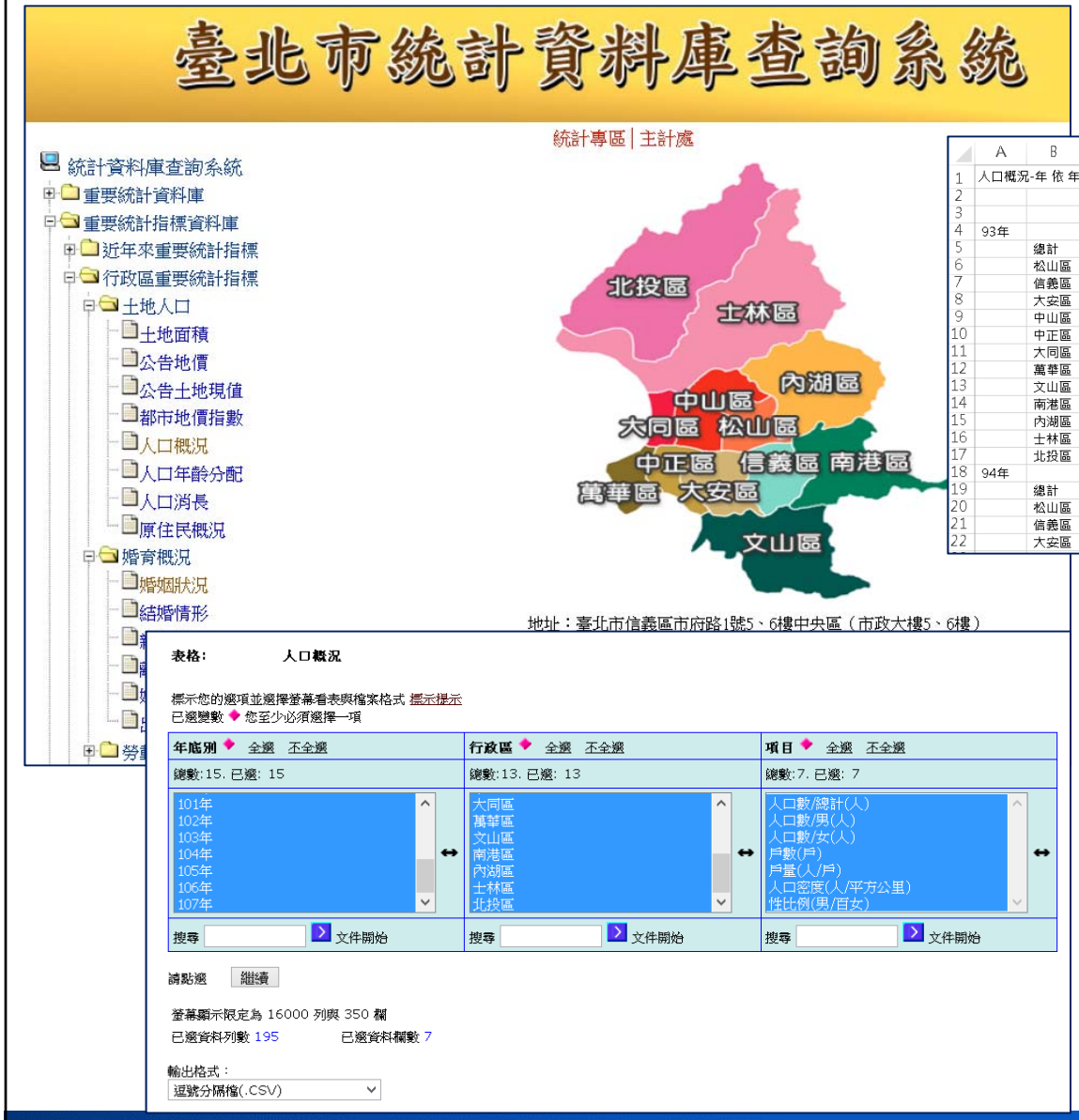
繪製統計資訊地圖 (3)

```
library(mapview)
mapview(my.taiwan.map.data["人口"])
mapview(my.taiwan.map.data["人口"], col.regions = tim.colors(100))
```





下載台北市行政區重要統計指標資料檔



	A	B	C	D	E	F	G	H	I
1	人口概況-年 依 年底別, 行政區 與 項目								
2									
3			人口數/總計(人)	人口數/男(人)	人口數/女(人)	戶數(戶)	戶量(人/戶)	人口密度(人/平方公里)	性比例(男/百女)
4	93年								
5		總計	2622472	1286303	1336169	923325	2.84	9649	96.27
6		松山區	205962	98839	107123	73768	2.79	22176	92.27
7		信義區	232506	114648	117858	83875	2.77	20745	97.28
8		大安區	312554	149602	162952	112428	2.78	27510	91.81
9		中山區	216868	104299	112569	84172	2.58	15850	92.65
10		中正區	158486	77676	80810	58841	2.69	20834	96.12
11		大同區	128512	64291	64221	44580	2.88	22619	100.11
12		萬華區	197445	100338	97107	71281	2.77	22305	103.33
13		文山區	258046	127225	130821	90972	2.84	8190	97.25
14		南港區	112982	56618	56364	37928	2.98	5173	100.45
15		內湖區	261201	127887	133314	87335	2.99	8271	95.93
16		士林區	288921	142259	146662	94924	3.04	4633	97
17		北投區	248989	122621	126368	83221	2.99	4382	97.03
18	94年								
19		總計	2616375	1279513	1336862	933110	2.8	9626	95.71
20		松山區	208101	99575	108526	75009	2.77	22406	91.75
21		信義區	230780	113351	117429	84605	2.73	20591	96.53
22		大安區	312166	148926	163240	113108	2.76	27476	91.23

人口概況:

年底別、行政區、人口數/總計(人)、人口數/男(人)、人口數/女(人)、戶數(戶)、戶量(人/戶)、人口密度(人/平方公里)、性比例(男/百女)。

"1.戶量=人口數÷戶數。"

"2.人口密度=人口數÷土地面積。"

