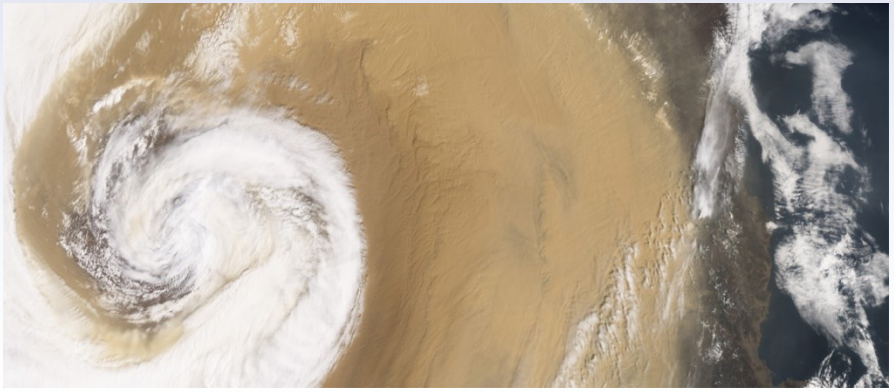
R语言数据可视化分析及大气科学应用

Data Visualization Analysis with R Language and its Application in Atmospheric Sciences

李成才

物理学院大气与海洋科学系



R语言数据可视化分析及大气科学应用

Data Visualization Analysis with R Language and its Application in Atmospheric Sciences

- 周学时：2
- 总学时：34
- 讲课学时：32
- 习题学时（课后）：16
- 上机学时（课后）：16

- 课程体系：任选

- 授课教师：李成才 （副教授）

课程基本目的

- 数据分析在各门学科的研究中都有着十分重要的地位，尤其是我们进入一个大数据的时代，掌握**快捷而便利的分析工具**常常是制胜法宝。
- R语言是近年来国际上快速发展的一个具有强大的统计分析和图形处理功能的、简便而容易掌握的编程语言。
- 本课程的目的就是让学生们掌握利用R语言数据分析方法，培养科学分析的兴趣，增强进行科学研究的基本技能。

为什么要学习R?

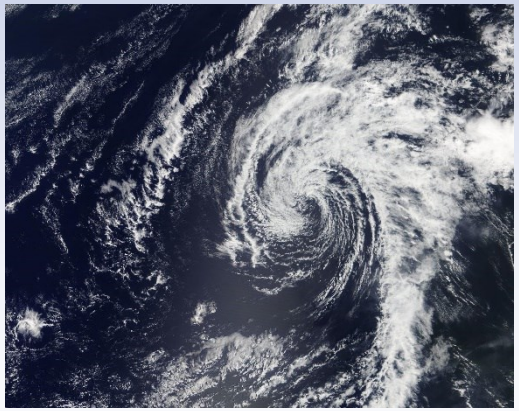
- R语言是一套完整的简便而强大的编程语言，是具有功能强大的数据处理计算、统计分析和制图的软件系统。
- R语言作为一种发展迅速的科研工具，是完全免费、开放源代码的，在GNU协议下系统发展迅速。新的功能和能力每隔几个月就会出现。R社区广泛来自应用领域专家和统计专家。
- 在本课程中，我们将首先学习R系统的基本使用方法，然后逐步对R语言数据分析和图形处理进入更深入的理解。最后，我们将研究如何使用R系统来解决大气科学中的数据分析问题，并且我们将以大气科学数据分析个例的代码实现来丰富R软件包，作为我们对R社区的贡献。

教学方式

- 每一章，教师主要讲授：基本数学和统计方法、基本思想和理论推导，然后介绍R语言实现；最后结合大气科学的应用，提出科学应用需求、初步设计。

- 由学生课后完成详细设计、上机实习、编写代码实现，完成结果和图形分析报告；在第二次课由教师在课堂展示部分同学的创作实现。

+



教学进度计划（初步）

本课程计划主讲32学时，平均每周2学时进行一章的教学。总体目标是让学生通过课程的学习，熟练掌握R平台，重点关注可视化方法理解数据和解决实际问题的能力培养。

第一部分 入门

- 第1章　R语言介绍、创建数据集、图形初阶
- 第2章　数据管理

第二部分 基本方法

- 第3章　图形处理、使用ggplot2进行高级绘图
- 第4章　基本统计分析

第三部分 中级方法

- 第5章　回归、方差分析
- 第6章　功效分析、重抽样与自助法

第四部分 高级方法

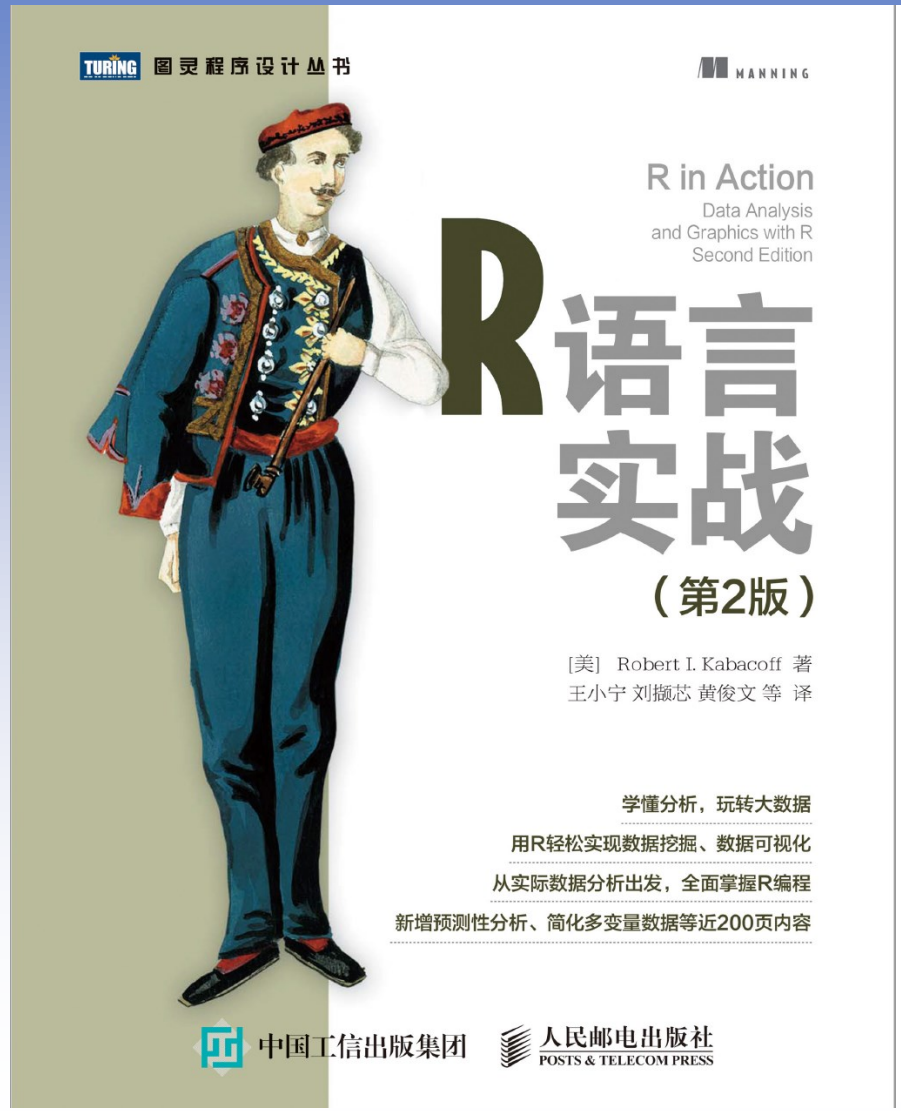
- 第7章　广义线性模型
- 第8章　主成分分析和因子分析
- 第9章　时间序列和聚类分析
- 第10章　分类
- 第11章　处理缺失数据的高级方法

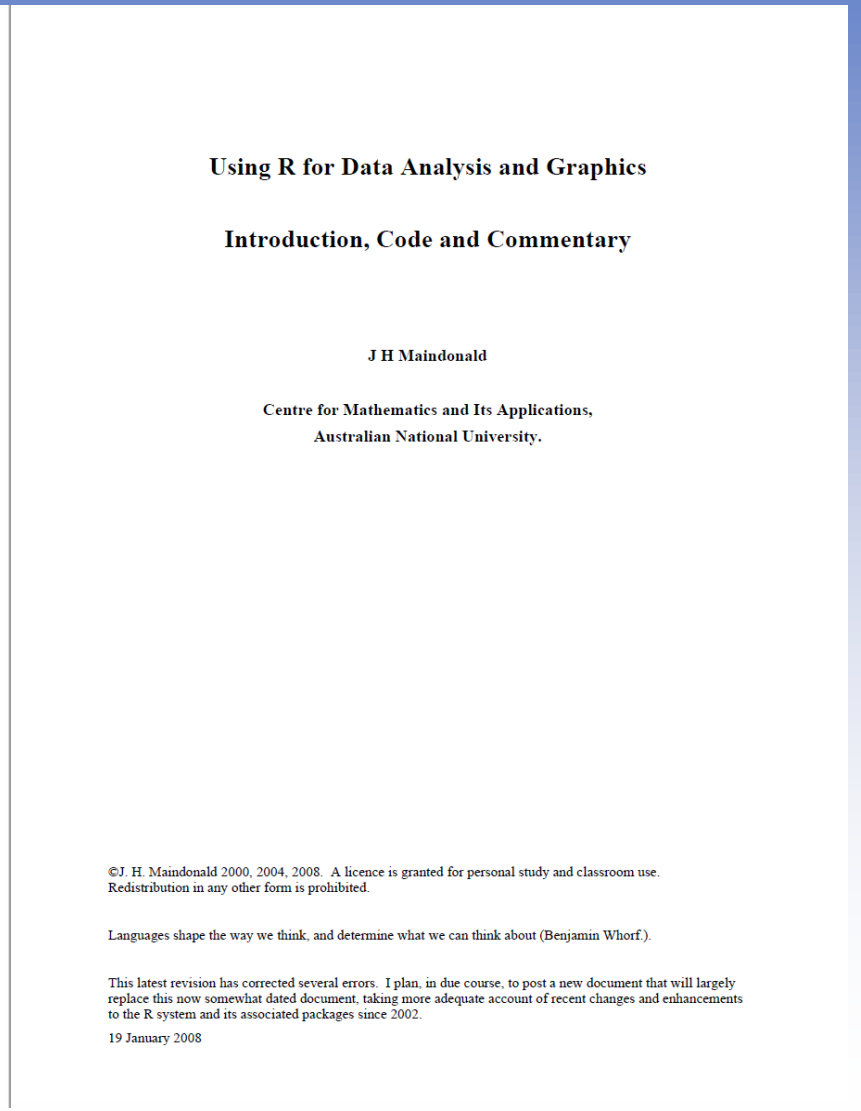
第五部分 技能拓展

- 第12章　高级编程
- 第13章　创建R包
- 第14章　创建动态报告
- 第15章　大气科学高级应用设计（一）
- 第16章　大气科学高级应用设计（二）

教材和参考书：

- 本课程鼓励同学们对比阅读R in Action的中、英文教材；
- 另外有多本英文参考资料可免费下载





Using R for Data Analysis and Graphics

Introduction, Code and Commentary

J H Maindonald

Centre for Mathematics and Its Applications,
Australian National University.

Languages shape the way we think, and determine what we can think about (Benjamin Whorf.).

This latest revision has corrected several errors. I plan, in due course, to post a new document that will largely replace this now somewhat dated document, taking more adequate account of recent changes and enhancements to the R system and its associated packages since 2002.

