中研院統計所2015統計研習營
2015/07/01~03

(6) R統計圖形及資料視覺化

吳漢銘
淡江大學 數學系
資料科學與數理統計組

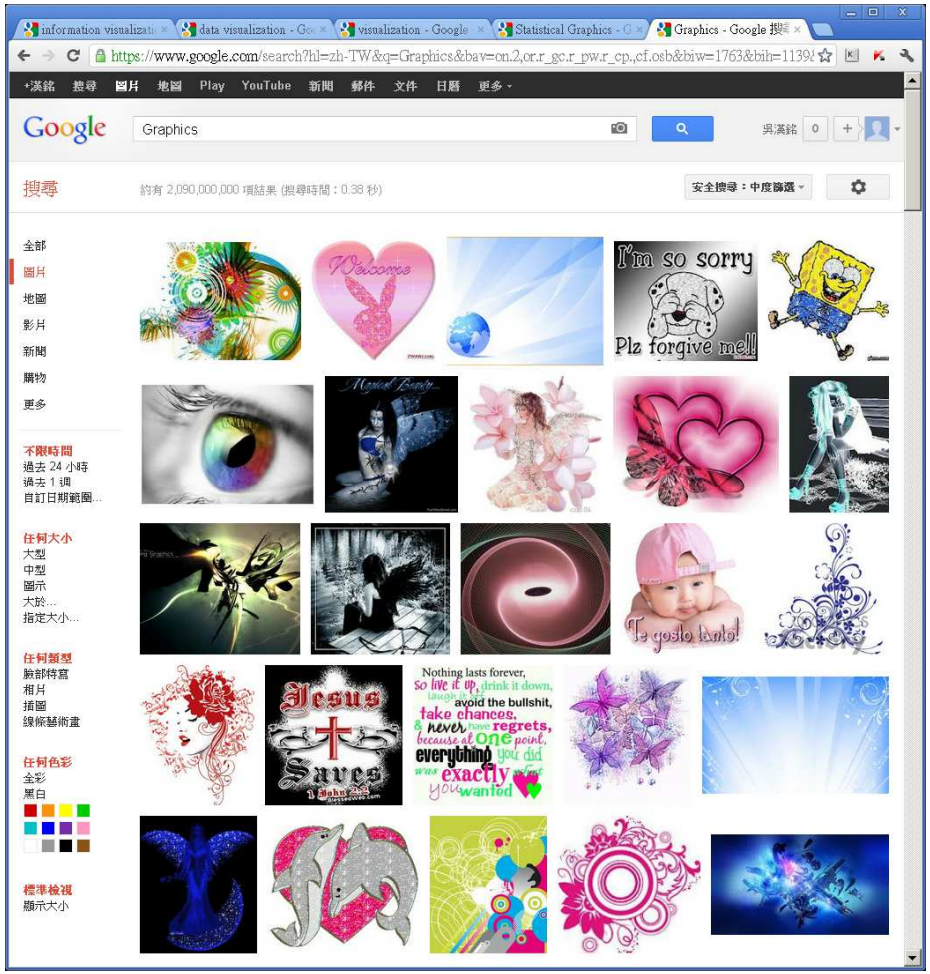
http://www.hmwu.idv.tw

本章大綱與學習目標

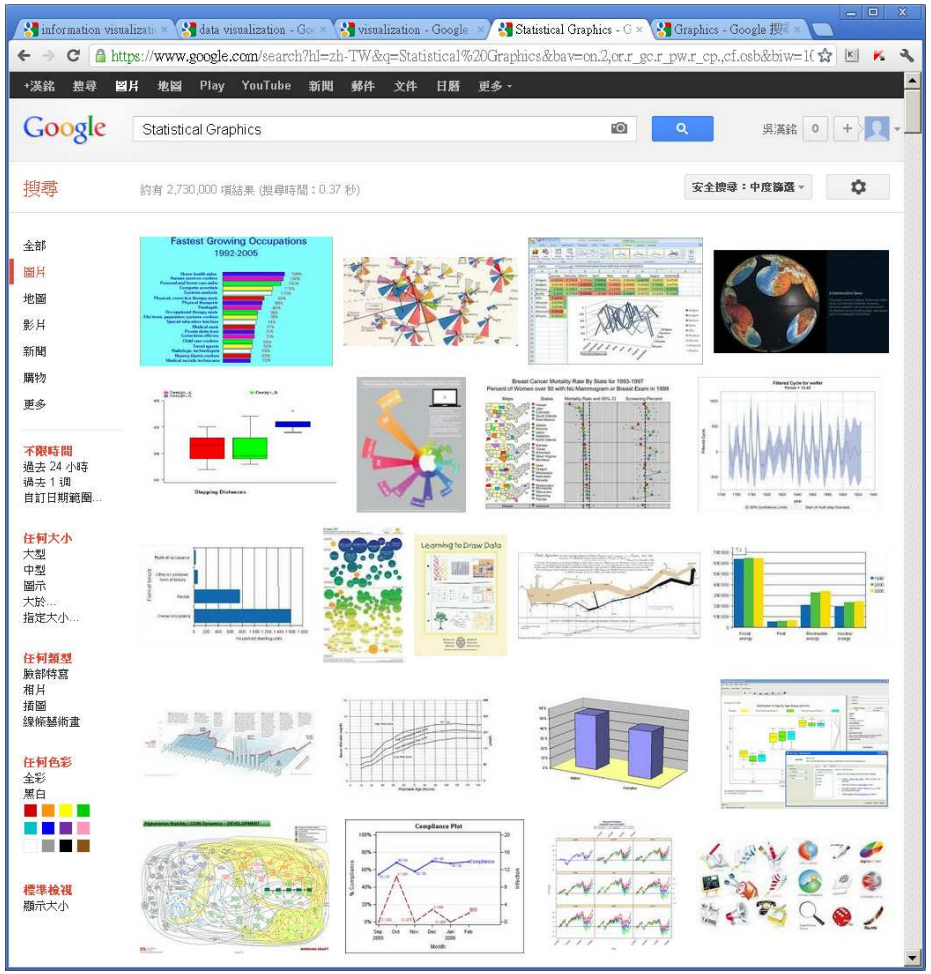
- **第一部份**
  - 簡介
  - R基礎統計圖形、圖形輸出與編輯
  - 單一樣本，雙樣本，多樣本的基礎統計圖形
- **第二部份**
  - 客制化基礎統計圖形
  - 顏色、資料符號、線條
  - 文字標號、圖例說明
  - 3D散佈圖、 影像、heatmap、等高線圖
- **第三部份**
  - Venn Diagrams、數學符號
  - ggplot2、Rgraphviz、igraph、畫 **Choropleth Maps**,
  - RgoogleMaps、maps, mapdata、googleVis、vcd
  - 參考書目

Graphics

Graphics

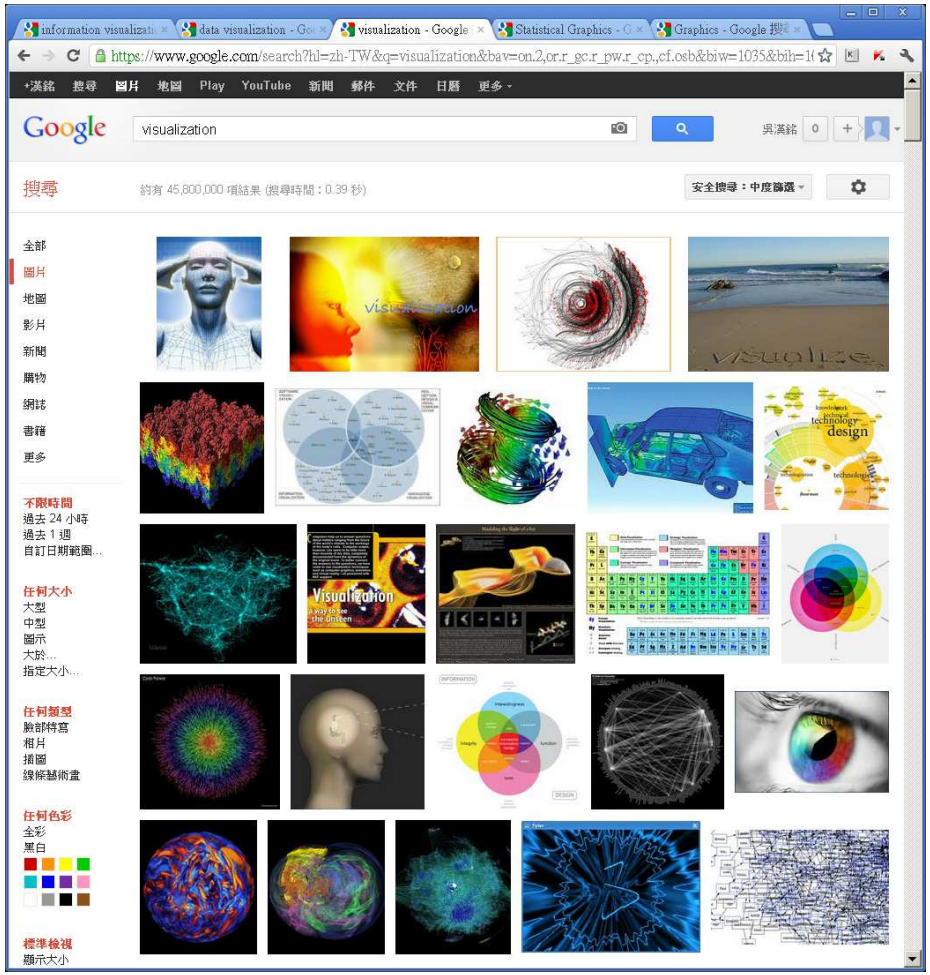


Statistical Graphics

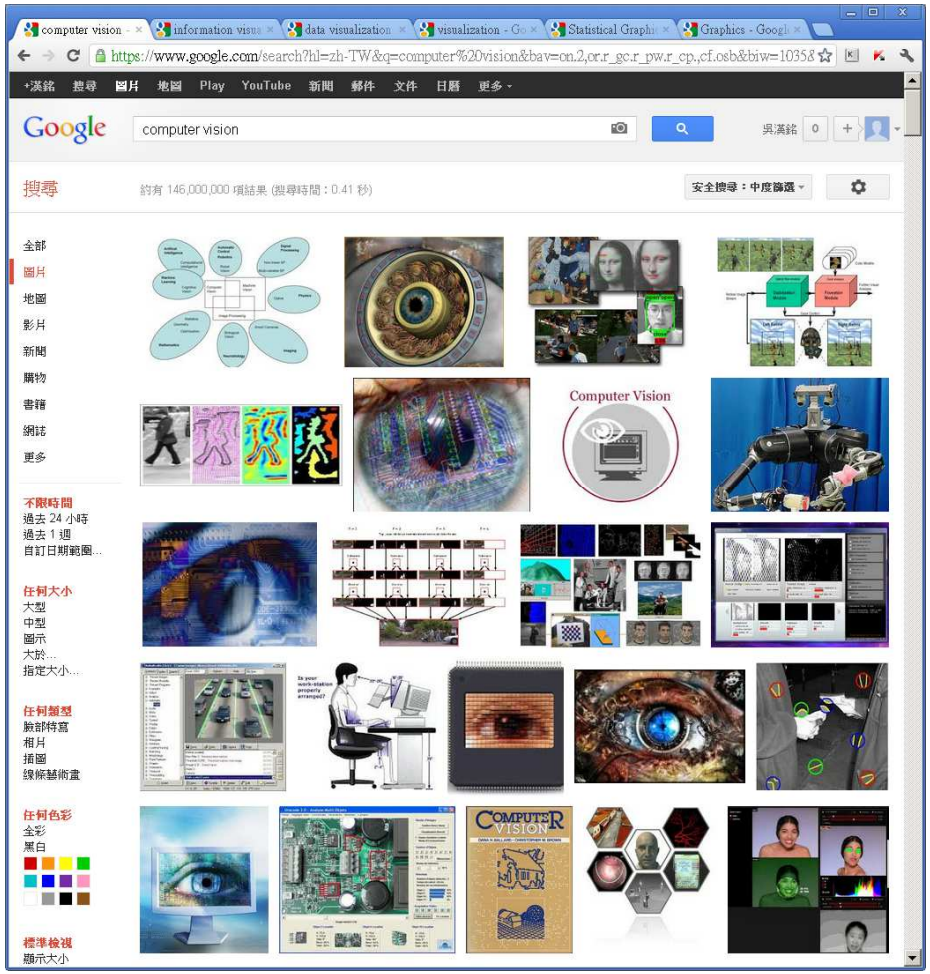


Visualization

Visualization

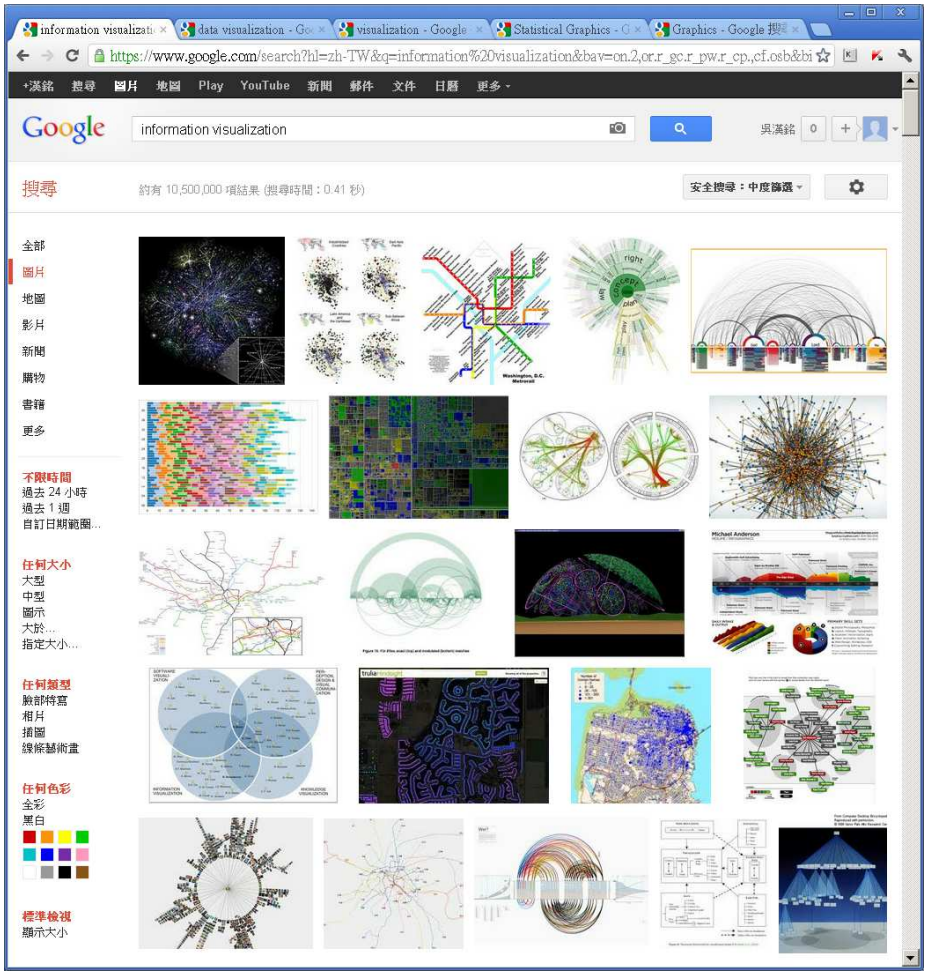


Computer Vision

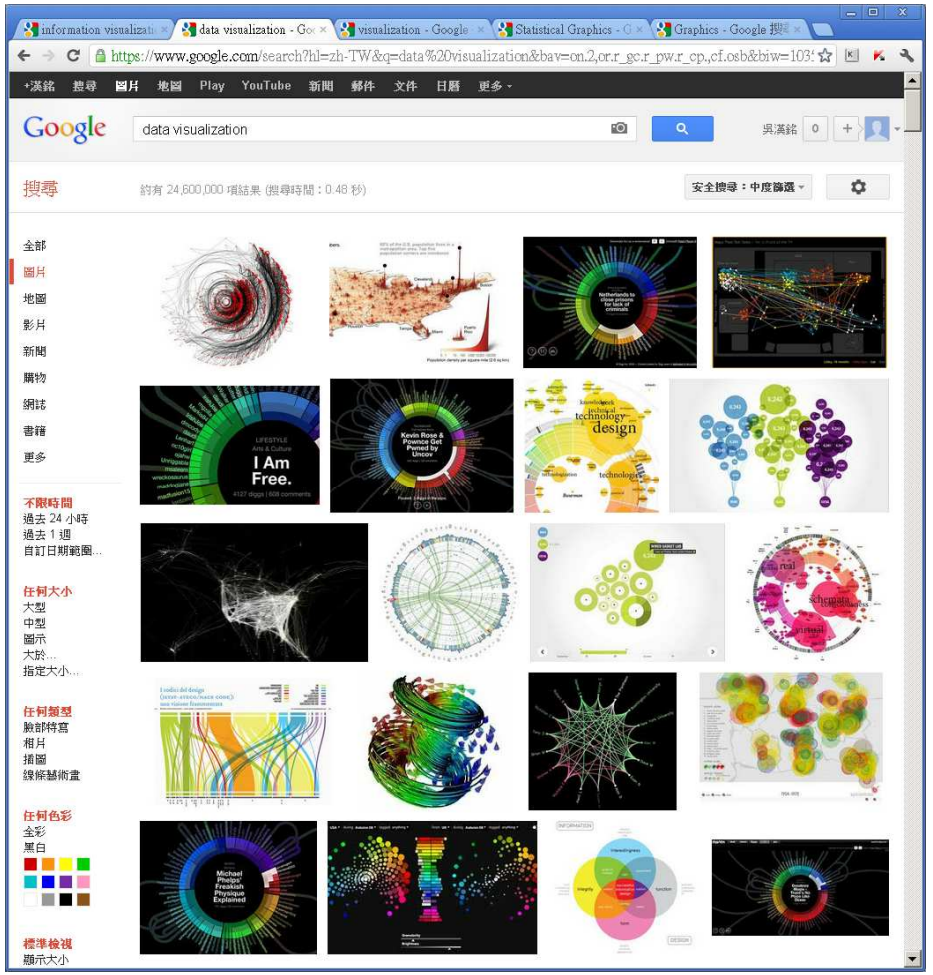


Data Visualization

Information Visualization



Data Visualization



What is Visualization?

People said

- Seeing is believing.
  (眼見為憑)
- Seeing is better than hearing a hundred times.
  (百聞不如一見)
- A picture is worth a thousand words.
  (一幅圖像勝過千言萬語)

The longest name of a city in New Zealand.

The shortest city name in the world is in Norway with one letter (A).

What is visualization?

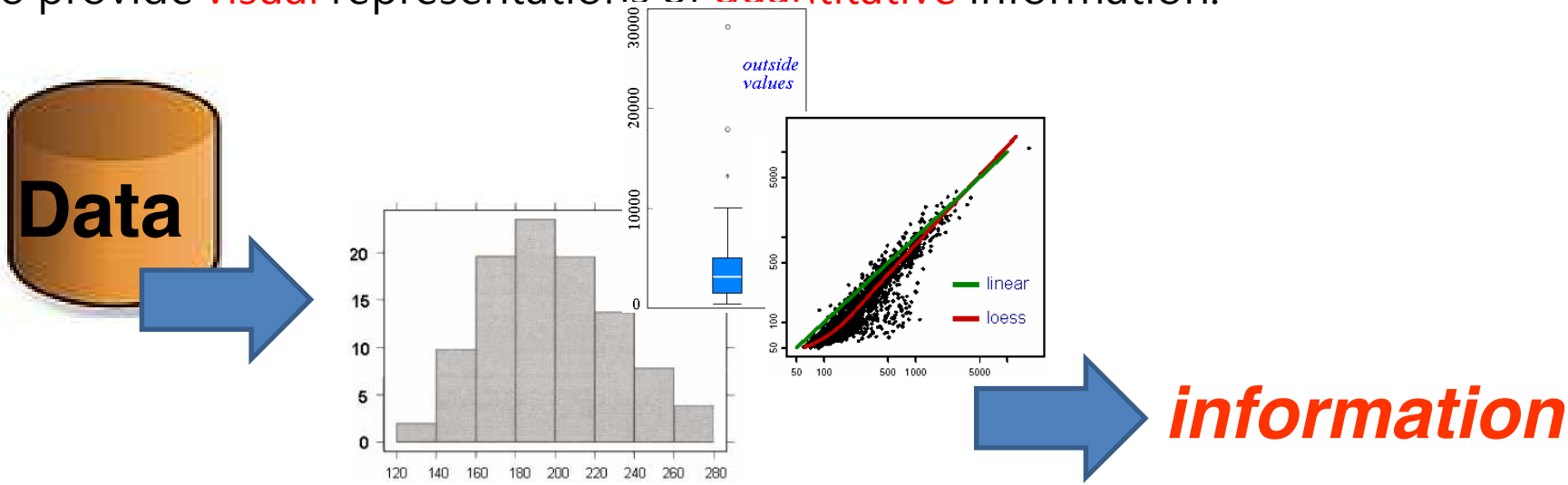
- Making things/processes/abstractions visible (to transform into pictures) that are not directly accessible by the human eye.
- Computer aided extraction and display of information from data.

Picture Source:
http://pictzz.blogspot.com/2009/03/pictures-is-worth-thousand-words.html

Graphical Methods

The purpose of statistical graphics is

to provide visual representations of quantitative information.



Exploratory Data Analysis (EDA) Tool

Statistical graphics comprise

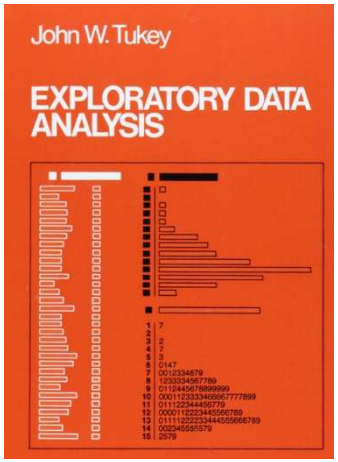
a set of strategies and techniques that provide the research with important insights about the data under examination and help guide the subsequent steps of the research process.

Visualization = Graphing for Data + Fitting + Graphing for Model

探索式資料分析
Exploratory Data Analysis, EDA

John Tukey (1915~2000) (統計學界的畢卡索)

「對正確的問題有個近似的答案，
勝過對錯的問題有精確的答案。」



- Summaries for large, complicated data sets.
- Revealing structure, patterns, features, trends, outliers, anomalies, and relationships in data .
- Extract important variables.
- Checking assumptions in statistical models.
- Interaction between the researcher and the data.
- Identifying the areas of interest.

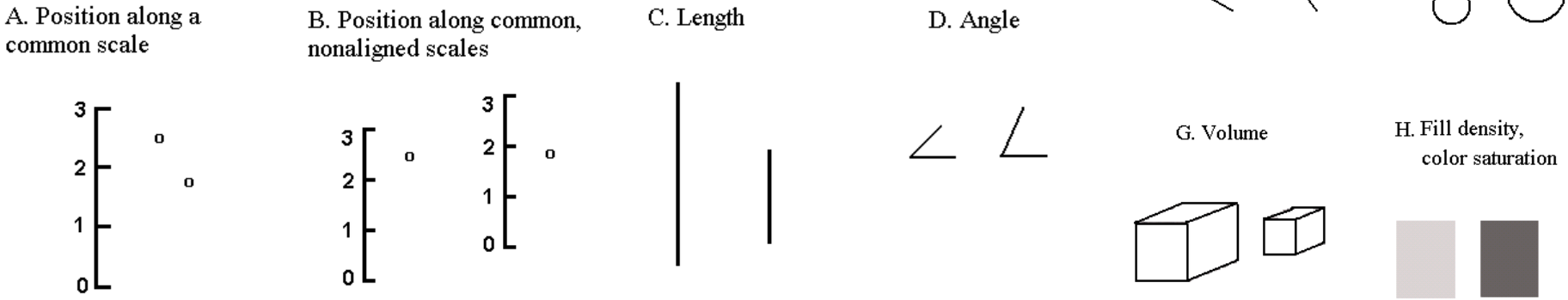
Visualization = Graphing for Data + Fitting + Graphing for Model

Graphical Perception

Human reception and comprehension of graphical information involves three fundamental perceptual task:

- Detection: the visual recognition of a geometric aspect that encodes a physical value. The basic information from the data must be discernible in the graph.
- Assembly: the process of discerning patterned regularities among the discrete elements of a graphical display.
- Estimation: the visual assessment of the relative magnitudes of two or more quantitative physical values.

Graphical Perception Tasks.
Ordered from the most accurate to the least accurate (Jacoby, 1997)

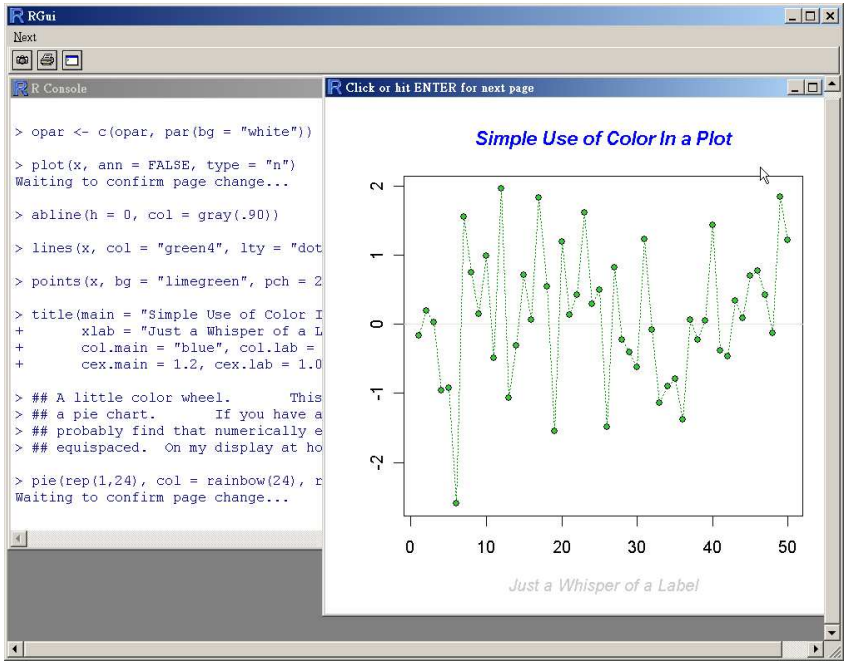


Demo: graphics

The R Graphics Package

```
> library(graphics)
> demo(graphics)#常見圖形
> demo(Hershey) #各種符號
> demo(image) #image和contours
> demo(Japanese) #日本字
> demo(persp) #曲面圖
> demo(plotmath) #數學符號
```

assocplot	Association Plots
Axis	Generic Function to Add an Axis to a Plot
axis	Add an Axis to a Plot
axis.POSIXct	Date and Date-time Plotting Functions
axTicks	Compute Axis Tickmark Locations
barplot	Bar Plots
box	Draw a Box around a Plot
boxplot	Box Plots
boxplot.matrix	Draw a Boxplot for each Column (Row) of a Matrix



程式碼:

C:\Program Files\R\R-3.x.x\library\graphics\demo

作圖準則: First calling a high-level function that creates a complete plot, then calling low-level functions to add more output if necessary.

R Graphics學習資源

- CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
  http://cran.r-project.org/web/views/Graphics.html
- R Graphics
  http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html
- R Graphics Gallary
  http://research.stowers-institute.org/efg/R/index.htm
  http://addictedtor.free.fr/graphiques/
- R Graphical manuals
  http://cged.genes.nig.ac.jp/RGM2/
- gRaphics! http://grrrraphics.blogspot.tw/

R 圖形

- Producing high-quality graphics is one of the main reasons for doing statistical computing.
- Plot function
  - Number of variables to plot
    - a single sample
    - two variables
    - multivariate plots
  - The pattern to highlight
  - Special plots for particular purpose
- First calling a high-level function that creates a complete plot, then calling low-level functions to add more output if necessary.

裝置函式 (Device Function)

Table 1.1
Graphics formats that R supports and the functions that open an appropriate graphics device

Device Function	Graphical Format
Screen/GUI Devices	
`x11()` or `X11()`	X Window window
`windows()`	Microsoft Windows window
`quartz()`	Mac OS X Quartz window
File Devices	
`postscript()`	Adobe PostScript file
`pdf()`	Adobe PDF file
`pictex()`	LaTeX PicTeX file
`xfig()`	XFIG file
`bitmap()`	GhostScript conversion to file
`png()`	PNG bitmap file
`jpeg()`	JPEG bitmap file
(Windows only)	
`win.metafile()`	Windows Metafile file
`bmp()`	Windows BMP file
Devices provided by add-on packages	
`devGTK()`	GTK window (`gtkDevice`)
`devJava()`	Java Swing window (`RJavaDevice`)
`devSVG()`	SVG file (`RSvgDevice`)

Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition

dev()

- Only one device is currently active and all graphics output is sent to that device.
- `dev.list()`
  #list open devices
- `dev.cur()`
  #return this information only for the currently active device
- `dev.set(number)`
  # set a device active by a number
- `dev.copy()`
  # copies all output from the active device to another device.
- `dev.next(), dev.prev()`
  #make the next/previous device on the device list the active device.
- `dev.off(number)`
  #close

```
> dev.list()
NULL
> plot(iris[,1])
> dev.list()
windows
      2
> dev.cur()
windows
      2
> windows()
> dev.set(2)
windows
      2
> dev.copy()
windows
      3
>
```

圖形輸出

Read in Data → Plot → Save → Edit → Finished

Example

方法1: Save as ...

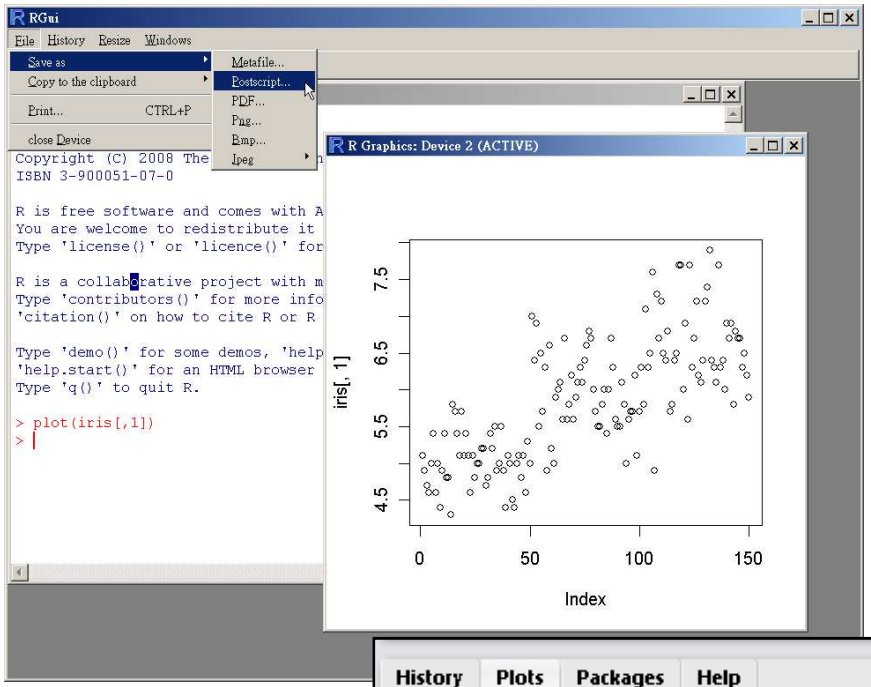
```
> plot(iris[,1])
```

方法2 : Activate device

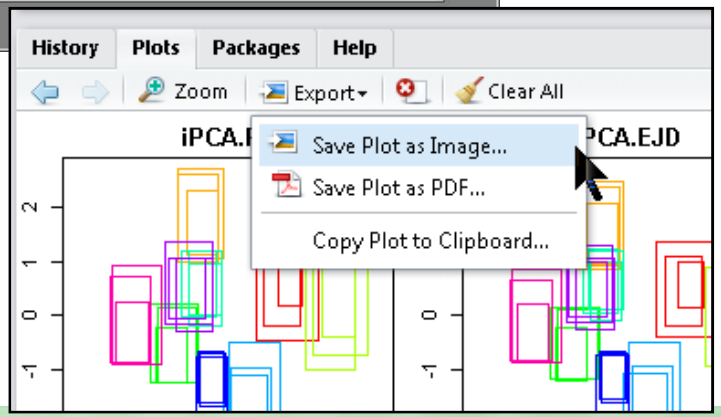
```
> postscript(file="iris2.ps")
> plot(iris[,2])
> dev.off()
```

or

```
> jpeg(file="iris3.jpg")
> plot(iris[,3])
> dev.off()
```



RGui



RStudio

向量圖與非向量圖格式

存檔格式及目的

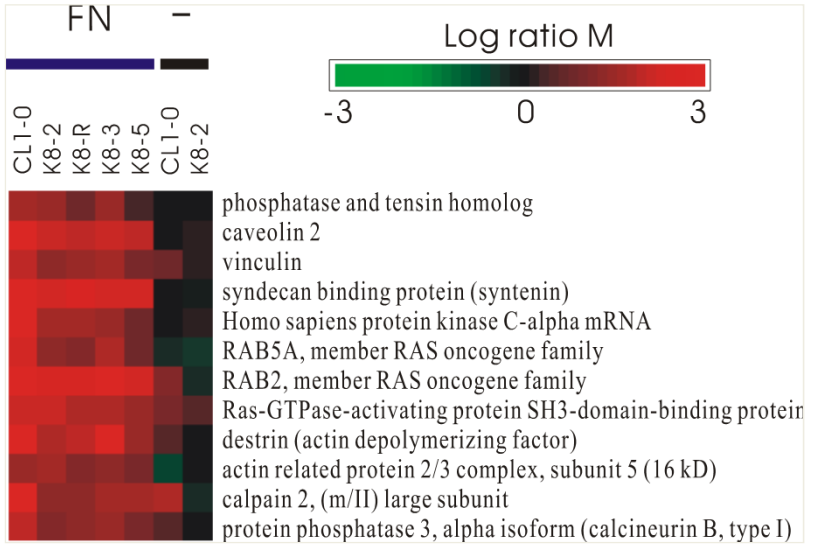
- 點陣圖檔 (壓縮失真: PNG, JPEG, TIFF, BMP) : web page, powerpoint, word,...
- 向量圖檔 (物件不失真: Metafile, SVG, EPS) : Publication, LaTeX.

