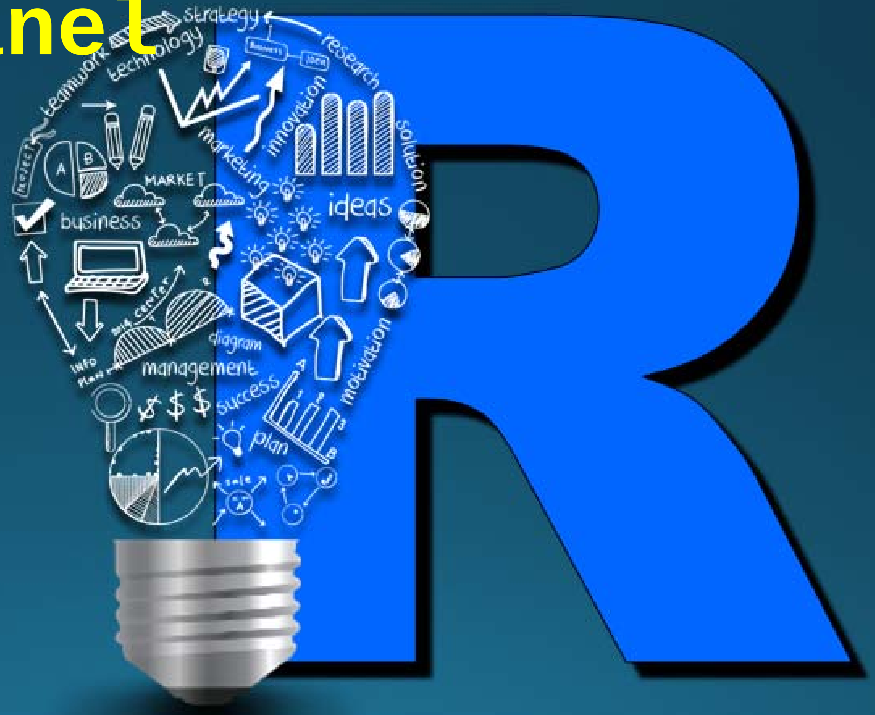
F01

R GUI
視窗程式設計
tcltk/tcltk2, rpanel

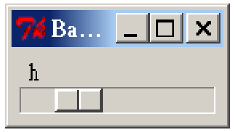


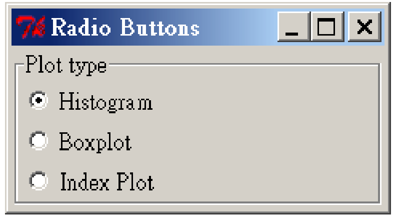
吳漢銘
國立政治大學 統計學系

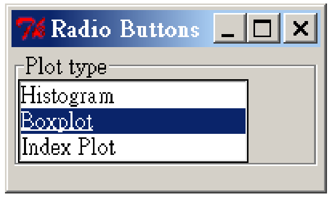
https://hmwu.idv.tw

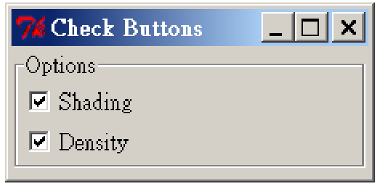
本章大綱&學習目標

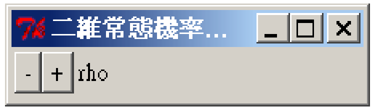
- **R GUI 簡介& 整合範例**
- 簡介使用tcltk package所設計R GUI 軟體。
- tcltk套件範例及指令查詢。
- 了解範例程式碼，並會利用tcltk實作R GUI軟體。
- **rpanel GUI元件**
  - **rp.slider, rp.radiogroup**
  - **rp.listbox, rp.checkbox**
  - **rp.doublebutton, rp.button**
  - **rp.messagebox, rp.menu**
  - **rp.textentry, rp.image**
  - **positioning controls**
  - **讀取檔案**
  - **display R graphics in a panel**
- **了解範例程式碼，並會利用rpanel實作R使用者介面(GUI)。**

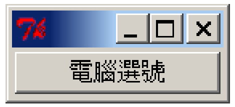


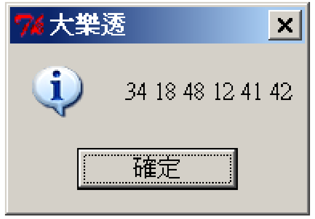




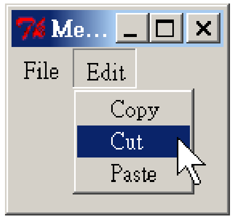


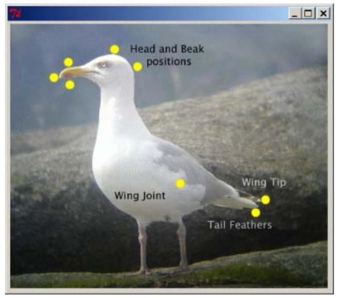


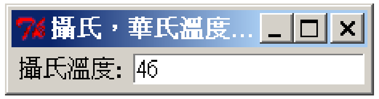




34 18 48 12 41 42

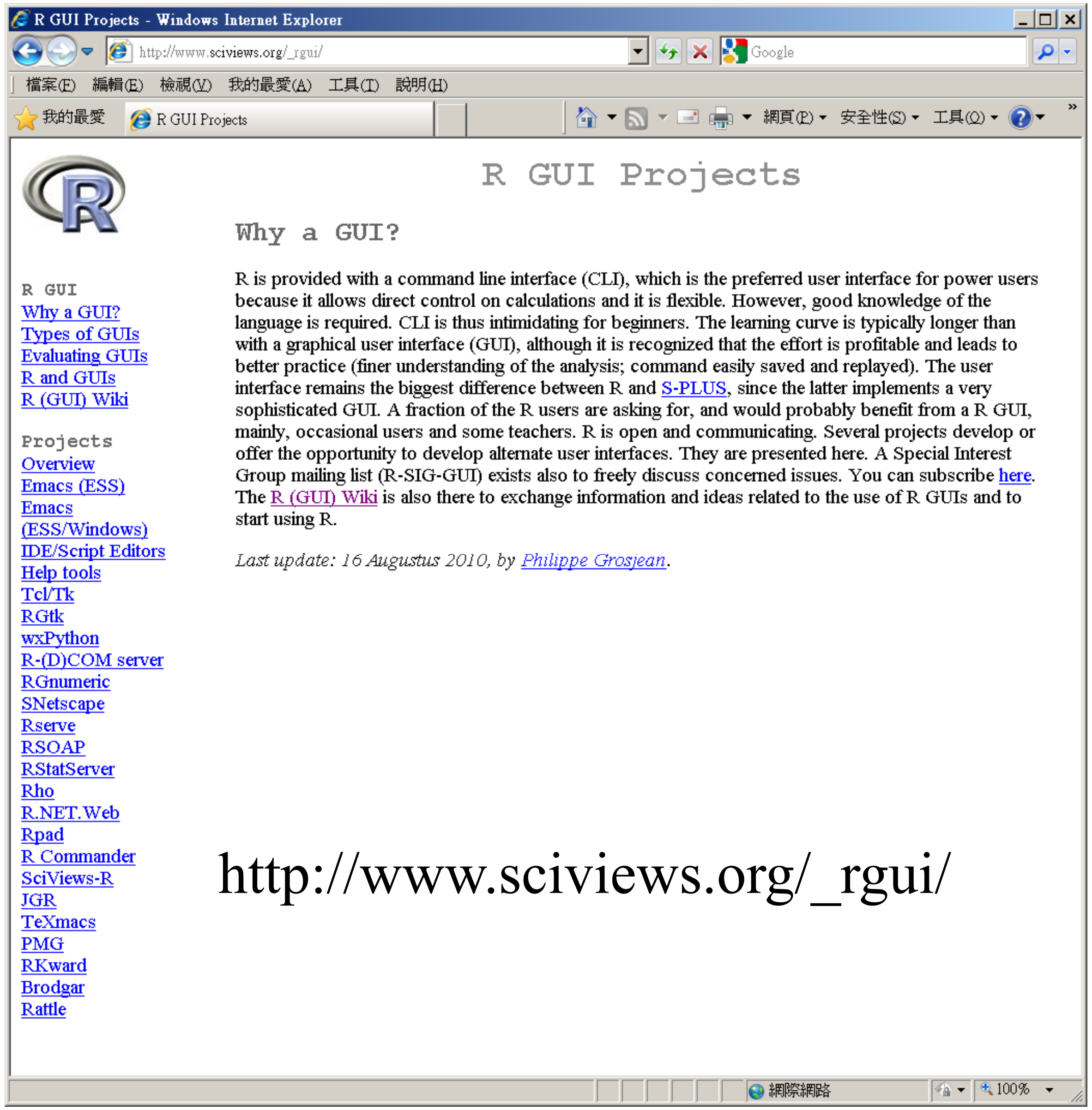






攝氏溫度: 46

R GUIs: Graphical User Interfaces for R


R GUI Projects - Windows Internet Explorer
http://www.sciviews.org/_rgui/
Google
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
我的最愛
R GUI Projects
網頁(P) 安全性(S) 工具(O)
網際網路
100%

R GUI Projects

Why a GUI?

R GUI
Why a GUI?
Types of GUIs
Evaluating GUIs
R and GUIs
R (GUI) Wiki

Projects
Overview
Emacs (ESS)
Emacs (ESS/Windows)
IDE/Script Editors
Help tools
Tcl/Tk
RGtk
wxPython
R-(D)COM server
RGnumeric
SNetscape
Rserve
RSOAP
RStatServer
Rho
R.NET.Web
Rpad
R Commander
SciViews-R
JGR
TeXmacs
PMG
RKward
Brodgar
Rattle

R is provided with a command line interface (CLI), which is the preferred user interface for power users because it allows direct control on calculations and it is flexible. However, good knowledge of the language is required. CLI is thus intimidating for beginners. The learning curve is typically longer than with a graphical user interface (GUI), although it is recognized that the effort is profitable and leads to better practice (finer understanding of the analysis; command easily saved and replayed). The user interface remains the biggest difference between R and S-PLUS, since the latter implements a very sophisticated GUI. A fraction of the R users are asking for, and would probably benefit from a R GUI, mainly, occasional users and some teachers. R is open and communicating. Several projects develop or offer the opportunity to develop alternate user interfaces. They are presented here. A Special Interest Group mailing list (R-SIG-GUI) exists also to freely discuss concerned issues. You can subscribe here. The R (GUI) Wiki is also there to exchange information and ideas related to the use of R GUIs and to start using R.

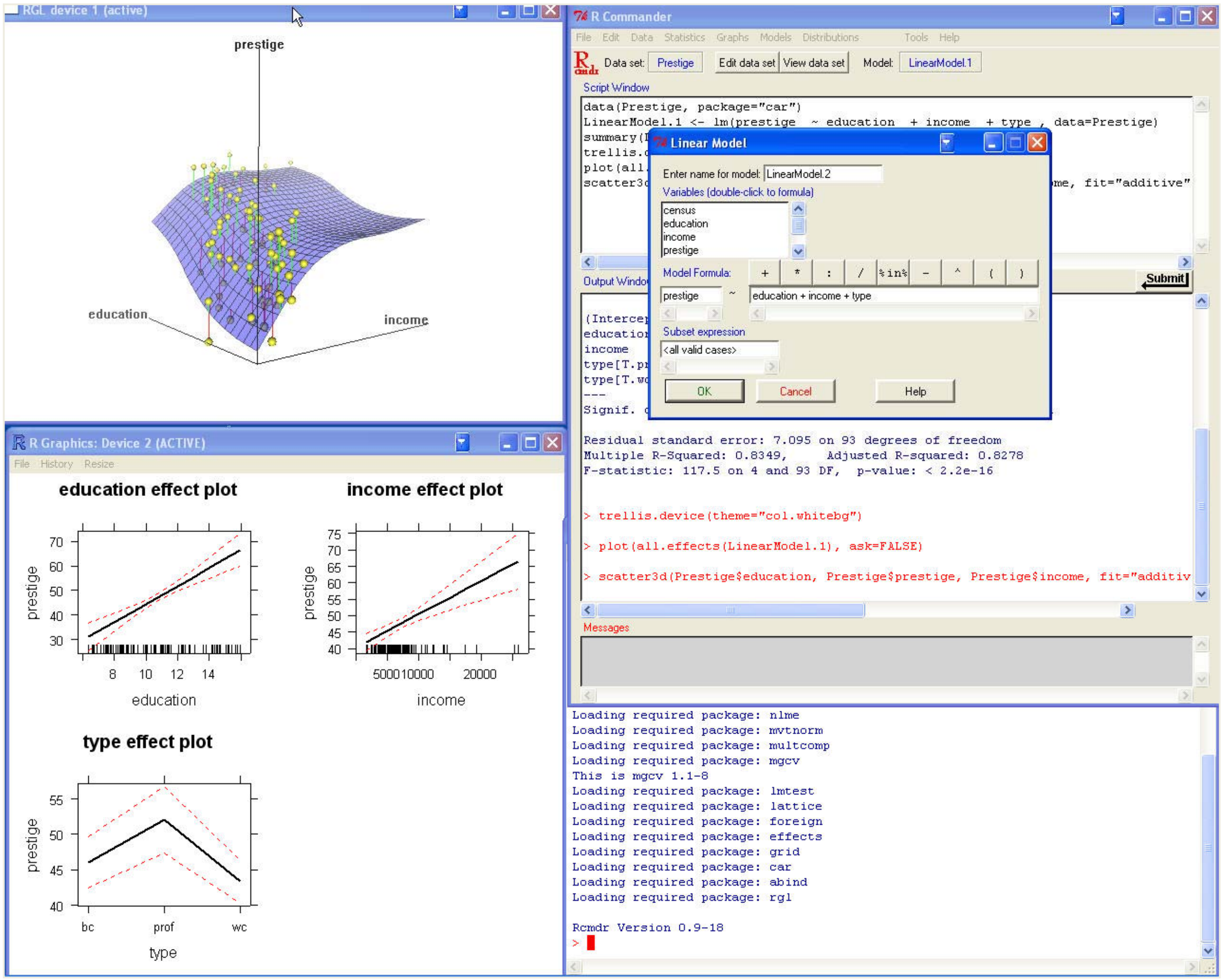
Last update: 16 Augustus 2010, by Philippe Grosjean.

http://www.sciviews.org/_rgui/

R GUI軟體設計實例 1
The R Commander: A Basic-Statistics GUI for R

(image source)

http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/



Data set: Prestige | Edit data set | View data set | Model: LinearModel.1

```
data(Prestige, package="car")
LinearModel.1 <- lm(prestige  ~ education  + income  + type , data=Prestige)
summary(L
trellis.d
plot(all.
scatter3d                                              me, fit="additive"
```

Enter name for model: LinearModel.2

Variables (double-click to formula): census, education, income, prestige

Model Formula: prestige ~ education + income + type

Subset expression: <all valid cases>

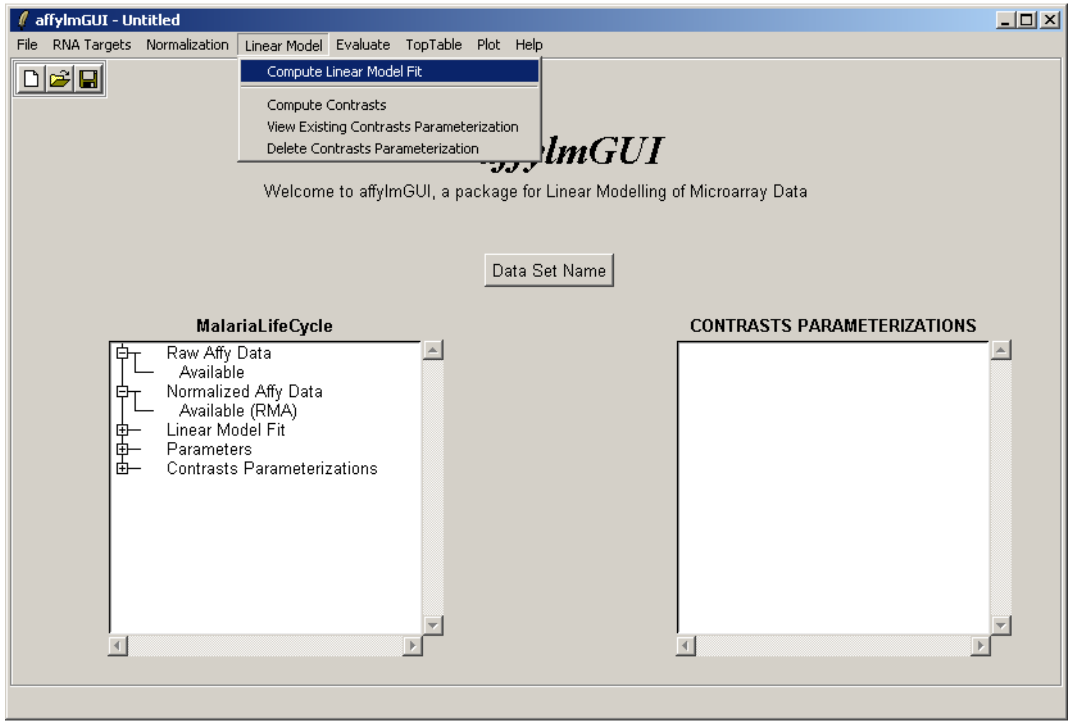
```
(Intercep
education
income
type[T.pr
type[T.wc
---
Signif. c

Residual standard error: 7.095 on 93 degrees of freedom
Multiple R-Squared: 0.8349,     Adjusted R-squared: 0.8278
F-statistic: 117.5 on 4 and 93 DF,  p-value: < 2.2e-16


> trellis.device(theme="col.whitebg")

> plot(all.effects(LinearModel.1), ask=FALSE)

> scatter3d(Prestige$education, Prestige$prestige, Prestige$income, fit="additiv
```

```
Loading required package: nlme
Loading required package: mvtnorm
Loading required package: multcomp
Loading required package: mgcv
This is mgcv 1.1-8
Loading required package: lmtest
Loading required package: lattice
Loading required package: foreign
Loading required package: effects
Loading required package: grid
Loading required package: car
Loading required package: abind
Loading required package: rgl

Rcmdr Version 0.9-18
>
```

R GUI軟體設計實例 2

affylmGUI: GUI for affy analysis using limma package

(image source)
http://bioinf.wehi.edu.au/.../LifeCycle/LifeCycle.html



affylmGUI - Untitled

File RNA Targets Normalization Linear Model Evaluate TopTable Plot Help

Compute Linear Model Fit
Compute Contrasts
View Existing Contrasts Parameterization
Delete Contrasts Parameterization

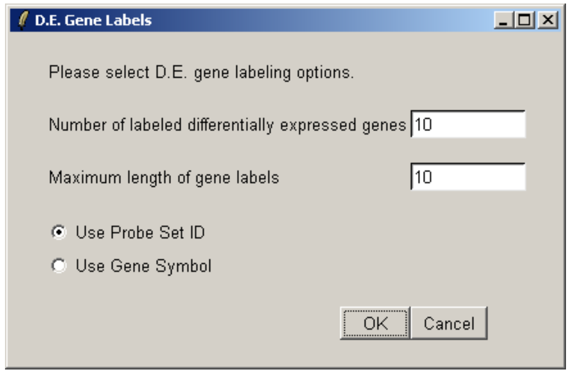
affylmGUI

Welcome to affylmGUI, a package for Linear Modelling of Microarray Data

Data Set Name

MalariaLifeCycle
- Raw Affy Data
  - Available
- Normalized Affy Data
  - Available (RMA)
- Linear Model Fit
- Parameters
- Contrasts Parameterizations

CONTRASTS PARAMETERIZATIONS



D.E. Gene Labels

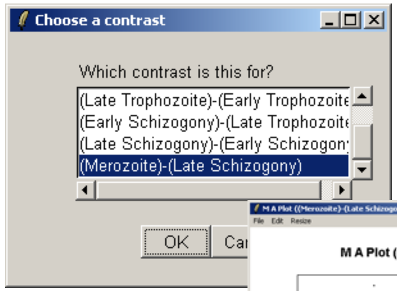
Please select D.E. gene labeling options.

Number of labeled differentially expressed genes: 10

Maximum length of gene labels: 10

[✓ Use Probe Set ID]
Use Gene Symbol

OK Cancel

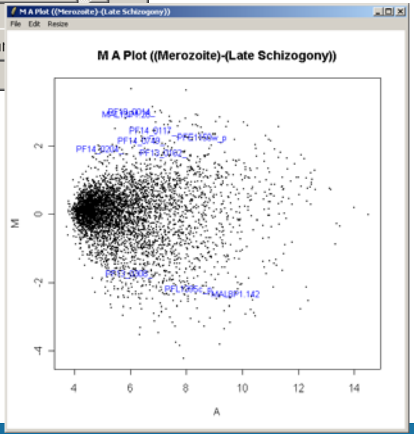


Choose a contrast

Which contrast is this for?

