



# 语言初步

## — 数据处理、绘图与编程

张金龙

[jinlongzhang01@gmail.com](mailto:jinlongzhang01@gmail.com)

June 3, 2010

# 报告内容

- 一 R简介
- 二 函数与对象
- 三 脚本编程
- 四 R绘图
- 五 编写函数
- 六 数据保存

# 一 R 简介

# 什么是R？

## The R Project for Statistical Computing

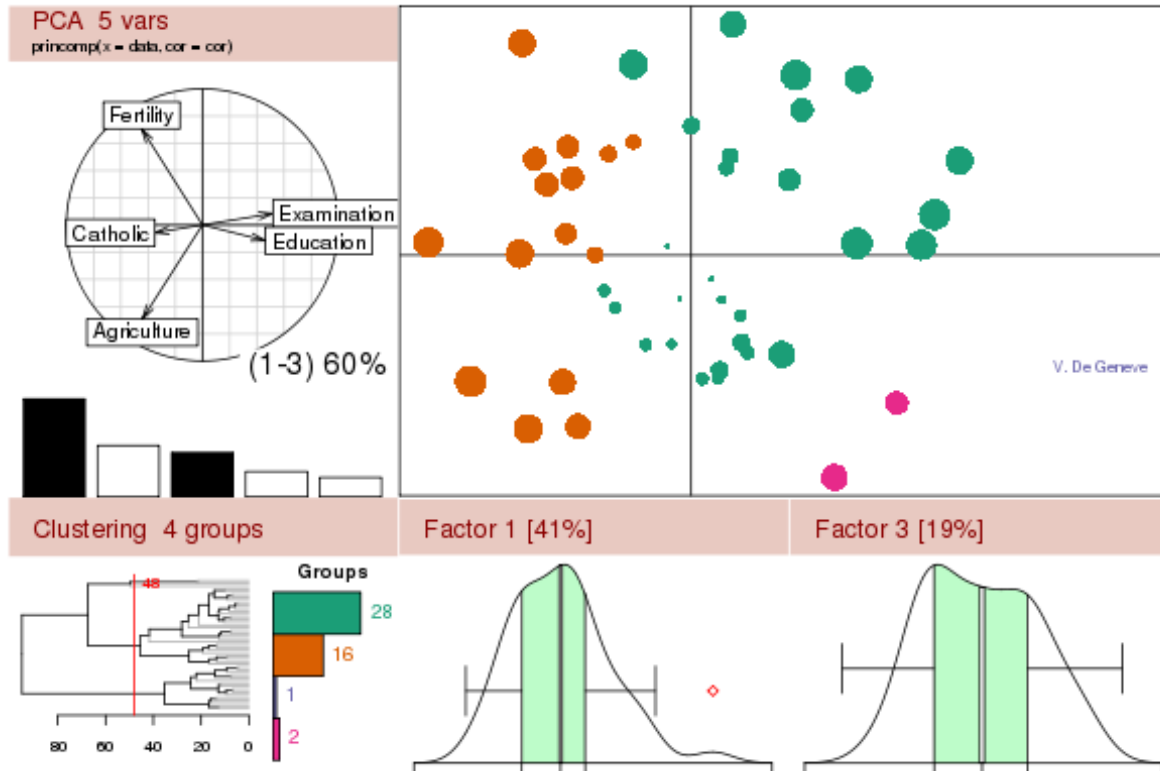


图1 R首页的图形

- R是一种统计绘图语言，也指实现该语言的软件。

# 简 史

R语言是从S统计绘图语言演变而来，可看作S的“方言”。

S语言上世纪70年代诞生于贝尔实验室，由Rick Becker, John Chambers, Allan Wilks开发。

基于S语言开发的商业软件**Plus**，可以方便的编写函数、建立模型，具有良好的扩展性，在国外学术界应用很广。

1995年由新西兰Auckland大学统计系的**R**obert Gentleman和**R**oss Ihaka，基于S语言的源代码，编写了一能执行S语言的软件，并将该软件的源代码全部公开，这就是R软件，其命令统称为R语言。

# R的特点

## 多领域的统计资源

目前在R网站上约有2400个程序包，涵盖了基础统计学、社会学、经济学、生态学、空间分析、系统发育分析、生物信息学等诸多方面。

## 跨平台

R可在多种操作系统下运行，如Windows、MacOS、多种Linux和UNIX等。

## 命令行驱动

R即时解释，输入命令，即可获得相应的结果。

# 为什么选择R？

- 丰富的资源

涵盖了多种行业数据分析中几乎所有的方法。

- 良好的扩展性

十分方便得编写函数和程序包，跨平台，可以胜任复杂的数据分析、绘制精美的图形。

- 完备的帮助系统

每个函数都有统一格式的帮助，运行实例。

- GNU软件

免费、软件本身及程序包的源代码公开。

# R与其他统计软件比较

- SAS:

速度快，有大量统计分析模块，可扩展性稍差，昂贵。

- SPSS:

复杂的用户图形界面，简单易学，但编程十分困难。

- Splus:

运行S语言，具有复杂的界面，与R完全兼容，昂贵。

.....



# R的缺点

- 用户需要对命令熟悉

与代码打交道，需要记住常用命令。

- 占用内存

所有的数据处理在内存中进行，不适于处理超大规模的数据。

- 运行速度稍慢

即时编译，约相当于C语言的1/20。

- 相比点击鼠标进行操作，R仍能够大大提高效率。



# The R Project for Statistical Computing

About R  
 What is R?  
 Contributors  
 Screenshots  
 What's new?

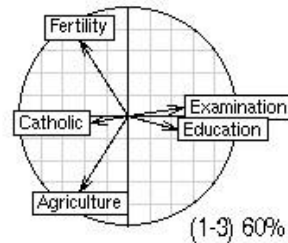
Download, Packages  
 CRAN

R Project  
 Foundation  
 Members & Donors  
 Mailing Lists  
 Bug Tracking  
 Developer Page  
 Conferences  
 Search

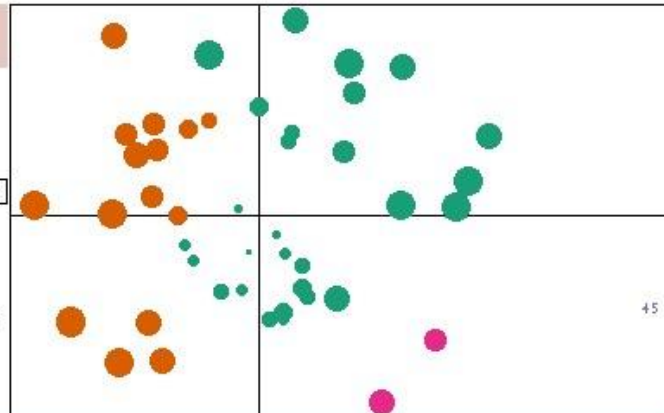
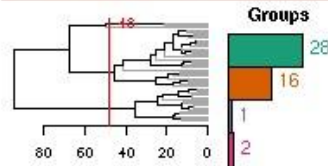
Documentation  
 Manuals  
 FAQs  
 The R Journal  
 Wiki  
 Books  
 Certification  
 Other

Misc  
 Bioconductor  
 Related Projects

PCA 5 vars  
 princomp(x = data, cor = cor)



Clustering 4 groups



Factor 1 [41%]

Factor 3 [19%]



## Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).
- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

图2 R软件首页 <http://www.r-project.org/>

# CRAN

The Comprehensive R Archive Network

简称**CRAN**，由世界几十个镜像网站组成网络，提供下载安装程序和相应软件包。

各镜像更新频率一般为1-2天

推荐镜像：

中国的镜像:数学所

<http://ftp.ctex.org/mirrors/CRAN/>

即时更新的CRAN源

<http://cran.r-project.org/>

# Windows下载和安装R

CRAN: Binaries>Windows>base

R-2.11.0 for Windows

[Download R 2.11.0 for Windows](#) (33 megabytes, 32 bits)

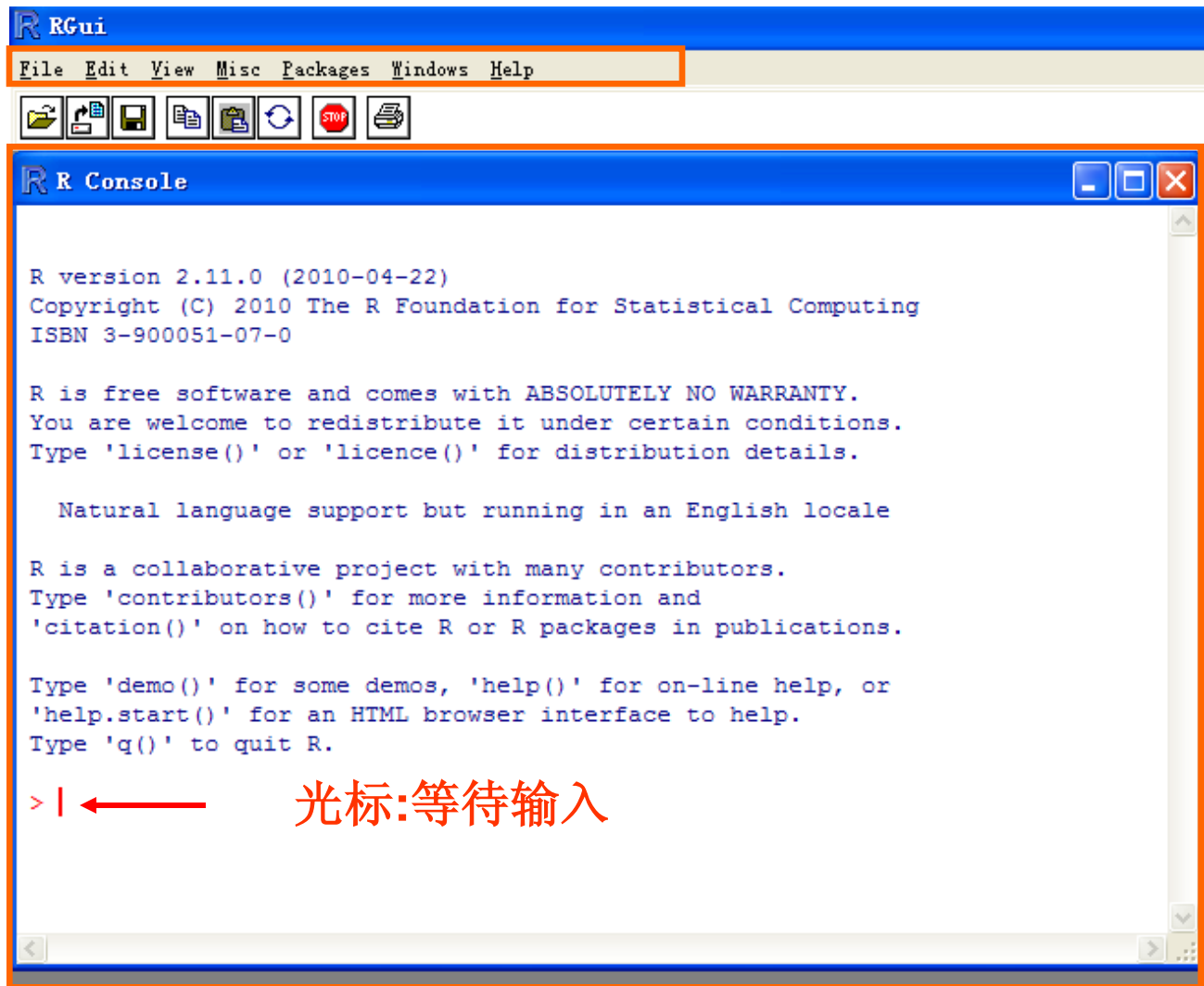
[Installation and other instructions](#)

New features in this version: [Windows specific](#), [all platforms](#).

图3 R2.11.0下载页面

下载完成后，双击[R-2.11.0-win32.exe](#)开始安装。

一直点击下一步，各选项默认，语言建议选英文。



菜单栏  
快捷按钮

控制台

图 4 R登陆界面(Windows版)

