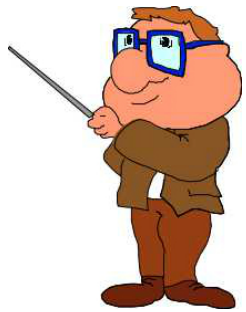
R 语言数据分析
Lecture 4 R 统计做图

刘 寅

统计与数学学院

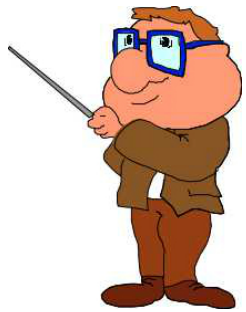
outline



1. 简介
1.1 示例

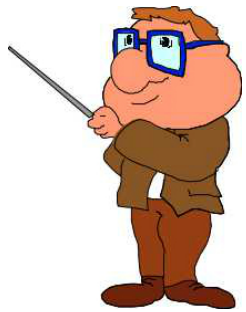
- 尝试以下代码:

```
> demo(graphics) # graphics是基础绘图包
> demo(persp) # persp绘制透视图
> library(lattice) # lattice是基于grid绘图系统的两个主流扩展包之一
> demo(lattice)
```



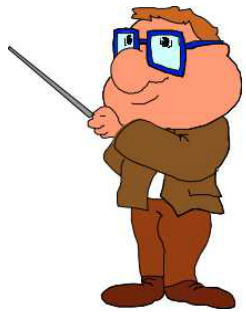
1.2 命令种类

- 高级绘图命令



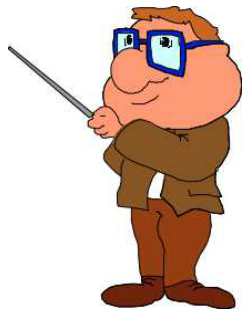
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等



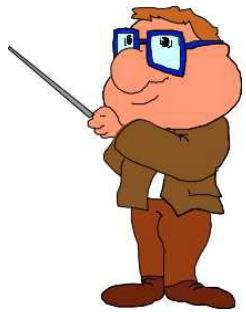
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令



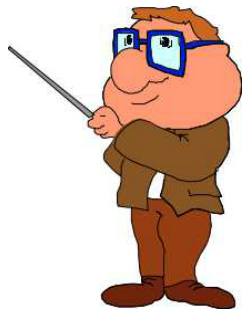
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签



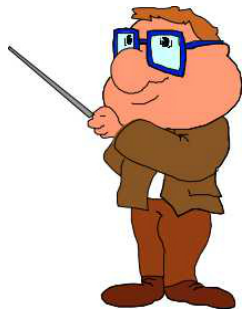
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签
- 图形参数



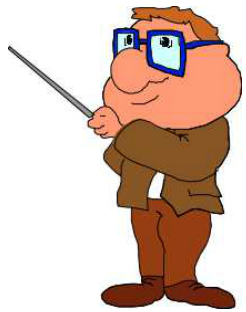
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签
- 图形参数
  - 图形参数可以被修改从而定制图形环境



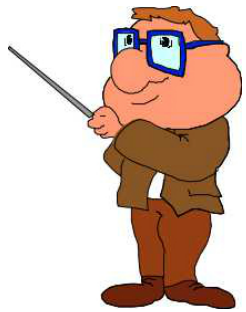
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签
- 图形参数
  - 图形参数可以被修改从而定制图形环境
- 网格作图命令



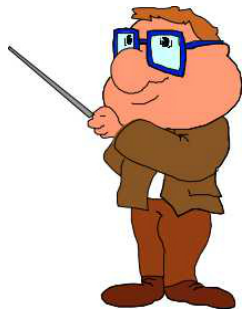
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签
- 图形参数
  - 图形参数可以被修改从而定制图形环境
- 网格作图命令
  - 使用**grid**、**lattice** 和**ggplot2**进行面板作图



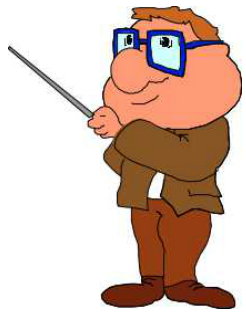
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签
- 图形参数
  - 图形参数可以被修改从而定制图形环境
- 网格作图命令
  - 使用**grid**、**lattice** 和**ggplot2**进行面板作图
- 图形设备管理命令



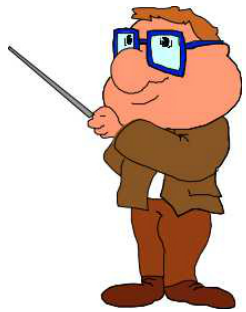
1.2 命令种类

- 高级绘图命令
  - 在图形设备上产生一个新的图区，它可能包括坐标轴，标签，标题等等
- 低级绘图命令
  - 在一个已经存在的图上加上更多的图形元素，如额外的点，线和标签
- 图形参数
  - 图形参数可以被修改从而定制图形环境
- 网格作图命令
  - 使用**grid**、**lattice** 和**ggplot2**进行面板作图
- 图形设备管理命令
  - 通过设备管理命令来保存R图形



2. 高级绘图函数

- plot(x)：以x 的元素值为纵坐标、以序号为横坐标绘图
- plot(x,y)：以x 的元素值为横坐标、以y的元素值为纵坐标绘图
- pie(x)：饼图
- boxplot(x)：盒形图
- hist(x)：x的频率直方图
- density(x)：x的密度曲线图
- barplot(x)：x的值的条形图
- pairs(x)：如果x是矩阵或数据框，作x的各列之间的二元图
- coplot(x~y|z)：关于z的每个数值(或数值区间)绘制x与y的二元图
- qqnorm(x)：正态分位数-分位数图
- image(x,y,z)：三元图
- contour(x,y,z)：三维等高线图
- persp(x,y,z)：三维表面曲线图
- heatmap(x)：热图



2.1 plot函数

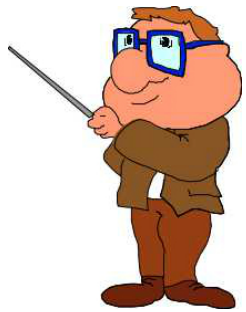
- 是R里面最常用的一个图形函数
- 是一个泛型函数：产生的图形依赖于第一个参数的类型或者类
- 使用方法
  - plot(x): 以x 的元素值为纵坐标、以序号为横坐标绘图
  - plot(x, y): x(在x-轴上)与y(在y-轴上)的二元作图
  - plot($y \sim x$): x(在x-轴上)与y(在y-轴上)的二元作图
  - plot(matrix): 矩阵散点图