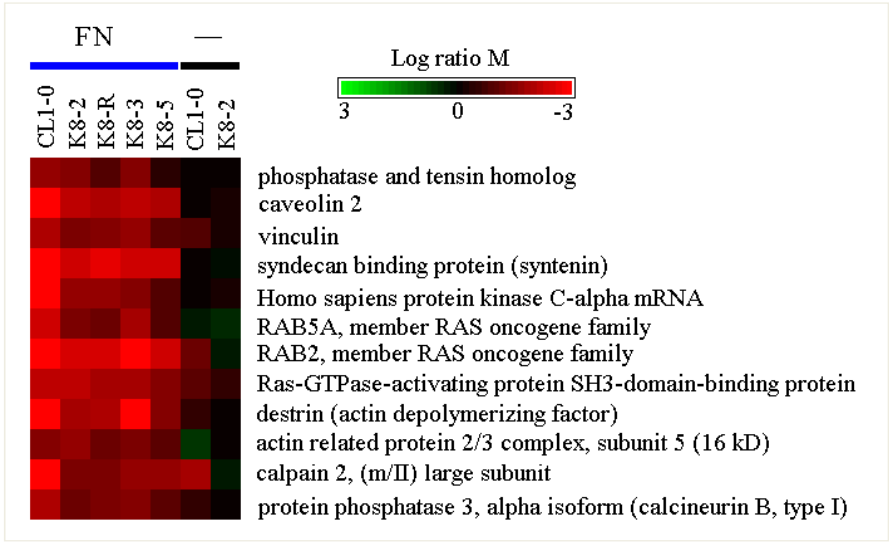
向量圖(在此為wmf格式):

可拉大或縮小不失真。
此時在營幕看起來可能會有點糊(平滑化結果)，其實列印出來會很清楚的。

點陣圖(在此為bmp格式):

與向量圖(在營幕上看的)最大差別是拉大縮小有明顯的鋸齒狀。此圖插入powerpoint後會被平滑化，所以看起來會霧霧的。





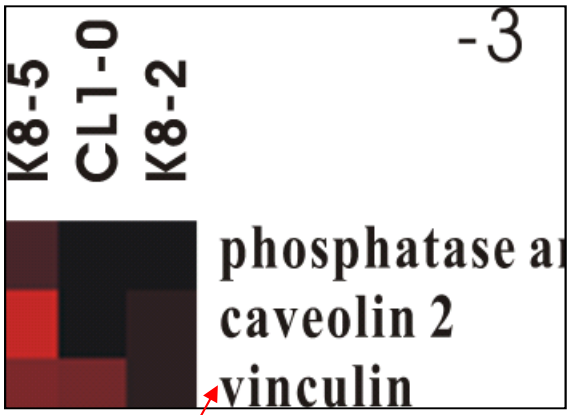
向量圖與非向量圖格式

如何放大圖片:

1.選按圖片=>滑鼠右鍵=>設定圖格式=>大小=> 縮放比例=>高度設為300% => OK

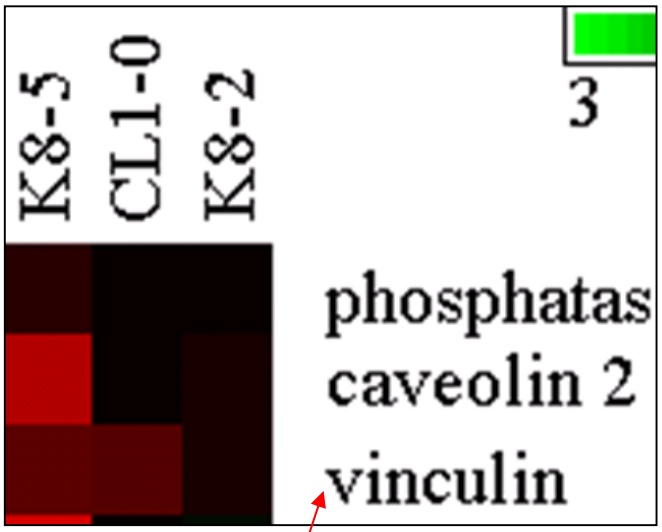
2. 這跟放大投影片(或全營幕)成300%是不一樣的。

向量圖放大變這樣:



無鋸齒狀, 邊界會糊糊的(或霧霧的是軟體自動平滑化的結果)

點陣圖放大變這樣:



有鋸齒狀

多重輸出 (Multiple Output)

- PostScript and PDF allow multiple pages.
- PNG does not.

```
for(i in 1: 4){
    name <-paste("iris",i, ".jpg", sep="")
    jpeg(name, width=800, height=800)
    plot(iris[,i])
    dev.off()
}
```

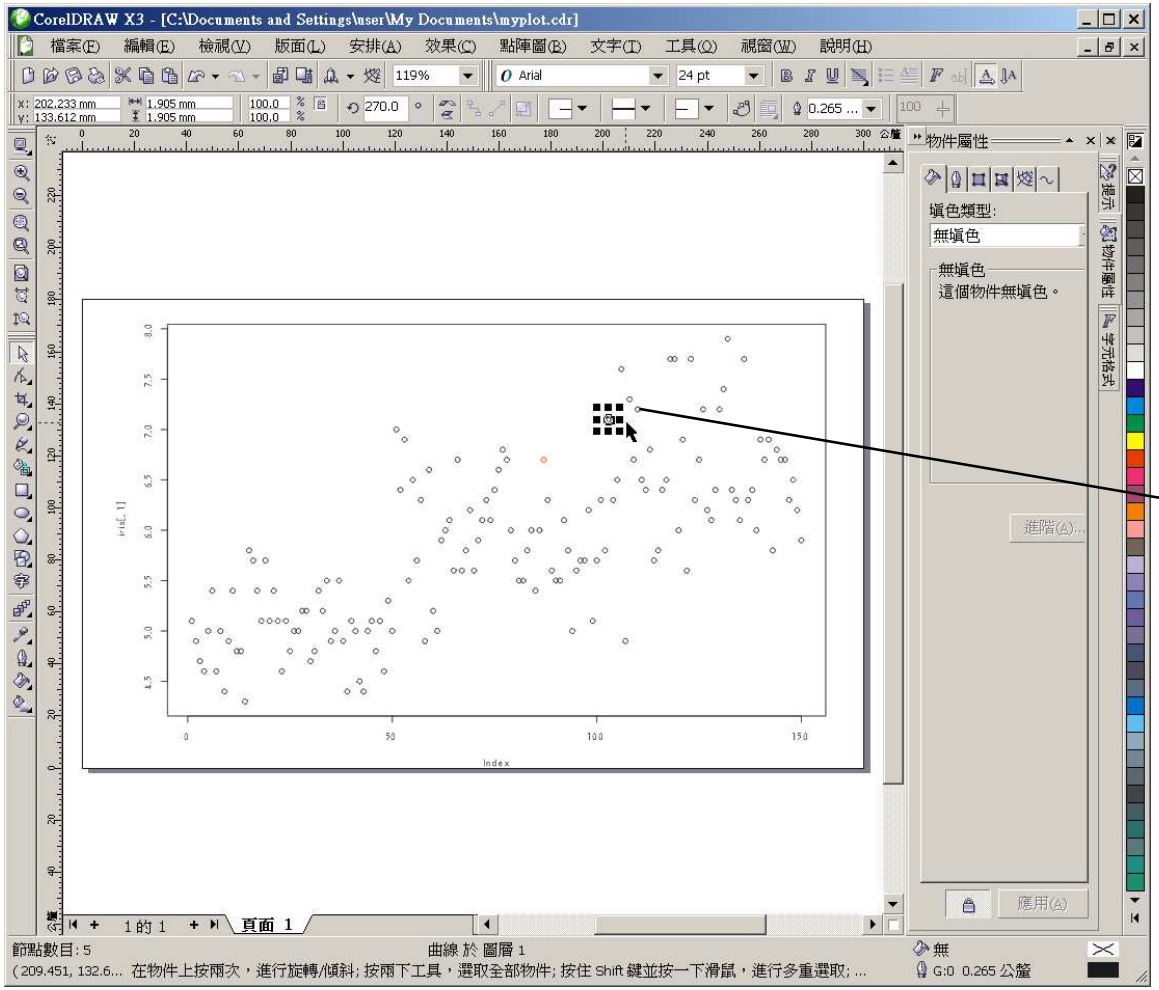
the %03d is replaced by a three-digit number indicating the page number for each file.

```
pdf("myplot%03d.pdf", onefile = FALSE)
for(i in 1: 4){
   plot(iris[,i])
}
dev.off()
```

```
pdf("myplot.pdf", onefile = TRUE)
for(i in 1: 4){
   plot(iris[,i])
}
dev.off()
```

向量圖編輯

Read in Data → **Plot** → **Save** → **Edit** → **Finished**



Example: CorelDraw

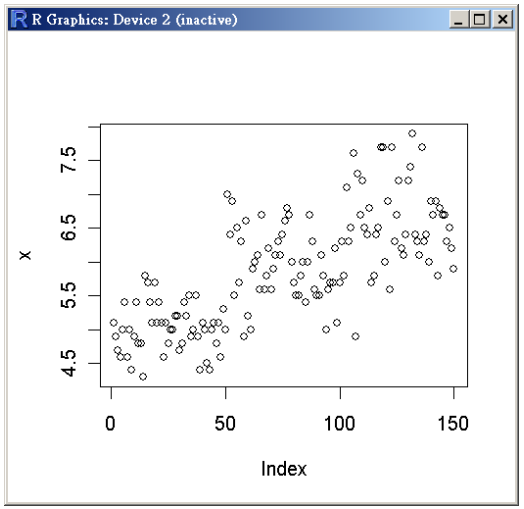
Plots for Single Samples

- 開啟一個繪圖視窗

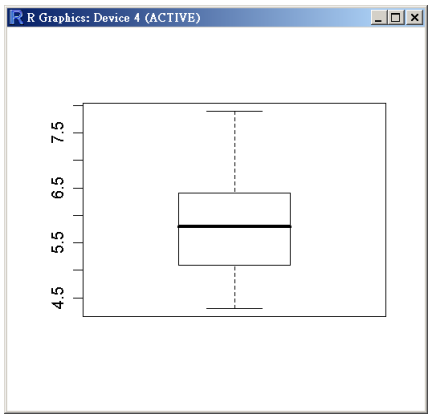
```
> windows()
> x11()
```

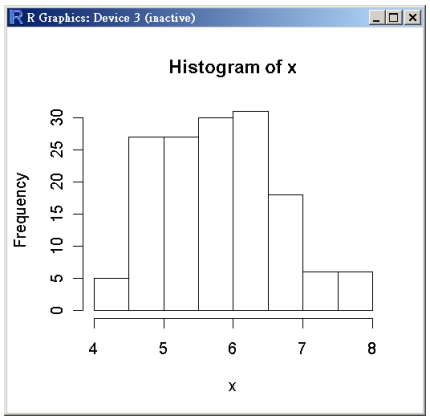
- 多張圖(mxn)一頁

```
> par(mfrow=c(m,n))
```



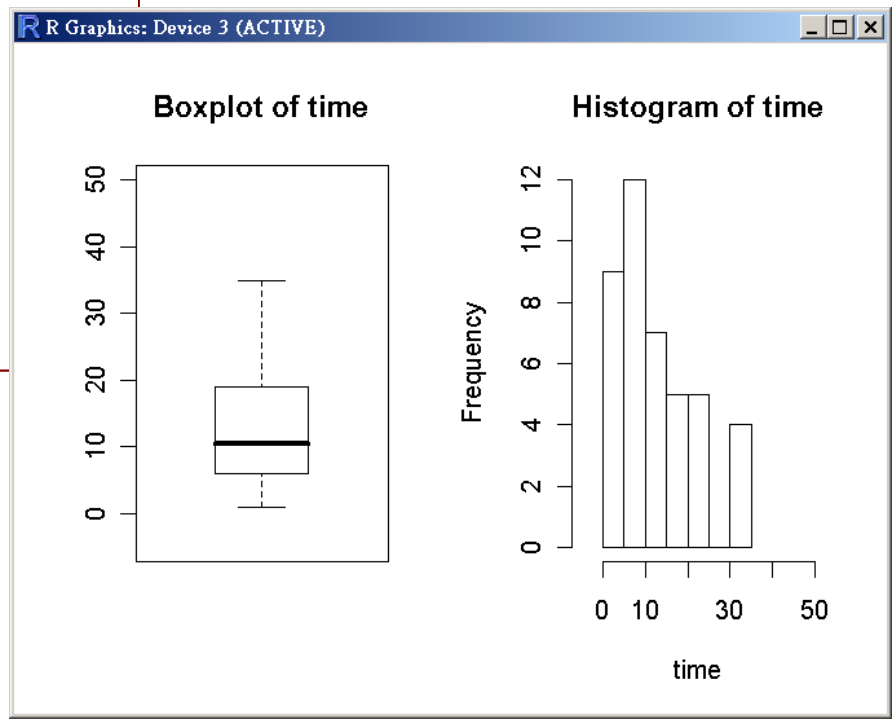

```
> x <- iris[,1]
> windows()
> plot(x) #index plots to show the values of y in sequence
> windows()
> hist(x) #histogram to show a frequency distribution
> windows()
> boxplot(x)
```





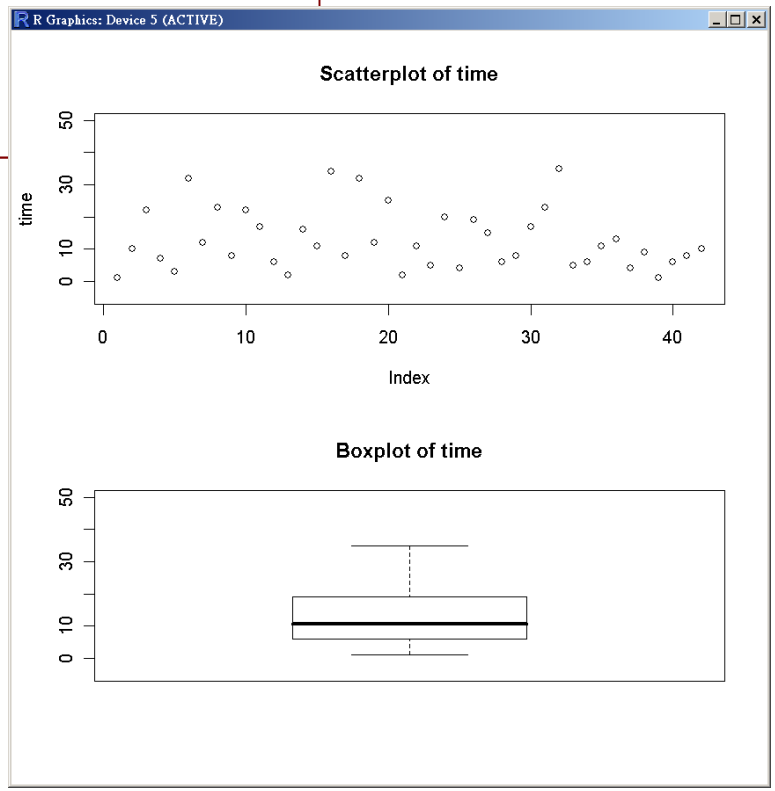
圖標題和範圍 (Title and Limit)

```
> library(MASS)
> data(gehan)
> time <- gehan$time
> windows()
> par(mfrow=c(1,2))
> boxplot(time, ylim=c(-5, 50))
> title("Boxplot of time")
> hist(time, xlim=c(-5, 50))
> title("Histogram of time")
```



圖標題和範圍 (Title and Limit)

```
> windows()
> par(mfrow=c(2,1))
> plot(time, ylim=c(-5, 50), main="Scatterplot of time")
> boxplot(time, ylim=c(-5, 50), main="Boxplot of time")

#or
> windows()
> par(mfrow=c(2,1))
> s.title <- "Scatterplot of time"
> plot(time, ylim=c(-5, 50), main=s.title)
> b.title <- "Boxplot of time"
> boxplot(time, ylim=c(-5, 50), main= b.title)
```



標準引數 (Standard Arguments)

- Standard Arguments: many high-level plot functions accept them.

```
> y <- iris[,1]
```

- **type="l"**: lines, **lwd**: line width

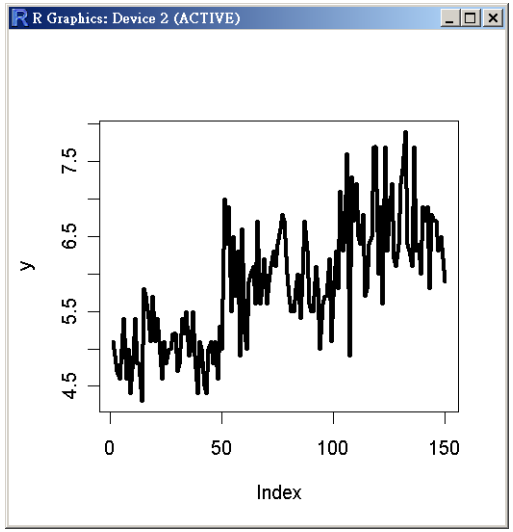
```
> plot(y, type="l", lwd=3)
```

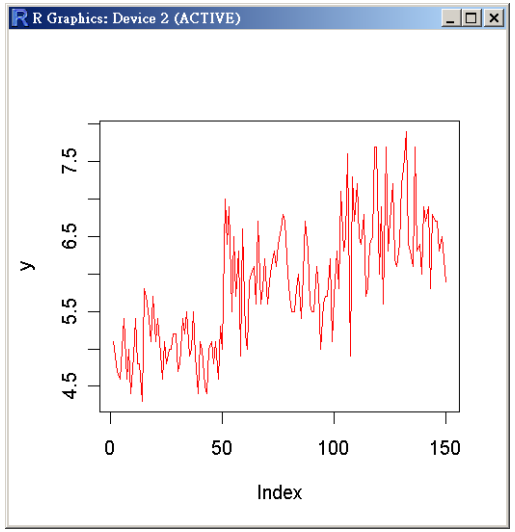
- col: color 顏色

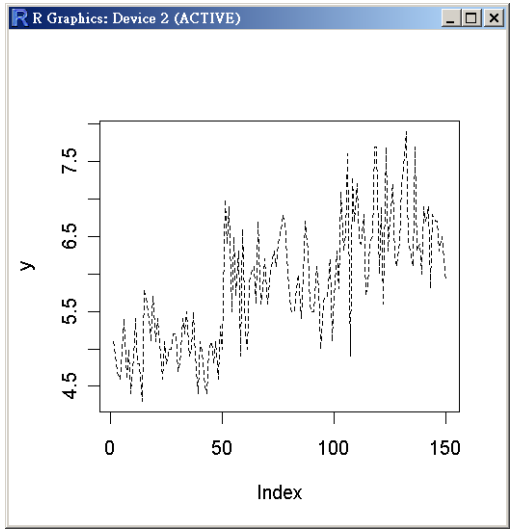
```
> plot(y, type="l", col="red")
```

- lty: line type 線的型式

```
> plot(y, type="l", lty="dashed")
```

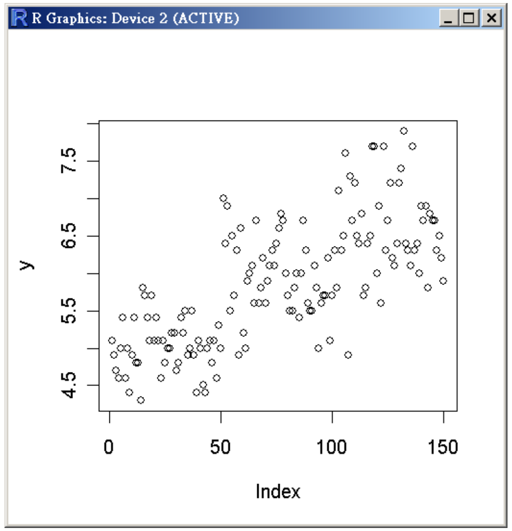


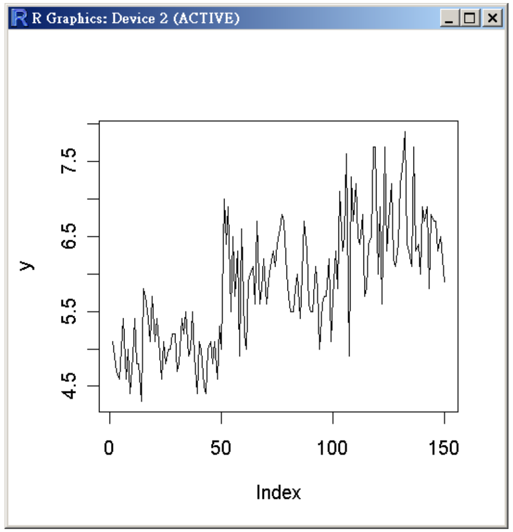


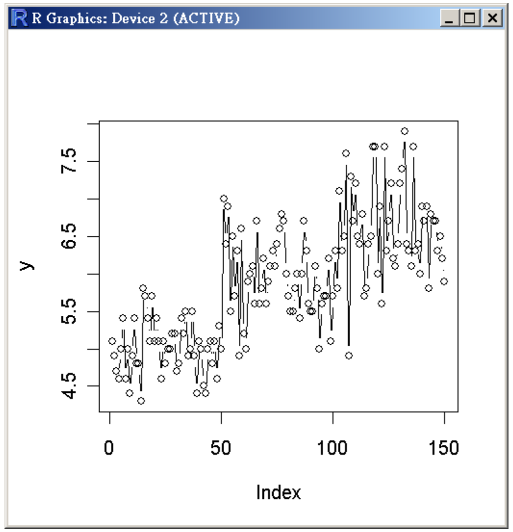


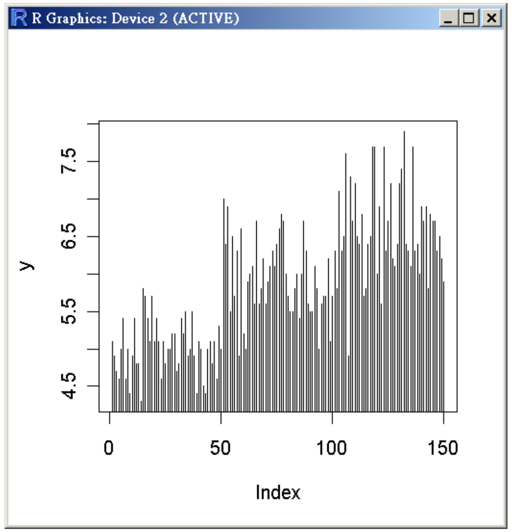
線的型式 (Line Type)

```
> y <- iris[,1]
> plot(y, type="p")
> plot(y, type="l")
> plot(y, type="b")
> plot(y, type="h")
> plot(y, type=“n")
```







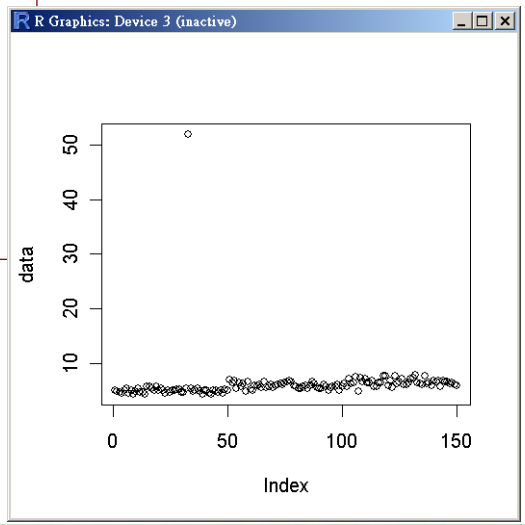


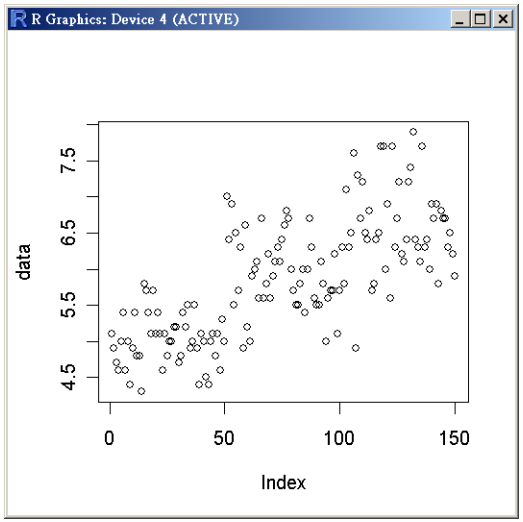
索引圖 (Index Plot)

- Index plot takes a single argument which is a continuous variable and plots the values on the y axis, with the x coordinate determined by the position of the number in the vector.

- Useful for error checking.

```
> data <- iris[,1]
> data[33] <- data[33]*10
> plot(data)
> ind <- which(data>15)
> data[ind]
> data[ind] <- 5.2
> windows()
> plot(data)
```



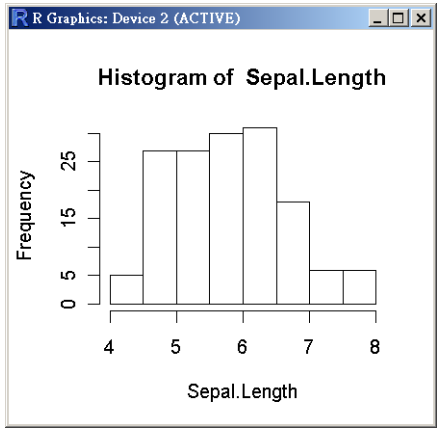


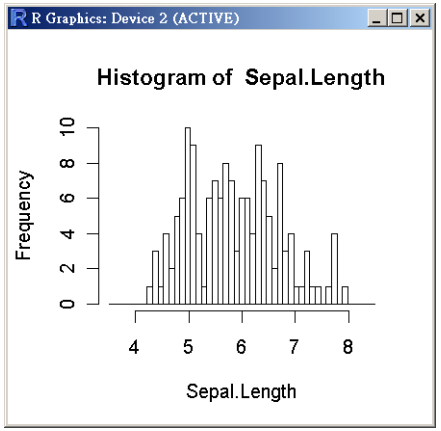
直方圖 (Histogram)

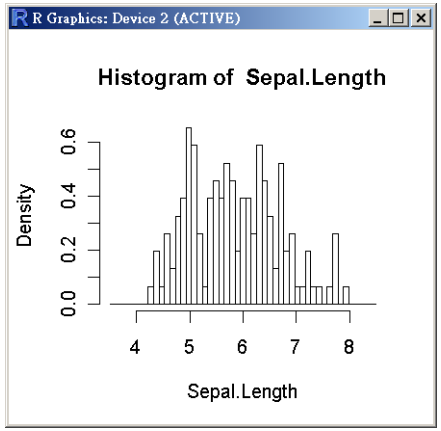
- Histogram are excellent for showing the mode, the spread and the symmetry (skew) of a set of data.

- Arguments
  - **breaks**
  - **pro=F** #default, 次數, **pro=T** #機率值

```
> lab <- names(iris)[1]
> title <- paste("Histogram of ", lab)
> hist(iris[,1], main=title, xlab=lab)
> range(iris[,1])
> hist(iris[,1], breaks=seq(3.5, 8.5, length=50),main=title, xlab=lab)
> hist(iris[,1], breaks=seq(3.5, 8.5, length=50),main=title, xlab=lab, pro=T)
```

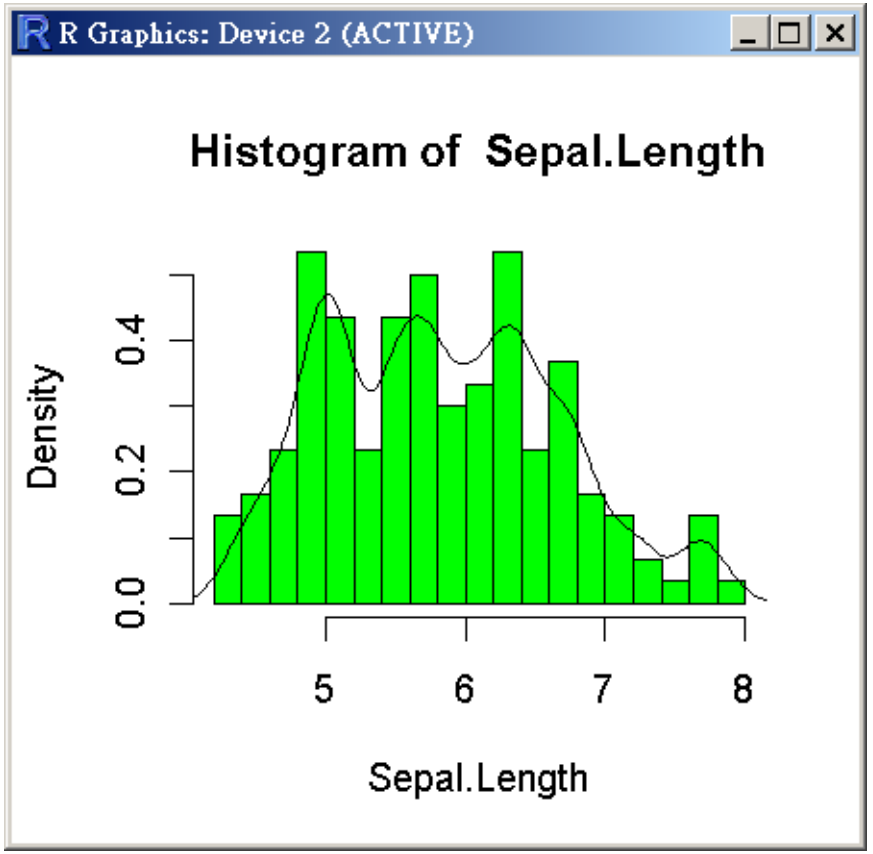






Density Estimation for Continuous Variables

```
> hist(iris[,1], breaks=15, main=title, xlab=lab, col="green", pro=T)

> lines(density(iris[,1], width=0.6, n=200))
```

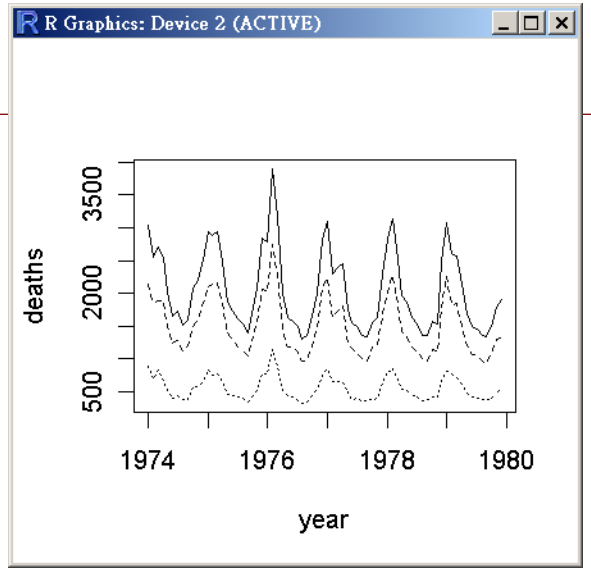


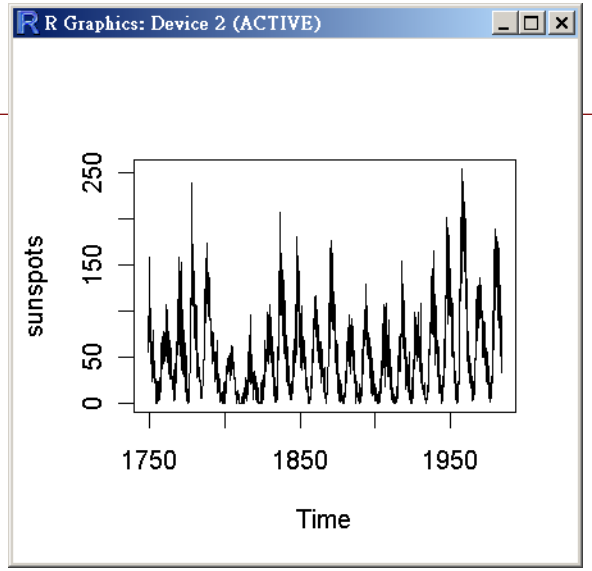
時間序列 (Time Series Plots)

- **ts.plot**: simple vector of numbers
- **plot.ts** works for plotting objects inheriting from class ts.

```
> data(UKLungDeaths) #total, male, female death
> ts.plot(ldeaths, mdeaths, fdeaths, xlab="year", ylab="deaths", lty=c(1:3))

> data(sunspots)
> plot(sunspots) #sunspots is ts class
> class(sunspots)
[1] "ts"
> is.ts(sunspots)
[1] TRUE
```

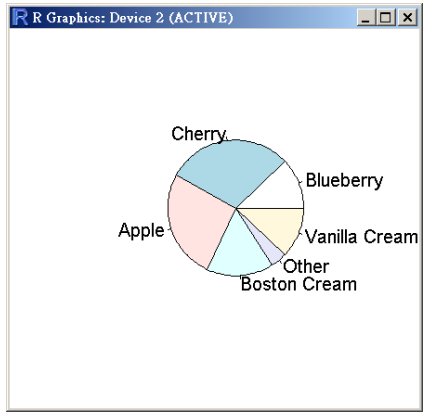
data(UKgas)
attach(UKgas)
names(UKgas)



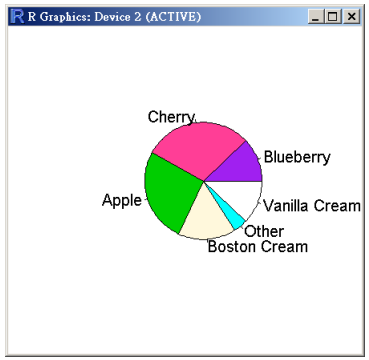


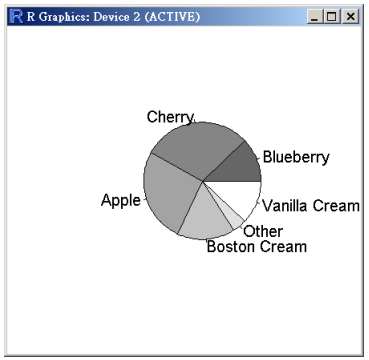
圓餅圖 (Pie Charts)

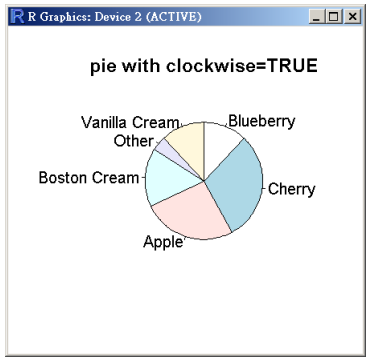
```
> pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
> sum(pie.sales)
> names(pie.sales) <- c("Blueberry", "Cherry",
    "Apple", "Boston Cream", "Other", "Vanilla Cream")

> pie(pie.sales) # default colours
```

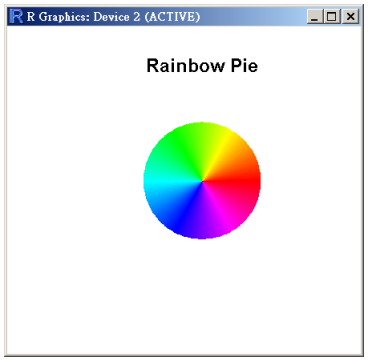



```
> pie(pie.sales, col = c("purple", "violetred1", "green3", "cornsilk", "cyan", "white"))
> pie(pie.sales, col = gray(seq(0.4,1.0,length=6)))
> pie(pie.sales, clockwise=TRUE, main="pie with clockwise=TRUE")
> pie(rep(1,200), labels="", col=rainbow(200), border=NA, main = "Rainbow Pie")
```



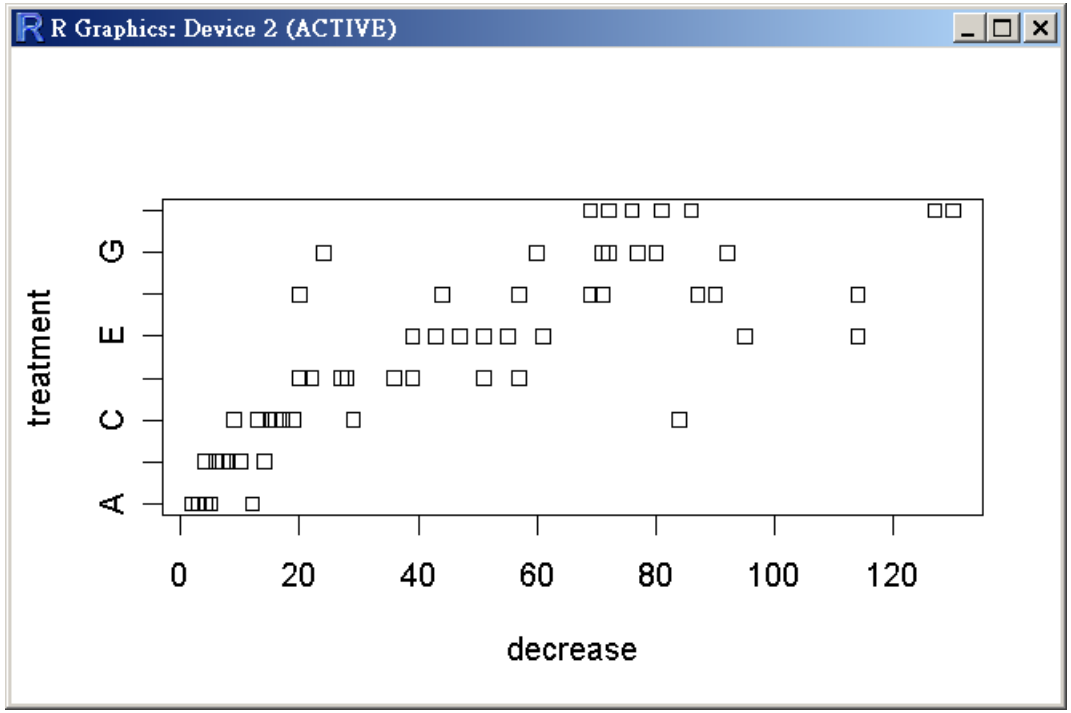






帶形圖 (Stripchart)

```
> attach(OrchardSprays)
> names(OrchardSprays)
[1] "decrease"  "rowpos"    "colpos"    "treatment"
> OrchardSprays[1:5,]
> stripchart(decrease~treatment, xlab="decrease", ylab="treatment")
```



Plots with Two Variables

- Response variables on the y-axis, explanatory variable on the x-axis.
  - explanatory variable is continuous => scatterplot.
  - explanatory variable is discrete => box-and-whisker plot, barplot

- `plot(x,y)`
  #scatterplot of y against x

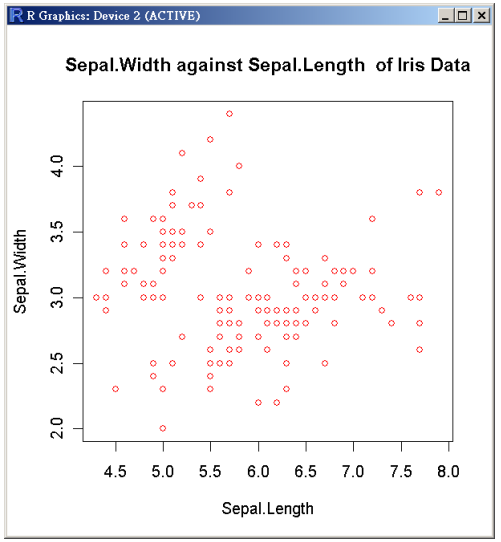
- `plot(factor, y)`
  #box-and-whisker plot of y at level of factor

