

R統計圖形與 資料視覺化

吳漢銘

國立政治大學 統計學系



本章大綱與學習目標

■ 第一部份: 客制化圖形

- 簡介、圖形技巧
- 圖形輸出與編輯、客制化基礎統計圖形
- 多重輸出、一頁多張圖形
- 顏色、資料符號、線條、座標軸
- 數學符號、文字標號、圖例說明

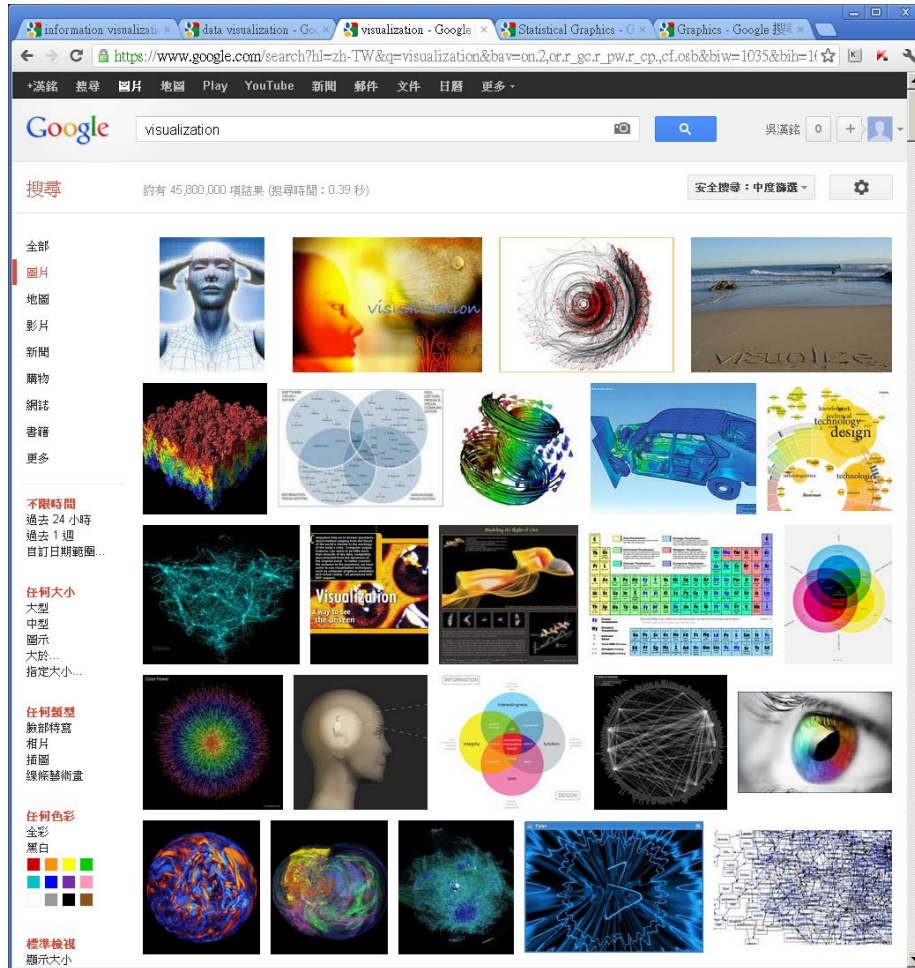
■ 第二部份: 基礎統計圖形

- 單一樣本、雙樣本及多樣本的基礎統計圖形
- 3D散佈圖、rgl、影像、heatmap、等高線圖

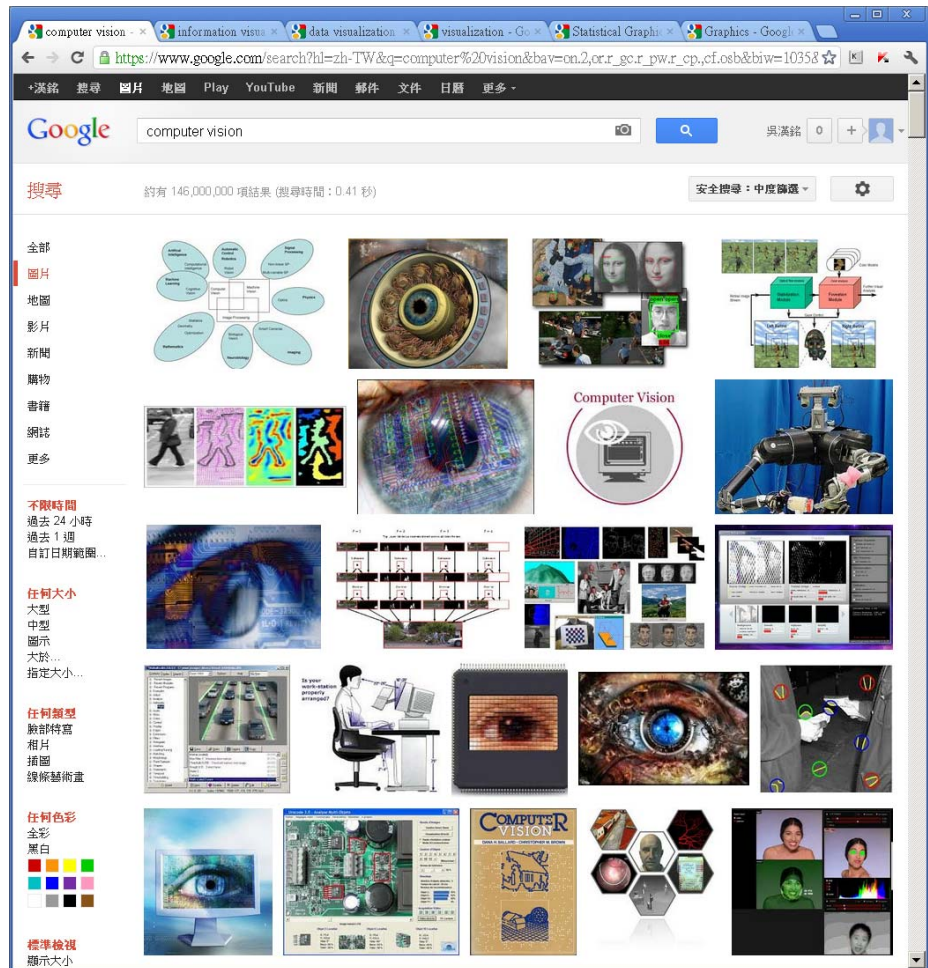
■ 第三部份: 其它主題

- Venn Diagrams、數學符號
- ggplot2、Rgraphviz、igraph、Choropleth Maps、
- RgoogleMaps、maps, mapdata、googleVis、vcd、ggvis、graphics.SDA
- 參考書目、Web Resource

Visualization

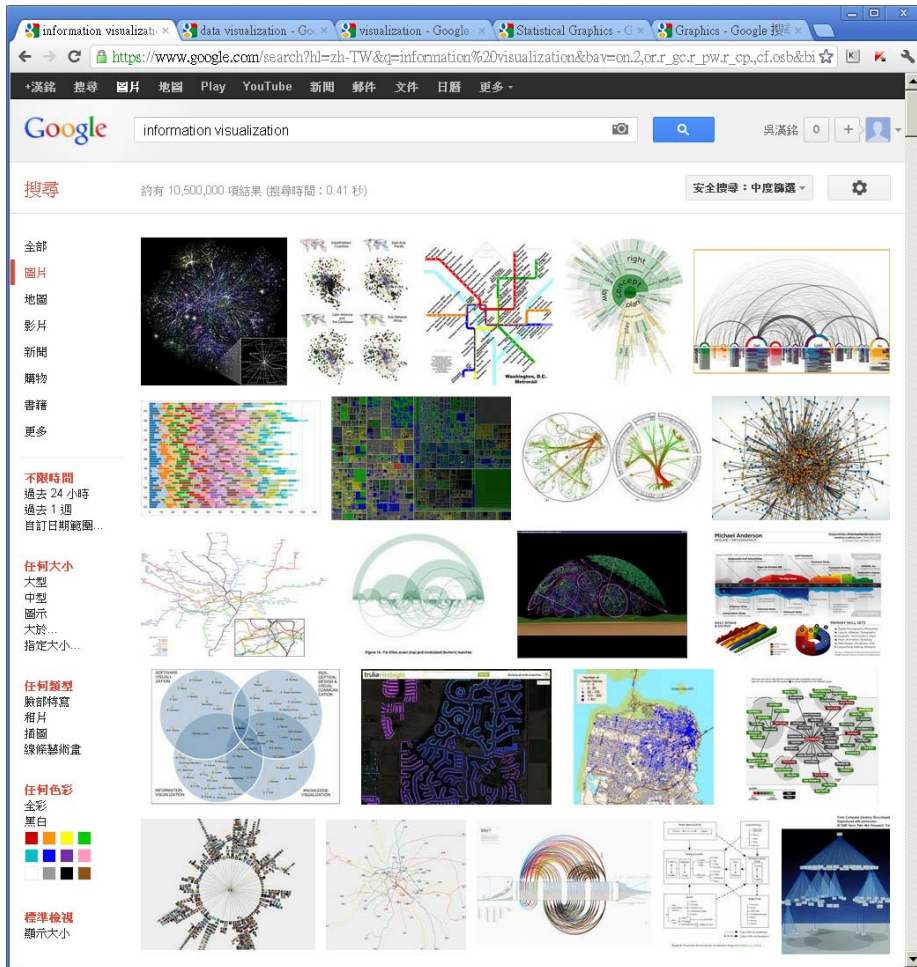


Computer Vision

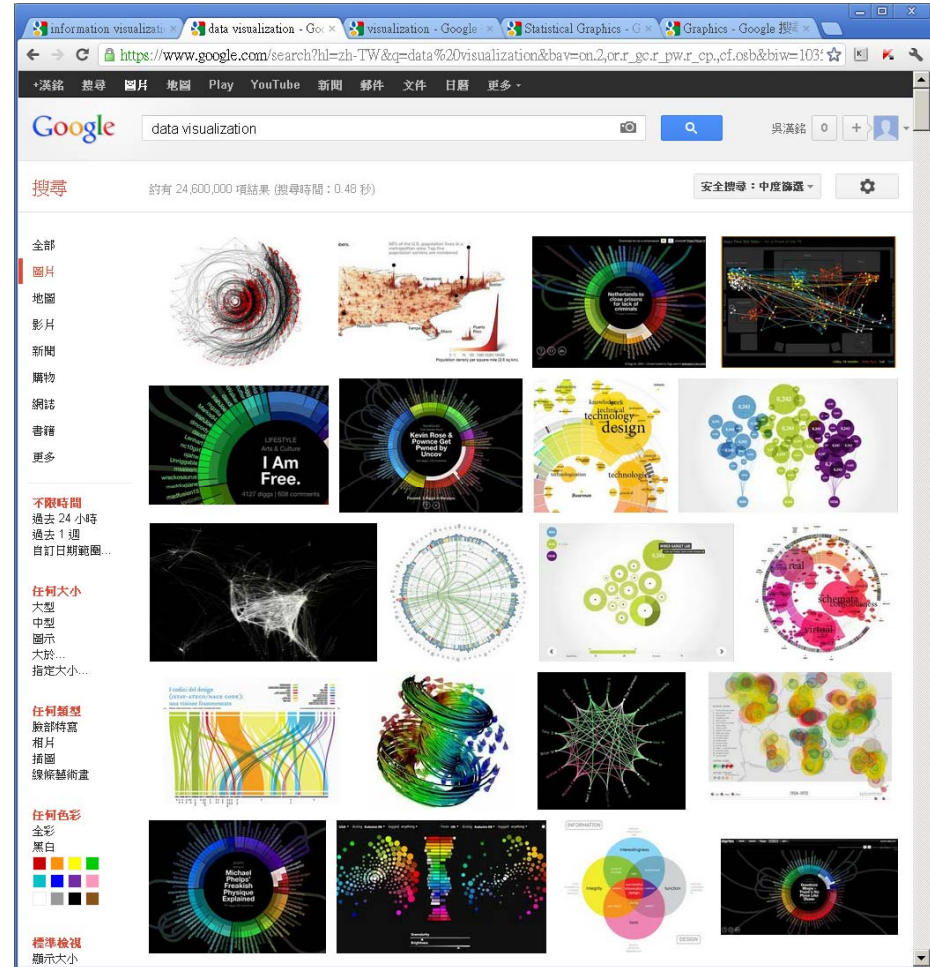


Data Visualization

Information Visualization



Data Visualization



What is Visualization?

People said

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



The longest name of a city in New Zealand.

What is visualization?

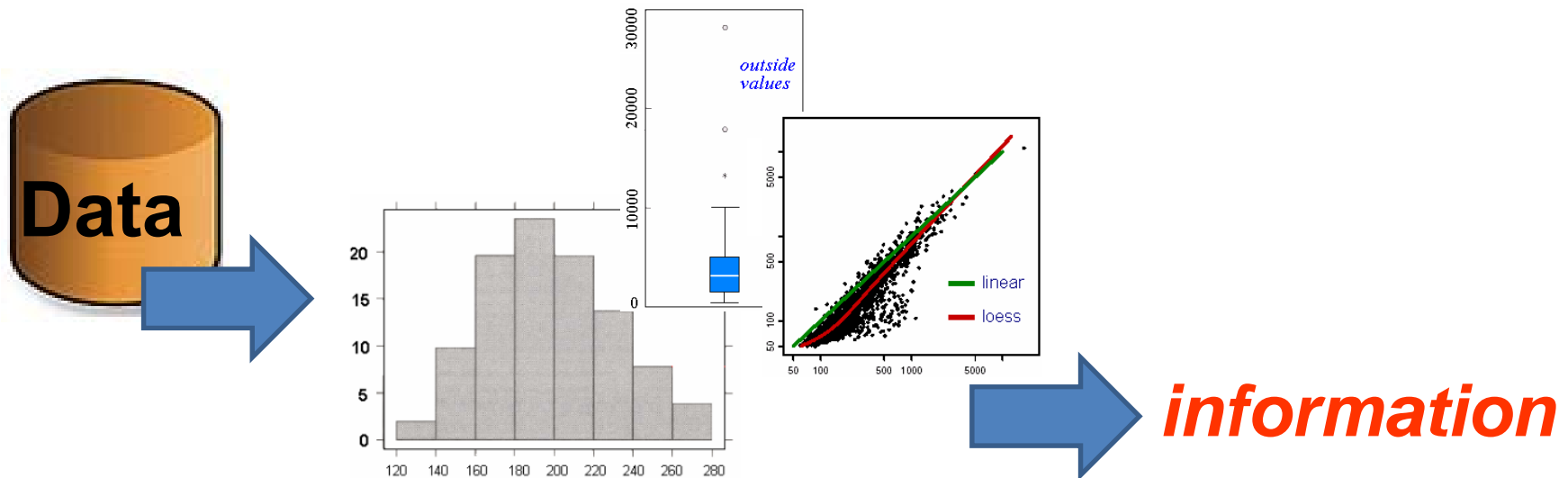
- Making things/processes/abstractions visible (to transform into pictures) that are not directly accessible by the human eye.
- Computer aided extraction and display of information from data.



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

Graphical Methods

The purpose of statistical graphics is to provide **visual** representations of **quantitative** information.



Exploratory Data Analysis (EDA) Tool

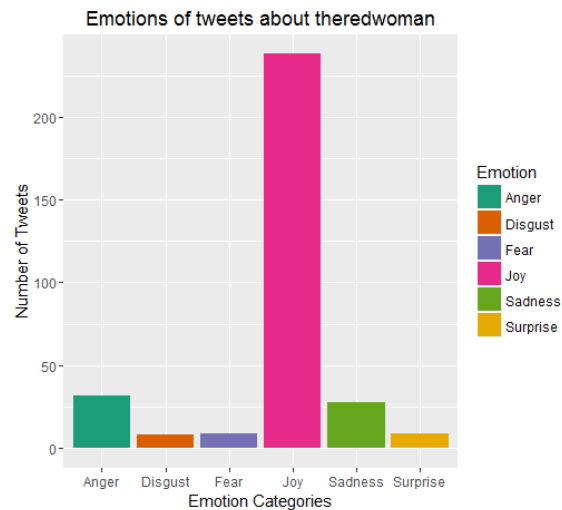
Statistical graphics comprise

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.



Journal of Computational and Graphical Statistics, Volume 22, 2013 - Issue 1

- **Infovis and Statistical Graphics: Different Goals, Different Looks**
Andrew Gelman & Antony Unwin, Pages: 2-28
- **InfoVis Is So Much More: A Comment on Gelman and Unwin and an Invitation to Consider the Opportunities**, Robert Kosara, Pages: 29-32
- **InfoVis and Statistical Graphics: Comment**
Paul Murrell, Pages: 33-37
- **Graphical Criticism: Some Historical Notes**
Hadley Wickham , Pages: 38-44
- **Tradeoffs in Information Graphics**
Andrew Gelman & Antony Unwin , Pages: 45-49



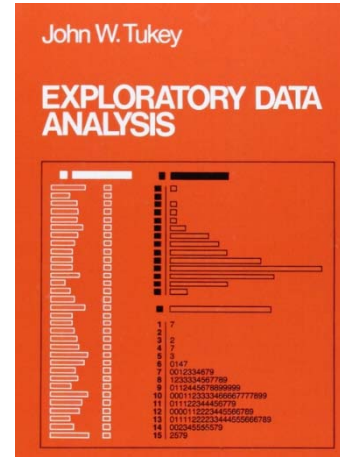
<http://emarketingwall.com/how-twitter-responded-to-the-latest-episode-of-game-of-thrones>

Exploratory Data Analysis, EDA



John Tukey (1915~2000) (統計學界的畢卡索)

「對正確的問題有個近似的答案，
勝過對錯的問題有精確的答案。」



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

Visualization = Graphing for Data + Fitting + Graphing for Model



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)

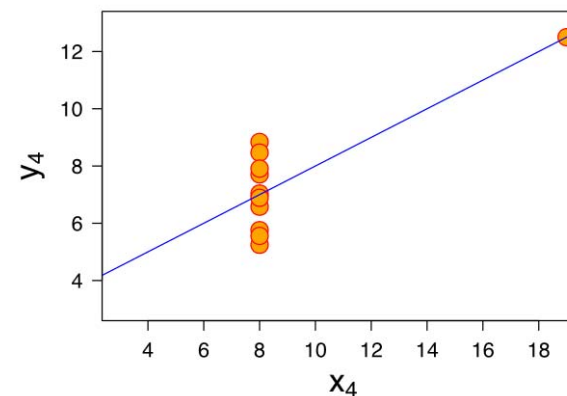
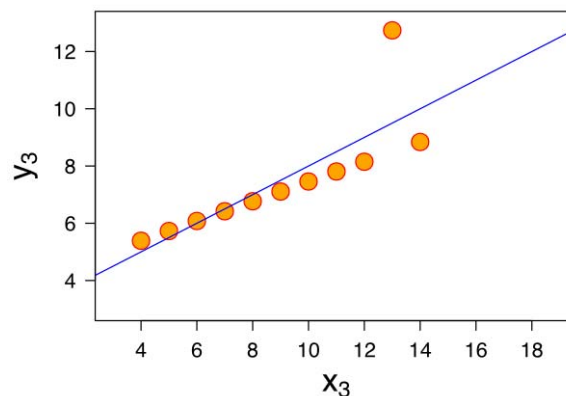
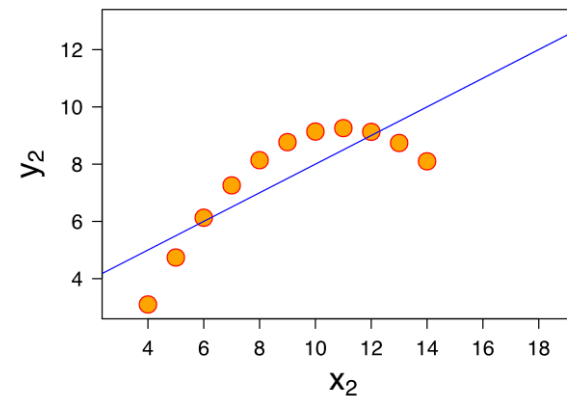
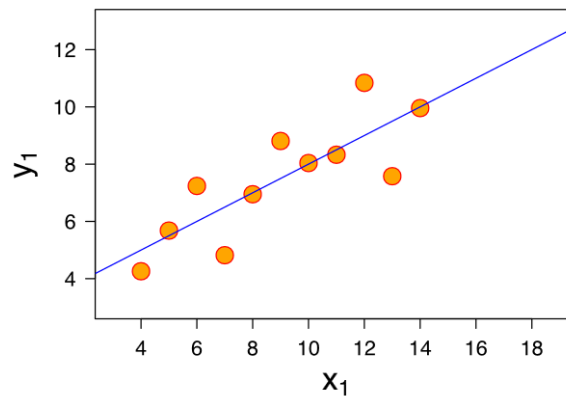
https://en.wikipedia.org/wiki/Anscombe's_quartet

<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'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

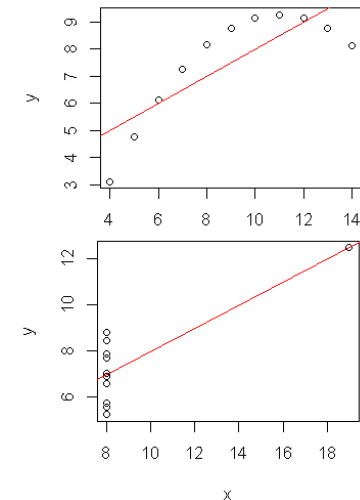
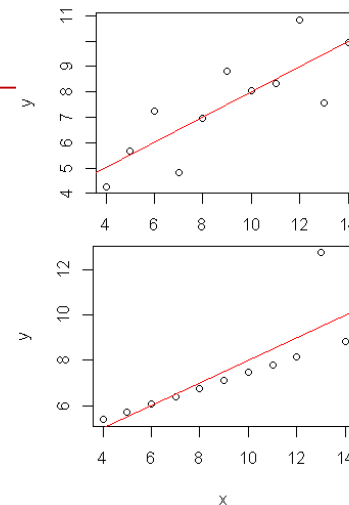
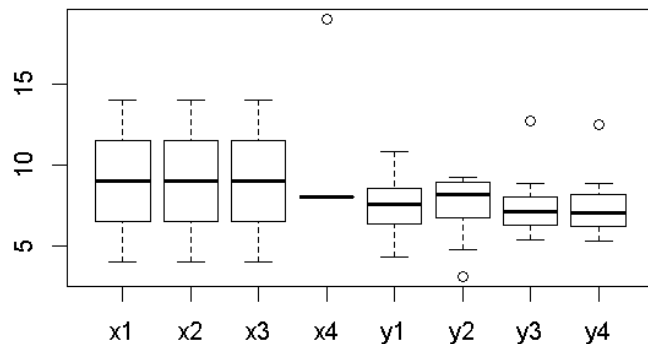
```

```

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])

```

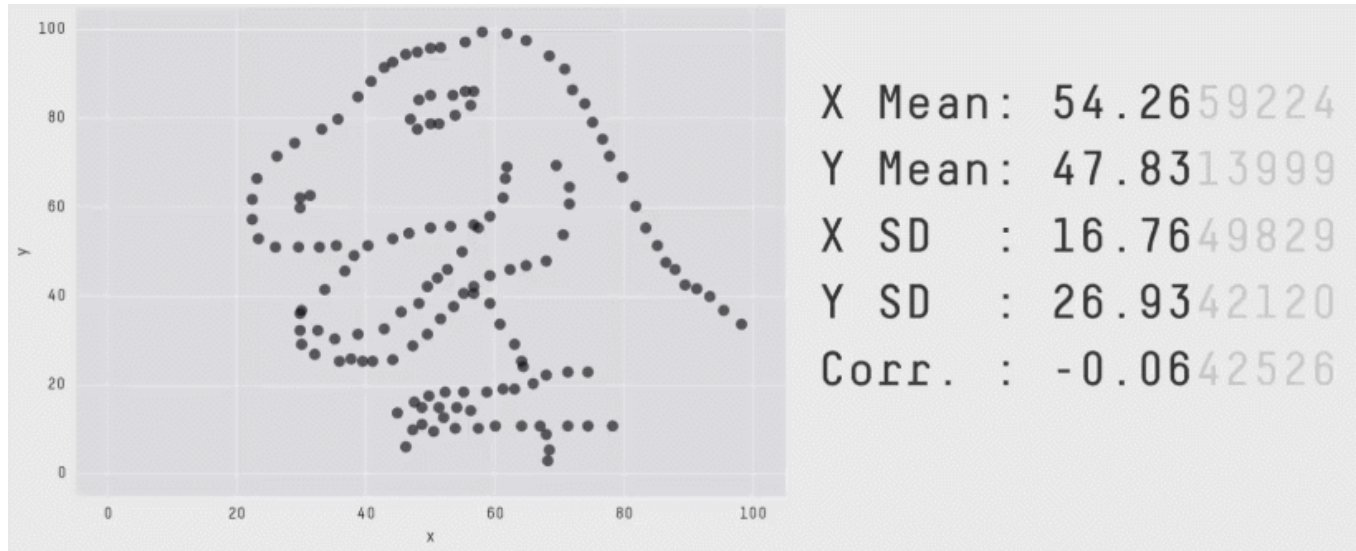
```
boxplot(anscombe)
```



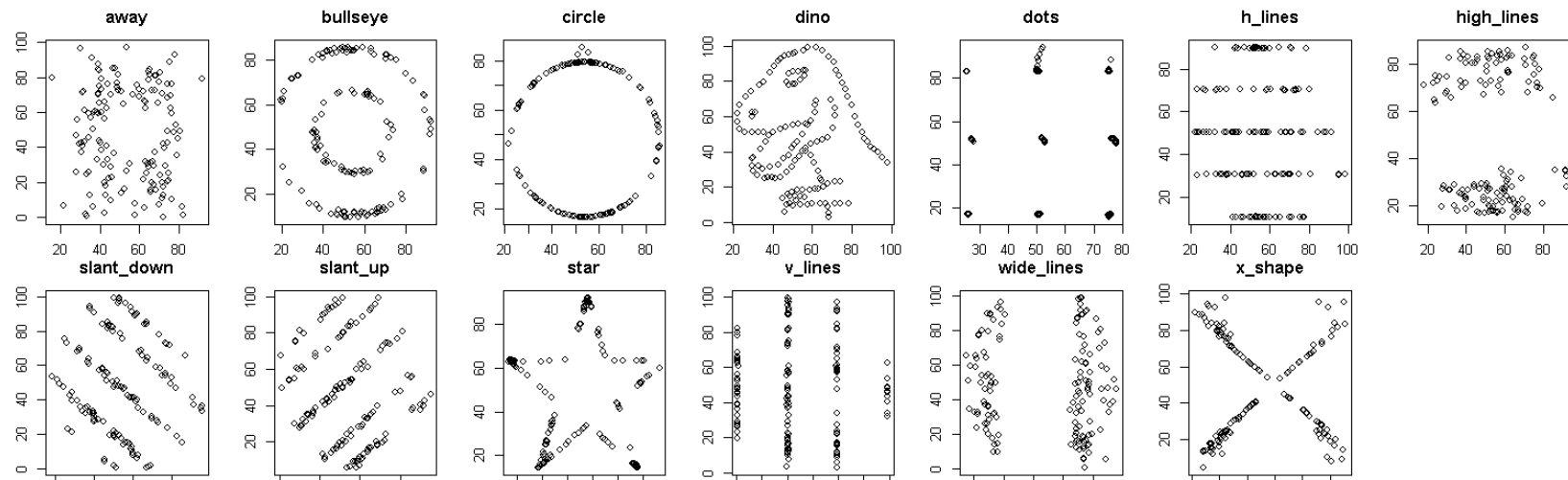


The Datasaurus Dozen

`install.packages("datasauRus")`



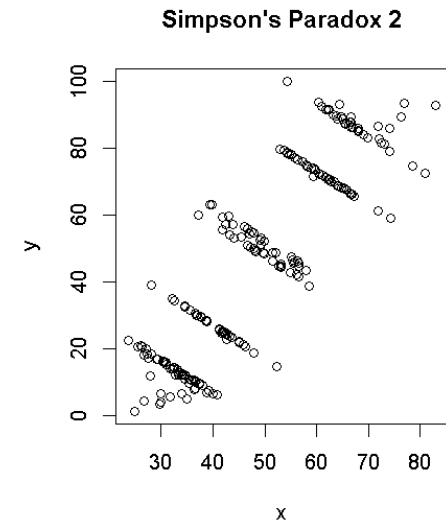
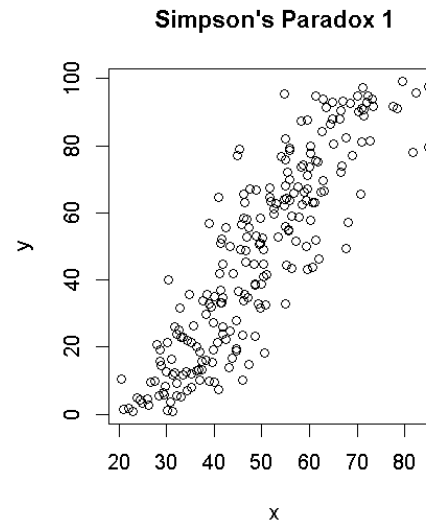
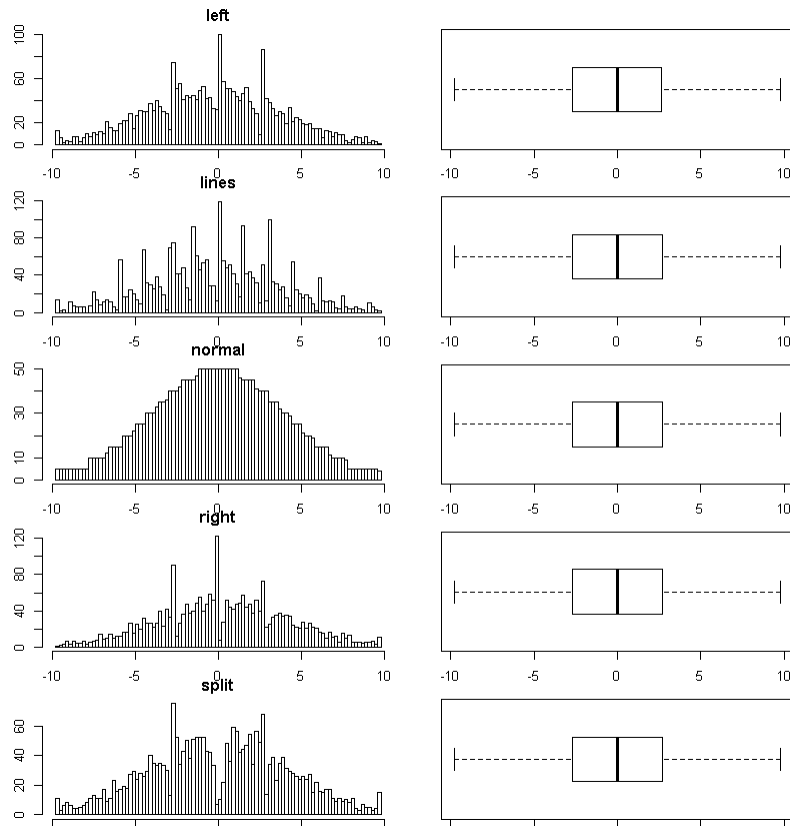
Justin Matejka and George Fitzmaurice, Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. <https://www.autodeskresearch.com/publications/samestats>





The Datasaurus Dozen

More examples





EDA and Visualization

- **Data visualization** is the presentation of data in a pictorial or graphical format.
 - Any effort to help people understand the significance of data by placing it in a visual context.
 - Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software.
- **Get to know your data:** distributions (symmetric, normal, skewed), data quality problems, outliers, correlations and inter-relationships, subsets of interest, suggest functional relationships.
- **Visualizing data:** One variable, Two variables, More than two variables, Other types of data, Dimension reduction.
- **Interactive data visualization:** using computers and mobile devices to drill down into charts and graphs for more details, and interactively (and immediately) changing what data you see and how it is processed.



Graphical Perception

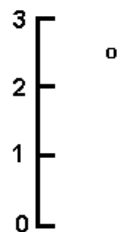
Human reception and comprehension of graphical information involves three fundamental perceptual task:

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

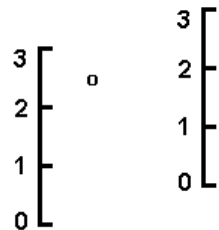
Graphical Perception Tasks.

Ordered from the most accurate to the least accurate (Jacoby, 1997)

A. Position along a common scale



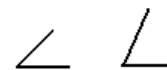
B. Position along common, nonaligned scales



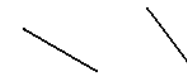
C. Length



D. Angle



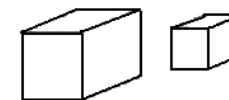
E. Slope, direction



F. Area



G. Volume




H. Fill density, color saturation



Demo (graphics)



The R Graphics Package 

Documentation for package graphics

- DESCRIPTION file.
- Code demos. Use `demo()` to run them.

graphics-package	The R Graphics Package
filled.contour	Level (Contour) Plots
Pars	Set or Query Graphics Parameters
abline	Add Straight Lines to a Plot
arrows	Add Arrows to a Plot
asp	Set up World Coordinates
assocplot	Association Plots
Axis	Generic Function to Add an Axis to a Plot
axis	Add an Axis to a Plot
axis.POSIXct	Date and Date-time Plotting Functions
axTicks	Compute Axis Tickmark Locations
barplot	Bar Plots
box	Draw a Box around a Plot
boxplot	Box Plots
boxplot.matrix	Draw a Boxplot for each Column (Row) of a Matrix

```
> library(graphics)
> demo(graphics) #常見圖
> demo(Hershey) #各種符號
> demo(image) #image和image2
> demo(Japanese) #日本字
> demo(persp) #曲面圖
> demo(plotmath) #數學符號
```

```
> opar <- c(opar, par(bg = "white"))
> plot(x, ann = FALSE, type = "n")
Waiting to confirm page change...
> abline(h = 0, col = gray(.90))
> lines(x, col = "green4", lty = "dot")
> points(x, bg = "limegreen", pch = 2)
> title(main = "Simple Use of Color In a Plot",
+       xlab = "Just a Whisper of a Label",
+       col.main = "blue", col.lab = "black",
+       cex.main = 1.2, cex.lab = 1.0)
> ## A little color wheel. This is a
> ## pie chart. If you have a
> ## probably find that numerically equal
> ## equispaced. On my display at home
> pie(rep(1,24), col = rainbow(24), radius = 1)
Waiting to confirm page change...
```

```
> .libPaths()
[1] "C:/Users/userpc/Documents/R/win-library/3.3"
[2] "C:/Program Files/R/R-3.3.1/library"
```

程式碼:

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

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



- **CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization**

<http://cran.r-project.org/web/views/Graphics.html>

- **Plotting:** `plotrix`, `vcd`, `hexbin`, `gclus`, `gplots`, `aplpack`, `lattice`, `scatterplot3d`, `misc3d`, `onion`.
- **Graphic Applications:**
 - **Effect ordering:** `gclus*`, `cba`, `seriation`, `biclust`.
 - **Large Data Sets:** `ash`, `hexbin*`, `scagnostics`.
 - **Trees and Graphs:** `ade4`, `ape`, `igraph`, `diagram`, `Rgraphviz`, `igraph`.
- **Graphics Systems:** `lattice*`, `ggplot2`.
- **Devices:** `cairoDevice`, `RGtk2`, `RSvgDevice`, `rgl`, `JavaGD`.
- **Colors:** `colorspace`, `vcd*`, `RColorBrewer`, `dichromat`.
- **Interactive Graphics:** `rggobi`, `iplots`, `JavaGD*`, `playwith`, `cairoDevice*`, `RGtk2*`, `rgl*`.
- **Development:** `rgl*`, `gridBase`.
- **Others:** `animation`, `Cairo`, `IDPmisc`, `klaR`, `latticeExtra`, `RGraphics`, `RSVGTipsDevice`, `tkrplot`, `vioplot`, `xgobi`.

- R Graphics: <http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html>

- R Graphics Gallery: <http://research.stowers-institute.org/efg/R/index.htm>

- R graph gallery: <http://rgraphgallery.blogspot.tw/>

- gRaphics! <http://grrrraphics.blogspot.tw/>

The R graph gallery

<http://www.r-graph-gallery.com/>



Produce (traditional)
graphics



Customize graphics
using high-level
graphics setting



Customize graphics
using low-level
graphics setting



Output

Table 1.1

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

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

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



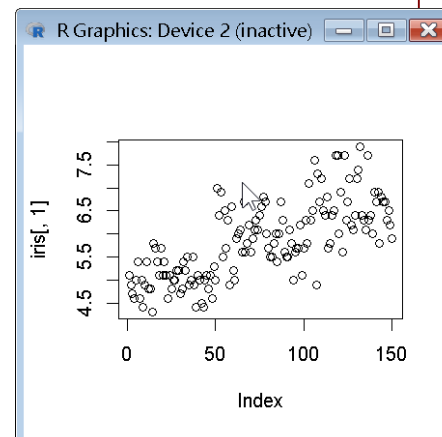
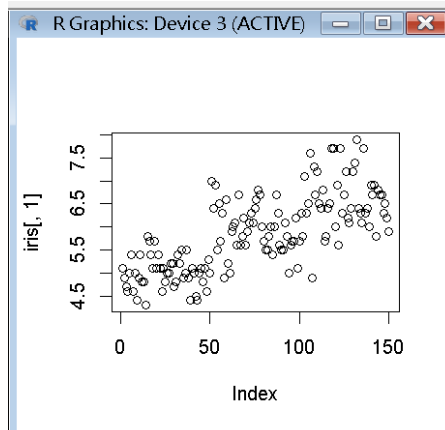
dev {grDevices}

Control Multiple Devices

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

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

RGui (64-bit)





圖形輸出



Example

方法1: Save as ...

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

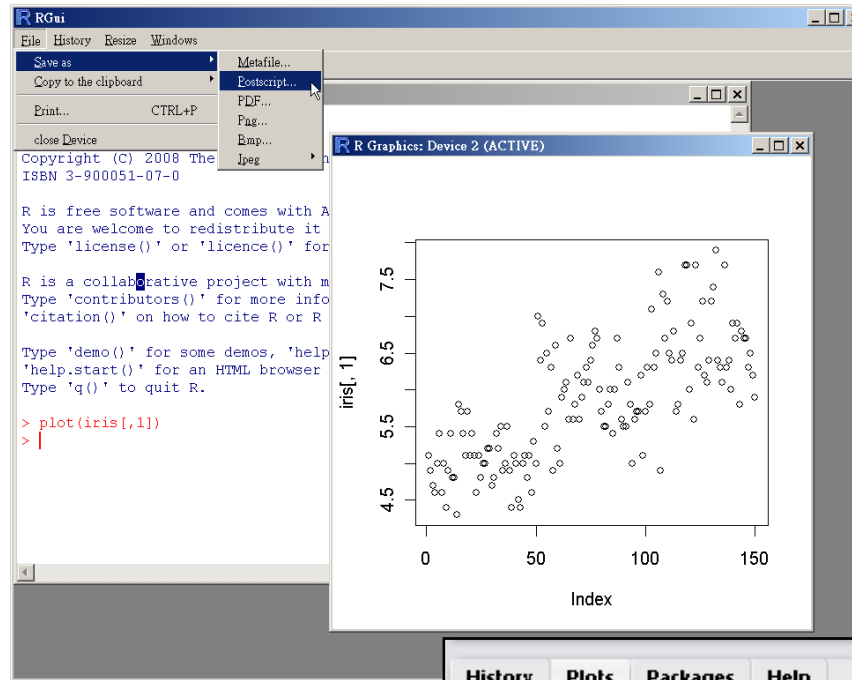
方法2: Activate device

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

or

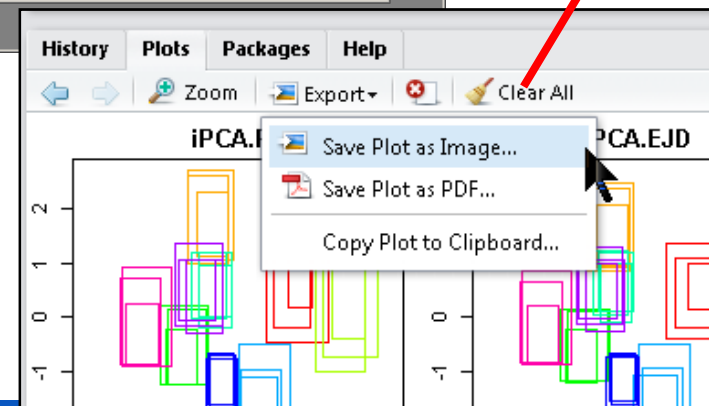
```
> jpeg(file="iris3.jpg")
> plot(iris[,3])
> dev.off()
```

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



RGui

```
> dev.off()
```



RStudio



向量圖與非向量圖格式

存檔格式及目的

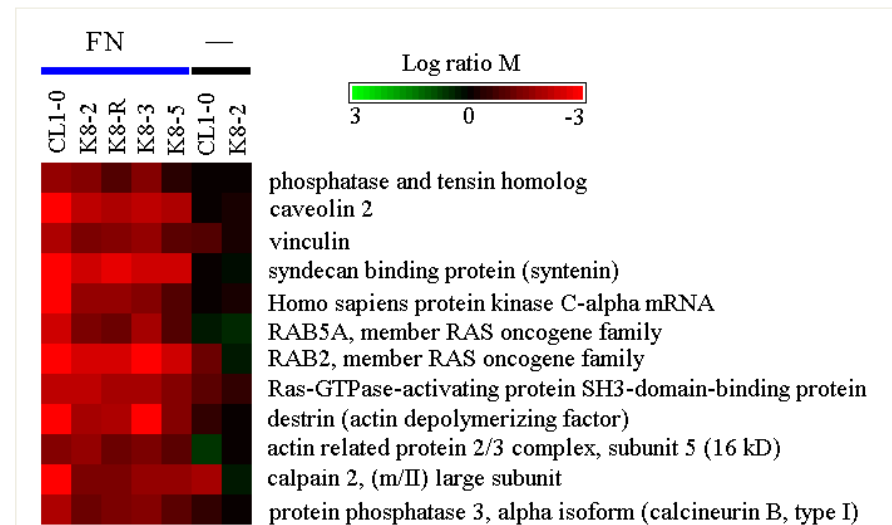
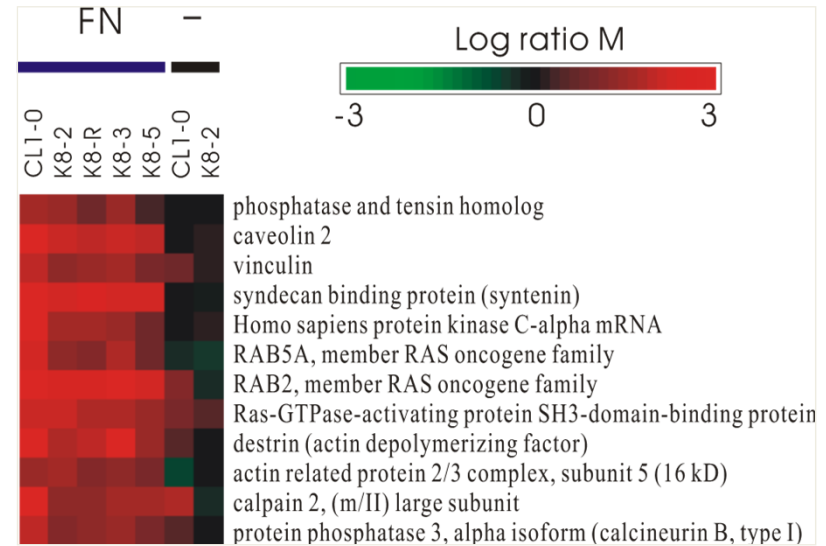
- ❑ 點陣圖檔 (壓縮失真: PNG, JPEG, TIFF, BMP) : web page, powerpoint, word,...
- ❑ 向量圖檔 (物件不失真: Metafile, SVG, EPS) : Publication, LaTeX.

向量圖(在此為wmf格式):

可拉大或縮小不失真。
此時在螢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(在螢幕上看的)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。



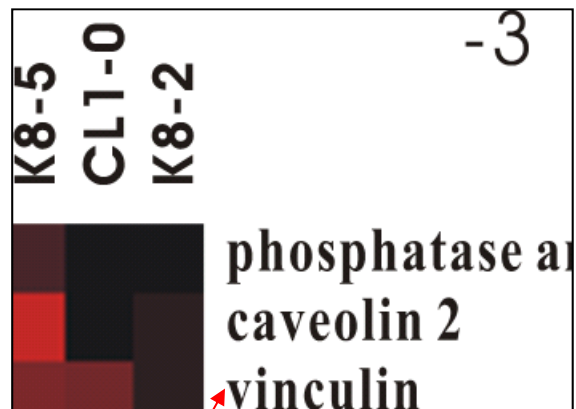


向量圖與非向量圖格式

如何放大圖片:

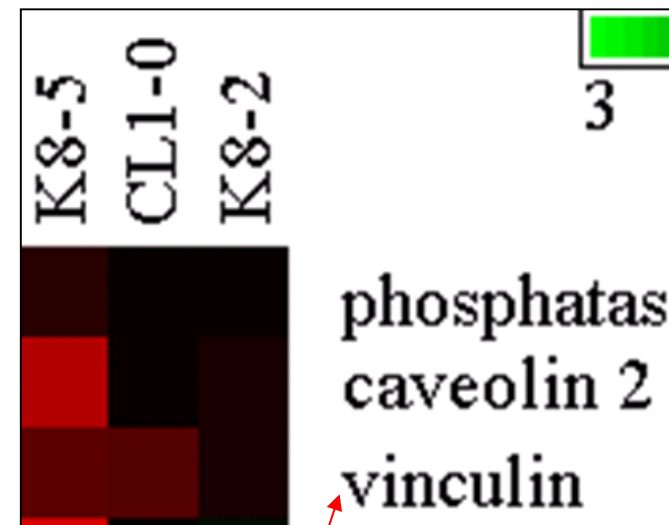
1. 選按圖片=>滑鼠右鍵=>設定圖格式=>大小=>縮放比例=>高度設為300%=>OK
2. 這跟放大投影片(或全螢幕)成300%是不一樣的。

向量圖放大變這樣:



無鋸齒狀, 邊界會糊糊的(或霧霧的是軟體自動平滑化的結果)

點陣圖放大變這樣:



有鋸齒狀



多重輸出 (Multiple Output)

24/216

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

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

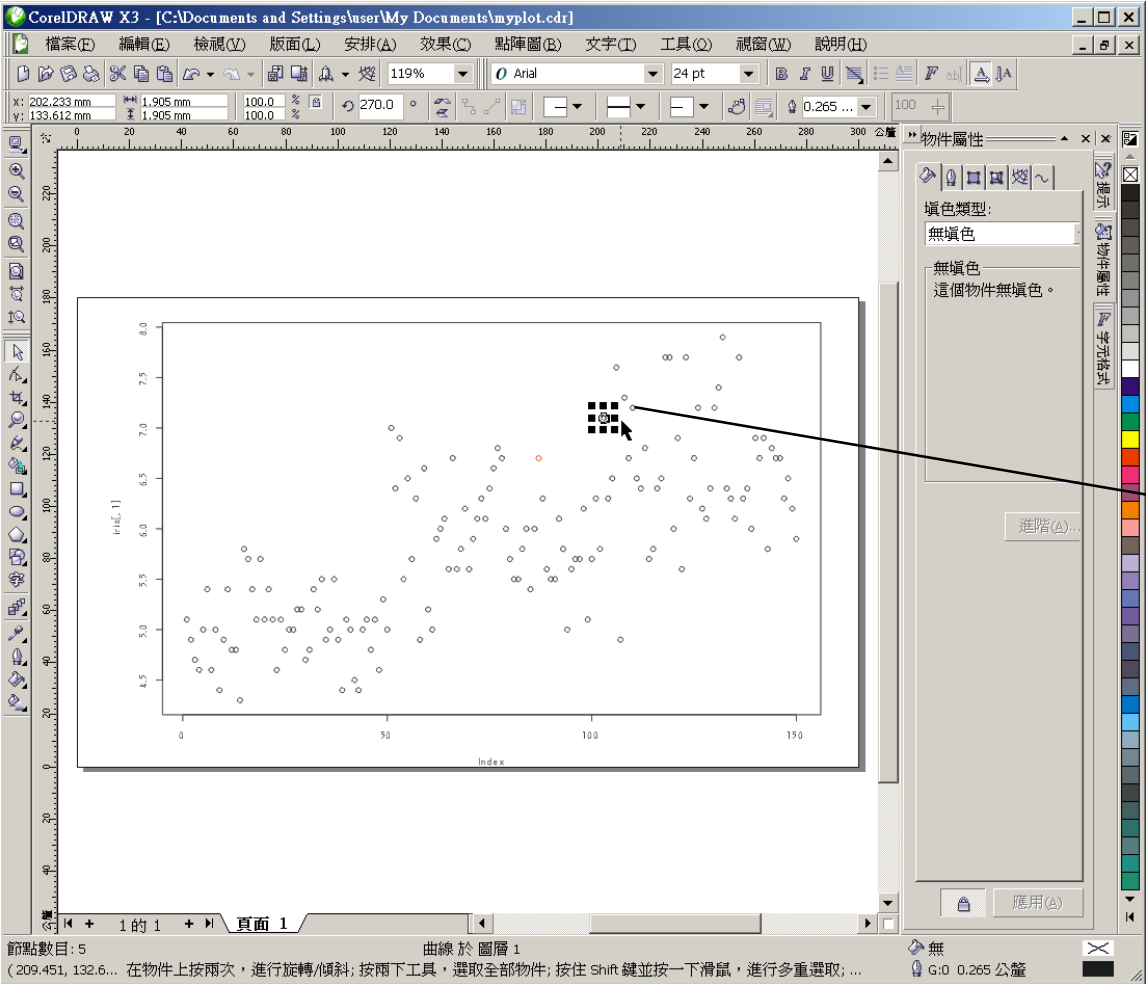
the **%03d** is replaced by a three-digit number indicating the page number for each file.

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

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

```
> getwd()
```

向量圖編輯

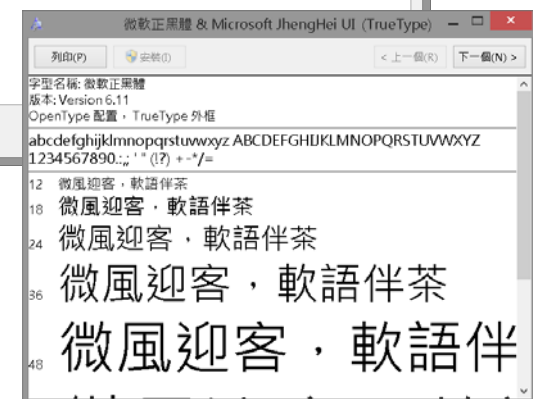
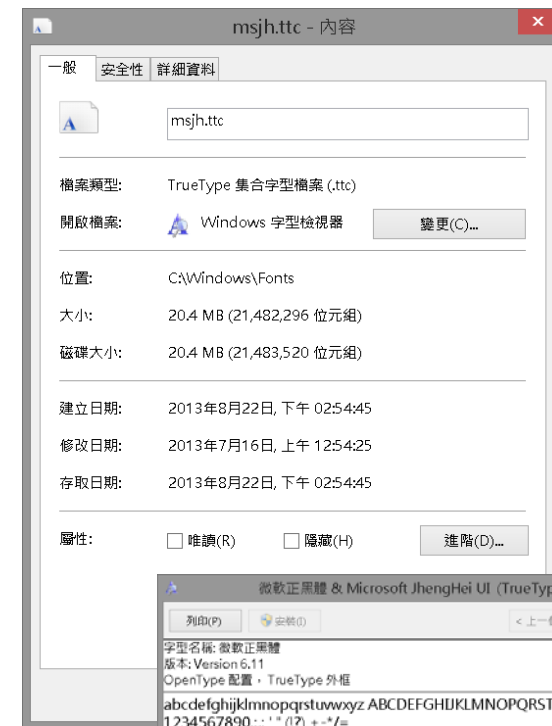
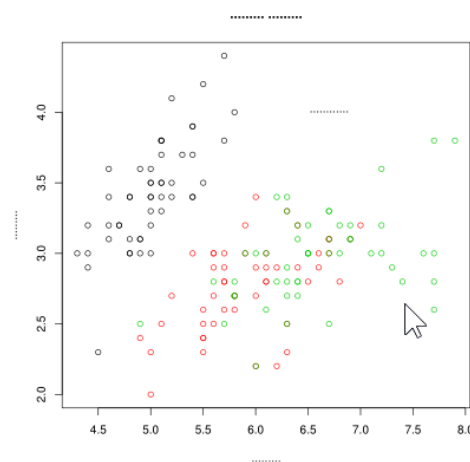
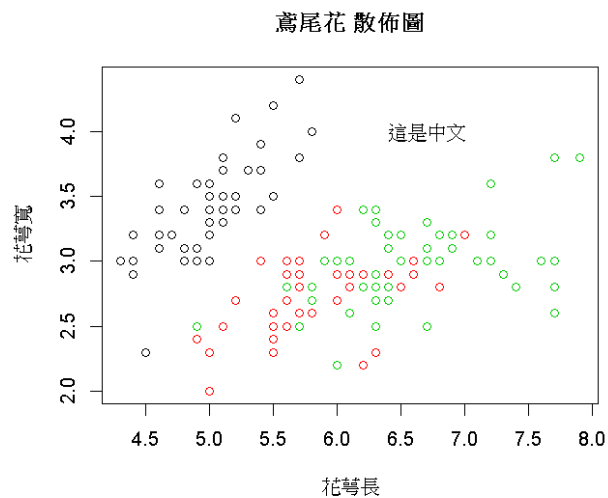


物件

Example: CorelDraw

圖形存檔及輸出中文問題

```
> 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()
```



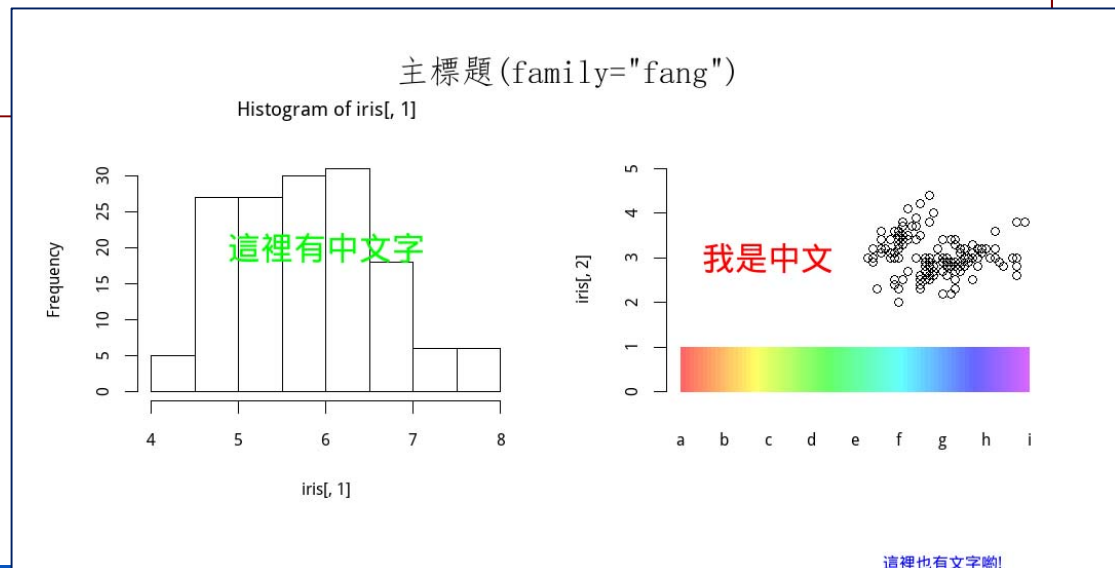
```
> #showtext: Using Fonts More Easily in R Graphs
> library(showtext)
> font_add("msjh", "msjh.ttc") ## 微軟正黑體
> showtext_auto(enable=TRUE)
```

圖形存檔及輸出中文問題

```

> 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("這裡也有文字啲!", 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

```



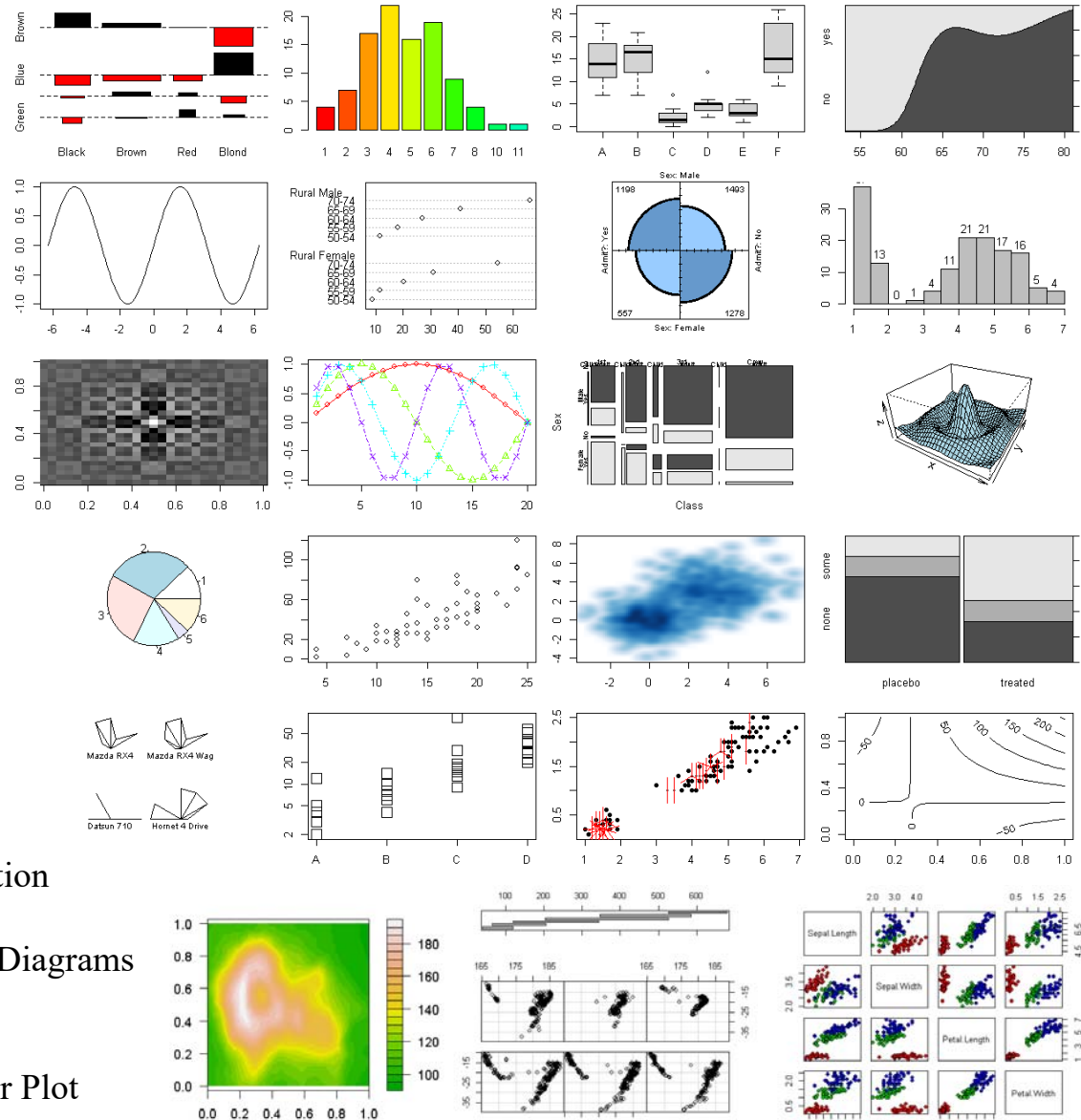


graphics: The R Graphics Package

28/216

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





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

- 新開啟一個繪圖視窗

```
> windows()
```

```
> x11()
```

- 多張圖(mxn)一頁

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

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

```
> windows()
```

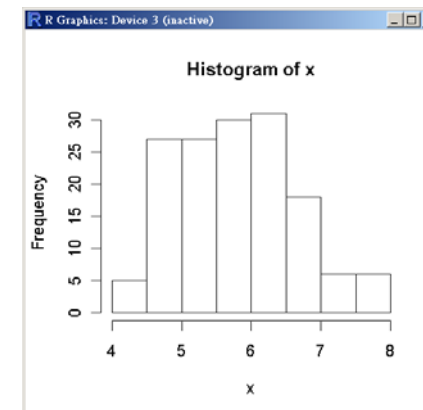
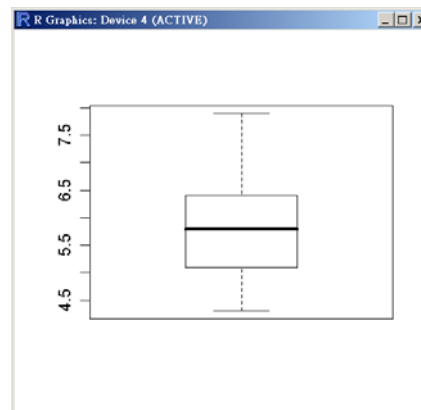
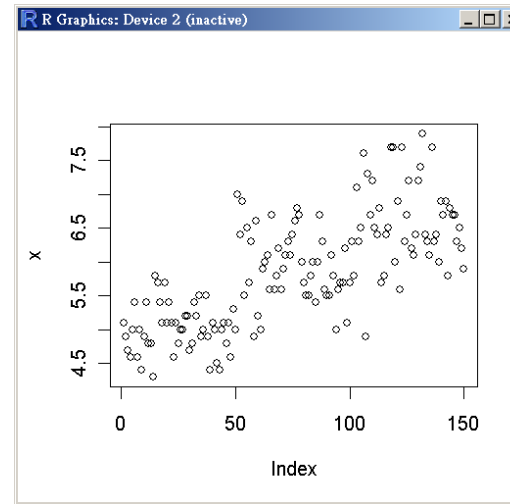
```
> plot(x) # index plots to show the values of y in sequence
```

```
> windows()
```

```
> hist(x) # histogram to show a frequency distribution
```

```
> windows()
```

```
> boxplot(x)
```



圖標題和範圍 (Title and Limit)

```

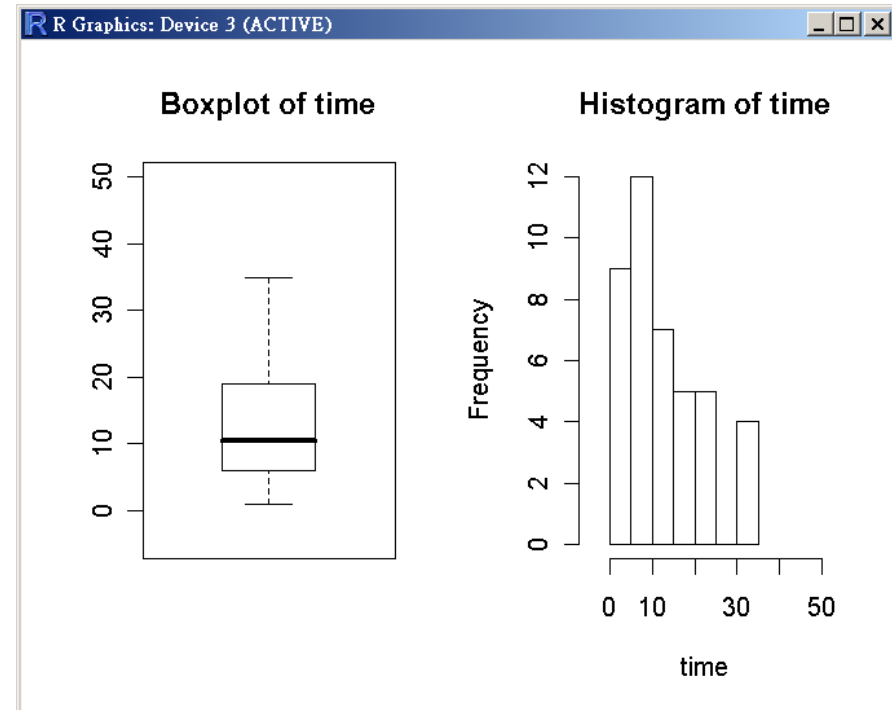
> library(MASS)
> data(gehan)
> time <- gehan$time
> windows()
> par(mfrow=c(1,2))
> boxplot(time, ylim=c(-5, 50))
> title("Boxplot of time")
> hist(time, xlim=c(-5, 50))
> title("Histogram of time")

```

```

> 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
>

```



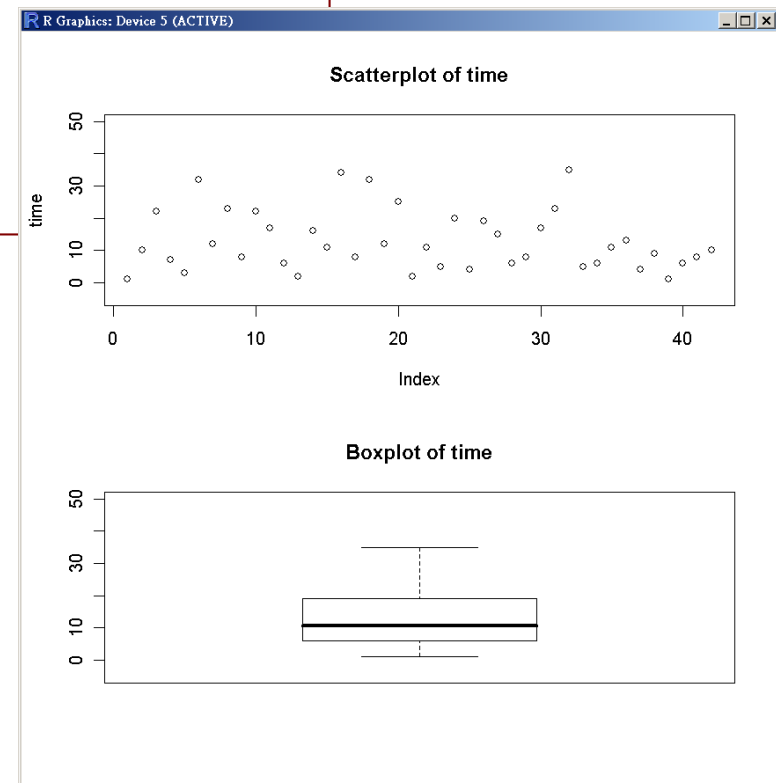
將繪圖視窗區域拉大

圖標題和範圍 (Title and Limit)

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

#or

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



標準引數 (Standard Arguments)

33/216

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

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

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

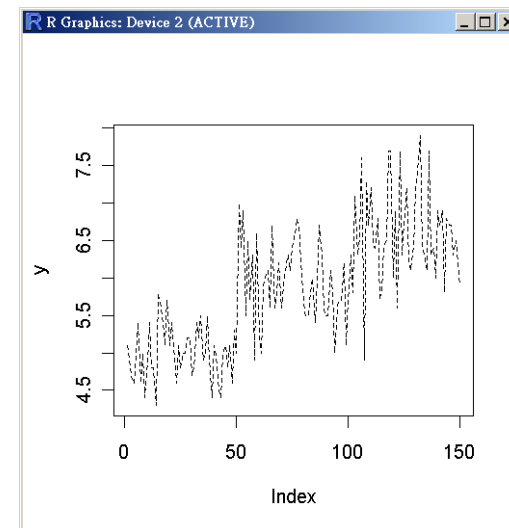
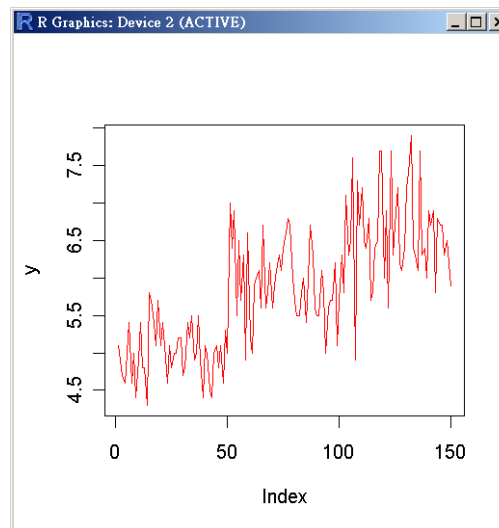
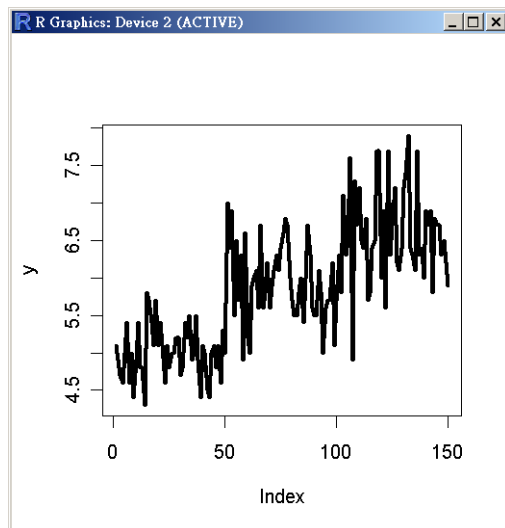
```
> plot(y, type="l", lwd=3)
```

- `col`: color 顏色

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

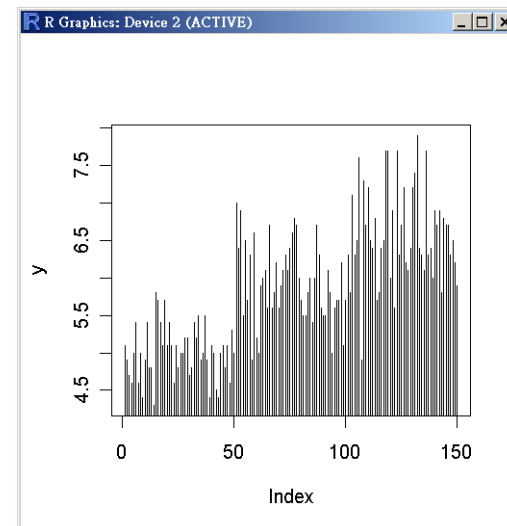
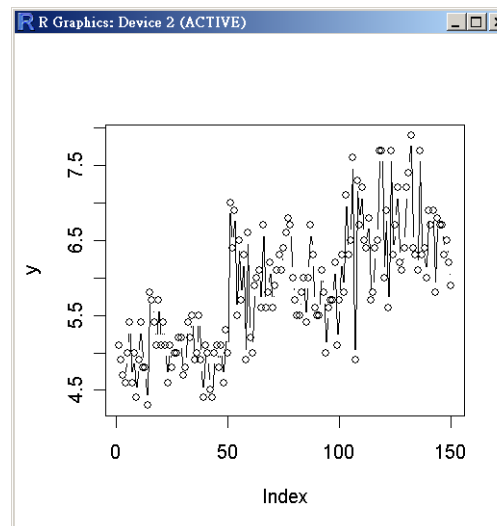
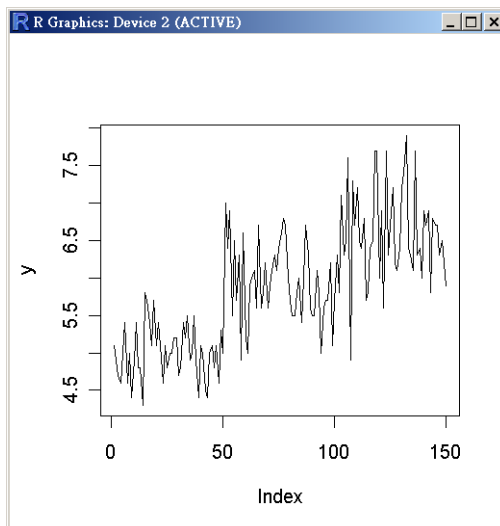
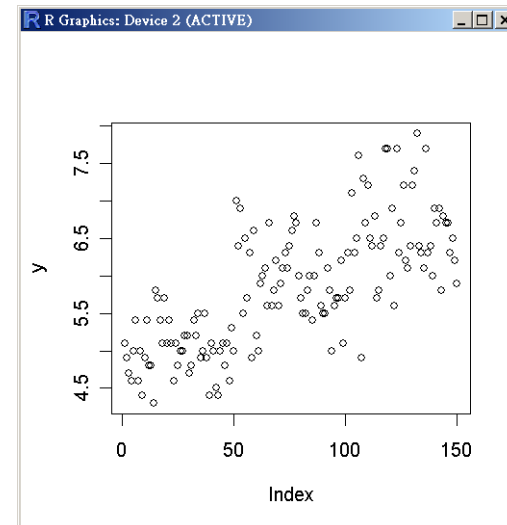
- `lty`: line type 線的型式