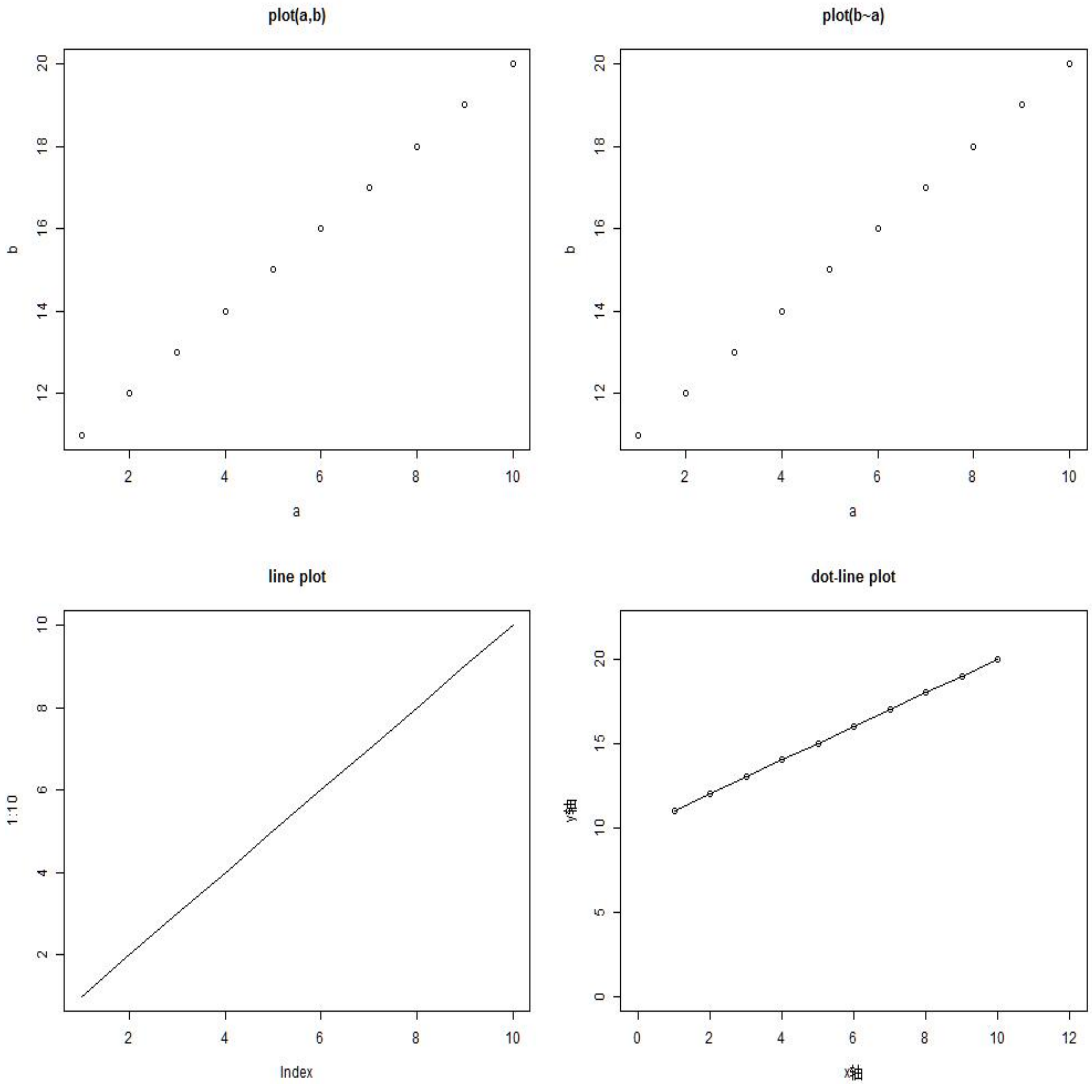
plot函数选项

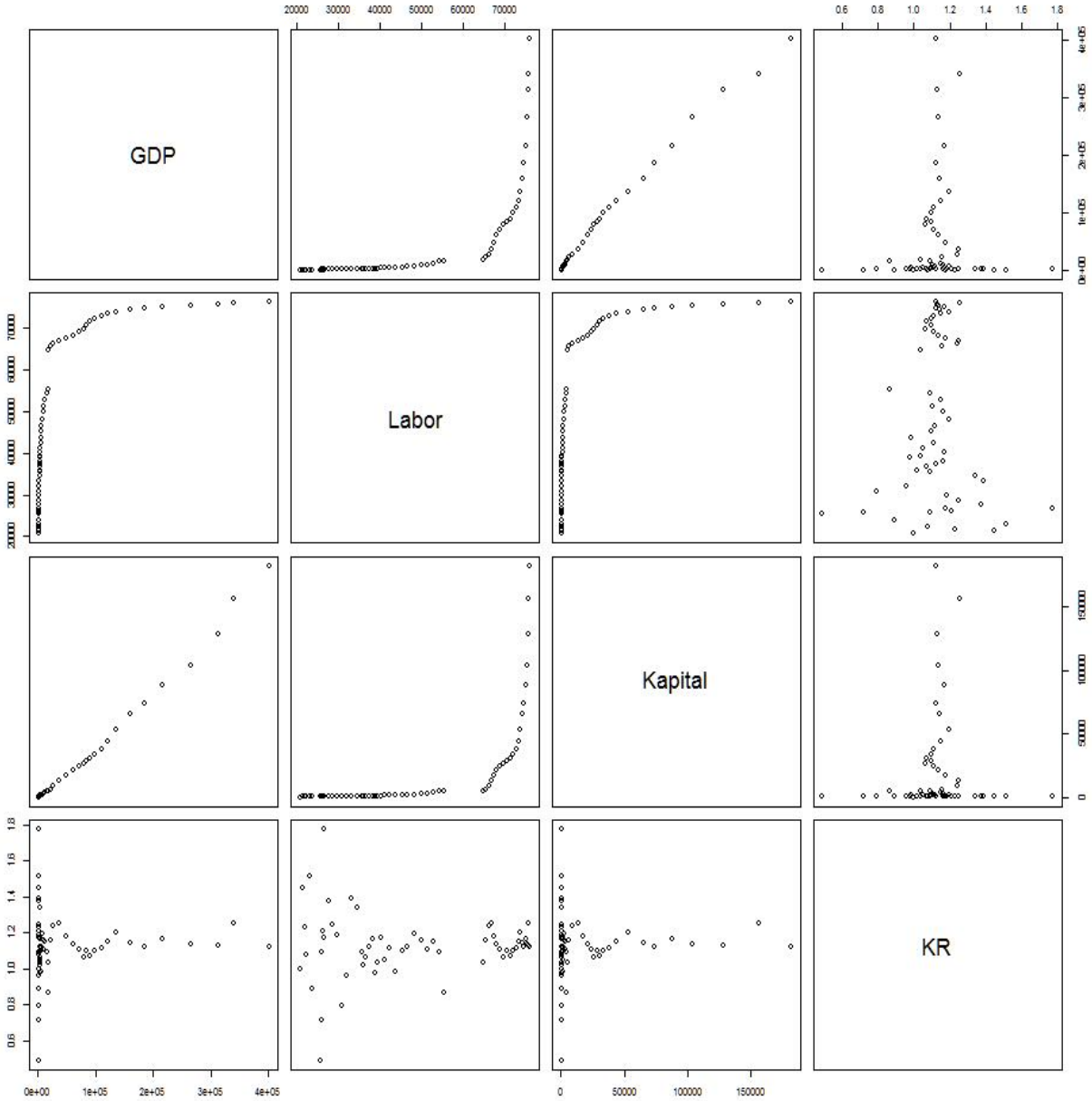
参数	作用
axes=T	如果是FALSE ，不绘制轴与边框
type= "p"	指定图形的类型，"p": 点，"l": 线，
	"b": 点连线，"o": 同上，但是线在点上
xlim=, ylim=	指定轴的上下限，例如xlim=c(1, 10)
xlab=, ylab=	坐标轴的标签，必须是字符型值
main=, sub=	指定主标题和副标题，必须是字符型值