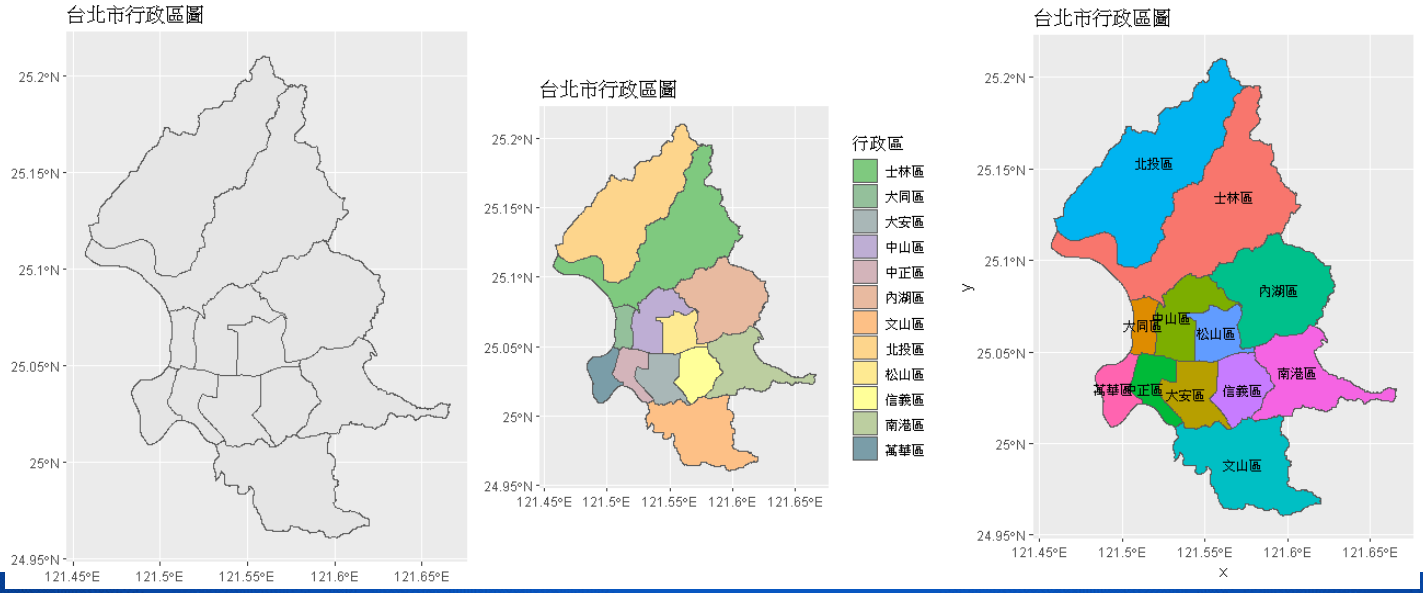
"3.性比例=男性人口數÷女性人口數×100。"

讀取「台灣鄉鎮市區界線」shapefile檔 並選取「台北市」之地圖資料

```
> taiwan.town.map <- st_read("data/mapdata201907310953/TOWN_MOI_1080726.shp")
Reading layer `TOWN_MOI_1080726' from data source
`E:\TaipeiCityMap_R\data\mapdata201907310953\TOWN_MOI_1080726.shp' using driver `ESRI Shapefile'
Simple feature collection with 368 features and 7 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 114.3593 ymin: 10.37135 xmax: 124.5611 ymax: 26.38528
epsg (SRID):    NA
proj4string:    +proj=longlat +ellps=GRS80 +no_defs
> head(taiwan.town.map)
Simple feature collection with 6 features and 7 fields
...
proj4string:    +proj=longlat +ellps=GRS80 +no_defs
  TOWNID TOWNCODE COUNTYNAME TOWNNAME            TOWNENG COUNTYID COUNTYCODE                       geometry
1    V02 10014020     臺東縣    成功鎮 Chenggong Township        V      10014 MULTIPOLYGON (((121.4098 23...
2    T21 10013210     屏東縣    佳冬鄉   Jiadong Township        T      10013 MULTIPOLYGON (((120.5485 22...
...
6    N07 10007120     彰化縣    田中鎮 Tianzhong Township        N      10007 MULTIPOLYGON (((120.5825 23...
> dim(taiwan.town.map)
[1] 368   8
>
> taipei.map <- taiwan.town.map[taiwan.town.map$COUNTYNAME == "臺北市",]
> head(taipei.map)
Simple feature collection with 6 features and 7 fields
...
proj4string:    +proj=longlat +ellps=GRS80 +no_defs
    TOWNID TOWNCODE COUNTYNAME TOWNNAME             TOWNENG COUNTYID COUNTYCODE                       geometry
152    A02 63000030     臺北市    大安區       Daan District        A      63000 MULTIPOLYGON (((121.5438 25...
153    A11 63000080     臺北市    文山區    Wenshan District        A      63000 MULTIPOLYGON (((121.5973 25...
...
346    A13 63000090     臺北市    南港區    Nangang District        A      63000 MULTIPOLYGON (((121.6143 25...
> dim(taipei.map)
[1] 12  8
```

繪製台北市行政區圖

```
library(gridExtra)
g1 <- ggplot(data = taipei.map) + geom_sf() + labs(title = "台北市行政區圖")

g2 <- ggplot(data = taipei.map) +
  geom_sf(aes(fill = TOWNNAME)) +
  scale_fill_manual(name = "行政區",
                    values = colorRampPalette(brewer.pal(8, "Accent"))(22)) +
  labs(title = "台北市行政區圖")

g3 <- ggplot(data = taipei.map) +
  geom_sf(aes(fill = TOWNNAME), show.legend= F) +
  geom_sf_text(aes(label = TOWNNAME), size = 3) +
  labs(title = "台北市行政區圖")

grid.arrange(g1, g2, g3, nrow=1, ncol=3)
```