```
> 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
```

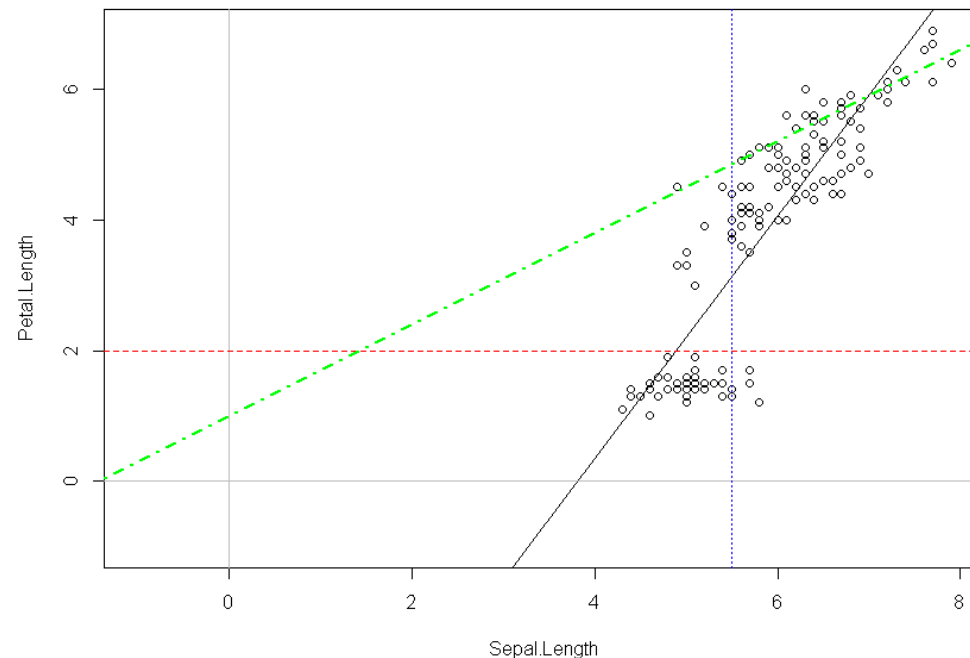




於圖中加畫出直線: **abline**

35/216

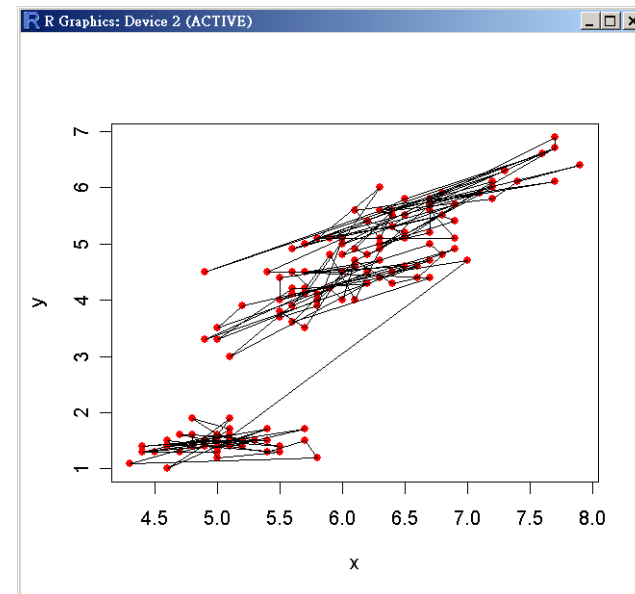
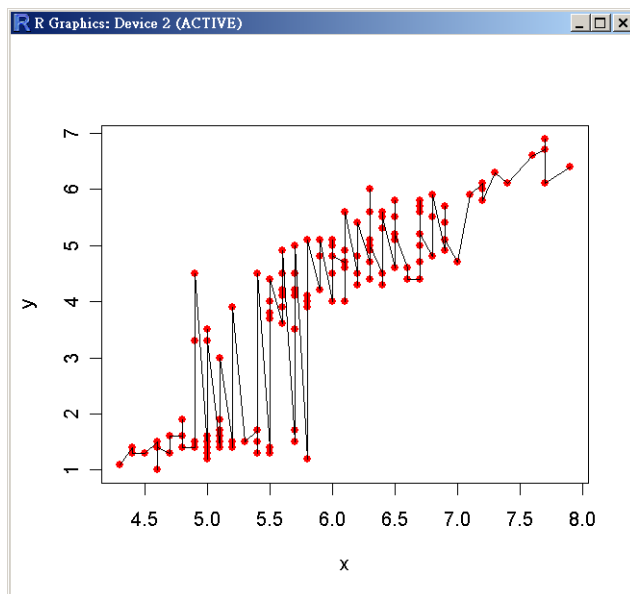
```
attach(iris)
plot(Sepal.Length, Petal.Length, xlim = c(-1, max(Sepal.Length)),
     ylim = c(-1, max(Petal.Length)))
abline(lm(Petal.Length ~ Sepal.Length), col = "black")
abline(h = 0, col = "grey")
abline(v = 0, col = "grey")
abline(h = 2, col = "red", lty = 2)
abline(v = 5.5, col = "blue", lty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



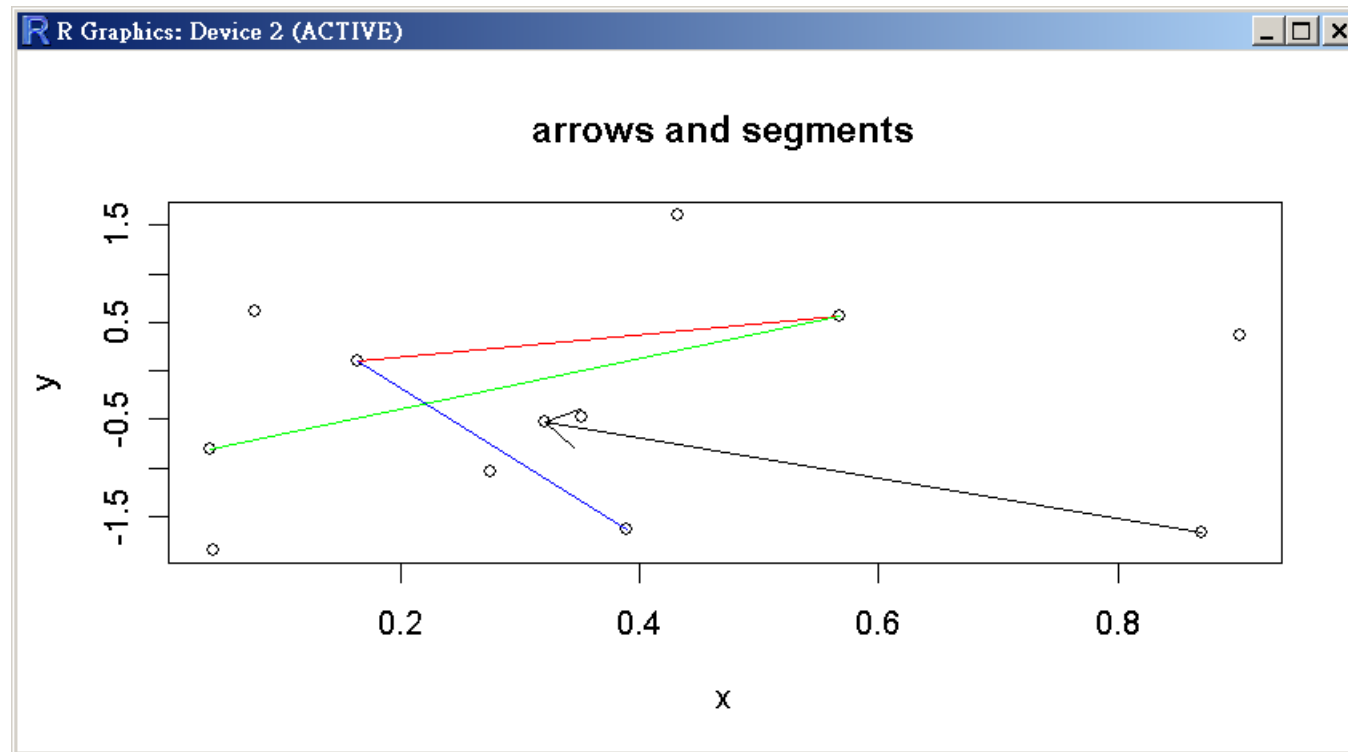
```
> sequence <- order(x)
> plot(x, y, pch=16, col="red")
> lines(x[sequence], y[sequence])
```



箭頭arrows與線段segments

37/216

```
> 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"))
```



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

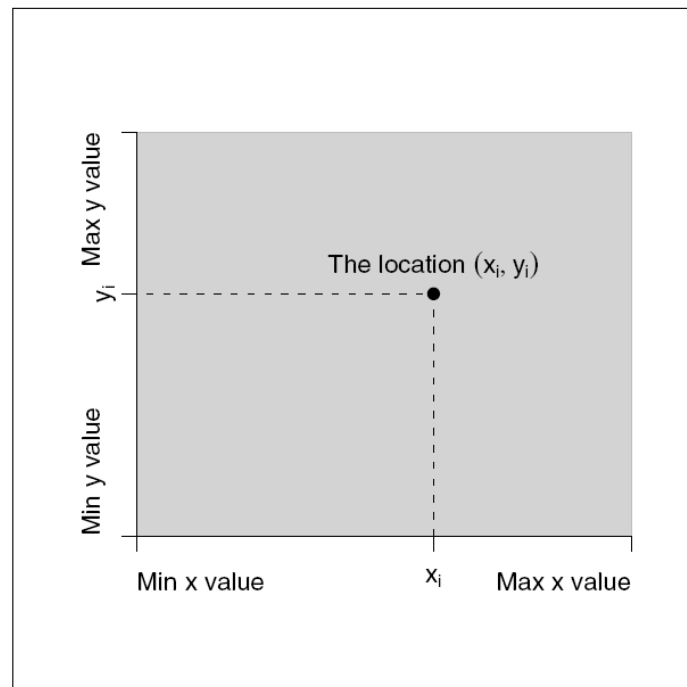
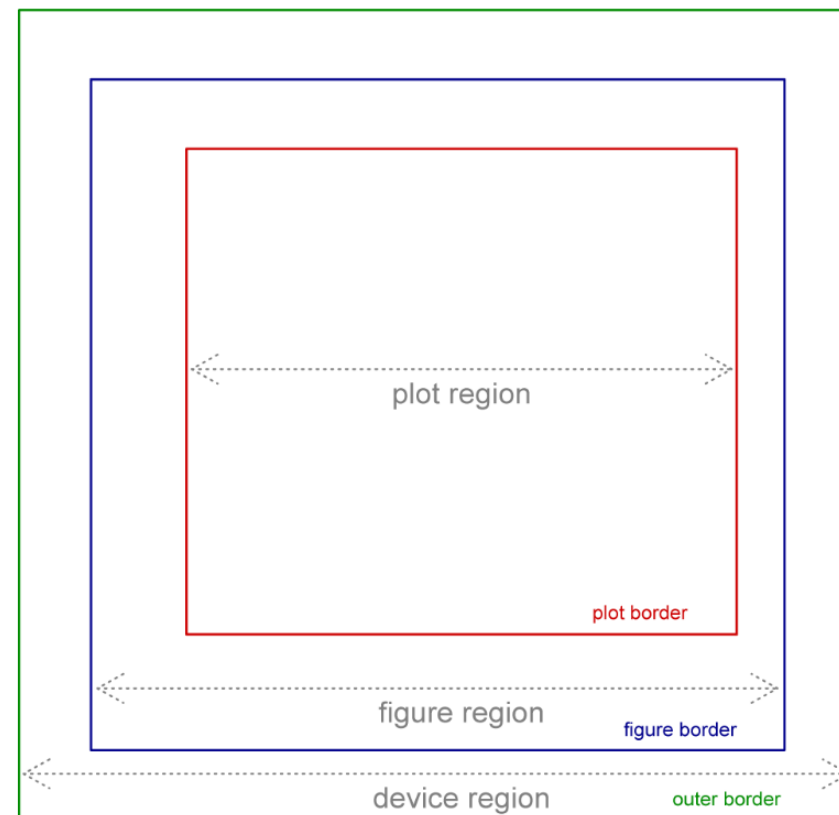


Figure 3.3

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



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

高階圖形調控項 (High-Level Graphics Setting)

Table 3.1

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

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

>?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
mfcpl	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.

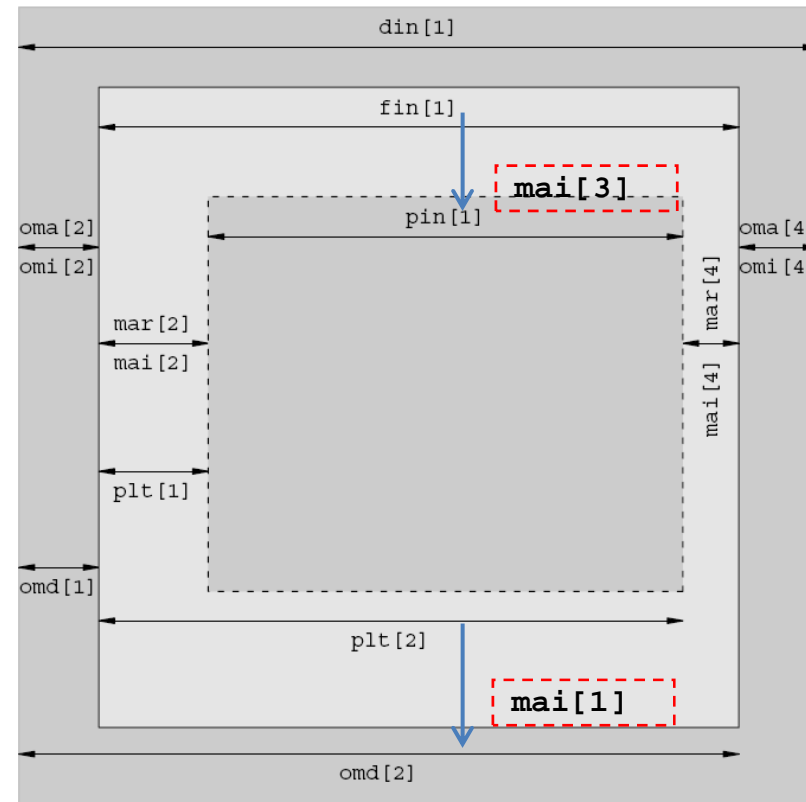


Figure 3.13

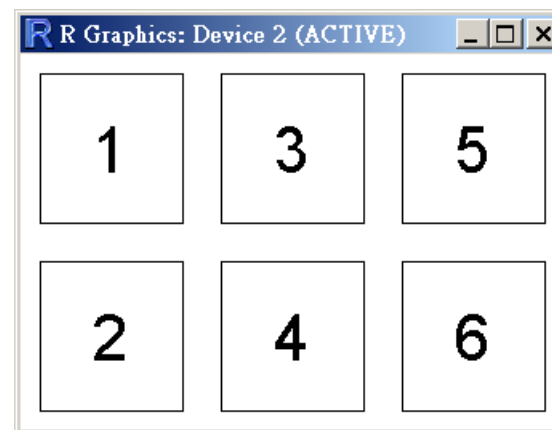
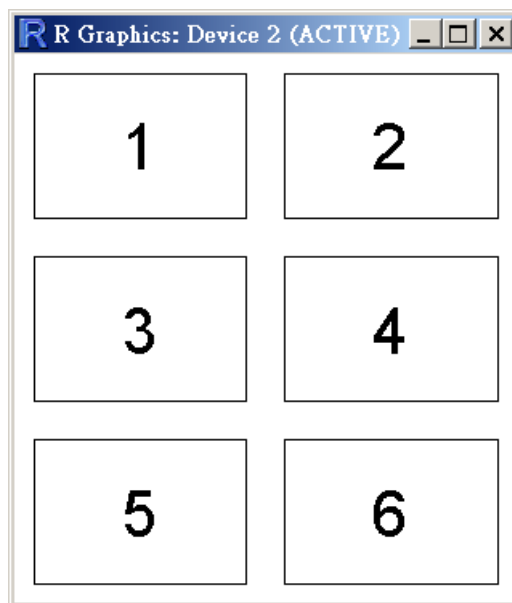
Graphics state settings controlling plot regions. These are some of the settings that control the widths and horizontal locations of the plot regions. For ease of comparison, this diagram has the same layout as Figure 3.1: the central grey rectangle represents the plot region, the lighter grey rectangle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



一頁多張圖形: `mfrow`, `mfcop`

42/216

```
> 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), mfcop=c(2, 3))
```



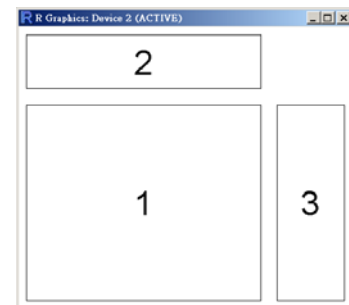
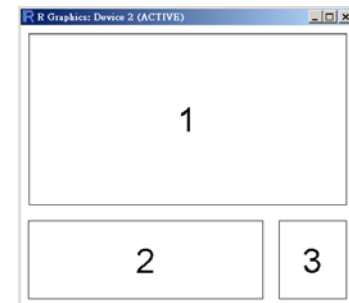
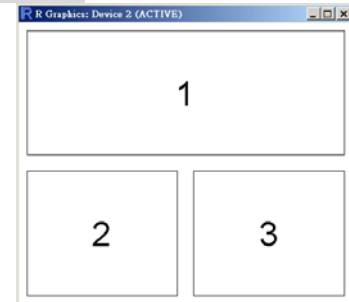
一頁多張圖形: layout

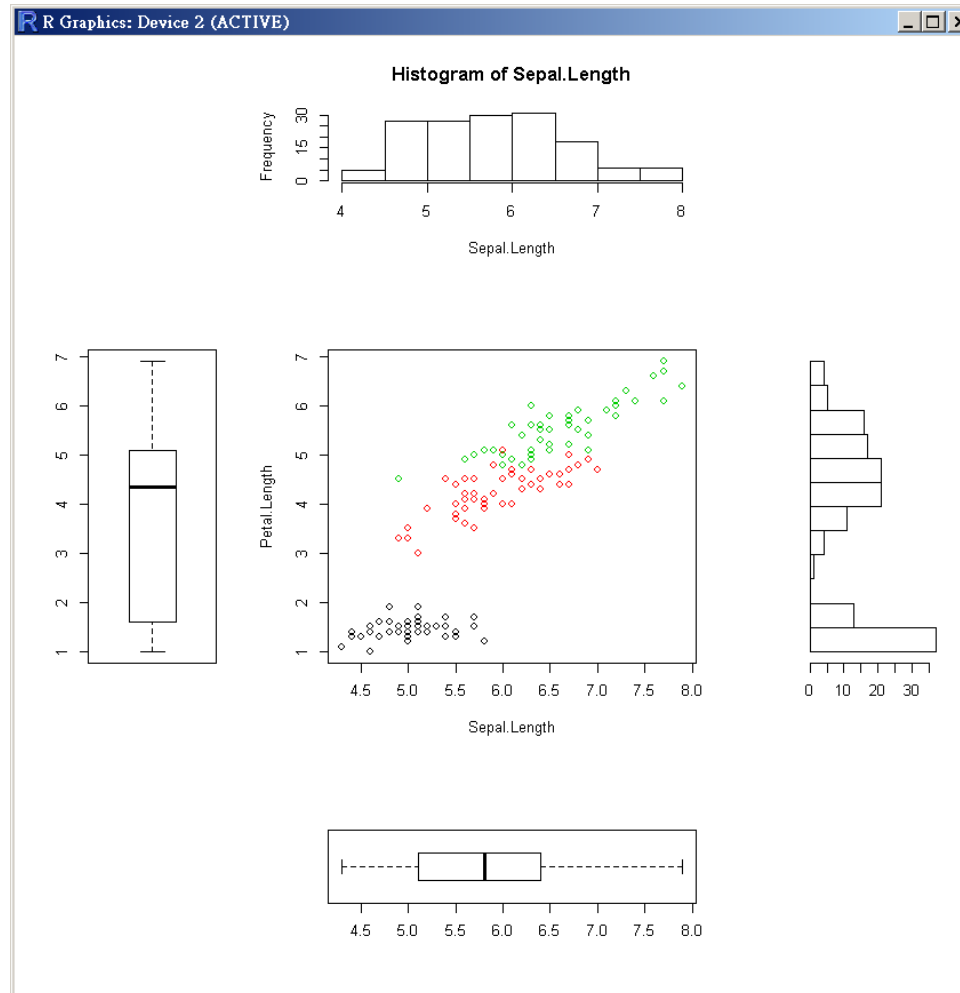
```
layout(mat, widths = rep.int(1, ncol(mat)),
       heights = rep.int(1, nrow(mat)), respect = FALSE)
```

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

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

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



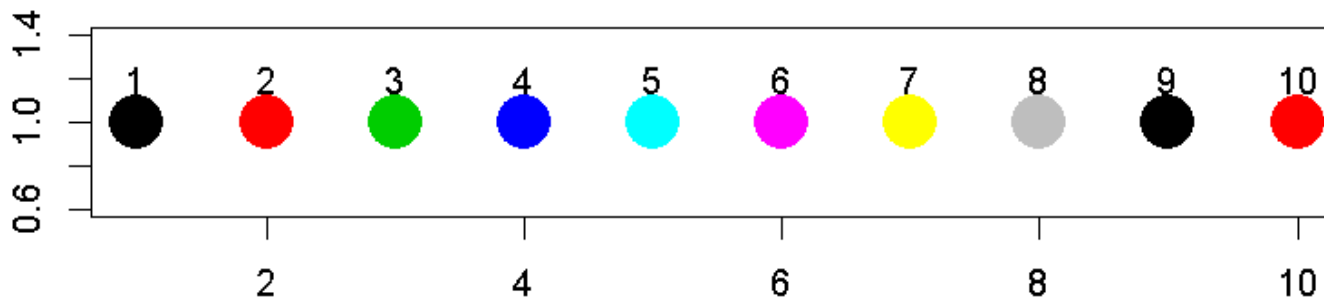


See also: `par(fig)`

顏色 (Color)

- Specify the color of data symbols, lines, text that are drawn in the plot region.
- `col`, `fg`, `bg`
 - `col.axis` (axes), `col.lab` (label), `col.main` (titles), `col.sub` (sub-titles)
 - `fg`: color of axes and borders on plots.
 - `bg`: the color of the background for base graphics output. This color is used to fill the entire page.

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





colors() # full list of known names.

```
> colors()  
[1] "white" "aliceblue" "antiquewhite"  
[4] "antiquewhite1" "antiquewhite2" "antiquewhite3"  
[7] "antiquewhite4" "aquamarine" "aquamarine1"  
[10] "aquamarine2" "aquamarine3" "aquamarine4"  
[13] "azure" "azure1" "azure2"  
[16] "azure3" "azure4" "beige"  
[19] "bisque" "bisque1" "bisque2"  
[22] "bisque3" "bisque4" "black"  
[25] "blanchedalmond" "blue" "blue1"  
.....  
  
[637] "turquoise2" "turquoise3" "turquoise4"  
[640] "violet" "violetred" "violetred1"  
[643] "violetred2" "violetred3" "violetred4"  
[646] "wheat" "wheat1" "wheat2"  
[649] "wheat3" "wheat4" "whitesmoke"  
[652] "yellow" "yellow1" "yellow2"  
[655] "yellow3" "yellow4" "yellowgreen"
```

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

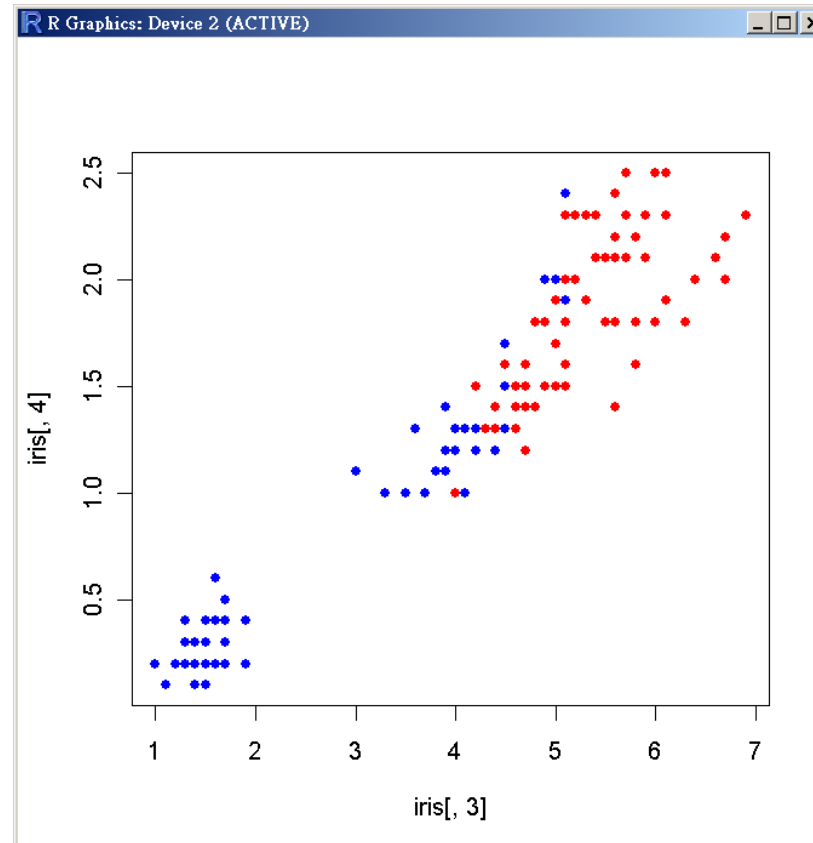

col2rgb()

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

Color Data Symbols

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

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





顏色集 (Color Sets)

49/216

Table 3.4

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

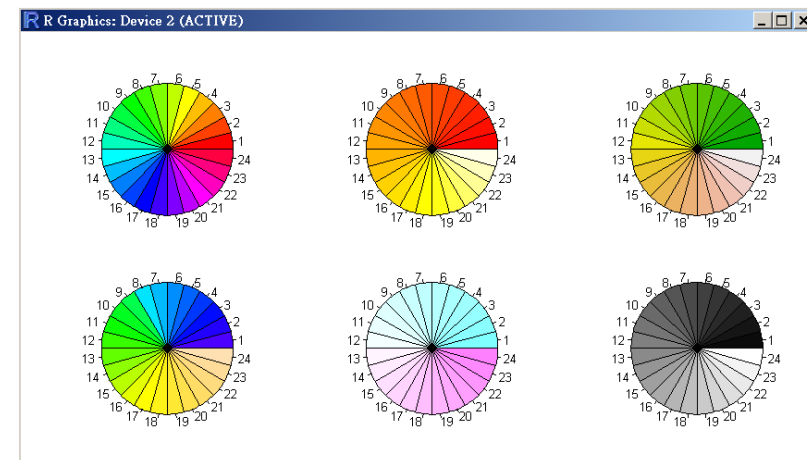
Name	Description
<code>rainbow()</code>	Colors vary from red through orange, yellow, green, blue, and indigo, to violet.
<code>heat.colors()</code>	Colors vary from white, through orange, to red.
<code>terrain.colors()</code>	Colors vary from white, through brown, to green.
<code>topo.colors()</code>	Colors vary from white, through brown then green, to blue.
<code>cm.colors()</code>	Colors vary from light blue, through white, to light magenta.
<code>grey()</code> or <code>gray()</code>	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

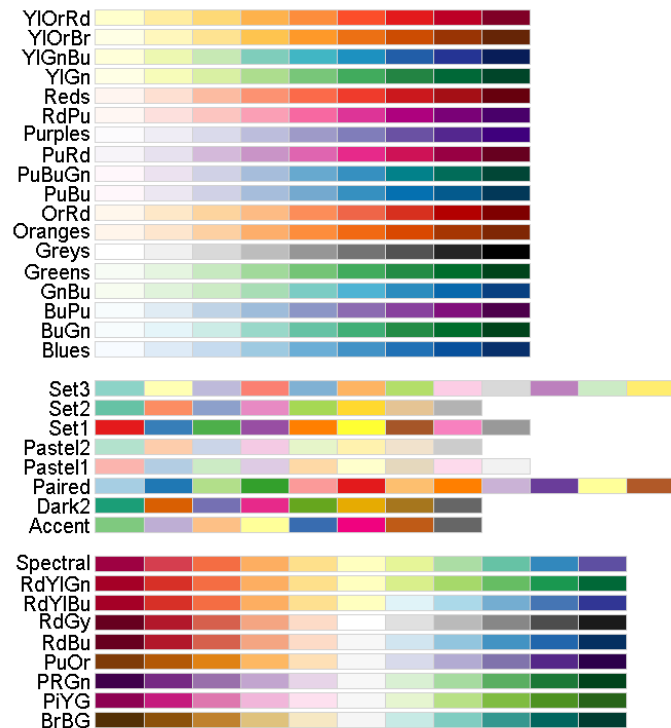
```
> par(mfrow=c(2,3))
> n <- 24
> pie(rep(1,n), col=rainbow(n))
> pie(rep(1,n), col=heat.colors(n))
> pie(rep(1,n), col=terrain.colors(n))
> pie(rep(1,n), col=topo.colors(n))
> pie(rep(1,n), col=cm.colors(n))
> pie(rep(1,n), col=grey(1:n/n))
```



RColorBrewer: ColorBrewer Palettes

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

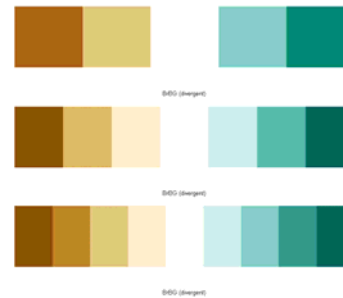
```
brewer.pal(n, name)
display.brewer.pal(n, name)
display.brewer.all(n=NULL, type="all", select=NULL, exact.n=TRUE, colorblindFriendly=FALSE)
brewer.pal.info
# n: Number of different colors in the palette, mim=3, maximum depends
# name: A palette name from the lists below
# type: One of the string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



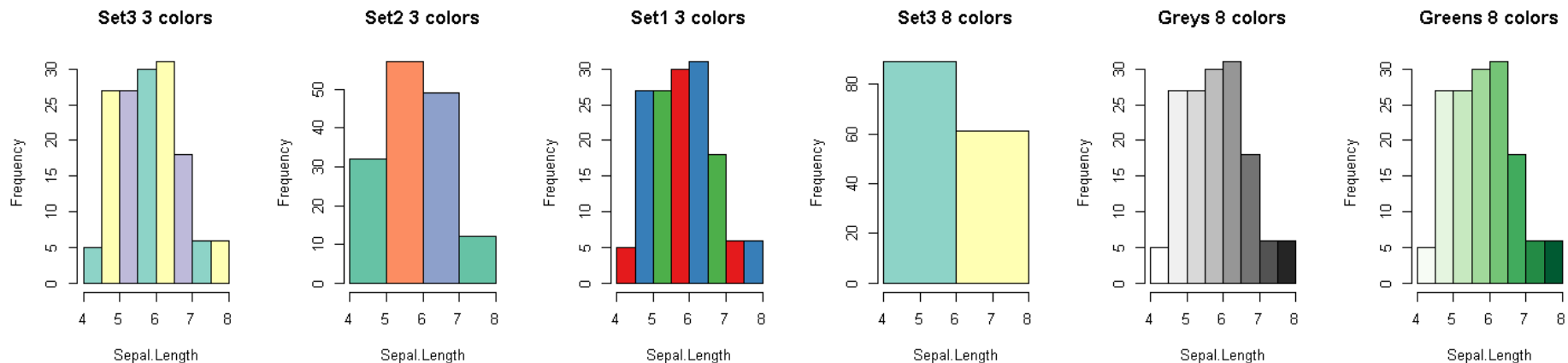
```
> library(RColorBrewer)
> brewer.pal.info
      maxcolors category colorblind
BrBG           11      div      TRUE
PiYG           11      div      TRUE
...
Spectral        11      div     FALSE
Accent          8      qual     FALSE
Dark2           8      qual      TRUE
...
YlOrBr          9     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

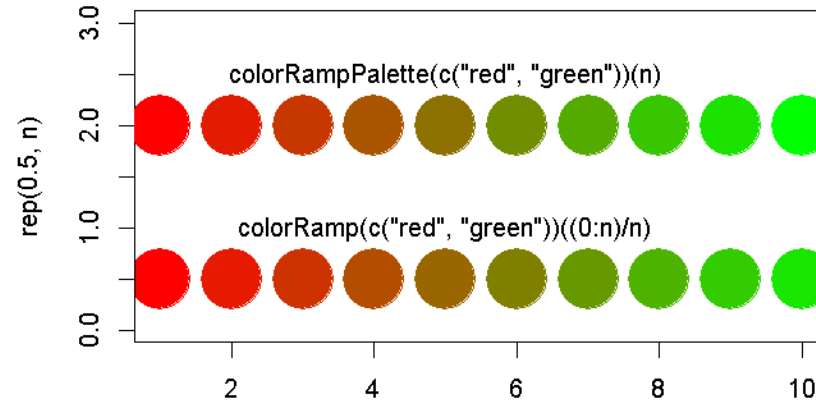


colorRamp {grDevices}: Color interpolation

```

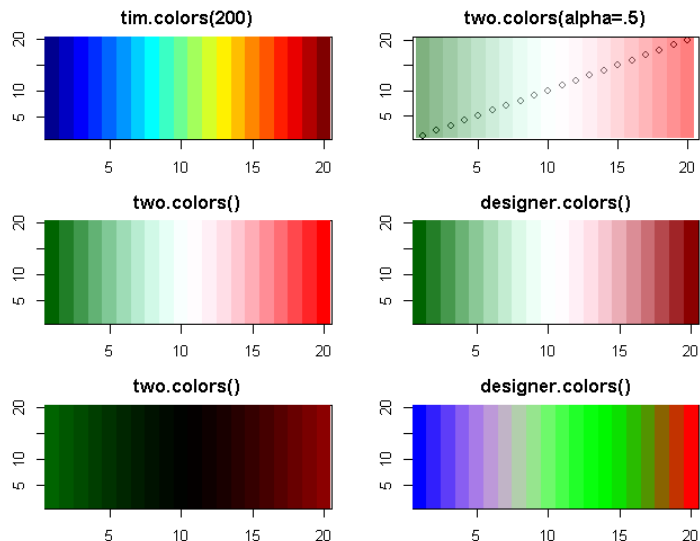
> n <- 10
> (col.a <- colorRamp(c("red", "green"))((0:n)/n)) #(x) , x in [0,1]
      [,1] [,2] [,3]
[1,] 255.0  0.0  0
[2,] 229.5 25.5  0
[3,] 204.0 51.0  0
[4,] 178.5 76.5  0
[5,] 153.0 102.0 0
[6,] 127.5 127.5 0
[7,] 102.0 153.0 0
[8,]  76.5 178.5 0
[9,]  51.0 204.0 0
[10,] 25.5 229.5 0
[11,]  0.0 255.0  0
> rgb(col.a/255)
      [,1] [,2] [,3]
[1] "#FF0000" "#E61A00" "#CC3300" "#B34D00" "#996600" "#808000" "#669900"
[8] "#4DB300" "#33CC00" "#19E600" "#00FF00"
> col.b <- colorRampPalette(c("red", "green"))(n) # (n)
> col.b
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
red      255  226  198  170  141  113   85   56   28    0
green     0   28   56   85  113  141  170  198  226  255
blue      0    0    0    0    0    0    0    0    0    0
> plot(1:n, rep(0.5, n), pch=20, col=rgb(col.a/255), cex=8, ylim=c(0, 3))
> text(5, 1, 'colorRamp(c("red", "green"))((0:n)/n)')
> points(1:n, rep(2, n), pch=20, col=col.b, cex=8)
> text(5, 2.5, 'colorRampPalette(c("red", "green"))(n)')

```



tim.colors {fields}: Some useful color tables 53/216 for images and tools to handle them

```
> 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()")
```



```
# 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)

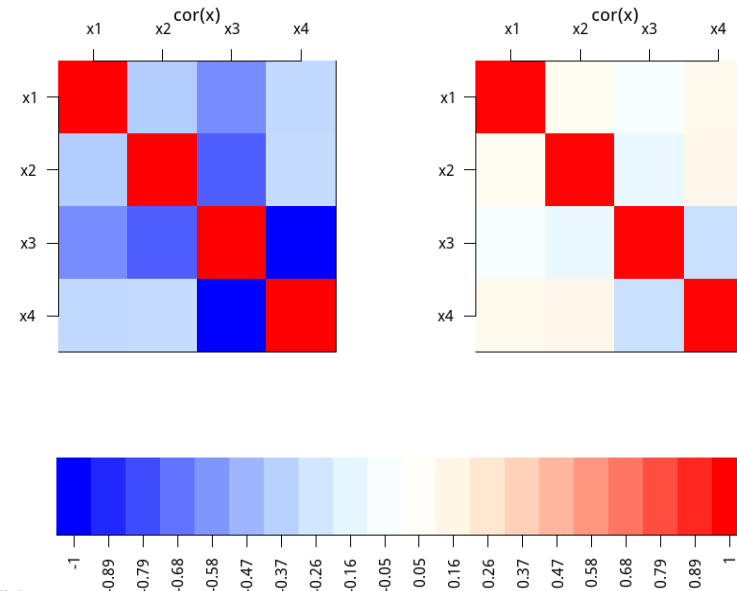
color.scale( z, col=tim.colors(256), zlim =NULL,
transparent.color="white", eps= 1e-8)
fieldsPlotColors( col,...)
```

A Color Spectrum within a range

```

> library(fields)
> cor.col <- two.colors(start="blue", middle="white", end="red")
> length(cor.col)
[1] 256
> range(cor.col)
[1] "#0000FF" "#FFFFFFE"
>
> n <- 100; p <- 4
> set.seed(12345)
> x <- matrix(rnorm(n*p), ncol=p)
> (rx <- cor(x))
      [,1] [,2] [,3] [,4]
[1,] 1.0000 0.1042 -0.0432 0.1406
[2,] 0.1042 1.0000 -0.1267 0.1510
[3,] -0.0432 -0.1267 1.0000 -0.2920
[4,] 0.1406 0.1510 -0.2920 1.0000
> #(-1, 0, 1) => (0, 1, 2) => (1, 128, 255)
> (range.col <- floor((1+range(rx))*127+1))
[1] 90 255
> # WRONG: image(t(rx)[,p:1], main="cor(x)", col=cor
> image(t(rx)[,p:1], main="cor(x)", col=cor.col[range.col[1]: range.col[2]],
+       xaxt="n", yaxt="n")
> axis(3, at=seq(0, 1, length.out=4), labels=paste0("x", 1:p))
> axis(2, at=seq(0, 1, length.out=4), labels=paste0("x", p:1), las=1)

```



```

> x <- 1:20; y <- 1:20
> z <- outer(x, rep(1,20), "+")
> obj <- list(x=x, y=y, z=z)
> image(obj, col=cor.col, xaxt="n", ylab="", yaxt="n")
> axis(1, at=x, labels=round(seq(-1, 1, length.out=20), 2), las=2)

```



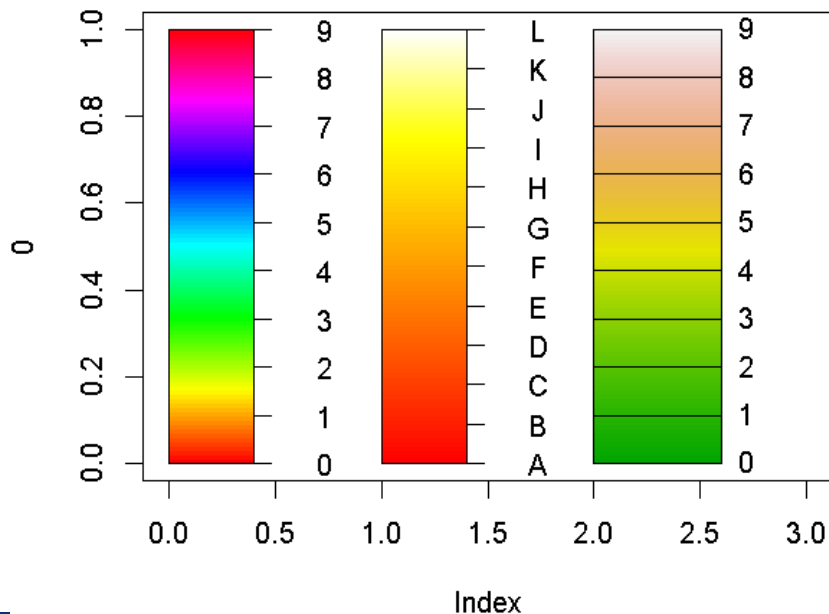
colorlegend {corrplot}

Draw color legend

Usage:

```
colorlegend(colbar, labels, at = NULL, xlim = c(0, 1), ylim = c(0, 1),  
vertical = TRUE, ratio.colbar = 0.4, lim.segment = "auto",  
align = c("c", "l", "r"), addlabels = TRUE, ...)
```

```
> # install.packages("corrplot")  
> library(corrplot)  
> # par(mar = rep(0,4))  
> plot(0, xlim = c(0, 3), ylim = c(0, 1), type = "n")  
> colorlegend(rainbow(100), 0:9)
```



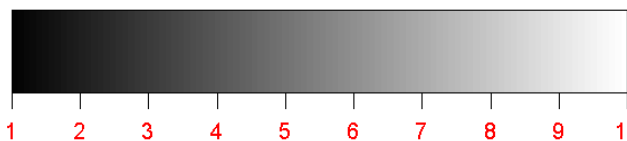
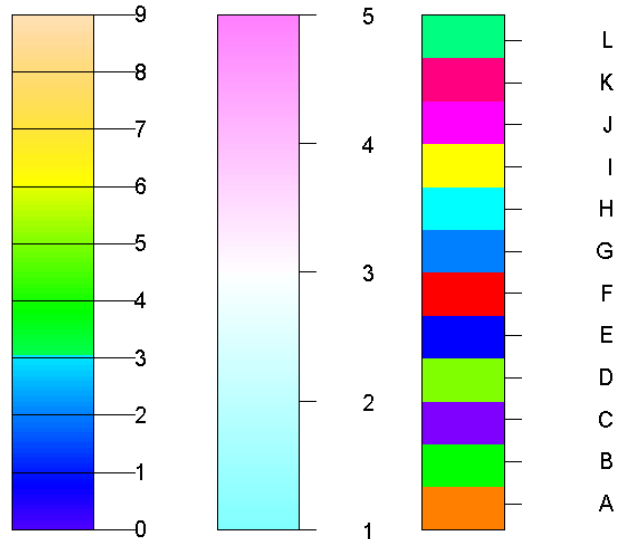
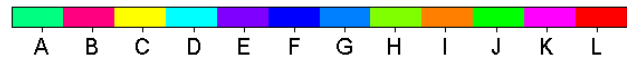
```
> colorlegend(heat.colors(100),  
+           LETTERS[1:12], xlim = c(1, 2))  
> colorlegend(terrain.colors(100),  
+           0:9,  
+           ratio.colbar = 0.6,  
+           lim.segment = c(0, 0.6),  
+           xlim = c(2, 3), align = "l")
```



colorlegend {corrplot}

Draw color legend

```
> par(mar = rep(0,4))  
> plot(0, xlim = c(3, 6), ylim = c(-0.5, 1.2), type = "n")  
> colorlegend(topo.colors(100), 0:9, lim.segment = c(0,0.6),  
+           xlim = c(3,4), align = "l", offset = 0)  
> colorlegend(cm.colors(100), 1:5, xlim = c(4,5))  
> colorlegend(sample(rainbow(12)), labels = LETTERS[1:12],  
+           at = seq(0.05, 0.95, len = 12), xlim = c(5, 6), align = "r")
```



```
> colorlegend(sample(rainbow(12)),  
+           labels = LETTERS[1:12],  
+           at = seq(0.05, 0.95, len = 12),  
+           xlim = c(3, 6),  
+           ylim = c(1.1, 1.2),  
+           vertical = FALSE)
```

```
> colorlegend(colbar = grey(1:100 / 100),  
+           1:10,  
+           col = "red",  
+           align = "l",  
+           xlim = c(3, 6),  
+           ylim = c(-0.5, -0.1),  
+           vertical = FALSE)
```


線條 (Lines)

Five graphics state settings for lines:

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

Customized line type via a string of digits:

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



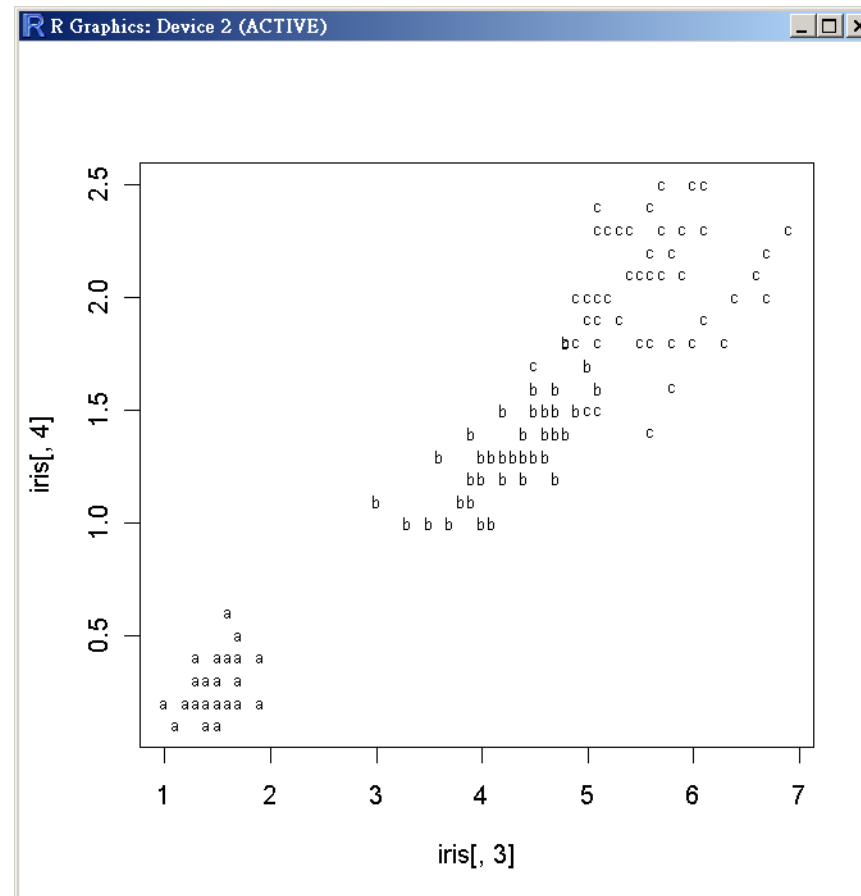
`lty="431313"`

Integer	Sample line	String
<i>Predefined</i>		
0		"blank"
1	—————	"solid"
2	- - - - -	"dashed"
3	"dotted"
4	.-.-.-.-.	"dotdash"
5	- - - - -	"longdash"
6	- . - . - .	"twodash"
<i>Custom</i>		
	"13"
	— — — — -	"F8"
	- . - . - .	"431313"
	- - - . - . - .	"22848222"

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

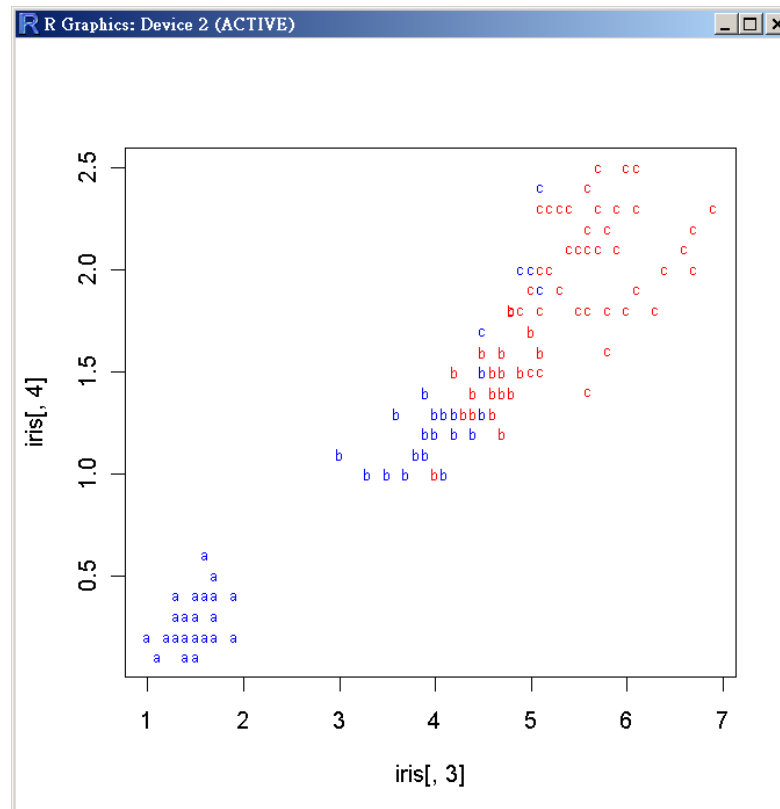
文字標號 (Text Labels)

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



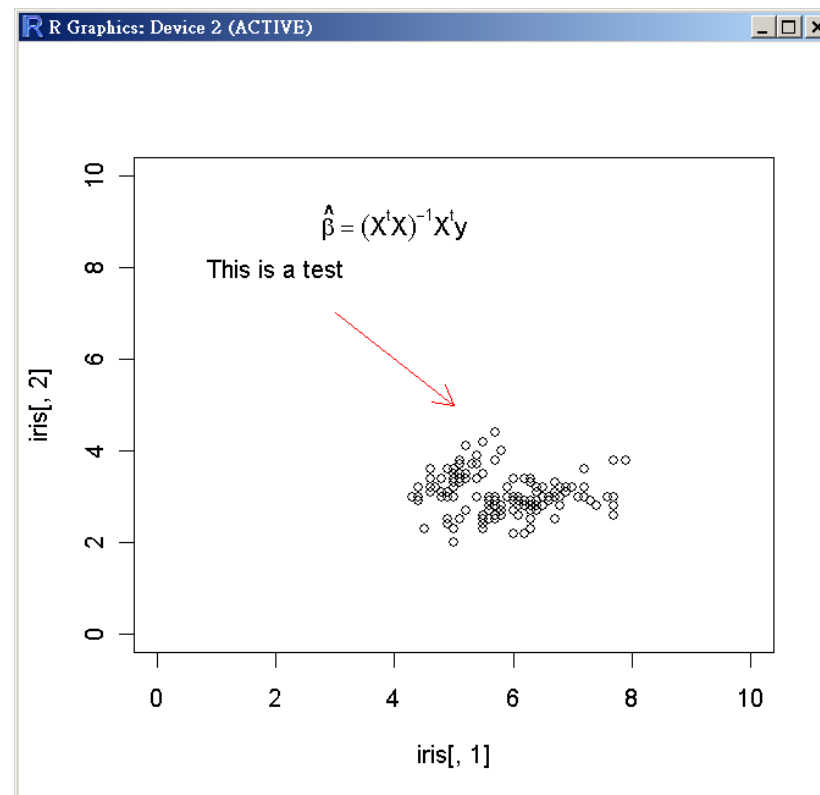
文字標號 (Text Labels)

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



圖上文字

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





Data Symbols

A fixed set of 26 data symbols:

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

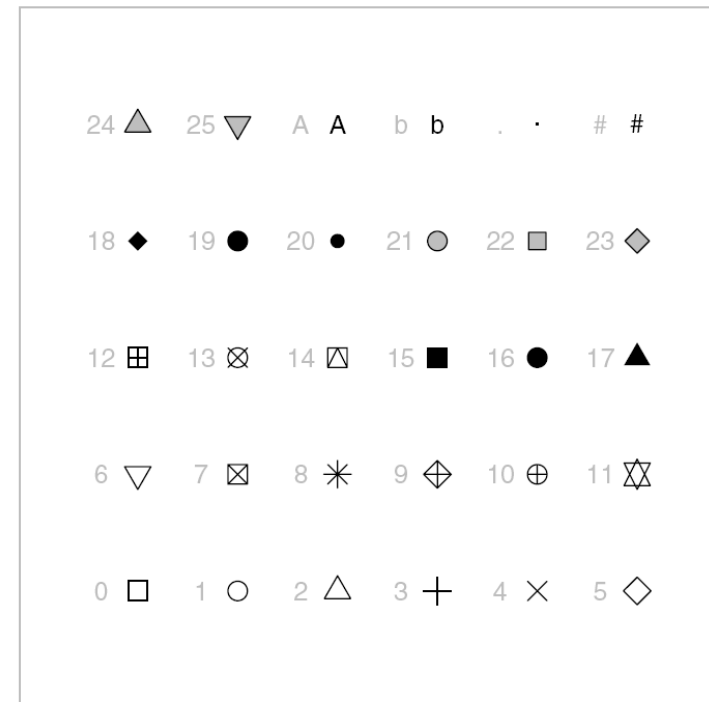
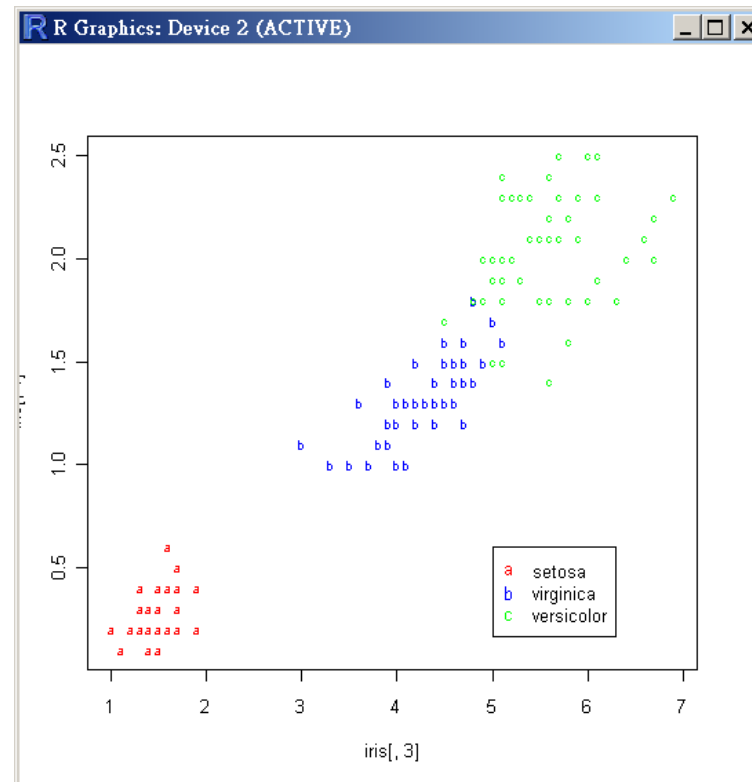


Figure 3.10
Data symbols available in R. A particular data symbol is selected by specifying an integer between 0 and 25 or a single character for the pch graphical setting. In the diagram, the relevant integer or character pch value is shown in grey to the left of the relevant symbol.

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

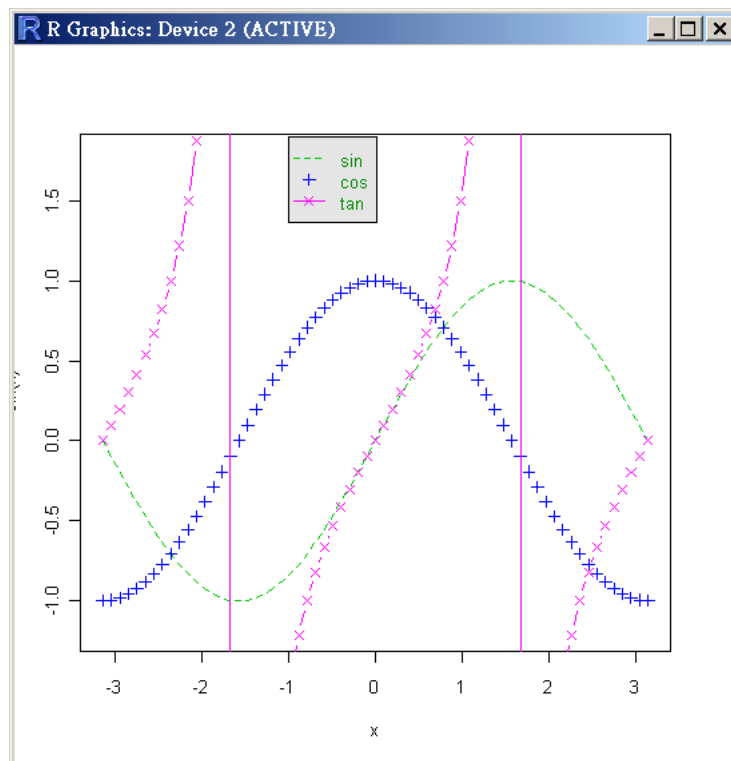
Legend

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



Legend

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



More Examples:
> ?legend

axis: 座標軸

■ 語法

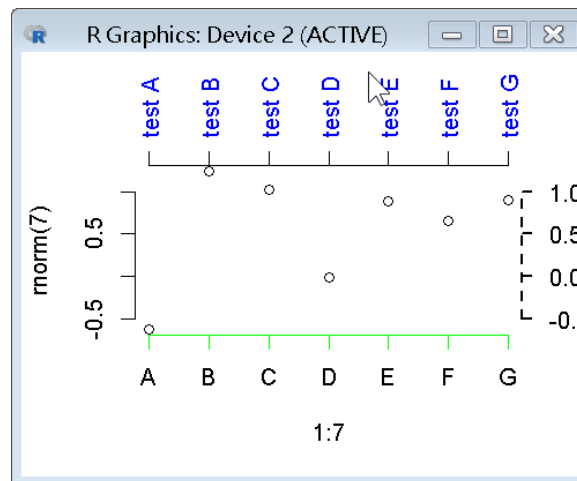
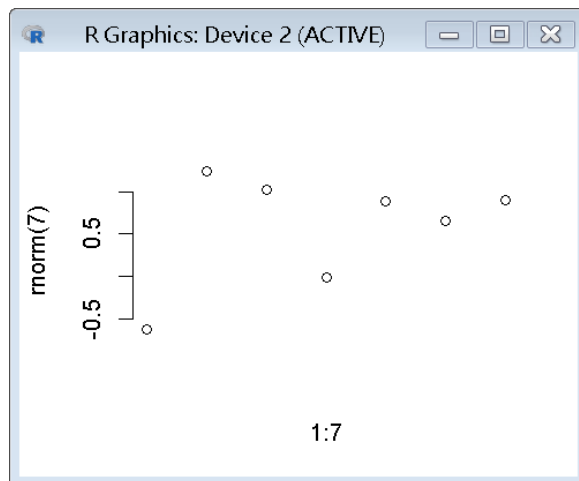
```
axis(side, at = NULL, labels = TRUE, tick = TRUE,  
line = NA, pos = NA, outer = FALSE, font = NA,  
lty = "solid", lwd = 1, lwd.ticks = lwd, col = NULL,  
col.ticks = NULL, hadj = NA, padj = NA, ...)
```

■ 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

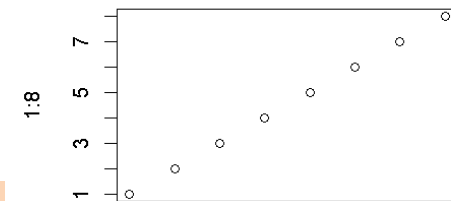
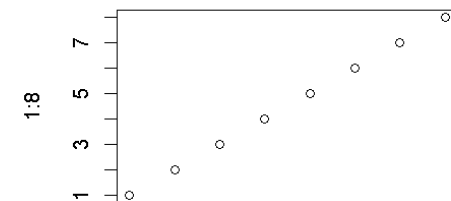
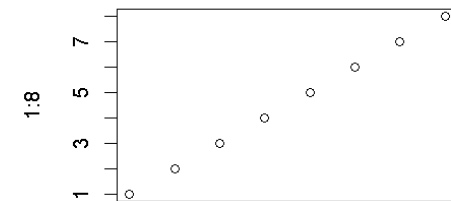
```
plot(1:7, rnorm(7), xaxt = "n", frame = FALSE)
axis(1, 1:7, LETTERS[1:7], col = "green")
axis(3, 1:7, paste("test", LETTERS[1:7]), col.axis = "blue", las=2)
axis(4, lty=2, lwd = 2, las=2)
```



```
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)
```

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

xpd: all plotting is clipped to the figure region



X Axis Label

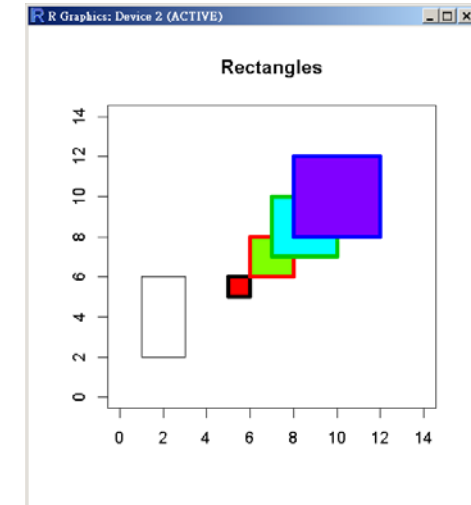


矩形 · Draw Symbols

66/216

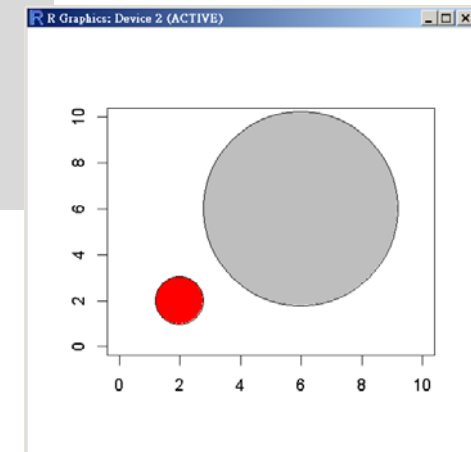
(Circles, Squares, Stars, Thermometers, Boxplots)

```
> 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}"*



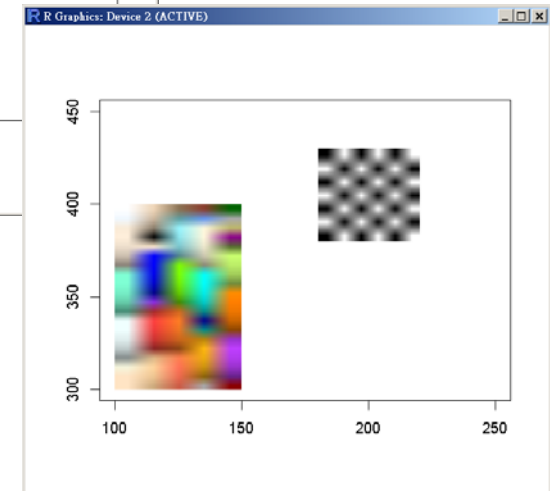
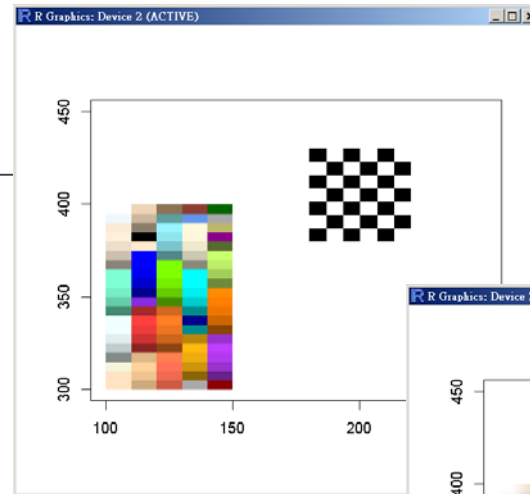
Draw One or More Raster Images

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

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

```
> matrix(0:1, ncol = 6, nrow = 7)
  [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0   1   0   1   0   1
[2,]  1   0   1   0   1   0
[3,]  0   1   0   1   0   1
[4,]  1   0   1   0   1   0
[5,]  0   1   0   1   0   1
[6,]  1   0   1   0   1   0
[7,]  0   1   0   1   0   1

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



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

彩色影像轉成灰階

```

> Transformers.f <- Image(flip(Transformers))
> # convert RGB to grayscale
> rgb.weight <- c(0.2989, 0.587, 0.114)
> Transformers.gray <- rgb.weight[1] * imageData(Transformers.f)[,,1] +
+                       rgb.weight[2] * imageData(Transformers.f)[,,2] +
+                       rgb.weight[3] * imageData(Transformers.f)[,,3]
> dim(Transformers.gray)
[1] 300 421
> Transformers.gray[1:5, 1:5]
      [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
[4,]    0    0    0    0    0
[5,]    0    0    0    0    0
> par(mfrow=c(1,2), mai=c(0.1, 0.1, 0.1, 0.1))
> image(Transformers.gray, col = grey(
+ seq(0, 1, length = 256)), xaxt="n", yaxt="n")
> image(Transformers.gray, col = rainbow(256),
+ xaxt="n", yaxt="n")

```

Converting RGB to grayscale/intensity

<http://stackoverflow.com/questions/687261/converting-rgb-to-grayscale-intensity>



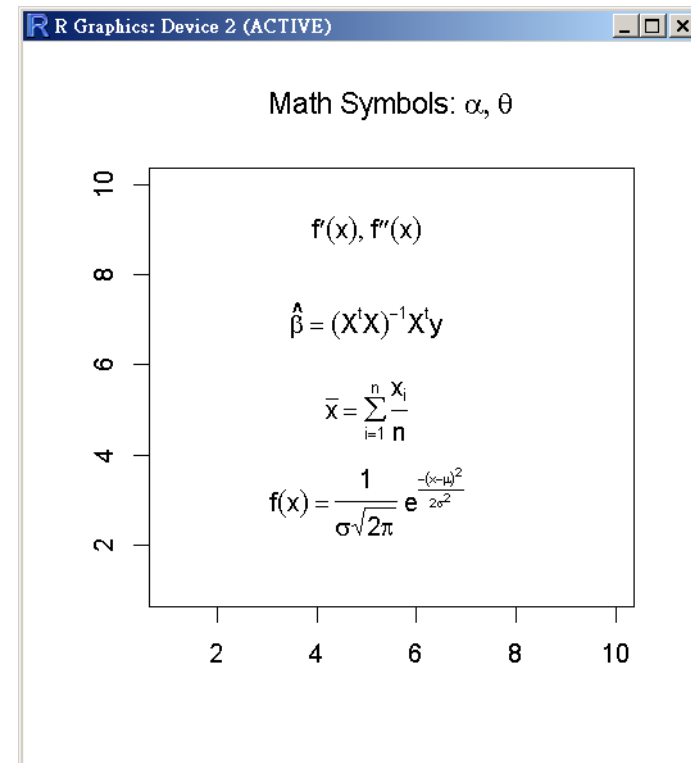


圖形上的數學符號: plotmath

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

demo(plotmath)