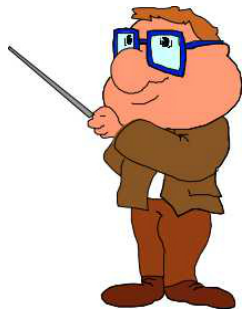
plot函数例子

```
>  plot(1:10)
>  a <- 1:10
>  b <- 11:20
>  win.graph(width=5, height=5, pointsize=8)
>  par(mfrow=c(2,2))
>  plot(a,b, main="plot(a,b)")
>  plot(b~a, main="plot(b~a)")
>  plot(1:10, type="l", main="line plot")
>  plot(b~a, type="o", xlim=c(0,12), ylim=c(0,22),
+  xlab="x轴", ylab="y 轴", main="dot-line plot")
>  GDPdata <- read.csv(file="...\\lecture 4\\GDP.csv")
>  str(GDPdata) # 查看GDPdata 的结构
'data.frame':   59 obs. of  10 variables:
 $ Year       : int  1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 ...
 $ GDP        : num  679 824 859 911 1029 ...
 $ GDPRealRate: num  NA 1.16 1.04 1.07 1.15 ...
 $ Labor      : num  20729 21364 21832 22328 23018 ...
 $ Kapital    : num  80.7 115.3 140.9 145.5 219.6 ...
 $ KR         : num  1 1.45 1.23 1.08 1.51 ...
 $ Technology : num  NA 0.56 1.22 2.13 5.23 ...
 $ Energy     : num  NA 5411 6234 6968 8800 ...
 $ HR         : num  11 19.2 20 19 26.5 ...
 $ CPI        : num  1.027 1.051 1.014 1.003 0.999 ...
>  win.graph(width=5, height=5, pointsize=8)
>  plot(GDPdata[, c("GDP", "Labor", "Kapital", "KR")])
```




plot(a,b)
b
a
plot(b~a)
b
a
line plot
1:10
Index
dot-line plot
y轴
x轴


GDP
Labor
Kapital
KR

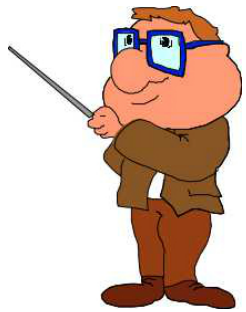


2.2 pie函数

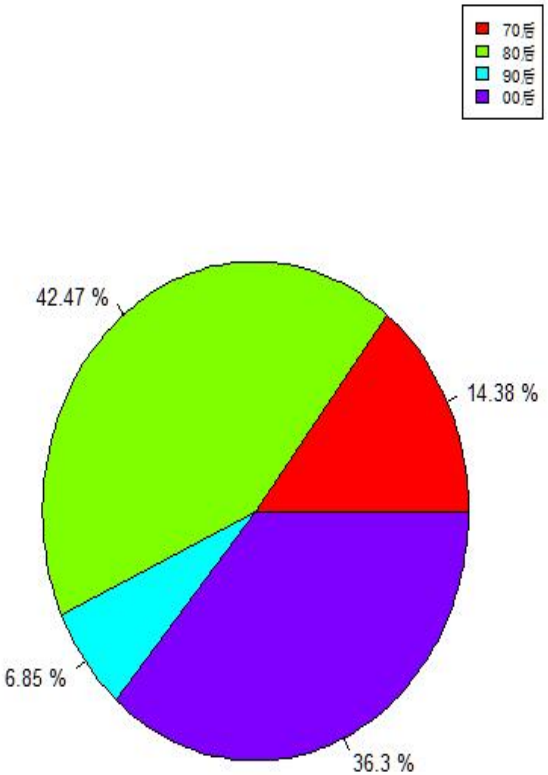
- 将数值向量的值以饼图的形式绘制出来。
- 使用方法: pie(x, labels, radius, main, col, clockwise)
  - x: 包含饼图中使用的数值的向量
  - labels: 用于描述切片的标签
  - radius: 用来表示饼图圆的半径(–1和+1 之间的值)
  - col: 表示调色板
  - clockwise: 是一个逻辑值，指示片是顺时针还是逆时针绘制

pie函数例子

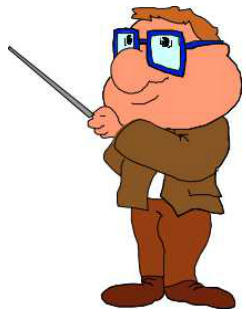
```
>  x <-  c(21, 62, 10, 53)
>  lbl <- c("70后", "80后", "90后", "00 后")
>  piepercent<- paste(round(100*x/sum(x), 2), "%")
>  win.graph(width=8, height=6, pointsize=8)
>  par(mfrow=c(1,2))
>  pie(x, labels = piepercent, main = "出生年龄段 - 饼状图",
+      col = rainbow(length(x)))
>  legend("topright", lbl, cex = 0.8, fill = rainbow(length(x)))

>  # 绘制3D饼图
>  library(plotrix)
>  pie3D(x, labels = lbl, explode = 0.1,
+        main = "出生年龄段 - 饼状图")
```



