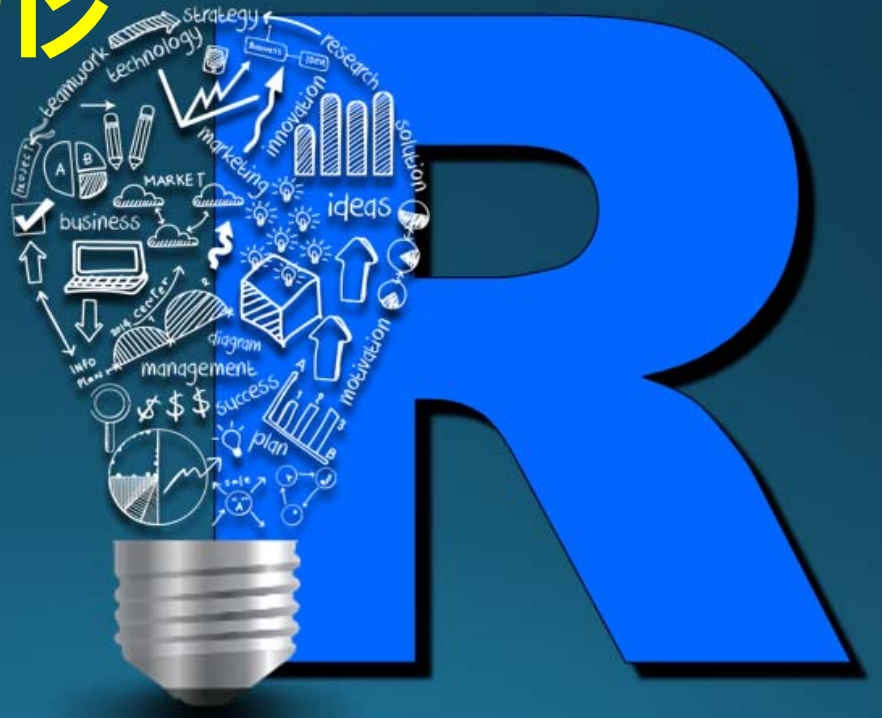
A05

R資料視覺化
基礎統計圖形



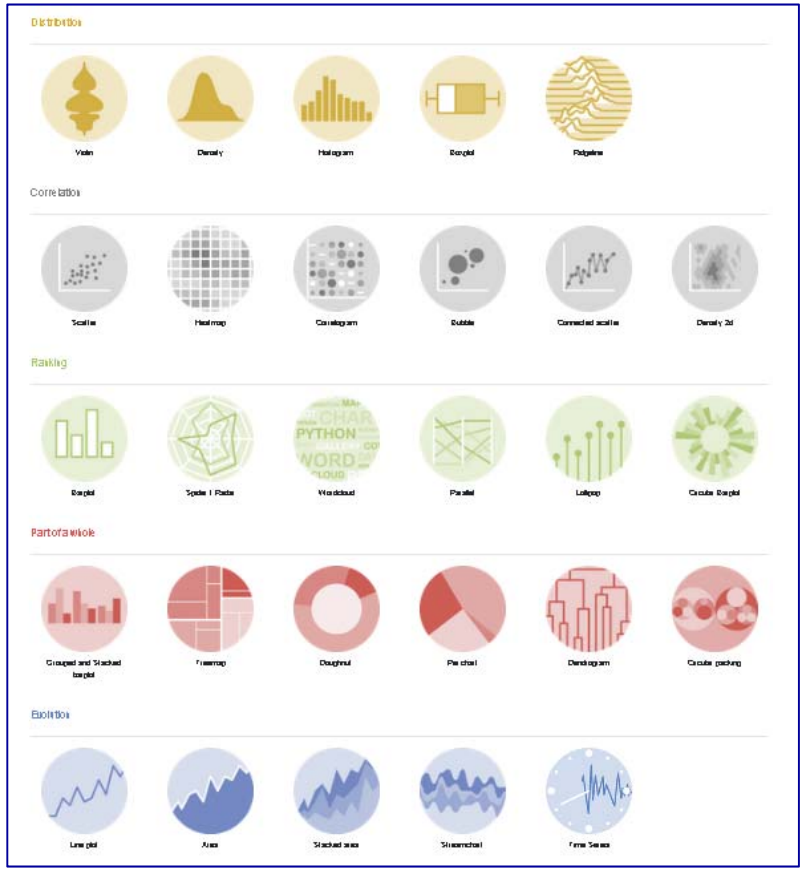
吳漢銘
國立政治大學 統計學系

http://www.hmwu.idv.tw

本章大綱與學習目標

- **簡介:** Why Data Visualization? 圖形輸出、向量圖與非向量圖格式
- **客製化圖形:** R圖形座標系統、圖標題和範圍、abline、lines、arrows、segments一頁多張圖形、顏色、文字標號、Data Symbols、圖例說明、座標軸、讀取外部影像檔案
- **基本統計圖形:** 索引圖、直方圖、Density Plots 、Quantile Plots、圓餅圖、散佈圖、curve、盒形圖、長條圖、散佈圖矩陣 、時間序列、 多邊形、Radar Plot、 3D透視圖、3D散佈圖、rgl、影像。
- **其它:** pheatmap、networkD3、Sankey Diagram、Word Cloud

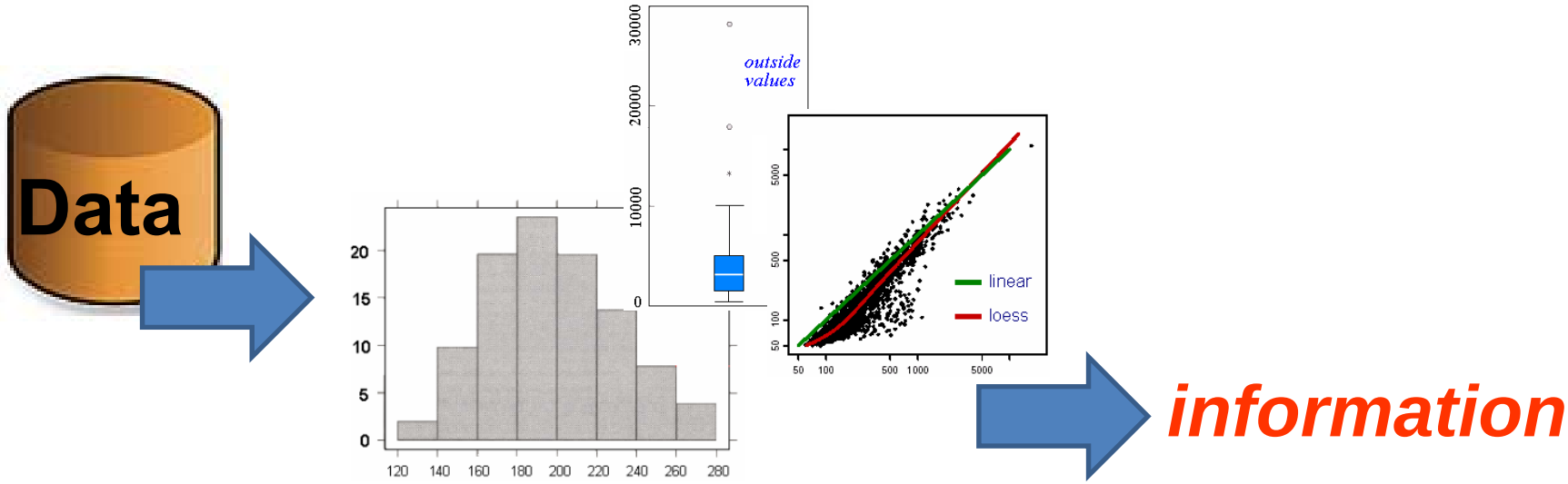
The R Graph Gallery
https://r-graph-gallery.com/



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.

Why Data Visualization?

- It is not about "**infographics**", the beautiful, heavily customized products of expert graphic designers.
- Data visualization can provide clear understanding of patterns in data, detect hidden structures in data, condense information.
- **Anscombe's quartet** comprises four datasets. They were constructed in 1973 by the statistician Francis Anscombe to demonstrate both the importance of graphing data before analyzing it and the effect of outliers on statistical properties.
- Four datasets have nearly identical simple statistical properties, yet appear very different when graphed.

	I		II		III		IV	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	10	8.04	10	9.14	10	7.46	8	6.58
2	8	6.95	8	8.14	8	6.77	8	5.76
3	13	7.58	13	8.74	13	12.74	8	7.71
4	9	8.81	9	8.77	9	7.11	8	8.84
5	11	8.33	11	9.26	11	7.81	8	8.47
6	14	9.96	14	8.1	14	8.84	8	7.04
7	6	7.24	6	6.13	6	6.08	8	5.25
8	4	4.26	4	3.1	4	5.39	19	12.5
9	12	10.84	12	9.13	12	8.15	8	5.56
10	7	4.82	7	7.26	7	6.42	8	7.91
11	5	5.68	5	4.74	5	5.73	8	6.89

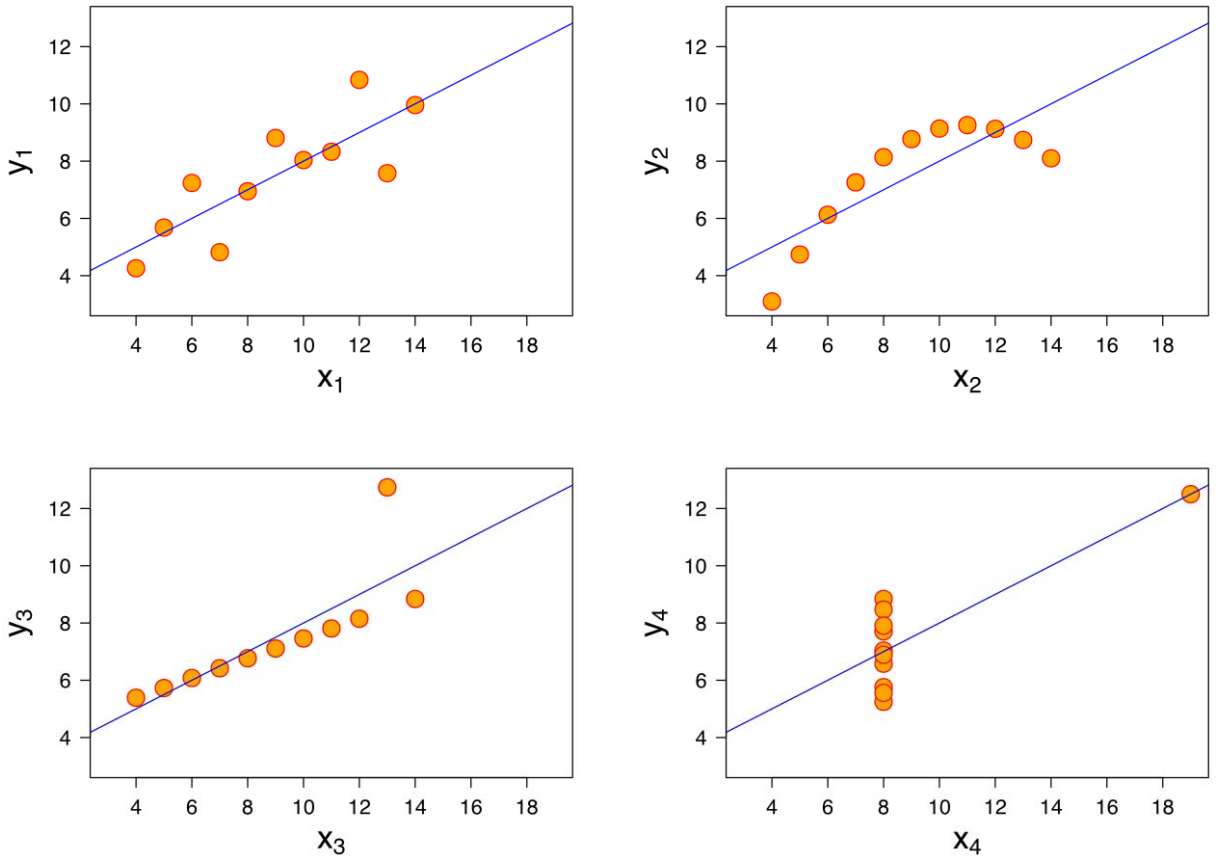
Mean of x in each case: 9 (exact)
Sample variance of x in each case: 11 (exact)
Mean of y in each case: 7.50 (to 2 decimal places)
Sample variance of y in each case: 4.122 or 4.127 (to 3 decimal places)
Correlation between x and y in each case: 0.816 (to 3 decimal places)
Linear regression line in each case: y = 3.00 + 0.500x (to 2 and 3 decimal places, respectively)

https://en.wikipedia.org/wiki/Anscombe's_quartet

http://ryanwomack.com/IASSIST/DataViz/

Anscombe's Quartet

- **Mean of x** in each case: 9 (exact)
- **Sample variance of x** in each case: 11 (exact)
- **Mean of y** in each case: 7.50 (to 2 decimal places)
- **Sample variance of y** in each case: 4.122 or 4.127 (to 3 decimal places)
- **Correlation** between x and y in each case: 0.816 (to 3 decimal places)
- **Linear regression line** in each case: y = 3.00 + 0.500x (to 2 and 3 decimal places, respectively)



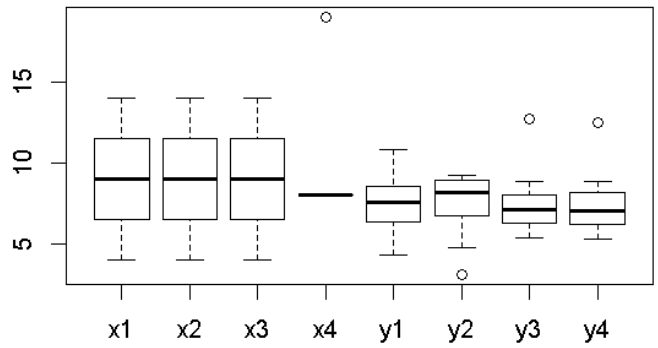
anscombe {datasets}

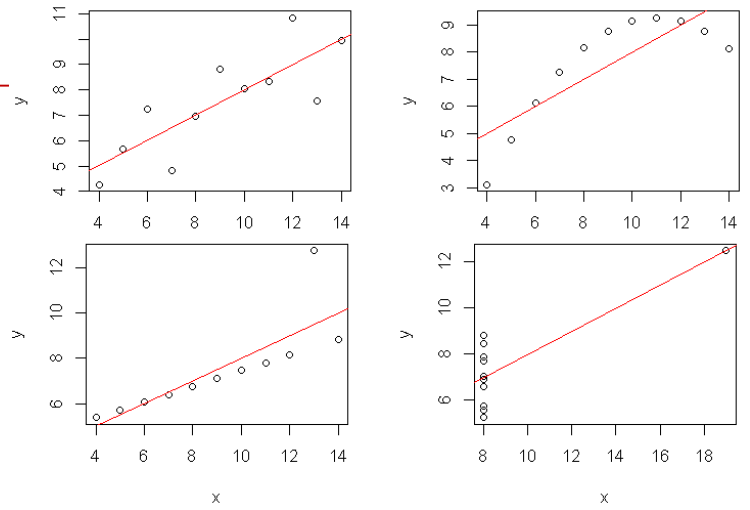
Anscombe's Quartet of 'Identical' Simple Linear Regressions

```
> head(anscombe, 3)
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
> apply(anscombe, 2, mean)
      x1       x2       x3       x4       y1       y2       y3       y4
9.000000 9.000000 9.000000 9.000000 7.500909 7.500909 7.500000 7.500909
> apply(anscombe, 2, sd)
      x1       x2       x3       x4       y1       y2       y3       y4
3.316625 3.316625 3.316625 3.316625 2.031568 2.031657 2.030424 2.030579
> mapply(cor, anscombe[,1:4], anscombe[,5:8])
       x1        x2        x3        x4
0.8164205 0.8162365 0.8162867 0.8165214
> mapply(function(x, y) lm(y~x)$coefficients, anscombe[, 1:4], anscombe[, 5:8])
                 x1       x2        x3        x4
(Intercept) 3.0000909 3.000909 3.0024545 3.0017273
x           0.5000909 0.500000 0.4997273 0.4999091
```

```
par(mfrow=c(2, 2))
regplot <- function(x, y){
  plot(y~x)
  abline(lm(y~x), col="red")
}
mapply(regplot, anscombe[, 1:4], anscombe[, 5:8])
```

boxplot(anscombe)





圖形輸出

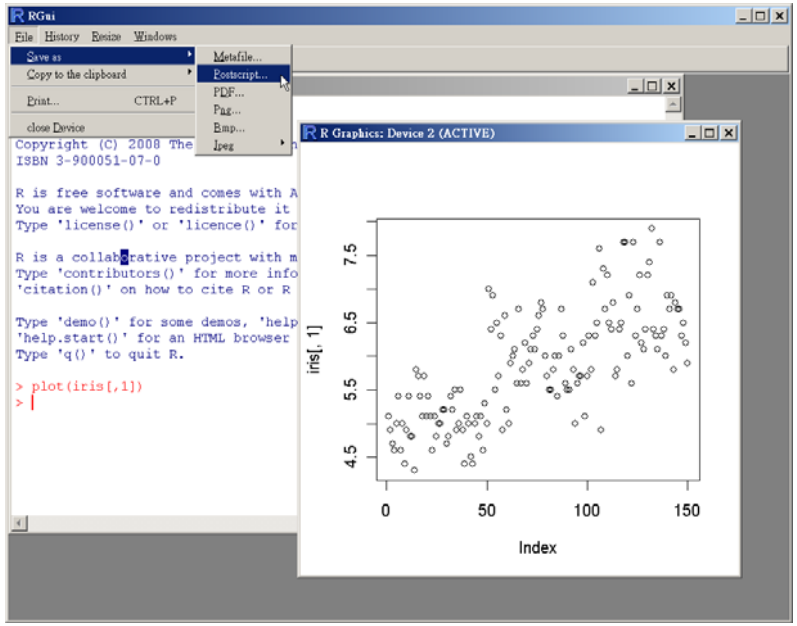
Read in Data → Plot → Save → Edit → Finished

Example

RGui

方法1: Save as ...

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



注意圖形的比例及大小

方法2 : Activate device

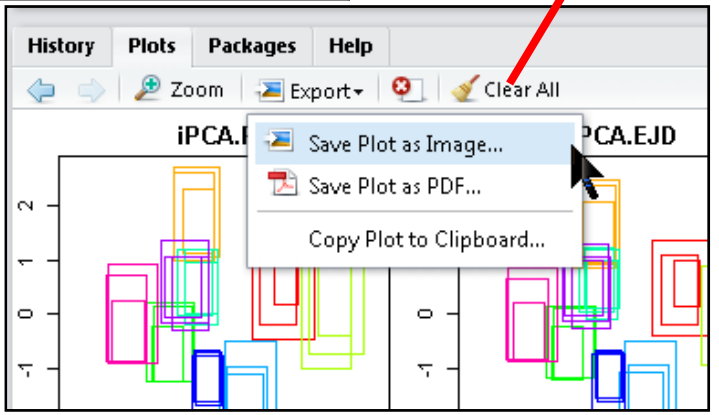
```
> postscript(file="iris2.ps")
> plot(iris[,2])
> dev.off()
```

or

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

```
# Graphics Device:
bmp, jpeg, png, tiff, pdf
```

RStudio



向量圖與非向量圖格式

存檔格式及目的

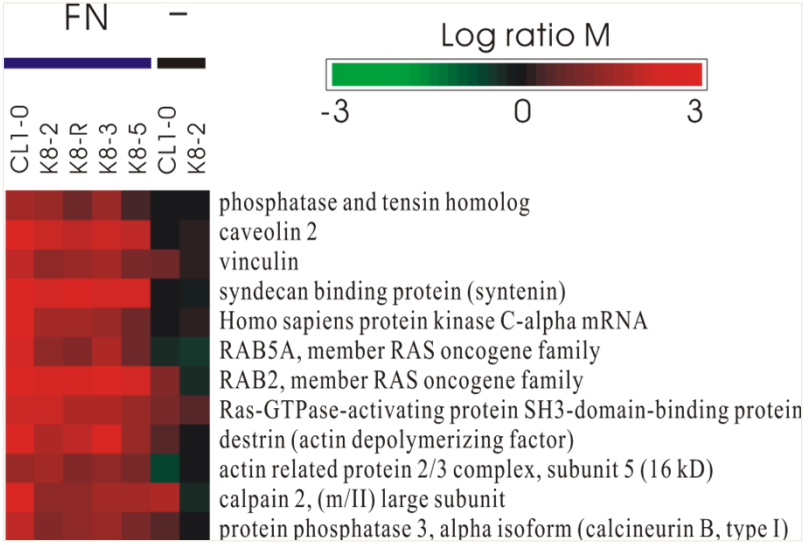
- 點陣圖檔 (壓縮失真: PNG, JPEG, TIFF, BMP) : web page, powerpoint, word,...
- 向量圖檔 (物件不失真: Metafile, SVG, EPS) : Publication, LaTeX.

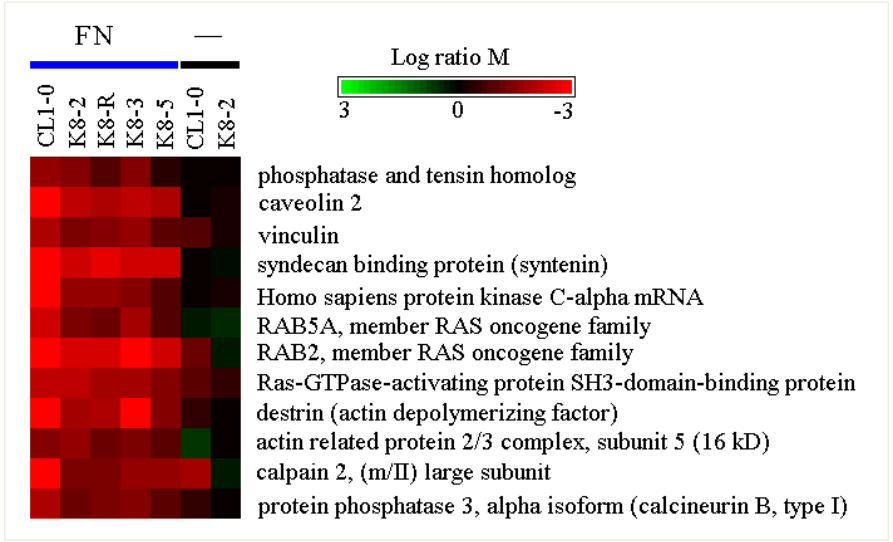
向量圖(在此為wmf格式):

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。





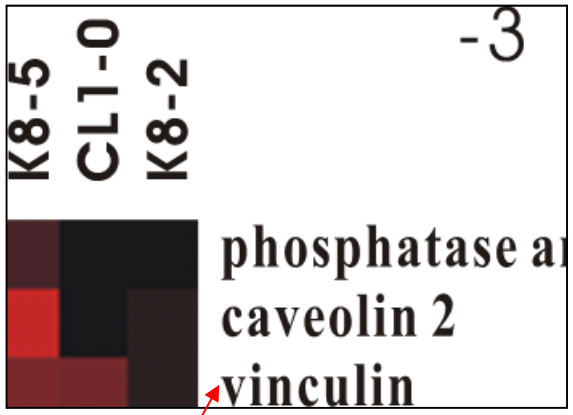
向量圖與非向量圖格式

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

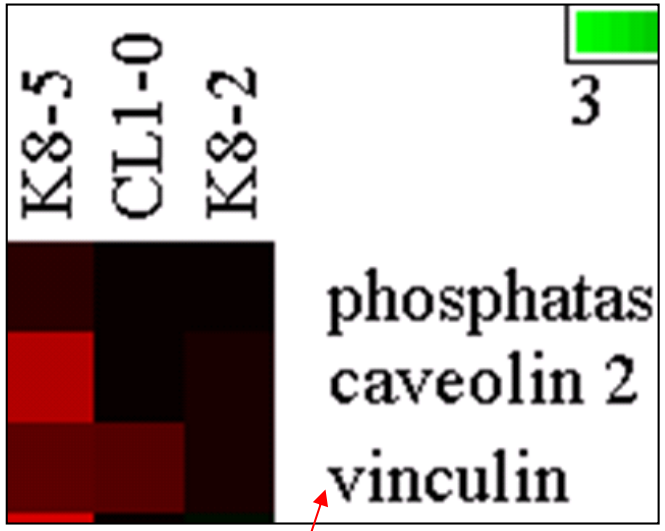
2. 這跟放大投影片(或全營幕)成300%是不一樣的。

向量圖放大變這樣:



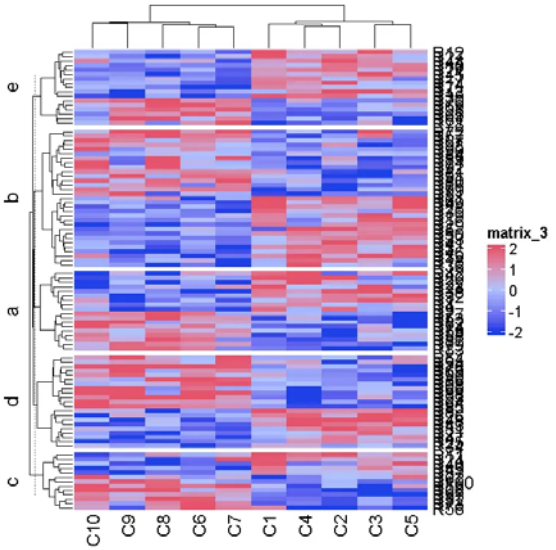
無鋸齒狀, 邊界會糊糊的(或霧霧的是軟體自動平滑化的結果)

點陣圖放大變這樣:



有鋸齒狀

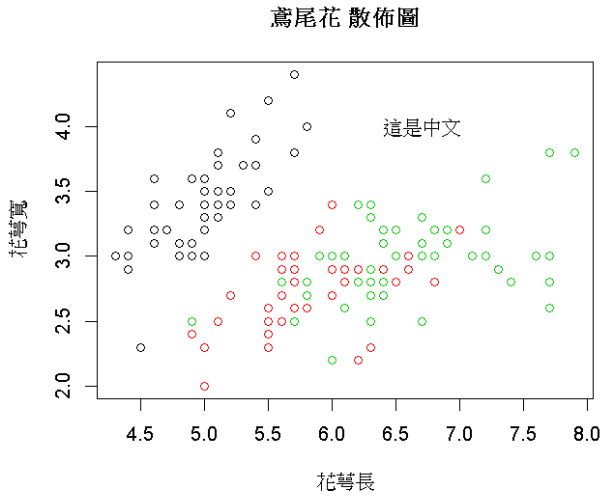
R畫的圖很醜?