路径： 开始>所有程序>R 2.11.0

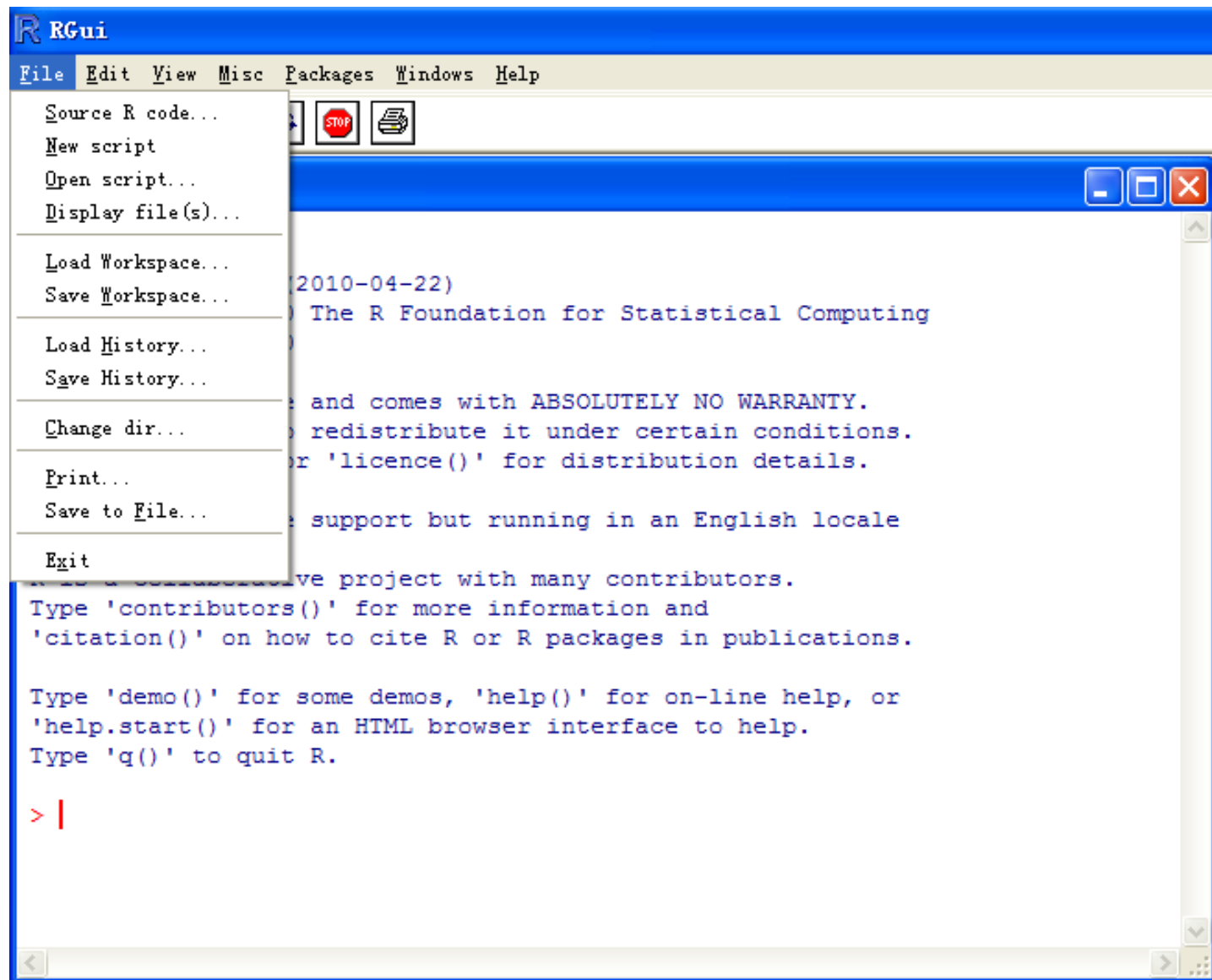


图 5 R Gui 的File菜单

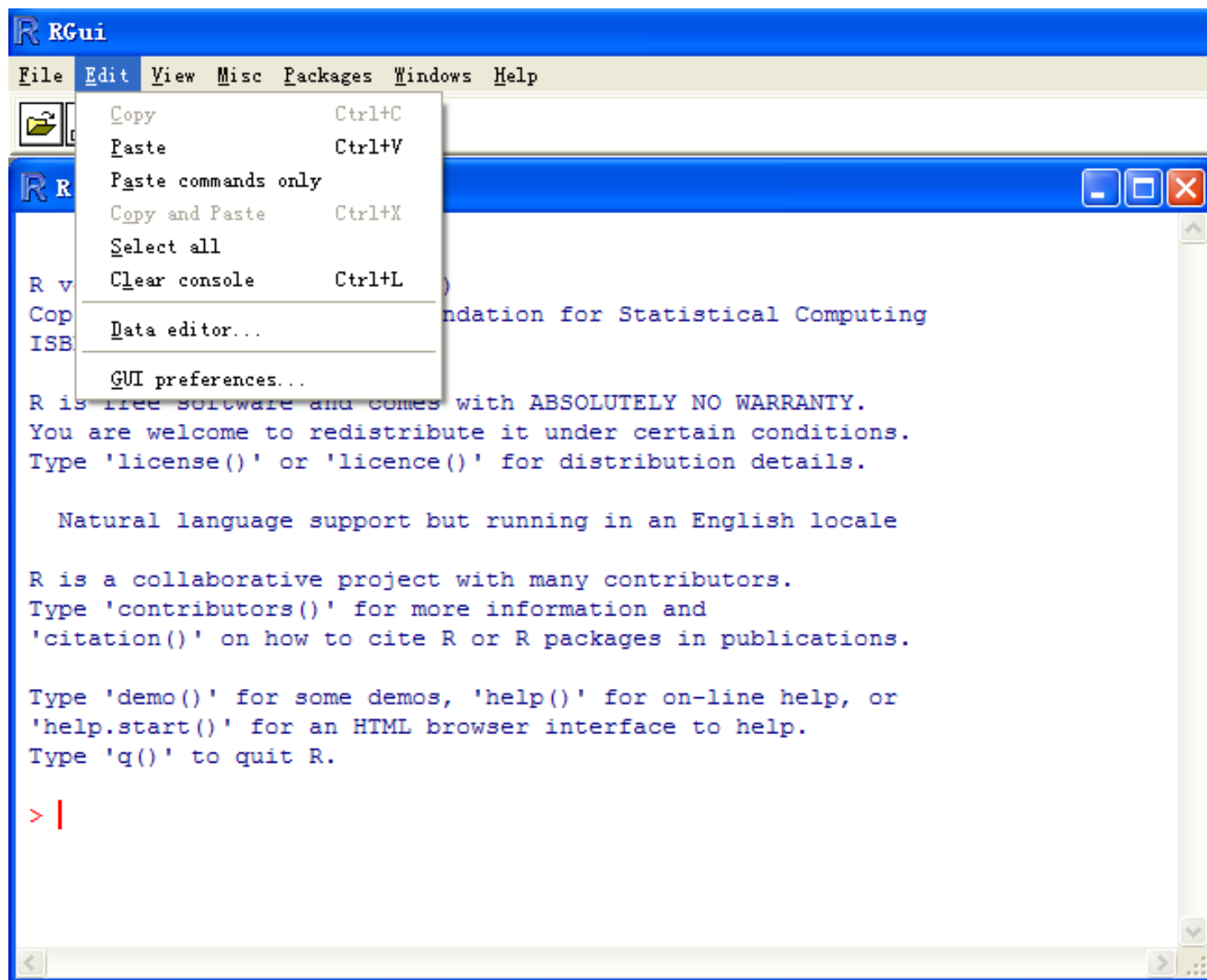


图 6 R Gui 的Edit菜单

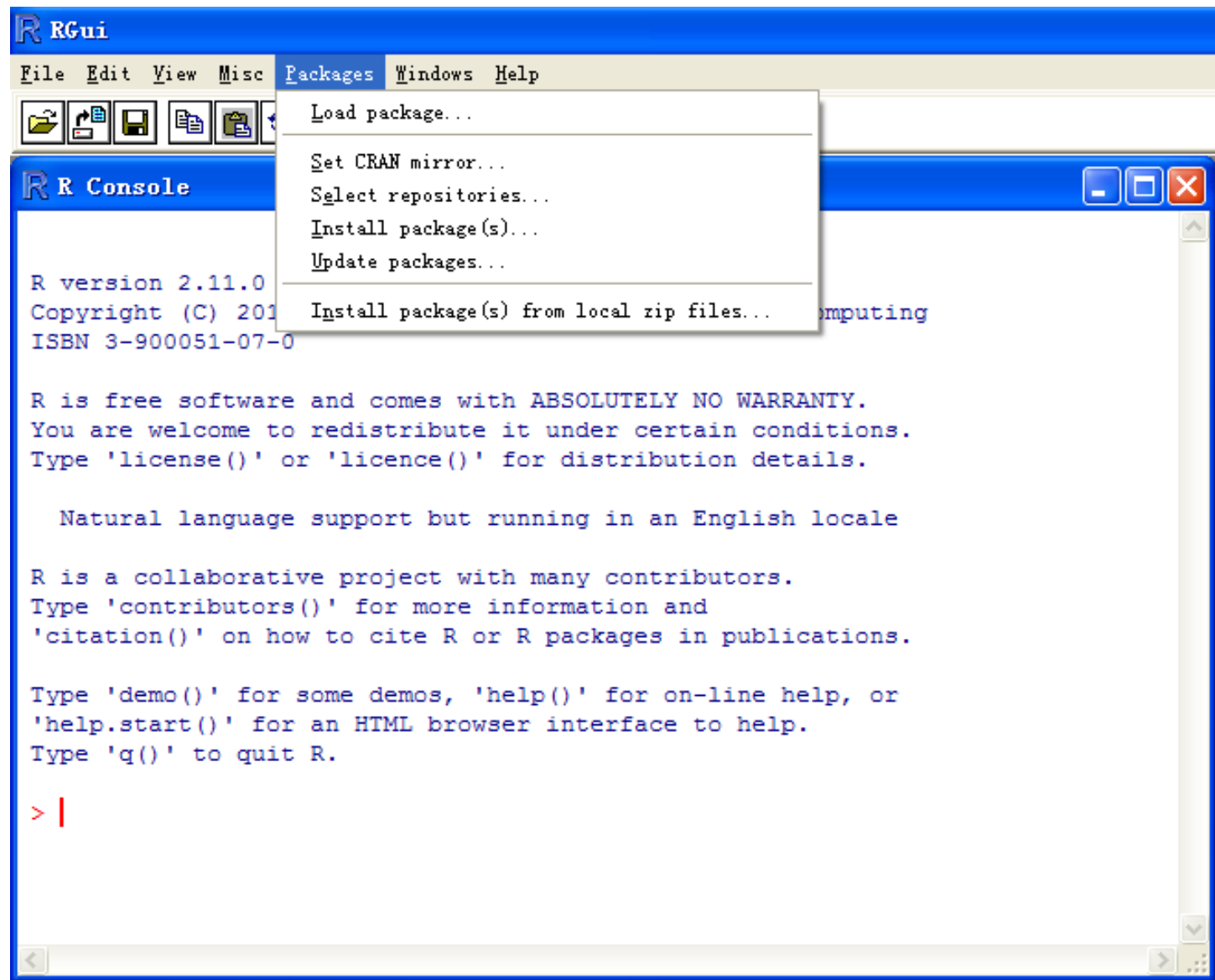


图 7 R Gui 的Packages菜单



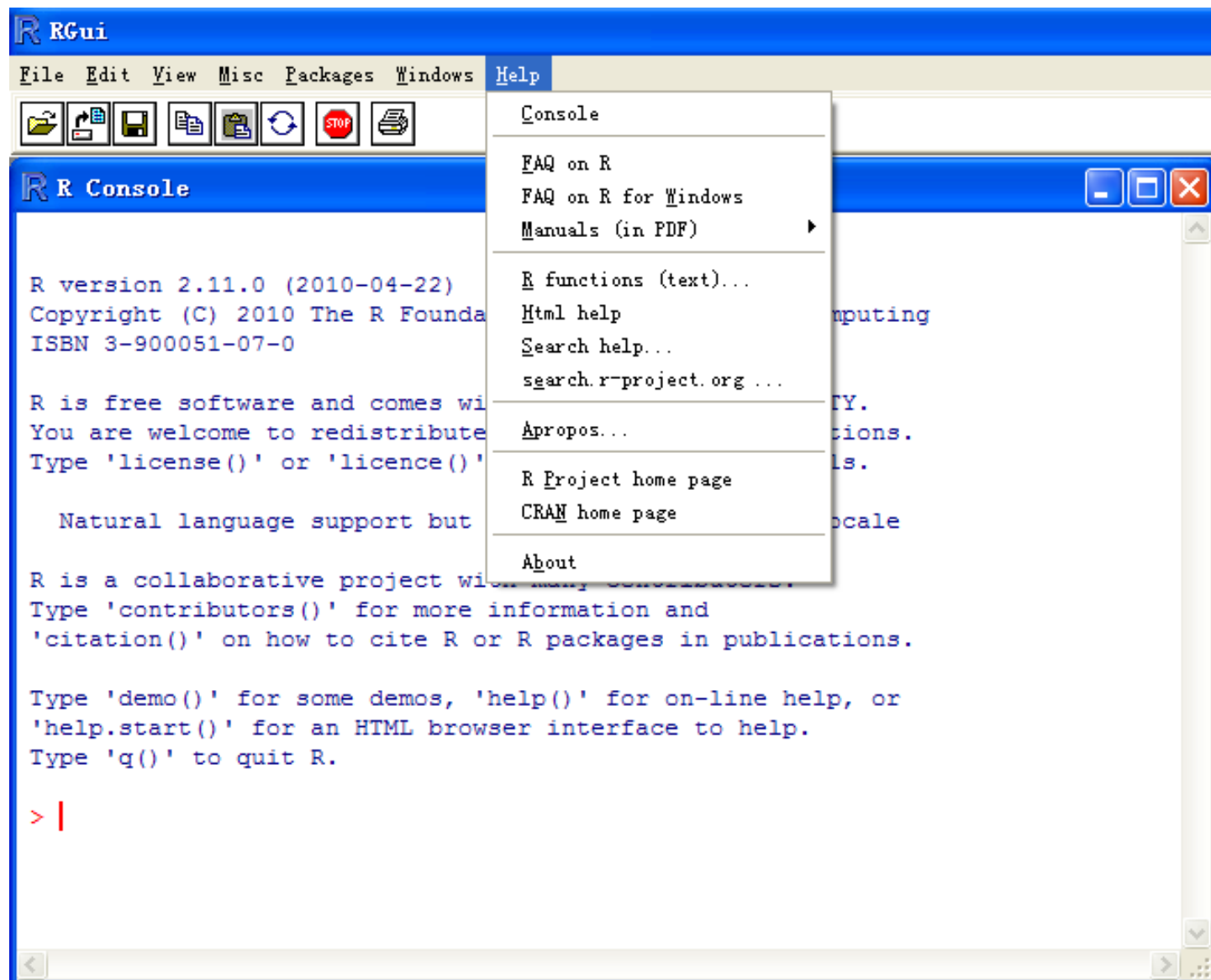


图 8 R Gui 的Help菜单

# R程序包（R Packages）

## 程序包是什么？

R程序包是多个函数的集合，具有详细的说明和示例。

Window下的R程序包是经过编译的zip包。

每个程序包包含R函数、数据、帮助文件、描述文件等。

## 为什么要安装程序包？

R程序包是R功能扩展，特定的分析功能，需要用相应的程序包实现。

例如：系统发育分析，常用到ape程序包，群落生态学vegan包等。

# 常用R程序包(I)

<b>ade4</b>	利用欧几里得方法进行生态学数据分析
<b>adephylo</b>	系统进化数据挖掘与比较方法
<b>ape</b>	系统发育与进化分析
<b>apTreeshape</b>	进化树分析
<b>boot</b>	<b>Bootstrap</b> 检验
<b>cluster</b>	聚类分析
<b>ecodist</b>	生态学数据相异性分析
<b>FD</b>	功能多样性分析
<b>geiger</b>	物种形成速率与进化分析

## 常用R程序包(II)

**Graphics**

绘图

**lattice**

栅格图

**maptools**

空间对象的读取和处理

**mefa**

生态学和生物地理学多元数据处理

**mgcv**

广义加性模型相关

**mvpart**

多变量分解

**nlme**

线性及非线性混合效应模型

**ouch**

系统发育比较

**pgirmess**

生态学数据分析

**phangorn**

系统发育分析

## 常用R程序包(III)

<b>picante</b>	群落系统发育多样性分析
<b>raster</b>	栅格数据分析与处理
<b>seqinr</b>	DNA序列分析
<b>sp</b>	空间数据处理
<b>spatstat</b>	空间点格局分析，模型拟合与检验
<b>splancs</b>	空间与时空点格局分析
<b>stats</b>	R统计学包
<b>SDMTools</b>	物种分布模型工具
<b>vegan</b>	植物与植物群落的排序，生物多样性计算

# CRAN Task Views

## CRAN Task Views

<a href="#">Bayesian</a>	Bayesian Inference
<a href="#">ChemPhys</a>	Chemometrics and Computational Physics
<a href="#">ClinicalTrials</a>	Design, Monitoring, and Analysis of Clinical Trials
<a href="#">Cluster</a>	Cluster Analysis & Finite Mixture Models
<a href="#">Distributions</a>	Probability Distributions
<a href="#">Econometrics</a>	Computational Econometrics
<a href="#">Environmetrics</a>	Analysis of Ecological and Environmental Data
<a href="#">ExperimentalDesign</a>	Design of Experiments (DoE) & Analysis of Experimenta
<a href="#">Finance</a>	Empirical Finance
<a href="#">Genetics</a>	Statistical Genetics
<a href="#">Graphics</a>	Graphic Displays & Dynamic Graphics & Graphic Device Visualization
<a href="#">gR</a>	gRaphical Models in R
<a href="#">HighPerformanceComputing</a>	High-Performance and Parallel Computing with R
<a href="#">MachineLearning</a>	Machine Learning & Statistical Learning
<a href="#">MedicalImaging</a>	Medical Image Analysis
<a href="#">Multivariate</a>	Multivariate Statistics
<a href="#">NaturalLanguageProcessing</a>	Natural Language Processing
<a href="#">Optimization</a>	Optimization and Mathematical Programming

图 9 CRAN Task Views: 对程序包的分类介绍

## vegan: Community Ecology Package

Ordination methods, diversity analysis and other functions for community and vegetation ecologists.