19 January 2008

成绩评定办法

- 课程的考核方式为平时作业+期末作业：

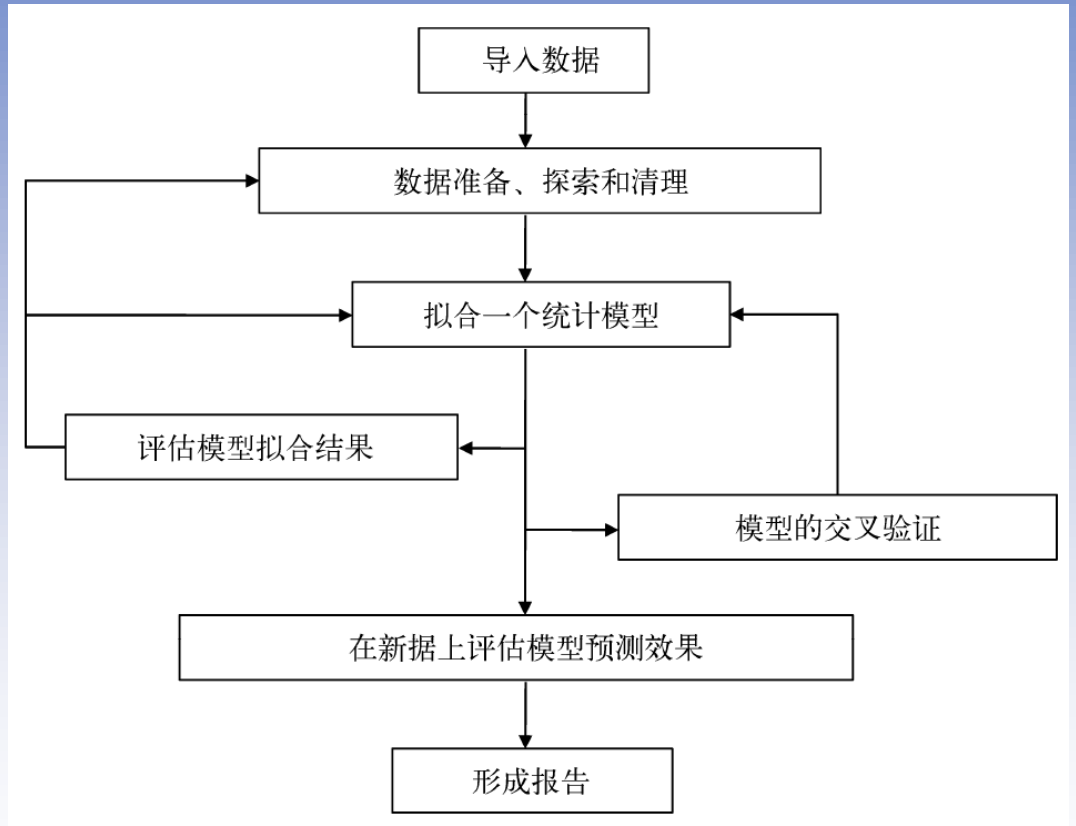
  平时作业占80%，主要考察学生是否掌握当堂所讲的方法。学生使用R语言对教师准备的数据（也可由学生自己准备）进行分析，并描述与解释结果，提交分析报告。

  期末作业占20%，由学生根据自己的研究方向选定一组数据，至少使用三种课堂讲授的方法对其进行分析，并撰写论文报告，规范与学术论文相同。

第1章　R语言介绍、创建数据集、图形初阶

1.1 R语言介绍

- 数据产生方式的发展+计算技术的发展
  =》海量数据形成
- 迫切要求数据分析技术的发展
- 典型的数据分析步骤如右图所示
  - 从广泛的数据源（数据库管理系统、文本文件、统计软件以及电子表格）获取数据
  - 将数据片段融合到一起、对数据做清理和标注
  - 用最新的方法进行分析、以有意义有吸引力的图形化方式展示结果
  - 将结果整合成令人感兴趣的报告并向利益相关者和公众发布。
- R语言：一个适合完成以上目标的理想而又功能全面的软件。



为什么要使用R?

- 多数商业统计软件价格不菲，而R是免费的！
- R是一个全面的统计研究平台，提供了各式各样的数据分析技术。几乎任何类型的数据分析工作皆可在R中完成。
- R拥有顶尖水准的制图功能。如果希望复杂数据可视化，那么R拥有最全面且最强大的一系列可用功能。
- R是一个可进行交互式数据分析和探索的强大平台。
- R可以轻松地从各种类型的数据源导入数据，包括文本文件、数据库管理系统、统计软件，乃至专门的数据仓库。
- R是一个无与伦比的平台，在其上可使用一种简单而直接的方式编写新的统计方法。它易于扩展，并为快速编程实现新方法提供了一套十分自然的语言。
- R囊括了在其他软件中尚不可用的、先进的统计计算例程。事实上，新方法的更新速度是以周来计算的。如果你是一位SAS用户，想象一下每隔几天就获得一个新SAS过程的情景。
- 如果你不想学习一门新的语言，有各式各样的GUI（Graphical User Interface，图形用户界面）工具通过菜单和对话框提供了与R语言同等的功能。
- R可运行于多种平台之上，包括Windows、UNIX和Mac OS X。这基本上意味着它可以运行于你所能拥有的任何计算机上。

R的获取和安装

- Comprehensive R Archive Network （CRAN）：

http://cran.r-project.org

- Linux、Mac OS X和Windows都有相应编译好的二进制版本。
- 安装包（package）作为可选模块同样可从CRAN下载，来增强R的功能。
- 建议同学们在Windows操作系统的个人电脑上安装R的运行平台：

Rgui 

RStudio 

R入门 （Getting started ）

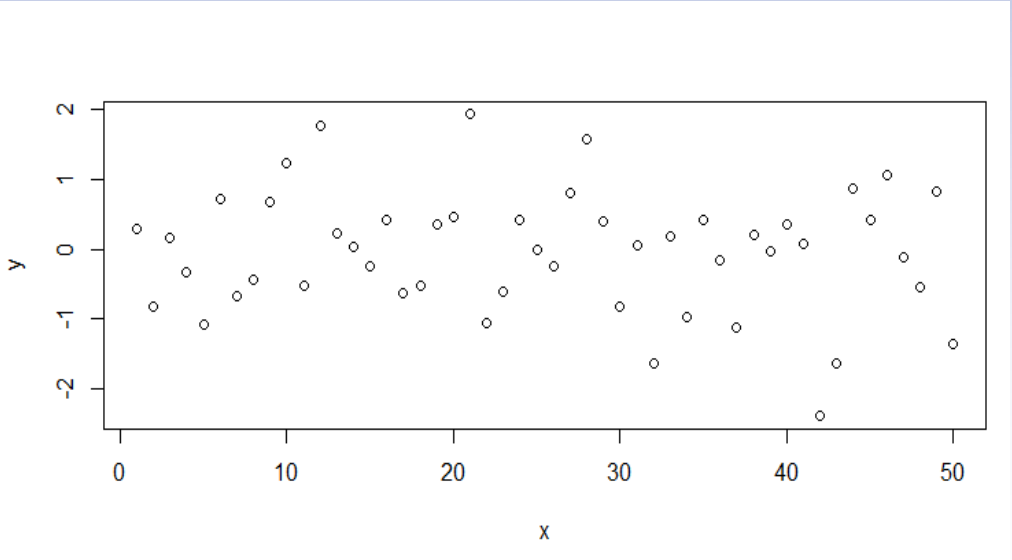
- 赋值语句：

```
>  y <- rnorm(50)
```

创建了一个名为y的向量对象，它包含50个来自标准正态分布的随机偏差。

```
>  x <- c(1:50)
```
创建了一个名为x的向量对象，它包含1~50共50个整数。

```
>  plot(x, y)
```
输出图形：



可以通过键入x或y查看具体数值

```
>  x
>  y
```

获取帮助 Getting help

表1-2　R中的帮助函数

函　　数	功　　能
help.start()	打开帮助文档首页
help("*foo*")或?*foo*	查看函数*foo*的帮助（引号可以省略）
help.search("*foo*")或??*foo*	以*foo*为关键词搜索本地帮助文档
example("*foo*")	函数*foo*的使用示例（引号可以省略）
RSiteSearch("*foo*")	以*foo*为关键词搜索在线文档和邮件列表存档
apropos("*foo*", mode="function")	列出名称中含有*foo*的所有可用函数
data()	列出当前已加载包中所含的所有可用示例数据集
vignette()	列出当前已安装包中所有可用的vignette文档
vignette("*foo*")	为主题*foo*显示指定的vignette文档

The workspace

- 工作空间，包括工作目录、工作空间中的对象、对象的存储和载入等
- 熟练掌握表1-3的函数

- 体味代码(code1.2.R):

```
> setwd("C:/myprojects/project1")
> options()
> options(digits=3)
> x <- runif(20)
> summary(x)
> hist(x)
> savehistory()
> save.image()
```

表1-3　用于管理R工作空间的函数

函　数	功　能
getwd()	显示当前的工作目录
setwd("*mydirectory*")	修改当前的工作目录为*mydirectory*
ls()	列出当前工作空间中的对象
rm(*objectlist*)	移除（删除）一个或多个对象
help(options)	显示可用选项的说明
options()	显示或设置当前选项
history(#)	显示最近使用过的#个命令（默认值为25）
savehistory("*myfile*")	保存命令历史到文件*myfile*中（默认值为.Rhistory）
loadhistory("*myfile*")	载入一个命令历史文件（默认值为.Rhistory）
save.image("*myfile*")	保存工作空间到文件*myfile*中（默认值为.RData）
save(*objectlist*, file="*myfile*")	保存指定对象到一个文件中
load("*myfile*")	读取一个工作空间到当前会话中（默认值为.RData）
q()	退出R。将会询问你是否保存工作空间

首先，当前工作目录被设置为C:/myprojects/project1，当前的选项设置情况将显示出来，而数字将被格式化为具有小数点后三位有效数字的格式。然后，创建了一个包含20个均匀分布随机变量的向量，生成了此数据的摘要统计量和直方图。最后，命令的历史记录保存到文件.Rhistory中，工作空间（包含向量x）保存到文件.RData中，会话结束。

Input and output

启动R后将默认开始一个交互式的会话，从键盘接受输入并从屏幕进行输出。不过你也可以将结果输出到多类目标中:

1. 输入

- 函数source("*filename*")可在当前会话中执行一个脚本。如果文件名中不包含路径，R将假设此脚本在当前工作目录中。举例来说，source("myscript.R")将执行包含在文件myscript.R中的R语句集合。依照惯例，脚本文件以.R作为扩展名，不过这并不是必需的。

2. 文本输出

- 函数sink("*filename*")将输出重定向到文件*filename*中。默认情况下，如果文件已经存在，则它的内容将被覆盖。使用参数append=TRUE可以将文本追加到文件后，而不是覆盖它。参数split=TRUE可将输出同时发送到屏幕和输出文件中。不加参数调用命令sink()将仅向屏幕返回输出结果。

3. 图形输出

- 虽然sink()可以重定向文本输出，但它对图形输出没有影响。要重定向图形输出，使用表1-4中列出的函数即可。最后使用dev.off()将输出返回到终端。

Input and output

表1-4　用于保存图形输出的函数

函　数	输　出
pdf("filename.pdf")	PDF文件
win.metafile("filename.wmf")	Windows图元文件
png("filename.png")	PBG文件
jpeg("filename.jpg")	JPEG文件
bmp("filename.bmp")	BMP文件
postscript("filename.ps")	PostScript文件

让我们通过一个示例来了解整个流程。假设我们有包含R代码的三个脚本文件script1.R、script2.R和script3.R。执行语句：

>source("script1.R")

- 将会在当前会话中执行script1.R中的R代码，结果将出现在屏幕上。

如果执行语句：

>sink("myoutput", append=TRUE, split=TRUE)

>pdf("mygraphs.pdf")

>source("script2.R")

文件script2.R中的R代码将执行，结果也将显示在屏幕上。除此之外，文本输出将被追加到文件myoutput中，图形输出将保存到文件mygraphs.pdf中。

Input and output

最后我们执行下面代码，执行script3.R，结果将显示在屏幕上。这一次，没有文本或图形输出保存到文件中：

```
>sink()
>dev.off()
>source("script3.R")
```

以上的例子流程如右图。

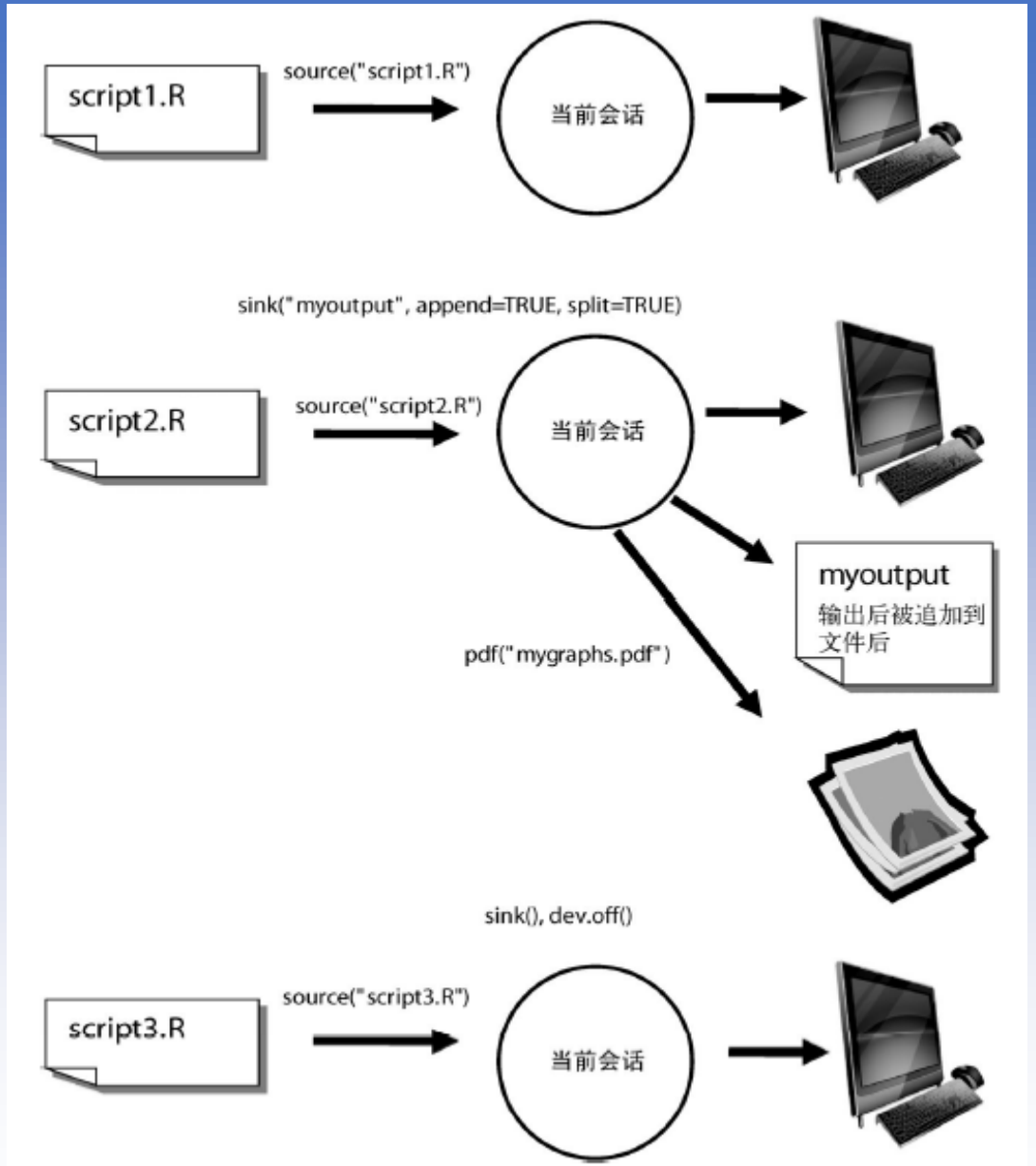


图1-6　使用函数source()进行输入并使用函数sink()进行输出

包 Packages

R最激动人心的一部分功能是通过可选模块的下载和安装来实现的。目前有几千个称为包（package）的用户贡献模块可从http://cran.r-project.org/web/packages下载。这些包提供了横跨各种领域、数量惊人的新功能，跨越多个学科。这也是我们要学习R的一个动机：借用和丰富前人的成果。