- `boxplot()`
- `barplot(y)`
  #heights from a vector of y values

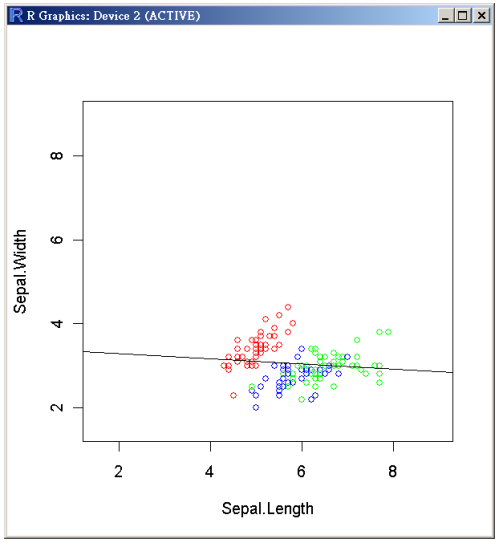
散佈圖 (Scatterplot)

plot(x, y) 或 plot(y~x)

```
> xlab <- names(iris)[1]
> ylab <- names(iris)[2]
> title <- paste(ylab, "against", xlab, " of Iris
    Data")
> x <- iris[,1]
> y <- iris[,2]
> plot(x, y, col="red", xlab=xlab, ylab=ylab,
    main=title)
```



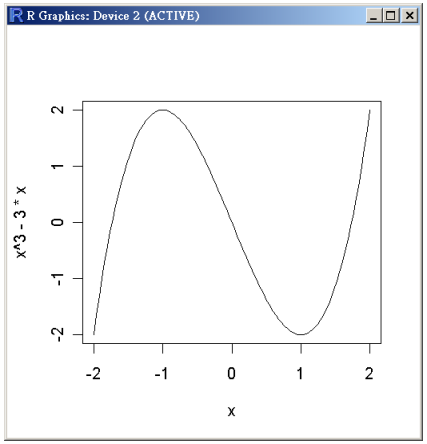

```
> range(x)
> range(y)
> plot(y~x, xlab=xlab, ylab=ylab, xlim=c(1.5,9),
     ylim=c(1.5,9), type="n")
> points(x[1:50], y[1:50], col="red")
> points(x[51:100], y[51:100], col="blue")
> points(x[101:150], y[101:150], col="green")
> abline(lm(y~x))
```



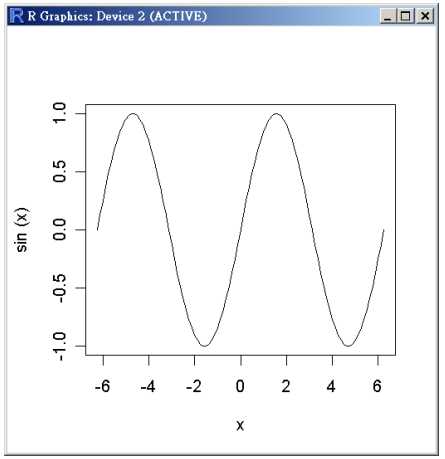
畫2D數學函數圖

- Show the response as a smooth curve.
- Plot f(x) = x^3-3x between x=-2 and x=2.

```
> curve(x^3-3*x, -2, 2)
```



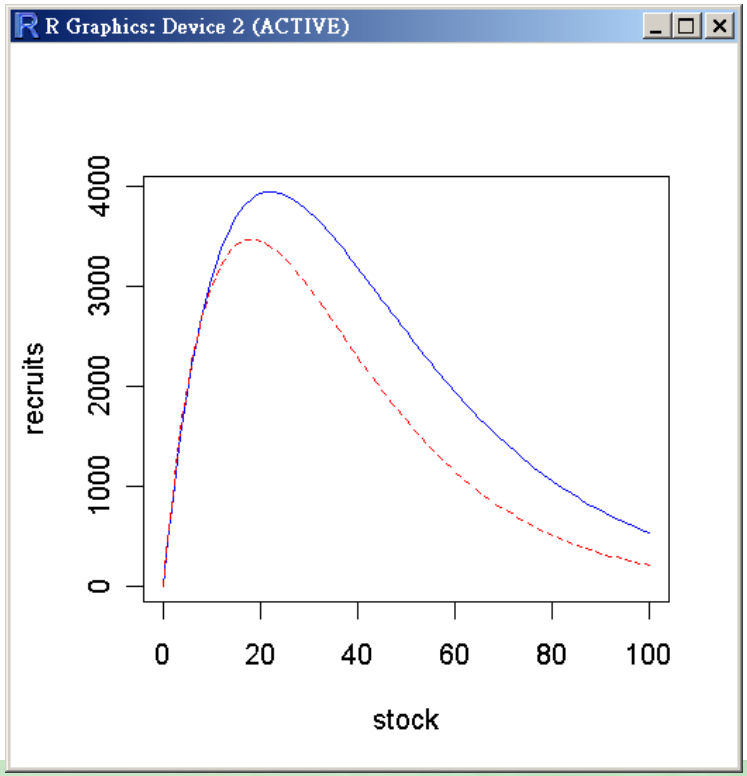

```
> curve(sin, -2*pi, 2*pi)
```




```
> x <- seq(-2, 2, 0.01)
> y <- x^3-3*x
> plot(x, y, type="l")
```

兩曲線在一張圖 (Two Curves)

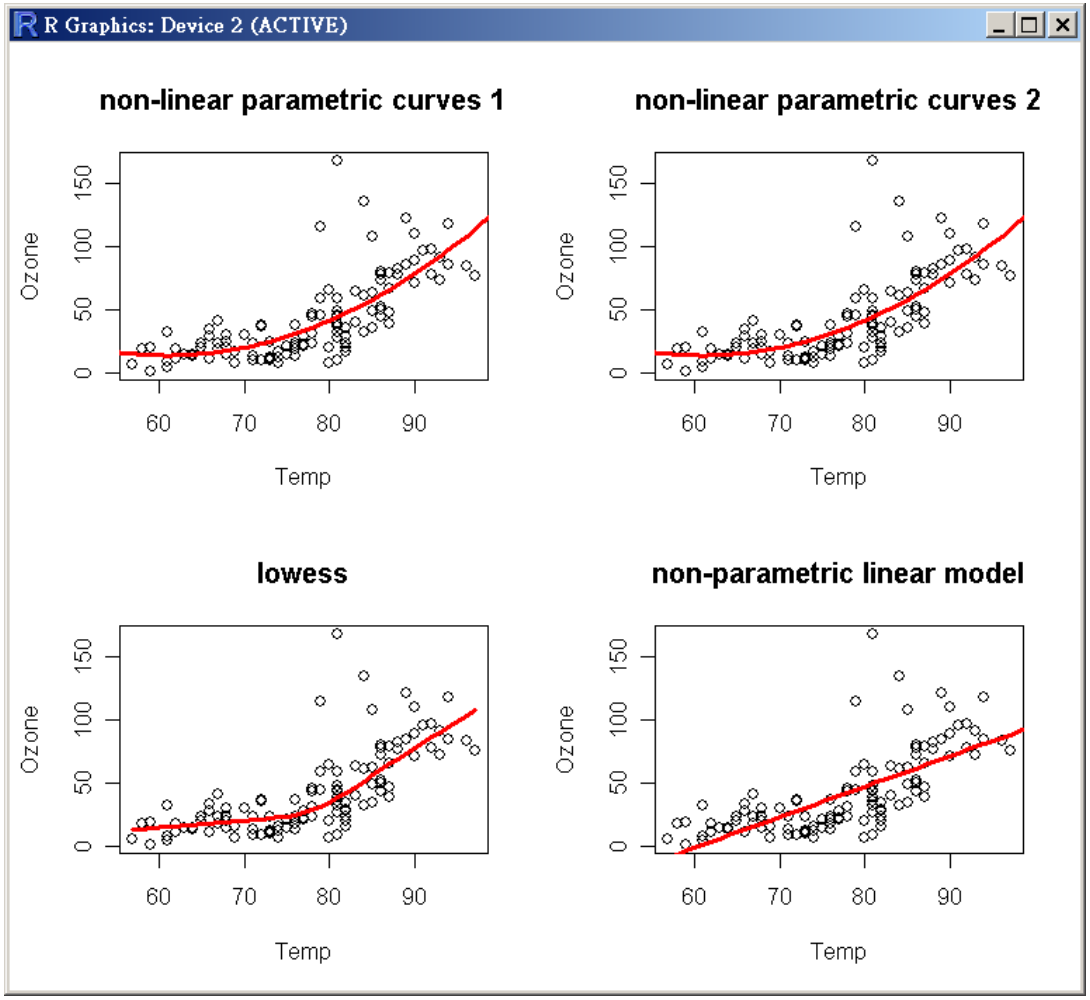
```
> xv <- 0:100
> yA <- 482*xv*exp(-0.045*xv)
> yB <- 518*xv*exp(-0.055*xv)
> plot(c(xv, xv), c(yA, yB), xlab="stock", ylab="recruits", type="n")
> lines(xv, yA, lty=1, col="blue")
> lines(xv, yB, lty=2, col="red")
```



曲線配適 (Fitting Curves)

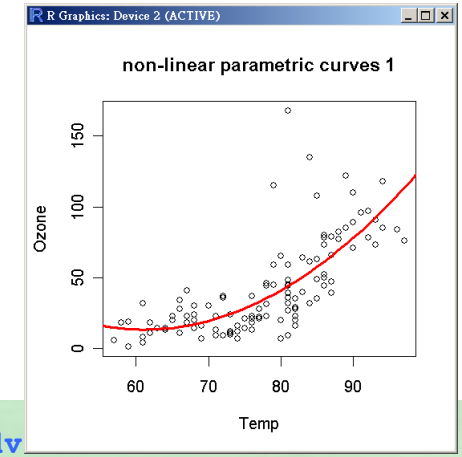
Example Methods

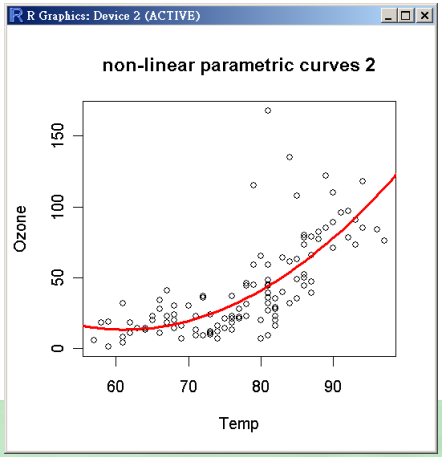
- non-linear parametric curves
- lowess (a non-parametric curve fitter)
- loess (a modelling tool)
- gam (fits generalized additive models)
- lm (linear model)



課堂練習1

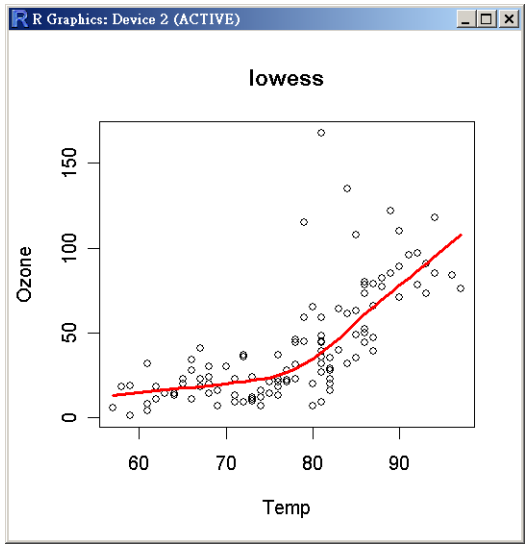
```
> data(airquality)
> id <- is.na(airquality$Ozone)
> Ozone <- airquality$Ozone[id==F]
> Temp <- airquality$Temp[id==F]

# Fitting non-linear parametric curves: non-linear least square nls
> plot(Ozone~Temp, main="non-linear parametric curves 1")
> model1 <- nls(Ozone~a+b*Temp+c*Temp*Temp, start=list(a=1, b=1, c=1))
> range(Temp)
> xv <- seq(55, 100, 1)
> lines(xv, predict(model1, list(Temp=xv)), col="red", lwd=2)

# or
> plot(Ozone~Temp, main="non-linear parametric curves 2")
> yv <- 305.48577-9.55060*xv+0.07807*xv*xv
> lines(xv, yv, col="red", lwd=2)
```

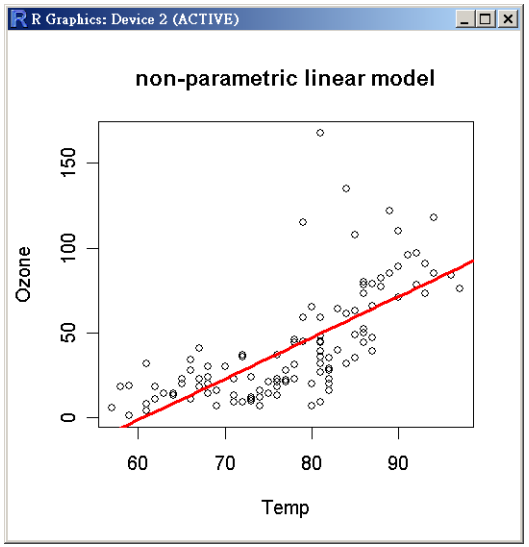




課堂練習2

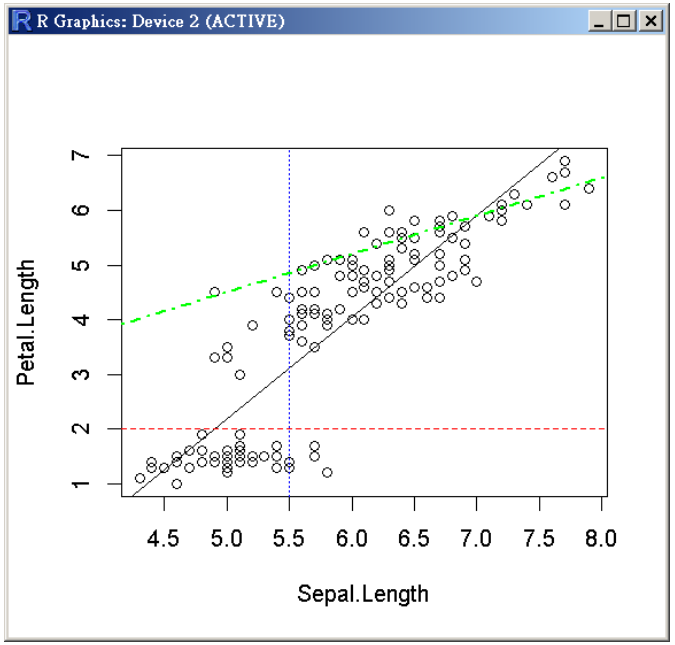
```
# lowess (a non-parametric curve fitter)
> plot(Ozone~Temp, main="lowess")
> lines(lowess(Ozone~Temp), col="red", lwd=2)

# lm
> plot(Ozone~Temp, main="non-parametric linear model")
> model2 <- lm(Ozone~Temp+(Temp^2)+(Temp^3))
> lines(xv, predict(model2, list(Temp=xv)), col="red", lwd=2)
```





於圖中加畫出直線: abline

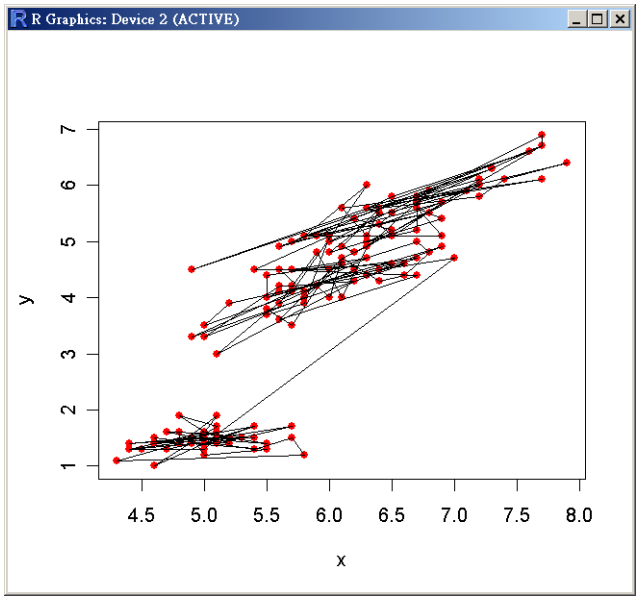
```
plot(Sepal.Length, Petal.Length)
abline(lm(Petal.Length~Sepal.Length), col="black")
abline(h=2, col="red", lty=2)
abline(v=5.5, col="blue", lty=3)
abline(a=1, b=0.7,, col="green", lty=4, lwd=2)
```

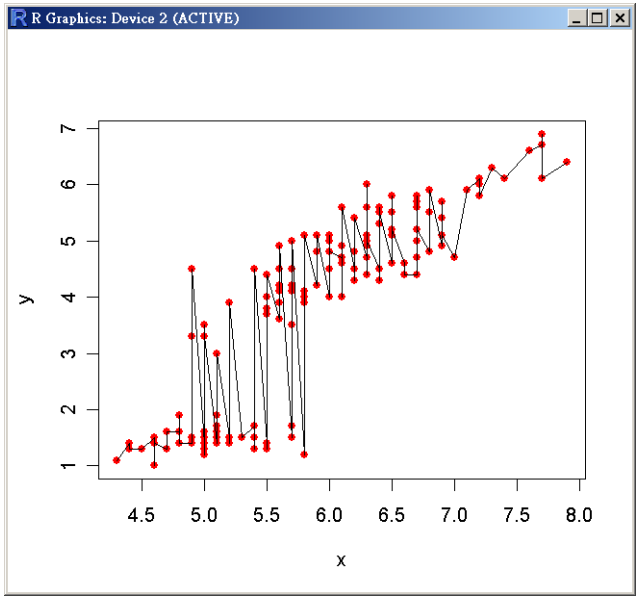


點的連結 (Join the Dots)

- Join the points on a scatterplot by lines: the trick is to ensure that the points on the x axis are ordered.
- if they are not ordered, the result is a mess.

```
> x <- iris[, 1]
> y <- iris[, 3]
> plot(x, y, pch=16, col="red")
> lines(x,y)
```

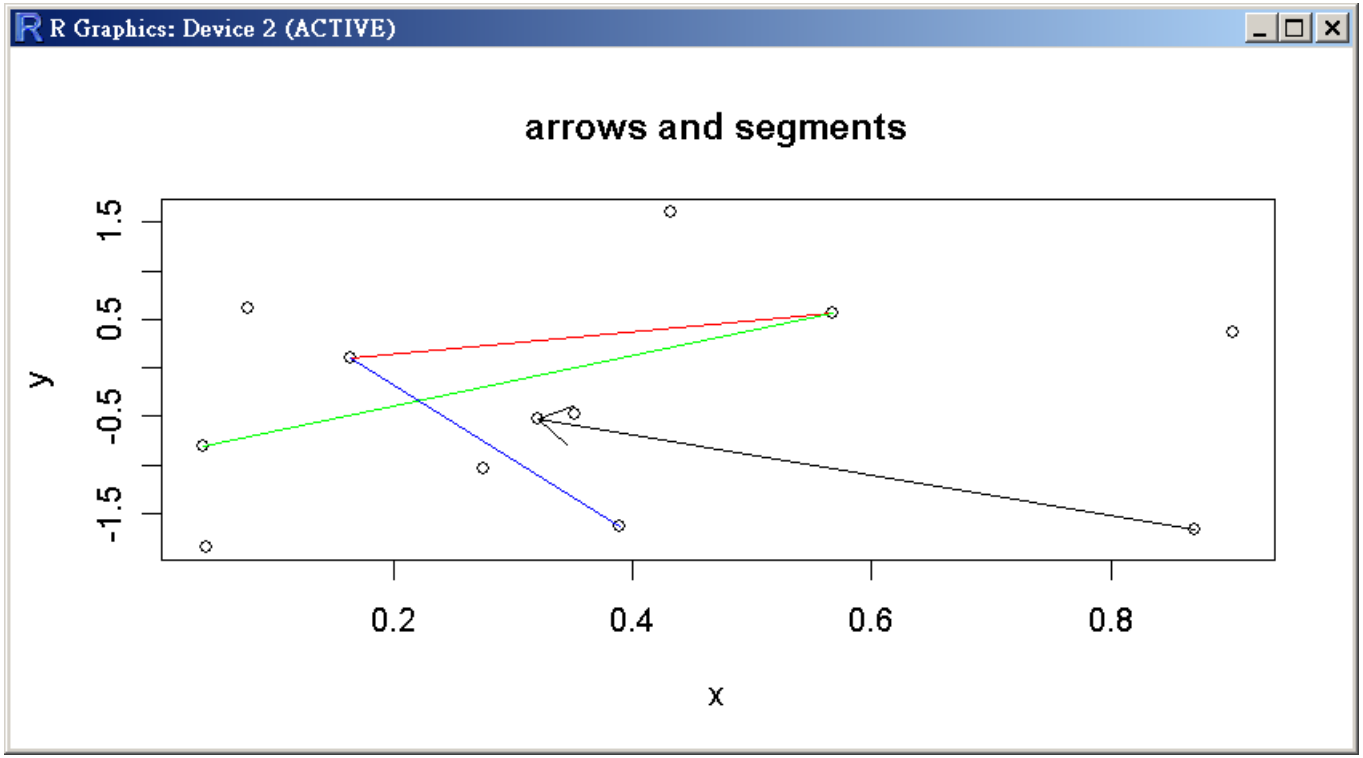





```
> sequence <- order(x)
> plot(x, y, pch=16, col="red")
> lines(x[sequence], y[sequence])
```

點的連結

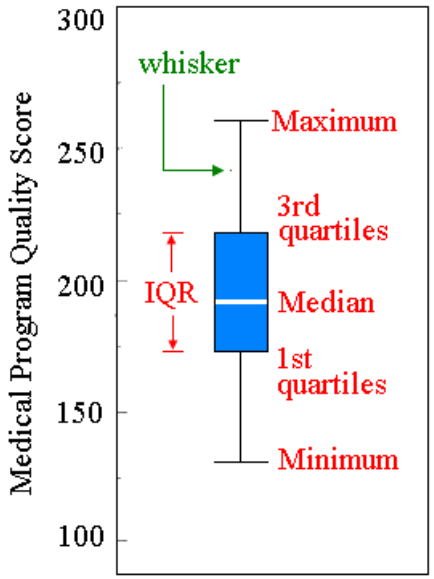
```
> x <- runif(12)
> y <- rnorm(12)
> plot(x, y, main="arrows and segments")
> arrows(x[1], y[1], x[2], y[2], col= "black", length=0.2)
> segments(x[3], y[3], x[4], y[4], col= "red")
> segments(x[3:4], y[3:4], x[5:6], y[5:6], col= c("blue", "green"))
```

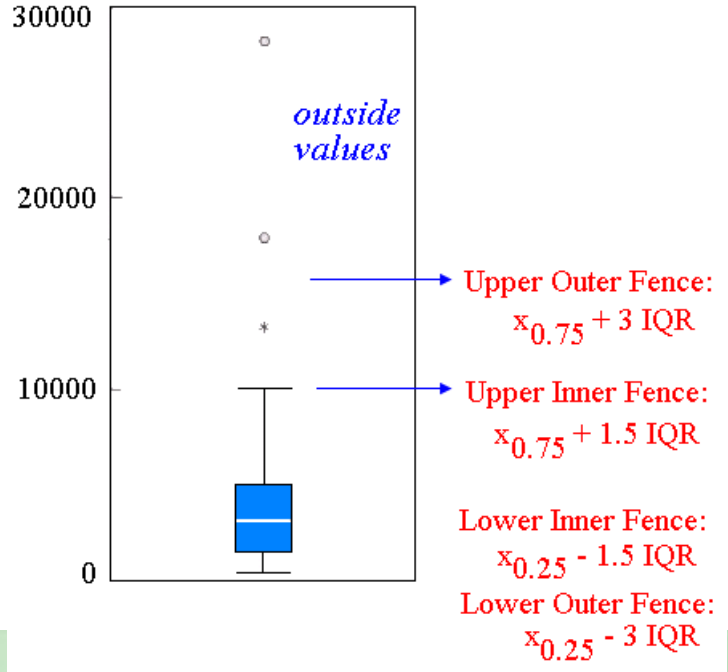


盒形圖 (Boxplot)

- Plotting with a categorical explanatory variable: boxplot.
- Categorical variables are factors with two or more levels.
- Boxplot
  - The horizontal line shows the median.
  - The bottom and top of the box show the 25th and 75th percentiles.
  - The vertical dashed lines are called the "whiskers".
  - Either maximum or 1.5 times the IQR.

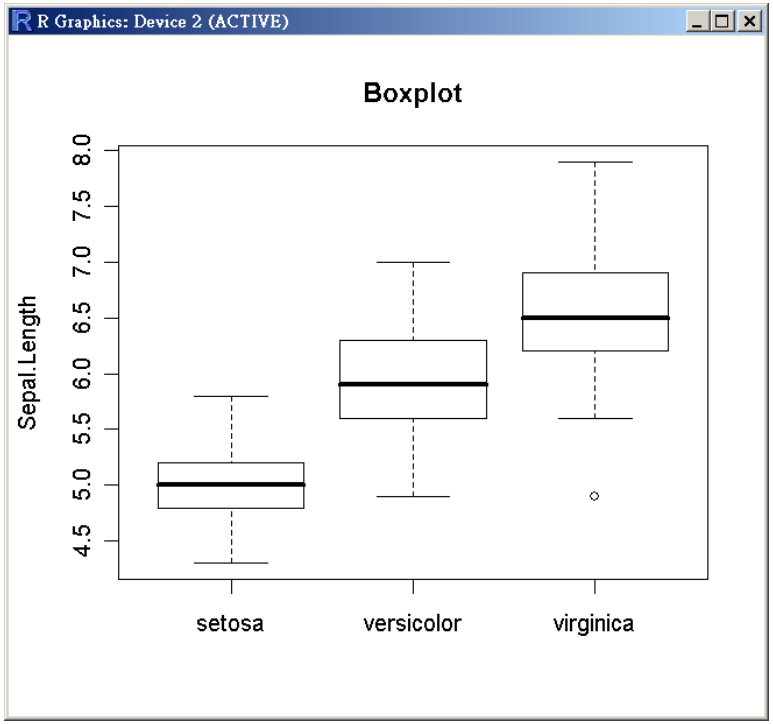
- boxplot not only show the location and spread of data but also indicate skewness.





使用plot畫盒形圖

```
> attach(iris)
> plot(factor(Species), Sepal.Length, ylab="Sepal.Length", main="Boxplot")
```



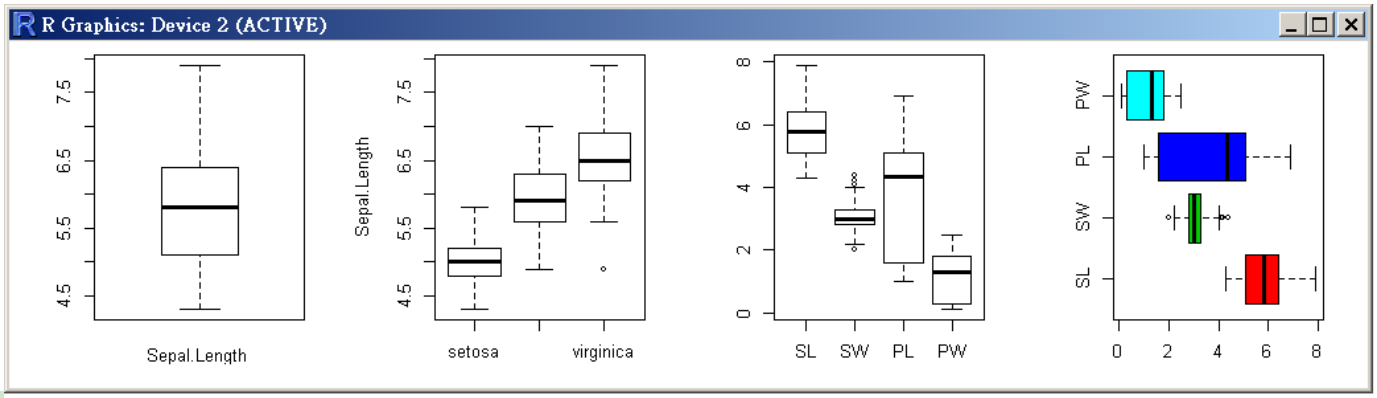
The box plot can provide answers to the following questions:

- Is a factor significant?
- Does the location differ between subgroups?
- Does the variation differ between subgroups?
- Are there any outliers?

boxplot(x, ...)

```
## S3 method for class 'formula'
boxplot(formula, data = NULL, ..., subset, na.action = NULL)

## Default S3 method:
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

```
> par(mfrow=c(1,4))
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> names(iris) <- c("SL", "SW", "PL", "PW", "Species")
> boxplot(Sepal.Length, xlab="Sepal.Length")
> boxplot(Sepal.Length~Species, ylab="Sepal.Length")
> boxplot(iris[,which(sapply(iris, is.numeric))])
> boxplot(iris[,which(sapply(iris, is.numeric))], horizontal=T, col=2:8)
```

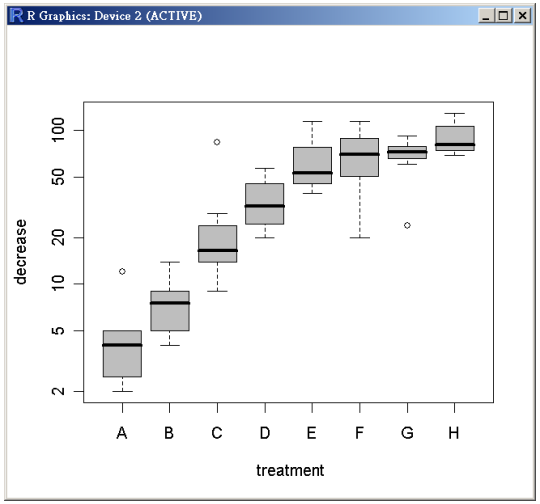


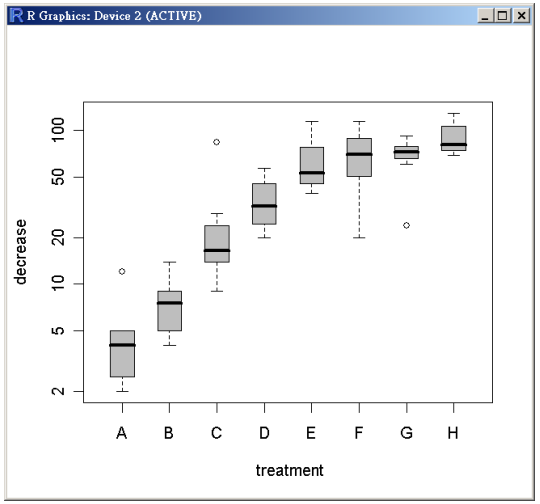
boxplot(formula)

```
> ylab <- "decrease"
> xlab <- " treatment"
> boxplot(decrease ~ treatment, data=OrchardSprays, log="y",
col="grey", xlab=xlab, ylab=ylab)

#control the width of the boxes
> boxplot(decrease ~ treatment, data=OrchardSprays, log="y",
col="grey", xlab=xlab, ylab=ylab, boxwex=0.5)
```

Data Editor

	decrease	rowpos	colpos	treatment
1	57	1	1	D
2	95	2	1	E
3	8	3	1	B
4	69	4	1	H
5	92	5	1	G
6	90	6	1	F
7	15	7	1	C
8	2	8	1	A
9	84	1	2	C
10	6	2	2	B
11	127	3	2	H
12	36	4	2	D
13	51	5	2	E
14	2	6	2	A
15	69	7	2	F
16	71	8	2	G
17	87	1	3	F
18	72	2	3	H
19	5	3	3	A
20	39	4	3	E
21	22	5	3	D
22	16	6	3	C
23	72	7	3	G
24	4	8	3	B

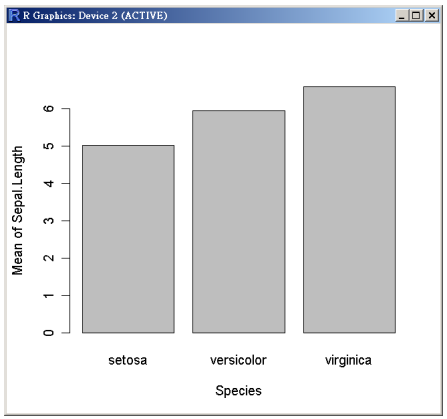


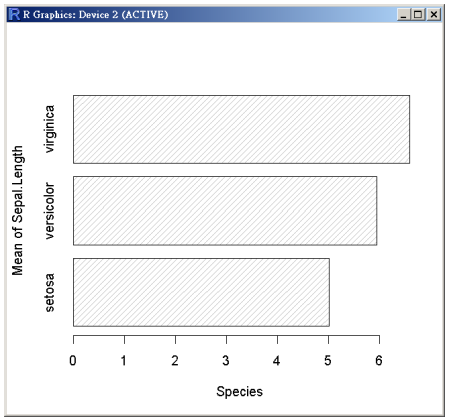


長條圖 (barplot)

```
barplot(height, width = 1, space = NULL,
        names.arg = NULL, legend.text = NULL, beside = FALSE,
        horiz = FALSE, density = NULL, angle = 45,
        col = NULL, border = par("fg"),
        main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
        axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
        add = FALSE, args.legend = NULL, ...)
```

```
> means <- tapply(iris$Sepal.Length, iris$Species, mean)
> barplot(means, xlab="Species", ylab="Mean of Sepal.Length")
# density:  a vector giving the density of shading lines
> barplot(means, xlab="Species", ylab="Mean of Sepal.Length", density=20,
    horiz=TRUE)
```

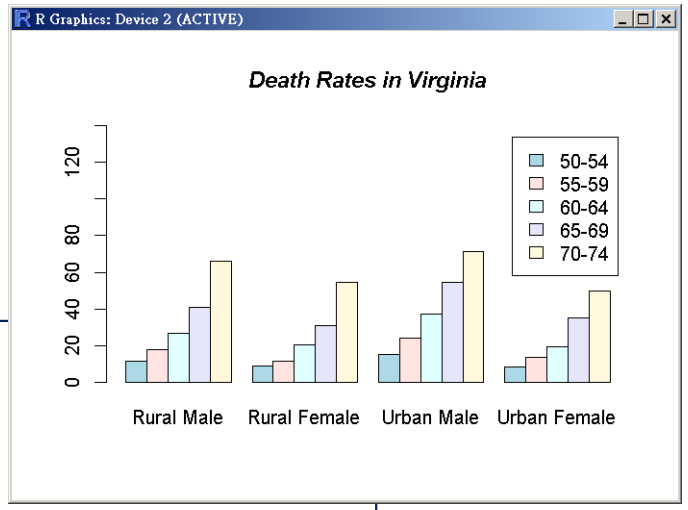




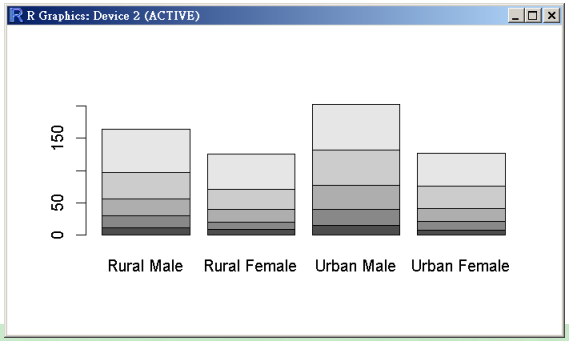
長條圖 (barplot)

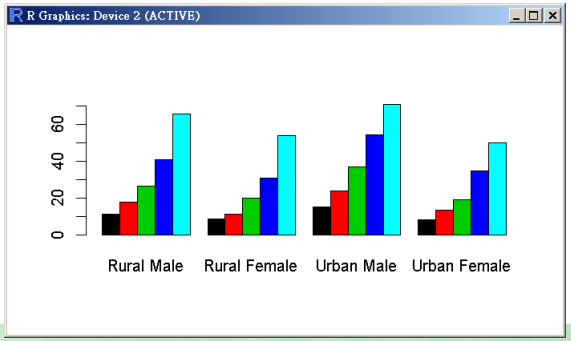
Death Rates in Virginia (1940): The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50–54, 55–59, 60–64, 65–69, 70–74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