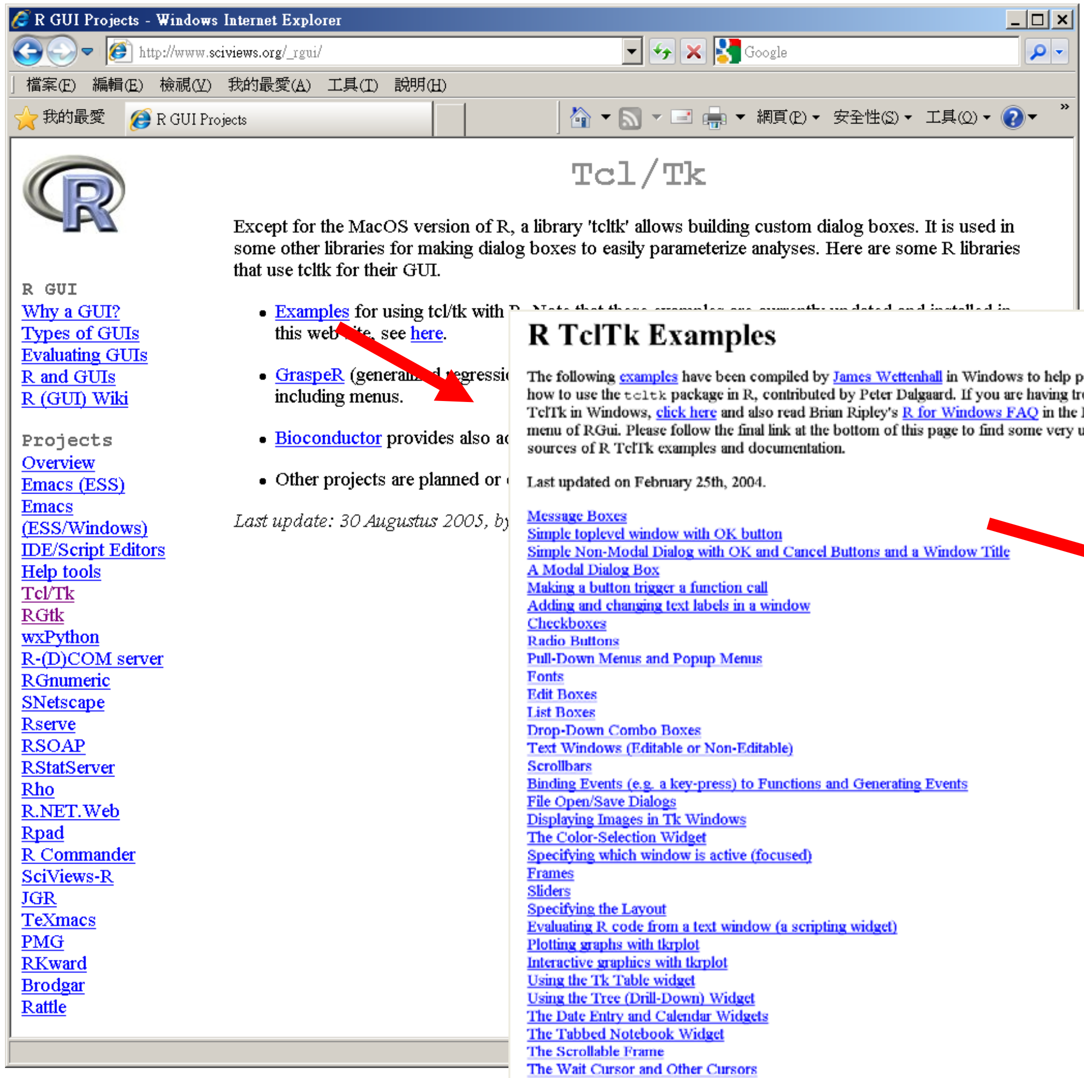
(Late Trophozoite)-(Early Trophozoite
(Early Schizogony)-(Late Trophozoite
(Late Schizogony)-(Early Schizogon
[✓ (Merozoite)-(Late Schizogony)]

OK Car



tcltk 套件

http://www.sciviews.org/_rgui/projects/TclTk.html


R GUI Projects - Windows Internet Explorer
http://www.sciviews.org/_rgui/
Google
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
我的最愛
R GUI Projects
網頁(P) 安全性(S) 工具(O)

R GUI
Why a GUI?
Types of GUIs
Evaluating GUIs
R and GUIs
R (GUI) Wiki

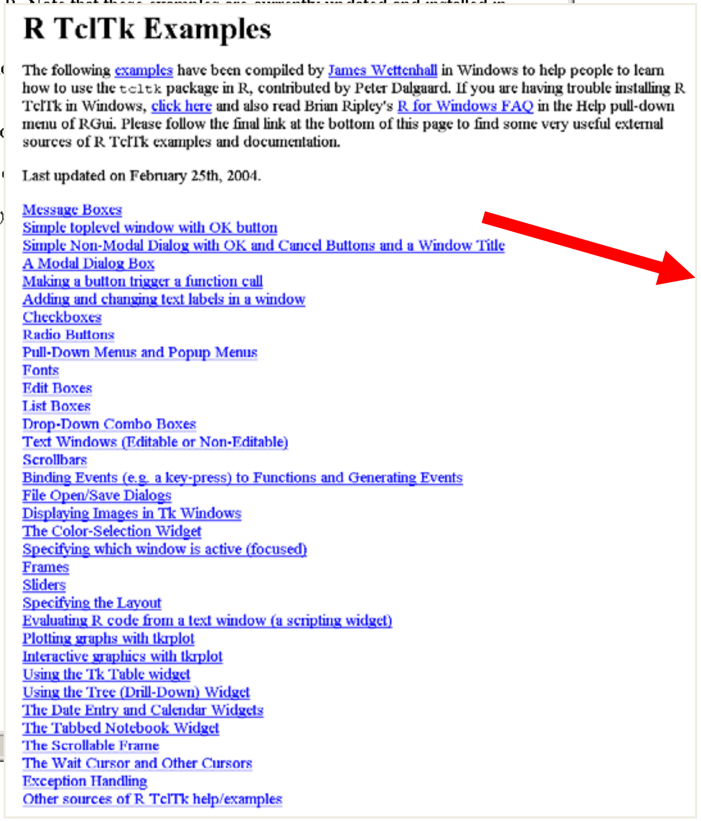
Projects
Overview
Emacs (ESS)
Emacs
(ESS/Windows)
IDE/Script Editors
Help tools
Tcl/Tk
RGtk
wxPython
R-(D)COM server
RGnumeric
SNetscape
Rserve
RSOAP
RStatServer
Rho
R.NET.Web
Rpad
R Commander
SciViews-R
JGR
TeXmacs
PMG
RKward
Brodgar
Rattle

Tcl/Tk

Except for the MacOS version of R, a library 'tcltk' allows building custom dialog boxes. It is used in some other libraries for making dialog boxes to easily parameterize analyses. Here are some R libraries that use tcltk for their GUI.

- Examples for using tcl/tk with R. Note that these examples are currently updated and installed in this web site, see here.
- GraspeR (generalized regressi including menus.
- Bioconductor provides also ac
- Other projects are planned or

Last update: 30 Augustus 2005, by

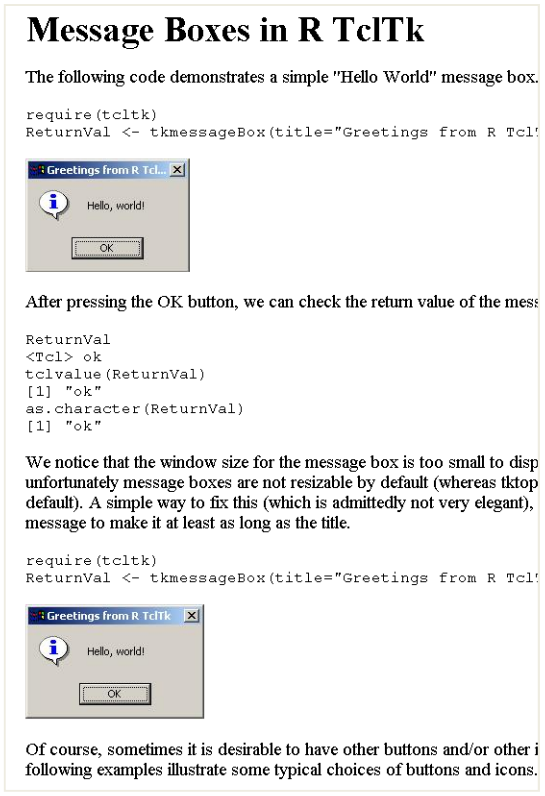


R TclTk Examples

The following examples have been compiled by James Wettenhall in Windows to help people to learn how to use the `tcltk` package in R, contributed by Peter Dalgaard. If you are having trouble installing R TclTk in Windows, click here and also read Brian Ripley's R for Windows FAQ in the Help pull-down menu of RGui. Please follow the final link at the bottom of this page to find some very useful external sources of R TclTk examples and documentation.

Last updated on February 25th, 2004.

Message Boxes
Simple toplevel window with OK button
Simple Non-Modal Dialog with OK and Cancel Buttons and a Window Title
A Modal Dialog Box
Making a button trigger a function call
Adding and changing text labels in a window
Checkboxes
Radio Buttons
Pull-Down Menus and Popup Menus
Fonts
Edit Boxes
List Boxes
Drop-Down Combo Boxes
Text Windows (Editable or Non-Editable)
Scrollbars
Binding Events (e.g. a key-press) to Functions and Generating Events
File Open/Save Dialogs
Displaying Images in Tk Windows
The Color-Selection Widget
Specifying which window is active (focused)
Frames
Sliders
Specifying the Layout
Evaluating R code from a text window (a scripting widget)
Plotting graphs with tkrplot
Interactive graphics with tkrplot
Using the Tk Table widget
Using the Tree (Drill-Down) Widget
The Date Entry and Calendar Widgets
The Tabbed Notebook Widget
The Scrollable Frame
The Wait Cursor and Other Cursors
Exception Handling
Other sources of R TclTk help/examples



Message Boxes in R TclTk

The following code demonstrates a simple "Hello World" message box.

```
require(tcltk)
ReturnVal <- tkmessageBox(title="Greetings from R Tcl
```

After pressing the OK button, we can check the return value of the mess

```
ReturnVal
<Tcl> ok
tclvalue(ReturnVal)
[1] "ok"
as.character(ReturnVal)
[1] "ok"
```

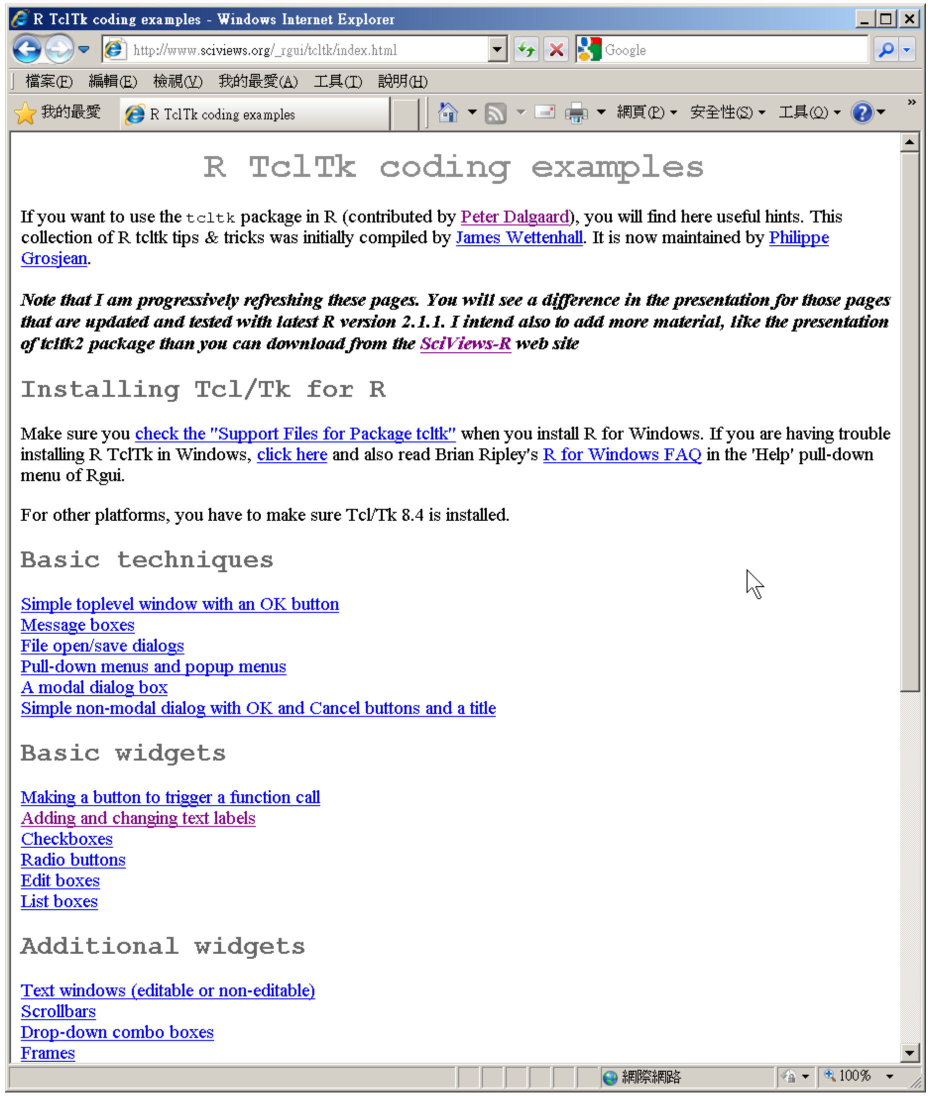
We notice that the window size for the message box is too small to disp
unfortunately message boxes are not resizable by default (whereas tktop
default). A simple way to fix this (which is admittedly not very elegant),
message to make it at least as long as the title.

```
require(tcltk)
ReturnVal <- tkmessageBox(title="Greetings from R Tcl
```

Of course, sometimes it is desirable to have other buttons and/or other i
following examples illustrate some typical choices of buttons and icons.

R TclTk coding examples

http://www.sciviews.org/_rgui/tcltk/index.html


R TclTk coding examples - Windows Internet Explorer
http://www.sciviews.org/_rgui/tcltk/index.html
Google
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
我的最愛
R TclTk coding examples
網頁(P) 安全性(S) 工具(O)
網際網路
100%

R TclTk coding examples

If you want to use the `tcltk` package in R (contributed by Peter Dalgaard), you will find here useful hints. This collection of R tcltk tips & tricks was initially compiled by James Wettenhall. It is now maintained by Philippe Grosjean.

Note that I am progressively refreshing these pages. You will see a difference in the presentation for those pages that are updated and tested with latest R version 2.1.1. I intend also to add more material, like the presentation of tcltk2 package than you can download from the SciViews-R web site

Installing Tcl/Tk for R

Make sure you check the "Support Files for Package tcltk" when you install R for Windows. If you are having trouble installing R TclTk in Windows, click here and also read Brian Ripley's R for Windows FAQ in the 'Help' pull-down menu of Rgui.

For other platforms, you have to make sure Tcl/Tk 8.4 is installed.

Basic techniques

Simple toplevel window with an OK button
Message boxes
File open/save dialogs
Pull-down menus and popup menus
A modal dialog box
Simple non-modal dialog with OK and Cancel buttons and a title

Basic widgets

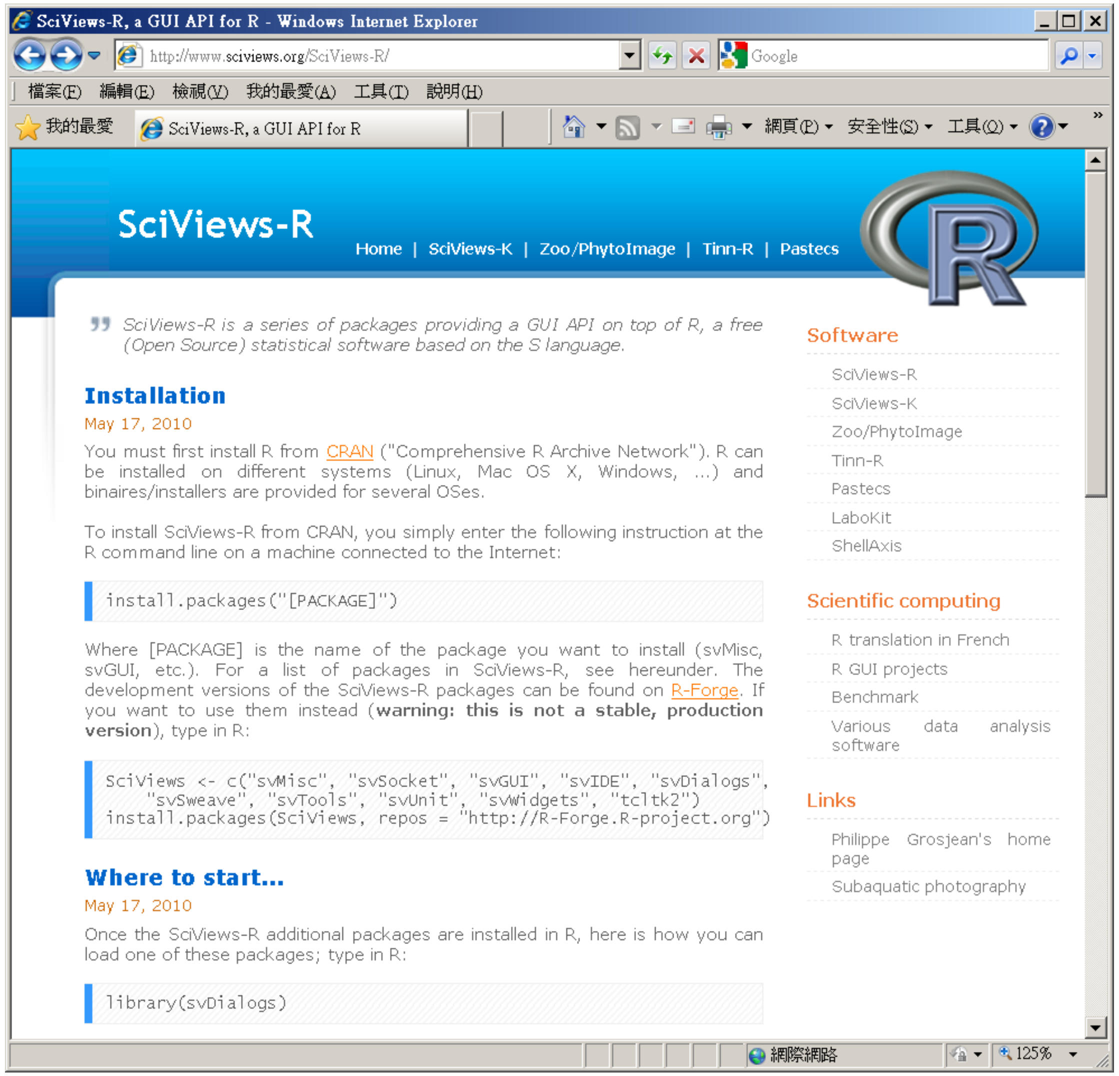
Making a button to trigger a function call
Adding and changing text labels
Checkboxes
Radio buttons
Edit boxes
List boxes

Additional widgets

Text windows (editable or non-editable)
Scrollbars
Drop-down combo boxes
Frames

SciViews-R (含tcltk2)

http://www.sciviews.org/SciViews-R/


SciViews-R, a GUI API for R - Windows Internet Explorer
http://www.sciviews.org/SciViews-R/
Google
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
我的最愛
SciViews-R, a GUI API for R
網頁(P) 安全性(S) 工具(O)
網際網路
125%

SciViews-R

Home | SciViews-K | Zoo/PhytoImage | Tinn-R | Pastecs

SciViews-R is a series of packages providing a GUI API on top of R, a free (Open Source) statistical software based on the S language.

Installation

May 17, 2010

You must first install R from CRAN ("Comprehensive R Archive Network"). R can be installed on different systems (Linux, Mac OS X, Windows, ...) and binaires/installers are provided for several OSes.

To install SciViews-R from CRAN, you simply enter the following instruction at the R command line on a machine connected to the Internet:

```
install.packages("[PACKAGE]")
```

Where [PACKAGE] is the name of the package you want to install (svMisc, svGUI, etc.). For a list of packages in SciViews-R, see hereunder. The development versions of the SciViews-R packages can be found on R-Forge. If you want to use them instead (**warning: this is not a stable, production version**), type in R:

```
SciViews <- c("svMisc", "svSocket", "svGUI", "svIDE", "svDialogs",
    "svSweave", "svTools", "svUnit", "svWidgets", "tcltk2")
install.packages(SciViews, repos = "http://R-Forge.R-project.org")
```

Where to start...

May 17, 2010

Once the SciViews-R additional packages are installed in R, here is how you can load one of these packages; type in R:

```
library(svDialogs)
```

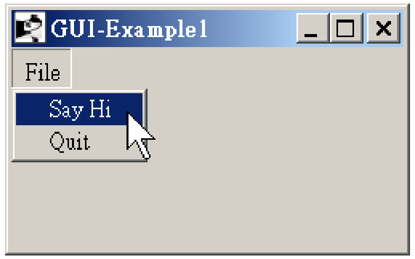
Software
- SciViews-R
- SciViews-K
- Zoo/PhytoImage
- Tinn-R
- Pastecs
- LaboKit
- ShellAxis

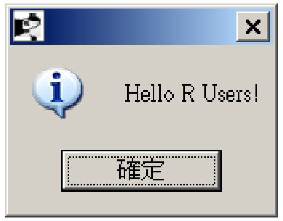
Scientific computing
- R translation in French
- R GUI projects
- Benchmark
- Various data analysis software

Links
- Philippe Grosjean's home page
- Subaquatic photography

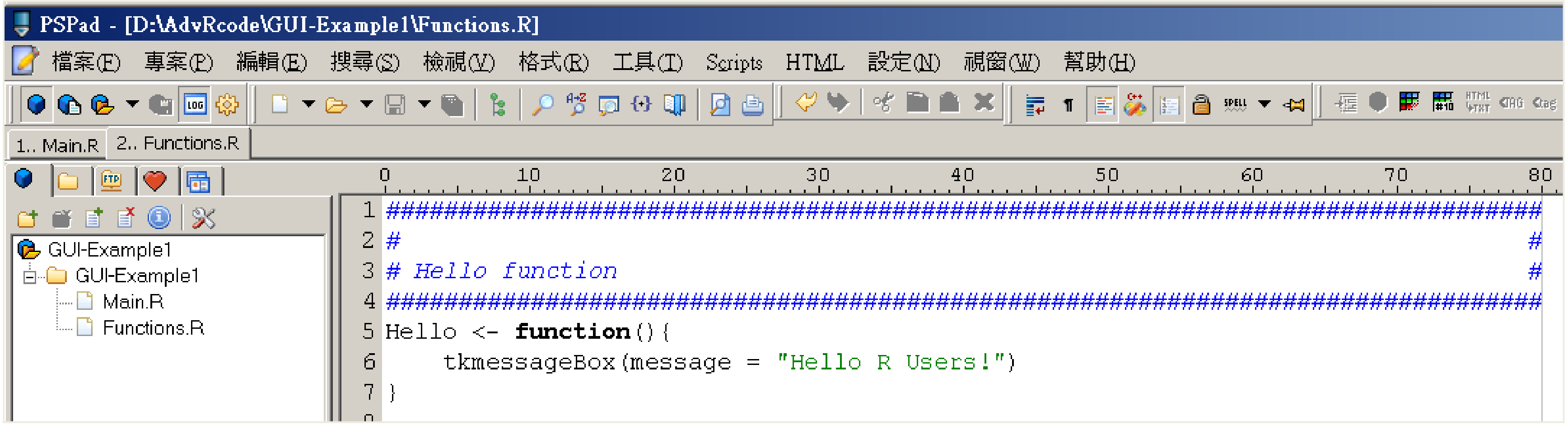
R GUI軟體設計範例 1

```
source("Main.R")
```




Hello R Users!

