R 语言简明教程

第三课、用 graphics 包绘图

张金龙
jinlongzhang01@gmail.com

2022 年 2 月 13 日

《R 语言简明教程》内容安排（I）

- 第一课、R 语言及其编程环境
- 第二课、数据类型、结构和基础操作
- **第三课、用 graphics 包绘图**
- 第四课、用 ggplot2 包绘图
- 第五课、tidydata 以及 tidyverse 系列程序包的使用
- 第六课、基础统计方法的实现
- 第七课、编写 R 函数
- 第八课、Rmarkdown 以及可重复性数据分析
- 第九课、git 与版本控制

《R 语言简明教程》内容安排（II）

- 第十课、R 程序包的编写、检查与发表
- 第十一课、生物多样性分析常用程序包
- 第十二课、空间数据处理和地图绘制
- 第十三课、系统发育研究的常用程序包
- 第十四课、课程总结

内容提要

1. 绘图是为了更好和读者交流
2. 绘图的一些基本概念
3. 图形的基本结构
4. 图的类型根据数据类型而定
5. R 常用绘图函数
6. 常用绘图函数的参数
7. 图形的进一步调整
8. 图像保存
9. 怎样设计出让人赏心悦目的图形

1

绘图是为了更好和读者交流

绘图是为了和读者交流

作者的信息 -> 读者

- 作者：为了传递从数据中提取到的信息（作者的认知）
- 读者：为了理解图中的信息说明了什么（作者要传递的信息）

准则：

- 将信息浓缩，只展示最重要的信息
- 每个图形要只展示一个信息点

R 的几个绘图系统

- *base + graphics*: 最早的绘图系统，至今仍然广泛应用
- 基于 *grid* 的 *ggplot2* 系列的各种程序包: 现在应用广泛，影响深远
- *Lattice*: 模仿 Splus 的 Trellis Plot，现已衰落
- *plotly*、*leaflet* 等，基于 HTML5 和 Javascript，主要用于网络环境，一般用其他程序包转换生成。

graphics 包

- graphics 包是 R 的核心程序包，由 R Core Team 编写和维护，用 graphics 绘图又称 R-base 绘图
- 提供了一系列 “高等级” 绘图函数，可以用来绘制多种复杂图形，包括线图、柱状图、散点图、热图等
- 提供了一系列 “低等级” 绘图函数，点、线、多边形、文本等，用户可通过这些函数来绘制各种图形
- 可生成 pdf、tiff、svg、png、jpeg、eps 等各种文件格式

graphics 包正被 ggplot2 替代?

ggplot2 程序包推出后，因：

- 输入数据格式统一
- 代码整齐易读
- 各图层之间能更好整合、联动
- 默认图例生成和管理更为人性化
- 提供绘制多种高级图形的方法
- ……

所绘制图形的质量空前提高，因此 graphics 包逐渐被冷落。

问题：*用 graphics 包绘图，是否还生活在”石器时代”?*

为什么仍然要学习 graphics 包?

- 代码简单，图形简洁，容易理解，很多情况下已经够用（“最小” 学习成本）
- “历史悠久”，早期的 R 绘图的大都用 graphics 包完成（“祖传” 代码）
- 仍然有很多用户写 graphics 代码，特别是在 ggplot2 推出之前就掌握了 R 的用户，目前仍然有大量图形是用 graphics 实现的（交流的需要）

graphics 包绘图的特点

1. 语句简单直接: `plot`（优点）
2. 图形简洁：也有人说”简陋”(优点? 缺点?)
3. 既有高级绘图功能，也有低级绘图功能（优点?）
4. 图形各部分可以精确控制（优点）
5. 参数”繁多”（优点? 缺点?）
6. 图形各部分之间相对独立，因此，图例和图形元素相对脱节，难以自动更新（缺点）
7. 有大量 R 的”传统”绘图代码（优点?）

2

绘图的一些基本概念

绘图设备

- R 绘图，是将处理结果输送到绘图 “设备”。设备可以是显示器，也可以是电脑上的某个文件。
- 每次运行完代码，设备显示的是代码的最终效果，要更改图形，就需要修改代码，重新绘图，直到满意为止

问题: 如果还不满意，咋办?

图形文件的属性

1. 文件格式：
   - 栅格图：tiff、jpg、jpeg、png、bmp 文件
   - 矢量图：svg、eps 或者 pdf
2. 大小：长、宽（画布的尺寸，像素（pixel）、厘米（cm）、英寸（inch））
3. 分辨率（只适用于栅格图）：单位 dpi，一般 300dpi-600dpi
4. 压缩方式：有些格式要求设定一定的压缩格式，如 tiff 文件一般较大，一般选择 lzw 方式压缩。
5. 颜色空间：RGB （红、绿、蓝叠加，适用于电脑屏幕看图）vs. CMYK（适用于印刷）

图片类型

- tiff：栅格图，无损压缩，一般用 lzw 压缩，多用于刊物投稿
- jpeg：栅格图，相机、手机一般默认保存为 jpeg，有时候也写作 jpg，多用于网页，图片较小
- png：栅格图，多用于网页，图片较小，质量却较高
- gif：栅格图，多用于制作“动图”，可嵌入网页，图片较小
- pdf：矢量图，多用于投稿、出版，质量高，文件较小
- svg：矢量图，一般用于编辑，也用于网络，质量高，文件较小

3

图形的基本结构

R-base 图形的基本结构

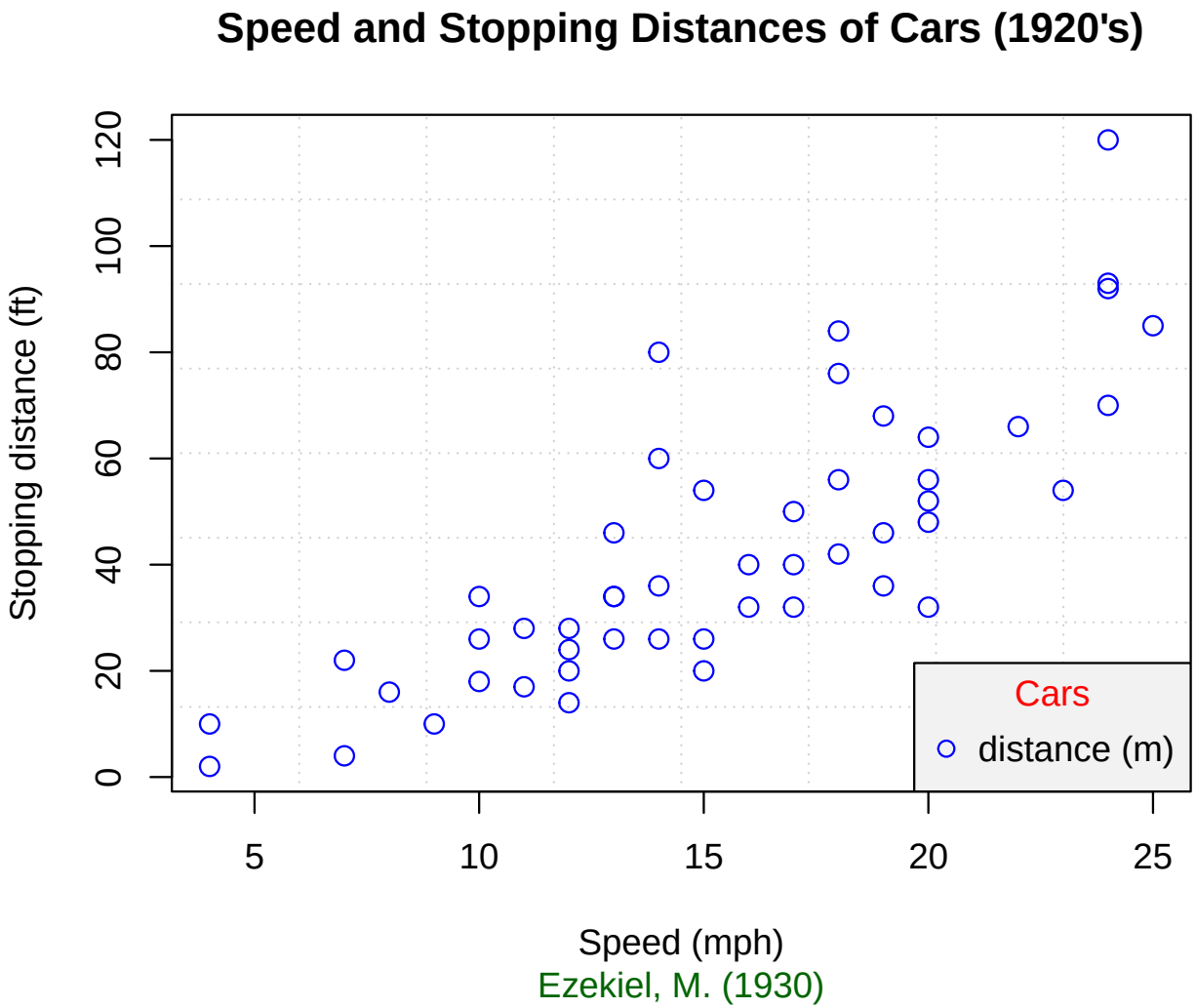


图 1: 图形的基本结构

R-base 图形的基本结构

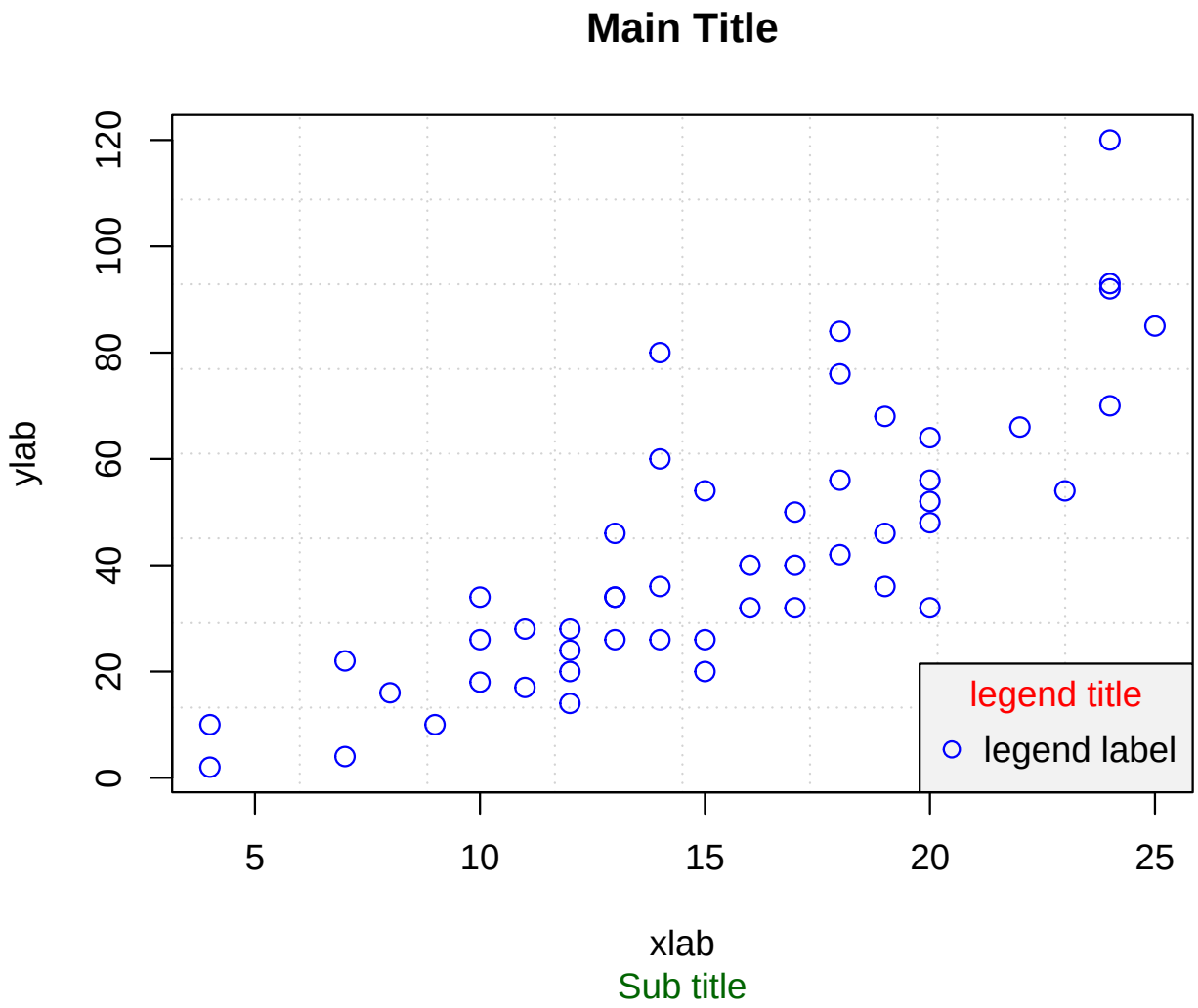


图 2: 图形的基本结构

图的基本要素

1. 绘图区域（边框）大小
2. x 轴的范围及刻度（刻度是向内还是向外）
3. y 轴的范围及刻度
4. 坐标轴名称和大小、颜色和方向
5. 点的形状（`pch`）、大小（`cex`）；线的粗细（`lwd`）、形状（`lty`）；颜色（`col`）
6. 标题和副标题
7. 边距
8. 字体（默认是 Sans Serif，无衬线字体）
9. 图例（颜色、形状以及对应的文字释义）

散点图 + 线性回归线

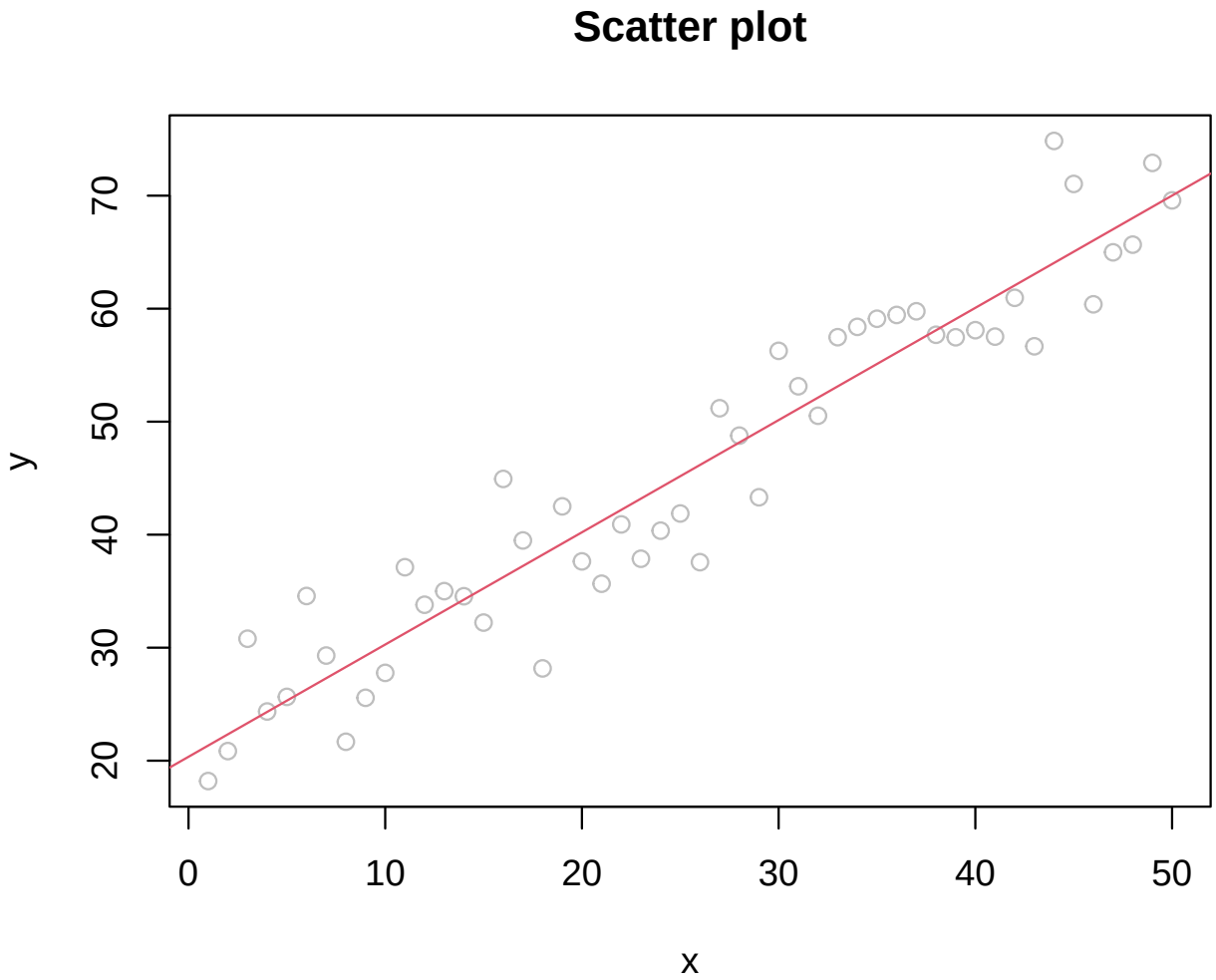


图 3: 散点图 + 线性回归线

散点图 + 线性回归线

```
y = 1:50 + (5*rnorm(50) + 20)
x = 1:50

plot(y ~ x, main = "Scatter plot", col = "gray")
abline(lm(y ~ x), col = 2)
```