```
> VADeaths
      Rural Male Rural Female Urban Male Urban Female
50-54       11.7          8.7       15.4          8.4
55-59       18.1         11.7       24.3         13.6
60-64       26.9         20.3       37.0         19.3
65-69       41.0         30.9       54.6         35.1
70-74       66.0         54.3       71.1         50.0
```



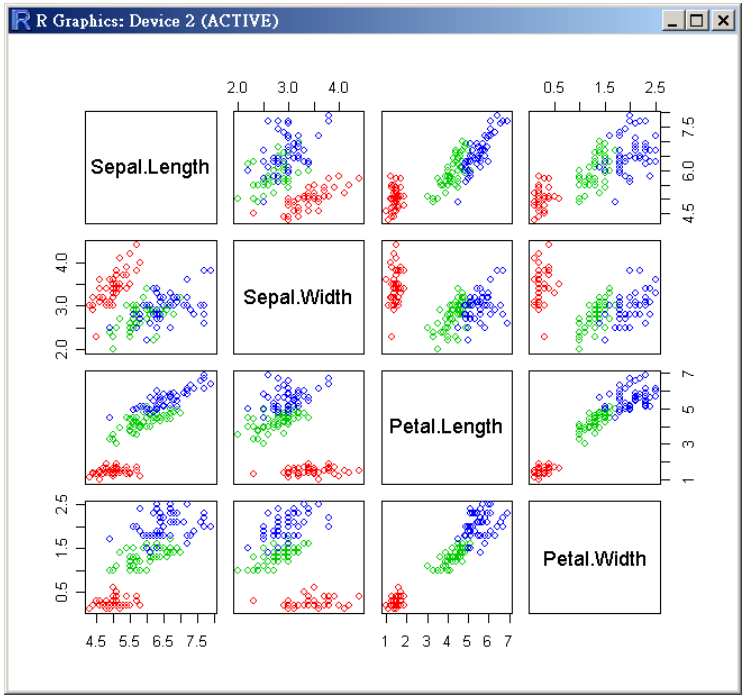

```
barplot(VADeaths)
barplot(VADeaths, beside = TRUE, col=1:5)

barplot(VADeaths, beside = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan",
                "lavender", "cornsilk"),
        legend = rownames(VADeaths), ylim = c(0, 140))
title(main = "Death Rates in Virginia", font.main = 4)
```

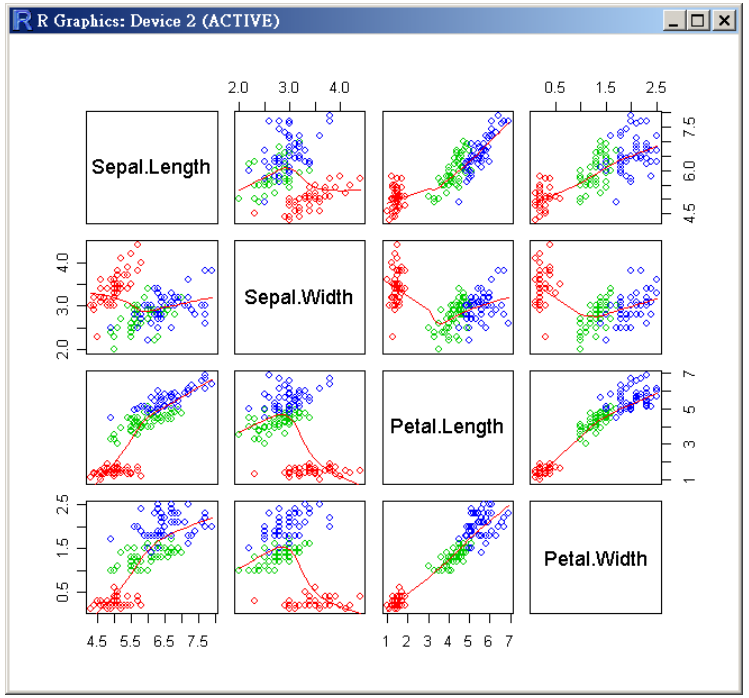




散佈圖矩陣 (Scatterplot Matrices)


```
> pairs(iris[,1:4], col=as.integer(iris[,5])+1)
> pairs(iris[,1:4], col=as.integer(iris[,5])+1,
  panel=panel.smooth)
```





See also: conditioning plot (**coplot**)

莖葉圖 (Stem Plot)

```
#a character-based stem-and leaf plot
> stem(iris[,1])

  The decimal point is 1 digit(s) to the left of the |

  42 | 0
  44 | 0000
  46 | 000000
  48 | 00000000000
  50 | 00000000000000000000
  52 | 00000
  54 | 0000000000000
  56 | 00000000000000
  58 | 0000000000
  60 | 000000000000
  62 | 0000000000000
  64 | 000000000000
  66 | 0000000000
  68 | 0000000
  70 | 00
  72 | 0000
  74 | 0
  76 | 00000
  78 | 0
```

plot函式總整理

- x為數值vector，`plot(x)`: 索引圖。
- x, y為vector，或xymat為兩個columns的矩陣，`plot(x, y)`或`plot(xymat)`: x vs y散佈圖。
- x為時間數列變數，`plot(x)`: 時間數列圖。
- f為因子變數，`plot(f)`: 長條圖。
- f為因子變數，y為數值變數，`plot(f, y)`: 比鄰盒形圖(side-by-side boxplot)。
- `plot(xdataframe)`或`plot(~x1+…+xk)`: x1,..., xk的散佈圖矩陣。
- `plot(y~x1+…+xk)`: y vs x1, y vs x2,..., y vs xk的散佈圖。

座標系統 (Coordinate System)

- The drawing of data symbols, lines, text, and so on in the plot region is relative to this user coordinate system.
- The scale on the axes: **`xlim`**, **`ylim`** or via **`par()`**.

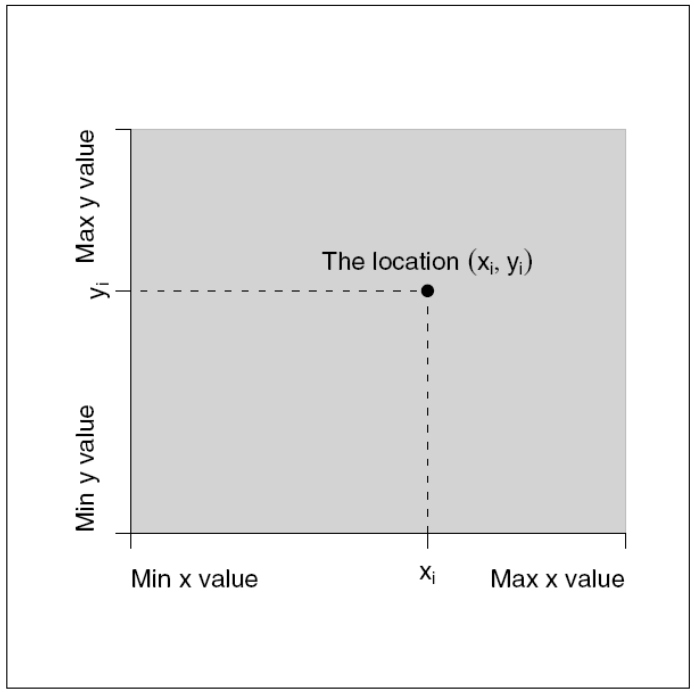


Figure 3.3
The user coordinate system in the plot region. Locations within this coordinate system are relative to the scales on the plot axes.

Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition

高階圖形調控項 (High-Level Graphics Setting)

Table 3.1
High-level traditional graphics state settings. This set of graphics state settings can be queried and set via the `par()` function *and* can be used as arguments to other graphics functions (e.g., `plot()` or `lines()`). Each setting is described in more detail in the relevant **Section**.

Setting	Description
`adj`	justification of text
`ann`	draw plot labels and titles?
`bg`	"background" color
`bty`	type of box drawn by `box()`
`cex`	size of text (multiplier)
`cex.axis`	size of axis tick labels
`cex.lab`	size of axis labels
`cex.main`	size of plot title
`cex.sub`	size of plot sub-title
`col`	color of lines and data symbols
`col.axis`	color of axis tick labels
`col.lab`	color of axis labels
`col.main`	color of plot title
`col.sub`	color of plot sub-title
`fg`	"foreground" color
`font`	font face (bold, italic) for text
`font.axis`	font face for axis tick labels
`font.lab`	font face for axis labels
`font.main`	font face for plot title
`font.sub`	font face for plot sub-title
`gamma`	gamma correction for colors
`lab`	number of ticks on axes
`las`	rotation of text in margins
`lty`	line type (solid, dashed)
`lwd`	line width
`mgp`	placement of axis ticks and tick labels
`pch`	data symbol type
`srt`	rotation of text in plot region
`tck`	length of axis ticks (relative to plot size)
`tcl`	length of axis ticks (relative to text size)
`tmag`	size of plot title (relative to other labels)
`type`	type of plot (points, lines, both)
`xaxp`	number of ticks on x-axis
`xaxs`	calculation of scale range on x-axis
`xaxt`	x-axis style (standard, none)
`xpd`	clipping region
`yaxp`	number of ticks on y-axis
`yaxs`	calculation of scale range on y-axis
`yaxt`	y-axis style (standard, none)

Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition

圖形設定: par()

> `par()` #list a complete list of the current graphics state.

- Setting will hold until a different setting is specified.

```
> par(c("col", "lty"))
> par(col="red", lty="dashed")
> y <- rnorm(20)
> plot(y, type="l") #line is dashed
> plot(y, type="l", lty="soild") #line is solid
> plot(y, type="l") #line is dashed
```

- `par()` will affect all subsequent graphical output.
  - High-level: `plot(..., col="red")`
    - The setting will affect the output just for that plot.
  - Low-level: `lines(..., col="red")`
    - Control the appearance of a single piece of graphical output.

低階圖形調控項 (Low-Level Graphics Setting)

```
>?par
```

Setting	Description
ask	prompt user before new page?
family	font family for text
fig	location of figure region (normalized)
fin	size of figure region (inches)
lend	line end style
lheight	line spacing (multiplier)
ljoin	line join style
lmitre	line mitre limit
mai	size of figure margins (inches)
mar	size of figure margins (lines of text)
mex	line spacing in margins
mfcol	number of figures on a page
mfg	which figure is used next
mfrow	number of figures on a page
new	has a new plot been started?
oma	size of outer margins (lines of text)
omd	location of inner region (normalized)
omi	size of outer margins (inches)
pin	size of plot region (inches)
plt	location of plot region (normalized)
ps	size of text (points)
pty	aspect ratio of plot region
usr	range of scales on axes
xlog	logarithmic scale on x-axis?
ylog	logarithmic scale on y-axis?

Table 3.2
Low-level traditional graphics state settings. This set of graphics state settings can be queried and set via the `par()` function. Each setting is described in more detail in the relevant **Section**.

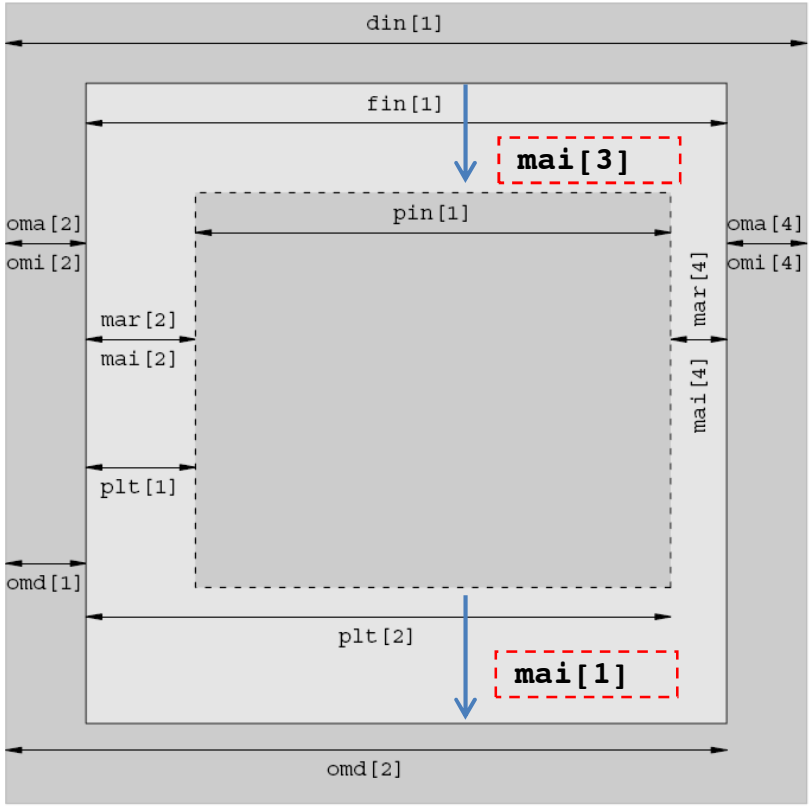
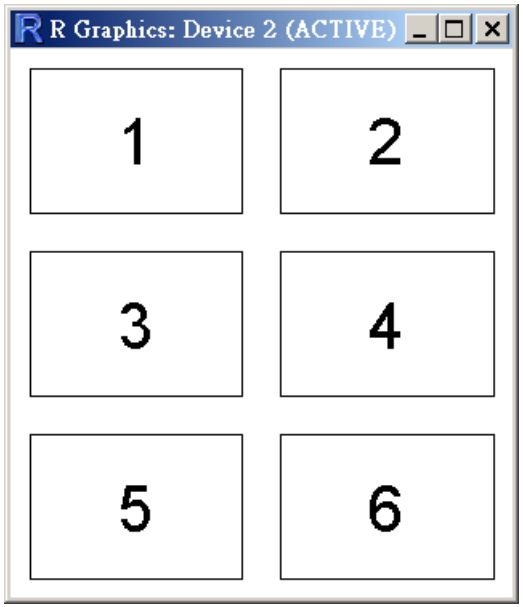


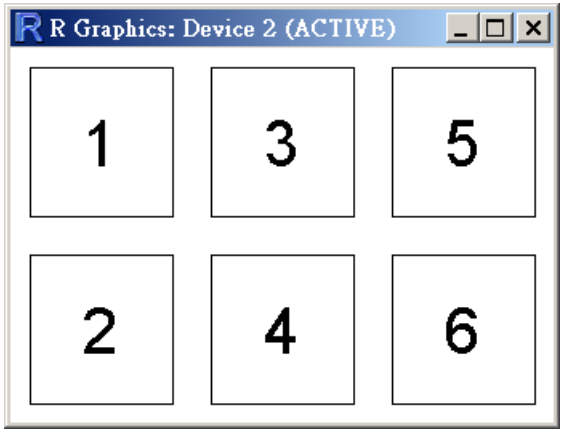
Figure 3.13
Graphics state settings controlling plot regions. These are some of the settings that control the widths and horizontal locations of the plot regions. For ease of comparison, this diagram has the same layout as Figure 3.1: the central grey rectangle represents the plot region, the lighter grey rectangle around that is the figure region, and the darker grey rectangle around that is the outer margins. A similar diagram could be produced for settings controlling heights and vertical locations.

Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition

一頁多張圖形: mfrow, mfcol

```
> myplot <- function(n){
+     for(i in 1:n){
+         plot(1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
+         text(1, 1, labels=paste(i), cex=3)
+     }
+ }
>
> orig.par <- par(mai=c(0.1, 0.1, 0.1, 0.1), mfrow=c(3, 2))
> myplot(6)
> par(orig.par)
> par(mai=c(0.1, 0.1, 0.1, 0.1), mfcol=c(2, 3))
```

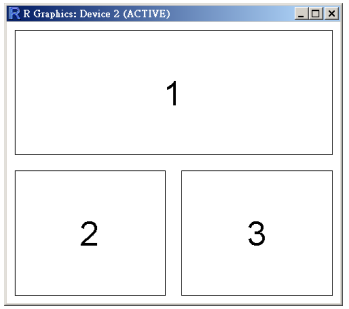




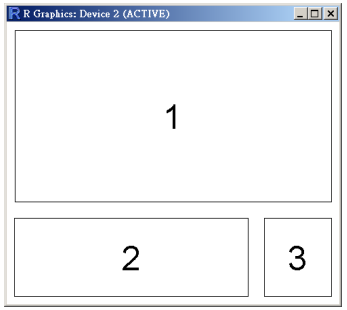
一頁多張圖形: layout

```
layout(mat, widths = rep.int(1, ncol(mat)),
       heights = rep.int(1, nrow(mat)), respect = FALSE)
```

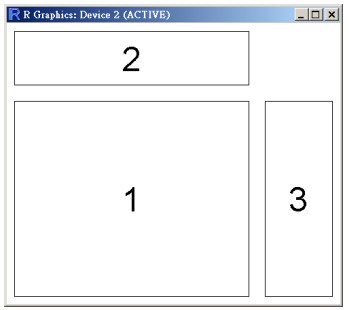
```
> orig.par <- par(mai=c(0.1, 0.1, 0.1, 0.1))
> (mat1 <- matrix(c(1,2,1,3), 2, 2))
     [,1] [,2]
[1,]    1    1
[2,]    2    3
> layout(mat1)
> myplot(3)
> par(orig.par)
```



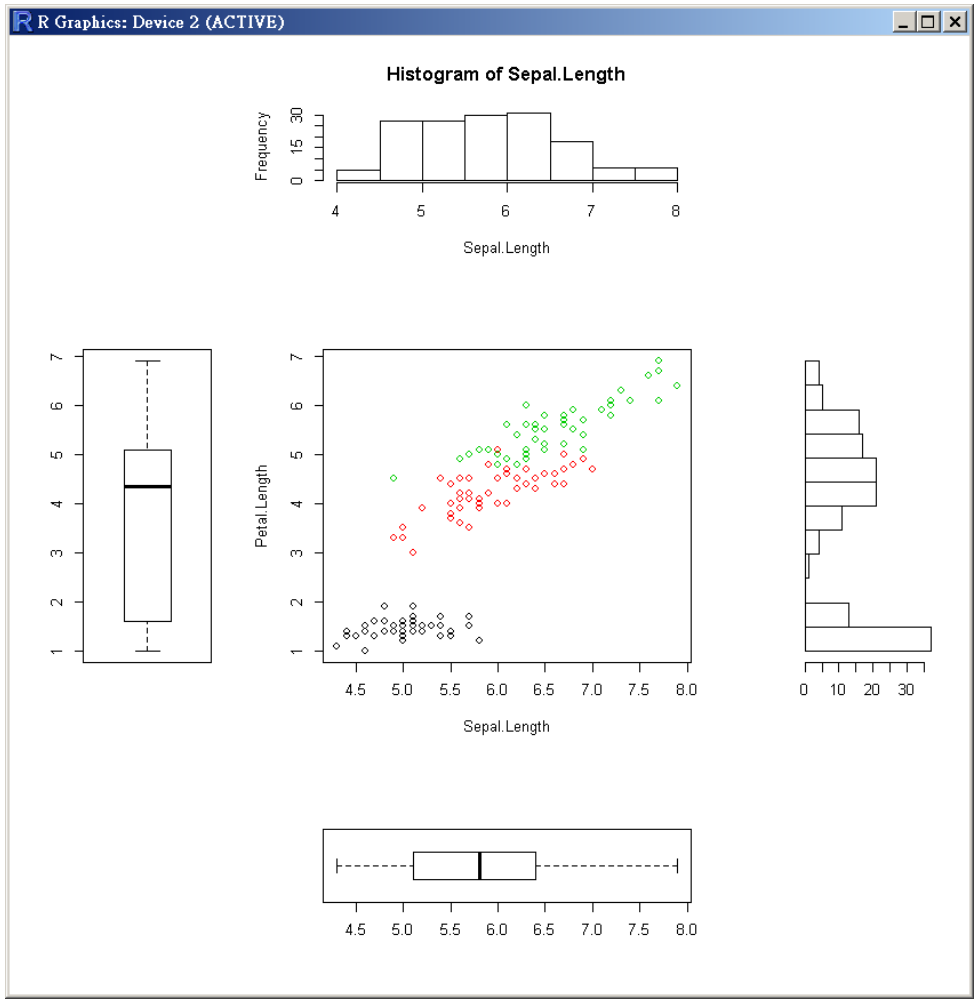

```
> layout(mat1, widths=c(3, 1), heights=c(2, 1))
> myplot(3)
```




```
> (mat2 <- matrix(c(2,1,0,3), 2, 2))
     [,1] [,2]
[1,]    2    0
[2,]    1    3
> layout(mat2, widths=c(3, 1), heights=c(1, 3))
> myplot(3)
```



課堂練習3



See also: **`par(fig)`**

作圖建議

Produce (traditional) graphics

Customize graphics using high-level graphics setting

Customize graphics
using low-level graphics setting

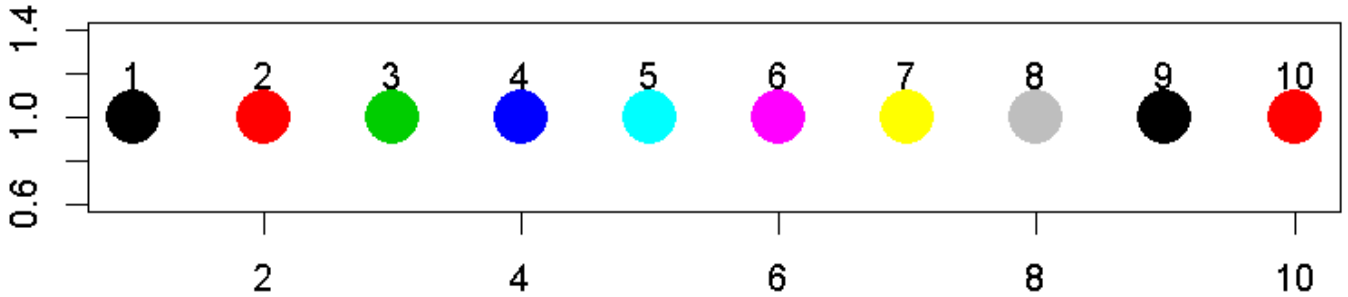
Output

顏色 (Color)

- Specify the color of data symbols, lines, text that are drawn in the plot region.

- `col, fg, bg`
  - **`col.axis`** (axes), **`col.lab`** (label), **`col.main`** (titles), **`col.sub`** (sub-titles)
  - **`fg`**: color of axes and borders on plots.
  - **`bg`**: the color of the background for base graphics output. This color is used to fill the entire page.

```
> plot(1:10, rep(1, 10), pch=20, col=1:10, cex=5, xlab="", ylab="")
> text(1:10, rep(1.2, 10), labels=1:10)
```



colors() # full list of known names.

```
> colors()
  [1] "white"                "aliceblue"            "antiquewhite"
  [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
  [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
 [10] "aquamarine2"          "aquamarine3"          "aquamarine4"
 [13] "azure"                "azure1"               "azure2"
 [16] "azure3"               "azure4"               "beige"
 [19] "bisque"               "bisque1"              "bisque2"
 [22] "bisque3"              "bisque4"              "black"
 [25] "blanchedalmond"       "blue"                 "blue1"
 [28] "blue2"                "blue3"                "blue4"
 [31] "blueviolet"           "brown"                "brown1"
 [34] "brown2"               "brown3"               "brown4"
.....

[637] "turquoise2"           "turquoise3"           "turquoise4"
[640] "violet"               "violetred"            "violetred1"
[643] "violetred2"           "violetred3"           "violetred4"
[646] "wheat"                "wheat1"               "wheat2"
[649] "wheat3"               "wheat4"               "whitesmoke"
[652] "yellow"               "yellow1"              "yellow2"
[655] "yellow3"              "yellow4"              "yellowgreen"
```

```
> rgb(1,0,0) # red
   [1] "#FF0000"
```

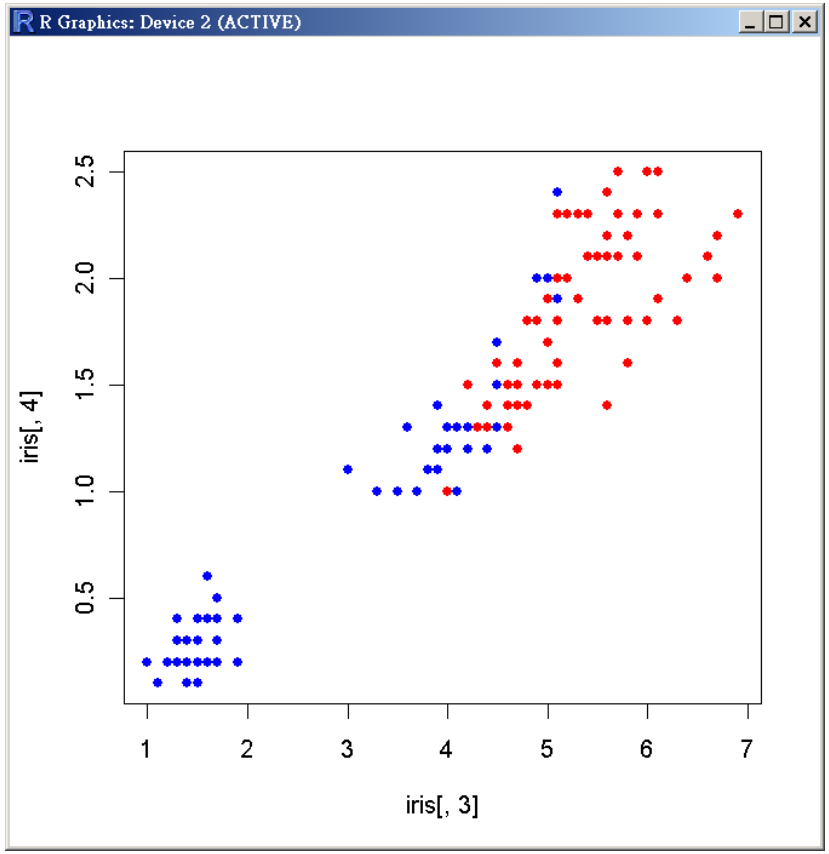
col2rgb()

```
> col2rgb("peachpuff")
      [,1]
red    255
green  218
blue   185
> col2rgb(c(myblue = "royalblue", reddish = "tomato")) # names kept
      myblue reddish
red       65     255
green    105      99
blue     225      71
> col2rgb(paste("gold", 1:4, sep=""))
      [,1] [,2] [,3] [,4]
red    255  238  205  139
green  215  201  173  117
blue     0    0    0    0
> col2rgb("#08a0ff")
      [,1]
red      8
green  160
blue   255
>
> palette() #predefined set of colors
[1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta" "yellow"
[8] "gray"
> col2rgb(1:8) # the ones from the palette()
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
red      0  255    0    0    0  255  255  190
green    0    0  205    0  255    0  255  190
blue     0    0    0  255  255  255    0  190
```

Color Data Symbols

- The points with iris[,1] above median in red, others in blue.

```
> plot(iris[,3], iris[,4], pch=16,
      col=ifelse(iris[,1] > median(iris[,1]), "red", "blue"))
```



顏色集 (Color Sets)

Table 3.4
Functions to generate color sets. R functions that can be used to generate coherent sets of colors

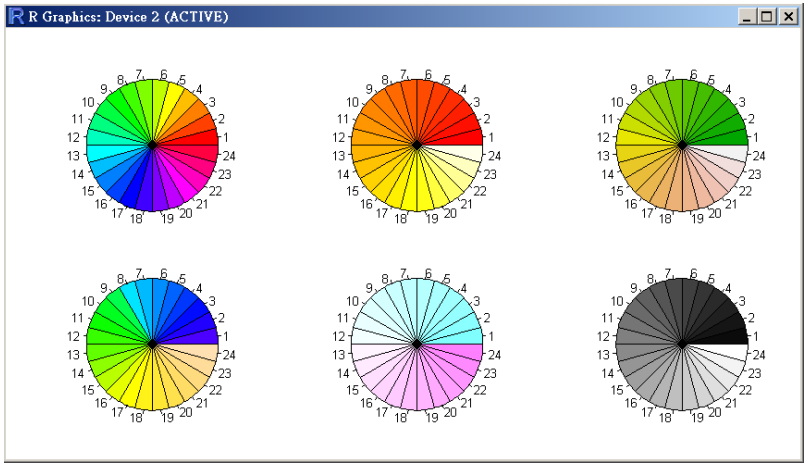
Name	Description
`rainbow()`	Colors vary from red through orange, yellow, green, blue, and indigo, to violet.
`heat.colors()`	Colors vary from white, through orange, to red.
`terrain.colors()`	Colors vary from white, through brown, to green.
`topo.colors()`	Colors vary from white, through brown then green, to blue.
`cm.colors()`	Colors vary from light blue, through white, to light magenta.
`grey()` or `gray()`	A set of shades of grey.

Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition

Fill pattern

- `rect(), polygon(), hist(), barplot(), pie(), legend()`
- `rainbow(n)` # n: the number of colors

```
> par(mfrow=c(2,3))
> n <- 24
> pie(rep(1,n), col=rainbow(n))
> pie(rep(1,n), col=heat.colors(n))
> pie(rep(1,n), col=terrain.colors(n))
> pie(rep(1,n), col=topo.colors(n))
> pie(rep(1,n), col=cm.colors(n))
> pie(rep(1,n), col=grey(1:n/n))
```



線條 (Lines)

Five graphics state settings for lines:

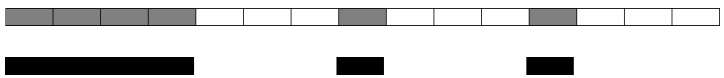
- `lty ="solid"` #type of line to draw (solid, dashed, dotted,...)
- `lwd=3` #width of line 3 pixel.
- `lend` #line ends, can be round, square
- `ljoin` #line joins, can be mitre, round
- `lmitre`
- These setting can only be specified via par()

Customized line type via a string of digits:

- Each digit is a hexadecimal value that indicates a number of units to draw either a line or a gap.
- odd digit: line lengths
- even digit: gap lengths
- Example: dotted line by `lty="13"`: length one unit then a gap of length three units.

Integer	Sample line	String
Predefined		
0		"blank"
1		"solid"
2		"dashed"
3		"dotted"
4		"dotdash"
5		"longdash"
6		"twodash"
Custom		
		"13"
		"F8"
		"431313"
		"22848222"

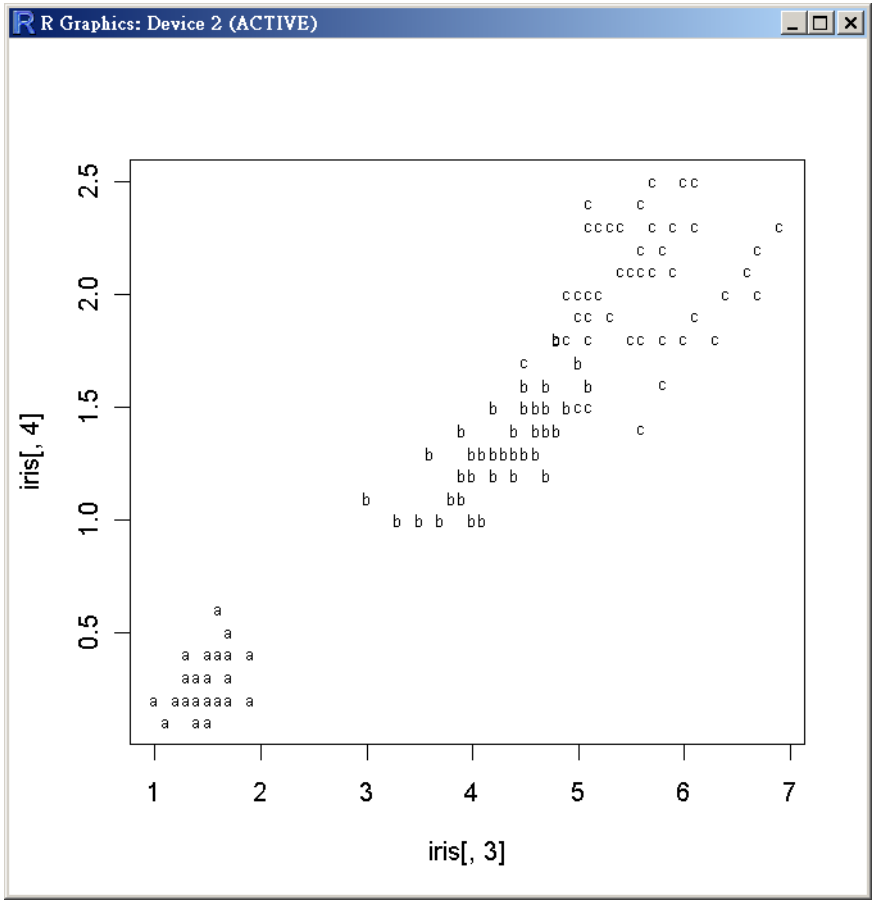
Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition



`lty="431313"`

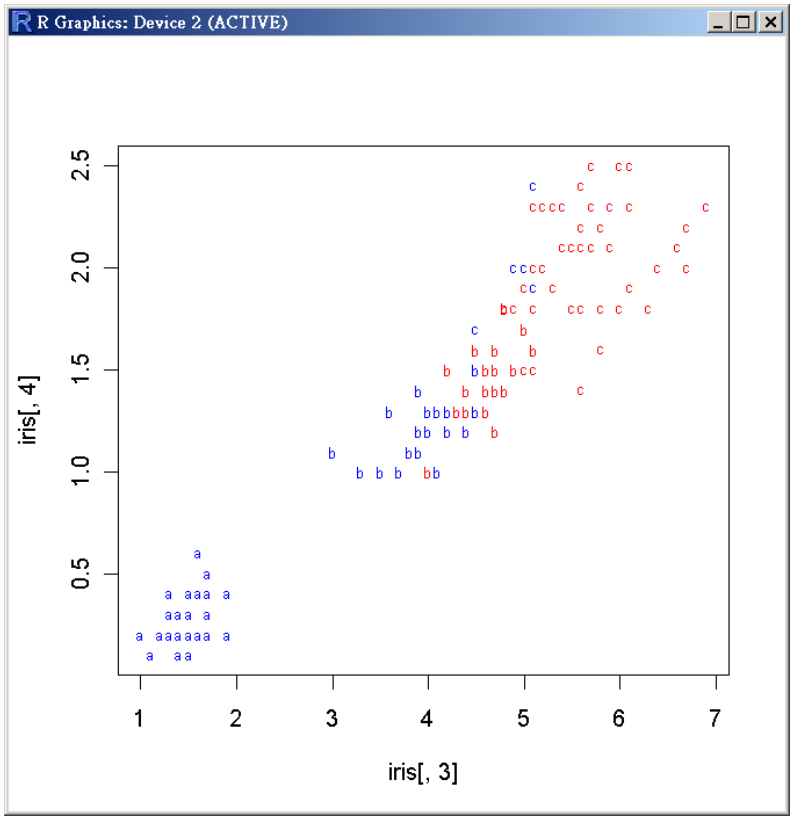
文字標號 (Text Labels)

```
> plot(iris[,3], iris[,4], type="n")
> my.label <- c(rep("a", 50), rep("b", 50), rep("c", 50))
> text(iris[,3], iris[,4], labels=my.label, cex=0.7)
```



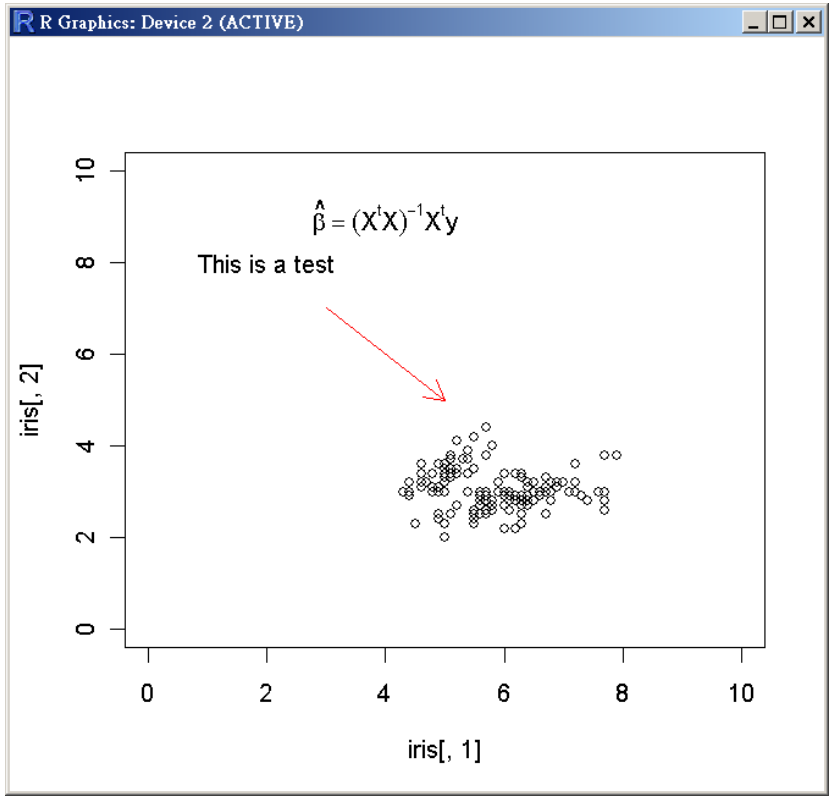
文字標號 (Text Labels)

```
> plot(iris[,3], iris[,4], type="n")
> my.label <- c(rep("a", 50), rep("b", 50), rep("c", 50))
> text(iris[,3], iris[,4], my.label, cex=0.7,
       col=ifelse(iris[,1] > median(iris[,1]), "red", "blue"))
```



圖上文字