Version: 1.17-2  
Suggests: [MASS](#), [mgcv](#), [lattice](#), [cluster](#), [scatterplot3d](#), [rgl](#), tcltk  
Published: 2010-03-08  
Author: Jari Oksanen, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, R. B. O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens, Helene Wagner  
Maintainer: Jari Oksanen <jari.oksanen at oulu.fi>  
License: [GPL-2](#)  
URL: <http://vegan.r-forge.r-project.org/>  
In views: [Environmetrics](#), [Multivariate](#), [Phylogenetics](#), [Psychometrics](#), [Spatial](#)  
CRAN  
checks: [vegan results](#)

### Downloads:

Package source: [vegan\\_1.17-2.tar.gz](#)  
MacOS X binary: [vegan\\_1.17-2.tgz](#)  
Windows binary: [vegan\\_1.17-2.zip](#)  
Reference manual: [vegan.pdf](#)

三种平台上的  
程序包

图 10 vegan包页面

# R程序包

在CRAN 提供了每个包的源代码和编译好的程序包

以vegan包为例，CRAN提供了：

Package source: [vegan\\_1.17-2.tar.gz](#)

MacOS X binary: [vegan\\_1.17-2.tgz](#)

Windows binary: [vegan\\_1.17-2.zip](#)

Reference manual: [vegan.pdf](#)

Window下程序包为zip文件，安装时不要解压缩。



# 安装程序包的方法

## 1 用函数 `install.packages()`,

如果已经连接到互联网, 在括号中输入要安装的程序包名称, 选择镜像后, 程序将自动下载并安装程序包。

例如: 要安装**picante**包, 在控制台中输入

```
install.packages("picante")
```

## 2 安装本地**zip**包

路径: **Packages>install packages from local files**

选择本地磁盘上存储**zip**包的文件夹。

# 程序包使用

程序包中的函数，都要先导入，再使用，因此导入程序包是第一步。

在控制台中输入如下命令：

```
library(vegan)
```

```
library(ade4)
```

程序包内的函数的用法与R内置的基本函数用法一样。

```
library(vegan)
```

```
This is vegan 1.17-2
```

# 查看程序包帮助文件

vegan 程序包内部都有哪些函数？分别有什么功能？

查询程序包内容最常用的方法：

1 菜单 帮助>Html帮助

2 查看pdf帮助文档

# 查看函数的帮助文件

函数的默认值是什么？ 怎么使用？ 使用时需要注意什么问题？  
需要查询函数的帮助。

1 `?t.test`

2 `RGui>Help>Html help`

3 `apropos("t.test")`

4 `help("t.test")`

5 `help.search("t.test")`

6 查看R包pdf手册

## Fitting Linear Models

### Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although [aov](#) may provide a more convenient interface for these).

### Usage

```
lm(formula, data, subset, weights, na.action,  
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,  
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

### Arguments

- |                      |  |
|----------------------|--|
| <code>formula</code> | an object of class " <a href="#">formula</a> " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under ‘Details’.   |
| <code>data</code>    | an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called. |
| <code>subset</code>  | an optional vector specifying a subset of observations to be used in the fitting process.  |

图 11 R帮助文件的内容与格式

# 帮助文件的内容

- `lm{stats}` #函数名及所在包
- `Fitting Linear Models` #标题
- `Description` #函数描述
- `Usage` #默认选项
- `Arguments` #参数
- `Details` #详情
- `Author(s)` #作者
- `References` #参考文献
- `Examples` #举例

# 练习一 安装R并导入程序包

1. 安装R软件、熟悉菜单
2. 安装程序包
3. 调用程序包，查看程序包的帮助

```
library(vegan)
```

```
library(ape)
```

查找ape包中plot.phylo函数的帮助

输入 `?plot.phylo`

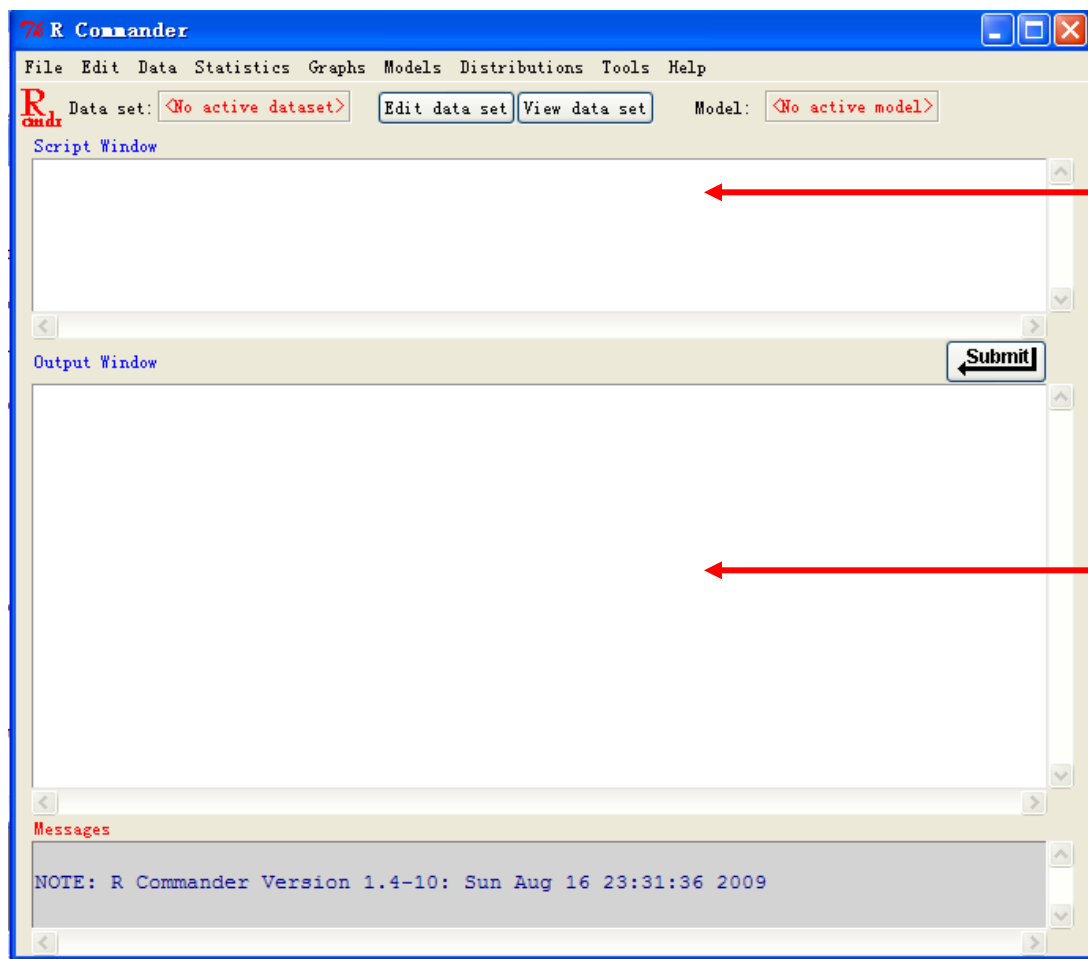
将其中的Example粘贴到控制台中，查看运行的结果。

# R图形界面： R commander

- R commander是R的图形界面之一，是John Fox教授编写的，适用于不希望R编程的用户。随着用户的操作，其窗口还可以显示出相应操作的R程序，对于初学者可能会有帮助。
- 安装R commander
- `install.packages("Rcmdr")`
- R将自动下载并安装Rcmdr所需的所有程序包



# 图形界面之一：R Commander



界面操作代码

结果输出

图 12 R commander 界面 `library(Rcmdr)`

# 为什么要学习编程？

界面操作直观易学，但也存在一些不足：

(1)操作的过程难以保存，数据处理不够灵活，在进行步骤繁多的数据处理工作时十分费时费力；

(2)在建立模型或自己编写函数时也会遇到困难。

而这些困难可以通过编程解决。

要学习R编程，首先要了解R的函数、对象及其操作。

## 二 函数与对象

# R的函数

R是一种解释性语言，输入后可直接给出结果。

功能靠函数实现。

函数形式：

函数(输入数据, 参数= )

如果没有指定，则参数的以默认值为准。

例如：

平均值 `mean(x, trim = 0, na.rm = FALSE, ...)`

线性模型 `lm(y~x, data=test)`

# R的函数

每一个函数执行特定的功能，后面紧跟括号，例如：

平均值     `mean()`

求和        `sum()`

绘图        `plot()`

排序        `sort()`

除了基本的运算之外，R的函数又分为”高级”和”低级”函数，高级函数可调用低级函数，这里的”高级”函数习惯上称为泛型函数。

如`plot()`就是泛型函数，可以根据数据的类型，调用底层的函数，应用相应的方法绘制相应的图形。这就是面向对象编程的思想。

# R有哪些函数？

查询的方法：Help>Html

help>packages

log()

log10()

exp()

sin()

cos()

tan()

asin()

acos()

binom.test()

fisher.test()

chisq.test()

glm(y ~ x1+x2+x3, binomial)

friedman.test()

mean()

sd()

var()

...

# R函数调用及其选项

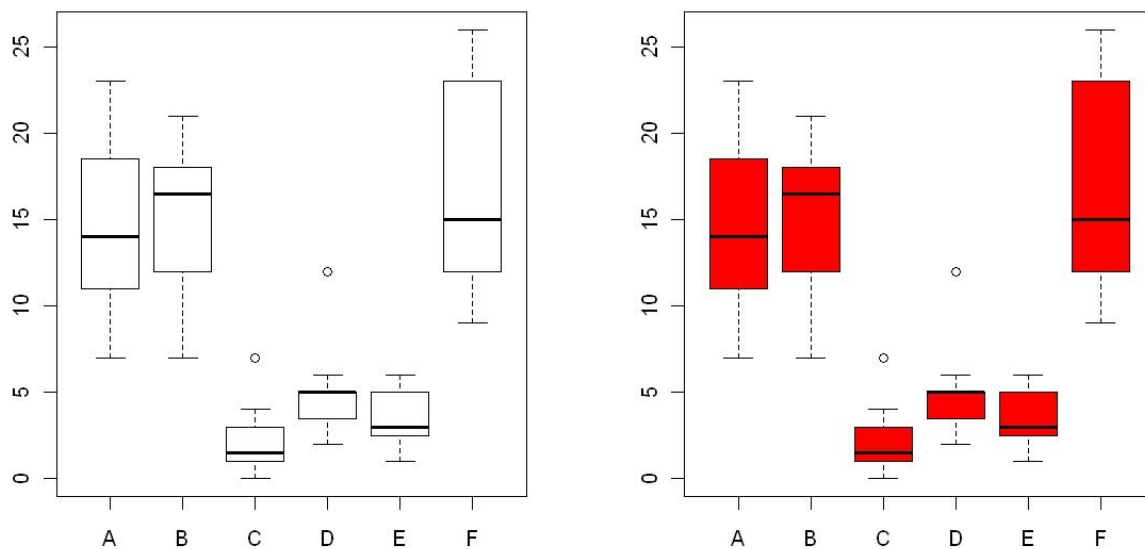


图13 箱线图修饰前后 (左: 默认值, 右: 修改属性后)

函数的调用方法, 函数名() 如 `plot()`, `lm()`, 并将对象放入括号中。

# R函数调用及其选项

箱线图绘制函数的调用

```
boxplot(day~type, data=bac, col="red", xlab="Virus",  
        ylab="days")
```

`day~type`, 以`type`为横轴, `day`为纵轴绘制箱线图。

`data=bac`            数据来源`bac`

`col="red"`           箱线图为红色

`xlab="Virus"`       横轴名称为`Virus`

`ylab="days"`       纵轴名称为`days`



## 练习二：查询函数帮助

查看**boxplot**的帮助文件

```
?boxplot
```

查看最后的**examples**

将帮助文件中的内容粘贴到控制台中，运行并观看运行结果。

```
boxplot(count ~ spray, data = InsectSprays, col =  
        "lightgray")
```

参数更改

```
boxplot(count ~ spray, data = InsectSprays, col = "red",  
        xlab="spray", ylab="counts")
```

# 赋值与注释

在控制台中键入如下命令