讀取北市之歷年人口統計資料檔

```
> TPE.Population <- read.csv("data/Taipei_Population_93-107.csv")
> head(TPE.Population)
  Year Admin.Area Pop.All Pop.Male Pop.Female Household AvePop.H Pop.Density Gender.Ratio
1   93     松山區  205962    98839     107123     73768     2.79       22176        92.27
2   93     信義區  232506   114648     117858     83875     2.77       20745        97.28
...
6   93     大同區  128512    64291      64221     44580     2.88       22619       100.11
>
> TPE.Population107 <- TPE.Population[TPE.Population$Year == 107,]
> TPE.Population107$Admin.Area
 [1] 松山區 信義區 大安區 中山區 中正區 大同區 萬華區 文山區 南港區 內湖區 士林區 北投區
Levels: 士林區 大同區 大安區 中山區 中正區 內湖區 文山區 北投區 松山區 信義區 南港區 萬華區
> TPE.Population107$Admin.Area <- as.character(TPE.Population107$Admin.Area)
>
> my.taipei.map <- taipei.map[c("TOWNNAME", "geometry")]
> my.taipei.map$TOWNNAME <- as.character(my.taipei.map$TOWNNAME)
> head(my.taipei.map)
Simple feature collection with 6 features and 1 field
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 121.4836 ymin: 24.9605 xmax: 121.6659 ymax: 25.06455
epsg (SRID):    NA
proj4string:    +proj=longlat +ellps=GRS80 +no_defs
    TOWNNAME                       geometry
152   大安區 MULTIPOLYGON (((121.5438 25...
153   文山區 MULTIPOLYGON (((121.5973 25...
343   信義區 MULTIPOLYGON (((121.58 25.0...
...
```

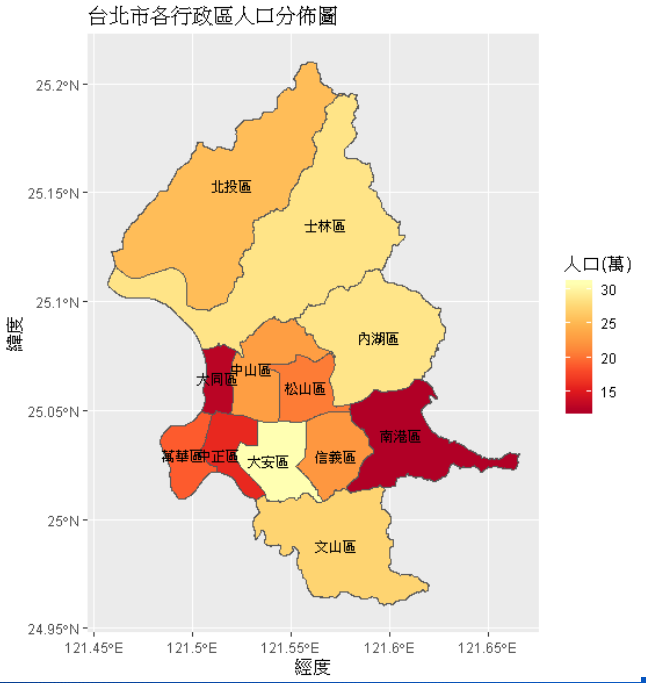
將統計資料與地圖資料結合

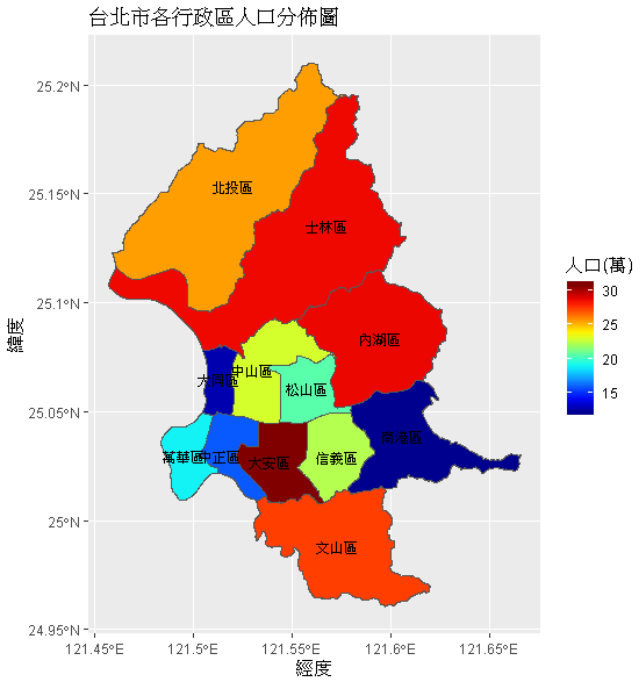
```
> library(dplyr)
> my.taipei.map.data <- left_join(my.taipei.map, TPE.Population107,
+                                 by= c("TOWNNAME"= "Admin.Area"))
> head(my.taipei.map.data)
Simple feature collection with 6 features and 9 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 121.4836 ymin: 24.9605 xmax: 121.6659 ymax: 25.06455
epsg (SRID):    NA
proj4string:    +proj=longlat +ellps=GRS80 +no_defs
  TOWNNAME Year Pop.All Pop.Male Pop.Female Household AvePop.H Pop.Density Gender.Ratio                       geometry
1   大安區  107  308843   144007     164836    121344     2.55       27184        87.36 MULTIPOLYGON (((121.5438 25...
2   文山區  107  273762   131145     142617    106640     2.57        8688        91.96 MULTIPOLYGON (((121.5973 25...
3   信義區  107  223406   106457     116949     89574     2.49       19933        91.03 MULTIPOLYGON (((121.58 25.0...
4   萬華區  107  189603    92969      96634     78770     2.41       21419        96.21 MULTIPOLYGON (((121.5052 25...
5   中正區  107  159000    75898      83102     65474     2.43       20902        91.33 MULTIPOLYGON (((121.5134 25...
6   南港區  107  121670    59439      62231     47386     2.57        5570        95.51 MULTIPOLYGON (((121.6143 25...
> dim(my.taipei.map.data)
[1] 12 10
```

https://spatialanalysis.github.io/lab_tutorials/4_R_Mapping.html

繪製台北市各行政區人口統計地圖 (1)