```
> plot(iris[,1], iris[,2], xlim=c(0, 10), ylim=c(0, 10))
> text(2,8, "This is a test")
> arrows(x0=3, y0=7, x1=5, y1=5, length = 0.15, col="red")
> text(4, 9, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
```



Data Symbols

A fixed set of 26 data symbols:

- **pch=numbers**
- **pch="a"**, a single character
- **pch="."**, small dot
- **cex:** size of data symbol
- **type:** how data is represented in a plot
  - **"p"**: data symbols are drawn at each (x,y) location
  - **"l"**: (x,y) locations are connected by lines
  - **"b"**: both data symbols and lines are drawn.
  - **"o"**: over-plot onlines
  - "h": vertical lines are drawn from x-axis to the (x,y)
  - **"s"**, "S": city-block fashion, step
  - **"n"**: nothing is drawn

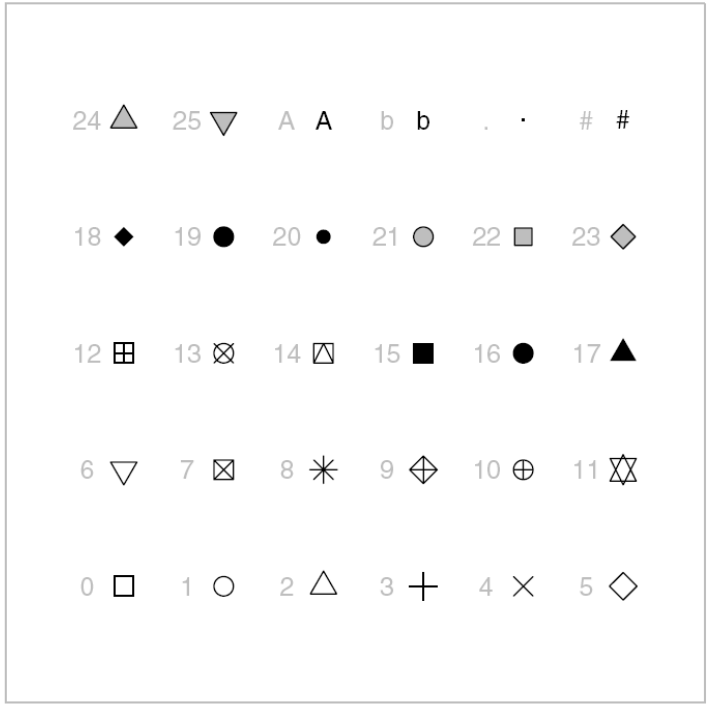
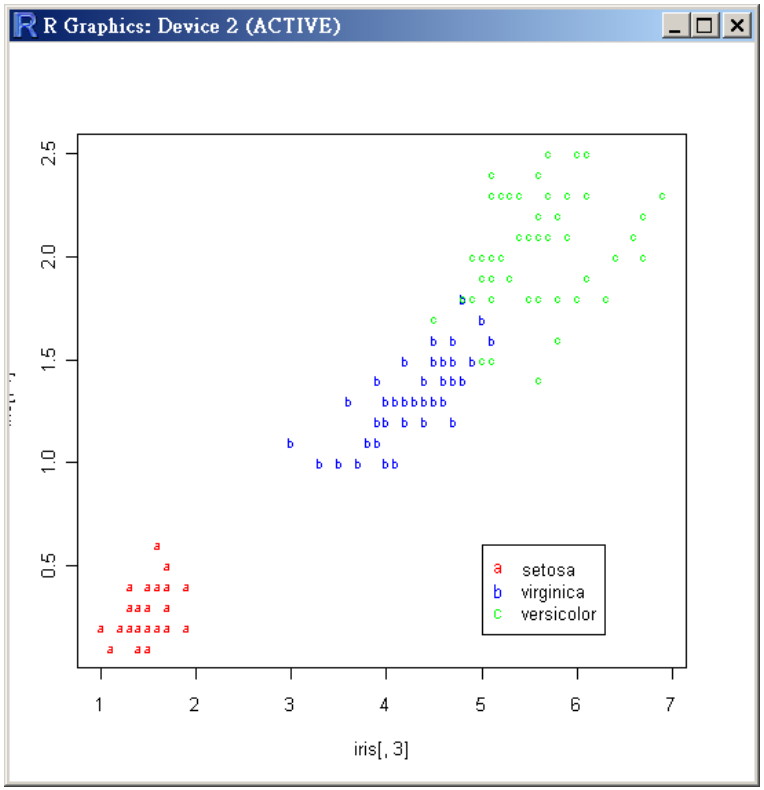


Figure 3.10
Data symbols available in R. A particular data symbol is selected by specifying an integer between 0 and 25 or a single character for the pch graphical setting. In the diagram, the relevant integer or character pch value is shown in grey to the left of the relevant symbol.

Murrell, P., 2005, R graphics, Chapman & Hall/CRC; 1 edition

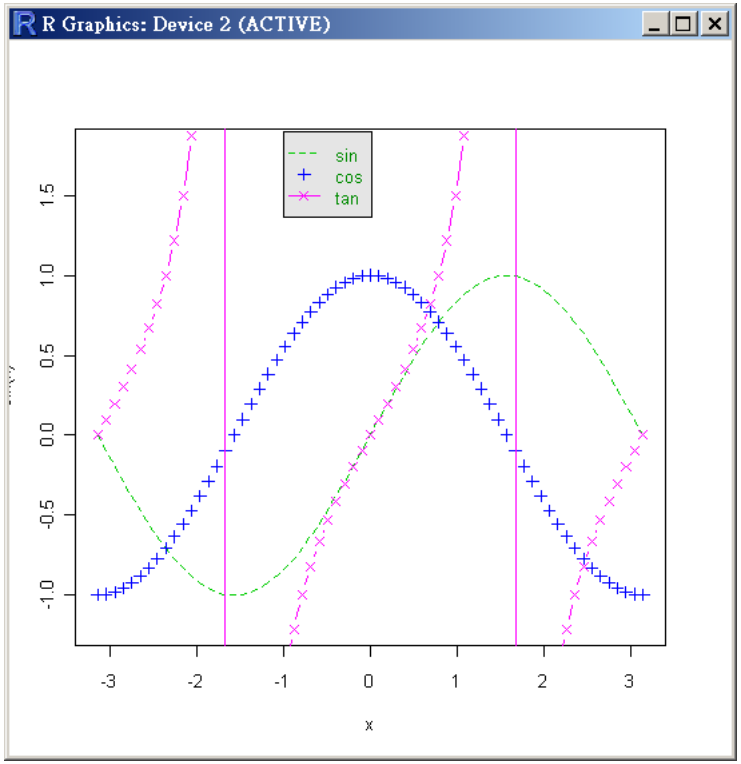
Legend

```
> plot(iris[,3], iris[,4], type="n")
> my.label <- c(rep("a", 50), rep("b", 50), rep("c", 50))
> my.color <- c(rep("red", 50), rep("blue", 50), rep("green", 50))
> text(iris[,3], iris[,4], my.label, cex=0.7, col=my.color)
> legend(5, 0.6, legend=c("setosa","versicolor", "virginica"), pch = "abc",
       col=c("red","blue","green"))
```



Legend

```
> x <- seq(-pi, pi, len = 65)
> plot(x, sin(x), type = "l", ylim = c(-1.2, 1.8), col = 3, lty = 2)
> points(x, cos(x), pch = 3, col = 4)
> lines(x, tan(x), type = "b", lty = 1, pch = 4, col = 6)
> legend(-1, 1.9, c("sin", "cos", "tan"), col = c(3,4,6), text.col = "green4",
         lty = c(2, -1, 1), pch = c(-1, 3, 4), bg = 'gray90')
```

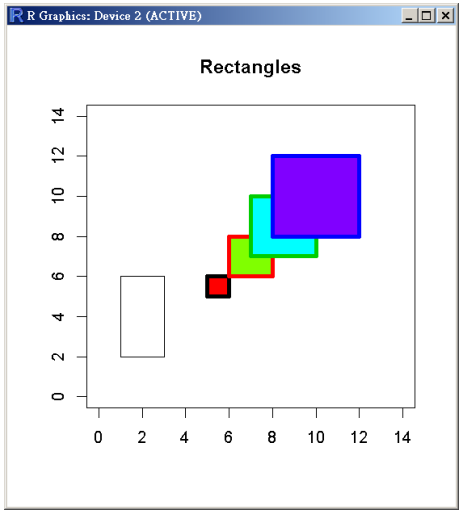


More Examples:
> ?legend

矩形，Draw Symbols

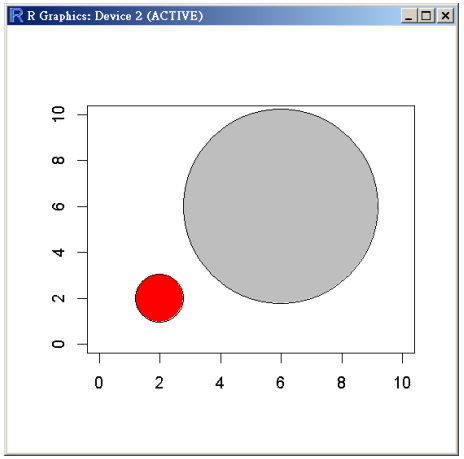
(Circles, Squares, Stars, Thermometers, Boxplots)

```
> plot(0, xlim=c(0,14), ylim=c(0, 14), type = "n",
xlab = "", ylab = "", main = "Rectangles")
> rect(1, 2, 3, 6)
> n <- 0:3
> rect(5+n, 5+n, 6+2*n, 6+2*n, col = rainbow(4),
border = n+1, lwd=4)
```




```
symbols(x, y = NULL, circles, squares, rectangles, stars,
        thermometers, boxplots, inches = TRUE, add = FALSE,
        fg = par("col"), bg = NA,
        xlab = NULL, ylab = NULL, main = NULL,
        xlim = NULL, ylim = NULL, ...)
```

```
> symbols(x = c(2, 6), y = c(2, 6), circles = c(1,
4), xlim=c(0, 10), ylim=c(0, 10), bg=c("red",
"gray"), xlab="", ylab="")
```

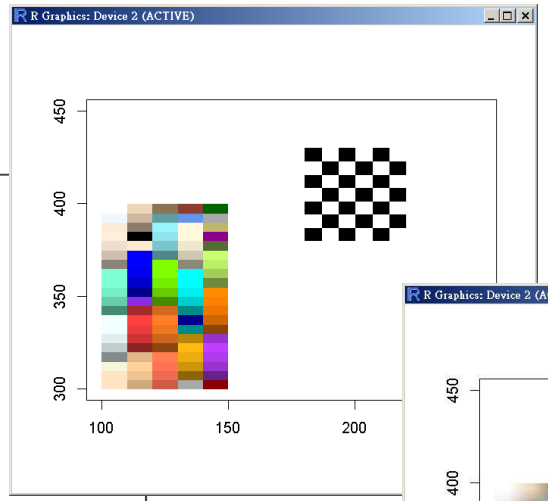


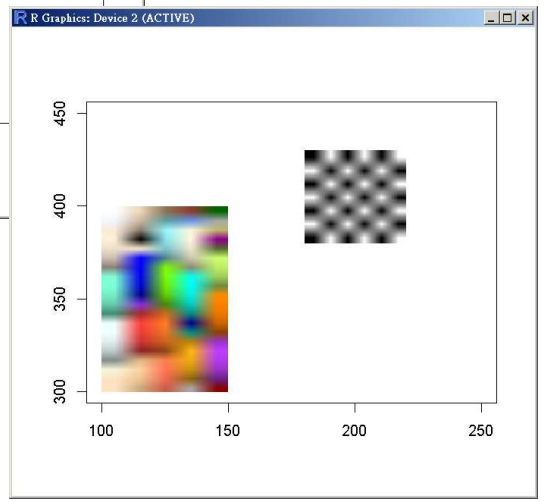
See also "ellipse" package

Draw One or More Raster Images

```
> plot(c(100, 250), c(300, 450), type = "n", xlab = "", ylab = "")
> i1 <- as.raster(matrix(0:1, ncol = 6, nrow = 7))
> rasterImage(i1, 180, 380, 220, 430, interpolate = FALSE)

> i2 <- as.raster(matrix(colors()[1:100], ncol = 5))
> rasterImage(i2, 100, 300, 150, 400, interpolate = FALSE)
```

```
> matrix(0:1, ncol = 6, nrow = 7)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    1    0    1    0    1
[2,]    1    0    1    0    1    0
[3,]    0    1    0    1    0    1
[4,]    1    0    1    0    1    0
[5,]    0    1    0    1    0    1
[6,]    1    0    1    0    1    0
[7,]    0    1    0    1    0    1
> as.raster(matrix(0:1, ncol = 6, nrow = 7))
     [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF"
[2,] "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000"
[3,] "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF"
[4,] "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000"
[5,] "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF"
[6,] "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000"
[7,] "#000000" "#FFFFFF" "#000000" "#FFFFFF" "#000000" "#FFFFFF"
```

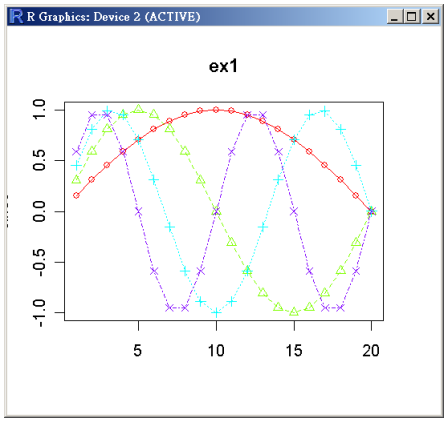


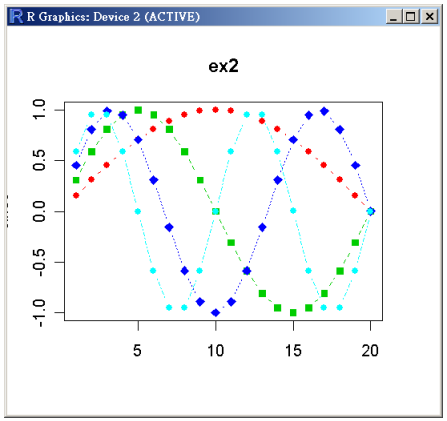


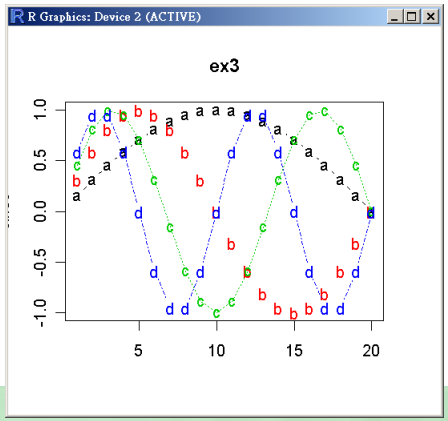
Plot Columns of Matrices

```
sines <- outer(1:20, 1:4, function(x, y) sin(x / 20 * pi * y))
dim(sines)
sines
matplot(sines, pch = 1:4, type = "o", col = rainbow(ncol(sines)), main="ex1")
matplot(sines, pch = 21:23, type = "b", col = 2:5, bg= 2:5, main="ex2")
```

```
> sines <- outer(1:20, 1:4, function(x, y) sin(x / 20 * pi * y))
> sines
              [,1]          [,2]          [,3]          [,4]
 [1,] 1.564345e-01  3.090170e-01  4.539905e-01  5.877853e-01
 [2,] 3.090170e-01  5.877853e-01  8.090170e-01  9.510565e-01
 [3,] 4.539905e-01  8.090170e-01  9.876883e-01  9.510565e-01
 [4,] 5.877853e-01  9.510565e-01  9.510565e-01  5.877853e-01
 [5,] 7.071068e-01  1.000000e+00  7.071068e-01  1.224606e-16
 [6,] 8.090170e-01  9.510565e-01  3.090170e-01 -5.877853e-01
 [7,] 8.910065e-01  8.090170e-01 -1.564345e-01 -9.510565e-01
 [8,] 9.510565e-01  5.877853e-01 -5.877853e-01 -9.510565e-01
 [9,] 9.876883e-01  3.090170e-01 -8.910065e-01 -5.877853e-01
[10,] 1.000000e+00  1.224606e-16 -1.000000e+00 -2.449213e-16
[11,] 9.876883e-01 -3.090170e-01 -8.910065e-01  5.877853e-01
[12,] 9.510565e-01 -5.877853e-01 -5.877853e-01  9.510565e-01
[13,] 8.910065e-01 -8.090170e-01 -1.564345e-01  9.510565e-01
[14,] 8.090170e-01 -9.510565e-01  3.090170e-01  5.877853e-01
[15,] 7.071068e-01 -1.000000e+00  7.071068e-01  3.673819e-16
[16,] 5.877853e-01 -9.510565e-01  9.510565e-01 -5.877853e-01
[17,] 4.539905e-01 -8.090170e-01  9.876883e-01 -9.510565e-01
[18,] 3.090170e-01 -5.877853e-01  8.090170e-01 -9.510565e-01
[19,] 1.564345e-01 -3.090170e-01  4.539905e-01 -5.877853e-01
[20,] 1.224606e-16 -2.449213e-16  3.673819e-16 -4.898425e-16
>
```



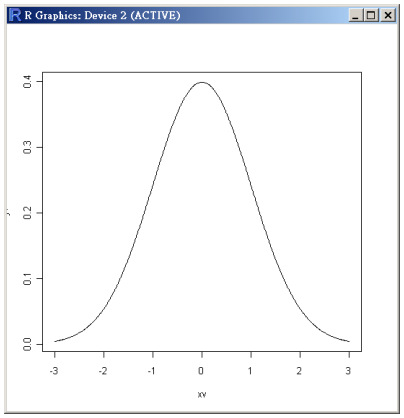


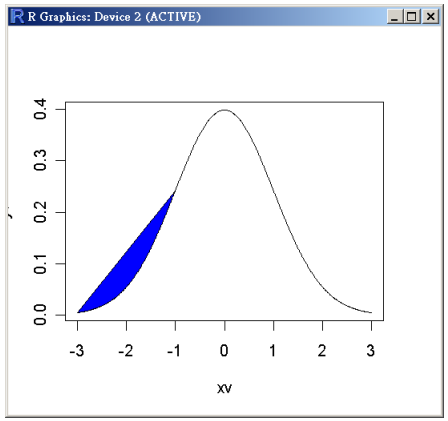


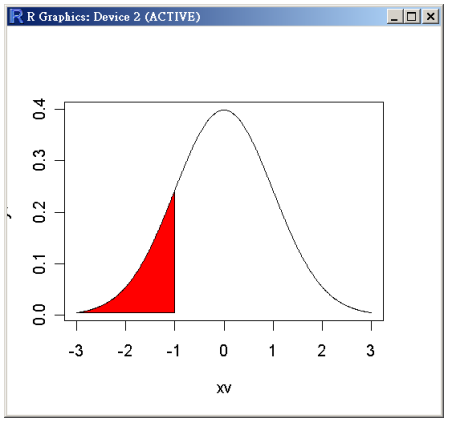
Exercise: adding legends?

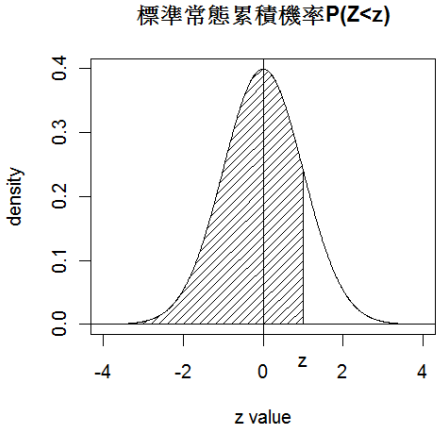
多邊形 (Polygon)

```
# draw a polygon using mouse
> locations <- locator(6)
> polygon(location, col="lavender")
```







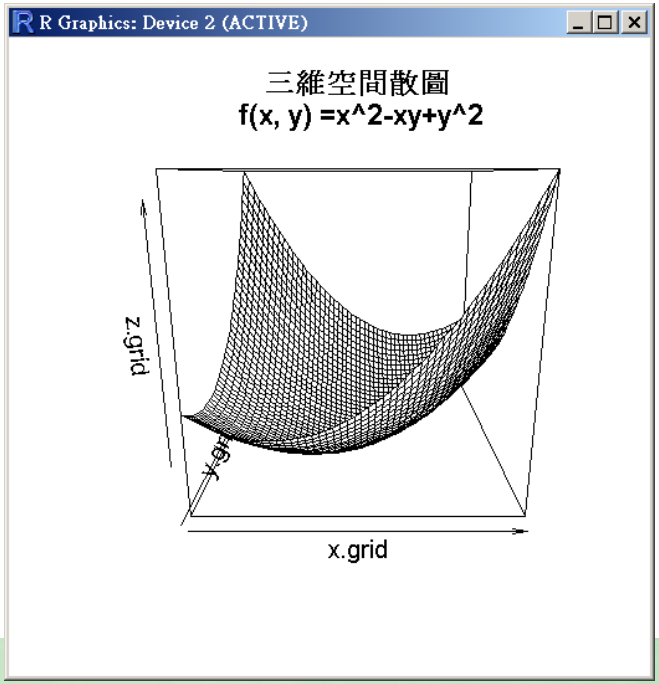



```
> xv <- seq(-3, 3, 0.01)
> yv <- dnorm(xv)
> plot(xv, yv, type="l")

> polygon(c(xv[xv <= -1]), c(yv[xv <= -1]), col="blue")

> x11()
> plot(xv, yv, type="l")
> polygon(c(xv[xv <= -1], -1), c(yv[xv <= -1], yv[xv==-3]), col="red")
```

3D透視圖: persp

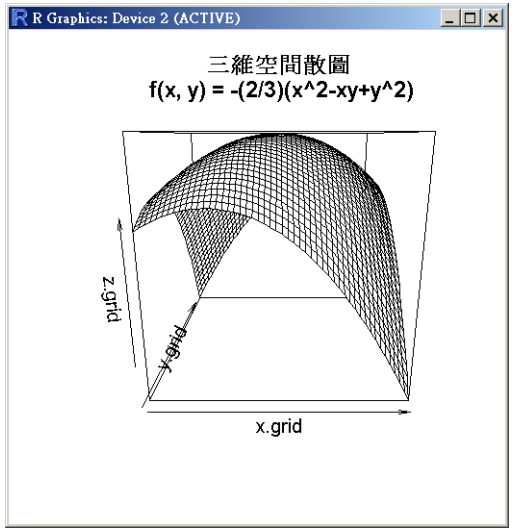
- 雙變數函數在三維空間的散佈圖
- NOTE: x-y平面上，格點數目愈多，散佈圖愈密集。
- plot f(x, y) =x^2-xy+y^2。

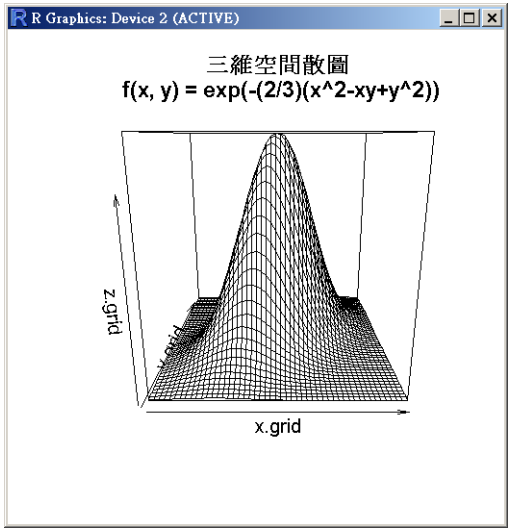
```
> ploy <- function(x, y){x^2-x*y+y^2}
> x.grid <- seq(-3, 3, length=50)
> y.grid <- seq(-3, 3, length=50)
> z.grid <- outer(x.grid, y.grid, FUN=ploy)
> ploy.title <- paste("三維空間散圖\n", "f(x, y) =x^2-xy+y^2")
> persp(x.grid, y.grid, z.grid, main= ploy.title)
```

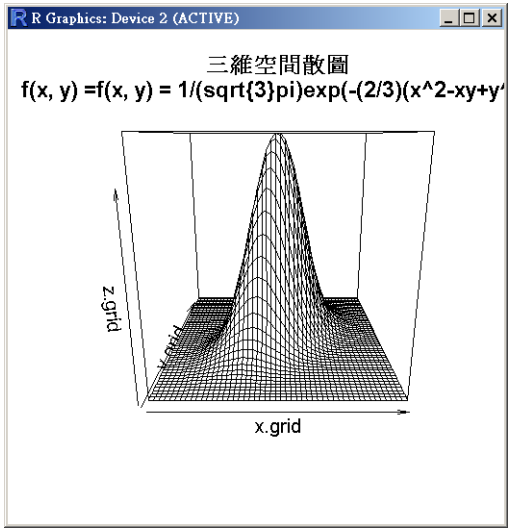


課堂練習4

- f(x, y) = -(2/3)(x^2-xy+y^2),
  xlim=c(-3, 3), ylim=c(-3, 3)
- f(x, y) = exp(-(2/3)(x^2-xy+y^2)),
  xlim=c(-3, 3), ylim=c(-3, 3)
- f(x, y) = 1/(sqrt{3}pi)exp(-(2/3)(x^2-xy+y^2)), xlim=c(-4, 4), ylim=c(-4, 4)

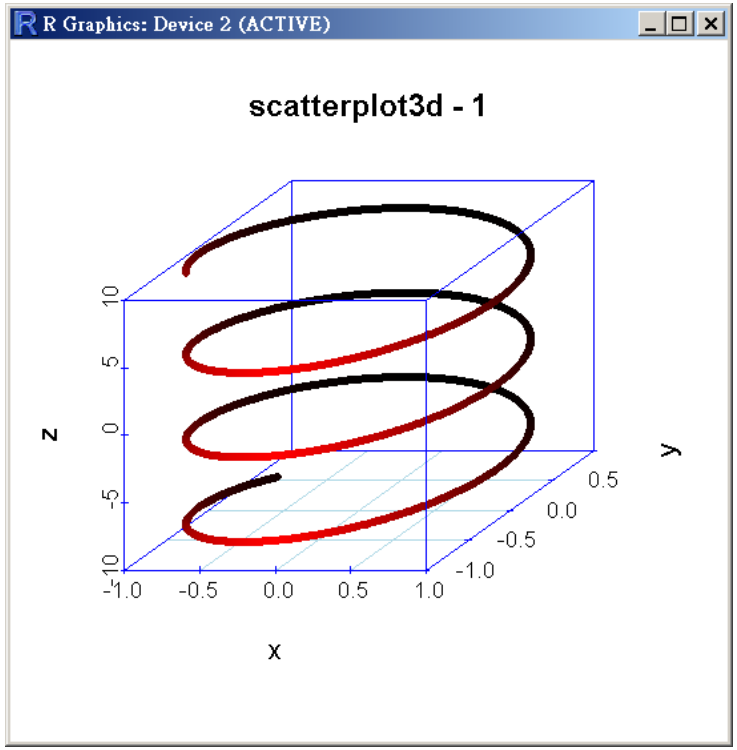






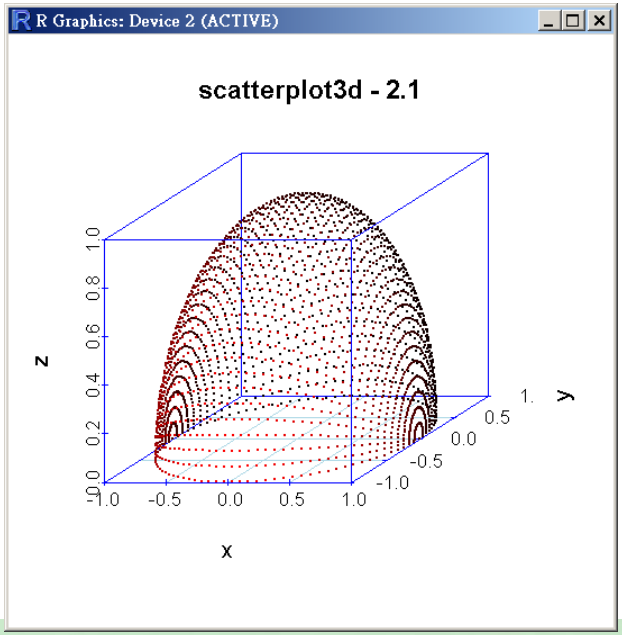
3D散佈圖

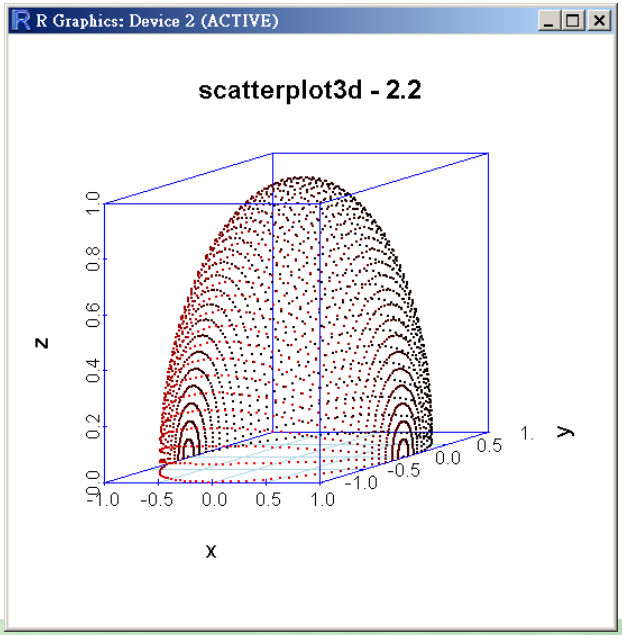
```
> library(scatterplot3d)
> z <- seq(-10, 10, 0.01)
> x <- cos(z)
> y <- sin(z)
> scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue",
   col.grid="lightblue", main="scatterplot3d - 1", pch=20)
```



3D散佈圖

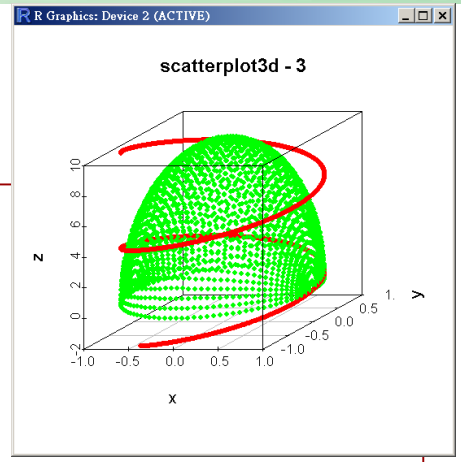
```
> temp <- seq(-pi, 0, length = 50)

> x <- c(rep(1, 50) %*% t(cos(temp)))
> y <- c(cos(temp) %*% t(sin(temp)))
> z <- c(sin(temp) %*% t(sin(temp)))

> scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue", col.grid="lightblue",
                main="scatterplot3d - 2.1", pch=20, cex.symbols=0.5)

> scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue", col.grid="lightblue",
                main="scatterplot3d - 2.2", pch=20, cex.symbols=0.5, angle= 20)
```



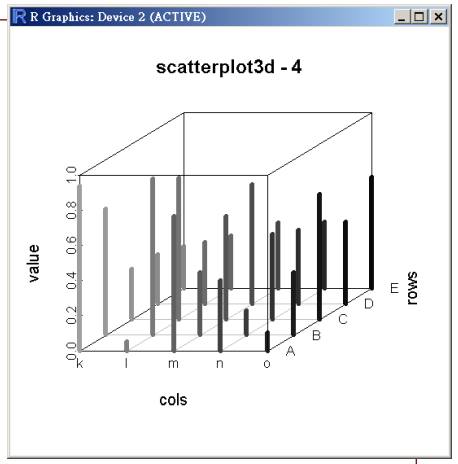


3D散佈圖

```
temp <- seq(-10, 10, 0.01)
x <- c(x, cos(temp))
y <- c(y, sin(temp))
z <- c(z, temp)
color <- c(color, rep("red", length(temp)))

scatterplot3d(x, y, z, color, pch=20, zlim=c(-2, 10), main="scatterplot3d - 3")
```

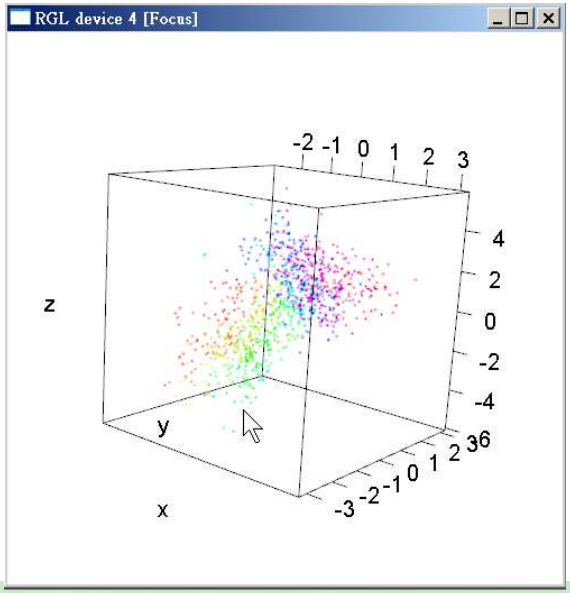


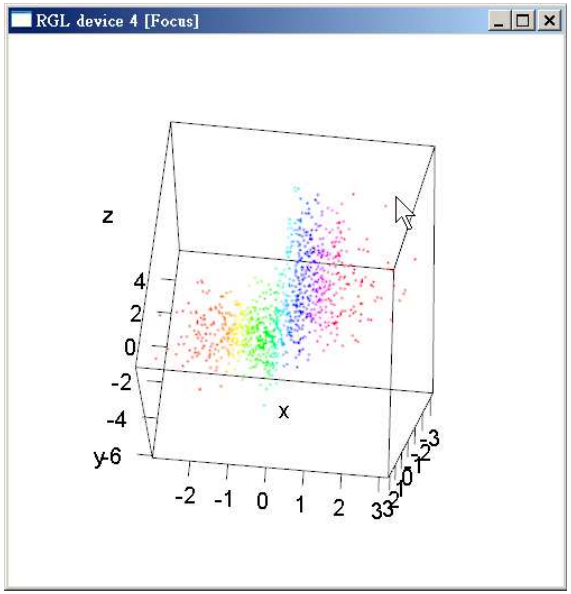

```
my.mat <- matrix(runif(25), nrow=5)
dimnames(my.mat) <- list(LETTERS[1:5], letters[11:15])
my.mat
col(my.mat)
row(my.mat)
cols <- as.vector(col(my.mat))
rows <- as.vector(row(my.mat))
v <- as.vector(my.mat)

s3d.dat <- data.frame(cols=cols, rows=rows, value=v)
scatterplot3d(s3d.dat, type="h", lwd=5, pch=" ", x.ticklabs=colnames(my.mat),
              y.ticklabs=rownames(my.mat),
              color=grey(25:1/40), main="scatterplot3d - 4")
```



3D visualization device system (OpenGL)

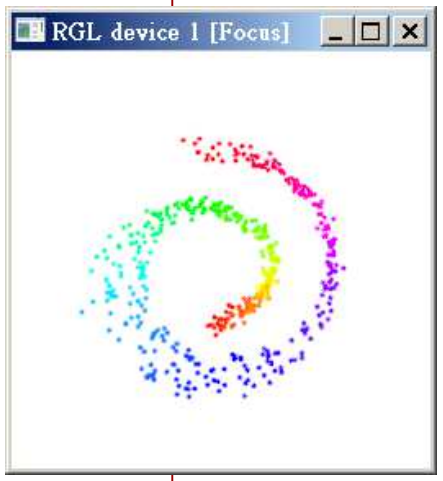
```
library(rgl)
open3d()
x <- sort(rnorm(1000))
y <- rnorm(1000)
z <- rnorm(1000) + atan2(x,y)
plot3d(x, y, z, col=rainbow(1000), size=2)

M <- par3d("userMatrix")
play3d(par3dinterp(userMatrix=list(M, rotate3d(M, pi/2, 1,
0, 0), rotate3d(M, pi/2, 0, 1, 0) ) ), duration=4)
```





Swiss Roll Data Set

```
library(rgl)
swissroll <- function(n, sigma=0.05){

        angle <- (3*pi/2)*(1+2*runif(n));
        height <- runif(n);
        xdata <-  cbind(angle*cos(angle), height, angle*sin(angle))
        xdata <- scale(xdata) + matrix(rnorm(n*3, 0, sigma), n, 3)

        order.angle <- order(angle)
        sort.angle <- sort(order.angle, index.return=TRUE)
        col.id <- rainbow(n)
        my.color <- col.id[sort.angle$ix]

        colnames(xdata) <- paste("x", 1:3, sep="")

    return(list(xdata=xdata, angle=angle, color=my.color))
}

swissdata <- swissroll(500)
xdata <- swissdata$xdata
x.color <- swissdata$color
open3d()
plot3d(xdata[,1], xdata[,2], xdata[,3], col=x.color, size=3, xlab="",
ylab="", zlab="", axes = FALSE)
```

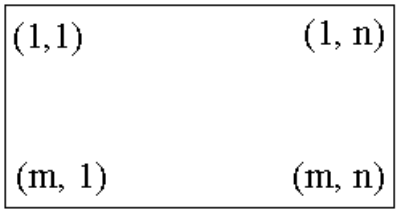


Image

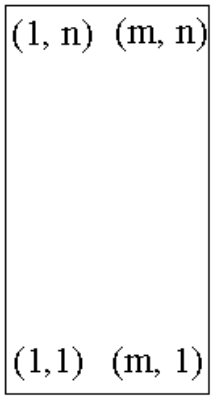
Display a Color Image

- `image(x, ...)`
- `image(x, y, z, zlim, xlim, ylim, col = heat.colors(12), add = FALSE, xaxs = "i", yaxs = "i", xlab, ylab, breaks, oldstyle = FALSE, ...)`

Data Matrix

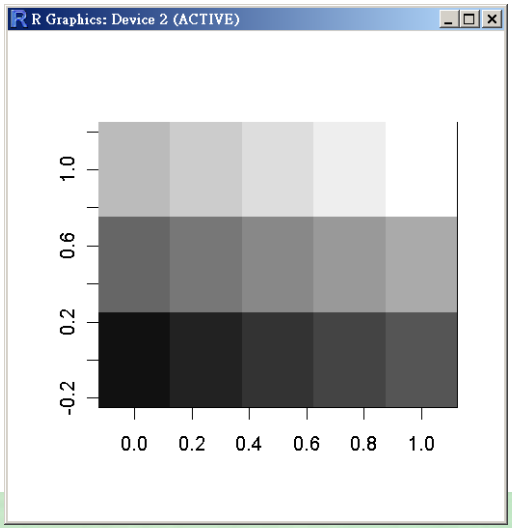


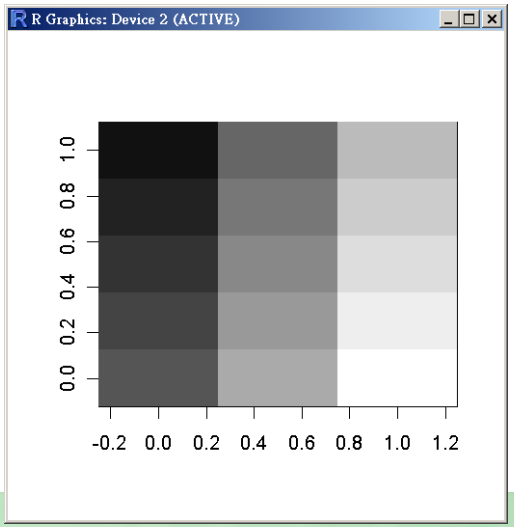
Image




```
> my.data <- matrix(c(1:15), ncol=3, nrow=5)
> my.data
> image(my.data, col=grey(1:15/15))

> image(t(my.data)[,nrow(my.data):1],
col=grey(1:15/15))
```

```
> my.data
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

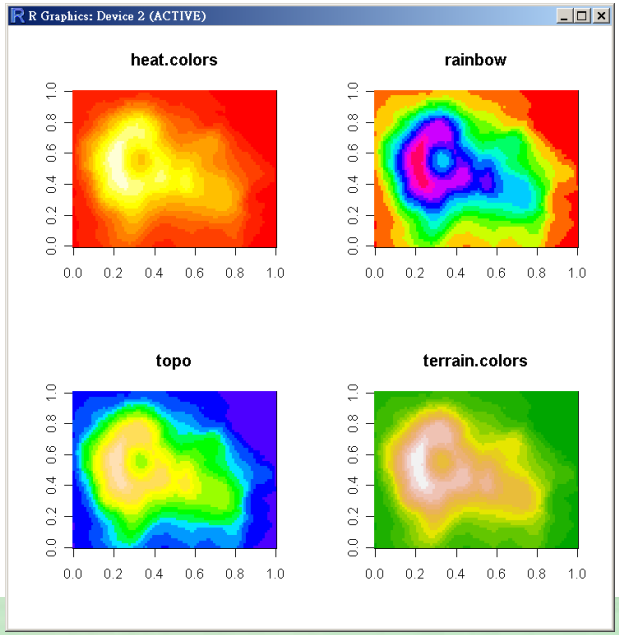




Example: volcano

- Maunga Whau (Mt Eden, 伊甸山) is one of about 50 volcanos in the Auckland volcanic field. This data set gives topographic information for Maunga Whau on a 10m by 10m grid.
  - A matrix with 87 rows and 61 columns,
  - rows: grid lines running east to west
  - columns: grid lines running south to north.