什么是包

- 包是R函数、数据、预编译代码以一种定义完善的格式组成的集合。计算机上存储包的目录称为库（library）。函数.libPaths()能够显示库所在的位置， 函数library()则可以显示库中有哪些包。
- R自带了一系列默认包（包括base、datasets、utils、grDevices、graphics、stats以及methods），它们提供了种类繁多的默认函数和数据集。其他包可通过下载来进行安装。安装好以后，它们必须被载入到会话中才能使用。命令search()可以告诉你哪些包已加载并可使用。

包的安装

- 有许多R函数可以用来管理包。第一次安装一个包，使用命令install.packages()即可。
- 不加参数执行install.packages()将显示一个CRAN镜像站点的列表，选择其中一个镜像站点之后，将看到所有可用包的列表，选择其中的一个包即可进行下载和安装。
- 如果知道自己想安装的包的名称，可以直接将包名作为参数提供给这个函数。例如，包gclus中提供了创建增强型散点图的函数。可以使用命install.packages("gclus")来下载和安装它。
- 一个包仅需安装一次。但和其他软件类似，包经常被其作者更新。使用命令update.packages()可以更新已经安装的包。、
- 要查看已安装包的描述，可以使用installed.packages()命令，这将列出安装的包，以及它们的版本号、依赖关系等信息。

包 Packages

包的载入

- 包的安装是指从某个CRAN镜像站点下载它并将其放入库中的过程。要在R会话中使用它，还需要使用library()命令载入这个包。例如，要使用gclus包，执行命令library(gclus)即可。当然，在载入一个包之前必须已经安装了这个包。
- 在一个会话中，包只需载入一次。如果需要，你可以自定义启动环境以自动载入会频繁使用的那些包。启动环境的自定义在教材附录B中有详细描述。

包的使用

- 载入一个包之后，就可以使用一系列新的函数和数据集了。包中往往提供了演示性的小型数据集和示例代码，能够让我们尝试这些新功能。帮助系统包含了每个函数的一个描述（同时带有示例），每个数据集的信息也被包括其中。
- 命令help(package="*package_name*")可以输出某个包的简短描述以及包中的函数名称和数据集名称的列表。使用函数help()可以查看其中任意函数或数据集的更多细节。这些信息也能以PDF帮助手册的形式从CRAN下载。

R语言常见错误 Common mistakes

有一些错误是R的初学者和经验丰富的R程序员都可能常犯的。如果程序出错了，请检查以下几方面。

- 使用了错误的大小写。如help()、Help()和HELP()是三个不同的函数（只有第一个是正确的）。
- 忘记使用必要的引号。install.packages("gclus")能够正常执行，然而Install.packages(gclus)将会报错。
- 在函数调用时忘记使用括号。例如，要使用help()而非help。即使函数无需参数，仍需加上()。
- 在Windows上，路径名中使用了\。R将反斜杠视为一个转义字符。setwd("c:\mydata")会报错。正确的写法是setwd("c:/mydata")或setwd("c:\\mydata")。
- 使用了一个尚未载入包中的函数。函数order.clusters()包含在包gclus中。如果还没有载入这个包就使用它，将会报错。

R的报错信息可能是含义模糊的，但如果谨慎遵守了以上要点，就应该可以避免许多错误。

Batch processing 批处理

批处理：重复的、标准化的、无人值守的方式执行某个R程序

- 如何以批处理模式运行R与使用的操作系统有关。在Linux或Mac OS X系统下，可以在终端窗口中使用如下命令：

  R CMD BATCH *options infile outfile*

- 其中*infile*是包含了要执行的R代码所在文件的文件名，*outfile*是接收输出文件的文件名，*options*部分则列出了控制执行细节的选项。依照惯例，*infile*的扩展名是.R，*outfile*的扩展名为.Rout。
- 对于Windows，则需使用：

  "C:\Program Files\R\R-2.13.0\bin\R.exe" CMD BATCH

  --vanilla --slave "c:\my projects\myscript.R"

1.2 创建数据集 Creating a dataset

数据结构：

- 向量
- 矩阵
- 数组
- 数据框
- 因子
- 列表

数据输入:

- 键盘输入
- 文本文件导入
- Excel表导入
- 网页抓取
- 导入SPSS
- 导入SAS
- 导入Stata
- 导入NetCDF
- 导入HDF5

处理数据对象的实用函数

数据结构

向量：Vectors，存储数值型、字符型或逻辑性的一维数组。

```
> a <- c(1, 2, 5, 3, 6, -2, 4)
> b <- c("one", "two", "three")
> c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
```

注意：同一向量中无法混杂不同模式的数据。

- 向量的访问：

```
> a <- c(1, 2, 5, 3, 6, -2, 4)
> a[3]
 5
> a[c(1, 3, 5)]
 1 5 6
> a[2:6]
 2 5 3 6 -2
```

数据结构

矩阵：Matrices，二维数组，只是每个元素都拥有相同的类型（数值型、字符型或逻辑型）。可通过函数matrix创建矩阵：

mymatrix <- matrix(*vector*, nrow=*number_of_rows*, ncol=*number_of_columns*, byrow=*logical_value*, dimnames = list(*char_vector_rownames*, *char_vector_colnames*))

```
> y <- matrix(1:20, nrow=5, ncol=4)
> y
     [,1] [,2] [,3] [,4]
[1,]  1   6   11  16
[2,]  2   7   12  17
[3,]  3   8   13  18
[4,]  4   9   14  19
[5,]  5 10   15  20
```

```
>   cells <- c(1,26,24,68)
>   rnames <- c("R1", "R2")
>   cnames <- c("C1", "C2")
>   mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,
    dimnames=list(rnames, cnames))
> mymatrix
     C1 C2
R1   1  26
R2 24  68
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=FALSE,
    dimnames=list(rnames, cnames))
> mymatrix
     C1 C2
R1   1 24
R2 26 68
```

数据结构

数组：Arrays，与矩阵类似，但是维度可以大于2。数组可通过array函数创建，形式如下：

myarray <- array(*vector, dimensions, dimnames*)

```
> dim1 <- c("A1", "A2")
> dim2 <- c("B1", "B2", "B3")
> dim3 <- c("C1", "C2", "C3", "C4")
> z <- array(1:24, c(2, 3, 4), dimnames=list(dim1, dim2, dim3))
> z
, , C1
   B1 B2 B3
A1  1  3  5
A2  2  4  6
, , C2
   B1 B2 B3
A1  7  9 11
A2  8 10 12
, , C3
   B1 B2 B3
A1 13 15 17
A2 14 16 18
, , C4
   B1 B2 B3
A1 19 21 23
A2 20 22 24
```

数据结构

- **数据框**：Data frames, 是R中**最常处理的数据结构**，不同的列可以包含不同的数据类型（数值型、字符型等），通过函数data.frame()创建：
- mydata <- data.frame(col1,col2,col3, ?)

```
> patientID <- c(1, 2, 3, 4)
> age <- c(25, 34, 28, 52)
> diabetes <- c("Type1", "Type2", "Type1", "Type1")
> status <- c("Poor", "Improved", "Excellent", "Poor")
> patientdata <- data.frame(patientID, age, diabetes, status)
> patientdata
  patientID age diabetes status
1         1  25    Type1   Poor
2         2  34    Type2 Improved
3         3  28    Type1 Excellent
4         4  52    Type1   Poor
```

```
>  patientdata[1:2]
  patientID age
1         1  25
2         2  34
3         3  28
4         4  52
>  patientdata[c("diabetes","status")]
  diabetes status
1    Type1   Poor
2    Type2 Improved
3    Type1 Excellent
4    Type1   Poor
>  patientdata$age
[1] 25 34 28 52
```

数据结构

- 数据框变量访问： attach()、detach()和with()用于简化代码

```
summary(mtcars$mpg)
plot(mtcars$mpg, mtcars$disp)
plot(mtcars$mpg, mtcars$wt)
```

利用attach():

```
attach(mtcars)
    summary(mpg)
    plot(mpg, disp)
    plot(mpg, wt)
detach(mtcars)
```

缺陷是数据框变量与全局环境变量重复时出错。

```
with (mtcars, {
      Print ( summary(mpg) )
      plot(mpg, disp)
      plot(mpg, wt)
      stats <- summary(mpg)
      stats
      } )
      stats
Error: object 'stats' not found
```

缺陷是赋值仅在括号内生效，如果需要赋值给全局变量，需要在括号内使用特殊赋值符 "<<-" 替代标准赋值符"<-"：

```
stats <<- summary(mpg)
```

数据结构

因子：factors, 用于处理名义型变量（如糖尿病类型Type1、Type2）和有序型变量（如病人状态poor、improved、excellent）

```
diabetes <- c("Type1", "Type2", "Type1", "Type1")
diabetes <- factor(diabetes)
```

将此名义向量存储为(1, 2, 1, 1)，并在内部将其关联为1=Type1和2=Type2。

ordered=TRUE选项用于有序变量：

```
status <- c("Poor", "Improved", "Excellent", "Poor")
status <- factor(status, ordered=TRUE)
```

将向量编码为(3, 2, 1, 3)，并在内部将这些值关联为1=Excellent、2=Improved以及3=Poor。另外，针对此向量进行的任何分析都会将其作为有序型变量对待，并自动选择合适的统计方法。可以通过指定levels选项来覆盖默认排序

```
status <- factor(status, order=TRUE, levels=c("Poor", "Improved", "Excellent"))
```

将指定1=Poor、2=Improved、3=Excellent

数据结构

- 因子的使用

```
> patientID <- c(1, 2, 3, 4)
> age <- c(25, 34, 28, 52)
> diabetes <- c("Type1", "Type2", "Type1", "Type1")
> status <- c("Poor", "Improved", "Excellent", "Poor")
> diabetes <- factor(diabetes)
> status <- factor(status, order=TRUE)
> patientdata <- data.frame(patientID, age, diabetes, status)
> str(patientdata)
'data.frame':   4 obs. of  4 variables:
 $ patientID: num  1 2 3 4
 $ age      : num  25 34 28 52
 $ diabetes : Factor w/ 2 levels "Type1","Type2": 1 2 1 1
 $ status   : Ord.factor w/ 3 levels "Excellent"<"Improved"<..: 3 2 1 3
> summary(patientdata)
   patientID          age         diabetes       status
 Min.   :1.00   Min.   :25.00   Type1:3   Excellent:1
 1st Qu.:1.75   1st Qu.:27.25   Type2:1   Improved :1
 Median :2.50   Median :31.00             Poor     :2
 Mean   :2.50   Mean   :34.75
 3rd Qu.:3.25   3rd Qu.:38.50
 Max.   :4.00   Max.   :52.00
```

❶ 以向量形式输入数据

❷ 显示对象的结构

❸ 显示对象的统计概要

代码以向量的形式输入了数据。然后，将diabetes和status分别指定为一个普通因子和一个有序型因子。最后，将数据合并为一个数据框。函数str(*object*)可提供R中某个对象的信息。它清楚地显示diabetes是一个因子，而status是一个有序型因子，以及此数据框在内部是如何进行编码的。注意，函数summary()会区别对待各个变量。它显示了连续型变量age的最小值、最大值、均值和各四分位数，并显示了类别型变量diabetes和有序型变量status（各水平）的频数值。