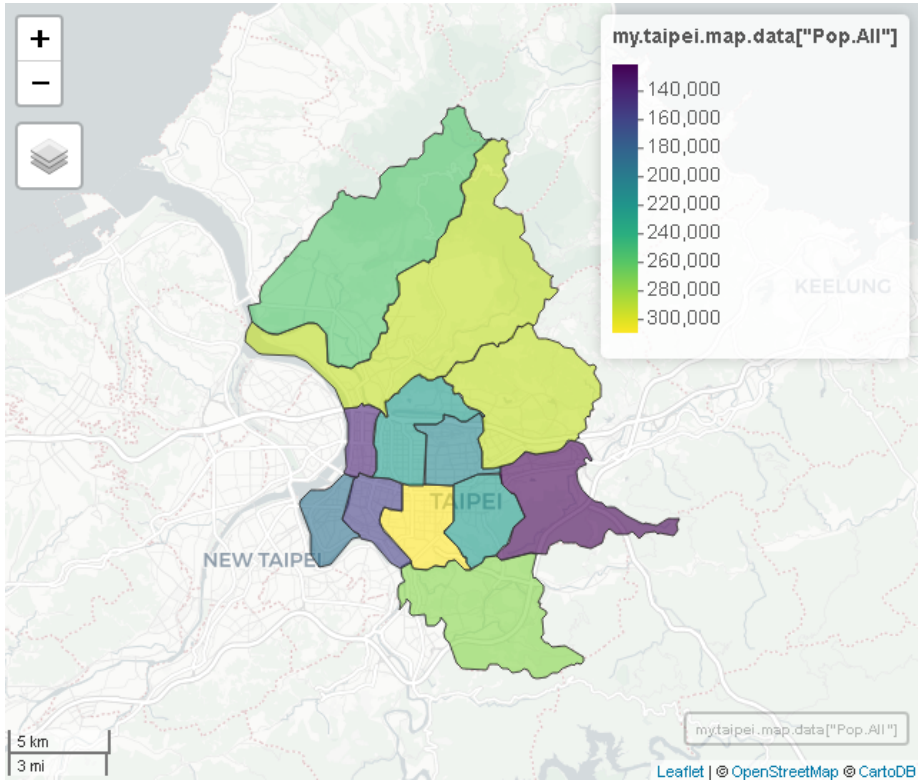
```
library(viridis)
library(fields)
ggplot(data = my.taipei.map.data) +
  geom_sf(aes(fill = Pop.All/10000)) +
  geom_sf_text(aes(label = TOWNNAME), size = 3) +
  #scale_fill_distiller(palette = "Spectral", name = "人口(萬)") +
  #scale_fill_gradientn(colours = tim.colors(22), name = "人口(萬)") +
  #scale_fill_viridis(name = "人口(萬)") +
  #scale_fill_distiller(palette = "YlOrRd", name = "人口(萬)") +
  scale_fill_distiller(palette = "YlOrRd", direction = -1, name = "人口(萬)") +
  labs(title="台北市各行政區人口分佈圖", x ="經度", y = "緯度")
```

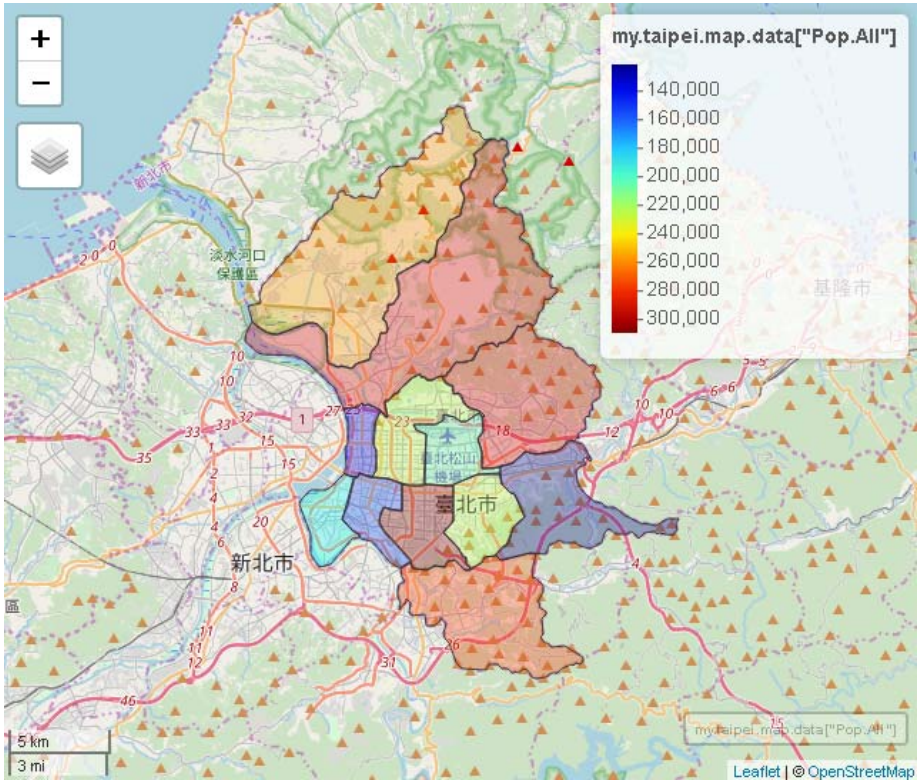




繪製台北市各行政區人口統計地圖 (2)

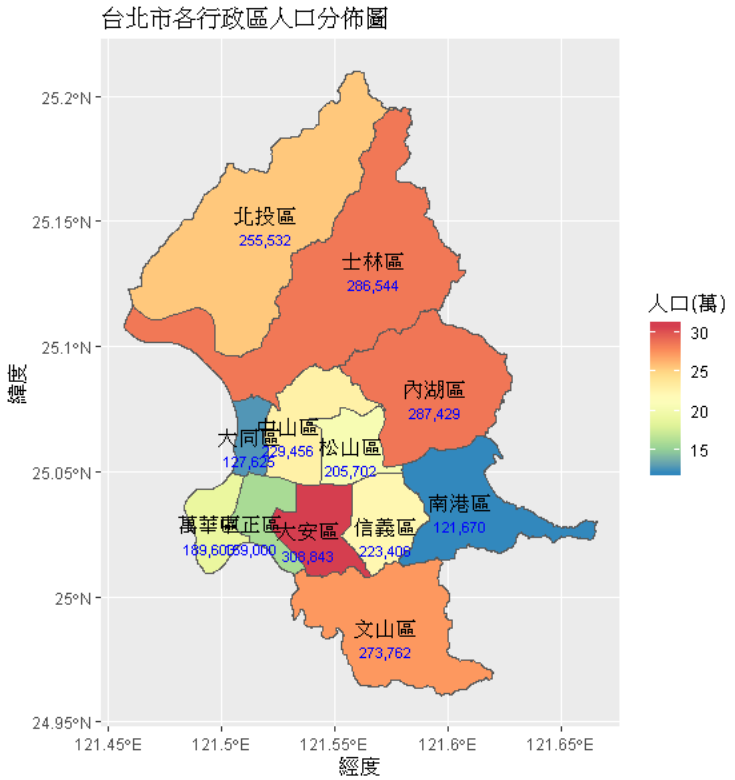
```
library(mapview)
mapview(my.taipei.map.data["Pop.All"])
mapview(my.taipei.map.data["Pop.All"], map.types = "OpenStreetMap",
        col.regions = tim.colors(100), alpha.regions = 0.3)
```