Grouping		group("(", list(a, b), ")")	(a, b)
$(x + y) * z$	$(x+y)z$	bgroup("(", atop(x, y), ")")	$\begin{pmatrix} x \\ y \end{pmatrix}$
$x^y + z$	$x^y + z$	group(lceil, x, rceil)	$\lceil x \rceil$
$x^{(y+z)}$	$x^{(y+z)}$	group(floor, x, rfloor)	$\lfloor x \rfloor$
$x^{\{y+z\}}$	x^{y+z}	group(" ", x, " ")	$ x $





demo(plotmath)

Arithmetic Operators		Lists	
$x + y$	<code>x+y</code>	<code>list(x, y, z)</code>	x, y, z
$x - y$	<code>x-y</code>	Relations	
$x * y$	<code>xy</code>	<code>x == y</code>	$x = y$
x / y	<code>x/y</code>	<code>x != y</code>	$x \neq y$
$x \%+-% y$	<code>x±y</code>	<code>x < y</code>	$x < y$
$x\%/y$	<code>x÷y</code>	<code>x <= y</code>	$x \leq y$
$x \%*% y$	<code>x×y</code>	<code>x > y</code>	$x > y$
$x \%.% y$	<code>x·y</code>	<code>x >= y</code>	$x \geq y$
$-x$	<code>-x</code>	<code>x %~~% y</code>	$x \approx y$
$+x$	<code>+x</code>	<code>x %~=% y</code>	$x \equiv y$
Sub/Superscripts		<code>x %==% y</code>	$x \equiv y$
$x[i]$	x_i	<code>x %prop% y</code>	$x \propto y$
x^2	x^2	<code>x %~% y</code>	$x \sim y$
Juxtaposition		Typeface	
$x * y$	<code>xy</code>	<code>plain(x)</code>	x
<code>paste(x, y, z)</code>	<code>xyz</code>	<code>italic(x)</code>	x
Radicals		<code>bold(x)</code>	\mathbf{x}
<code>sqrt(x)</code>	\sqrt{x}	<code>bolditalic(x)</code>	\mathbf{x}
<code>sqrt(x, y)</code>	$\sqrt[y]{x}$	<code>underline(x)</code>	\underline{x}

Ellipsis		Arrows	
<code>list(x[1], ..., x[n])</code>	x_1, \dots, x_n	<code>x %<=>% y</code>	$x \leftrightarrow y$
<code>x[1] + ... + x[n]</code>	$x_1 + \dots + x_n$	<code>x %->% y</code>	$x \rightarrow y$
<code>list(x[1], cdots, x[n])</code>	x_1, \dots, x_n	<code>x %<-% y</code>	$x \leftarrow y$
<code>x[1] + ldots + x[n]</code>	$x_1 + \dots + x_n$	<code>x %up% y</code>	$x \uparrow y$
Set Relations		<code>x %down% y</code>	$x \downarrow y$
<code>x %subset% y</code>	$x \subset y$	<code>x %<=>% y</code>	$x \Leftrightarrow y$
<code>x %subsetq% y</code>	$x \subseteq y$	<code>x %=>% y</code>	$x \Rightarrow y$
<code>x %supset% y</code>	$x \supset y$	<code>x %<=% y</code>	$x \Leftarrow y$
<code>x %supsetq% y</code>	$x \supseteq y$	<code>x %dblup% y</code>	$x \Uparrow y$
<code>x %notsubset% y</code>	$x \not\subset y$	<code>x %dbldown% y</code>	$x \Downarrow y$
<code>x %in% y</code>	$x \in y$	Symbolic Names	
<code>x %notin% y</code>	$x \notin y$	Alpha - Omega	$A - \Omega$
Accents		alpha - omega	$\alpha - \omega$
<code>hat(x)</code>	\hat{x}	phi1 + sigma1	$\varphi + \varsigma$
<code>tilde(x)</code>	\tilde{x}	Upsilon1	Υ
<code>ring(x)</code>	$\overset{\circ}{x}$	infinity	∞
<code>bar(xy)</code>	\overline{xy}	32 * degree	32°
<code>widehat(xy)</code>	\widehat{xy}	60 * minute	$60'$
<code>widetilde(xy)</code>	\widetilde{xy}	30 * second	$30''$

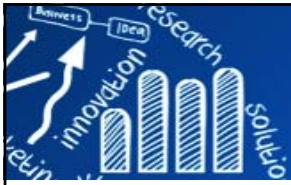


demo(plotmath)

Ellipsis		Arrows	
<code>list(x[1], ..., x[n])</code>	x_1, \dots, x_n	<code>x %<=>% y</code>	$x \leftrightarrow y$
<code>x[1] + ... + x[n]</code>	$x_1 + \dots + x_n$	<code>x %>=% y</code>	$x \rightarrow y$
<code>list(x[1], cdots, x[n])</code>	x_1, \cdots, x_n	<code>x %<=% y</code>	$x \leftarrow y$
<code>x[1] + ldots + x[n]</code>	$x_1 + \dots + x_n$	<code>x %up% y</code>	$x \uparrow y$
Set Relations		<code>x %down% y</code>	$x \downarrow y$
<code>x %subset% y</code>	$x \subset y$	<code>x %<=>% y</code>	$x \Leftrightarrow y$
<code>x %subsetq% y</code>	$x \subseteq y$	<code>x %=>% y</code>	$x \Rightarrow y$
<code>x %supset% y</code>	$x \supset y$	<code>x %<=% y</code>	$x \Leftarrow y$
<code>x %supsetq% y</code>	$x \supseteq y$	<code>x %dblup% y</code>	$x \Uparrow y$
<code>x %notsubset% y</code>	$x \not\subset y$	<code>x %dbldown% y</code>	$x \Downarrow y$
<code>x %in% y</code>	$x \in y$	Symbolic Names	
<code>x %notin% y</code>	$x \notin y$	Alpha - Omega	$A - \Omega$
Accents		alpha - omega	$\alpha - \omega$
<code>hat(x)</code>	\hat{x}	phi1 + sigma1	$\varphi + \varsigma$
<code>tilde(x)</code>	\tilde{x}	Upsilon1	Υ
<code>ring(x)</code>	$\overset{\circ}{x}$	infinity	∞
<code>bar(xy)</code>	\overline{xy}	32 * degree	32°
<code>widehat(xy)</code>	\widehat{xy}	60 * minute	$60'$
<code>widetilde(xy)</code>	\widetilde{xy}	30 * second	$30''$

Style	
<code>displaystyle(x)</code>	x
<code>textstyle(x)</code>	x
<code>scriptstyle(x)</code>	x
<code>scriptscriptstyle(x)</code>	x
Spacing	
<code>x ~ ~y</code>	$x \quad y$
<code>x + phantom(0) + y</code>	$x + \quad + y$
<code>x + over(1, phantom(0))</code>	$x + \frac{1}{-}$
Fractions	
<code>frac(x, y)</code>	$\frac{x}{y}$
<code>over(x, y)</code>	$\frac{x}{y}$
<code>atop(x, y)</code>	$\frac{x}{y}$

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



Graphical Techniques: By Problem Category

73/216

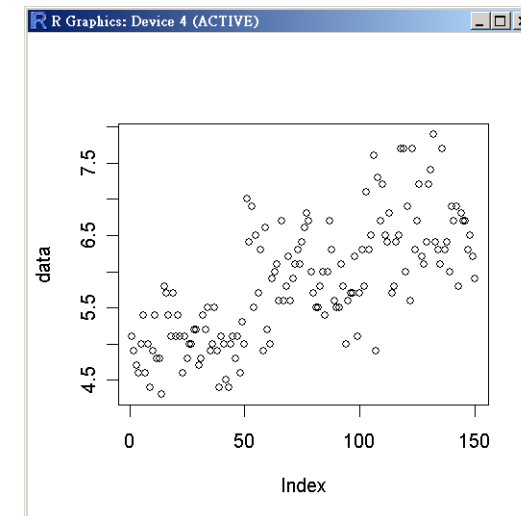
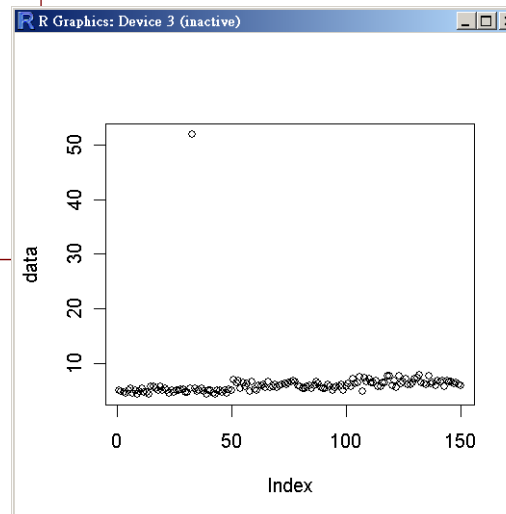
- **Univariate:** $y = c + e$: Run Sequence Plot (Index plot), Lag Plot, Histogram, Normal Probability Plot, Probability Plot Correlation Coefficient Plot (PPCC) Plot, Weibull Plot, Probability Plot, Box-Cox Linearity Plot, Box-Cox Normality Plot, Bootstrap Plot
- **Time Series:** $y = f(t) + e$: Run Sequence Plot, Spectral Plot, Autocorrelation Plot, Complex Demodulation Amplitude Plot, Complex Demodulation Phase Plot
- **One Factor:** $y = f(x) + e$: Scatter Plot, Box Plot, Bihistogram, Quantile-Quantile Plot, Mean Plot, Standard Deviation Plot
- **Multi-Factor/Comparative:** $y = f(x_p, x_1, x_2, \dots, x_k) + e$: Block Plot
- **Multi-Factor/Screening:** $y = f(x_1, x_2, x_3, \dots, x_k) + e$: DOE Scatter Plot, DOE Mean Plot, DOE Standard Deviation Plot, Contour Plot
- **Regression:** $y = f(x_1, x_2, x_3, \dots, x_k) + e$: Scatter Plot, 6-Plot (Scatter plot of the response and predicted values versus the independent variable; Scatter plot of the residuals versus the independent variable; Scatter plot of the residuals versus the predicted values; Lag plot of the residuals; Histogram of the residuals; Normal probability plot of the residuals.) Linear Correlation Plot, Linear Intercept Plot, Linear Slope Plot, Linear Residual Standard Deviation Plot
- **Interlab:** $(y_1, y_2) = f(x) + e$: Youden Plot
- **Multivariate:** (y_1, y_2, \dots, y_p) : Star Plot



索引圖 (Index Plot)

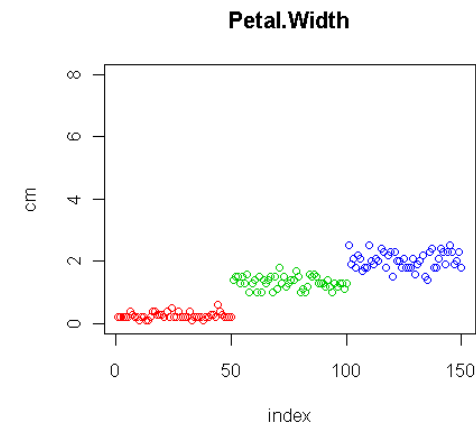
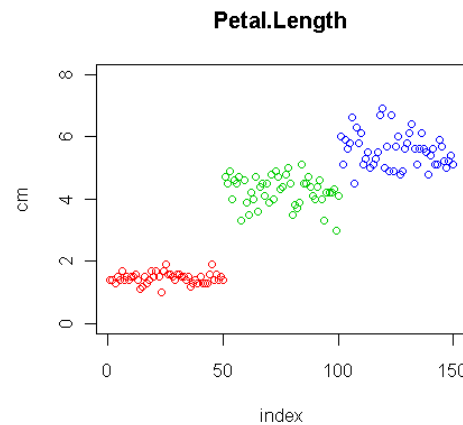
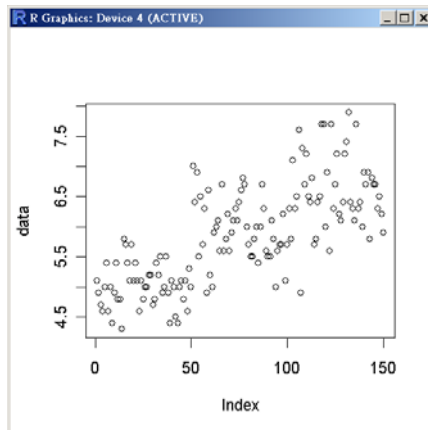
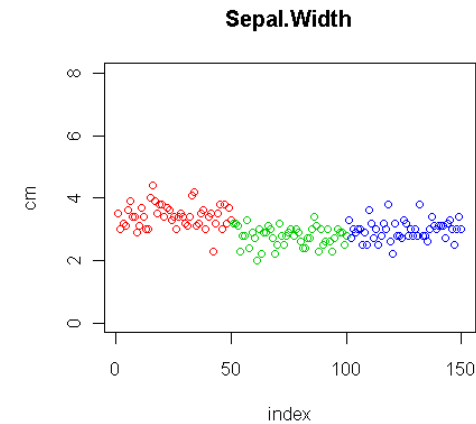
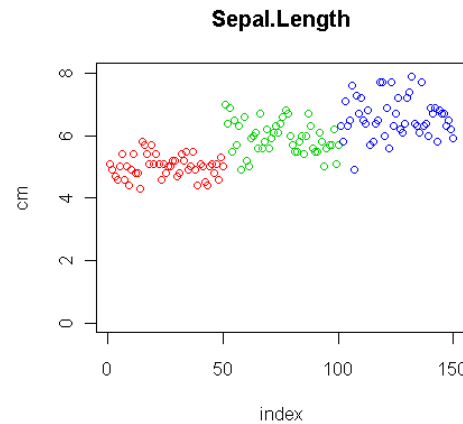
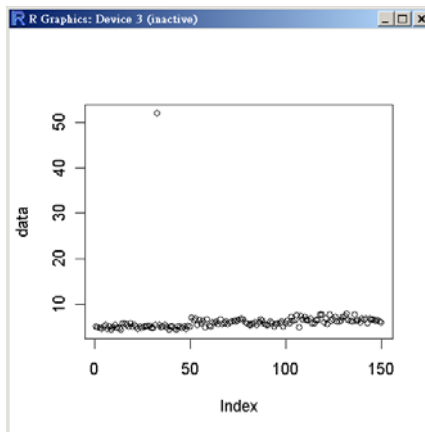
- Index plot takes a single argument which is a **continuous variable** and plots the **values** on the y axis, with the x coordinate determined by the **position** of the number in the vector.
- Useful for error checking.

```
> 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

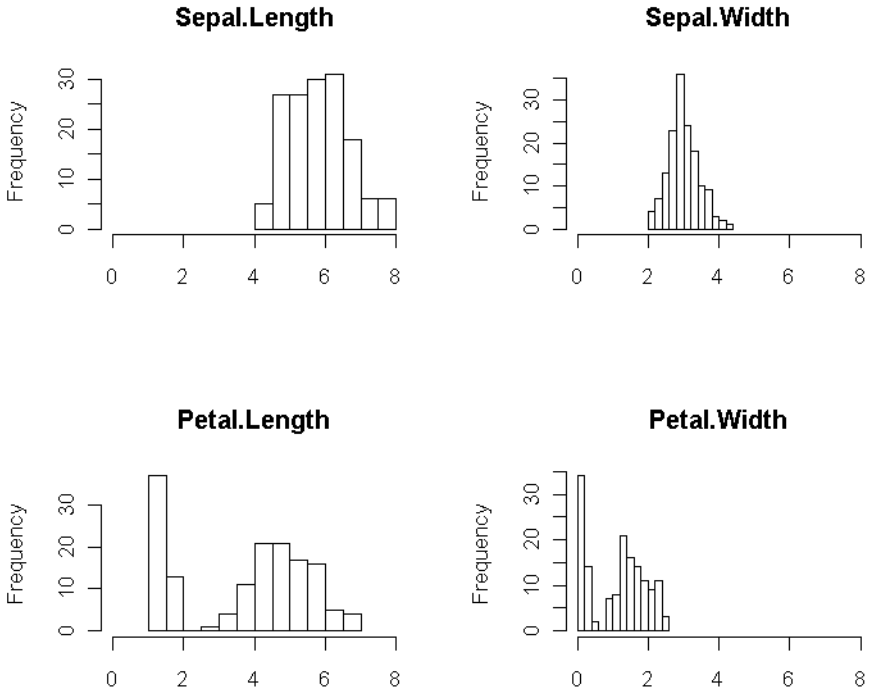
- Index plot takes a single argument which is a **continuous variable** and plots the **values** on the y axis, with the x coordinate determined by the **position** of the number in the vector.
- Useful for error checking.



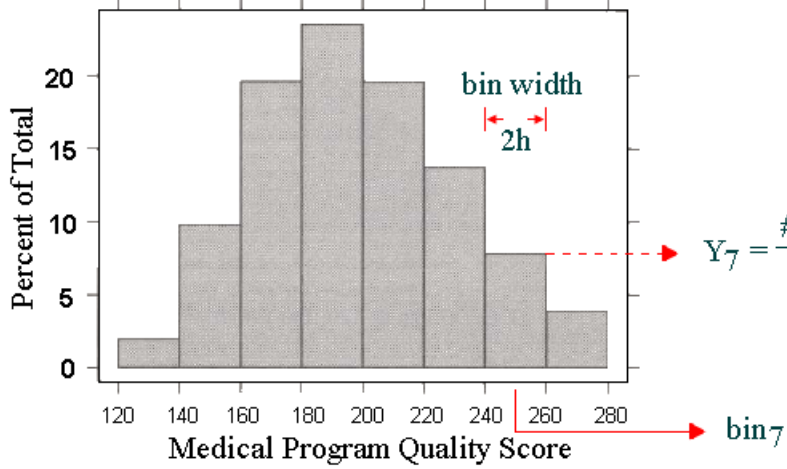
直方圖 (Histogram) (1/3)

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.



O. Bin origin at 120, bin widths of 20.



```
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]))
```

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

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

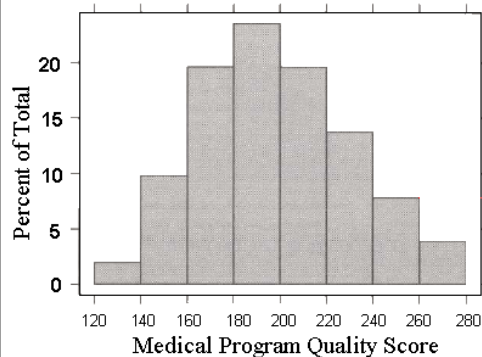
直方圖 (Histogram) (2/3)

- $1/2h$ 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.

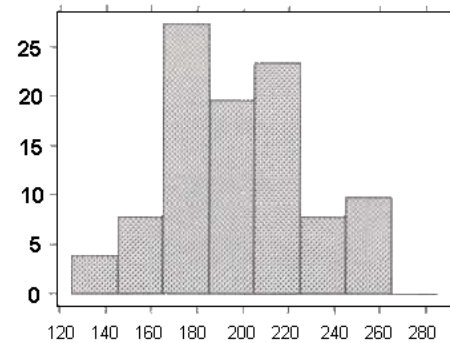
Disadvantage for displaying a variable's distribution:

- 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.

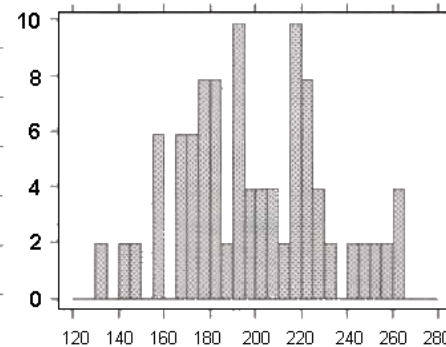
O. Bin origin at 120, bin widths of 20.



A. Bin origin at 125, bin widths of 20.



B. Bin origin at 120, bin widths of 5.



C. Bin origin at 120, bin widths of 10.

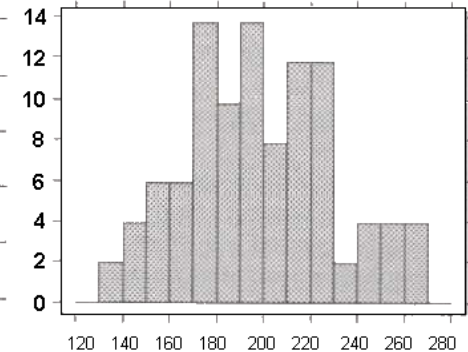
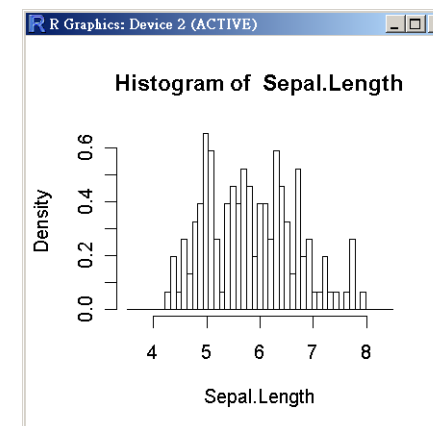
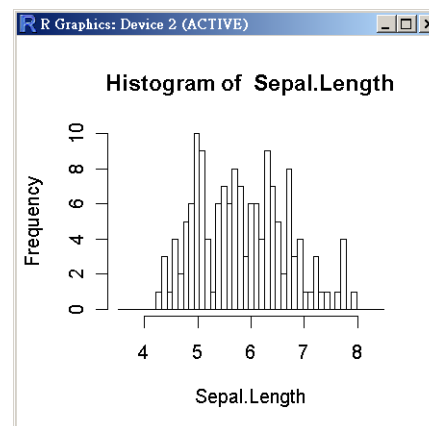
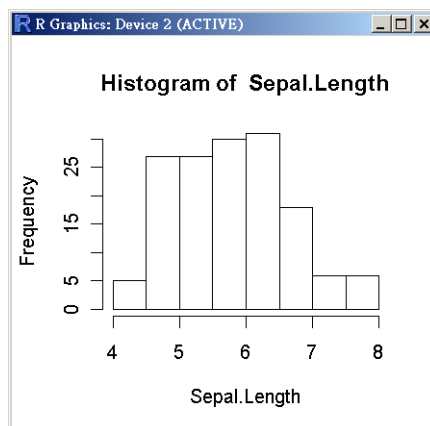


Figure Sources: Jacoby (1997).

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)
```



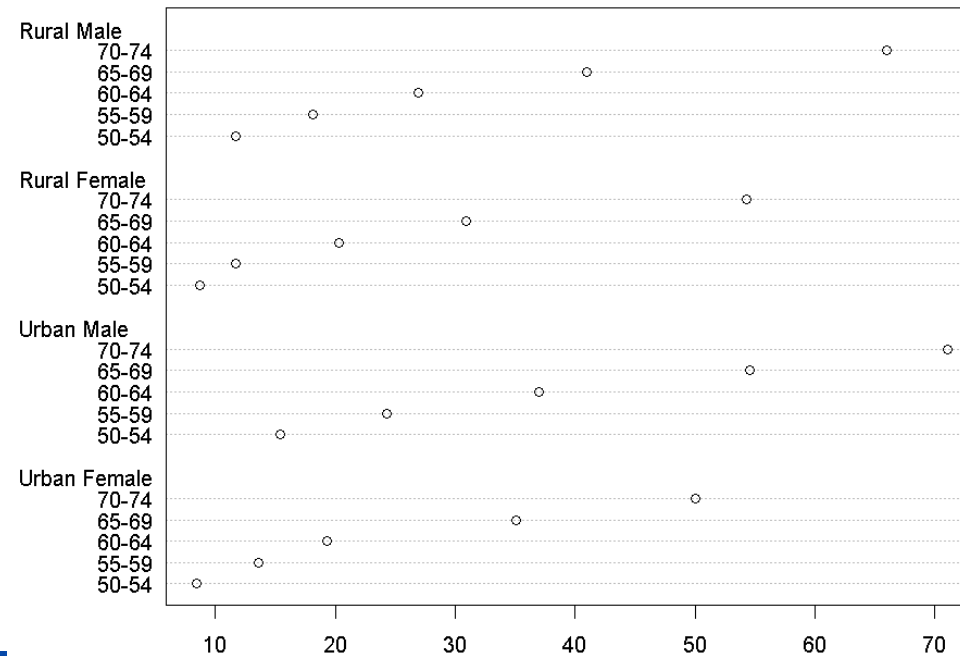
點圖 (Dotplot/Dotchart)

```

> dotchart(VADeaths, main = "Death Rates in Virginia - 1940")
> 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
> ?VADeaths

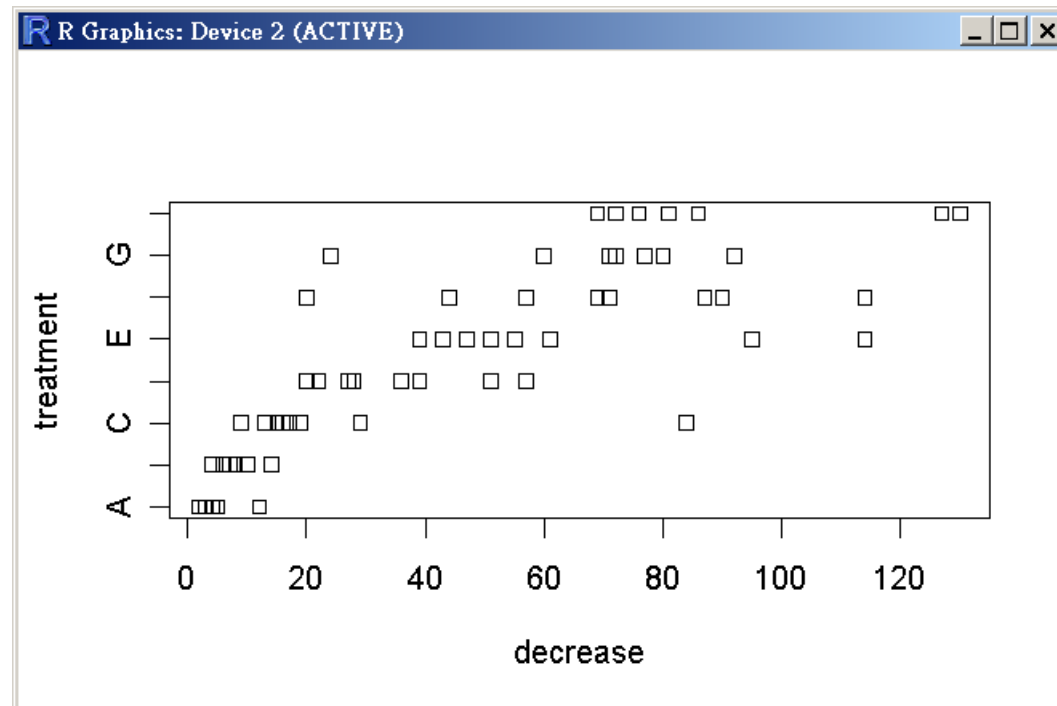
```

Death Rates in Virginia - 1940



帶形圖 (Stripchart)

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

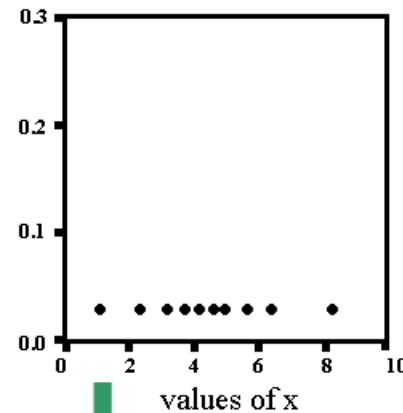


Density Plots (Smoothed Histograms) (1/3)

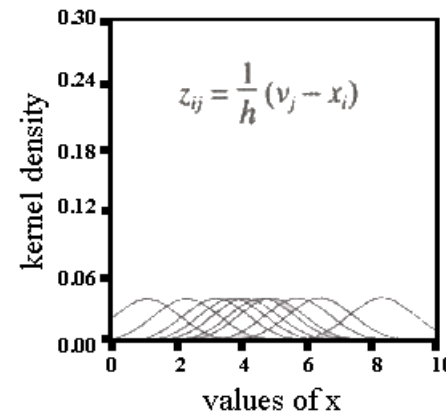
- Smoothed histograms overcome some of the disadvantages caused by the arbitrary, discrete bins used in traditional histogram.
- The relative height of the smooth curve corresponds to the local density.
- The overall height is adjusted so that the total area under the curve is approximately equal to 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.

Constructing a Smoothed Histogram (Jacoby, 1997)

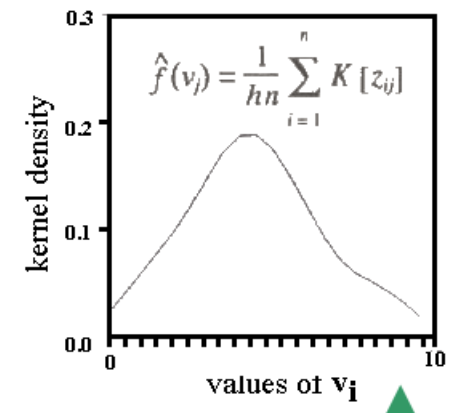
A. Unidimensional scatterplot of 10 data points



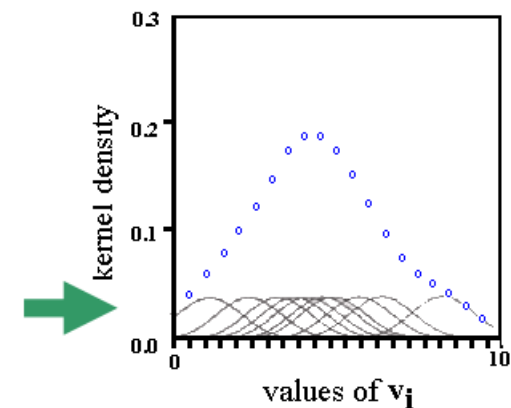
B. Data points shown as kernel densities



D. Final smoothed histogram



C. Summing kernel densities at the 20 v_i



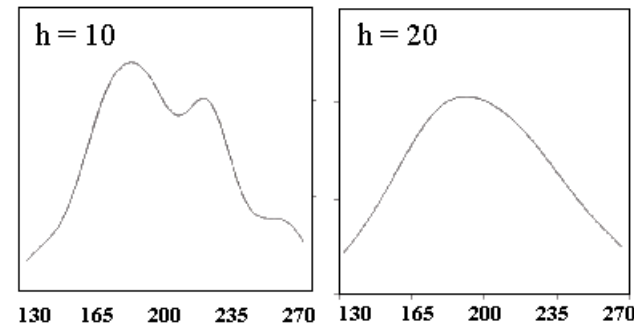
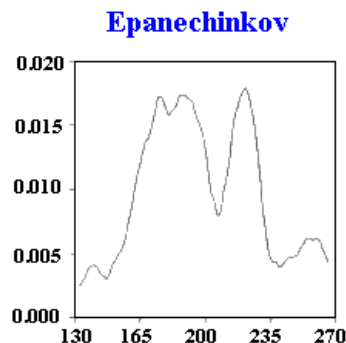
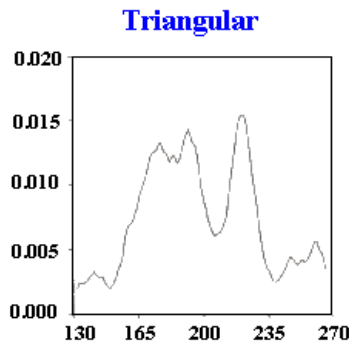
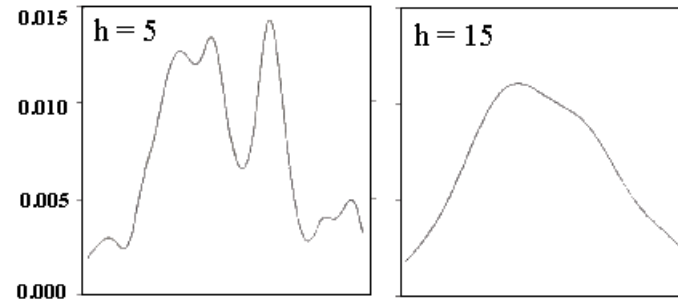
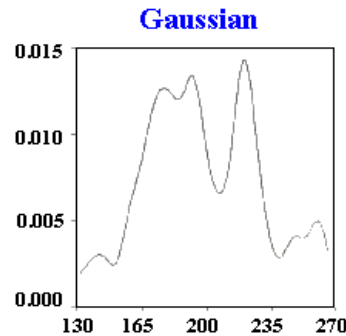
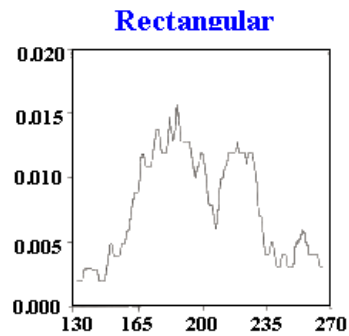
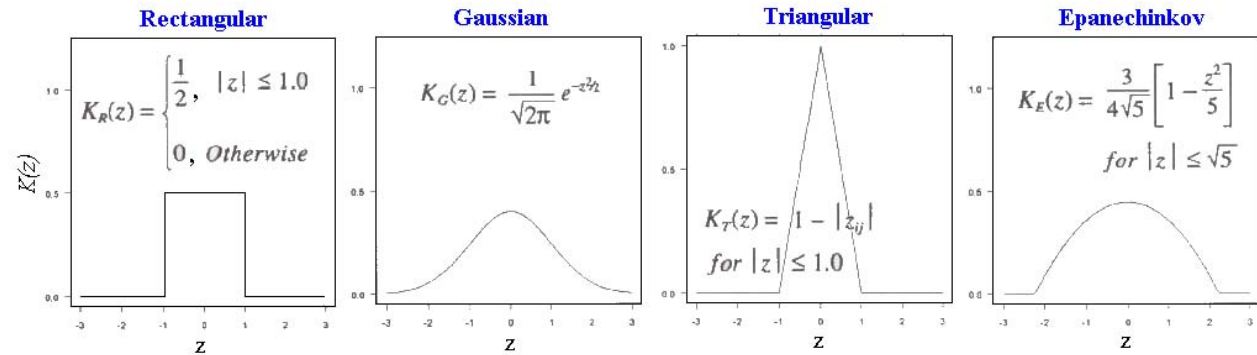


Density Plots (2/3)

- 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)

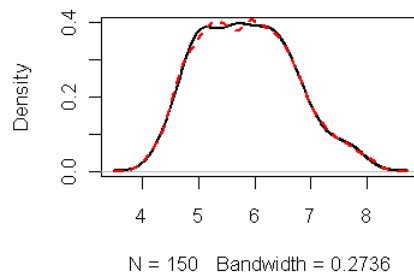
Density Plots (3/3)

```
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, ...)
```

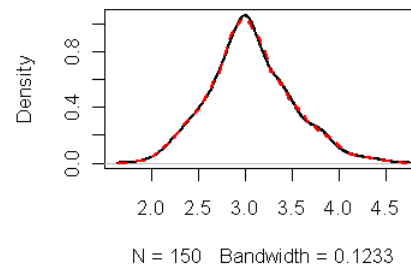
— gaussian
- - - epanechnikov

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

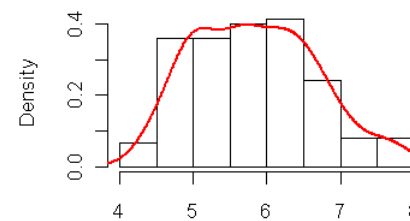
Sepal.Length



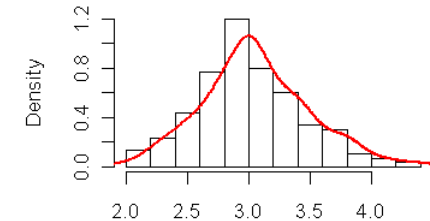
Sepal.Width



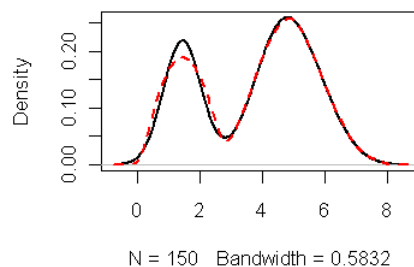
Sepal.Length



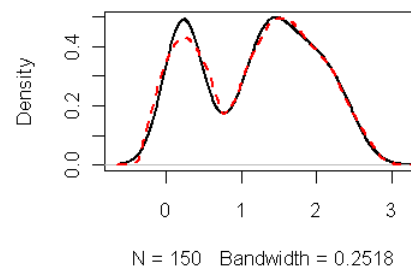
Sepal.Width



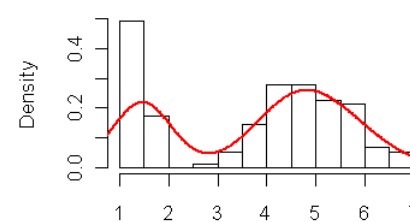
Petal.Length



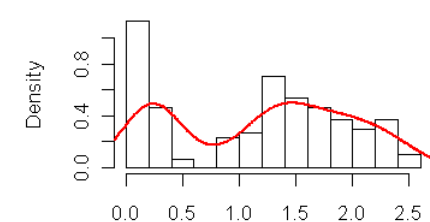
Petal.Width



Petal.Length

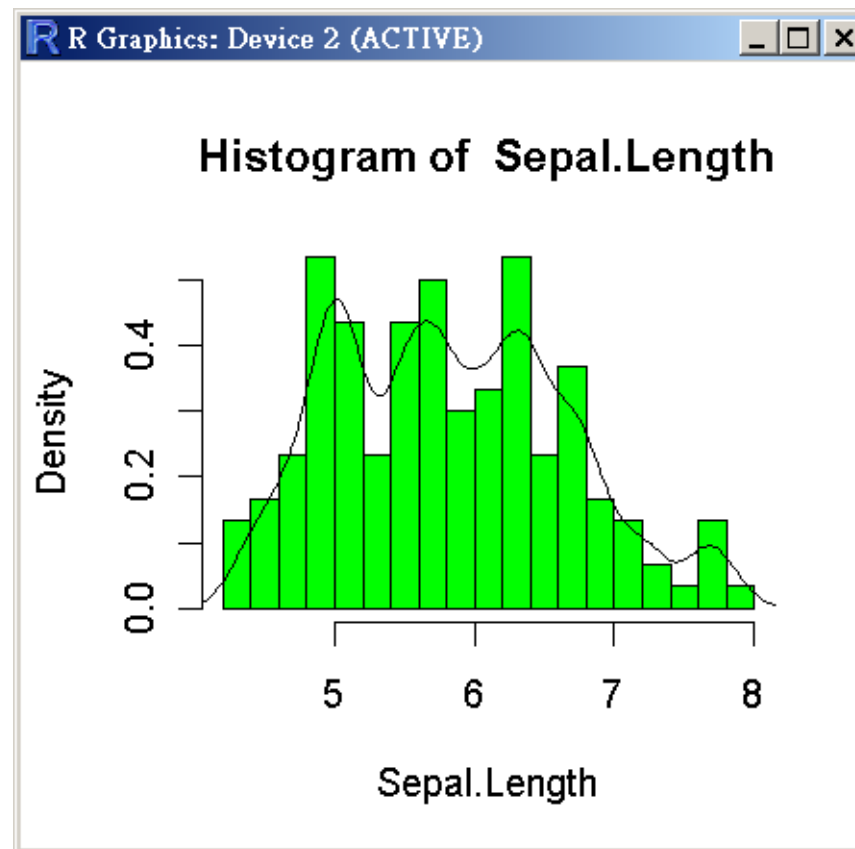


Petal.Width



Density Estimation for Continuous Variables

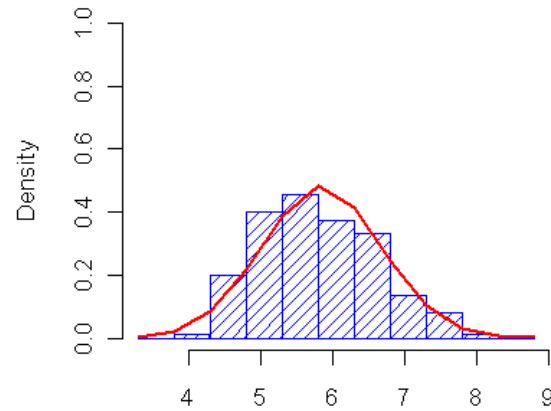
```
> hist(iris[,1], breaks=15, main=title, xlab=lab, col="green", pro=T)  
> lines(density(iris[,1], width=0.6, n=200))
```



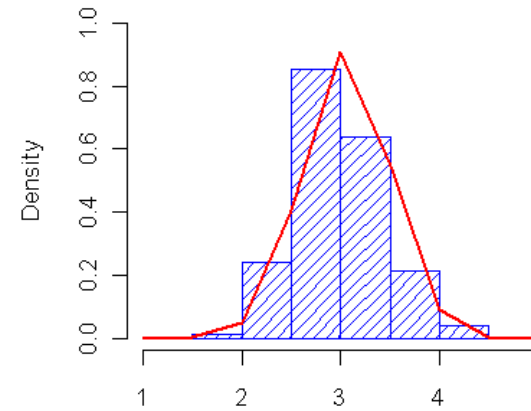


Comparing Data with a Normal Distribution

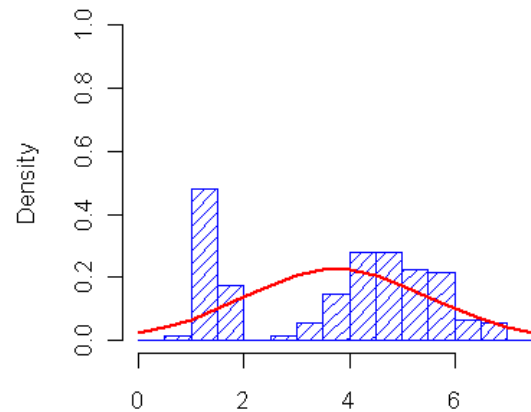
Sepal.Length



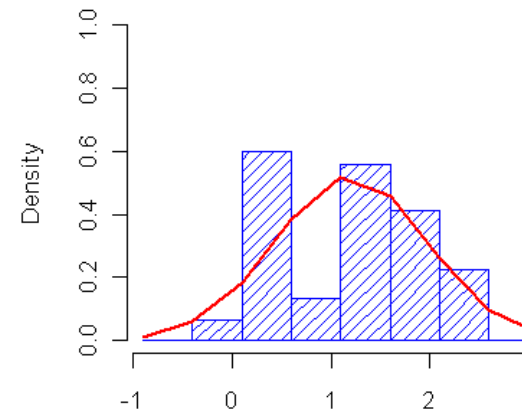
Sepal.Width



Petal.Length



Petal.Width





Andrews' Plot

- Data $\{x_{ij} : i=1, \dots, n; j=1, \dots, p\}$, (vector observations in p -dimensions so x_{ij} is the j^{th} element of the i^{th} observation).

$$f_{x_i}(t) = \frac{1}{\sqrt{2}} x_{i1} + x_{i2} \sin t + x_{i3} \cos t + x_{i4} \sin 2t + x_{i5} \cos 2t + \dots + x_{ip} \frac{\sin\left(\left[\frac{p}{2}\right]t\right)}{\cos\left(\left[\frac{p}{2}\right]t\right)}$$

This maps p -dimensional data $\{x_{ij}\}$ onto 1-dimensional $\{f_{x_i}(t)\}$ for any t .

If we plot $f_{x_i}(t)$ over $-\pi < t < \pi$ we obtain a 1-dimensional representation of the data

Properties:

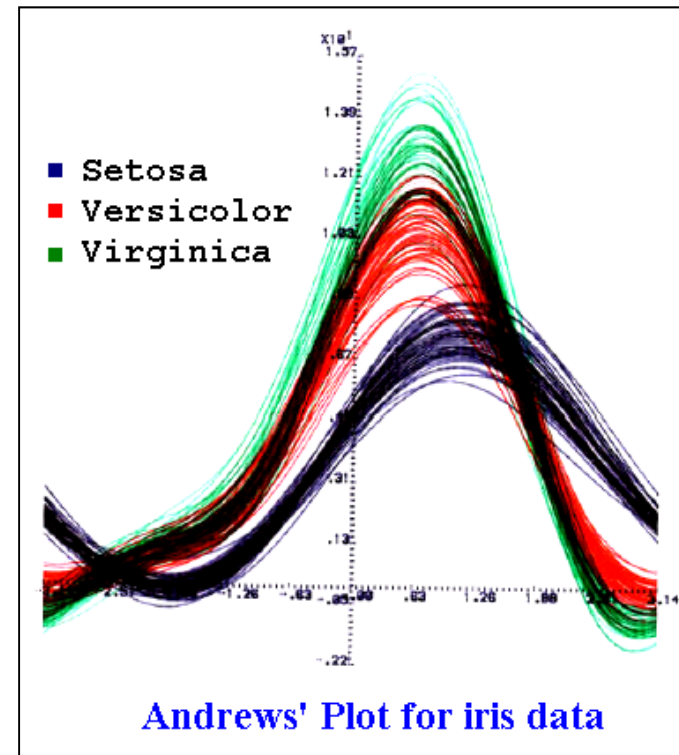
- (i) preserves means,

$$f_{\bar{x}}(t) = \frac{1}{n} \sum_{i=1}^n f_{x_i}(t)$$

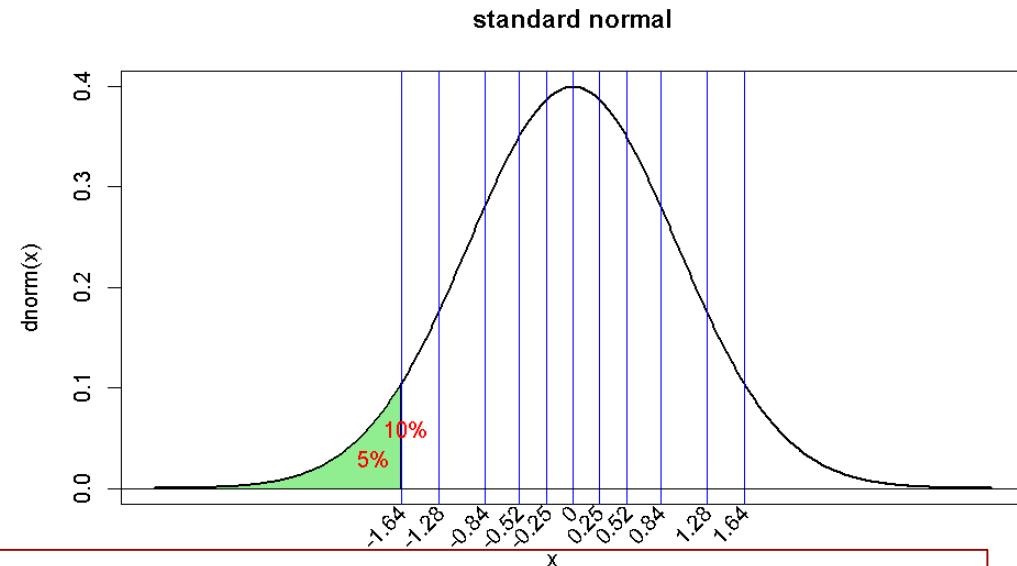
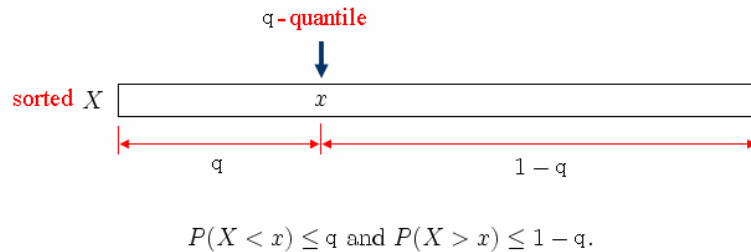
- (ii) preserves distances;

$$\|f_{x_1}(t) - f_{x_2}(t)\|^2 = \int_{-\pi}^{+\pi} (f_{x_1}(t) - f_{x_2}(t))^2 dt = \pi \sum_{j=1}^p (x_{1j} - x_{2j})^2$$

- (iii) yields 1-dimensional views of the data: at $t=t_0$ we obtain the projection of the data onto the vector $f_1(t_0) = (1/\sqrt{2}, \sin t_0, \cos t_0, \sin 2t_0, \dots)'$



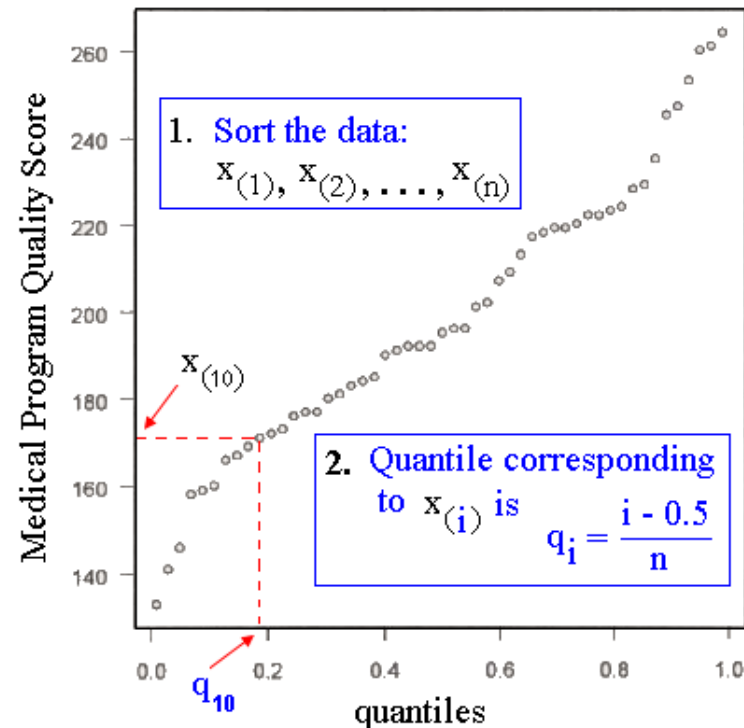
Normal Quantiles



```
> x <- seq(-3, 3, 0.01)
> plot(x, dnorm(x), main="standard normal", type="l", lwd=2, xaxt = "n")
> p <- c(0.05, seq(0.1, 0.9, 0.1), 0.95)
> q <- round(qnorm(p), 2)
> rbind(p, q)
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
p  0.05 0.10 0.20 0.30 0.40 0.5 0.60 0.70 0.80 0.90 0.95
q -1.64 -1.28 -0.84 -0.52 -0.25 0.0 0.25 0.52 0.84 1.28 1.64
> abline(v=q, col="blue")
> abline(h=0, col="black")
> text(q, -0.02, srt = 45, adj = 1, labels = q, xpd = TRUE)
> y <- dnorm(x)
> polygon(c(x[x <= qnorm(0.05)], qnorm(0.05)), c(y[x <= qnorm(0.05)], y[x== -3]),
col="lightgreen")
> text(-1.9, 0.03, "5%", col="red")
> text(-1.6, 0.06, "10%", col="red")
```

Quantile Plots

The empirical quantiles



- 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.

Quantile-Quantile Plots

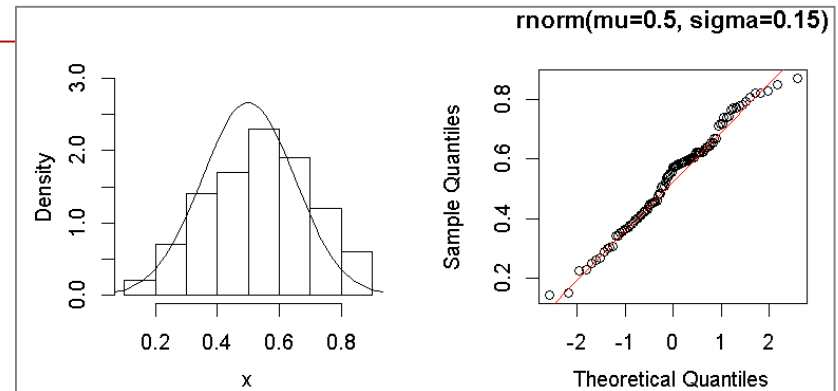
- The quantile-quantile (Q-Q) plot is used to determine if two data sets come from populations with a common density.
- Q-Q plots are sometimes called **probability plots**, especially when data are examined against a theoretical density.
- **qqnorm()**: produces a normal QQ plot of the values in sample
- **qqline()**: adds a line which passes through the first and third quartiles.
 - Use the diagonal line would not make sense because the first axis is scaled in terms of the theoretical quantiles of a $N(0,1)$ distribution.
 - Using the first and third quartiles to set the line gives a robust approach for estimating the parameters of the normal distribution, when compared with using the empirical mean and variance, say.
 - Departures from the line (except in the tails) are indicative of a lack of normality.
- **qqplot()**: qqplot produces a QQ plot of two datasets.

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)

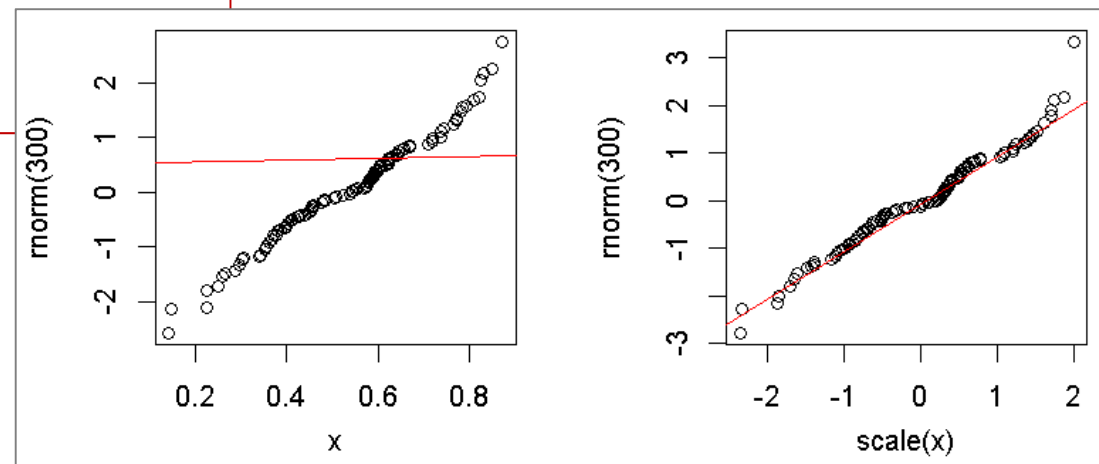
```



```

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

```





實作 Quantile-Quantile Plots

(X_1, X_2, \dots, X_n)

1. 計算樣本平均數及樣本變異數。

$$\bar{X} = \frac{\sum X_i}{n}$$

$$S^2 = \frac{1}{n-1} \sum (X_i - \bar{X})^2$$

2. 將隨機樣本標準化並排序。

$$d_{(1)}, d_{(2)}, \dots, d_{(n)}$$

$$d_{(i)} = \frac{X_i - \bar{X}}{S}$$

3. 查出 n 個標準常態值: (將標準態分配, 區分成 $n+1$ 區塊, 最左及最右區塊的機率分別為 $1/2n$, 中間的 $n-1$ 區塊, 機率分別為 $1/n$)。

$$q_{(1)} = z_{\frac{1}{2n}}, q_{(2)} = z_{\frac{3}{2n}}, \dots, q_{(n)} = z_{\frac{2n-1}{2n}}$$

$$q_{(i)} = z_{\frac{2i-1}{2n}}$$

$$P(Z < q_{(i)}) = \frac{2i-1}{2n}$$

4. 畫散佈圖: x軸: 排序的標準化樣本, y軸: 標準常態值。

$$(d_{(i)}, q_{(i)})$$

5. 加入一條由 $(q_{(i)}, q_{(i)})$ 產生的標準常態直線。
(或加入一條通過第20%及第75% quantiles 的直線)

$$(q_{(i)}, q_{(i)})$$

課堂練習

```
> qqnorm(iris[,1])
> qqline(iris[,1])

> qqnorm(scale(iris[,1]))
> qqline(scale(iris[,1]))

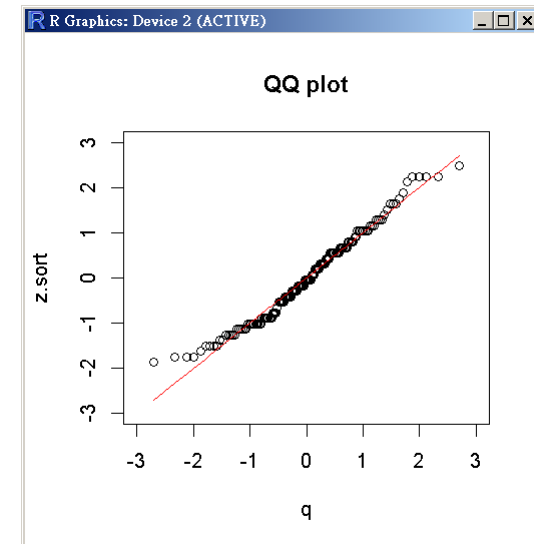
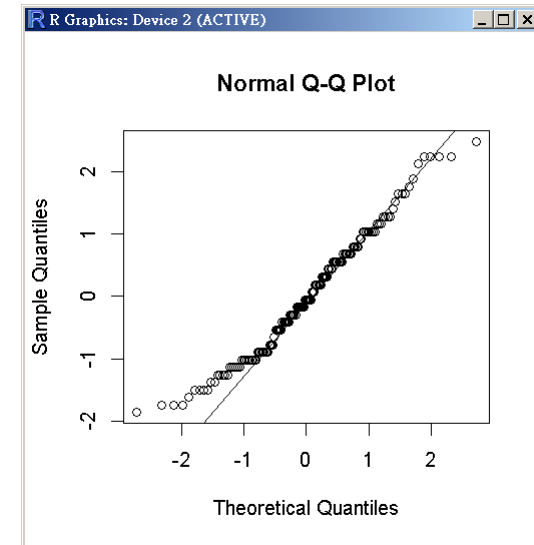
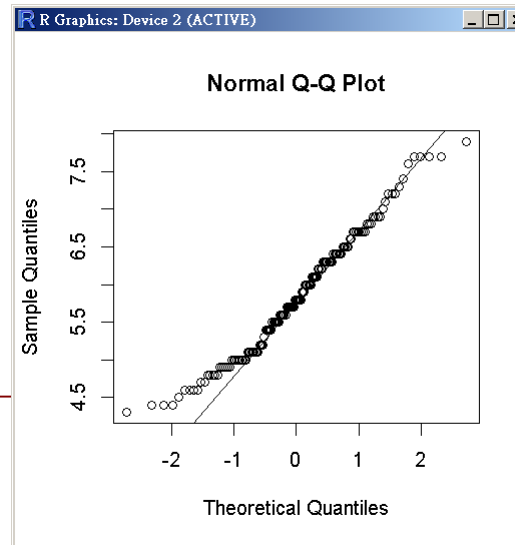
> my.qqplot(iris[,1])
```

```
my.qqplot <- function(x){
  x.mean <- mean(x)
  x.var <- var(x)
  n <- length(x)

  z <- (x-x.mean)/sqrt(x.var)
  z.mean <- mean(z)
  z.var <- var(z)
  z.sort <- sort(z)

  k <- 1:n
  p <- (k-0.5)/n
  q <- qnorm(p)

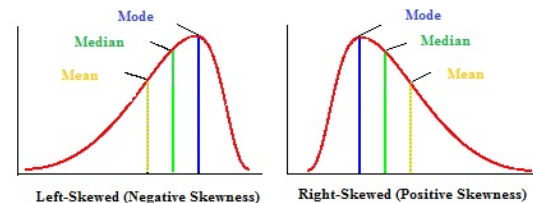
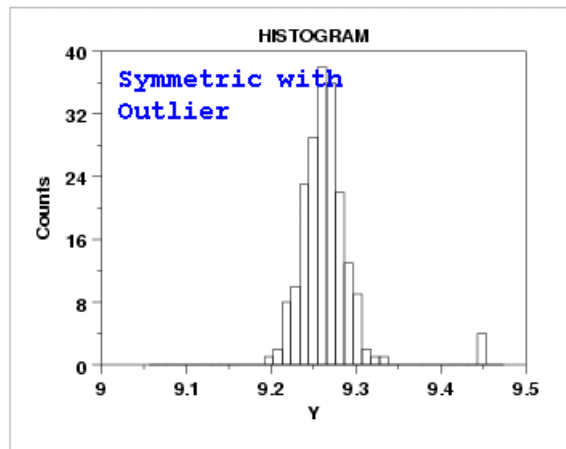
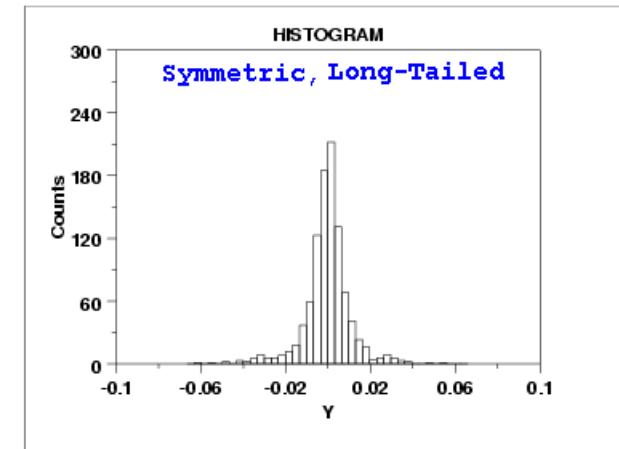
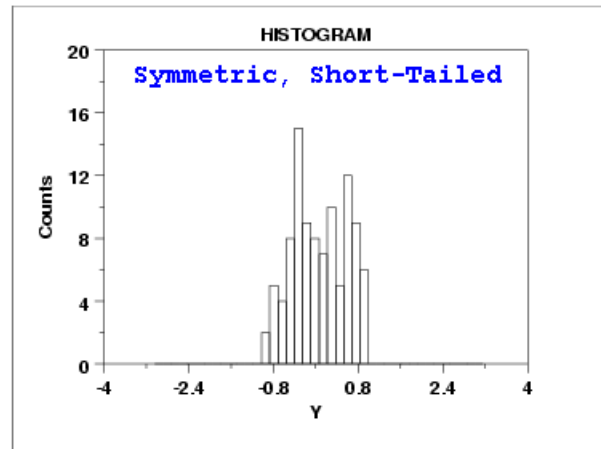
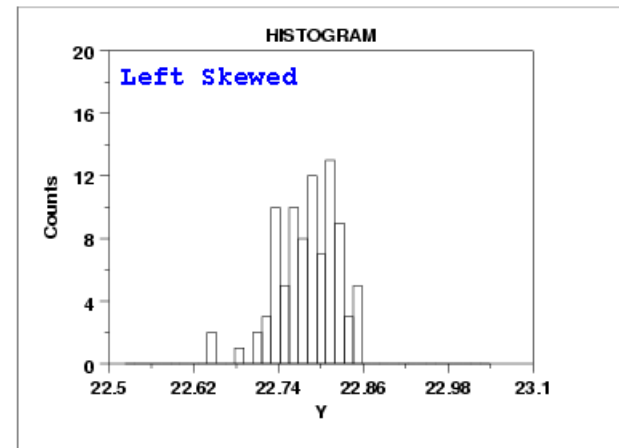
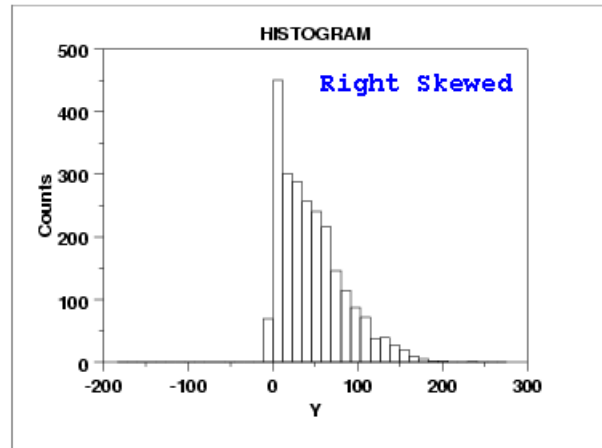
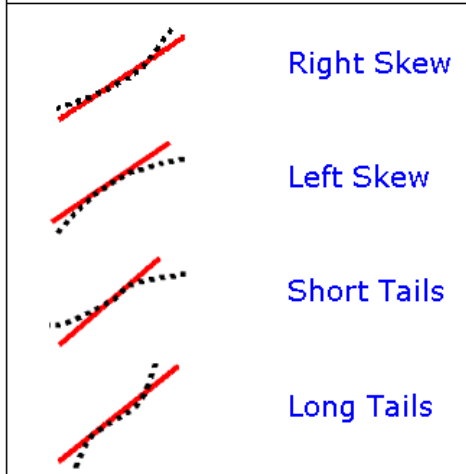
  plot(q, z.sort, xlim=c(-3, 3), ylim=c(-3, 3))
  title("QQ plot")
  lines(q, q, col=2)
}
```





QQplot Diagnostics (1/2)

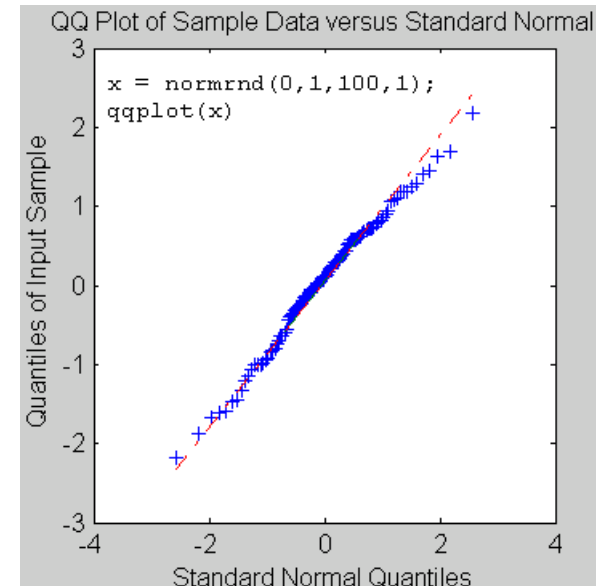
If your Normal distribution Plot looks like: has





QQplot Diagnostics (2/2)

- The QQplot of the **sample quantiles** of X versus **theoretical quantiles** from a (normal) distribution.
 - If the distribution of X is normal, the plotted points fall on or near the line $y = x$.
 - Normal probability plot for graphical normality testing.
- The QQplot of **two samples**. If the samples do come from the same distribution, the plot will be linear.



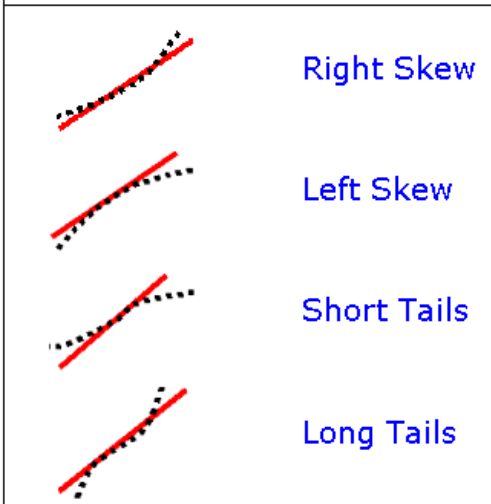
Quantile-Quantile Plot Diagnostics

Description of Point Pattern	Possible Interpretation
All but a few points fall on a line	Outliers in the data
Left end of pattern is below the line; right end of pattern is above the line	Long tails at both ends of the data distribution
Left end of pattern is above the line; right end of pattern is below the line	Short tails at both ends of the data distribution
Curved pattern with slope increasing from left to right	Data distribution is skewed to the right
Curved pattern with slope decreasing from left to right	Data distribution is skewed to the left
Staircase pattern (plateaus and gaps)	Data have been rounded or are discrete

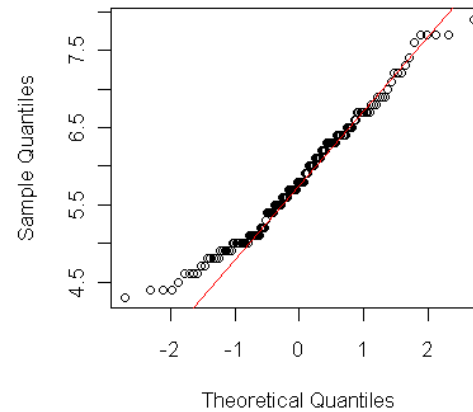


QQplot for Iris Data

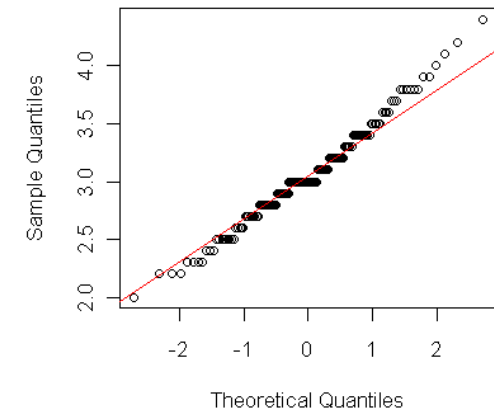
If your Normal distribution Plot looks like:



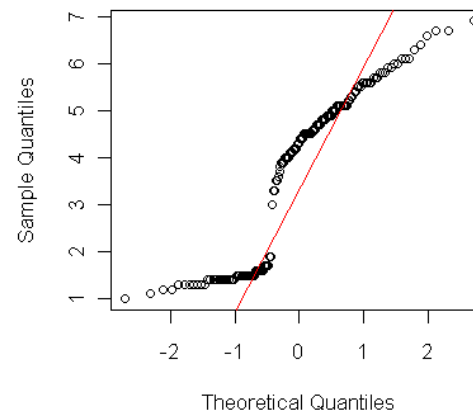
QQplot(Sepal.Length)



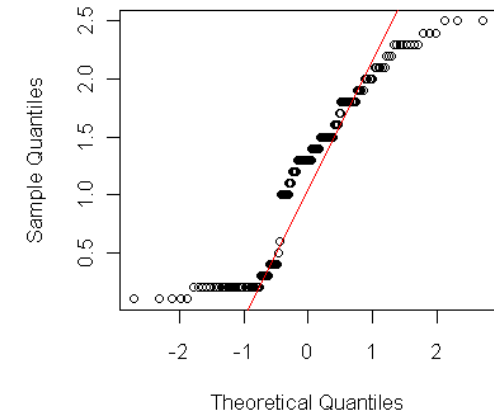
QQplot(Sepal.Width)



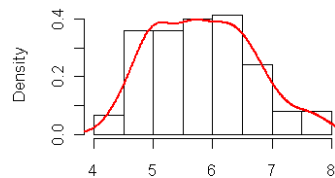
QQplot(Petal.Length)



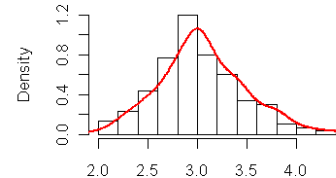
QQplot(Petal.Width)



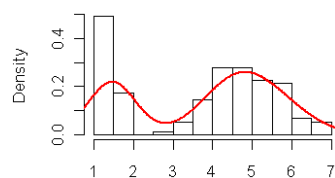
Sepal.Length



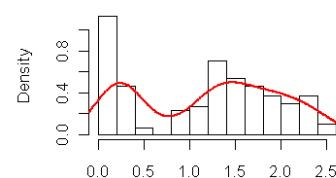
Sepal.Width



Petal.Length



Petal.Width

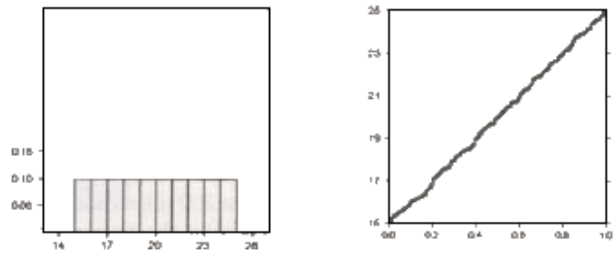




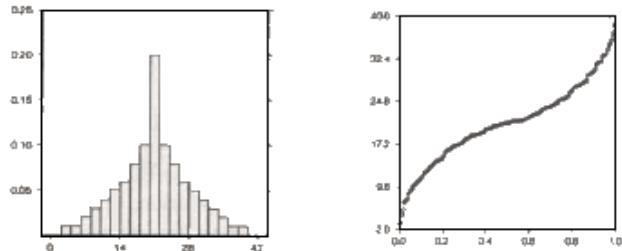
Comparison of Histogram and Quantile Plots

Comparison of histogram and Quantile plots for differently shaped data distribution

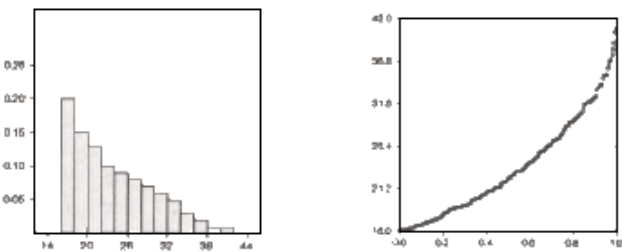
Uniform distribution



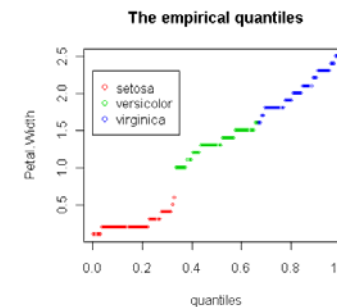
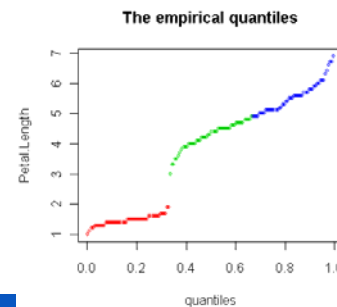
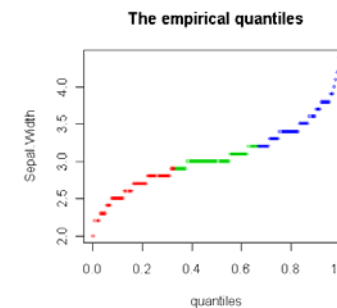
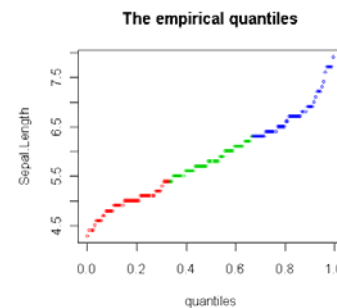
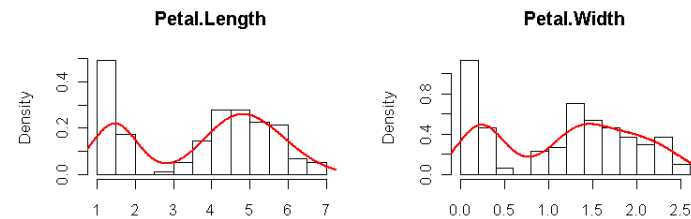
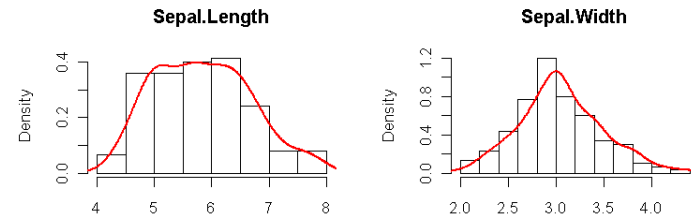
Symmetric, bell-shaped distribution



Positively skewed distribution

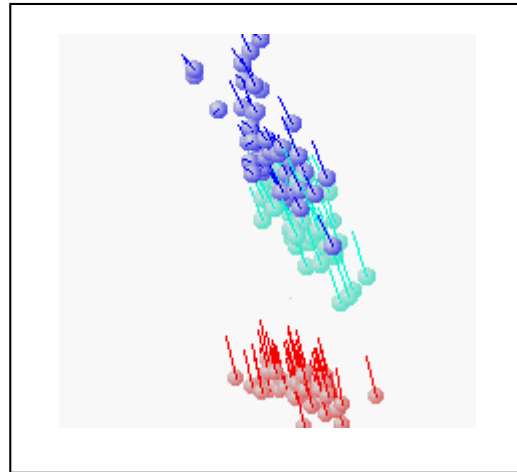


Figures modified from Jacoby (1997)



Glyph Plot

- A "tail" can also be added to each data point, in which the value of the **fourth dimension** is indicated by the angle of the tail.
- The data points represented by complex symbols are called "glyphs".



Glyph graph for Fisher Iris data

x: Petal length.
y: Sepal length.
z: Sepal width.
Angle: Petal width.
Color: three species of iris.

- **Multiple-symbol:** usually one display panel simultaneously shows values of multiple variables that are represented by different shapes, sizes, colors, and locations of symbols (Tukey & Tukey, 1988).



時間序列 (Time Series Plots)

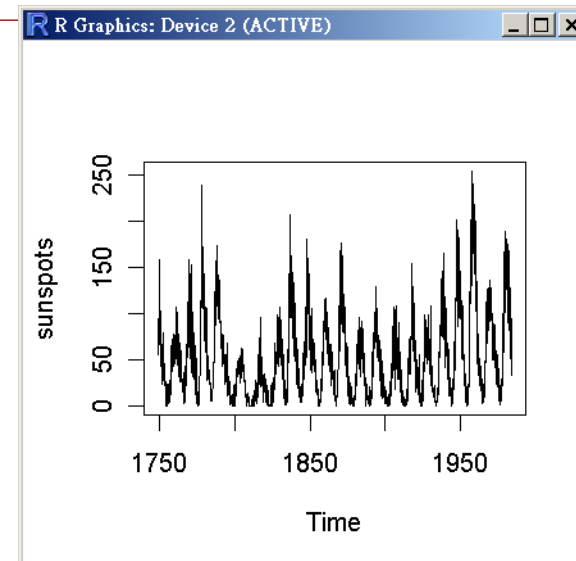
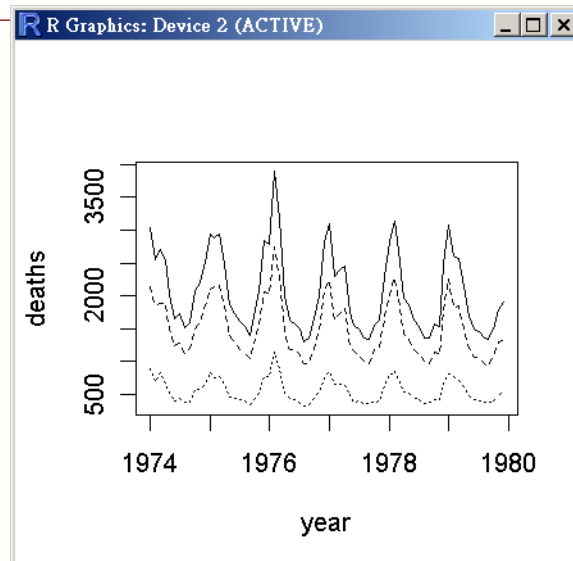
98/216

- `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)
```