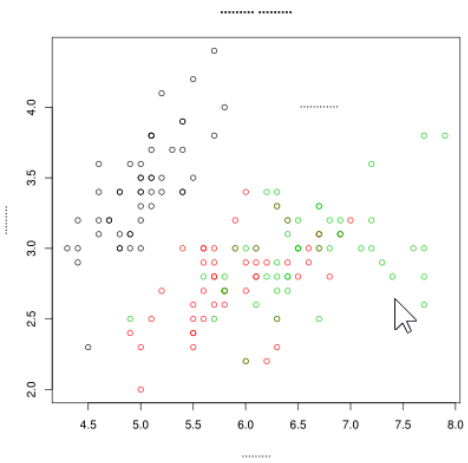


圖形存檔及輸出中文問題

選讀

```
> pdf("iris.pdf") # jpeg, bmp OK
> plot(iris[, 1:2], xlab="花萼長", ylab="花萼寬", main="鳶尾花 散佈圖", col=iris[,5])
> text(6.7, 4, "這是中文")
> dev.off()
```



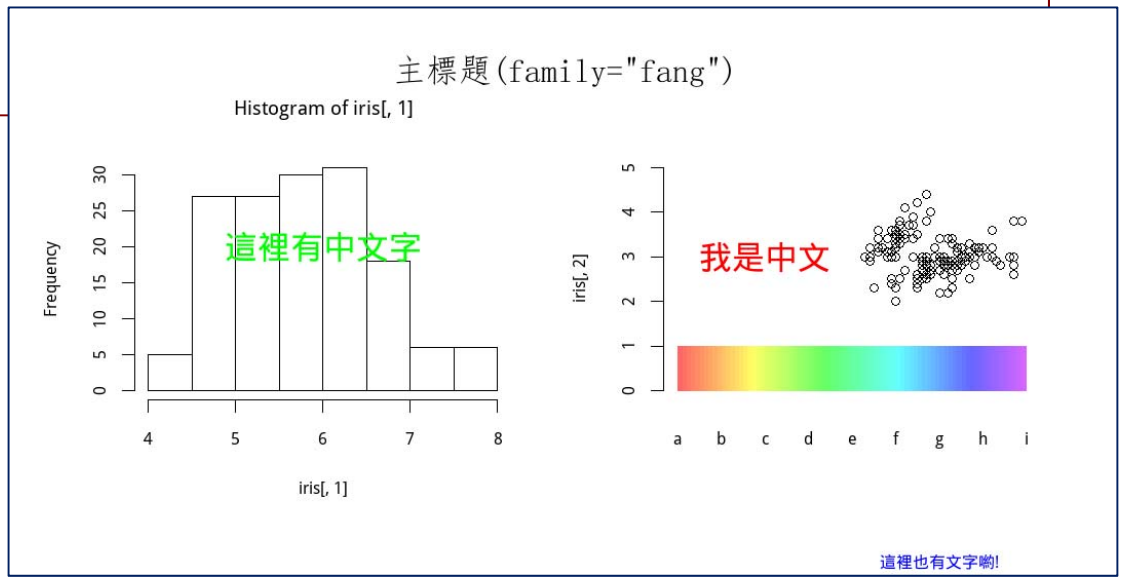



```
> #showtext: Using Fonts More Easily in R Graphs
> library(showtext)
> font_add("msjh", "msjh.ttc") ## 微軟正黑體
> showtext_auto(enable=TRUE)
```

圖形存檔及輸出中文問題

選讀

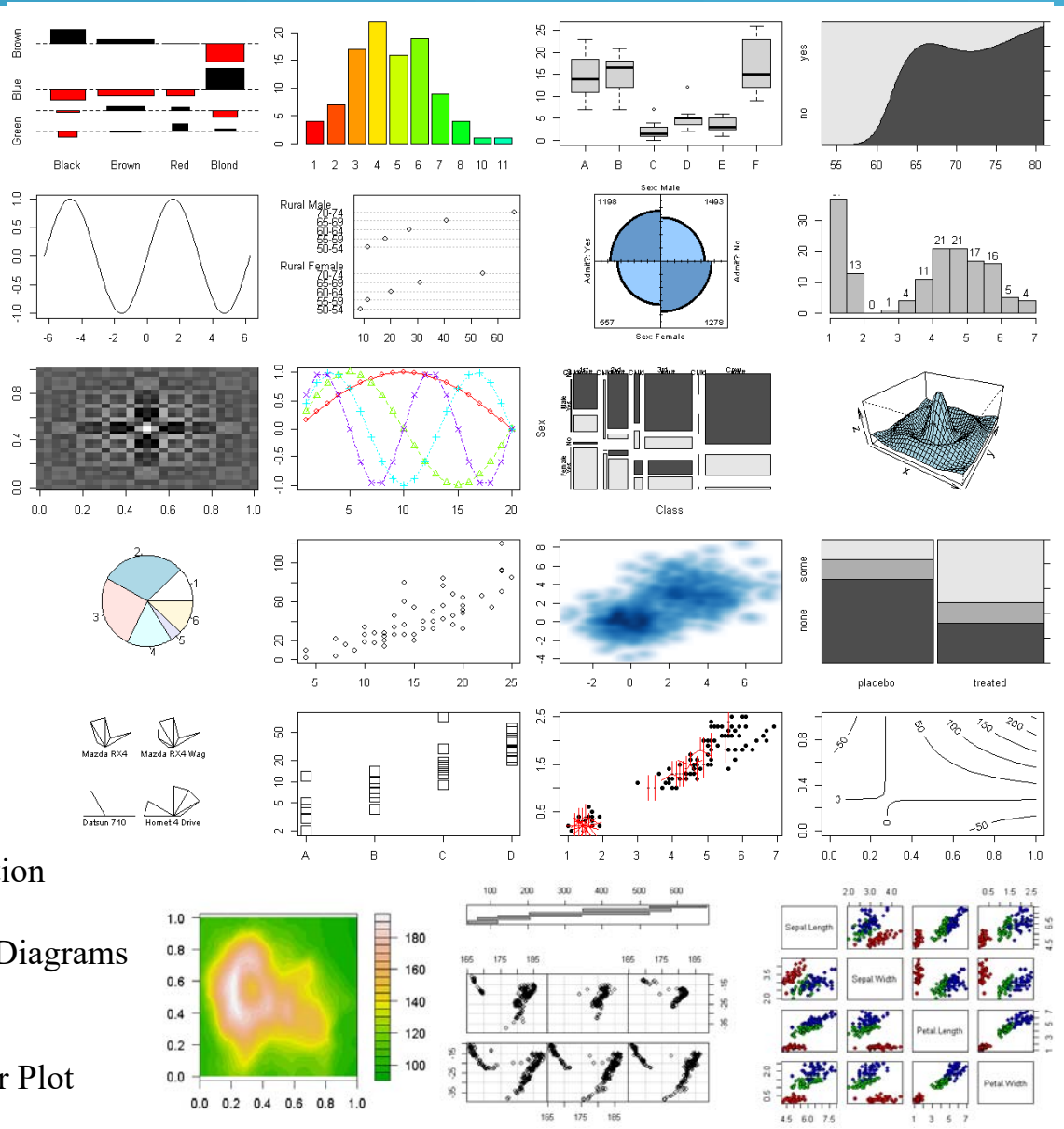
```
> pdf("test.pdf", width=12, height=8)
> # ?bmp (jpeg, png, tiff)
> par(family="msjh") ## 微軟正黑體
> # outer: 兩個圖對中間
> # line=-2 與圖形之空格
> par(mfrow=c(1,2), oma=c(2, 1, 2, 1))
> hist(iris[, 1])
> text(6, 20, "這裡有中文字", col="green", cex=2)
> plot(iris[, 1], iris[, 2], xaxt="n", bty="n", xlab="", xlim=c(0, 8), ylim=c(0, 5))
> axis(1, at=0:8 , labels=letters[1:9], tick=FALSE)
> text(2, 3, "我是中文", col="red", cex=2)
> title('主標題(family="fang")', family="fang", outer=TRUE,  line=-1, cex.main=2)
> mtext("這裡也有文字喲!", side=1, line=6, at=6, col="blue")
> mycolor <- rainbow(100, alpha=0.6)[1:80]
> rasterImage(t(mycolor), 0, 0, 8, 1, interpolate=FALSE)
> dev.off()
windows
      2
```



graphics: The R Graphics Package

Plots:

assocplot: Association Plots
barplot: Bar Plots
boxplot: Box Plots
cdplot: Conditional Density Plots
contour: Display Contours
coplot: Conditioning Plots
curve: Draw Function Plots
dotchart: Cleveland's Dot Plots
filled.contour: Level (Contour) Plots
fourfoldplot: Fourfold Plots
hist: Histograms
image: Display a Color Image
matplot: Plot Columns of Matrices
mosaicplot: Mosaic Plots
pairs: Scatterplot Matrices
persp: Perspective Plots
pie: Pie Charts
plot: Generic X-Y Plotting
smoothScatter: Scatterplots with Smoothed Densities Color Representation
spineplot: Spine Plots and Spinograms
stars: Star (Spider/Radar) Plots and Segment Diagrams
stem: Stem-and-Leaf Plots
stripchart: 1-D Scatter Plots
sunflowerplot: Produce a Sunflower Scatter Plot



graphics: The R Graphics Package

Decoration:

abline: Add Straight Lines to a Plot
arrows: Add Arrows to a Plot
axis.POSIXct: Date and Date-time Plotting Functions
axis: Add an Axis to a Plot
box: Draw a Box around a Plot
grid: Add Grid to a Plot
legend Add Legends to Plots
lines: Add Connected Line Segments to a Plot
matlines: Plot Columns of Matrices
matpoints: Plot Columns of Matrices
mtext: Write Text into the Margins of a Plot
panel.smooth: Simple Panel Plot
points: Add Points to a Plot
polygon: Polygon Drawing
polypath: Path Drawing
rasterImage: Draw One or More Raster Images
rect: Draw One or More Rectangles
rug: Add a Rug to a Plot
segments: Add Line Segments to a Plot
symbols: Draw Symbols (Circles, Squares, Stars, Thermometers, Boxplots)
text: Add Text to a Plot
title: Plot Annotation
xspline: Draw an X-spline

Utilities:

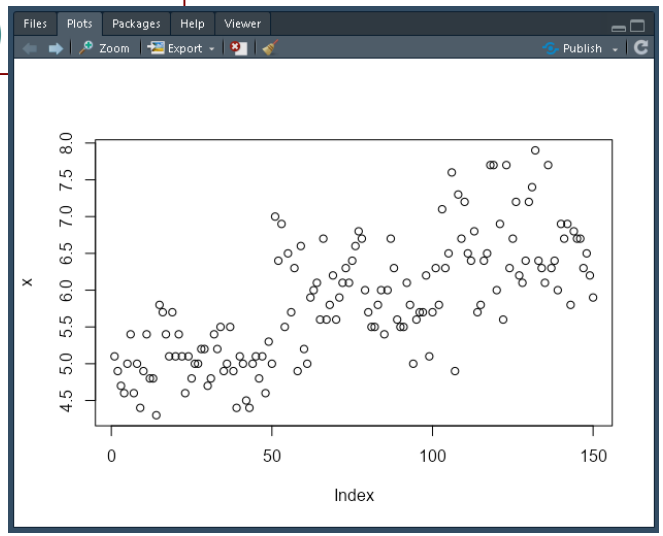
axTicks: Compute Axis Tickmark Locations
close.screen: Creating and Controlling Multiple Screens on a Single Device
erase.screen: Creating and Controlling Multiple Screens on a Single Device
frame: Create/Start a New Plot Frame
grconvertX: Convert between Graphics Coordinate Systems
grconvertY: Convert between Graphics Coordinate Systems
identify: Identify Points in a Scatter Plot
layout: Specifying Complex Plot Arrangements
lcm: Specifying Complex Plot Arrangements
locator: Graphical Input
screen: Creating and Controlling Multiple Screens on a Single Device
split.screen: Creating and Controlling Multiple Screens on a Single Device
strheight: Plotting Dimensions of Character Strings and Math Expressions
strwidth: Plotting Dimensions of Character Strings and Math Expressions

Parameters:

asp: Set up World Coordinates for Graphics Window
clip: Set Clipping Region
par: Set or Query Graphical Parameters
pch: Add Points to a Plot
xinch: Graphical Units
xlim: Set up World Coordinates for Graphics Window
xyinch: Graphical Units
yinch: Graphical Units
ylim: Set up World Coordinates for Graphics Window

Plots for Single Samples

```
> x <- iris[,1]
> plot(x)
> hist(x)
> boxplot(x)
```

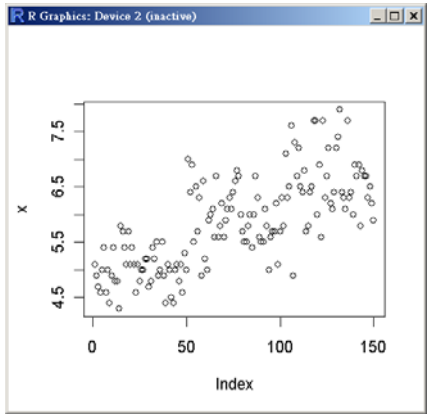


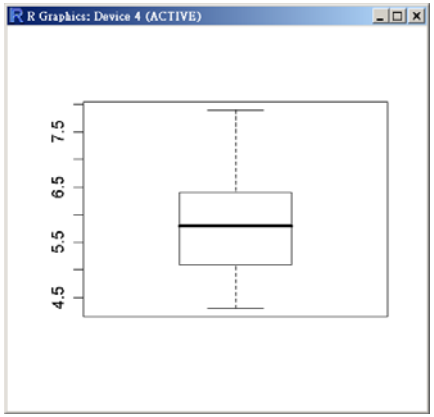

```
> title("Boxplot of time")
> hist(time, xlim=c(-5, 50))
> title("Histogram of time")
>
> hist(time, xlim=c(-5, 50))
> hist(time, xlim=c(-5, 50))
Error in plot.new() : figure margins too large
>
```

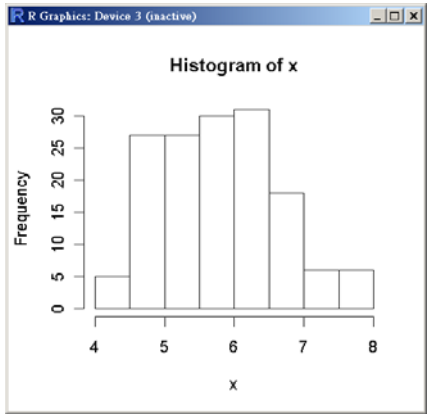
Plots Packages Help Viewer

Zoom Export Publish

將繪圖視窗區域拉大



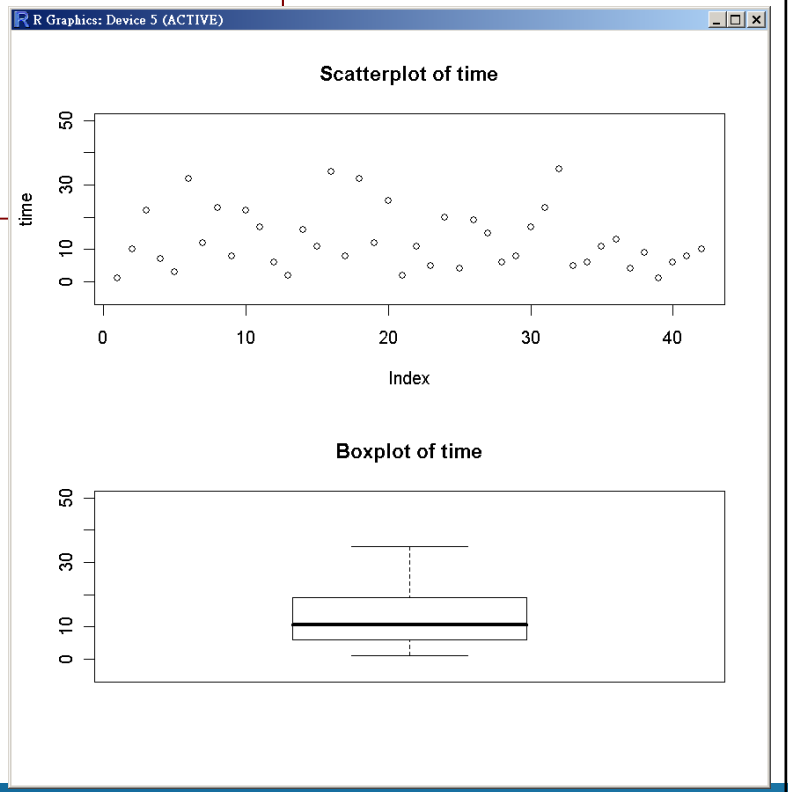




圖標題和範圍 (Title and Limit)

```
> head(gehan)
  pair time cens   treat
1    1    1    1 control
2    1   10    1    6-MP
3    2   22    1 control
4    2    7    1    6-MP
5    3    3    1 control
6    3   32    0    6-MP
```

```
> library(MASS)
> data(gehan)
> time <- gehan$time
>
> 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")

> 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

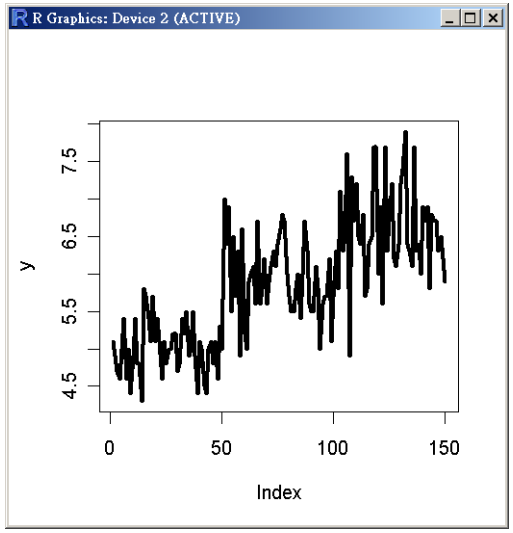
```
> plot(y, type="l", lwd=3)
```

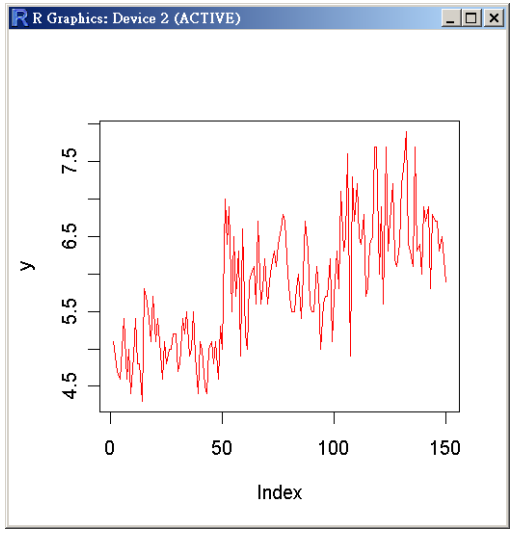
- col: color 顏色

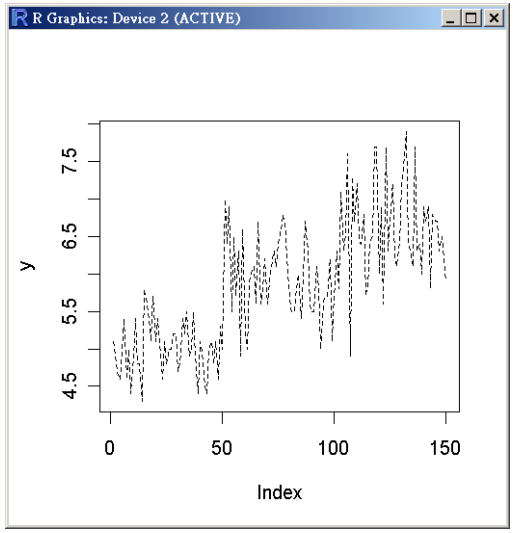
```
> plot(y, type="l", col="red")
```

- lty: line type 線的型式

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

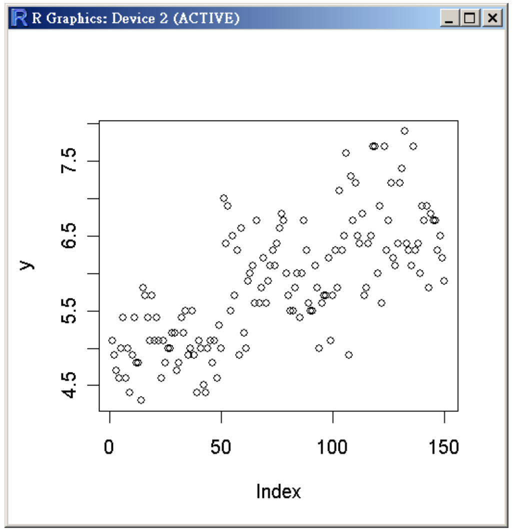


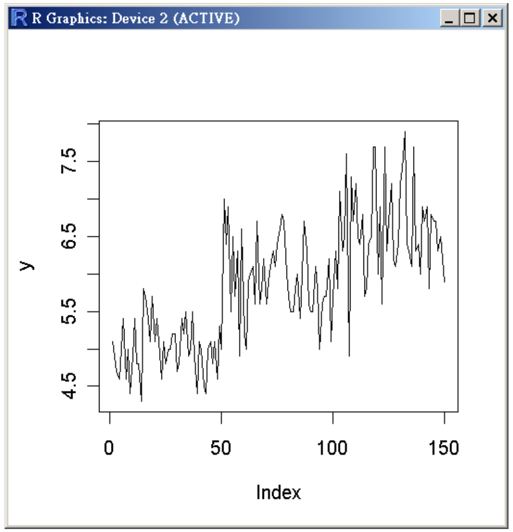


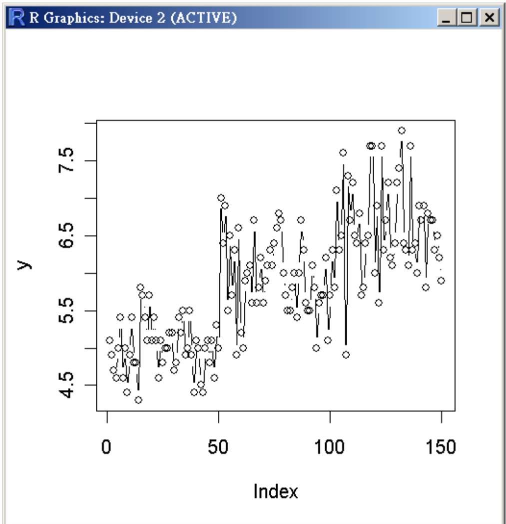


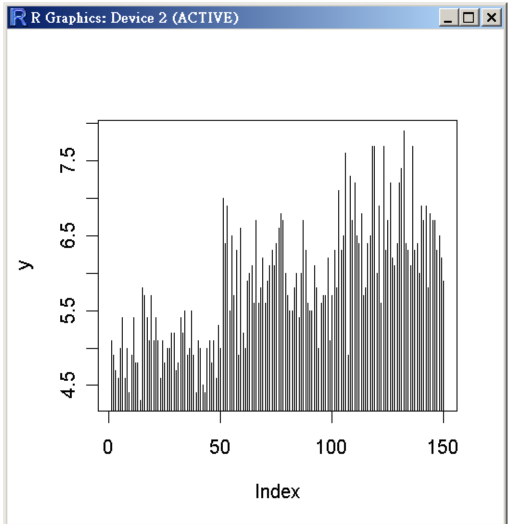
線的型式 (Line Type)

```
> y <- iris[,1]
> plot(y, type="p") # points
> plot(y, type="l") # lines
> plot(y, type="b") # both
> plot(y, type="h") #histogram-like
> plot(y, type=“n") # none
```



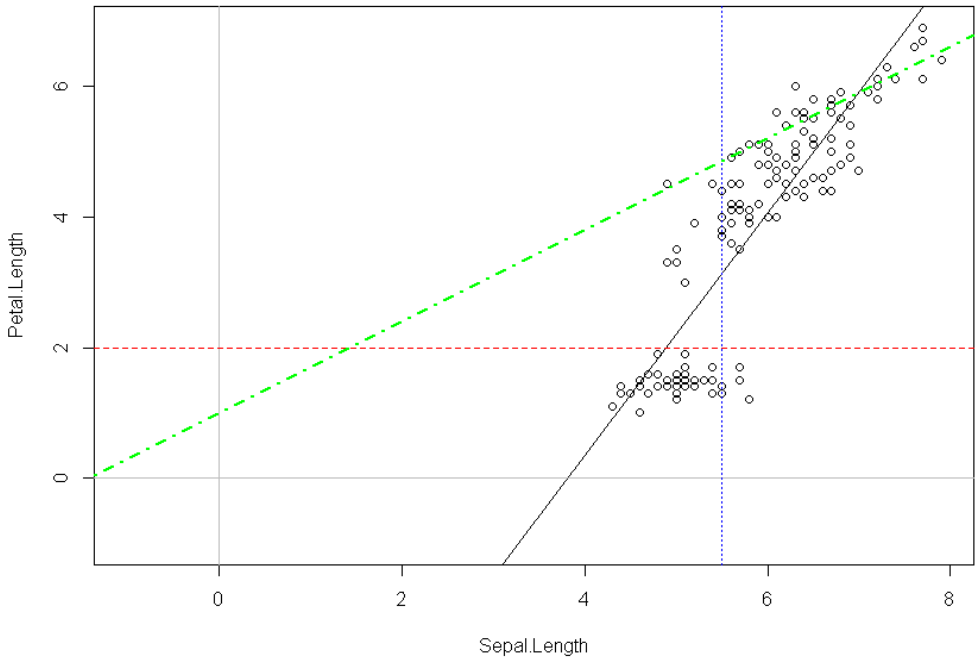






於圖中加畫出直線: abline

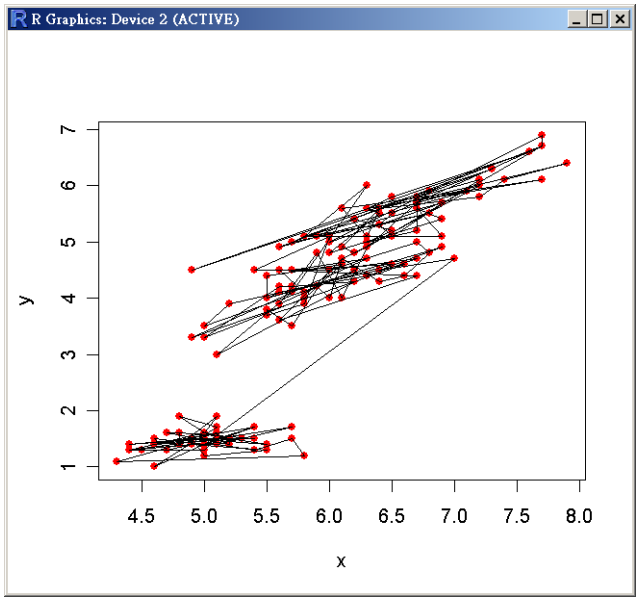
```
attach(iris)
plot(Sepal.Length, Petal.Length, xlim = c(-1, max(Sepal.Length)),
     ylim = c(-1, max(Petal.Length)))
abline(lm(Petal.Length ~ Sepal.Length), col = "black")
abline(h = 0, col = "grey")
abline(v = 0, col = "grey")
abline(h = 2, col = "red", lty = 2)
abline(v = 5.5, col = "blue", lty = 3)
abline(a = 1, b = 0.7,  col = "green", lty = 4, lwd = 2)
detach(iris)
```

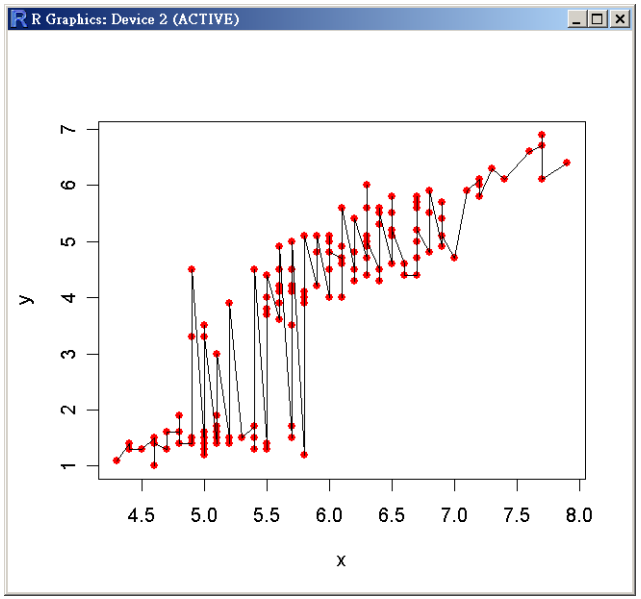


lines(): 點的連結

- 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)
```

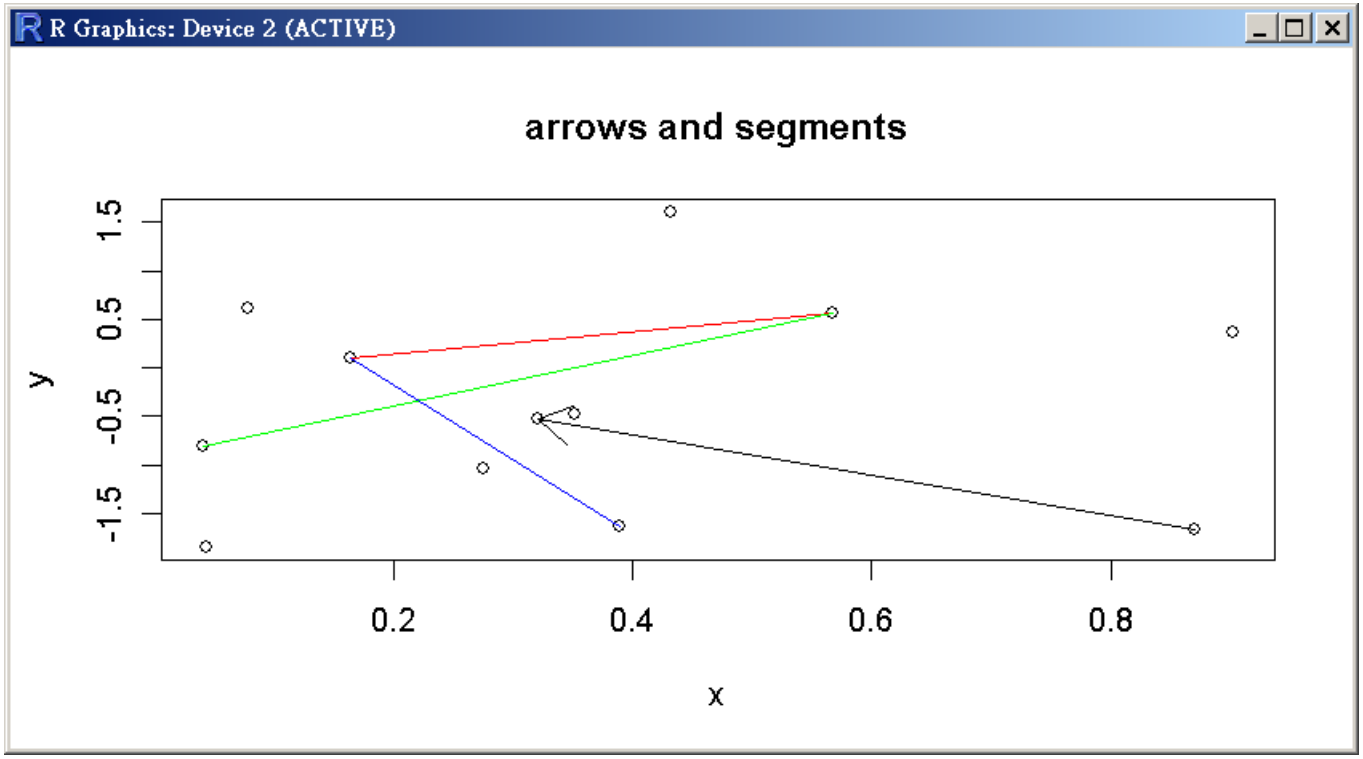





```
> sequence <- order(x)
> plot(x, y, pch=16, col="red")
> lines(x[sequence], y[sequence])
```

箭頭arrows與線段segments

```
> 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"))
```



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()`**.

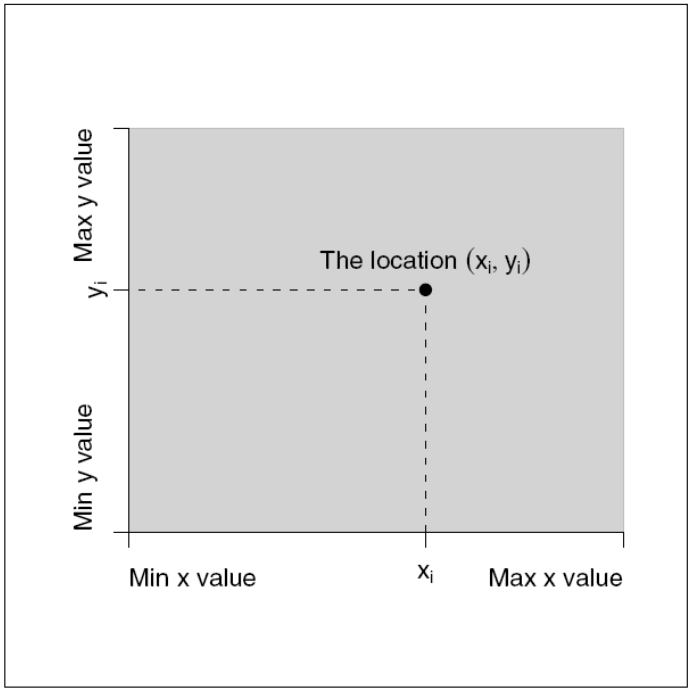
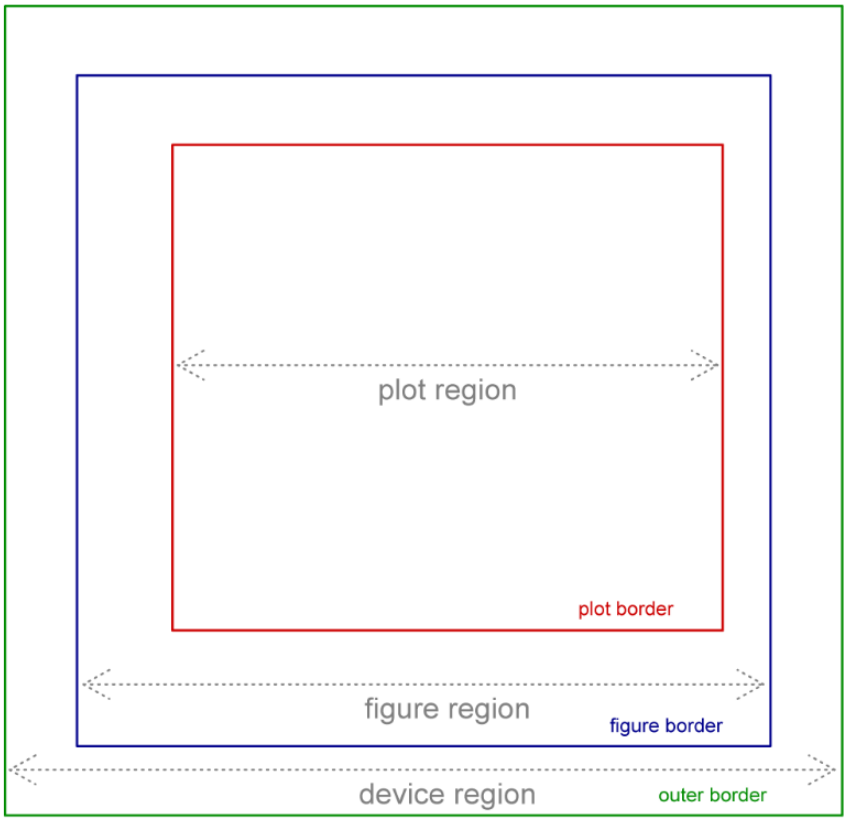


Figure 3.3
The user coordinate system in the plot region. Locations within this coordinate system are relative to the scales on the plot axes.