於圖上標記文字、數字

```
p <- ggplot(data = my.taipei.map.data) +
  geom_sf(aes(fill = Pop.All/10000)) +
  geom_sf_text(aes(label = TOWNNAME), size = 4) +
  scale_fill_distiller(palette = "Spectral", name = "人口(萬)") +
  labs(title="台北市各行政區人口分佈圖", x ="經度", y = "緯度")

p + geom_sf_text(aes(label = format(Pop.All, big.mark = ",", scientific = FALSE)),
                 colour = "blue", size = 3,
                 position = position_nudge(y = -0.01))
```

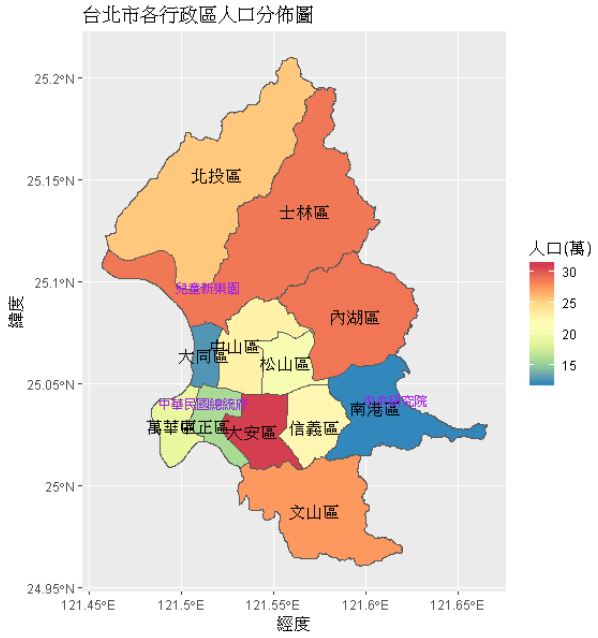


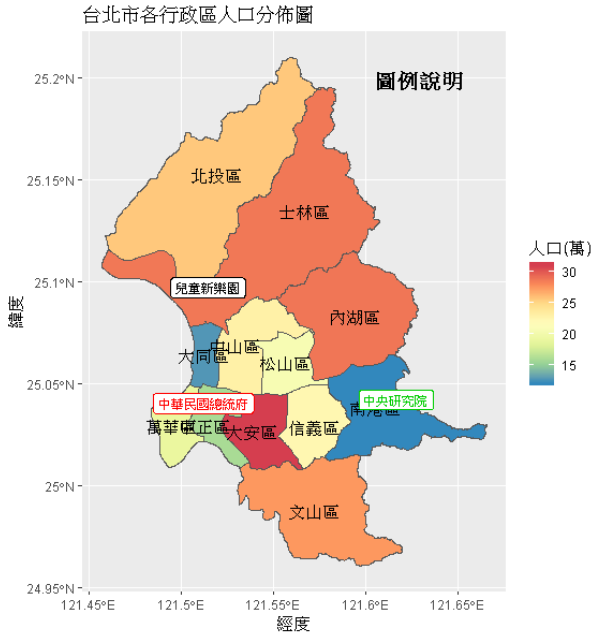
讀取地址資料，轉換成經緯度

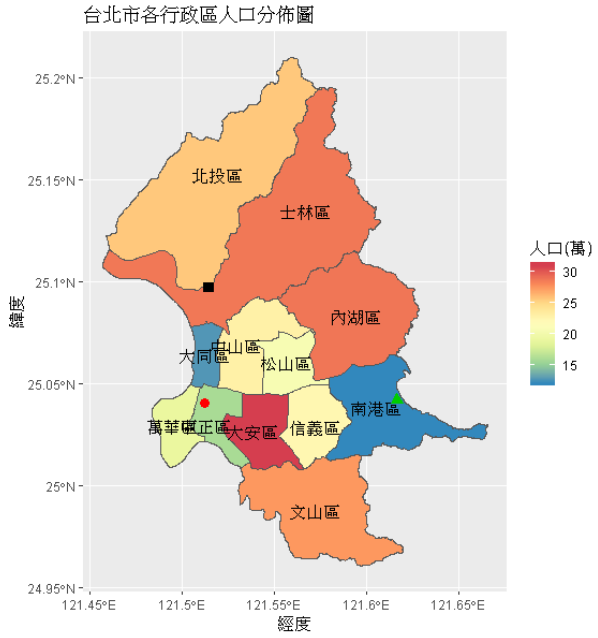
```
> address.df <- read.csv("data/address.csv", stringsAsFactors=FALSE, header=FALSE)
> has_google_key()
[1] TRUE
> address.df[1,2]
[1] " 11169台北市士林區承德路五段55號"
> geocode(address.df[1,2])
Source :
https://maps.googleapis.com/maps/api/geocode/json?address=11169%E5%8F%B0%E5%8C%97%E5%B8%82%E5%A
3%AB%E6%9E%97%E5%8D%80%E6%89%BF%E5%BE%B7%E8%B7%AF%E4%BA%94%E6%AE%B555%E8%99%9F&key=xxx
# A tibble: 1 x 2
    lon   lat
  <dbl> <dbl>
1  122.  25.1
> address.df
             V1                                    V2
1     兒童新樂園      11169台北市士林區承德路五段55號
2 中華民國總統府     100台北市中正區重慶南路一段122號
3     中央研究院   11529台北市南港區研究院路二段128號
> address.xy <- data.frame(name = address.df$V1, mutate_geocode(address.df["V2"], V2))
Source : https://maps.googleapis.com/maps/api/geocode/json?address=11169%E5%...8%99%9F&key=xxx
Source : https://maps.googleapis.com/maps/api/geocode/json?address=100%E5%8F...8%99%9F&key=xxx
Source : https://maps.googleapis.com/maps/api/geocode/json?address=11529%E5%...8%99%9F&key=xxx
> address.xy
            name                                    V2      lon      lat
1     兒童新樂園      11169台北市士林區承德路五段55號 121.5147 25.09709
2 中華民國總統府     100台北市中正區重慶南路一段122號 121.5123 25.04036
3     中央研究院   11529台北市南港區研究院路二段128號 121.6167 25.04208
```