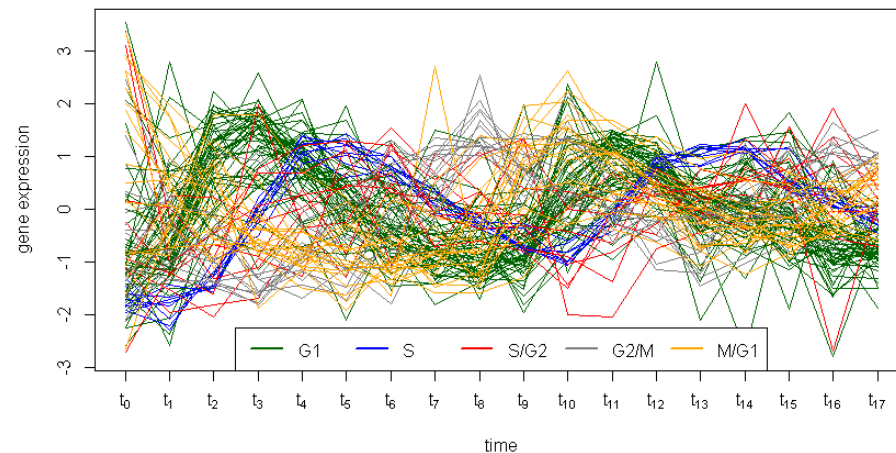
Time Series Plots

```

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)
    
```

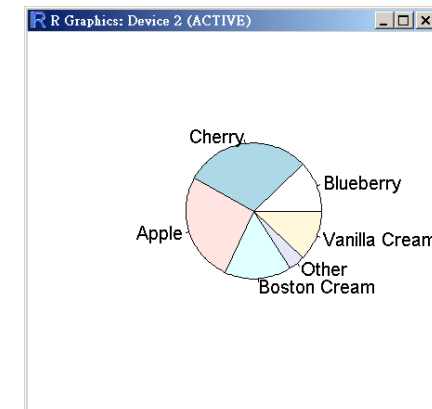
	A	B	C	D	E	F	G
1	UID	Phase	alpha0	alpha7	alpha14	alpha21	alpha28
2	YAR007C_G1	0	-0.48	-0.42	0.87	0.92	0.67
3	YBL035C_G1	0	-0.39	-0.58	1.08	1.21	0.52
4	YBR023C_G1	0	0.87	0.25	-0.17	0.18	-0.13
5	YBR067C_G1	0	1.57	1.03	1.22	0.31	0.16
6	YBR088C_G1	0	-1.15	-0.86	1.21	1.62	1.12
7	YBR278W_G1	0	0.04	-0.12	0.31	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

Time series

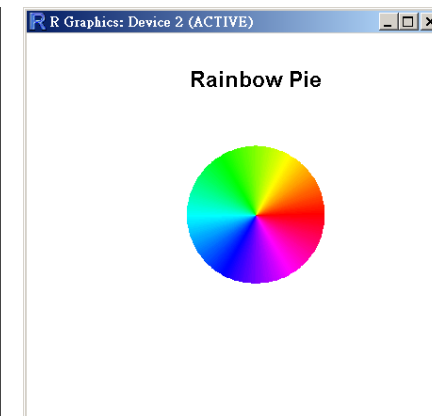
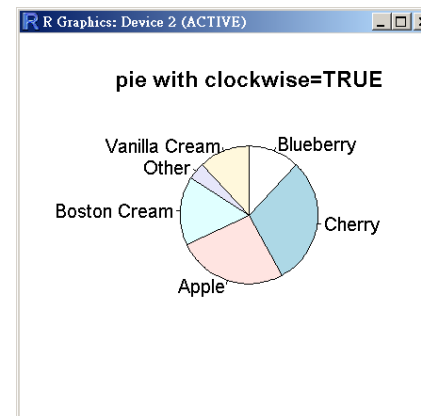
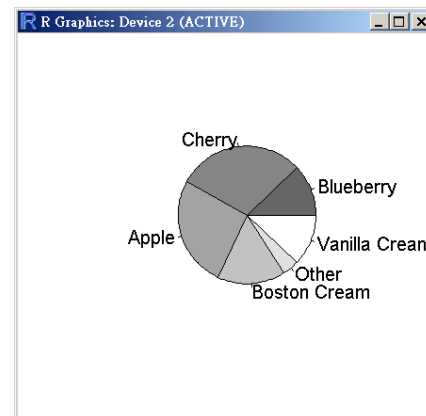
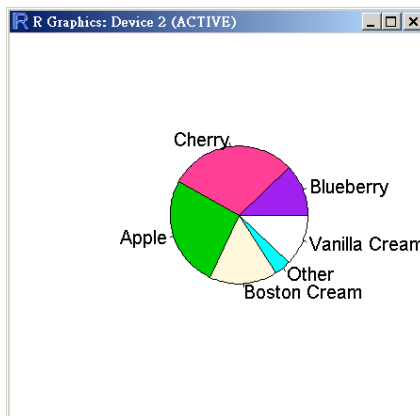


圓餅圖 (Pie Charts)

```
> 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")
```





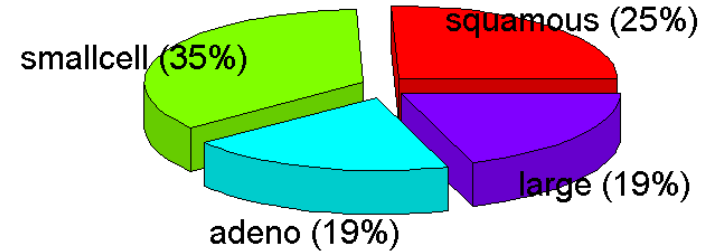
Pie Charts and Their Variants

```

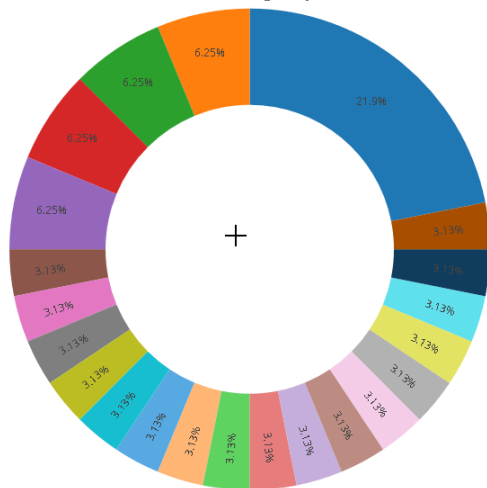
> library(plotrix)
> library(survival)
> head(veteran)
  trt celltype time status karno diagtime age prior
1   1 squamous  72      1    60         7  69     0
2   1 squamous 411      1    70         5  64    10
3   1 squamous 228      1    60         3  38     0
4   1 squamous 126      1    60         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)

```

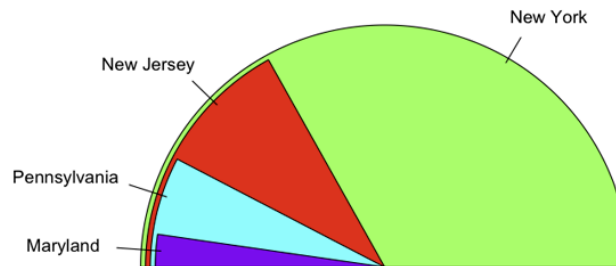


Donut charts using Plotly

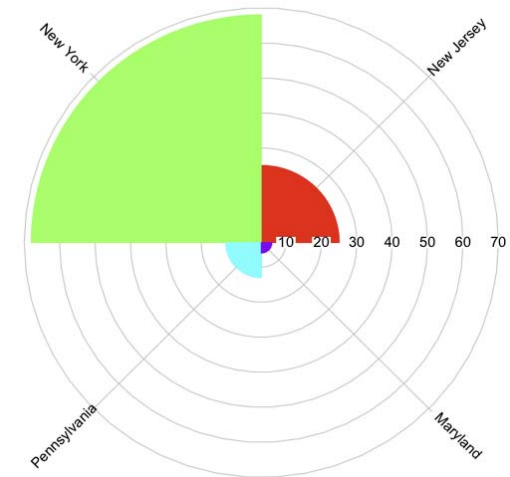


<https://plot.ly/r/pie-charts/>

fan.plot



radial.pie



Plots with Two Variables

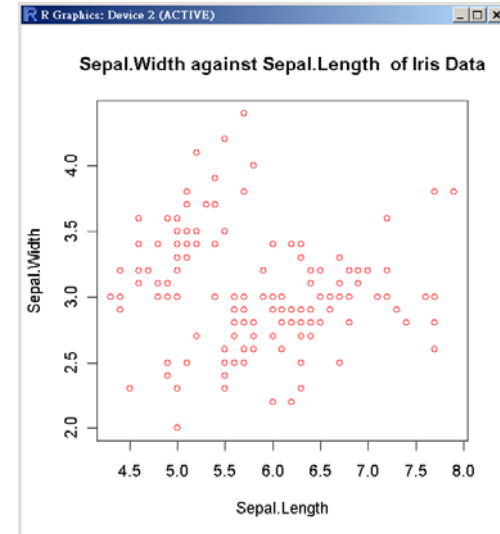
- Response variables on the y-axis, explanatory variable on the x-axis.
 - explanatory variable is continuous => **scatterplot**.
 - explanatory variable is discrete => **box-and-whisker plot, barplot**
- **plot(x, y)**
#scatterplot of y against x
- **plot(factor, y)**
#box-and-whisker plot of y at level of factor
- **boxplot()**
- **barplot(y)**
#heights from a vector of y values



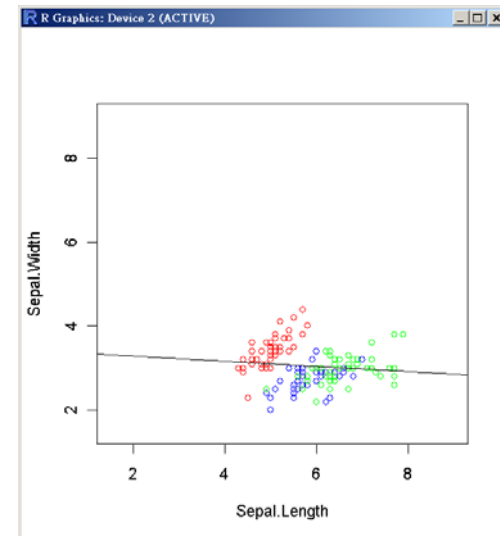
散佈圖 (Scatterplot)

`plot(x, y)` 或 `plot(y~x)`

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



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

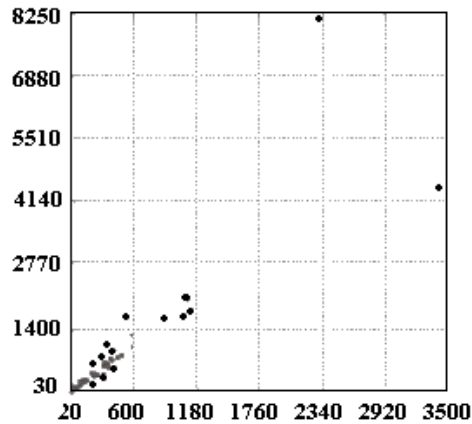




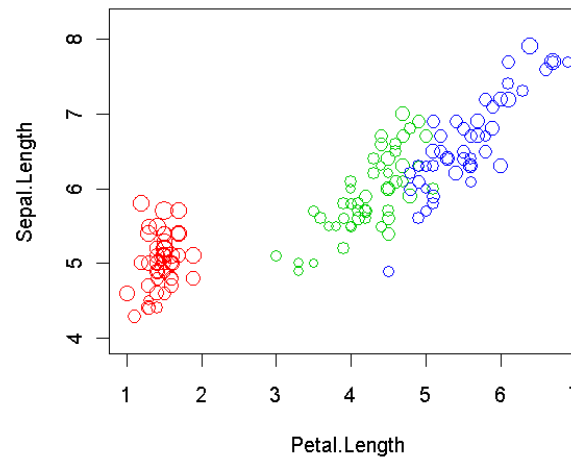
Extensions of Scatterplots

With a smoothing curve

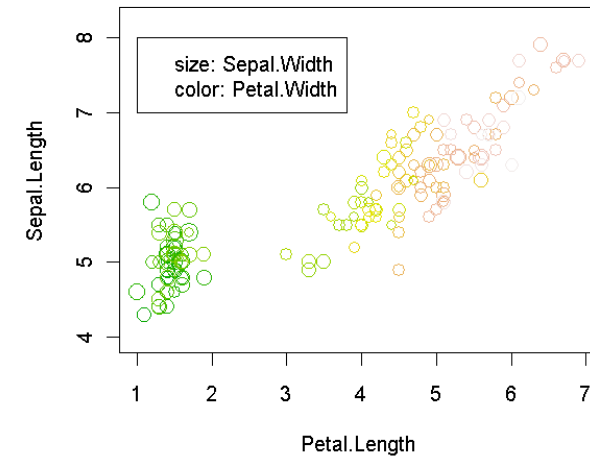
Poorly-Constructed



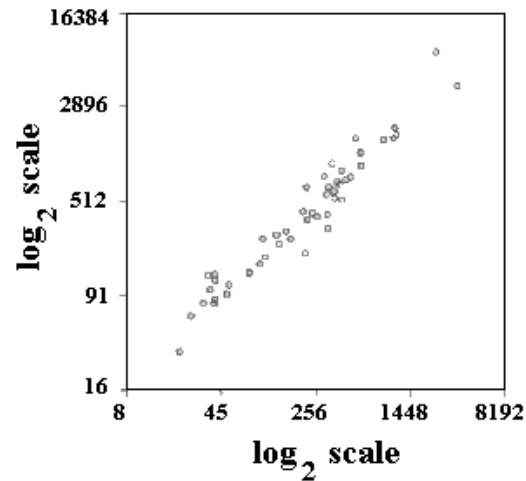
Bubbleplot: Sepal.Width



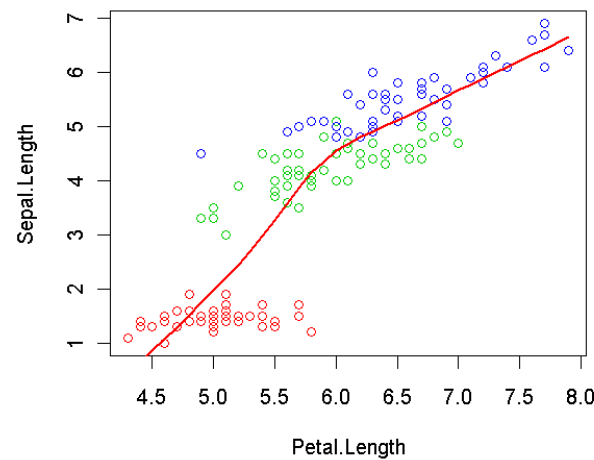
Color Bubbleplot



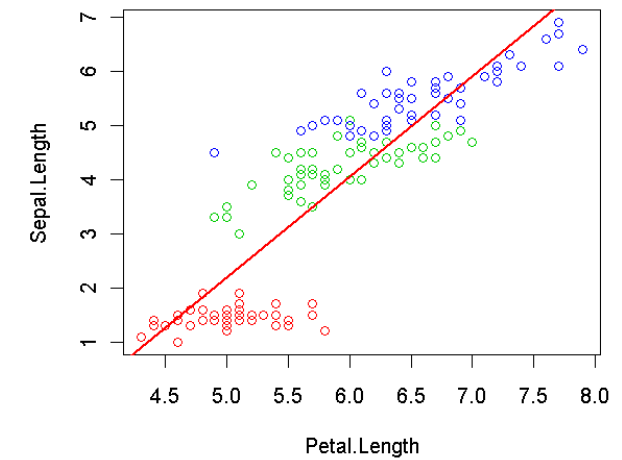
Better



LOWESS



Simple Linear Regression

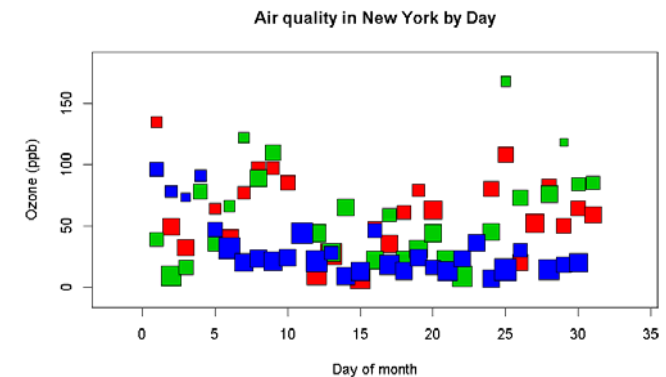
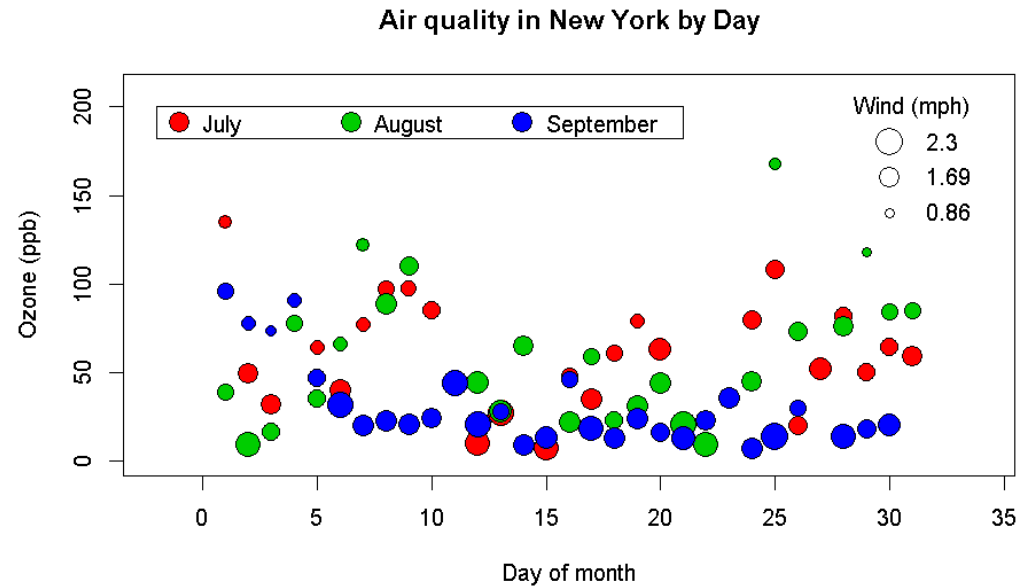




範例: Bubble plots using symbols

105/216

```
> data(airquality)
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41    190  7.4   67     5   1
2    36    118  8.0   72     5   2
3    12    149 12.6   74     5   3
> aq <- airquality[airquality$Month
+             %in% c(7,8,9),]
> aq$Month <- factor(aq$Month,
+                   labels = c("July",
+                               "August",
+                               "September"))
> attach(aq)
> # area of a circle = pi r^2,
> # size Wind by area
> radius <- sqrt(Wind/pi)
> # symbols() sizes the largest bubble to one inch
> symbols(Day, Ozone, circles=radius,
+         inches=0.1, fg="black", bg=as.integer(Month)+1,
+         xlab="Day of month", ylab="Ozone (ppb)",
+         main="Air quality in New York by Day",
+         ylim=c(0, 210))
> legend(-2, 200, legend=c("July", "August", "September"),
+       pch=21, pt.bg=2:4, col="black", pt.cex=2, horiz=T)
> x.loc <- rep(30, 3)
> y.loc <- seq(140, 180, length.out=3)
> s <- summary(radius)[c(1, 4, 6)]
> symbols(x.loc, y.loc, circles=s, inches=0.1, add = T)
> text(x.loc+1, y.loc, labels=round(s,2), pos=4)
> text(31, 200, labels="Wind (mph)")
> detach(aq)
```



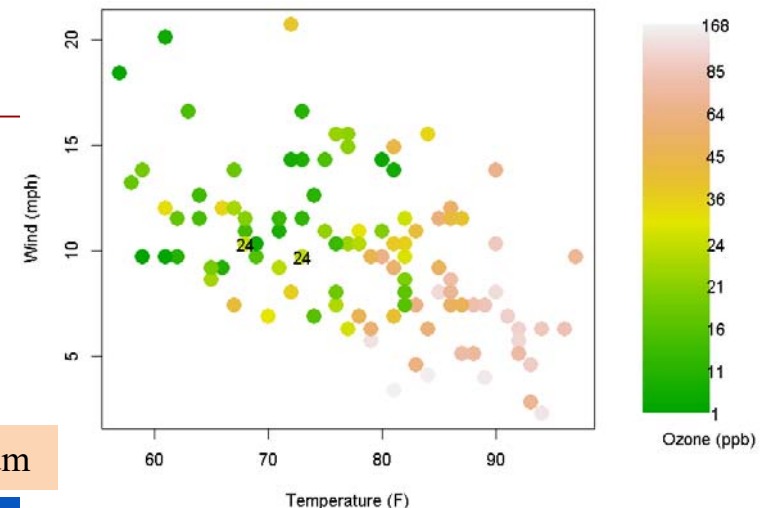
範例: scatterplot with colors (1)

106/216

```
> aq.complete <- airquality[complete.cases(airquality), ]
> np <- dim(aq.complete) # [1] 111 6
> attach(aq.complete)
> layout(matrix(c(1, 2), ncol=2), width=c(4, 1))
> Ozone.col <- get_order_color(Ozone)
> plot(Wind ~ Temp, xlab="Temperature (F)", ylab="Wind (mph)", pch=16, col= Ozone.col, cex=2)
> at <- which(Ozone==24)
> text(Temp[at], Wind[at], "24")
> par("mai")
[1] 1.02 0.82 0.82 0.42
> par.orig <- par(mai=c(1.02, 0, 0.82, 0.1))
> mycolor <- terrain.colors(n)
> plot(0, 0, xlab="", ylab="", xaxt="n", yaxt="n", bty="n", type="n")
> rasterImage(rev(mycolor), -1, -1, 0, 1, interpolate=FALSE)
> id <- rev(floor(seq(1, np[1], length.out=10)))
> id.scale <- (id-min(id))/(max(id)-min(id))*2-1 # scale to (-1, 1)
> text(rep(0.1, 10), id.scale, label= sort(Ozone)[id])
> mtext("Ozone (ppb)", side = 1)
> par(par.orig)
> detach(aq.complete)
```

```
get_order_color <- function(x){
  n <- length(x)
  mycol <- terrain.colors(n)
  sorted.x <- sort(x); order.x <- order(x)
  id <- 1:n
  sorted.id <- id[order.x]
  x.col <- mycol[order(sorted.id)]
  x.col
}
```

use rank of Ozone for a continuous color spectrum

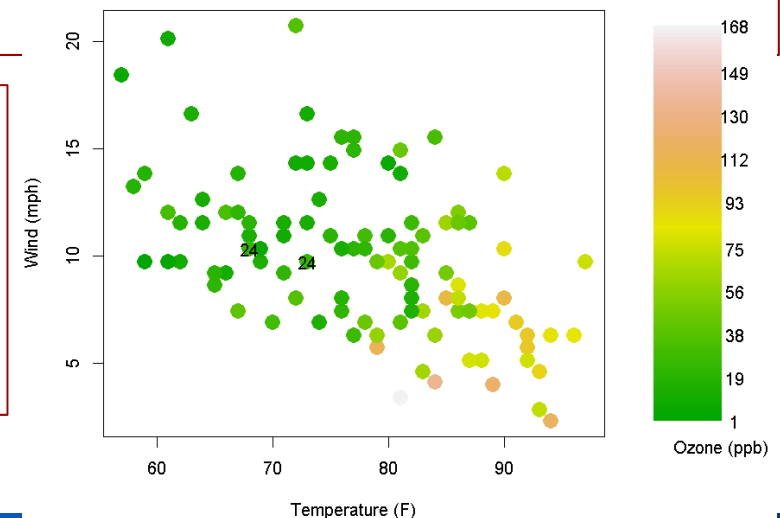


範例: scatterplot with colors (2) 107/216

```
> attach(aq.complete)
> layout(matrix(c(1, 2), ncol=2), width=c(4, 1))
> Ozone.col <- variable_to_color(Ozone)
> plot(Wind ~ Temp, xlab="Temperature (F)", ylab="Wind (mph)", pch=16, col= Ozone.col, cex=2)
> at <- which(Ozone==24)
> text(Temp[at], Wind[at], "24")
>
> par.orig <- par(mai=c(1.02, 0, 0.82, 0.1))
> mycolor <- terrain.colors(n)
> plot(0, 0, xlab="", ylab="", xaxt="n", yaxt="n", bty="n", type="n")
> rasterImage(rev(mycolor), -1, -1, 0, 1, interpolate=FALSE)
> id <- rev(floor(seq(1, np[1], length.out=10)))
> lab <- rev(floor(seq(min(Ozone), max(Ozone), length.out=10)))
> id.scale <- (id-min(id))/(max(id)-min(id))*2-1 # scale to (-1, 1)
> text(rep(0.2, 10), id.scale, label= lab)
> mtext("Ozone (ppb)", side = 1)
> par(par.orig)
> detach(aq.complete)
```

```
variable_to_color <- function(x){
  n <- length(x)
  mycol <- terrain.colors(n)
  # scale x to (1, n)
  scale.x <- floor((x-min(x))/(max(x)-min(x))*(n-1))+1
  x.col <- mycol[scale.x]
  x.col
}
```

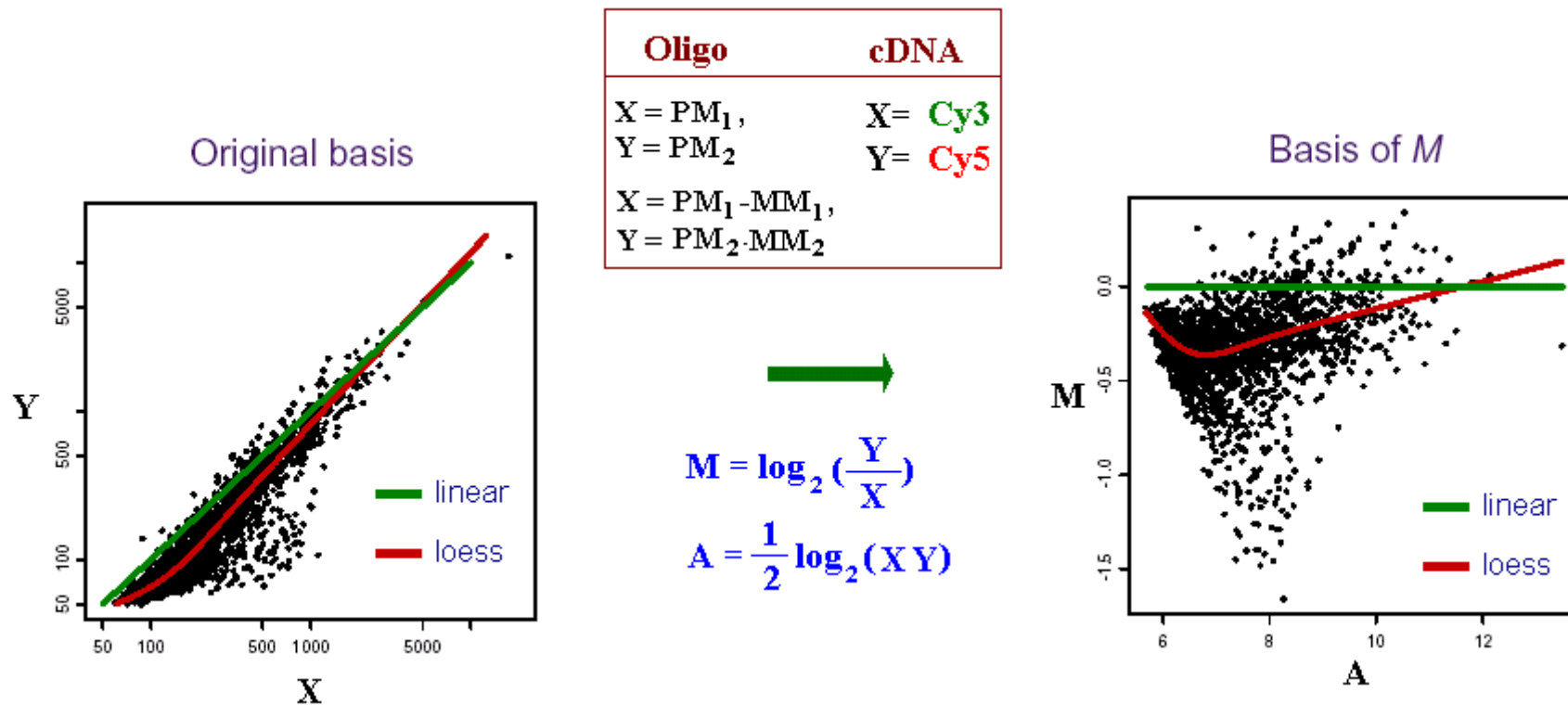
use numeric numbers of Ozone for a continuous color spectrum



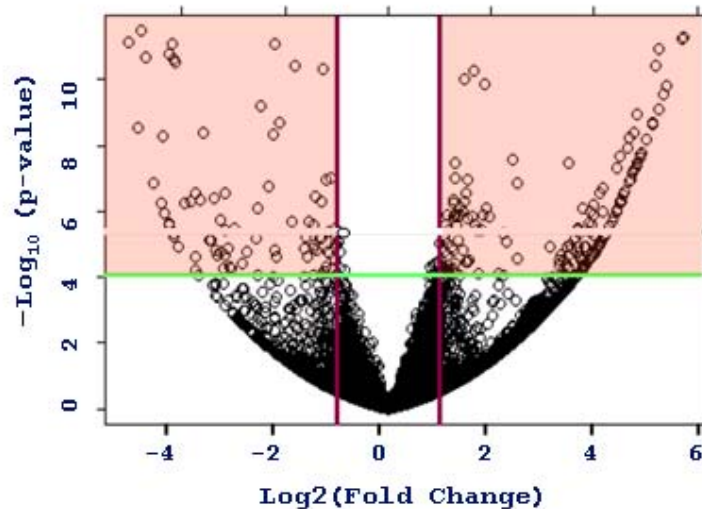


Scatterplot for Gene Expression Data

- **MA plots** can show the intensity-dependent ratio of raw microarray data.



Volcano Plot



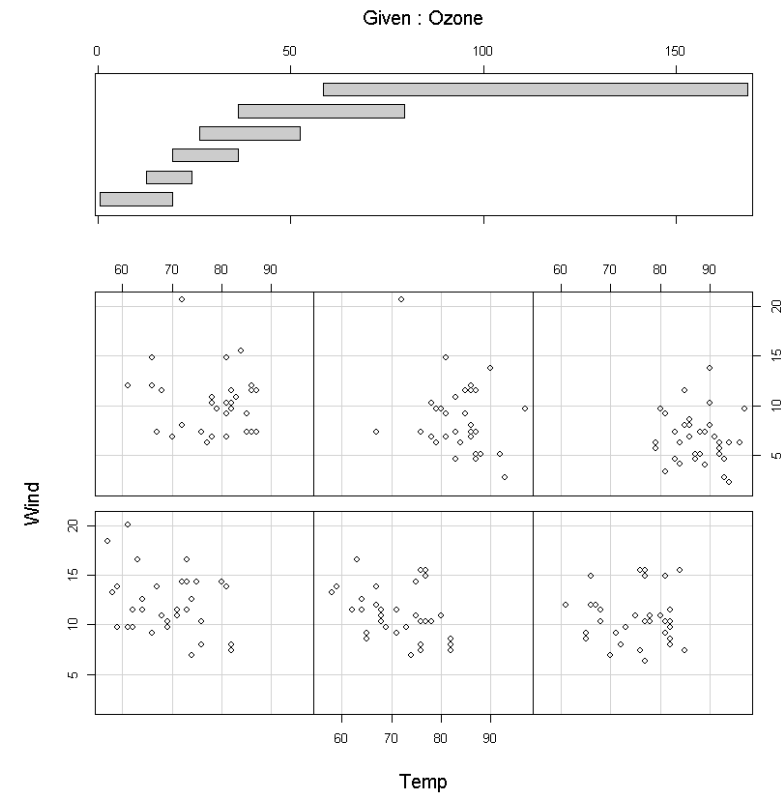
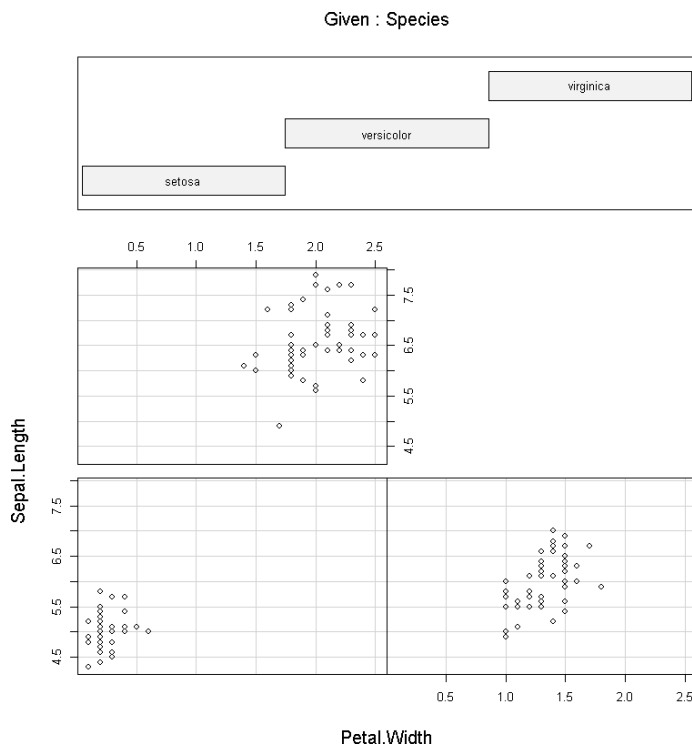
- A volcano plot is a heuristic device that arranges genes along dimensions of **biological** and **statistical significance**.
 - A volcano plot is helpful in identifying significance and magnitude of change in expression of a set of genes between two conditions.
-
- A volcano plot displays the negative log of p-values from a t-test on one axis and the log₂ of change between two conditions on the other axis on the scatterplot view.
 - The researcher can then make judgments about the most promising candidates for follow-up studies, by trading off both these criteria by eye.



coplot {graphics}: Conditioning Plots

110/216

```
> coplot(Sepal.Length ~ Petal.Width | Species, data = iris)
> coplot(Wind ~ Temp | Ozone, data = airquality)
```



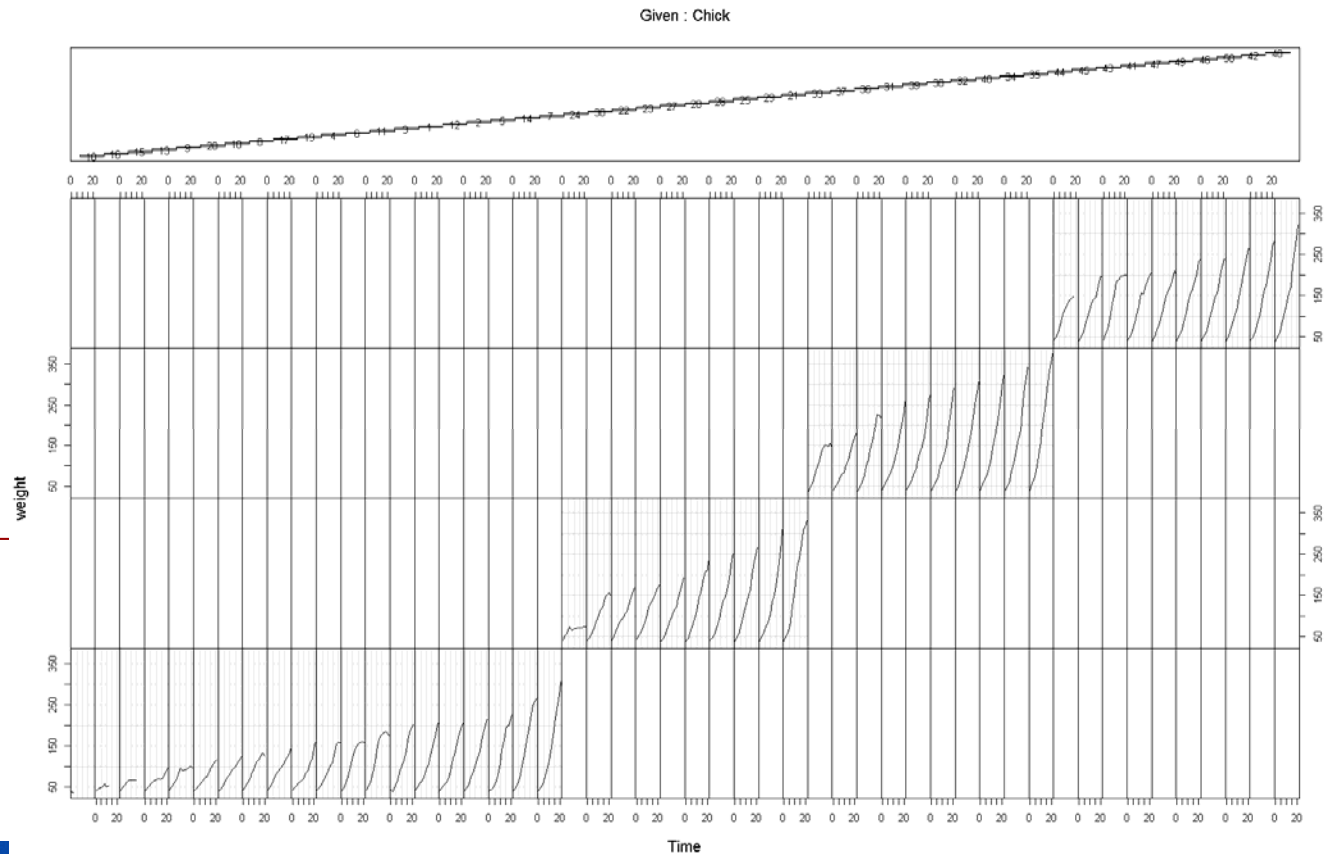


coplot {graphics}: Conditioning Plots

```
> head(ChickWeight, 3)
  weight Time Chick Diet
1     42    0     1    1
2     51    2     1    1
3     59    4     1    1
> coplot(weight ~ Time | Chick * Diet, type = "l", data = ChickWeight)
```

```
> table(ChickWeight$Chick, ChickWeight$Diet)
```

	1	2	3	4
18	2	0	0	0
16	7	0	0	0
15	8	0	0	0
13	12	0	0	0
9	12	0	0	0
20	12	0	0	0
10	12	0	0	0
8	11	0	0	0
17	12	0	0	0
19	12	0	0	0
4	12	0	0	0
...				



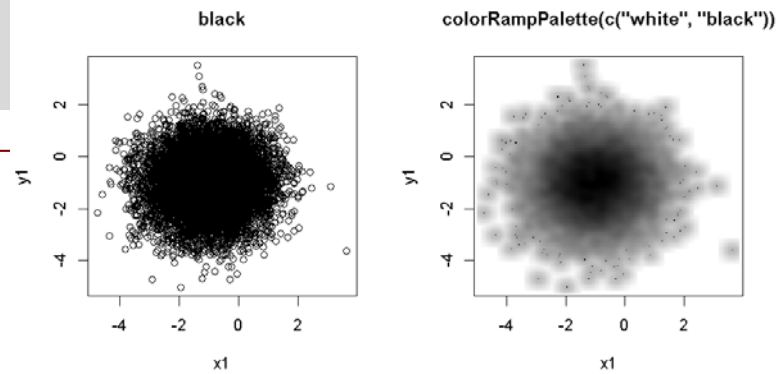


smoothScatter {graphics}: Scatterplots with Smoothed Densities Color Representation

- **smoothScatter** produces a smoothed color density representation of a scatterplot, obtained through a (2D) kernel density estimate.
 - **nbin**: numeric vector of length one (for both directions) or two (for x and y separately) specifying the number of equally spaced grid points for the density estimation; directly used as gridsize in bkde2D().
 - **bandwidth**: numeric vector (length 1 or 2) of smoothing bandwidth(s). If missing, a more or less useful default is used. bandwidth is subsequently passed to function bkde2D.
 - **nrpoints**: number of points to be superimposed on the density image. The first nrpoints points from those areas of lowest regional densities will be plotted. Adding points to the plot allows for the identification of outliers. If all points are to be plotted, choose nrpoints = Inf.

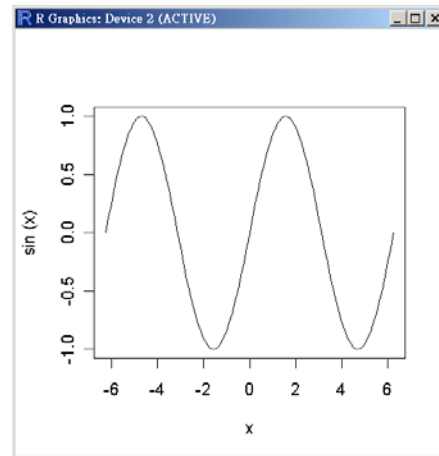
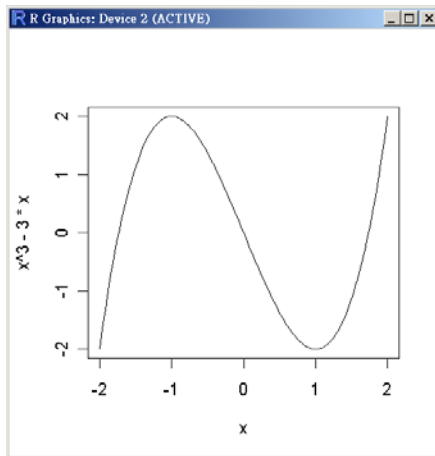
```
smoothScatter(x, y = NULL, nbin = 128, bandwidth,  
              colramp = colorRampPalette(c("white", blues9)),  
              nrpoints = 100, ret.selection = FALSE,  
              pch = ".", cex = 1, col = "black",  
              transformation = function(x) x^.25,  
              postPlotHook = box,  
              xlab = NULL, ylab = NULL, xlim, ylim,  
              xaxs = par("xaxs"), yaxs = par("yaxs"),
```

```
> n <- 1e+04  
> x1 <- rnorm(n, mean = -1, sd = 1)  
> y1 <- rnorm(n, mean = -1, sd = 1)  
> x2 <- rnorm(n, mean = 2, sd = 1)  
> y2 <- rnorm(n, mean = 2, sd = 1)  
> par(mfrow=c(1, 2))  
> plot(x1, y1, main="black")  
> smoothScatter(x1, y1, col="black", colramp=colorRampPalette(c("white", "black")),  
               main='colorRampPalette(c("white", "black"))')
```



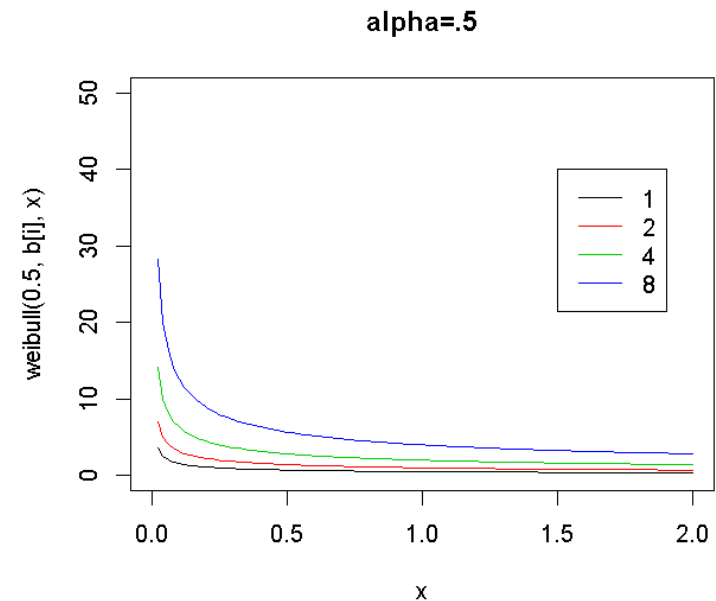
curve {graphics}: 畫2D數學函數圖 113/216

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



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

```
> 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)
```

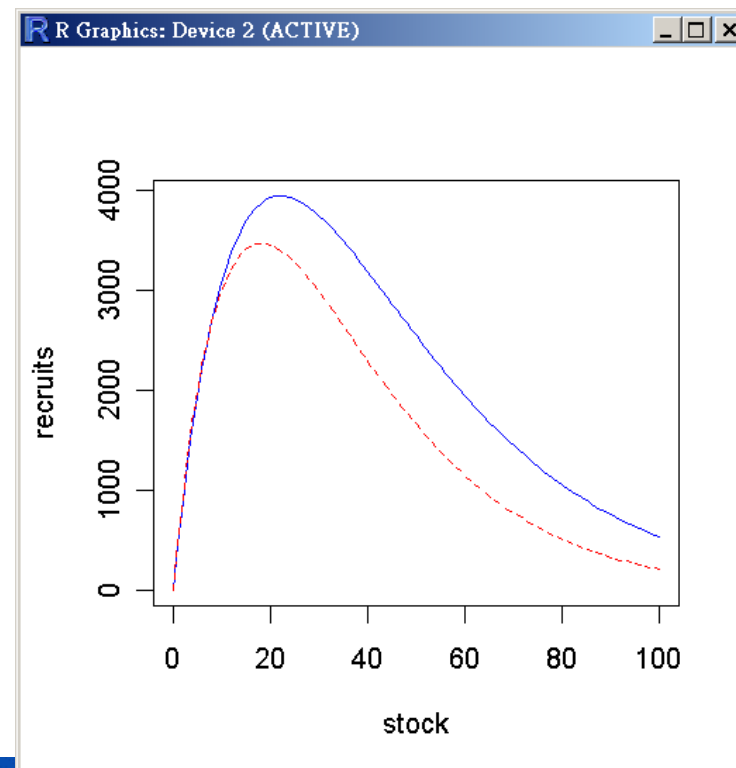




兩曲線在一張圖 (Two Curves)

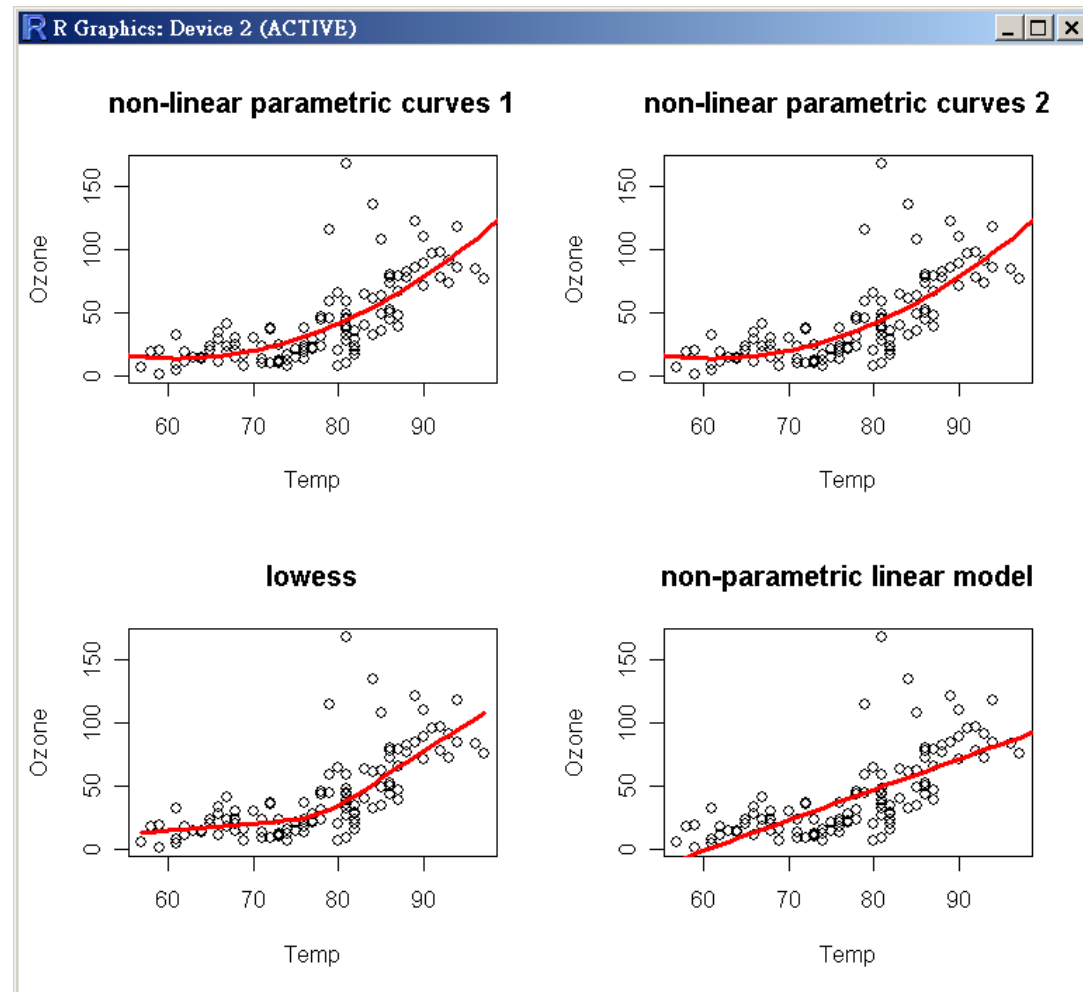
114/216

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



Example Methods

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

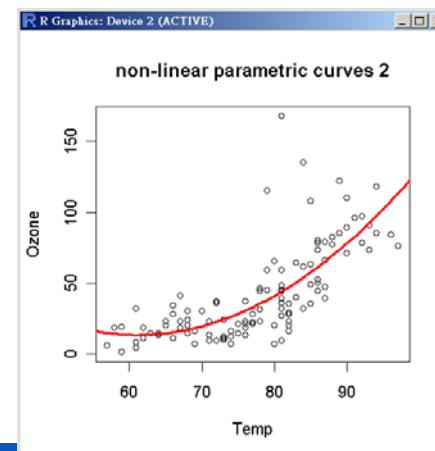
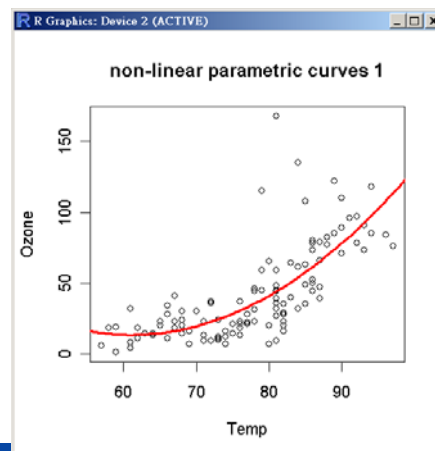


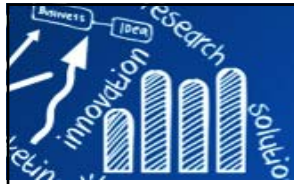
課堂練習

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

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

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

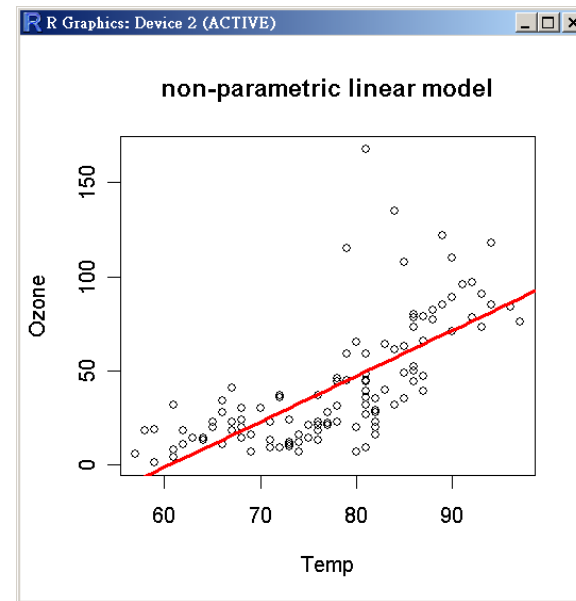
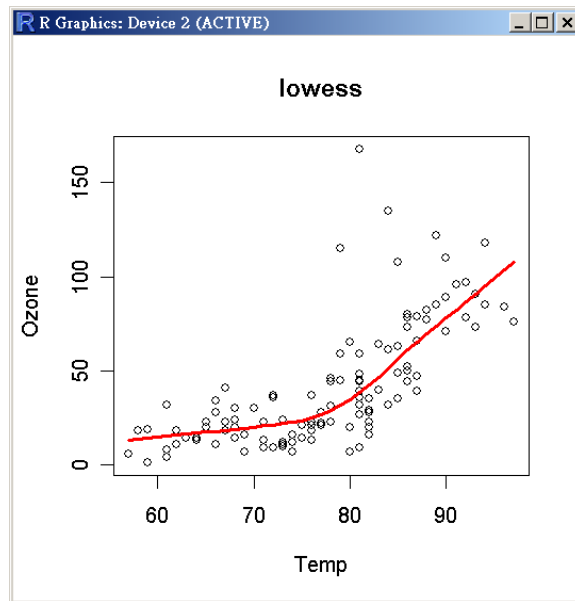




課堂練習

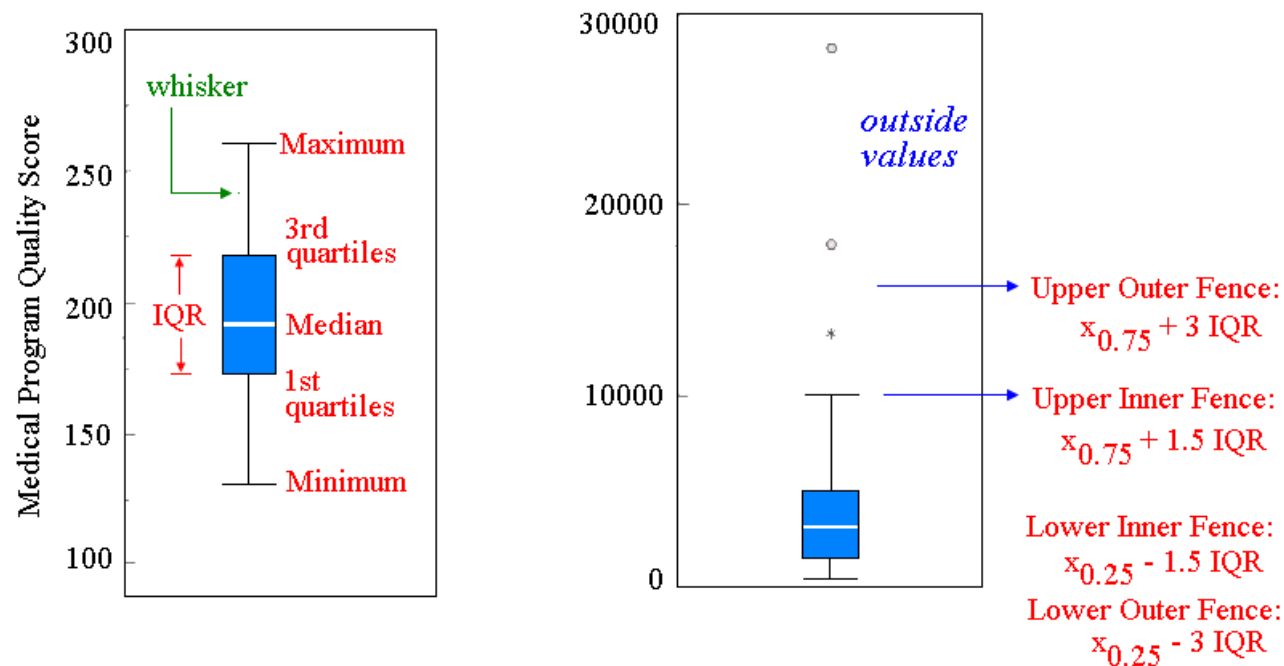
```
> # lowess (a non-parametric curve fitter)
> plot(Ozone~Temp, main="lowess")
> lines(lowess(Ozone~Temp), col="red", lwd=2)

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

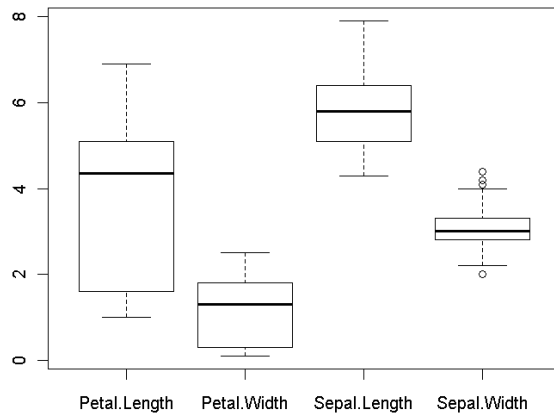


盒形圖 (Boxplot)

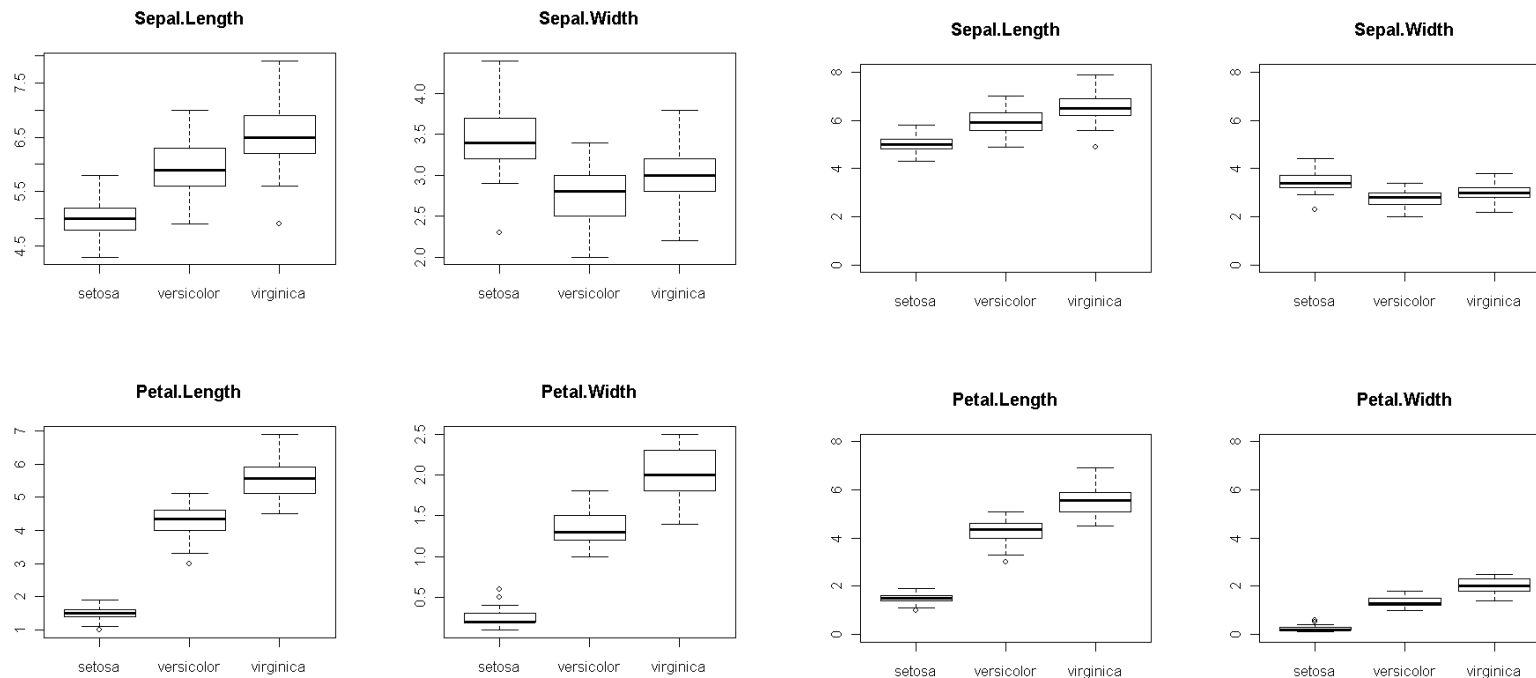
- Plotting with a categorical explanatory variable: boxplot.
- Categorical variables are factors with two or more levels.
- Boxplot
 - The horizontal line shows the median.
 - The bottom and top of the box show the 25th and 75th percentiles.
 - The vertical dashed lines are called the "whiskers".
 -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



Boxplot for Iris Data

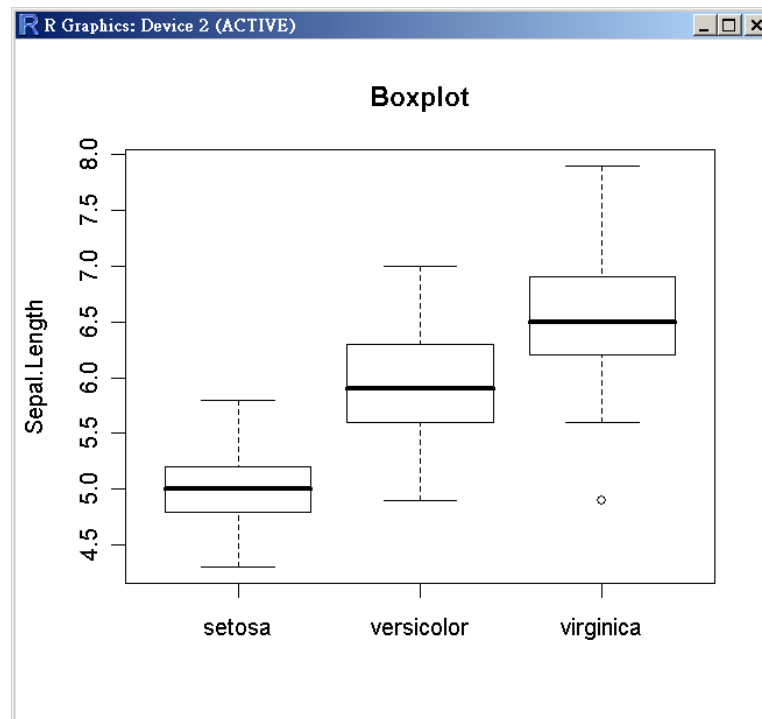


- Is a factor significant?
- Does the location differ between subgroups?
- Does the variation differ between subgroups?
- Are there any outliers?



使用plot畫盒形圖

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



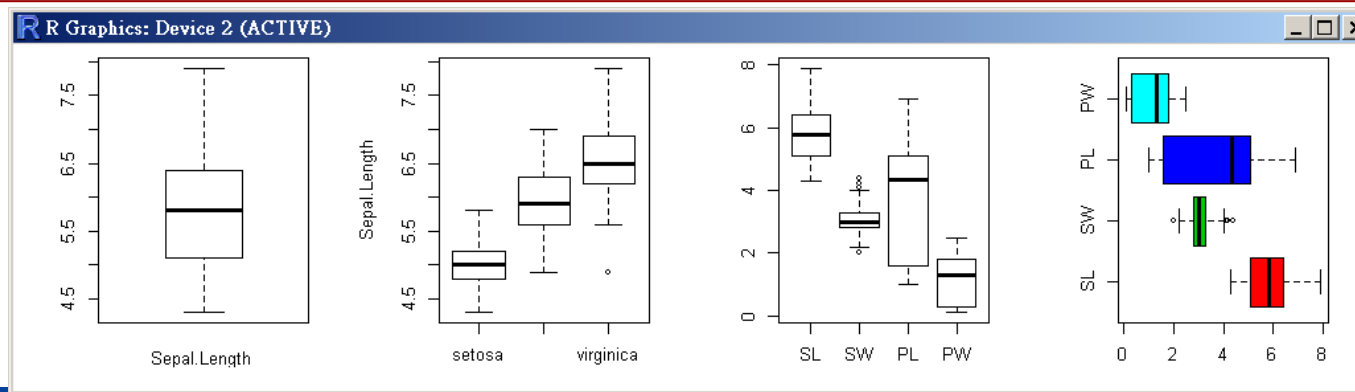
The box plot can provide answers to the following questions:

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

boxplot(x, ...)

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

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



boxplot (formula)

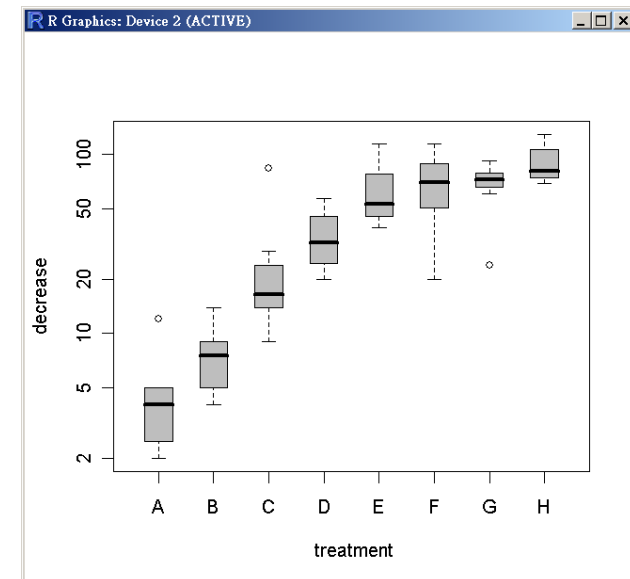
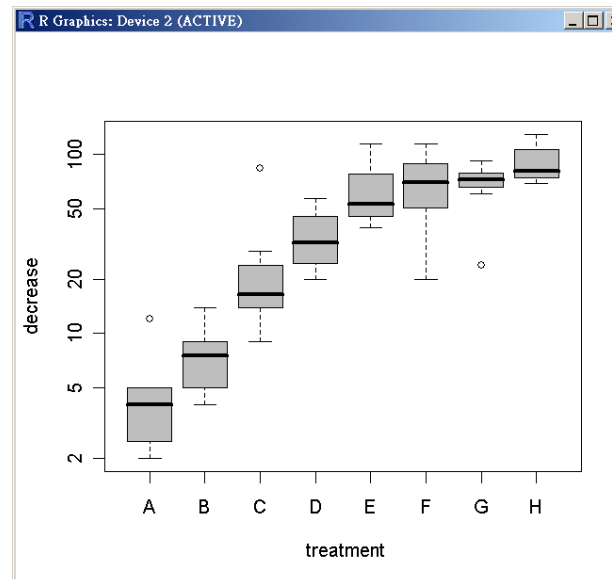
```