Hello 副程式:


PSPad - [D:\AdvRcode\GUI-Example1\Functions.R]
檔案(F) 專案(P) 編輯(E) 搜尋(S) 檢視(V) 格式(R) 工具(T) Scripts HTML 設定(N) 視窗(W) 幫助(H)
1.. Main.R
2.. Functions.R
GUI-Example1
GUI-Example1
Main.R
Functions.R

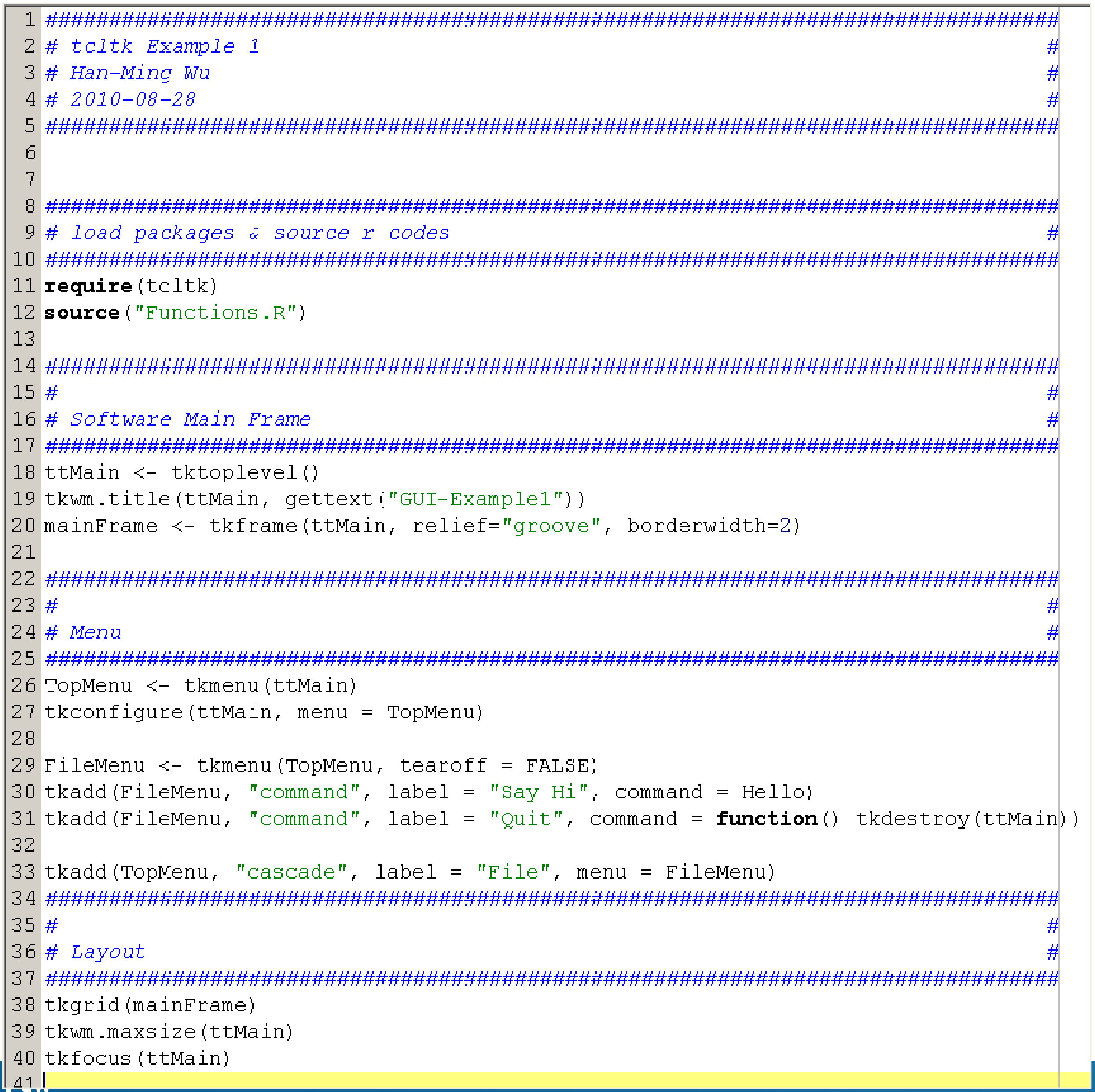
```
################################################################################
#                                                                              #
# Hello function                                                               #
################################################################################
Hello <- function(){
    tkmessageBox(message = "Hello R Users!")
}
```

R GUI軟體設計範例 1

主框架

選單

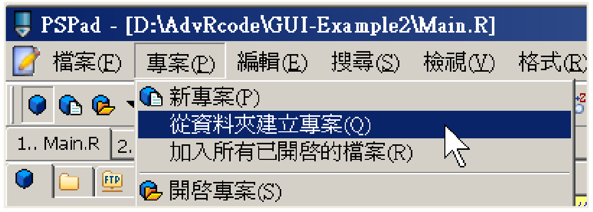
版面配置

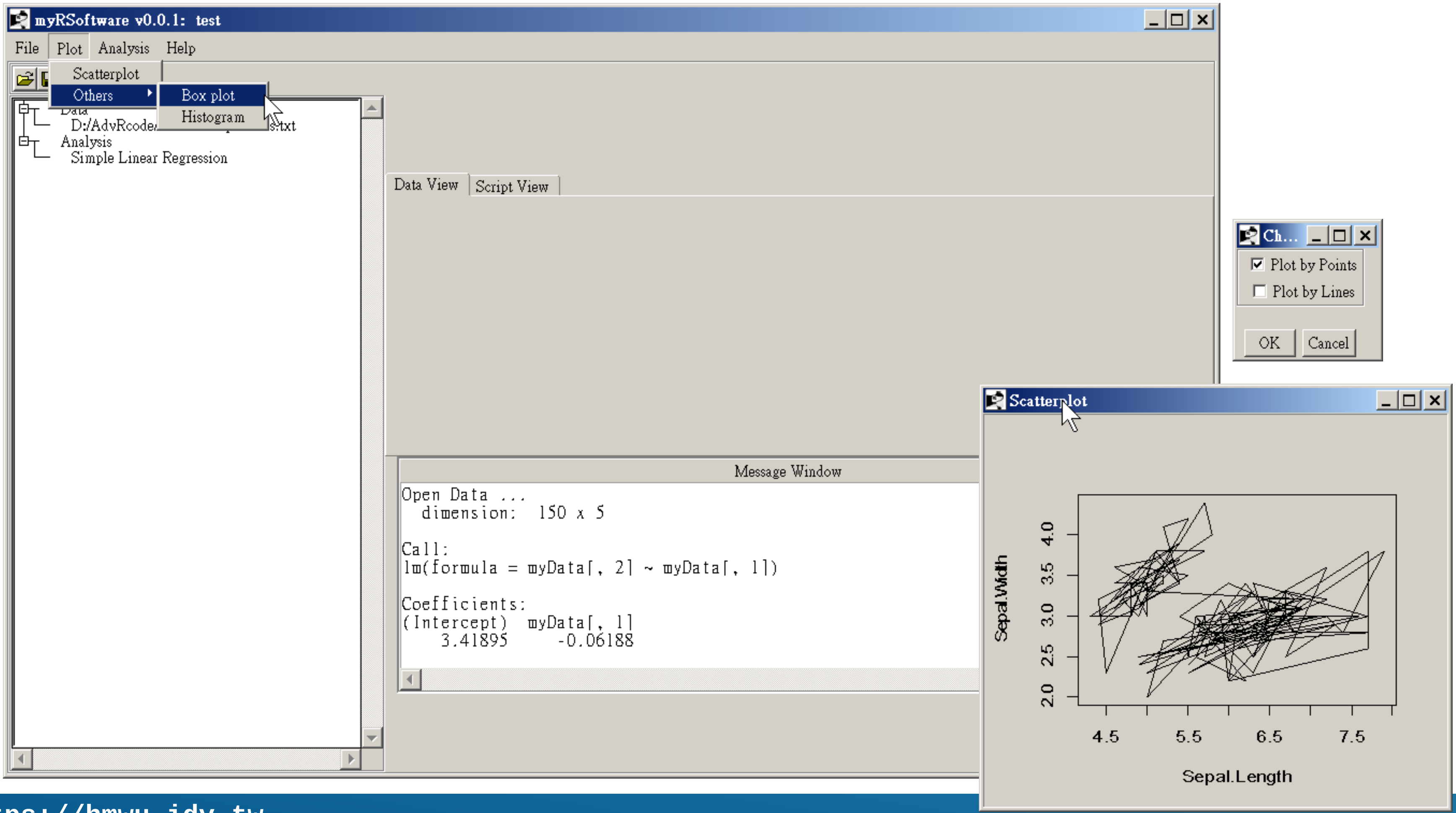


```
##############################################################################
# tcltk Example 1                                                            #
# Han-Ming Wu                                                                #
# 2010-08-28                                                                 #
##############################################################################


##############################################################################
# load packages & source r codes                                             #
##############################################################################
require(tcltk)
source("Functions.R")

##############################################################################
#                                                                            #
# Software Main Frame                                                        #
##############################################################################
ttMain <- tktoplevel()
tkwm.title(ttMain, gettext("GUI-Example1"))
mainFrame <- tkframe(ttMain, relief="groove", borderwidth=2)

##############################################################################
#                                                                            #
# Menu                                                                       #
##############################################################################
TopMenu <- tkmenu(ttMain)
tkconfigure(ttMain, menu = TopMenu)

FileMenu <- tkmenu(TopMenu, tearoff = FALSE)
tkadd(FileMenu, "command", label = "Say Hi", command = Hello)
tkadd(FileMenu, "command", label = "Quit", command = function() tkdestroy(ttMain))

tkadd(TopMenu, "cascade", label = "File", menu = FileMenu)
##############################################################################
#                                                                            #
# Layout                                                                     #
##############################################################################
tkgrid(mainFrame)
tkwm.maxsize(ttMain)
tkfocus(ttMain)
```

R GUI軟體設計範例 2

請下載程式碼練習：





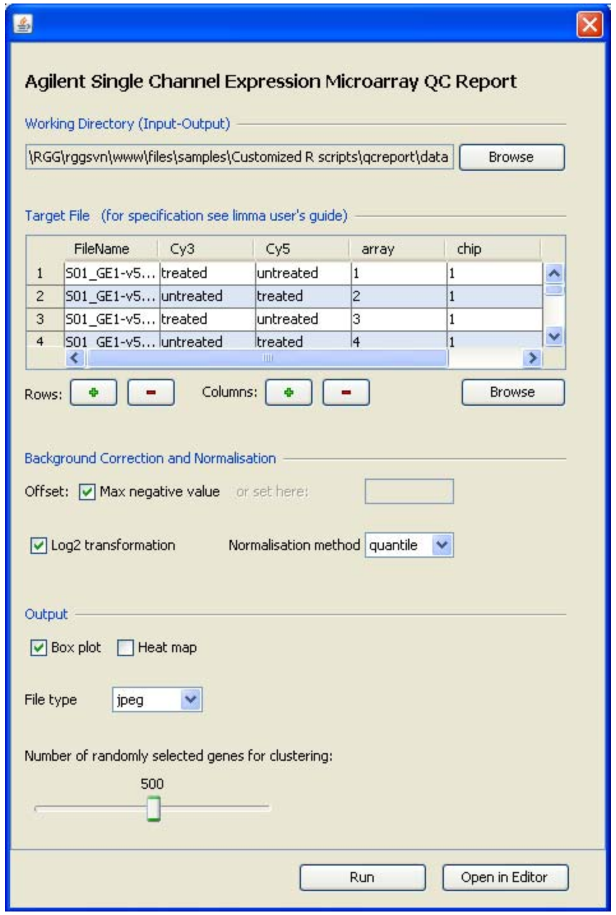
Plot by Points [✓] ; Plot by Lines [ ]

```
Open Data ...
  dimension:  150 x 5

Call:
lm(formula = myData[, 2] ~ myData[, 1])

Coefficients:
(Intercept)  myData[, 1]
    3.41895     -0.06188
```

其它參考資料

- Ilhami Visne, Erkan Dilaveroglu, Klemens Vierlinger, Martin Lauss, Ahmet Yildiz, Andreas Weinhaeusel, Christa Noehammer, Friedrich Leisch, and Albert Kriegner. **RGG: A general GUI framework for R scripts in bioinformatics**. BMC Bioinformatics, 10(1):74, 2009.

- Nan M. Laird & Thomas J. Hoffmann, . **fgui: A Method for Automatically Creating Graphical User Interfaces for Command-Line R Packages**, Journal of Statistical Software, American Statistical Association, vol. 30(i02).


Agilent Single Channel Expression Microarray QC Report

Working Directory (Input-Output)

\RGG\rggsvn\www\files\samples\Customized R scripts\qcreport\data

Target File (for specification see limma user's guide)

	FileName	Cy3	Cy5	array	chip
1	S01_GE1-v5...	treated	untreated	1	1
2	S01_GE1-v5...	untreated	treated	2	1
3	S01_GE1-v5...	treated	untreated	3	1
4	S01_GE1-v5...	untreated	treated	4	1

Background Correction and Normalisation

Offset: [✓] Max negative value   or set here:

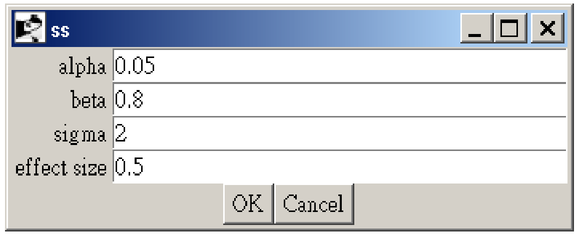
[✓] Log2 transformation   Normalisation method: quantile

Output

[✓] Box plot   [ ] Heat map

File type: jpeg

Number of randomly selected genes for clustering: 500

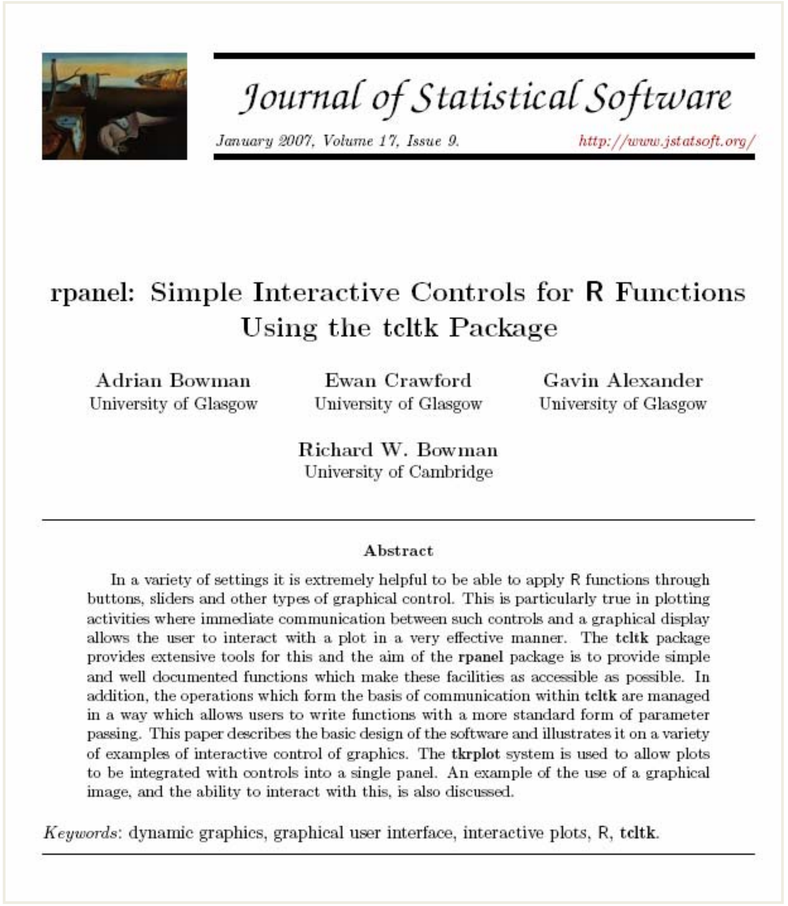

ss

alpha	0.05
beta	0.8
sigma	2
effect size	0.5

```
library(fgui)
ss <- function(alpha = 0.05, beta = 0.8, sigma = 2, effect_size = 0.5){
      ceiling((qnorm(1 - alpha / 2) + qnorm(1 - beta)) ^ 2 * sigma ^ 2 /effect_size ^ 2)
}
guiv(ss)
```

簡介 The rpanel package (1)

- Adrian Bowman, Ewan Crawford, Gavin Alexander, Richard W. Bowman, 2007, rpanel: Simple Interactive Controls for R Functions Using the tcltk Package, *Journal of Statistical Software*, January 2007, Volume 17, Issue 9.
- The rpanel package is built on **rtcltk** and manages the process of communication so that controls can be constructed directly by R simple function calls.



Journal of Statistical Software

January 2007, Volume 17, Issue 9. http://www.jstatsoft.org/

rpanel: Simple Interactive Controls for R Functions Using the tcltk Package

Adrian Bowman, University of Glasgow
Ewan Crawford, University of Glasgow
Gavin Alexander, University of Glasgow
Richard W. Bowman, University of Cambridge

Abstract

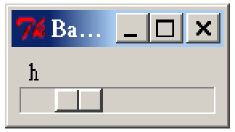
In a variety of settings it is extremely helpful to be able to apply R functions through buttons, sliders and other types of graphical control. This is particularly true in plotting activities where immediate communication between such controls and a graphical display allows the user to interact with a plot in a very effective manner. The **tcltk** package provides extensive tools for this and the aim of the **rpanel** package is to provide simple and well documented functions which make these facilities as accessible as possible. In addition, the operations which form the basis of communication within **tcltk** are managed in a way which allows users to write functions with a more standard form of parameter passing. This paper describes the basic design of the software and illustrates it on a variety of examples of interactive control of graphics. The **tkrplot** system is used to allow plots to be integrated with controls into a single panel. An example of the use of a graphical image, and the ability to interact with this, is also discussed.

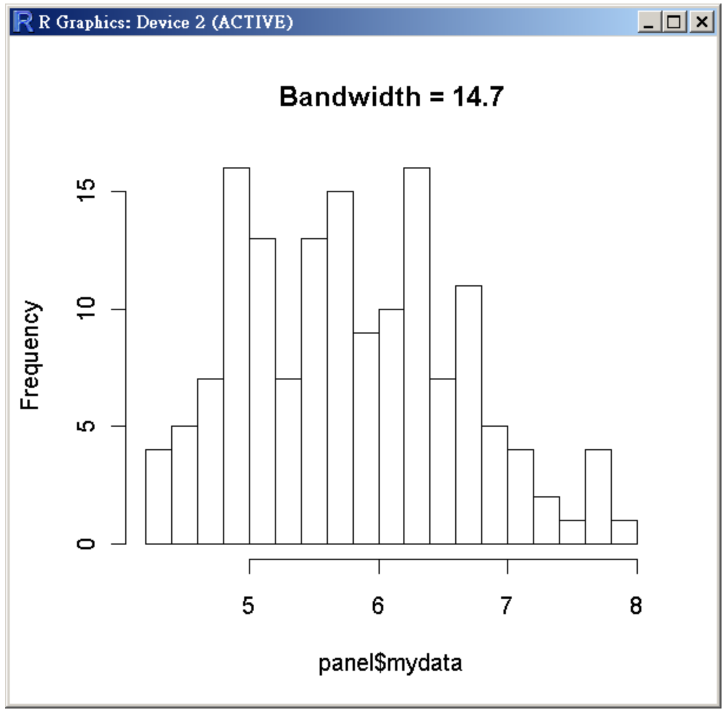
Keywords: dynamic graphics, graphical user interface, interactive plots, R, **tcltk**.

rpanel-paper-scripts.r

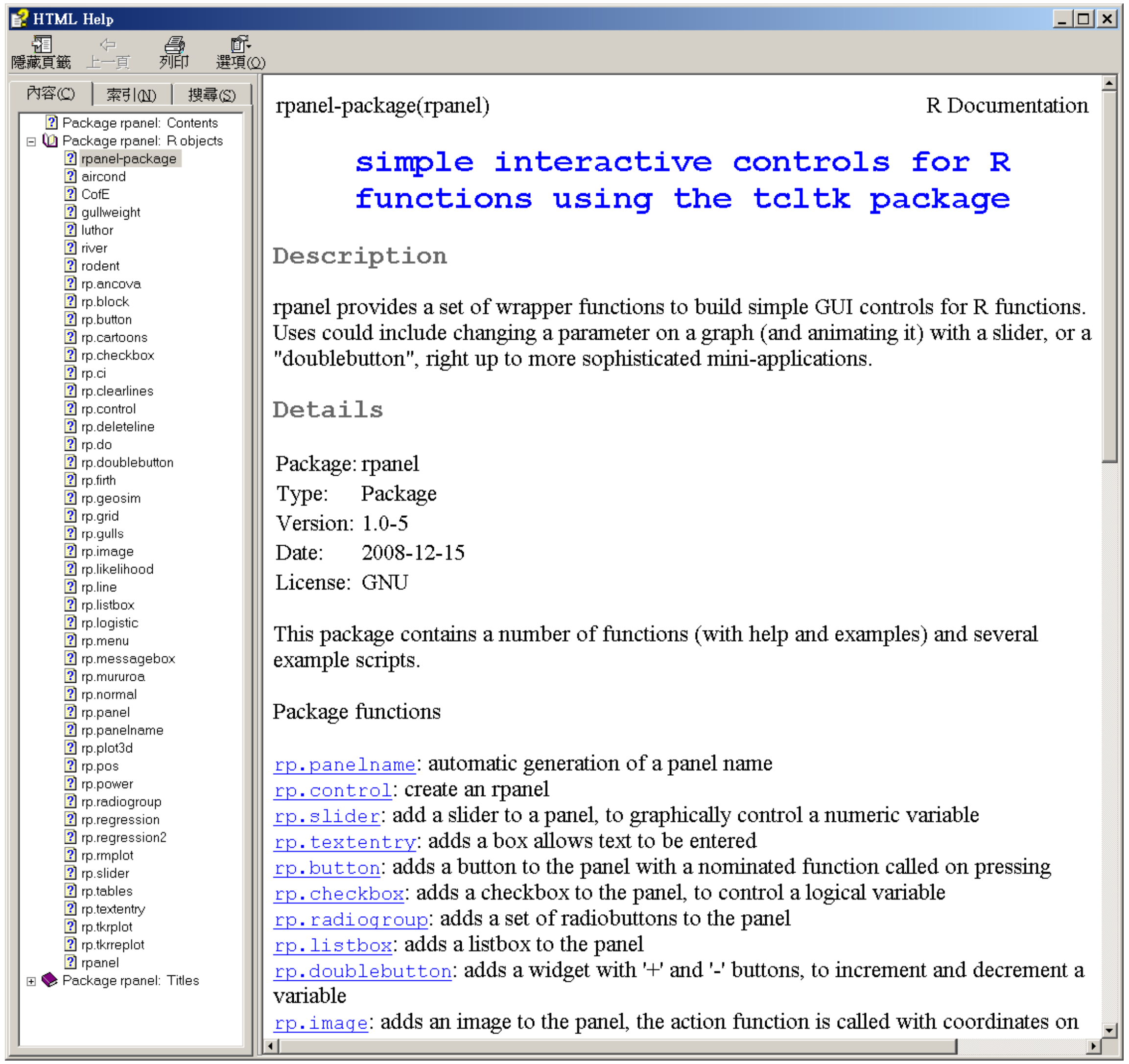
簡介 The rpanel package (2)

- **Aim 1**: Construction of GUI control panels for R applications.
- **Aim 2**: Use these gui construction tools to provide specific applications. (e.g., teaching of statistics)





?rpanel



rpanel-package(rpanel) R Documentation

simple interactive controls for R functions using the tcltk package

Description

rpanel provides a set of wrapper functions to build simple GUI controls for R functions. Uses could include changing a parameter on a graph (and animating it) with a slider, or a "doublebutton", right up to more sophisticated mini-applications.