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

高階圖形調控項

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

```
> op <- par()
> par(col="red")
> plot(1,2)
> par(op)
```

- `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

>?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**.

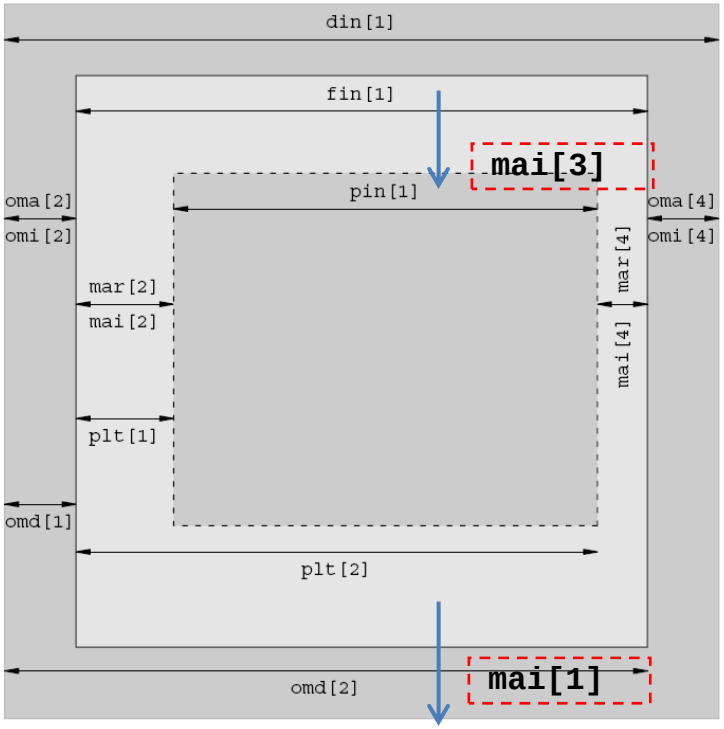
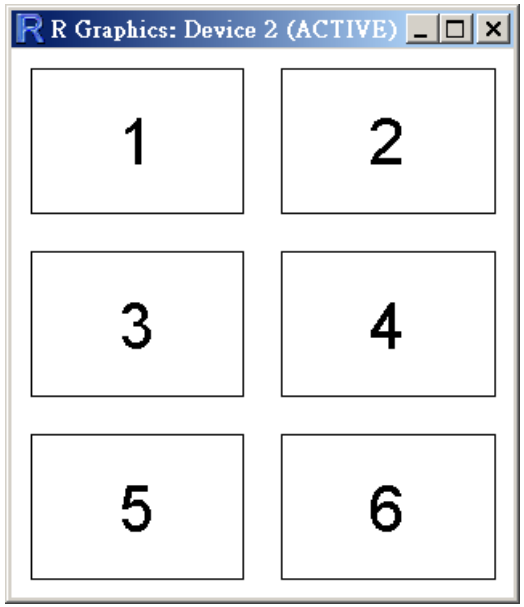


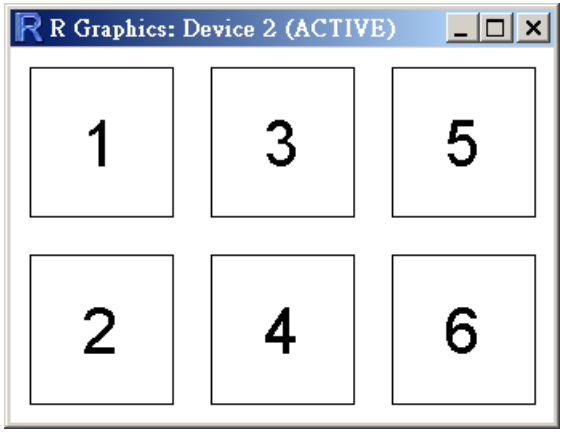
Figure 3.13
Graphics state settings controlling plot regions. These are some of the settings that control the widths and horizontal locations of the plot regions. For ease of comparison, this diagram has the same layout as Figure 3.1: the central grey rectangle represents the plot region, the lighter 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))
> myplot(6)
```



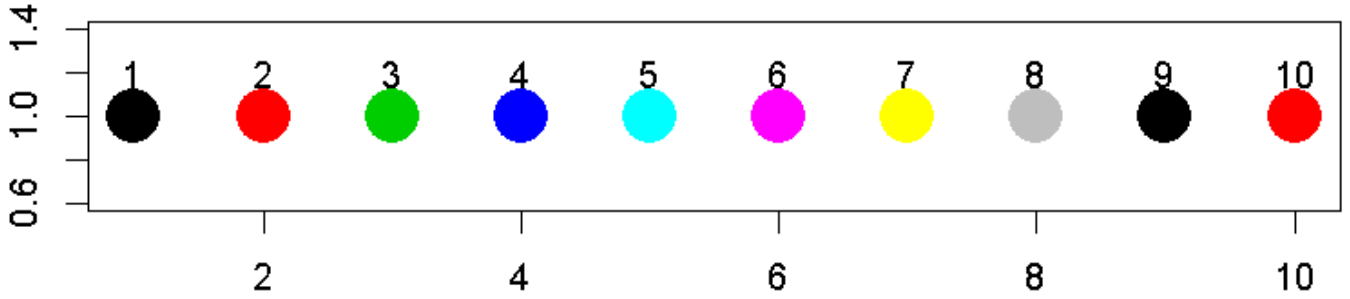


顏色 (Color)

- 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)
```



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

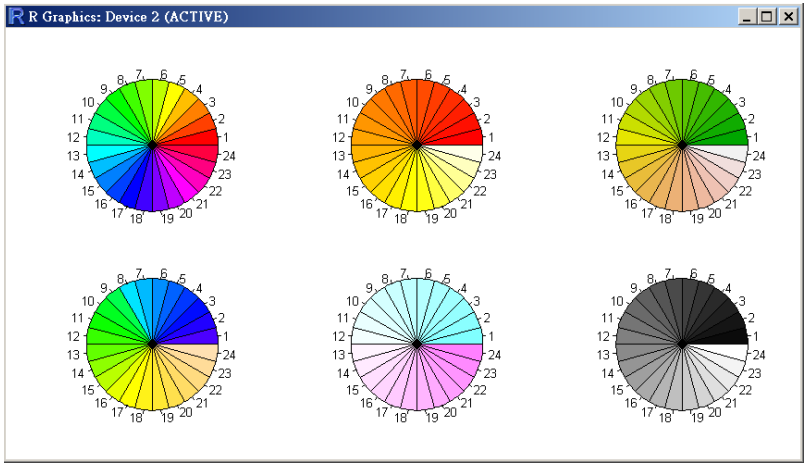
Name	Description
`rainbow()`	Colors vary from red through orange, yellow, green, blue, and indigo, to violet.
`heat.colors()`	Colors vary from white, through orange, to red.
`terrain.colors()`	Colors vary from white, through brown, to green.
`topo.colors()`	Colors vary from white, through brown then 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*

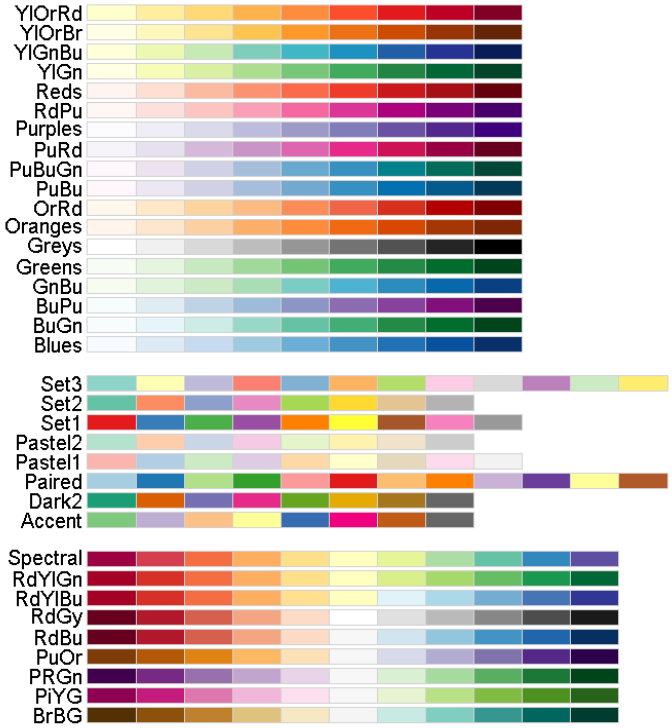
```
> par(mfrow=c(2,3))
> n <- 24
> pie(rep(1,n), col=rainbow(n))
> pie(rep(1,n), col=heat.colors(n))
> pie(rep(1,n), col=terrain.colors(n))
> pie(rep(1,n), col=topo.colors(n))
> pie(rep(1,n), col=cm.colors(n))
> pie(rep(1,n), col=grey(1:n/n))
```



RColorBrewer : ColorBrewer Palettes

- **RColorBrewer**: Provides color schemes for maps (and other graphics) designed by Cynthia Brewer as described at http://colorbrewer2.org

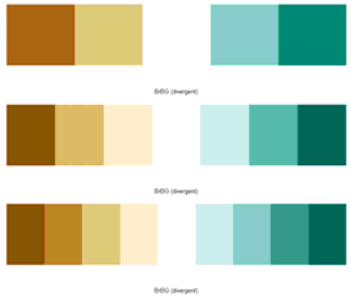
```
brewer.pal(n, name)
display.brewer.pal(n, name)
display.brewer.all(n=NULL, type="all", select=NULL, exact.n=TRUE, colorblindFriendly=FALSE)
brewer.pal.info
# n: Number of different colors in the palette, mim=3, maximum depends
# name: A palette name from the lists below
# type: One of the string "div", "qual", "seq", or "all"
```



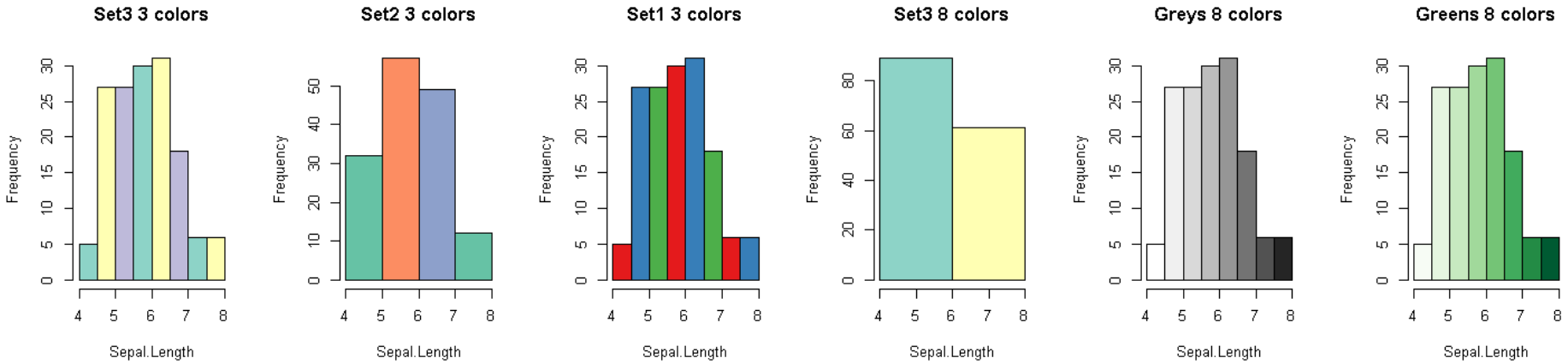

```
> library(RColorBrewer)
> brewer.pal.info
         maxcolors category colorblind
BrBG            11      div       TRUE
PiYG            11      div       TRUE
...
Spectral        11      div      FALSE
Accent           8     qual      FALSE
Dark2            8     qual       TRUE
...
YlOrBr           9      seq       TRUE
YlOrRd           9      seq       TRUE
> display.brewer.all()
```

RColorBrewer: ColorBrewer Palettes

```
> display.brewer.pal(5, "BrBG")
> display.brewer.pal(7, "BrBG")
> display.brewer.pal(9, "BrBG")
```



```
> attach(iris)
> par(mfrow=c(1,6))
> hist(Sepal.Length, breaks=10, col=brewer.pal(3, "Set3"), main="Set3 3 colors")
> hist(Sepal.Length, breaks=3 ,col=brewer.pal(3, "Set2"), main="Set2 3 colors")
> hist(Sepal.Length, breaks=7, col=brewer.pal(3, "Set1"), main="Set1 3 colors")
> hist(Sepal.Length, breaks= 2, col=brewer.pal(8, "Set3"), main="Set3 8 colors")
> hist(Sepal.Length, col=brewer.pal(8, "Greys"), main="Greys 8 colors")
> hist(Sepal.Length, col=brewer.pal(8, "Greens"), main="Greens 8 colors")
```

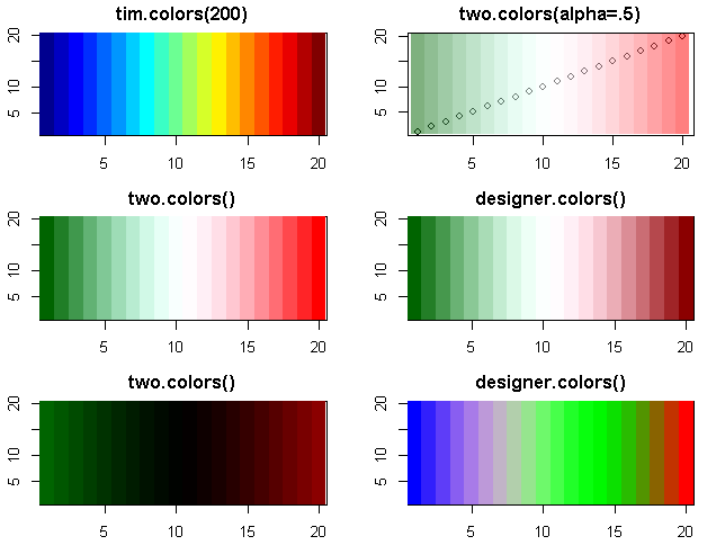


tim.colors {fields}: Some useful color tables for images and tools to handle them

選讀

```
> library(fields)
> par(mfcol=c(3,2))
> x <- 1:20
> y <- 1:20
> z <- outer(x, rep(1,20), "+")
> obj <- list(x=x, y=y, z=z)
> image(obj, col=tim.colors(200), main="tim.colors(200)")
> image(obj, col=two.colors(), main="two.colors()")
> image(obj, col=two.colors(start="darkgreen", end="darkred", middle="black"),
+       main="two.colors()")
> plot(x, y,  main="two.colors(alpha=.5)")
> image(obj, col=two.colors(alpha=.5), add=TRUE)
> image(obj, col=designer.colors(), main="designer.colors()")
> coltab <- designer.colors(col=c("blue", "grey", "green", "red"))
> image(obj, col= coltab, main="designer.colors()")
```

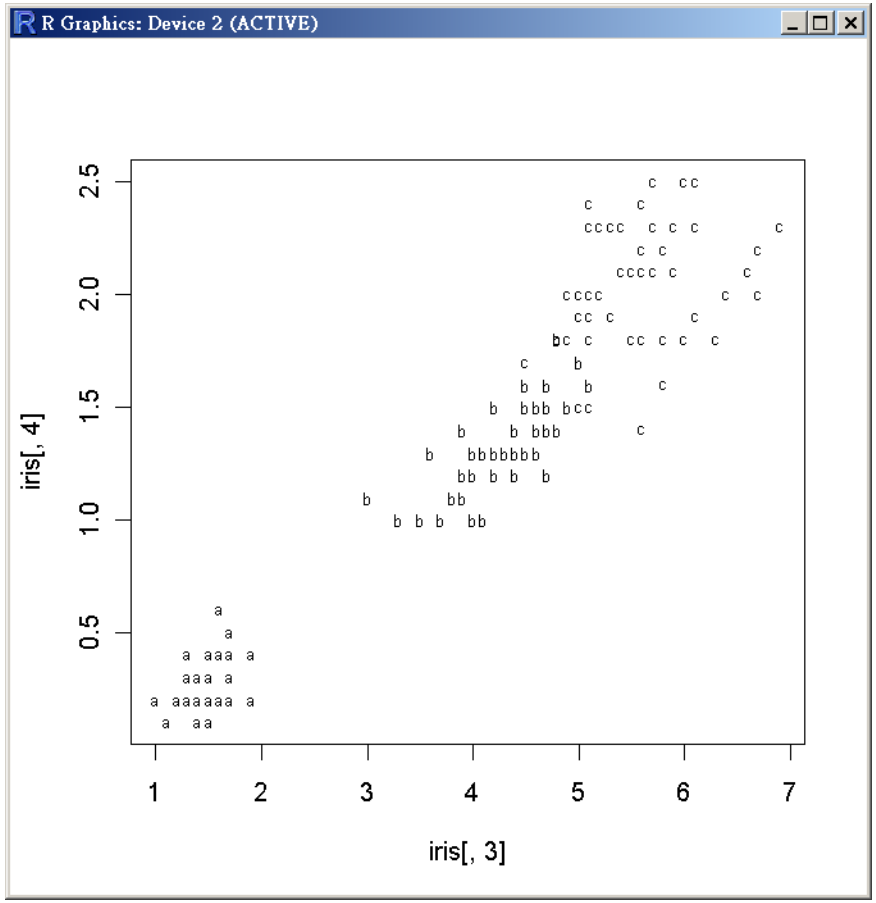
```
> obj
$x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

$y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

$z
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
 [1,]    2    2    2    2    2    2    2    2    2    2    2    2
 [2,]    3    3    3    3    3    3    3    3    3    3    3    3
 [3,]    4    4    4    4    4    4    4    4    4    4    4    4
 [4,]    5    5    5    5    5    5    5    5    5    5    5    5
 [5,]    6    6    6    6    6    6    6    6    6    6    6    6
 [6,]    7    7    7    7    7    7    7    7    7    7    7    7
```




```
# fields: Tools for Spatial Data
tim.colors(n = 64, alpha=1.0)
larry.colors()
two.colors(n=256, start="darkgreen", end="red", middle="white",
alpha=1.0)
designer.colors(n=256, col= c("darkgreen", "white", "darkred"),
                x=seq(0, 1, length(col)) ,alpha=1.0)

color.scale( z, col=tim.colors(256), zlim =NULL,
transparent.color="white", eps= 1e-8)
fieldsPlotColors( col,...)
```

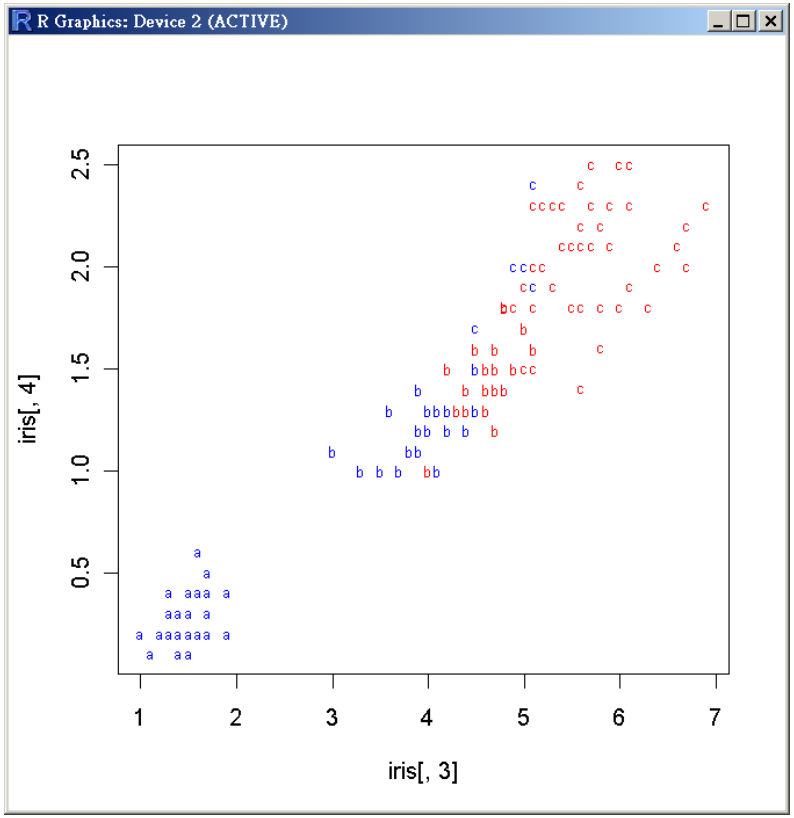
文字標號 (Text Labels)

```
> 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)
```



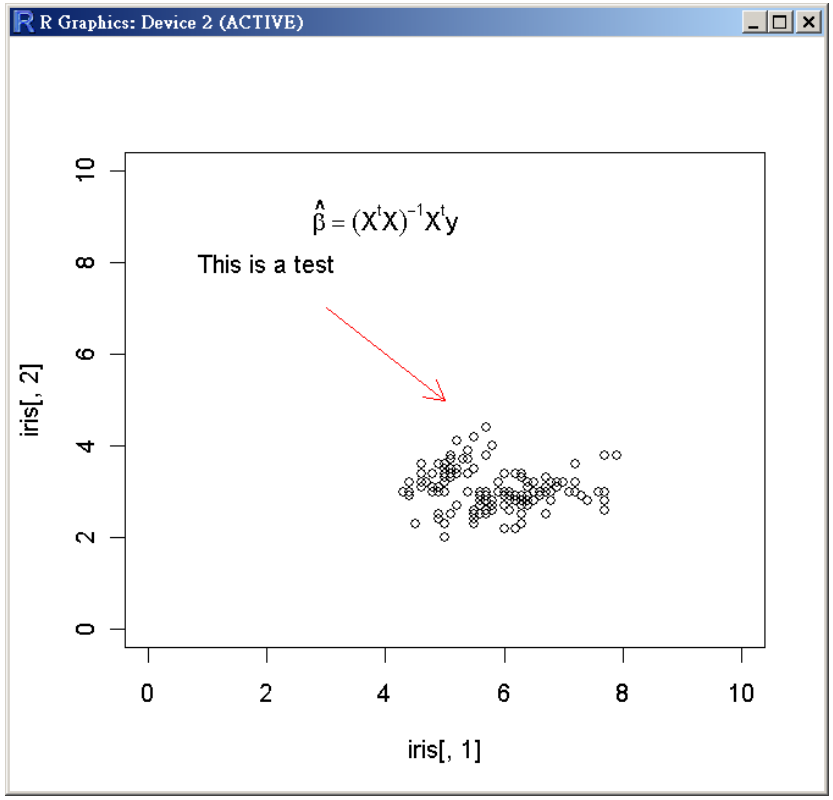
文字標號 (Text Labels)

```
> 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

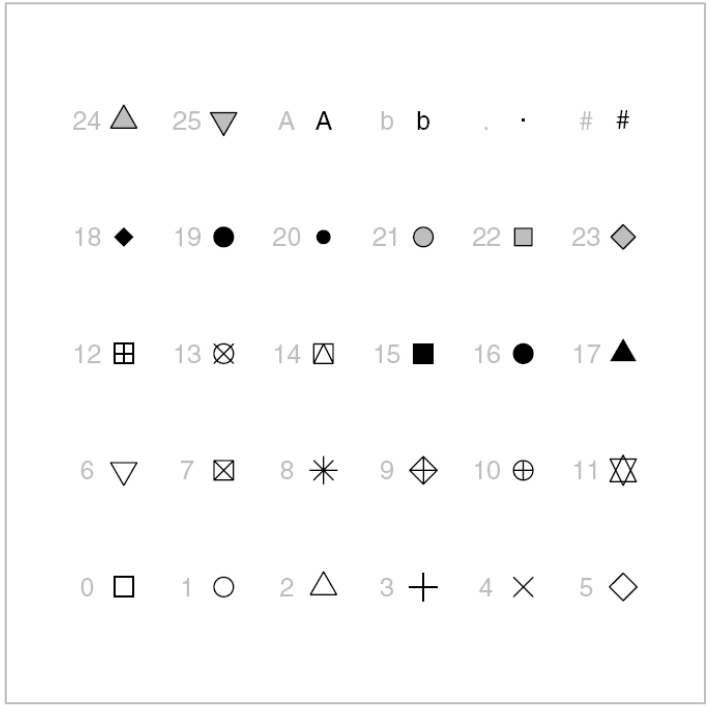
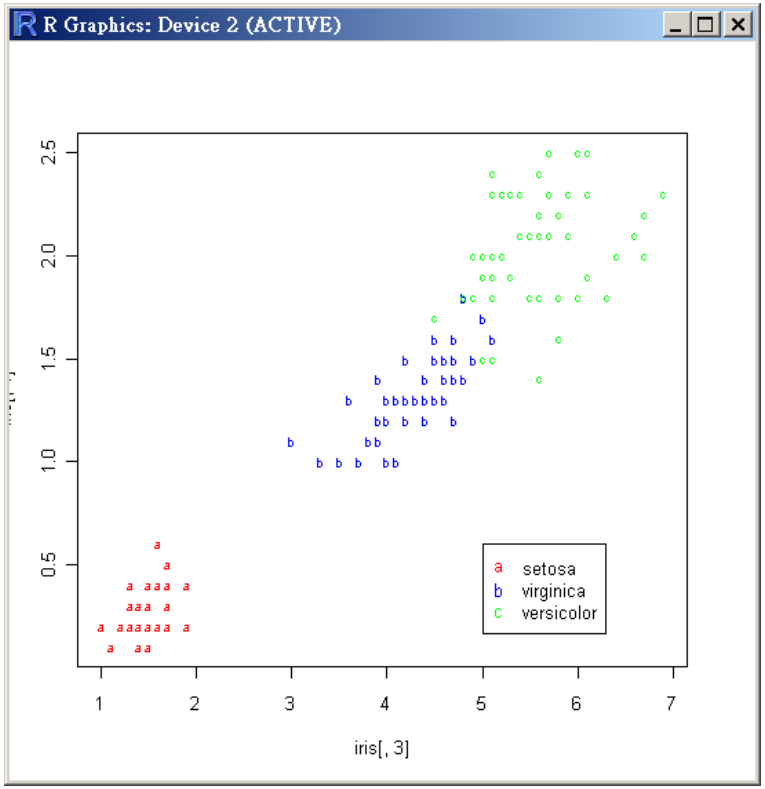


Figure 3.10
Data symbols available in R. A particular data symbol is selected by specifying an integer between 0 and 25 or a single character for the `pch` graphical setting. In the diagram, the relevant integer or character `pch` value is shown in grey to the left of the relevant symbol.

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"))
```



axis: 座標軸

- 語法

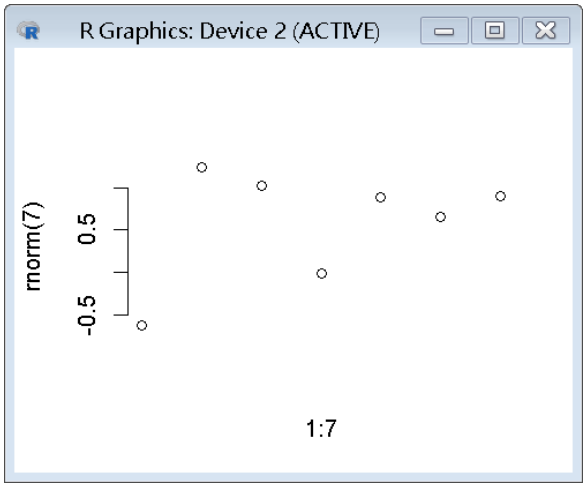
```
axis(side, at = NULL, labels = TRUE, tick = TRUE,
line = NA,  pos = NA, outer = FALSE, font = NA,
lty = "solid",lwd = 1, lwd.ticks = lwd, col = NULL,
col.ticks = NULL, hadj = NA, padj = NA, ...)
```

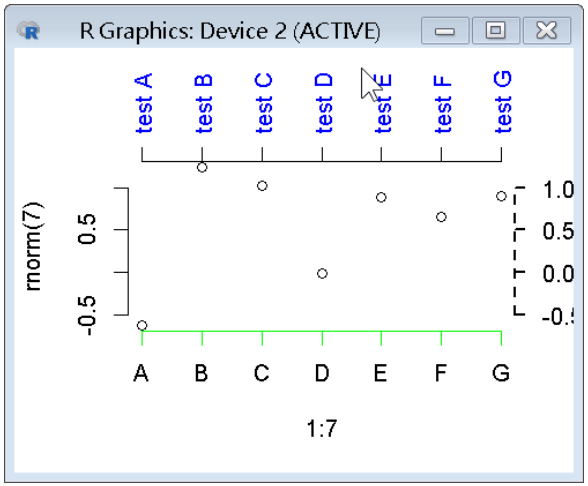
- Arguments
  - side: 為一整數，指定座標軸畫在圖形的哪一邊， 1=below, 2=left, 3=above and 4=right.
  - at: 指定圖上的tick-marks所在位置
  - labels: 指定tick-marks的標號
  - tick: (logical) tickmarks，axis line 需要畫出長來嗎?
  - pos: (coordinate) axis line畫的位置
  - hadj: adjustment for all labels parallel (‘horizontal’) to the reading direction.
  - padj: adjustment for each tick label perpendicular to the reading direction
  - las: labels are parallel (=0) or perpendicular(=2) to axis