```
2 + 2
```

```
a <- 2
```



赋值符号

<-也可用=, 甚至->代替

```
b <- 2
```

```
c <- a+b
```

```
c
```

#注释

# 如何为对象起名？

R处理的所有数据、变量、函数和结果都以对象的形式保存。

1. 区分大小写, 注意China和china的不同。
2. 不能用数字作为变量, 对象也不能用数字开头, 但是数字可以放在中间或结尾, 如2result与result2, 后者是合法的。
3. 建议不要用过短的名称。可以用”.”作为间隔, 例如 anova.result1。
4. 不要使用保留名:

NA, NaN, pi, LETTERS, letters, month.abb, month.name

# 元素的类型

对象是由各元素组成的。每个元素，都有自己的数据类型

数值型 **Numeric**      如 100, 0, -4.335

字符型 **Character**    如 “China”

逻辑型 **Logical**      如 TRUE, FALSE

因子型 **Factor**        表示不同类别

复数型 **Complex**      如:  $2 + 3i$

# 对象的类(class)

向量 (**vector**) 一系列元素的组合。

如 `c(1,2,3)` ; `c("a","a","b","b","c")`

因子 (**factor**) 因子是一个分类变量

`c("a","a","b","b","c")`

矩阵 (**matrix**) 二维的数据表，是数组的一个特例

`x <- 1:12 ; dim(x) <- c(3,4)`

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

# 对象的类

## 数组 (**array**)

数组是 $k$ 维的数据表 ( $k \text{ in } 1:n, n$  为正整数)。

向量 ( $n = 1$ ) 矩阵 ( $n = 2$ ) 高维数组 ( $n \geq 3$ )

## 数据框 (**dataframe**)

是由一个或几个向量和（或）因子构成，它们必须是等长的，但可以是不同的数据类型。

## 列表 (**list**)

列表可以包含任何类型的对象。

可以包含向量、矩阵、高维数组，也可以包含列表

# 运算符

数学运算 运算后给出数值结果

**+, -, \*, /, ^ (幂)**

比较运算 运算后给出判别结果 (**TRUE FALSE**)

**>, <, <=, >=, ==, !=**

逻辑运算 与、或、非

**!, &, &&, |, ||**

# 数据表的行与列

表1 数据表、数据框与向量

列名  
Column  
names

	物种数	科数	属数	海拔	坡度	类型
样方1	40	15	22	600	25	山顶
样方2	51	12	26	350	30	山坡
样方3	46	11	20	390	45	山坡
样方4	38	12	24	260	20	低地
样方5	49	10	25	220	33	低地

每行作  
为一个  
Entry

行名Row names  
字符串

每列可看做带名称  
的向量

字符串、因子



# 数据框的组成

- 每个数据表可以看作一个数据框 (**dataframe**)。
- 每一列 (**column**) 作为一个向量 (**vector**)。
- 由很多不同类型的向量组成，如字符型，因子型，数值型。
- 每一行 (**row**) 作为一个记录 (**entry**)
- 如何生成数据框？
- 两种办法：
  - (1)从外部数据读取
  - (2)各类型因子组合成数据框

# 外部数据读取

最为常用的数据读取方式是用`read.table()` 函数或`read.csv()` 函数读取外部`txt`或`csv`格式的文件。

`txt`文件，制表符间隔

`csv`文件，逗号间隔

一些R程序包（如`foreign`）也提供了直接读取`Excel`，`SAS`，`dbf`，`Matlab`，`spss`，`systat`，`Minitab`文件的函数。

# read.csv() 的使用

例: `test.data<-read.csv("D:/R/test2.csv",header=T)`

`header=T`表示将数据的第一行作为标题。

`read.table(file=file.choose(),header=T)` 可以弹出对话框, 选择文件。

## 例：从数据输入到t检验

现有6名患者的身高和体重，检验体重除以身高的平方是否等于22.5。

表2 六名患者的身高和体重

编号	1	2	3	4	5	6
身高 m	1.75	1.80	1.65	1.90	1.74	1.91
体重kg	60	72	57	90	95	72

# 第一种方式：从控制台输入数据

数据量较少时可以从控制台直接输入：

```
height<-c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
```

```
weight<-c(60, 72, 57, 90, 95, 72)
```

```
sq.height<-height^2
```

```
ratio<-weight/sq.height
```

```
t.test(ratio, mu=22.5)
```

## 第二种方式 从外部读取数据

数据量较大时用`read.table`函数从外部`txt`文件读取

第1步 将Excel中的数据另存为`.txt`格式（制表符间隔）或`.csv`格式。

第2步 用`read.table()`或`read.csv()`函数将数据读入R工作空间，并赋值给一个对象。

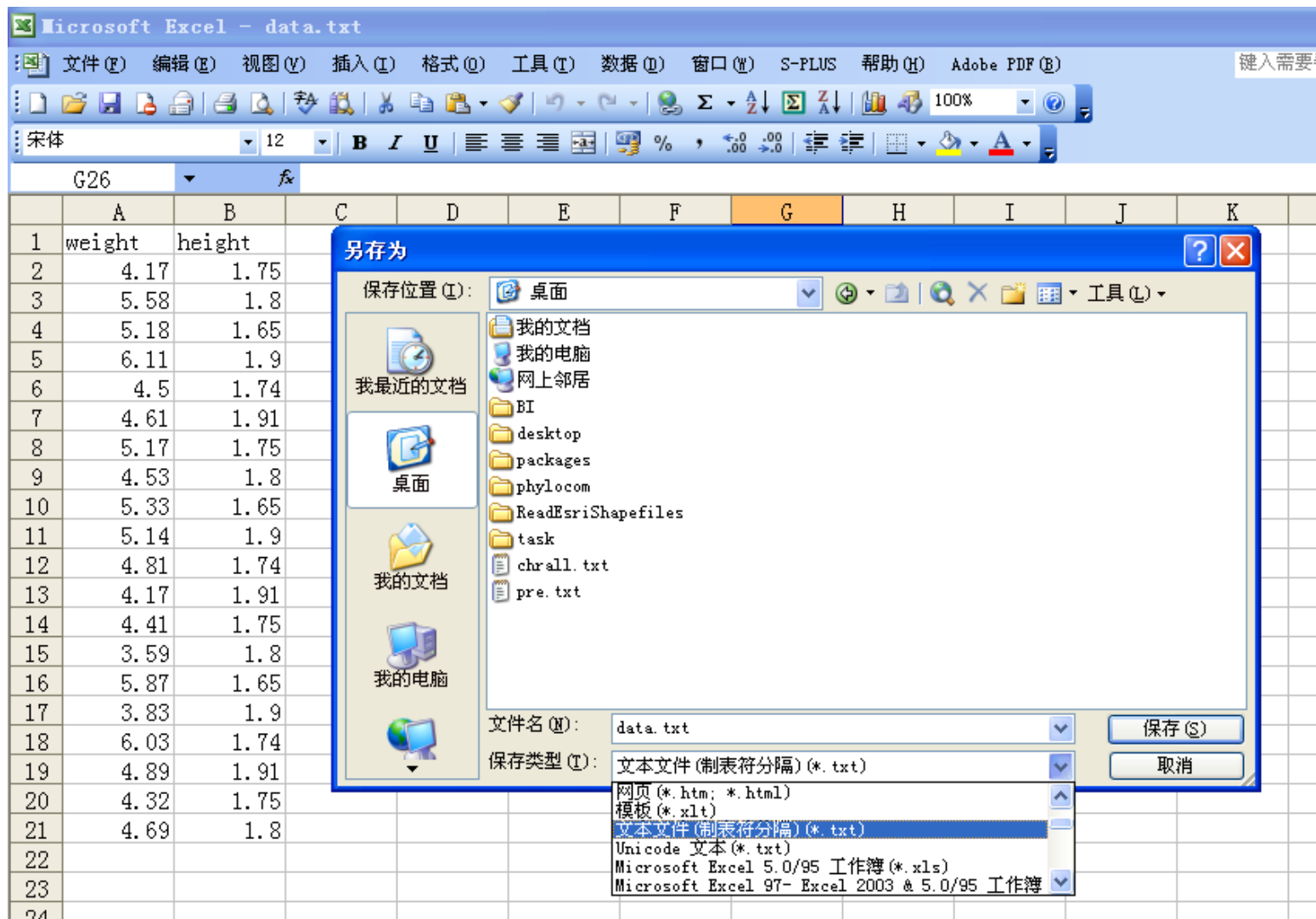


图14 在Excel中将数据存为txt文件

## 例：t检验(续)

一般从txt文档读取数据。每一行作为一个观测值。每一行的变量用制表符，空格或逗号间隔开。

```
read.table("位置", header=T)
```

```
read.csv("位置", header=T)
```

#从外部读取数据

```
data1<-read.table("d:/t.test.data.txt", header=T)
```

```
bmi<- data1$weight/data1$height^2
```

```
t.test(bmi, mu=22.5) #t检验
```



## 练习三：数据读取和t检验

将表2中的数据录入Excel中，另存为t.test.txt文件。

用read.table函数读取该文件。

```
t.test.data <- read.table("X:/t.test.txt", header=TRUE)
```

对变量t.test.data中的

```
attach(t.test.data)
```

```
ratio<-weight/height^2
```

```
t.test(ratio)
```

## 例：单因素方差分析

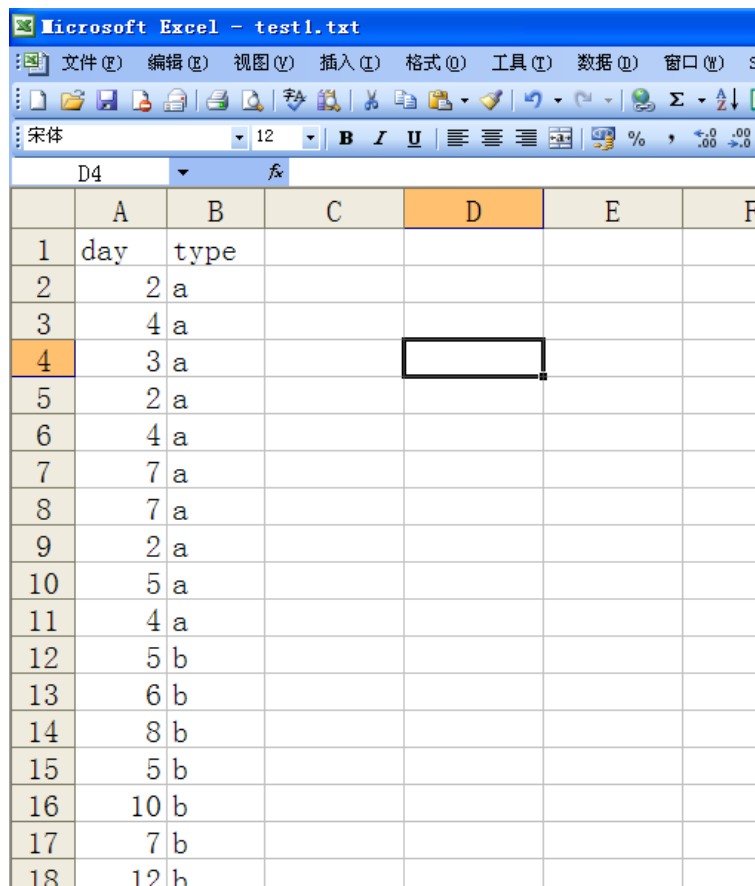
将三种不同菌型的伤寒病毒a,b,c分别接种于10, 9, 和11只小白鼠上, 观察其存活天数, 问三种菌型下小白鼠的平均存活天数是否有显著差异。

a菌株: 2, 4, 3, 2, 4, 7, 7, 2, 5, 4

b菌株: 5, 6, 8, 5, 10, 7, 12, 6, 6

c菌株: 7, 11, 6, 6, 7, 9, 5, 10, 6, 3, 10

# 准备数据表



	A	B	C	D	E	F
1	day	type				
2	2	a				
3	4	a				
4	3	a				
5	2	a				
6	4	a				
7	7	a				
8	7	a				
9	2	a				
10	5	a				
11	4	a				
12	5	b				
13	6	b				
14	8	b				
15	5	b				
16	10	b				
17	7	b				
18	12	b				

图15 数据表的准备

day和type 各为一列

## 例：方差分析(续)

#数据读取，将test1.txt中的内容保存到bac中， header=T表示保留标题行。

```
bac<-read.table("d:/anova.data.txt",header=T)
```

#将ba数据框中的type转换为因子(factor)

```
bac$type<-as.factor(bac$type)
```

```
ba.an<-aov(lm(day~type, data=bac))
```

```
summary(ba.an)
```

```
boxplot(day~type,data=bac,col="red")
```

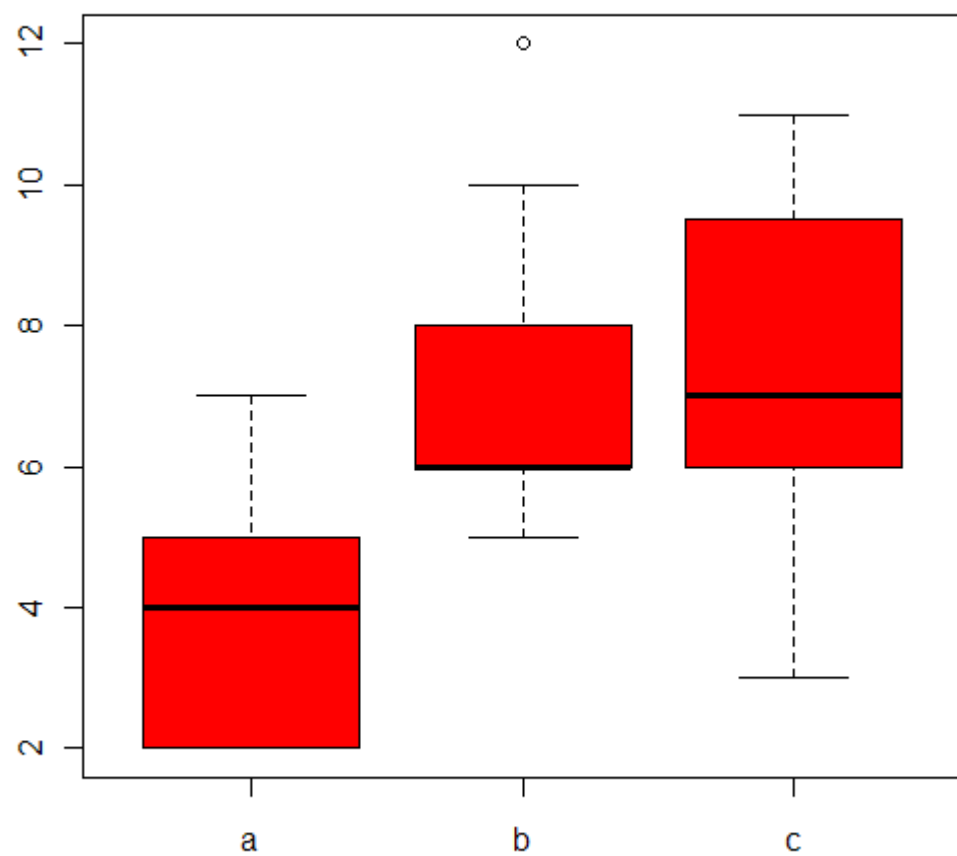


图16 三种菌型对小白鼠影响的箱线图

## 练习四：方差分析和箱线图绘制

- 1 在**excel**中准备数据表
- 2 用**R**读取数据表
- 3 输入如下命令进行方差分析、绘制箱线图

```
boxplot(day~type,data=bac,col="red")
```

```
ba.an<-aov(lm(day~type,data=bac))
```

```
summary(ba.an)
```

# 向量、矩阵和数据框的创建

有时需要对读入的数据进行操作，将某一向量转换成矩阵，如条件筛选，此时将遇到向量、矩阵和数据框的生成、条件筛选等。

例如：提取前面群落数据表中，物种数 $>30$ 的行，提取其中的某一行，进行分析等。

# 向量的创建

四种类型的向量

字符型

```
character<-c("China", "Korea", "Japan", "UK", "USA",  
"France", "India", "Russia")
```

数值型

```
numeric<-c(1, 3, 6, 7, 3, 8, 6, 4)
```

逻辑型

```
logical<-c(T, F, T, F, T, F, F, T)
```

复数型 略



# 向量的创建

生成向量的函数 `c()` , `rep()` , `seq()` , `" : "`