出生年龄段 - 饼状图

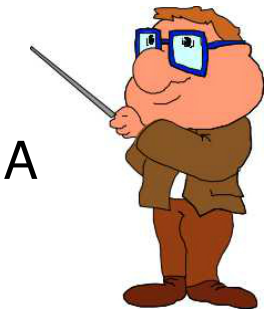


出生年龄段 - 饼状图



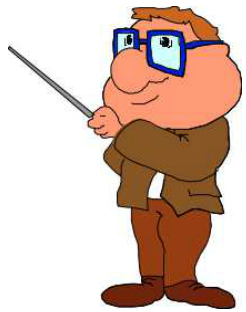
2.3 boxplot函数

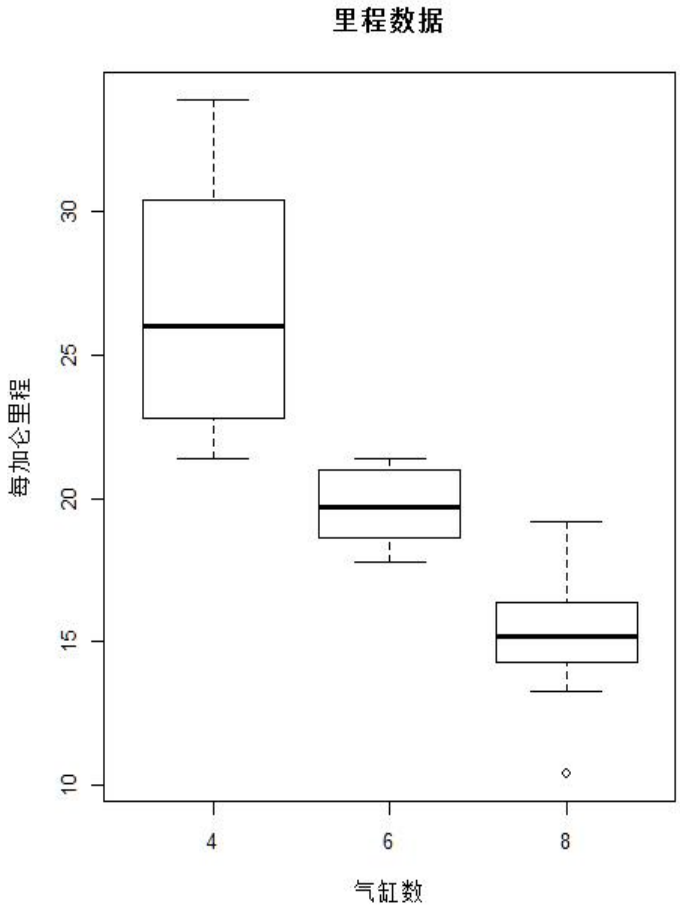
- 是数据集中数据分布情况的衡量标准。它将数据集分为三个四分位数。盒形图表示数据集中的最小值，最大值，中值，第一四分位数和第三四分位数。
- 使用方法: boxplot(formula, data, notch, varwidth, names, main,, na.action=NULL)
  - formula: 是公式，例如y ~ grp，其中y是由数据构成的向量，grp 为数据的分组，通常为因子
  - data: 是数据框
  - notch: 是一个逻辑值，设置为TRUE可以画出一个缺口
  - varwidth: 是一个逻辑值，设置为true以绘制与样本大小成比例的框的宽度
  - names: 是将在每个箱形图下打印的组标签
  - main: 用于给图表标题
  - na.action: 表示数据含有NAs时如何处理，默认情况下忽略NA

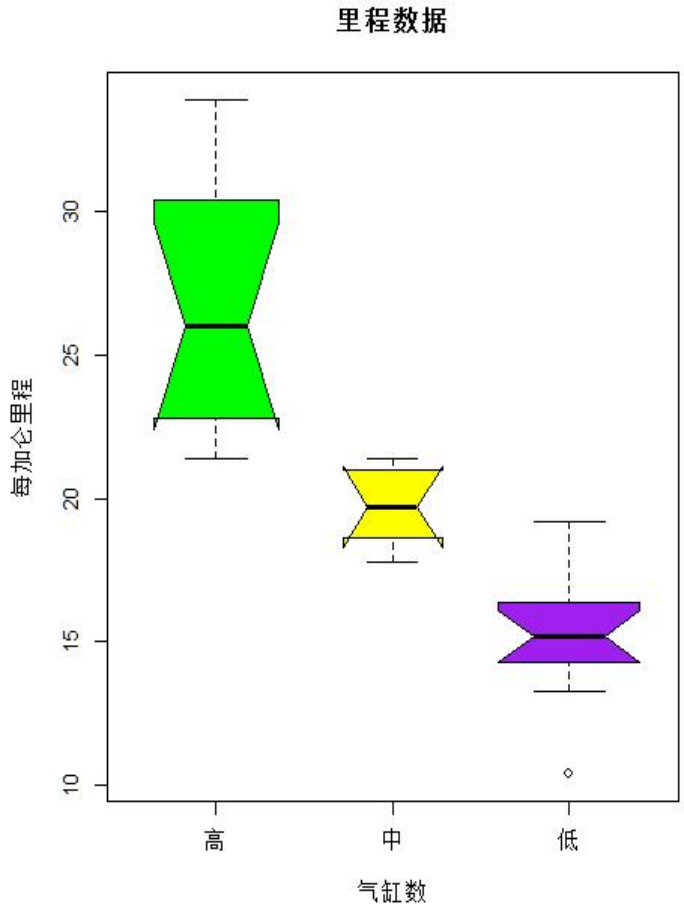


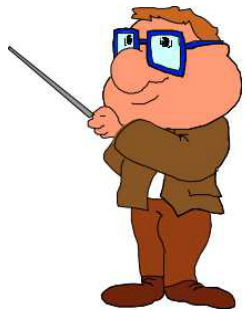
boxplot函数例子

```
>  win.graph(width=8, height=5, pointsize=8)
>  par(mfrow=c(1,2))
>  boxplot(mpg ~ cyl, data = mtcars, xlab = "气缸数",
+          ylab = "每加仑里程", main = "里程数据")

>  # 绘制带有凹槽的盒形图
>  boxplot(mpg ~ cyl, data = mtcars, xlab = "气缸数",
+          ylab = "每加仑里程", main = "里程数据",
+          notch = TRUE, varwidth = TRUE,
+          col = c("green","yellow","purple"),
+          names = c(" 高","中","低"))
```