給定經緯度，於圖上標記文字、數字及符號

```
p + geom_text(data = address.xy, aes(x = lon, y = lat, label = name),
              colour = "purple", size = 3)

p + geom_label(data = address.xy, aes(x = lon, y = lat, label = name),
               colour = 1:nrow(address.xy), size = 3) +
      annotate("text", label = "圖例說明", x = 121.63, y = 25.2, size = 5,
               fontface = "bold")

p + geom_point(data = address.xy, aes(x = lon, y = lat, label = name),
                      colour = 1:nrow(address.xy), size = 3,
                      shape = 15:17)
```







choroplethr: Simplify the Creation of Choropleth Maps in R

Choropleths are thematic maps where geographic regions, such as states, are colored according to some metric, such as the number of people who live in that state. This package simplifies this process by

1. Providing ready-made functions for creating choropleths of common maps.
2. Providing data and API connections to interesting data sources for making choropleths.
3. Providing a framework for creating choropleths from arbitrary shapefiles.
4. Overlaying those maps over reference maps from Google Maps.

NOTE:
choroplethrMaps:
Contains Maps Used by the 'choroplethr' Package Contains 3 maps: US States, US Counties, Countries of the world.

The US Mexcian American Population data: the percentage of Mexican Americans by US state from the 2010 Census.

state_choropleth function: the function requires that the **data frame** to be plotted has a column named region to represent state, and a column named value.

Additionally, the entries in the **region** column must exactly match how the entries are named in the **region** column of the dataset **state.map** from the **choroplethrMaps** package.

```
> library(choroplethrMaps)
> data(state.map)
> head(state.map)
       long      lat order  hole piece group id      GEO_ID STATE  region LSAD CENSUSAREA
1 -112.5386 37.00067     1 FALSE     1   0.1  0 0400000US04    04 arizona <NA>   113594.1
2 -112.5345 37.00068     2 FALSE     1   0.1  0 0400000US04    04 arizona <NA>   113594.1
...
5 -111.4128 37.00148     5 FALSE     1   0.1  0 0400000US04    04 arizona <NA>   113594.1
6 -111.4059 37.00148     6 FALSE     1   0.1  0 0400000US04    04 arizona <NA>   113594.1
> dim(state.map)
[1] 50763    12
```