R-base 图形的代码（调用 x、y 参数）

```
plot(x = Speed, y = Distance)
```

R-base 图形的代码（用公式，表示变量间的关系）

```
plot(Distance ~ Speed)
```

绘图中的”公式”

```
## Default S3 method:
plot(x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
     log = "", main = NULL, sub = NULL,
     xlab = NULL, ylab = NULL, ...)
```

也可以写作

```
y ~ x
Distance ~ Speed
```

plot 函数，会根据数据的类型，自动猜测要绘制的图形。

4

图的类型根据数据类型而定

数据的类型

- 连续变量（例如植株的高度）
- 离散变量
  - 有序离散变量 (例如植物的濒危等级)
  - 无序离散变量 (例如花朵的颜色)

散点图 scatter plot

一般用来展示两个变量之间的关系，若关系显著，常添加一些趋势线

```
Speed <- cars$speed
Distance <- cars$dist
plot(Speed, Distance,
  panel.first = lines(stats::lowess(Speed, Distance), lty = "d
  pch = 0, cex = 1.2, col = "blue"
)
```

散点图 scatter plot

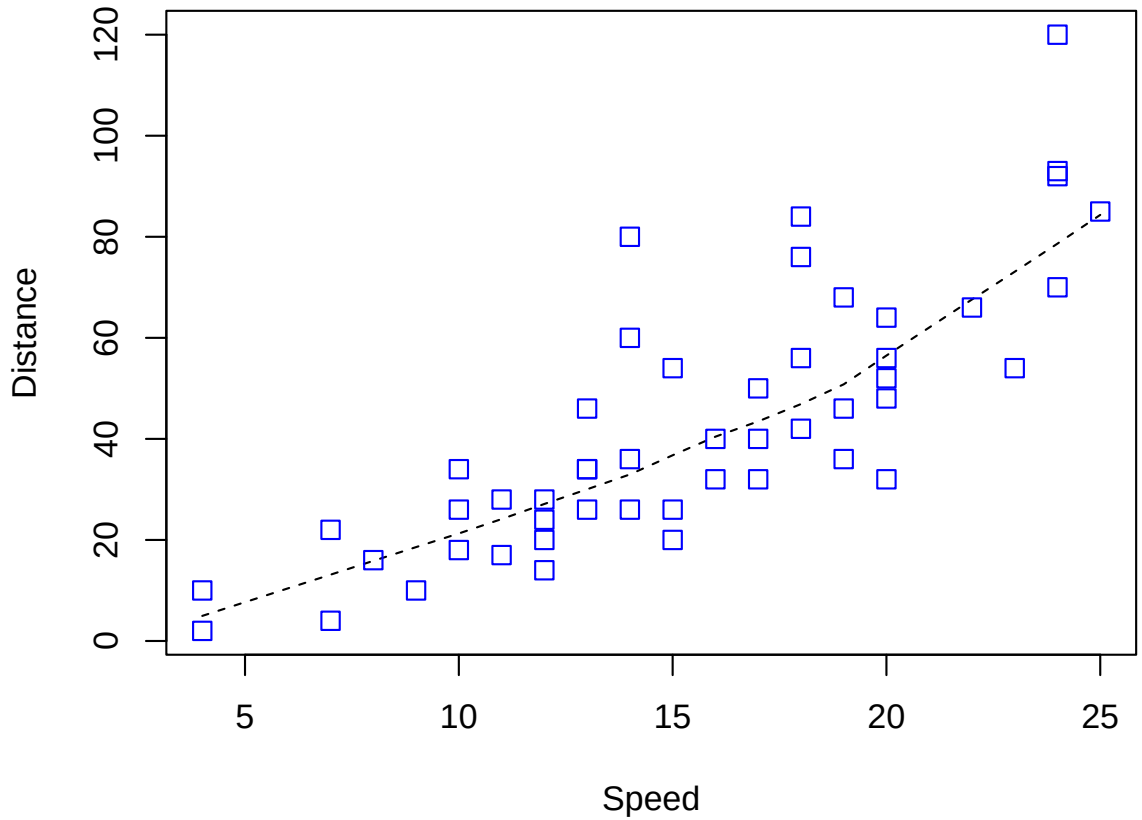


图 4: 散点图 + 趋势线

折线图 line chart

多用来表示某一个或者几个变量随时间的变化。

```
z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), freque
class(z)
head(z) # as "matrix"
plot(z)
```

折线图 line chart

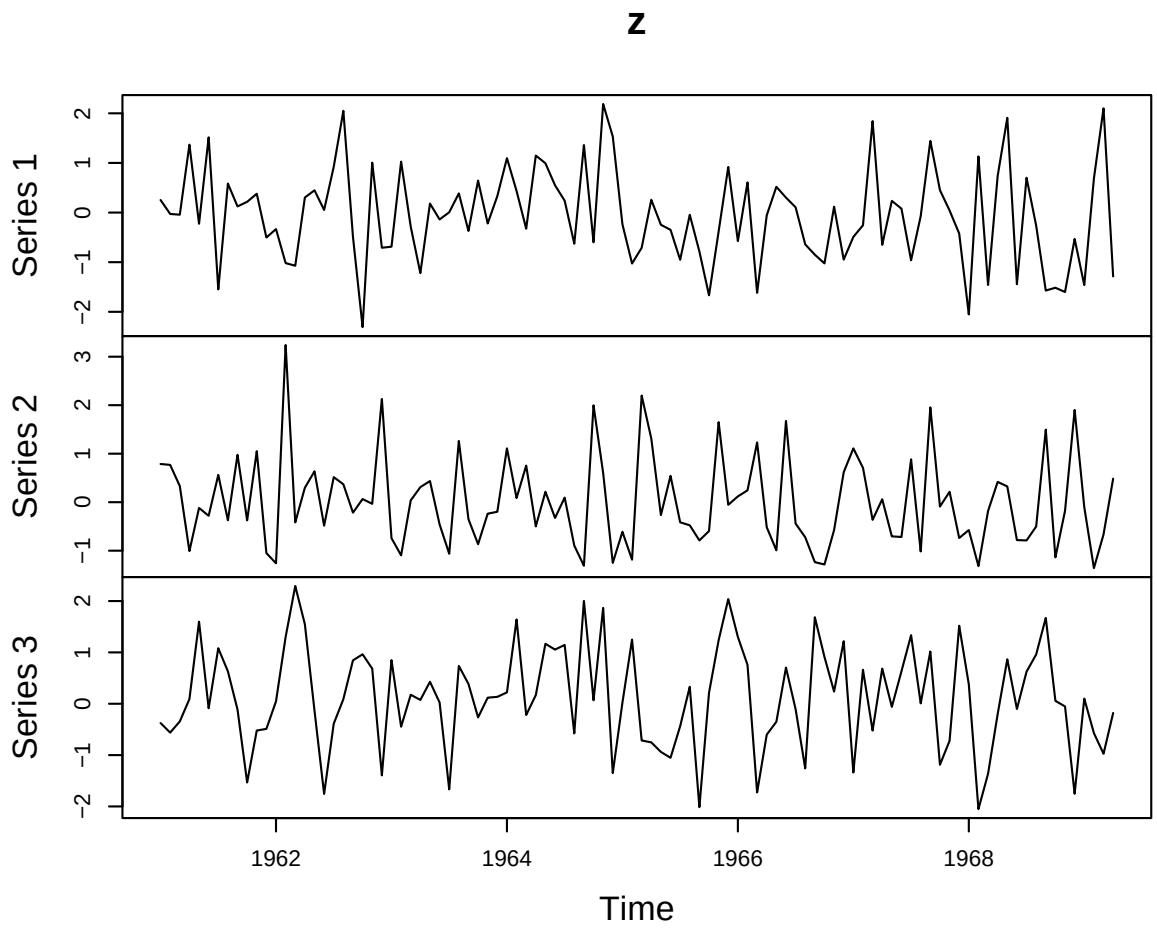


图 5: 折线图

柱状图 barplot

表示不同类型数据的高度，便于不同组的比较。如做方差分析、多重比较，常常会用到柱状图，并用 a、b、c 等标注出组与组之间差异是否显著。

```
barplot(GNP ~ Year, data = longley)
```

柱状图 barplot

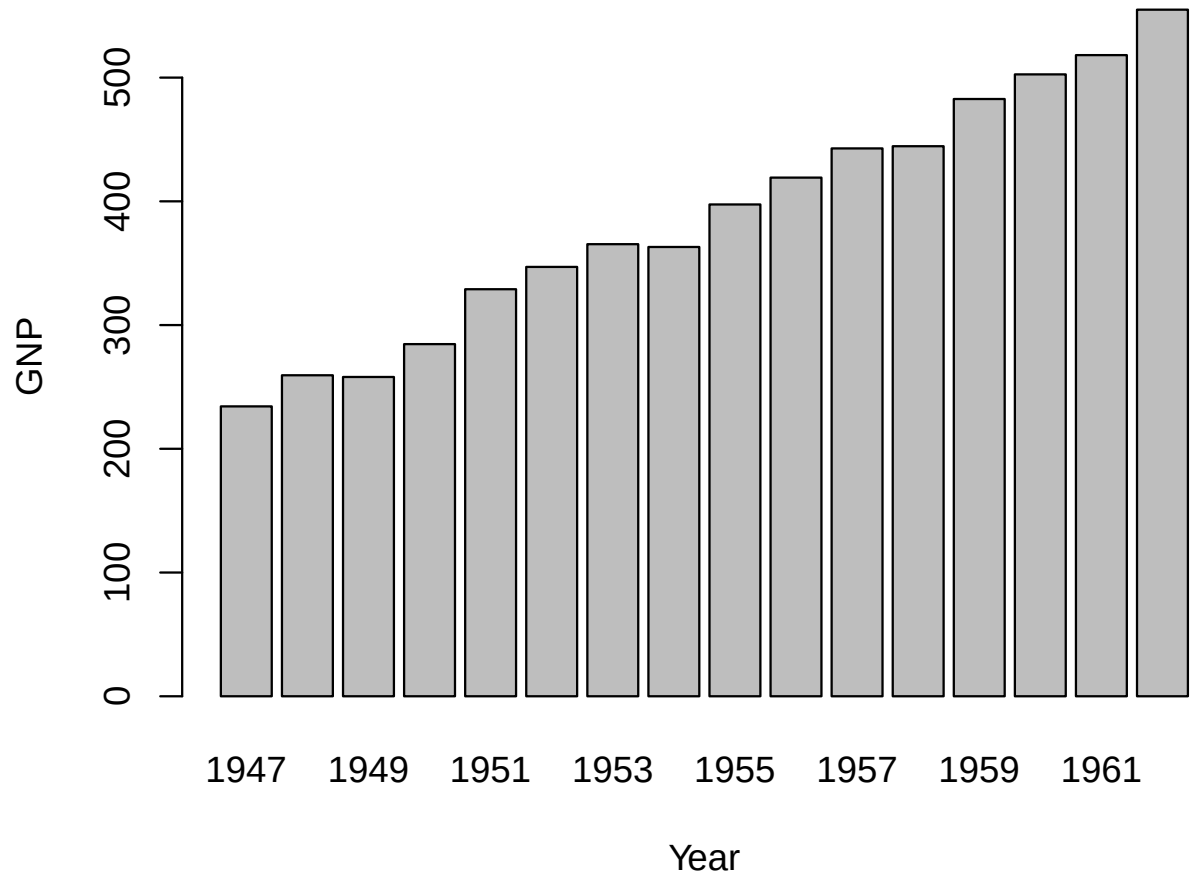


图 6: 柱状图

饼图 pie chart

一般用来表示不同类型数据所占的比例，有些统计学家非常讨厌饼图，认为其扇形的角度不太直观，因此极力推荐柱状图。

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

饼图 pie chart

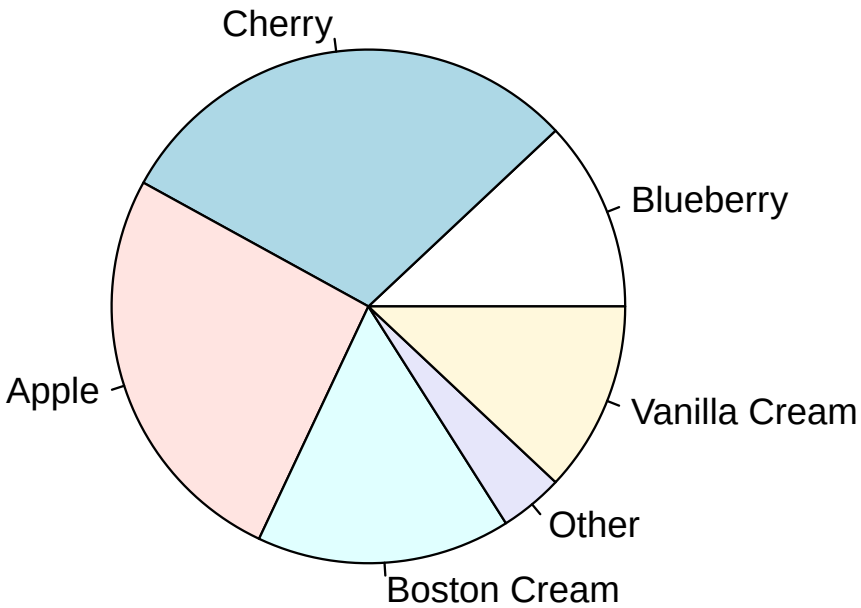


图 7: 饼图

频度直方图 histogram

将数据分成若干区间，展示每个区间出现的次数。

```
hist(rnorm(100))
```

频度直方图 histogram

将数据分成若干区间，展示每个区间出现的次数。

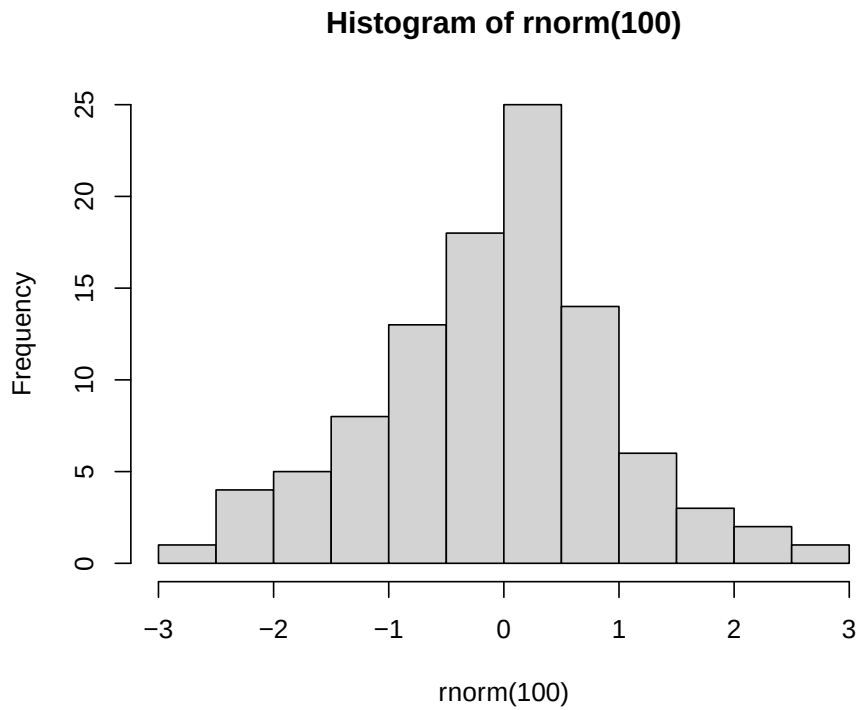


图 8: 频度直方图

箱线图 boxplot

表示一系列数字的统计分布，上中位数、中位数、下中位数、异常值等。

```
rb <- boxplot(decrease ~ treatment, data = OrchardSprays, col
title("OrchardSprays data")
```