里程数据
每加仑里程
10
15
20
25
30
4
6
8
气缸数


里程数据
每加仑里程
10
15
20
25
30
高
中
低
气缸数

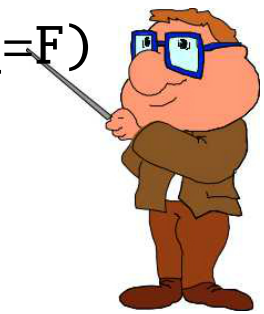


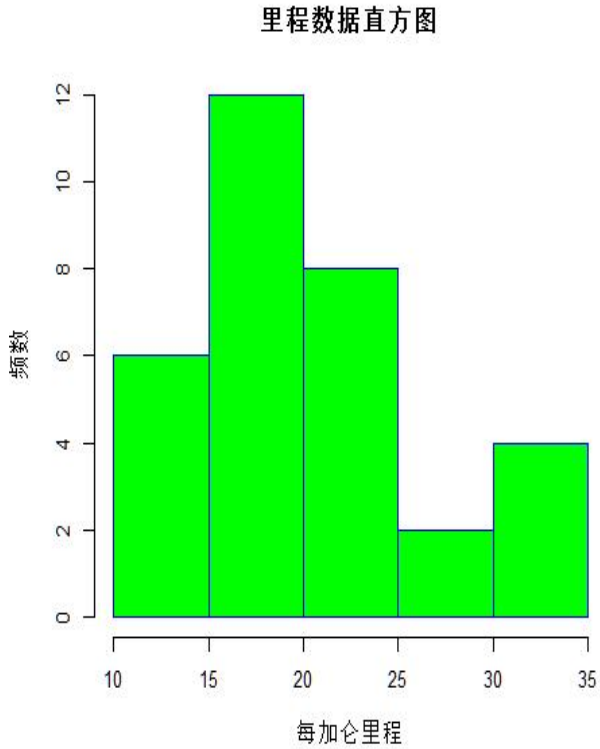
2.4 hist函数

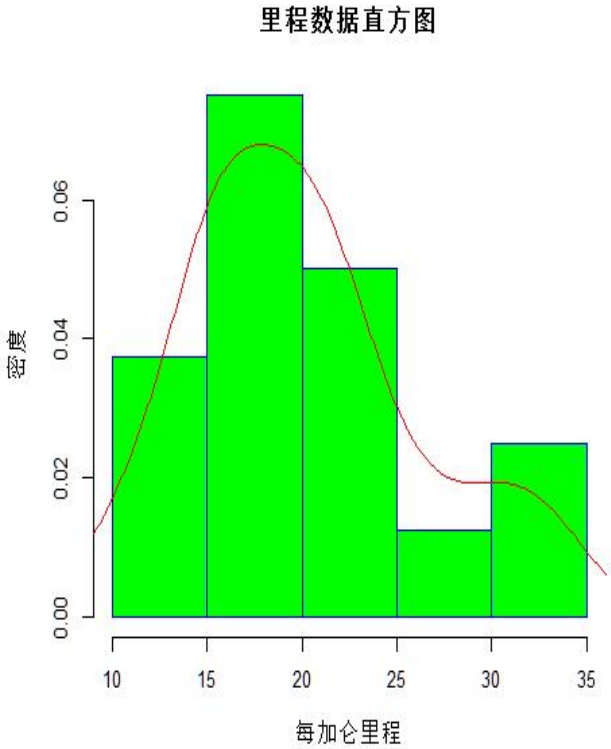
- 将数据取值的范围分成若干区间(一般为等间隔区间)，在等间隔的情况下，每个区间的长度成为组距。考察数据落入每一区间的频数或频率，在每个区间上画一个矩形，它的宽度是组距，高度为频数、频率或频率/组距，而在高度是频率/组距时，每一个矩形的面积恰为数据落入区间的频率，此时的直方图可以用来估计总体的概率密度。
- 使用方法: hist(x, main, xlab, xlim, ylim, breaks, col, border, freq)
  - x: 是包含直方图中使用数值的向量
  - main: 用于给图表标题
  - col: 用于设置条的颜色
  - border: 用于设置每个栏的边框颜色
  - xlab: 用于描述x轴
  - xlim: 用于指定x轴上的值范围
  - ylim: 用于指定y轴上的值范围
  - breaks: 是用来提及每个栏的宽度
  - freq: 是一个逻辑值，设置为FALSE表示绘制密度直方图，设置为TRUE表示绘制频数直方图

hist函数例子

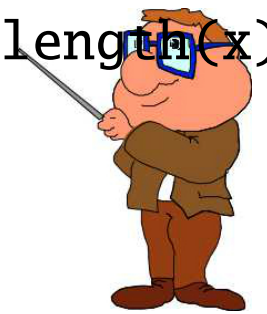
```
>  data(mtcars) # 提取R中已有数据集
>  input <- mtcars[,c('mpg','cyl')]
>  head(input)
>  win.graph(width=8, height=4, pointsize=8)
>  par(mfrow=c(1,2))
>  hist(input[,1], main="里程数据直方图", xlab="每加仑里程",
+       ylab="频数", col="green", border="blue", breaks=5)

>  # 绘制密度估计曲线图
>  hist(input[,1], main="里程数据直方图", xlab="每加仑里程",
+       ylab="密度", col="green", border="blue", freq=F)
>  lines(density(input[,1]), col="red")
```

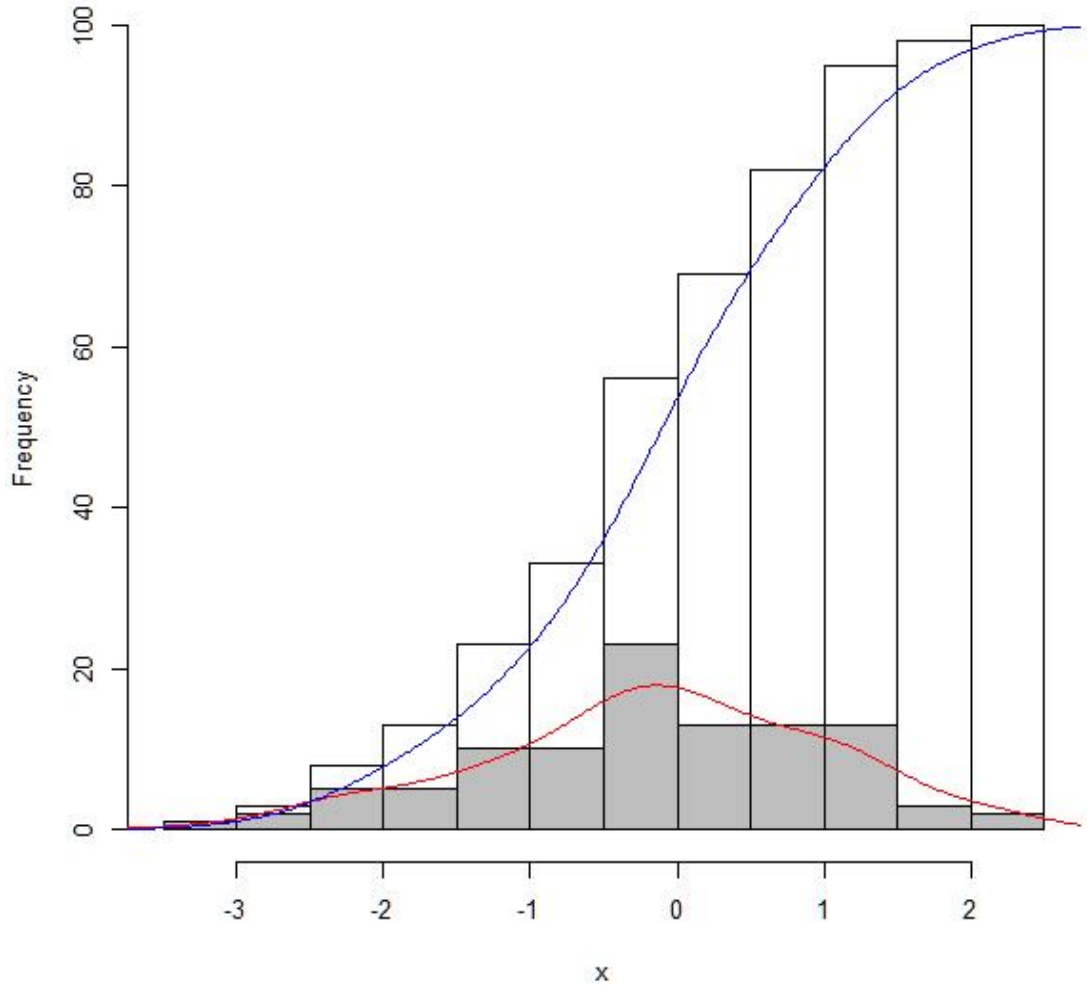


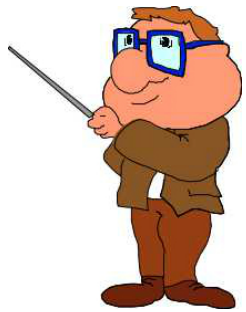

里程数据直方图
频数
0
2
4
6
8
10
12
10
15
20
25
30
35
每加仑里程


里程数据直方图
密度
0.00
0.02
0.04
0.06
10
15
20
25
30
35
每加仑里程

```
>  # Make some sample dat
>  x <- rnorm(100)
>  # Calculate and plot the two histograms
>  hcum <- h <- hist(x, plot=FALSE)
>  hcum$counts <- cumsum(hcum$counts)
>  win.graph(width=5, height=5, pointsize=8)
>  plot(hcum, main="")
>  plot(h, add=T, col="grey")
>  # Plot the density and cumulative density
>  d <- density(x)
>  lines(x = d$x, y = d$y * length(x) * diff(h$breaks)[1],
+        lwd = 1, col="red")
>  lines(x = d$x, y = cumsum(d$y)/max(cumsum(d$y)) * length(x)
+        lwd = 1, col="blue")
```