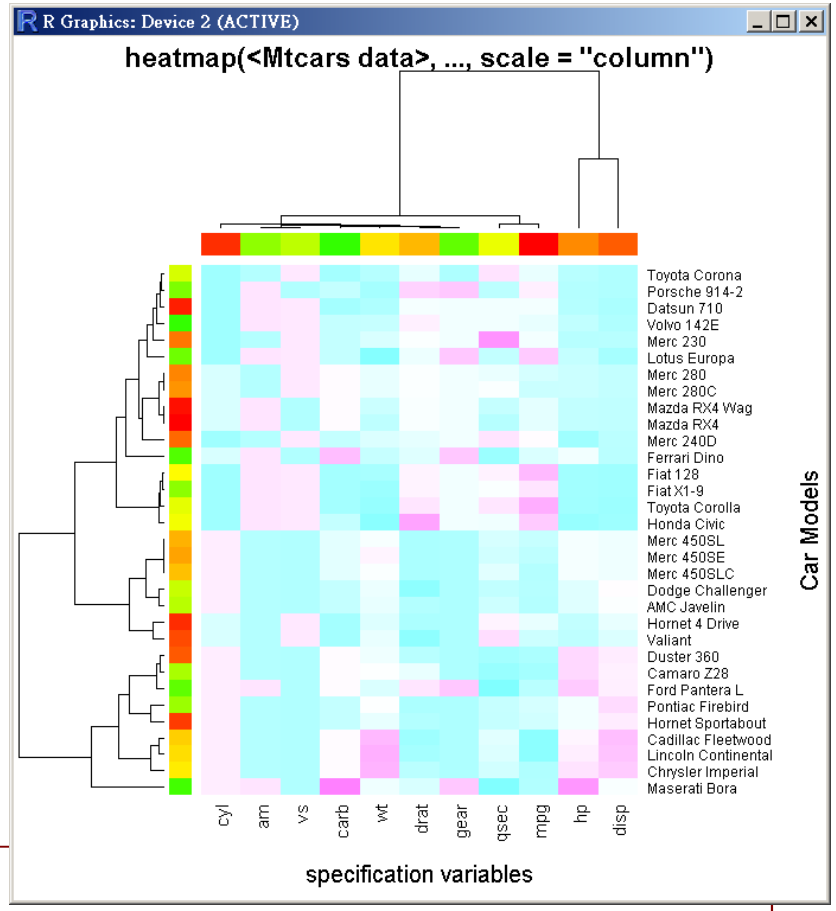
```
> data(volcano)
> par(mfrow = c(2, 2))
> image(volcano, main = "heat.colors")
> image(volcano, main = "rainbow", col = rainbow(15))
> image(volcano, main = "topo", col = topo.colors(15))
> image(volcano, main = "terrain.colors",col = terrain.colors(15))
```



Heatmap

- **heatmap {stats}**: A heat map is a false color image (basically **image(t(x))**) with a dendrogram added to the left side and to the top.
- A deafult dendrograms are computed as **as.dendrogram(hclustfun(distfun(X)))**

```
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

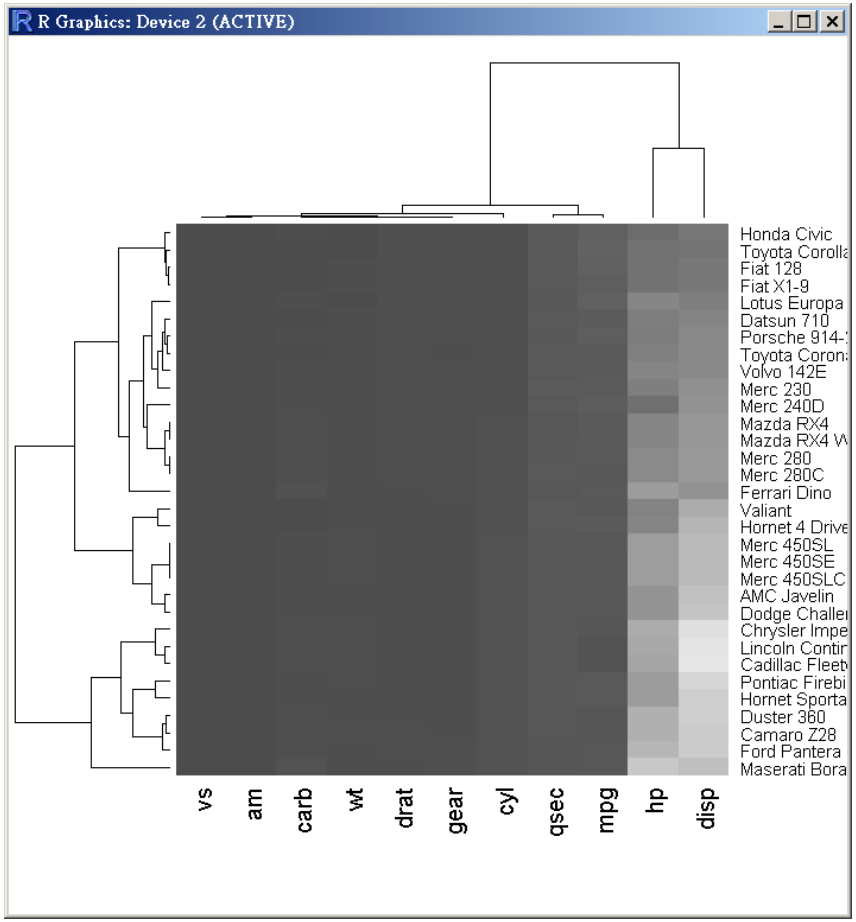



```
x  <- as.matrix(mtcars)
rc <- rainbow(nrow(x), start = 0, end = .3)
cc <- rainbow(ncol(x), start = 0, end = .3)
hv <- heatmap(x, col = cm.colors(256), scale = "column",
              RowSideColors = rc, ColSideColors = cc, margins = c(5,10),
              xlab = "specification variables", ylab =  "Car Models",
              main = "heatmap(<Mtcars data>, ..., scale = \"column\")")
```

hmap{seriation}

Plot heat map reordered by different algorithms

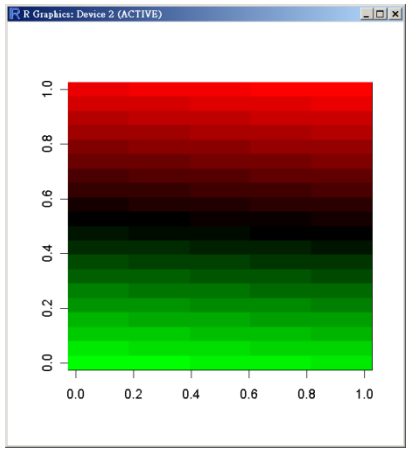
```
hmap(x, distfun = dist, hclustfun = hclust,
    method = NULL, control = NULL, options = NULL, ...)
```

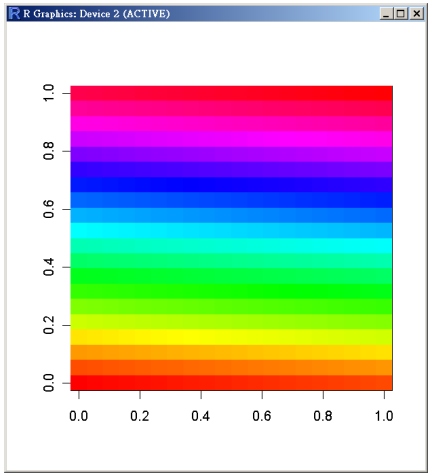


課堂練習5: color.Palette

```
> GBRcol <- color.Palette(low="green", mid="black", high="red")
> image(matrix(1:400, ncol=20), col=GBRcol)
> image(matrix(1:400, ncol=20), col=rainbow(400))
```

```
color.Palette <- function(low = "black",
                          high = c("green", "red"),
                          mid="black",
                          k =50)
{
    low <- col2rgb(low)/255
    high <- col2rgb(high)/255

    if(is.null(mid)){
        r <- seq(low[1], high[1], len = k)
        g <- seq(low[2], high[2], len = k)
        b <- seq(low[3], high[3], len = k)
      }
    if(!is.null(mid)){
      k2 <- round(k/2)
      mid <- col2rgb(mid)/255
      r <- c(seq(low[1], mid[1], len = k2),
             seq(mid[1], high[1], len = k2))
      g <- c(seq(low[2], mid[2], len = k2),
             seq(mid[2], high[2], len = k2))
      b <- c(seq(low[3], mid[3], len = k2),
             seq(mid[3], high[3], len = k2))
    }
    rgb(r, g, b)
}
```



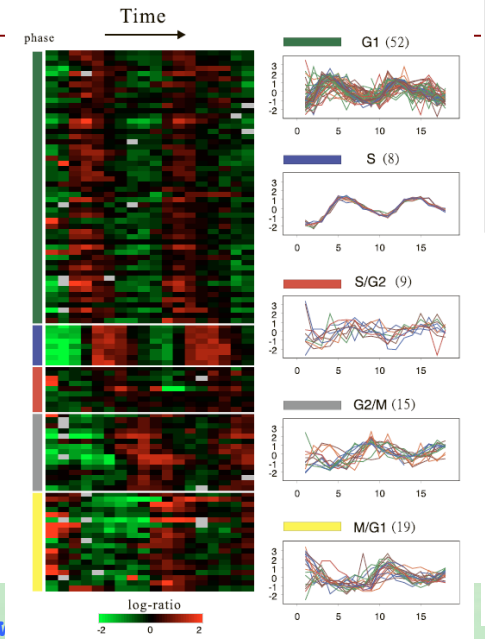


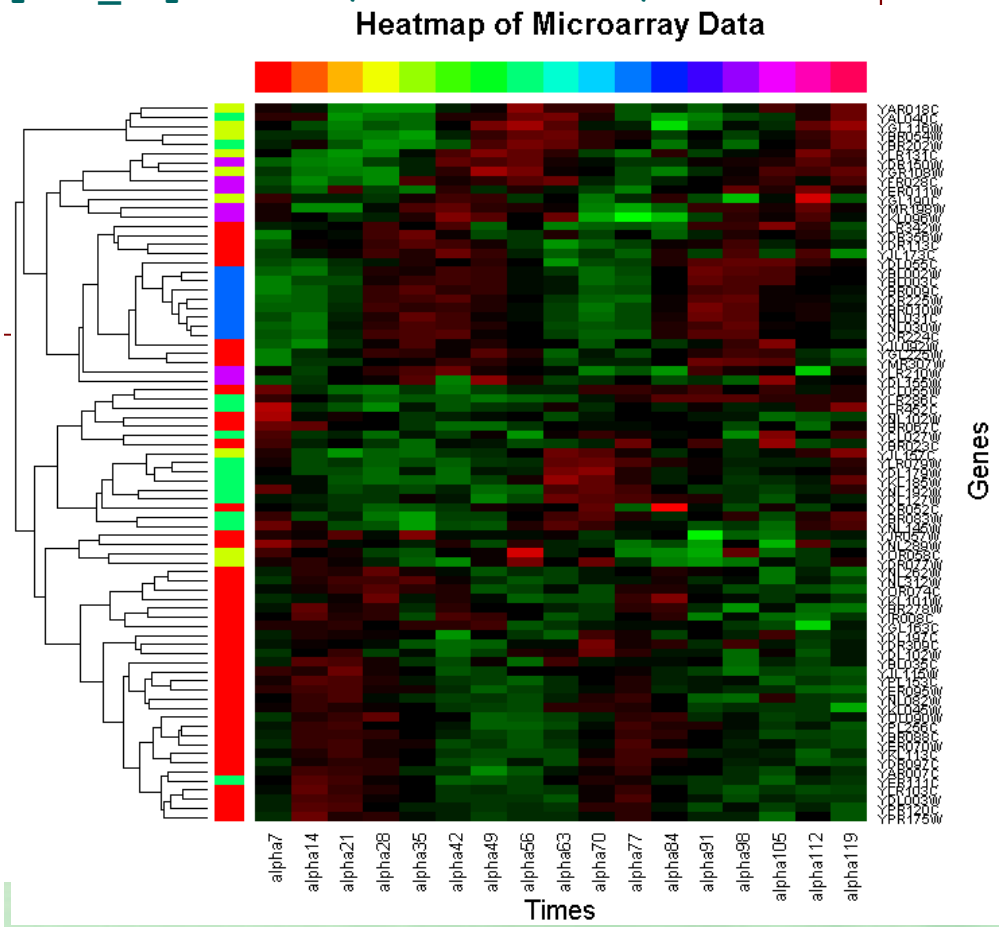
課堂練習6: Microarray Data

```
cell.sdata <- (cell.data – apply(cell.data, 1, mean))/
                          sqrt(apply(cell.data, 1, var))
```

- Lu and Wu (2010)
  - Time course data: every 7 minutes and totally 18 time points.
  - Known genes: there are 103 cell cycle-regulated genes by traditional method in G1, S, S/G2, G2/M, or M/G1. (Remove NA's: 79.)

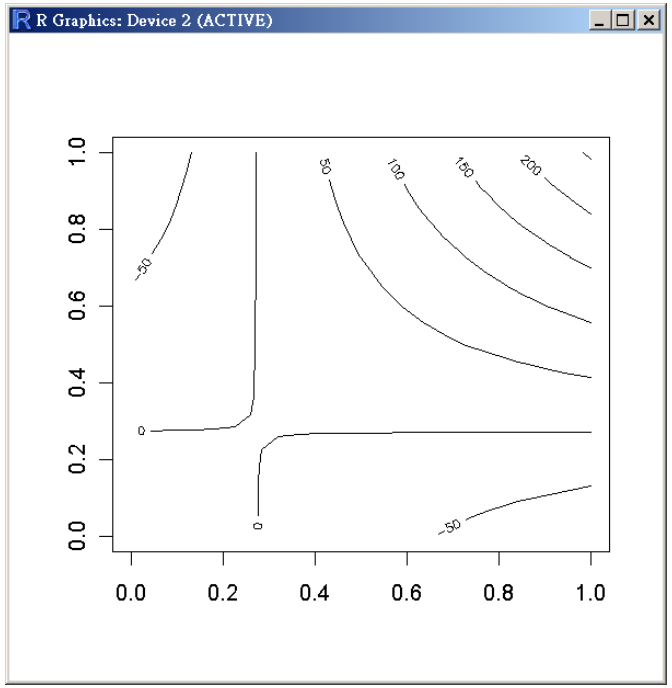
```
cell.matrix  <- read.table("YeastCellCycle_alpha.txt", header=TRUE,
row.names=1)
n <- dim(cell.matrix)[1]
p <- dim(cell.matrix)[2]-1
cell.data <- cell.matrix[,2:p+1]
gene.phase <- cell.matrix[,1]
#cell.sdata <- scale(cell.data)
...
```



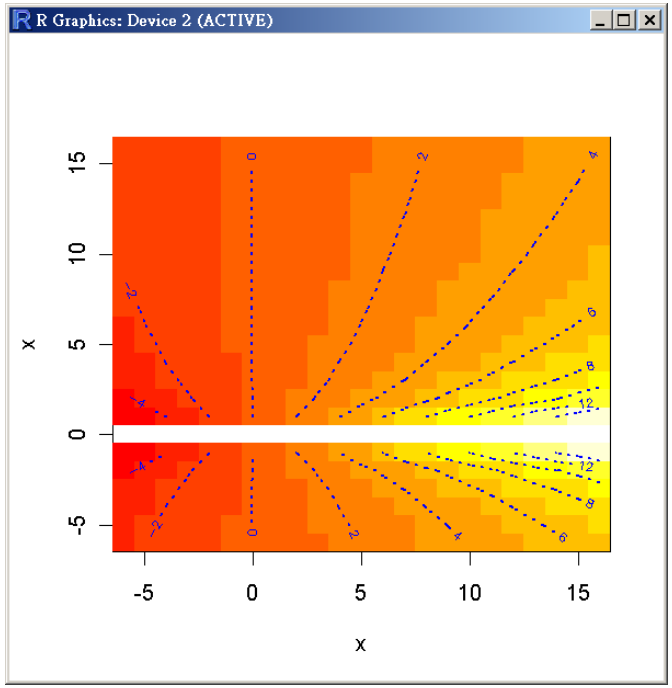


等高線圖 (Contours)

```
> x <- -6:16; length(x)
> my.data <- outer(x, x);dim(my.data)
> my.data
> contour(my.data, method = "edge")
```

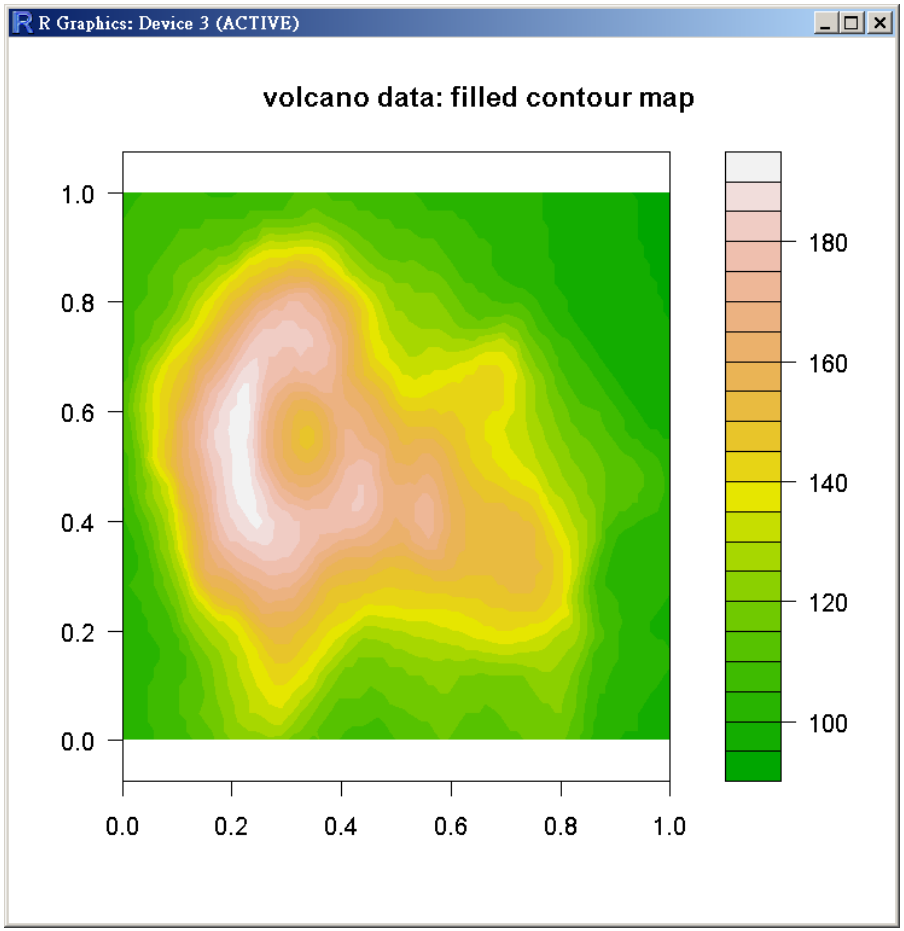



```
> image(x, x, z)
> contour(x, x, z, col = "blue", add = TRUE,
          method = "edge", lwd=1.5, lty=3)
```



Filled Contours

```
> filled.contour(volcano, color.palette = terrain.colors, asp = 1)
> title(main = "volcano data: filled contour map")
# asp: the y/x aspect ratio.
```



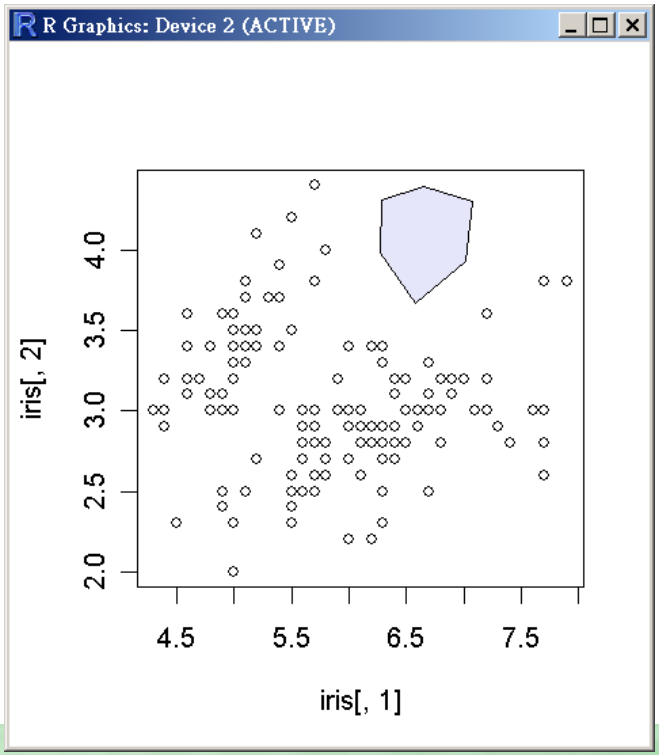
Interactive Graphics

`> par(ask=TRUE)` #the user is prompted before a new page of output is begin.

`> locator()` #allow users to click within a plot and return the coordinates where the mouse click occurred.

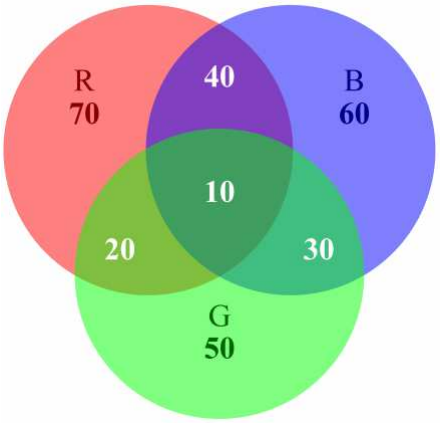
`> identify()` #use to add labels to data symbols on a plot. The data point closet to the mouse click gets labelled.

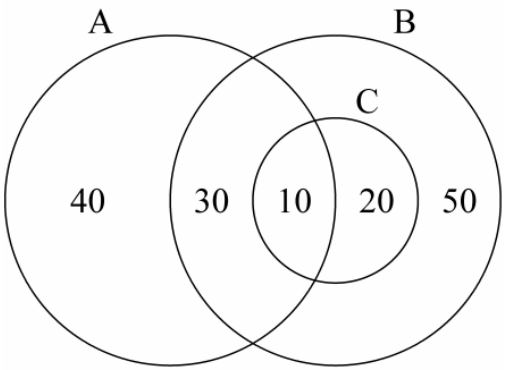
```
> plot(iris[,1], iris[,2])
> locations <- locator(6)
> polygon(locations, col="lavender")
```

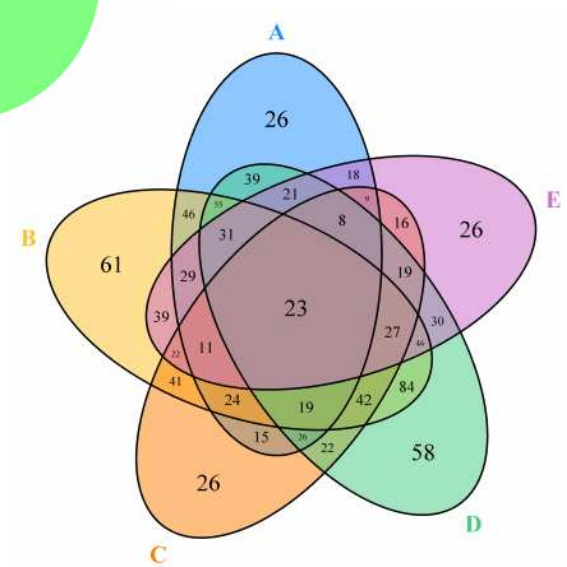


Venn Diagrams

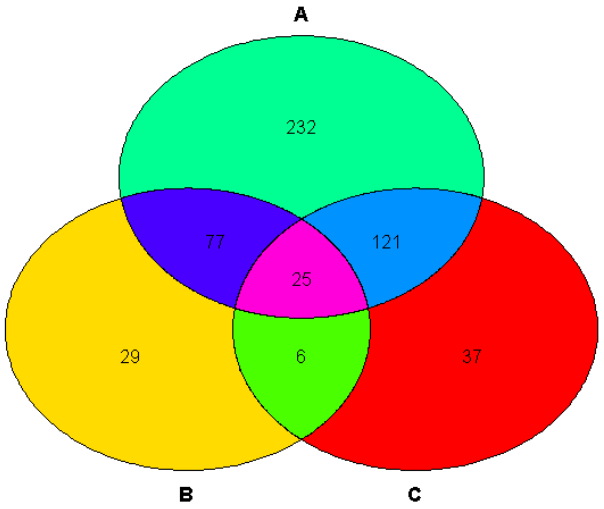
Package **VennDiagram**

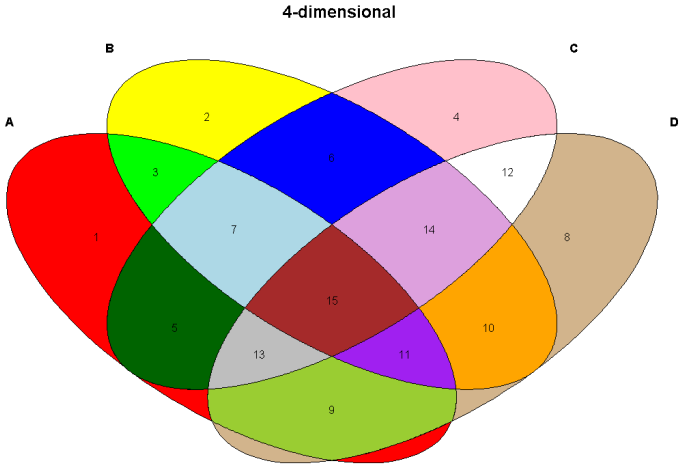






Package **colorfulVennPlot**



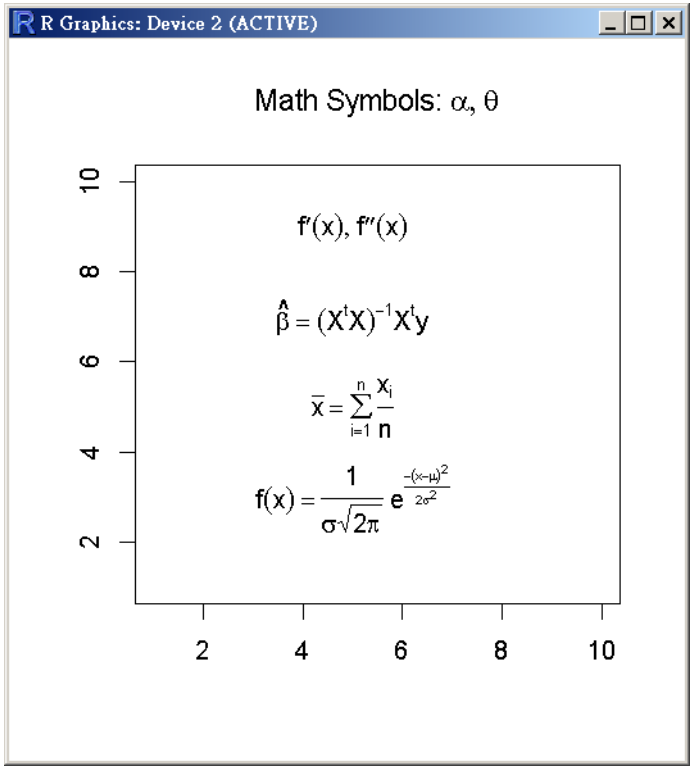


圖形上的數學符號: plotmath