```
c(2,5,6,9)
```

```
rep(2,times=4)
```

```
seq(from=3, to=21, by=3 )
```

```
[1]  3  6  9 12 15 18 21
```

```
" : "
```

```
1:15
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

通过与向量的组合，产生更为复杂的向量。

```
rep(1:2,c(10,15))
```

# 向量创建——产生随机数

```
runif(10, min = 0, max= 1)
```

```
rnorm(10, mean = 0, sd = 1)
```

几个随机数的相关函数

概率密度

```
dunif(x, min=0, max=1, log = FALSE)
```

累积函数

```
punif(q, min=0, max=1, ...)
```

分位数

```
qunif(p, min=0, max=1, ...)
```

随机均匀分布

```
runif(n, min=0, max=1)
```

# 矩阵的创建

生成矩阵的函数 `dim()` 和 `matrix()`

`dim()` 定义矩阵的行列数，例如：

```
x <- 1:12
```

```
dim(x) <- c(3,4)
```

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

```
matrix.x <- matrix(1:12,nrow=3,byrow=T)
```

`t(x)` #转置

为行或列添加名称：

```
row.names()
```

# 数据框的创建

创建数据框的函数: `data.frame()`, `as.data.frame()`, `cbind()`,  
`rbind()`

`cbind()` # 按列组合成数据框

`rbind()` # 按行组合成数据框

`data.frame()` #生成数据框

`head()` #默认访问数据的前6行

# 列表的创建

列表可以是不同类型甚至不同长度的向量 (数值型, 逻辑型, 字符型等等)、数据框甚至是列表的组合。

`list()`

例如

`list(character, numeric, logical, matrix.x)`

# 类的判断

## 对象类型判断

**mode()** 判断存储的类型

**class()** 判断数据的类

根据数据的类，采用相应的处理方法。

以下函数，主要用在函数处理异常时使用，目的是增强程序的稳健性。

**is.numeric()** #返回值为**TRUE**或**FALSE**

**is.logical()** #是否为逻辑值

**is.character()** #是否为字符串

**is.null()** #是否为空

**is.na()** #是否为na

# 类的转换

`as.numeric()` #转换为数值型

`as.logical()` #转换为逻辑型

`as.character()` #转换为字符串

`as.matrix()` #转换为矩阵

`as.data.frame()` #转换为数据框

`as.factor()` #转换为因子

# 向量内的元素引用

### Dalgaard书中的例子

```
intake.pre <- c(5260, 5470, 5640, 6180, 6390, 6515, 6805,  
               7515, 7515, 8230, 8770)
```

```
intake.post <- c(3910, 4220, 3885, 5160, 5645, 4680, 5265,  
                5975, 6790, 6900, 7335)
```

```
intake.pre[5];           #引用第5个元素
```

```
intake.pre[c(3,5,7)]     #引用第3, 5, 7个元素
```

```
v <- c(3,5,7); intake.pre[v];
```

```
intake.pre[1:5];         #引用第1到5个元素
```

```
intake.pre[-c(3,5,7)]    #去除第3, 5, 7元素
```



# 数据框内元素的引用

```
intake <- data.frame(intake.pre, intake.post)
```

引用数据框中的元素

(1) `$` 引用列，后面为列的名称

例如 `intake$intake.pre`

(2) `[,]` 方括号引用，逗号前为行，逗号后为列

`intake[,1];` 引用第1列

`intake[5,];` 引用第5行

`intake[5,1];` 引用第1列，第5行

`i = 1:5; intake[i,]` 引用1到5行

# 列表内元素的引用

列表内元素的引用可以用”`[[ ]]`”

如 `list1[[c(1,2,3)]]`

上述放于”`[]`”内的数字，称为下标。通过下标的变化，可以方便的访问向量、数据框、矩阵、列表内的各元素。熟悉下标的用法对掌握循环结构是非常重要的。

# 访问数据框内的元素

直接调用数据框内的列，以列的名称作为向量的名称

**attach()**

**detach()**

在函数内部，对数据进行相应调整

**with()** #with函数内部形成一个空间，在这个空间中，函数可以对列访问。

**within()**

**transform()** #数据的转换, 如取log

**subset()** #取数据的子集

**apply()** #对数据表或矩阵应用某个函数，可减少循环

# 条件筛选

条件筛选是先对变量否满足条件进行判断，满足为**TRUE**，不满足为**FALSE**。  
之后再用逻辑值对向量内的元素进行筛选。

```
intake.pre > 7000
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE  
TRUE TRUE TRUE
```

```
intake.pre[intake.pre > 7000]
```

```
intake.pre > 7000 & intake.pre <= 8000
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE  
TRUE FALSE FALSE
```

```
intake.post[intake.pre > 7000 & intake.pre <= 8000]
```

## 练习五：下标和条件筛选

创建一个2到50的向量 `vector1`

2, 4, 6, 8, ..., 48, 50

```
vector1<-seq(from=2, to=50, by=2)
```

选取`vector1`中的第20个元素 `vector1[20]`

选取`vector1`中的第10, 15, 20个元素

```
vector1[c(10,15,20)]
```

选取`vector1`中的第10到20个元素

```
vector1[10:20]
```

选取`vector1`中值大于40的元素

```
vector1[vector>40]
```

# 排 序

将向量中的元素按照一定顺序排列。

`sort()` 按数值大小排序

举例：

```
intake$intake.pre
```

```
sort(intake$intake.pre)
```

`order()` 默认给出从小到大的出现序号。

```
order(intake$post)
```

```
o <- intake[order(intake$intake.pre),]
```

# 工作空间image

R的所有对象都在计算机内存的工作空间中。

**ls()** 列出工作空间中的对象

**rm()** 删除工作空间中的对象

**rm(list=ls())** 删除空间中所有对象

**save.image()** 保存工作镜像

**sink()** 将运行结果保存到指定文件中

**getwd()** 显示当前工作文件夹

**setwd()** 设定工作文件夹

可将结果保存在**image**中，形式为**.Rdata**文件，里面保存了R当前工作空间中的各种对象，包括函数。

# 输入历史 history

- 输入的命令，在R中作为历史**history**保存，可输入函数
- **history()**      # 查看输入历史
- **history(Inf)**    # 查看打开R之后所有的输入
- 可用向上或向下的箭头切换输入的行。



# 练习六：了解工作路径

## 1 查看当前R工作的空间目录

```
getwd()
```

## 2 将R工作的路径设置为 d:/data/

```
setwd("d:/data")
```

```
save.image("example.Rdata")
```

```
load("example.Rdata")
```

## 3 历史

```
history(Inf)
```

# 三 脚本编程

**Scripting**

# 脚本语言

## 脚本语言

脚本语言 (**scripting language**) 又称动态语言，是依靠解释器完成相应的功能的一类计算机语言，通常以**ASCII**码的文本格式保存源程序。

## 特点

脚本语言语法和结构通常比较简单，不需要编译，通过解释器对脚本进行解释，从而给出结果，能用简单的代码完成复杂的功能，但是速度较慢。

## 常见的脚本语言

**Windows**批处理程序, **PHP**, **Perl**, **Python**, **Ruby**, **JavaScript**等。

# 集成开发环境

很多计算机语言都有IDE (integrated development environment), 即集成开发环境,如 Windows的VisualStudio, Visual Basic等等。

但是R通常无需集成开发环境, 脚本在一般的文本编辑器里即可编辑。

如 Windows自带的记事本,Notepad++, UltraEdit, TinnR。

Linux下的Vi, Vim, Emacs等等。

在代码较多的情况下, 常需要对行数、函数、括号、函数选项等进行高亮显示, 设置成不同的颜色, 以减少错误的发生。

# 编辑器

R自带的脚本编辑器

Editplus ( [www.editplus.com](http://www.editplus.com) )

TinnR ( <http://www.sciviews.org/Tinn-R/> )

Ultraedit ( [www.ultraedit.com/](http://www.ultraedit.com/) )

Emacs ( [www.gnu.org/software/emacs/](http://www.gnu.org/software/emacs/) )

Notepad++ 与NpptoR组合

( <http://notepad-plus.sourceforge.net/> )

记事本或写字板 等等

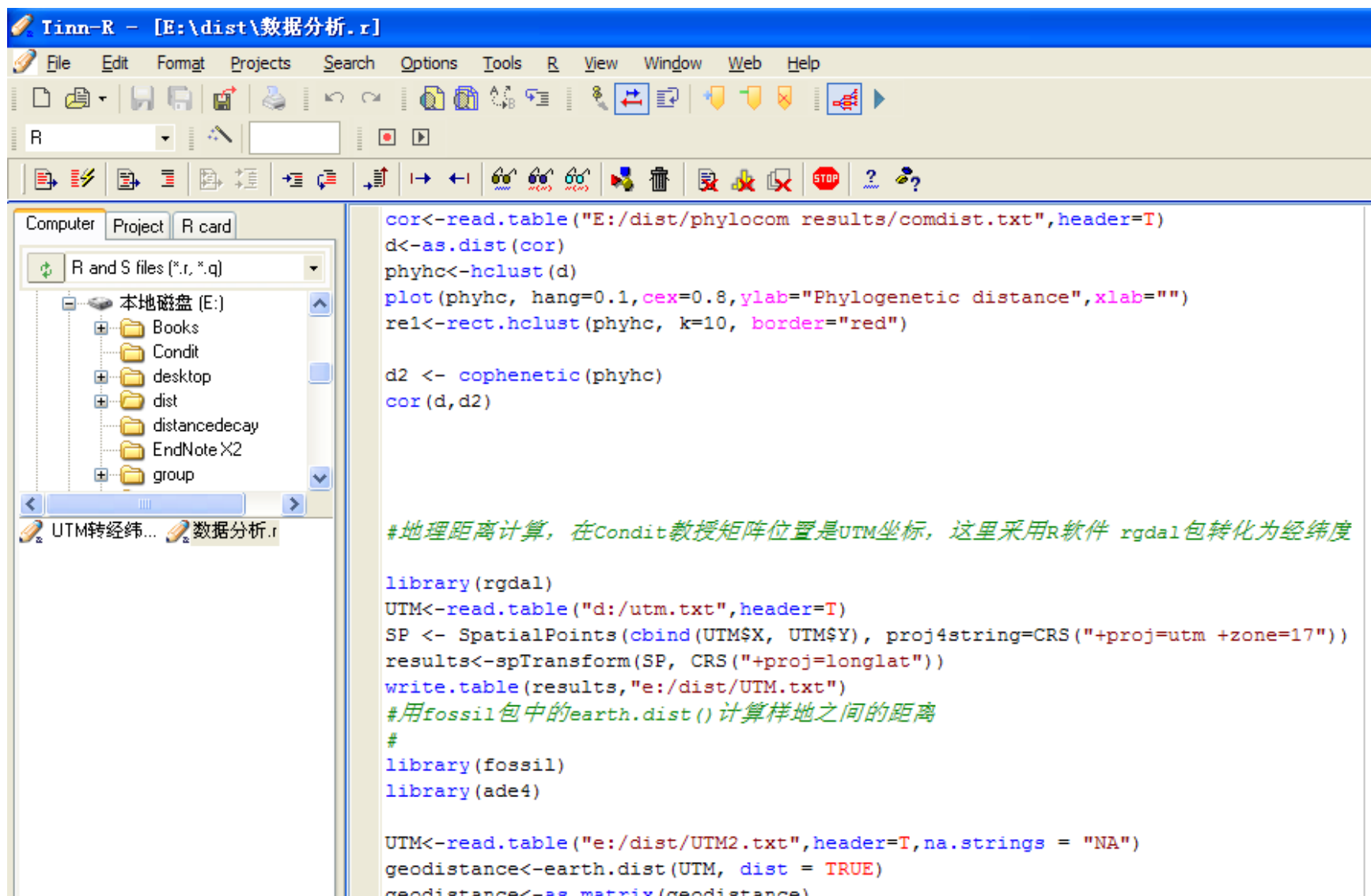
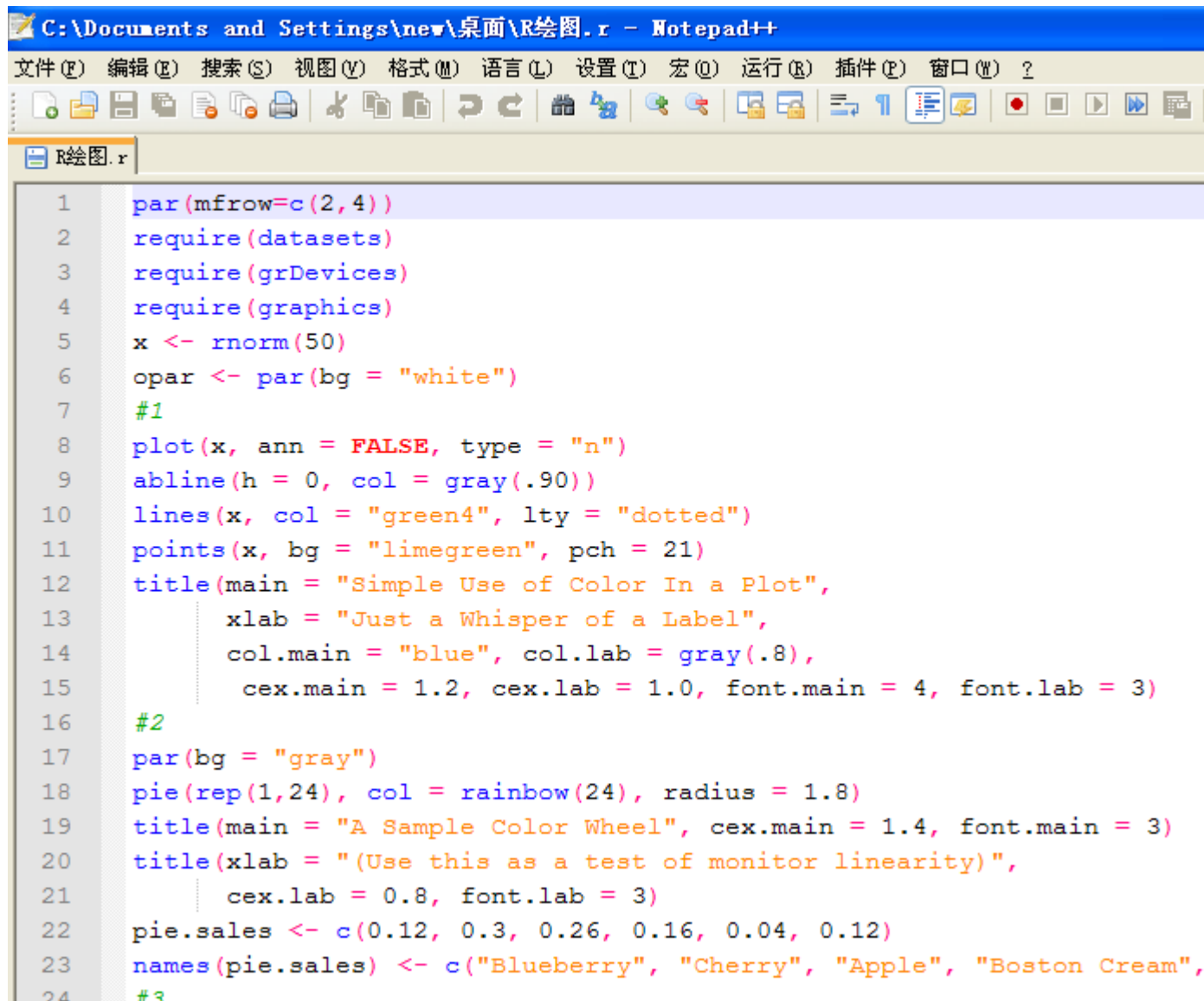