Details

Package:	rpanel
Type:	Package
Version:	1.0-5
Date:	2008-12-15
License:	GNU

This package contains a number of functions (with help and examples) and several example scripts.

Package functions

`rp.panelname`: automatic generation of a panel name
`rp.control`: create an rpanel
`rp.slider`: add a slider to a panel, to graphically control a numeric variable
`rp.textentry`: adds a box allows text to be entered
`rp.button`: adds a button to the panel with a nominated function called on pressing
`rp.checkbox`: adds a checkbox to the panel, to control a logical variable
`rp.radiogroup`: adds a set of radiobuttons to the panel
`rp.listbox`: adds a listbox to the panel
`rp.doublebutton`: adds a widget with '+' and '-' buttons, to increment and decrement a variable
`rp.image`: adds an image to the panel, the action function is called with coordinates on

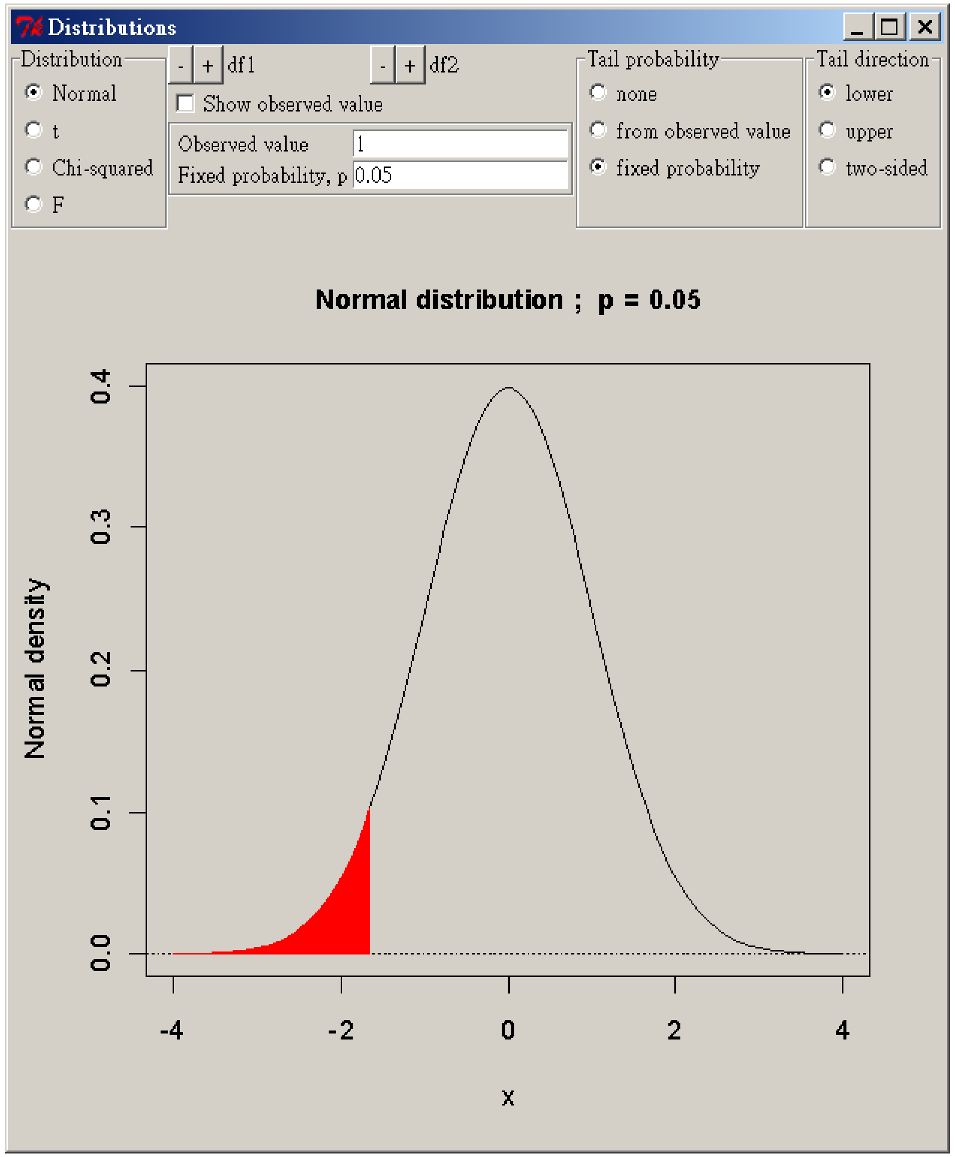
整合GUI設計範例:

Applications functions in rpanel

- **`rp.gulls()`**: STEPS module - the Birds and the Bees
- **`rp.ci()`**: Confidence intervals
- **`rp.ancova()`**: Analysis of covariance
- **`rp.power()`**: Power calculations for a two-sample t-test
- **`rp.normal()`**: Fitting a normal distribution to a single sample
- **`rp.rmplot()`**: Plotting of repeated measurement data
- **`rp.tables()`**: Interactive statistical tables
- **`rp.regression()`**: Regression with one or two covariates
- **`rp.plot3d()`**: Interactive display of a plot of three variables
- **`rp.likelihood()`**: Exploration of one and two parameter likelihood functions
- **`rp.logistic()`**: Interactive display of logistic regression with a single covariate
- **`rp.cartoons()`**: A menu-driven set of rpanel illustrations
- **`rp.geosim()`**: Simulation of spatial processes
- **`rp.mururoa()`**: Sampling in Mururoa Atoll
- **`rp.firth()`**: Sampling in a firth

```
# install.packages("rpanel")
> library(rpanel)
> rp.tables()
```

整合GUI設計範例 1: rp.tables()



Distribution: [✓ Normal] t Chi-squared F

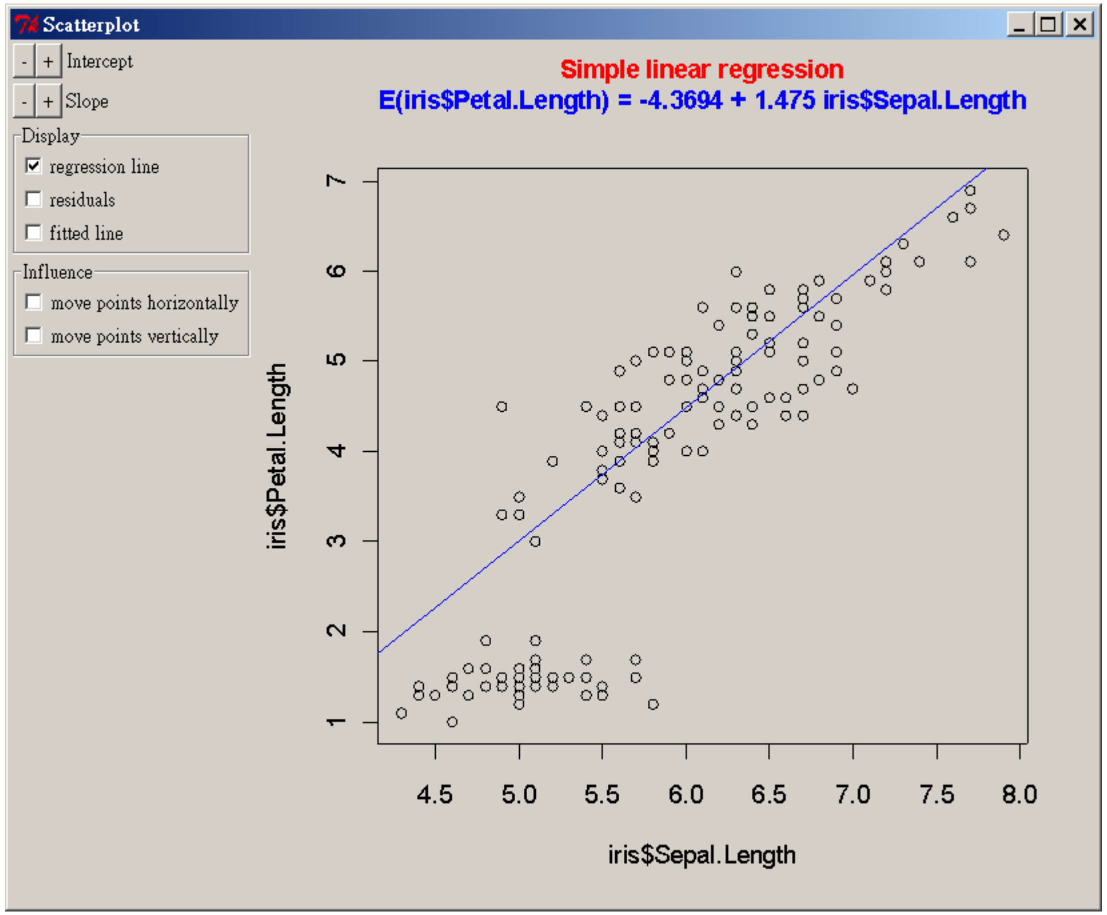
Observed value: 1
Fixed probability, p: 0.05

Tail probability: none from observed value [✓ fixed probability]

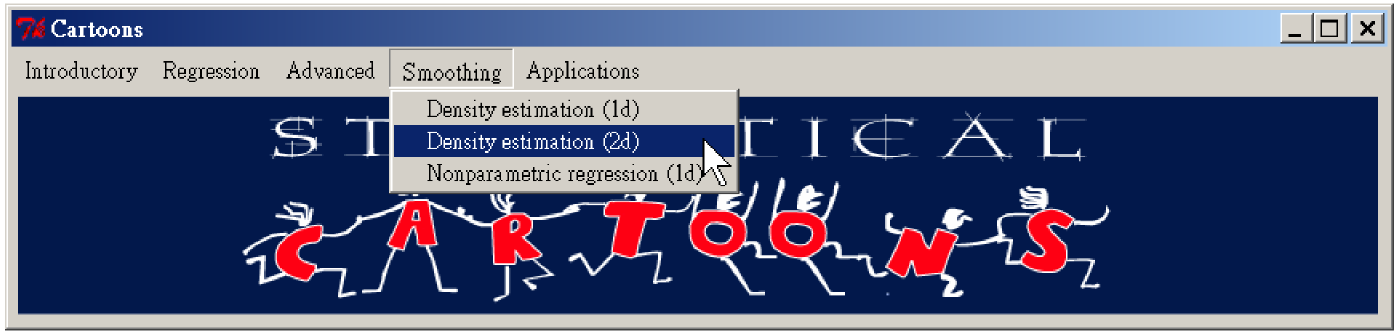
Tail direction: [✓ lower] upper two-sided

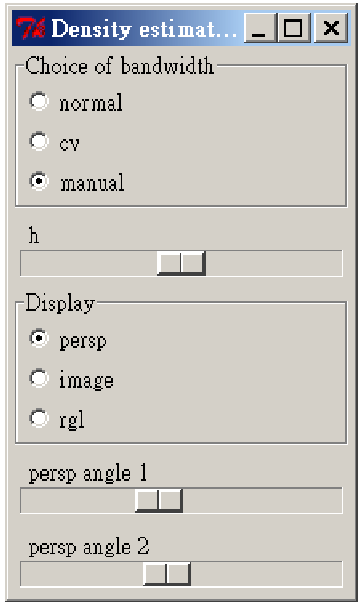
整合GUI設計範例 2: rp.regression()

```
rp.regression(iris$Sepal.Length, iris$Petal.Length)
```



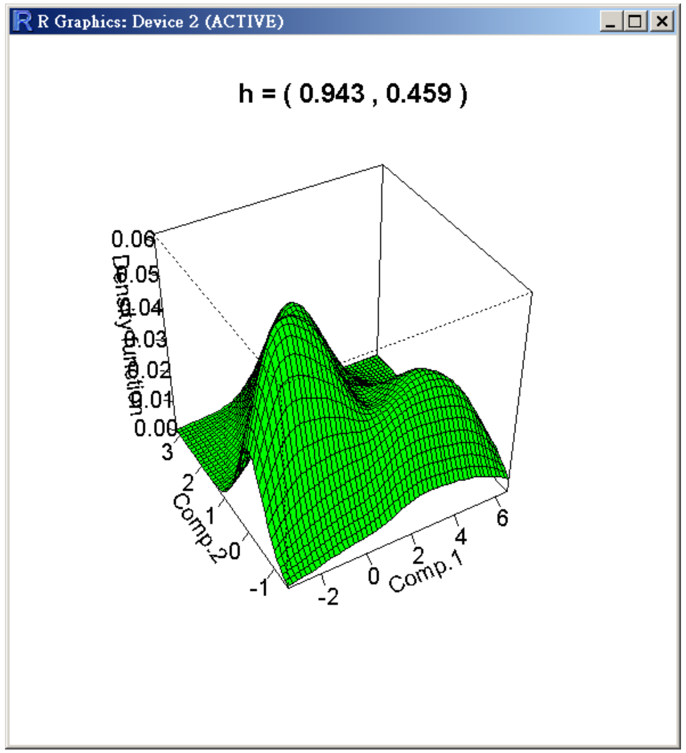
整合GUI設計範例 3: rp.cartoons()





Choice of bandwidth: ( ) normal ( ) cv [✓ manual]

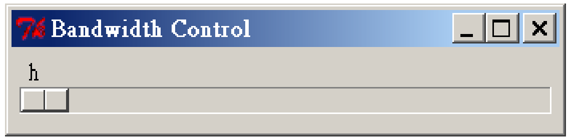
Display: [✓ persp] ( ) image ( ) rgl



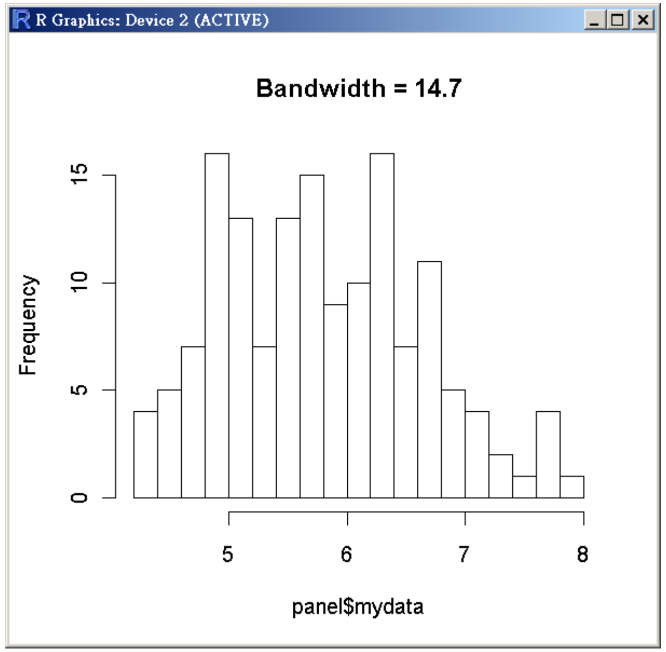
rp.slider

範例 1.1: 長條圖的帶寬 (bandwidth)

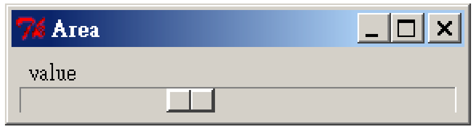
```r
# install.packages("rpanel")
library(rpanel)

my.draw <- function(panel) {
    hist(x = panel$mydata, breaks = panel$h,
        main = paste("Bandwidth =", round(panel$h, 2)))
    panel
}
my.panel <- rp.control("Bandwidth Control", mydata = iris$Sepal.Length, h = 4)
rp.slider(panel = my.panel, var = h, from = 4,  to = 50, action = my.draw)
```

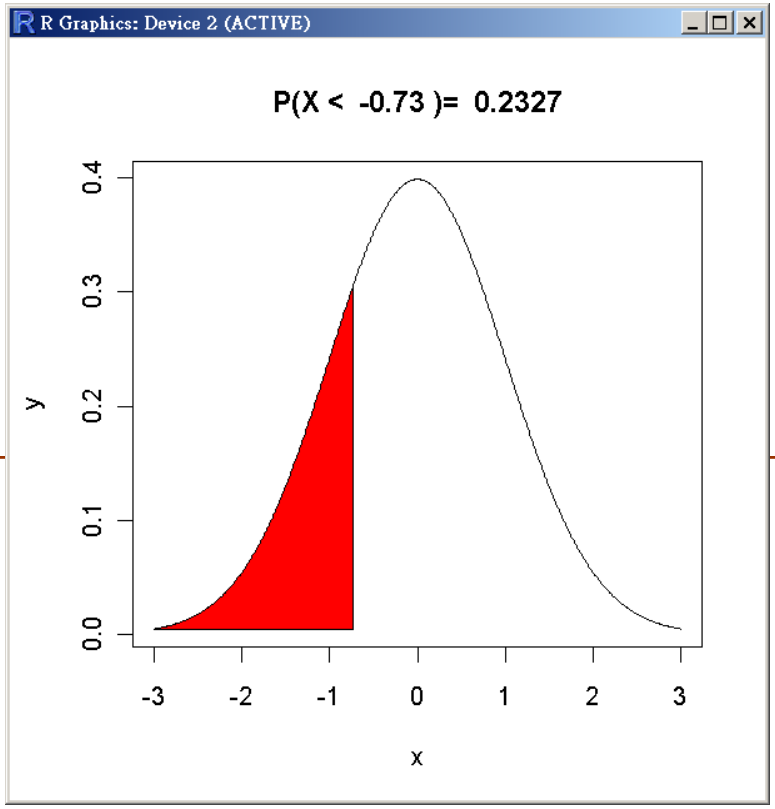


```
> str(my.panel)
List of 6
 $          : chr "window35"
 $ mydata   : num [1:150] 5.1 4.9 4.7 4.6 5 5.4
4.6 5 4.4 4.9 ...
 $ h        : num 4
 $ .handle  :List of 2
  ..$ ID : chr ".9"
  ..$ env:<environment: 0x000000004e41c6d8>
  ..- attr(*, "class")= chr "tkwin"
 $ .type    : chr "window"
 $ panelname: chr "window35"
```



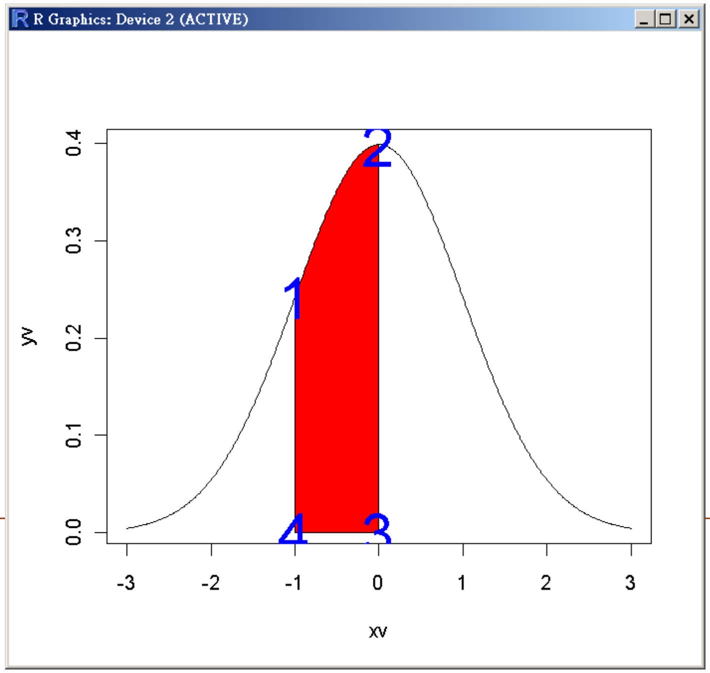
範例 1.2: 常態分佈機率圖





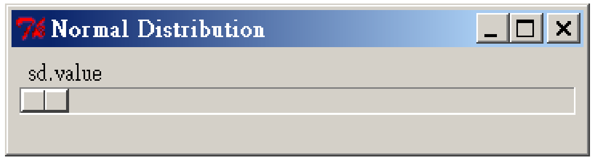
```
xv <- seq(-3, 3, 0.01)
yv <- dnorm(xv)
xyv <- cbind(xv, yv)

my.draw <- function(panel) {
    x <- panel$mydata[,1]
    y <- panel$mydata[,2]
    v <- panel$value
    p <- round(pnorm(v), 4)
    plot(x, y, type="l", main=paste("P(X < ", v, ")= ", p))
    polygon(c(x[x <= v], v), c(y[x <= v], y[x == -3]), col="red")
    panel
}
my.panel <- rp.control("Area", mydata = xyv, value = -1)
rp.slider(panel = my.panel, var = value, from = -3,  to = 3, action = my.draw)
```

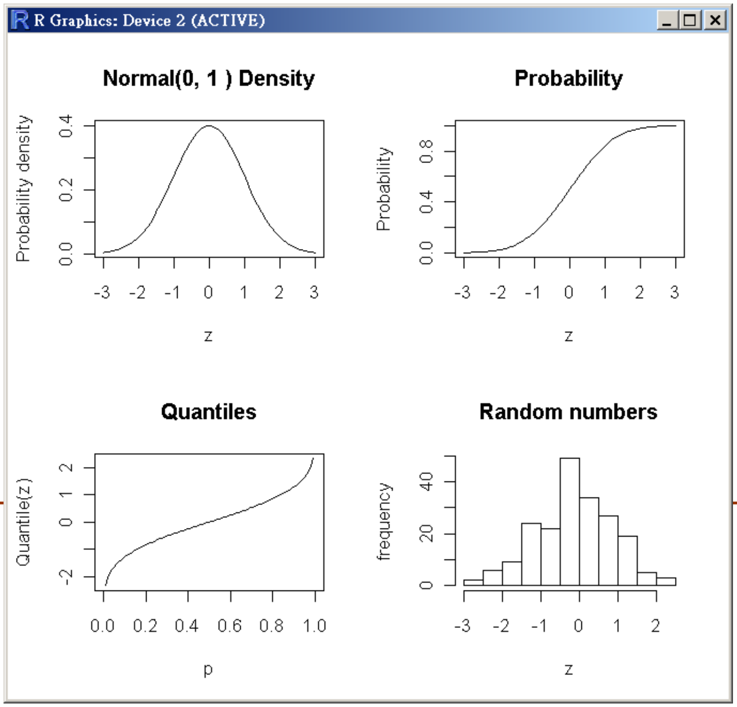
polygon()



```
xv <- seq(-3, 3, 0.01)
yv <- dnorm(xv)
plot(xv, yv, type="l")
id <- (xv <= 0) & (xv >= -1)
polygon(c(xv[id], 0, -1), c(yv[id], 0, 0), col="red")

points(xv[id][1],yv[id][1], pch="1", col="blue", cex=3)
points(xv[id][length(xv[id])],yv[id][length(xv[id])], pch="2",
   col="blue", cex=3)
points(0,0, pch="3", col="blue", cex=3)
points(-1,0, pch="4", col="blue", cex=3)
```