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

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

```

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

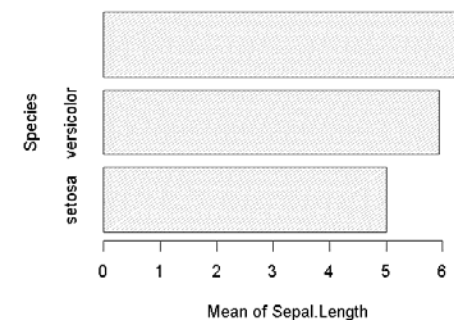
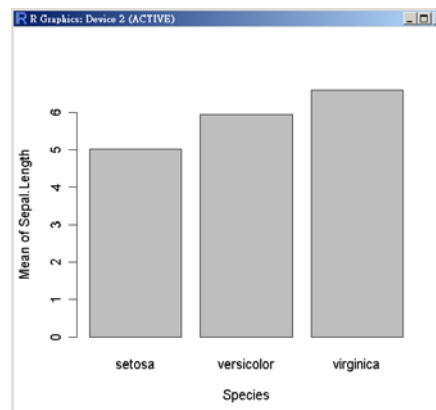


長條圖 (barplot)

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

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

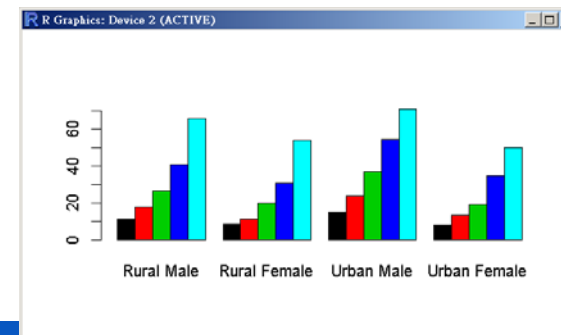
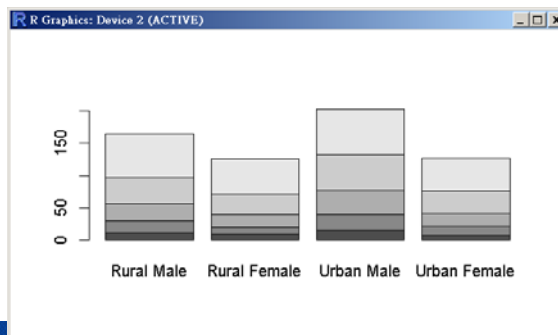
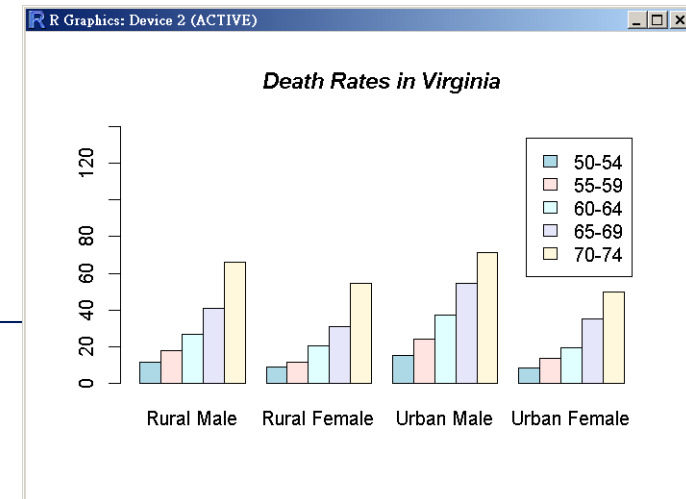
Death Rates in Virginia (1940): The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50–54, 55–59, 60–64, 65–69, 70–74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

```
> 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)

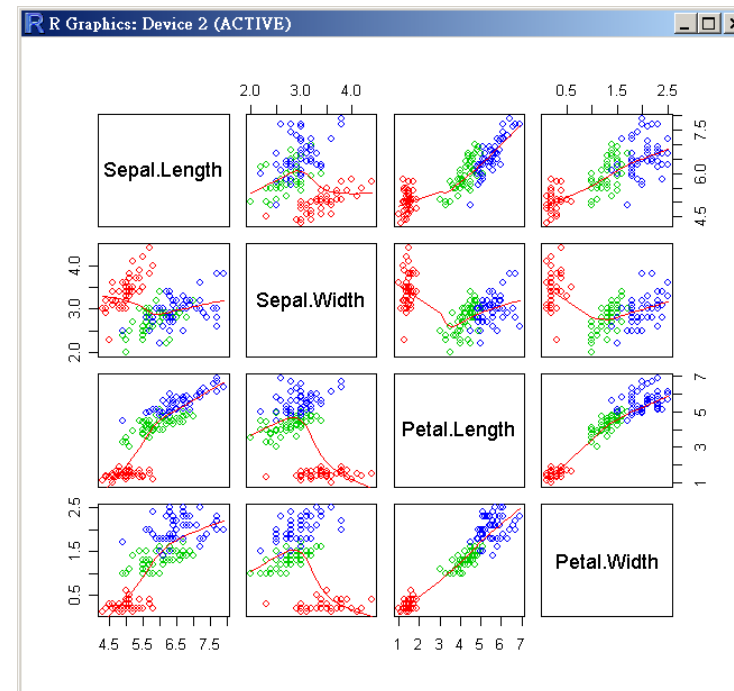
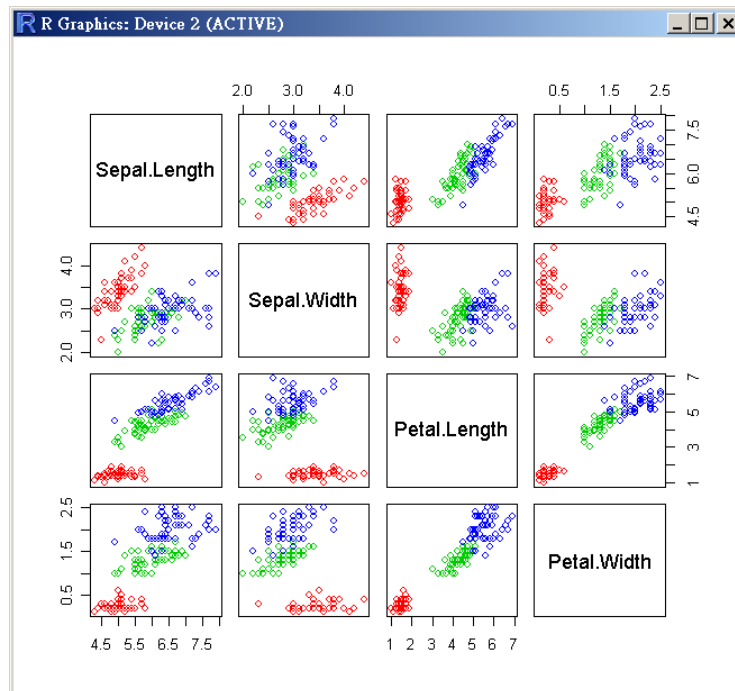
barplot(VADeaths, beside = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan",
               "lavender", "cornsilk"),
        legend = rownames(VADeaths), ylim = c(0, 140))
title(main = "Death Rates in Virginia", font.main = 4)
```



散佈圖矩陣 (Scatterplot Matrices)

125/216

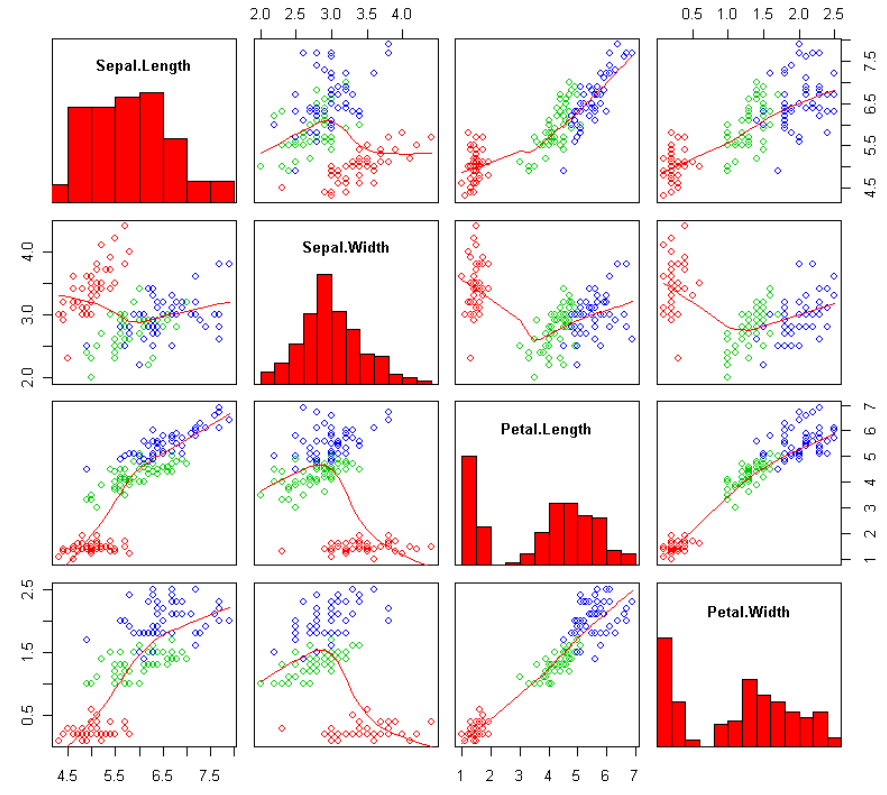
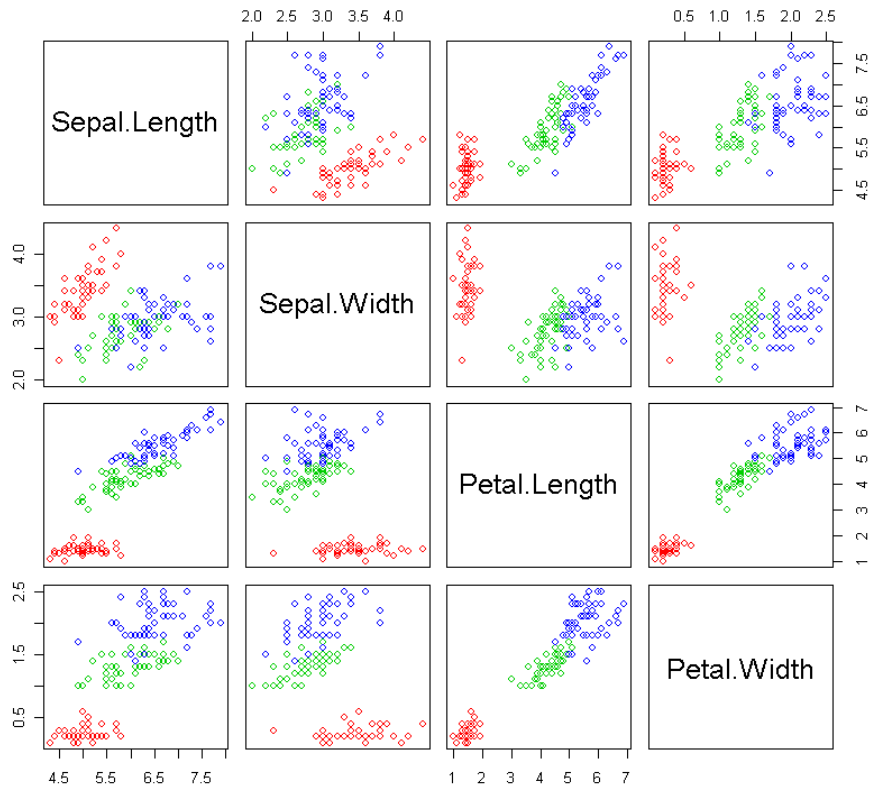
```
> pairs(iris[,1:4], col=as.integer(iris[,5])+1)
> pairs(iris[,1:4], col=as.integer(iris[,5])+1,
       panel=panel.smooth)
```



See also: conditioning plot (`coplot`)



課堂練習





莖葉圖 (Stem and Leaf Plot)

The stem and leaf plot provides the same information as a histogram:

- The plot can be constructed without computer.
- The values of each individual data point can be recovered from the plot.
- The data is arranged compactly since the stem is not repeated in multiple data points.

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

The decimal point is 1 digit

```
42 | 0
44 | 0000
46 | 000000
48 | 00000000000
50 | 00000000000000000000
52 | 00000
54 | 000000000000000
56 | 0000000000000000
58 | 00000000000
60 | 000000000000000
62 | 000000000000000
64 | 000000000000000
66 | 00000000000
68 | 0000000
70 | 00
72 | 0000
74 | 0
76 | 00000
78 | 0
```

Sepal.Length

```
4 | 3444
4 | 5666677888889999999
5 | 00000000001111111122223444444
5 | 5555556666667777777888888999
6 | 0000001111112222333333334444444
6 | 555556677777778889999
7 | 0122234
7 | 677779
```

Sepal.Width

```
4 | 3444
4 | 5666677888889999999
5 | 00000000001111111122223444444
5 | 5555556666667777777888888999
6 | 0000001111112222333333334444444
6 | 555556677777778889999
7 | 0122234
7 | 677779
```

Petal.Length

```
4 | 3444
4 | 5666677888889999999
5 | 00000000001111111122223444444
5 | 5555556666667777777888888999
6 | 0000001111112222333333334444444
6 | 555556677777778889999
7 | 0122234
7 | 677779
```

Petal.Width

```
4 | 3444
4 | 5666677888889999999
5 | 00000000001111111122223444444
5 | 5555556666667777777888888999
6 | 0000001111112222333333334444444
6 | 555556677777778889999
7 | 0122234
7 | 677779
```



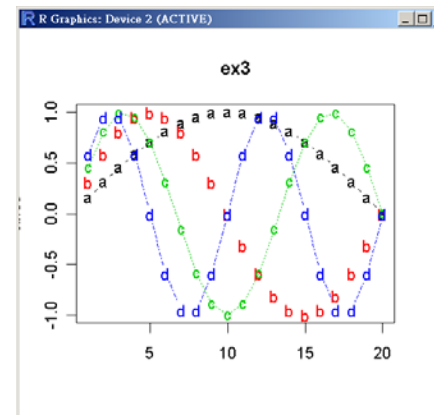
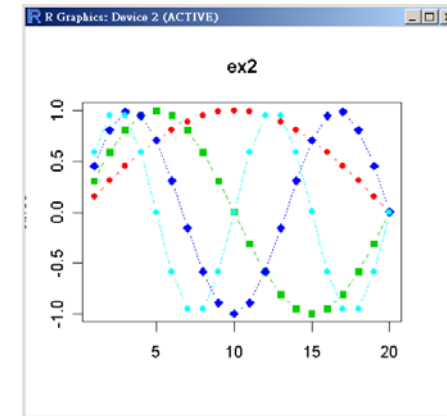
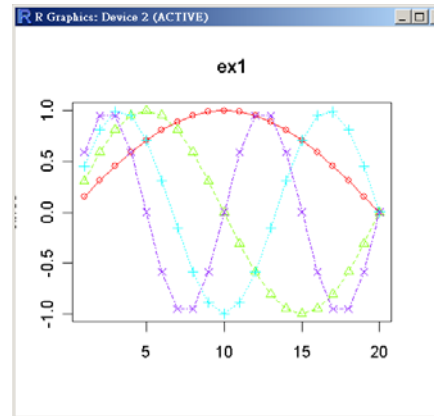
- x 為數值vector，`plot(x)`: 索引圖。
- x, y 為vector，或 `xymat` 為兩個columns的矩陣，`plot(x, y)` 或 `plot(xymat)`: x vs y 散佈圖。
- x 為時間數列變數，`plot(x)`: 時間數列圖。
- f 為因子變數，`plot(f)`: 長條圖。
- f 為因子變數， y 為數值變數，`plot(f, y)`: 比鄰盒形圖(side-by-side boxplot)。
- `plot(xdataframe)` 或 `plot(~x1+...+xk)`: x_1, \dots, x_k 的散佈圖矩陣。
- `plot(y~x1+...+xk)`: y vs x_1, y vs x_2, \dots, y vs x_k 的散佈圖。



Plot Columns of Matrices

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

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

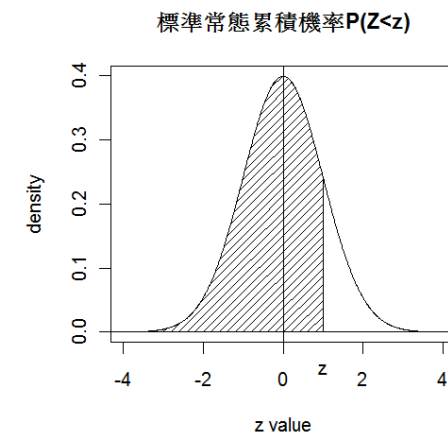
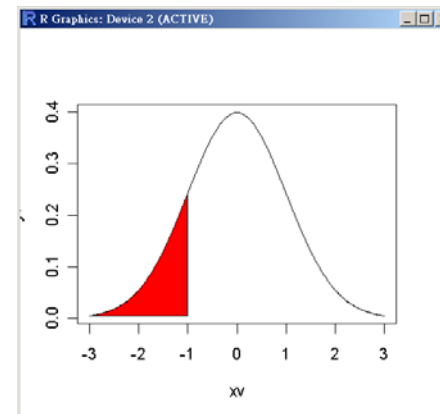
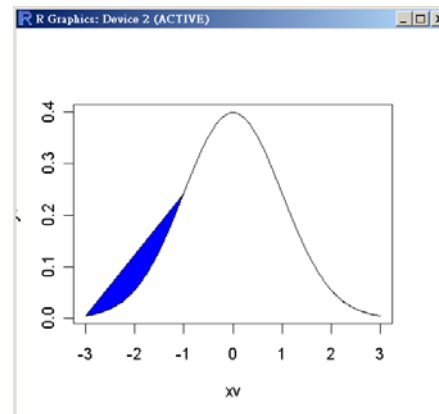
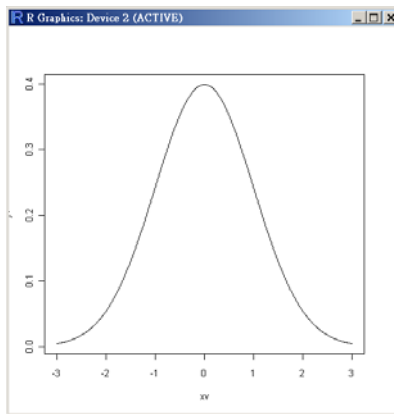


Exercise: adding legends?

多邊形 (Polygon)

130/216

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

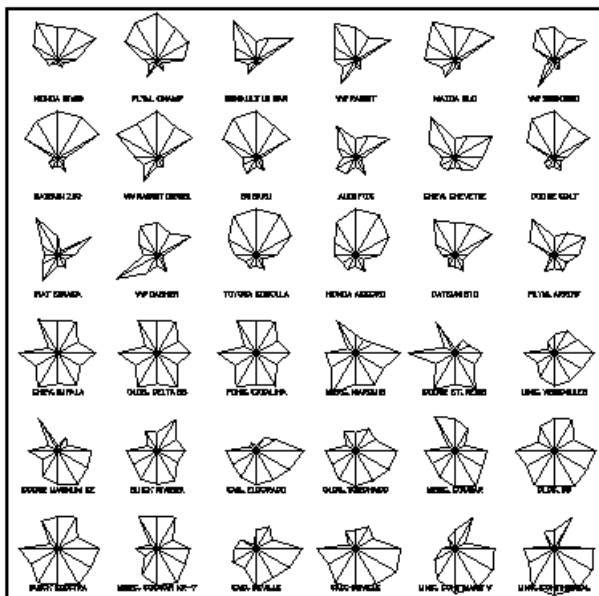


```
> xv <- seq(-3, 3, 0.01)  
> yv <- dnorm(xv)  
> plot(xv, yv, type="l")  
  
> polygon(c(xv[xv <= -1]), c(yv[xv <= -1]), col="blue")  
  
> x11()  
> plot(xv, yv, type="l")  
> polygon(c(xv[xv <= -1], -1), c(yv[xv <= -1], yv[xv == -3]), col="red")
```

Star Plot (Chambers 1983)

- The star plot consists of a sequence of equi-angular spokes, called radii, with each spoke representing one of the variables.
 - The data length of a spoke is proportional to the magnitude of the variable for the data point.
 - A line is drawn connecting the data values for each spoke.
- Typically, star plots are generated in a multi-plot format with many stars on each page and each star representing one observation.

Star plot of Automobile Data



- Each star represents one car model.
- Each ray in the star is proportional to one variable.
- The dominant pattern is that the star symbols in the top rows have long rays on the top (good price and performance) and short rays on the bottom (small in size variables), but the reverse is generally true for the heaviest models in the bottom rows.
- The primary weakness of star plot is that their effectiveness is limited to data sets with less than a few hundred points.

Star Plot

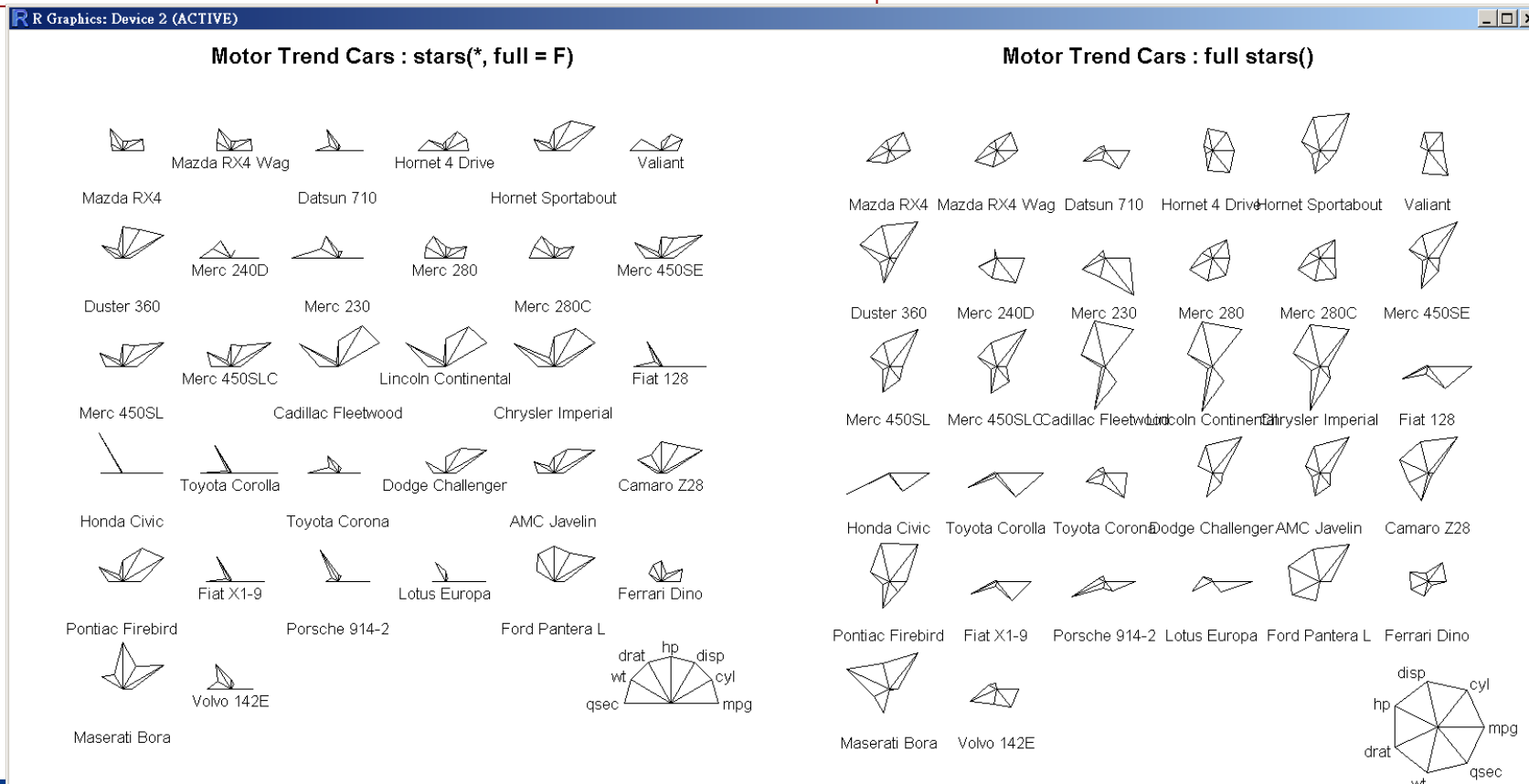
```
> head(mtcars[, 1:7])
```

	mpg	cyl	disp	hp	drat	wt	qsec
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02
Datsun 710	22.8	4	108	93	3.85	2.320	18.61
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02
Valiant	18.1	6	225	105	2.76	3.460	20.22

stars {graphics}:

Star (Spider/Radar) Plots and Segment Diagrams

?stars



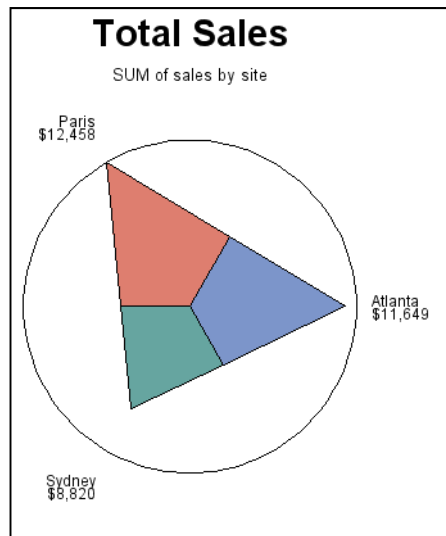


Variants of Star Plot

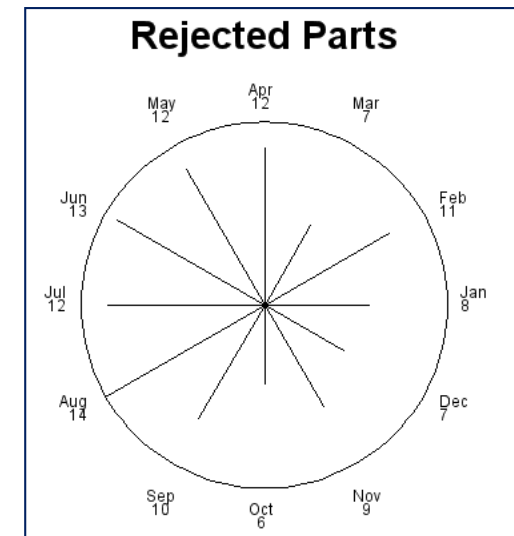
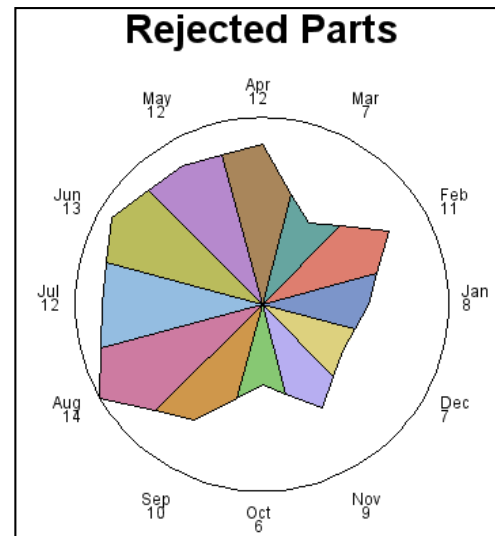
The star plot can show:

- What variables are dominant for a given observation?
- Which observations are most similar, i.e., are there clusters of observations?
- Are there outliers?

Specifying the Sum Statistic
in a Star Chart



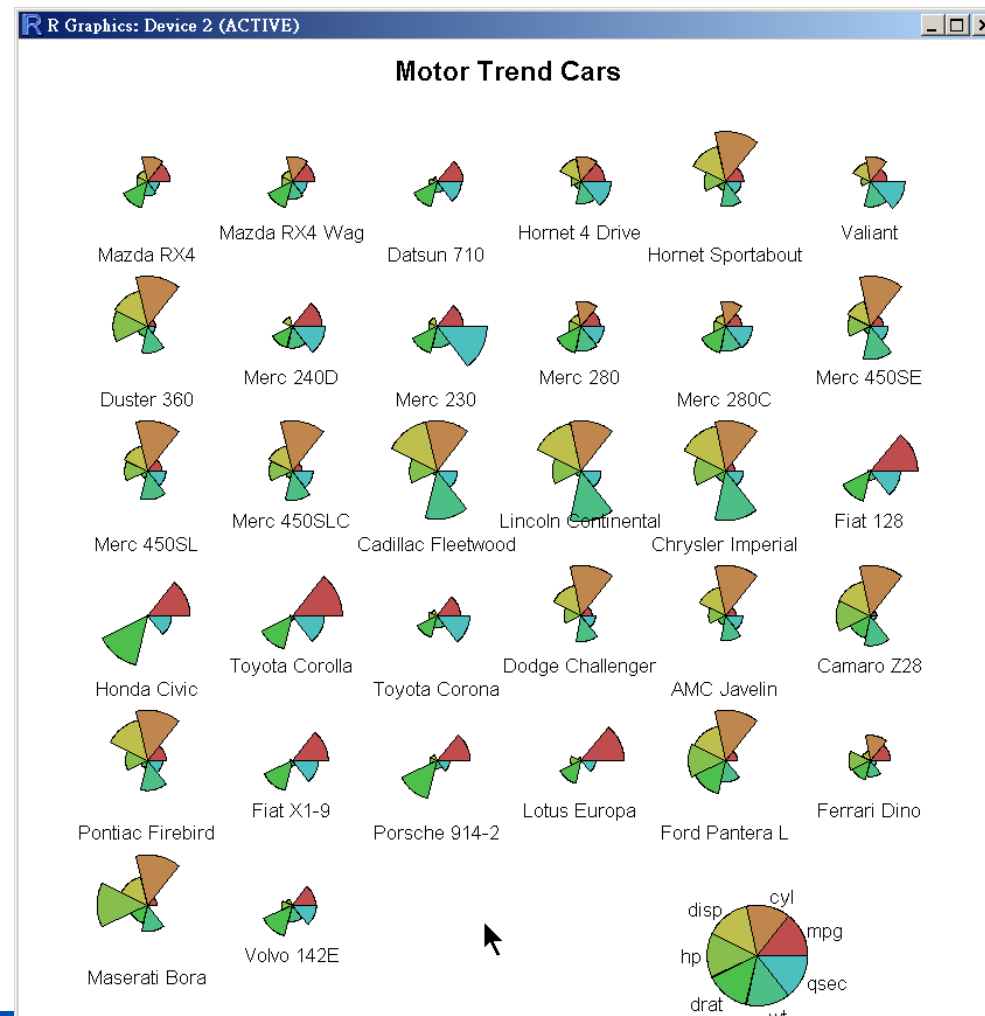
Charting a Discrete Numeric Variable in a Star Chart



Figures Source: SAS/GRAPH® 9.2 Reference, 2nd Edition

Segment Diagrams

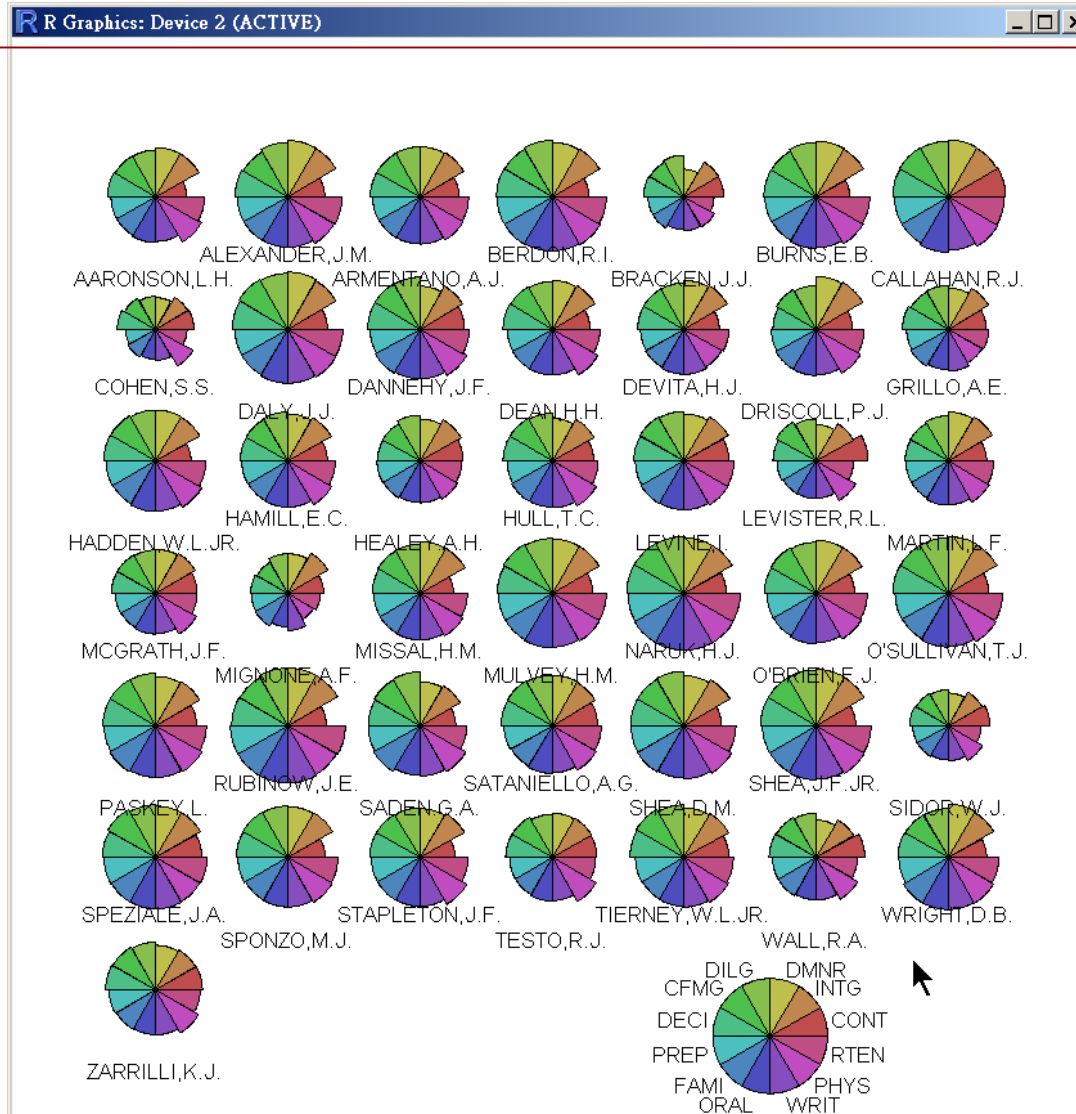
```
palette(rainbow(12, s = 0.6, v = 0.75))
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),
+     main = "Motor Trend Cars", draw.segments = TRUE)
```





Segments

```
stars(USJudge, draw.segments = TRUE, scale = FALSE, key.loc =  
c(13,1.5))
```



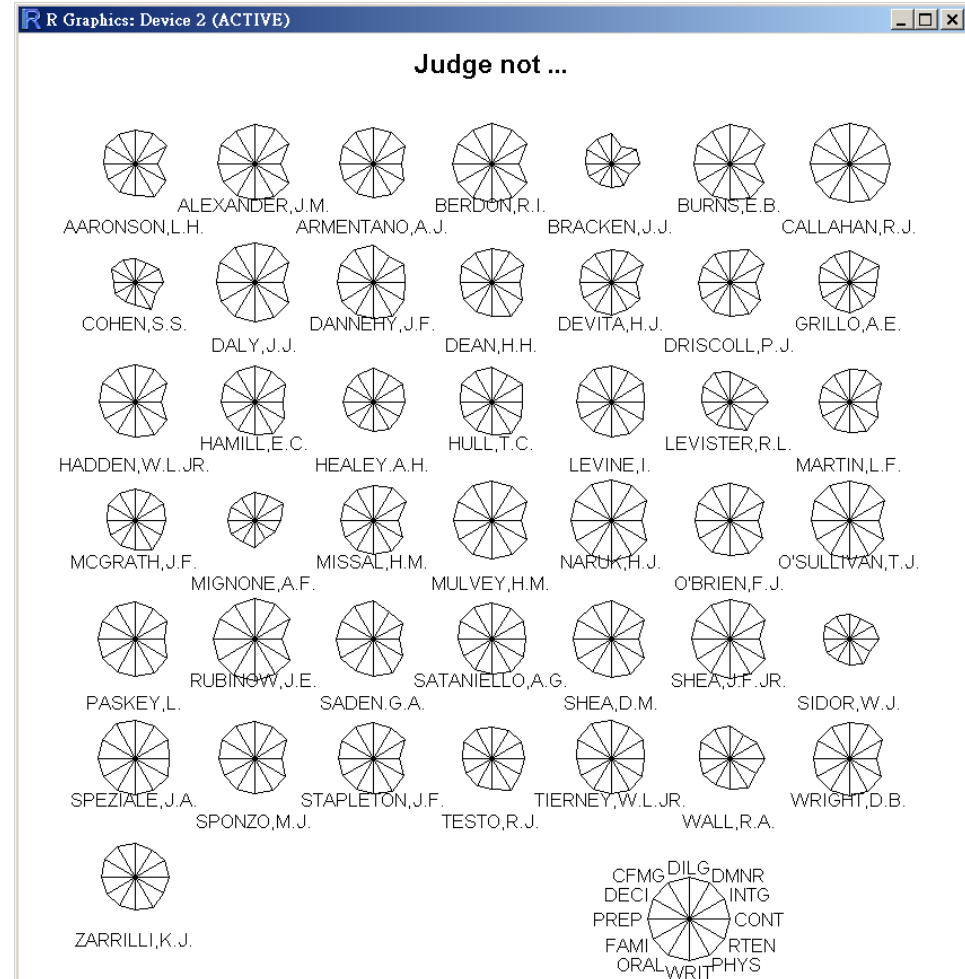
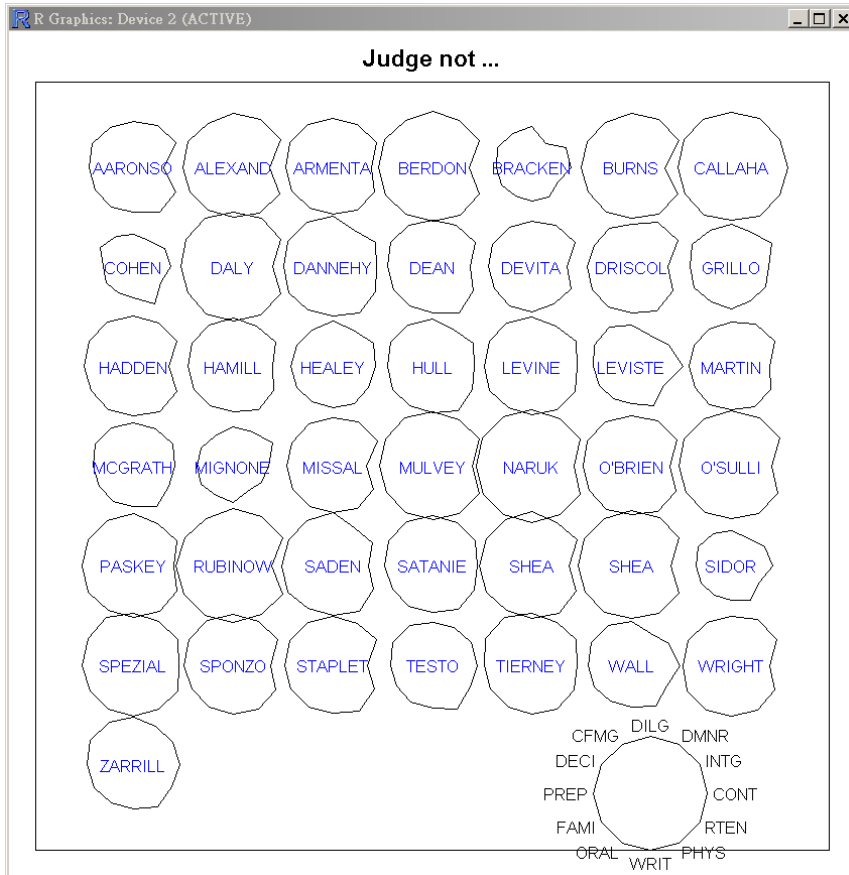


Scale linearly (not affinely) to [0, 1] 136/216

stars {graphics}:

Star (Spider/Radar) Plots and Segment Diagrams

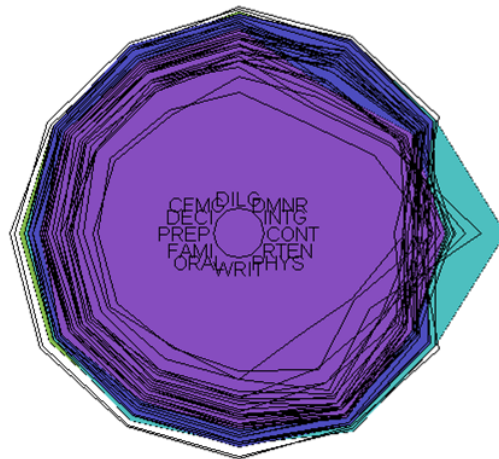
?starts



Spider

```
> par(mfrow=c(1,2))
> stars(USJudgeRatings, locations = c(0, 0), scale = FALSE,
+ radius = FALSE, col.stars = 1:10, key.loc = c(0, 0),
+ main = "US Judges rated")
> # Same as above, but with colored lines instead of filled polygons.
> stars(USJudgeRatings, locations = c(0, 0), scale = FALSE,
+ radius = FALSE, col.lines = 1:10, key.loc = c(0, 0),
+ main = "US Judges rated")
```

US Judges rated

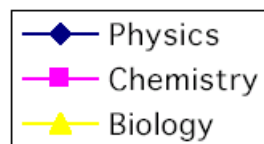


US Judges rated

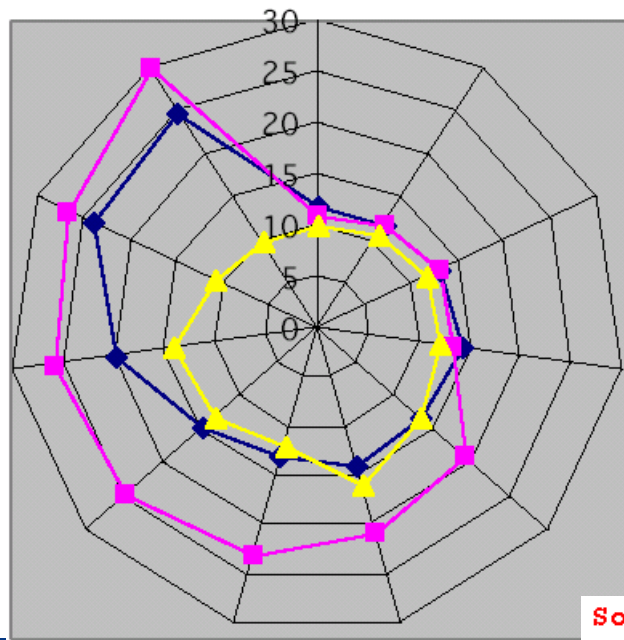


Radar Plot

- The idea of a radar plot is similar to that of the star plot.
- In a radar plot, the value of the measurement is also represented by radii stretching out from the center of a circle.
- **However, here each radius stands for a subject instead of a variable.**
- The subjects' response on each variable is displayed by points of different shapes, colors, or both.



Achievement Scores in three classes of eleven subjects.

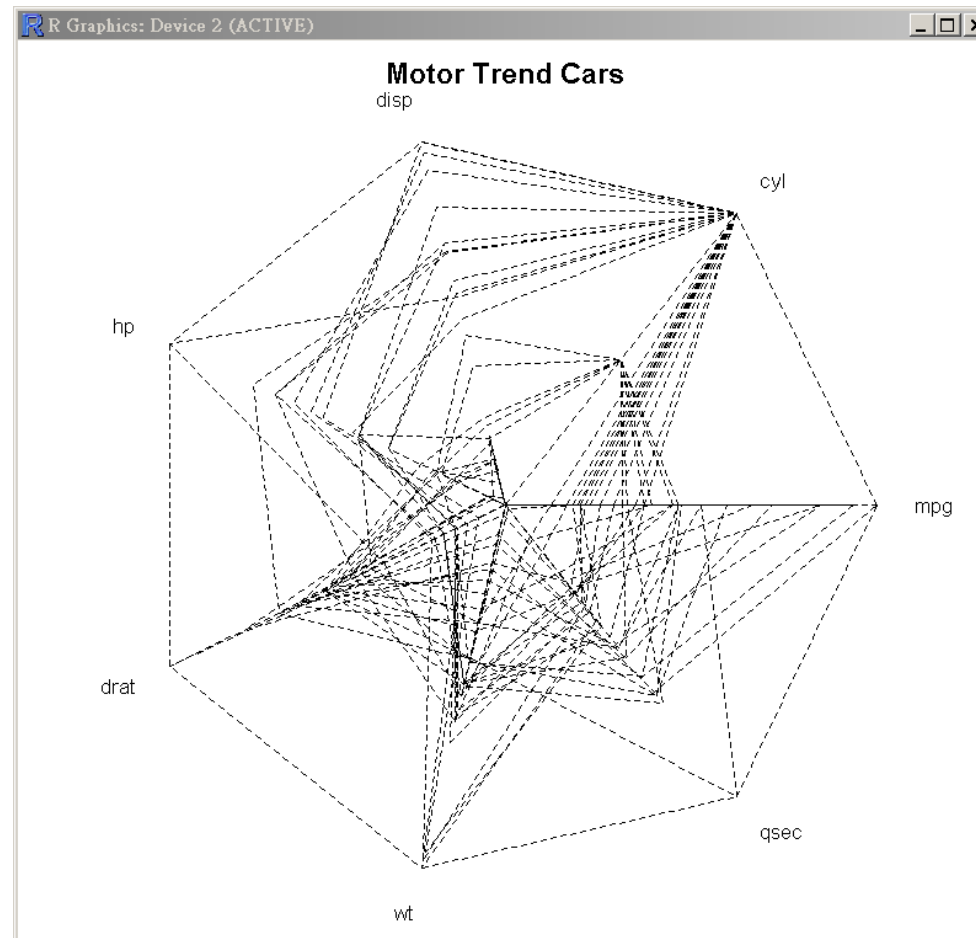


Source: ATKOSoft: Survey on Visualisation methods and software tools

- The graph shows the frequencies of data series relative to one another.
- Apparently, the biology scores are the lowest and are not correlated to neither physics nor chemistry scores.
- However, physics and chemistry scores are good predictors to each other.
This approach suffers the same shortcoming as the star graph. When there are too many variables and subjects, the data pattern will be concealed.

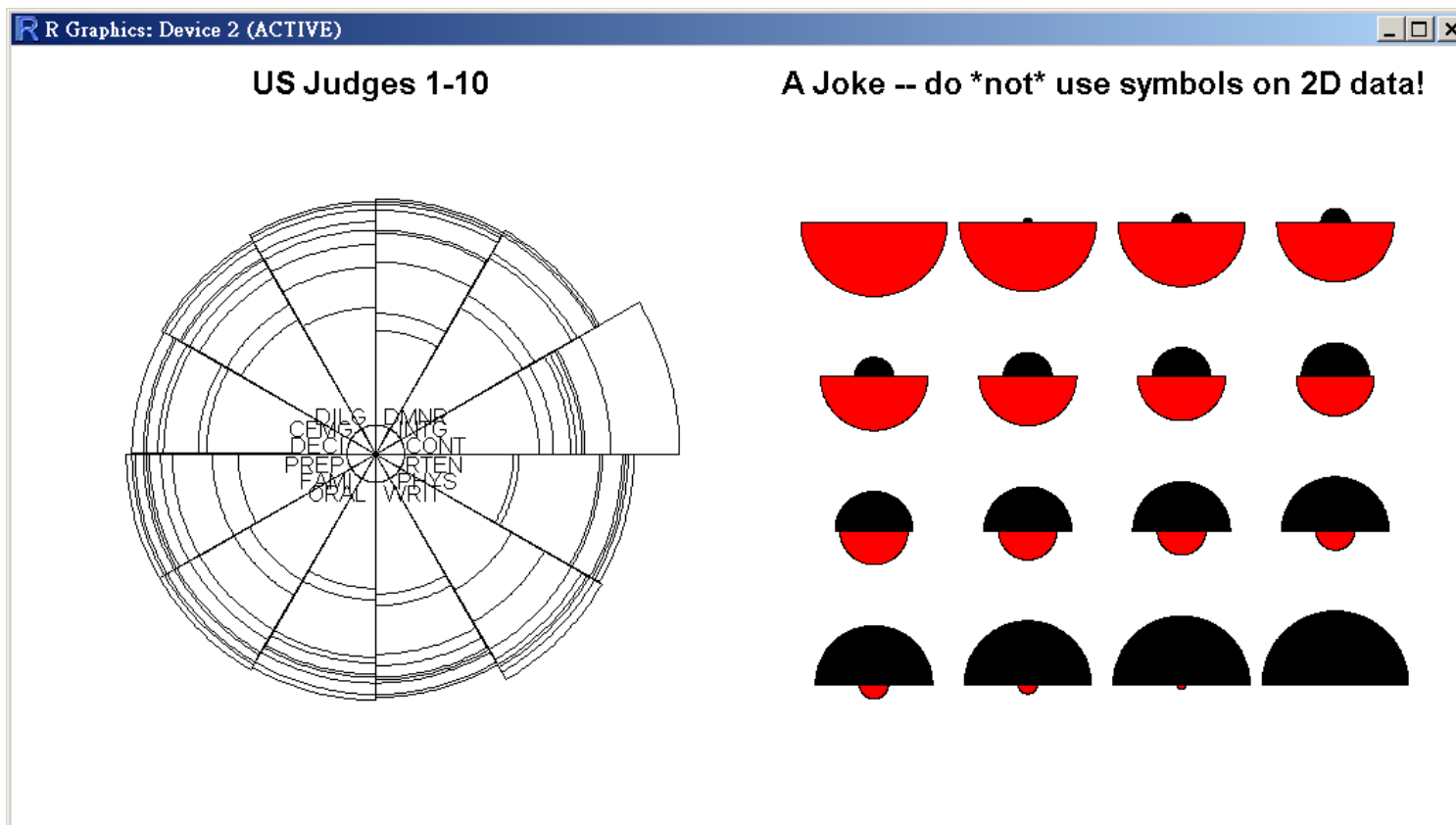
Spider or Radar Plot

```
stars(mtcars[, 1:7], locations = c(0, 0), radius = FALSE,  
      key.loc = c(0, 0), main = "Motor Trend Cars", lty = 2)
```



Radar-Segments

```
stars(USJudgeRatings[1:10,], locations = 0:1, scale = FALSE,
      draw.segments = TRUE, col.segments = 0, col.stars = 1:10, key.loc = 0:1,
      main = "US Judges 1-10 ")
palette("default")
stars(cbind(1:16, 10*(16:1)), draw.segments = TRUE,
      main = "A Joke -- do *not* use symbols on 2D data!")
```

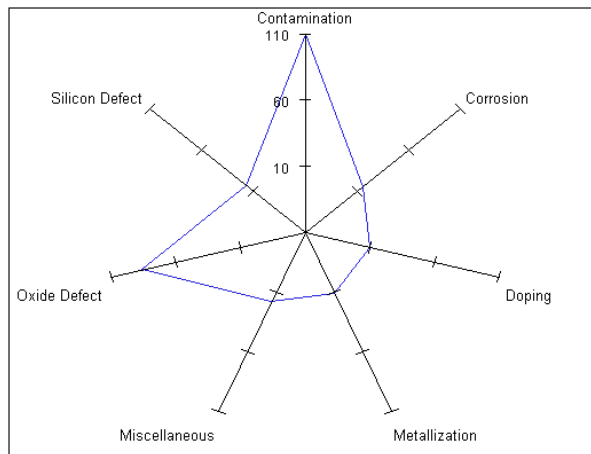




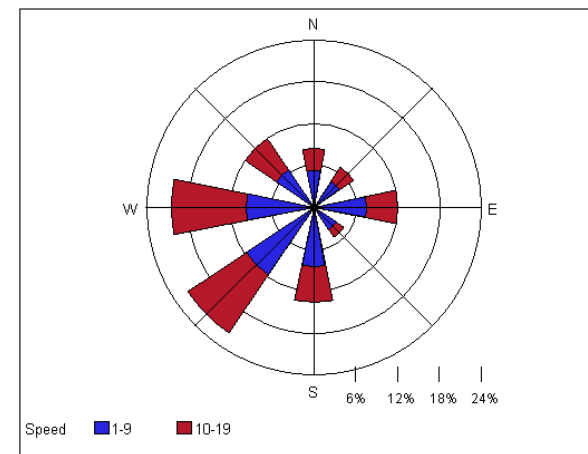
Variants of Radar Plot

Figures Source: SAS/GRAPH® 9.2 Reference, 2nd Edition

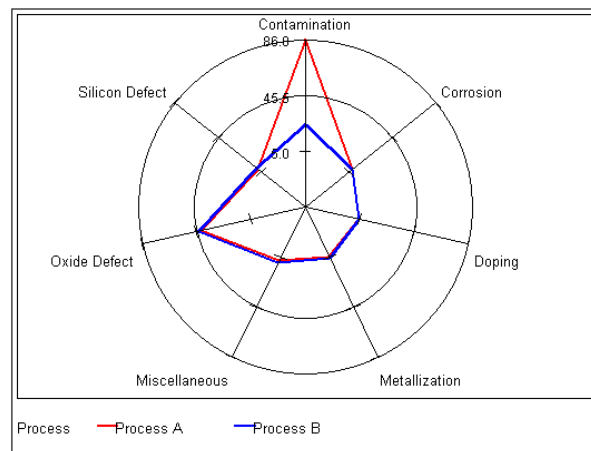
Producing a Basic Radar Chart



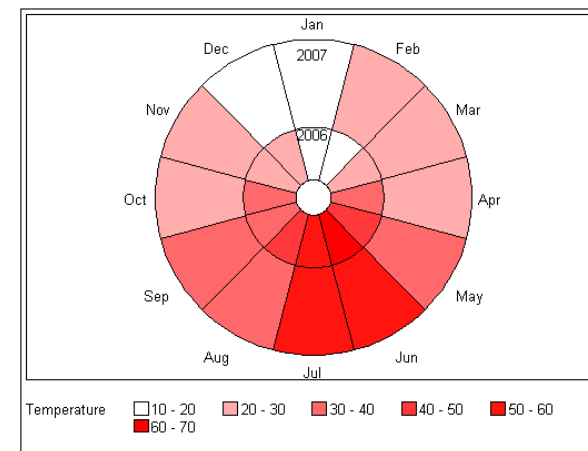
Creating a Windrose Chart



Modifying the Appearance of Radar Charts



Creating a Calendar Chart

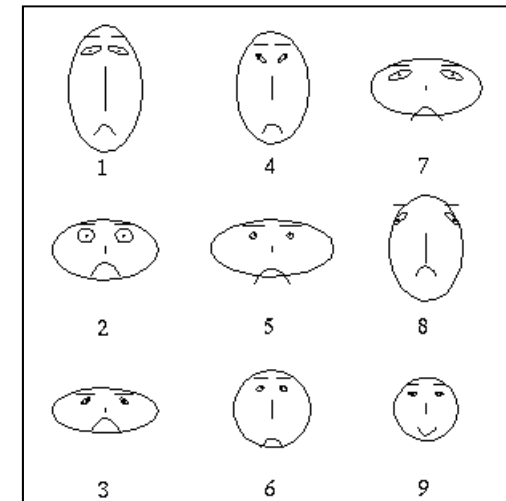


Chernoff Faces Plot (Chernoff, 1973)

- Each facial feature denotes a particular variable.
 - For example, X1 can be associated with the size of the mouth, X2 with the size of the nose, X3 with the size of the eyes, and so on.
- The power of Chernoff face is its highly condensation of data and its interesting way of presentation.
- A **major drawback**: the subjective assignment of facial expressions to variables affects on the shape of the face.

Chernoff and Rizvi (1975) found that the permutations of the assignment of features caused an error rate of as high as 25 for the task of classifying faces into groups.

- It means that classifying two faces as "fairly similar" is greatly influenced by the assignment of variables to specific features.
- Some researchers criticised that the symmetrical feature of Chernoff faces is redundant.
- Like the star graph, the power of showing multiple relationships in Chernoff faces are limited in a still mode.



Source: ATKOSoft: Survey on Visualisation methods and software tools

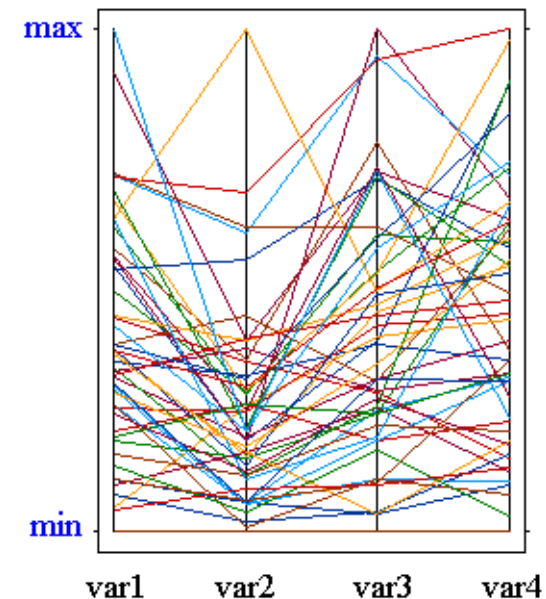


Parallel Coordinates Plot

- The Parallel Plot is a series of **parallel Dotplots** with lines linking the data values of each of the zones instead of individual symbols on each axis.
 - Variables to be visualized are represented by a series of parallel axes as replacement of the standard orthogonal approach.
 - Then, each row from the multidimensional dataset is represented as a series of unbroken line segments, which connect the parallel axes.

Parallel plot is useful to quickly identify interactions between variables:

- Clusters of observations with the similar lines across all axes.
- Direct relationship between a pair of variables appears in the plot as two axes connected by a series of parallel lines.
- Inverse relationship between two variables should be displayed as a series of lines, which cross each other.

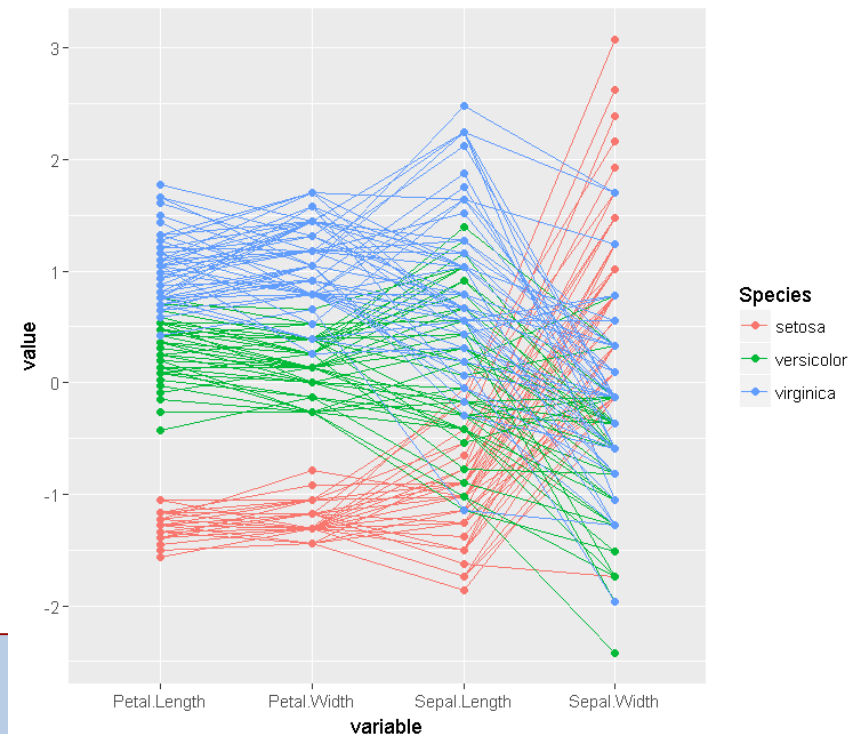
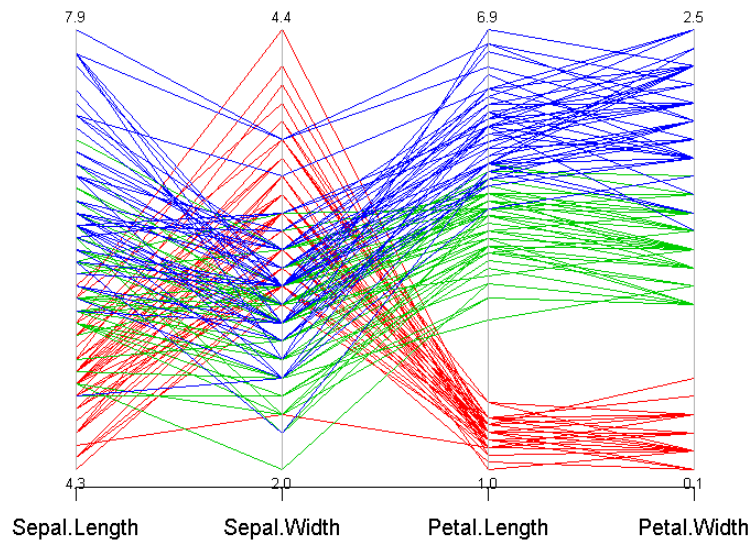




範例: Parallel Coordinates Plot

144/216

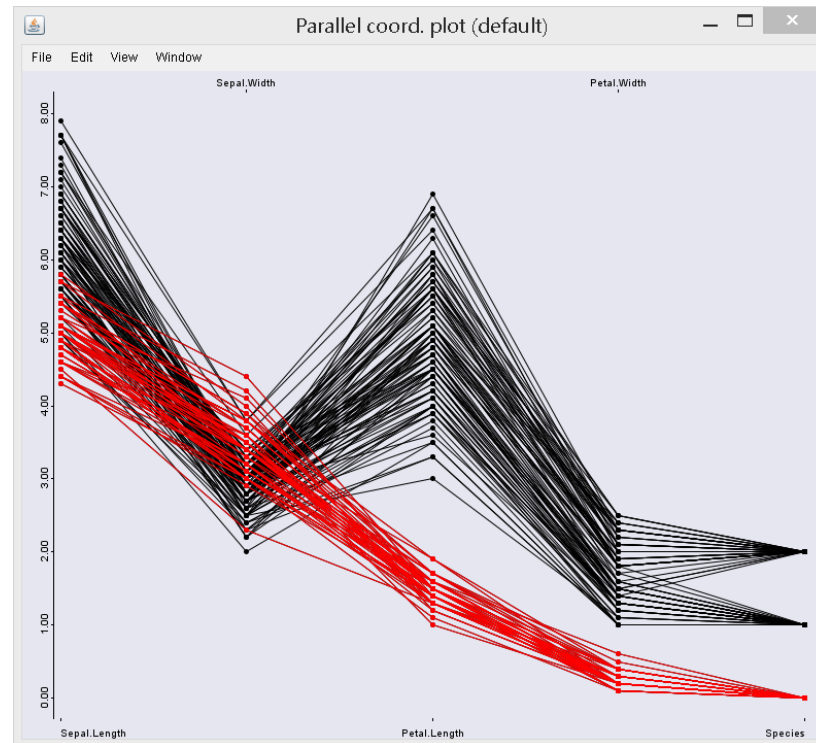
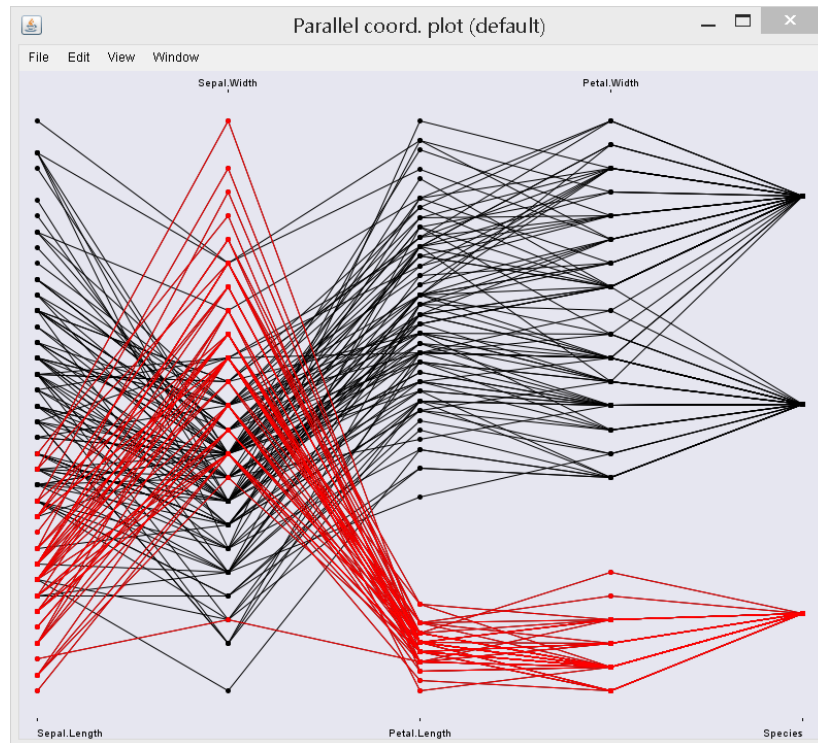
```
> library(MASS)
> parcoord(iris[,1:4], col=as.integer(iris[,5])+1, var.label = T)
> library(GGally) # Extension to 'ggplot2'
> ggparcoord(data = iris, columns = 1:4, groupColumn = 5)
> ggparcoord(data = iris, columns = 1:4, groupColumn = 5, boxplot = T)
> ggparcoord(data = iris, columns = 1:4, groupColumn = 5, order = "anyClass",
+ showPoints = TRUE)
```