数据结构

- 列表：lists，是R数据类型中最为复杂的一种，是一些对象（或成分，components）的有序集合。它允许你整合若干对象到单个对象名下。某个列表中可能是若干向量、矩阵、数据框，甚至其他列表的组合，其中的对象可以是目前为止讲到的任何结构：
- mylist <- list(*object1, object2, …*)　　mylist <- list(*name1=object1, name2=object2, …*)

```
> g <- "My First List"
> h <- c(25, 26, 18, 39)
> j <- matrix(1:10, nrow=5)
> k <- c("one", "two", "three")
> mylist <- list(title=g, ages=h, j, k)
> mylist
$title
[1] "My First List"
$ages
[1] 25 26 18 39
```

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


[[4]]
[1] "one" "two" "three"
```

```
> mylist[[2]]
[1] 25 26 18 39


> mylist[["ages"]]
[1] 25 26 18 39
```

列表的必要：用于组织和调用不同结构的信息；许多R函数的运行结果以列表的形式返回。

数据输入

- 键盘:

> mydata <- data.frame(age=numeric(0), gender=character(0), weight=numeric(0))

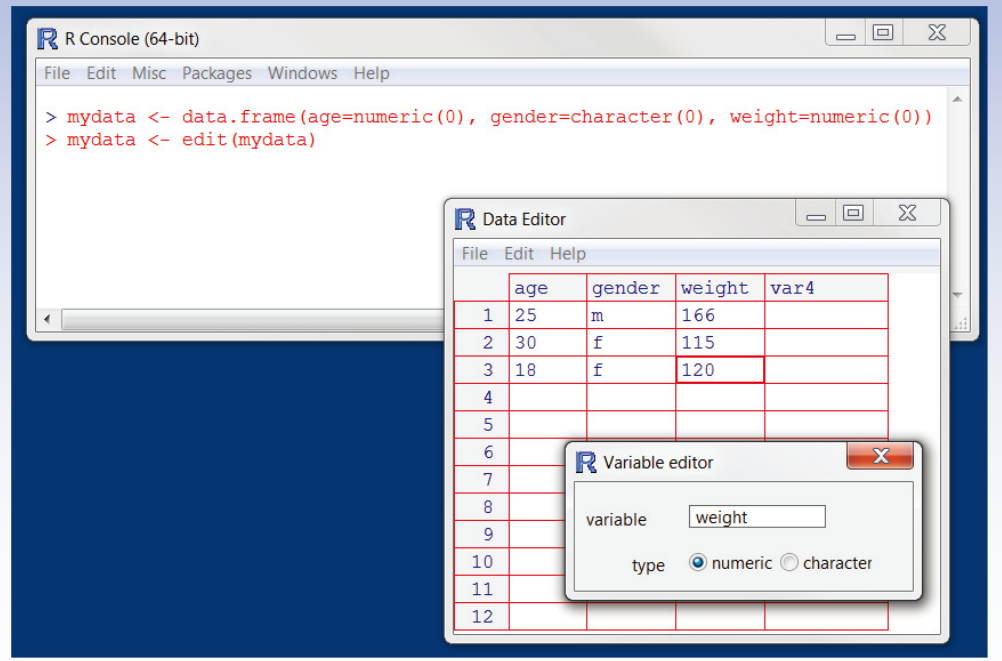
> mydata <- edit(mydata)

Warning message: In edit.data.frame(mydata) : 在'gender'里加上了因子水准

> mydata

age gender weight

- 1 1 m 1
- 2 2 f 2
- 3 3 m 3
- 4 4 f 4



数据输入

- 键盘在代码嵌入:

```
mydatatxt <-  “
      age gender weight
       25  m       166
       30   f       115
       18   f       120
  “

Mydata <-  read.table(header=TRUE, text=mydatatxt)
```

数据输入：从带分隔符的文本文件导入

mydataframe <- read.table(*file*, options)

mydataframe <- read.table(*file*, header=*logical_value*, sep="*delimiter*", row.names="*name*")

- *file*是一个带分隔符的ASCII文本文件，header是一个表明首行是否包含了变量名的逻辑值（TRUE或FALSE），sep用来指定分隔数据的分隔符，row.names是一个可选参数，用以指定一个或多个表示行标识符的变量。

- 举例：

```
grades <- read.table("studentgrades.csv", header=TRUE, sep=",",
+                    row.names="STUDENTID")
```

数据输入：其他文件和方式

- Excel表导入
- 导入XML
- 从网页抓取数据
- 导入SPSS
- 导入SAS
- 导入Stata
- 导入NetCDF
- 导入HDF5

R语言具有很多灵活的文件输入方式，一般都是需要下载并安装相应的包。我们暂时不需要完全掌握，将来具体任务需要时可以再了解。

1.3 图形初阶

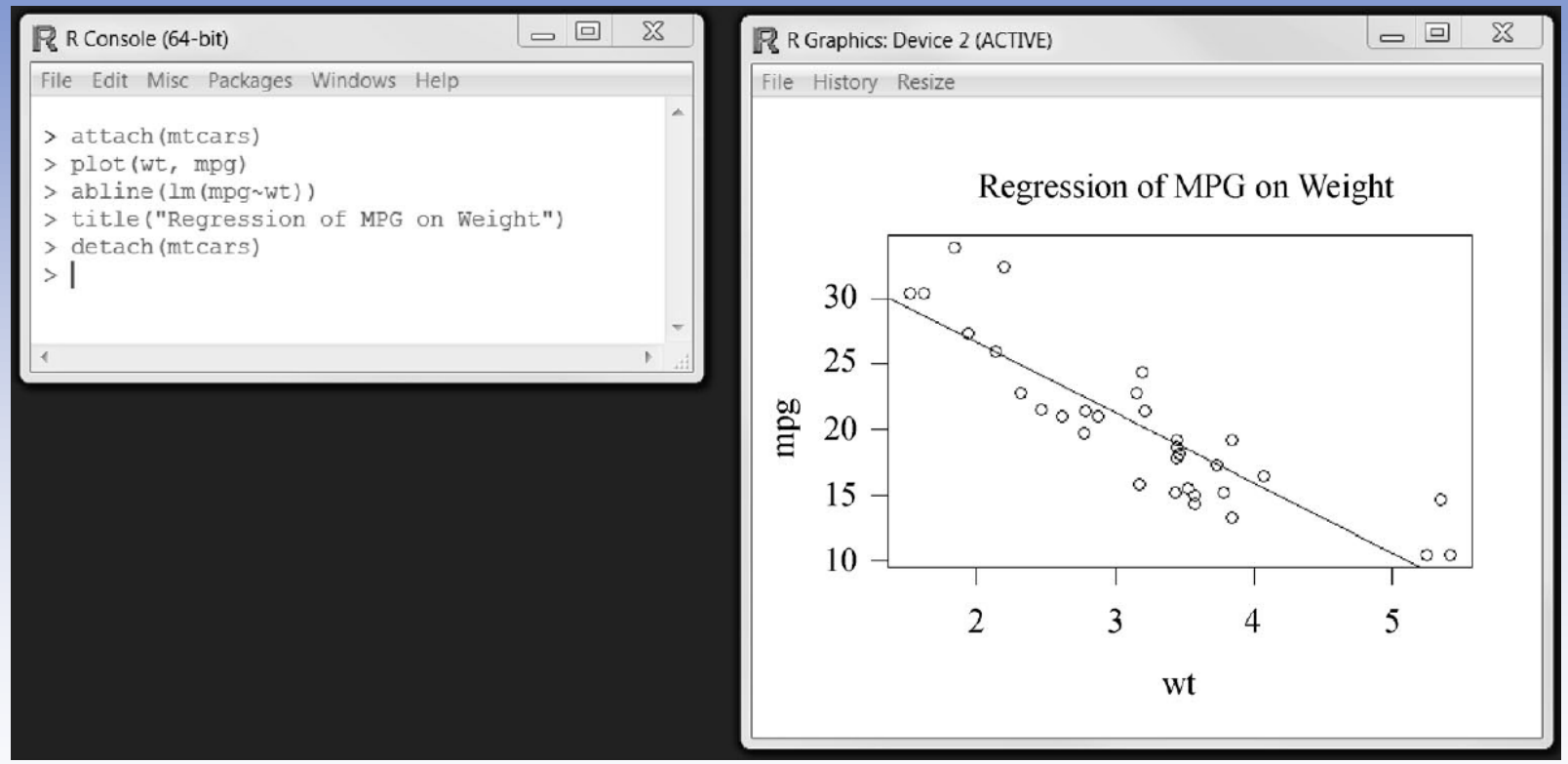
- 图形的创建和保存
- 自定义符号、线条、颜色和坐标轴
- 标注文本和标题
- 控制图形维度
- 组合多个图形

为什么需要图形？ Why graphics?

- Human beings are remarkably adept at discerning relationships from visual representations. A well-crafted graph can help you make meaningful comparisons among thousands of pieces of information, extracting patterns not easily found through other methods.
- 人类非常善于从视觉呈现中洞察关系。一幅精心绘制的图形能够帮助你在数以千计的零散信息中做出有意义的比较，提炼出使用其他方法时不那么容易发现的模式。
- 本部分的内容：
  - 如何创建和保存图形
  - 如何修改那些存在于所有图形中的特征，包括图形的标题、坐标轴、标签、颜色、线条、符号和文本标注。重点关注那些可以应用于所有图形的通用方法。
  - 研究组合多幅图形为单幅图形的各种方法。

简单图形

- 绑定数据框mtcars
- 打开了一个图形窗口并生成了一幅散点图，横轴表示车身重量，纵轴为每加仑汽油行驶的英里数
- 向图形添加一条最优拟合曲线
- 添加标题
- 为数据框解除绑定



```
> pdf("mygraph.pdf")
>       ……
> dev.off()
```

- 除了pdf()，还可以使用函数win.metafile()、png()、jpeg()、bmp()、tiff()、xfig()和postscript()将图形保存为其他格式; 另外，可以通过图形用户界面保存图形。

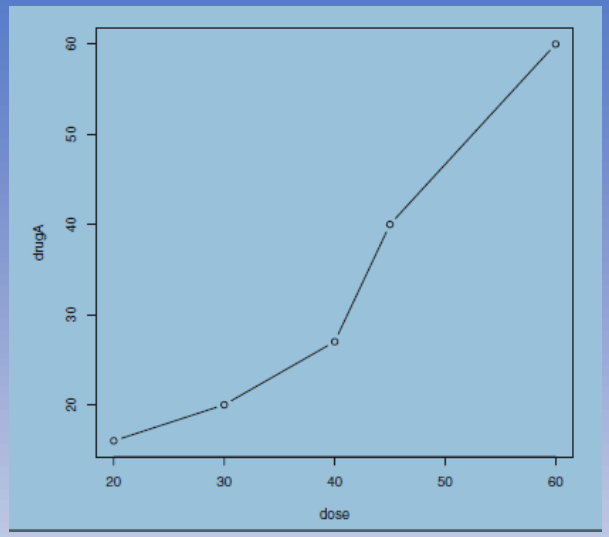
窗口控制

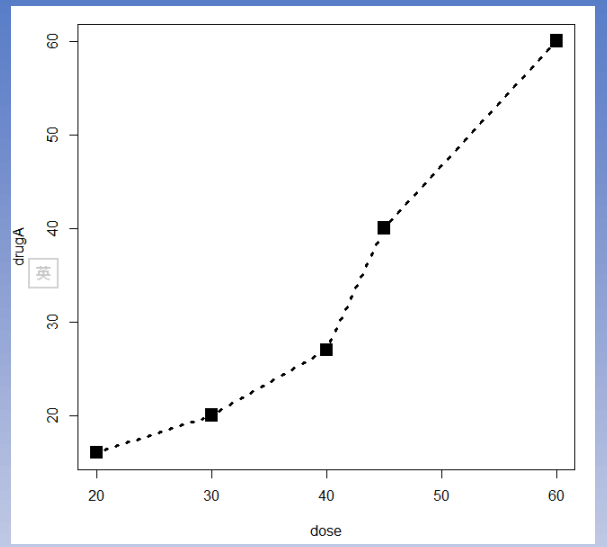
- dev.new()

- dev.new()、dev.next()、dev.prev()、dev.set()和dev.off()同时打开多个图形窗口，并选择将哪个输出发送到哪个窗口中。
- 这种方法全平台适用，关于这种方法的更多细节，请参考help(dev.cur)。

图形参数 plot