图17 TinnR对R脚本的高亮显示

The image shows a Notepad++ window with the title bar 'C:\Documents and Settings\new\桌面\R绘图.r - Notepad++'. The menu bar includes '文件(F)', '编辑(E)', '搜索(S)', '视图(V)', '格式(M)', '语言(L)', '设置(T)', '宏(O)', '运行(R)', '插件(P)', '窗口(W)', and '?'. The toolbar contains various icons for file operations and editing. The active file is 'R绘图.r'. The code is as follows:

```
1  par(mfrow=c(2,4))
2  require(datasets)
3  require(grDevices)
4  require(graphics)
5  x <- rnorm(50)
6  opar <- par(bg = "white")
7  #1
8  plot(x, ann = FALSE, type = "n")
9  abline(h = 0, col = gray(.90))
10 lines(x, col = "green4", lty = "dotted")
11 points(x, bg = "limegreen", pch = 21)
12 title(main = "Simple Use of Color In a Plot",
13       xlab = "Just a Whisper of a Label",
14       col.main = "blue", col.lab = gray(.8),
15       cex.main = 1.2, cex.lab = 1.0, font.main = 4, font.lab = 3)
16 #2
17 par(bg = "gray")
18 pie(rep(1,24), col = rainbow(24), radius = 1.8)
19 title(main = "A Sample Color Wheel", cex.main = 1.4, font.main = 3)
20 title(xlab = "(Use this as a test of monitor linearity)",
21       cex.lab = 0.8, font.lab = 3)
22 pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
23 names(pie.sales) <- c("Blueberry", "Cherry", "Apple", "Boston Cream",
24 #3
```

图18 Notepad++对R脚本的高亮显示

## 例-线性回归

- 对一批涂料进行研究，确定搅拌速度对杂质含量的影响，数据如下，试进行回归分析

表3 搅拌速度对涂料中杂质的影响

转速 rpm	20	22	24	26	28	30	32	34	36	38	40	42
杂质率 %	8.4	9.5	11.8	10.4	13.3	14.8	13.2	14.7	16.4	16.5	18.9	18.5



# 脚本举例

#将以下代码粘贴到编辑器中，另存为**regression.r**文件。

```
rate<-c(20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42)
impurity <-c(8.4, 9.5, 11.8, 10.4, 13.3, 14.8, 13.2, 14.7,
            16.4, 16.5, 18.9, 18.5)

plot(impurity~rate)

reg<-lm(impurity~rate)

abline(reg,col="red")

summary(reg)
```

# 运行脚本

## 三种运行方式

### 1 通过**source()**函数运行

```
source("d:/regression.r")
```

### 2 通过R脚本编辑器运行

路径: RGui>File>Open Script #Ctrl+R运行

### 3 直接粘贴到R控制台

```
ctrl+c, ctrl+v
```

第三种最为简单直接

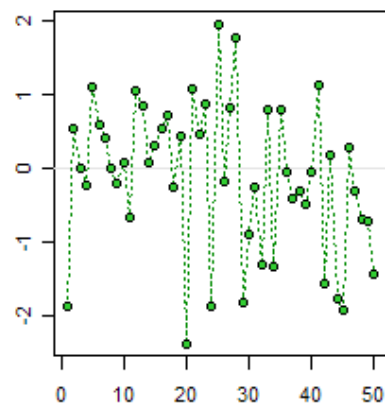
## 练习十：R脚本运行

将R命令粘贴到记事本中，另存为**regression.R**文件。

分别通过三种方式运行R脚本。

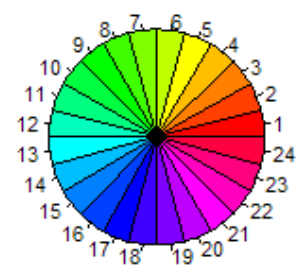
## 四 R绘图

Simple Use of Color In a Plot



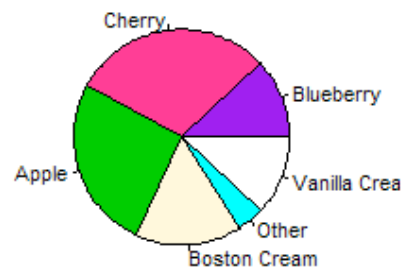
*Just a Whisper of a Label*

A Sample Color Wheel



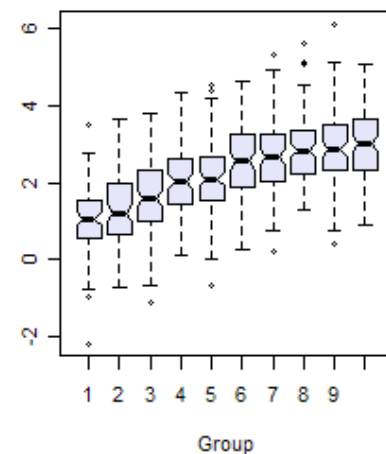
*(Use this as a test of monitor linearity)*

January Pie Sales

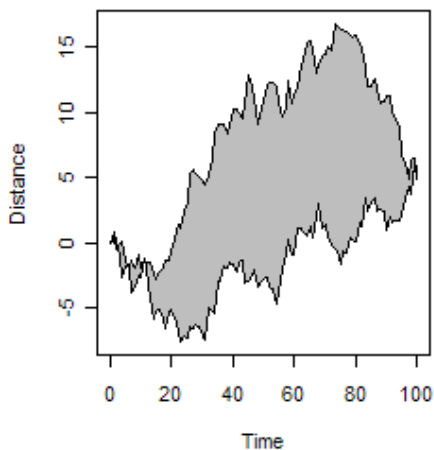


*(Don't try this at home kids)*

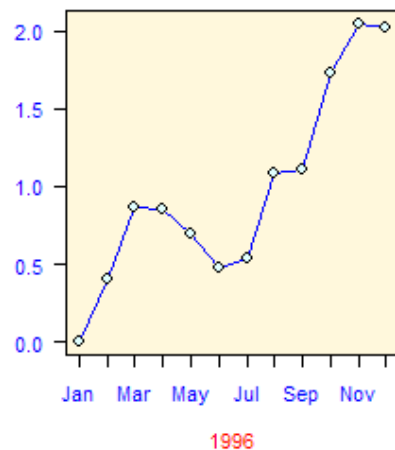
Notched Boxplots



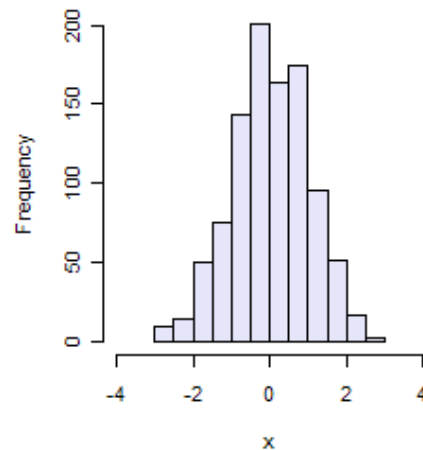
Distance Between Brownian Motions



The Level of Interest in R



1000 Normal Random Variates



A Topographic Map of Maunga Whau  
10 Meter Contour Spacing

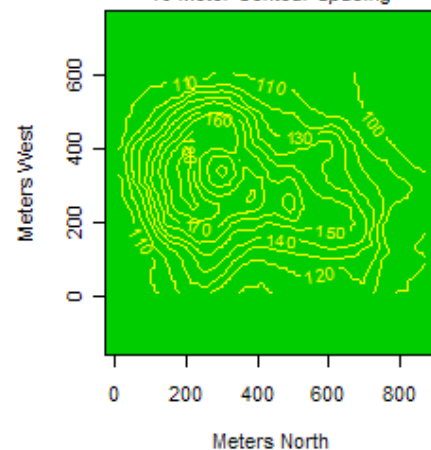
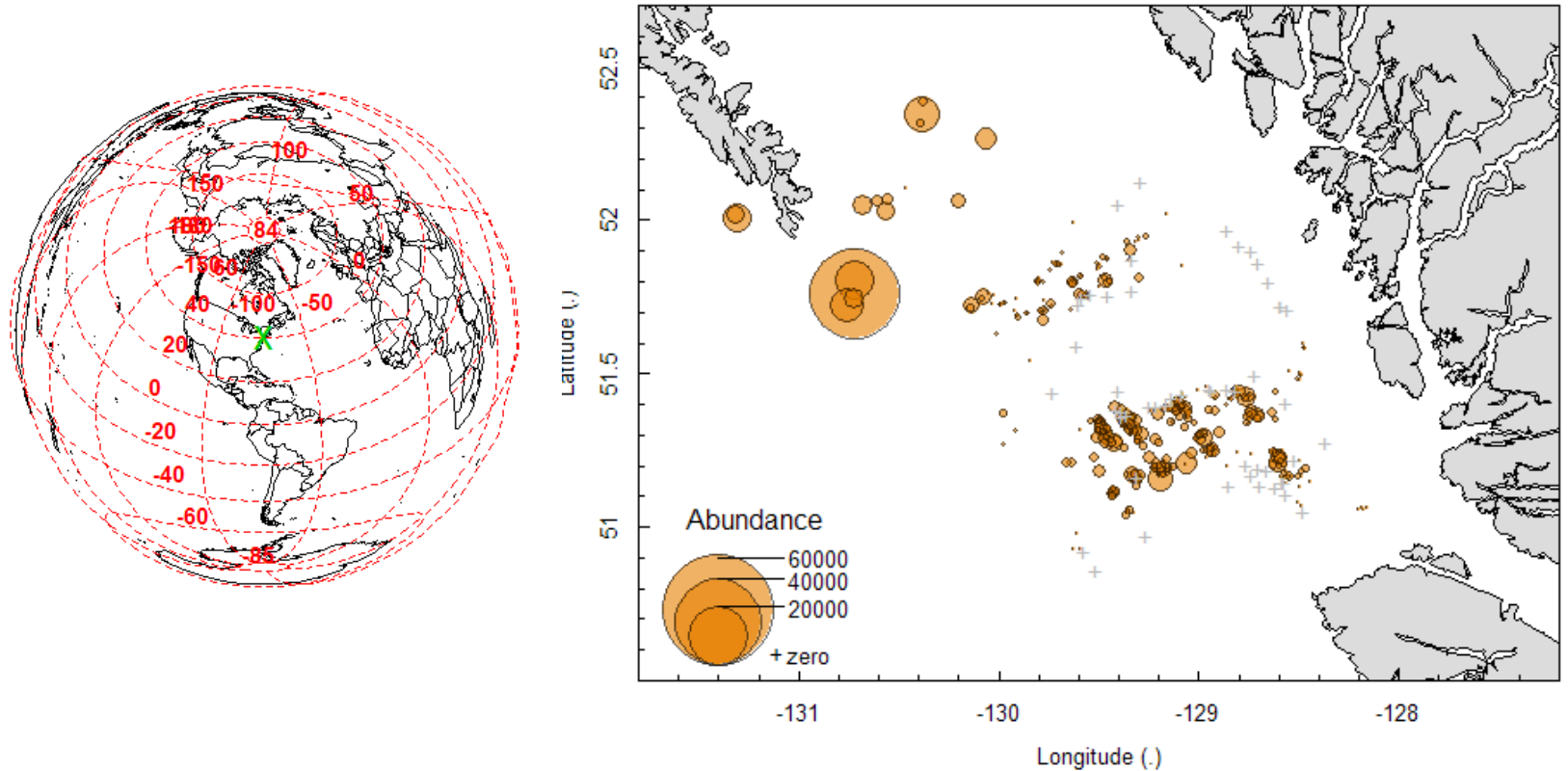


图19 R绘制的图形

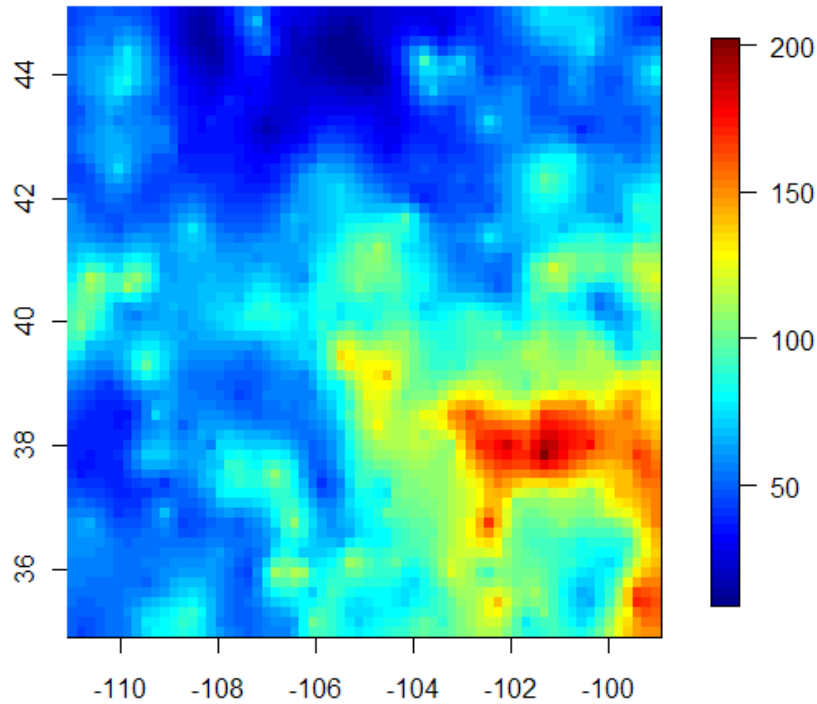
## 绘制地图



- 图20 左图 maps包 map() 右图 PBSmapping包 addBubbles()