箱线图 boxplot

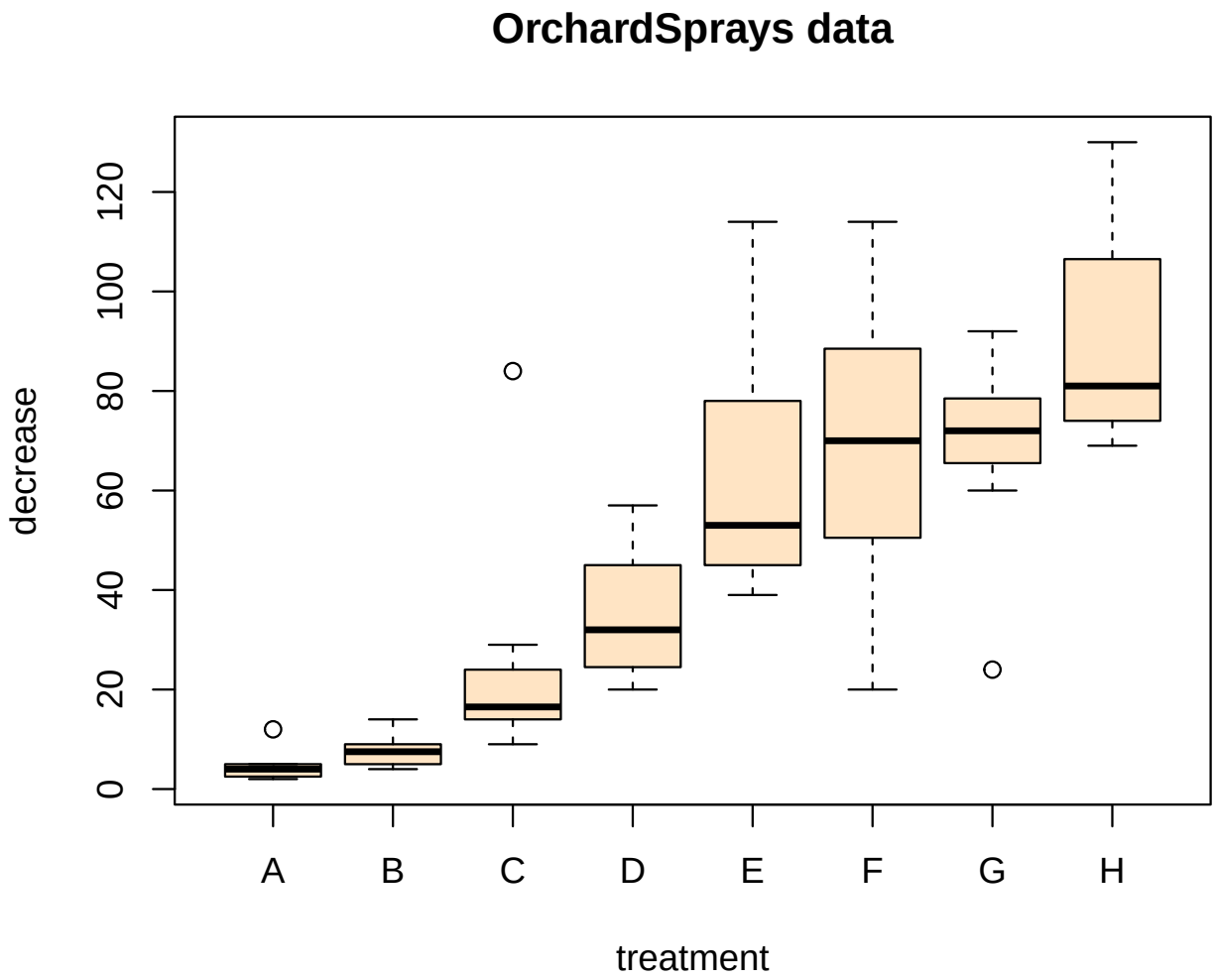


图 9: 箱线图

树状图 dendrogram

表示各操作单元的相似程度，常用于聚类分析。

进化树图是一种特殊的树状图，表示分类单元之间的系统发育关系。

树状图 dendrogram

```
hc <- hclust(dist(USArrests), "ave")
(dend1 <- as.dendrogram(hc)) # "print()" method
plot(dend1,
  nodePar = list(pch = 2:1, cex = .4 * 2:1, col = 2:3),
  horiz = TRUE,
  cex = 0.5
)
```

树状图 dendrogram

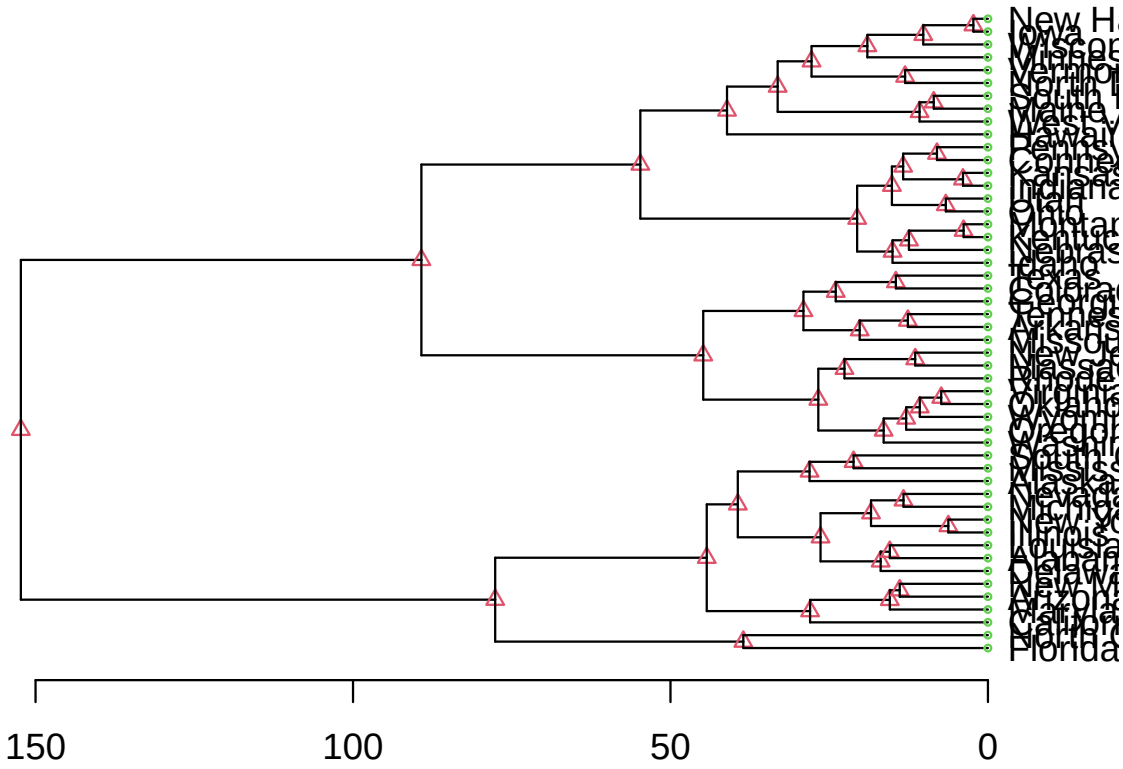


图 10: 树状图

双标图 Biplot

在群落生态学中常用，PCA、CA、CCA、RDA、NMDS 等。

表示转换之后，点与点之间，在前两个维度的”距离”。

双标图 Biplot

```
library(vegan)
data(varespec)
data(varechem)
vare.cca <- cca(varespec, varechem)
vare.cca
plot(vare.cca)
```

双标图 Biplot

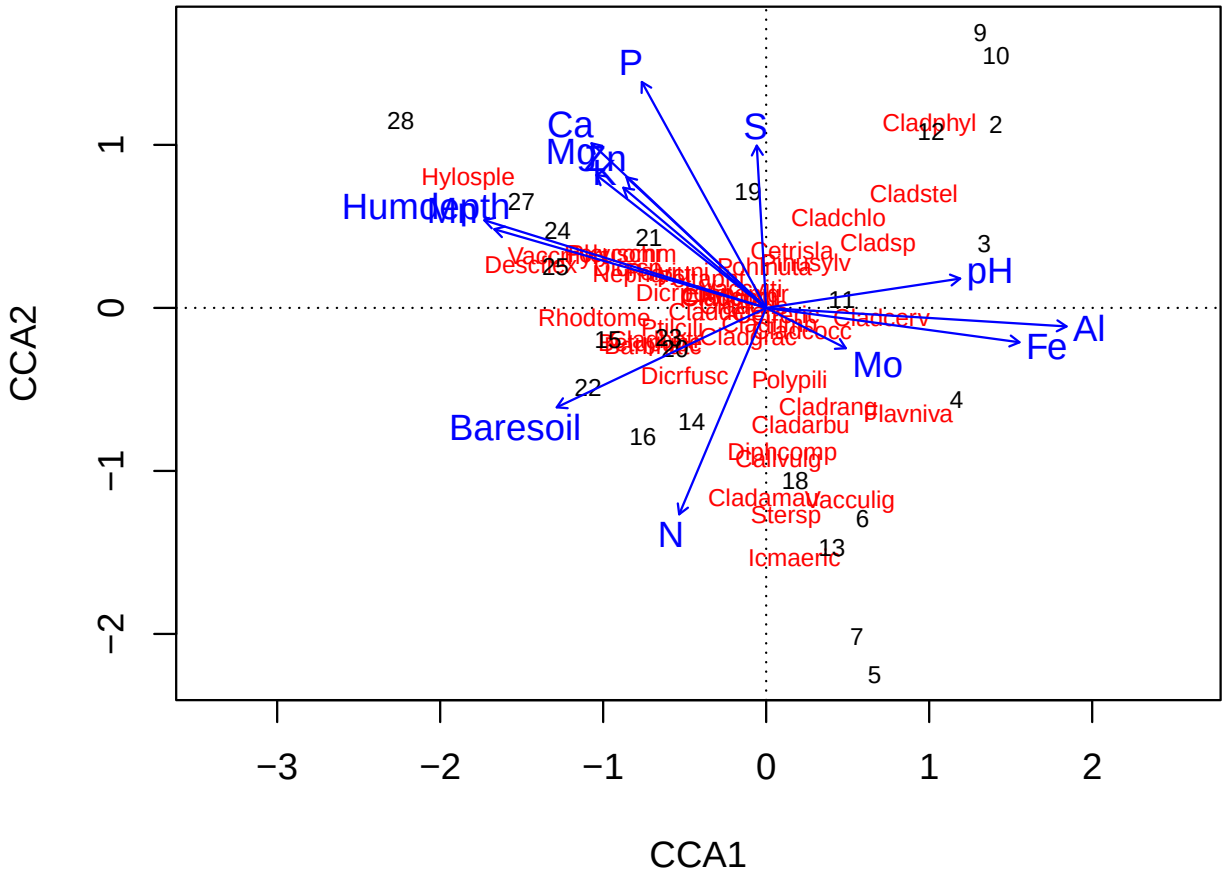


图 11: 双标图

地图 map

样地位置、研究范围、采样点位置、研究区域的地形、降水、温度等，以及模型预测的结果

其要素包括边界、地形、图例尺、指北针、标题

```
library(tmap)
data(land)
plot(land, main = "Land Cover")
```

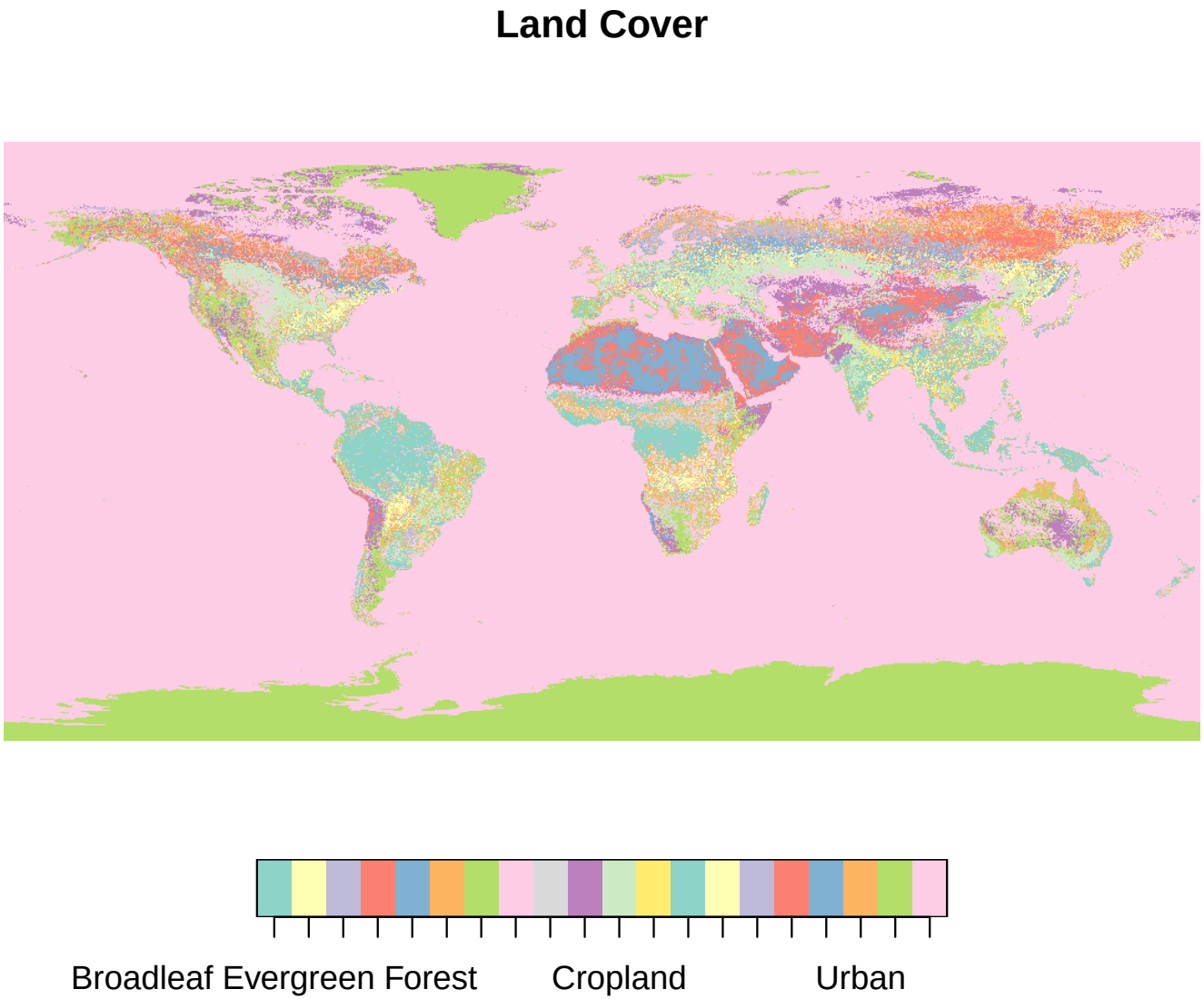


图 12: 地图

双 y 坐标图 plot with two y axis （尽量不要使用）

```
set.seed(101)
x <- 1:10
y <- rnorm(10)
## second data set on a very different scale
z <- runif(10, min = 1000, max = 10000)
par(mar = c(5, 4, 4, 4) + 0.3) # Leave space for z axis
plot(x, y) # first plot
par(new = TRUE)
plot(x, z, type = "l", axes = FALSE, bty = "n", xlab = "", yla
axis(side = 4, at = pretty(range(z)))
mtext("z", side = 4, line = 3)
```

双 y 坐标图 plot with two y axis （尽量不要使用）

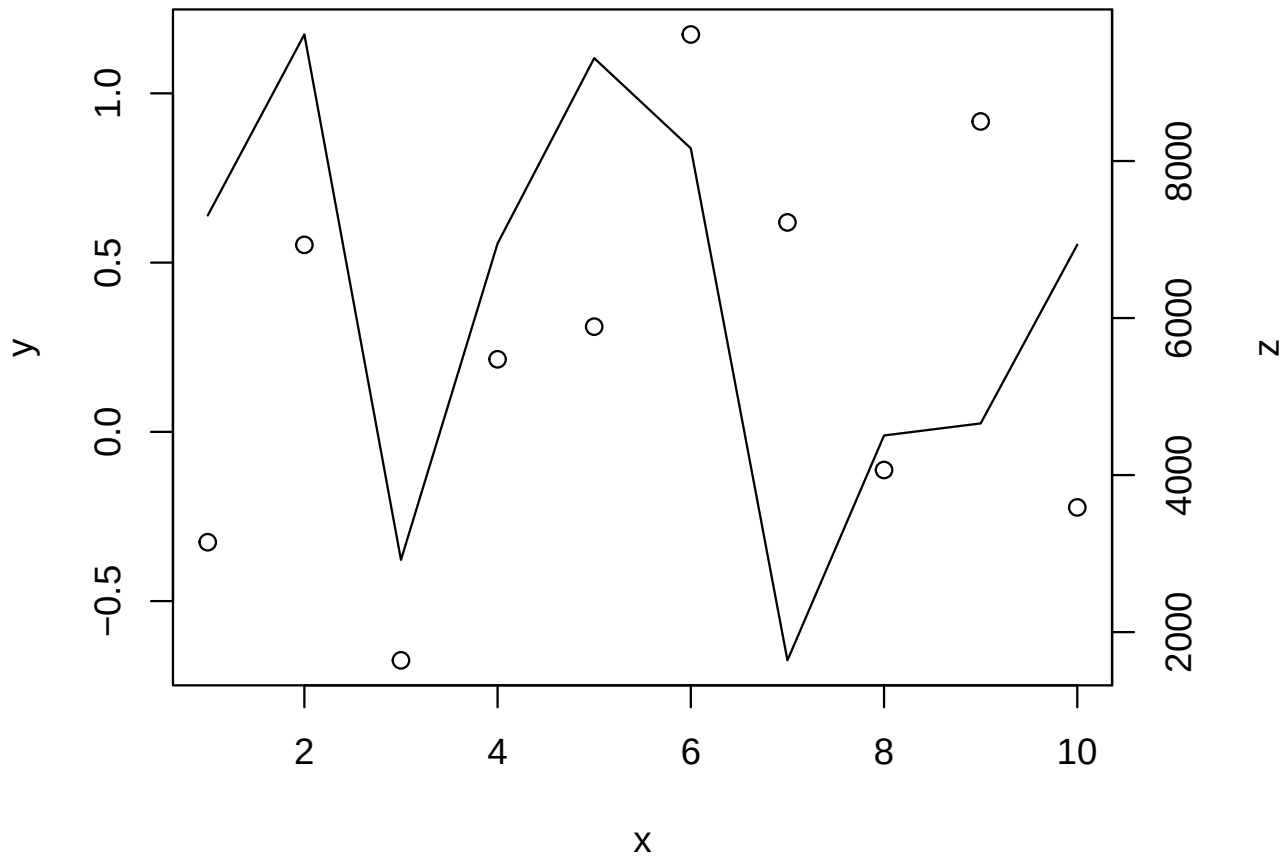


图 13: 双 y 坐标图

气候图（双 Y 坐标）

```
library(climatol)
data(datcli)
diagwl(datcli, est = "Example station",
       alt = 100, per = "1961-90",
       mlab = "en")
```