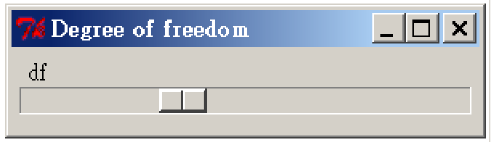
範例 1.3: 常態分佈，不同的變異數

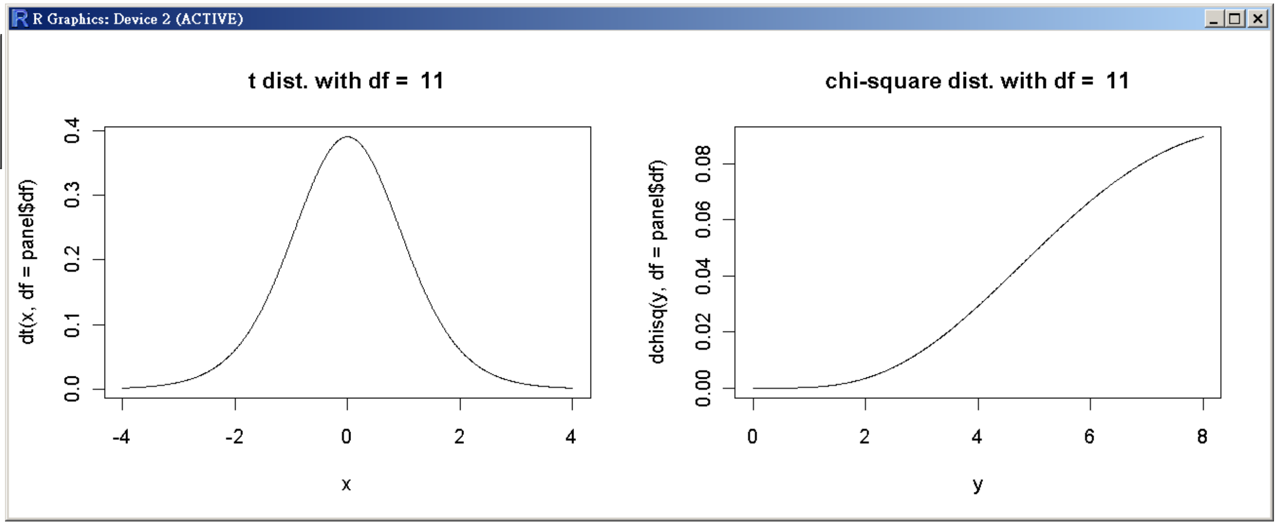




```
par(mfrow=c(2,2))

my.draw <- function(panel) {
    n <- 200
    m <- 0
    s <- panel$sd.value
    curve(dnorm(x, m, s), -3, 3, xlab="z", ylab="Probability density",
        main=paste("Normal(0,",s*s,") Density"))
    curve(pnorm(x, m, s), -3, 3, xlab="z", ylab="Probability", main="Probability")
    curve(qnorm(x, m, s), 0, 1, xlab="p", ylab="Quantile(z)", main="Quantiles")
    hist(rnorm(n, m, s), xlab="z", ylab="frequency", main="Random numbers")
    panel
}

my.panel <- rp.control("Normal Distribution", sd.value = 1)
rp.slider(panel = my.panel, var = sd.value, from = 1,  to = 10, action = my.draw)
```

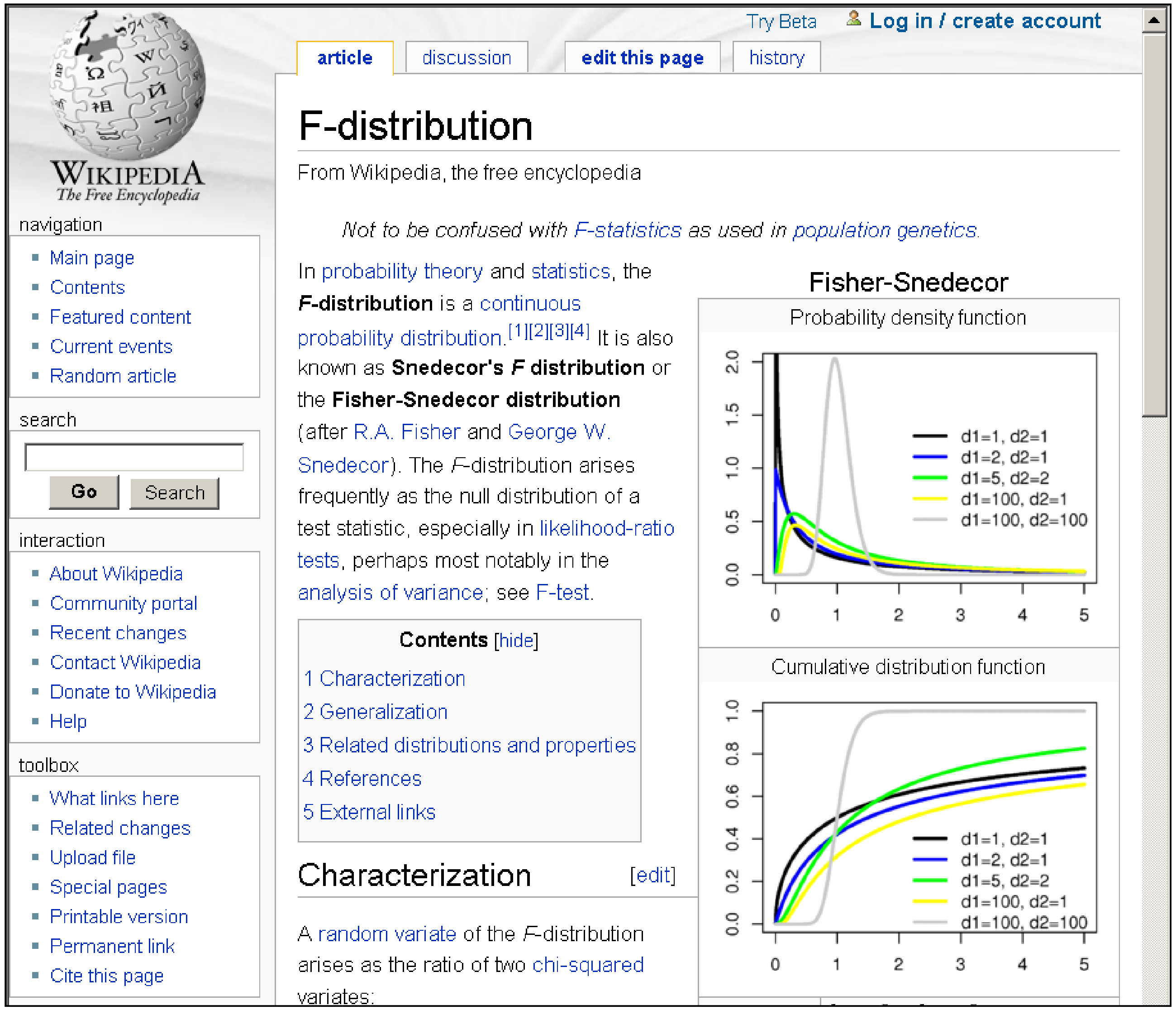
範例 1.4: t and chi-square 分佈





```
par(mfrow=c(1,2))
my.draw <- function(panel) {
    x <- seq(-4, 4, 0.01)
    y <- seq(0, 8, 0.01)
    plot(x, dt(x, df = panel$df), type = "l",
        main = paste("t dist. with df = ", panel$df))
    plot(y, dchisq(y, df = panel$df), type = "l",
        main = paste("chi-square dist. with df = ", panel$df))
    panel
}
my.panel <- rp.control("Degree of freedom", df = 1)
rp.slider(panel = my.panel, var = df, from = 1,  to = 30, action = my.draw)
```

範例 1.5: F 分佈 (1)



F-distribution

From Wikipedia, the free encyclopedia

Not to be confused with F-statistics as used in population genetics.

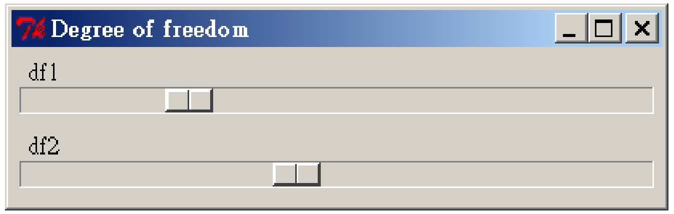
In probability theory and statistics, the ***F*-distribution** is a continuous probability distribution.[1][2][3][4] It is also known as **Snedecor's *F* distribution** or the **Fisher-Snedecor distribution** (after R.A. Fisher and George W. Snedecor). The *F*-distribution arises frequently as the null distribution of a test statistic, especially in likelihood-ratio tests, perhaps most notably in the analysis of variance; see F-test.

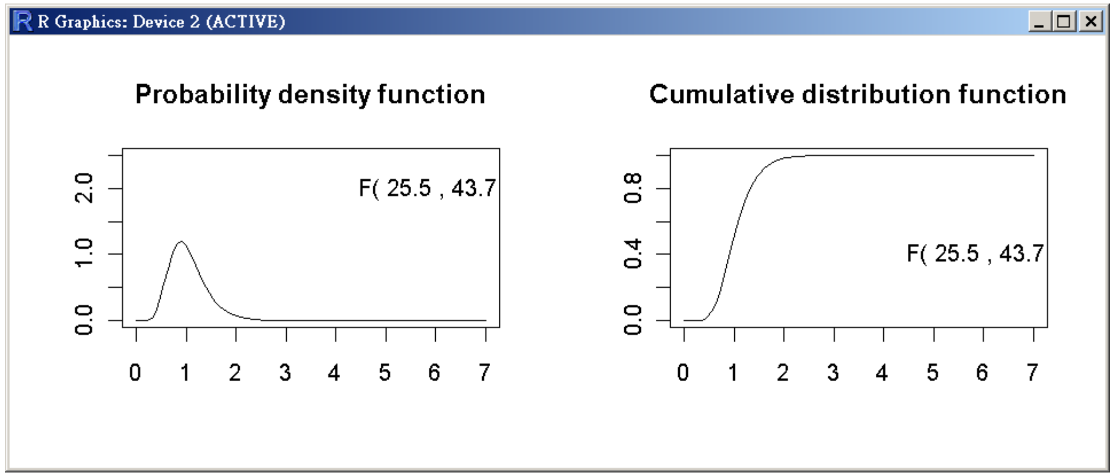
Contents [hide]

Characterization [edit]

A random variate of the *F*-distribution arises as the ratio of two chi-squared variates:

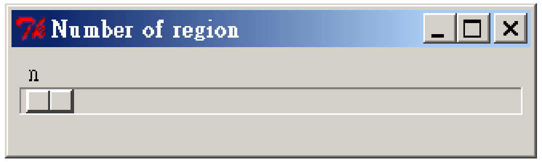
範例 1.5: F 分佈 (2)

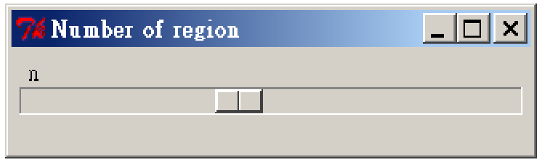


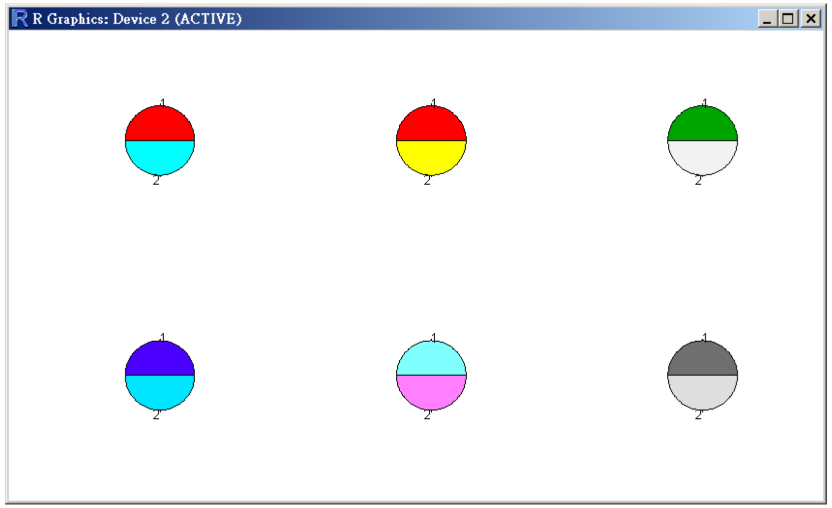


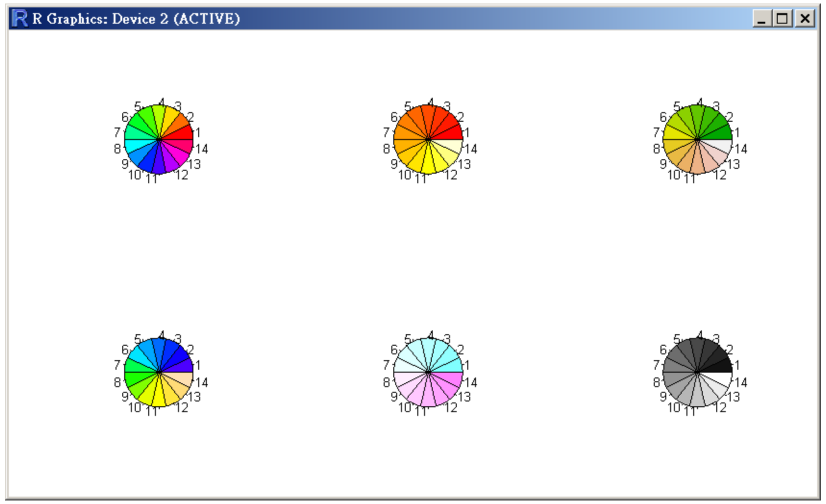
```
par(mfrow=c(1,2))
my.draw <- function(panel) {
    df1 <- panel$df1
    df2 <- panel$df2
    curve(df(x, df1, df2), 0, 7, xlab="", ylab="",
        main="Probability density function", ylim = c(0, 2.5))
    text(6, 2, paste("F(",df1,",",df2,")"))
    curve(pf(x, df1, df2), 0, 7, xlab="", ylab="",
        main="Cumulative distribution function", ylim = c(0, 1))
    text(6, 0.4, paste("F(",df1,",",df2,")"))
    panel
}
my.panel <- rp.control("Degree of freedom", df1 = 1, df2 = 1)
rp.slider(panel = my.panel, var = df1, from = 1,  to = 100, action = my.draw)
rp.slider(panel = my.panel, var = df2, from = 1,  to = 100, action = my.draw)
```

課堂練習 1.1: 餅圖





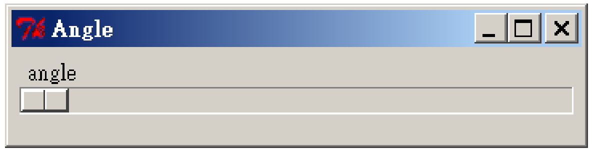


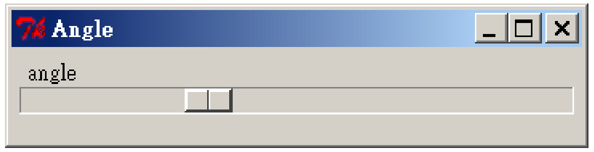


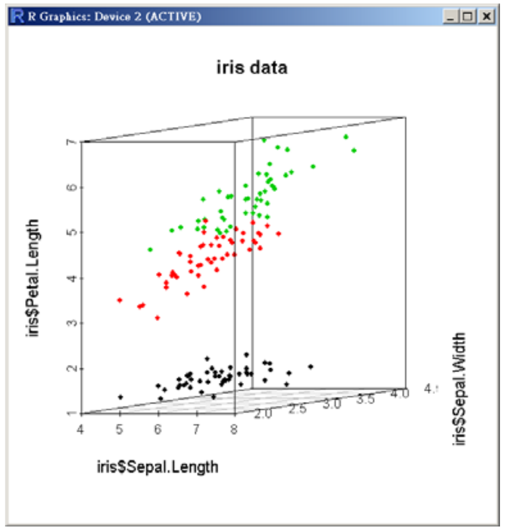
提示:

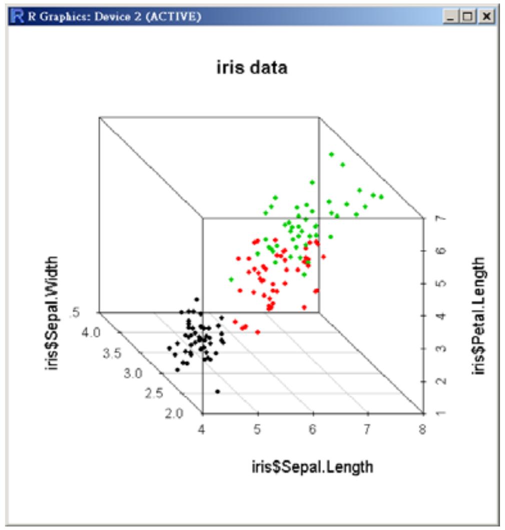
```
? rainbow
? pie

par(mfrow=c(2,3))
my.draw <- function(panel) {
    ...
}
my.panel <- rp.control(...)
rp.slider(...)
```

課堂練習 1.2: 3D-scatterplot





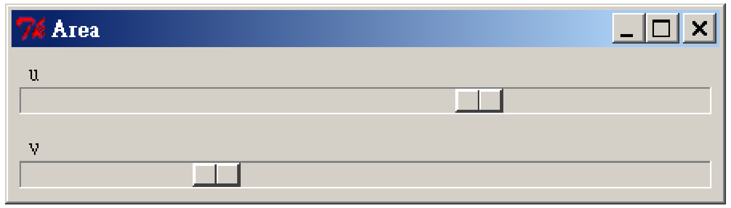


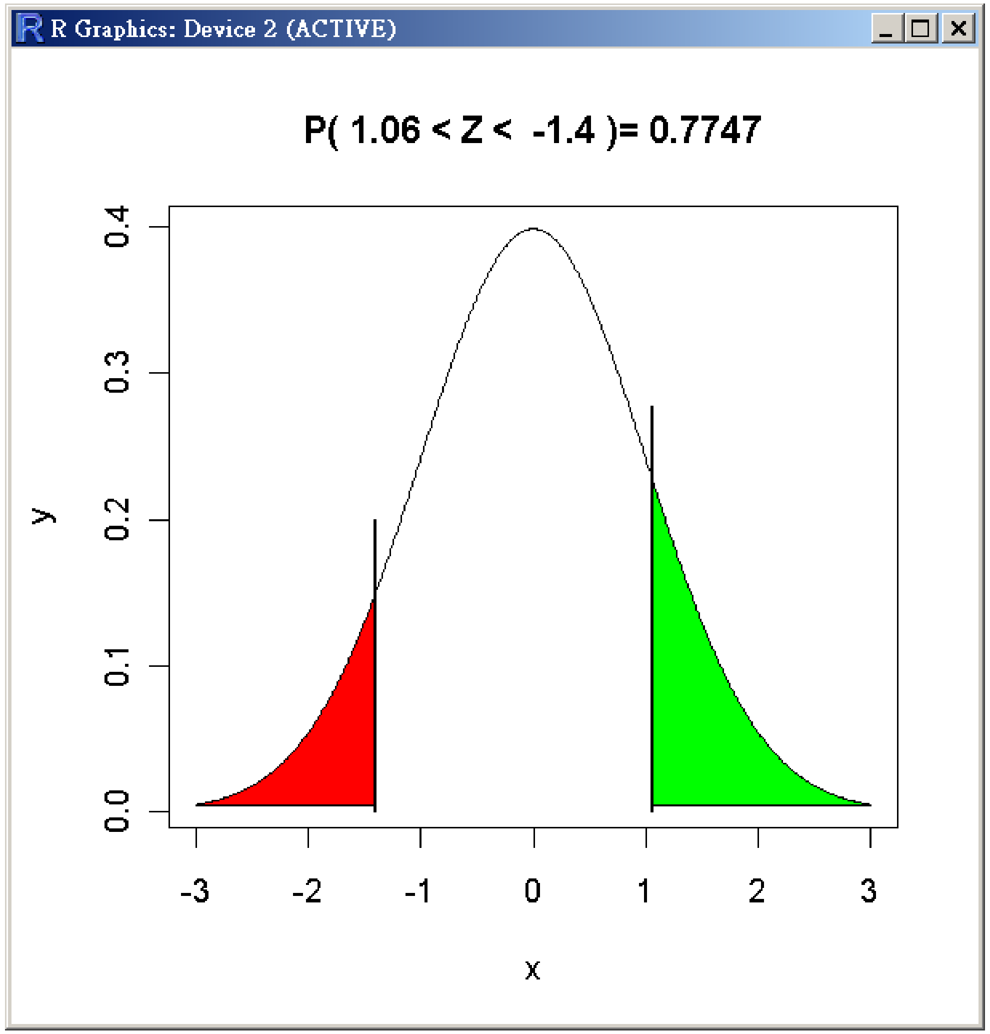


提示:

```
library(scatterplot3d)

my.draw <- function(panel){
    scatterplot3d(..., color=as.integer(iris$Species), ...)
...
}
my.panel <- rp.control(...)
rp.slider(...)
```

課堂練習 1.3: 常態分佈

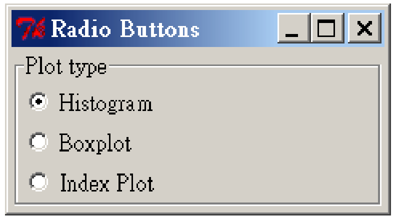


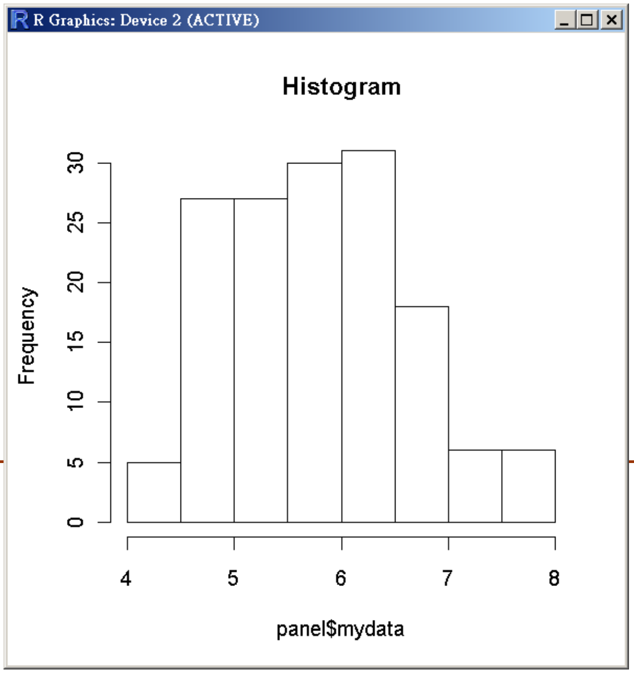


提示:

```
xv <- seq(-3, 3, 0.01)
yv <- dnorm(xv)
xyv <- cbind(xv, yv)

my.draw <- function(panel) {
    ...
    polygon(...)
    segments(...)
    polygon(...)
    segments(...)
    panel
}
my.panel <- rp.control(...)
rp.slider(...)
rp.slider(...)
```