# 绘制地图

Plot of first (black) and second (red)  
nearest neighbours



- 图21 **fields** 包实例      **spdep** 包实例

# 空间分析绘图

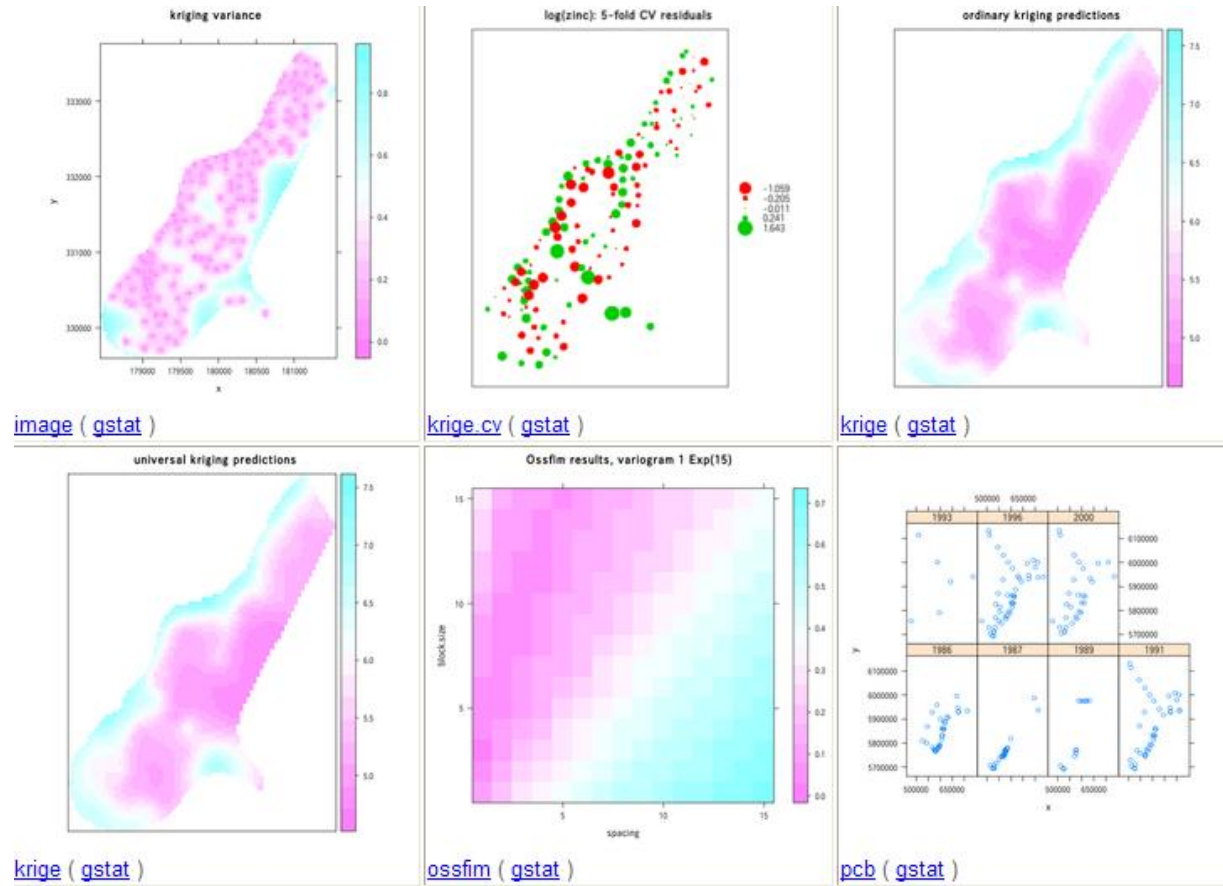


图22 `gstat`程序包实例



# R绘图功能

R具备卓越的绘图功能，通过参数设置对图形进行精确控制。绘制的图形能满足出版印刷的要求，可以输出Jpg、tiff、eps、emf、pdf、png等各种格式。

通过与GhostScript软件的结合，可以生成600dpi，1200dpi的等各种分辨率和尺寸的图形。

绘图是通过绘图函数结合相应的选项完成的。

绘图函数包括高级绘图函数和低级绘图函数。

# 高级绘图函数

`plot()`

绘制散点图等多种图形, 根据数据的类, 调用相应的函数绘图

`hist()`

频率直方图

`boxplot()`

箱线图

`stripchart()`

点图

`barplot()`

柱状图

`dotplot()`

点图

`piechart()`

饼图

`matplot()`

数学图形

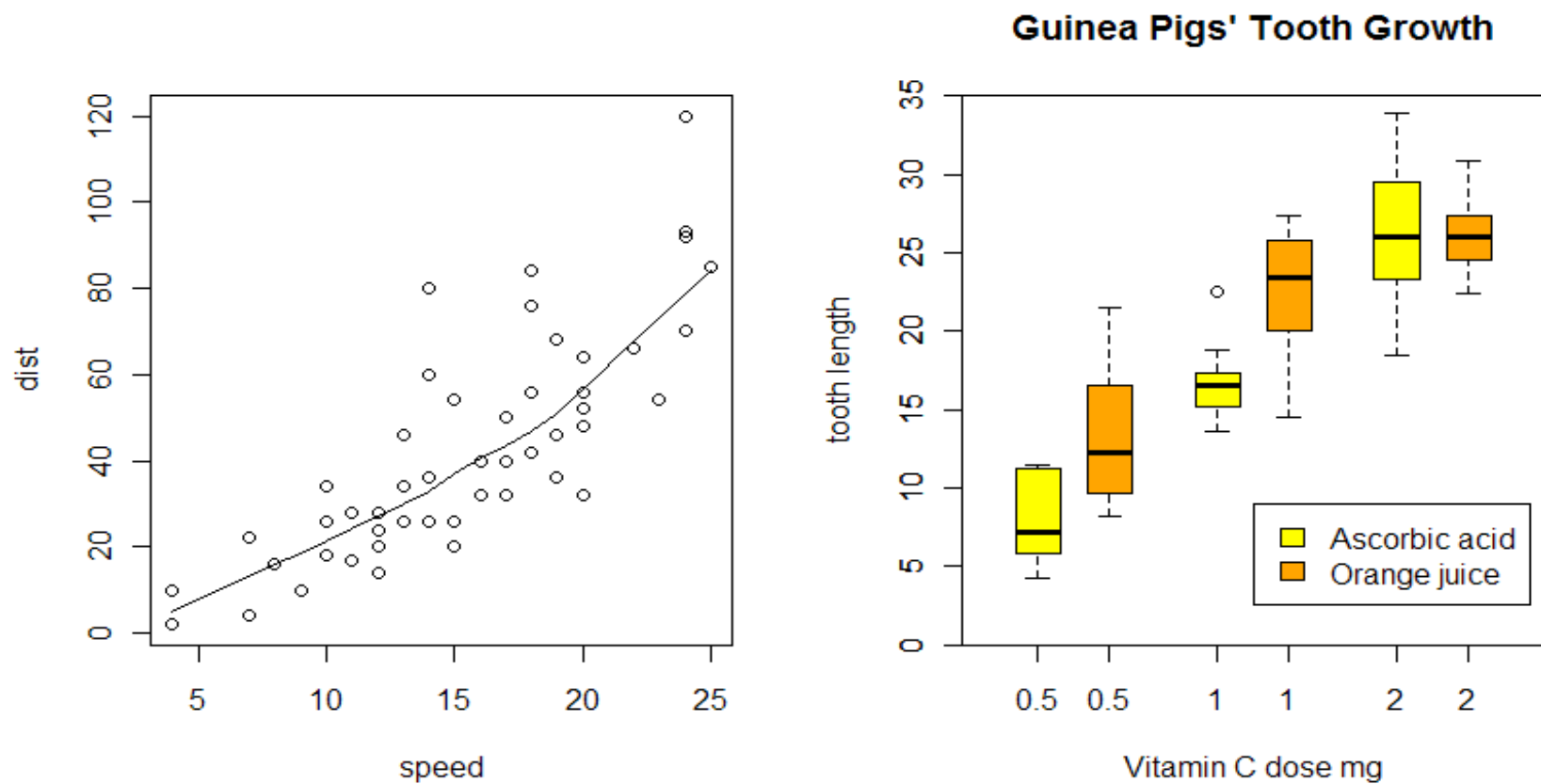


图23 散点图与箱线图

## 低级绘图函数

<code>lines()</code>	添加线
<code>curve()</code>	添加曲线
<code>abline()</code>	添加给定斜率的线
<code>points()</code>	添加点
<code>segments()</code>	折线
<code>arrows()</code>	箭头
<code>axis()</code>	坐标轴
<code>box()</code>	外框
<code>title()</code>	标题
<code>text()</code>	文字
<code>mtext()</code>	图边文字

# 绘图参数

参数用在函数内部，在没有设定值时使用默认值。

`font` = 字体

`lty` = 线类型

`lwd` = 线宽度

`pch` = 点的类型,

`xlab` = 横坐标

`ylab` = 纵坐标

`xlim` = 横坐标范围

`ylim` = 纵坐标范围

也可以对整个要绘制图形的各种参数进行设定

参见 `par()`

## 举例：绘图

生成0到2之间的50个随机数，分别命名为 $x$ ,  $y$

```
x <- runif(50,0,2)
```

```
y <- runif(50,0,2)
```

绘图：将主标题命名为“散点图”，横轴命名为“横坐标”，纵轴命名为“纵坐标”

```
plot(x, y, main="散点图", xlab="横坐标", ylab="纵坐标")
```

```
text(0.6,0.6,"text at (0.6,0.6)")
```

```
abline(h=.6,v=.6)
```

散点图

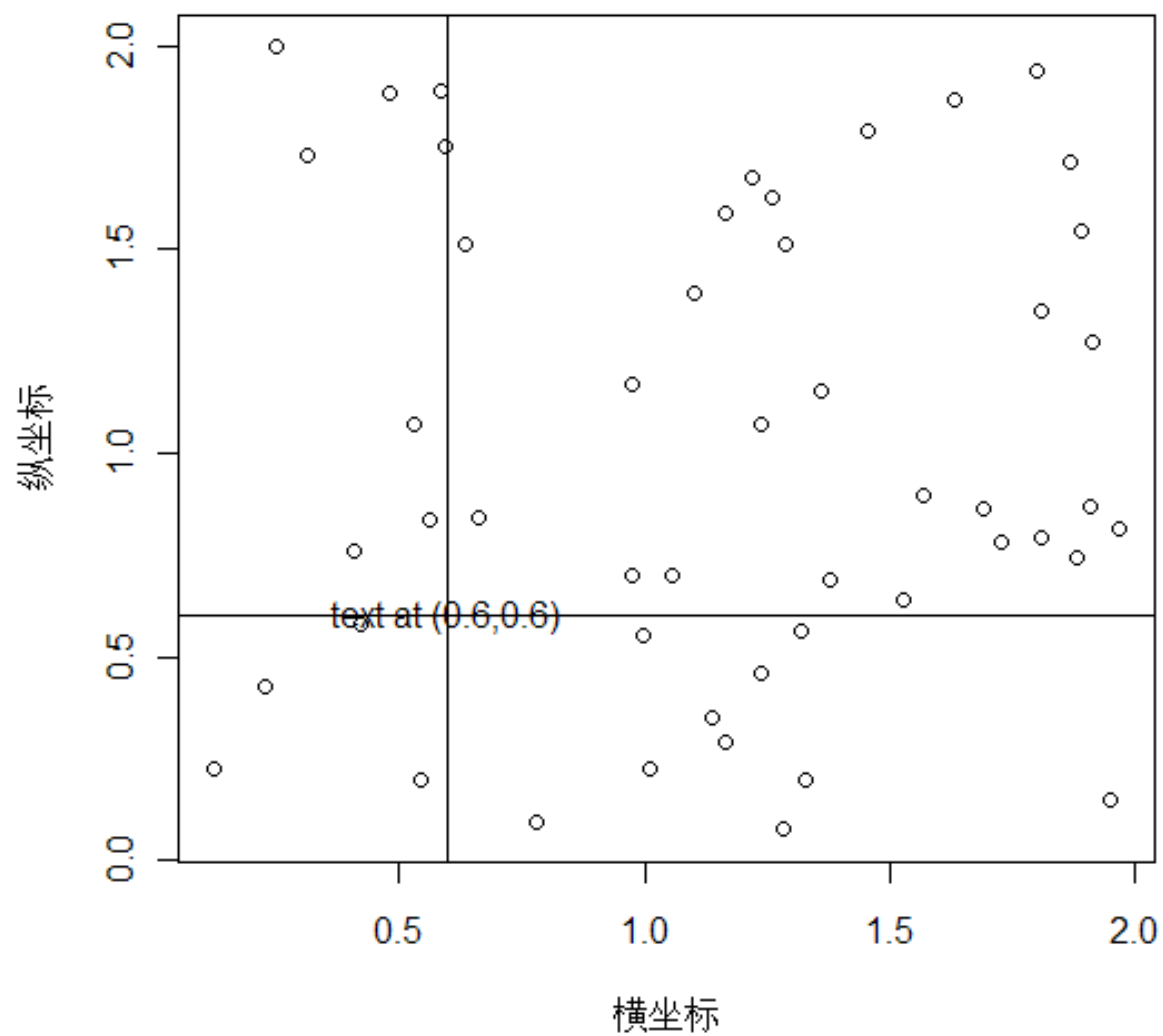


图24 绘图举例

## 例：分步绘图

1. 打开绘图窗口，不绘制任何对象

```
plot(x, y, type="n", xlab="", ylab="", axes=F)
```

2. 添加坐标点

```
points(x,y)
```

3. 添加坐标轴

```
axis(1); axis(at=seq(0.2,1.8,0.2), side=2)
```

4. 补齐散点图的边框

```
box()
```

5. 添加标题、副标题、横轴说明、纵轴说明

```
title(main="Main title", sub="subtitle", xlab="x-label",  
      ylab="y-label")
```