气候图（双 Y 坐标）

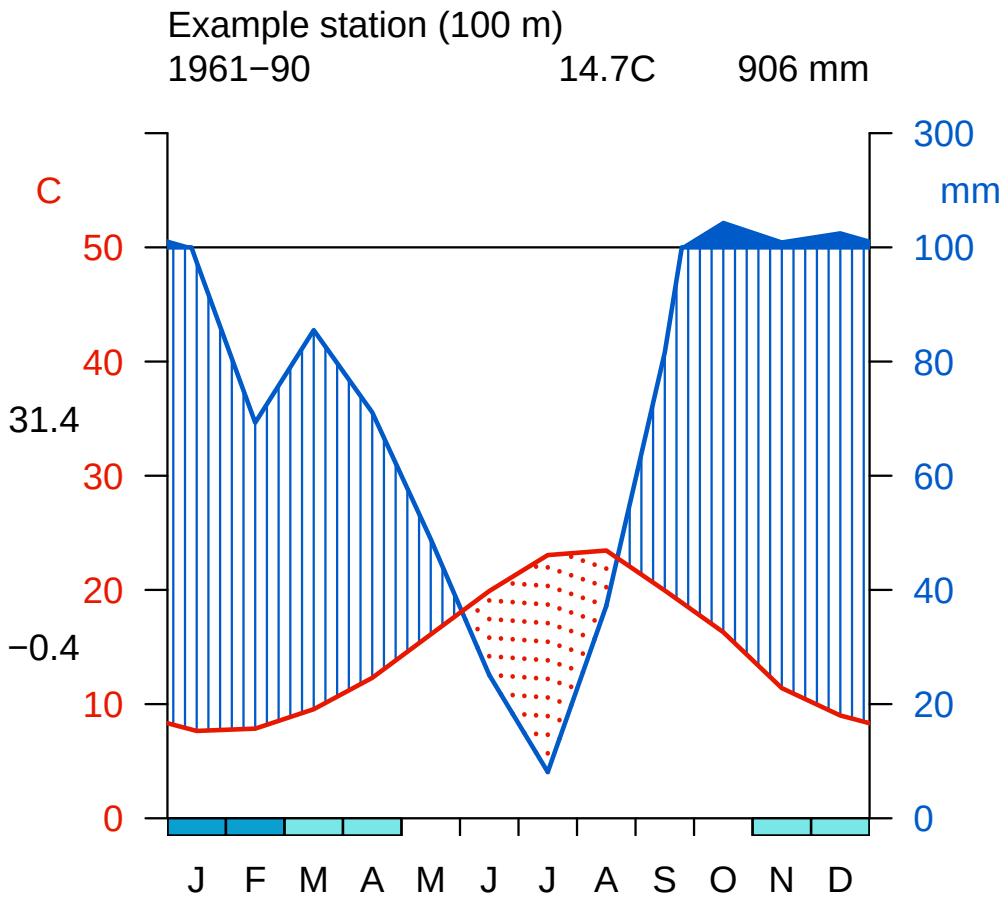


图 14: 气候图（双 Y 坐标）

三维图 3-D plot

```
library(plot3D)
x <- seq(1, nrow(volcano), by = 2)
y <- seq(1, ncol(volcano), by = 2)
V <- volcano[x, y]

persp3D(z = V)
```

三维图 3-D plot

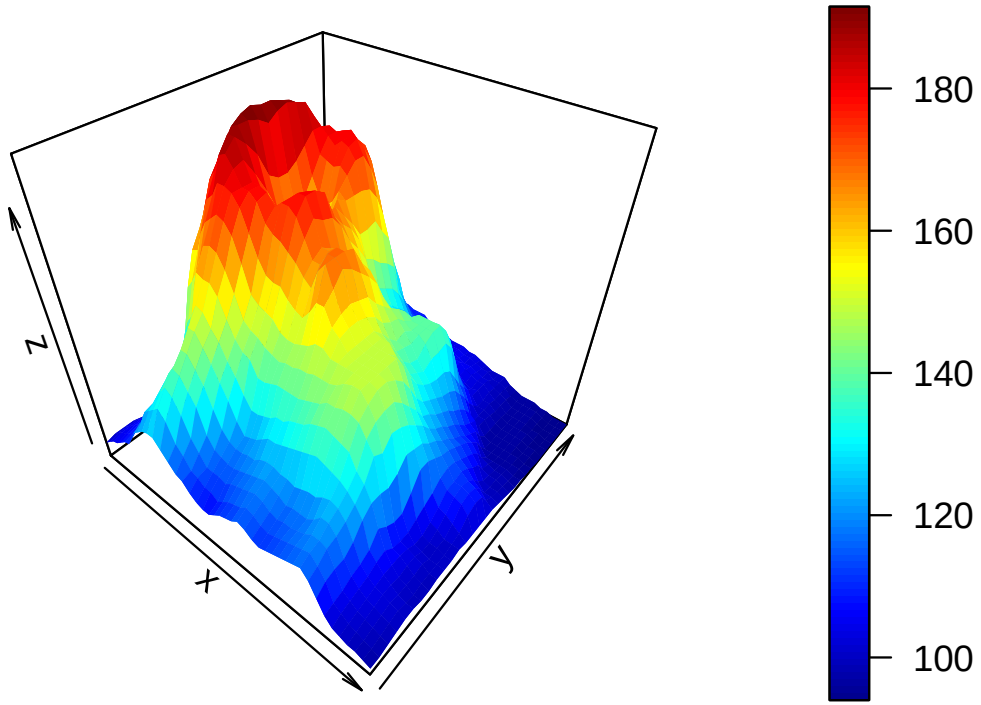


图 15: 三维图

三维图举例

```
x <- seq(-10, 10, length= 30)
y <- x

f <- function(x, y) {
    r <- sqrt(x ^ 2+y ^ 2);
    10 * sin(r)/r
}

z <- outer(x, y, f)
z[is.na(z)] <- 1
op <- par(bg = "white")
persp(x, y, z, theta = 30, phi = 30,
    expand = 0.5, col = "lightblue")
```

三维图举例

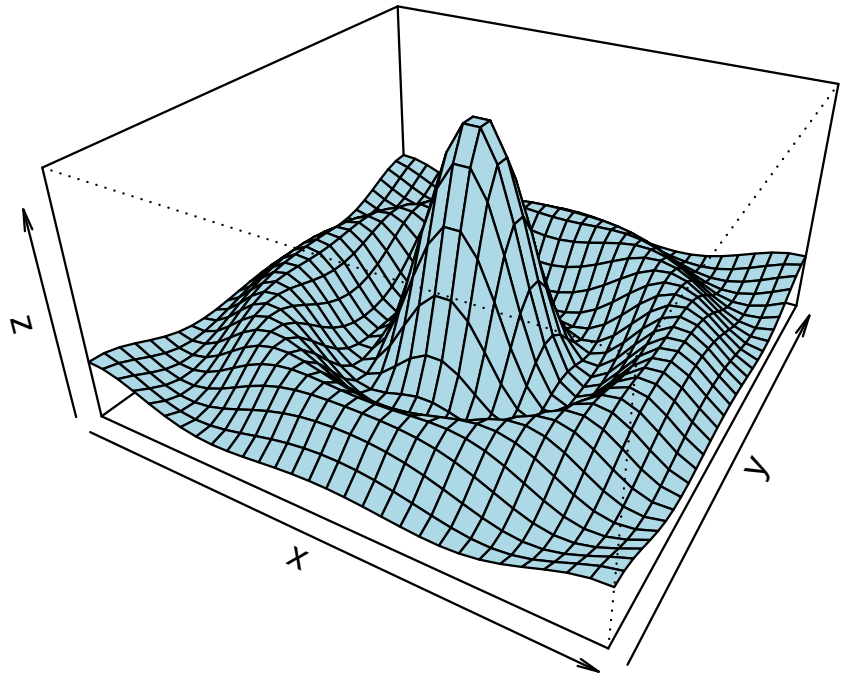


图 16: 三维图举例

三维图

- 是否需要三维图？除非确有必要，不要用三维图。
- 绘制三维图的函数/包
  - `persp()` 函数
  - `plot3D`
  - `lattice` 包 `wireframe()`
  - `scatterplot3d` 包
  - `rgl`
  - …

5

R 常用绘图函数

高等级绘图函数

高等级绘图函数，一般直接可以打开 R 的绘图设备，按照默认的参数（如字体、字体大小、线条粗细、颜色、图例大小、位置）等，按照预先设定，绘制对应的图形。

但是，一般来说，这些图形仍然要做一定调整，才能达到出版要求。

常见高等级绘图函数

```
plot()              # 散点图
hist()              # 频度直方图
boxplot()           # 箱线图
barplot()           # 柱状图
piechart()          # 饼图
legend()            # 图例
curve()             # 曲线
contour()           # 等高线
filled.contour()    # 等高线设色图
image()             # 栅格图
persp()             # 三维图
```

低等级/底层绘图函数

绘制点、线、多边形等，一般用于在高等级绘图函数设定好图形的基本参数后，往图形上添加组件。

```
lines()       # 折线
abline()      # 给定斜率，添加斜线
points()      # 加点
segments()    # 加线段
axis()        # 坐标轴
box()         # 添加图的框
title()       # 标题
text()        # 添加文字
```

用低等级绘图函数绘制散点图

```
x <- runif(50,0,2); y <- runif(50,0,2) # 生成数据
# 空白背景
plot(x, y, type="n", xlab="", ylab="", axes=FALSE)
points(x,y) # 添加点
axis(1);    # 添加横轴
axis(at=seq(0.2,1.8,0.2), side=2) # 添加纵轴
box()       # 添加边框
title(main="Main title", sub="subtitle", xlab="x-label",
ylab="y-label") # 添加标题
```

分步绘图 1

图 17: 在设备上设定绘图区域

分步绘图 2

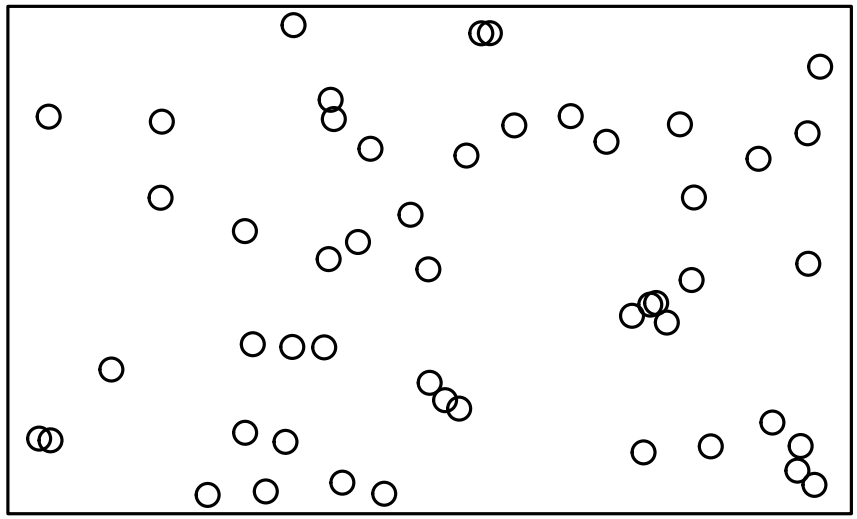


图 18: 绘制边框和添加点

分步绘图 3

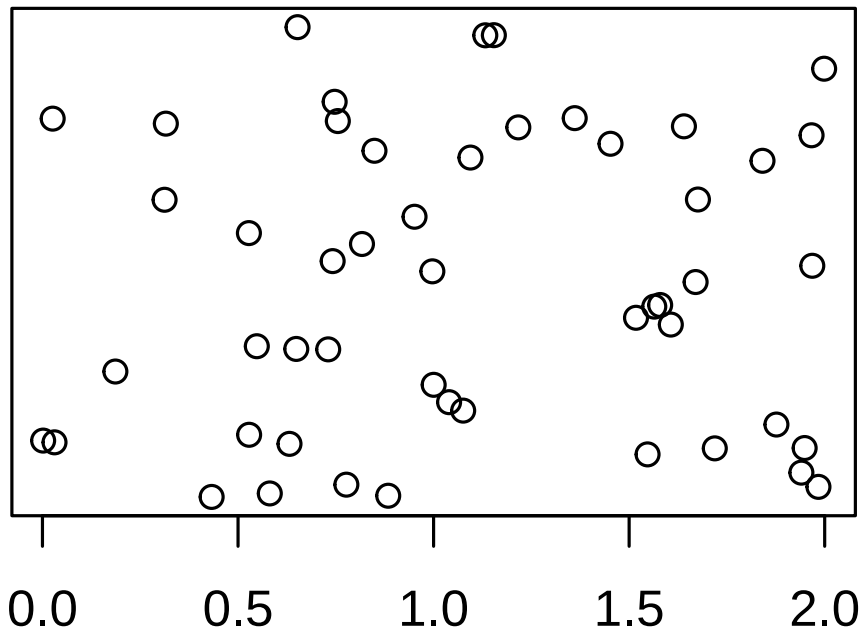


图 19: 添加 x 轴

分步绘图 4

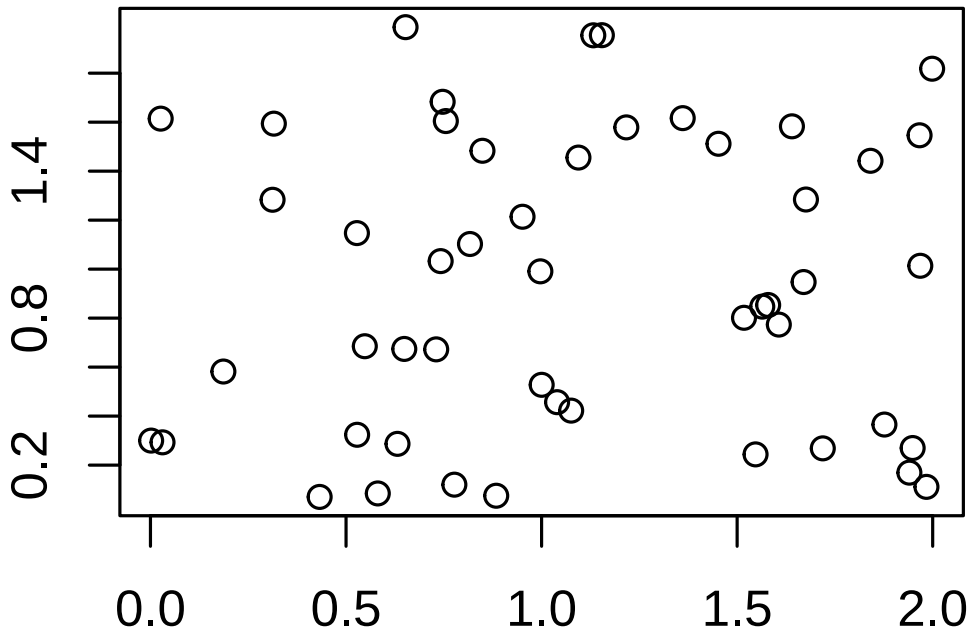


图 20: 添加 y 轴

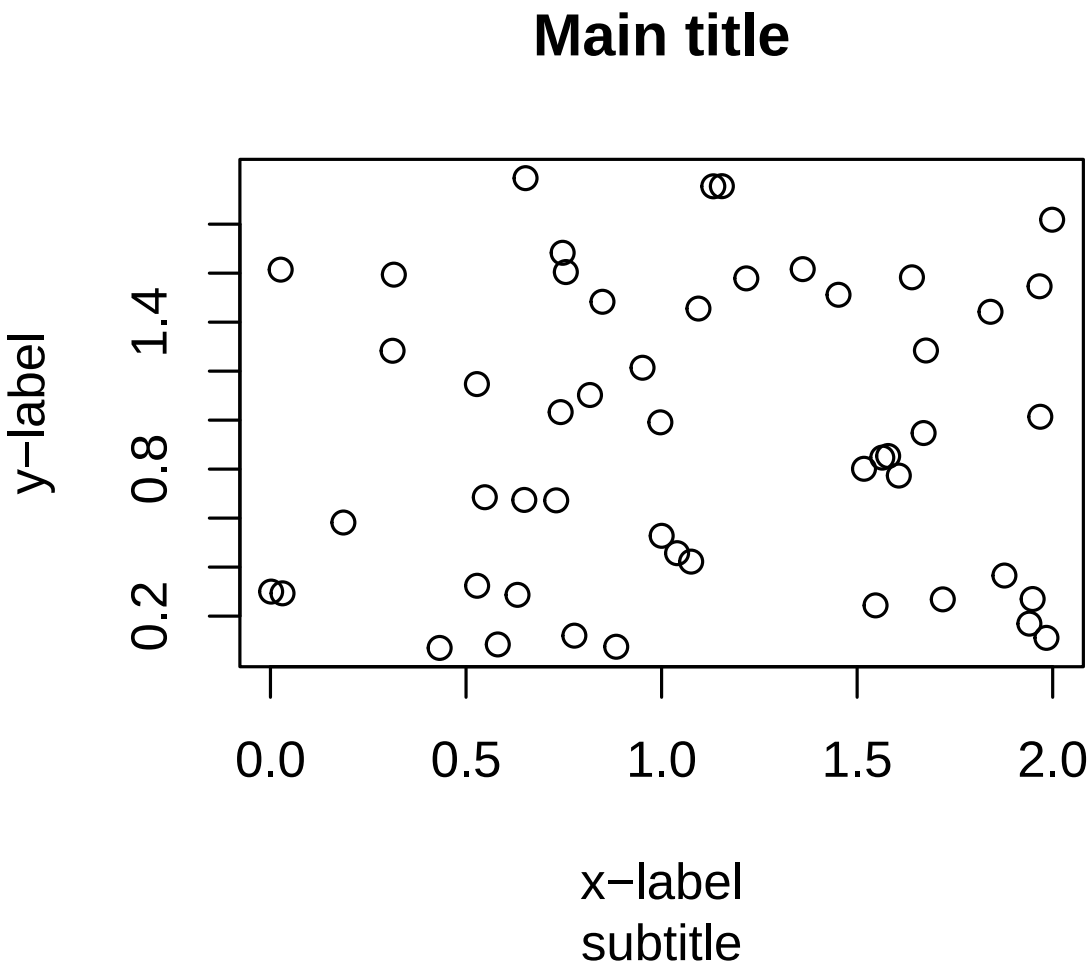


图 21: 添加标题和副标题

6

常用绘图函数的参数

plot 的常用参数

```
## Default S3 method:
plot(x, y = NULL, type = "p"
     xlim = NULL, ylim = NULL,
     log = "", main = NULL,
     sub = NULL, xlab = NULL,
     ylab = NULL, ...)
```