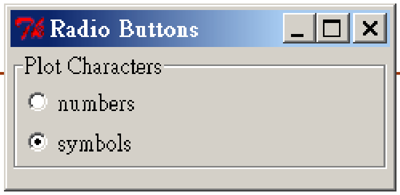
範例 2.1: rp.radiogroup



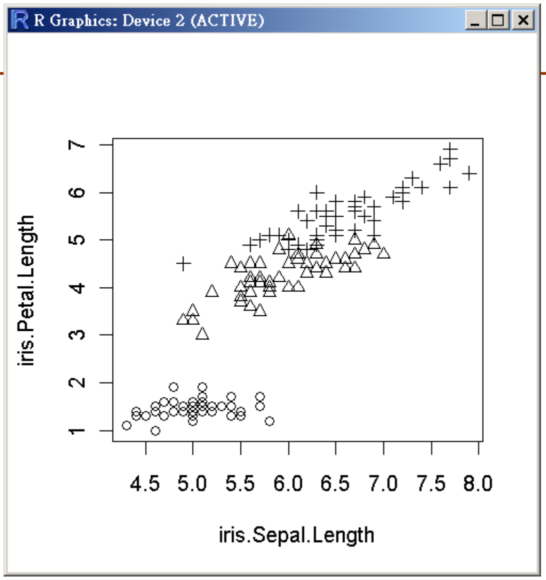


```
my.draw <- function(panel) {
     plot.title <- panel$plot.type
     if (panel$plot.type == "Histogram"){
       hist(panel$mydata, main = plot.title)
     }else if (panel$plot.type == "Boxplot"){
         boxplot(panel$mydata, main = plot.title)
     }else{
         plot(panel$mydata, main = plot.title)
     }
     panel
}
my.panel <- rp.control(title = "Radio Buttons", mydata = iris$Sepal.Length)
rp.radiogroup(panel = my.panel, var = plot.type,
    values = c("Histogram", "Boxplot", "Index Plot"),
    action = my.draw, title = "Plot type")
rp.do(my.panel, my.draw)
```

範例 2.2: rp.radiogroup



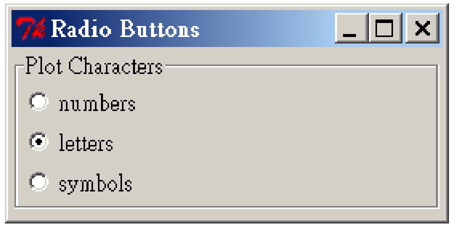
Plot Characters: numbers [✓ symbols]



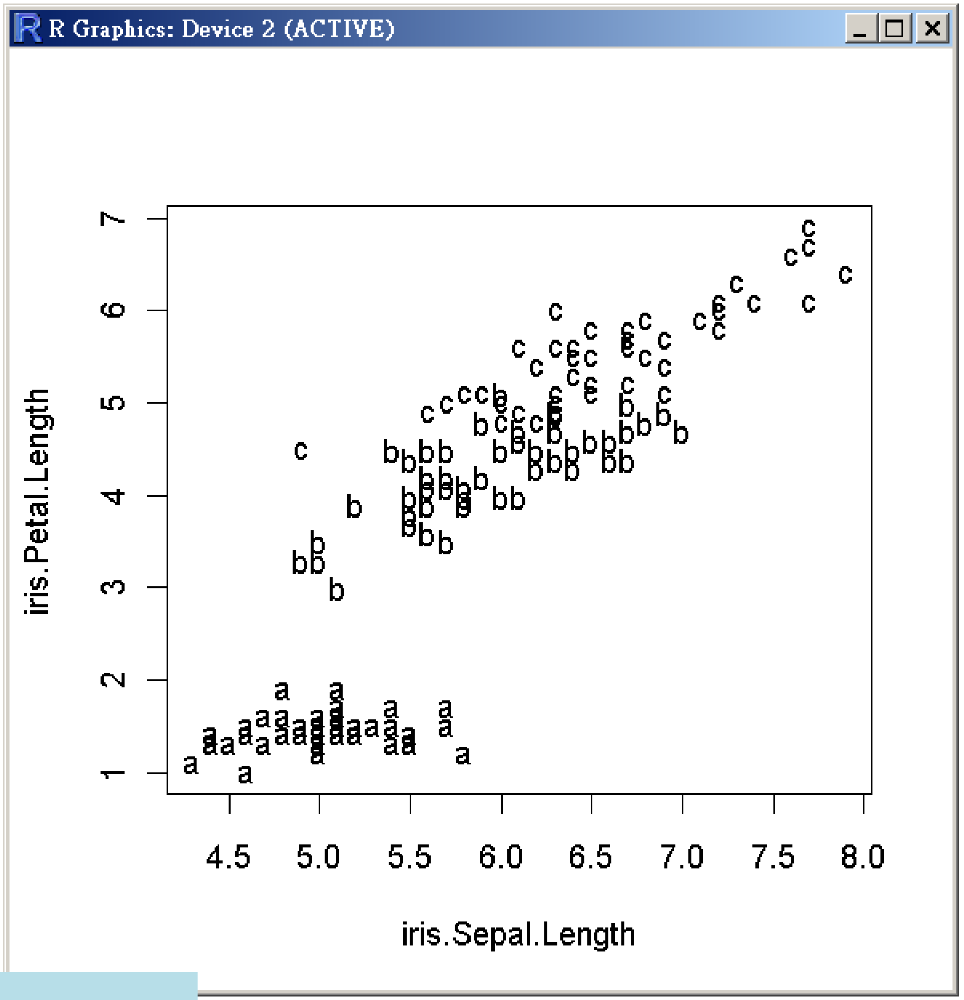
```
my.draw <- function(panel) {
    mydata <- panel$mydata
    x <- panel$mydata[,1]
    y <- panel$mydata[,2]
    plot(x, y, xlab = names(mydata)[1],
        ylab = names(mydata)[2], type="n")
    if(panel$plot.pch == "numbers"){
        my.label <- panel$group
        text(x, y, labels = my.label)
    }else if(panel$plot.pch == "symbols"){
        my.label <- panel$group
        points(x, y, pch = my.label)
    }
    panel
}
my.panel <- rp.control(title = "Radio Buttons",
    mydata = data.frame(iris$Sepal.Length, iris$Petal.Length),
    group = as.integer(iris$Species))
rp.radiogroup(panel = my.panel, var = plot.pch,
    values = c("numbers", "symbols"),
    action = my.draw, title = "Plot Characters")
rp.do(my.panel, my.draw)
```

課堂練習 2.1: rp.radiogroup

add letters

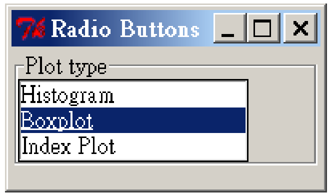


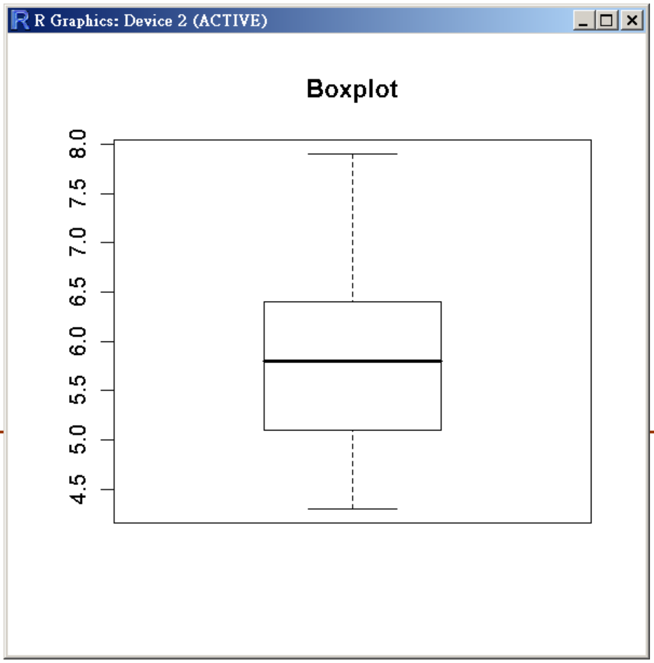
Plot Characters: numbers [✓ letters] symbols



hint:
```
my.label <- letters[panel$group]
```

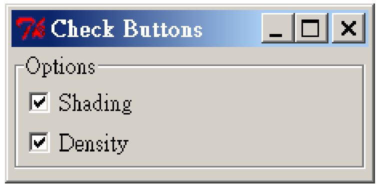
範例 3: rp.listbox

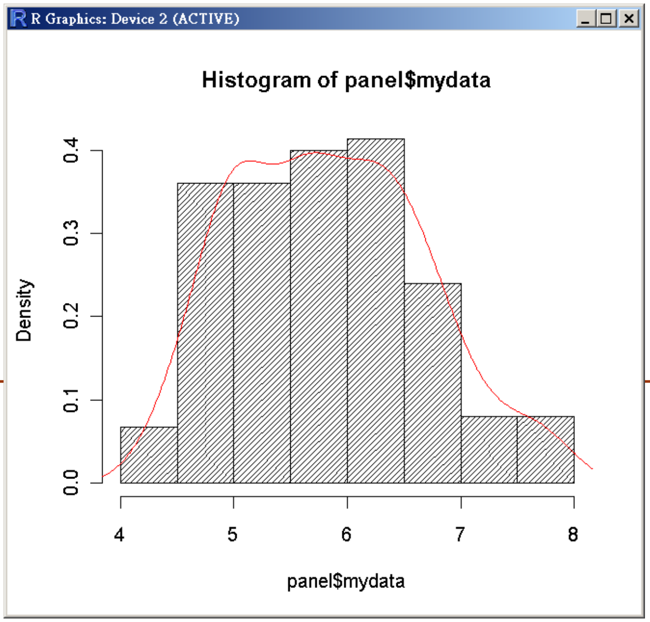




```
my.draw <- function(panel) {
     plot.title <- panel$plot.type
     if (panel$plot.type == "Histogram"){
       hist(panel$mydata, main = plot.title)
     }else if (panel$plot.type == "Boxplot"){
         boxplot(panel$mydata, main = plot.title)
     }else{
         plot(panel$mydata, main = plot.title)
     }
     panel
}
my.panel <- rp.control(title = "Radio Buttons", mydata = iris$Sepal.Length)
rp.listbox(panel = my.panel, var = plot.type,
    vals = c("Histogram", "Boxplot", "Index Plot"),
    action = my.draw, title = "Plot type")
rp.do(my.panel, my.draw)
```

範例 4: rp.checkbox

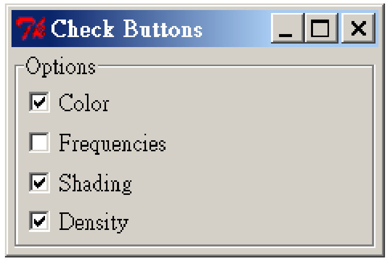




```
my.draw <- function(panel) {

    is.shading <- NULL
    if (panel$options[1]){
       is.shading <- 30
    }
    hist(panel$mydata, freq = F, density = is.shading)
    if (panel$options[2]){
       lines(density(panel$mydata), col = "red")
    }
    panel
}
my.panel <- rp.control(title = "Check Buttons", mydata =  iris$Sepal.Length)
rp.checkbox(panel = my.panel, var = options, action = my.draw,
      labels = c("Shading", "Density"), title = "Options")
rp.do(my.panel, my.draw)
```

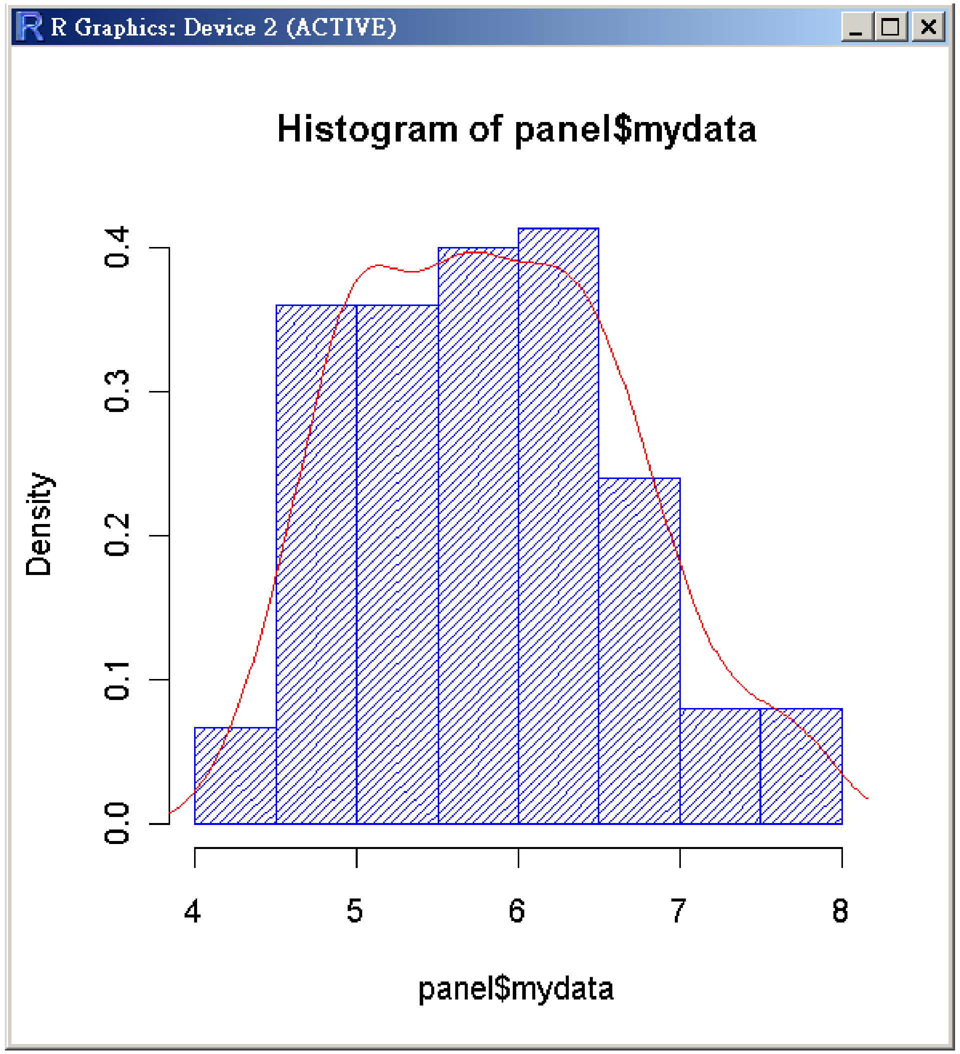
課堂練習 4.1: rp.checkbox

more checkbox


Check Buttons

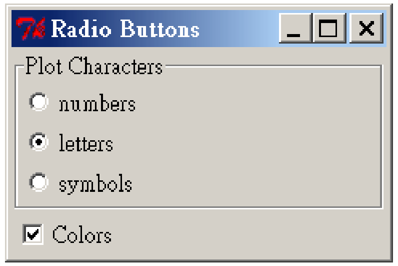
Options

- [✓] Color
- [ ] Frequencies
- [✓] Shading
- [✓] Density



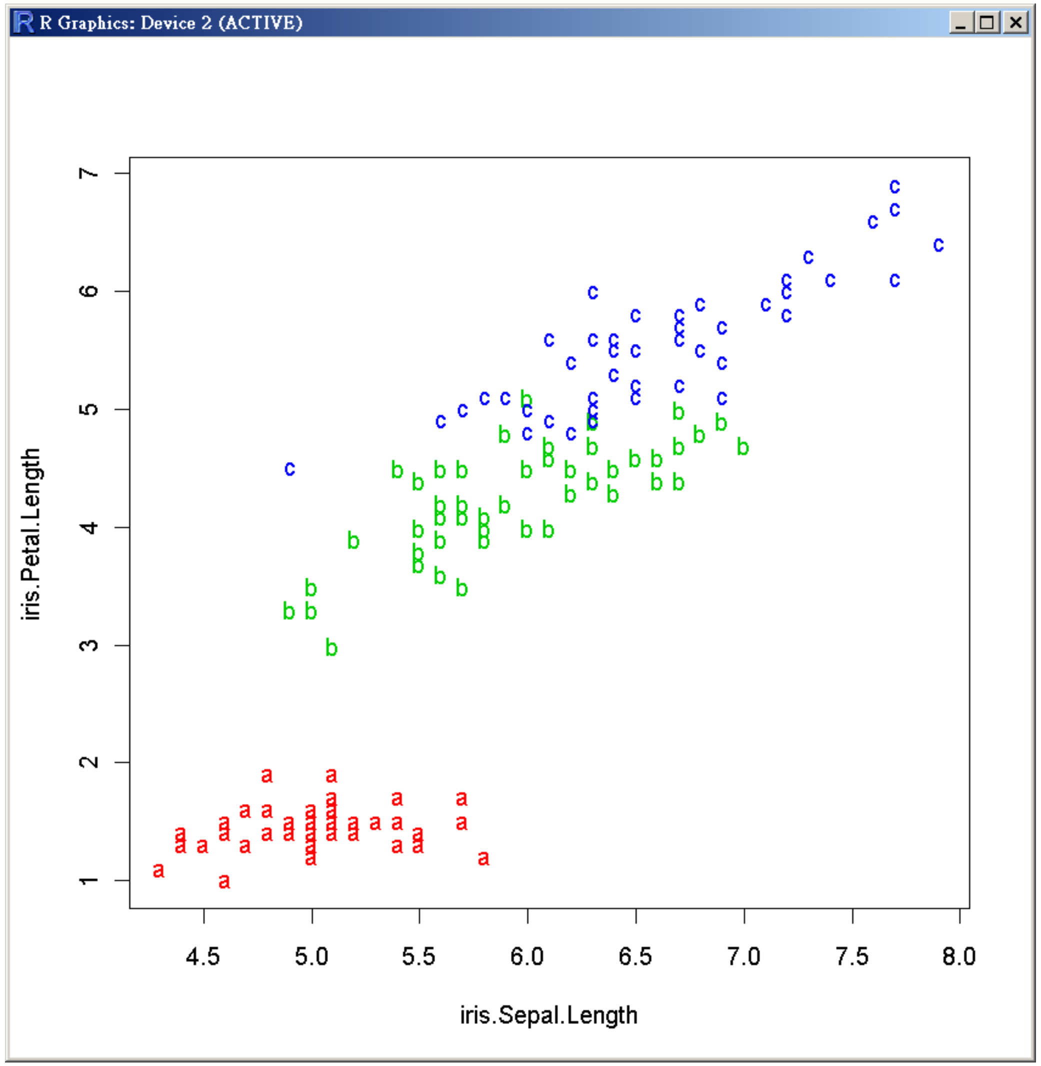
課堂練習 4.2: rp.radiogroup & rp.checkox

colors


Radio Buttons

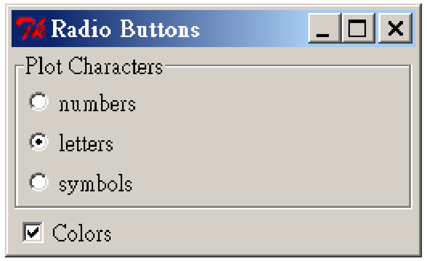
Plot Characters
- [ ] numbers
- [✓] letters
- [ ] symbols

[✓] Colors



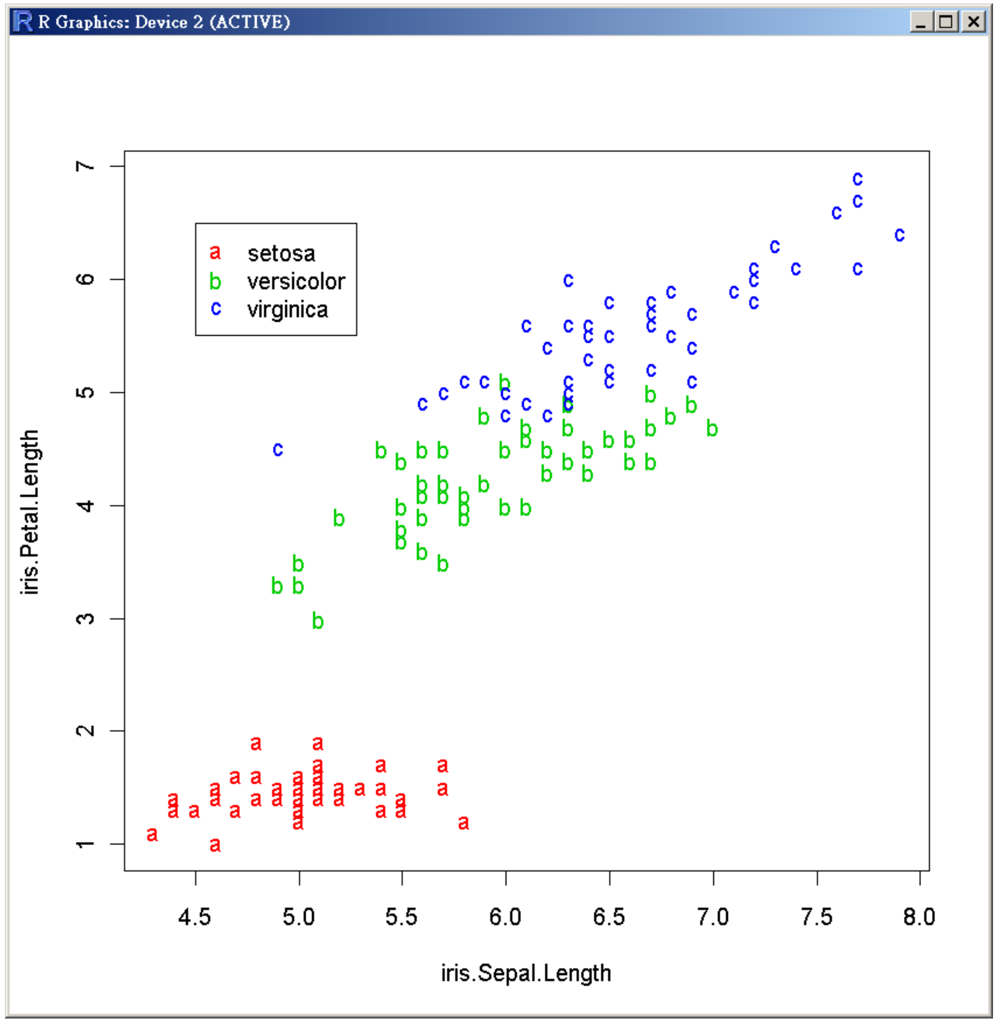
課堂練習 4.3: rp.radiogroup & rp.checkox

Legend


Radio Buttons

Plot Characters
- [ ] numbers
- [✓ letters]
- [ ] symbols

[✓ Colors]



範例 5: rp.doublebutton: 二元常態機率密度函數 (1)

The function $f(x,y)$ with the quadratic form $Q(x,y)$ gives the joint density function of a bivariate normal distribution.