```
>dose <- c(20, 30, 40, 45, 60)
>drugA <- c(16, 20, 27, 40, 60)
>drugB <- c(15, 18, 25, 31, 40)
>plot(dose, drugA, type="b")
```



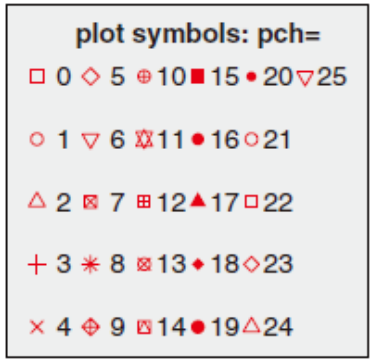


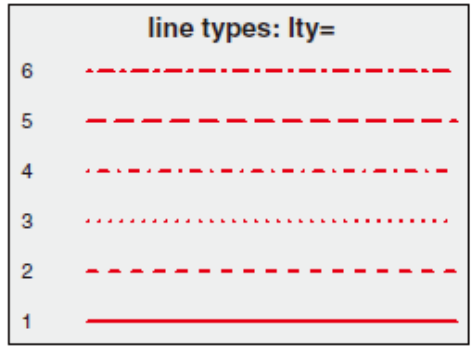
>par(lty=2, pch=17) 设置虚线及三角图标

or: plot(dose, drugA, type="b", lty=3, lwd=3, pch=15, cex=2)

表3-2 用于指定符号和线条类型的参数

参 数	描 述
pch	指定绘制点时使用的符号（见图3-4）
cex	指定符号的大小。cex是一个数值，表示绘图符号相对于默认大小的缩放倍数。默认大小为1，1.5表示放大为默认值的1.5倍，0.5表示缩小为默认值的50%，等等
lty	指定线条类型（参见图3-5）
lwd	指定线条宽度。lwd是以默认值的相对大小来表示的（默认值为1）。例如，lwd=2将生成一条两倍于默认宽度的线条





颜色

表3-3　用于指定颜色的参数

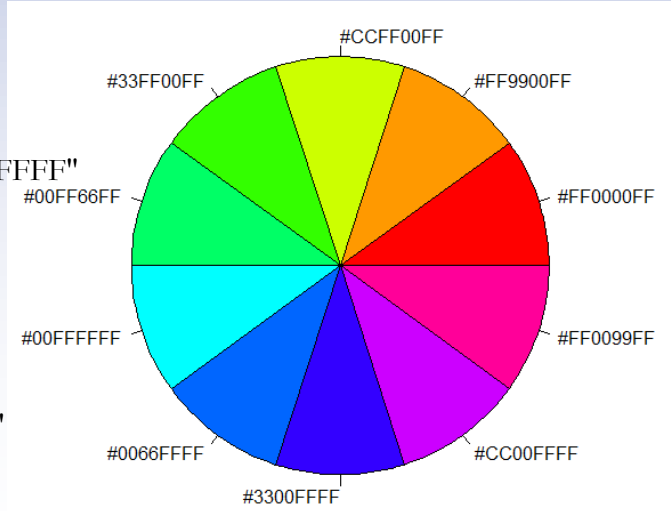
参　数	描　述
col	默认的绘图颜色。某些函数（如lines和pie）可以接受一个含有颜色值的向量并自动循环使用。例如，如果设定col=c("red", "blue")并需要绘制三条线，则第一条线将为红色，第二条线为蓝色，第三条线又将为红色
col.axis	坐标轴刻度文字的颜色
col.lab	坐标轴标签（名称）的颜色
col.main	标题颜色
col.sub	副标题颜色
fg	图形的前景色
bg	图形的背景色

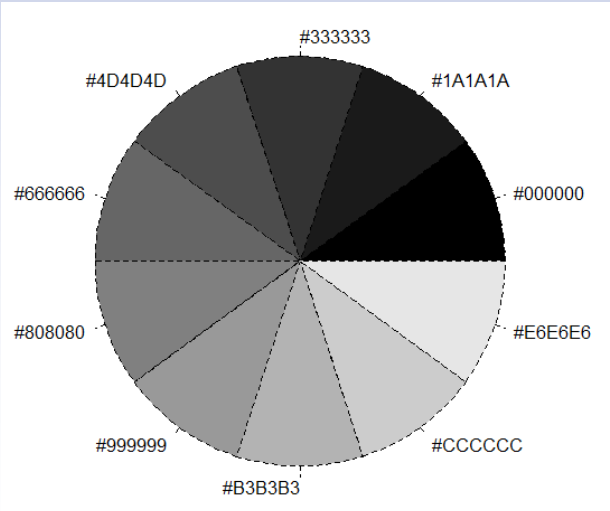
```
>n <- 10
>mycolors <- rainbow(n)
>pie(rep(1, n), labels=mycolors, col=mycolors)
>mygrays <- gray(0:n/n)
>pie(rep(1, n), labels=mygrays, col=mygrays)
```

```
> mycolors
[1] "#FF0000FF" "#FF9900FF" "#CCFF00FF" "#33FF00FF" "#00FF66FF" "#00FFFFFF"

[7] "#0066FFFF" "#3300FFFF" "#CC00FFFF" "#FF0099FF"
```

```
>mygrays
[1] "#000000" "#1A1A1A" "#333333" "#4D4D4D" "#666666" "#808080" "#999999"
[8] "#B3B3B3" "#CCCCCC" "#E6E6E6" "#FFFFFF"
```





颜色

- 可以通过颜色下标、颜色名称、十六进制的颜色值、RGB值或HSV值来指定颜色。例如，col=1、col="white"、col="#FFFFFF"、col=rgb(1,1,1)和col=hsv(0,0,1)都表示白色。
- 函数rgb()可基于红—绿—蓝三色值生成颜色，而hsv()则基于色相—饱和度—亮度值来生成颜色。
- 创建连续型颜色向量的函数，包括rainbow()、heat.colors()、terrain.colors()、topo.colors()以及cm.colors()。例如，rainbow(10)可以生成10种连续的“彩虹型”颜色。多阶灰度色可使用gray()函数生成，如上例。

文本属性

- 表3-4阐释了用于控制文本大小的参数。字
- 体族和字样可以通过字体选项进行控制（见表3-5）。
- 如：

par(font.lab=3, cex.lab=1.5, font.main=4, cex.main=2)之后创建的所有图形都将拥有斜体、1.5倍于默认文本大小的坐标轴标签，以及粗斜体、2倍于默认文本大小的标题。

表3-4　用于指定文本大小的参数

参　数	描　述
cex	表示相对于默认大小缩放倍数的数值。默认大小为1，1.5表示放大为默认值的1.5倍，0.5表示缩小为默认值的50%，等等
cex.axis	坐标轴刻度文字的缩放倍数。类似于cex
cex.lab	坐标轴标签（名称）的缩放倍数。类似于cex
cex.main	标题的缩放倍数。类似于cex
cex.sub	副标题的缩放倍数。类似于cex

表3-5　用于指定字体族、字号和字样的参数

参　数	描　述
font	整数。用于指定绘图使用的字体样式。1=常规，2=粗体，3=斜体，4=粗斜体，5=符号字体（以Adobe符号编码表示）
font.axis	坐标轴刻度文字的字体样式
font.lab	坐标轴标签（名称）的字体样式
font.main	标题的字体样式
font.sub	副标题的字体样式
ps	字体磅值（1磅约为1/72英寸）。文本的最终大小为ps*cex
family	绘制文本时使用的字体族。标准的取值为serif（衬线）、sans（无衬线）和mono（等宽）

图形尺寸与边界尺寸

- par(pin=c(4,3), mai=c(1,.5, 1, .2))
- 生成一幅4英寸宽、3英寸高、上下边界为1英寸、左边界为0.5英寸、右边界为0.2英寸的图形。

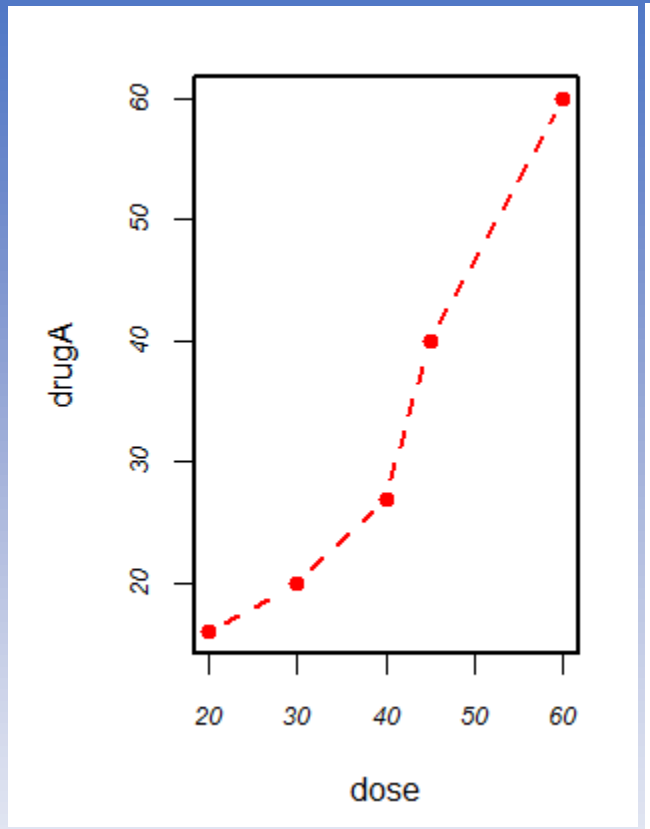
表3-6　用于控制图形尺寸和边界大小的参数

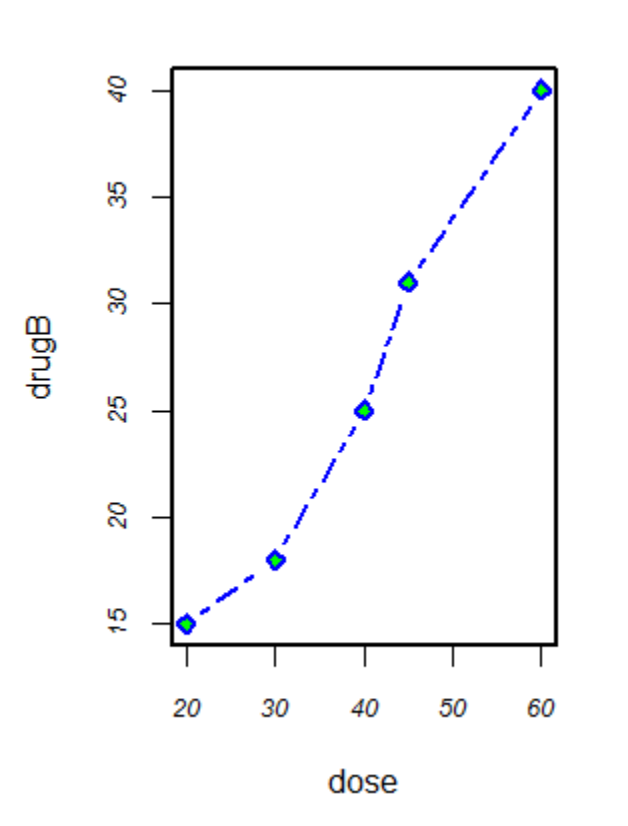
参　数	描　述
pin	以英寸表示的图形尺寸（宽和高）
mai	以数值向量表示的边界大小，顺序为“下、左、上、右”，单位为英寸
mar	以数值向量表示的边界大小，顺序为“下、左、上、右”，单位为英分*。默认值为c(5, 4, 4, 2) + 0.1

*一英分等于十二分之一英寸。——译者注

练习

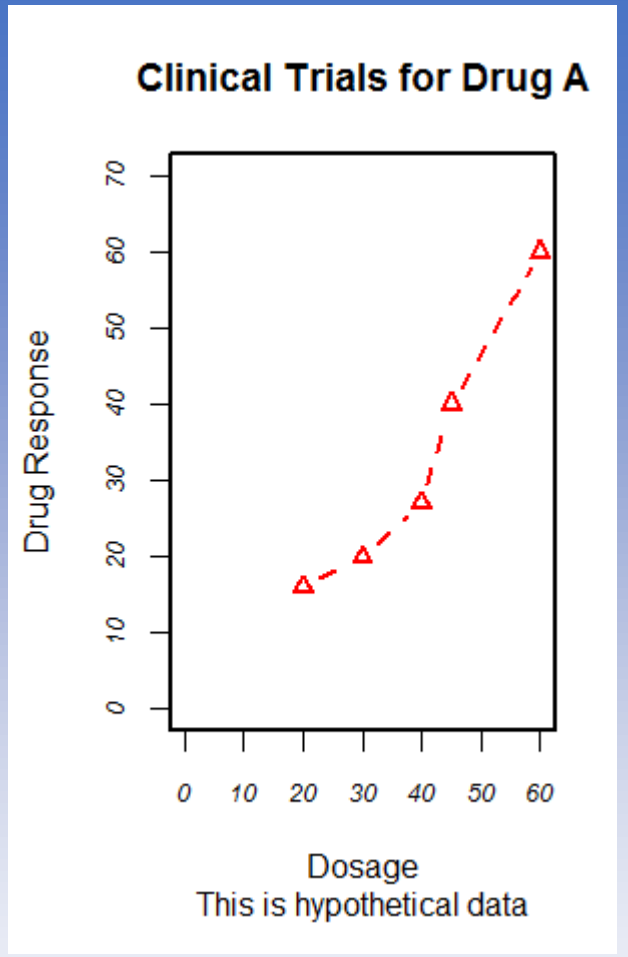
- dose <- c(20, 30, 40, 45, 60)
- drugA <- c(16, 20, 27, 40, 60)
- drugB <- c(15, 18, 25, 31, 40)
- opar <- par(no.readonly=TRUE)
- par(pin=c(2, 3))
- par(lwd=2, cex=1.5)
- par(cex.axis=.75, font.axis=3)
- plot(dose, drugA, type="b", pch=19, lty=2, col="red")
- plot(dose, drugB, type="b", pch=23, lty=6, col="blue", bg="green")
- par(opar)





坐标轴