```
main.ex <- expression(paste("Math Symbols: ", list(alpha, theta)))
plot(1:10, 1:10, type="n", main=main.ex, xlab="", ylab="")
text(5, 9, expression(list({f * minute}(x), {f * second}(x))))
text(5, 7, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
text(5, 5, expression(bar(x) == sum(frac(x[i], n), i==1, n)))
ex <- expression(f(x)==paste(frac(1, sigma*sqrt(2*pi)), " ",
                             plain(e)^{frac(-(x-mu)^2, 2*sigma^2)}))
text(5, 3, labels = ex)
```

demo(plotmath)

Grouping		group("(", list(a, b), "]")	(a, b]
(x + y) * z	(x+y)z	bgroup("(", atop(x, y), ")"	$\binom{x}{y}$
x^y + z	x^y+z	group(lceil, x, rceil)	$\lceil x \rceil$
x^(y + z)	$x^{(y+z)}$	group(lfloor, x, rfloor)	$\lfloor x \rfloor$
x^{y + z}	x^{y+z}	group("|", x, "|")	|x|



demo(plotmath)

Arithmetic Operators	
x + y	$x+y$
x - y	$x-y$
x * y	xy
x/y	x/y
x %+-% y	$x \pm y$
x%/%y	$x \div y$
x %*% y	$x \times y$
x %.% y	$x \cdot y$
-x	$-x$
+x	$+x$
Sub/Superscripts	
x[i]	x_i
x^2	x^2
Juxtaposition	
x * y	xy
paste(x, y, z)	xyz
Radicals	
sqrt(x)	$\sqrt{x}$
sqrt(x, y)	$\sqrt[y]{x}$

Lists	
list(x, y, z)	x, y, z
Relations	
x == y	$x = y$
x != y	$x \neq y$
x < y	$x < y$
x <= y	$x \leq y$
x > y	$x > y$
x >= y	$x \geq y$
x %~~% y	$x \approx y$
x %=~% y	$x \cong y$
x %==% y	$x \equiv y$
x %prop% y	$x \propto y$
x %~% y	$x \sim y$
Typeface	
plain(x)	x
italic(x)	*x*
bold(x)	**x**
bolditalic(x)	***x***
underline(x)	x

Ellipsis	
list(x[1], ..., x[n])	$x_1, \ldots, x_n$
x[1] + ... + x[n]	$x_1 + \cdots + x_n$
list(x[1], cdots, x[n])	$x_1, \cdots, x_n$
x[1] + ldots + x[n]	$x_1 + \ldots + x_n$
Set Relations	
x %subset% y	$x \subset y$
x %subseteq% y	$x \subseteq y$
x %supset% y	$x \supset y$
x %supseteq% y	$x \supseteq y$
x %notsubset% y	$x \not\subset y$
x %in% y	$x \in y$
x %notin% y	$x \notin y$
Accents	
hat(x)	$\hat{x}$
tilde(x)	$\tilde{x}$
ring(x)	$\mathring{x}$
bar(xy)	$\overline{xy}$
widehat(xy)	$\widehat{xy}$
widetilde(xy)	$\widetilde{xy}$

Arrows	
x %<->% y	$x \leftrightarrow y$
x %->% y	$x \rightarrow y$
x %<-% y	$x \leftarrow y$
x %up% y	$x \uparrow y$
x %down% y	$x \downarrow y$
x %<=>% y	$x \Leftrightarrow y$
x %=>% y	$x \Rightarrow y$
x %<=% y	$x \Leftarrow y$
x %dblup% y	$x \Uparrow y$
x %dbldown% y	$x \Downarrow y$
Symbolic Names	
Alpha - Omega	$A - \Omega$
alpha - omega	$\alpha - \omega$
phi1 + sigma1	$\varphi + \varsigma$
Upsilon1	Υ
infinity	∞
32 * degree	32°
60 * minute	60′
30 * second	30″

demo(plotmath)

Ellipsis	
list(x[1], ..., x[n])	$x_1, \ldots, x_n$
x[1] + ... + x[n]	$x_1 + \cdots + x_n$
list(x[1], cdots, x[n])	$x_1, \cdots, x_n$
x[1] + ldots + x[n]	$x_1 + \ldots + x_n$

Set Relations	
x %subset% y	$x \subset y$
x %subseteq% y	$x \subseteq y$
x %supset% y	$x \supset y$
x %supseteq% y	$x \supseteq y$
x %notsubset% y	$x \not\subset y$
x %in% y	$x \in y$
x %notin% y	$x \notin y$

Accents	
hat(x)	$\hat{x}$
tilde(x)	$\tilde{x}$
ring(x)	$\mathring{x}$
bar(xy)	$\overline{xy}$
widehat(xy)	$\widehat{xy}$
widetilde(xy)	$\widetilde{xy}$

Arrows	
x %<->% y	$x \leftrightarrow y$
x %->% y	$x \rightarrow y$
x %<-% y	$x \leftarrow y$
x %up% y	$x \uparrow y$
x %down% y	$x \downarrow y$
x %<=>% y	$x \Leftrightarrow y$
x %=>% y	$x \Rightarrow y$
x %<=% y	$x \Leftarrow y$
x %dblup% y	$x \Uparrow y$
x %dbldown% y	$x \Downarrow y$

Symbolic Names	
Alpha - Omega	$A - \Omega$
alpha - omega	$\alpha - \omega$
phi1 + sigma1	$\varphi + \varsigma$
Upsilon1	Υ
infinity	∞
32 * degree	32°
60 * minute	$60'$
30 * second	$30''$

Style	
displaystyle(x)	x
textstyle(x)	x
scriptstyle(x)	x
scriptscriptstyle(x)	x

Spacing	
x ~ ~y	$x\ y$
x + phantom(0) + y	$x+\ \ +y$
x + over(1, phantom(0))	$x+\frac{1}{\ }$

Fractions	
frac(x, y)	$\frac{x}{y}$
over(x, y)	$\frac{x}{y}$
atop(x, y)	$\begin{matrix}x\\y\end{matrix}$

C:\Program Files\R\R-3.1.0\library\graphics\demo\plotmath.R

Advance Graphics

- Lattice: Multivariate Data Visualization with R
- Grid package
- R Graphics 2nd Edition, https://www.stat.auckland.ac.nz/~paul/RG2e/

Table 4.1
The plotting functions available in lattice

Lattice Function	Description	Traditional Analogue
`barchart()`	Barcharts	`barplot()`
`bwplot()`	Boxplots Box-and-whisker plots	`boxplot()`
`densityplot()`	Conditional kernel density plots Smoothed density estimate	*none*
`dotplot()`	Dotplots Continuous versus categorical	`dotchart()`
`histogram()`	Histograms	`hist()`
`qqmath()`	Quantile–quantile plots Data set versus theoretical distribution	`qqnorm()`
`stripplot()`	Stripplots One-dimensional scatterplot	`stripchart()`
`qq()`	Quantile–quantile plots Data set versus data set	`qqplot()`
`xyplot()`	Scatterplots	`plot()`
`levelplot()`	Level plots	`image()`
`contourplot()`	Contour plots	`contour()`
`cloud()`	3-dimensional scatterplot	*none*
`wireframe()`	3-dimensional surfaces	`persp()`
`splom()`	Scatterplot matrices	`pairs()`
`parallel()`	Parallel coordinate plots	*none*

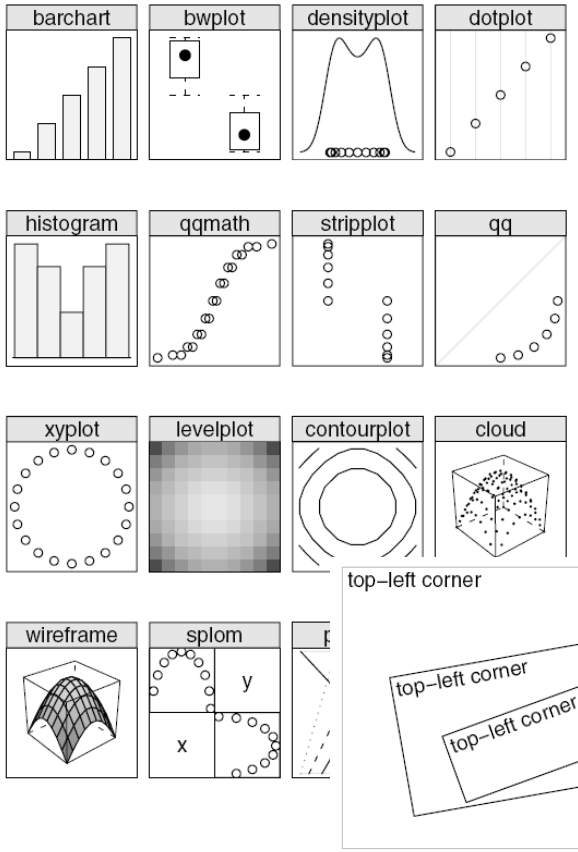
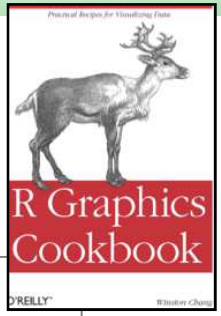


Figure 5.12
Popping a viewport. When a viewport is popped, the drawing context reverts to the parent viewport. In this figure, the second viewport (pushed in Figure 5.11) has been popped to go back to the first viewport (pushed in Figure 5.10). This time text has been drawn in the bottom-right corner.

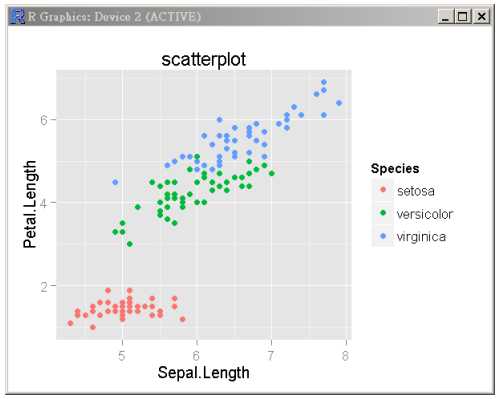
Package: ggplot2

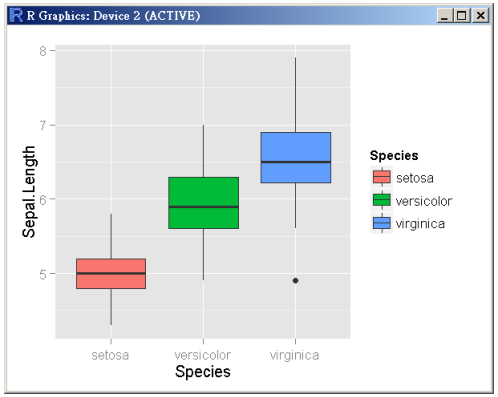
- Hadley Wickham, ggplot2: Elegant Graphics for Data Analysis: http://ggplot2.org/
- Cookbook for R: http://www.cookbook-r.com/Graphs/

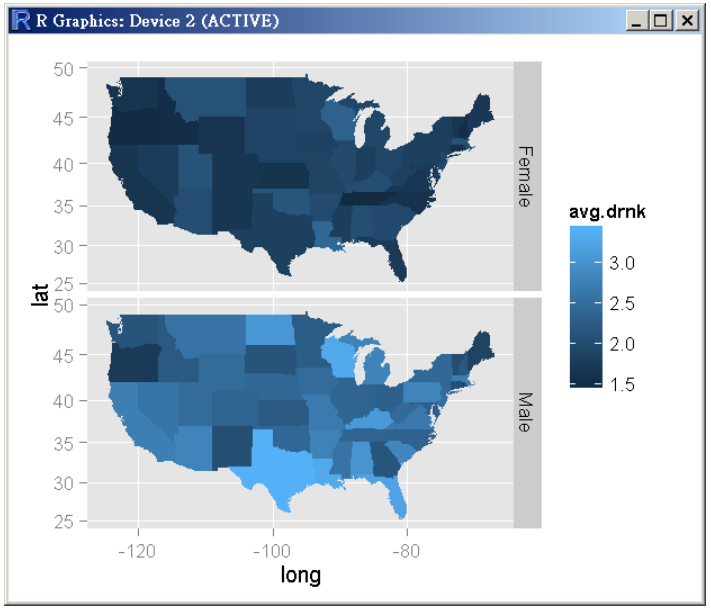



```
qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE,
  geom = "auto", stat = list(NULL), position = list(NULL), xlim = c(NA,
  NA), ylim = c(NA, NA), log = "", main = NULL,
  xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), asp = NA)
```

```
library(ggplot2)
qplot(Sepal.Length, Petal.Length, geom="point",
 data=iris, colour = Species, main="scatterplot")
qplot(Species, Sepal.Length, geom="boxplot",
 fill=Species, data=iris)
```

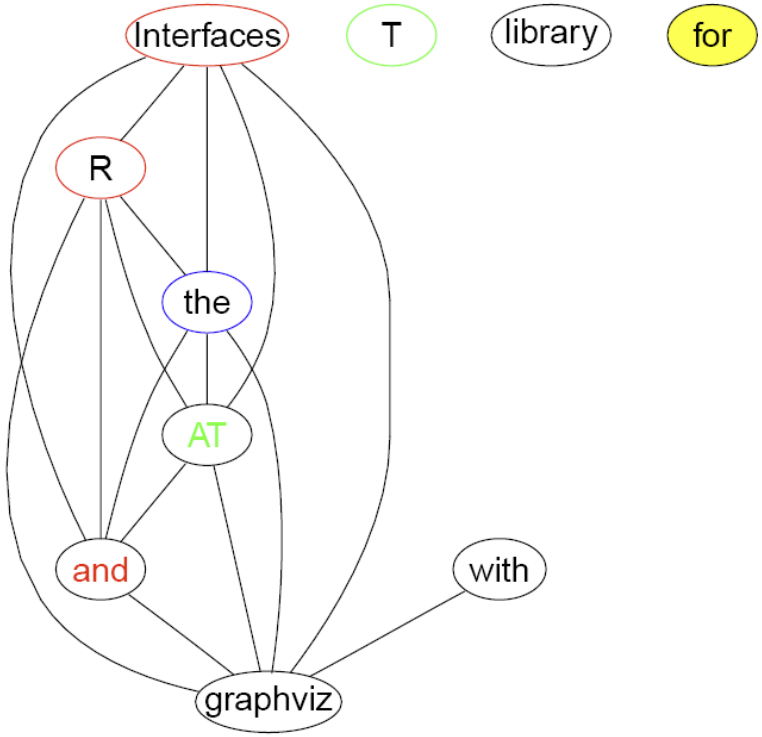






Networks: Rgraphviz, igraph

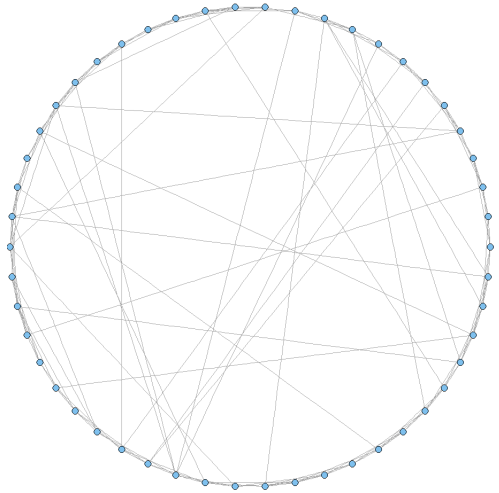
Rgraphviz: Interfaces R with the AT and T graphviz library for plotting R graph objects from the graph package.

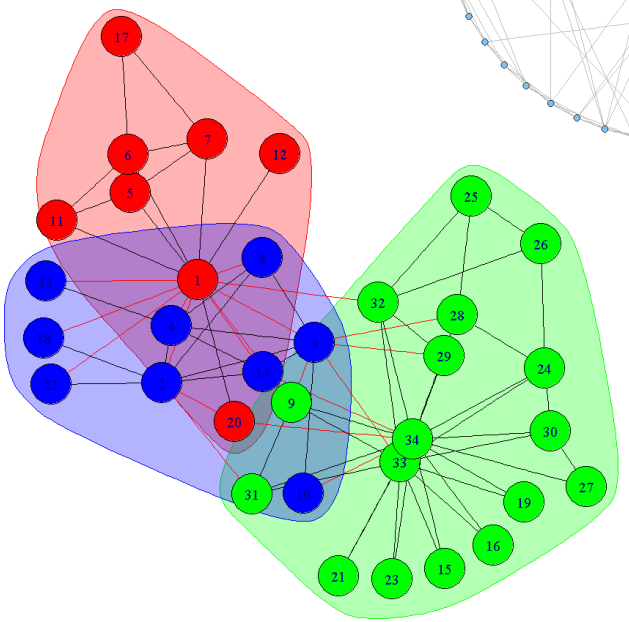


The network analysis package

```
demo(package="igraph")
demo(package="igraph", community)
demo(package="igraph", smallworld)
```

http://igraph.org





Plotting on Google Static Maps in R: RgoogleMaps

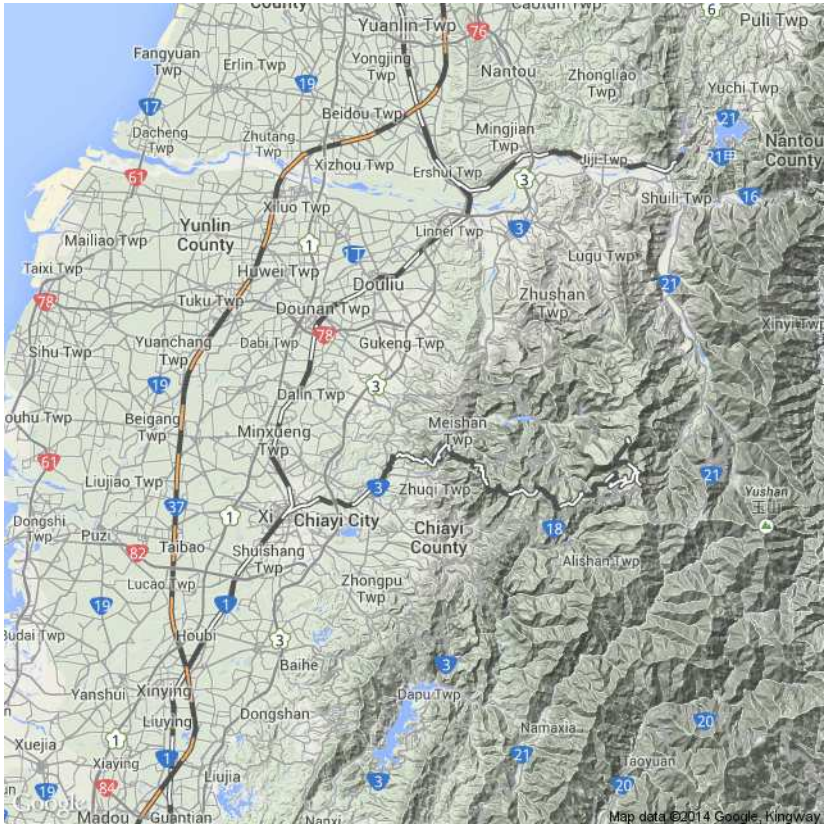
語法:

```
GetMap(center = c(lat = 42, lon = -76), size = c(640, 640), destfile,
    zoom = 12, markers, path = "", span, frame, hl, sensor = "true",
    maptype = c("roadmap", "mobile", "satellite", "terrain",
        "hybrid", "mapmaker-roadmap", "mapmaker-hybrid")[2],
    format = c("gif", "jpg", "jpg-baseline", "png8", "png32")[5],
    RETURNIMAGE = TRUE, GRAYSCALE = FALSE, NEWMAP = TRUE, SCALE = 1,
    verbose = 0)
```


```
library(RgoogleMaps)
WorldMap <- GetMap(center=c(0,0), zoom =1,
                   destfile = "World1.png")
```

台灣地圖

```
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")
```

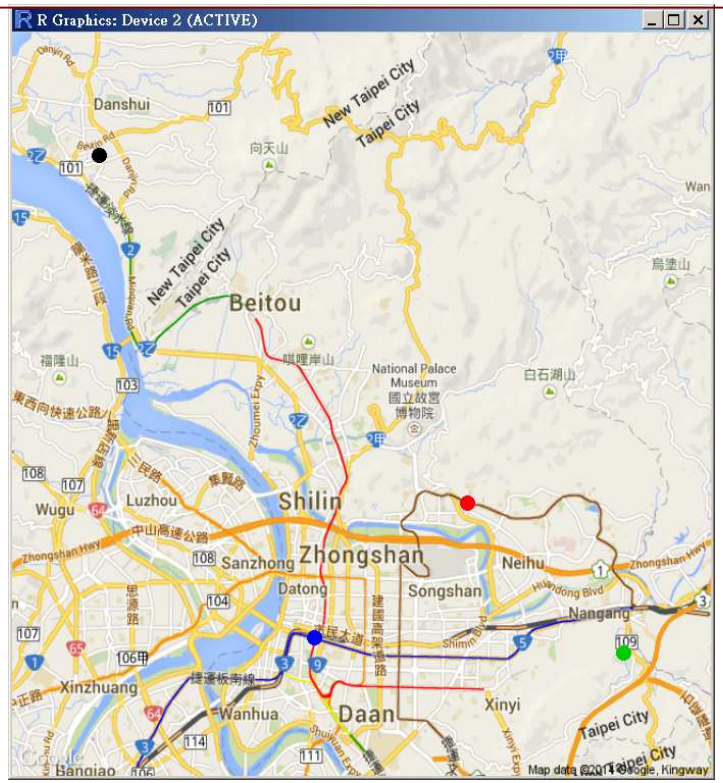


於地圖上標記

```
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)
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)
```

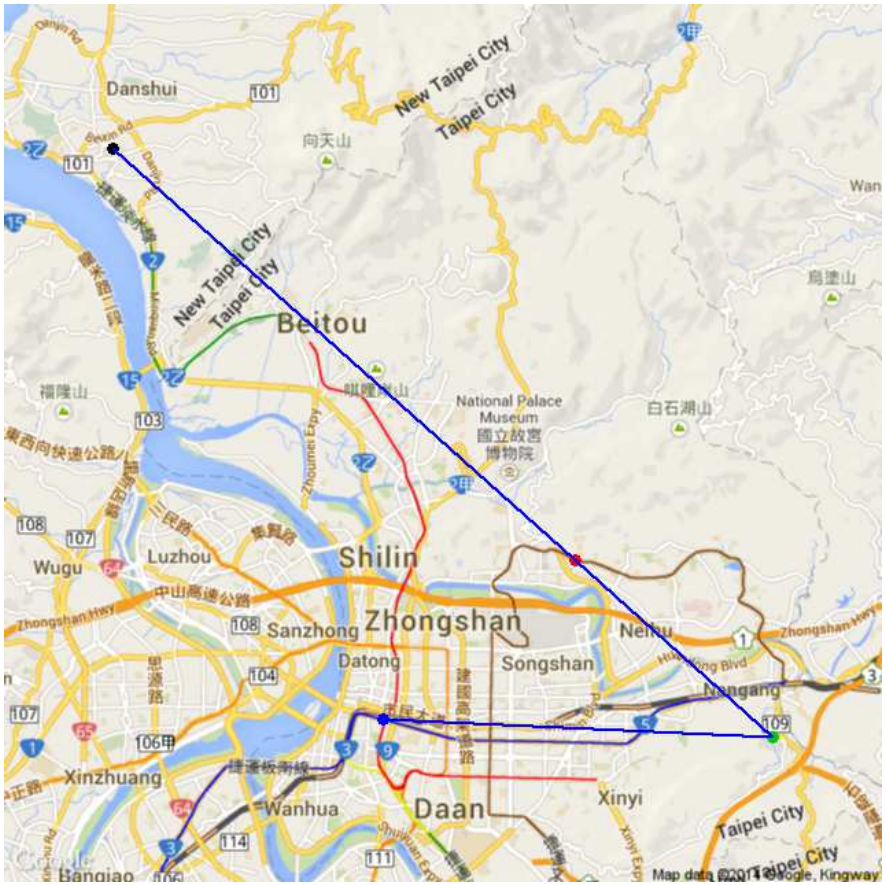
查詢經緯度
http://card.url.com.tw/realads/map_latlng.php

- 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

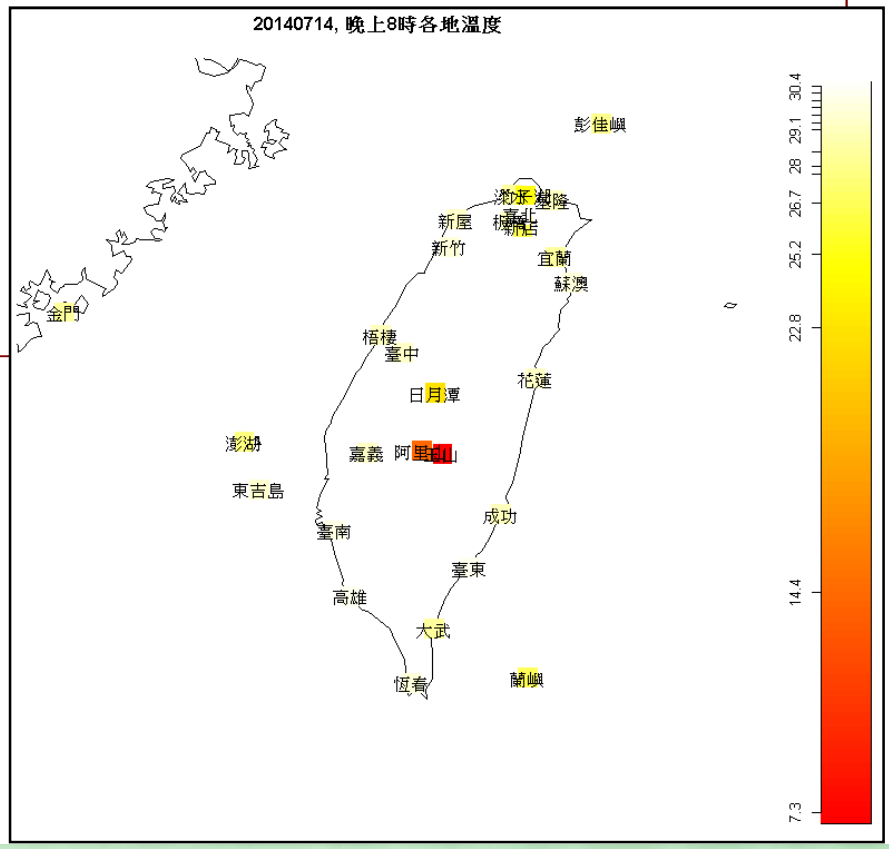
```
png("my2.png", 640, 640)
tmp <-  PlotOnStaticMap(MyMap, lat = My.markers[,"lat"], lon = My.markers[,"lon"],
cex=2.5, pch=20, col=1:4, add=F)
tmp <-  PlotOnStaticMap(MyMap, lat = My.markers[,"lat"], lon = My.markers[,"lon"],
col="blue", add=T, FUN = lines, lwd = 2)
dev.off()
```



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)
```

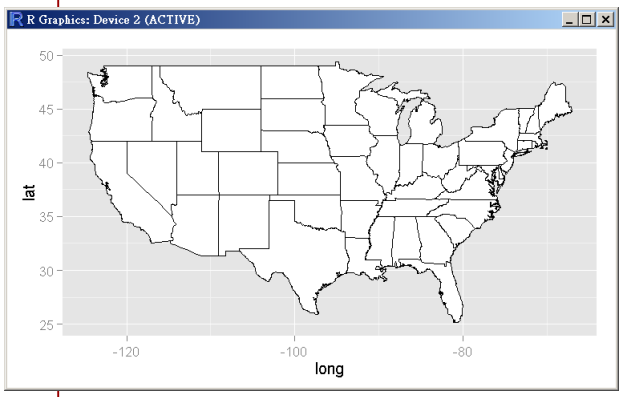


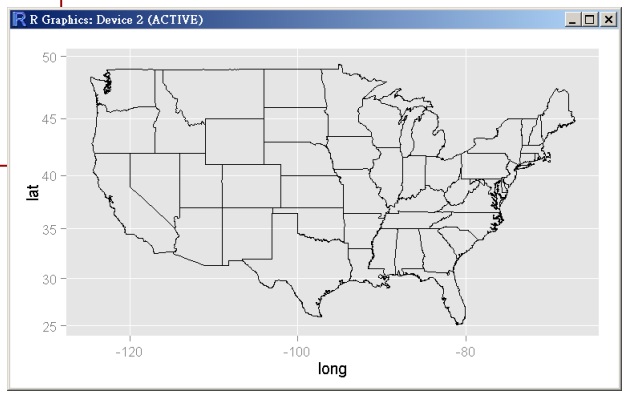
See also:

Spatial data in R: Using R as a GIS
http://pakillo.github.io/R-GIS-tutorial/

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")
```



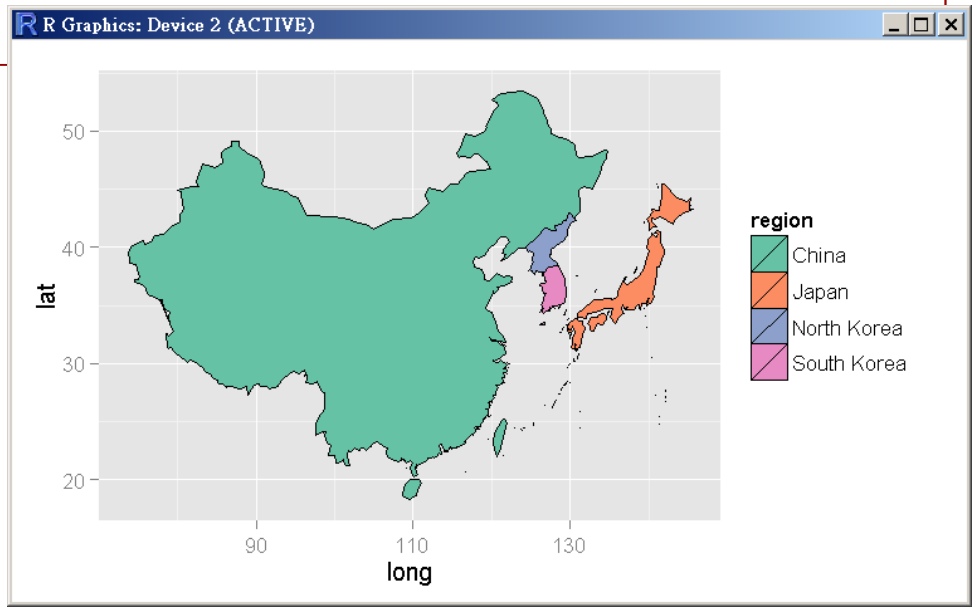


mercator: equally spaced straight meridians, conformal, straight compass courses

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")
```



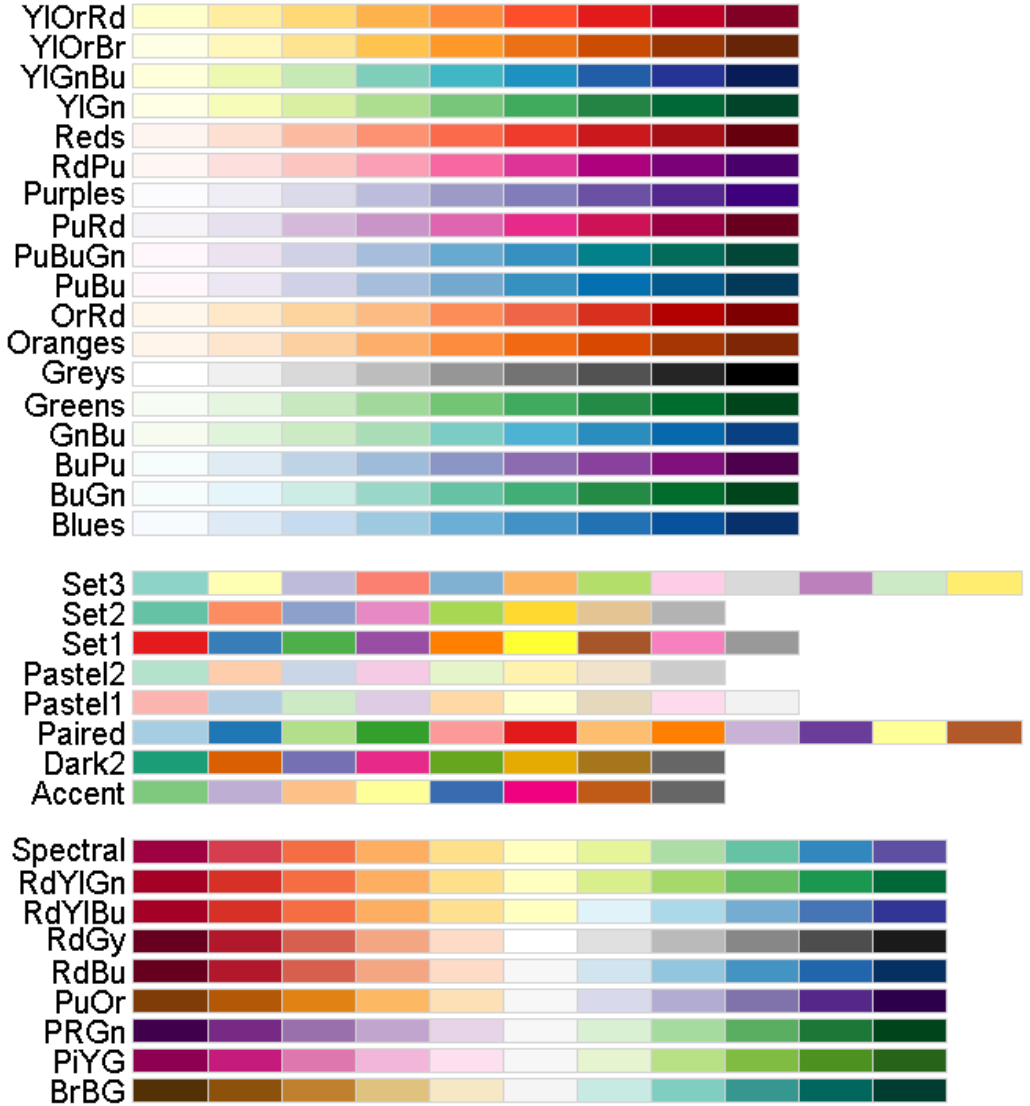
NOTE:

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**

RColorBrewer

```
> library(RColorBrewer)
> display.brewer.all()
```



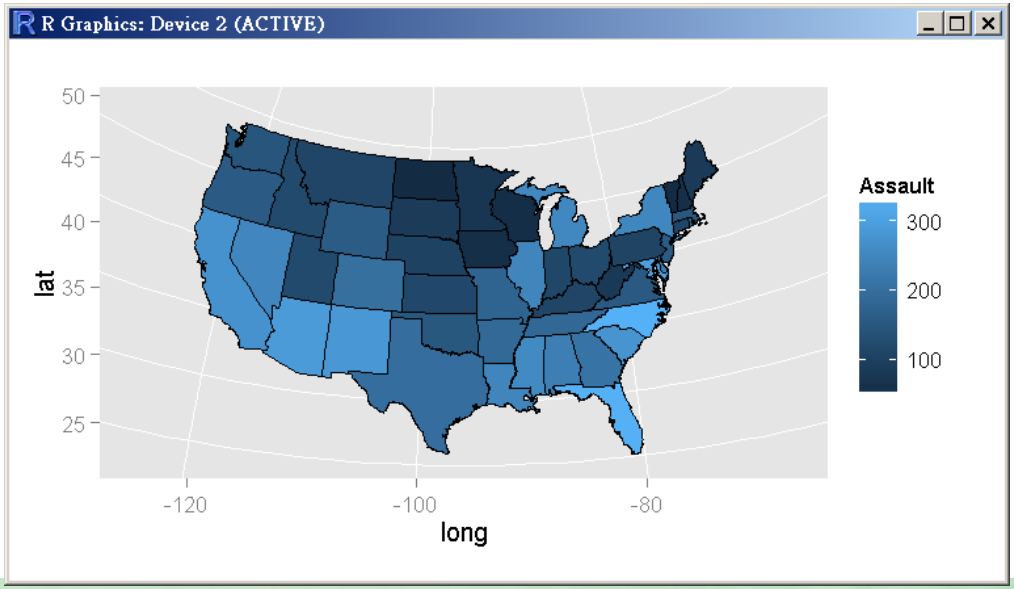
分級著色圖 (Choropleth Maps)

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)
> 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
```

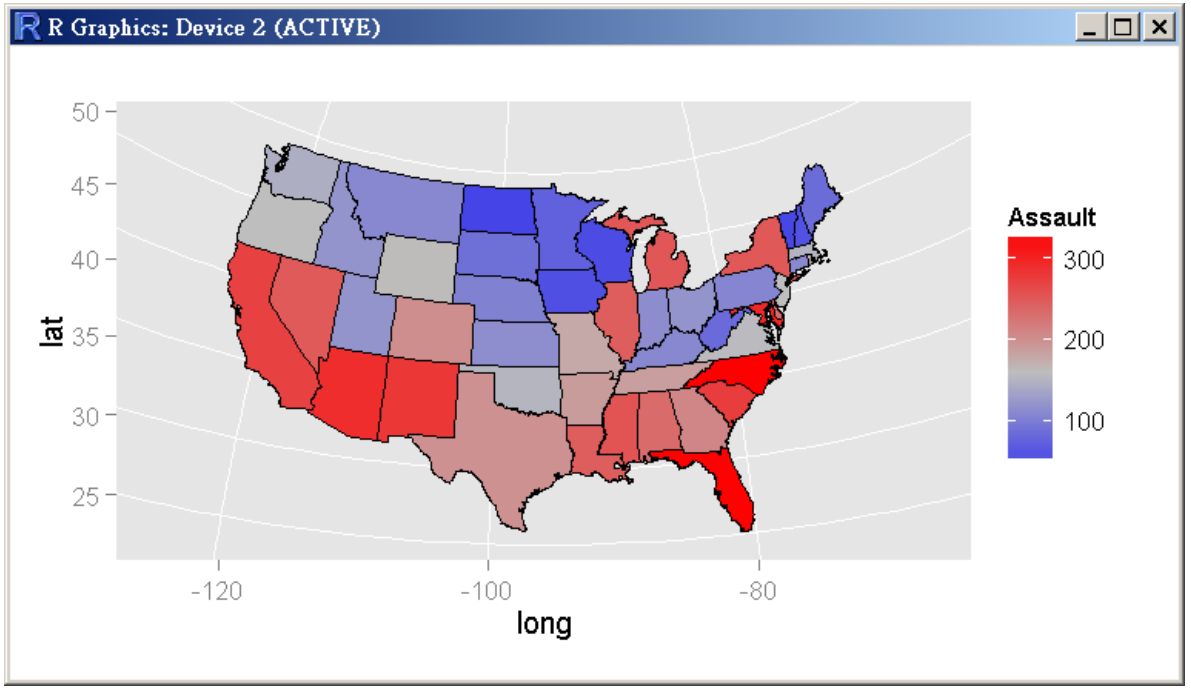
分級著色圖 (Choropleth Maps)