绘图函数的常用参数

```
type   # 图的类型：散点图，折线图？
xlim   # x 轴的范围
ylim   # y 轴的范围
log    # 是否对某一轴取对数
main   # 标题
sub    # 副标题
```

绘图函数的常用参数

```
xlab   # x 坐标
ylab   # y 坐标
axes   # 是否添加坐标刻度
col    # 点、线等的颜色
pch    # 点的类型
cex    # 字体的大小
lty    # 线的类型
lwd    # 线的宽度
...    # 其他可以传递给本函数的函数。
```

绘图函数的参数传递

- 如果函数 A 的参数有...，就表示这个函数没有写全参数。
- 此时，在函数 A 内部，也必然有一个函数 B，其参数为...。
- 这样，参数就得以传递，从而保证调用函数 A 时直接可以使用函数 B 的参数。

绘图函数的参数传递举例

```
plot222 <- function(x, ...){
    plot.default(x, ...)
}

plot222(cars, main = "Cars",
        xlab = "Speed",
        ylab = "Distance")
```

plot222 函数并没有定义 main, xlab, ylab 等参数，但是可以直接调用 plot.default 函数中的参数。

点的类型

怎样修改散点图中点的形状?

答: pch

```
plot(1:20, pch = 1:20, ylim = c(0, 21), main = "pch")
text(1:20, 2:21, labels = 1:20)
```

思考: 怎样改变点的大小和颜色?

点的类型

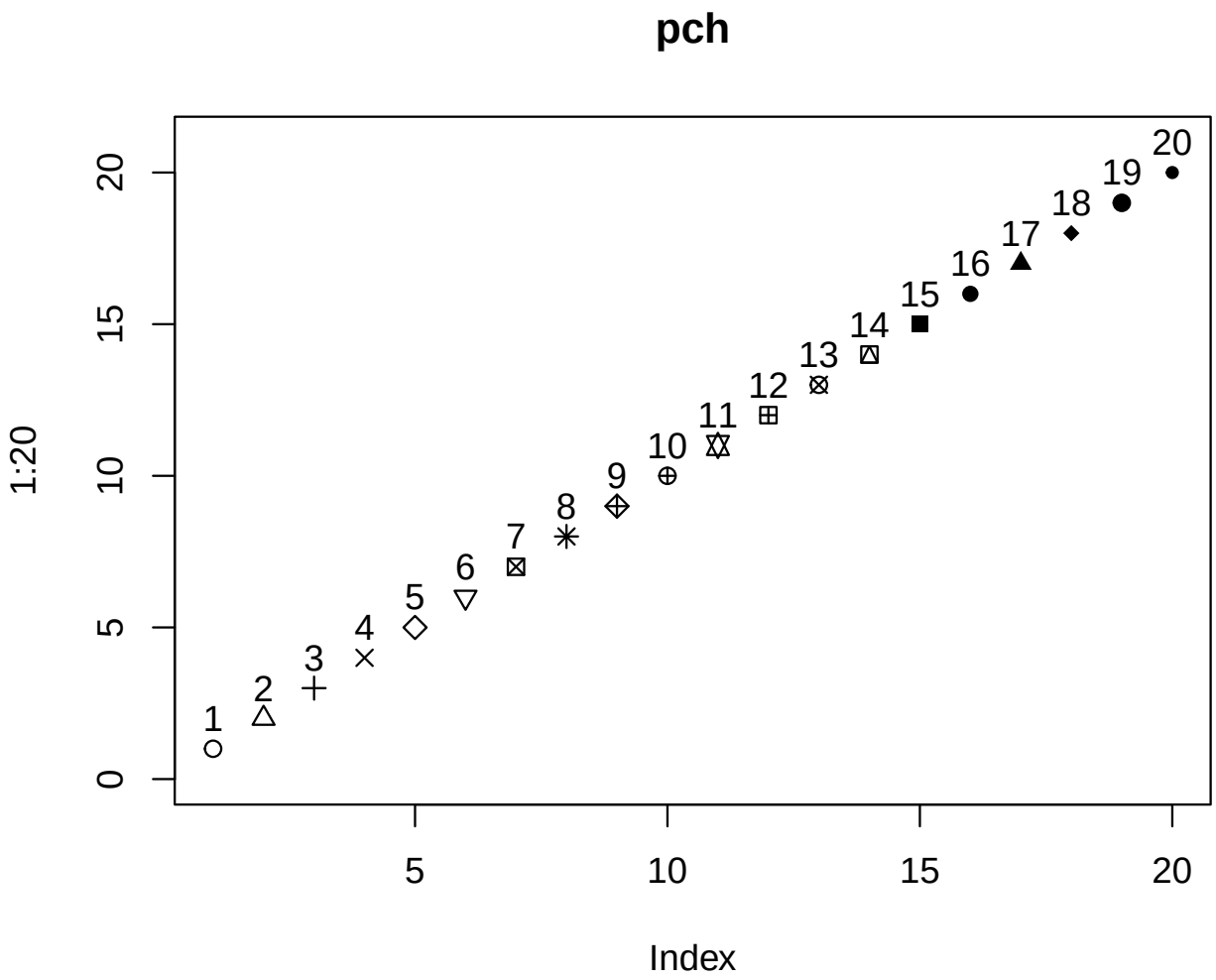


图 22: 点的类型

线的类型

```
plot(cars, main = "Stopping Distance
    \nversus Speed")
lines(stats::lowess(cars),
    col = "red", lwd = 2)
```

线的类型（实线）

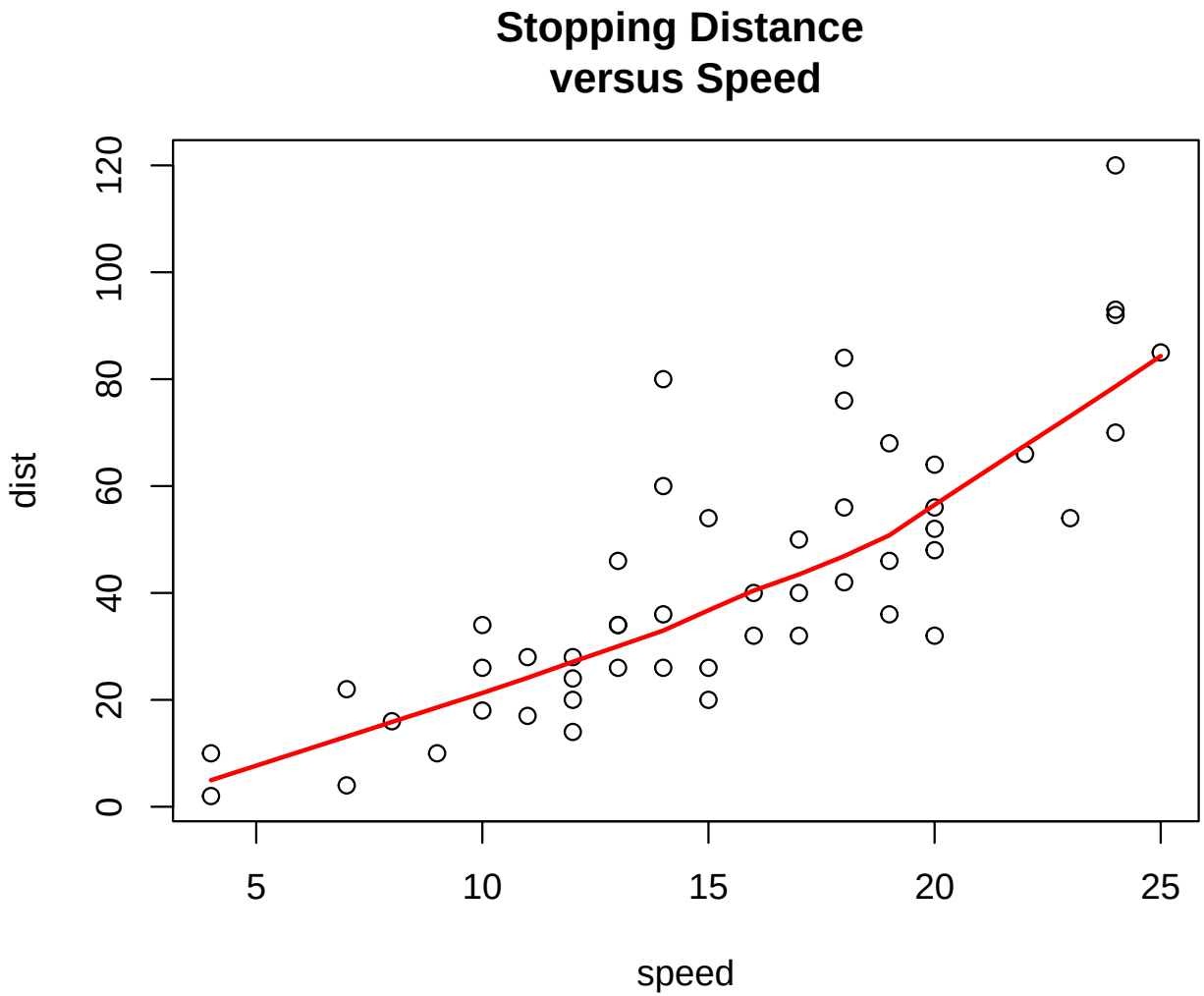


图 23: 添加趋势线

线的类型（虚线）

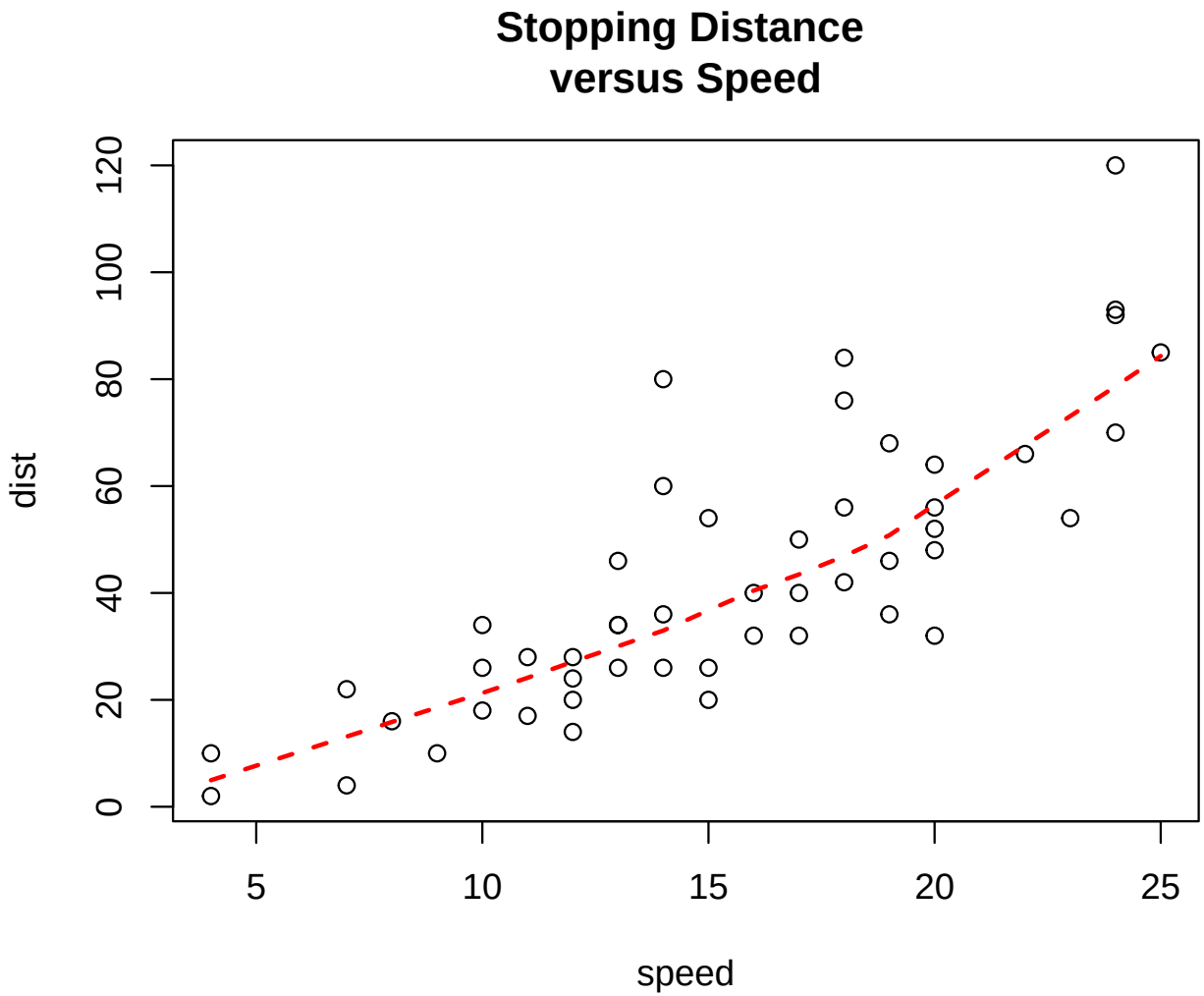


图 24: 趋势线为虚线

线的类型

```
plot(cars, main = "Stopping Distance versus Speed")
lines(stats::lowess(cars), col = "red", lty = 2)
```

- 0 "blank" : 空白
- 1 "solid" : 实线
- 2 "dashed": 由短线段构成的虚线
- 3 "dotted": 有点构成的虚线
- 4 "dotdash": 点和线构成的虚线
- 5 "longdash": 长线段构成的虚线
- 6 "twodash": 双断线构成的虚线

颜色

颜色，通常是用 `col` 参数来指定。

有三种指定方法：

- 用颜色名：`"red","green","blue","grey","yellow"`。输入 `colors()` 查看所有颜色名
- 用 RGB 值，参考 `col2rgb("green")`。
- 用数字（颜色的索引），默认 1:8，参考 `palette()`

颜色序列 (调色板)

graphis 包提供了一系列生成不同色系过渡颜色的函数。

```
hcl.colors()
rainbow()
heat.colors()
terrain.colors()
topo.colors()
cm.colors()
```

内置的调色板 (hcl.colors)