Frequency
0
20
40
60
80
100
-3
-2
-1
0
1
2
x

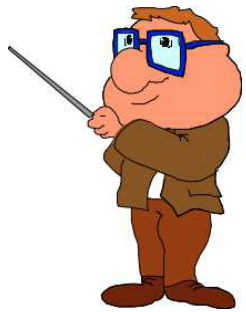


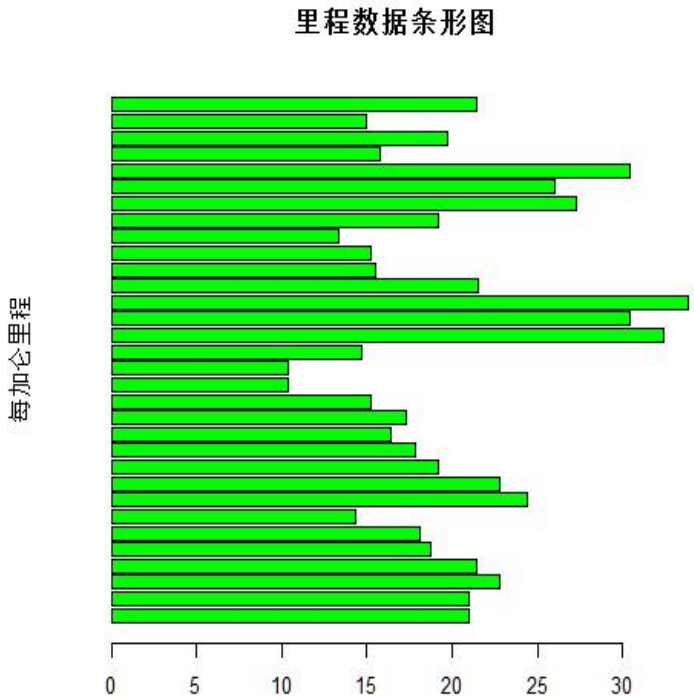
2.5 barplot函数

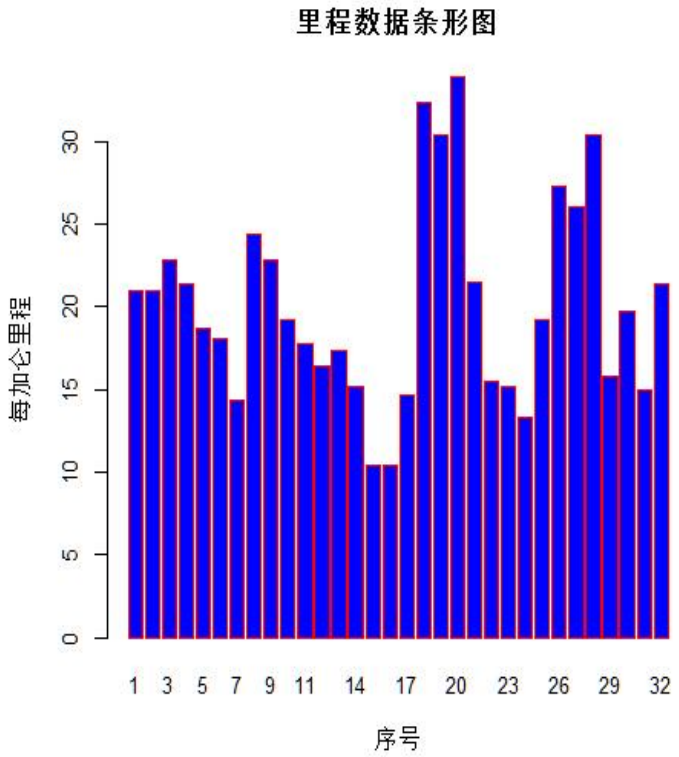
- 条形图表示矩形条中的数据，其长度与变量的值成比例。
- 使用方法: barplot(x, xlab, ylab, main, names.arg, col)
  - x: 是包含直方图中使用数值的向量或矩阵
  - xlab: 用于描述x轴
  - ylab: 用于描述y轴
  - main: 用于给图表标题
  - names.arg: 是在每个栏下显示的名称向量
  - col: 用于设置条的颜色

barplot函数例子

```
win.graph(width=8, height=4, pointsize=8)
par(mfrow=c(1,2))
barplot(input[,1], main="里程数据条形图", ylab="每加仑里程",
     horiz=T, col="green")
index <- 1:length(input[,1])
barplot(input[,1], main="里程数据条形图", ylab="每加仑里程",
     xlab="序号", col="blue", border="red", names.arg=index)
```