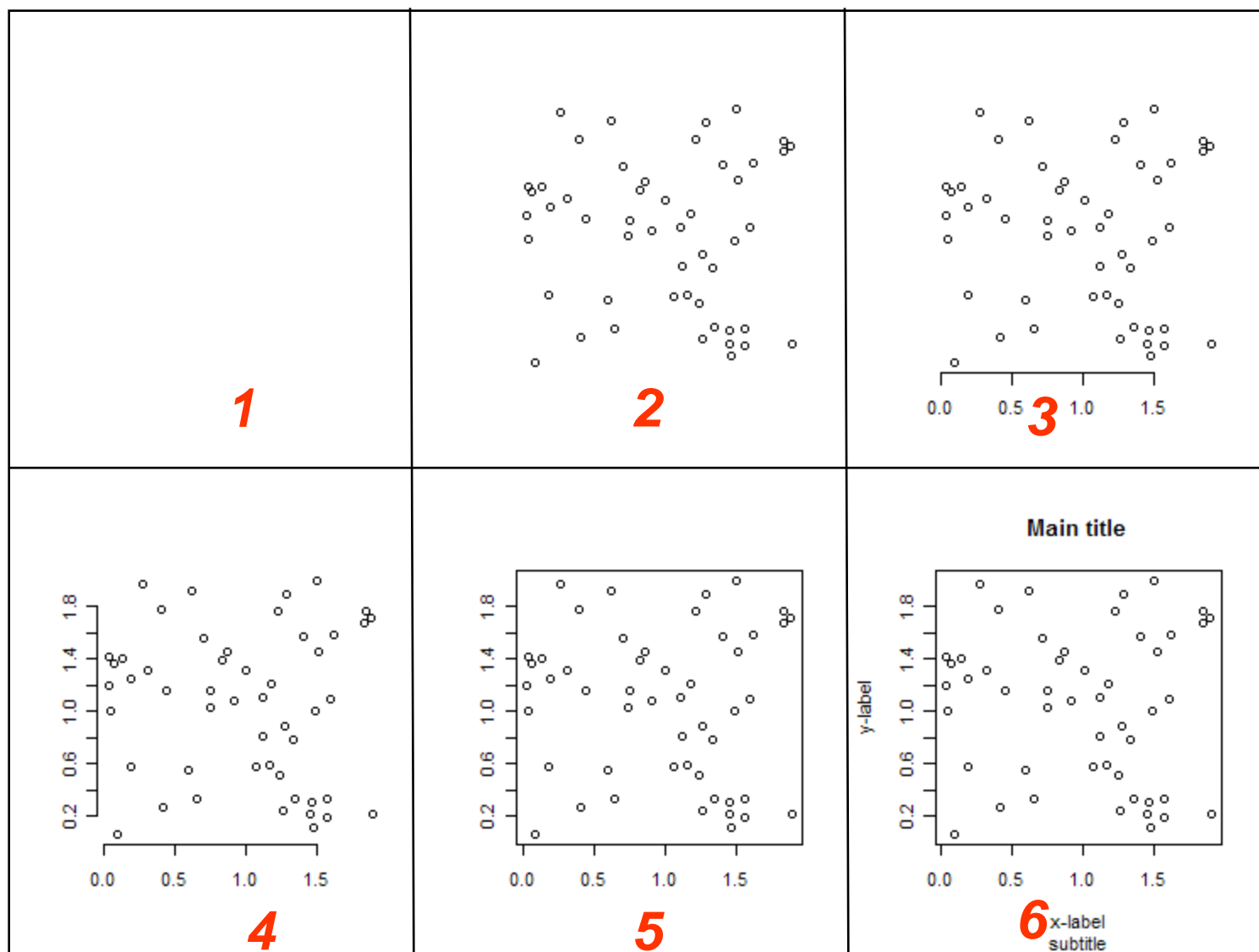


图25 分步绘图

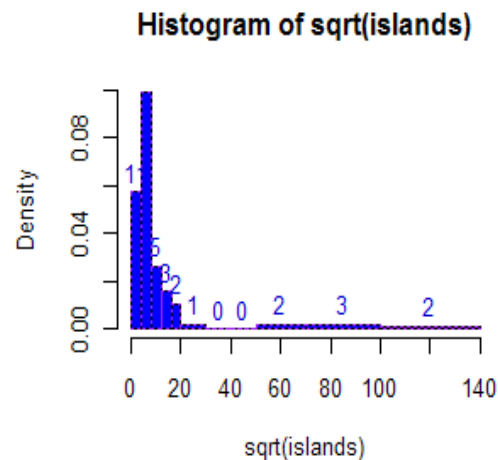
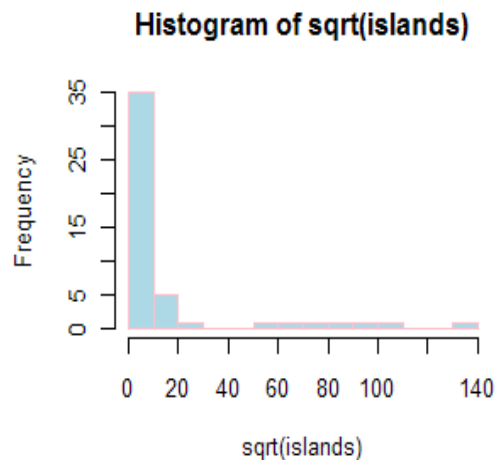
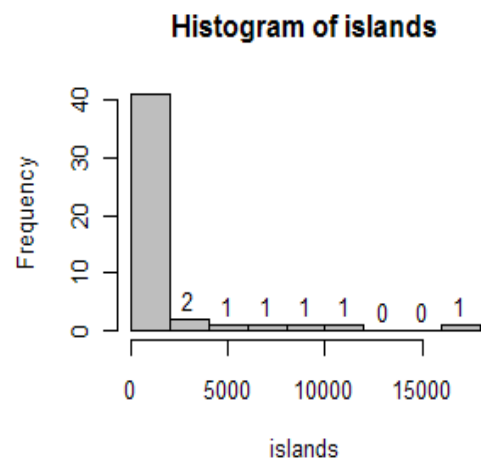
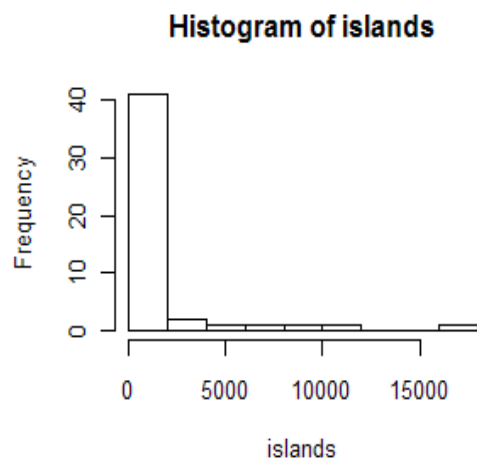
# 一页多图

图26 一页多图

`par()`

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

...



# 在原有图形上添加元素

举例:

```
x <- rnorm(100)           # 生成随机数
hist(x, freq=F)           # 绘制直方图
curve(dnorm(x), add=T)    # 添加曲线
h <- hist(x, plot=F)      # 绘制直方图
ylim <- range(0, h$density, dnorm(0)) # 设定纵轴的取值范围
hist(x, freq=F, ylim=ylim) # 绘制直方图
curve(dnorm(x), add=T, col="red") # 添加曲线
```

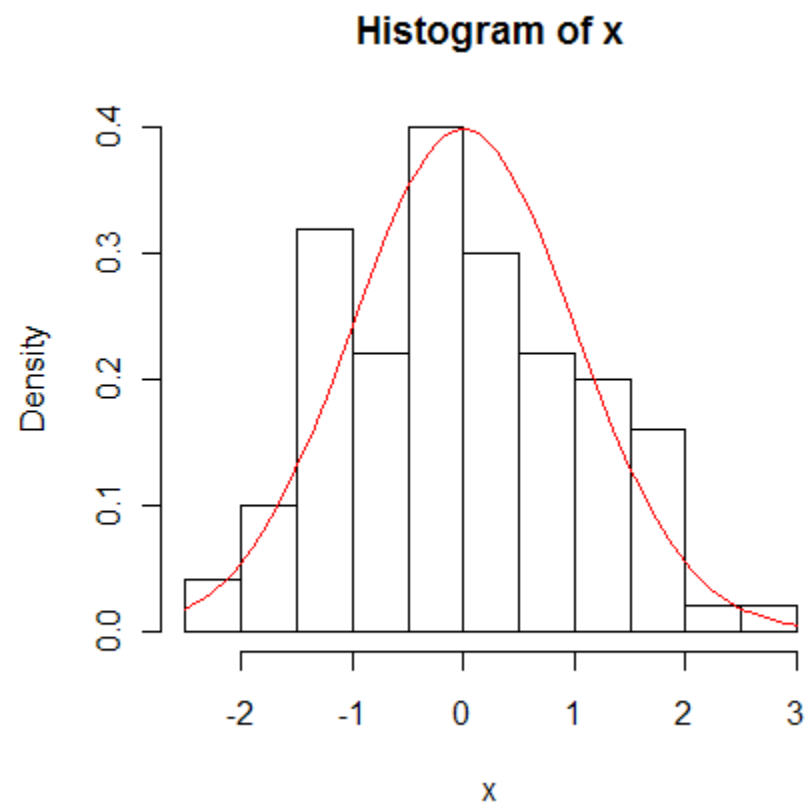
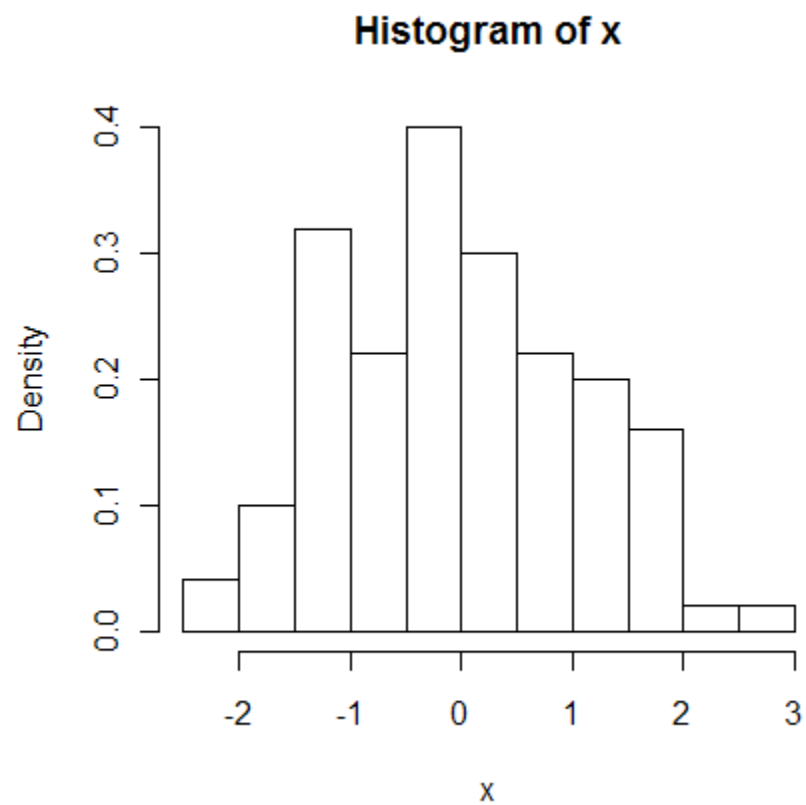


图27 在原有直方图上添加曲线

## 练习八：绘图练习

练习1 将**Rplots.r**中的代码拷贝到R控制台中，查看R绘制的图形。

练习2 对例进行回归分析，并绘制散点图，并为散点图添加回归直线。

```
plot(impurity~rate)
```

```
reg<-lm(impurity~rate)
```

```
abline(reg,col="red")
```

```
summary(reg)
```

## 五 编写函数

# 编程基础

R可以灵活的编写程序，用户自己编写的程序可以直接调用。编程时无需声明变量的类型，这与C, C++等语言不同。

## 基本格式

```
函数名 <- function(数据, 参数1= 默认值, ...)  
{  
    异常处理;  
    表达式(循环/判别);  
    return(返回值);  
}
```

函数内部也可用#添加注释

# 函数实例

```
data2mat <-  
function(data = data)  
{  
  if (!any(colnames(data) == "abundance"))  
    stop("A column named \"abundance\" must be speciefied.")  
  if (!any(is.integer(data$abundance)))  
    stop("Number of individuals must be integer!")  
  col <- which(colnames(data) == "abundance")  
  data1 <- data[,-col]  
  abundance <- as.numeric(data[,col])  
  result1 <- data.frame(rep(NA, sum(abundance)))  
  colnames(result1) <- "plots"  
  for (i in 1:(ncol(data)-1))  
  {  
    result1[, i] <- rep(as.character(data[, i]), abundance)  
  }  
  result <- table(result1)  
  return(result)  
}
```

- 图28 函数实例 data2mat()



# 程序流程控制 `if`

`if` (条件) 表达式

`if` (条件) 表达式1 `else` 表达式2

举例

```
p = 0.03  
  
{  
  
    if (p<=0.05)  
  
        print ("p <= 0.05!")  
  
    else  
  
        print ("p > 0.05!")  
  
}
```

# 循环 for, while

**for**(变量 in 向量) 表达式

用法:

```
for(i in 1:10) print(i)
```

**while**(条件) 表达式

用法:

```
i <- 1
```

```
while(i<10)
```

```
{
```

```
  print(i)
```

```
  i <- i + 1
```

```
}
```

# 返回值

- 返回值表示函数输出的结果。
- 返回值必须是一个对象。
- R默认将最后一行作为返回值。
- 如果函数的结果需要有多个返回值，可以创建一个`list()`，并返回该对象。
- 也可以用`return()`函数，设定返回值。
- 但是一个函数的返回的对象只有一个。

# 异常处理

- 如数据输入不能满足要求，或者参数设定错误等等，可能造成函数给出错误的结果，则需要对函数的运行过程发出警告或终止，以提高程序的稳健性。
- 警告的写法

```
if (any (is.na (inputdata)))
```

```
inputdata <- na.omit(inputdata)
```

```
cat("NAs are found in the input data, and has been  
removed.\n")
```

- 终止的写法

```
if (any (is.na (xx)))
```

```
stop("NAs are not allowed!\n")
```

# 函数举例

问题：输入直角三角形的两个边长，求其斜边长。

定义函数：

```
rca1 <- function(x,y)
{
  z <- x^2 + y^2
  result <- sqrt(z)
  return(result)
}
```

调用函数：

```
rca1(3,4)
```

## 练习九 编写函数

编写一个函数，给出两个数之后，直接给出这两个数的平方和。

```
sqtest<-function(x, y)
```

```
{
```

```
  z1=x^2;
```

```
  z2=y^2;
```

```
  z3=z1+z2;
```

```
  z3
```

```
}
```

## 六 数据保存

# 数据保存

`write.table()`

`write.csv()`

`save.image()`

`sink()`

`unlink()`

若有**LaTeX**基础，可以用

**Sweave()** 函数

该函数能将脚本、程序说明和运算结果直接保存成**.tex**文件，用**LaTeX**编译成**pdf**文件。



# 主要讲了什么？

1. R是开源的统计绘图软件，也是一种脚本语言，有大量的程序包可以利用。
2. R中的向量、列表、数组、函数等都是对象，可以方便的查询和引用，并进行条件筛选。
3. R具有精确控制的绘图功能，生成的图可以另存为多种格式。
4. R编写函数无需声明变量的类型，能利用循环、条件语句，控制程序的流程。

# 推荐的教材

- Crawley *Statistics an introduction using R*
- Peter Dalgaard *Introductory statistics with R*
- E. Paradis *R for Beginners*
- Verzani *SimpleR.*
- D G Rossiter *Introduction to the R Project for Statistical Computing for use at ITC*
- J. Maindonald *Using R for data analysis and graphics introduction code and commentary using R*
- Venables, W. N. & Ripley, B. D.  
*Modern Applied Statistics with S*

# R网络资源

R主页: <http://www.r-project.org>

R资源列表 NCEAS <http://www.nceas.ucsb.edu/scicomp/software/r>

R Graphical Manual <http://bm2.genes.nig.ac.jp/RGM2/index.php>

统计之都: <http://cos.name/>

QuikR <http://www.statmethods.net/>

丁国徽的R文档: <http://www.biosino.org/R/R-doc/>

R语言中文论坛 <http://rbbs.biosino.org/Rbbs/forums/list.page>

谢谢各位  
敬请指正！