```
ggparcoord(data, columns = 1:ncol(data),
groupColumn = NULL,
scale = "std", scaleSummary = "mean", centerObsID = 1,
missing = "exclude", order = columns, showPoints = FALSE,
splineFactor = FALSE, alphaLines = 1, boxplot = FALSE,
shadeBox = NULL, mapping = NULL, title = "")
```

範例: Interactive PCP

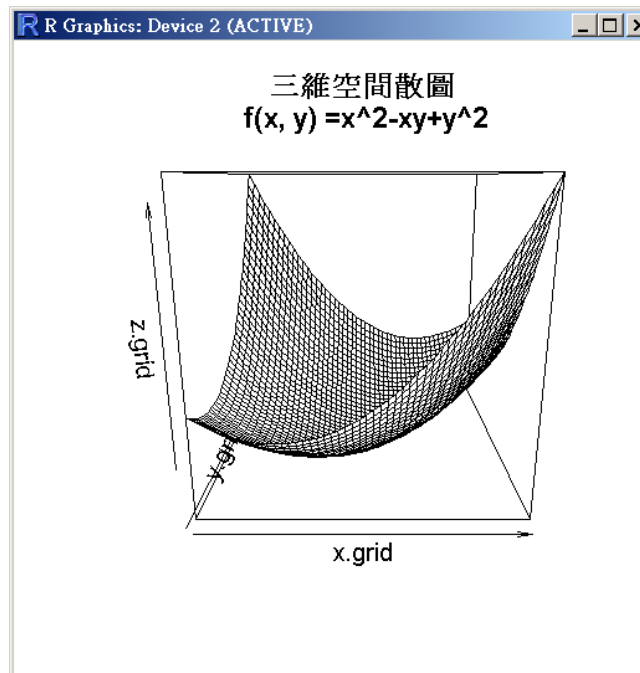
```
> library(iplots) # Interactive Parallel Coordinates Plot
Loading required package: rJava
Warning messages:
1: package 'iplots' was built under R version 3.4.2
2: package 'rJava' was built under R version 3.4.1
> ipcp(iris)
ID:1 Name: "Parallel coord. plot (default)"
```



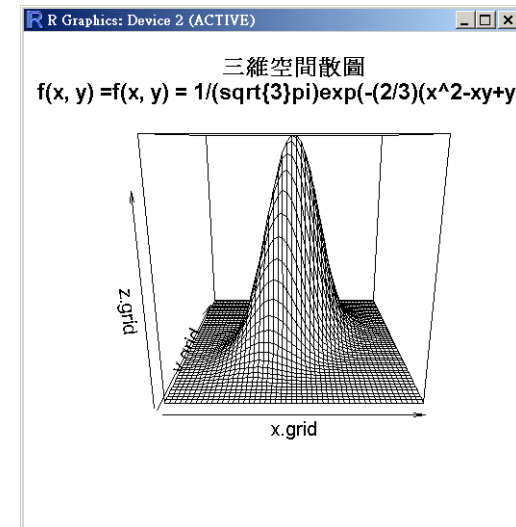
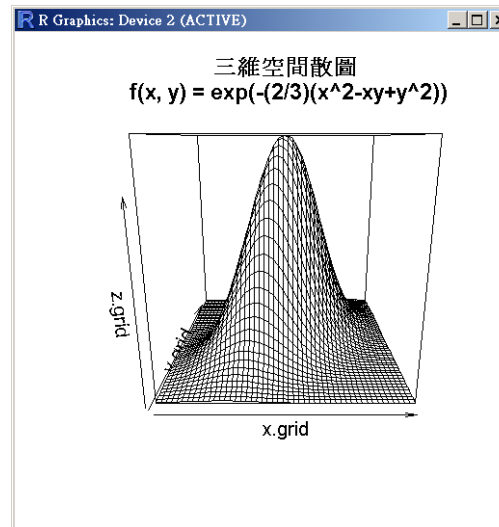
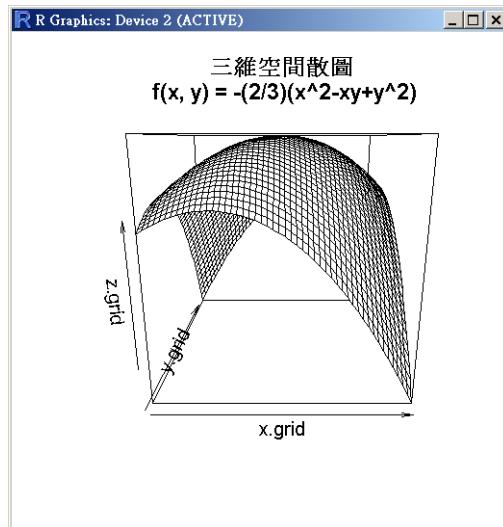
3D透視圖: persp

- 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)
```



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

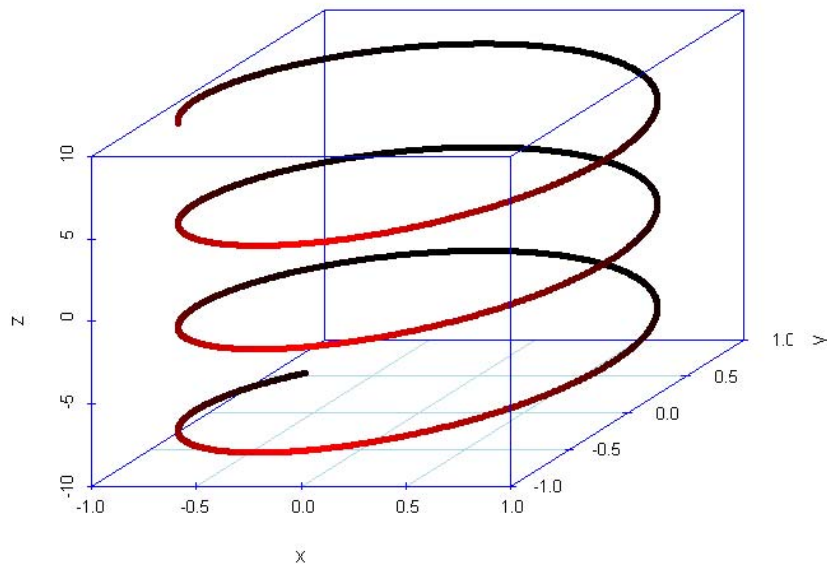




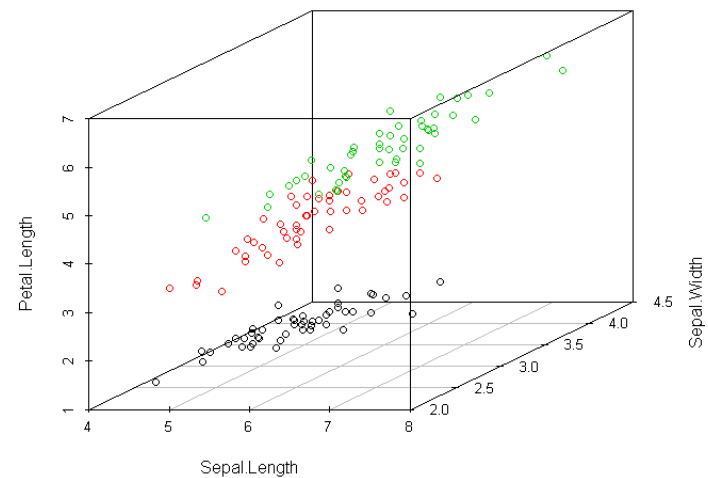
3D散佈圖

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

scatterplot3d - 1



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



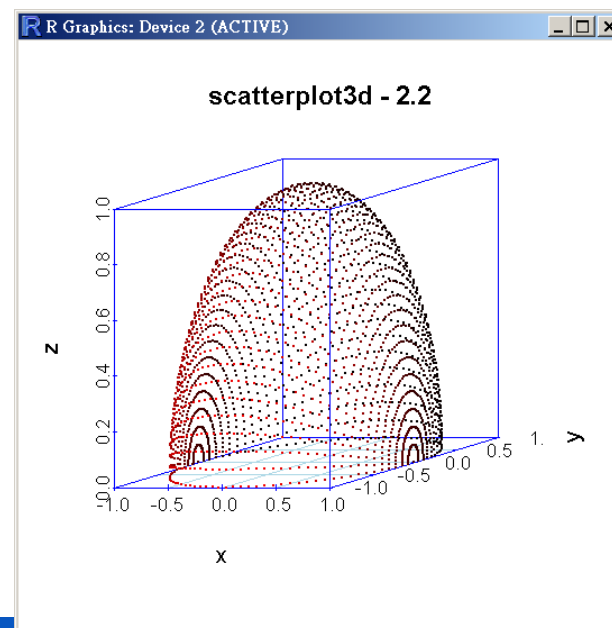
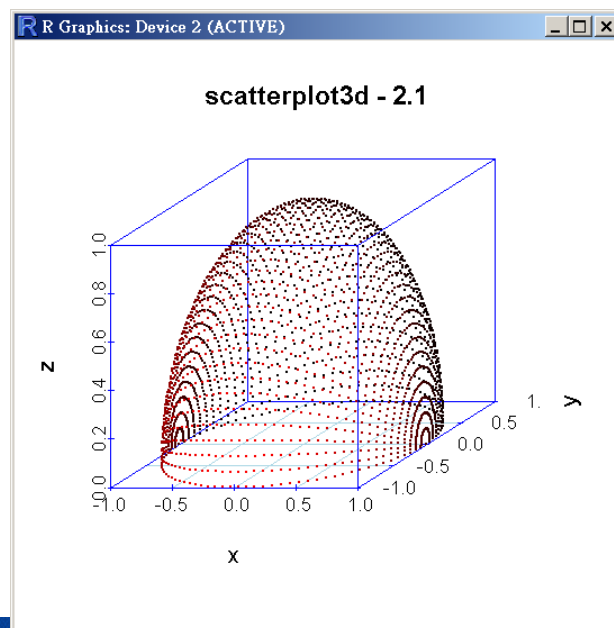
3D散佈圖

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

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

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

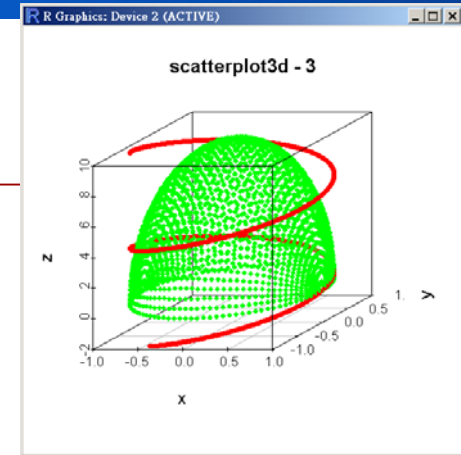
> scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue", col.grid="lightblue",
  main="scatterplot3d - 2.2", pch=20, cex.symbols=0.5, angle= 20)
```





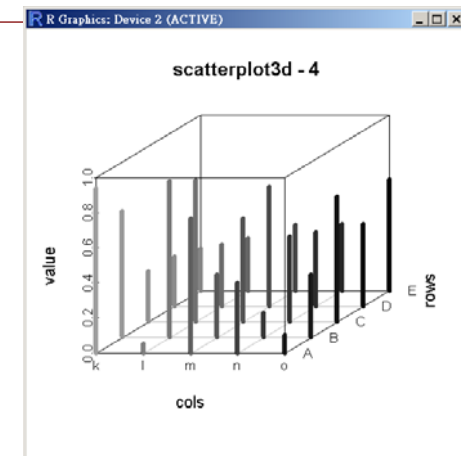
3D散佈圖

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



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

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



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



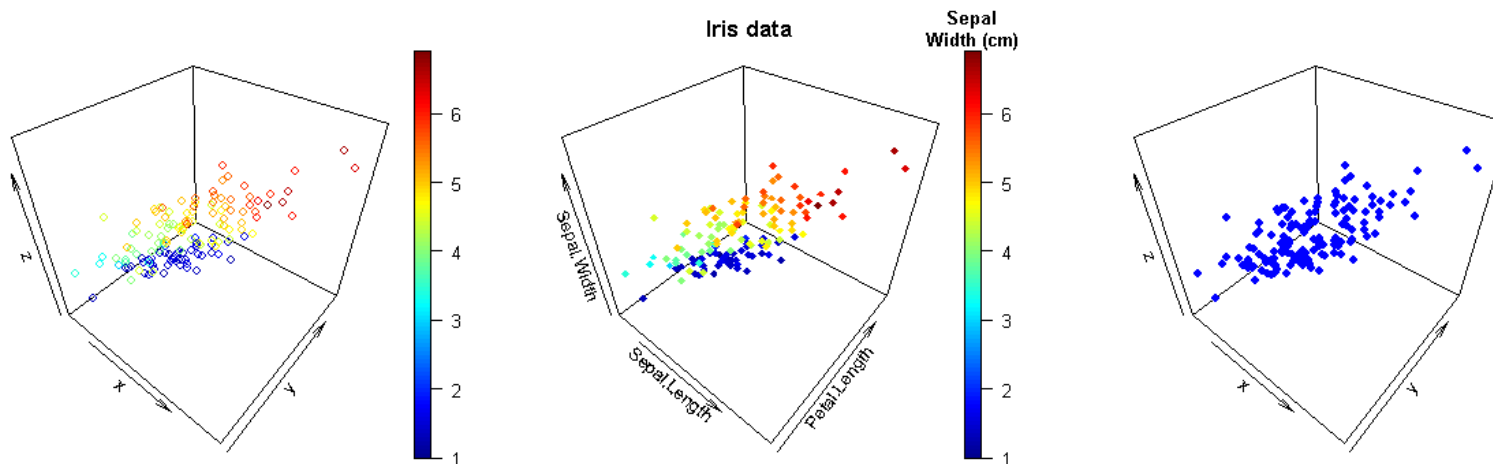
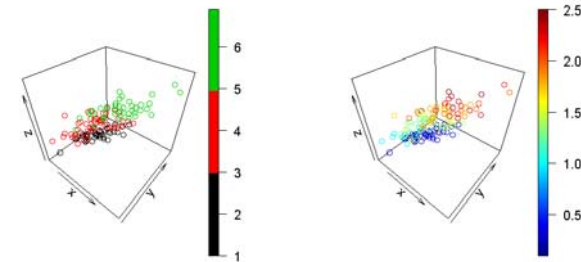
plot3D package

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

```
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)
```

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



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

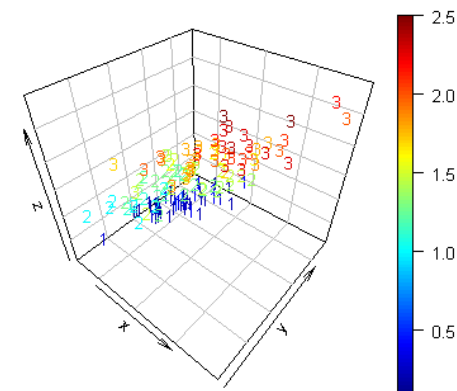
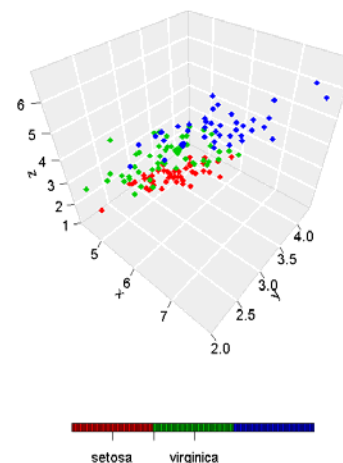
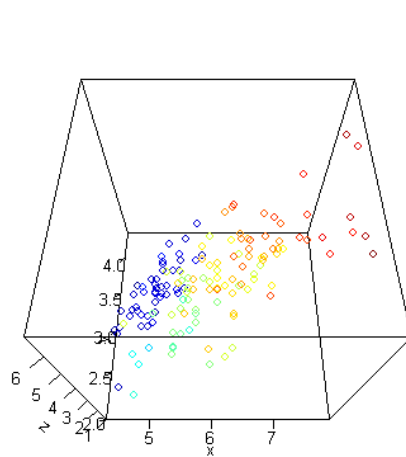
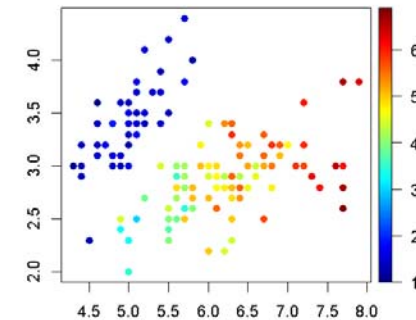
plot3D package

```

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)

```

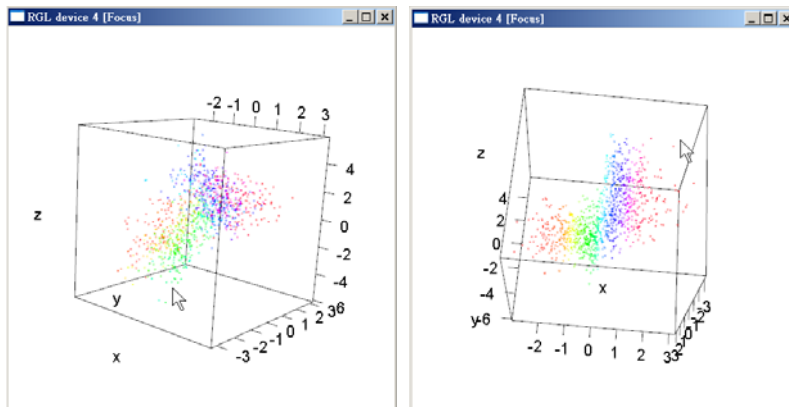
> library(rgl)
> open3d()
> x <- sort(rnorm(1000))
> y <- rnorm(1000)
> z <- rnorm(1000) + atan2(x,y)
> plot3d(x, y, z, col=rainbow(1000), size=2)
>
> M <- par3d("userMatrix")
> play3d(par3dinterp(userMatrix=list(M, rotate3d(M, pi/2, 1, 0, 0),
+ rotate3d(M, pi/2, 0, 1, 0))),
+ duration=4) # 90 degree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

```

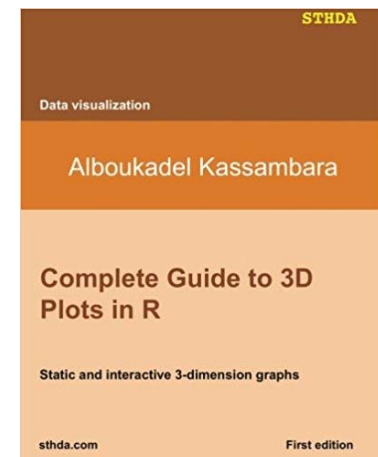
```

> 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/>



Simulated Swissroll Data Set

154/216

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

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

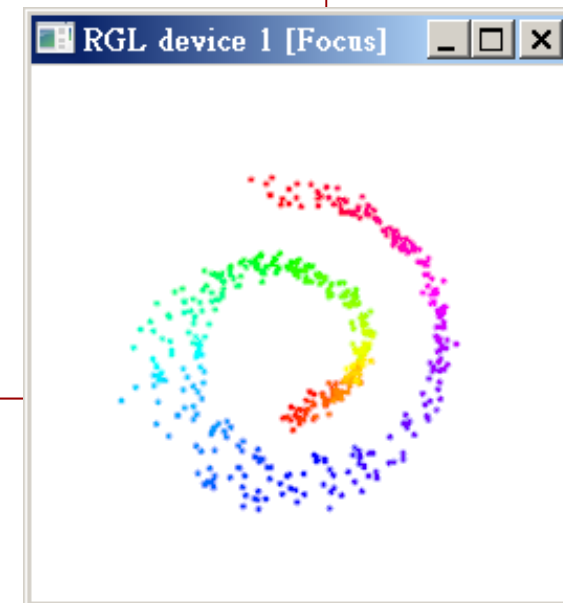
  order.angle <- order(angle)
  sort.angle <- sort(order.angle, index.return=TRUE)
  col.id <- rainbow(n)
  my.color <- col.id[sort.angle$ix]

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

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

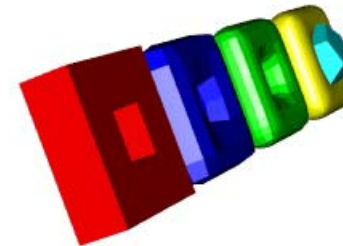
swissdata <- swissroll(500)
xdata <- swissdata$xdata
x.color <- swissdata$color
open3d()
plot3d(xdata[,1], xdata[,2], xdata[,3], col=x.color,
       size=3, xlab="", ylab="", zlab="", axes = FALSE)
```

```
# export graphics
rgl.snapshot("test.png")
rgl.postscript("test.pdf", "pdf")
```



範例(1): rgl

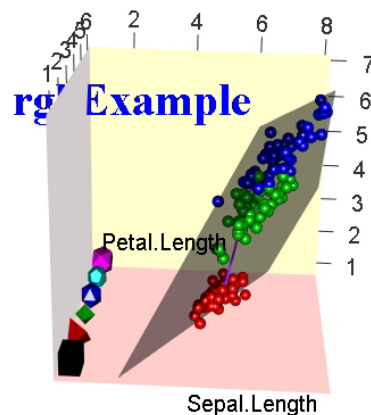
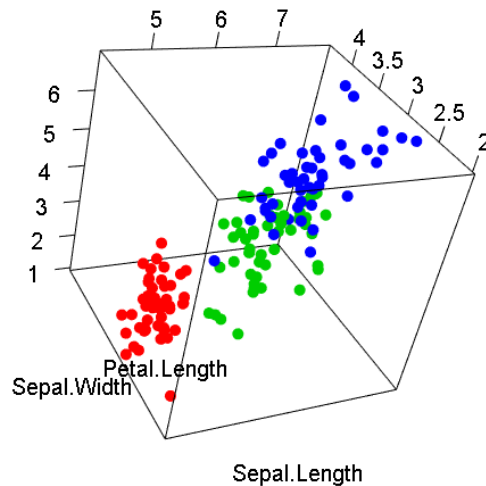
OpenGL uses homogeneous coordinates to handle perspective and affine transformations. The homogeneous point (x, y, z, w) corresponds to the Euclidean point $(x/w, y/w, z/w)$.



```
> library(rgl)
> open3d()
> o1 <- oh3d(tran = par3d("userMatrix"), color="red") # "o" objects
> shade3d(translate3d(o1, -6, 0, 0))
> o2 <- subdivision3d(o1) # divides and refines a given mesh
> shade3d(translate3d(o2, -2, 0, 0), color="blue")
> o3 <- subdivision3d(o2)
> shade3d(translate3d(o3, 2, 0, 0), color="green")
> o4 <- subdivision3d(o3)
> shade3d(translate3d(o4, 6, 0, 0), color="yellow")
> shade3d(translate3d(dodecahedron3d(col = "cyan"), 6, 0, 0)) # 十二面體
>
> # Platonic solids
> # tetrahedron3d (四面體), cube3d, octahedron3d (八面體),
> # dodecahedron3d (十二面體), icosahedron3d (二十面體)
> play3d(spin3d(), duration=10)
```


範例 (2): rgl

```
> library("rgl")
> open3d()
> plot3d(iris[,1:3], col=as.integer(iris[,5])+1, type="p", size=10)
```



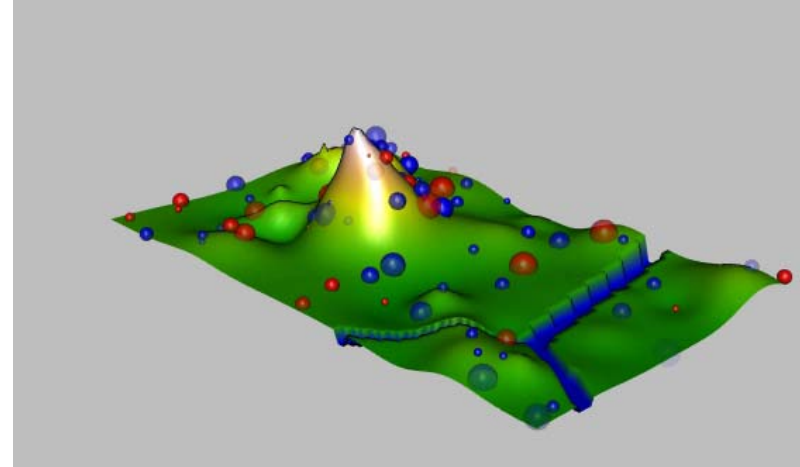
```
> plot3d(iris[,1:3], col=as.integer(iris[,5])+1, type="s",
+        radius=0.15)
> bbox3d(color=c("red", "black"), emission="gray",
+        specular="yellow", shininess=5, alpha=0.8, nticks = 3)
> aspect3d(1,1,1)
>
> lines3d(iris[c(1, 150), 1:3], col="purple", lwd=2)
> # points3d, lines3d, segments3d, triangles3d, quads3d.
>
> shapes <- list(cube3d(), tetrahedron3d(), octahedron3d(),
+               icosahedron3d(), dodecahedron3d(), cuboctahedron3d())
> shapelist3d(shapes, x=1, y=1:6, z=1, size=0.3, col=1:6)
> aspect3d(1,1,1)
>
> texts3d(x=2, y=6, z=6, texts="rgl Example", font=2,
+        color="blue", cex=2, family="serif")
>
> # Show regression plane with z as dependent variable
> fit <- lm(iris[,3] ~ iris[,1] + iris[,2])
> coefs <- coef(fit)
> planes3d(a=coefs[2], b=coefs[3], c=-1, d= coefs["(Intercept)"],
+         alpha = 0.5)
> # planes3d draws planes using ax + by + cz + d = 0.
>
> play3d(spin3d(axis = c(0, 0, 1), rpm = 20), duration = 4)
```

範例 (3): rgl

```

> terrain <- as.matrix(read.table("terrain_data.txt", header=F))
> dim(terrain)
[1] 100 100
> animal <- read.table("animal_data.txt", header=T)
> dim(animal)
[1] 100 5
> attach(animal)
> head(animal, 3)
      loc.x    loc.y number sex    index
1  1.421804 0.1536418     4  1 7.571144
2 86.589918 0.7205304     3  1 6.855152
3 58.427946 2.1297322     2  0 7.526096
> # Define colors for terrain
> # see ? surface3d for how to assign colors
> terrain.scale <- floor((terrain - min(terrain))/(max(terrain) - min(terrain))*99)+1
> terrain.color <- terrain.colors(100)[terrain.scale]
> terrain.color[terrain==0] <- rgb(0, 0, 1) # set color for river
> open3d()
> clear3d("all")
> bg3d(col="gray") # setup background
> light3d() # setup head-light
> surface3d(1:100, seq(1,60,length=100), terrain, col=terrain.color, back="lines")
> sex.col <- ifelse(sex==0, rgb(0, 0, 1), rgb(1, 0, 0)) # males: blue, females: red
>
> z <- terrain[cbind(ceiling(loc.x), ceiling(loc.y*10/6))]
> alpha.index <- (index-min(index))/(max(index)-min(index))
> spheres3d(loc.x, loc.y, z + 0.5,
+           radius=0.3*number, col=sex.col,
+           alpha=alpha.index)
> detach(animal)
> play3d(spin3d(), duration=10)

```



```

# this example is modified
from rgl.demo.abundance()
# see rgl.demo.abundance

```



範例 (4): rgl, explore a comet

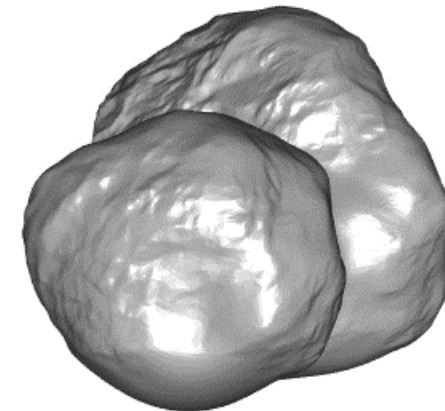
158/216

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")
```

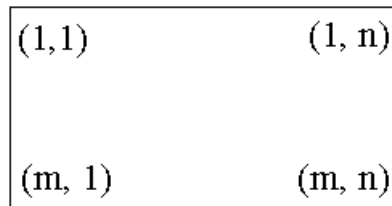
it: indices for triangular faces
ib: indices for quad faces
vb: matrix of vertices: 4xn matrix (rows x, y, z, h) or equivalent vector, where h indicates scaling of each plotted quad



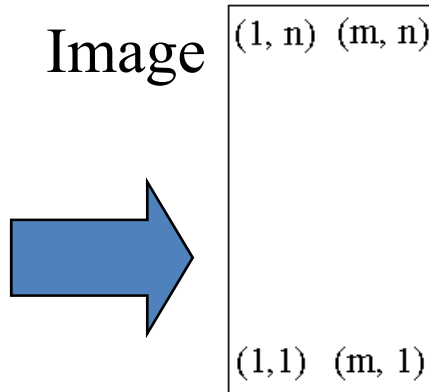
Display a Color Image

- `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

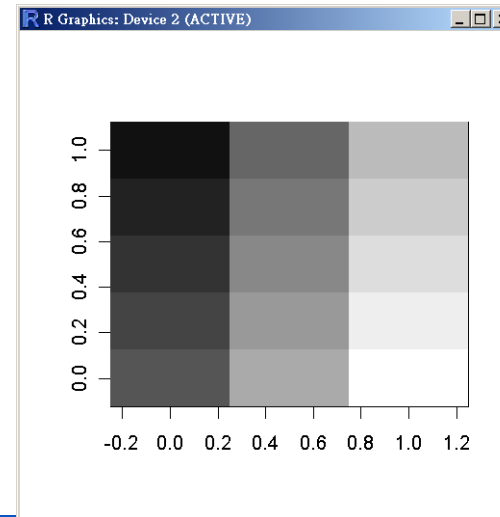
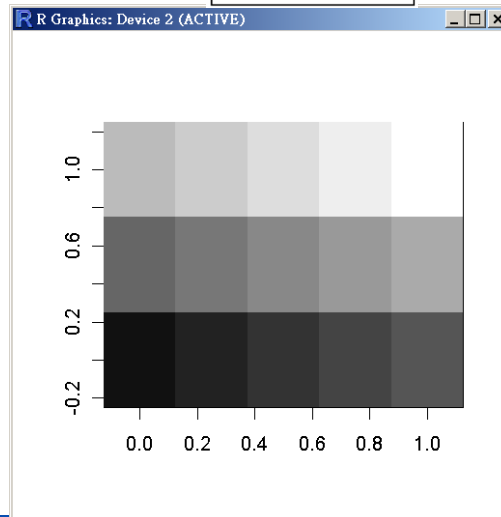


Image



```
> 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
```



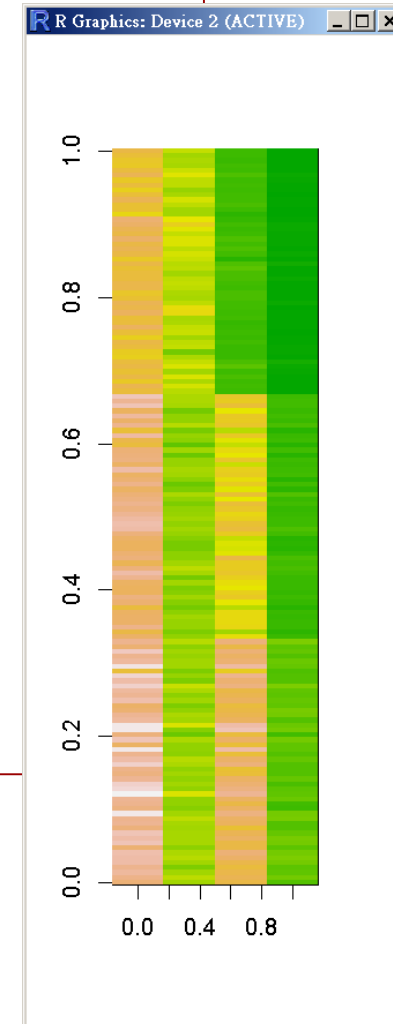


練習

```
> 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
```



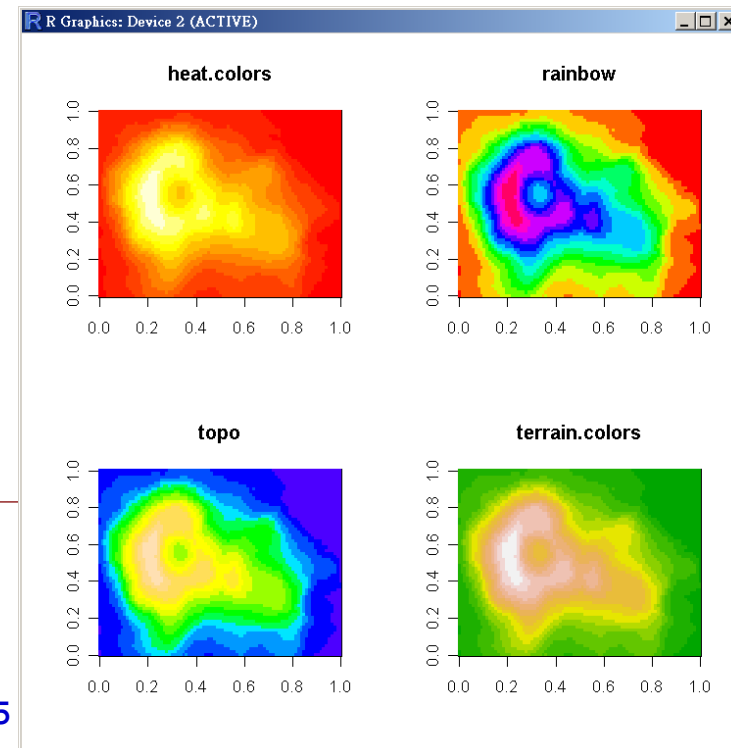
Example: volcano

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

```

> data(volcano)
> dim(volcano)
[1] 87 61
> str(volcano)
  num [1:87, 1:61] 100 101 102 103 104 105 105
> par(mfrow = c(2, 2))
> image(volcano, main = "heat.colors")
> image(volcano, main = "rainbow", col = rainbow(15))
> image(volcano, main = "topo", col = topo.colors(15))
> image(volcano, main = "terrain.colors", col = terrain.colors(15))

```

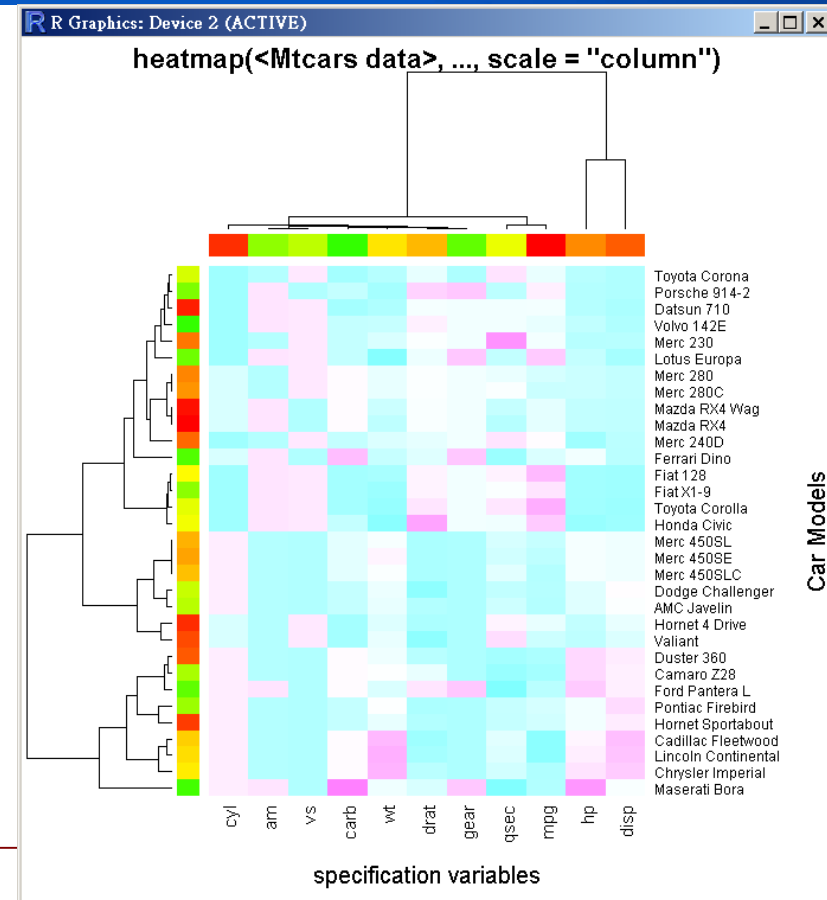


熱圖 (Heatmap)

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

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

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



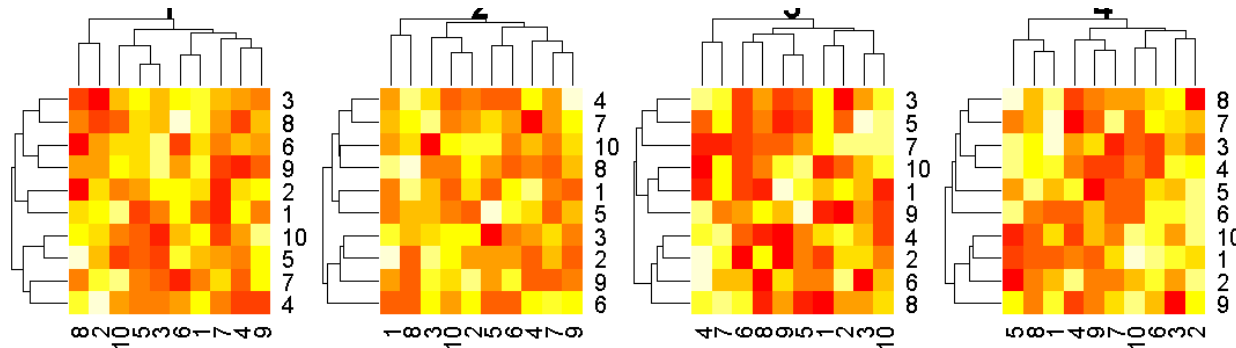


Multiple heatmaps on one page

```
> # par(mfrow=c(1, 2)) # not working
> # install.packages("gridGraphics")
> library(gridGraphics)
> library(grid)
>
> no.data <- 4
> random.data <- function() { matrix(sample(1:100), nrow=10, ncol=10) }
> simulated.data <- replicate(no.data, random.data(), simplify = FALSE)
> class(simulated.data) # simplify = FALSE: "list"; ow. "array"
[1] "list"
>
> grid.newpage() # erases the current device or moves to a new page
> library(gridExtra) # install.packages("gridExtra")
> grid.arrange(grobs=myplots, ncol=no.data)
```

```
> myplots <- lapply(1:no.data, function(i){
+   heatmap(simulated.data[[i]], main=paste(i))
+   grid.echo()
+   grid.grab()
+ })
```

grid.echo is for base graphics



```
arrangeGrob(..., grobs = list(...), layout_matrix, vp = NULL,
  name = "arrange", as.table = TRUE, respect = FALSE, clip = "off",
  nrow = NULL, ncol = NULL, widths = NULL, heights = NULL, top = NULL,
  bottom = NULL, left = NULL, right = NULL, padding = unit(0.5, "line"))

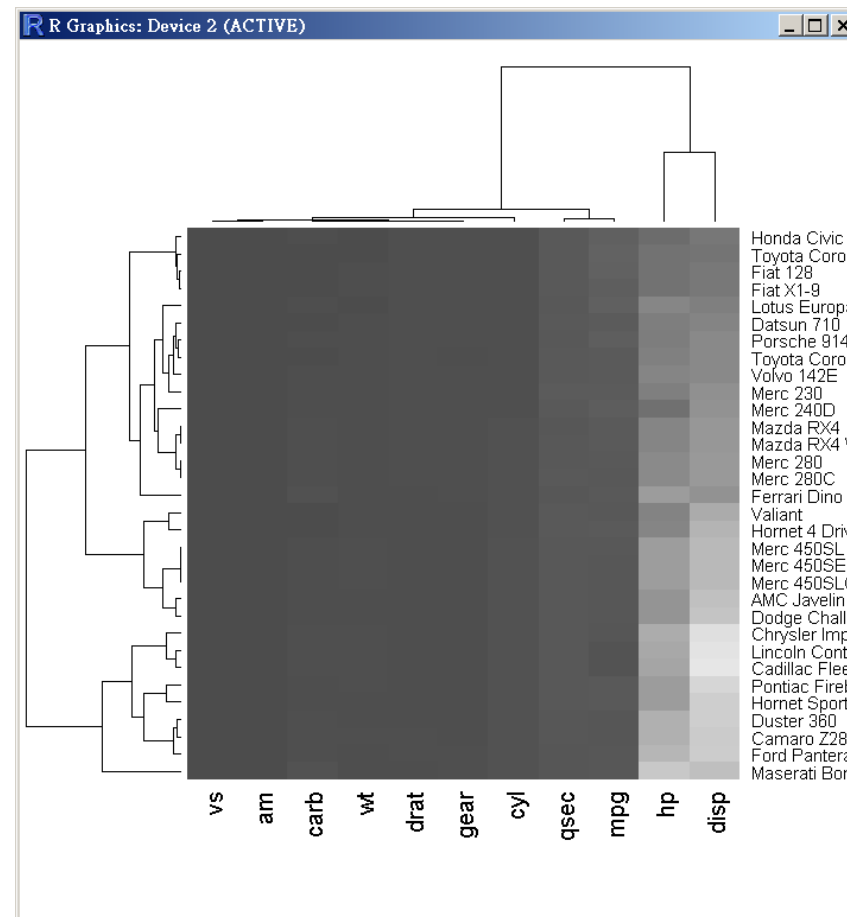
grid.arrange(..., newpage = TRUE)
```



hmap{seriation}

Plot heat map reordered by different algorithms

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

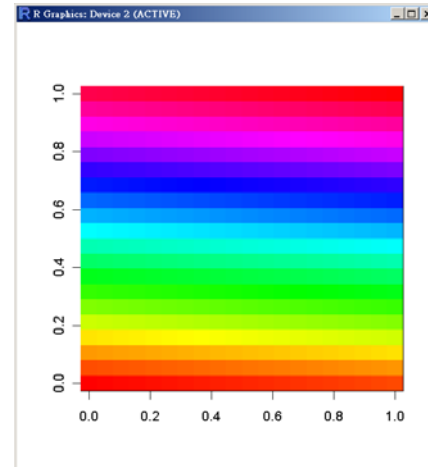
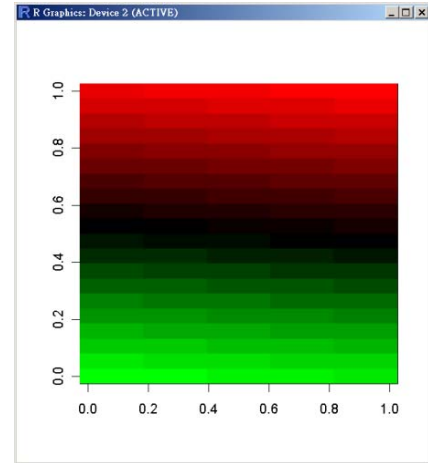




課堂練習: color.Palette

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

```
color.Palette <- function(low = "black",  
                          high = c("green", "red"),  
                          mid="black",  
                          k =50)  
{  
  low <- col2rgb(low)/255  
  high <- col2rgb(high)/255  
  
  if(is.null(mid)){  
    r <- seq(low[1], high[1], len = k)  
    g <- seq(low[2], high[2], len = k)  
    b <- seq(low[3], high[3], len = k)  
  }  
  if(!is.null(mid)){  
    k2 <- round(k/2)  
    mid <- col2rgb(mid)/255  
    r <- c(seq(low[1], mid[1], len = k2),  
           seq(mid[1], high[1], len = k2))  
    g <- c(seq(low[2], mid[2], len = k2),  
           seq(mid[2], high[2], len = k2))  
    b <- c(seq(low[3], mid[3], len = k2),  
           seq(mid[3], high[3], len = k2))  
  }  
  rgb(r, g, b)  
}
```

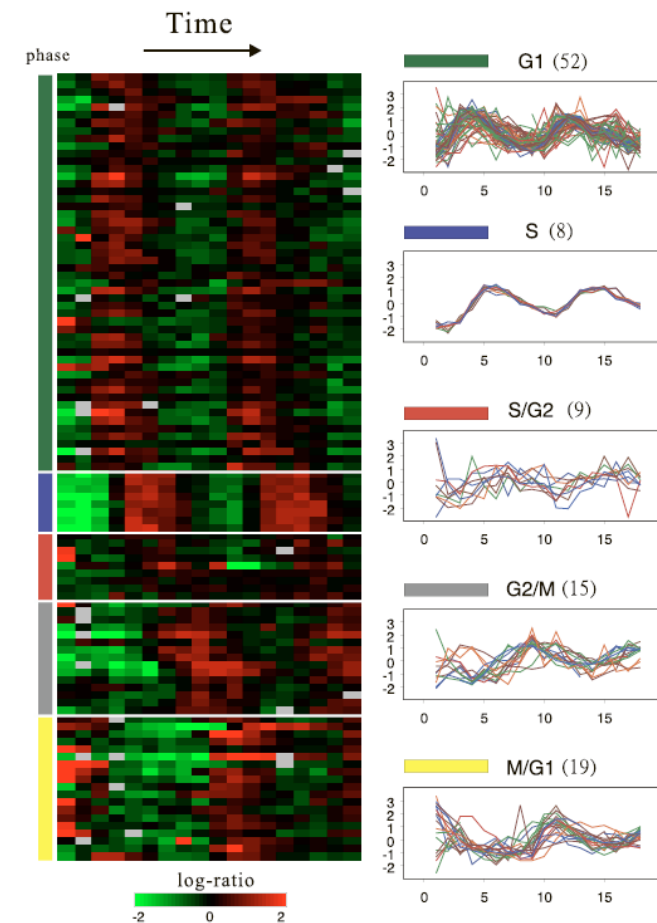
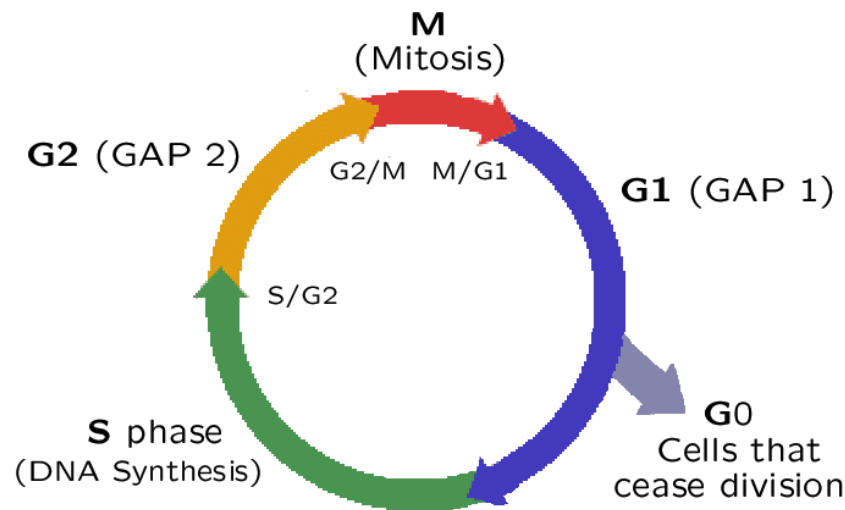




課堂練習: Microarray Data

166/216

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



See also: Using R to draw a Heatmap from Microarray Data

http://www2.warwick.ac.uk/fac/sci/moac/people/students/peter_cock/r/heatmap/



課堂練習: Microarray Data

167/216

```
# install.packages("fields")
library(fields)
gbr <- two.colors(start="green", middle="black", end="red")
cell.raw <- read.table("trad_alpha103.txt", row.names=1, header=T)
cell.data <- t(scale(t(cell.raw[,2:19]), center=T, scale=T))
n <- nrow(cell.data)
p <- ncol(cell.data)
gene.phase <- cell.raw[,1]
range(cell.data)
cell.data[cell.data > 2.802712] <- 2.802712
cellcycle.color <- c("darkgreen", "blue", "red", "gray50", "orange")
rc <- cellcycle.color[gene.phase+1]
cc <- rainbow(ncol(cell.data))

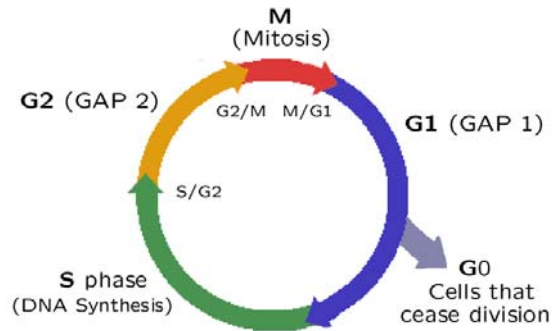
hv1 <- heatmap(cell.data[n:1,], col = gbr, Colv=NA, Rowv=NA,
              RowSideColors = rc,
              ColSideColors = cc, margins = c(5,10),
              xlab = "Times", ylab = "Genes",main = "Heatmap of Microarray Data")

hv2 <- heatmap(cell.data, col = gbr, Colv=NA, Rowv=NULL,
              RowSideColors = rc,
              ColSideColors = cc, margins = c(5,10),
              xlab = "Times", ylab = "Genes",main = "Heatmap of Microarray Data")

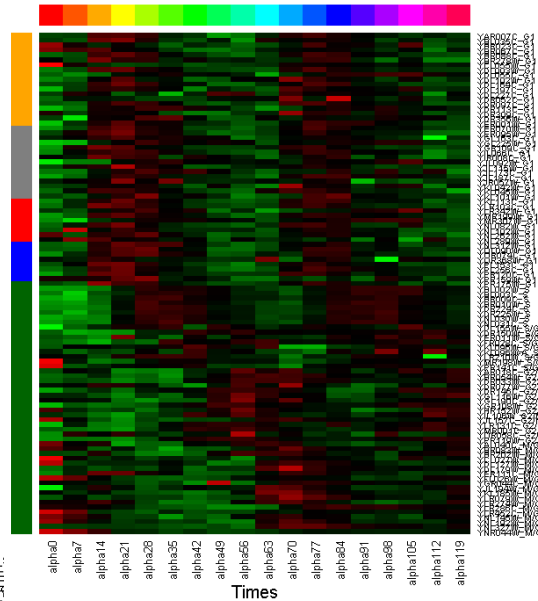
dd <- as.dendrogram(hclust(as.dist(1-cor(t(cell.data))))))
hv3 <- heatmap(cell.data, col = gbr, Colv=NA, Rowv=dd,
              RowSideColors = rc,
              ColSideColors = cc, margins = c(5,10),
              scale = "row",
              xlab = "Times", ylab = "Genes",main = "Heatmap of Microarray Data")
```




課堂練習: Microarray Data

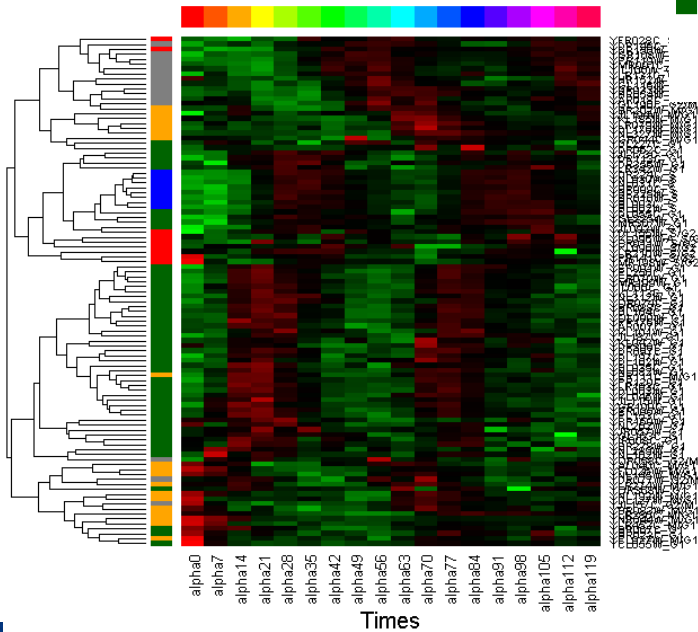


Heatmap of Microarray Data



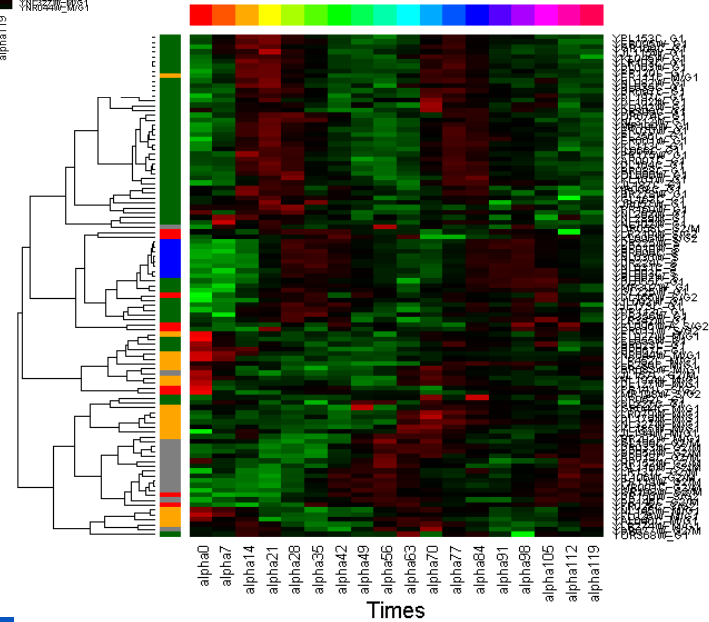
Genes

Heatmap of Microarray Data



Times

Heatmap of Microarray Data



Times

Heatmaps in R

■ Static

- `image {graphics}`
- `heatmap {stats}`
- `pheatmap {pheatmap} # pretty`
- `heatmap.2 {gplots} # Enhanced Heat Map`
- `aheatmap {NMF} # annotated heatmap`
- `heatmap3 {heatmap3}`
- `annHeatmap2 {Heatplus}, heatmap_2 {Heatplus}, heatmap_plus {Heatplus}`
- `d3heatmap`
- `Heatmap {ComplexHeatmap}`
- `plot_ly {plotly} # type = "heatmap"`
- `heatmap.plus {heatmap.plus}`
- Heat map produced by `xyplot()` function
- `corrplot {corrplot}`
- `levelplot {lattice}`

■ Interactive

- `heatmaply, fheatmap, gapmap, superheat`
- `shinyheatmap`: Ultra fast low memory heatmap web interface for big data genomics

■ Web Application

- A heatmap is created with the `geom_tile` geom from `ggplot`
- `Autoimage`



Heatmaps: Software-related Literature

- 2010, **neatmap** : non-clustering heat map alternatives in R
- 2011, **gitools**: analysis and visualisation of genomic data using interactive heat-maps
- 2014, advanced heat map and clustering analysis using **heatmap3**
- 2014, **hemi**: a toolkit for illustrating heatmaps
- 2014, **jheatmap** : an interactive heatmap viewer for the web
- 2015, an interactive cluster heat map to visualize and explore multidimensional metabolomic data
- 2015, **clustvis** : a web tool for visualizing clustering of multivariate data using principal component analysis and heatmap
- 2016, **complex heatmaps** reveal patterns and correlations in multidimensional genomic data
- 2017, **Autoimage** : multiple heat maps for projected coordinates
- 2017, **clustergrammer** : a web-based heatmap visualization and analysis tool for high-dimensional biological data
- 2017, **shinyheatmap** : ultra fast low memory heatmap web interface for big data genomics
- 2017, a galaxy implementation of next-generation clustered heatmaps for interactive exploration of molecular profiling data
- 2018, **heatmaply** : an R package for creating interactive cluster heatmaps for online publishing
- 2018, **superheat**: an R package for creating beautiful and extendable heatmaps for visualizing complex data

pheatmap: Pretty Heatmaps

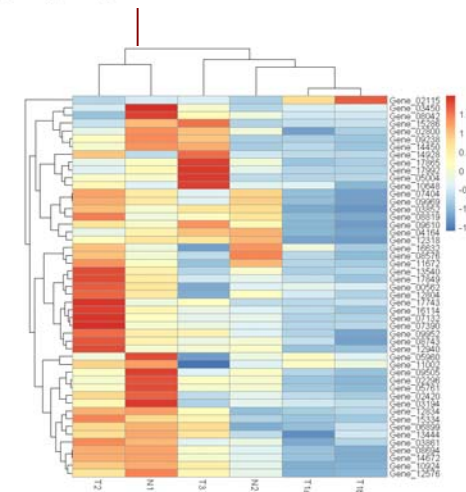
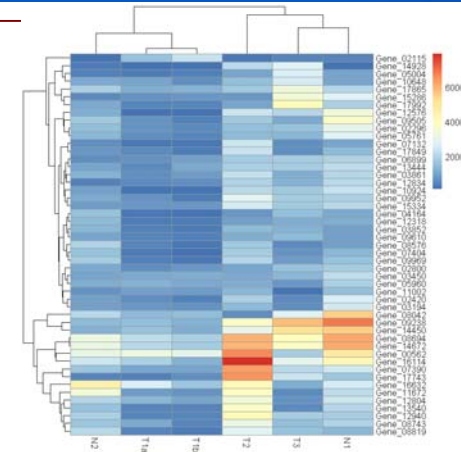
171/216

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

```
> 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

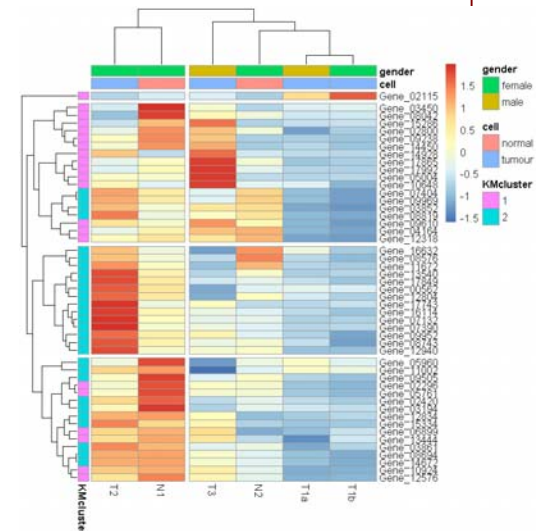
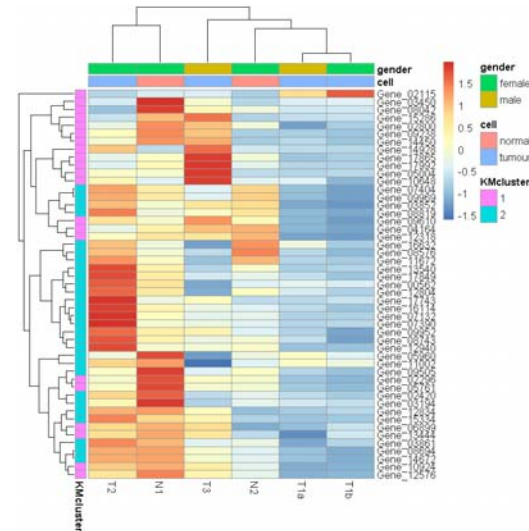
<https://dovetong.org/muse/2018/05/15/making-a-heatmap-in-r-with-the-pheatmap-package/>

<http://www.hmwu.idv.tw>



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)
```

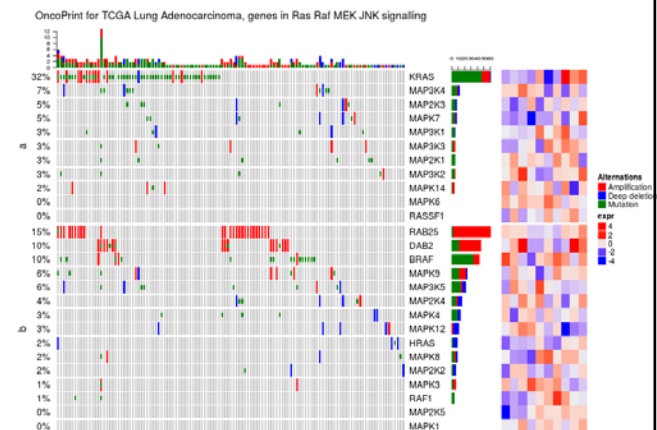
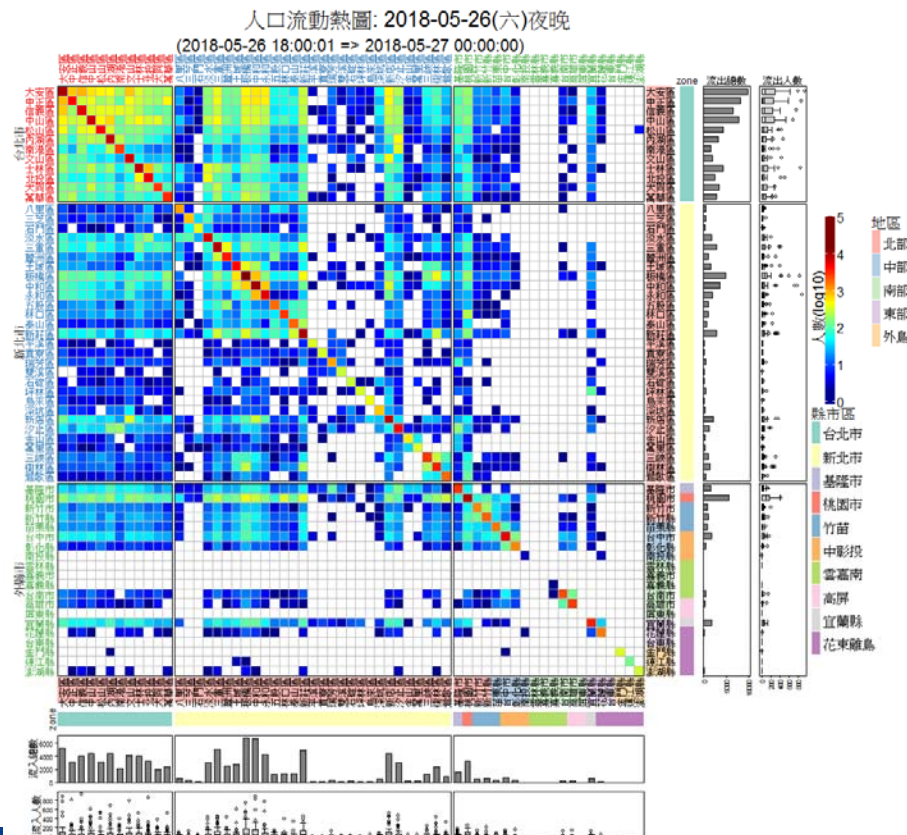


ComplexHeatmap

<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.

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
> BiocManager::install("ComplexHeatmap")
> library(ComplexHeatmap)
> Heatmap(exprs(selected.eset))
```

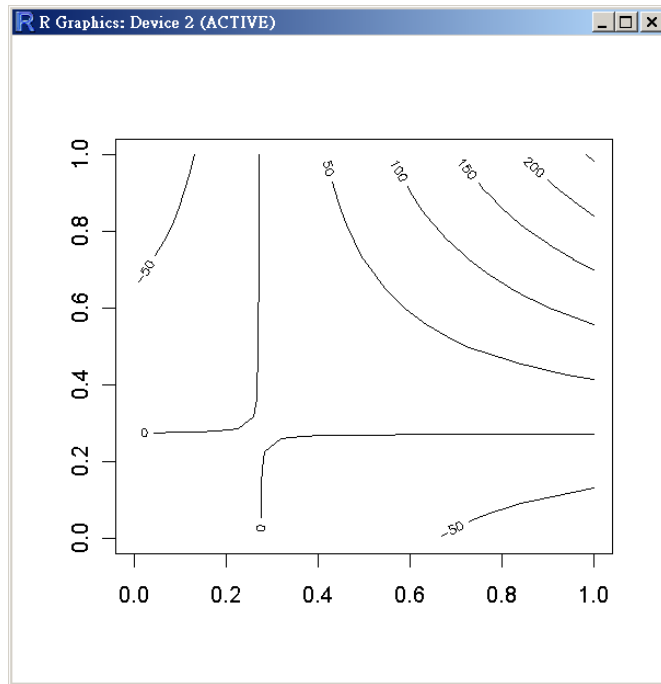


visualize multiple genomic alteration events by heatmap

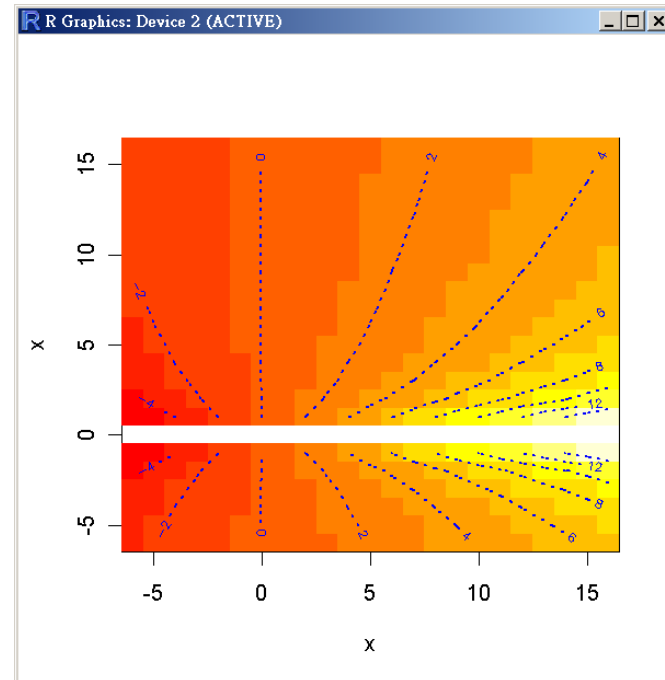


等高線圖 (Contours)

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

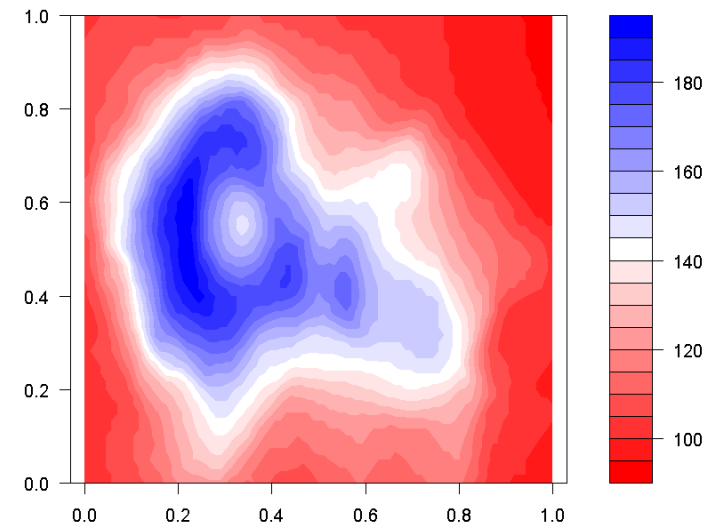
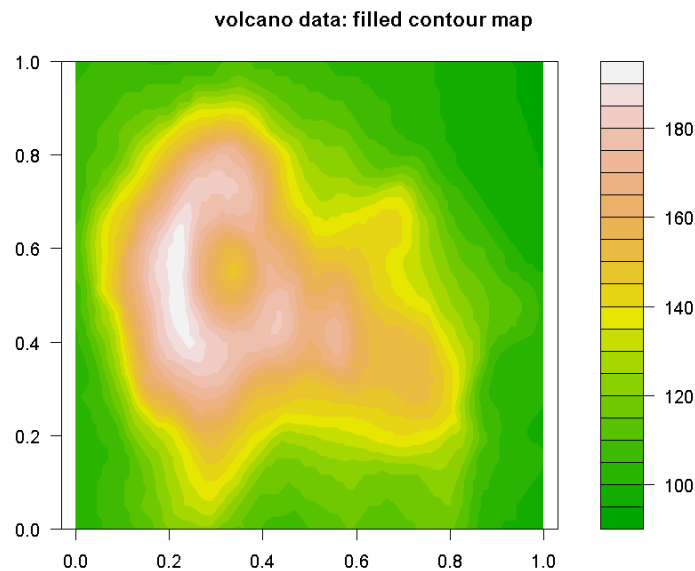


```
> image(x, x, z)
> contour(x, x, z, col = "blue", add = TRUE,
          method = "edge", lwd=1.5, lty=3)
```



Filled Contours

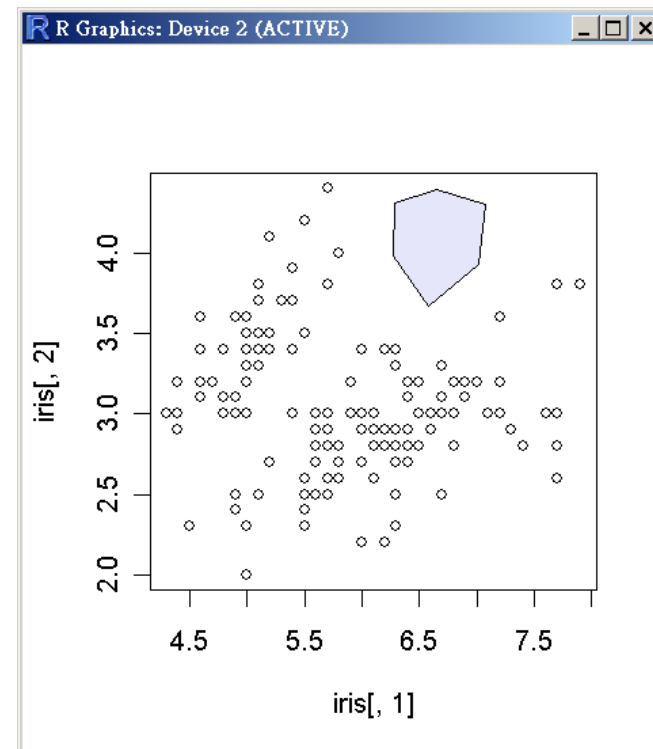
```
> filled.contour(volcano, color.palette = terrain.colors, asp = 1)
> title(main = "volcano data: filled contour map")
> # asp: the y/x aspect ratio.
> filled.contour(volcano, color.palette = colorRampPalette(c("red", "white",
"blue")), asp = 1)
```



Interactive Graphics

- > `par(ask=TRUE)` #the user is prompted before a new page of output is begin.
- > `locator()` #allow users to click within a plot and return the coordinates where the mouse click occurred.
- > `identify()` #use to add labels to data symbols on a plot. The data point closet to the mouse click gets labelled.