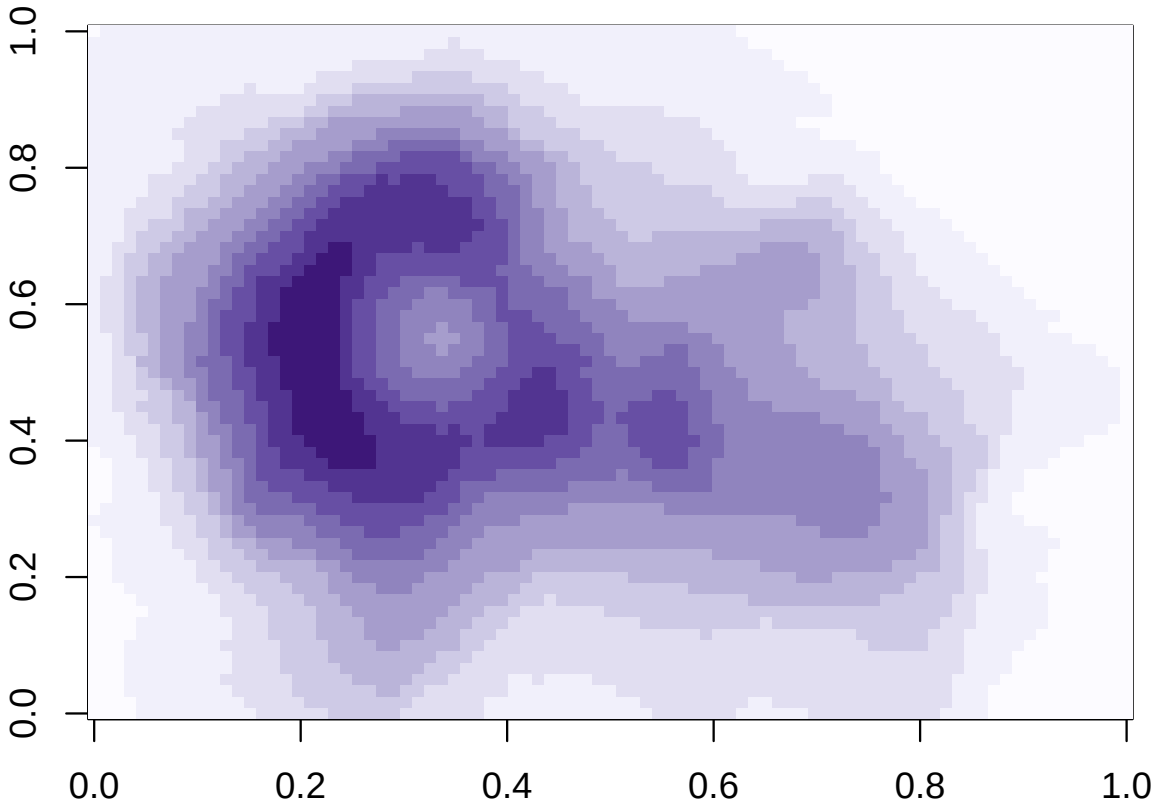


图 25: hcl.colors

内置的调色板 (rainbow)

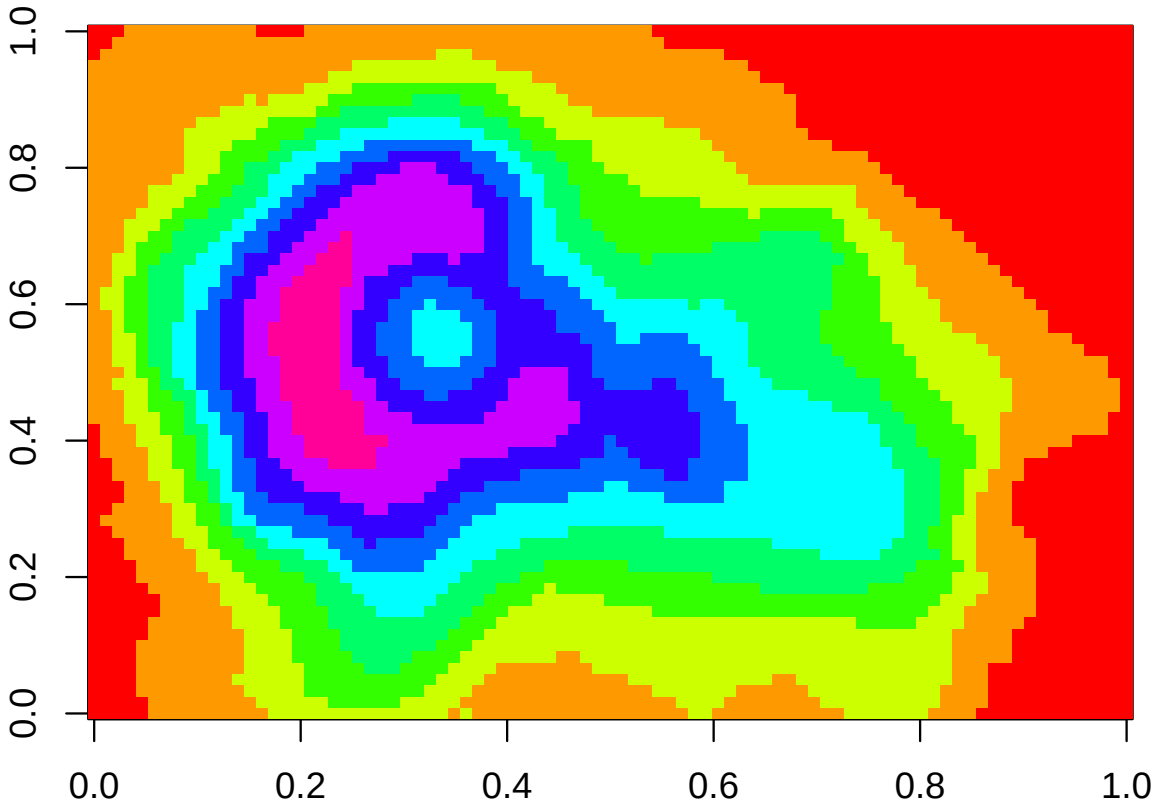


图 26: rainbow

内置的调色板 (heat.colors)

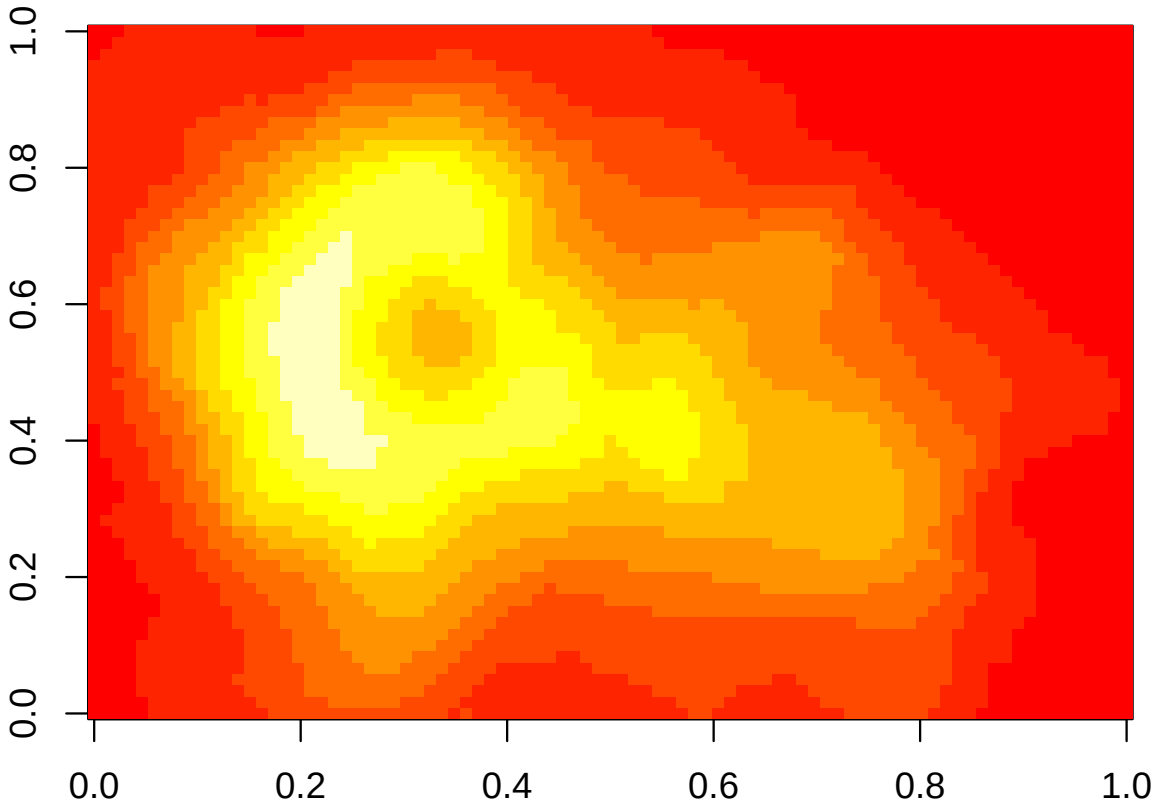


图 27: heat.colors

内置的调色板 (terrain.colors)

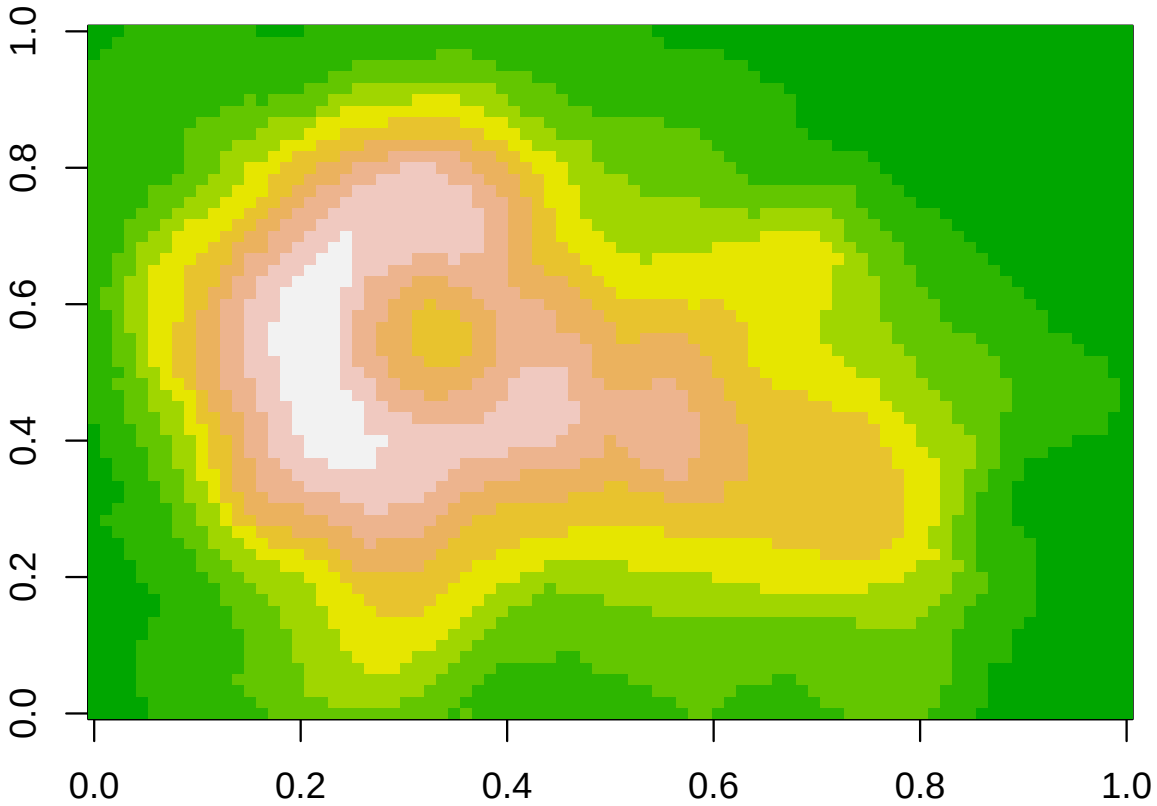


图 28: terrain.colors

内置的调色板 (topo.colors)

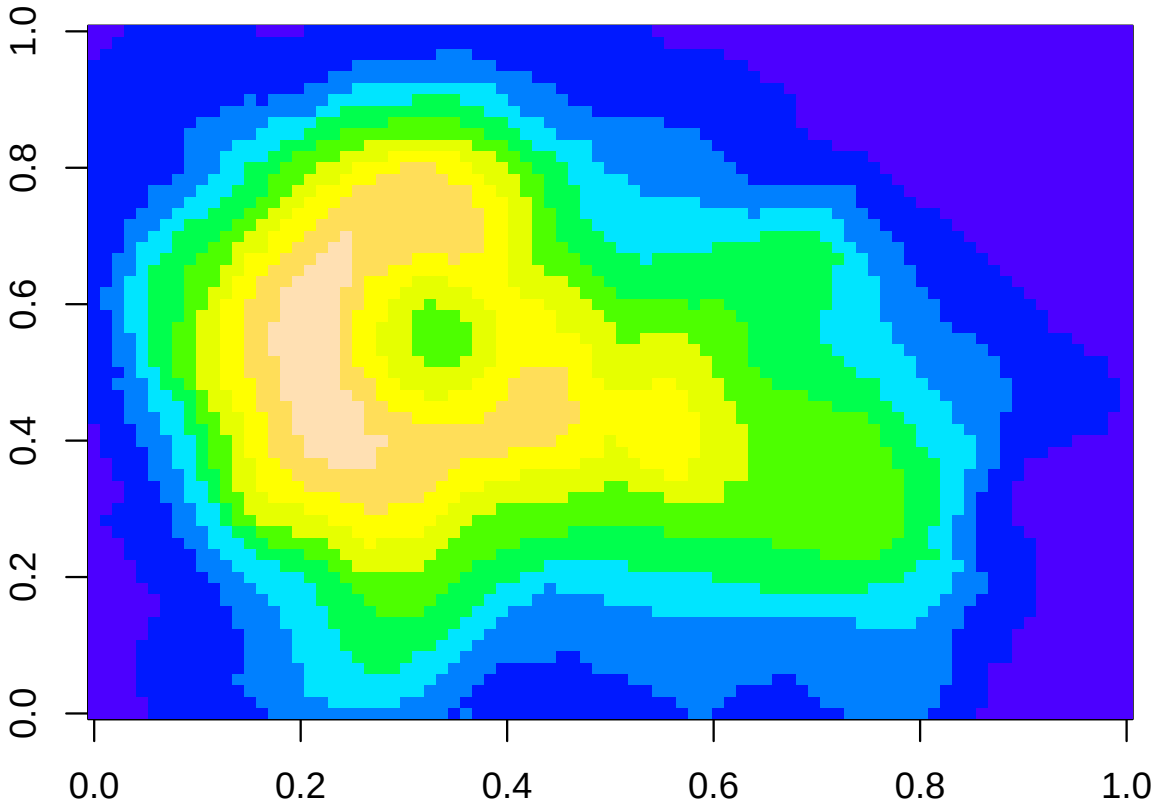


图 29: topo.colors

内置的调色板 (cm.colors)

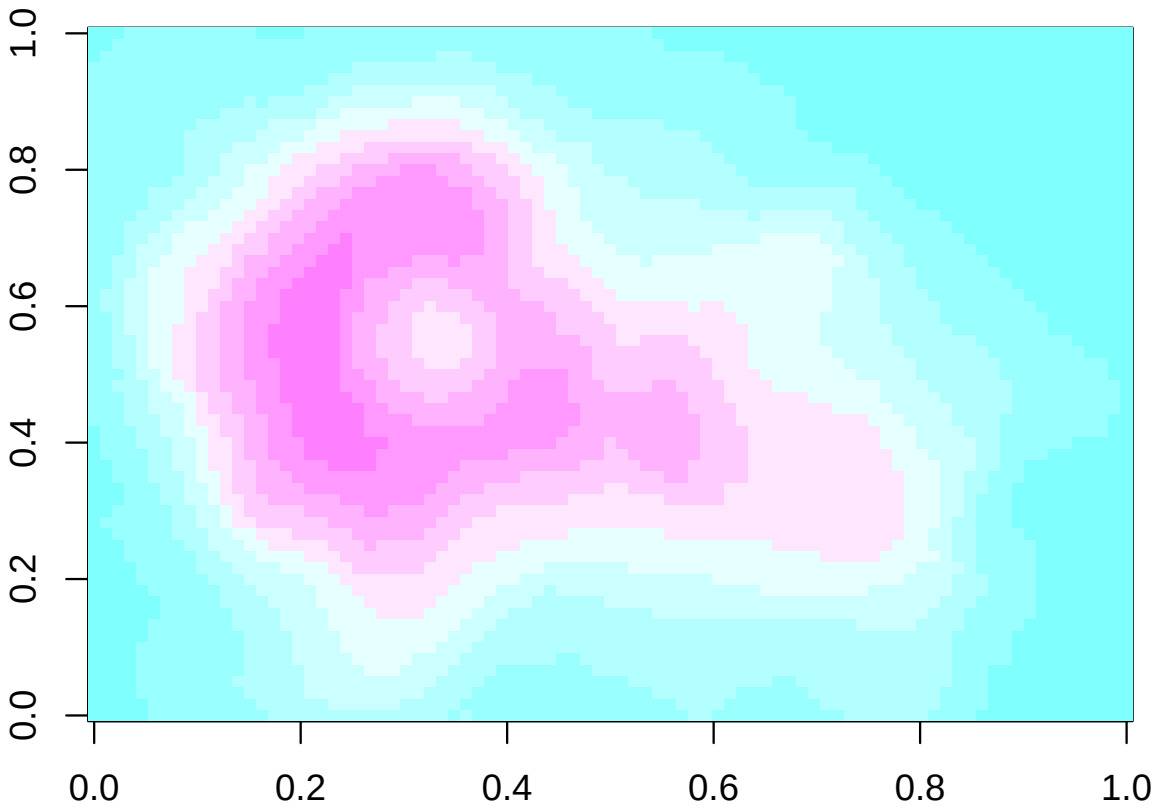


图 30: cm.colors

颜色的编号

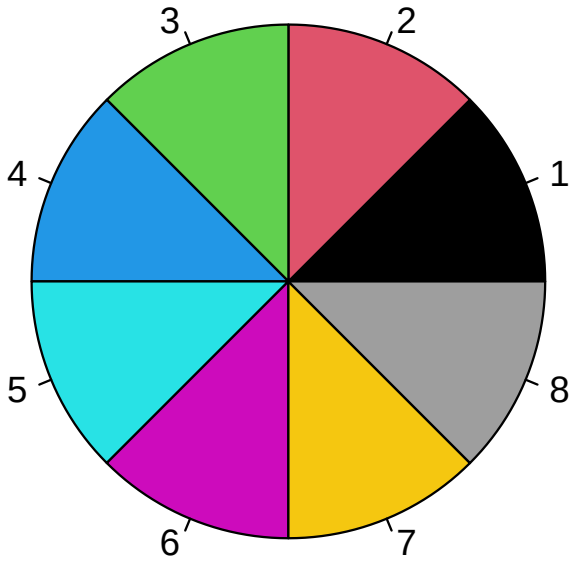
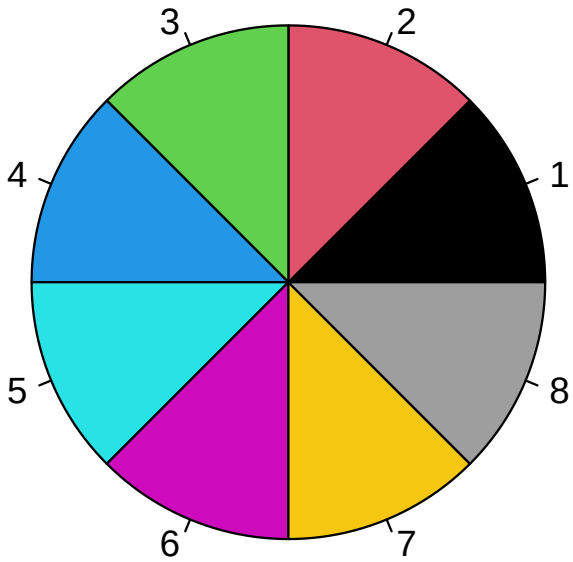