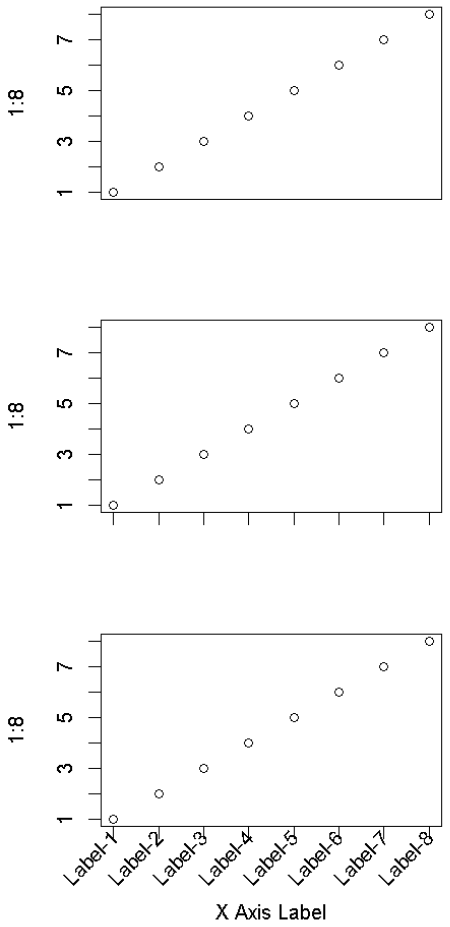
範例

選讀

```
plot(1:7, rnorm(7), xaxt = "n", frame = FALSE)
axis(1, 1:7, LETTERS[1:7], col = "green")
axis(3, 1:7, paste("test", LETTERS[1:7]), col.axis = "blue", las=2)
axis(4, lty=2, lwd = 2, las=2)
```








```
plot(1:8, xaxt = "n",  xlab = "")
axis(1, labels = FALSE)
my.labels <- paste("Label", 1:8, sep = "-")
text(1:8, par("usr")[3] - 0.25, srt = 45, adj = 1,
     labels = my.labels, xpd = TRUE)
mtext(1, text = "X Axis Label", line = 3)
```

srt: string rotation

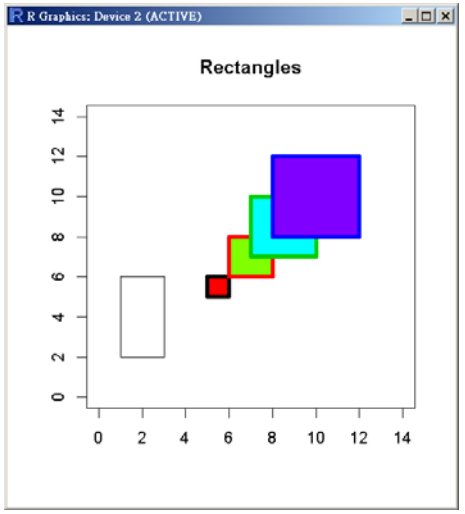
```
> par("usr")
[1] 0.72 8.28 0.72 8.28
```

xpd: all plotting is clipped to the figure region

矩形，Draw Symbols

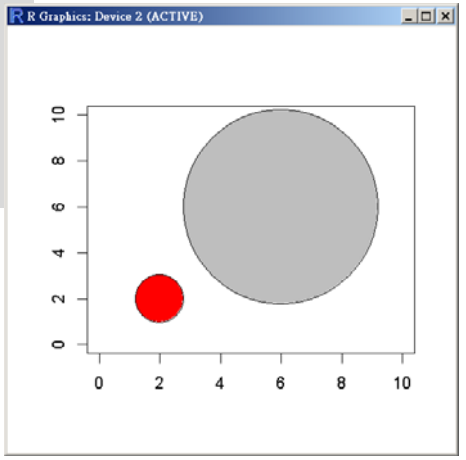
(Circles, Squares, Stars, Thermometers, Boxplots)

```
> 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="")
```

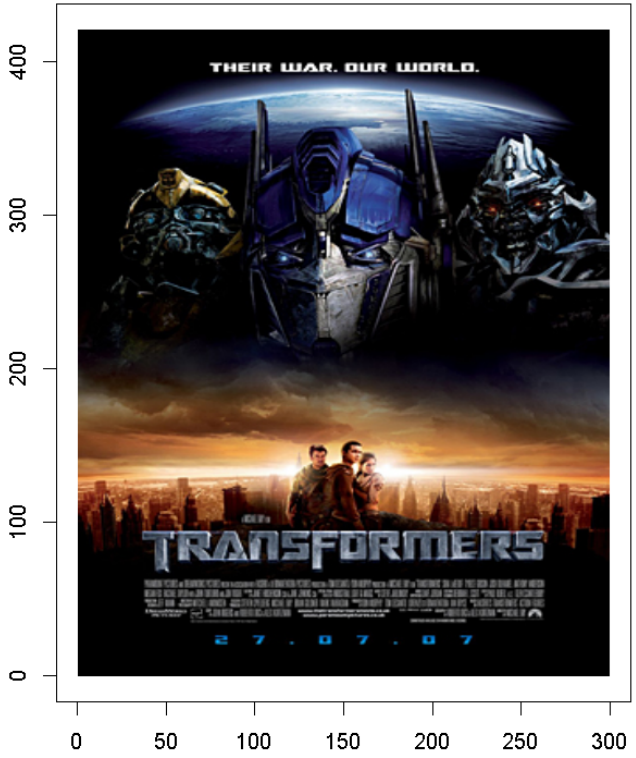


See also "draw.ellipse {plotrix}"

讀取外部影像檔案

```
> install.packages(c("tiff", "jpeg", "png", "fftwtools"),
repos="http://cran.csie.ntu.edu.tw")
> library(EBImage) # (Repositories: BioC Software)
> Transformers <- readImage("Transformers07.jpg")
> (dims <- dim(Transformers))
[1] 300 421   3
> Transformers
Image
  colorMode    : Color
  storage.mode : double
  dim          : 300 421 3
  frames.total : 3
  frames.render: 1

imageData(object)[1:5,1:6,1]
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    0    0    0    0    0
[2,]    0    0    0    0    0    0
[3,]    0    0    0    0    0    0
[4,]    0    0    0    0    0    0
[5,]    0    0    0    0    0    0
> plot(c(0, dims[1]), c(0, dims[2]), type='n',
+ xlab="", ylab="")
> rasterImage(Transformers, 0, 0, dims[1], dims[2])
```

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

BiocManager::install("EBImage")
```




```
> #install.packages("jpeg")
> library(jpeg)
> Transformers <- readJPEG("Transformers07.jpg")
```

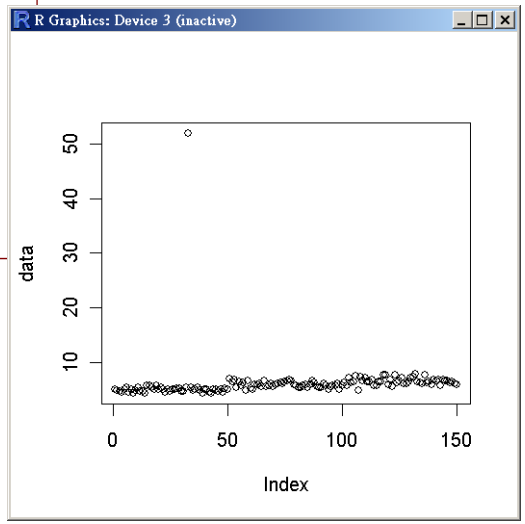
https://en.wikipedia.org/wiki/Transformers_(film)

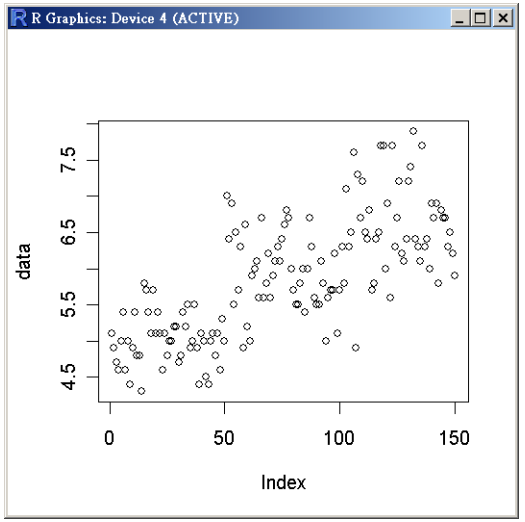
索引圖 (Index Plot)

- 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.

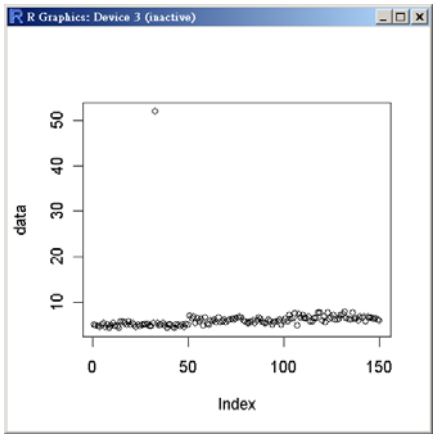
```
> data <- iris[,1]
> data[33] <- data[33]*10
> plot(data)
> ind <- which(data>15)
> data[ind]
> data[ind] <- 5.2
> windows()
> plot(data)
```

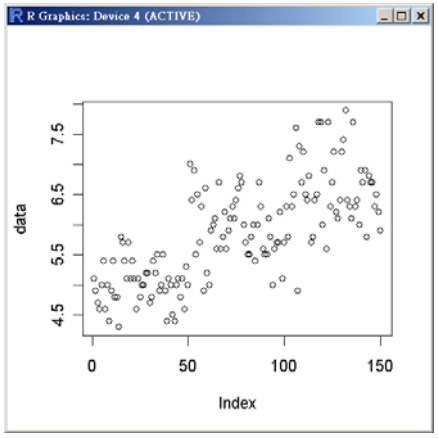


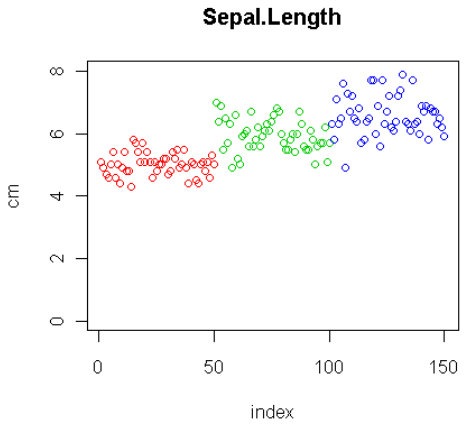


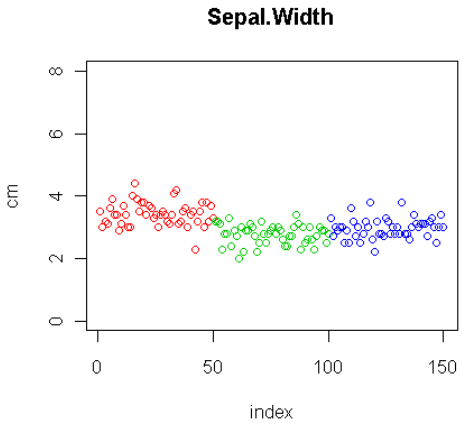
Index Plot

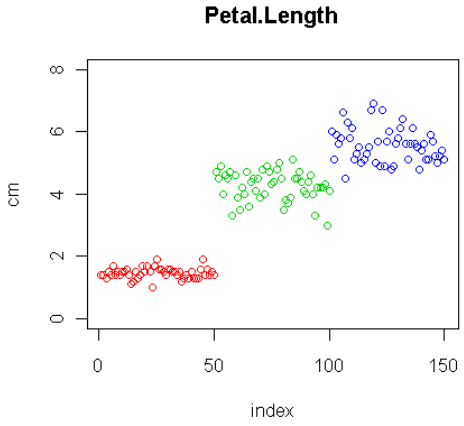
- Index plot takes a single argument which is a continuous variable and plots the values on the y axis, with the x coordinate determined by the position of the number in the vector.
- Useful for error checking.

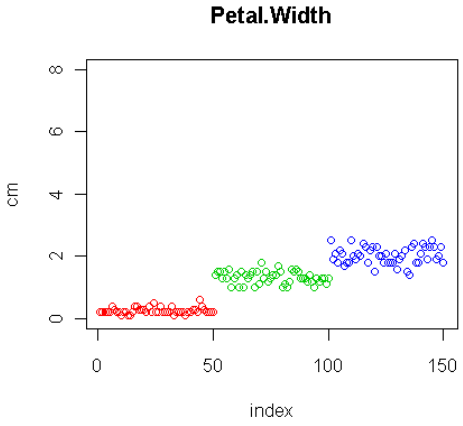








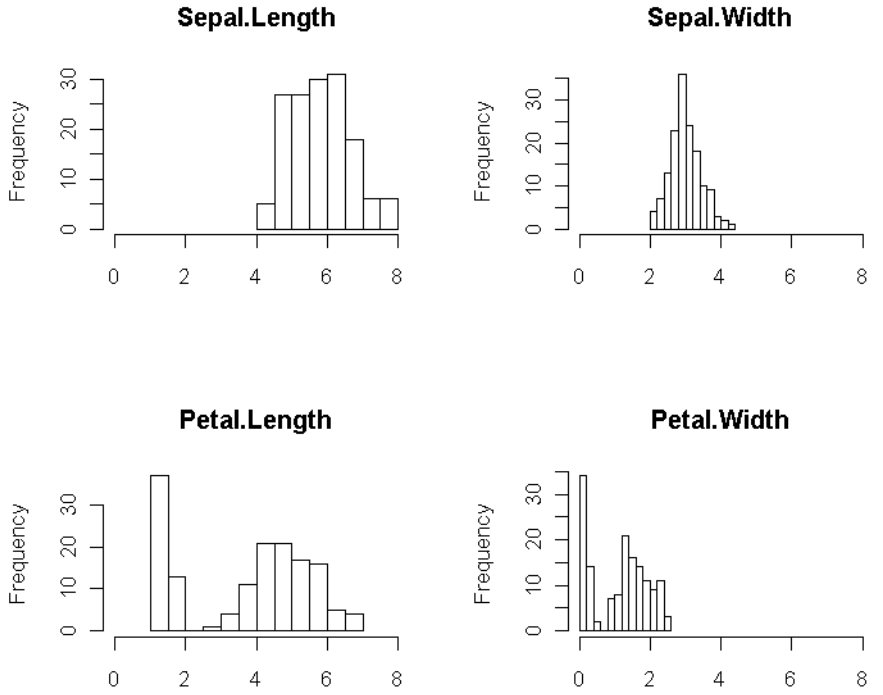


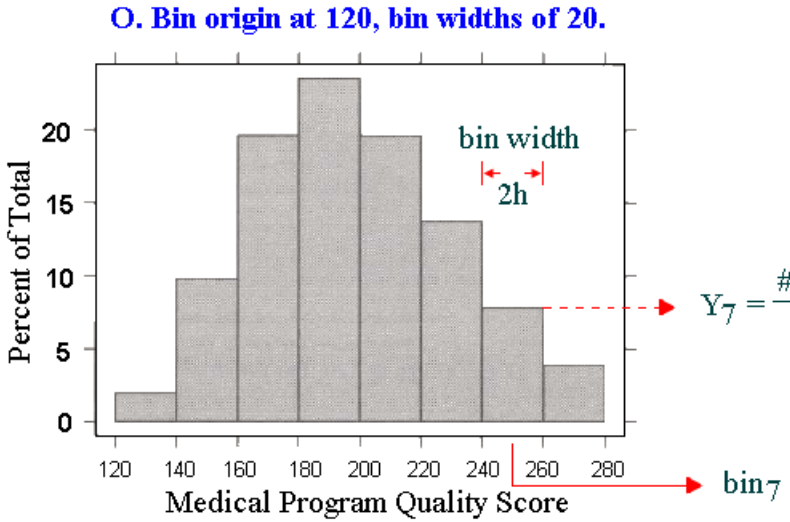


直方圖 (Histogram) (1/3)

The histogram shows:

1. center of the data (location)
2. spread of the data (scale)
3. skewness of the data
4. presence of outliers
5. presence of multiple modes in the data.





$$Y_7 = \frac{\#\text{ observations within bin}_7}{(2h)\,n}$$

```
par(mfrow=c(2, 2))
lapply(1:4, function(x) hist(iris[,x], xlim=c(0, 8),
ylim=c(0, 30), xlab="x", main=names(iris)[x]))
```

Changes in bin origin and bin widths affect the shape of the histogram

直方圖 (Histogram) (2/3)

- 1/2h adjusts the height of each bar so that the total area enclosed by the entire histogram is 1.
- The area covered by each bar can be interpreted as the probability of an observation falling within that bar.

Disadvantage for displaying a variable's distribution:

- selection of origin of the bins.
- selection of bin widths.
- the very use of the bins is a distortion of information because any data variability within the bins cannot be displayed in the histogram.

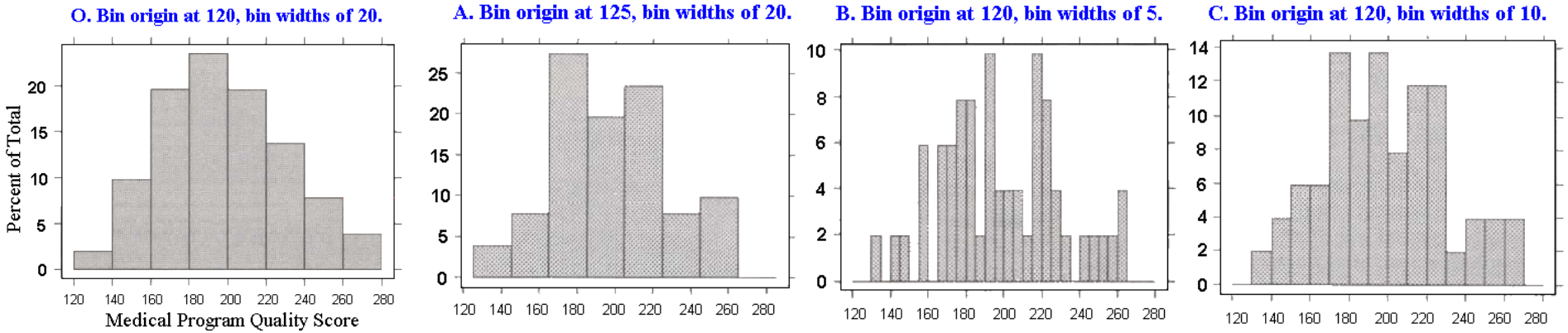


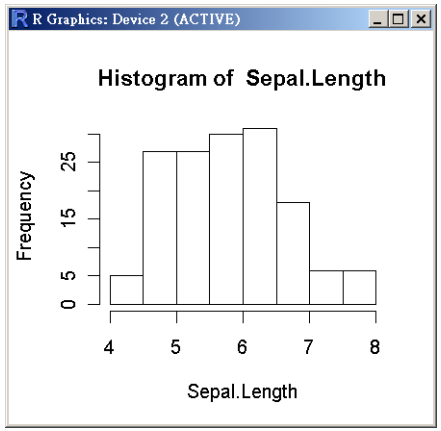
Figure Sources: Jacoby (1997).

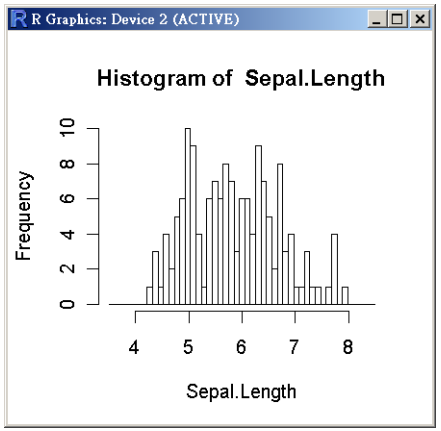
直方圖 (Histogram) (3/3)

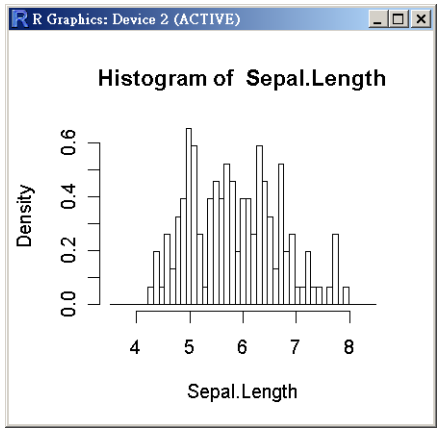
- Histogram are excellent for showing the mode, the spread and the symmetry (skew) of a set of data.

- 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)
```

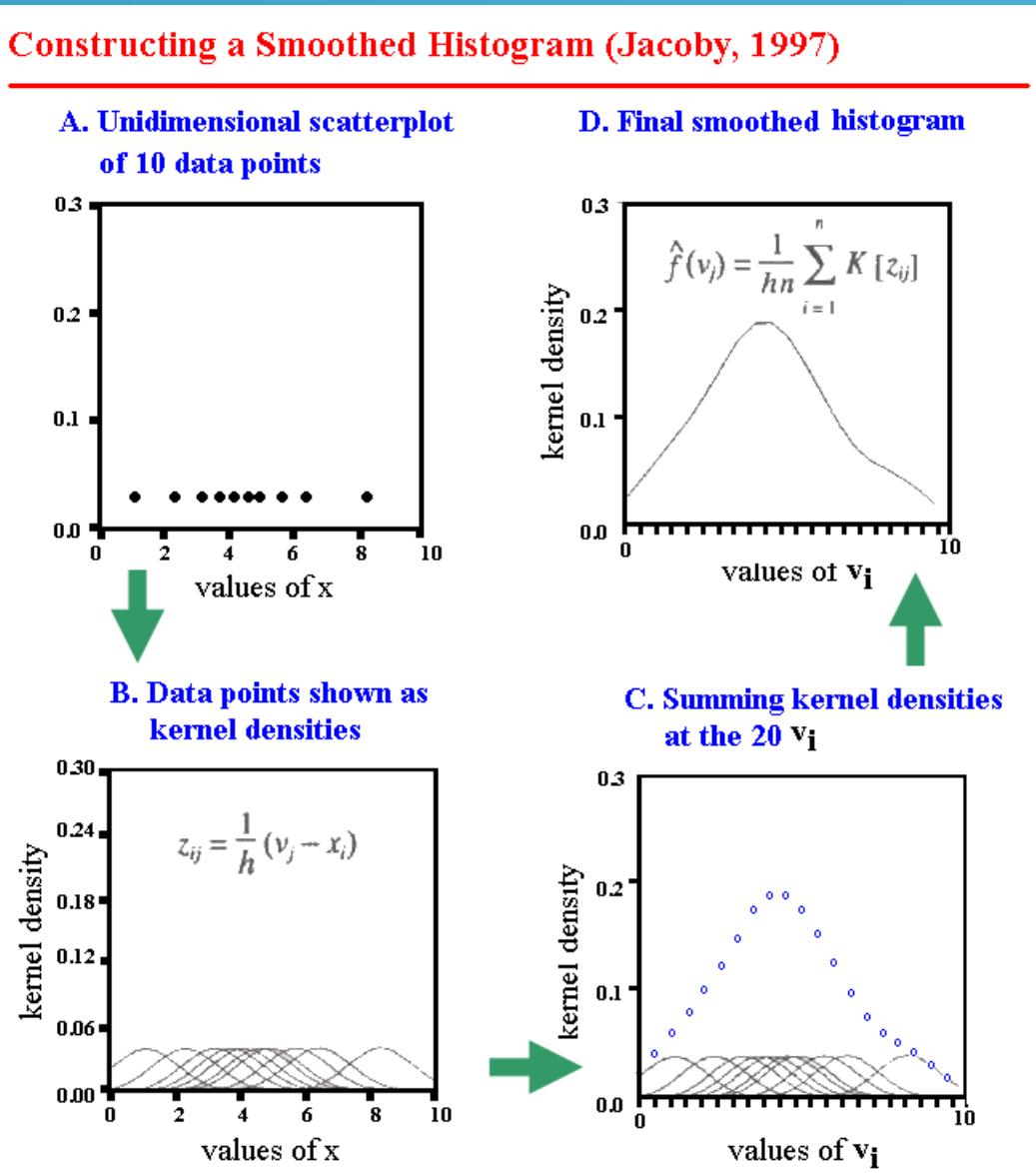






Density Plots (Smoothed Histograms) (1/3)

- Smoothed histograms overcome some of the disadvantages caused by the arbitrary, discrete bins used in traditional histogram.
- The relative height of the smooth curve corresponds to the local density.
- The overall height is adjusted so that the total area under the curve is approximately equal to 1.
- The area under the curve between any two points along the horizontal scale can be interpreted as the probability that an observation falls within that interval of data values.

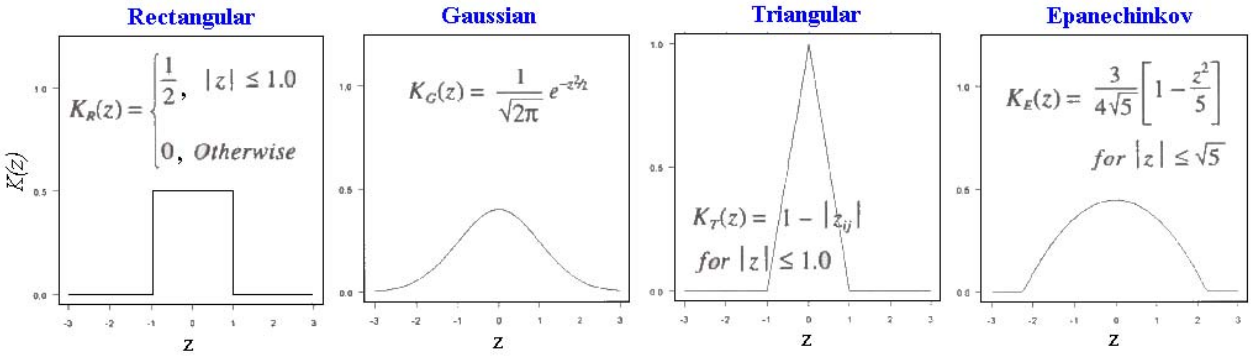


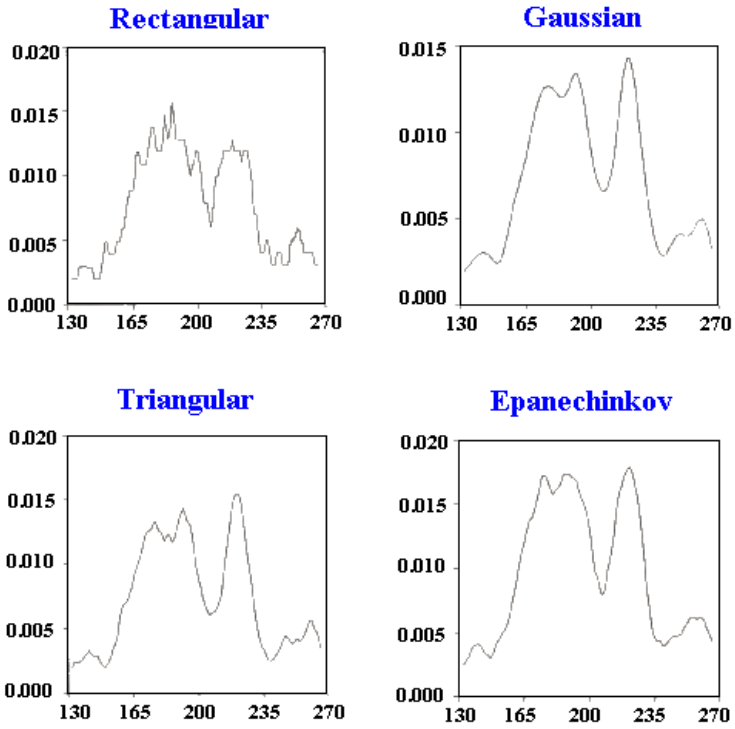
Density Plots (2/3)

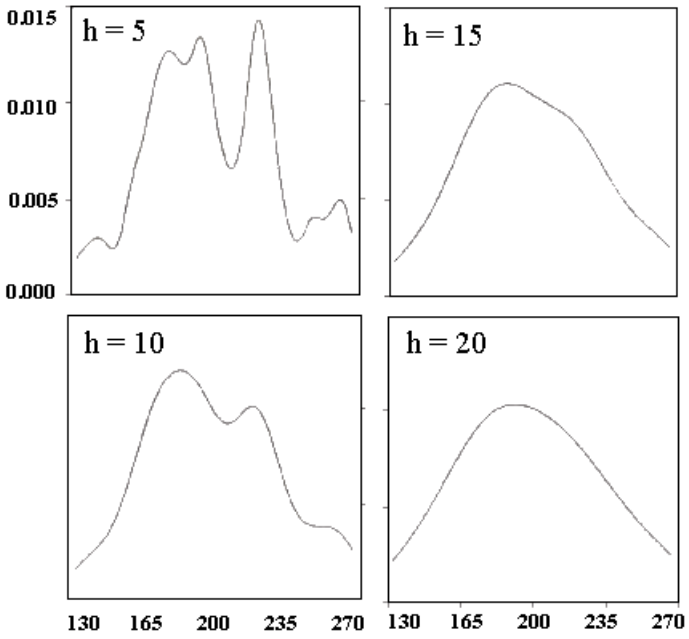
- Selection of kernels
- Selection of bandwidth

$$\hat{f}(v_j) = \frac{1}{hn} \sum_{i=1}^{n} K[z_{ij}]$$

$$z_{ij} = \frac{1}{h}(v_j - x_i)$$



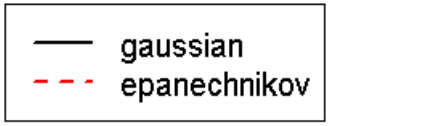




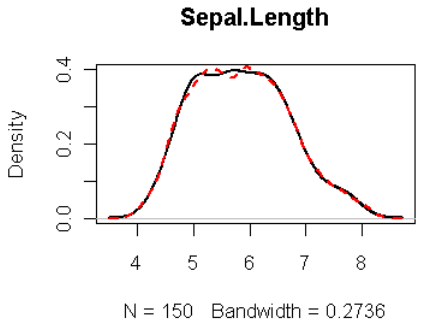
Figures modified from Jacoby (1997)

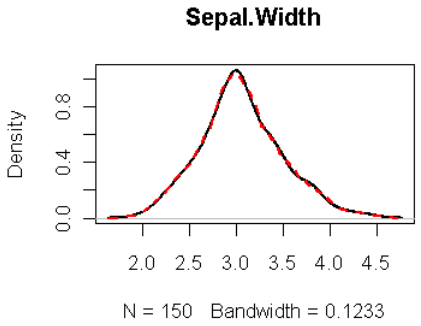
Density Plots (3/3)