```
>plot(dose, drugA, type="b",
  col="red", lty=2, pch=2, lwd=2,
  main="Clinical Trials for Drug A",
  sub="This is hypothetical data",
  xlab="Dosage", ylab="Drug Response",
  xlim=c(0, 60), ylim=c(0, 70))
```



- 在图形上添加了标题（main）、副标题（sub）、坐标轴标签（xlab、ylab）并指定了坐标轴范围（xlim、ylim）

标题

- 函数title()中可指定其他图形参数（如文本大小、字体、旋转角度和颜色）。例如，以下代码将生成红色的标题和蓝色的副标题，以及较默认大小小25%的绿色x轴、y轴标签：
- title(main="My Title", col.main="red",
  sub="My Sub-title", col.sub="blue",
  xlab="My X label", ylab="My Y label",
  col.lab="green", cex.lab=0.75)

坐标轴

axis(side, at=, labels=, pos=, lty=, col=, las=, tck=, ...)

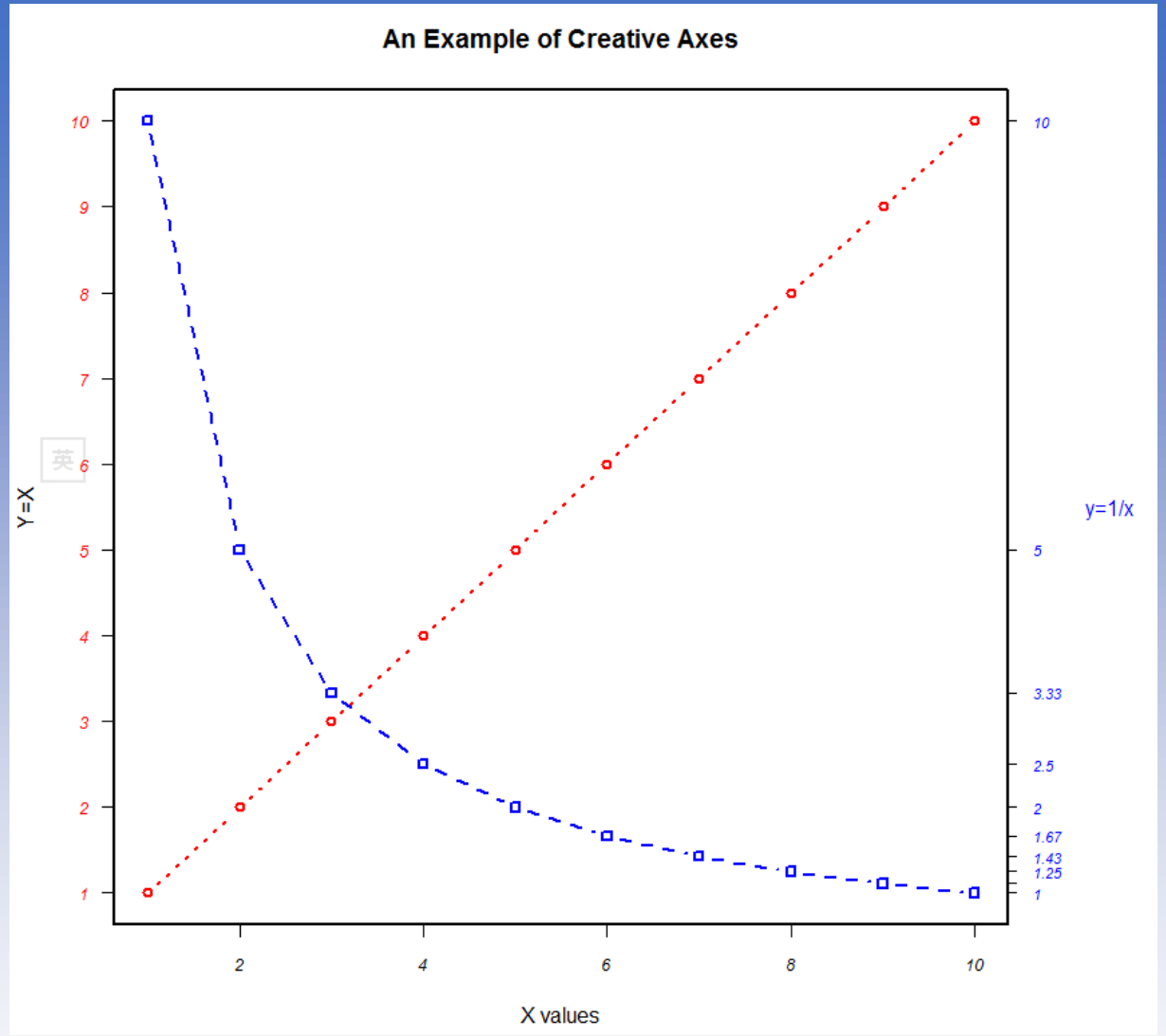
- 使用函数axis()来创建自定义坐标轴，应当禁用高级绘图函数自动生成的坐标轴。
- 参数axes=FALSE将禁用全部坐标轴（包括坐标轴框架线，除非你添加了参数frame.plot=TRUE）。参数xaxt="n"和yaxt="n"将分别禁用*X*轴或*Y*轴（会留下框架线，只是去除了刻度）。

表3-7　坐标轴选项

选　项	描　述
side	一个整数，表示在图形的哪边绘制坐标轴（1=下，2=左，3=上，4=右）
at	一个数值型向量，表示需要绘制刻度线的位置
labels	一个字符型向量，表示置于刻度线旁边的文字标签（如果为NULL，则将直接使用at中的值）
pos	坐标轴线绘制位置的坐标（即与另一条坐标轴相交位置的值）
lty	线条类型
col	线条和刻度线颜色
las	标签是否平行于（=0）或垂直于（=2）坐标轴
tck	刻度线的长度，以相对于绘图区域大小的分数表示（负值表示在图形外侧，正值表示在图形内侧，0表示禁用刻度，1表示绘制网格线）；默认值为–0.01
(…)	其他图形参数

练习

```
> x <- c(1:10)
> y <- x
> z <- 10/x
> opar <- par(no.readonly=TRUE)
> par(mar=c(5, 4, 4, 8) + 0.1)
> plot(x, y, type="b", pch=21, col="red",
+ yaxt="n", lty=3, ann=FALSE)
> lines(x, z, type="b", pch=22, col="blue", lty=2)
> axis(2, at=x, labels=x, col.axis="red", las=2)
> axis(4, at=z, labels=round(z, digits=2),
> col.axis="blue", las=2, cex.axis=0.7, tck=-.01)
> mtext("y=1/x", side=4, line=3, cex.lab=1, las=2, col="blue")
> title("An Example of Creative Axes", xlab="X values", ylab="Y=X")
> par(opar)
```



次要刻度线

- 需要使用Hmisc包中的minor.tick()函数：

> install.packages("Hmisc")

>library(Hmisc)

>minor.tick(nx=2, ny=3, tick.ratio=0.5)

- 在X轴的每两条主刻度线之间添加1条次要刻度线，并在Y轴的每两条主刻度线之间添加2条次要刻度线。

参考线

- 函数abline()可以用来为图形添加参考线。

  abline(h=*yvalues*, v=*xvalues*)

- 函数abline()中也可以指定其他图形参数（如线条类型、颜色和宽度）。举例来说：

  abline(h=c(1,5,7))

  在y为1、5、7的位置添加了水平实线，而代码：

  abline(v=seq(1, 10, 2), lty=2, col="blue")

  则在x为1、3、5、7、9的位置添加了垂直的蓝色虚线。

图例

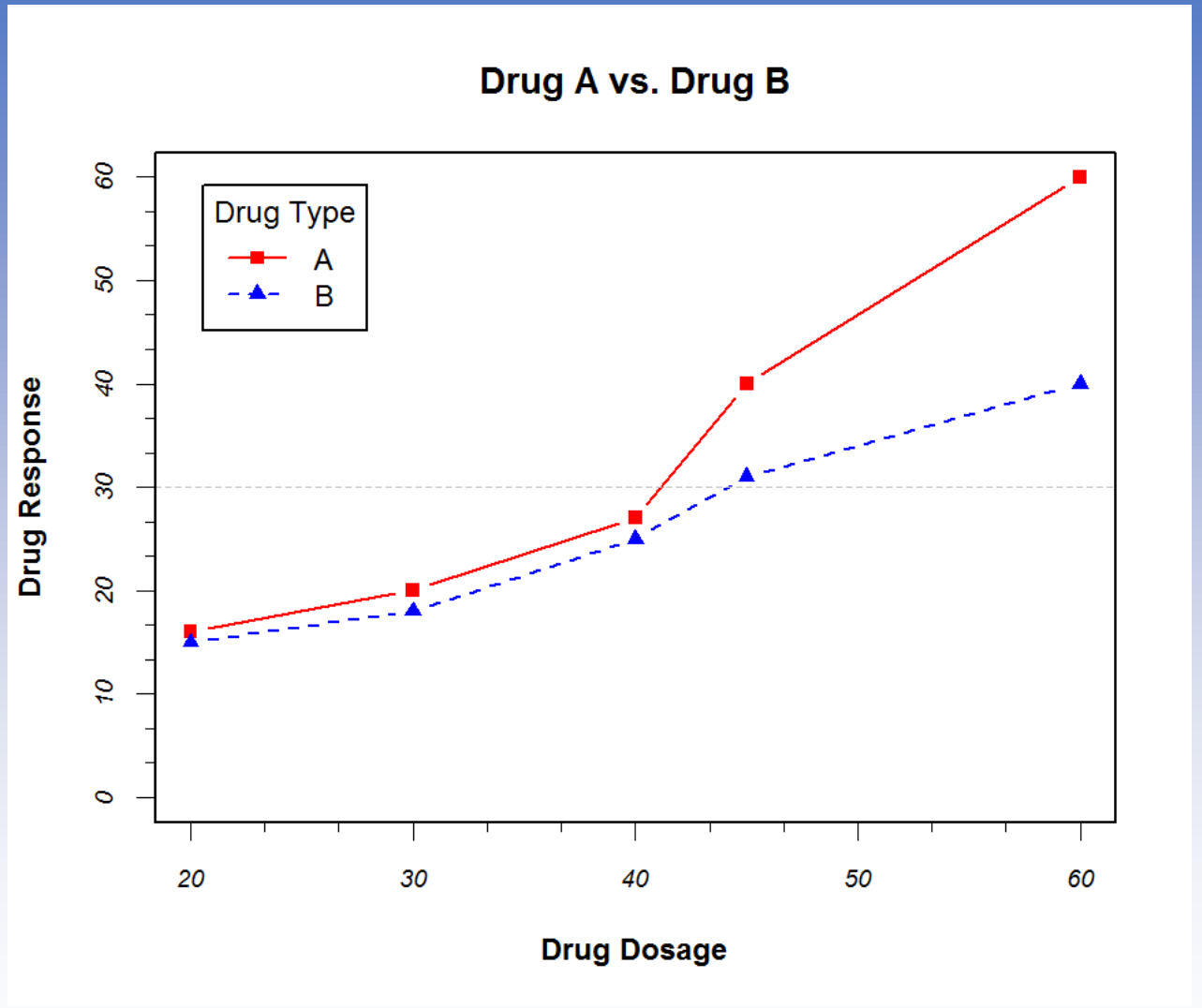
- legend(*location*, *title*, *legend*, ...)

表3-8　图例选项

选　项	描　述
location	有许多方式可以指定图例的位置。你可以直接给定图例左上角的x、y坐标，也可以执行locator(1)，然后通过鼠标单击给出图例的位置，还可以使用关键字bottom、bottomleft、left、topleft、top、topright、right、bottomright或center放置图例。如果你使用了以上某个关键字，那么可以同时使用参数inset=指定图例向图形内侧移动的大小（以绘图区域大小的分数表示）
title	图例标题的字符串（可选）
legend	图例标签组成的字符型向量
…	其他选项。如果图例标示的是颜色不同的线条，需要指定col=加上颜色值组成的向量。如果图例标示的是符号不同的点，则需指定pch=加上符号的代码组成的向量。如果图例标示的是不同的线条宽度或线条类型，请使用lwd=或lty=加上宽度值或类型值组成的向量。要为图例创建颜色填充的盒形（常见于条形图、箱线图或饼图），需要使用参数fill=加上颜色值组成的向量

练习