```
> # After merging, the order has changed, which would lead to polygons drawn in
> # the incorrect order. So, we sort the data.
> library(plyr) # For arrange() function
> # Sort by group, then order
> 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")
```



分級著色圖 (Choropleth Maps)

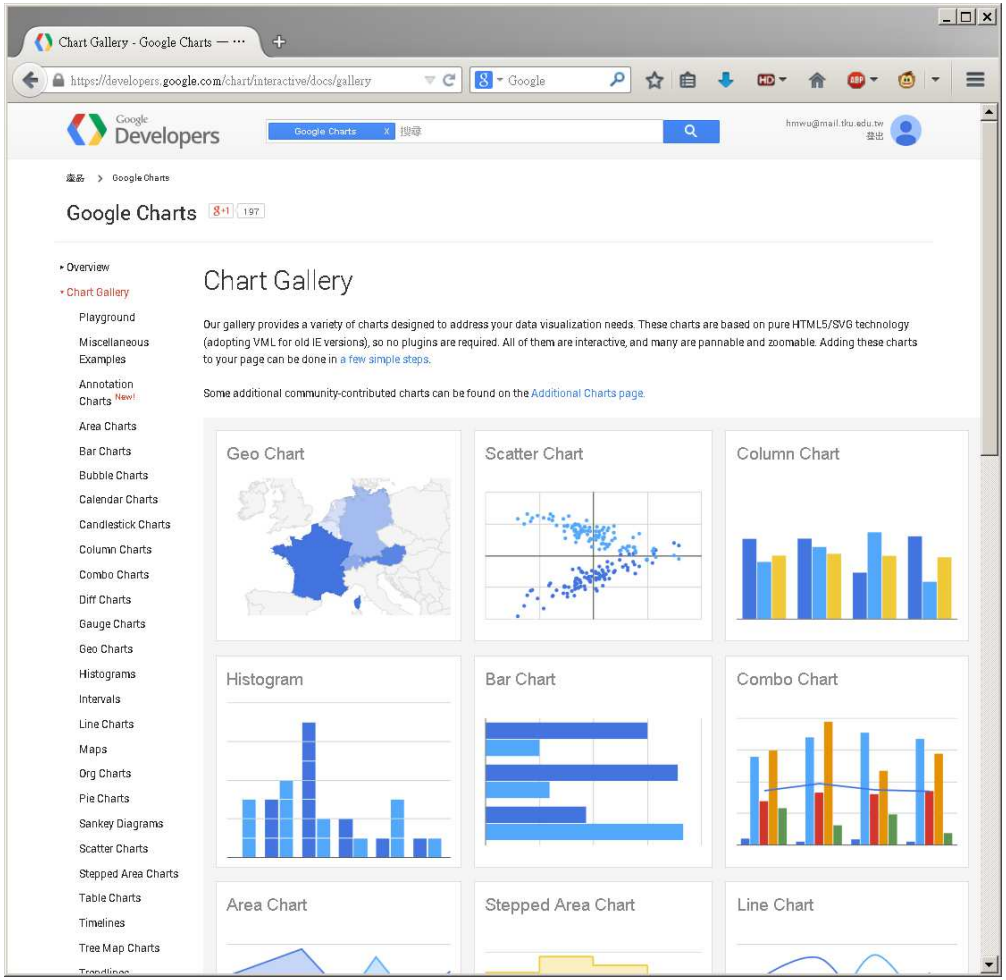
```
ggplot(crime.map, aes(x=long, y=lat, group=group, fill=Assault)) +
    geom_polygon(colour="black") +
    coord_map("polyconic") +
    scale_fill_gradient2(low="blue", mid="grey", high="red",
                         midpoint=median(crimes$Assault))
```



Package: googleVis

Interface between R and Google Charts

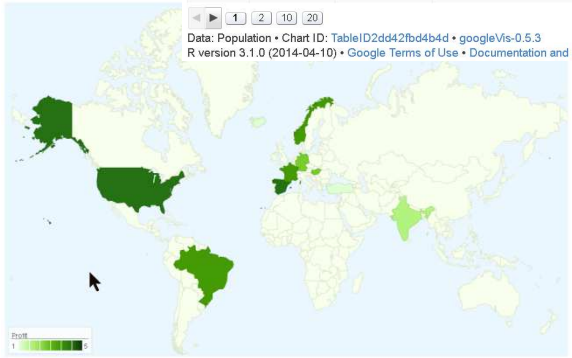
```
library(googleVis)
demo(googleVis)
```



Rank	Country	Population	% of World Population	Flag	Mode	Date
1	China	1,339,940,000	1950%		✓	2010年10月9日
2	India	1,188,650,000	1730%		✓	2010年10月9日
3	United States	310,438,000	452%		✓	2010年10月9日
4	Indonesia	237,556,363	346%		✓	2010年10月9日
5	Brazil	193,626,000	282%		✓	2010年10月9日
6	Pakistan	170,745,000	248%		✓	2010年10月9日

1 2 10 20

Data: Population • Chart ID: TableID2dd42fbd4b4d • googleVis-0.5.3
R version 3.1.0 (2014-04-10) • Google Terms of Use • Documentation and Data Policy



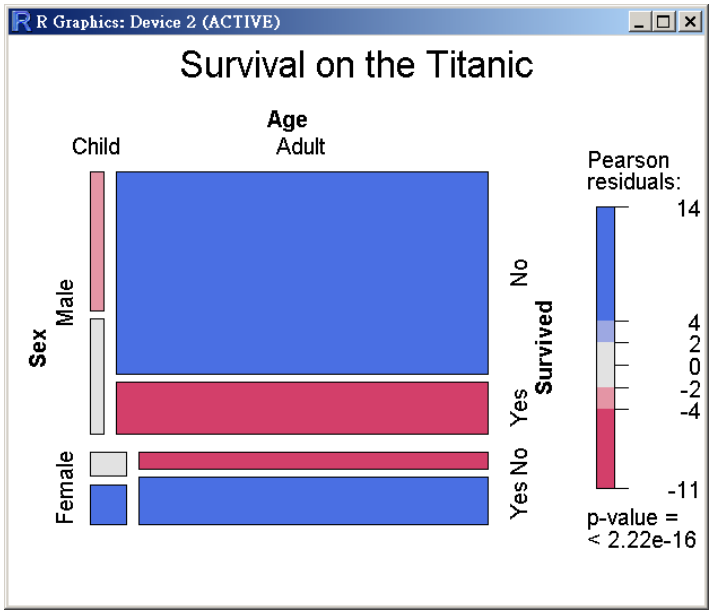
Data: Exports • Chart ID: GeoMapID2dd41d096c80 • googleVis-0.5.3
R version 3.1.0 (2014-04-10) • Google Terms of Use • Documentation and Data Policy

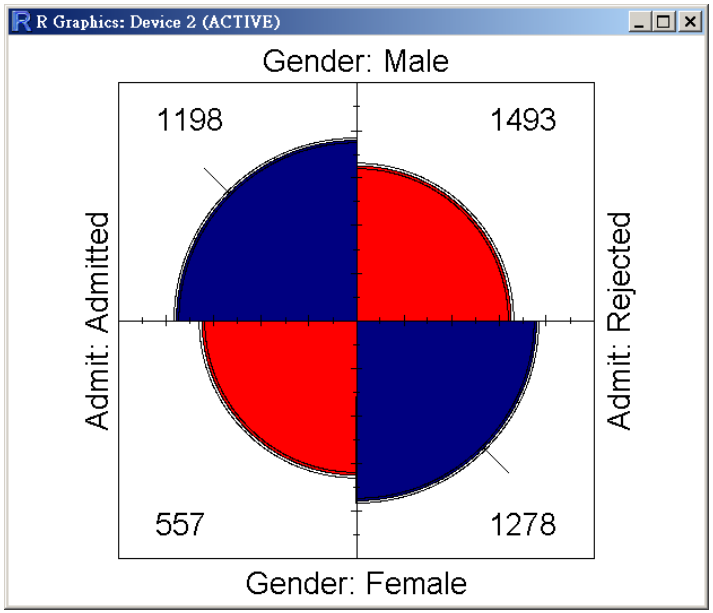
Data: Stock • Chart ID: AnnotatedTimeLineID2dd46d218a5 • googleVis-0.5.3
R version 3.1.0 (2014-04-10) • Google Terms of Use • Documentation and Data Policy

http://cran.r-project.org/web/packages/googleVis/vignettes/googleVis_examples.html

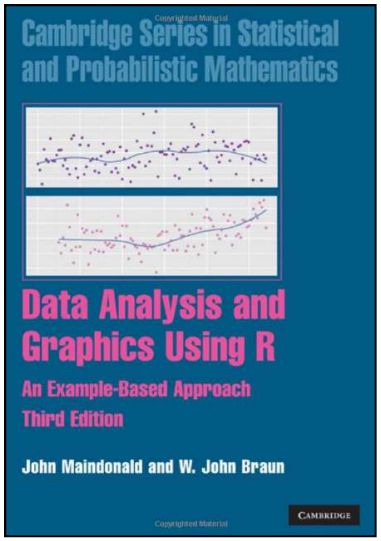
類別資料的視覺化: vcd

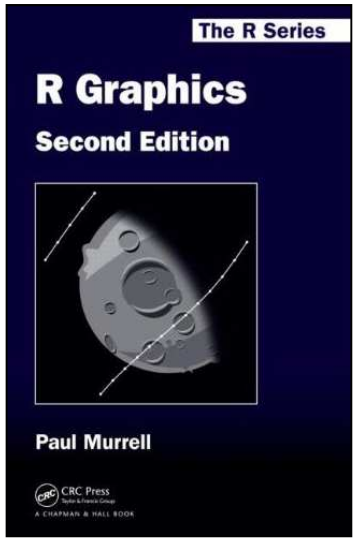
- **vcd**: Visualizing Categorical Data
- Visualizing Categorical Data with SAS and R: http://www.datavis.ca/courses/VCD/
- Visualizing Categorical Data: http://www.datavis.ca/books/vcd/

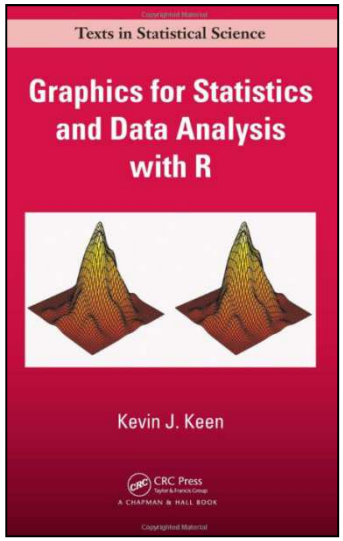


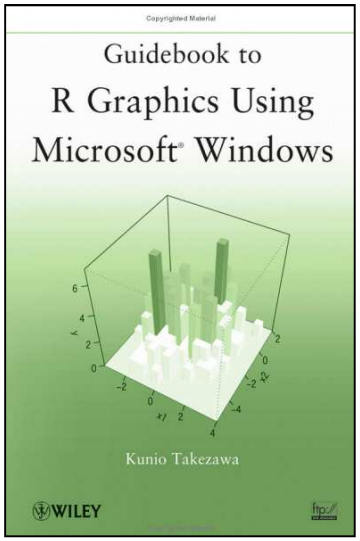


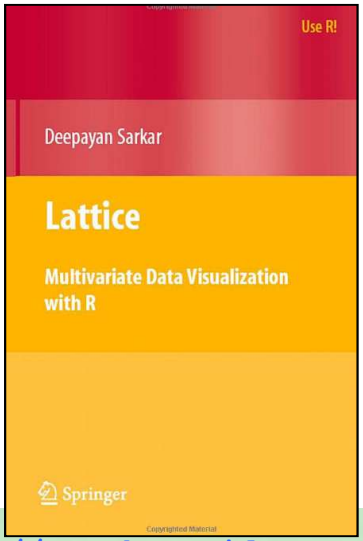
R圖形參考書目

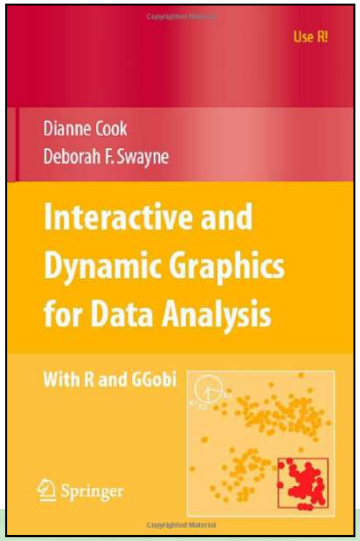

Cambridge Series in Statistical and Probabilistic Mathematics
Data Analysis and Graphics Using R
An Example-Based Approach
Third Edition
John Maindonald and W. John Braun
CAMBRIDGE

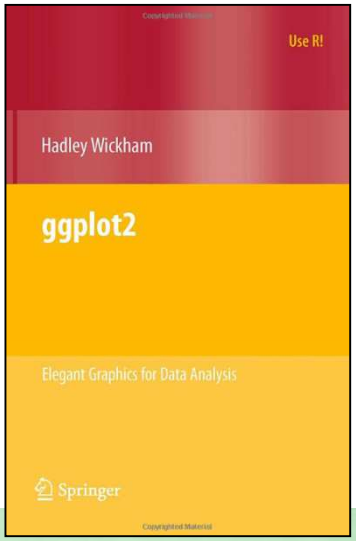

The R Series
R Graphics
Second Edition
Paul Murrell
CRC Press
A CHAPMAN & HALL BOOK

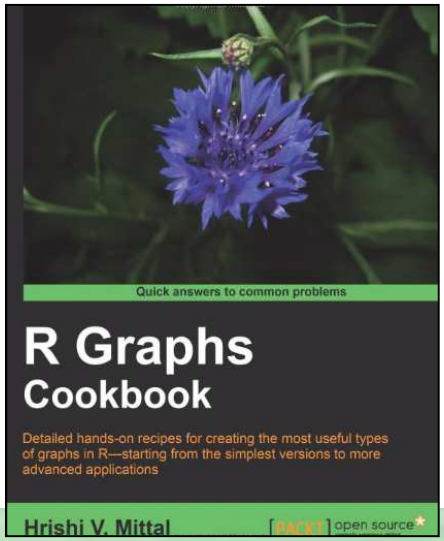

Texts in Statistical Science
Graphics for Statistics and Data Analysis with R
Kevin J. Keen
CRC Press
A CHAPMAN & HALL BOOK


Guidebook to
R Graphics Using
Microsoft® Windows
Kunio Takezawa
WILEY

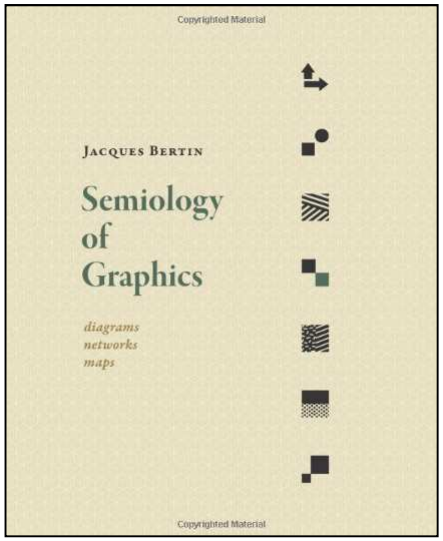

Use R!
Deepayan Sarkar
Lattice
Multivariate Data Visualization with R
Springer

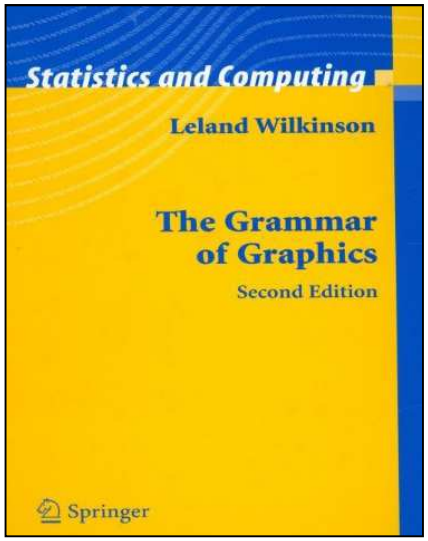

Use R!
Dianne Cook
Deborah F. Swayne
Interactive and Dynamic Graphics for Data Analysis
With R and GGobi
Springer

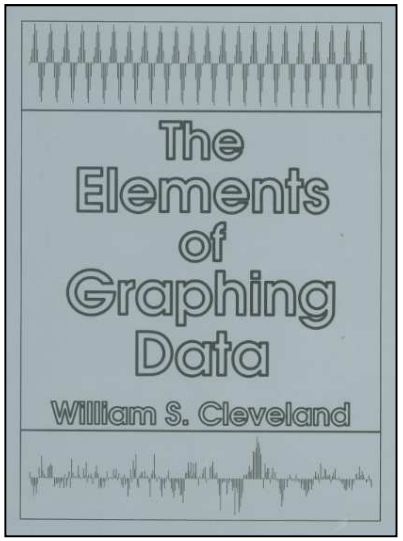

Use R!
Hadley Wickham
ggplot2
Elegant Graphics for Data Analysis
Springer

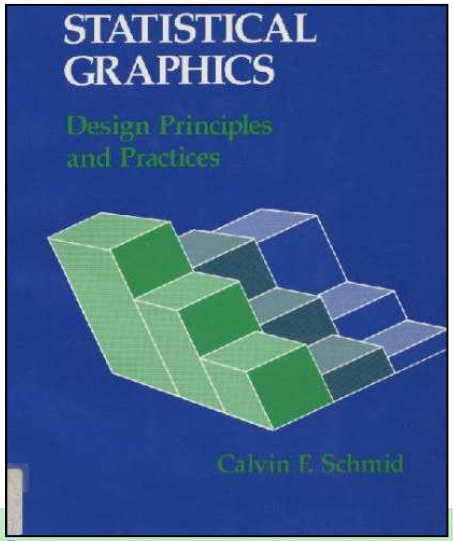

Quick answers to common problems
R Graphs Cookbook
Detailed hands-on recipes for creating the most useful types of graphs in R—starting from the simplest versions to more advanced applications
Hrishi V. Mittal
PACKT open source

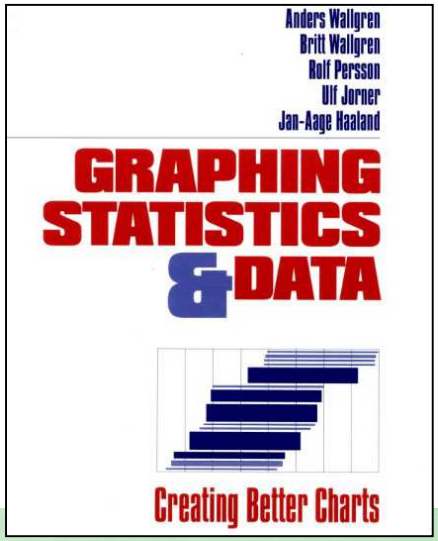
Fundations, Principle



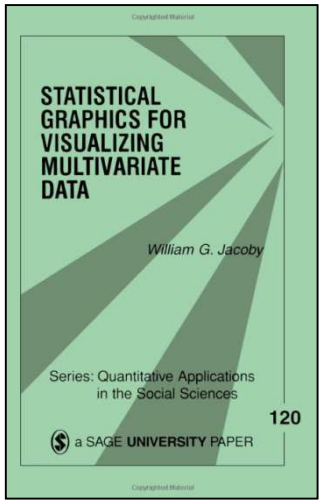


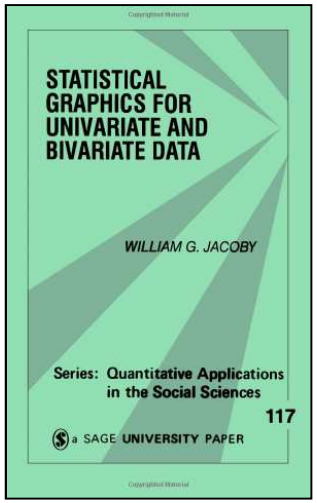


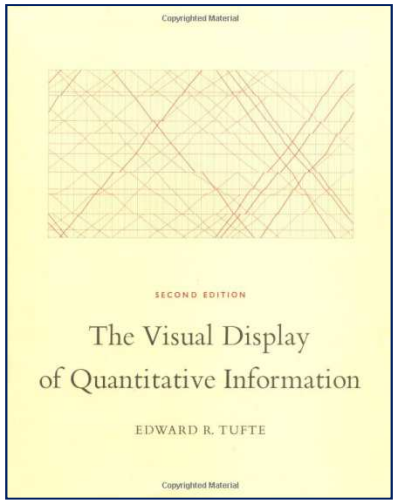


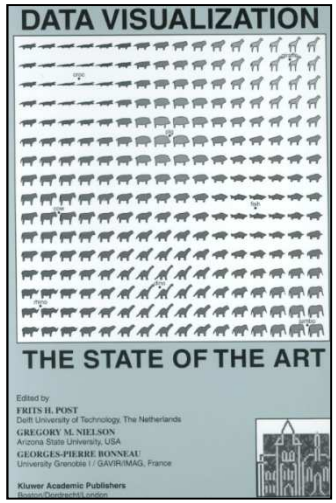


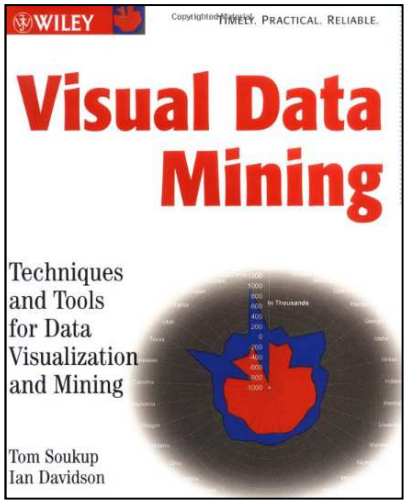
Statistics and Graphics

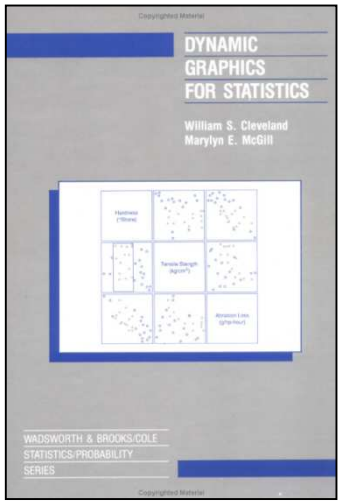


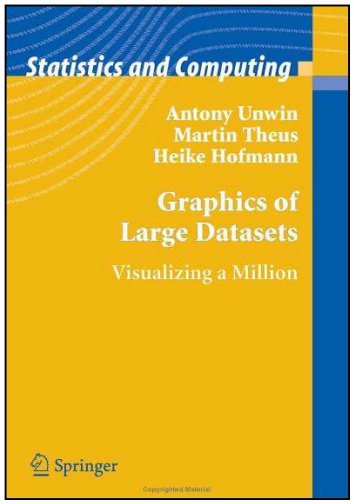


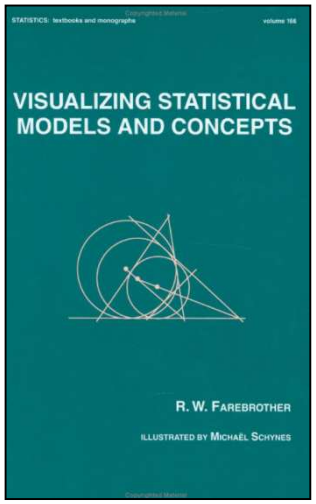


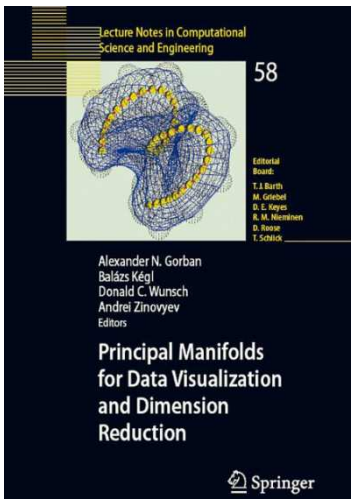


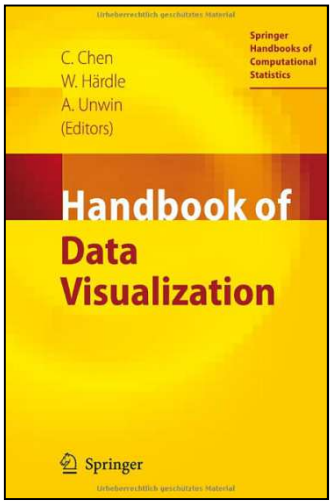




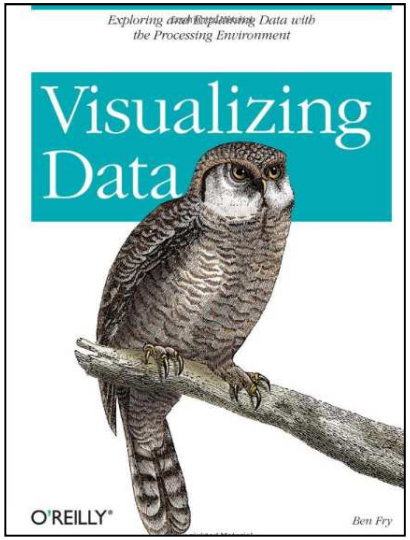


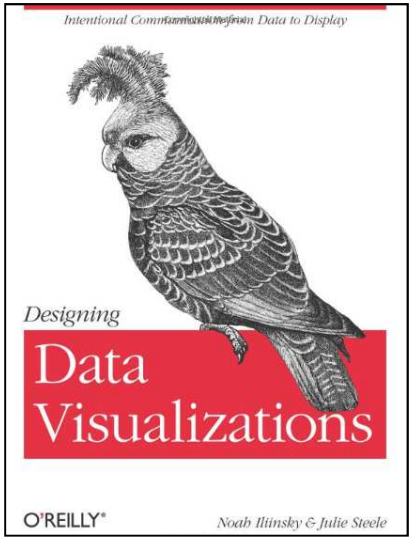


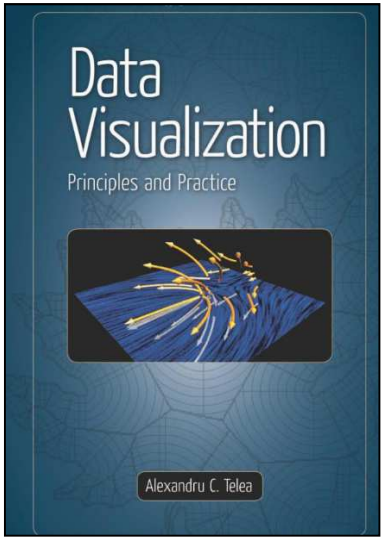


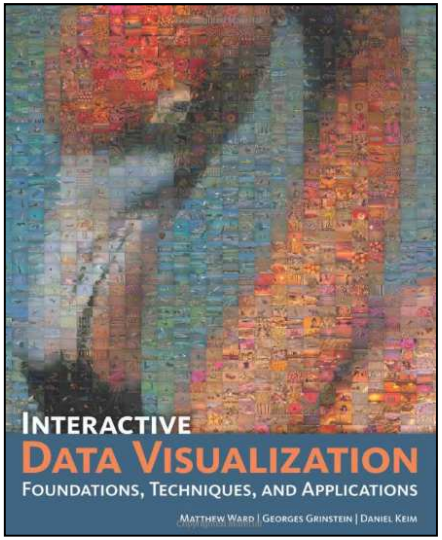


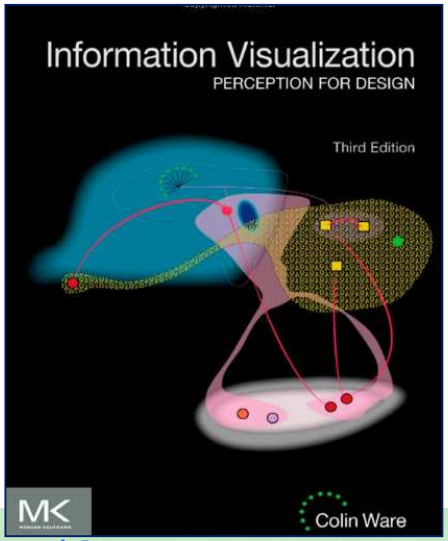
Data Visualization

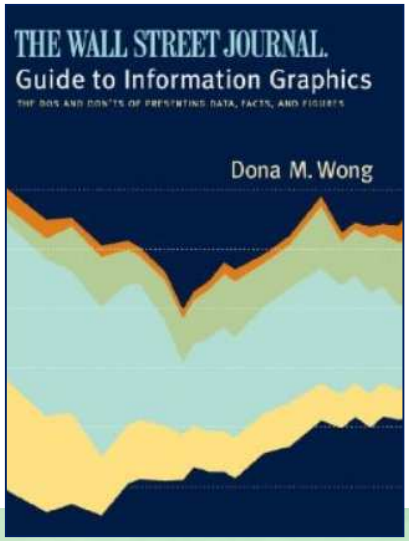


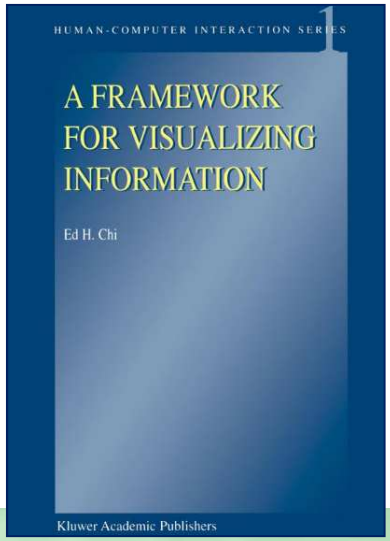




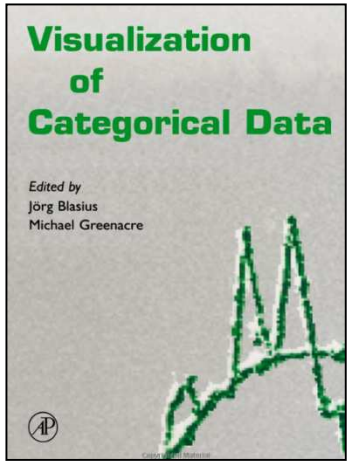


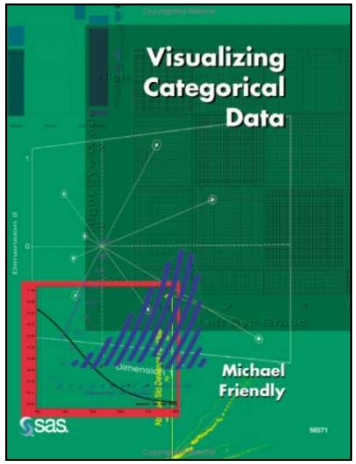


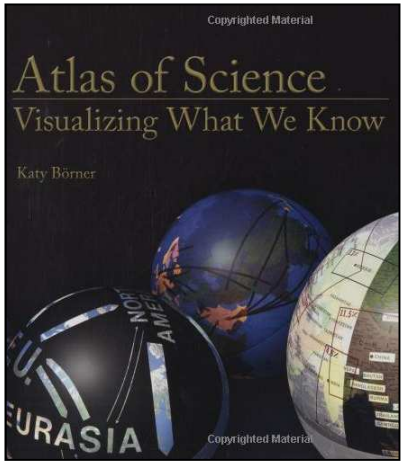


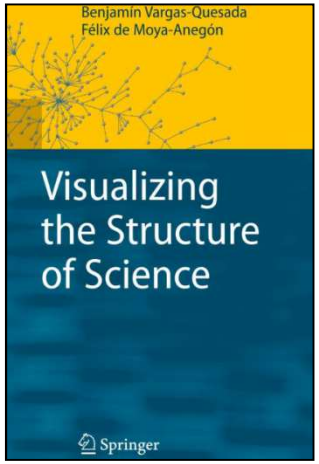


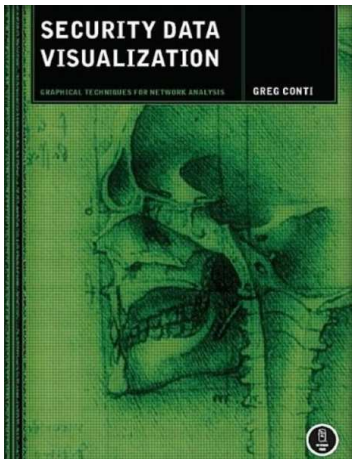
Specified Data Types

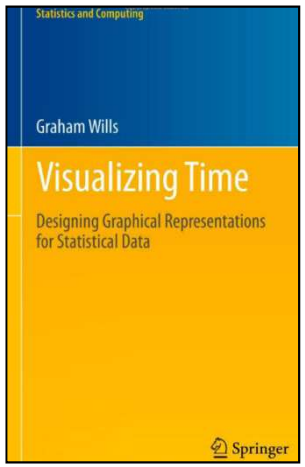


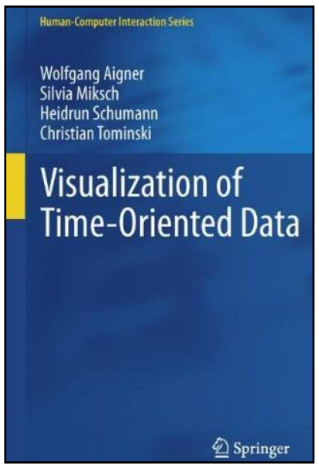


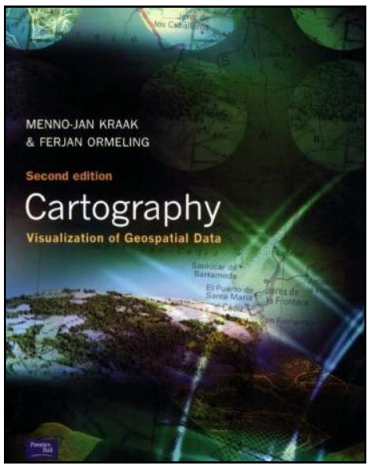


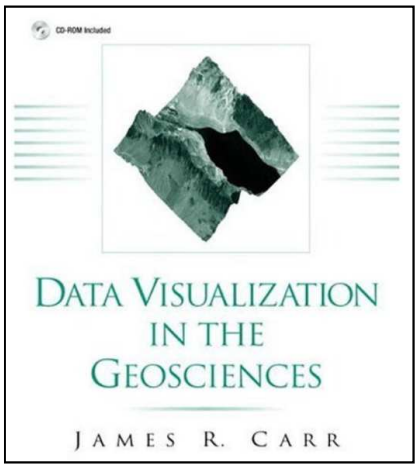


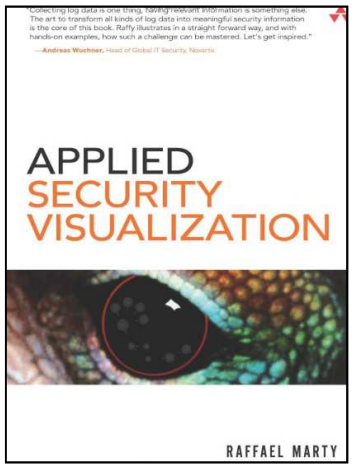




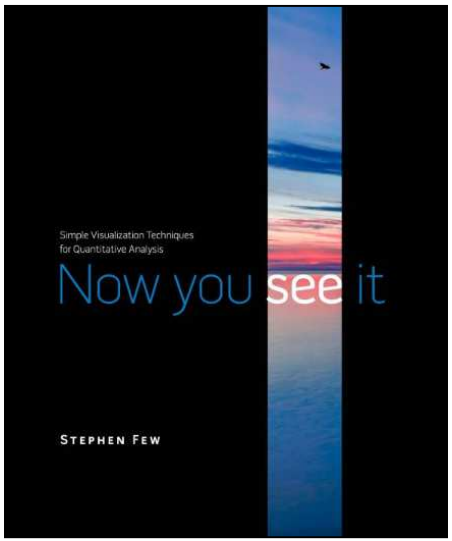


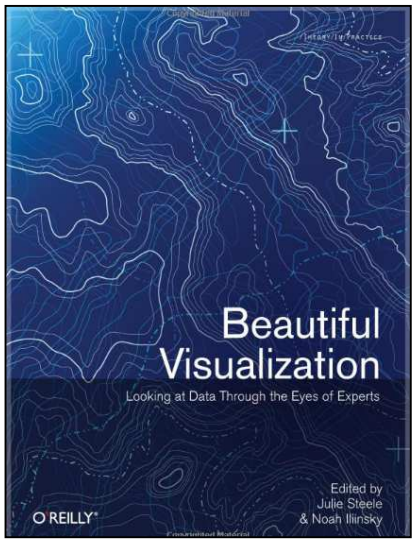


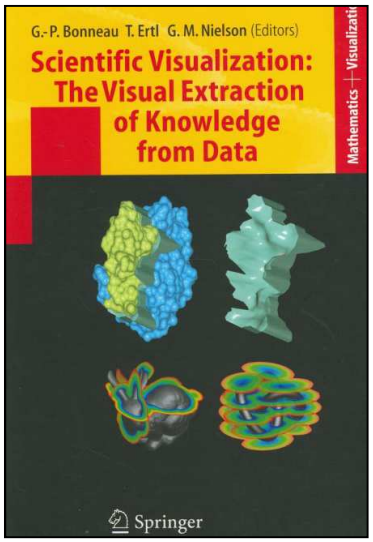


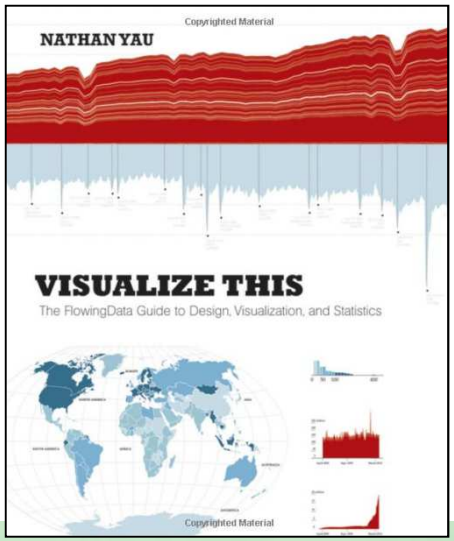


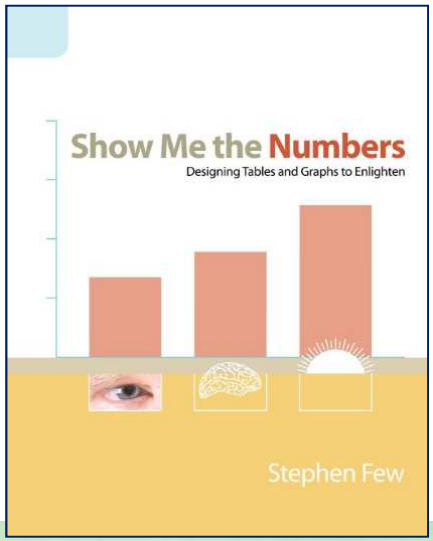
Others

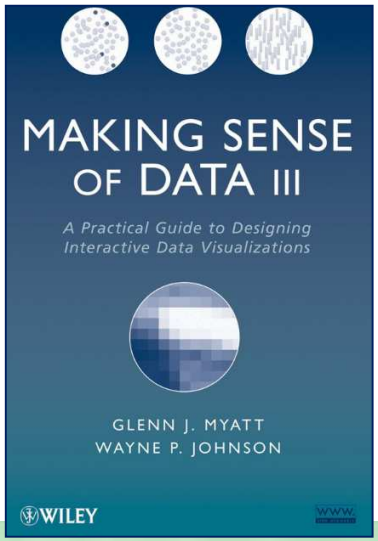

Simple Visualization Techniques
for Quantitative Analysis
Now you see it
Stephen Few


Beautiful
Visualization
Looking at Data Through the Eyes of Experts
O'REILLY
Edited by
Julie Steele
& Noah Iliinsky


G.-P. Bonneau T. Ertl G. M. Nielson (Editors)
Scientific Visualization:
The Visual Extraction
of Knowledge
from Data
Springer


Copyrighted Material
NATHAN YAU
VISUALIZE THIS
The FlowingData Guide to Design, Visualization, and Statistics
Copyrighted Material


Show Me the Numbers
Designing Tables and Graphs to Enlighten
Stephen Few


MAKING SENSE
OF DATA III
A Practical Guide to Designing
Interactive Data Visualizations
GLENN J. MYATT
WAYNE P. JOHNSON
WILEY