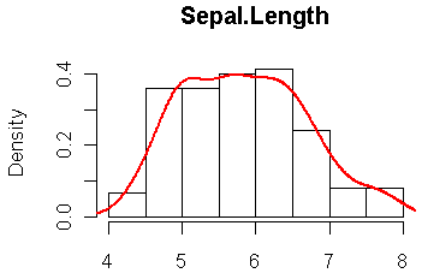
```
density(x, bw = "nrd0", adjust = 1,
        kernel = c("gaussian", "epanechnikov", "rectangular",
                   "triangular", "biweight",
                   "cosine", "optcosine"),
        weights = NULL, window = kernel, width,
        give.Rkern = FALSE,
        n = 512, from, to, cut = 3, na.rm = FALSE, ...)
```

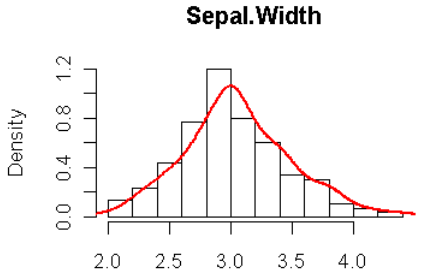


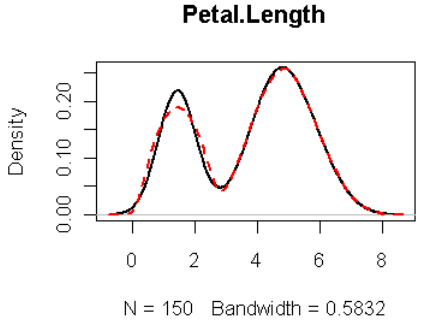

```
> plot(density(iris$Sepal.Length))
```

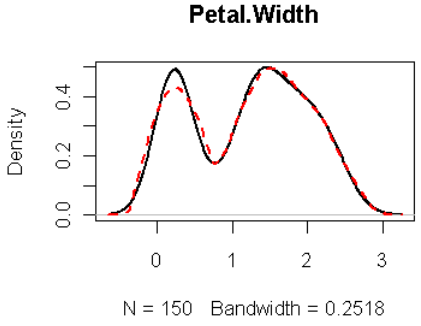


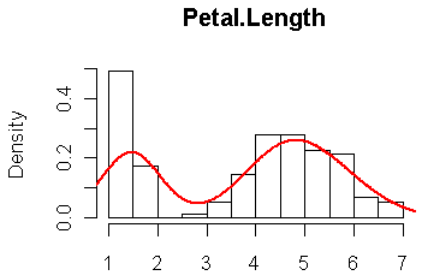


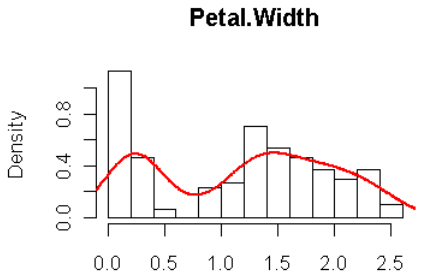






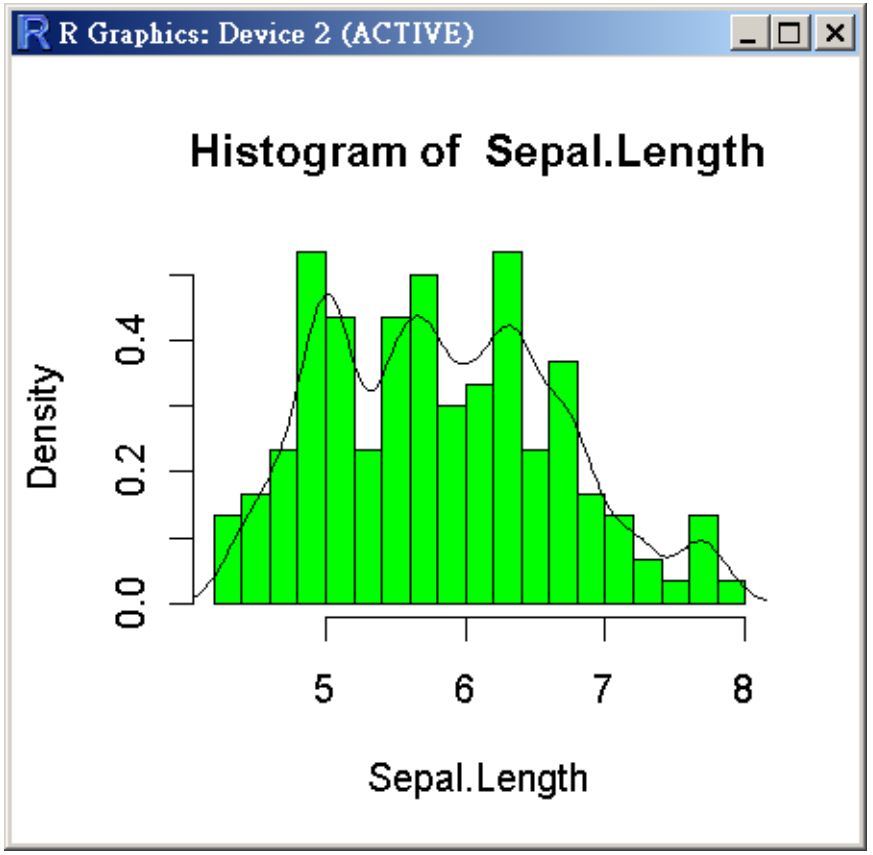




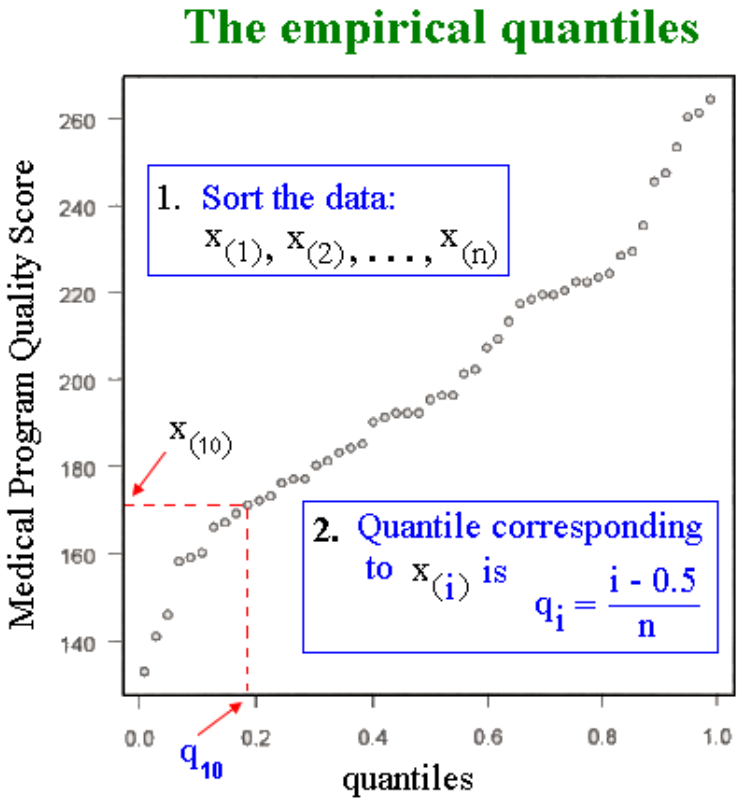


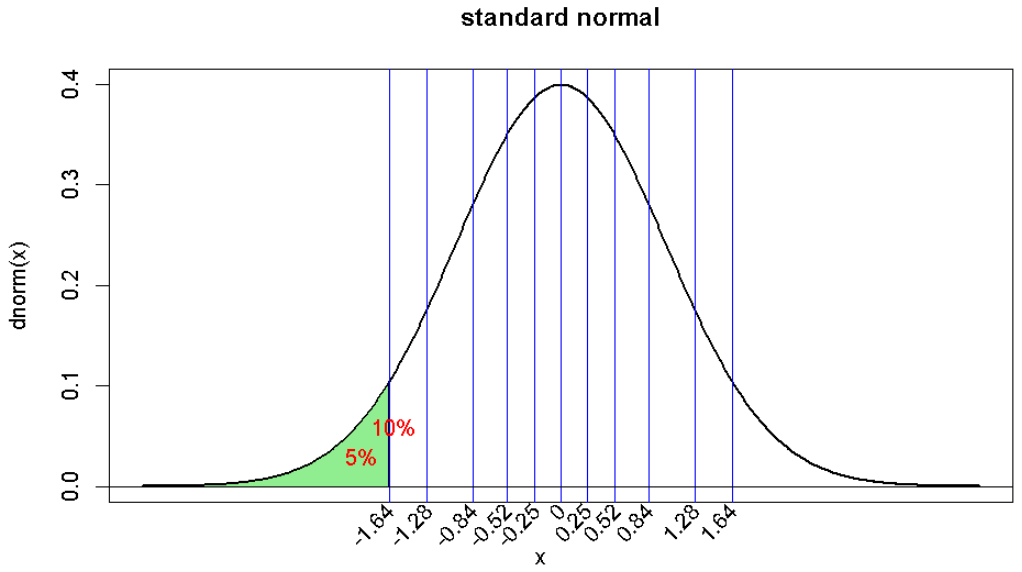
Density Estimation for Continuous Variables

```
> hist(iris[,1], breaks=15, main=title, xlab=lab, col="green", pro=T)
> lines(density(iris[,1], width=0.6, n=200))
```



Quantile Plots

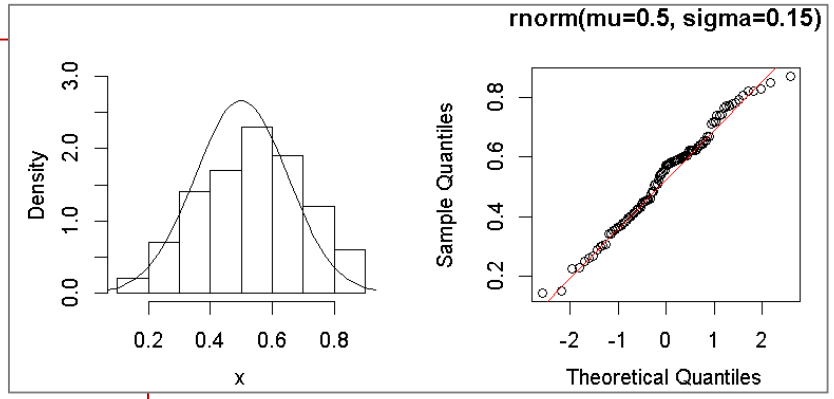




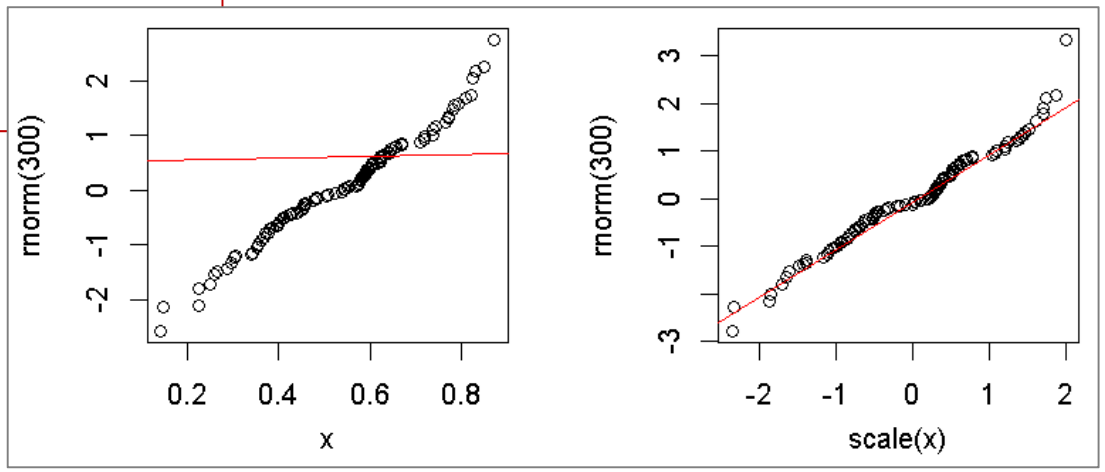
- 0.5 is subtracted from each i value to avoid extreme quantiles of exactly 0 or 1.
- The latter would cause problems if empirical quantiles were to be compared against quantiles derived from a theoretical asymptotic distribution such as the normal.
- This adjustment has no effect on the shape of any graphical display.

qqnorm, qqline, qqplot

```
> par(mfrow = c(1, 2))
> set.seed(12345);
> n <- 100; mu <- 0.5; sigma <- 0.15
> x <- rnorm(n, mu, sigma)
> hist(x, freq=FALSE, ylim=c(0, 3), main="")
> y <- seq(0, 1, length = n)
> lines(y, dnorm(y, mu, sigma), type = 'l')
> qqnorm(x, main = "rnorm(mu=0.5, sigma=0.15)");
> qqline(x)
```

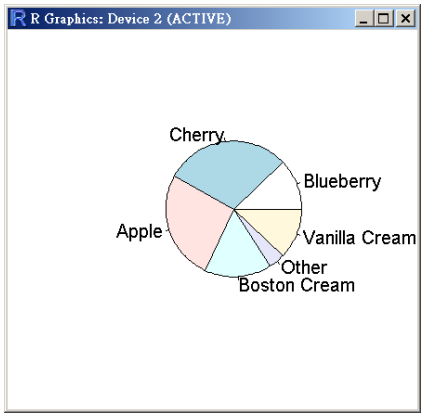



```
> qqplot(x, rnorm(300))
> qqline(x, col = 2)
> qqplot(scale(x), rnorm(300))
> qqline(scale(x), col = 2)
```

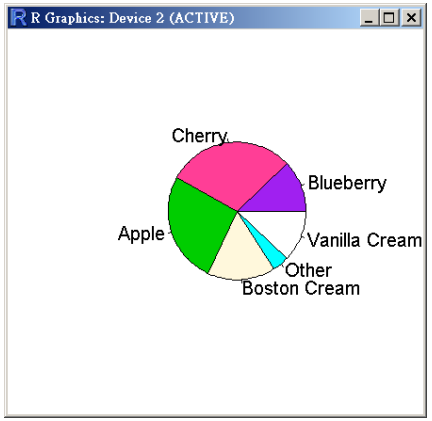


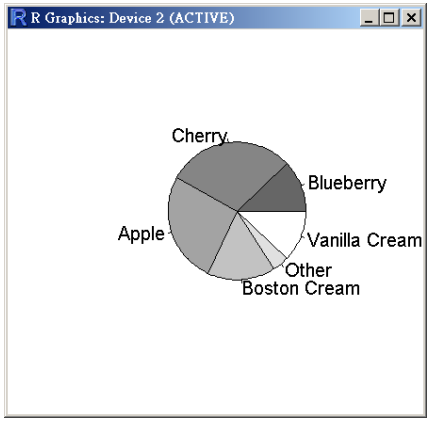
圓餅圖 (Pie Charts)

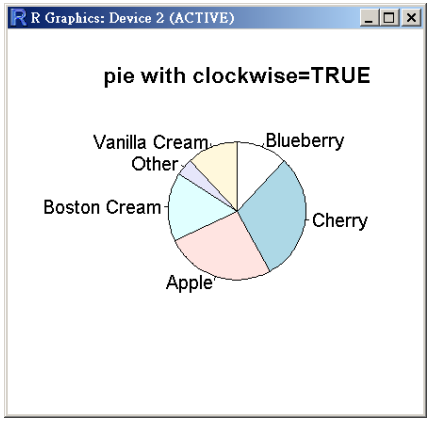
```
> pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
> sum(pie.sales)
> names(pie.sales) <- c("Blueberry", "Cherry",
    "Apple", "Boston Cream", "Other", "Vanilla Cream")

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

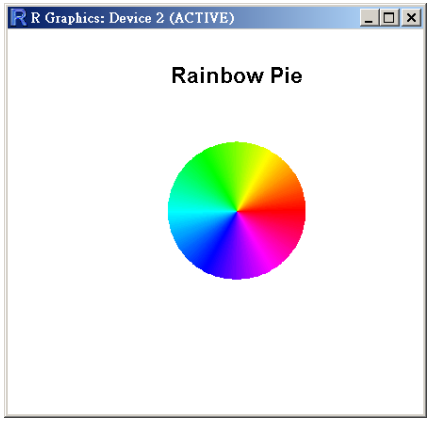



```
> pie(pie.sales, col = c("purple", "violetred1", "green3", "cornsilk", "cyan", "white"))
> pie(pie.sales, col = gray(seq(0.4,1.0,length=6)))
> pie(pie.sales, clockwise=TRUE, main="pie with clockwise=TRUE")
> pie(rep(1,200), labels="", col=rainbow(200), border=NA, main = "Rainbow Pie")
```



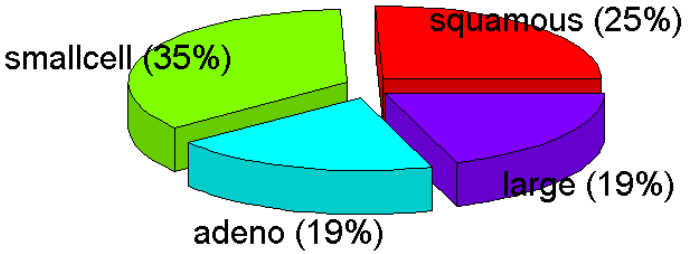


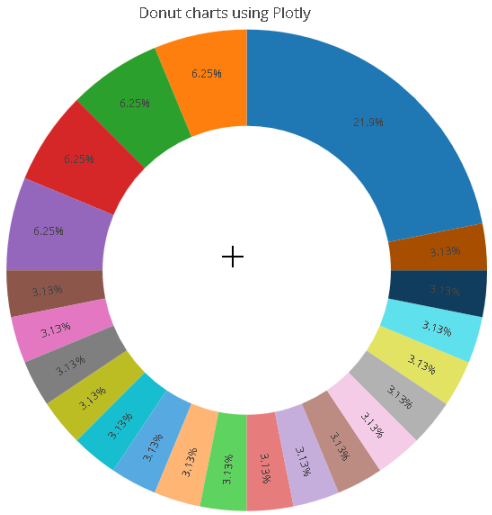




Pie Charts and Their Variants

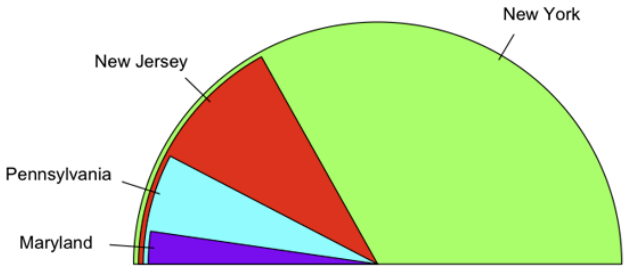
```
> library(plotrix)
> library(survival)
> head(veteran)
  trt celltype time status karno diagtime age prior
1   1 squamous   72      1    60         7  69     0
2   1 squamous  411      1    70         5  64    10
3   1 squamous  228      1    60         3  38     0
4   1 squamous  126      1    60         9  63    10
5   1 squamous  118      1    70        11  65    10
6   1 squamous   10      1    20         5  49     0
> slices <- summary(veteran$celltype)
> p <- floor(100*slices/sum(slices))
> pie3D(slices, labels=paste0(names(slices), " (",p, "%)"), explode=0.1)
```



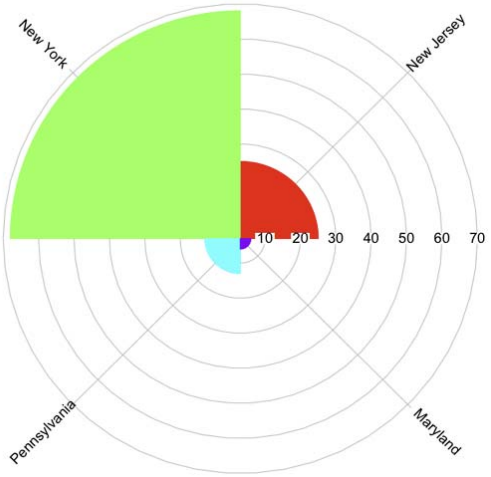


https://plot.ly/r/pie-charts/

fan.plot



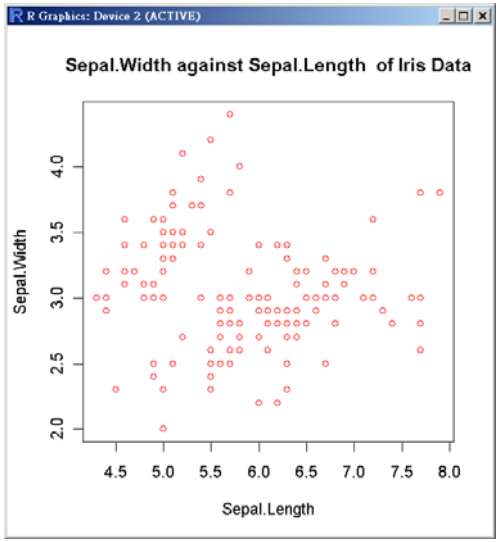
radial.pie



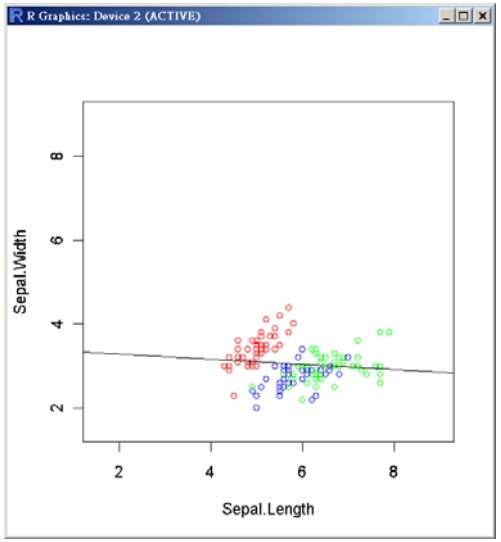
散佈圖 (Scatterplot)

`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)
```



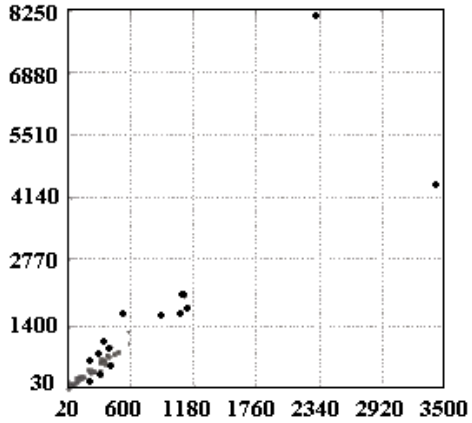

```
> range(x)
> range(y)
> plot(y~x, xlab=xlab, ylab=ylab, xlim=c(1.5,9),
     ylim=c(1.5,9), type="n")
> points(x[1:50], y[1:50], col="red")
> points(x[51:100], y[51:100], col="blue")
> points(x[101:150], y[101:150], col="green")
> abline(lm(y~x))
```



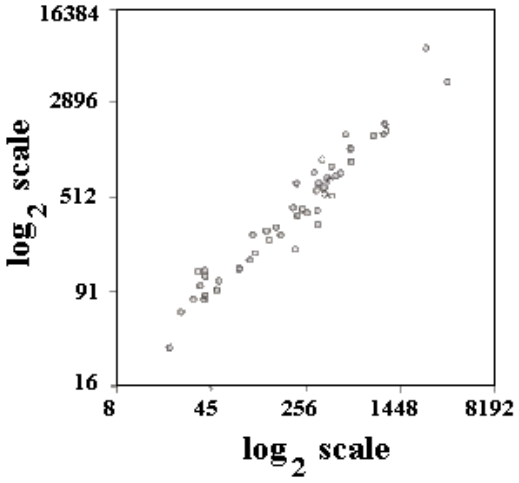
Extensions of Scatterplots

With a smoothing curve

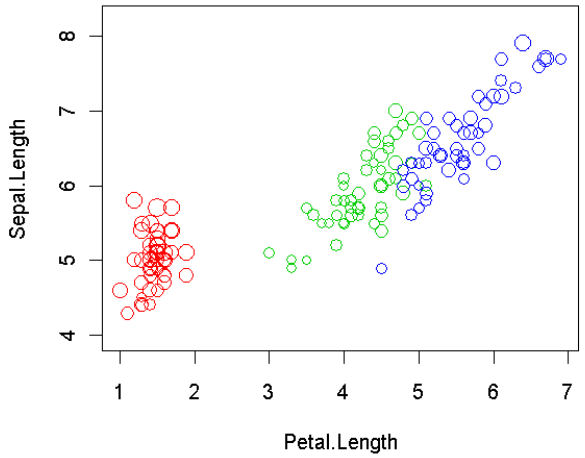
Poorly-Constructed



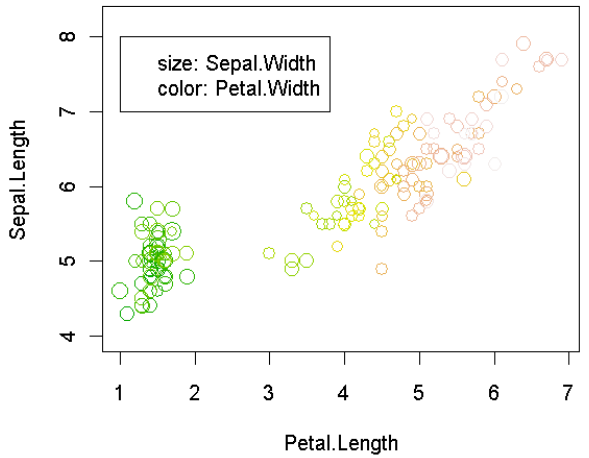
Better



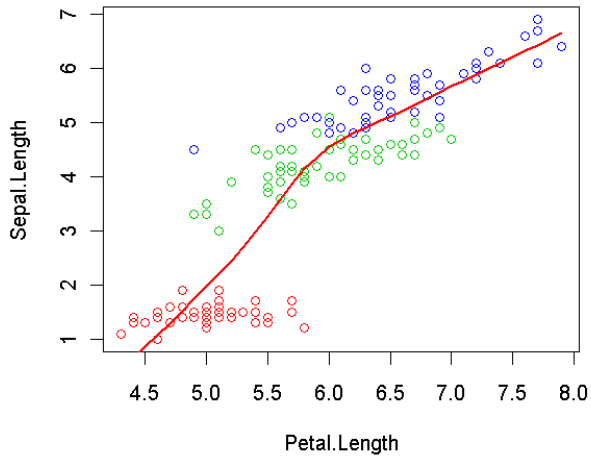
Bubbleplot: Sepal.Width



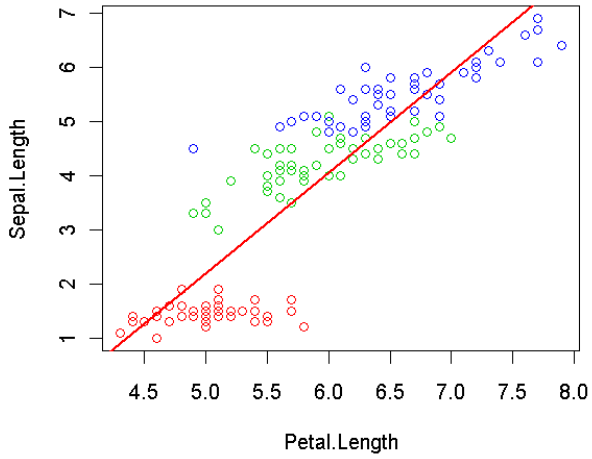
Color Bubbleplot



LOWESS

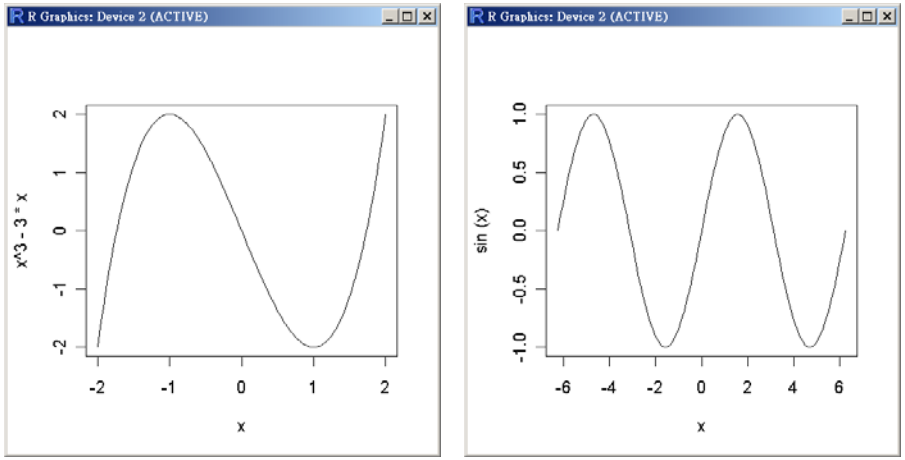


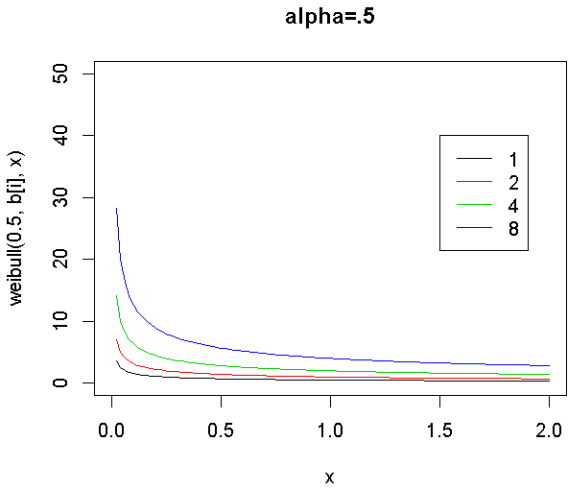
Simple Linear Regression



curve {graphics}: 畫2D數學函數圖

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





$$f(x;k,b) = bkx^{k-1}e^{-bx^k},$$

the hazard function is $h(x;k,b) = bkx^{k-1},$

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

The probability density function of a Weibull random variable is:[1]