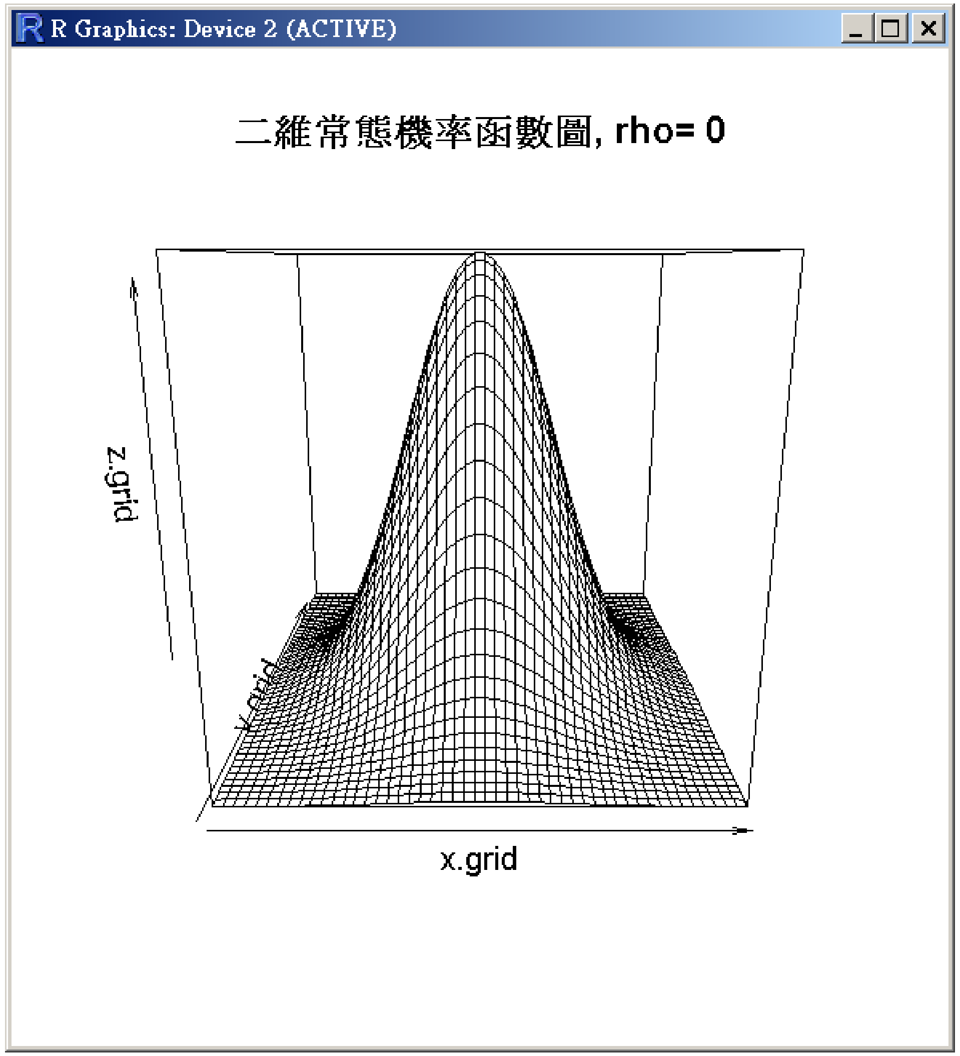
$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\{-\frac{1}{2}\, Q(x,y))\}$$

where

$$Q(x,y) = \frac{1}{1-\rho^2}[(\frac{x-\mu_x}{\sigma_x})^2 - 2\rho(\frac{x-\mu_x}{\sigma_x})(\frac{y-\mu_y}{\sigma_y}) + (\frac{y-\mu_y}{\sigma_y})^2]$$

```
mu.x <- 0
mu.y <- 0
sigma.x <- 1
sigma.y <- 1
rho <- 0

Q <- function(x, y){
    s.x <- (x-mu.x)/sigma.x
    s.y <- (y-mu.y)/sigma.y
    return(s.x^2 - 2 * rho * s.x * s.y + s.y^2)
}
f <- function(x, y){
    a <- 2 * pi * sigma.x * sigma.y * sqrt(1-rho^2)
    return(a * exp(-0.5* Q(x, y)))
}

x.grid <- seq(-3, 3, length=50)
y.grid <- seq(-3, 3, length=50)
z.grid <- outer(x.grid, y.grid, FUN = f)
my.title <- paste("二維常態機率函數圖,", "rho=", round(rho, 2))
persp(x.grid, y.grid, z.grid, main= my.title)
```

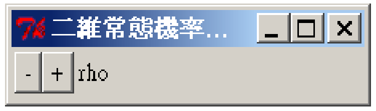
範例 5: rp.doublebutton: 二元常態機率密度函數 (2)

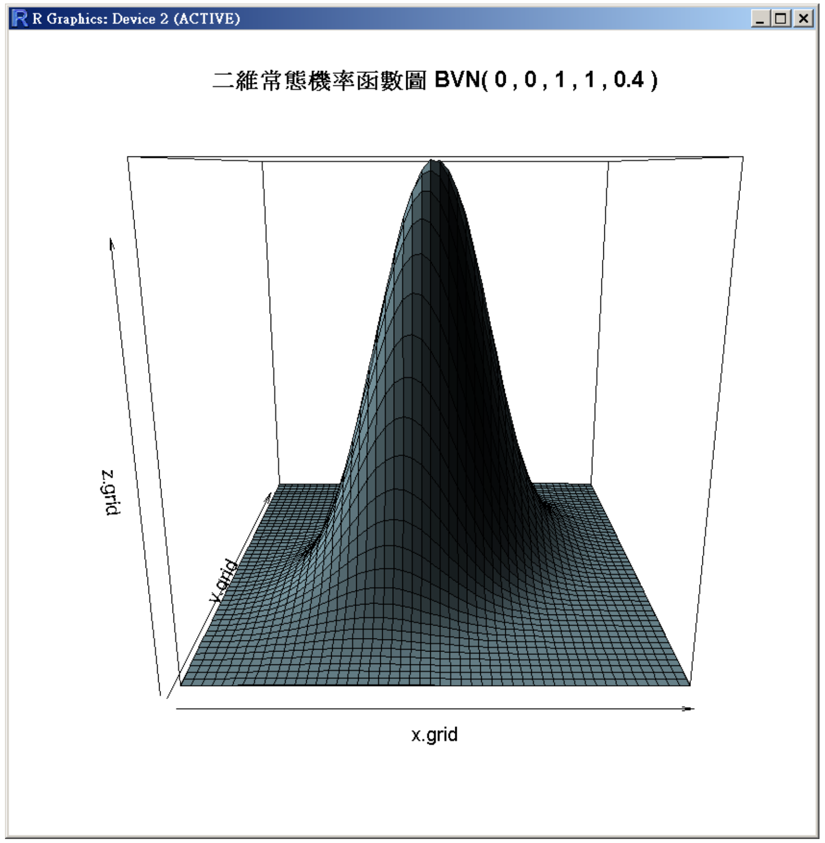


範例 5: rp.doublebutton: 二元常態機率密度函數 (3)

```
my.draw <- function(panel) {
    mu.x <- 0; mu.y <- 0
    sigma.x <- 1; sigma.y <- 1
    rho <- panel$rho

    Q <- function(x, y){
        s.x <- (x-mu.x)/sigma.x
        s.y <- (y-mu.y)/sigma.y
        return(s.x^2 - 2 * rho * s.x * s.y + s.y^2)
    }
    f <- function(x, y){
        a <- 2 * pi * sigma.x * sigma.y * sqrt(1-rho^2)
        return(a * exp(-0.5* Q(x, y)))
    }
    x.grid <- seq(-4, 4, length=50)
    y.grid <- seq(-4, 4, length=50)
    z.grid <- outer(x.grid, y.grid, FUN = f)
    my.title <- paste("二維常態機率函數圖", "BVN(",
        round(mu.x, 2), ",", round(mu.y, 2), ",",
        round(sigma.x^2, 2), ",", round(sigma.y^2, 2), ",",
        round(rho, 2), ")")
    persp(x.grid, y.grid, z.grid, main= my.title,
        col = "lightblue", shade = 0.75)
    panel
}
```

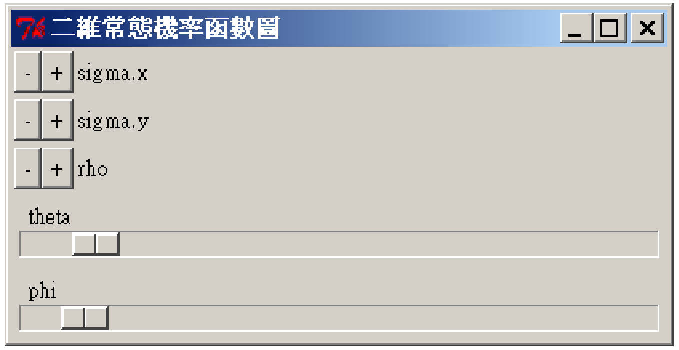
範例 5: rp.doublebutton: 二元常態機率密度函數 (4)

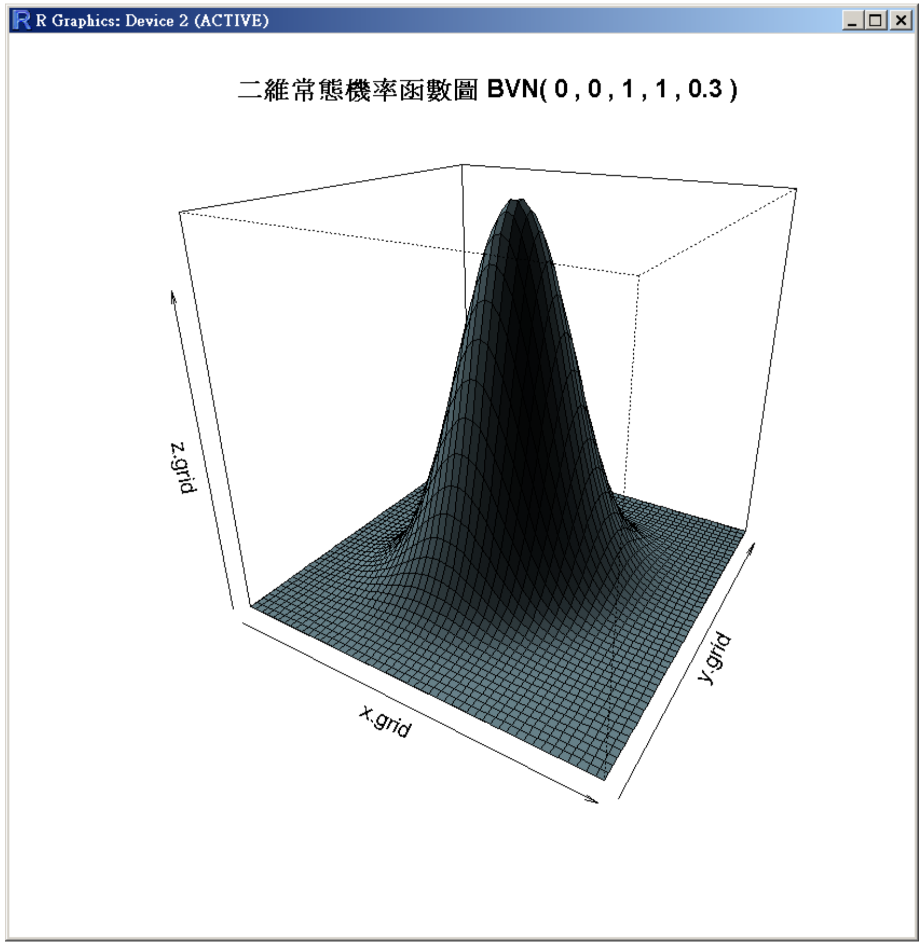




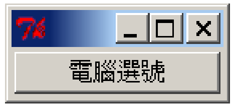
```
my.panel <- rp.control(title = "二維常態機率函數圖", rho = 0)
rp.doublebutton(panel = my.panel, var = rho, step = 0.1, range = c(-1, 1),
    title = "rho", action = my.draw)
rp.do(my.panel, my.draw)
```

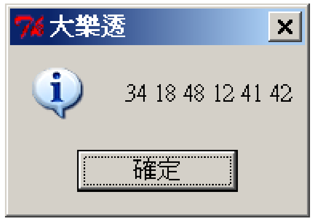
課堂練習 5: 二元常態機率密度函數

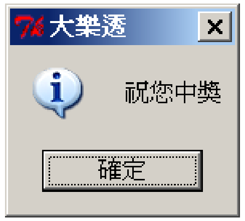




範例 6, 範例 7: rp.button & rp.messagebox

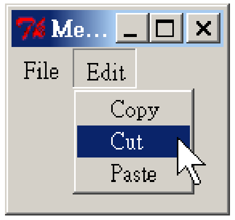


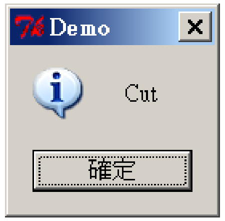




```
my.fun <- function(panel) {
    numbers <- sample(1:49, 6, replace = FALSE)
    rp.messagebox(numbers, title = "大樂透")
    rp.messagebox("祝您中獎", title = "大樂透")
    panel
}
my.panel <- rp.control()
rp.button(panel = my.panel, action = my.fun, title = "電腦選號")
```

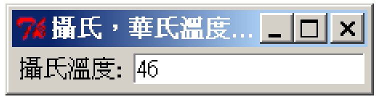
範例 8: rp.menu & rp.messagebox



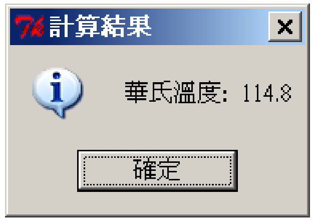


```
my.menu <- function(panel) {
    rp.messagebox(panel$menu, title = "Demo")
    panel
}
my.panel <- rp.control(title = "Menu Demo")
rp.menu(panel = my.panel, var = menu,
    labels = list(list("File", "Quit"),
        list("Edit", "Copy", "Cut", "Paste")),
    action = my.menu)
```

範例 9: rp.textentry


攝氏，華氏溫度...

攝氏溫度: 46


計算結果

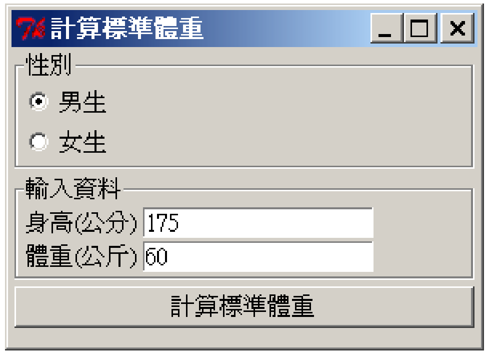
華氏溫度: 114.8

```
my.fun <- function(panel) {
    ctemp <- as.numeric(panel$ctemp)
    ftemp = (ctemp*9/5) + 32
    rp.messagebox(paste("華氏溫度: ", ftemp),  title = "計算結果")
    panel
}
my.panel <- rp.control(title = "攝氏，華氏溫度換算")
rp.textentry(panel = my.panel, var = ctemp,
    labels = "攝氏溫度: ", initval = 23, action = my.fun)
```

課堂練習 9:

世界衛生組織計算標準體重之方法

- 男性： 標準體重 = 0.7×身高cm - 56
- 女性： 標準體重 = 0.6×身高cm - 42
  - 標準體重正負10％為「體重正常」
  - 標準體重正負10％～20％為「體重過重」或「過輕」
  - 標準體重正負20％以上為「肥胖」或「體重不足」



計算標準體重

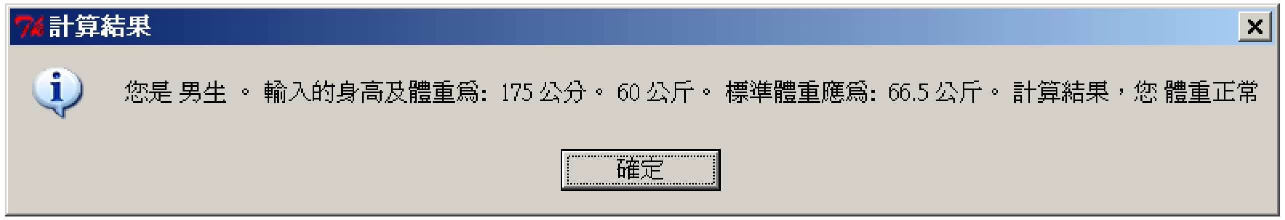
性別

(● 男生) ○ 女生

輸入資料

身高(公分) 175

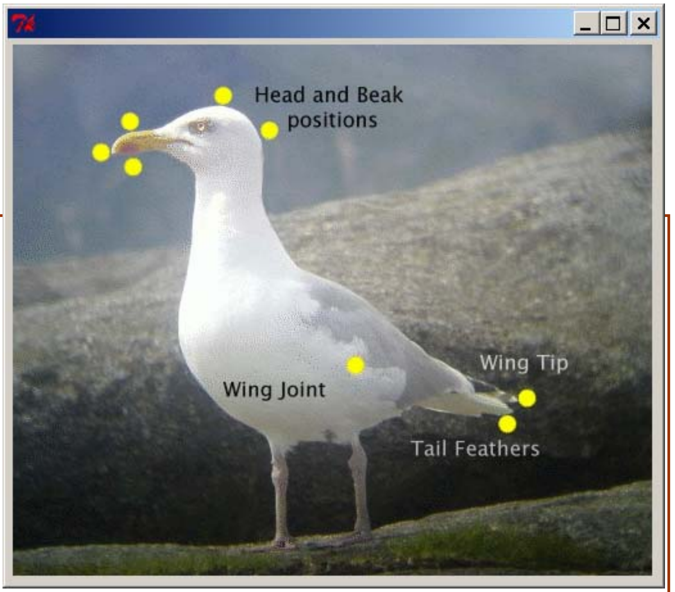
體重(公斤) 60



計算結果

您是 男生 。 輸入的身高及體重爲: 175 公分。 60 公斤。 標準體重應爲: 66.5 公斤。 計算結果，您 體重正常

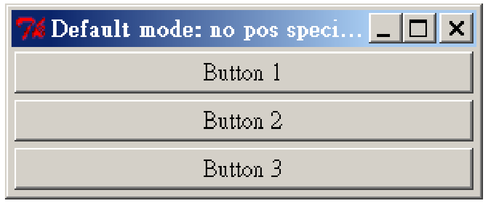
範例 10: rp.image: Placement of an image within rpanel



```
my.click <- function(panel, x, y) {
     print(paste("click 座標: (", x,",", y,")"))
     panel
}
my.drag <- function(panel, x, y) {
     print(paste("drag 座標: (", x,",", y,")"))
     panel
}
my.release <- function(panel, x, y) {
     print(paste("release 座標: (", x,",", y,")"))
     panel
}

my.panel <- rp.control()
image.file <- file.path(system.file(package = "rpanel"), "images",
    "gulllmks.gif")
rp.image(panel = my.panel, filename = image.file, id = "gulls.image",
    action = my.click, mousedrag = my.drag, mouseup = my.release)
```

範例 11.1: Positioning controls: default

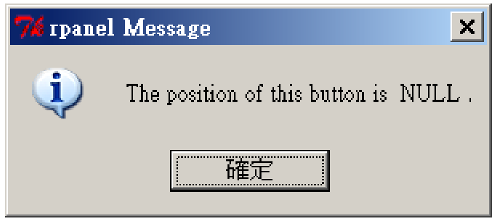
deafult: pos is not specified


Default mode: no pos speci...

Button 1

Button 2

Button 3


rpanel Message

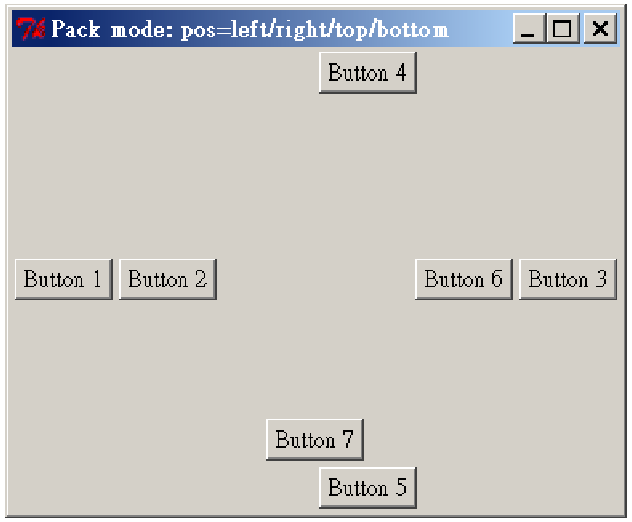
The position of this button is NULL .

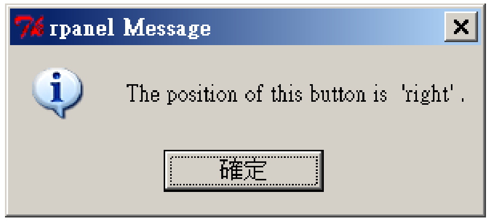
確定

```
showpos <- function(pos){
  function(panel,...) {
    rp.messagebox("The position of this button is ",pos,".")
    panel
  }
}

panel1 <- rp.control(title='Default mode: no pos specified')
rp.button(panel1, action = showpos('NULL'), title = "Button 1")
rp.button(panel1, action = showpos('NULL'), title = "Button 2")
rp.button(panel1, action = showpos('NULL'), title = "Button 3")
```

範例 11.2: Positioning controls: pack

pos is set to "left", "right", "top" or "bottom"



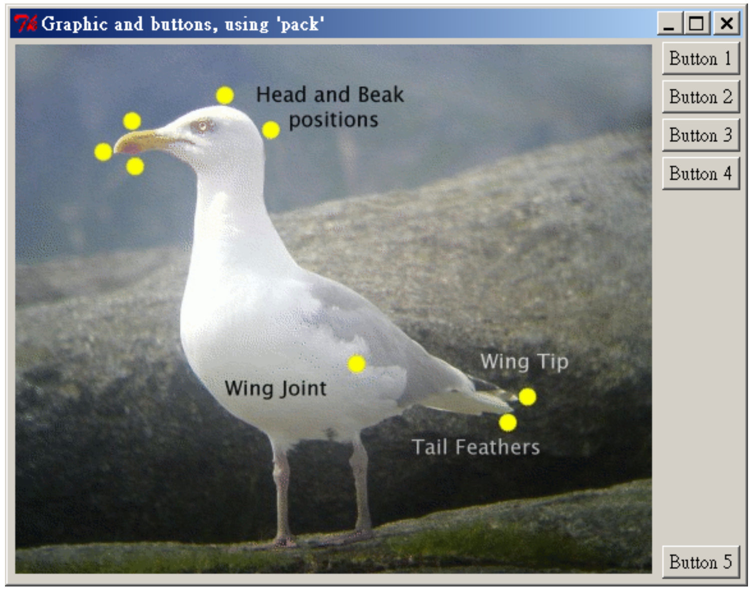

rpanel Message

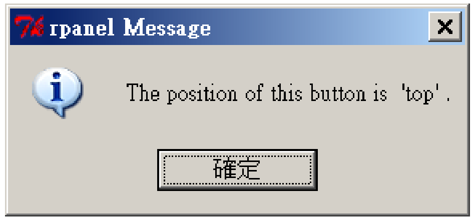
The position of this button is 'right'.