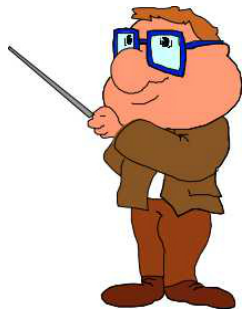

里程数据条形图
每加仑里程
0
5
10
15
20
25
30

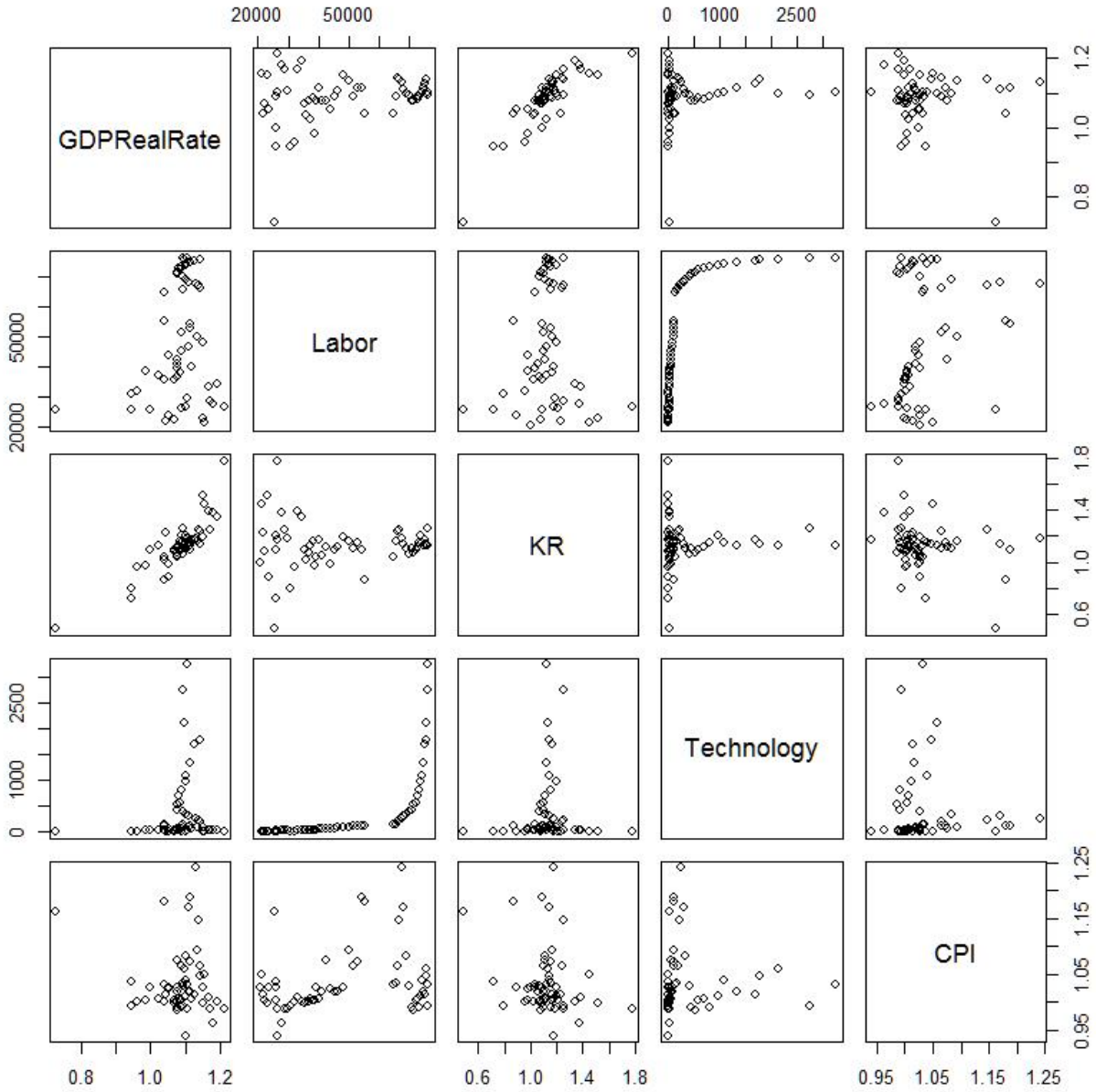

里程数据条形图
每加仑里程
0
5
10
15
20
25
30
1
3
5
7
9
11
14
17
20
23
26
29
32
序号

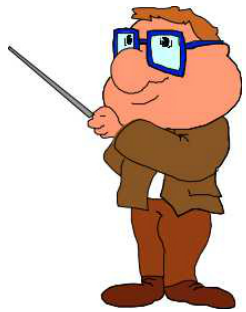
2.6 pairs函数

- 作多个变量的散点图矩阵.
- 参数为数据框对象.
- 效果与plot函数使用数据框参数效果相同
- 使用方法: pairs(x, ...)
  - x: 是数值型矩阵或数据框

```
>  win.graph(width=5, height=5, pointsize=8)
>  pairs(GDPdata[, c("GDPRealRate", "Labor", "KR",
+        "Technology", "CPI")])
>  plot(GDPdata[, c("GDPRealRate", "Labor", "KR",
+       "Technology", "CPI")])
```

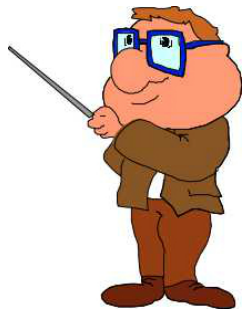



GDPRealRate
Labor
KR
Technology
CPI
20000
50000
0
1000
2500
0.8
1.0
1.2
0.6
1.4
1.8
0.95
1.05
1.15
1.25



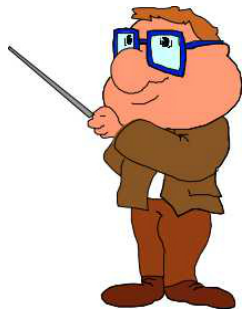
pairs函数：panel参数

- 默认散点图矩阵存在的问题：空间比较浪费，没有揭示更多内容
  - 矩阵图中上三角和下三角的内容雷同
  - 矩阵对角线只有变量的名称



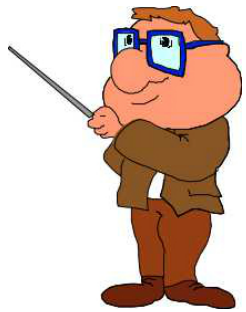
pairs函数：panel参数

- 默认散点图矩阵存在的问题：空间比较浪费，没有揭示更多内容
  - 矩阵图中上三角和下三角的内容雷同
  - 矩阵对角线只有变量的名称
- 解决方法：使用panel参数
  - panel定义每个矩阵元素图中的图形，默认为散点图
  - lower.panel 定义下三角矩阵的图形，默认为散点图
  - upper.panel 定义上三角矩阵的图形，默认为散点图
  - diag.panel定义对角线的图形，默认为不绘制图形



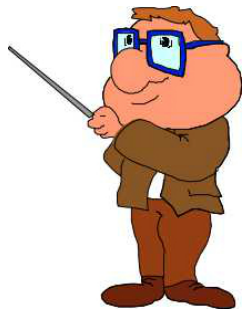
pairs函数：panel参数

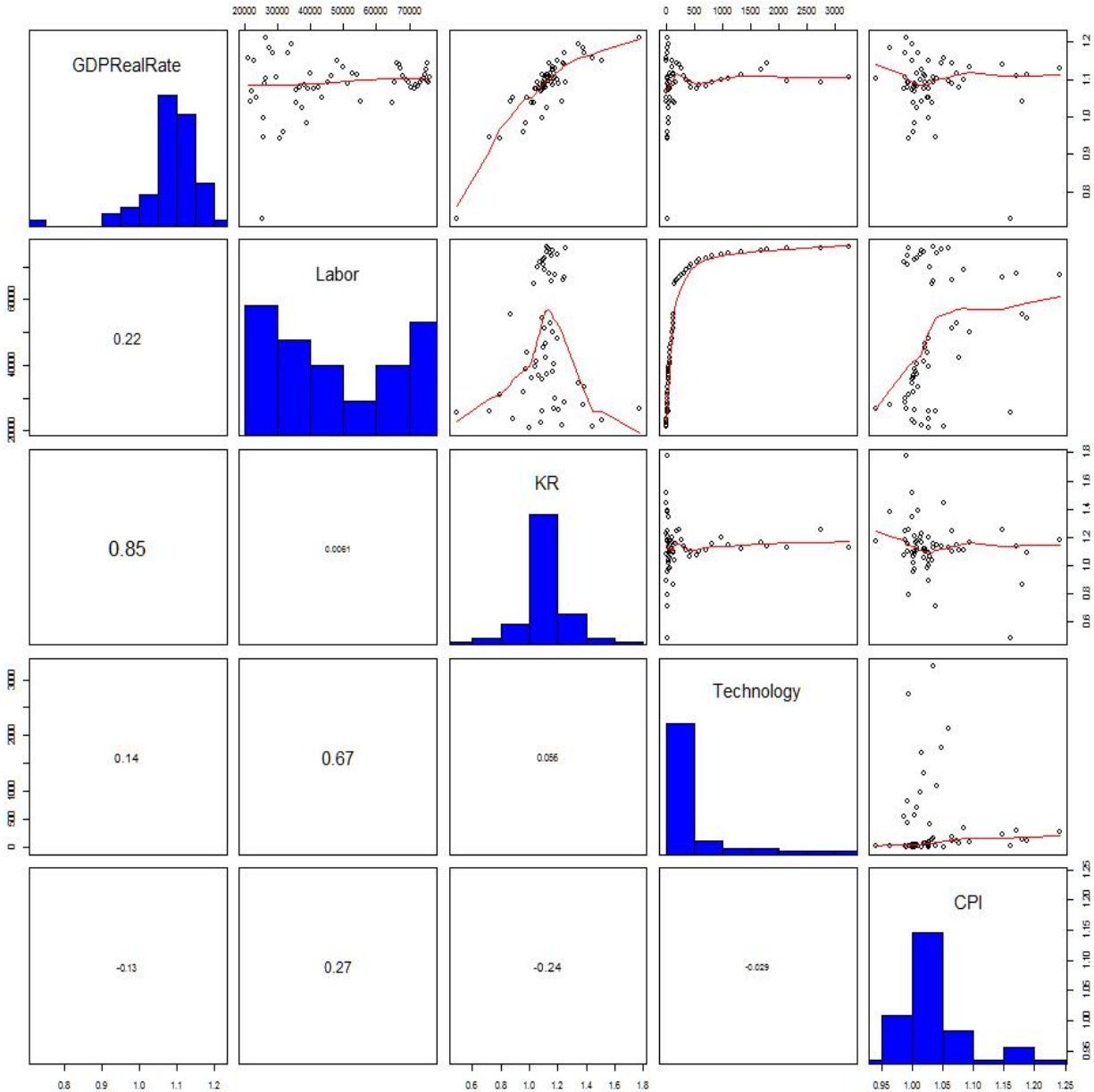
- 默认散点图矩阵存在的问题：空间比较浪费，没有揭示更多内容
  - 矩阵图中上三角和下三角的内容雷同
  - 矩阵对角线只有变量的名称
- 解决方法：使用panel参数
  - panel定义每个矩阵元素图中的图形，默认为散点图
  - lower.panel 定义下三角矩阵的图形，默认为散点图
  - upper.panel 定义上三角矩阵的图形，默认为散点图
  - diag.panel定义对角线的图形，默认为不绘制图形
- 上面几个参数应设置为作图函数，可以为已有的作图函数，也可以自己定义