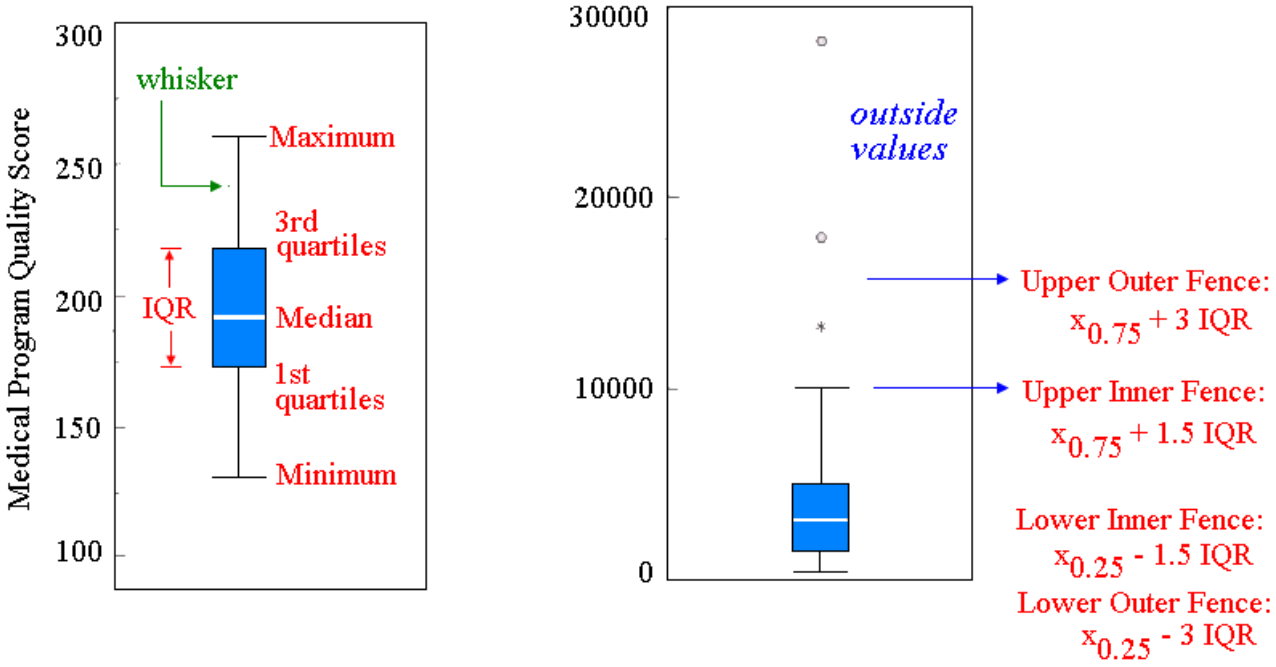
$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & x \geq 0, \\ 0, & x < 0, \end{cases}$$

where $k > 0$ is the *shape parameter* and $\lambda > 0$ is the *scale parameter* of the distribution. Its

```
> weibull <- function(alpha, beta, x){
+   alpha * beta * (x^(alpha-1))
+ }
>
> b <- c(1, 2, 4, 8)
> for(i in 1:length(b)) {
+   curve(weibull(0.5, b[i], x), from=0, to=2,
+         add=(i!=1),
+         col=i, ylim=c(0, 50), main="alpha=.5")
+ }
>
> legend(1.5, 40, legend=b, col=1:length(b), lty=1)
```

盒形圖 (Boxplot)

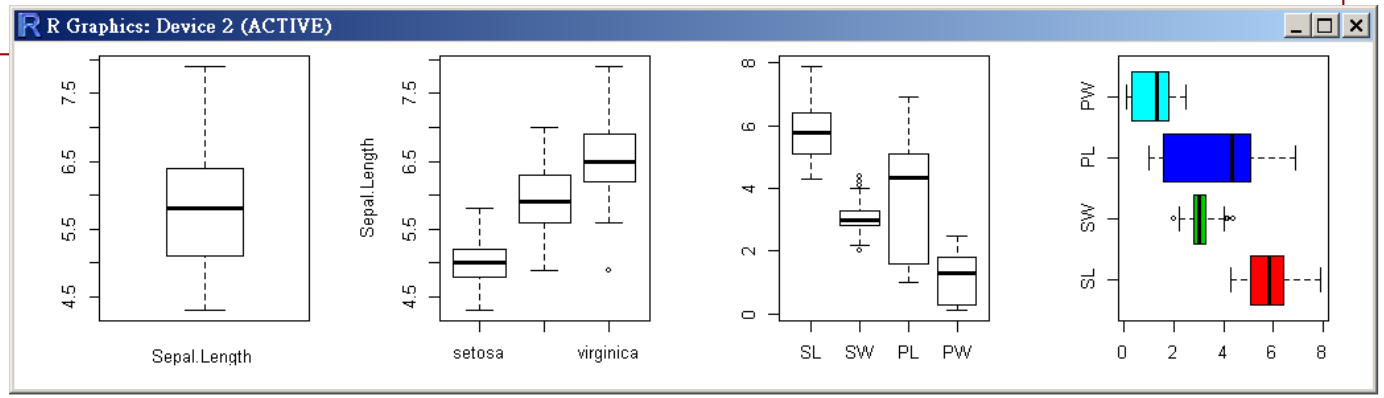
- Plotting with a categorical explanatory variable: boxplot.
- Categorical variables are factors with two or more levels.
- Boxplot
  - The horizontal line shows the median.
  - The bottom and top of the box show the 25th and 75th percentiles.
  - The vertical dashed lines are called the "whiskers".
  - Either maximum or 1.5 times the IQR.
- boxplot not only show the location and spread of data but also indicate skewness.



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

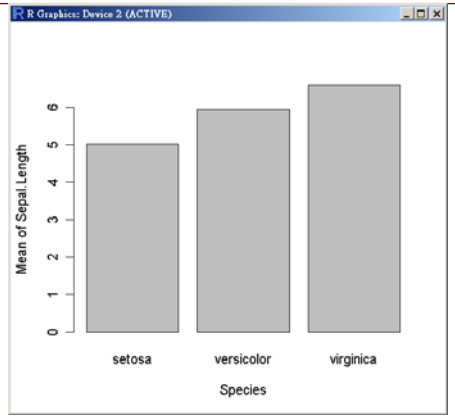


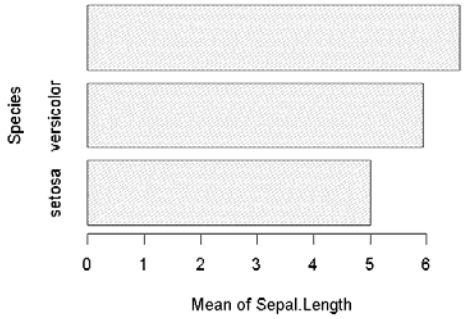
長條圖 (barplot)

```
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, ...)
```

```
> data(iris)
> 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, ylab="Species", xlab="Mean of Sepal.Length", density=20,
    horiz=TRUE)
```

```
> means
    setosa versicolor  virginica
     5.006      5.936      6.588
```

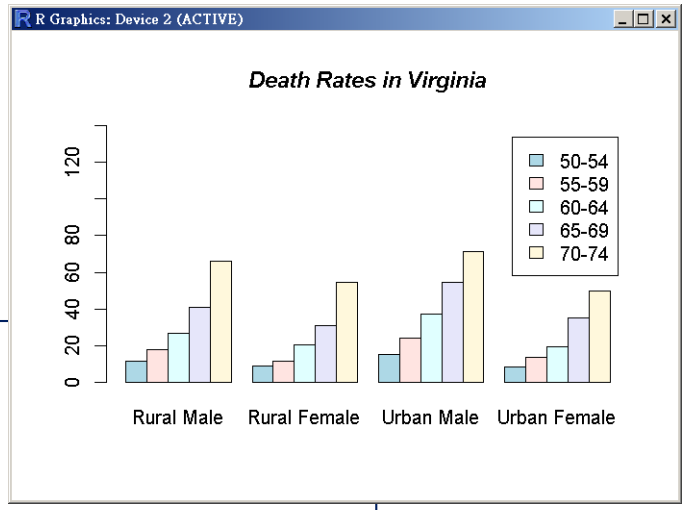




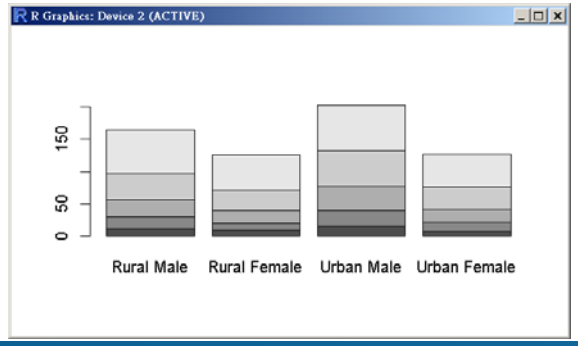
長條圖 (barplot)

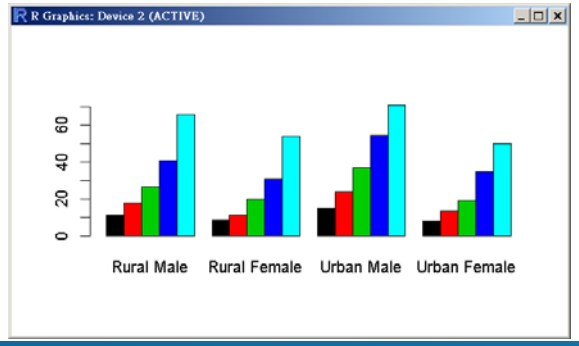
Death Rates in Virginia (1940): The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50–54, 55–59, 60–64, 65–69, 70–74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

```
> 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
```



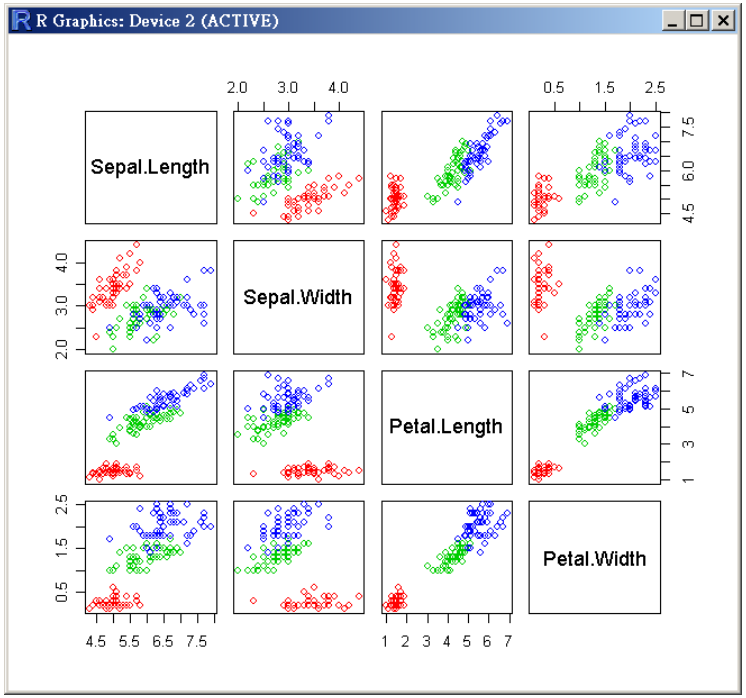

```
barplot(VADeaths)
barplot(VADeaths, beside = TRUE, col=1:5)

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)
```

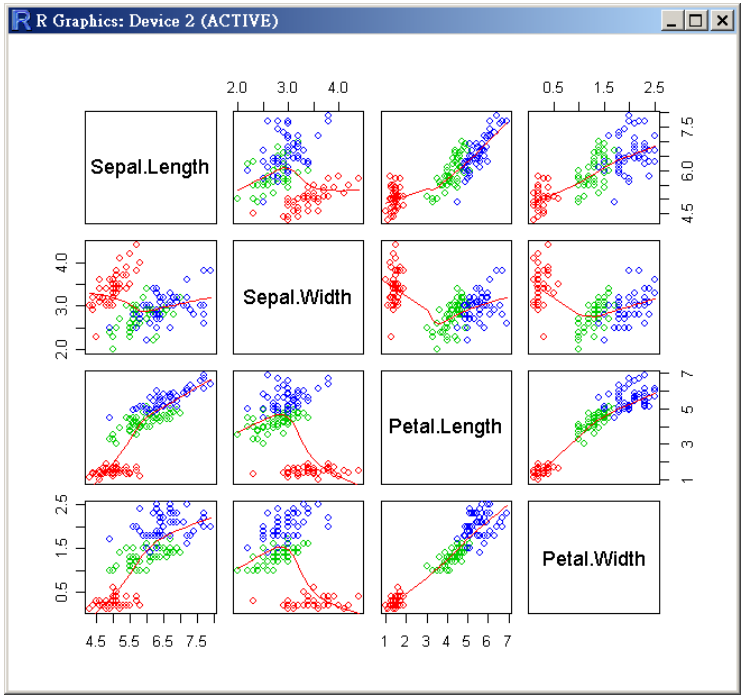




散佈圖矩陣 (Scatterplot Matrices)

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



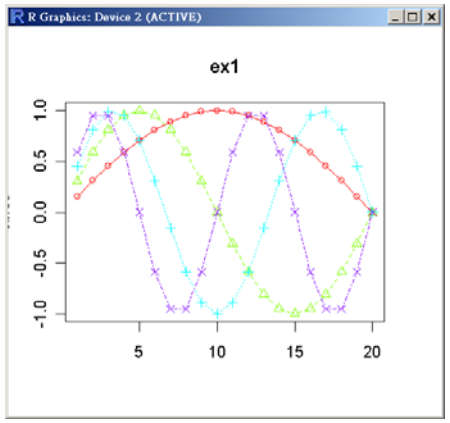


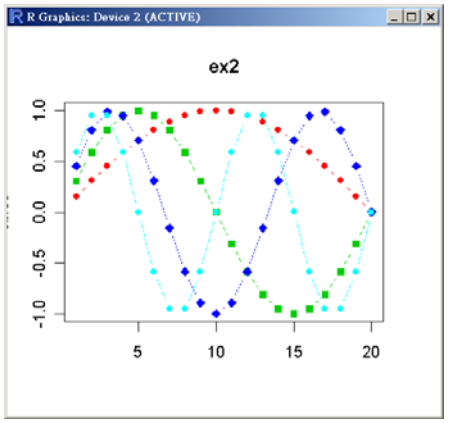
See also: conditioning plot (`coplot`)

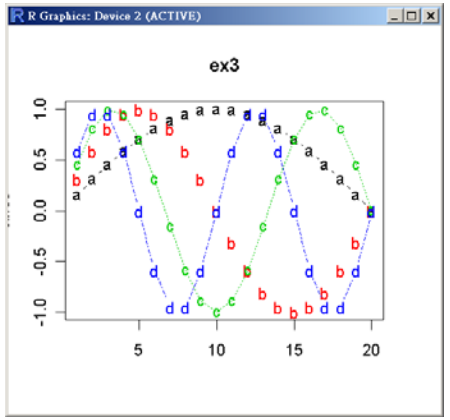
Plot Columns of Matrices

```
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
>
```







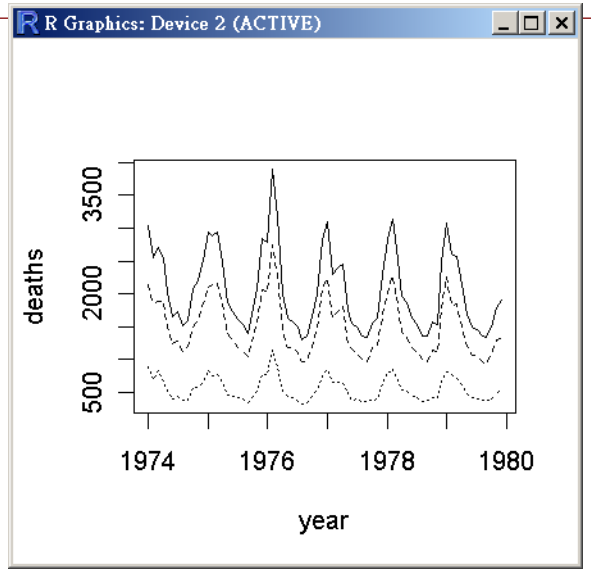
Exercise: adding legends?

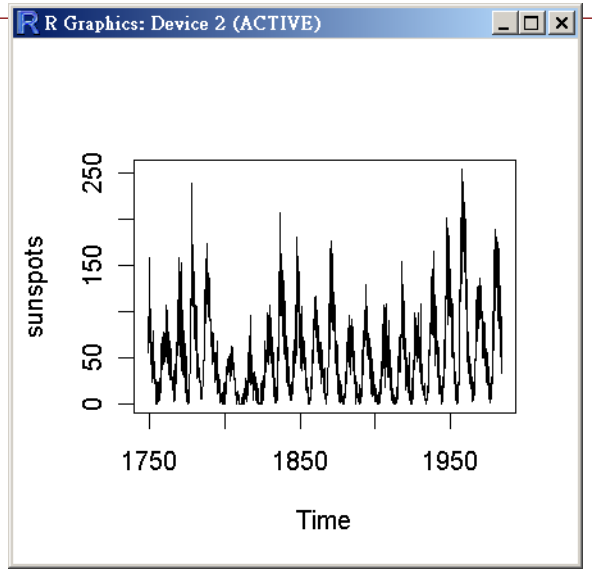
時間序列 (Time Series Plots)

- **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)
```

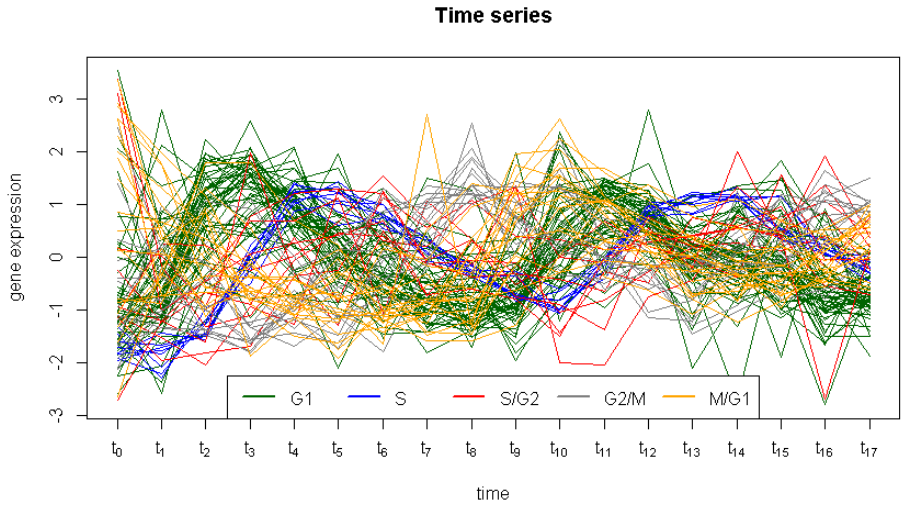




Time Series Plots

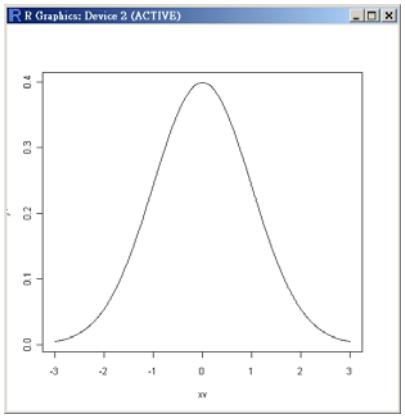
```
cell.raw <- read.table("trad_alpha103.txt", row.names=1, header=T)
head(cell.raw)
cell.xdata <- t(scale(t(cell.raw[,2:19]), center=T, scale=T))
y.C <-  as.integer(cell.raw[,1])
table(y.C)
no.cluster <- length(unique(y.C))
p <- ncol(cell.raw) - 1
cellcycle.color <- c("darkgreen", "blue", "red", "gray50", "orange")
ycolors <- cellcycle.color[y.C+1]
my.pch <- c(1:no.cluster)[y.C+1]
phase <- c("G1", "S", "S/G2", "G2/M", "M/G1")
matplot(t(cell.xdata), lty=1, type = "l", ylab="gene expression",
            col=ycolors, xlab="time", main="Time series", xaxt="n")
time.label <- parse(text=paste("t[",0:p,"]",sep=""))
axis(1, 1:(p+1), time.label)
legend("bottom", legend=phase, col=cellcycle.color, lty=1, horiz = T, lwd=2)
```

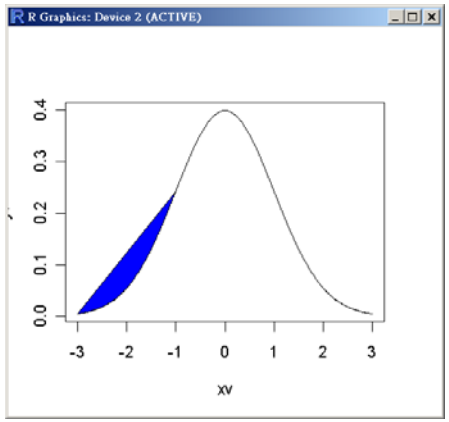
	A	B	C	D	E	F	G
1	UID	Phase	alpha0	alpha7	alpha14	alpha21	alpha28
2	YAR007C_G1	0	-0.48	-0.42	0.87	0.92	0.67
3	YBL035C_G1	0	-0.39	-0.58	1.08	1.21	0.52
4	YBR023C_G1	0	0.87	0.25	-0.17	0.18	-0.13
5	YBR067C_G1	0	1.57	1.03	1.22	0.31	0.16
6	YBR088C_G1	0	-1.15	-0.86	1.21	1.62	1.12
7	YBR278W_G1	0	0.04	-0.12	0.31	0.16	0.17
8	YCL055W_G1	0	2.95	0.45	-0.40	-0.66	-0.59
9	YDL003W_G1	0	-1.22	-0.74	1.34	1.50	0.63
10	YDL055C_G1	0	-0.73	-1.06	-0.79	-0.02	0.16
11	YDL102W_G1	0	-0.58	-0.40	0.13	0.58	-0.09
12	YDL164C_G1	0	-0.50	-0.42	0.66	1.05	0.68
13	YDL197C_G1	0	-0.86	-0.29	0.42	0.46	0.30
14	YDL227C_G1	0	-0.16	0.29	0.17	-0.28	-0.02
15	YDR052C_G1	0	-0.36	-0.03	-0.03	-0.08	-0.23

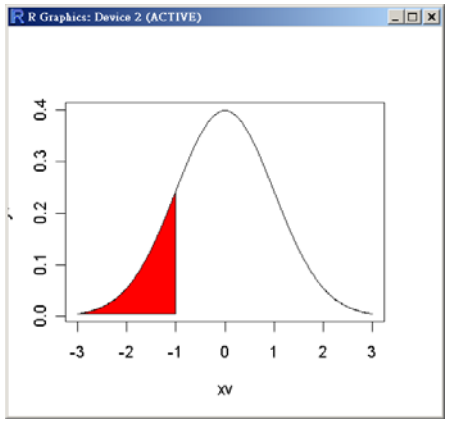


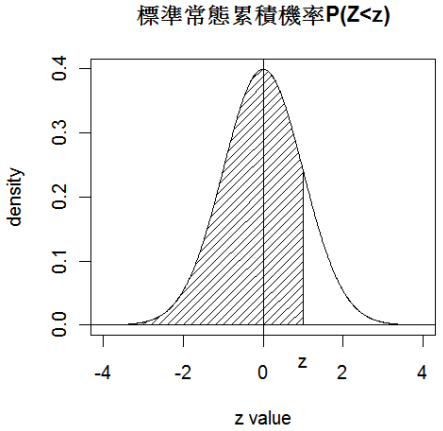
多邊形 (Polygon)

```
# draw a polygon using mouse
> plot(1)
> locations <- locator(6)
> polygon(locations, col="lavender")
```









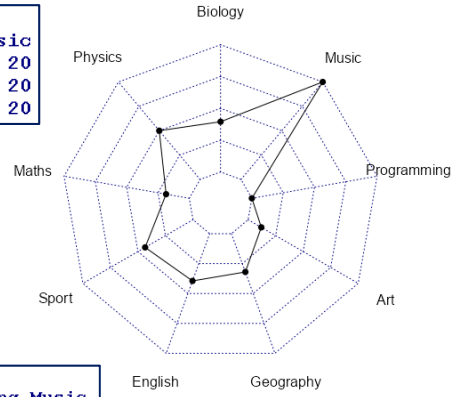

```
> xv <- seq(-3, 3, 0.01)
> yv <- dnorm(xv)
> plot(xv, yv, type="l")

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

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

Radar Plot using fmsb Package

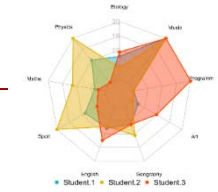
```
scores <- data.frame(
    row.names = c("Student.1", "Student.2", "Student.3"),
    Biology = c(7.9, 3.9, 9.4),
    Physics = c(10, 20, 0),
    Maths = c(3.7, 11.5, 2.5),
    Sport = c(8.7, 20, 4),
    English = c(7.9, 7.2, 12.4),
    Geography = c(6.4, 10.5, 6.5),
    Art = c(2.4, 0.2, 9.8),
    Programming = c(0, 0, 20),
    Music = c(20, 20, 20)
)
scores

max.min <- data.frame(
    Biology = c(20, 0), Physics = c(20, 0), Maths = c(20, 0),
    Sport = c(20, 0), English = c(20, 0), Geography = c(20, 0),
    Art = c(20, 0), Programming = c(20, 0), Music = c(20, 0)
)
rownames(max.min) <- c("Max", "Min")
max.min

mydata <- rbind(max.min, scores)
mydata

library(fmsb)
student1.data <- mydata[c("Max", "Min", "Student.1"), ]
student1.data
radarchart(student1.data)
```

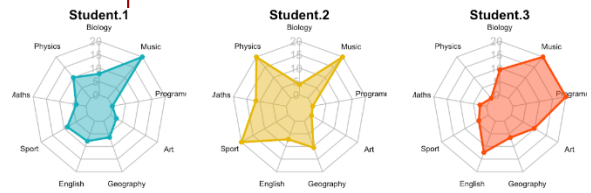
- library(fmsb)
- The data should be organized as follow:
- The row 1 must contain the maximum values for each variable
- The row 2 must contain the minimum values for each variable
- Data for cases or individuals should be given starting from row 3
- The number of columns or variables must be more than 2.

```
> scores
          Biology Physics Maths Sport English Geography Art Programming Music
Student.1     7.9      10   3.7   8.7     7.9       6.4 2.4           0    20
Student.2     3.9      20  11.5  20.0     7.2      10.5 0.2           0    20
Student.3     9.4       0   2.5   4.0    12.4       6.5 9.8          20    20
```




```
> mydata
          Biology Physics Maths Sport English Geography  Art Programming Music
Max          20.0      20  20.0  20.0    20.0      20.0 20.0          20    20
Min           0.0       0   0.0   0.0     0.0       0.0  0.0           0     0
Student.1     7.9      10   3.7   8.7     7.9       6.4  2.4           0    20
Student.2     3.9      20  11.5  20.0     7.2      10.5  0.2           0    20
Student.3     9.4       0   2.5   4.0    12.4       6.5  9.8          20    20
```



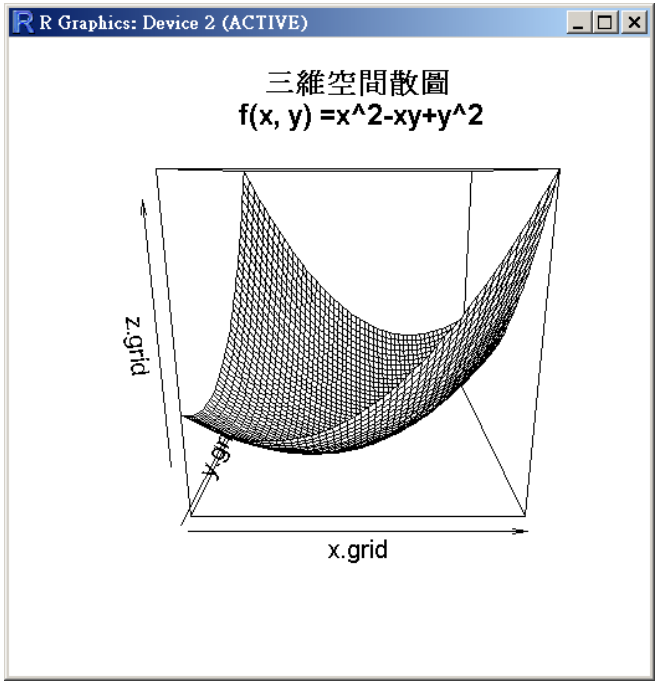


Beautiful Radar Chart in R using FMSB and GGPlot Packages
https://www.datanovia.com/en/blog/beautiful-radar-chart-in-r-using-fmsb-and-ggplot-packages/

3D透視圖: persp

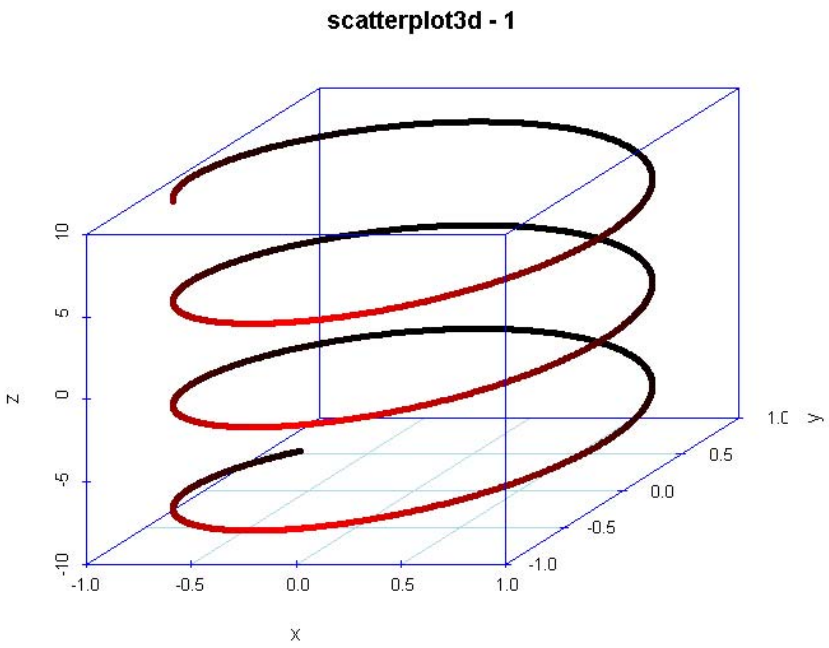
- 雙變數函數在三維空間的散佈圖
- NOTE: x-y平面上，格點數目愈多，散佈圖愈密集。
- plot f(x, y) =x^2-xy+y^2。

```
> 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)
```