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

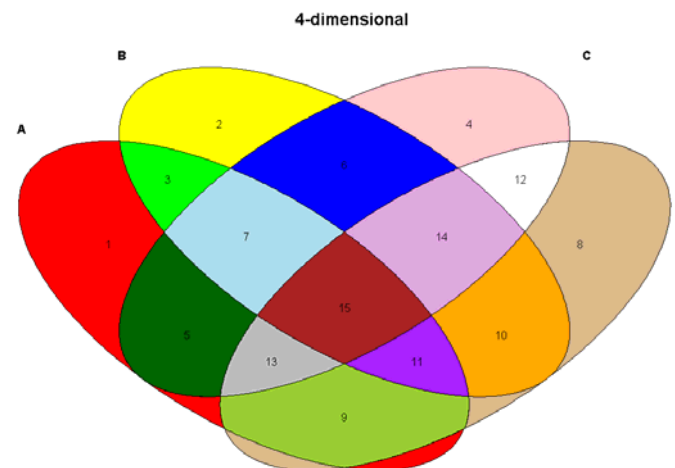
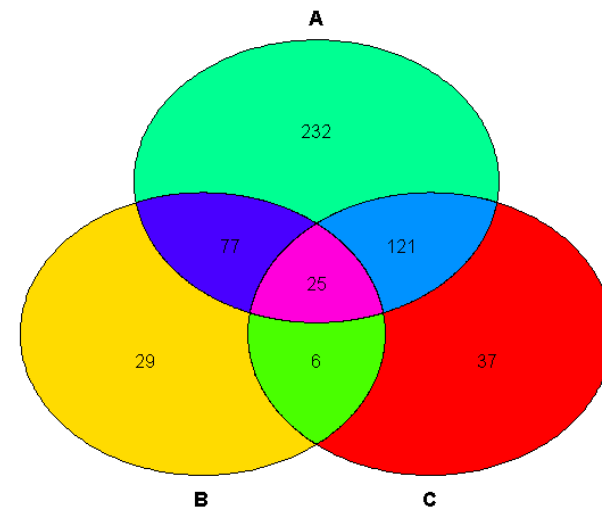
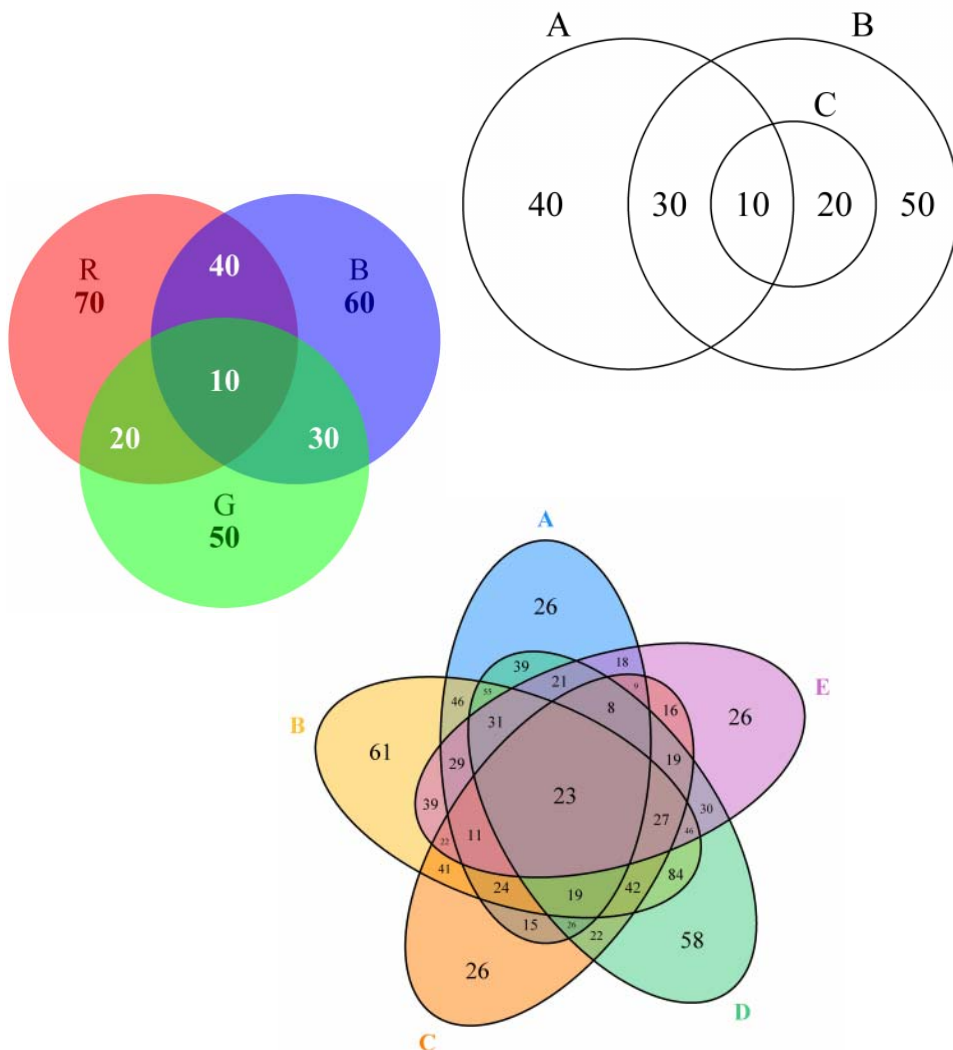




Venn Diagrams

Package **VennDiagram**

Package **colorfulVennPlot**





- **Lattice: Multivariate Data Visualization with R**
- Grid package
- R Graphics 2nd Edition, <https://www.stat.auckland.ac.nz/~paul/RG2>

Table 4.1
The plotting functions available in lattice

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

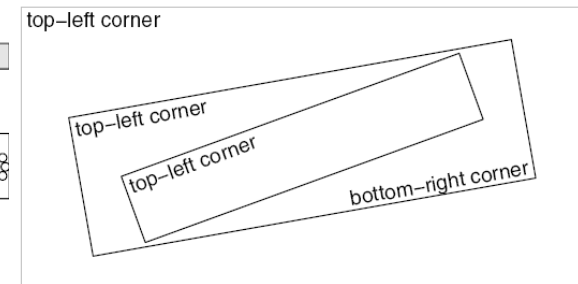
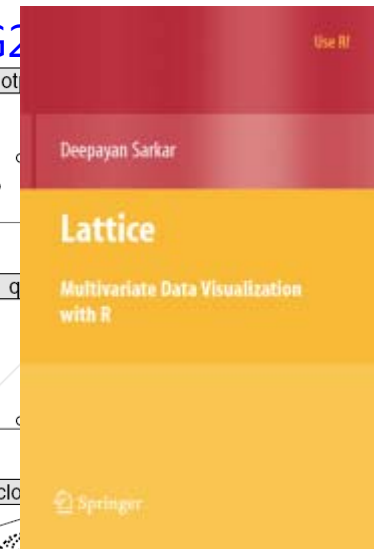
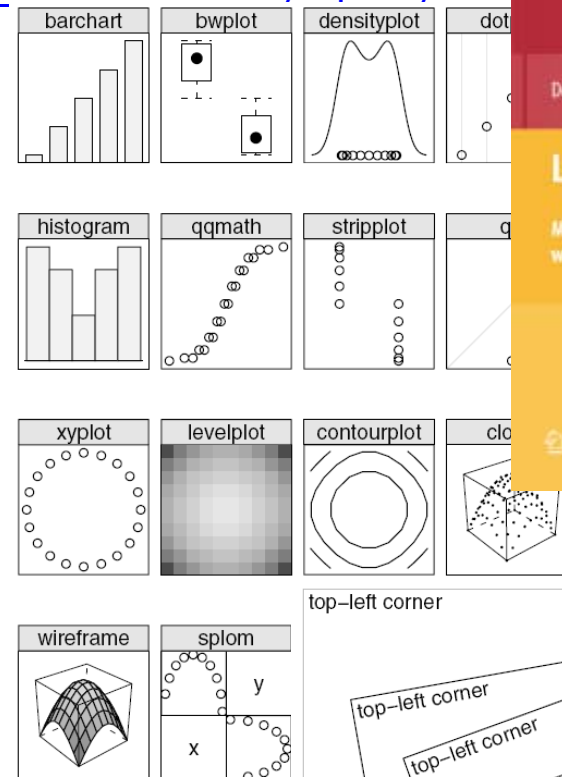


Figure 5.12
Popping a viewport. When a viewport is popped, the drawing context reverts to the parent viewport. In this figure, the second viewport (pushed in Figure 5.11) has been popped to go back to the first viewport (pushed in Figure 5.10). This time text has been drawn in the bottom-right corner.

ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics

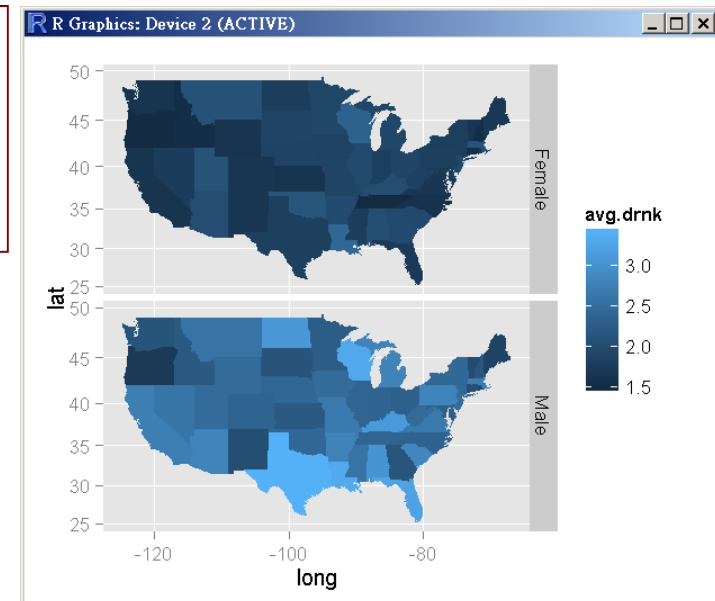
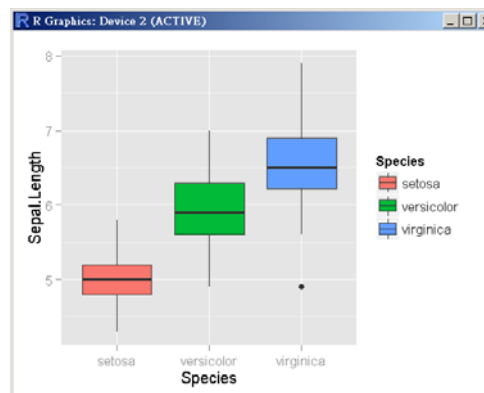
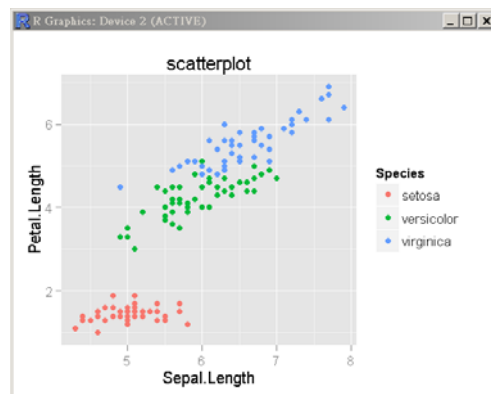
179/216

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

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



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



<http://www.youtube.com/watch?v=HeqHMM4ziXA>

<http://www.youtube.com/watch?v=n8kYa9vu1l8>

Data Visualization Cheat Sheet by RStudio

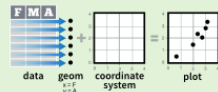
<https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf>

Data Visualization with ggplot2 Cheat Sheet

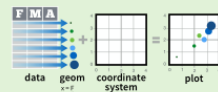


Basics

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



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



Complete the template below to build a graph.

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

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

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

last_plot()
Returns the last plot

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

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

Graphical Primitives

- a** <- ggplot(economics, aes(date, unemploy))
- b** <- ggplot(seals, aes(x = long, y = lat))
- a** + **geom_blank()**
(Useful for expanding limits)
- b** + **geom_curve**(aes(yend = lat + 1, xend = long + 1, curvature = z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size
- a** + **geom_path**(lineend = "butt", linejoin = "round", linemitre = 1)
x, y, alpha, color, group, linetype, size
- a** + **geom_polygon**(aes(group = group))
x, y, alpha, color, fill, group, linetype, size
- b** + **geom_rect**(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmin, xmax, ymin, ymax, alpha, color, fill, linetype, size
- a** + **geom_ribbon**(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

Line Segments

- common aesthetics: x, y, alpha, color, linetype, size
- b** + **geom_abline**(aes(intercept = 0, slope = 1))
- b** + **geom_hline**(aes(yintercept = lat))
- b** + **geom_vline**(aes(xintercept = long))
- b** + **geom_segment**(aes(yend = lat + 1, xend = long + 1))
- b** + **geom_spoke**(aes(angle = 1:1155, radius = 1))

One Variable

Continuous

- c** <- ggplot(mpg, aes(hwy)); **c2** <- ggplot(mpg)
- c** + **geom_area**(stat = "bin")
x, y, alpha, color, fill, linetype, size
- c** + **geom_density**(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight
- c** + **geom_dotplot**
x, y, alpha, color, fill
- c** + **geom_freqpoly**
x, y, alpha, color, group, linetype, size
- c** + **geom_histogram**(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
- c2** + **geom_qq**(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight

Discrete

- d** <- ggplot(mpg, aes(fl))
- d** + **geom_bar**
x, alpha, color, fill, linetype, size, weight

Two Variables

Continuous X, Continuous Y

- e** <- ggplot(mpg, aes(cty, hwy))
- e** + **geom_label**(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- e** + **geom_jitter**(height = 2, width = 2)
x, y, alpha, color, fill, shape, size
- e** + **geom_point**
x, y, alpha, color, fill, shape, size, stroke
- e** + **geom_quantile**
x, y, alpha, color, group, linetype, size, weight
- e** + **geom_rug**(sides = "b")
x, y, alpha, color, linetype, size
- e** + **geom_smooth**(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight
- e** + **geom_text**(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

- f** <- ggplot(mpg, aes(class, hwy))
- f** + **geom_col**
x, y, alpha, color, fill, group, linetype, size
- f** + **geom_boxplot**
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- f** + **geom_dotplot**(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill, group
- f** + **geom_violin**(scale = "area")
x, y, alpha, color, fill, group, linetype, size, weight

Discrete X, Discrete Y

- g** <- ggplot(diamonds, aes(cut, color))
- g** + **geom_count**
x, y, alpha, color, fill, shape, size, stroke

Continuous Bivariate Distribution

- h** <- ggplot(diamonds, aes(carat, price))
- h** + **geom_bin2d**(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight
- h** + **geom_density2d**
x, y, alpha, colour, group, linetype, size
- h** + **geom_hex**
x, y, alpha, colour, fill, size

Continuous Function

- i** <- ggplot(economics, aes(date, unemploy))
- i** + **geom_area**
x, y, alpha, color, fill, linetype, size
- i** + **geom_line**
x, y, alpha, color, group, linetype, size
- i** + **geom_step**(direction = "hv")
x, y, alpha, color, group, linetype, size

Visualizing error

- df** <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
- j** <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
- j** + **geom_crossbar**(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- j** + **geom_errorbar**
x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh**)
- j** + **geom_linerange**
x, ymin, ymax, alpha, color, group, linetype, size
- j** + **geom_pointrange**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

Maps

- data** <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests)))
- map** <- map_data("state")
- k** <- ggplot(data, aes(fill = murder))
- k** + **geom_map**(aes(map_id = state), map = map) + **expand_limits**(x = map\$long, y = map\$lat)
map_id, alpha, color, fill, linetype, size

Three Variables

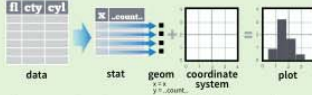
- sealsSz** <- with(seals, sqrt(delta_long^2 + delta_lat^2))
- l** <- ggplot(seals, aes(long, lat))
- l** + **geom_raster**(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill
- l** + **geom_tile**(aes(fill = z))
x, y, alpha, color, fill, linetype, size, width

Data Visualization Cheat Sheet by RStudio



Stats - An alternative way to build a layer

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



Visualize a stat by changing the default stat of a geom function, `geom_bar(stat="count")` or by using a stat function, `stat_count(geom="bar")`, which calls a default geom to make a layer (equivalent to a geom function).

Use `..name..` syntax to map stat variables to aesthetics.

geom to use | **stat function** | **geom mappings**
`stat_density2d(aes(fill = ..level..), geom = "polygon")`

1D distributions

- `c + stat_bin(binwidth = 1, origin = 10)`
- `x, y | ..count.., ..density..`
- `c + stat_count(width = 1) x, y, | ..count.., ..prop..`
- `c + stat_density(adjust = 1, kernel = "gaussian") x, y, | ..count.., ..density.., ..scaled..`

2D distributions

- `e + stat_bin_2d(bins = 30, drop = T)`
- `x, y, fill | ..count.., ..density..`
- `e + stat_bin_hex(bins=30) x, y, fill | ..count.., ..density..`
- `e + stat_density_2d(contour = TRUE, n = 100) x, y, color, size | ..level..`
- `e + stat_ellipse(level = 0.95, segments = 51, type = "t")`

3 Variables

- `l + stat_contour(aes(z = z)) x, y, z, order | ..level..`
- `l + stat_summary_hex(aes(z = z), bins = 30, fun = max) x, y, z, fill | ..value..`
- `l + stat_summary_2d(aes(z = z), bins = 30, fun = mean) x, y, z, fill | ..value..`

Comparisons

- `f + stat_boxplot(coef = 1.5) x, y | ..lower.., ..middle.., ..upper.., ..width.., ..ymin.., ..ymax..`
- `f + stat_ydensity(kernel = "gaussian", scale = "area") x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..`

Functions

- `e + stat_ecdf(n = 40) x, y | ..x.., ..y..`
- `e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") x, y | ..quantile..`
- `e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95) x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..`

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

- `e + stat_identity(na.rm = TRUE)`
- `ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) sample, x, y | ..sample.., ..theoretical..`
- `e + stat_sum() x, y, size | ..n.., ..prop..`
- `e + stat_summary(fun.data = "mean_cl_boot")`
- `h + stat_summary_bin(fun.y = "mean", geom = "bar")`
- `e + stat_unique()`

Scales

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

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

scale_... aesthetic to adjust | **prepackaged scale to use** | **scale specific arguments**

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

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

General Purpose scales

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

X and Y location scales

- Use with x or y aesthetics (x shown here)
- `scale_x_log10()` - Plot x on log10 scale
- `scale_x_reverse()` - Reverse direction of x axis
- `scale_x_sqrt()` - Plot x on square root scale

Color and fill scales (Discrete)

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

Color and fill scales (Continuous)

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

Shape and size scales

- `p <- e + geom_point(aes(shape = fl, size = cyl))`
- `p + scale_shape() + scale_size()`
- `p + scale_shape_manual(values = c(3:7))`
- `0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25`
`□ ○ △ × ◇ ▽ ☆ * ✱ ✲ ✳ ✴ ✵ ✶ ✷ ✸ ✹ ✺ ✻ ✼ ✽ ✾ ✿`
- `p + scale_radius(range = c(1,6))` Maps to radius of circle, or area
- `p + scale_size_area(max_size = 6)`

Coordinate Systems

- `r <- d + geom_bar()`
- `r + coord_cartesian(xlim = c(0, 5))`
`xlim, ylim`
The default cartesian coordinate system
- `r + coord_fixed(ratio = 1/2)`
`ratio, xlim, ylim`
Cartesian coordinates with fixed aspect ratio between x and y units
- `r + coord_flip()`
`xlim, ylim`
Flipped Cartesian coordinates
- `r + coord_polar(theta = "x", direction=1)`
`theta, start, direction`
Polar coordinates
- `r + coord_trans(ytrans = "sqrt")`
`xtrans, ytrans, limx, limy`
Transformed cartesian coordinates. Set `xtrans` and `ytrans` to the name of a window function.
- `tt + coord_quickmap()`
- `tt + coord_map(projection = "ortho", orientation=c(41, -74, 0))`
`projection, orientation, xlim, ylim`
Map projections from the `mapproj` package (mercator (default), `azequalarea`, `lagrange`, etc.)

Faceting

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

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

- `t + facet_grid(. ~ fl)`
facet into columns based on `fl`
- `t + facet_grid(year ~ .)`
facet into rows based on `year`
- `t + facet_grid(year ~ fl)`
facet into both rows and columns
- `t + facet_wrap(~ fl)`
wrap facets into a rectangular layout

Set `scales` to let axis limits vary across facets

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

Set `labeller` to adjust facet labels

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

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

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

α^c	α^d	α^e	α^p	α^r
------------	------------	------------	------------	------------

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

c	d	e	p	r
---	---	---	---	---

Labels

- `t + labs(x = "New x axis label", y = "New y axis label", title = "Add a title above the plot", subtitle = "Add a subtitle below title", caption = "Add a caption below plot", <AES> = "New <AES> legend title")`
- Use scale functions to update legend labels
- `t + annotate(geom = "text", x = 8, y = 9, label = "A")`
- geom to place | manual values for geom's aesthetics

Legends

- `n + theme(legend.position = "bottom")`
Place legend at "bottom", "top", "left", or "right"
- `n + guides(fill = "none")`
Set legend type for each aesthetic: colorbar, legend, or none (no legend)
- `n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"))`
Set legend title and labels with a scale function.

Zooming

- Without clipping (preferred)**
`t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))`
- With clipping (removes unseen data points)**
`t + xlim(0, 100) + ylim(10, 20)`
`t + scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 100))`

Position Adjustments

- Position adjustments determine how to arrange geoms that would otherwise occupy the same space.
- `s <- ggplot(mpg, aes(fl, fill = drv))`
- `s + geom_bar(position = "dodge")`
Arrange elements side by side
- `s + geom_bar(position = "fill")`
Stack elements on top of one another, normalize height
- `e + geom_point(position = "jitter")`
Add random noise to X and Y position of each element to avoid overplotting
- `e + geom_label(position = "nudge")`
Nudge labels away from points
- `s + geom_bar(position = "stack")`
Stack elements on top of one another
- Each position adjustment can be recast as a function with manual `width` and `height` arguments
- `s + geom_bar(position = position_dodge(width = 1))`

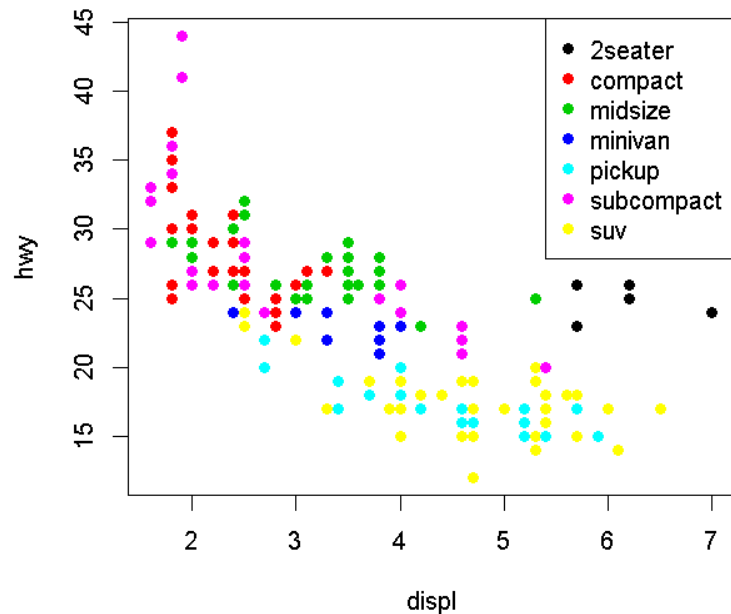
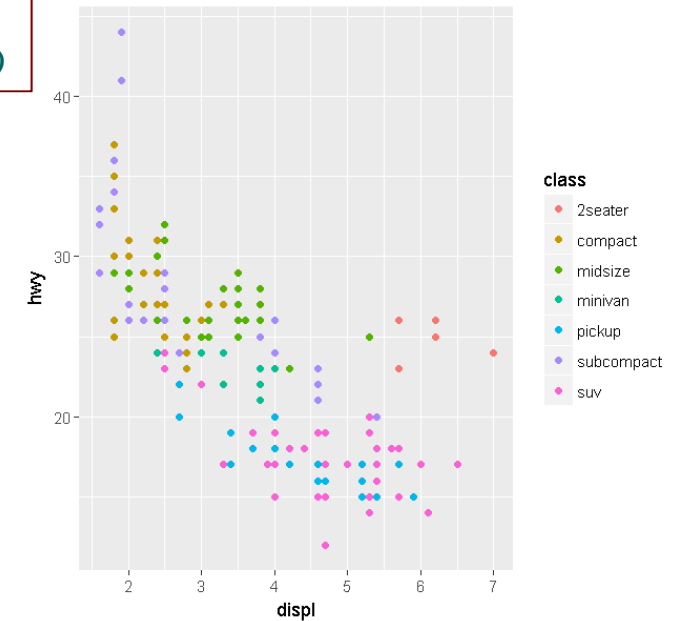
Themes

- `r + theme_bw()`
White background with grid lines
- `r + theme_classic()`
`r + theme_light()`
- `r + theme_gray()`
Grey background (default theme)
- `r + theme_minimal()`
Minimal themes
- `r + theme_dark()`
dark for contrast
- `r + theme_void()`
Empty theme

Base Graphics or ggplot2 ?

```
> ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

```
> mpg.df <- as.data.frame(mpg)
> attach(mpg.df)
> group <- as.factor(class)
> plot(displ, hwy, col=group, pch=16)
> legend("topright", legend=levels(group),
+   col=1:length(levels(group)), pch=16)
> detach(mpg.df)
```



10 reasons to switch to ggplot

<https://mandymejia.wordpress.com/2013/11/13/10-reasons-to-switch-to-ggplot-7/>

Comparing Base Graphics with ggplot2

<https://sakai.duke.edu/access/content/group/7a48cfac-b05c-4291-8e13-0091b5cd1479/Reference/BaseGraphicsGGPlotComparison.html>

Why I use ggplot2

<http://varianceexplained.org/r/why-i-use-ggplot2/>

Why I don't use ggplot2

<http://simplystatistics.org/2016/02/11/why-i-dont-use-ggplot2/>



R Base Graphics Cheatsheet

<http://publish.illinois.edu/johnrgallagher/files/2015/10/BaseGraphicsCheatsheet.pdf>

R Base Graphics Cheatsheet

SET GRAPHICAL PARAMETERS

the following can only be set with par()

par (...)

<i>multiple plots</i>	<code>mfc</code> = <code>c(nrow, ncol)</code>	<i>plot margins (outer)</i>	<code>oma</code> = <code>c(bottom, left, top, right)</code> default: <code>c(0, 0, 0, 0)</code> lines
<i>plot margins</i>	<code>mar</code> = <code>c(bottom, left, top, right)</code> default: <code>c(5.1, 4.1, 4.1, 2.1)</code> lines	<i>query x & y limits</i>	<code>par ("usr")</code>

CREATE A NEW PLOT

Bar charts	<code>barplot</code> (<i>height, ...</i>)	Histograms	<code>hist</code> (<i>x, ...</i>)
<i>bar labels</i>	<code>names.arg</code> =	<i>breakpts</i>	<code>breaks</code> =
<i>border</i>	<code>border</code> =		
<i>fill color</i>	<code>col</code> =	Line charts	<code>plot</code> (<i>x, type = "l"</i>)
<i>horizontal</i>	<code>horiz</code> = TRUE	<i>line type</i>	<code>lty</code> = "blank" 0 "solid" 1 "dashed" 2 "dotted" 3
Box plots	<code>boxplot</code> (<i>x, ...</i>)	<i>line width</i>	<code>lwd</code> =
<i>horizontal</i>	<code>horizontal</code> = TRUE		
<i>box labels</i>	<code>names</code> =	Scatterplots	<code>plot</code> (<i>x, ...</i>)
Dot plots	<code>dotchart</code> (<i>x, ...</i>)	<i>symbol</i>	<code>pch</code> =
<i>dot labels</i>	<code>labels</code> =		

REMOVE

<i>axis labels</i>	<code>ann</code> = FALSE
<i>axis, tickmarks, and labels</i>	<code>xaxt</code> = "n" <code>yaxt</code> = "n"
<i>plot box</i>	<code>bty</code> = "n"

NOTE: Many of the parameters here can be also be set in `par()`. See R help for more options.

ADJUST

<i>allow plotting out of plot region</i>	<code>xpd</code> = TRUE
<i>aspect ratio</i>	<code>asp</code> =
<i>axis limits</i>	<code>xlim</code> =, <code>ylim</code> =
<i>axis lines to match axis limits</i>	<code>xaxs</code> = "i", <code>yaxs</code> = "i" (internal axis calculation)

ADD TEXT

location	<code>xlab</code> =, <code>ylab</code> =	size	(magnification factor)
<i>axis labels</i>	<code>sub</code> =	<i>all elements</i>	<code>cex</code> =
<i>subtitle</i>	<code>main</code> =	<i>axis labels</i>	<code>cex.lab</code> =
<i>title</i>		<i>subtitle</i>	<code>cex.sub</code> =
		<i>tick mark labels</i>	<code>cex.axis</code> =
		<i>title</i>	<code>cex.main</code> =
style	<code>font</code> = 1 (plain) 2 (bold) 3 (italic) 4 (bold italic)	position	
<i>font face</i>		<i>text direction</i>	<code>las</code> = 1 (horizontal)
<i>font family</i>	<code>family</code> = "serif" "sans" "mono"	<i>justification</i>	<code>adj</code> = 0 .5 1 (left, center, right)

ADD TO AN EXISTING PLOT

Add new plot	[<i>any plot function</i>]	Lines	<code>lines</code> (<i>x, ...</i>)
	(..., <code>add</code> = TRUE)	<i>line style</i>	<code>lty</code> =
	ex. <code>barplot(x, add = TRUE)</code>	<i>line width</i>	<code>lwd</code> =
Axes	<code>axis</code> (<i>side, ...</i>)	<i>color</i>	<code>col</code> =
<i>location</i>	<code>side</code> = 1 2 3 4 (bottom, left, top, right)	Points	<code>points</code> (<i>x, ...</i>)
<i>tick mark:</i>		<i>symbol</i>	<code>pch</code> =
<i>labels</i>	<code>labels</code> =		
<i>location</i>	<code>at</code> =		
<i>remove</i>	<code>tick</code> = FALSE	<i>color</i>	<code>col</code> =
<i>rotate text</i>	<code>las</code> = 1 (horizontal)	<i>fill color</i>	<code>bg</code> = (pch: 21-25 only)
Axis labels	<code>mtext</code> (<i>text, ...</i>)	Text	<code>text</code> (<i>x, y, text, ...</i>)
<i>location</i>	<code>side</code> = 1 2 3 4 (bottom, left, top, right)	<i>position (rel. to x,y)</i>	<code>pos</code> = 1 2 3 4 (below, left, above, right) (default=center)
<i>lines to skip</i>	<code>line</code> = (from plot region, default=0)	Title	<code>title</code> (<i>main, ...</i>)
<i>position</i>	<code>at</code> = <i>x or y-coord</i> (depending on side)	<i>axis labels</i>	<code>xlab</code> =, <code>ylib</code> =
<i>justification</i>	<code>adj</code> = 0 .5 1 (left, center, right)	<i>subtitle</i>	<code>sub</code> =
		<i>title</i>	<code>main</code> =

Joyce Robbins, joycerobbins1@gmail.com



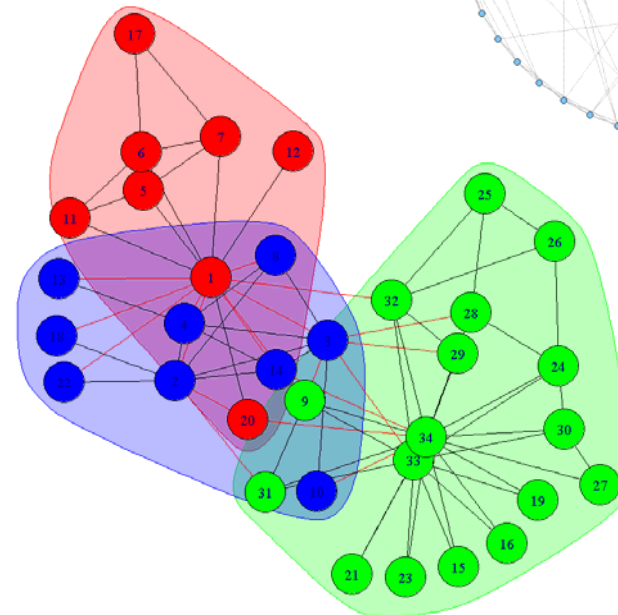
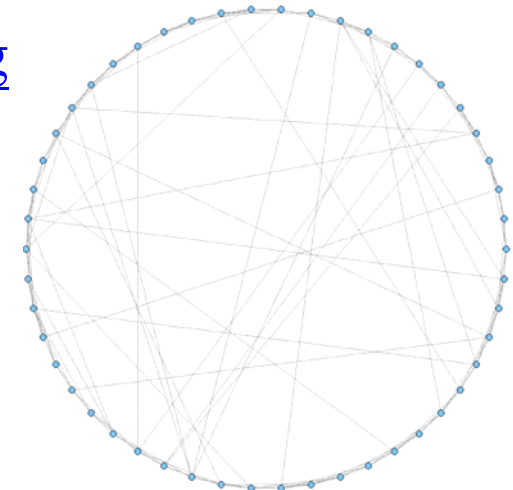
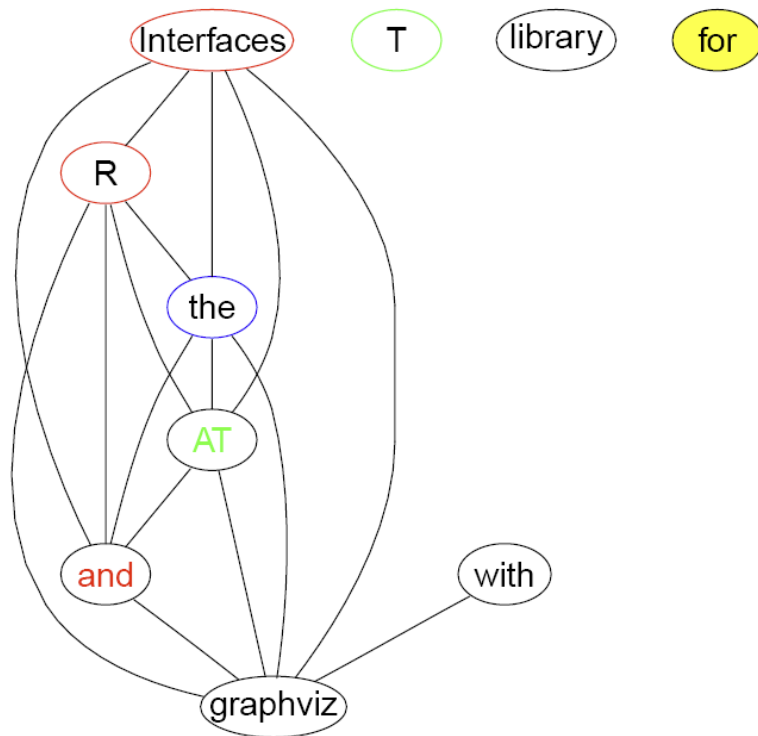
Networks: Rgraphviz, igraph

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

The network analysis package

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

<http://igraph.org>



Plotting on Google Static Maps in R: 185/216

RgoogleMaps

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



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

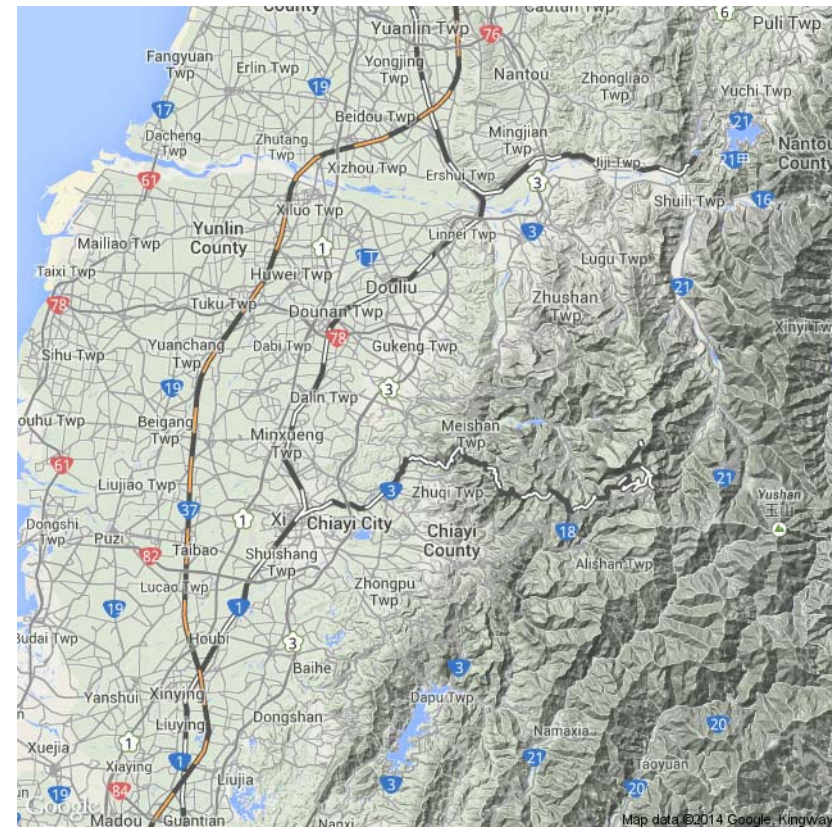


```
> library(ggmap)  
> twmap <- get_map(location = 'Taiwan', zoom = 7, language = "zh-TW")  
Error: Google now requires an API key.  
See ?register_google for details.
```




台灣地圖

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





於地圖上標記

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

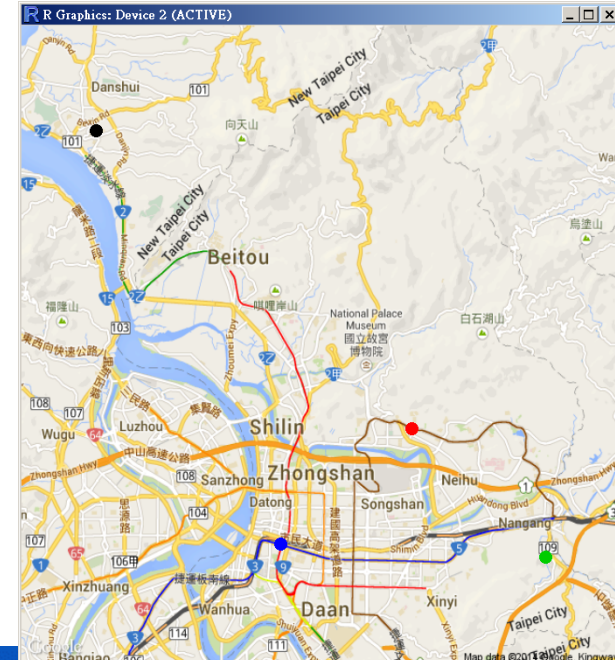
$lonR
[1] 121.4492 121.6154

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

查詢經緯度

http://card.url.com.tw/realads/map_latlng.php

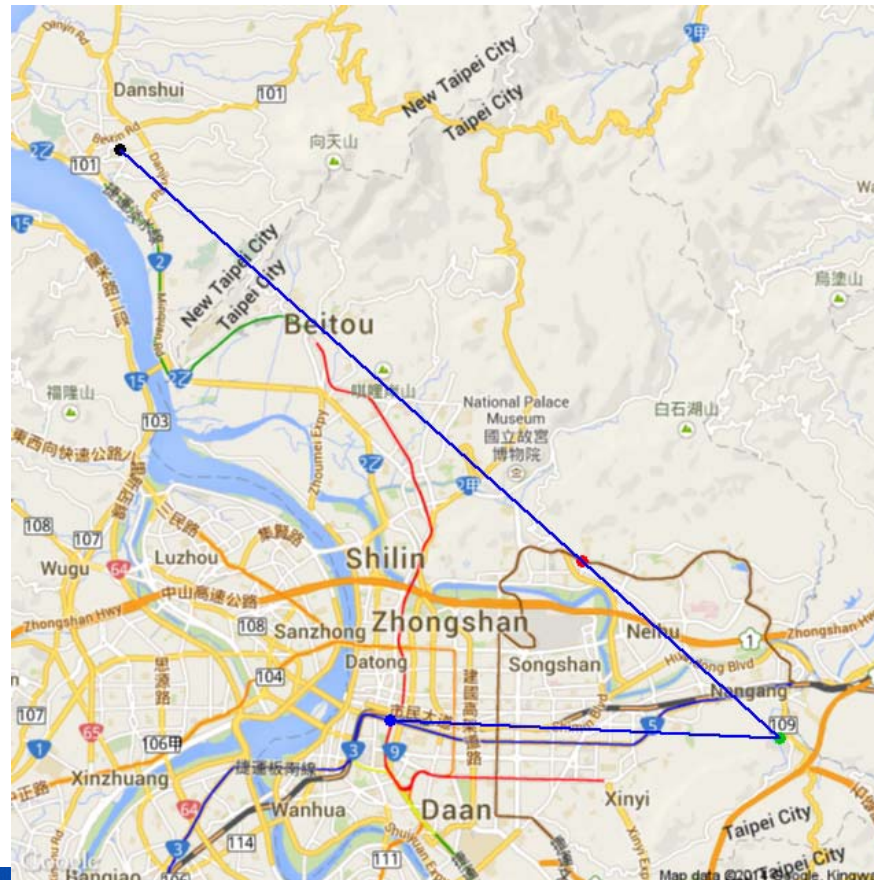
- 淡江大學 25.175339, 121.450003
- 台北市的地理中心位置: 內湖區環山路和內湖路一段
跟基湖路口: 25.082288, 121.565481
- 中研院 25.042185, 121.614548
- 捷運台北站: 25.046254, 121.517532





於地圖上標記

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

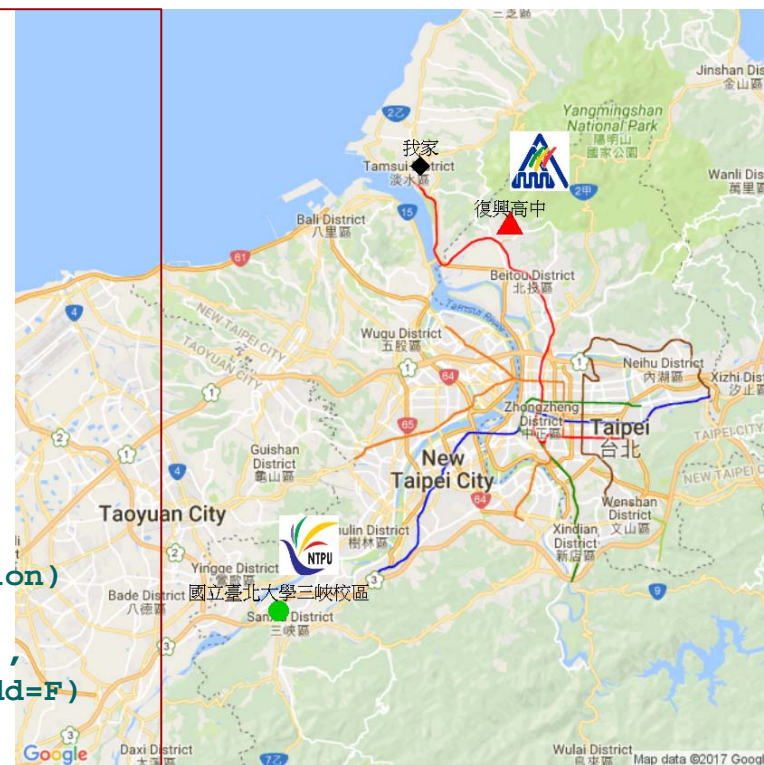


於地圖上加文字，加圖片

189/216

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

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



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

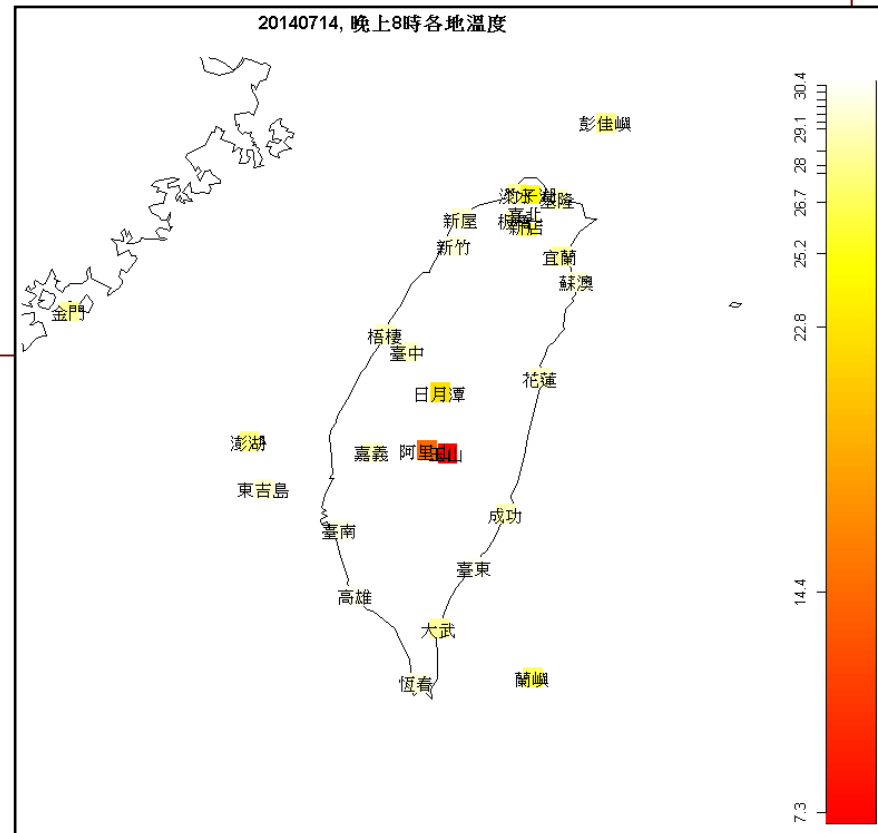


Package: `maps`, `mapdata`

190/216

氣象局開放資料平台 <http://opendata.cwb.gov.tw/>

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



See also:

Spatial data in R: Using R as a GIS

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

Creating a Map

```

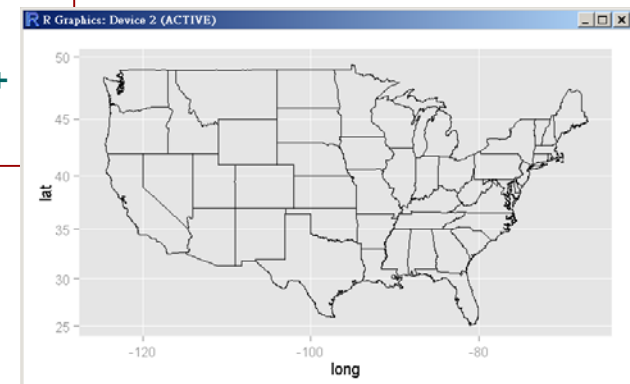
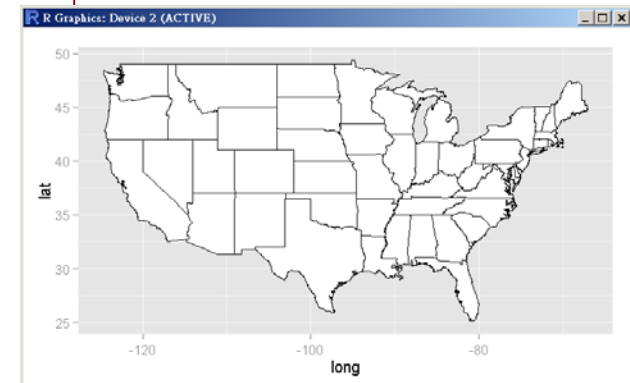
> library(ggplot2)
> library(maps)
> library(mapproj)
> states.map <- map_data("state")
> head(states.map, 3)
      long      lat group order  region subregion
1 -87.46201 30.38968     1     1  alabama    <NA>
2 -87.48493 30.37249     1     2  alabama    <NA>
3 -87.52503 30.37249     1     3  alabama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

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

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

```

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



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

Creating a Map

```

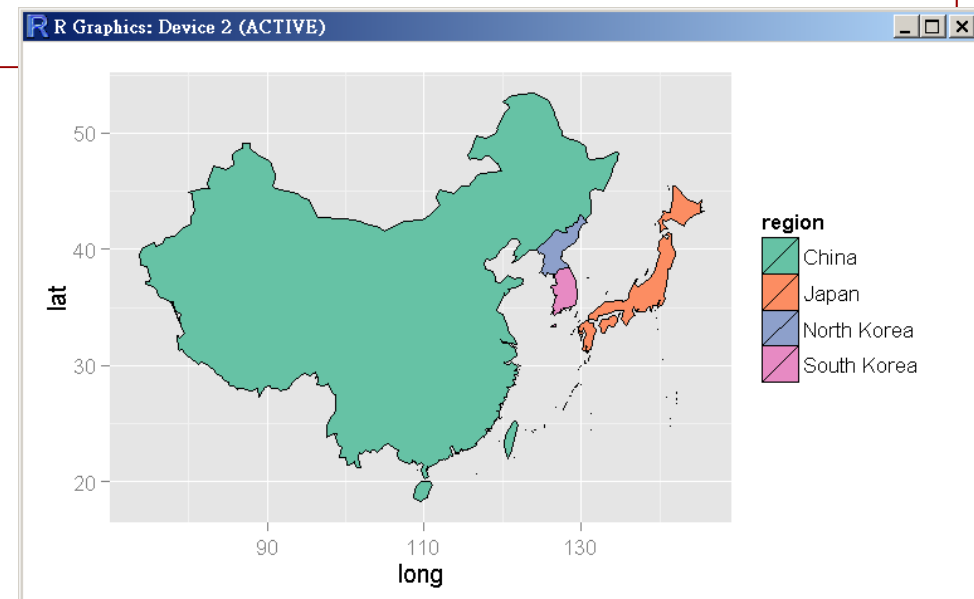
> # map_data: county, france, italy, nz, state, usa, world, world2.
> # region names
> world.map <- map_data("world")
> sort(unique(world.map$region))
[1] "Afghanistan"      "Albania"
[3] "Algeria"          "American Samoa"
[5] "Andaman Islands"  "Andorra"
...
> east.asia <- map_data("world", region=c("Japan",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

```

NOTE:

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

See Also: `mapproject`, `map`



分級著色圖 (Choropleth Maps)

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

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

```
      state Murder Assault UrbanPop Rape
Alabama  alabama  13.2    236     58 21.2
Alaska   alaska  10.0    263     48 44.5
Arizona  arizona   8.1    294     80 31.0
```

```
> library(maps); library(ggmap)
> states.map <- map_data("state")
> head(states.map, 3)
```

```
      long      lat group order  region subregion
1 -87.46201 30.38968     1     1  alabama    <NA>
2 -87.48493 30.37249     1     2  alabama    <NA>
3 -87.52503 30.37249     1     3  alabama    <NA>
```

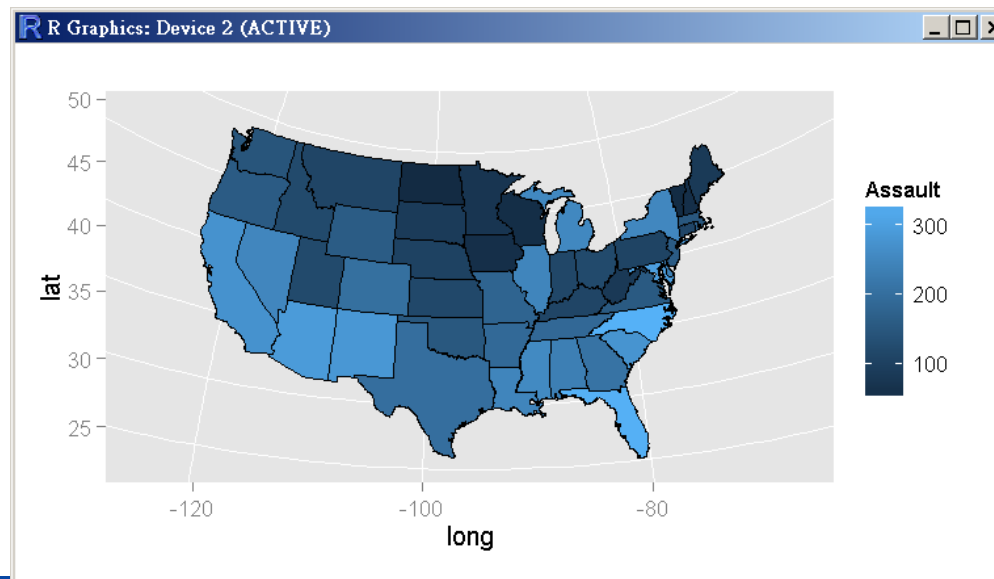
```
> crime.map <- merge(states.map, crimes, by.x="region", by.y="state")
> head(crime.map, 3)
```

```
      region      long      lat group order subregion Murder Assault UrbanPop Rape
1  alabama -87.46201 30.38968     1     1    <NA>   13.2    236     58 21.2
2  alabama -87.48493 30.37249     1     2    <NA>   13.2    236     58 21.2
3  alabama -87.95475 30.24644     1    13    <NA>   13.2    236     58 21.2
```



分級著色圖 (Choropleth Maps)

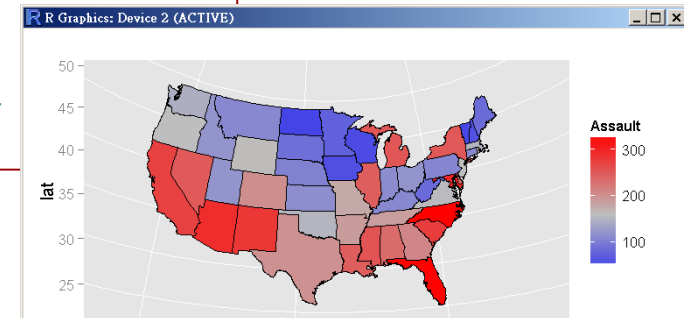
```
> # After merging, the order has changed, which would lead to polygons drawn in
> # the incorrect order. So, we sort the data.
> library(plyr) # For arrange() function
> # Sort by group, then order
> crime.map <- arrange(crime.map, group, order)
> head(crime.map, 3)
  region      long      lat group order subregion Murder Assault UrbanPop Rape
1 alabama -87.46201 30.38968     1     1      <NA>   13.2    236     58 21.2
2 alabama -87.48493 30.37249     1     2      <NA>   13.2    236     58 21.2
3 alabama -87.52503 30.37249     1     3      <NA>   13.2    236     58 21.2
> ggplot(crime.map, aes(x=long, y=lat, group=group, fill=Assault)) +
  geom_polygon(colour="black") +
  coord_map("polyconic")
```



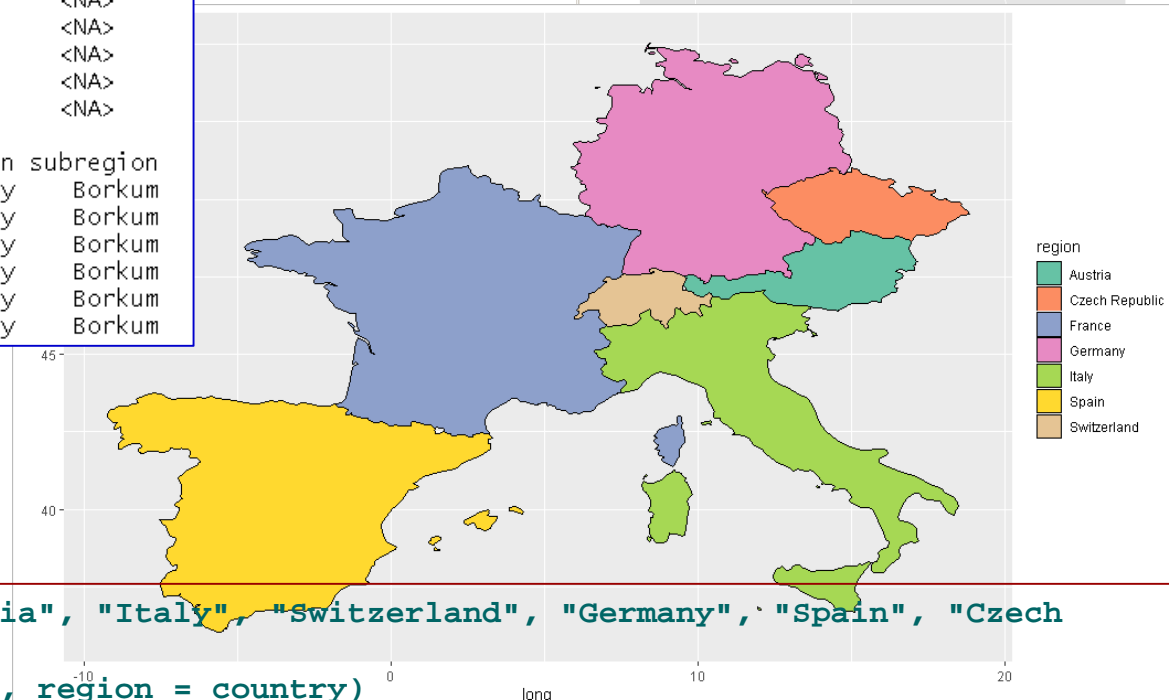
分級著色圖 (Choropleth Maps)

195/216

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



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



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

Package: googleVis

Interface between R and Google Charts

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

Chart Gallery - Google Charts

https://developers.google.com/chart/interactive/docs/gallery

Google Developers

Google Charts

Overview

Chart Gallery

Playground

Miscellaneous

Examples

Annotation Charts

Area Charts

Bar Charts

Bubble Charts

Calendar Charts

Candlestick Charts

Column Charts

Combo Charts

Diff Charts

Gauge Charts

Geo Charts

Histograms

Intervals

Line Charts

Maps

Org Charts

Pie Charts

Sankey Diagrams

Scatter Charts

Stepped Area Charts

Table Charts

Timelines

Tree Map Charts

Transitions

Chart Gallery

Our gallery provides a variety of charts designed to address your data visualization needs. These charts are based on pure HTML5/SVG technology (adopting VML for old IE versions), so no plugins are required. All of them are interactive, and many are pannable and zoomable. Adding these charts to your page can be done in a few simple steps.

Some additional community-contributed charts can be found on the Additional Charts page.

Geo Chart

Scatter Chart

Column Chart

Histogram

Bar Chart

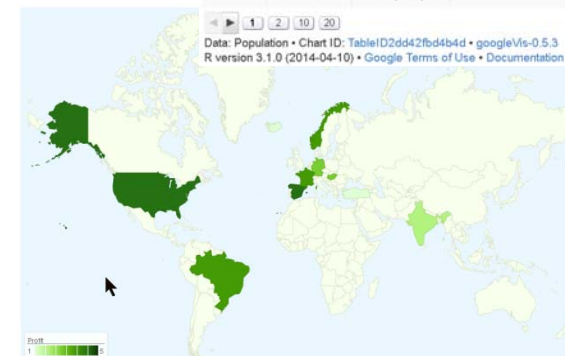
Combo Chart

Area Chart

Stepped Area Chart

Line Chart

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



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



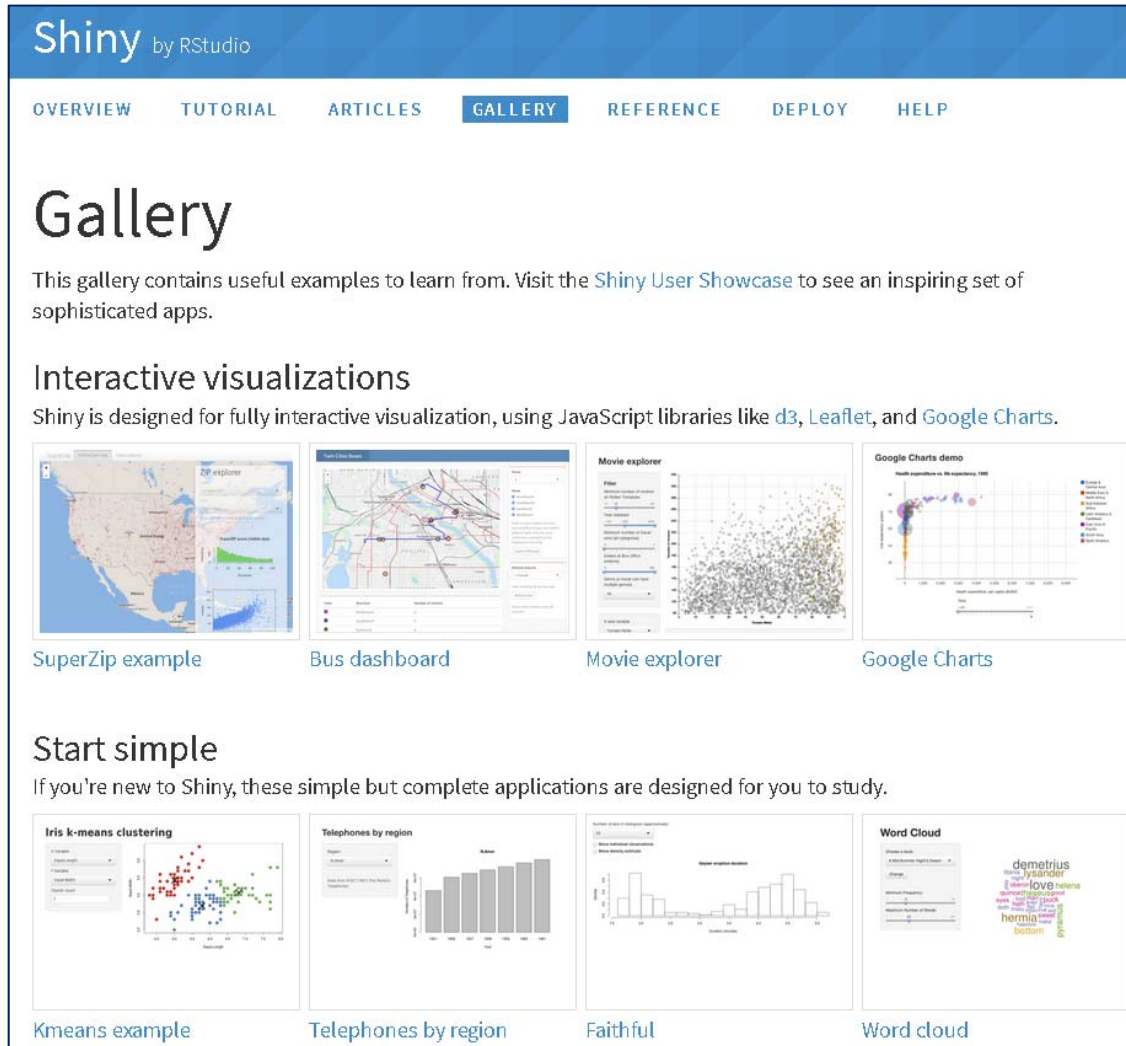
B. Bought pencils Bought 200k pencils 2009-1-5

A. Out of stock Ran out of stock of pens at 4pm 2009-1-4

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

<http://www.hmwu.idv.tw>

A web application framework for R



Shiny by RStudio

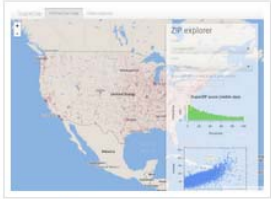
OVERVIEW TUTORIAL ARTICLES **GALLERY** REFERENCE DEPLOY HELP

Gallery

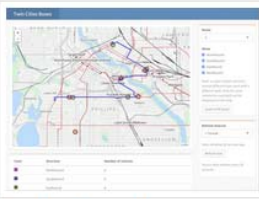
This gallery contains useful examples to learn from. Visit the [Shiny User Showcase](#) to see an inspiring set of sophisticated apps.

Interactive visualizations

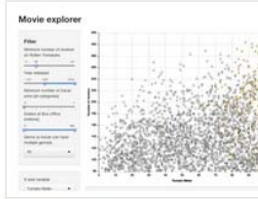
Shiny is designed for fully interactive visualization, using JavaScript libraries like [d3](#), [Leaflet](#), and [Google Charts](#).



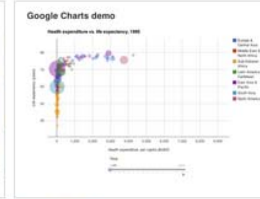
SuperZip example



Bus dashboard



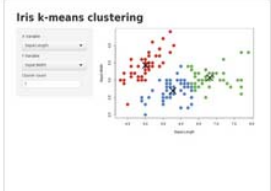
Movie explorer



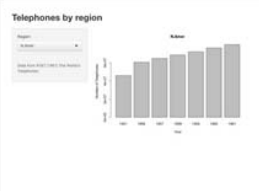
Google Charts

Start simple

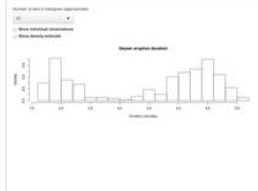
If you're new to Shiny, these simple but complete applications are designed for you to study.




Kmeans example



Telephones by region



Faithful



Word cloud

<http://shiny.rstudio.com/>

Turn analyses into interactive web applications

- Widgets: Sliders, File Download
- Application layout
- Dynamic user interface
- Interactive plots
- ...



Interactive Grammar of Graphics

ggvis 0.4 overview

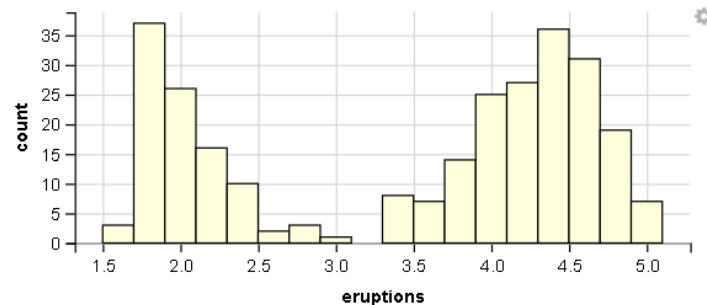
ggvis is a data visualization package for R which lets you:

- Declaratively describe data graphics with a syntax similar in spirit to ggplot2.
- Create rich interactive graphics that you can play with locally in Rstudio or in your browser.
- Leverage shiny's infrastructure to publish interactive graphics usable from any browser (either locally or remotely).

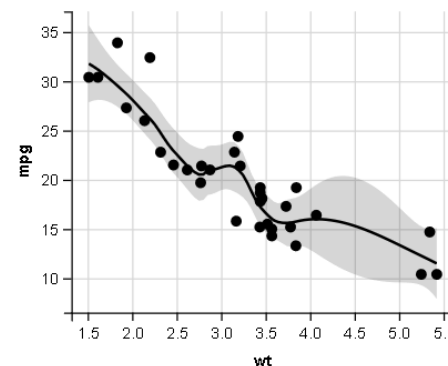
The goal is to combine the best of R (e.g. every modelling function you can imagine) and the best of JavaScript (e.g. every interaction and manipulation you can imagine). All data manipulation and transformation are done in R, and the graphics are rendered in a web browser. The graphics are displayed in a viewer panel, which is possible because RStudio is a web browser.

Examples of ggvis graphics

Histogram:



Scatterplot with smooth curve and interactive control:



Smoothing span



Visualizing Categorical Data

Visualizing Categorical Data - Windows Internet Explorer

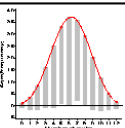
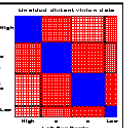
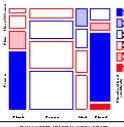
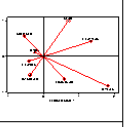
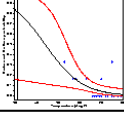
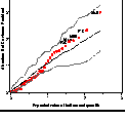
http://www.math.yorku.ca/SCS/vcd/

Visualizing Categorical Data

Visualizing Categorical Data

by **Michael Friendly**

SAS Institute (Dec, 2000), [Order code 56571](#), ISBN 1-58025-660-0

Visualizing		
	Categorical	
		Data

Contents

1. Introduction [[View Ch. 1 in .pdf \(264k\)](#)]
2. Fitting and graphing discrete distributions
3. Two-way contingency tables
4. Mosaic displays for n-way tables
5. Correspondence analysis
6. Logistic regression
7. Loglinear and logit models

A. SAS programs and macros
B. Data sets
C. Tables

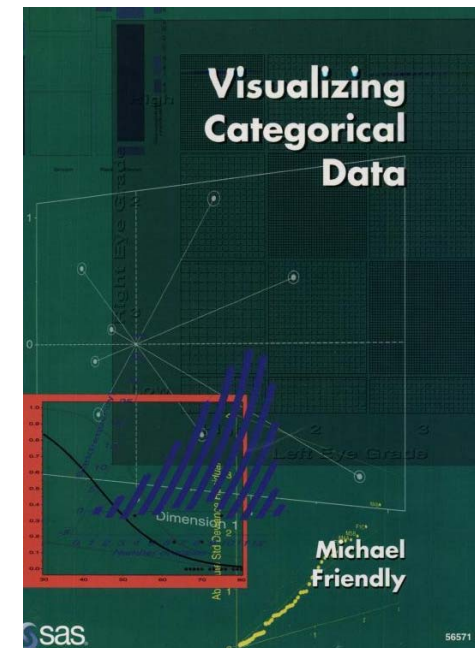
[Errata and Updates](#)

See also:

- Reviews of [Visualizing Categorical Data](#)
- [Reader's Guide to Visualizing Categorical Data](#) (SUGI 26 paper, PDF [137k])
- VCD Sampler: [Visualizing Categorical Data: Data, Stories, and Pictures](#) (SUGI 25 paper, PDF [177k])
- [VCD Source Online](#) [550k] (Source code for datasets and macros from the initial release, all in one big file, but no longer maintained)
- [VCD Archive](#) (VCD archive, vcdprog.zip, for purchasers of the book)
- [Mosaic displays web applet](#)
- [Mosaic displays User's Guide](#) (.pdf [318k])
- [Sieve diagrams web applet](#)
- [Gallery of Data Visualization](#)

Next

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



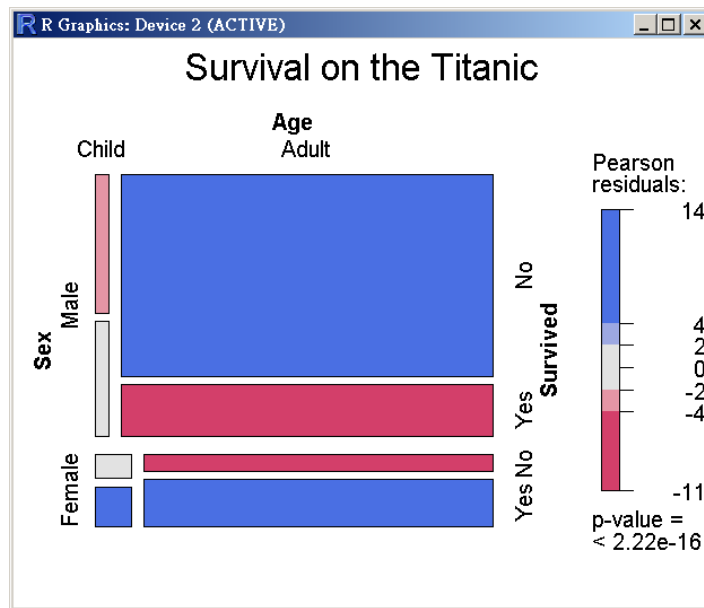
```
> library(vcd)
```

vcd: Visualizing Categorical Data

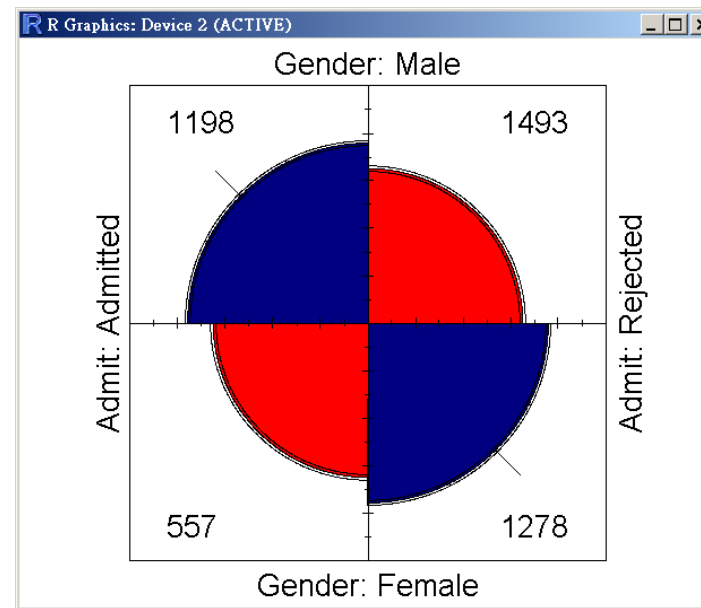
<http://cran.r-project.org/web/packages/vcd/index.html>

- **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/>

Mosaic Displays



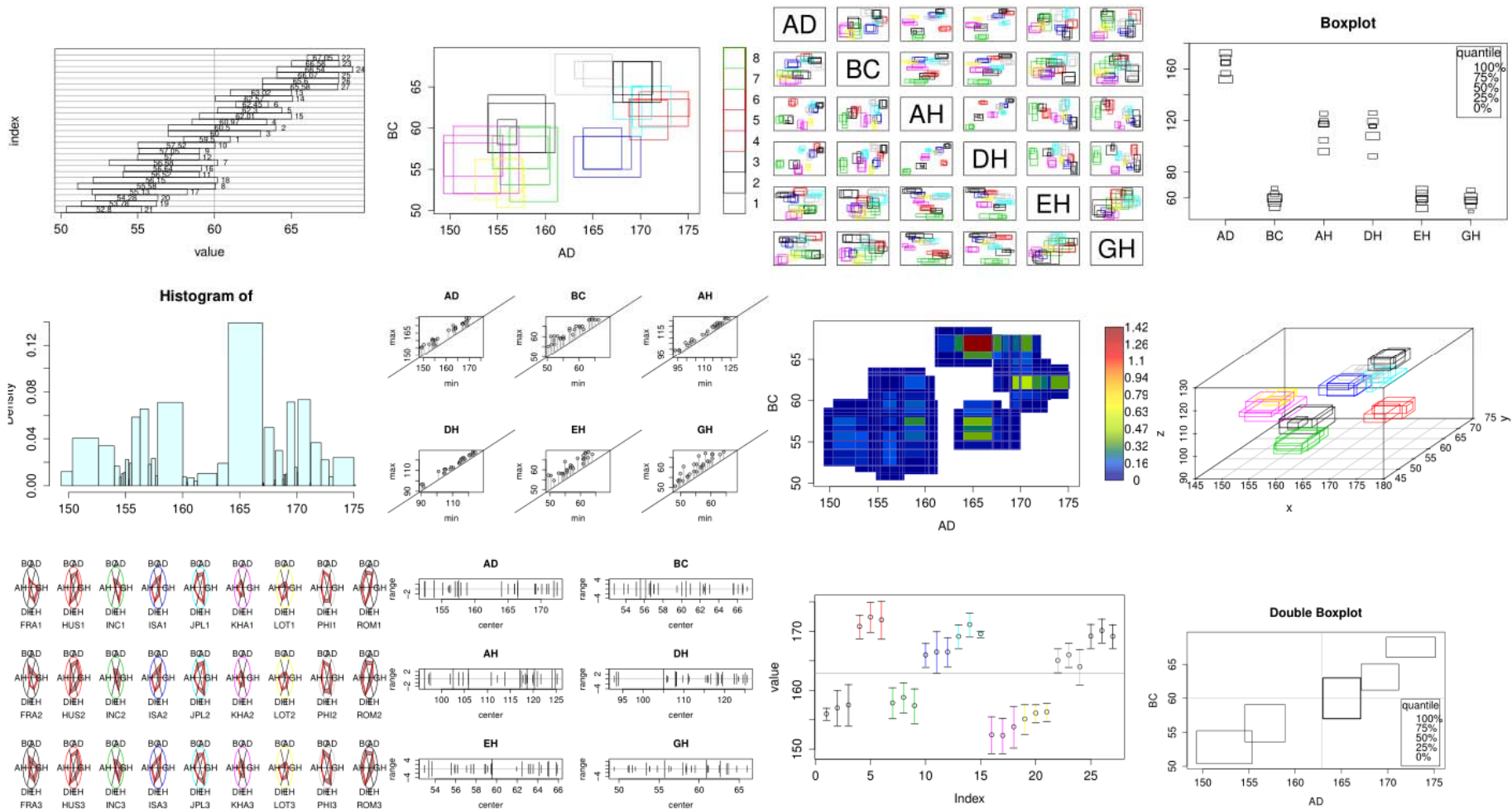
Fourfold Display





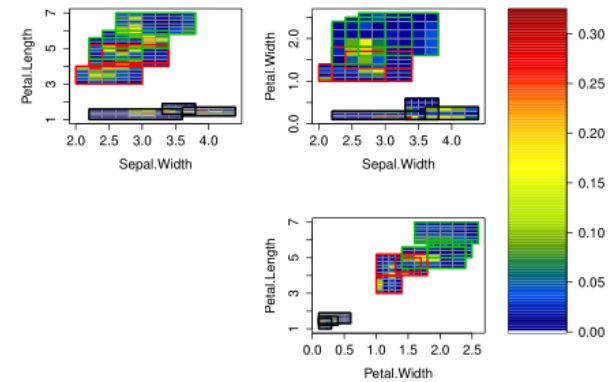
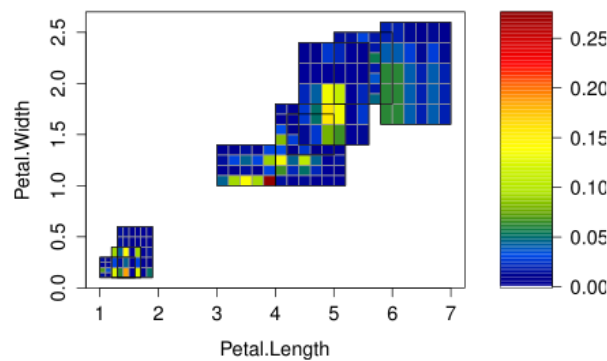
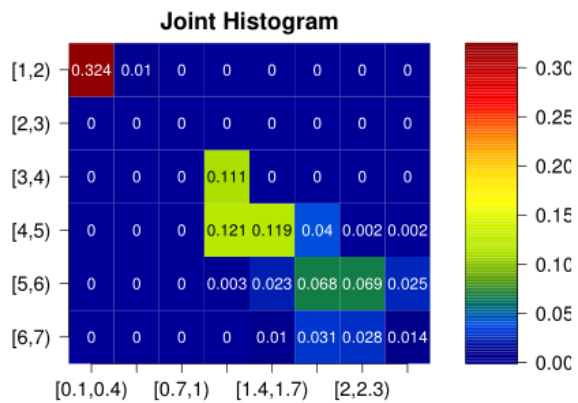
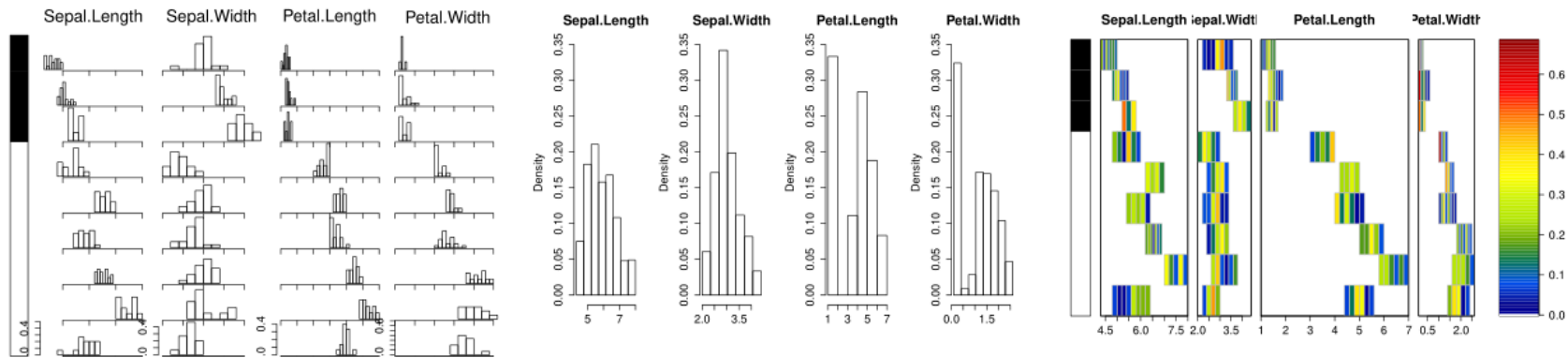
the R base graphics package companion for symbolic data analysis