图 31: 颜色的编号

颜色的编号

```
pie(rep(1, 8), col = 1:8)
```



颜色序列 (RColorBrewer 调色板)

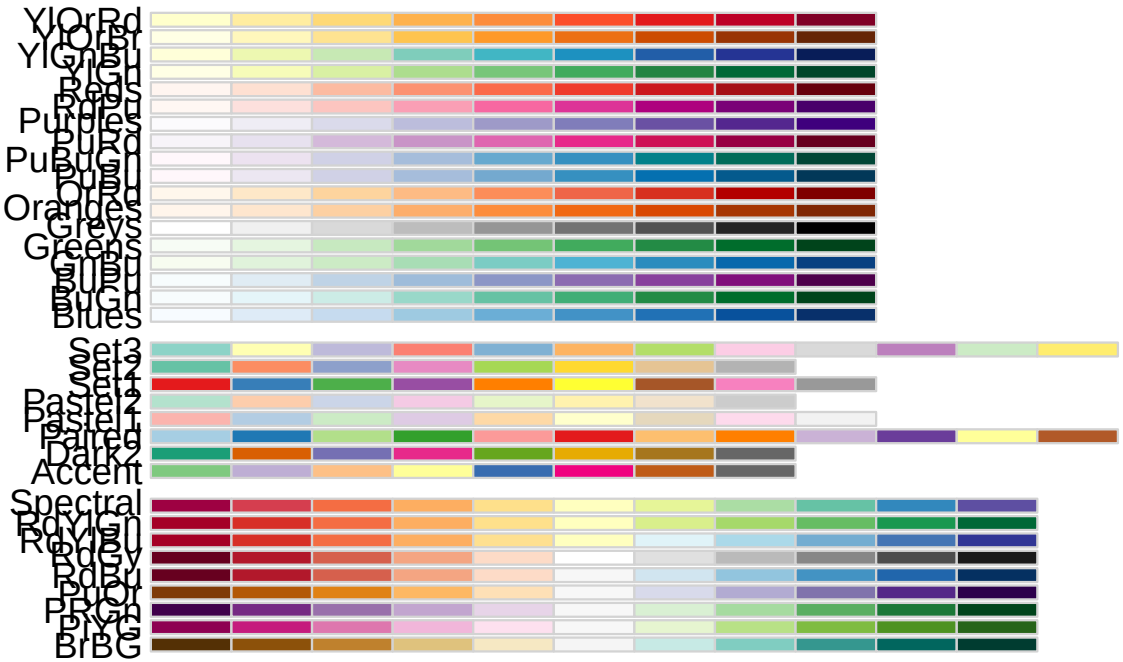


图 33: 调色板

更多颜色序列

有若干程序包提供了更多颜色序列，例如：

viridis、wesanderson、ggsci

参考：`tmaptools::palette_explorer()`

R color cheatsheet：

https://github.com/EmilHvitfeldt/r-color-palettes

如果提供的颜色数量不够，会怎样?

```
plot(cars, main = "Stopping Distance versus Speed", pch = 1,
     col = c("blue", "red", "green"))
```

答：短的序列将会被重复若干次

如果提供的颜色数量不够……

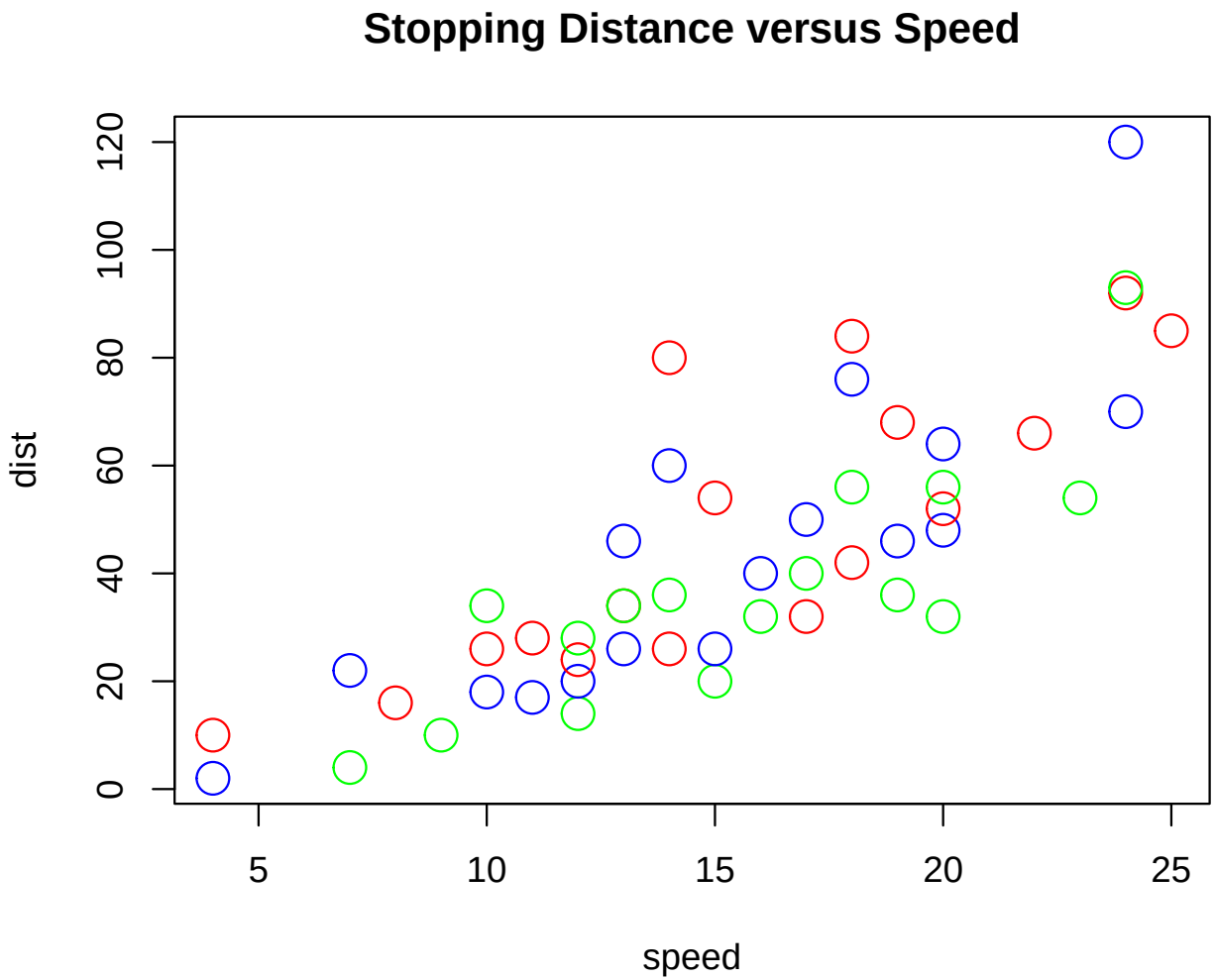


图 34: 如果提供的颜色不够，现有颜色将会被重复

字体 family

- 默认是 `Sans Serif`（类似 Arial）
- 可将 `family` 改为 `Serif`（类似 Times New Roman），可在 `plot`、`par` 中更改
- 在图片中显示中文，请参考 `showtext` 包

7

图形的进一步调整

添加图例

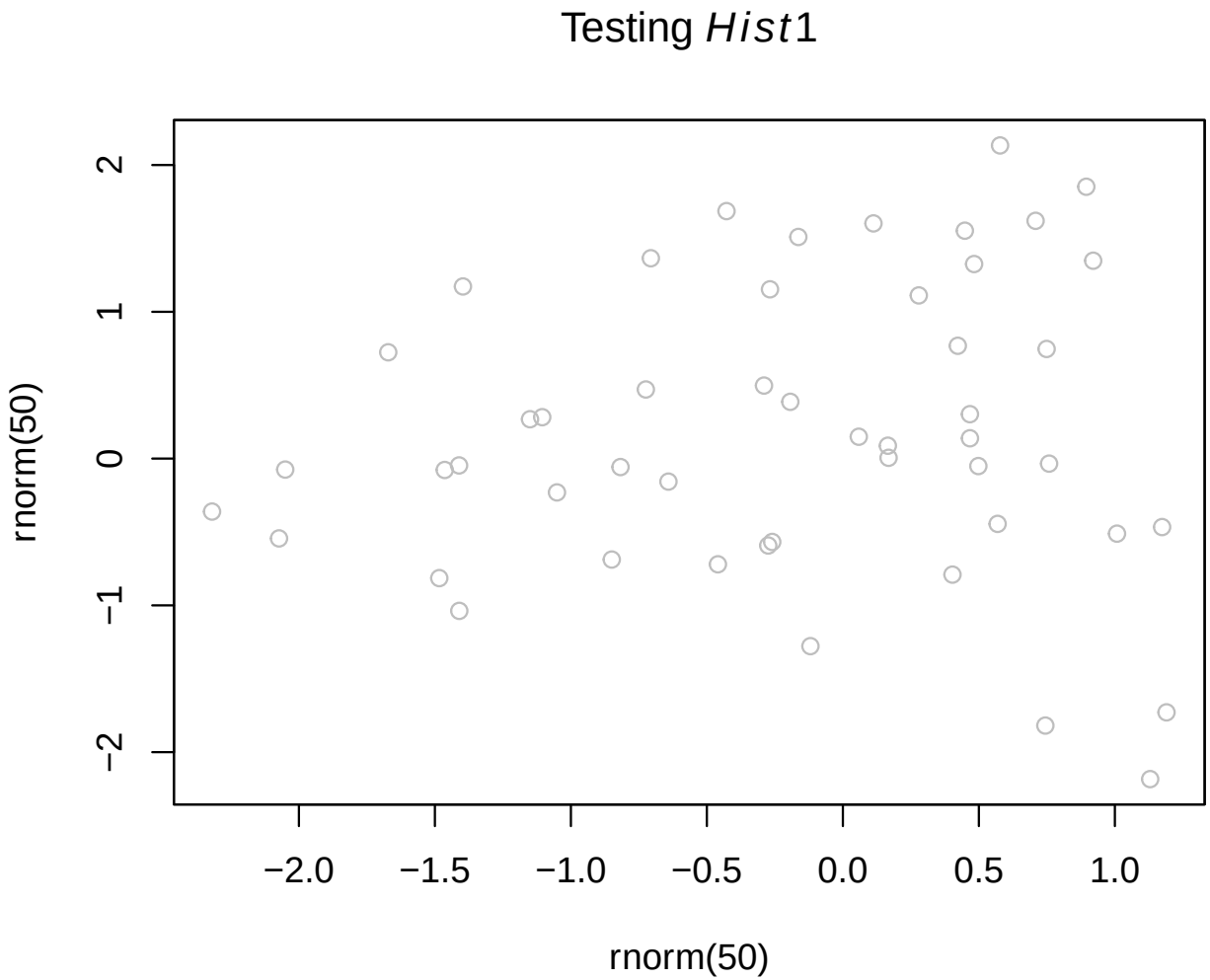


图 35: 添加图例

添加图例

```
## 标题部分斜体
plot(x = rnorm(50), y = rnorm(50), main = expression(paste(
  "Testing ",
  italic(Hist1))), col = "gray")
# 添加边框
box()

## 图中添加图例
legend.label <- c("Type I", "Type II", "Type III")
legend(1.5, 30, legend = legend.label,
    pch = c(19, 20, 21), col = c(1, 2, 3))
```

添加文字标注

需要用到若干低等级（底层）绘图函数

```
text()
mtext()
expression()
```

用 expression() 添加公式

```
hist(rnorm(200),
    main = expression(paste(plain(sin) * phi,
    " and ",
    plain(cos) * phi)),
    xlab = expression(paste("Latitude ",
    degree)), col = "gray")

text(-2, 30, expression(bar(x) ==
    sum(frac(x[i], n), i==1, n)), cex = 1.2)
```

添加标注、公式和特殊字符等

一般是 text() 以及 expression(), substitute() 等

```
## 标题中显示度等特殊字符，标题中显示希腊字母
plot(x = rnorm(50), y = rnorm(50), main = expression(paste("Te
## 图中添加公式
text(0, 0, expression(bar(x) ==
     sum(frac(x[i], n), i == 1, n)),
     cex = 1.2)
```