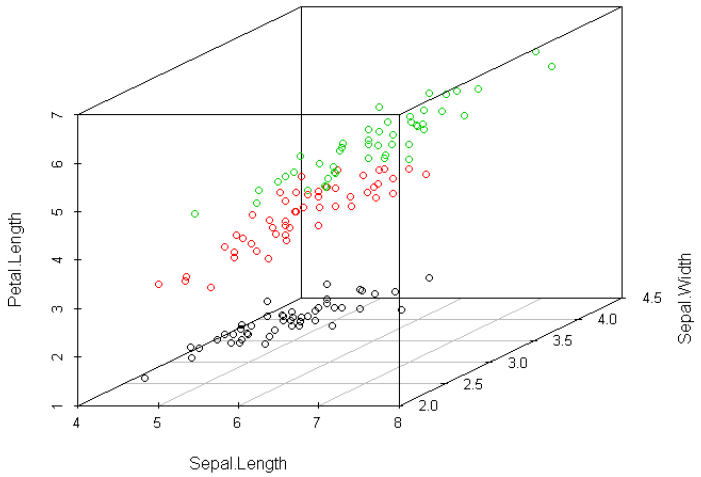


3D散佈圖

```
> 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)
```



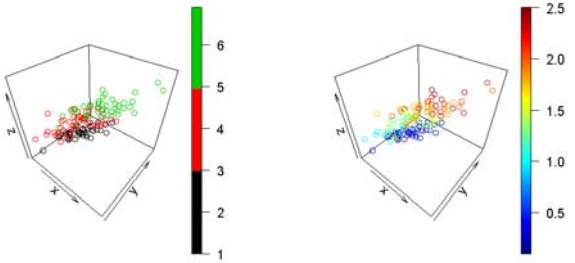

```
> scatterplot3d(iris[,1:3],
   color=as.integer(iris[,5]))
```



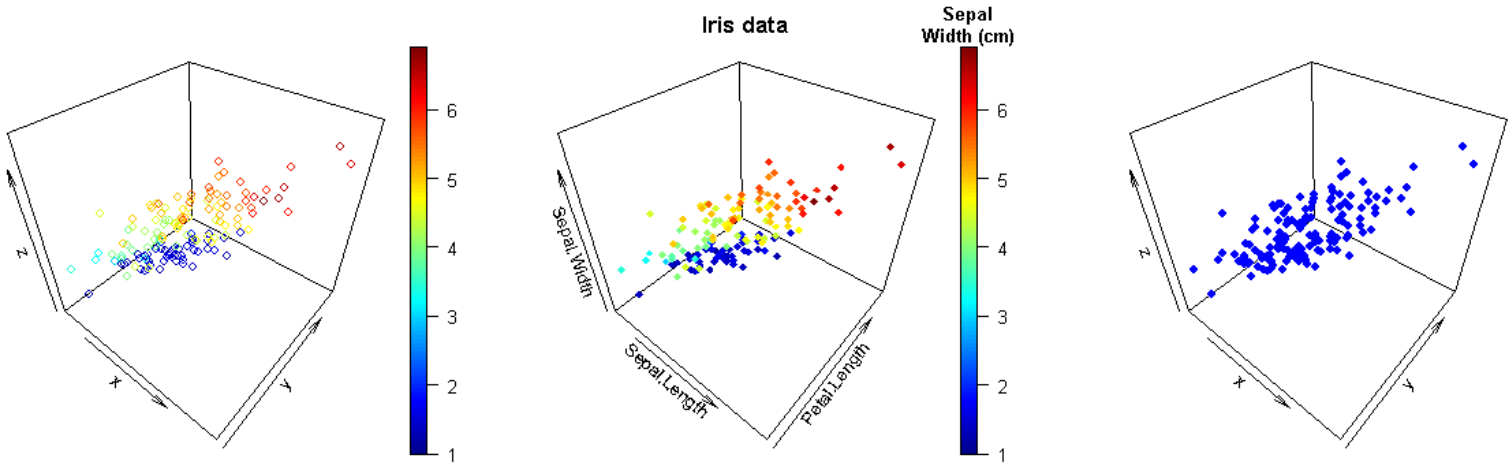
plot3D package

```
install.packages("plot3D")
library("plot3D")
```

```
> scatter3D(x, y, z, col = as.integer(s))
> scatter3D(x, y, z, colvar = w)
```



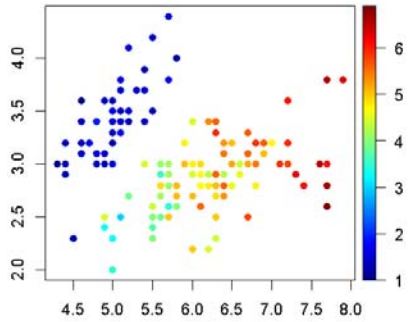
```
x <- iris$Sepal.Length
y <- iris$Sepal.Width
z <- iris$Petal.Length
w <- iris$Petal.Width
s <- iris$Species

par(mfrow = c(1,3), mai = c(0.3, 0.3, 0.3, 0.3))
scatter3D(x, y, z)
scatter3D(x, y, z, pch = 18, clab = c("Sepal", "Width (cm)"), main = "Iris data",
          xlab = "Sepal.Length", zlab = "Petal.Length", ylab = "Sepal.Width")
scatter3D(x, y, z, colvar = as.integer(s), col = "blue", pch = 19, cex = 1)
```

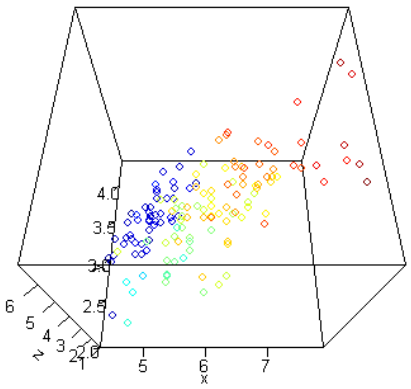


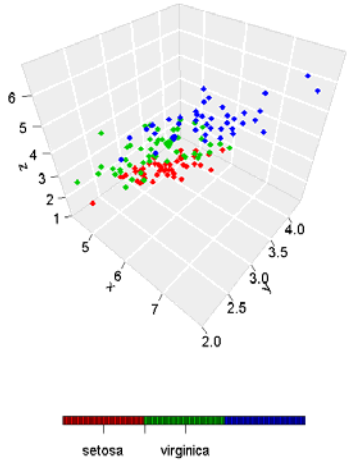
http://www.sthda.com/english/wiki/impressive-package-for-3d-and-4d-graph-r-software-and-data-visualization

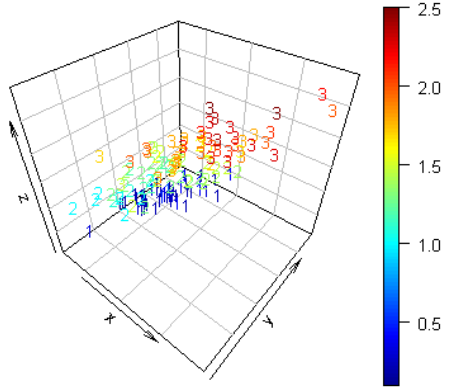
plot3D package

```
par(mfrow = c(1,3), mai = c(0.3, 0.3, 0.2, 0.3))
# grey background with white grid lines
scatter3D(x, y, z, bty = "f", colkey = FALSE, ticktype = "detailed",
          theta=0, phi=60)
# theta: the azimuthal direction, phi: the co-latitude.
scatter3D(x, y, z, bty = "g", pch = 18,
          col = as.integer(s) + 1,
          ticktype = "detailed",
          colkey = list(at = c(2, 3, 4), side = 1,
                        addlines = TRUE, length = 0.5, width = 0.5,
                        labels = c("setosa", "versicolor", "virginica"),,
text3D(x, y, z, labels = as.integer(s), colvar = w, bty = "b2")
# "b2": back panels and grid lines are visible

# hist3D, text3D
# scatter2D(x, y, colvar = z, pch = 16)
```







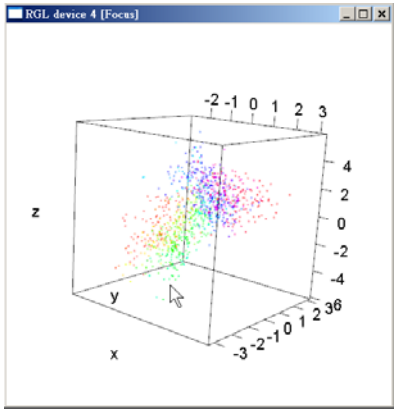


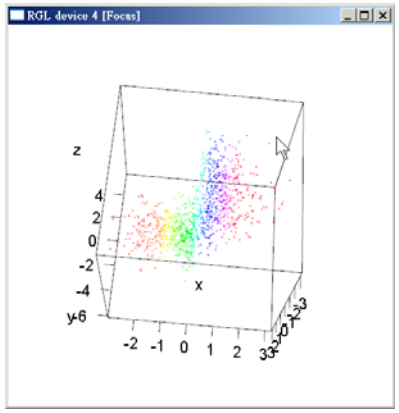
More examples: http://www.sthda.com/english/wiki/impressive-package-for-3d-and-4d-graph-r-software-and-data-visualization

3D visualization device system (OpenGL)

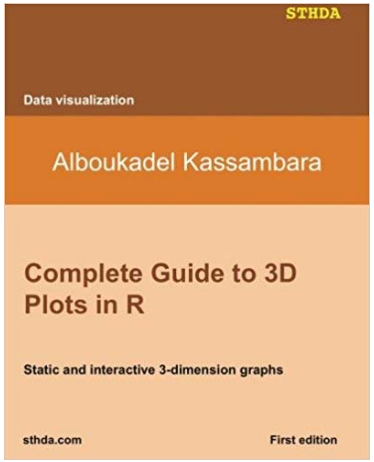
```
> library(rgl)
> demo(rgl)
> open3d()
> x <- sort(rnorm(1000))
> y <- rnorm(1000)
> z <- rnorm(1000) + atan2(x,y)
> plot3d(x, y, z, col=rainbow(1000), size=2)
>
> M <- par3d("userMatrix")
> play3d(par3dinterp(userMatrix=list(M, rotate3d(M, pi/2, 1, 0, 0),
+                                    rotate3d(M, pi/2, 0, 1, 0)),
+                                    # 90 degree rotation about the x axis
+                    duration=4)
```

```
> M
             [,1]       [,2]       [,3] [,4]
[1,] -0.98849827 -0.1478247 -0.0319229    0
[2,] -0.01036166 -0.1443882  0.9894670    0
[3,] -0.15087697  0.9784170  0.1411958    0
[4,]  0.00000000  0.0000000  0.0000000    1
```





- 截取靜態2d圖: `rgl.postscript {rgl}`, 或 `rgl.snapshot {rgl}`.
- 存出動態圖: `writeWebGL {rgl}`.



STHDA: rgl

http://www.sthda.com/english/wiki/a-complete-guide-to-3d-visualization-device-system-in-r-r-software-and-data-visualization

http://www.sthda.com/english/download/3-ebooks/6-complete-guide-to-3d-plots-in-r/

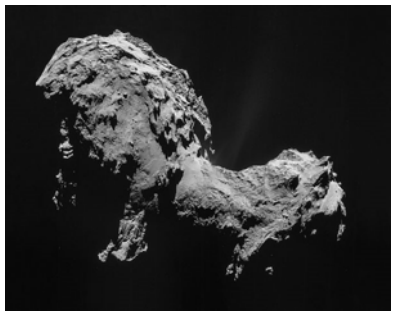
範例: rgl, explore a comet

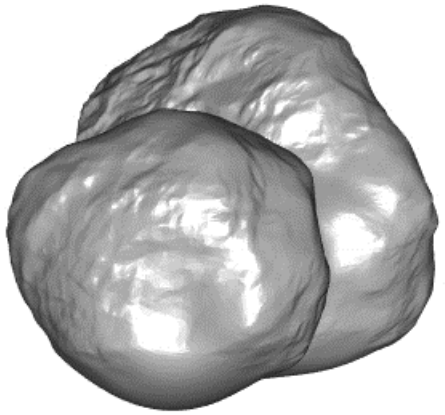
Explore a comet with R's "rgl" package

December 24, 2014

http://blog.revolutionanalytics.com/2014/12/explore-a-comet-with-rs-rgl-package.html

"Last month, the Philae lander touched down on comet Churyumov–Gerasimenko. In the process, the lander and the orbiting Rosetta probe captured detailed data on the geometry of the comet, which the ESA published as a shape file. ..."





https://en.wikipedia.org/wiki/67P/Churyumov%E2%80%93Gerasimenko

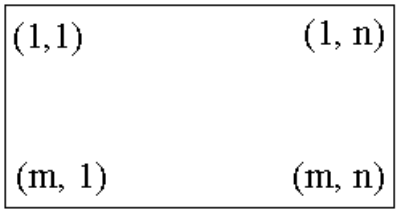
```
> open3d()
> # comet <- readOBJ(url("http://sci.esa.int/science-e/www/object/doc.cfm?fobjectid=54726"))
> comet <- readOBJ("ESA_Rosetta_OSIRIS_67P_SHAP2P.obj")
> class(comet)
[1] "mesh3d"  "shape3d"
> str(comet)
List of 6
 $ vb           : num [1:4, 1:31456] -0.394 0.402 0.443 1 -0.163 ...
 $ it           : num [1:3, 1:62908] 14327 6959 18747 8258 15598 ...
 $ primitivetype: chr "triangle"
 $ material     : NULL
 $ normals      : NULL
 $ texcoords    : NULL
 - attr(*, "class")= chr [1:2] "mesh3d" "shape3d"
> shade3d(comet, col="gray")
```

```
# it: indices for triangular faces
# ib: indices for quad faces
# vb: matrix of vertices: 4xn matrix (rows
x, y, z, h) or equivalent vector, where h
indicates scaling of each plotted quad
```

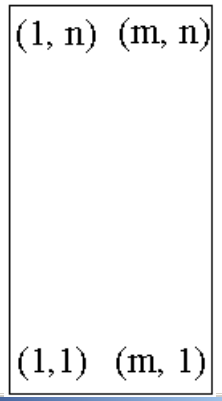
Display a Color Image

- `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

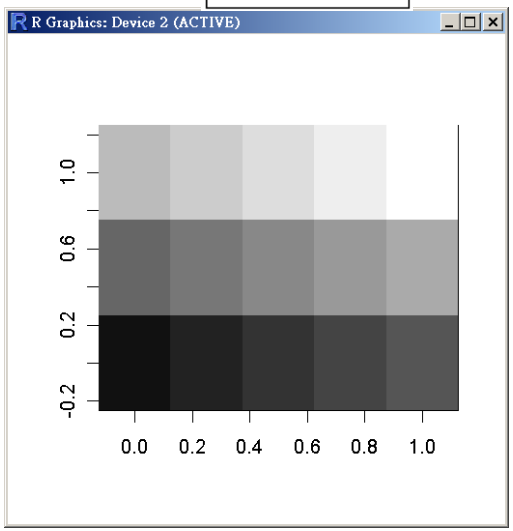


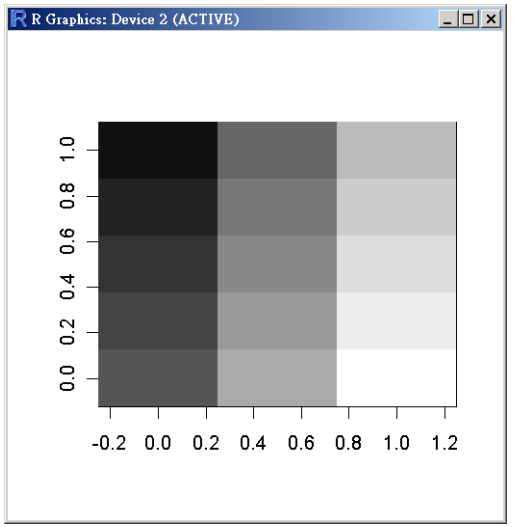
Image




```
> 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))
```

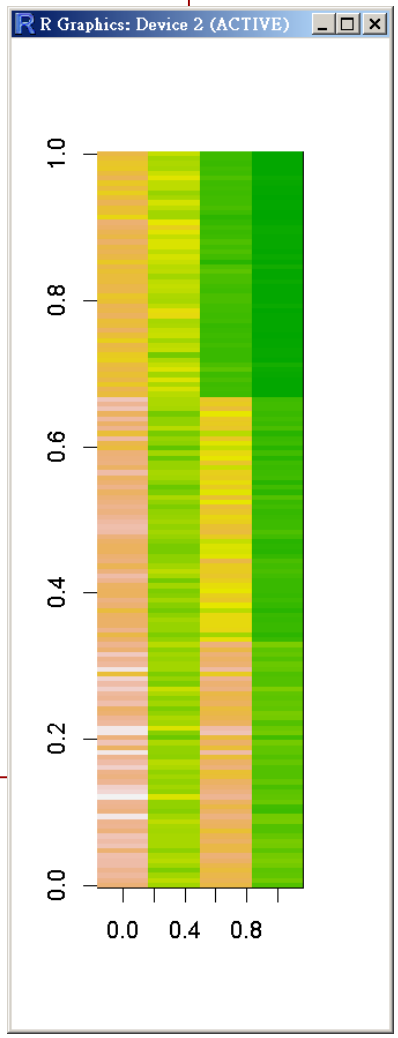
```
> my.data
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```





練習

```
> image(t(as.matrix(iris[,1:4]))[,150:1], col=terrain.colors(100))
> head(iris[,1:4])
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
> tail(iris[,1:4])
    Sepal.Length Sepal.Width Petal.Length Petal.Width
145          6.7         3.3          5.7         2.5
146          6.7         3.0          5.2         2.3
147          6.3         2.5          5.0         1.9
148          6.5         3.0          5.2         2.0
149          6.2         3.4          5.4         2.3
150          5.9         3.0          5.1         1.8
```

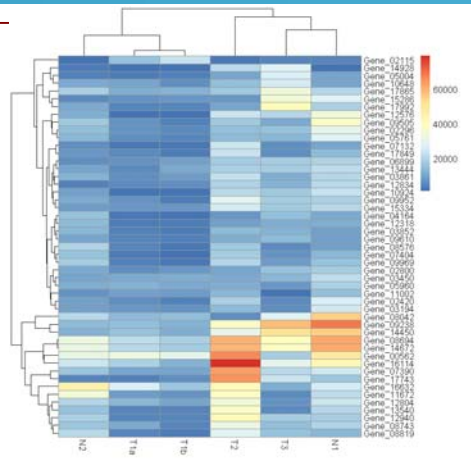


pheatmap: Pretty Heatmaps

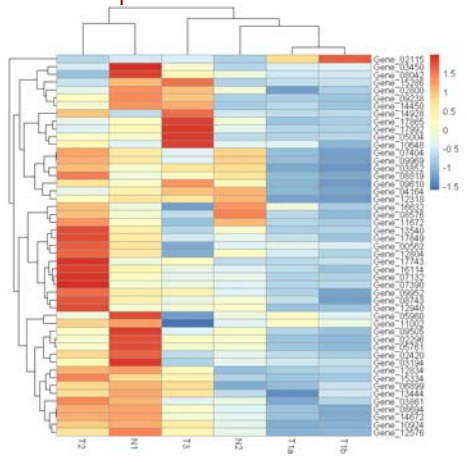
Implementation of heatmaps that offers more control over dimensions and appearance.

選讀

```
> library(pheatmap)
> DESeq_subset <- read.csv("DESeq_subset.csv")
> dim(DESeq_subset)
[1] 49  7
> head(DESeq_subset)
          X   T1a   T1b    T2    T3    N1    N2
1 Gene_00562 32314 29693 66140 17973 47994 30878
2 Gene_02115 15261 23301  1944  4578  4087  1072
...
6 Gene_03194  7611  6806 13506  5727 25020  9235
> DESeq.X <- as.matrix(DESeq_subset[,2:ncol(DESeq_subset)])
> colnames(DESeq.X)
[1] "T1a" "T1b" "T2"  "T3"  "N1"  "N2"
> rownames(DESeq.X) <- DESeq_subset[,1]
> dimnames(DESeq.X)
[[1]]
 [1] "Gene_00562" "Gene_02115" "Gene_02296" "Gene_02420" ...
[46] "Gene_17743" "Gene_17849" "Gene_17865" "Gene_17992"

[[2]]
[1] "T1a" "T1b" "T2"  "T3"  "N1"  "N2"

> str(DESeq.X)
 int [1:49, 1:6] 32314 15261 6730 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:49] "Gene_00562"...
  ..$ : chr [1:6] "T1a" "T1b" "T2" "T3" ...
> pheatmap(DESeq.X)
```



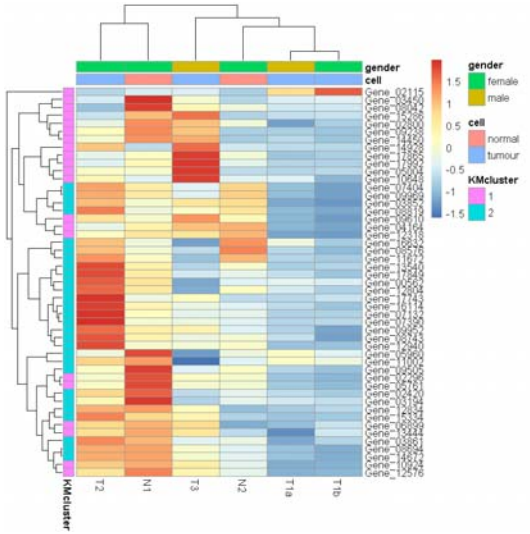
```
> DESeq.X.std <- t(apply(DESeq.X, 1, scale))
> class(DESeq.X.std)a
[1] "matrix"
> dimnames(DESeq.X.std) <- dimnames(DESeq.X)
> pheatmap(DESeq.X.std) # note the color spectrum
```

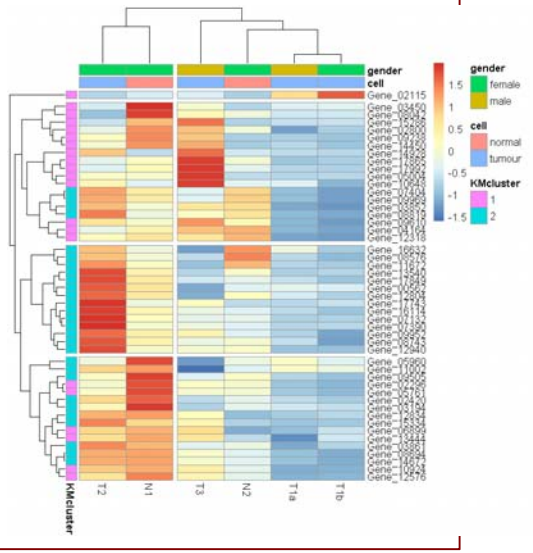


Making a heatmap in R with the pheatmap package
https://davetang.org/muse/2018/05/15/making-a-heatmap-in-r-with-the-pheatmap-package/

pheatmap: Pretty Heatmaps

```
> sample.group <- data.frame(cell = rep(c("tumour", "normal"), c(4,2)),
+                            gender = sample(c("male", "female"), 6, replace=T))
> row.names(sample.group) <- colnames(DESeq.X)
> sample.group
      cell gender
T1a tumour   male
T1b tumour female
T2  tumour female
T3  tumour   male
N1  normal female
N2  normal female
> km <- as.character(kmeans(DESeq.X.std, 2)$cluster
> gene.cluster <- data.frame(KMcluster = km)
> row.names(gene.cluster) <- rownames(DESeq.X)
> head(gene.cluster)
           KMcluster
Gene_00562         2
Gene_02115         1
Gene_02296         1
Gene_02420         2
Gene_02800         1
Gene_03194         2
>
> pheatmap(DESeq.X.std,
+          annotation_row = gene.cluster,
+          annotation_col = sample.group)
>
> pheatmap(DESeq.X.std,
+          annotation_row = gene.cluster,
+          annotation_col = sample.group,
+          cutree_rows = 4,
+          cutree_cols = 2)
```



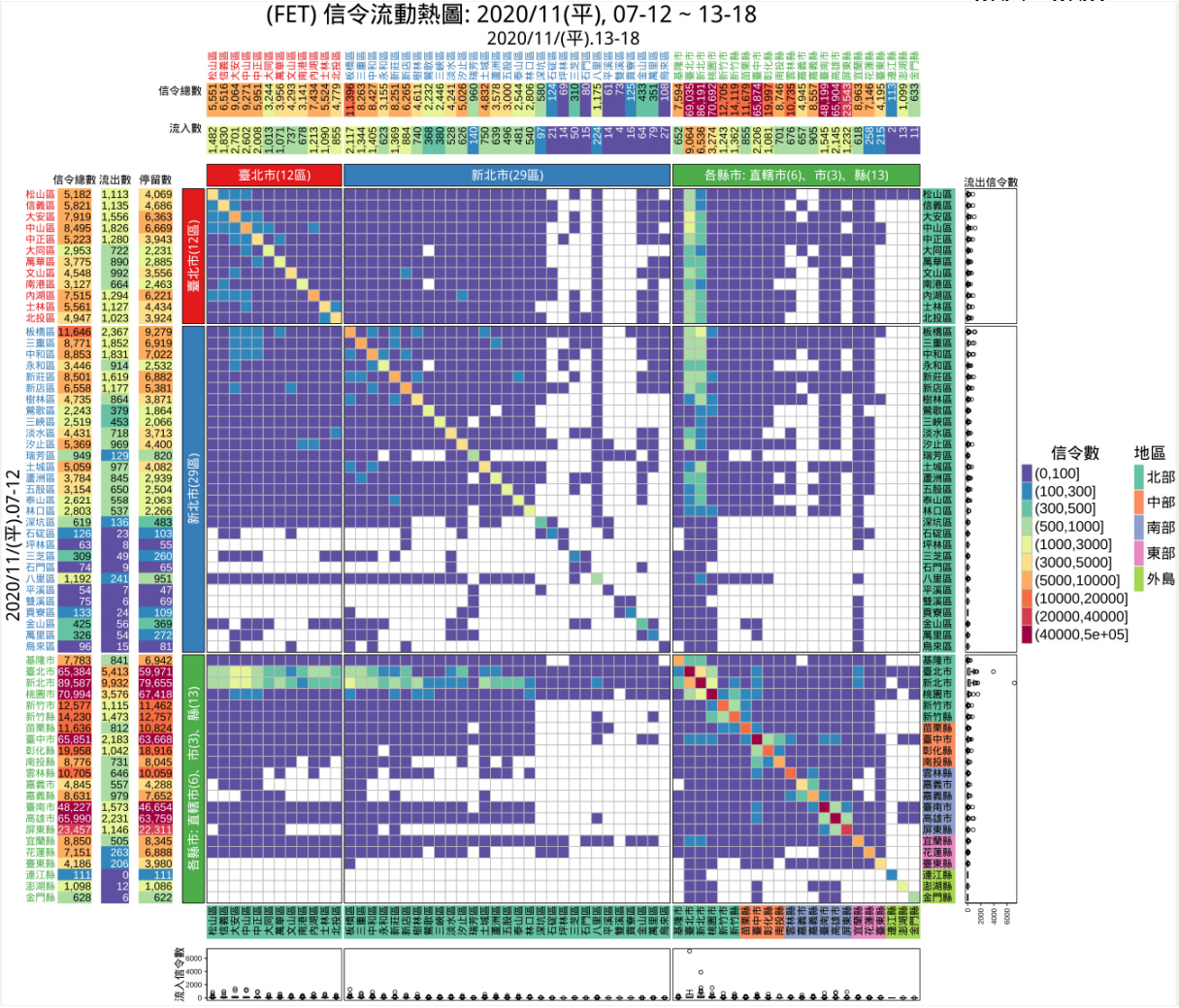


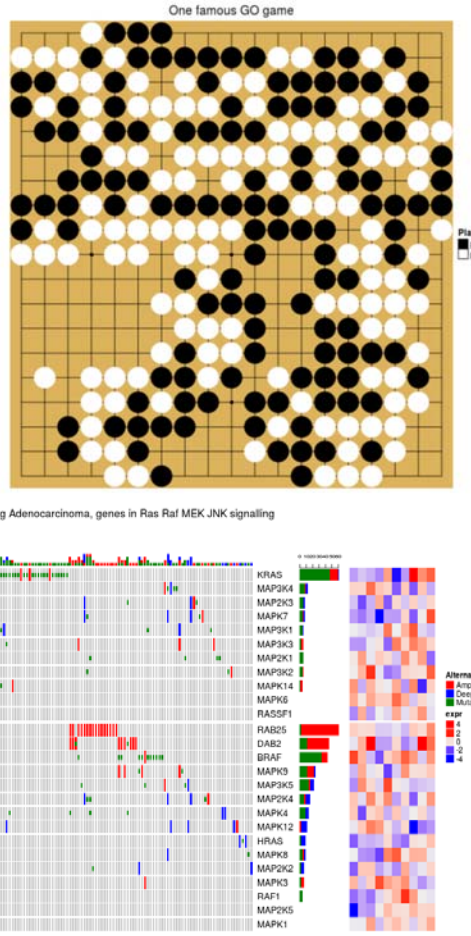
ComplexHeatmap

https://jokergoo.github.io/ComplexHeatmap-reference/book/

http://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html

Zuguang Gu, Roland Eils, Matthias Schlesner, Complex heatmaps reveal patterns and correlations in multidimensional genomic data, Bioinformatics, Volume 32, Issue 18, 15 September 2016, Pages 2847–2849

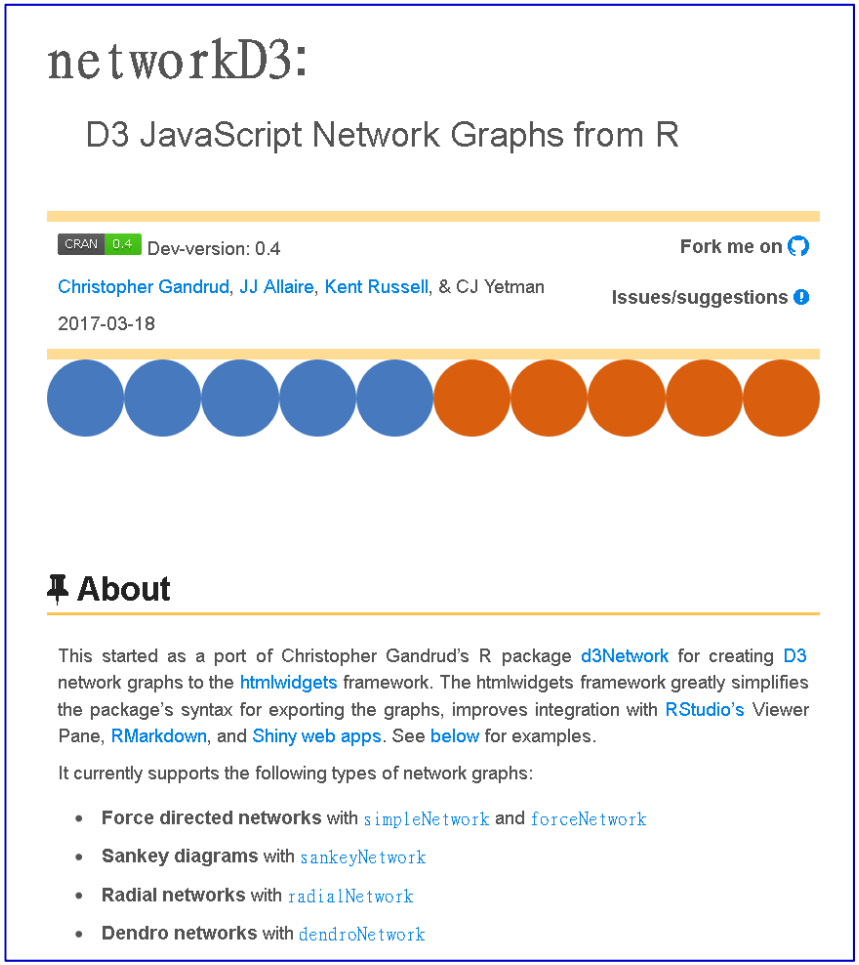




visualize multiple genomic alteration events by heatmap

networkD3:
D3 JavaScript Network Graphs from R

https://christophergandrud.github.io/networkD3/


networkD3:

D3 JavaScript Network Graphs from R

CRAN 0.4 Dev-version: 0.4

Fork me on

Christopher Gandrud, JJ Allaire, Kent Russell, & CJ Yetman

Issues/suggestions

2017-03-18

About

This started as a port of Christopher Gandrud's R package d3Network for creating D3 network graphs to the htmlwidgets framework. The htmlwidgets framework greatly simplifies the package's syntax for exporting the graphs, improves integration with RStudio's Viewer Pane, RMarkdown, and Shiny web apps. See below for examples.

It currently supports the following types of network graphs:

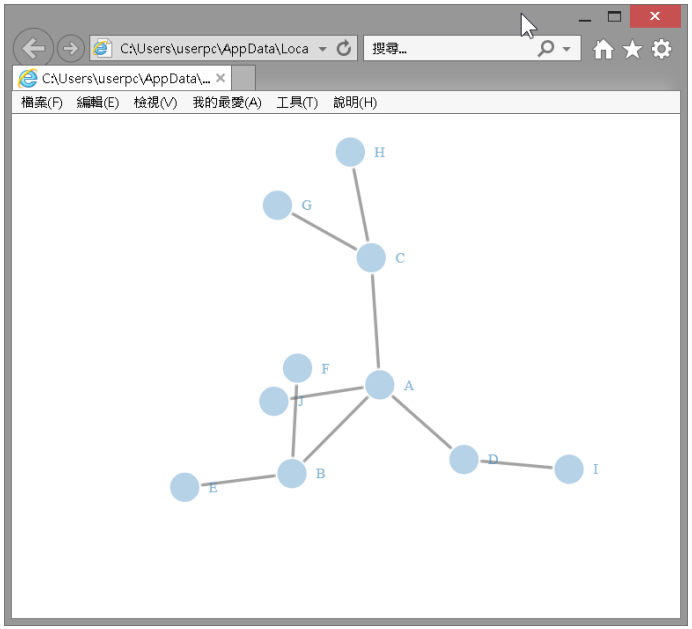
- **Force directed networks** with simpleNetwork and forceNetwork
- **Sankey diagrams** with sankeyNetwork
- **Radial networks** with radialNetwork
- **Dendro networks** with dendroNetwork

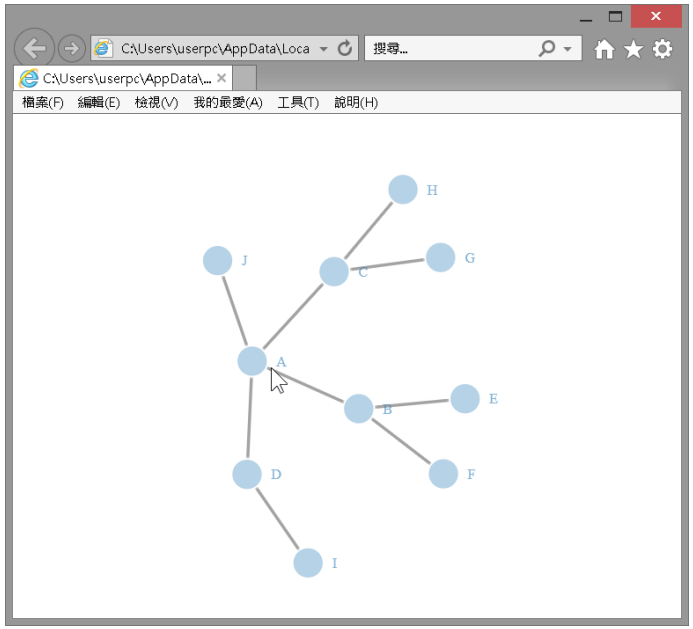
- **networkD3** supports the following types of network graphs:
  - Force directed networks with **simpleNetwork** and **forceNetwork**
  - Sankey diagrams with **sankeyNetwork**
  - Radial networks with **radialNetwork**
  - Dendro networks with **dendroNetwork**

simpleNetwork