```
plot(interval_object); hist(interval_object); ...
```

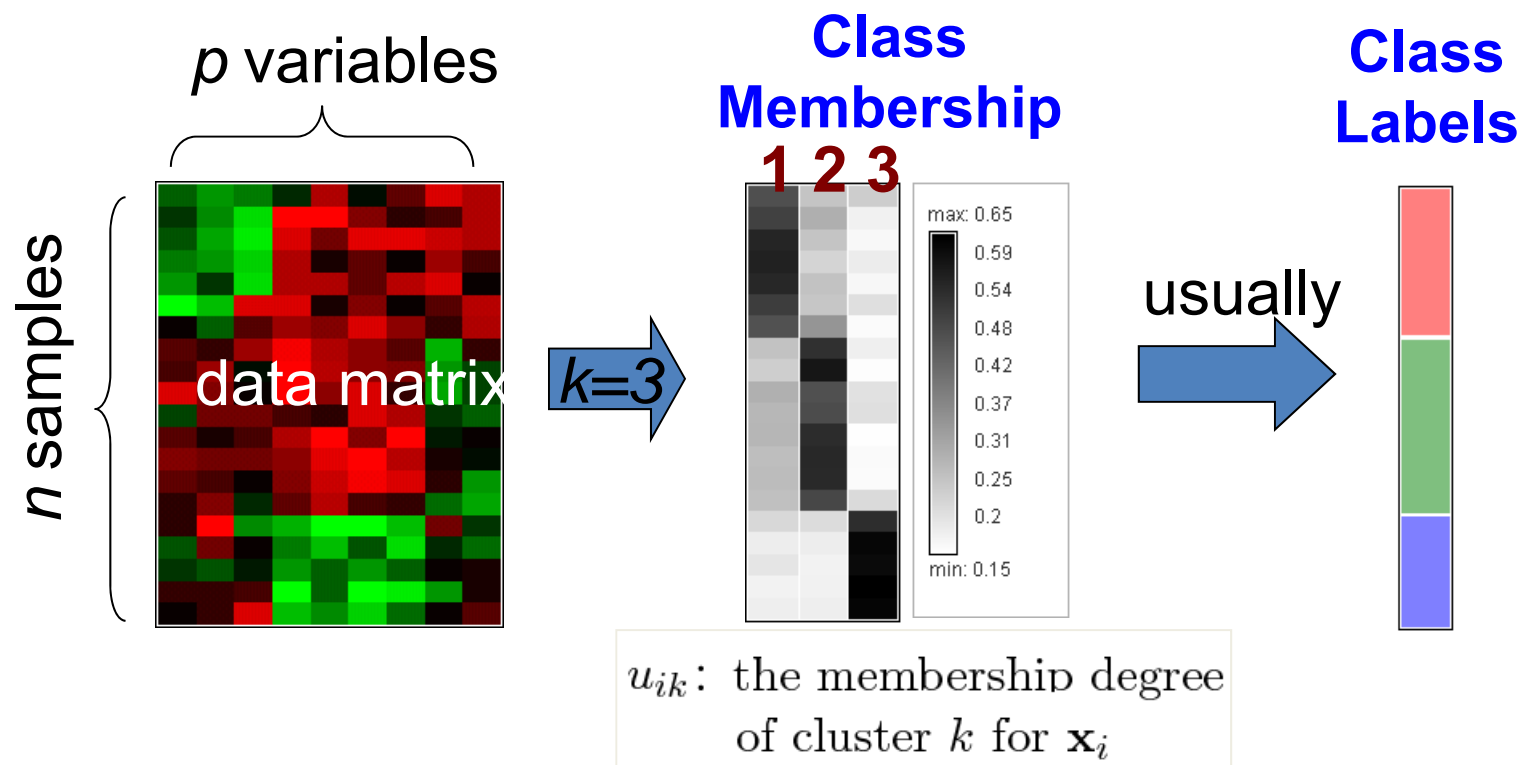




`plot(histogram_object); hist(histogram_object); ...`



Visualization of Fuzzy (Soft) Clustering Results



- **Fuzzy Clustering methods:**
Fuzzy c-means, model-based clustering, ensemble clustering,...
- **Visualization methods:**





Visualization Method



PCA



$\text{Proj}\mathbf{x}_i$

FCM



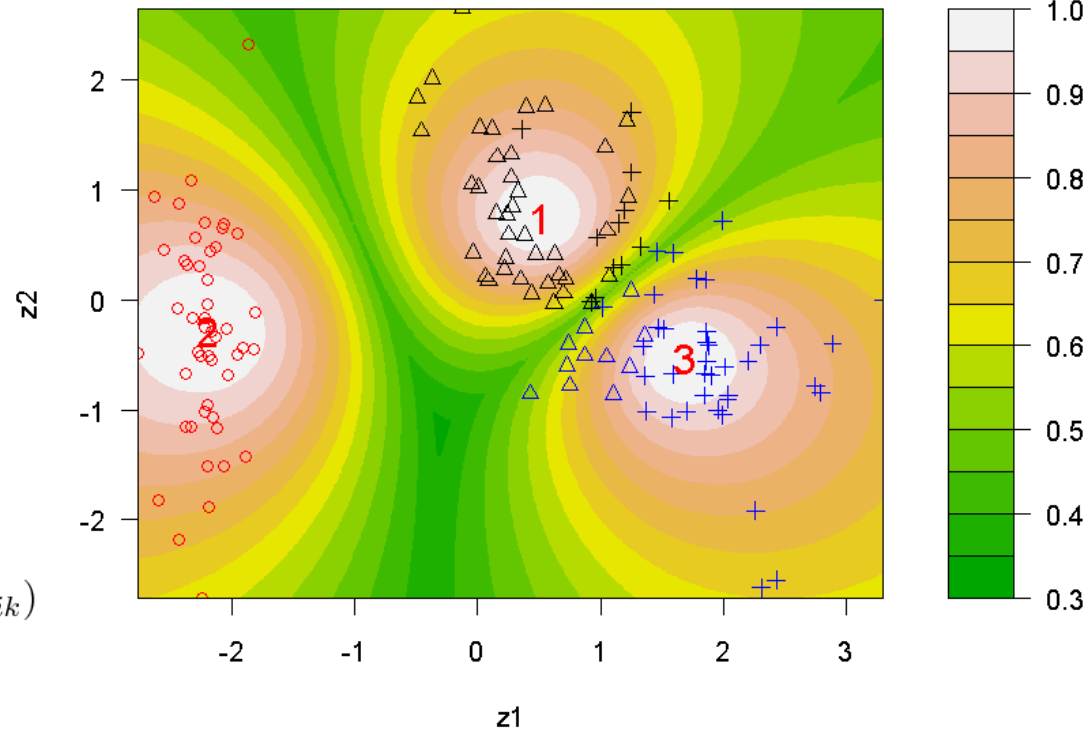
u_{ik}

$$\text{Proj}\mathbf{c}_k = \frac{\sum_{i=1}^n u_{ik}^m \cdot \text{Proj}\mathbf{x}_i}{\sum_{i=1}^n u_{ik}^m}$$

$$\tilde{y}_i = \arg \max_k (u_{ik})$$

$$\tilde{u}_{ik} = \left(\sum_{j=1}^K \left[\frac{d(\mathbf{g}_i, \text{Proj}\mathbf{c}_k)}{d(\mathbf{g}_i, \text{Proj}\mathbf{c}_j)} \right]^{\frac{2}{m-1}} \right)^{-1}$$

PCA+FCM

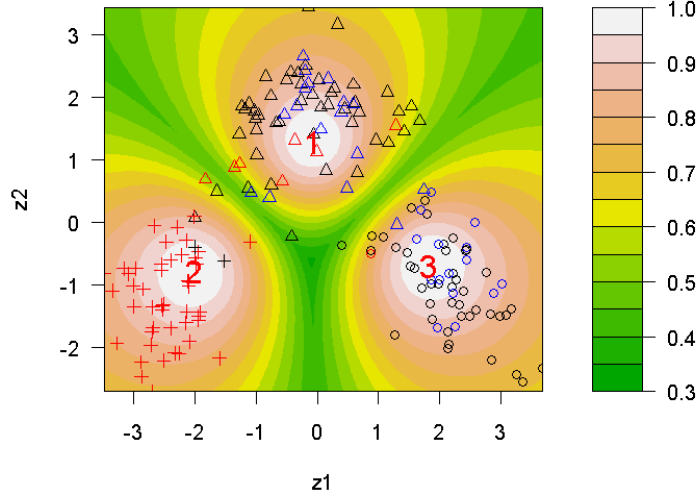




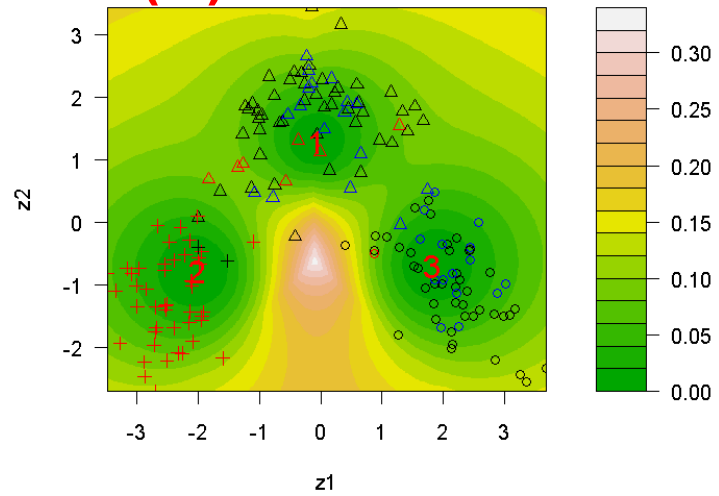
Example: Wine Data

(178x13: K=3)

max(U) FSIR+FCM

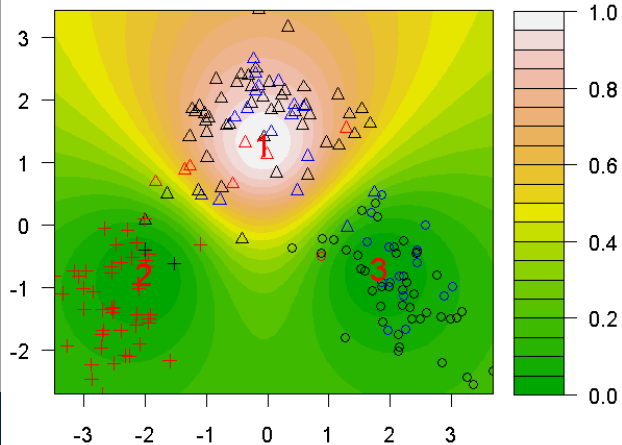


min(U) FSIR+FCM



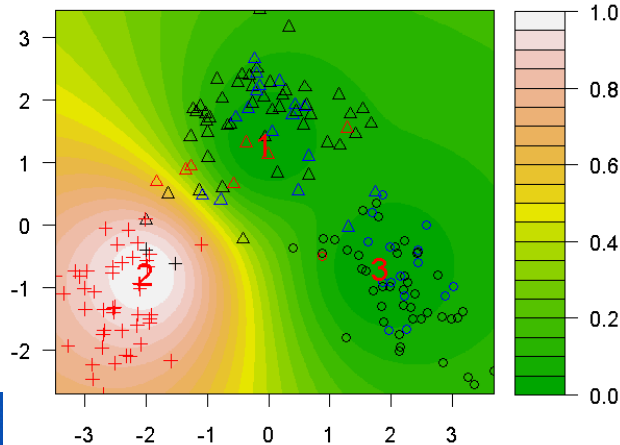
U[,1]

FSIR+FCM



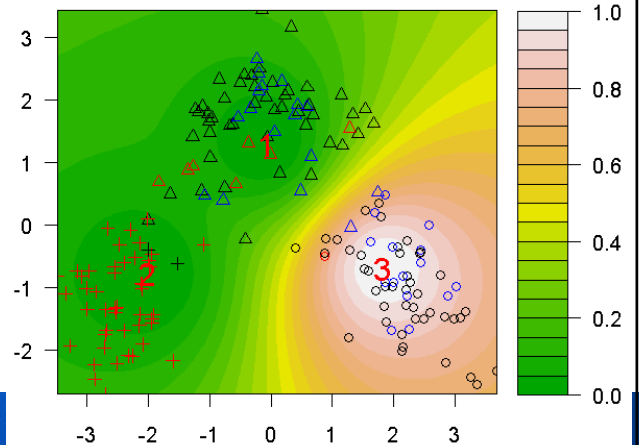
U[,2]

FSIR+FCM



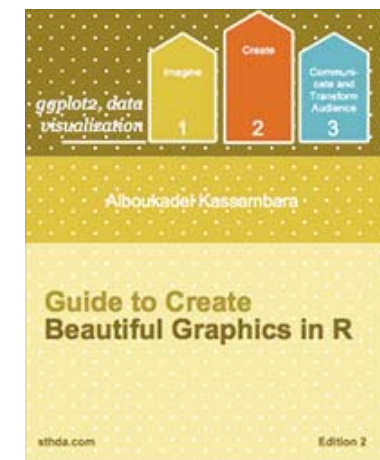
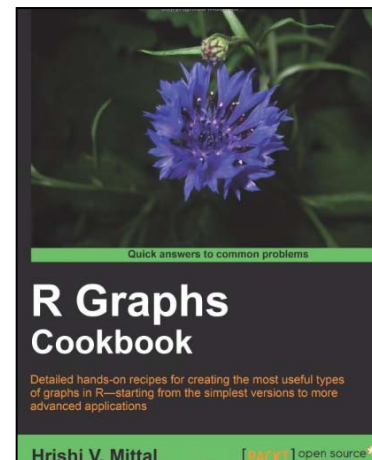
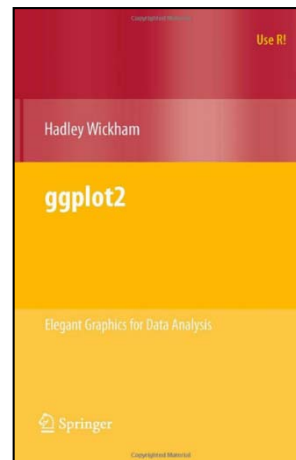
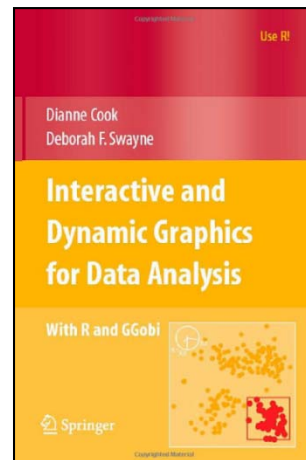
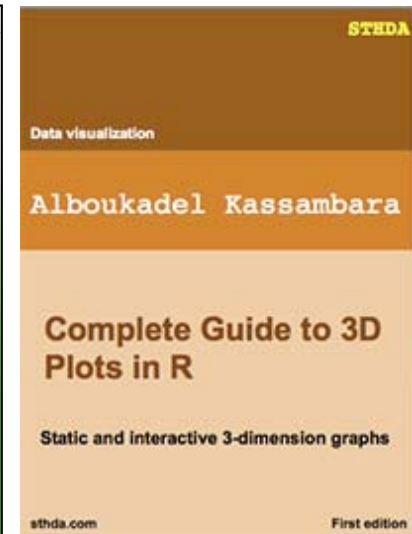
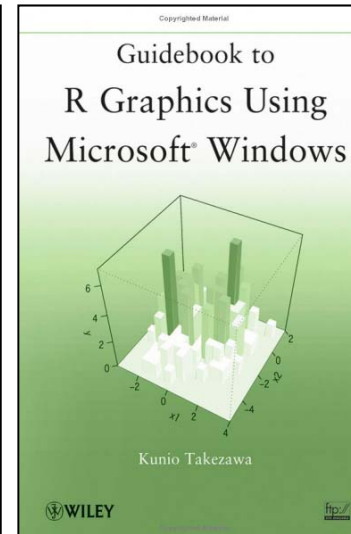
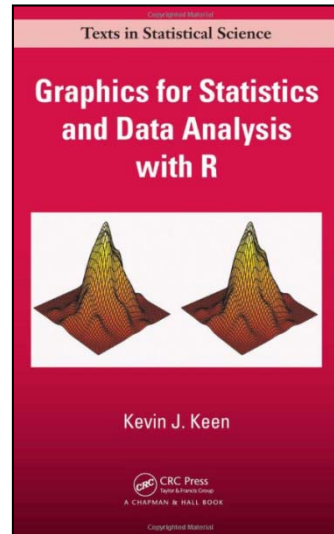
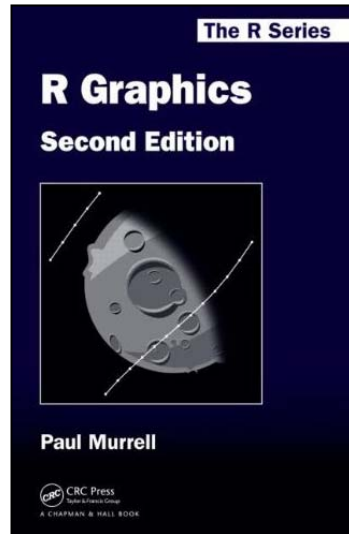
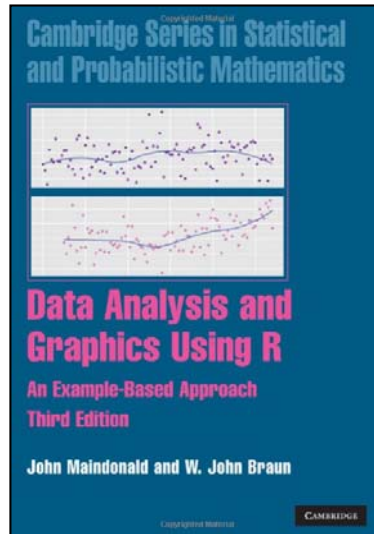
U[,3]

FSIR+FCM

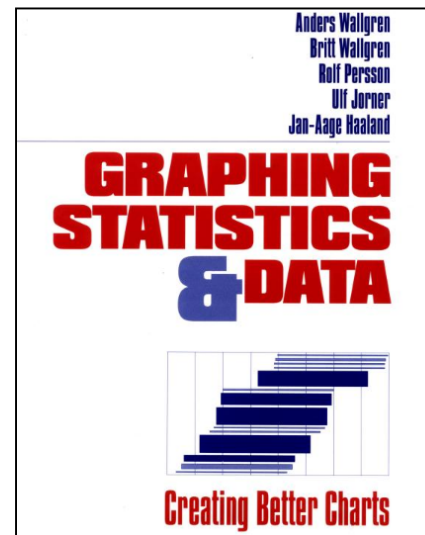
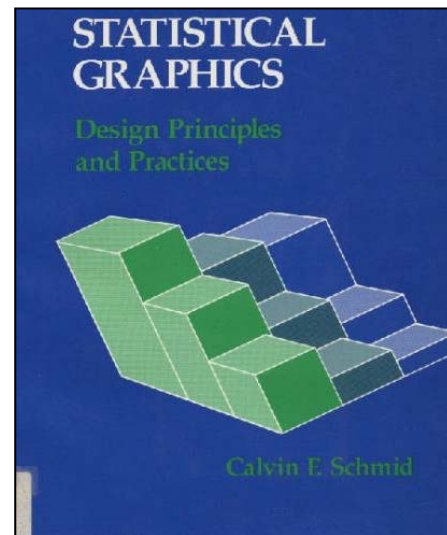
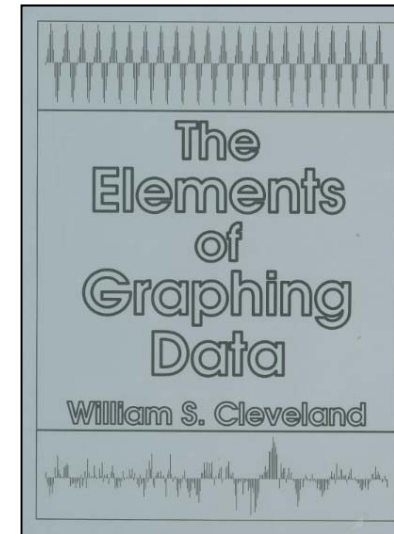
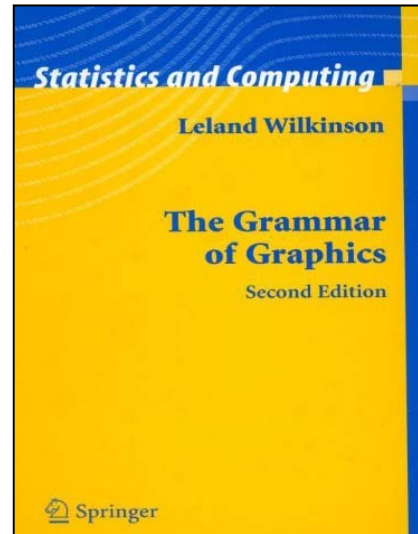
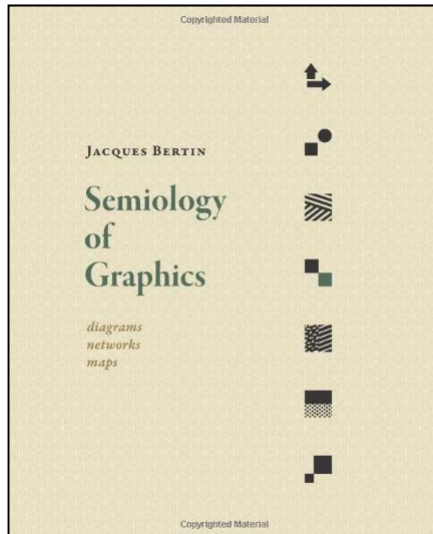


R圖形參考書目

206/216

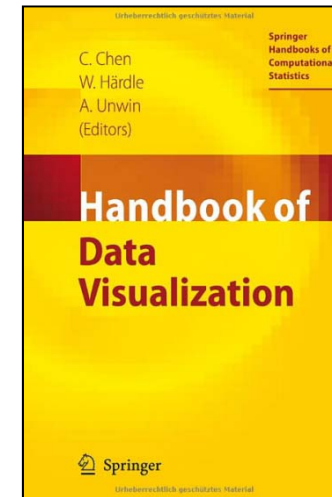
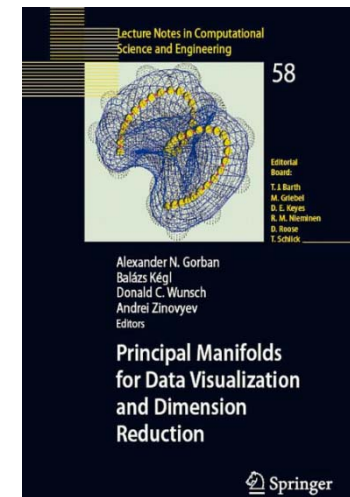
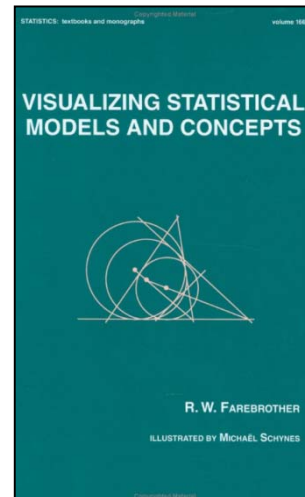
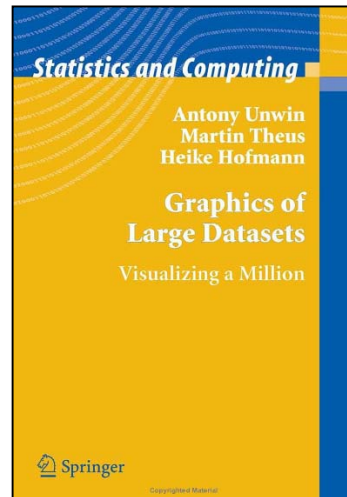
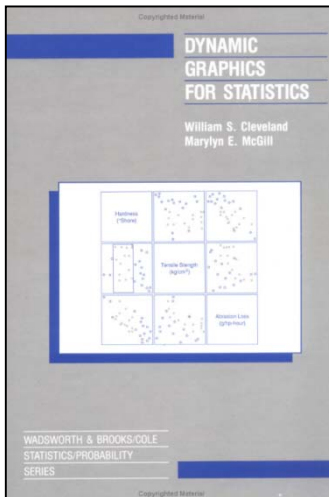
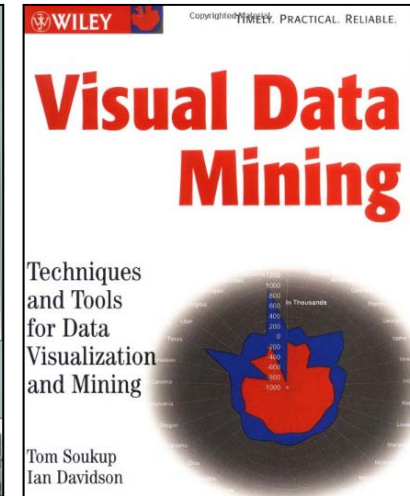
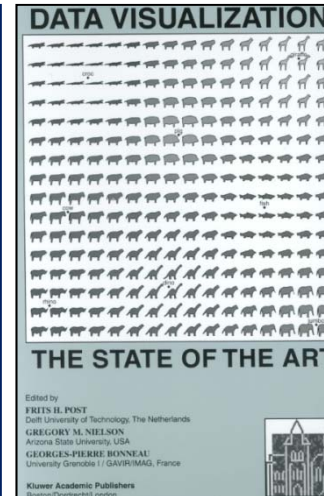
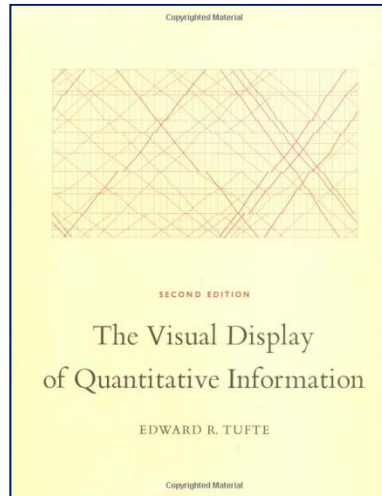
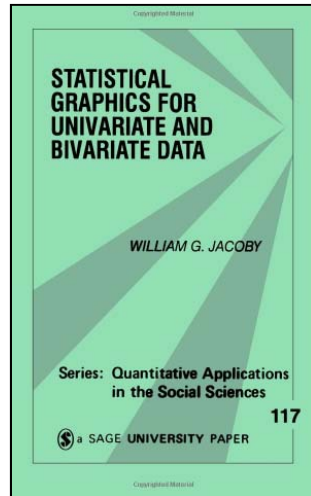
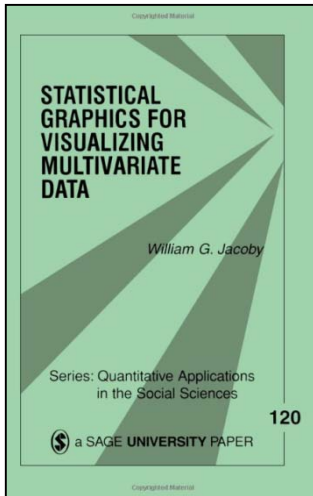


Fundations, Principle



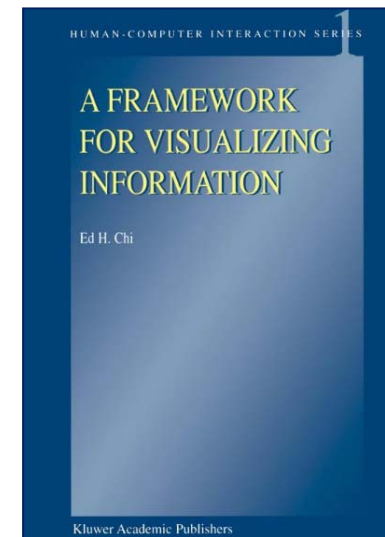
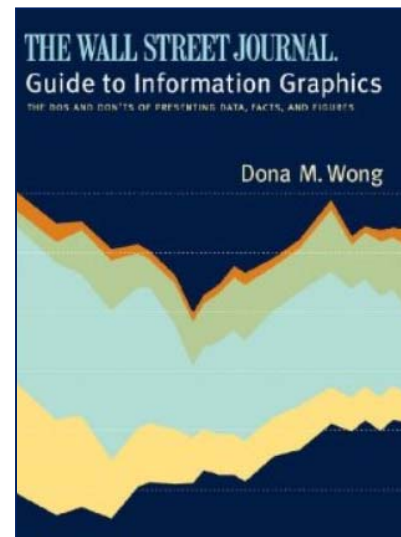
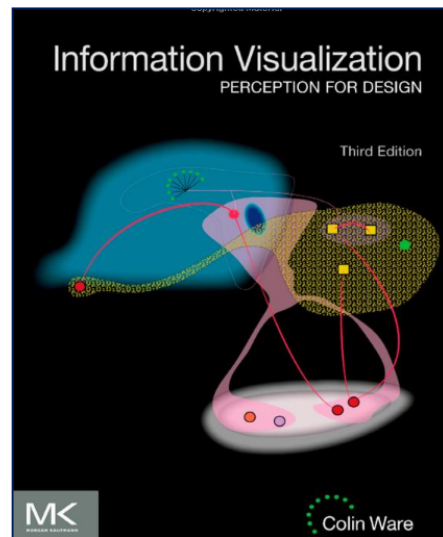
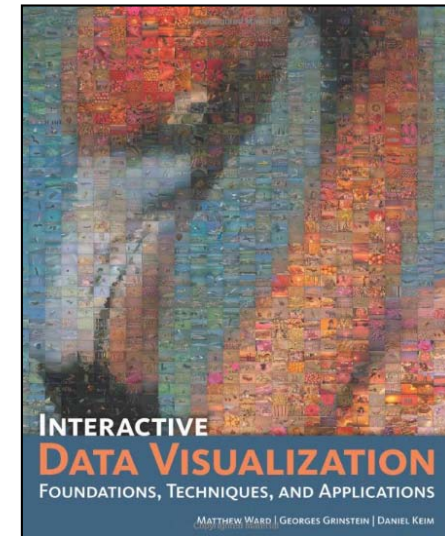
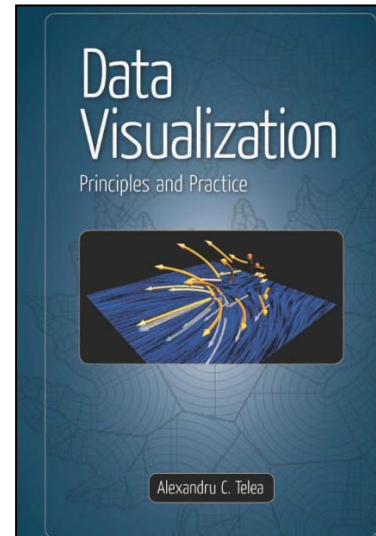
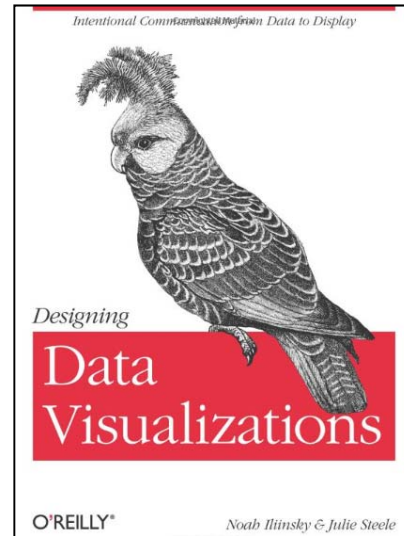
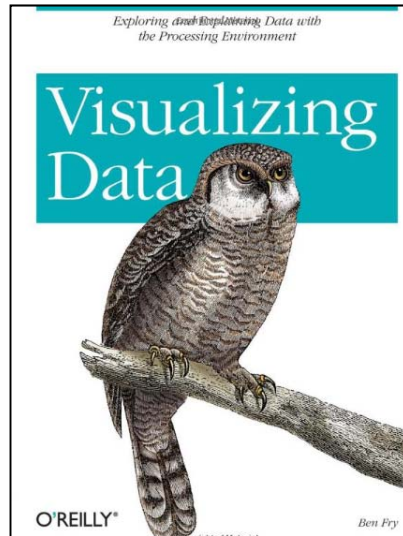
Statistics and Graphics

208/216



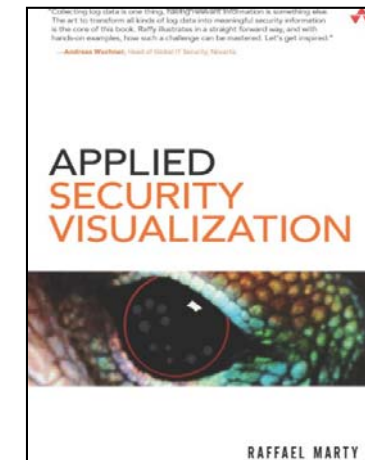
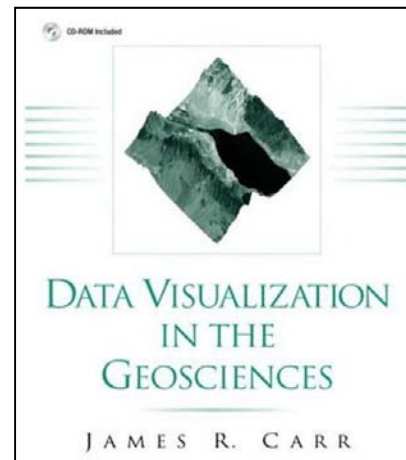
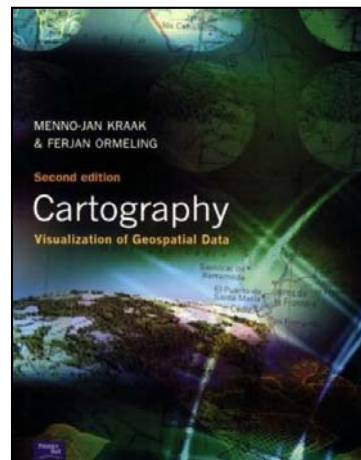
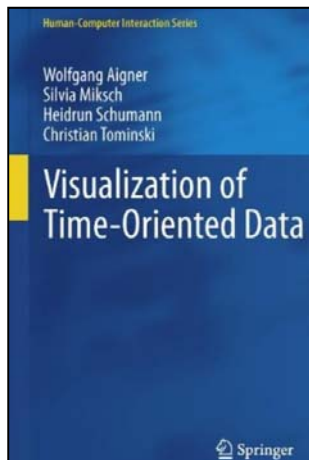
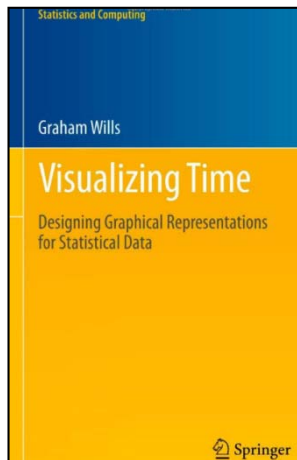
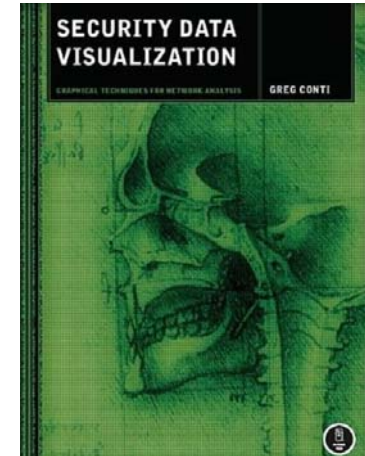
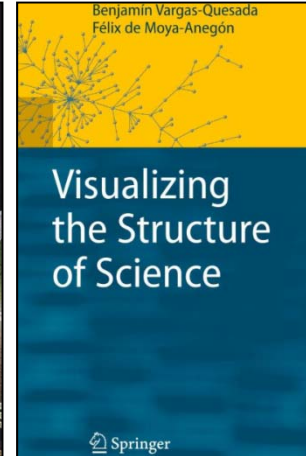
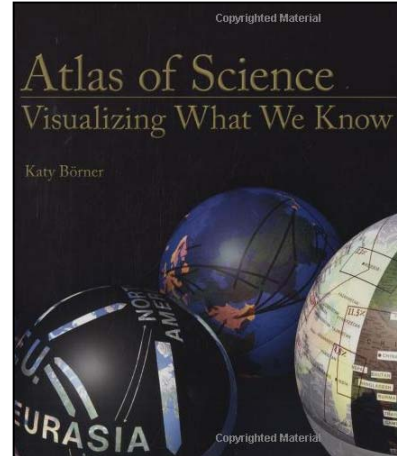
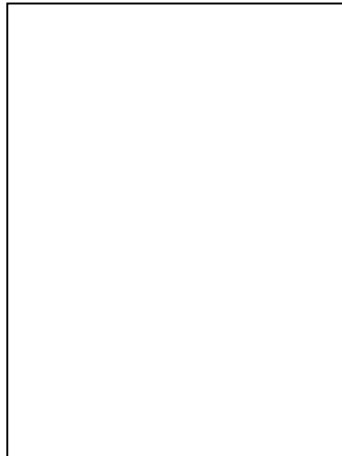
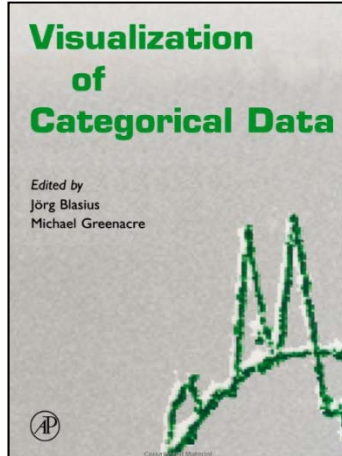


Data Visualization

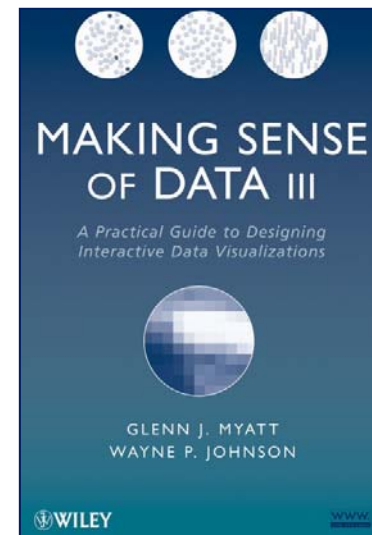
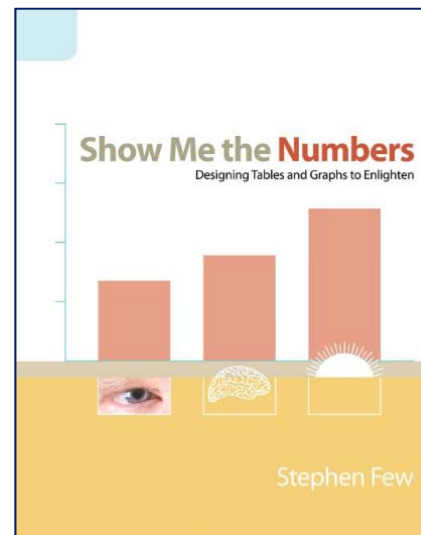
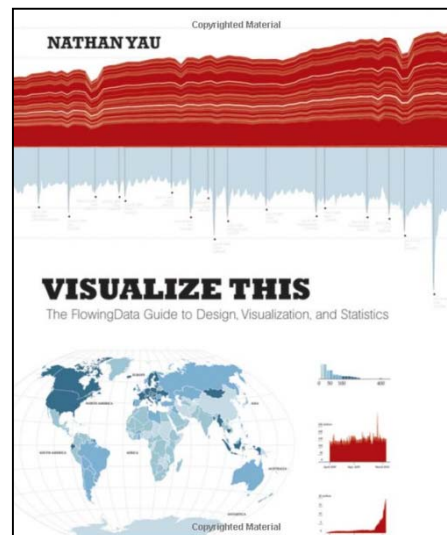
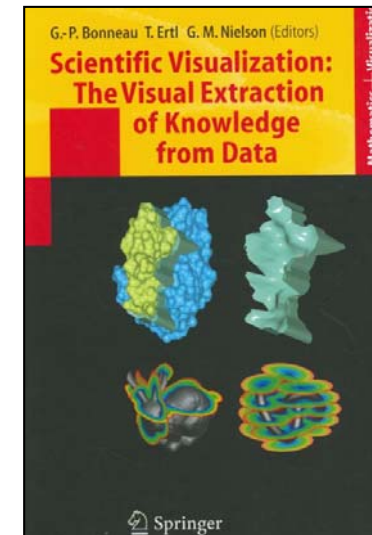
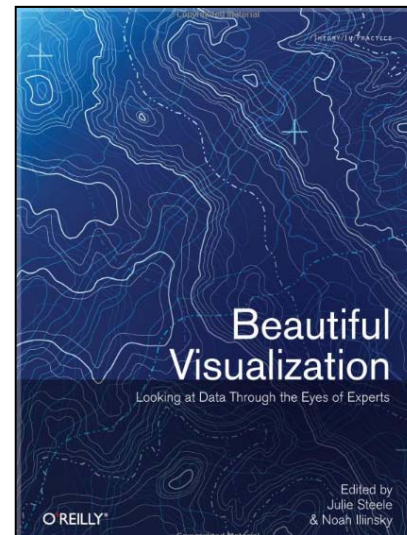
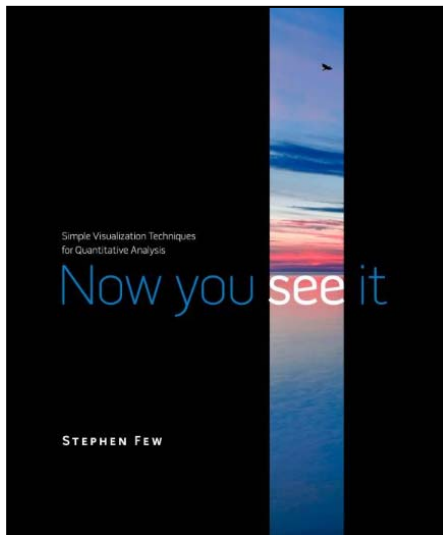




Specified Data Types



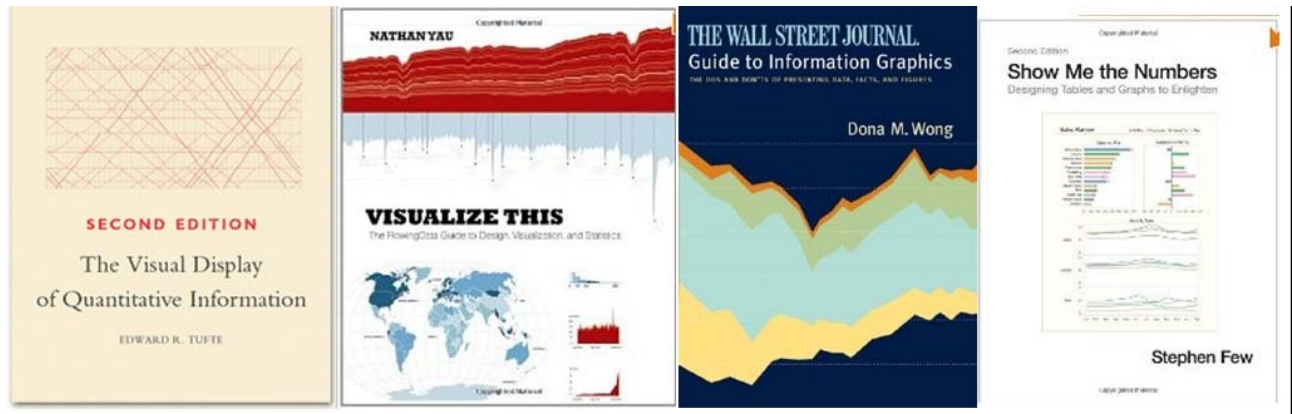
Others





Must read books on data visualization

<https://www.analyticsvidhya.com/blog/2013/09/read-books-visualization/>



Sadly, not many analysts spend time on improving and thinking about visualization.

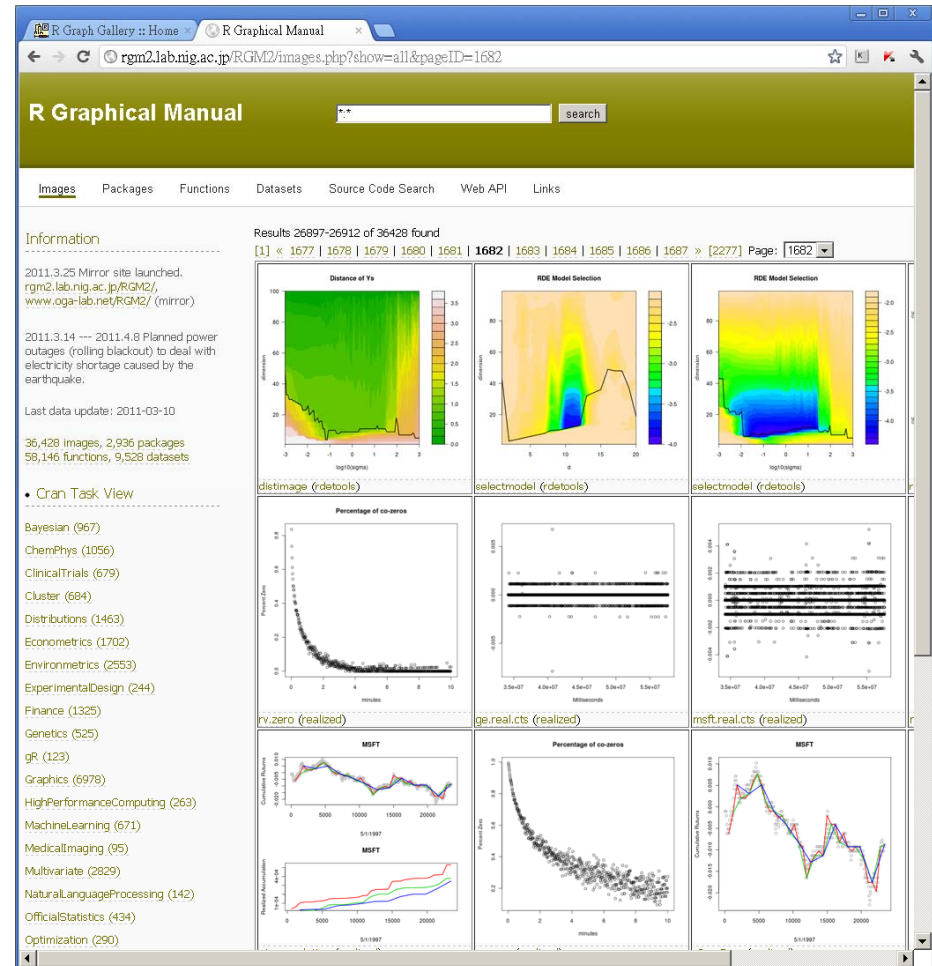
Must read books on data visualization



- Data Visualization – How to Pick the Right Chart Type? <https://eazybi.com/blog/data-visualization-and-chart-types/>
- Introduction to Data Visualization: Chart Dos and Don'ts <http://guides.library.duke.edu/datavis/top10>



The screenshot shows the homepage of the R Graph Gallery. At the top, there is a navigation bar with 'Home', 'List View', and 'Thumbnails View'. Below this is a search bar and a 'Donate' button. The main content area is divided into 'Last entries' and 'Random entries', each displaying a grid of various R plots and charts. On the right side, there is a social media section with a 'Tweets about the R Graph Gallery' box and a 'Recent news from R-bloggers' section with a link to <http://www.r-bloggers.com/>. At the bottom, there is a 'Find us on Facebook' section with the R Graph Gallery logo and a list of 353 people who like the page, including Antonio, Masato, Kang, Huang-Mi, and Tokhir.



The screenshot shows the R Graphical Manual page. The title is 'R Graphical Manual' and the URL is <http://rgm2.lab.nig.ac.jp/RGM2/images.php?show=all&pageID=1682>. The page features a search bar and a navigation menu with 'Images', 'Packages', 'Functions', 'Datasets', 'Source Code Search', 'Web API', and 'Links'. The main content area displays 'Results 26897-26912 of 36428 found' and a grid of 12 plots. The plots include 'Distance of Ts', 'RDE Model Selection', 'Percentage of co-zeros', 'rv.zero (realized)', 'ge.real.cts (realized)', 'metf.real.cts (realized)', and 'MSFT'. The page also includes an 'Information' section with text about the 2011.3.25 Mirror site launch and a 'Cran Task View' section listing various R packages and their counts, such as Bayesian (967), ChemPhys (1056), and OfficialStatistics (434).

Web Resource (1)

Michael Friendly's Home Page

DataVis.ca
Michael Friendly
York University

Milestones Project
The Milestones Project is a comprehensive, visual compendium of significant events in the histories of data visualization, statistical graphics and thematic cartography. This new version features an interactive timeline.
[Visit Site](#)

Data Visualization Gallery
This Gallery of Data Visualization displays some examples of the Best and Worst of Statistical Graphics, with the view that the contrast may be useful, inform current practice, and provide some pointers to both historical and current work.
[Visit Site](#)

Books
Here are links to my books on data visualization and statistical graphics, as well as other related books of interest

- SAS System for Statistical Graphics
- Visualizing Categorical Data
- Visual Statistics

[More](#)

Courses & Short Courses
I teach a variety of courses in the Psychology Department at York University and short courses on statistical topics through the Statistical Consulting Service at

Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization
An illustrated chronology of innovations by Michael Friendly and Daniel J. Denis

Introduction | Milestones Project | Varieties of Data Visualization | Related | References | Keyword Index |

Pre-1600 | 1600s | 1700s | 1800+ | 1850+ | 1900+ | 1950+ | 1975+

Timeline

This page provides a graphic overview of the events in the history of data visualization that we call "milestones." These milestones are shown below in the form of an *interactive timeline*. The timeline is divided into *two vertical sections*. You can *drag each section left or right* to see milestones of different time periods. You can also click one of the links at the bottom of the timeline to jump to a particular epoch.

Each of the milestone's in the timeline can be clicked to reveal its summary that includes both a link to its full details and a category to which it belongs. The category can also be clicked to initiate a search of other milestone's based on that category.

Item categories: ● Cartography ● Statistics and graphics ● Technology ● Other

Chartbook, Linked brushing, Brushing, Grand tour, Treemaps, Table lens, Grammar of Graphics, Scatterplot, S language, Mosaic display, Parallel coordinates plot, Lisp-Stat, Cartographic data visualiser, Sparkline, RIM-9, Cartesian rectangle, Sieve diagram, Categorical data graphics, Tag cloud, Word cloud, Committee on graphics, Graphical esthetics, Enhanced mosaic display, Xgobi, VISTA, Chartbook, Visibility base map, Multivariate grand tours, Circular display, USA Today weather map, Textured dot strips, Chernoff faces, Fisheye view, Lexis pencil, Graphics versus tables, Draftsman display, Parallel coordinates plot theory, Scatterplot matrix, Bifocal display, Interactive maps, Interactive grand tours, Hivariate matrix, Geographic correlation diagram, Nested dimensions, Comparative graphics, Indicator chartbook

Timeline © StatL: 1860 1870 1880 1890 1900 1910 1920 1930 1940 1950 1960 1970 1980 1990 2000 2010 2020 2030 2040 2050 2060 2070 2080 2090 2100

Pre-1600 | 1600s | 1700s | 1800+ | 1850+ | 1900+ | 1950+ | 1975+

This web version is dedicated to Arthur H. Robinson (1915-2004), who inspired and encouraged our interest; to Antoine de Falguerolles, who initiated it,

Web Resource (2)

215/216

VizWiz
Data Visualization Done Right

May 24, 2012

Creating an interactive monthly calendar in Tableau is easier than you might think

If you're not following the InterWorks blog, you should be. They routinely crank out fantastic tips and tricks for Tableau. Dustin Wyers, a BI Analyst for InterWorks, recently wrote about "Creating Calendar Views in Tableau". Dustin's post does an excellent job of taking you through creating a calendar view step-by-step. But I felt it a bit short in the end since you didn't actually see a calendar. Dustin's end product looks like this:

Columns: MY(Order Date) WEEKDAY(Order Date)
Rows: WEEK(Order Date)

Week	1	2	3	4	5	6	7
Week 1	1	2	3	4	5	6	7
Week 2	8	9	10	11	12	13	14
Week 3	15	16	17	18	19	20	21
Week 4	22	23	24	25	26	27	28
Week 5	29	30	31				

I wanted something that looked more like a true calendar. I did so utilizing some of the techniques I outlined recently for creating a heat map, but also adding in some of the suggestions by Joe Mako. The end result, if you follow the tutorial below, will look like this:

April 2012						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Wolfram Demonstrations Project

7922 Interactive Demonstrations
Powered by CDF Technology

Bring ideas to life
from recreation & education
to research & industry
Start exploring >>

FEATURED DEMONSTRATIONS

View latest >

- Mathematics: Algebra | Calculus & Analysis
- Business & Social Systems: Economics | Finance ...
- Creative Arts: Art | Architecture | Music ...
- Computation: Algorithms | Computer Science ...
- Systems, Models & Methods: Discrete Models | Networks ...
- Kids & Fun: For Kids | Puzzles | Optical Illusions ...
- Physical Sciences
- Engineering & Technology
- Mathematica Functionality

FEATURED CONTRIBUTORS

- Karl Scherer: Auckland, New Zealand (88 Demonstrations)
- Abraham Gadalla: Minneapolis, Minnesota, USA (17 Demonstrations)

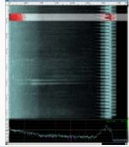
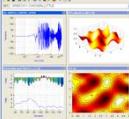
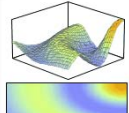
List of Information Graphics Software 216/216

Wikipedia article: List of information graphics software

This article may require cleanup to meet Wikipedia's quality standards. No cleanup reason specified. Please add a `|reason=` parameter to this template. Please help improve this article if you can. The talk page may contain suggestions. (June 2008)

This is a list of software to create any kind of information graphics:

- either includes the ability to create one or more infographics from a provided data set
- either it is provided specifically for information visualization

Software	Example(s)	Interface	Licence(s)		Operating system	Distinguishing features
			License	Open Source (yes/no)		
Algebraator		GUI	Proprietary	No	\$58.99	Linux, Mac OS-X, Sugar, Windows
Baudline		GUI	Proprietary	No, source available	Free	FreeBSD, Linux, Mac OS-X, Solaris
DADISP		GUI, command line, SPL script language	Proprietary	No		UNIX, Windows
DAP						Statistics
DataScene		GUI	Shareware	No	Free express, \$169 - \$299 std. and pro.	Linux, Unix, Windows

DecisionStats article: Top 10 Graphical User Interfaces in Statistical Software

SPONSOR: ANALYTICS TRAINING

JIGSAW ACADEMY

www.jigsawacademy.in

Online Analytics Training

Learn advanced analytics with tools such as SAS, R, Excel, Angss and SPSS

TOP 10 GRAPHICAL USER INTERFACES IN STATISTICAL SOFTWARE

APRIL 29, 2010 BY AJAY OHRI | 9 COMMENTS

Here is a list of top 10 GUIs in Statistical Software. The overall criterion is based on-

- User Friendly Nature for a New User to begin click and point and learn.
- Cleanliness of Automated Code or Log generated.
- Practical application in consulting and corporate world.
- Cost and Ease of Ownership (including purchase, install, training, maintainability, renewal)
- Aesthetics (or just plain pretty)

However this list is not in order of ranking- (as beauty (of GUI) lies in eyes of the beholder). For a list of top 10 GUI in R language only please see - <http://foranalytics.wordpress.com/graphical-user-interfaces-for-r/>

This is only a GUI based list so it **excludes** notable *command line* or *text editor* *submit* *commands* based softwares which are also very powerful and user friendly.

1. JMP -

While critics of SAS Institute often complain on the premium pricing of the basic model (especially AFTER the entry of another SAS language software WPS from <http://www.teamwpc.co.uk/products/wps> - they should try out JMP from <http://jmp.com> - it has a 1 month free evaluation, is much less expensive and the GUI makes it very very easy to do basic statistical analysis and testing. The learning curve is surprisingly fast to pick it up (as it should be for well designed interfaces) and it allows for very good quality output graphics as well.