添加标注、公式和特殊字符等

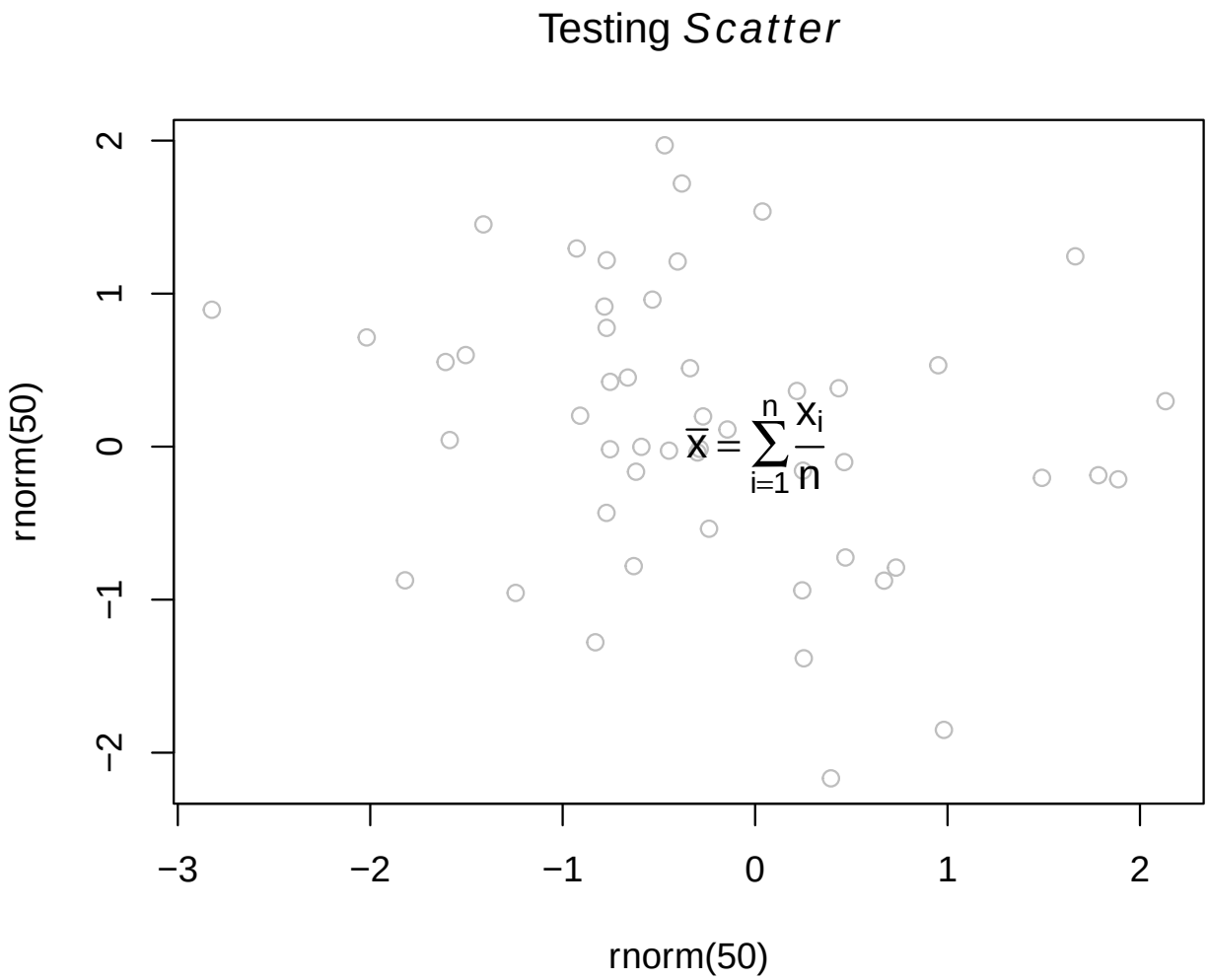


图 36: 添加标注、公式和特殊字符等

par 调整图的页面边距

默认的图边距

```
par("mar")
```

```
## [1] 5.1 4.1 4.1 2.1
```

修改图边距

顺序：**下：2、左：3、上：1、右：4 （行）**

```
par(mar = c(2, 3, 1, 4))
```

页面边距

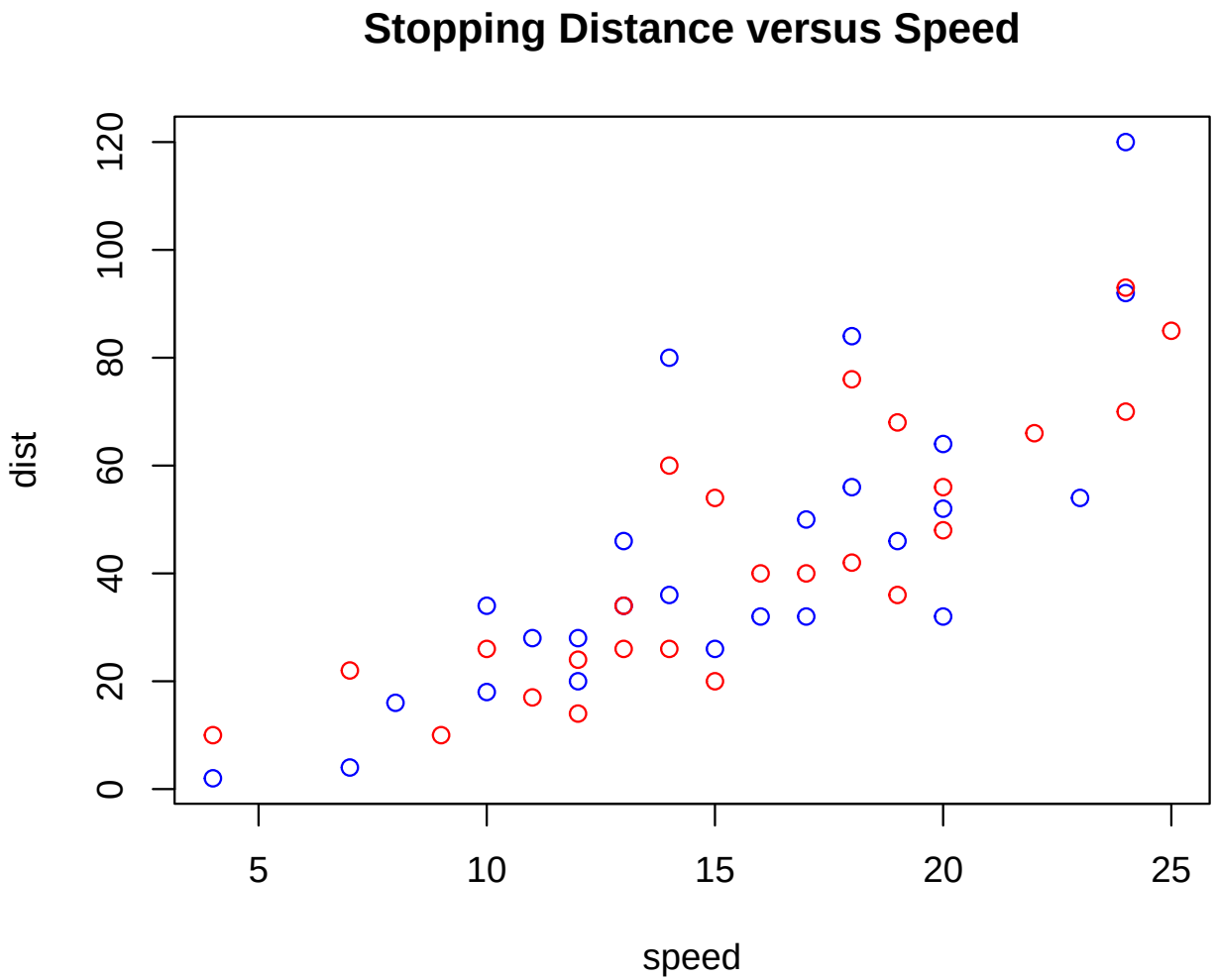


图 37: 页面边距

一行两列

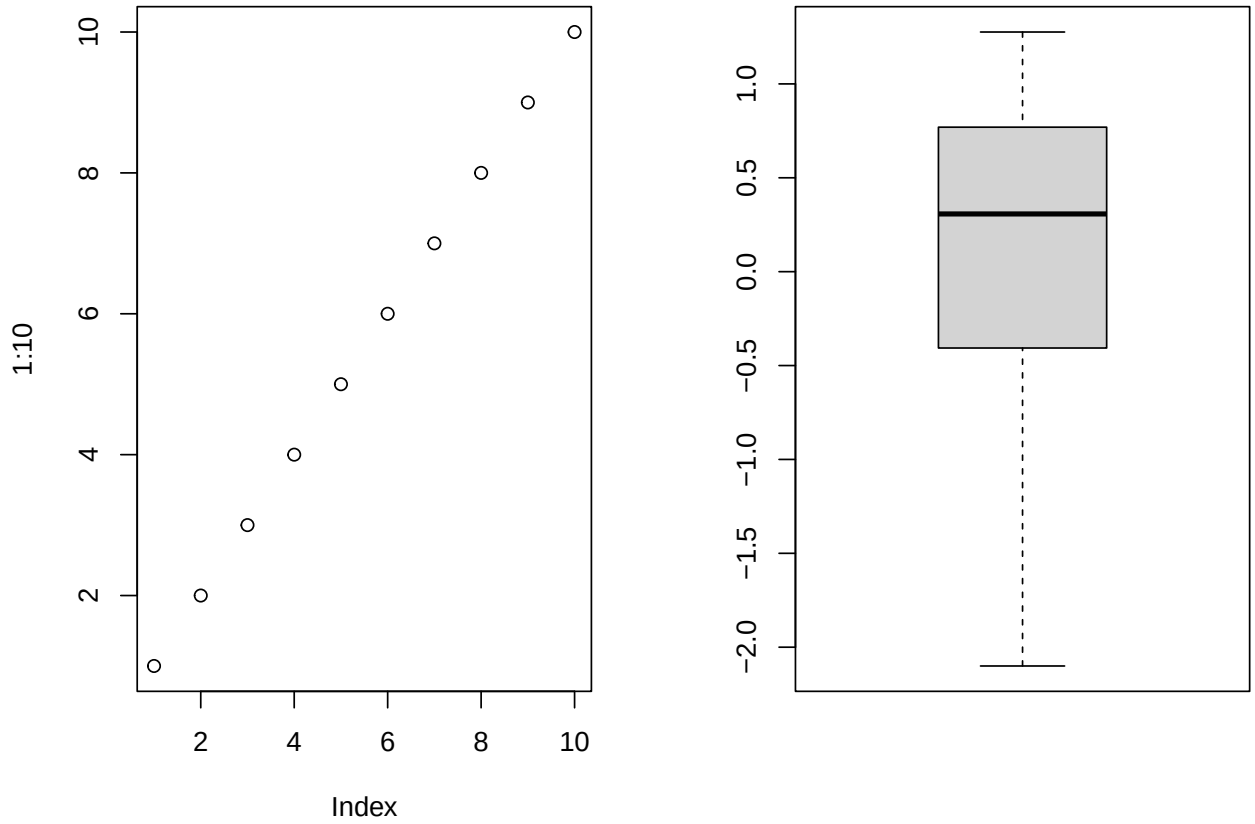


图 38: 一行两列

一行两列代码

```
par(mfrow = c(1,2))
# layout.show(2)
plot(1:10)
boxplot(rnorm(30))
```

两行、三列

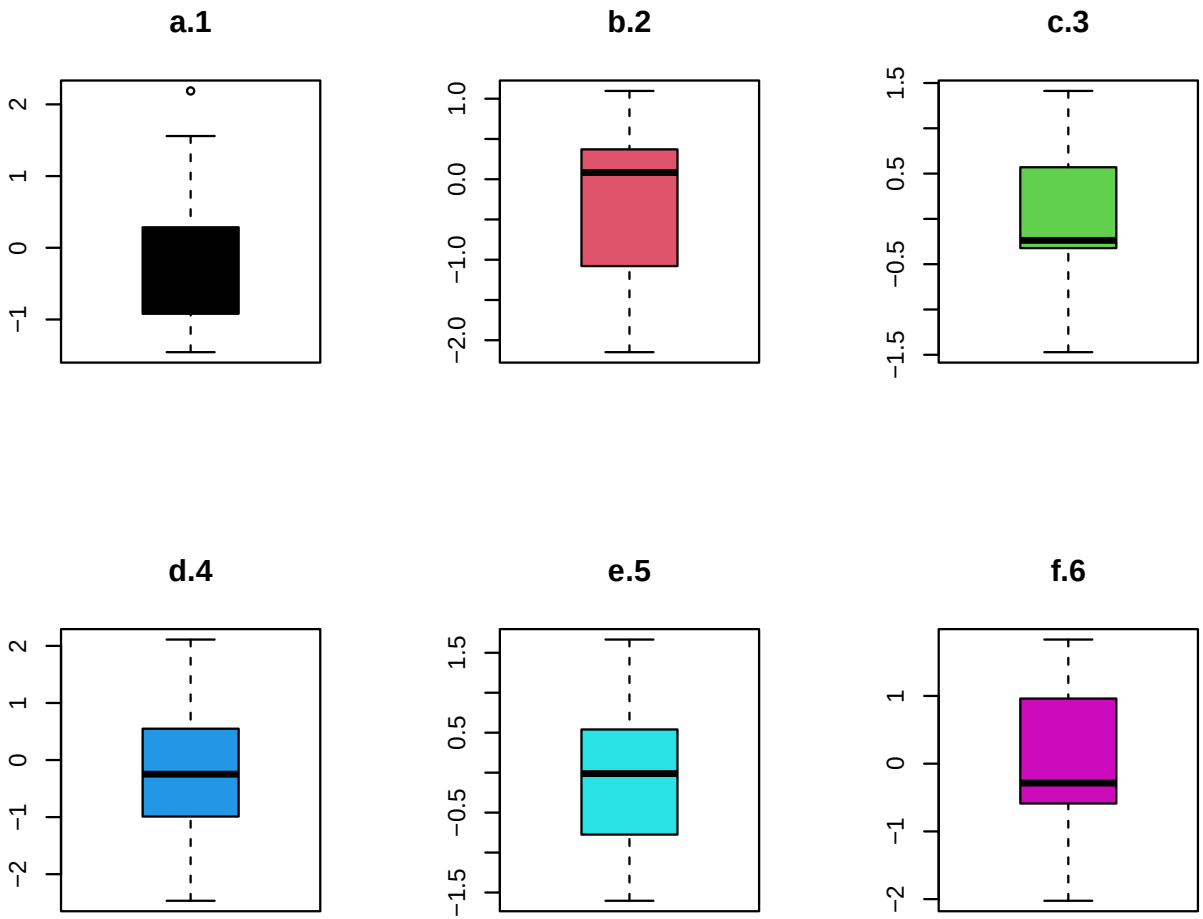


图 39: 一页多图

两行、三列

```
par(mfrow = c(2, 3))
# layout.show(6)

for (i in 1:6) {
  boxplot(rnorm(20), col = i,
  main = paste0(letters[i], ".", i))
}
```

调整各图所占比例

图 40: 绘图区域的分配

调整各图所占比例

```
layout(matrix(c(1,1,0,2), 2, 2, byrow = TRUE))
layout.show(2)
```

用 par() 设定绘图设备的其他参数

```
bg   # 背景的颜色
cex  # 字体大小
col  # 颜色
font # 字体
las  # 文字相对于坐标轴的方向
lty  # 默认的线类型
(0=blank, 1=solid (default),
 2=dashed, 3=dotted,
 4=dotdash, 5=longdash,
 6=twodash)
lwd  # 默认线粗细
```

par 设定图形的默认环境

```
mar   # 图四边的边距
mfcol, mfrow # 一页多图，设定列数、行数
pch   # 默认的点类型
srt   # 旋转
xaxs # x 轴坐标是否从 0 开始
yaxs # x 轴坐标是否从 0 开始
```

8

图像保存

图像保存

- 图像保存的函数
  1. 开启绘图设备
  2. 绘图 `plot...`
  3. 用 dev.off() 关闭绘图设备
- 开启绘图设备的函数
  - tiff: `tiff()`
  - jpeg: `jpg()`
  - png: `png()`
  - pdf: `pdf()`
  - svg: `svg()`

图片的分辨率

- 默认分辨率: 72dpi
- 栅格图: 发表的要求，一般 300dpi，即每英寸像素的密度为 300
- 矢量图：无需考虑分辨率

举例：保存为 pdf

```
pdf("lines_examples2.pdf", width = 6, height = 5)
plot(cars, main = "Stopping Distance vs. Speed")
lines(stats::lowess(cars), col = "red", lty = 2)
dev.off()
```

保存为 tiff

- 保存为 tiff 的时候，默认单位是像素，width 和 height 需要根据图像尺寸和分辨率换算
- 默认分辨率（DPI）为 72，论文投稿要求一般为 300-600dpi
- 默认不压缩，但是应尽量选择 `lzw` 压缩

保存为 tiff（600dpi）

```
tiff(filename = "line_types.tiff", width = 4800,
     height = 4800, units = "px", pointsize = 12,
     compression = "lzw", bg = "white", res = 600)
plot(cars, main = "Stopping Distance versus Speed")
lines(stats::lowess(cars), col = "red", lty = 2)
dev.off()
```

怎样修改图形?

1. 修改代码，运行代码，自动保存图形
2. 导出到文件，用其他软件编辑

- 导出为栅格图（`tiff()`、`jpeg()`、`png()`），再用 Adobe Photoshop、paint.net、GIMP 等编辑
- 导出为矢量图（`pdf`、`svg`），再用 Adobe Illustrator、Inkscape、Krita 等编辑
- 导出为 ppt 文件（`library(export)`），再用 MS Powerpoint 编辑

9

怎样设计出让人赏心悦目的图形

选择合适的图形

- Midway, S. R. (2020). Principles of effective data visualization. Patterns, 1(9), 100141.(https://www.sciencedirect.com/science/article/pii/S2666389920301896)

绘图先要明确主题

一个图，首先要有一个主题，也就是向读者传递的信息。画图的人先要想清楚希望读者看到图以后明白什么。

图形的种类以及图形内所有的信息，包括点、线的类型、颜色、大小、横纵坐标的文字、标注、标题等所有信息，都是为了主题服务的，为的就是把主题清晰地表达出来。

参考：
https://writingcenter.unc.edu/tips-and-tools/figures-and-charts/

绘图的三个原则

简洁 simplicity：数据是复杂的，图形是将复杂中的信息抽象出来，提取出最重要的信息，展现给读者。最好是”一目了然”

清晰 clarity：每一幅图，就是一个独立的作品，只看图不看正文，就应该将图形的意思表达清楚。要根据需要做适当说明，以便读者能理解。

准确 accuracy：图中不能有任何错误，更不能为了好看而筛选数据。

参考：
https://writingcenter.unc.edu/tips-and-tools/figures-and-charts/

怎样设计出让人赏心悦目的图形?

科学绘图，其实是在制作艺术品。

设计的原则以及常见要避开的问题：

- Franconeri, S. L., Padilla, L. M., Shah, P., Zacks, J. M., & Hullman, J. (2021). The science of visual data communication: What works. *Psychological Science in the Public Interest*, 22(3), 110-161.
- Rougier, N. P., Droettboom, M., & Bourne, P. E. (2014). Ten simple rules for better figures. *PLoS computational biology*, 10(9), e1003833.

哪些图好？哪些图差？

Karl Broman [2013] Top ten worst graphs in the scientific literature

https://www.biostat.wisc.edu/~kbroman/presentations/topten.pdf

怎样画很差的图

- Adapted from Wainer H. How to Display Data Badly. *The American Statistician* 1984; 38: 137-147.

参考

- Wilke, C. O. (2019). Fundamentals of data visualization: a primer on making informative and compelling figures. O'Reilly Media. https://clauswilke.com/dataviz/
- Wickham, H., & Grolemund, G. (2016). R for data science: import, tidy, transform, visualize, and model data. " O'Reilly Media, Inc.".
- https://r4ds.had.co.nz/data-visualisation.html
- https://github.com/hadley/ggplot2-book

下一节课 ggplot2

1. graphics 还是 ggplot2?
2. ggplot2 的基本用法

内容回顾

1. 绘图是为了更好和读者交流
2. 绘图的一些基本概念
3. 图形的基本结构
4. 图的类型根据数据类型而定
5. R 常用绘图函数
6. 常用绘图函数的参数
7. 图形的进一步调整
8. 图像保存
9. 怎样设计出让人赏心悦目的图形