```
> library(networkD3)
> src <- c("A", "A", "A", "A", "B", "B", "C", "C", "D")
> target <- c("B", "C", "D", "J", "E", "F", "G", "H", "I")
> networkData <- data.frame(src, target)
> simpleNetwork(networkData)
```

```
> networkData
  src target
1   A      B
2   A      C
3   A      D
4   A      J
5   B      E
6   B      F
7   C      G
8   C      H
9   D      I
```

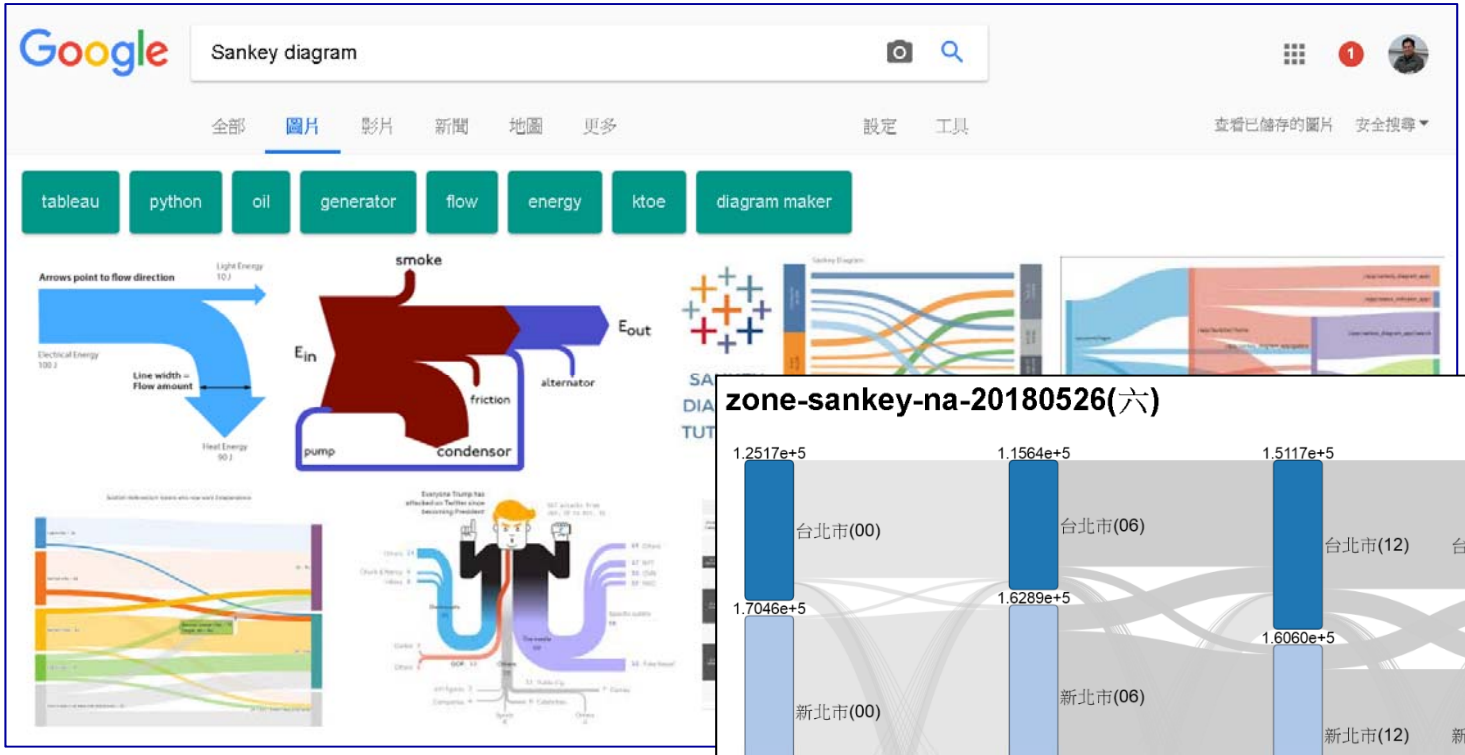


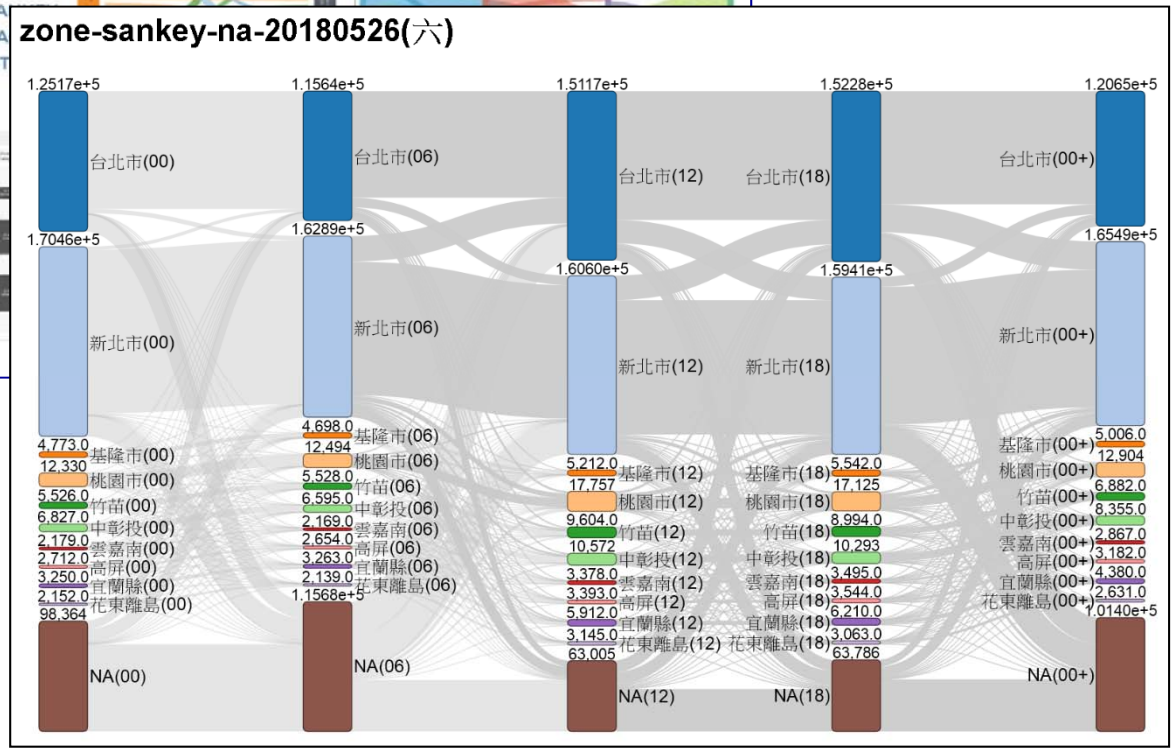


https://christophergandrud.github.io/networkD3/

桑基圖 (Sankey Diagram)

http://www.sankey-diagrams.com/





sankeyNetwork {networkD3}:
Create a D3 JavaScript Sankey diagram

Usage

```
sankeyNetwork(Links, Nodes, Source, Target, Value, NodeID, NodeGroup = NodeID,
  LinkGroup = NULL, units = "",
  colourScale = JS("d3.scaleOrdinal(d3.schemeCategory20);"), fontSize = 7,
  fontFamily = NULL, nodeWidth = 15, nodePadding = 10, margin = NULL,
  height = NULL, width = NULL, iterations = 32, sinksRight = TRUE)
```

Arguments

- **Links**: a data frame object with the links between the nodes. It should have include the Source and Target for each link. An optional Value variable can be included to specify how close the nodes are to one another.
- **Nodes**: a data frame containing the node id and properties of the nodes. If no ID is specified then the nodes must be in the same order as the Source variable column in the Links data frame. Currently only grouping variable is allowed.
- **Source**: character string naming the network source variable in the Links data frame.
- **Target**: character string naming the network target variable in the Links data frame.
- **Value**: character string naming the variable in the Links data frame for how far away the nodes are from one another.
- **NodeID**: character string specifying the node IDs in the Nodes. data frame. Must be 0-indexed.
- **NodeGroup**: character string specifying the node groups in the Nodes. Used to color the nodes in the network.
- **LinkGroup**: character string specifying the groups in the Links. Used to color the links in the network.

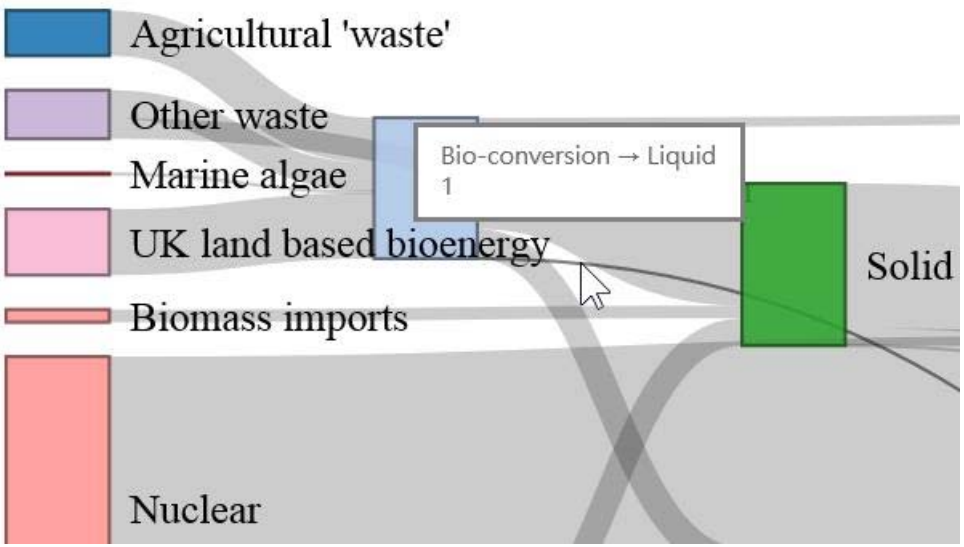
sankeyNetwork {networkD3}:
Create a D3 JavaScript Sankey diagram

```
> URL <- paste0(
+         "https://cdn.rawgit.com/christophergandrud/networkD3/",
+         "master/JSONdata/energy.json")
> Energy <- jsonlite::fromJSON(URL)
> str(Energy)
List of 2
 $ nodes:'data.frame':  48 obs. of  1 variable:
  ..$ name: chr [1:48] "Agricultural 'waste'" "Bio-conversion" "Liquid" "Losses" ...
 $ links:'data.frame':  68 obs. of  3 variables:
  ..$ source: int [1:68] 0 1 1 1 1 6 7 8 10 9 ...
  ..$ target: int [1:68] 1 2 3 4 5 2 4 9 9 4 ...
  ..$ value : num [1:68] 124.729 0.597 26.862 280.322 81.144 ...
> lapply(Energy, head)
$nodes
                  name
1 Agricultural 'waste'
2       Bio-conversion
3               Liquid
4               Losses
5                Solid
6                  Gas

$links
  source target   value
1      0      1 124.729
2      1      2   0.597
3      1      3  26.862
4      1      4 280.322
5      1      5  81.144
6      6      2  35.000
```

```
> sankeyNetwork(Links = Energy$links, Nodes = Energy$nodes,
+               Source = "source", Target = "target",
+               Value = "value", NodeID = "name",
+               fontSize = 12, nodeWidth = 30)
```

sankeyNetwork

```
> Energy
$nodes
                                  name
1                 Agricultural 'waste'
2                      Bio-conversion
3                              Liquid
4                              Losses
5                               Solid
6                                 Gas
7                      Biofuel imports
8                      Biomass imports
...
46              UK land based bioenergy
47                                Wave
48                                Wind

$links
   source target   value
1       0      1 124.729
2       1      2   0.597
3       1      3  26.862
4       1      4 280.322
5       1      5  81.144
6       6      2  35.000
7       7      4  35.000
...
67     46     15  19.013
68     47     15 289.366
```

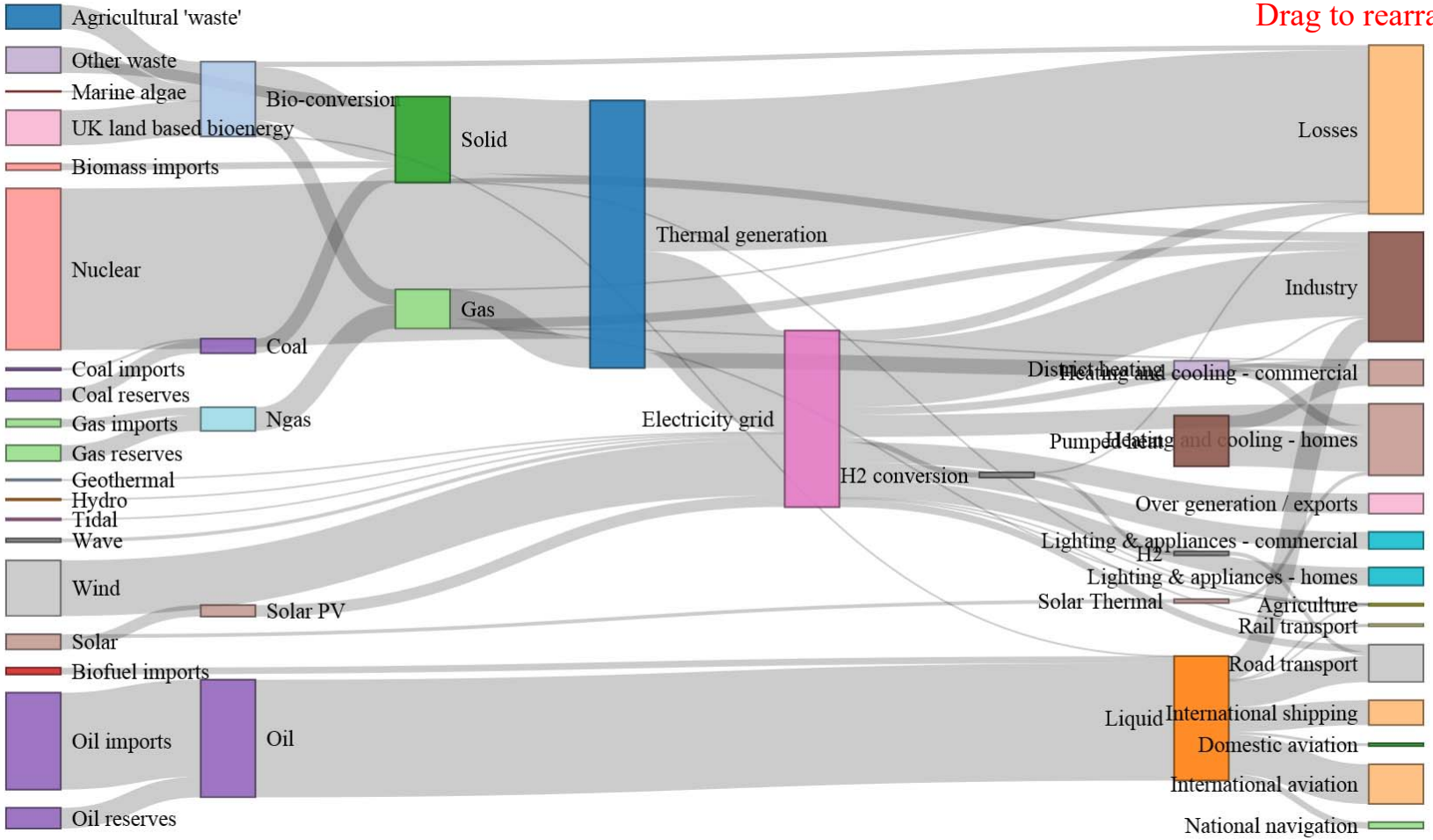


Sankey diagrams are closely related to alluvial diagrams (沖積圖), which show how network structure changes over time. https://en.wikipedia.org/wiki/Alluvial_diagram

sankeyNetwork

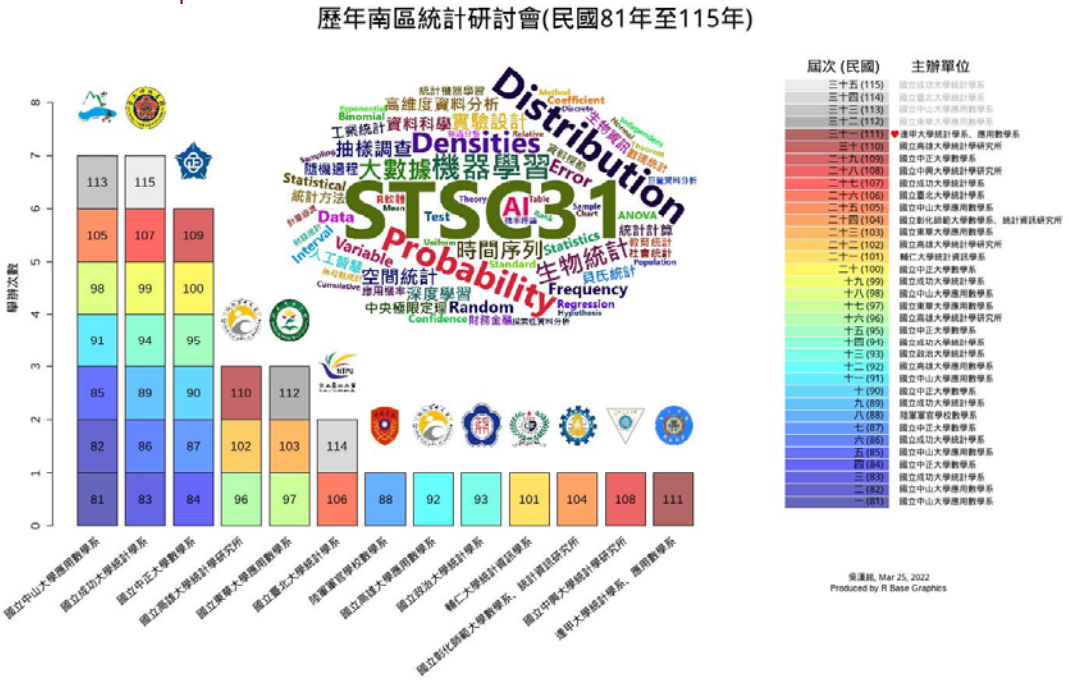
```
> sankeyNetwork(Links = Energy$links, Nodes = Energy$nodes,
+               Source = "source", Target = "target",
+               Value = "value", NodeID = "name",
+               fontSize = 12, nodeWidth = 30)
```

Drag to rearrange nodes.



Word Cloud: wordcloud2 Package

```
> # require(devtools)
> # install_github("lchiffon/wordcloud2")
> library(wordcloud2)
> head(demoFreq)
         word freq
oil       oil   85
said     said   73
prices prices   48
opec     opec   42
mln       mln   31
the       the   26
> str(demoFreq)
'data.frame':   1011 obs. of  2 varia
 $ word: Factor w/ 1011 levels
"100000","108",..: 613 808 686 617 56
149 300 241 ...
 $ freq: num  85 73 48 42 31 26 24 23
...
> wordcloud2(data = demoFreq)
```



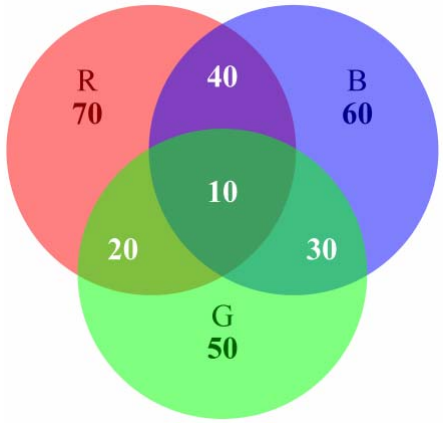
Wordcloud2 introduction
https://cran.r-project.org/web/packages/wordcloud2/vignettes

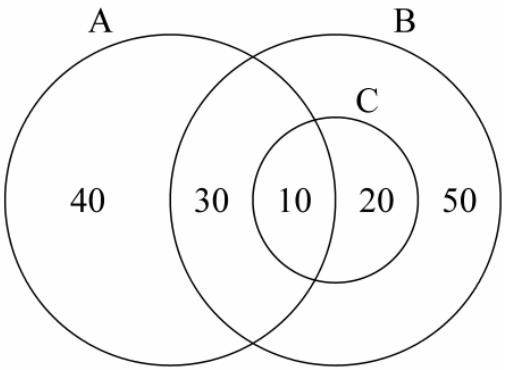
The Wordcloud2 library
https://r-graph-gallery.com/196-the-wordcloud2-library.html

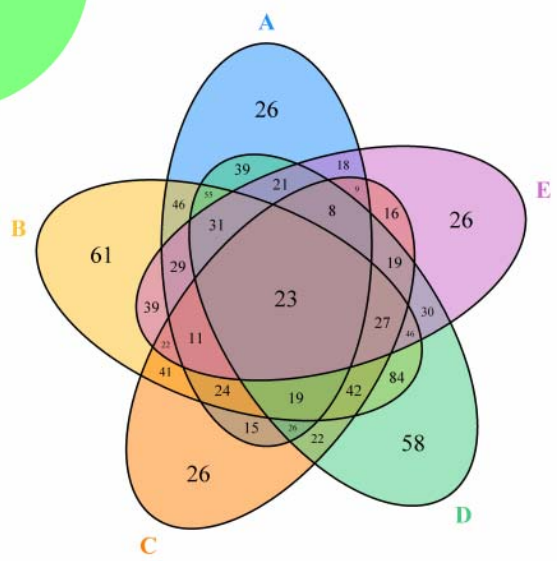
第三十一屆南區統計研討會: https://www.stsc31.info/

Venn Diagrams

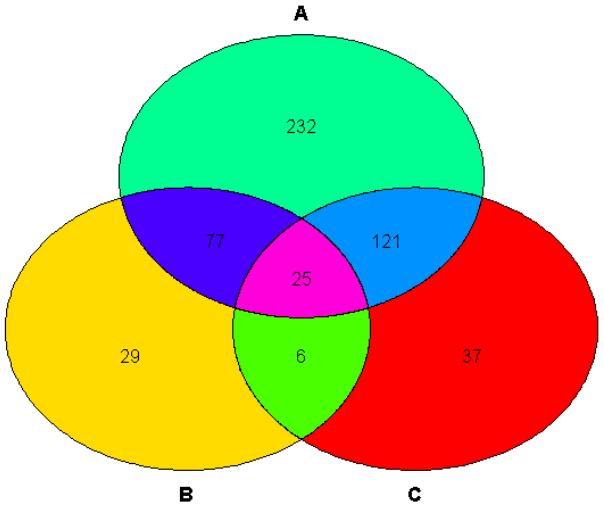
Package **VennDiagram**

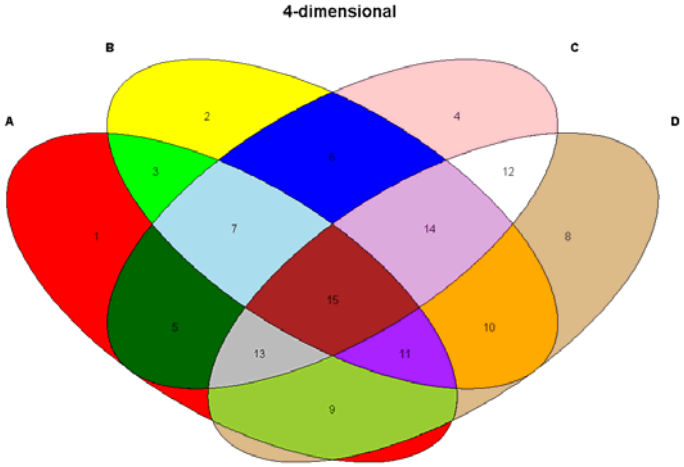






Package **colorfulVennPlot**





類別資料的視覺化: vcd

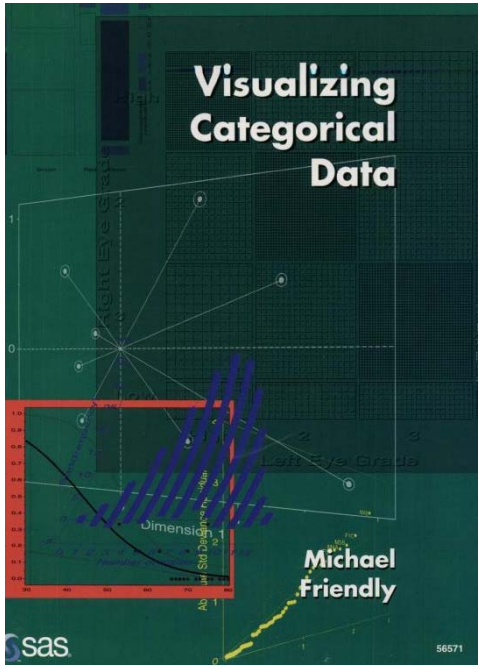
- **vcd**: Visualizing Categorical Data
- Visualizing Categorical Data with SAS and R: http://www.datavis.ca/courses/VCD/
- Visualizing Categorical Data: http://www.datavis.ca/books/vcd/

```
> library(vcd)
```

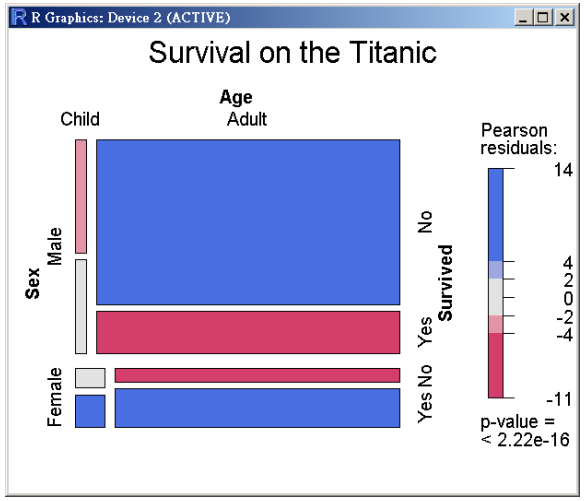
vcd: Visualizing Categorical Data

http://cran.r-project.org/web/packages/vcd/index.html

- Fourfold Display for 2x2 Tables
- Association Plots
- Mosaic Display



Mosiac Displays



Fourfold Display