```
> dose <- c(20, 30, 40, 45, 60)
> drugA <- c(16, 20, 27, 40, 60)
> drugB <- c(15, 18, 25, 31, 40)
> opar <- par(no.readonly=TRUE)
> par(lwd=2, cex=1.5, font.lab=2)
> plot(dose, drugA, type="b",
+ pch=15, lty=1, col="red", ylim=c(0, 60),
+ main="Drug A vs. Drug B", xlab="Drug Dosage",
+ ylab="Drug Response")
> lines(dose, drugB, type="b",
+ pch=17, lty=2, col="blue")
> abline(h=c(30), lwd=1.5, lty=2, col="gray")
> library(Hmisc)
> minor.tick(nx=3, ny=3, tick.ratio=0.5)
> legend("topleft", inset=.05, title="Drug Type",
+ c("A","B"), lty=c(1, 2), pch=c(15, 17), col=c("red", "blue"))
> par(opar)
```



文本标注

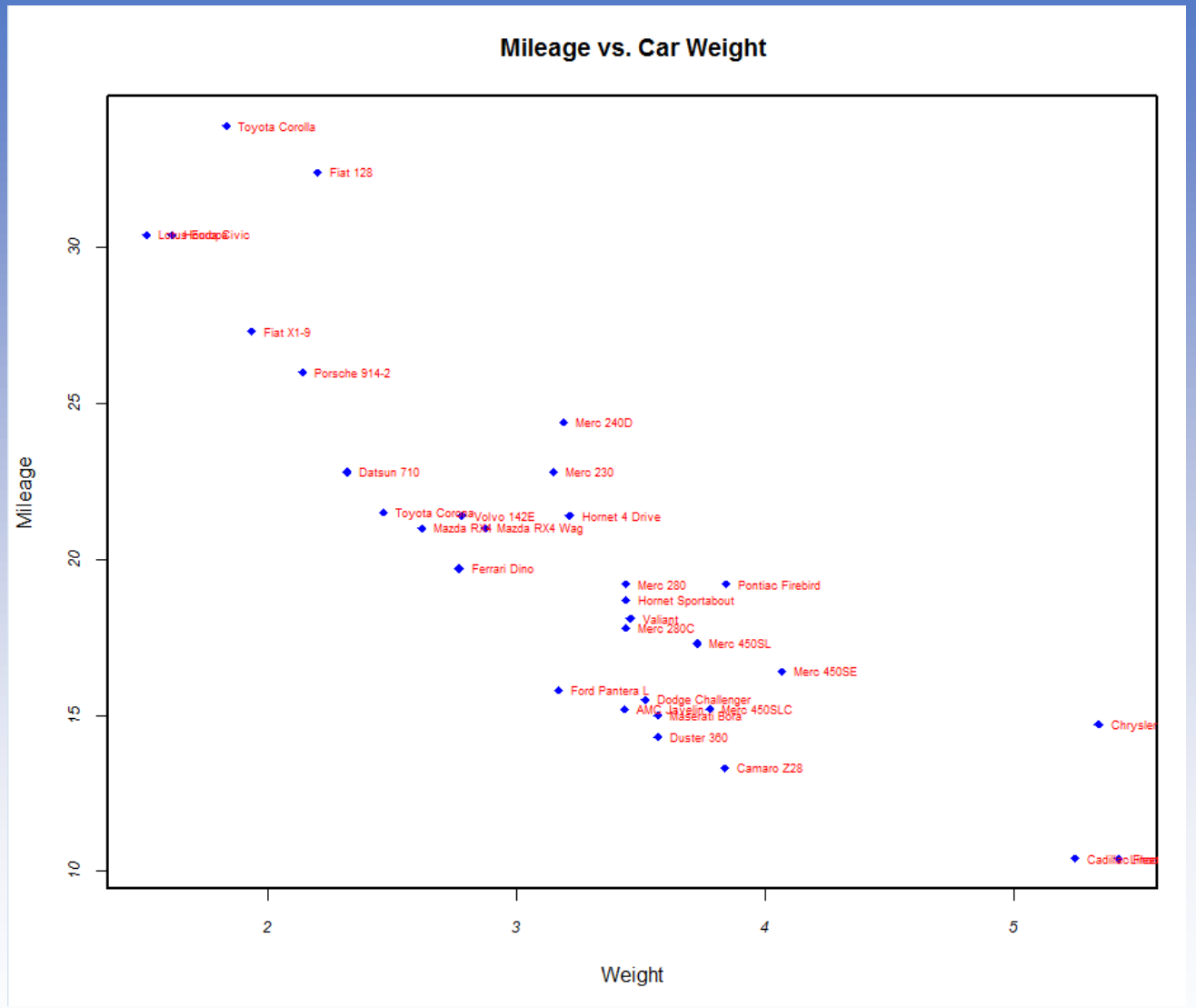
- text(*location*, "*text to place*", *pos*, ...)
- mtext("*text to place*", *side*, line=*n*, ...)
- 通过函数text()和mtext()将文本添加到图形上。text()可向绘图区域内部添加文本，而mtext()则向图形的四个边界之一添加文本:

表3-9　函数text()和mtext()的选项

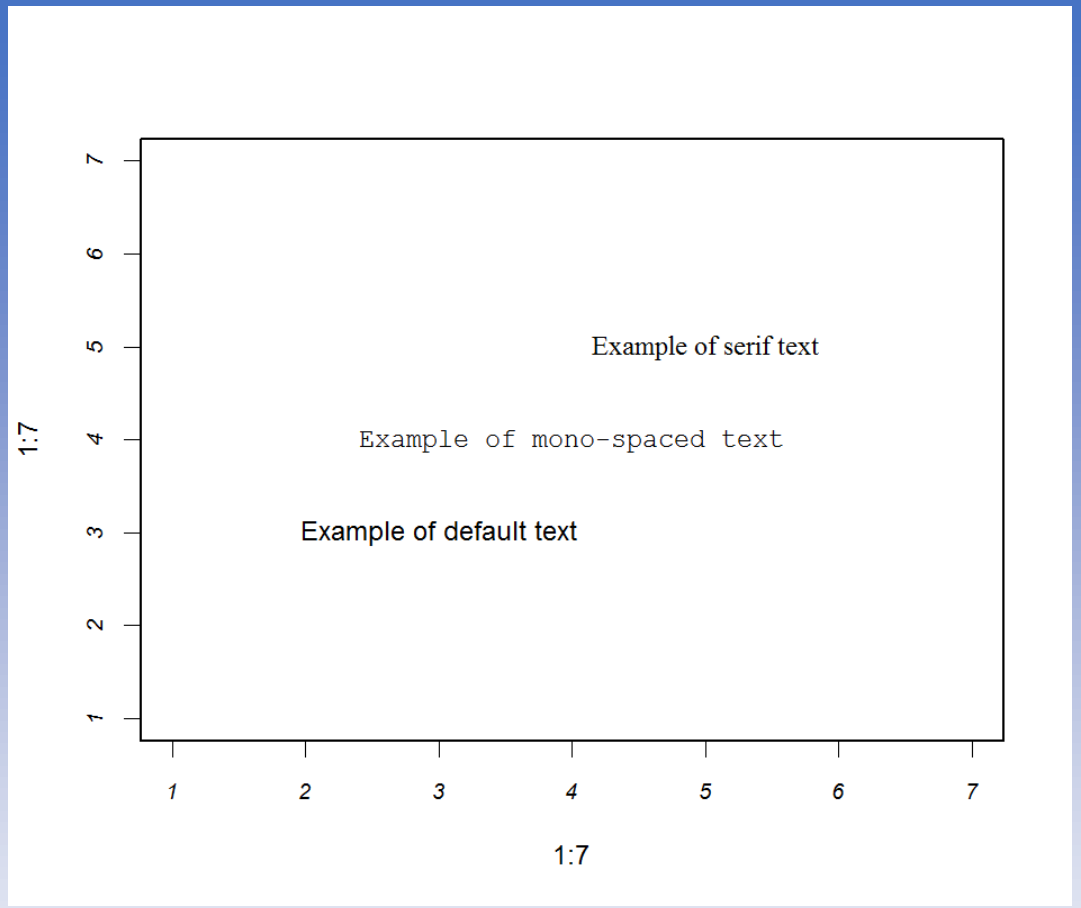
选　项	描　述
location	文本的位置参数。可为一对x,y坐标，也可通过指定location为locator(1)使用鼠标交互式地确定摆放位置
pos	文本相对于位置参数的方位。1=下,2=左,3=上,4=右。如果指定了pos,就可以同时指定参数offset=作为偏移量，以相对于单个字符宽度的比例表示
side	指定用来放置文本的边。1=下，2=左，3=上，4=右。你可以指定参数line=来内移或外移文本，随着值的增加，文本将外移。也可使用adj=0将文本向左下对齐，或使用adj=1右上对齐

文本标注

```
>attach(mtcars)
>plot(wt, mpg, main="Mileage vs. Car Weight", xlab="Weight", ylab="Mileage", pch=18, col="blue")
>text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")
>detach(mtcars)
```



```
> opar <- par(no.readonly=TRUE)
>par(cex=1.5)
>plot(1:7,1:7,type="n")
>text(3,3,"Example of default text")
>text(4,4,family="mono",
+    "Example of mono-spaced text")
>text(5,5,family="serif","Example of serif text")
>par(opar)
```



数学标注

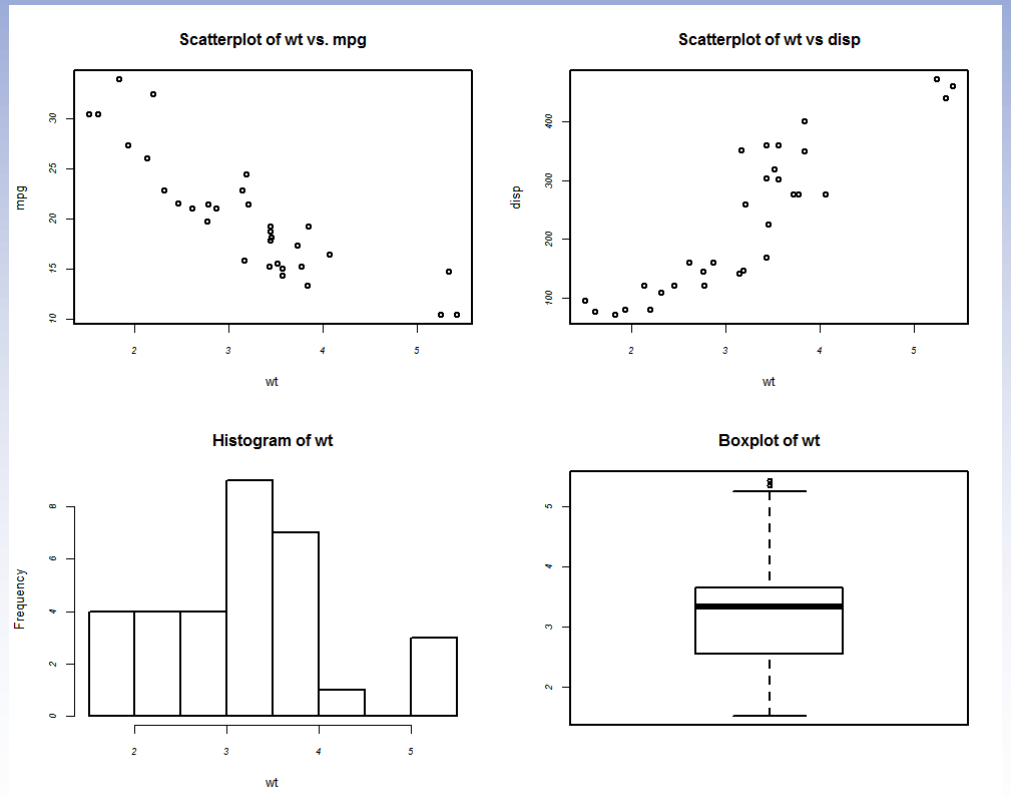
- 可以使用类似于TeX中的写法为图形添加数学符号和公式
- 函数plotmath()可以为图形主体或边界上的标题、坐标轴名称或文本标注添加数学符号

Arithmetic Operators		Radicals	
x + y	x + y	sqrt(x)	$\sqrt{x}$
x − y	x − y	sqrt(x, y)	$\sqrt[y]{x}$
x * y	xy	Relations	
x/y	x/y	x == y	x = y
x %+-% y	x ± y	x != y	x ↑ y
x%/%y	x √ y	x < y	x < y
x %*% y	x × y	x <= y	x ″ y
x %.% y	x · y	x > y	x > y
−x	−x	x >= y	x ⩾ y
+x	+x	x %~~% y	x ⊕ y
Sub/Superscripts		x %=~% y	x ≅ y
x[i]	x_i	x %==% y	x ≡ y
x^2	x^2	x %prop% y	x ∝ y
Juxtaposition		Typeface	
x * y	xy	plain(x)	x
paste(x,y,z)	xyz	italic(x)	*x*
Lists		bold(x)	**x**
list(x,y,z)	x, y, z	bolditalic(x)	***x***
		underline(x)	x̲

图形的组合

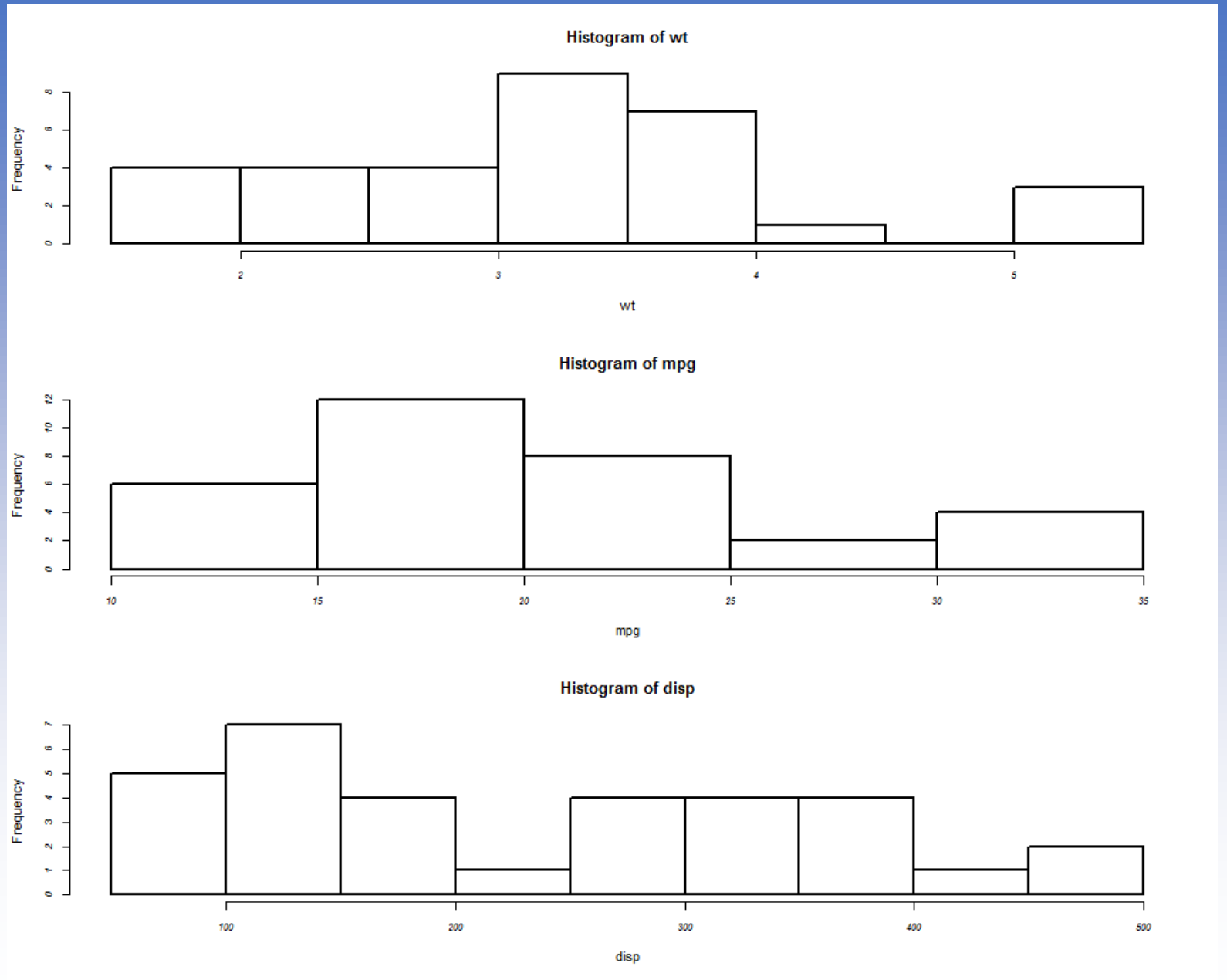
- 使用函数par()或layout()可以容易地组合多幅图形为一幅总括图形
- 可以在par()函数中使用图形参数mfrow=c(*nrows*, *ncols*)来创建按行填充的、行数为*nrows*、列数为*ncols*的图形矩阵。或使用nfcol=c(*nrows*, *ncols*)按列填充矩阵。