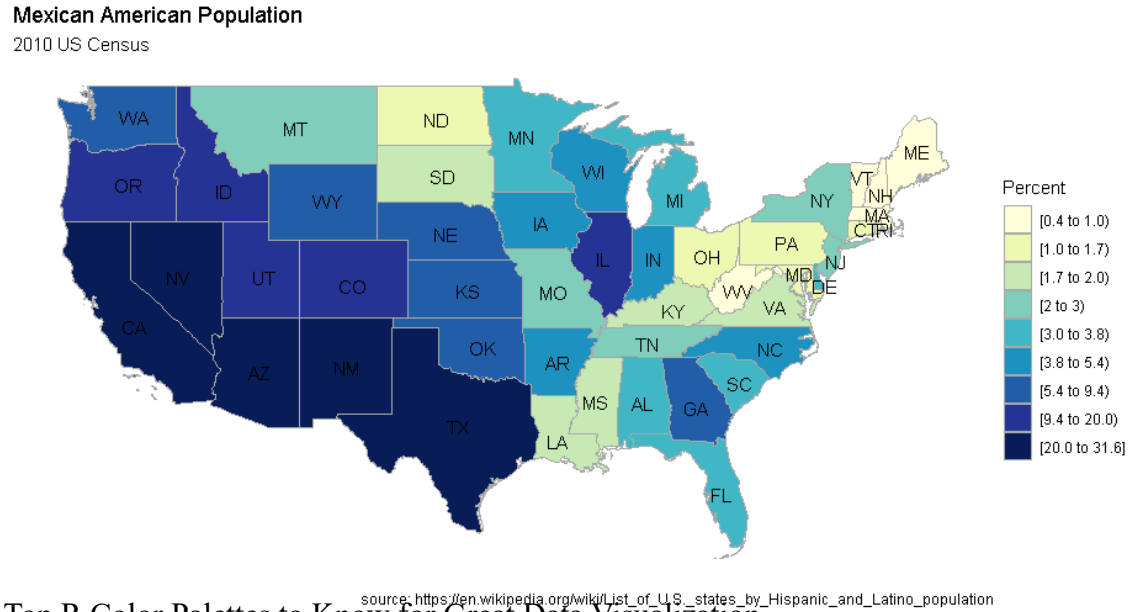
The US Mexcian American Population data

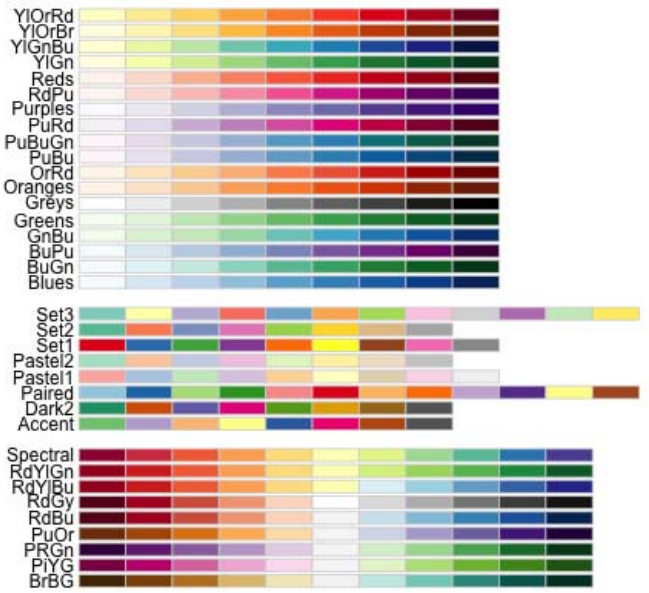
```
> library(ggplot2)
> library(choroplethr)
> data(continental_us_states)
> head(continental_us_states)
[1] "alabama"     "arkansas"    "arizona"     "california"  "colorado"
[6] "connecticut"
>
> mex.am <- read.csv("data/mexican_american.csv")
> head(mex.am)
       state population percent
1    Alabama     185602     3.7
2     Alaska     214642     3.0
3    Arizona    1957668    25.9
4   Arkansas     138164     4.7
5 California   11423146    30.7
6   Colorado     757181    15.1
> dim(mex.am)
[1] 51  3
>
> mex.am$region <- tolower(mex.am$state)
> mex.am$value <- mex.am$percent
> head(mex.am)
       state population percent     region value
1    Alabama     185602     3.7    alabama   3.7
2     Alaska     214642     3.0     alaska   3.0
3    Arizona    1957668    25.9    arizona  25.9
4   Arkansas     138164     4.7   arkansas   4.7
5 California   11423146    30.7 california  30.7
6   Colorado     757181    15.1   colorado  15.1
```

state_choropleth {choroplethr}:
Create a choropleth of US States

```
> state_choropleth(mex.am,
+                  num_colors = 9,
+                  zoom = continental_us_states) +
+   scale_fill_brewer(palette="YlGnBu") +
+   labs(title = "Mexican American Population",
+        subtitle = "2010 US Census",
+        caption = "source: https://en.wikipedia.org/wiki/List_of_U.S._states_by_Hispanic_and_Latino_population",
+        fill = "Percent")
```

RColorBrewer package palette





Top R Color Palettes to Know for Great Data Visualization
https://www.datanovia.com/en/blog/top-r-color-palettes-to-know-for-great-data-visualization/

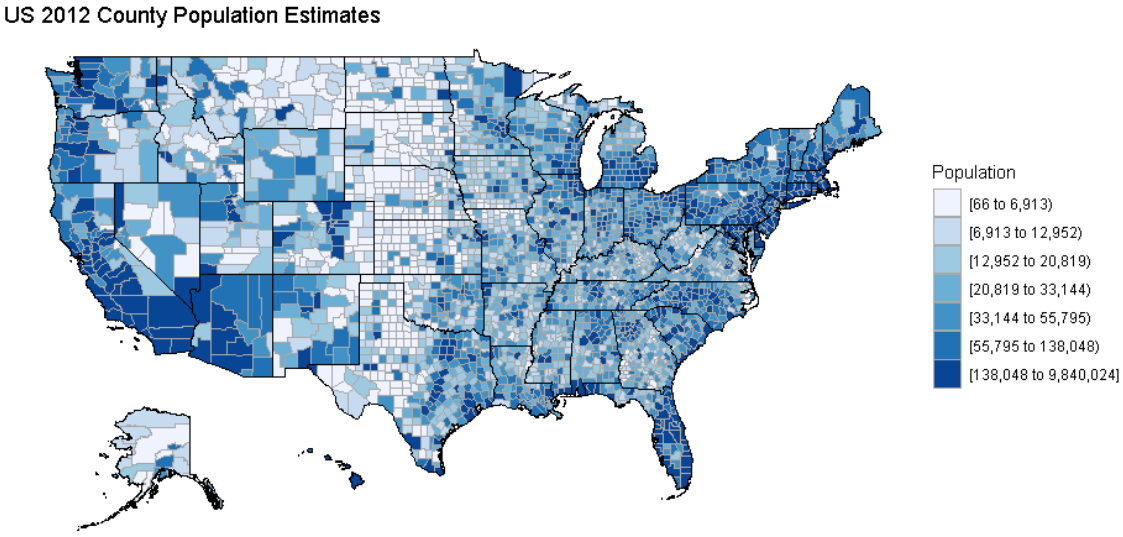
county_choropleth {choroplethr}:

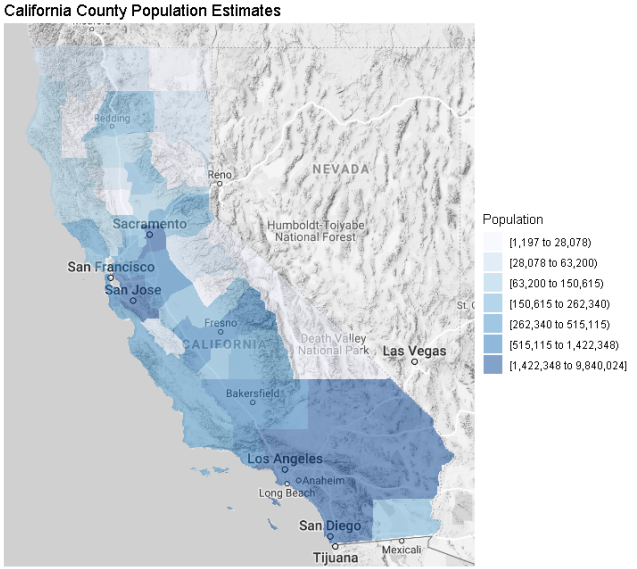
Create a choropleth of US Counties

```
> data(df_pop_county)
> head(df_pop_county)
  region  value
1   1001  54590
2   1003 183226
3   1005  27469
4   1007  22769
5   1009  57466
6   1011  10779
> dim(df_pop_county)
[1] 3143    2
```

```
> county_choropleth(df_pop_county,
+                   title = "US 2012 County Population Estimates",
+                   legend = "Population")
> county_choropleth(df_pop_county,
+                   title = "California County Population Estimates",
+                   legend = "Population",
+                   state_zoom = "california",
+                   reference_map = TRUE)
```

```
Source : https://maps.googleapis.com/maps/api/staticmap?center=37.010625,-120.716466&zoo
m=6&size=640x640&scale=2&maptype=terrain&language=en-EN&key=xxx
Scale for 'x' is already present. Adding another scale for 'x', which will replace
the existing scale.
Scale for 'y' is already present. Adding another scale for 'y', which will replace
the existing scale.
Coordinate system already present. Adding new coordinate system, which will replace the
 existing one.
Warning message:
Removed 1 rows containing missing values (geom_rect).
```