確定

```
panel2 <- rp.control(title="Pack mode: pos=left/right/top/bottom")
rp.button(panel2, action = showpos("'left'"), title = "Button 1", pos = "left")
rp.button(panel2, action = showpos("'left'"), title = "Button 2", pos = "left")
rp.button(panel2, action = showpos("'right'"), title = "Button 3", pos = "right")
rp.button(panel2, action = showpos("'top'"), title = "Button 4", pos = "top")
rp.button(panel2, action = showpos("'bottom'"), title = "Button 5", pos =
"bottom")
rp.button(panel2, action = showpos("'right'"), title = "Button 6", pos = "right")
rp.button(panel2, action = showpos("'bottom'"), title = "Button 7", pos =
"bottom")
```

範例 11.3: Positioning controls: pack



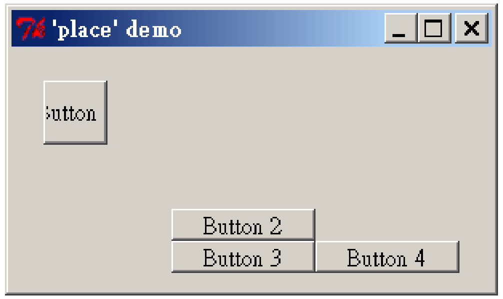


The position of this button is 'top'.

```
panel3 <- rp.control(title="Graphic and buttons, using 'pack'")
image.file <- file.path(system.file(package = "rpanel"), "images",
    "gulllmks.gif")
rp.image(panel3, image.file, pos = "left", id = "gulls.image",
    action = showpos("'left'"))
rp.button(panel3, action = showpos("'top'"), title = "Button 1", pos = "top")
rp.button(panel3, action = showpos("'top'"), title = "Button 2", pos = "top")
rp.button(panel3, action = showpos("'top'"), title = "Button 3", pos = "top")
rp.button(panel3, action = showpos("'top'"), title = "Button 4", pos = "top")
rp.button(panel3, action = showpos("'bottom'"), title = "Button 5", pos =
"bottom")
```

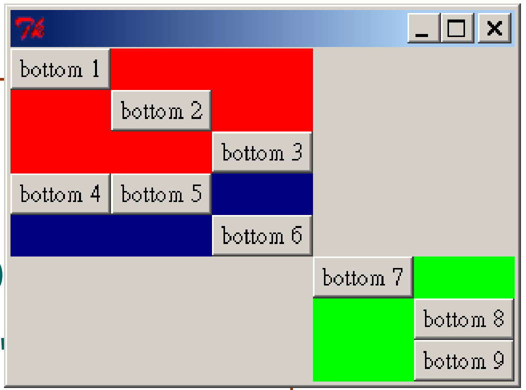
範例 11.4: Positioning : place

```
pos = c(x.axis, y.axis, width, height)
```



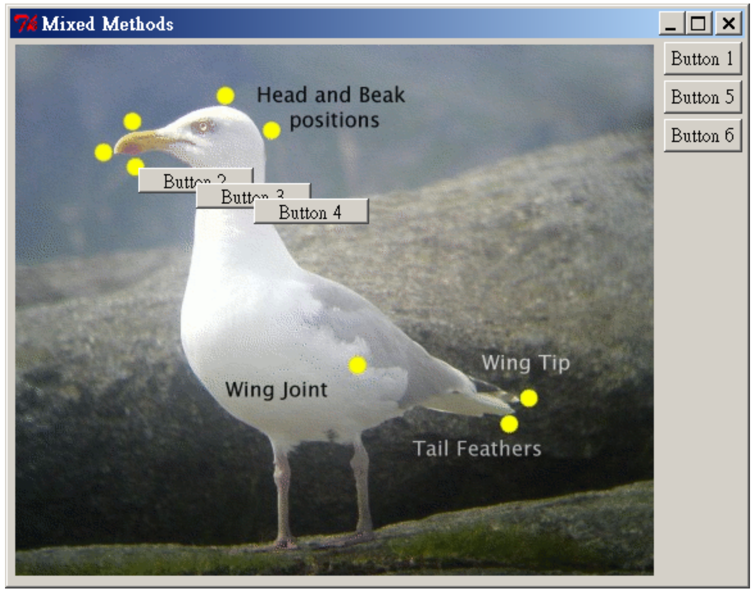
```
panel4 <- rp.control(title="'place' demo", size = c(300,150))
rp.button(panel4, action = showpos("c(20,20,40,40"), title = "Button 1",
    pos = c(20,20,40,40))
rp.button(panel4, action = showpos("c(100,100,90,20)"), title = "Button 2",
    pos = c(100,100,90,20))
rp.button(panel4, action = showpos("c(100,120,90,20"), title = "Button 3",
    pos = c(100,120,90,20))
rp.button(panel4, action = showpos("c(190,120,90,20"), title = "Button 4",
    pos = c(190,120,90,20))
```

範例 11.5: Positioning controls: grid



```
panel7 <- rp.control()

rp.grid(panel7, "g1", pos = list(row = 0, column = 0), bg = "red")
rp.grid(panel7, "g2", pos = list(row = 1, column = 0), bg = "navy")
rp.grid(panel7, "g3", pos = list(row = 2, column = 1), bg = "green")

rp.button(panel7, action = showpos("'bottom 1'"), title = "
    pos = list(row = 0, column = 0, grid = "g1"))
rp.button(panel7, action = showpos("'bottom 2'"), title = "bottom 2",
    pos = list(row = 1, column = 1, grid = "g1"))
rp.button(panel7, action = showpos("'bottom 3'"), title = "bottom 3",
    pos = list(row = 2, column = 2, grid = "g1"))
rp.button(panel7, action = showpos("'bottom 4'"), title = "bottom 4",
    pos = list(row = 1, column = 0, grid = "g2"))
rp.button(panel7, action = showpos("'bottom 5'"), title = "bottom 5",
    pos = list(row = 1, column = 1, grid = "g2"))
rp.button(panel7, action = showpos("'bottom 6'"), title = "bottom 6",
    pos = list(row = 2, column = 2, grid = "g2"))
rp.button(panel7, action = showpos("'bottom 7'"), title = "bottom 7",
    pos = list(row = 0, column = 0, grid = "g3"))
rp.button(panel7, action = showpos("'bottom 8'"), title = "bottom 8",
    pos = list(row = 1, column = 2, grid = "g3"))
rp.button(panel7, action = showpos("'bottom 9'"), title = "bottom 9",
    pos = list(row = 2, column = 2, grid = "g3"))
```

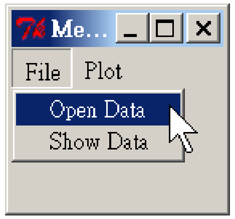
範例 11.6: Positioning controls: 混合範例

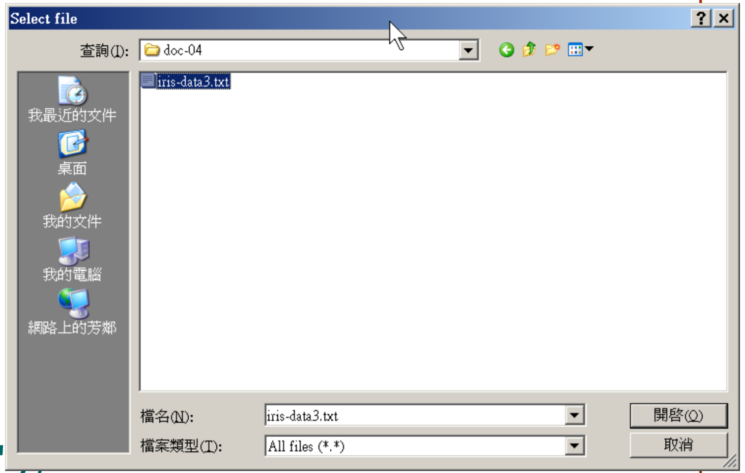


```
panel6 <- rp.control(title="Mixed Methods", size = c(500,416))
rp.image(panel6, image.file, pos = "left", id = "gulls.image",
    action = showpos("'left'"))
rp.button(panel6, action = showpos("NULL"), title = "Button 1")
rp.button(panel6, action = showpos("c(100,100,90,20)"), title = "Button 2",
    pos = c(100,100,90,20))
rp.button(panel6, action = showpos("c(145,112,90,20"), title = "Button 3",
    pos = c(145,112,90,20))
rp.button(panel6, action = showpos("c(190,124,90,20"), title = "Button 4",
    pos = c(190,124,90,20))
rp.button(panel6, action = showpos("NULL"), title = "Button 5")
rp.button(panel6, action = showpos("NULL"), title = "Button 6")
```

範例 12: 讀取檔案

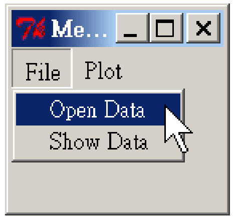
```
my.menu <- function(panel) {
    if(panel$menu == "Open Data"){
        my.file <- file.choose()
        my.data <- read.table(my.file, header=TRUE)
        fix(my.data)
    }
    if(panel$menu == "Show Data"){
        fix(my.data)
    }
    if(panel$menu == "2D plot"){
        plot(my.data[,1], my.data[, 2])
    }
    if(panel$menu == "Histogram"){
        hist(my.data[,1])
    }
    panel
}
my.panel <- rp.control(title = "Menu Demo")
rp.menu(panel = my.panel, var = menu,
    labels = list(list("File", "Open Data", "Show Data")),
                  list("Plot", "2D plot", "Histogram")),
    action = my.menu)
```

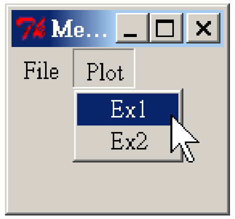


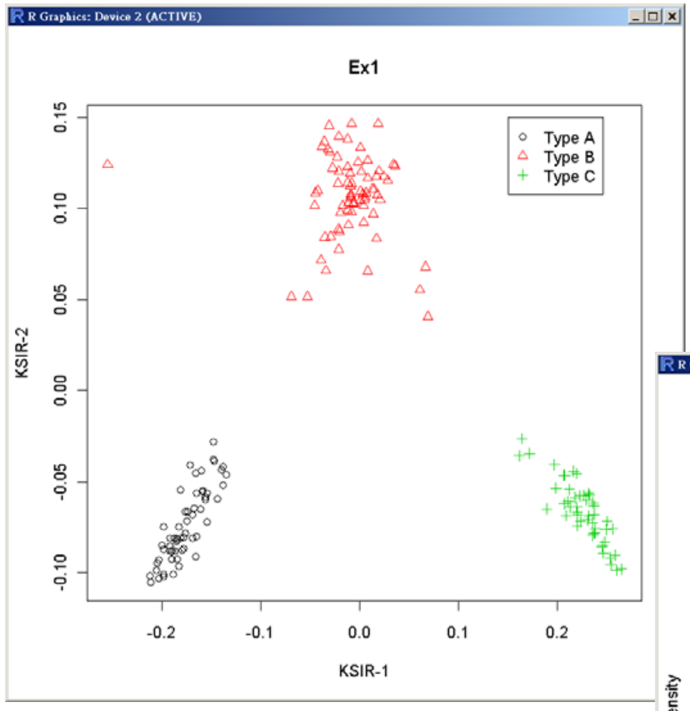


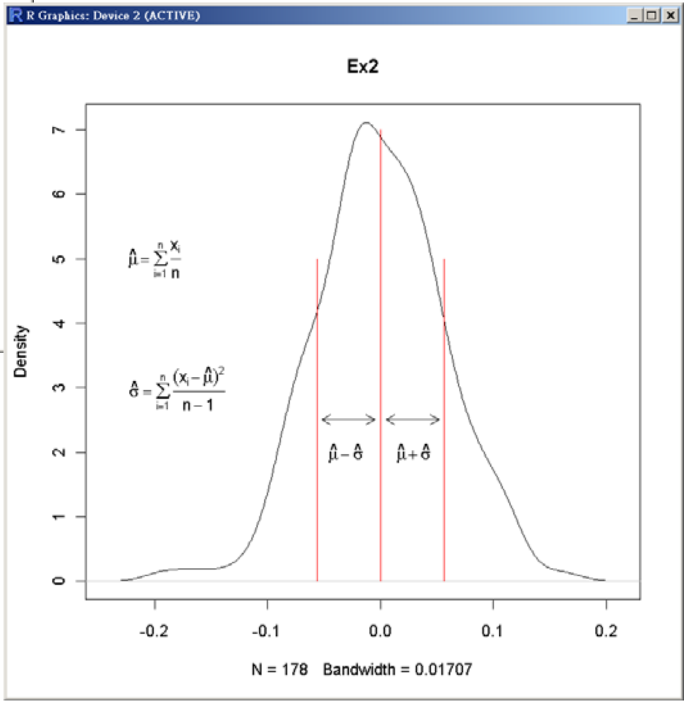
課堂練習 12: 讀取檔案

wine.data.txt

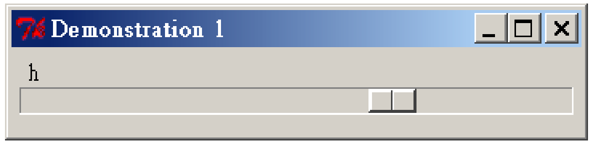


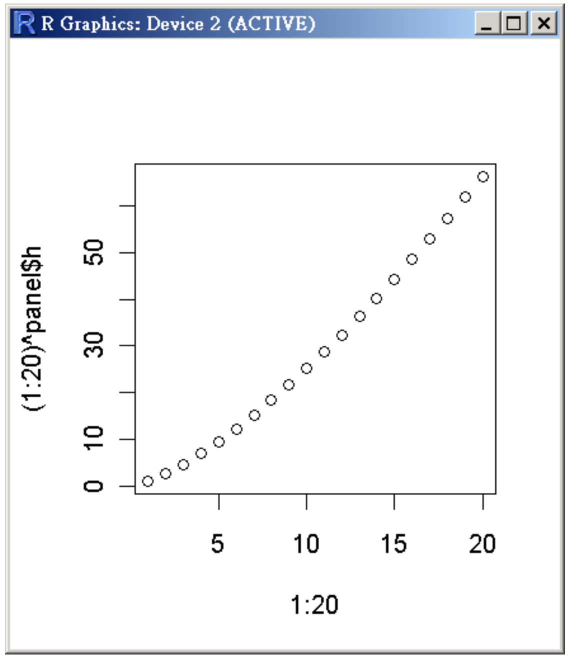






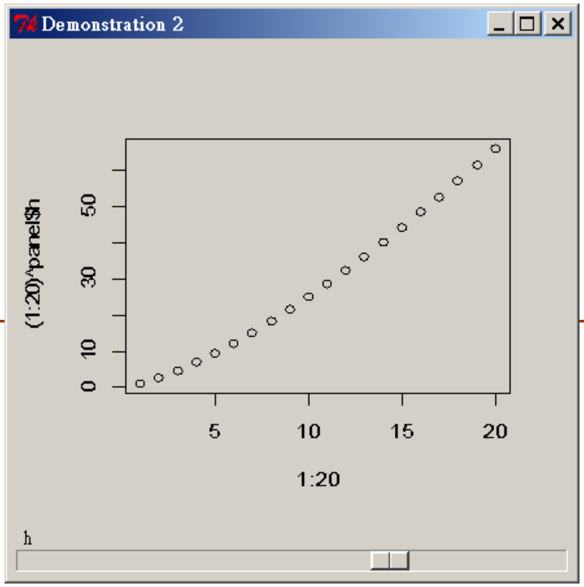
範例 13.1: display R graphics in a panel





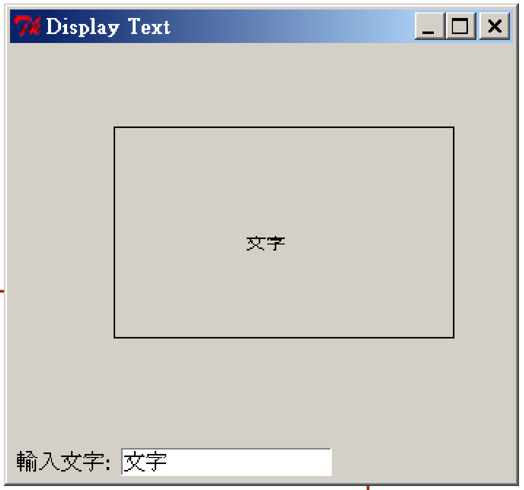
```
my.plot <- function(panel) {
      plot(1:20, (1:20)^panel$h)
      panel
}
my.panel <- rp.control(title = "Demonstration 1", h = 1)
rp.slider(panel = my.panel, var = h, from = 0.05, to = 2.00,
    resolution = 0.05, action = my.plot)
```

範例 13.2: rp.tkrplot



```
my.plot <- function(panel) {
      plot(1:20, (1:20)^panel$h)
      panel
}

my.call <- function(panel) {
      rp.tkrreplot(panel, ex1)
      panel
}

my.panel <- rp.control(title = "Demonstration 2", h = 1)
rp.tkrplot(panel = my.panel, name = ex1, plotfun = my.plot)
rp.slider(panel = my.panel, var = h, from = 0.05, to = 2.00,
    resolution = 0.05, action = my.call)
```

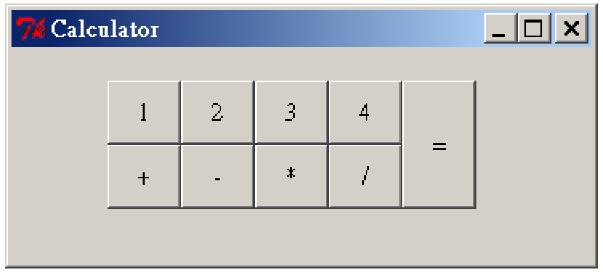
範例 13.3: rp.tkrplot

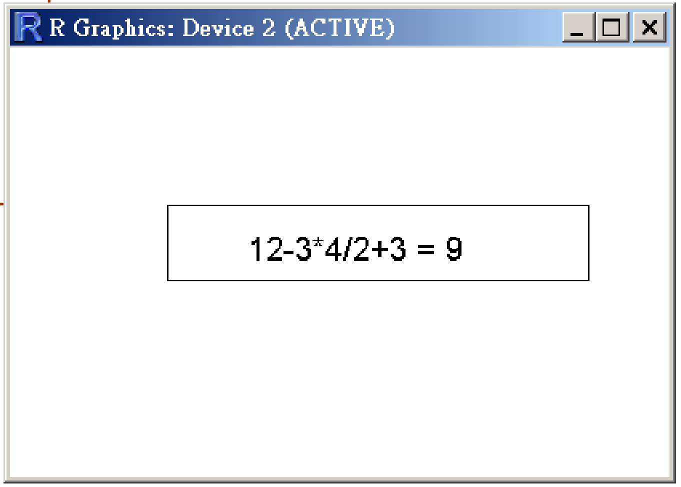


```
my.plot <- function(panel) {
    plot(1:10, 1:10, type="n", xlab="", ylab="",
        axes=FALSE, frame = TRUE)
    text(5, 5, panel$my.text)
    cat(panel$my.text)
    panel
}

my.call <- function(panel) {
    rp.tkrreplot(panel, ex2)
    panel
}

my.panel <- rp.control(title = "Display Text")
rp.tkrplot(panel = my.panel, name = ex2, plotfun = my.plot)
rp.textentry(panel = my.panel, var = my.text,
    labels = "輸入文字: ", action = my.call)
```

範例 13.4: 計算機

```
my.panel <- rp.control(title = "Calculator", math = 0)

rp.button(my.panel, action = my.fun(1), title = "1",
    pos = c(60,20,46,40))
rp.button(my.panel, action = my.fun(2), title = "2",
    pos = c(106,20,46,40))
rp.button(my.panel, action = my.fun(3), title = "3",
    pos = c(152,20,46,40))
rp.button(my.panel, action = my.fun(4), title = "4",
    pos = c(198,20,46,40))
rp.button(my.panel, action = my.fun("+"), title = "+",
    pos = c(60,60,46,40))
rp.button(my.panel, action = my.fun("-"), title = "-",
    pos = c(106,60,46,40))
rp.button(my.panel, action = my.fun("*"), title = "*",
    pos = c(152,60,46,40))
rp.button(my.panel, action = my.fun("/"), title = "/",
    pos = c(198,60,46,40))
rp.button(my.panel, action = my.fun("="), title = "=",
    pos = c(244,20,46,80))
```




R Graphics: Device 2 (ACTIVE)

12-3*4/2+3 = 9

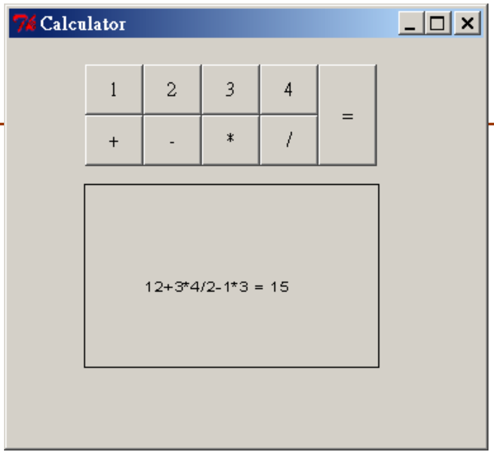
範例 13.4: 計算機

```
my.fun <- function(input){
    function(panel,...) {

        plot(1:10, 1:10, type="n", xlab="", ylab="",
            axes=FALSE, frame = TRUE)

        if(input != "="){
            if(panel$math == 0){
                panel$math <- input
            }else{
                panel$math <- paste(panel$math, input, sep="")
            }
            text(5, 5, panel$math)
        }else{
           s <- parse(file = "",n = NULL, text = panel$math)
           answer <- eval(s[1])
           text(5, 5, paste(panel$math, "=", answer))
           panel$math <- 0
        }
        panel
    }
}
```

課堂練習 13: 計算機

提示:

```
my.fun <- function(panel){

    plot(1:10, 1:10, type="n", xlab="",
        ylab="", axes=FALSE, frame = TRUE)

    if(panel$input != "="){
        text(5, 5, panel$math)
    }else{
       s <- parse(file = "",n = NULL, text = panel$math)
       answer <- eval(s[1])
       text(5, 5, paste(panel$math, "=", answer))
    }
    panel
}
```



12+3*4/2-1*3 = 15

```
my.call <- function(input){
     function(panel) {
     ...
     }
}
```

```
my.panel <- rp.control(...)
rp.tkrplot(...)
rp.button(...)
```