```
>attach(mtcars)
>opar <- par(no.readonly=TRUE)
>par(mfrow=c(2,2))
>plot(wt,mpg, main="Scatterplot of wt vs. mpg")
>plot(wt,disp, main="Scatterplot of wt vs disp")
>hist(wt, main="Histogram of wt")
>boxplot(wt, main="Boxplot of wt")
>par(opar)
>detach(mtcars)
```



图形的组合

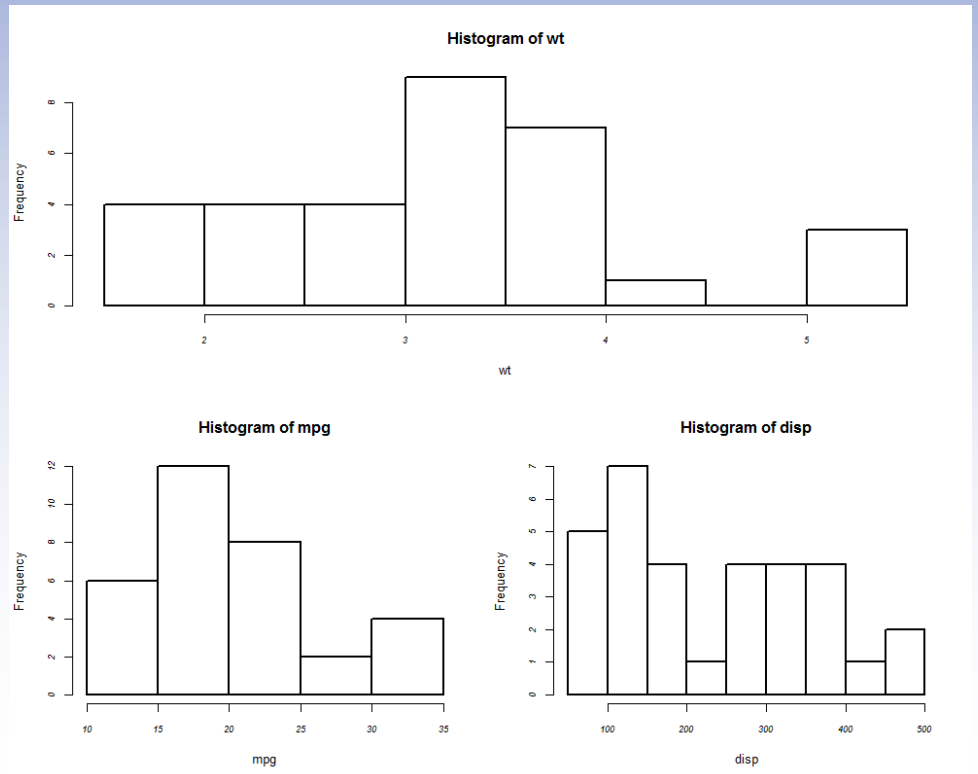
```
> attach(mtcars)
> opar <- par(no.readonly=TRUE)
> par(mfrow=c(3,1))
> hist(wt)
> hist(mpg)
> hist(disp)
> par(opar)
> detach(mtcars)
```



图形的组合 layout()

- 函数layout()的调用形式为layout(*mat*)，其中的*mat*是一个矩阵，它指定了所要组合的多个图形的所在位置。
- 在以下代码中，一幅图被置于第1行，另两幅图则被置于第2行：

```
> attach(mtcars)
> layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
> hist(wt)
> hist(mpg)
> hist(disp)
> detach(mtcars)
```



图形的组合 layout()

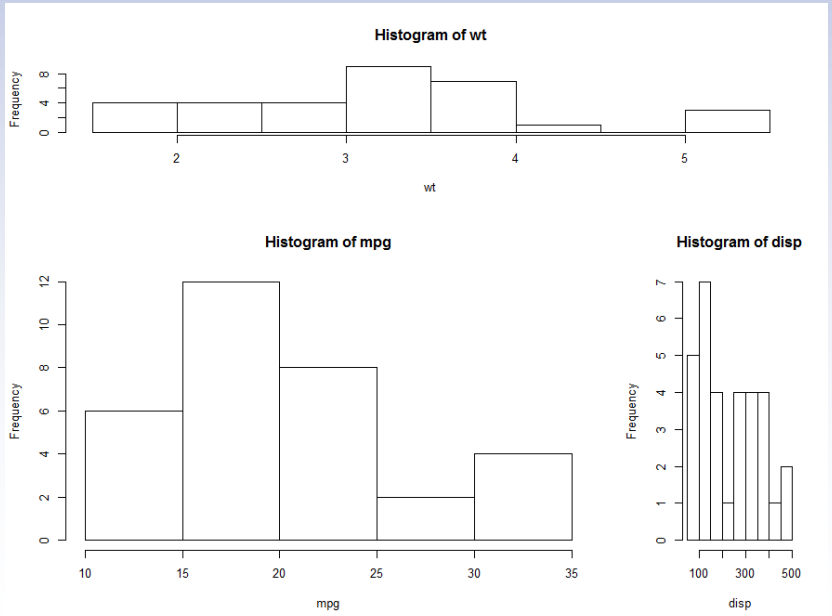
- 更精确地控制每幅图形的大小，可以有选择地在layout()函数中使用widths=和heights=两个参数。其形式为：

widths = 各列宽度值组成的一个向量

heights = 各行高度值组成的一个向量

- 相对宽度可以直接通过数值指定，绝对宽度（以厘米为单位）可以通过函数lcm()来指定。

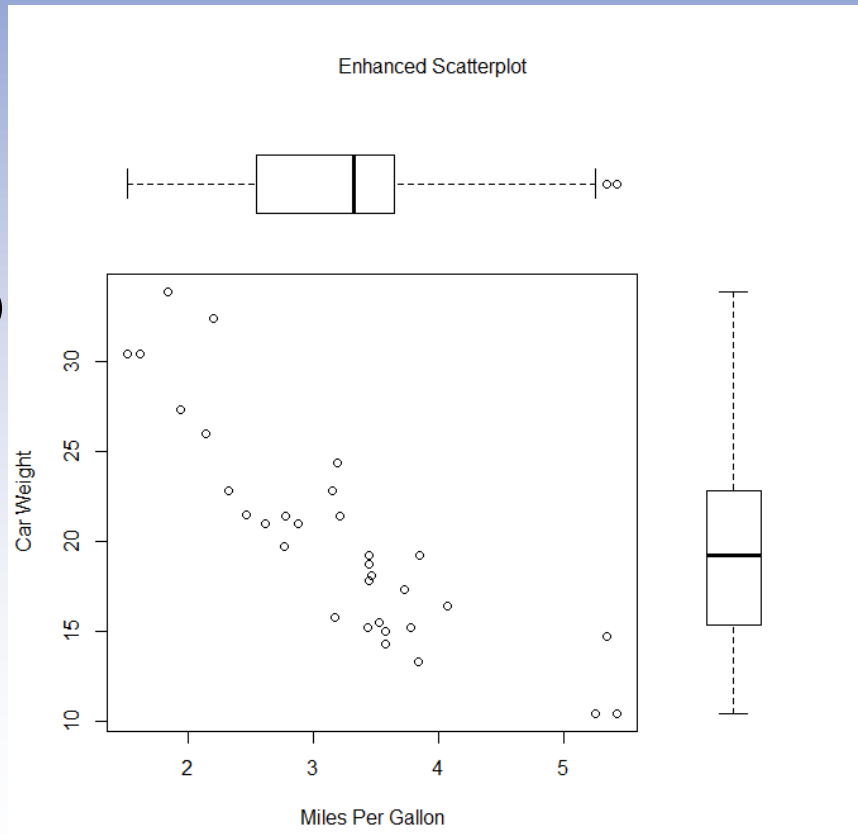
```
> attach(mtcars)
> layout(matrix(c(1, 1, 2, 3), 2, 2, byrow = TRUE),
+  widths=c(3, 1), heights=c(1, 2))
> hist(wt)
> hist(mpg)
> hist(disp)
> detach(mtcars)
```



图形布局的精细控制

- 很多时候，你想通过排布或叠加若干图形来创建单幅的、有意义的图形，这需要有对图形布局的精细控制能力。你可以使用图形参数fig=完成这个任务。

```
> opar <- par(no.readonly=TRUE)
> par(fig=c(0, 0.8, 0, 0.8))
> plot(mtcars$wt, mtcars$mpg, xlab="Miles Per Gallon", ylab="Car Weight")
> par(fig=c(0, 0.8, 0.55, 1), new=TRUE)
> boxplot(mtcars$wt, horizontal=TRUE, axes=FALSE)
> par(fig=c(0.65, 1, 0, 0.8), new=TRUE)
> boxplot(mtcars$mpg, axes=FALSE)
> mtext("Enhanced Scatterplot", side=3, outer=TRUE, line=-3)
> par(opar)
```



总结

- 本章内容，涵盖教材“R语言介绍、创建数据集、图形初阶”三章内容。
- 需要了解：R的安装、包的安装、运行平台使用、交互和批处理两种方式运行程序、如何将工作保存到文本和图形文件。
- 数据结构：向量、矩阵、数据框和列表等概念的表达、将数据导入到R中的方法
- 如何修改一幅图形的坐标轴、字体、绘图符号、线条和颜色，以及如何添加标题、副标题、标签、文本、图例和参考线，如何指定图形和边界的大小，以及将多幅图形组合为实用的单幅图形。

- 需要熟练掌握R的数据结构、了解图形功能；
- 不要求死记硬背，具体应用可以通过查教材、使用在线帮助等实现相应功能即可；
- 后边将安排一次利用实际观测数据进行绘图的作业。

谢谢各位！