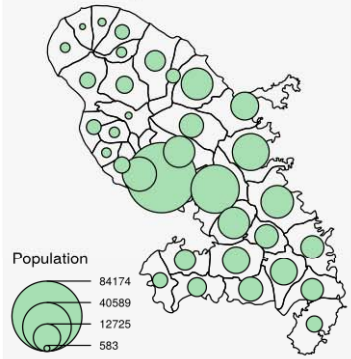
cartography: Thematic Cartography

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

Thematic maps with cartography : : CHEAT SHEET

Use cartography with spatial objects from sf or sp packages to create thematic maps.

```
library(cartography)
library(sf)
mtq <- st_read("martinique.shp")
plot(st_geometry(mtq))
propSymbolsLayer(x = mtq, var = "P13_POP",
 legend.title.txt = "Population",
 col = "#a7dfb4")
```



Classification

Available methods are: quantile, equal, q6, fisher-jenks, mean-sd, sd, geometric progression...

```
bks1 <- getBreaks(v = var, nclass = 6,
 method = "quantile")
bks2 <- getBreaks(v = var, nclass = 6,
 method = "fisher-jenks")
pal <- carto.pal("green.pal",3, "wine.pal", 3)
hist(var, breaks = bks1, col = pal)
```


```
hist(var, breaks = bks2, col = pal)
```

Symbology

In most functions the x argument should be an sf object. sp objects are handled through spdf and df arguments.

Choropleth
```
choroLayer(x = mtq, var = "myvar",
 method = "quantile", nclass = 8)
```

Typology
```
typoLayer(x = mtq, var = "myvar")
```

Proportional Symbols
```
propSymbolsLayer(x = mtq, var = "myvar",
 inches = 0.1, symbols = "circle")
```

Colorized Proportional Symbols (relative data)
```
propSymbolsChoroLayer(x = mtq, var = "myvar",
 var2 = "myvar2")
```

Colorized Proportional Symbols (qualitative data)
```
propSymbolsTypoLayer(x = mtq, var = "myvar",
 var2 = "myvar2")
```

Double Proportional Symbols
```
propTrianglesLayer(x = mtq, var1 = "myvar",
 var2 = "myvar2")
```

OpenStreetMap Basemap (see rosm package)
```
tiles <- getTiles(x = mtq, type = "osm")
tilesLayer(tiles)
```

Isopleth (see SpatialPosition package)
```
smoothLayer(x = mtq, var = "myvar",
 typefct = "exponential", span = 500,
 beta = 2)
```

Discontinuities
```
discLayer(x = mtq.borders, df = mtq,
 var = "myvar", threshold = 0.5
```

Flows
```
propLinkLayer(x = mtq_link, df = mtq_df,
 var = "fij")
```

Dot Density
```
dotDensityLayer(x = mtq, var = "myvar")
```

Labels
```
labelLayer(x = mtq, txt = "myvar",
 halo = TRUE, overlap = FALSE)
```

Transformations

Polygons to Grid
```
mtq_grid <- getGridLayer(x = mtq, cellsize = 3.6e+07,
  type = "hexagonal", var = "myvar")
```

Grids layers can be used by choroLayer() or propSymbolsLayer().

Points to Links
```
mtq_link <- getLinkLayer(x = mtq, df = link)
```

Links layers can be used by *LinkLayer().

Polygons to Borders
```
mtq_border <- getBorders(x = mtq)
```

Borders layers can be used by discLayer() function

Polygons to Pencil Lines
```
mtq_pen <- getPencilLayer(x = mtq)
```

Legends

legendChoro()
```
legendChoro(pos = "topleft",
 title.txt = "legendChoro()",
 breaks = c(0,20,40,60,80,100),
 col = carto.pal("green.pal", 5),
 nodata = TRUE, nodata.txt = "No Data")
```

legendTypo()
```
legendTypo(title.txt = "legendTypo()",
 col = c("peru", "skyblue", "gray77"),
 categ = c("type 1", "type 2", "type 3"),
 nodata = FALSE)
```

legendCirclesSymbols()
```
legendCirclesSymbols(var = c(10,100),
 title.txt = "legendCirclesSymbols()",
 col = "#a7dfb4ff", inches = 0.3)
```

See also legendSquaresSymbols(), legendBarsSymbols(), legendGradLines(), legendPropLines() and legendPropTriangles().

Map Layout

North Arrow:
```
north(pos = "topright")
```

Scale Bar:
```
barscale(size = 5)
```

Full Layout:
```
layoutLayer(
 title = "Martinique",
 tabtitle = TRUE,
 frame = TRUE,
 author = "Author",
 sources = "Sources",
 north = TRUE,
 scale = 5)
```

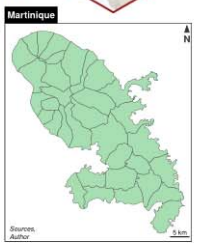
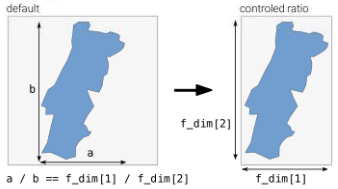


Figure Dimensions

Get figure dimensions based on the dimension ratio of a spatial object, figure margins and output resolution.

```
f_dim <- getFigDim(x = sf_obj, width = 500,
  mar = c(0,0,0,0))
png("fig.png", width = 500, height = f_dim[2])
par(mar = c(0,0,0,0))
plot(sf_obj, col = "#729fcf")
dev.off()
```



Color Palettes

```
carto.pal(pal1 = "blue.pal", n1 = 5,
 pal2 = sand.pal, n2 = 3)
```

```
display.carto.all(n = 8)
```

cartography - https://github.com/riatelab/cartography - CC BY SA Timothée Giraud - 2019-01