pairs函数例子

```
>  panel.hist <- function(x, ...)
+  {
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(usr[1:2], 0, 1.5) )
+     h <- hist(x, plot = FALSE)
+     breaks <- h$breaks; nB <- length(breaks)
+     y <- h$counts; y <- y/max(y)
+     rect(breaks[-nB], 0, breaks[-1], y, col="blue", ...)
+  }
>  panel.cor <- function(x, y, digits=2, cex.cor)
+  {
+     usr <- par("usr"); on.exit(par(usr))
+     par(usr = c(0, 1, 0, 1))
+     r <- cor(x, y, use="complete.obs"); r1 <- abs(r)
+     txt <- format(c(r, 0.123456789), digits=digits)[1]
+     if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
+     text(0.5, 0.5, txt, cex = cex.cor*(r1+1)/5)
+  }
>  win.graph(width=5, height=5, pointsize=8)
>  pairs(GDPdata[, c("GDPRealRate", "Labor", "KR",
+        "Technology", "CPI")], diag.panel=panel.hist,
+        upper.panel=panel.smooth, lower.panel=panel.cor)
```



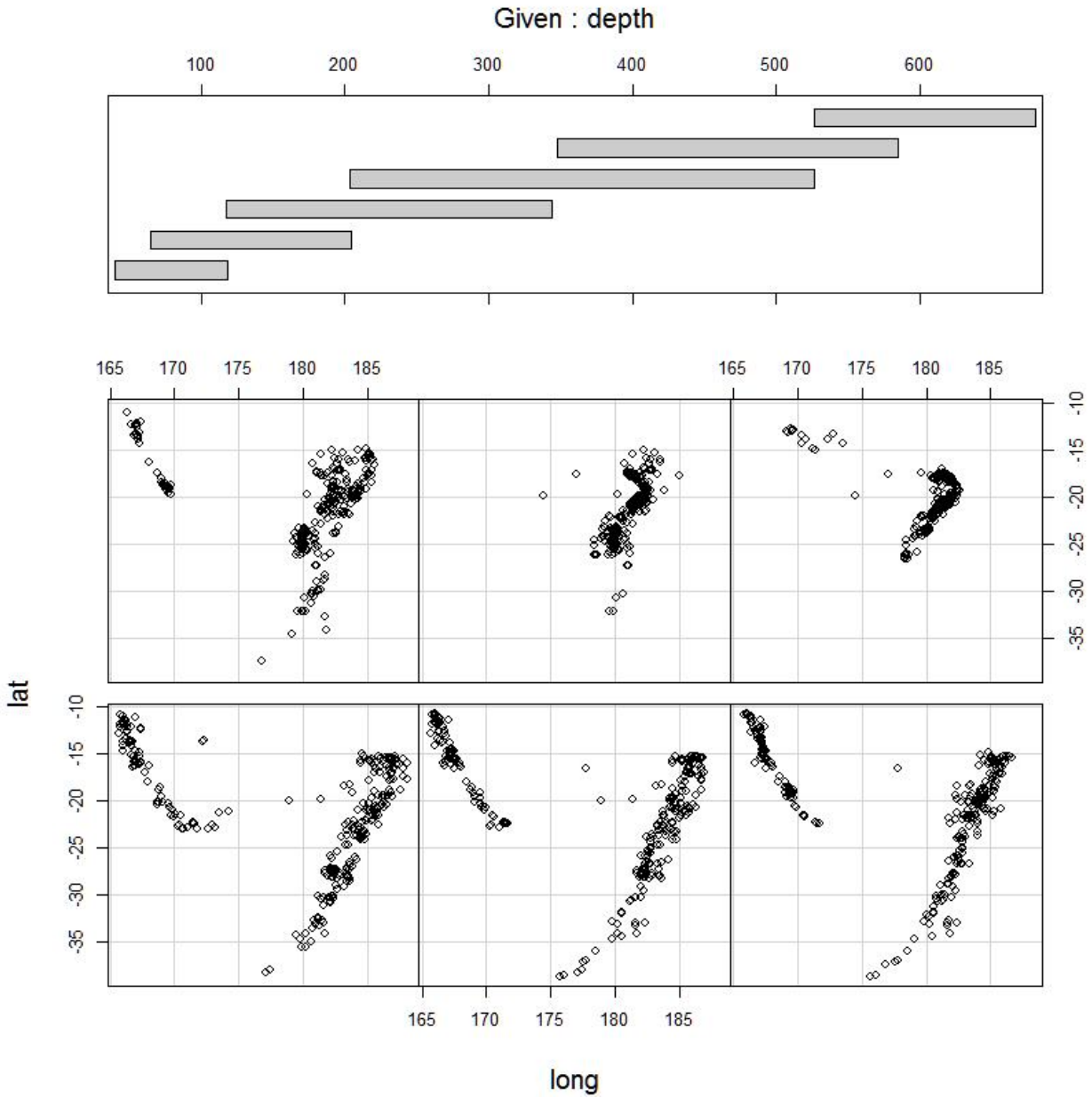

GDPRealRate
Labor
KR
Technology
CPI
0.22
0.85
0.0061
0.14
0.67
0.056
-0.13
0.27
-0.24
-0.029

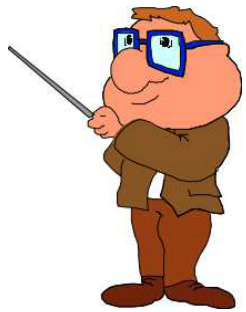
2.7 coplot函数

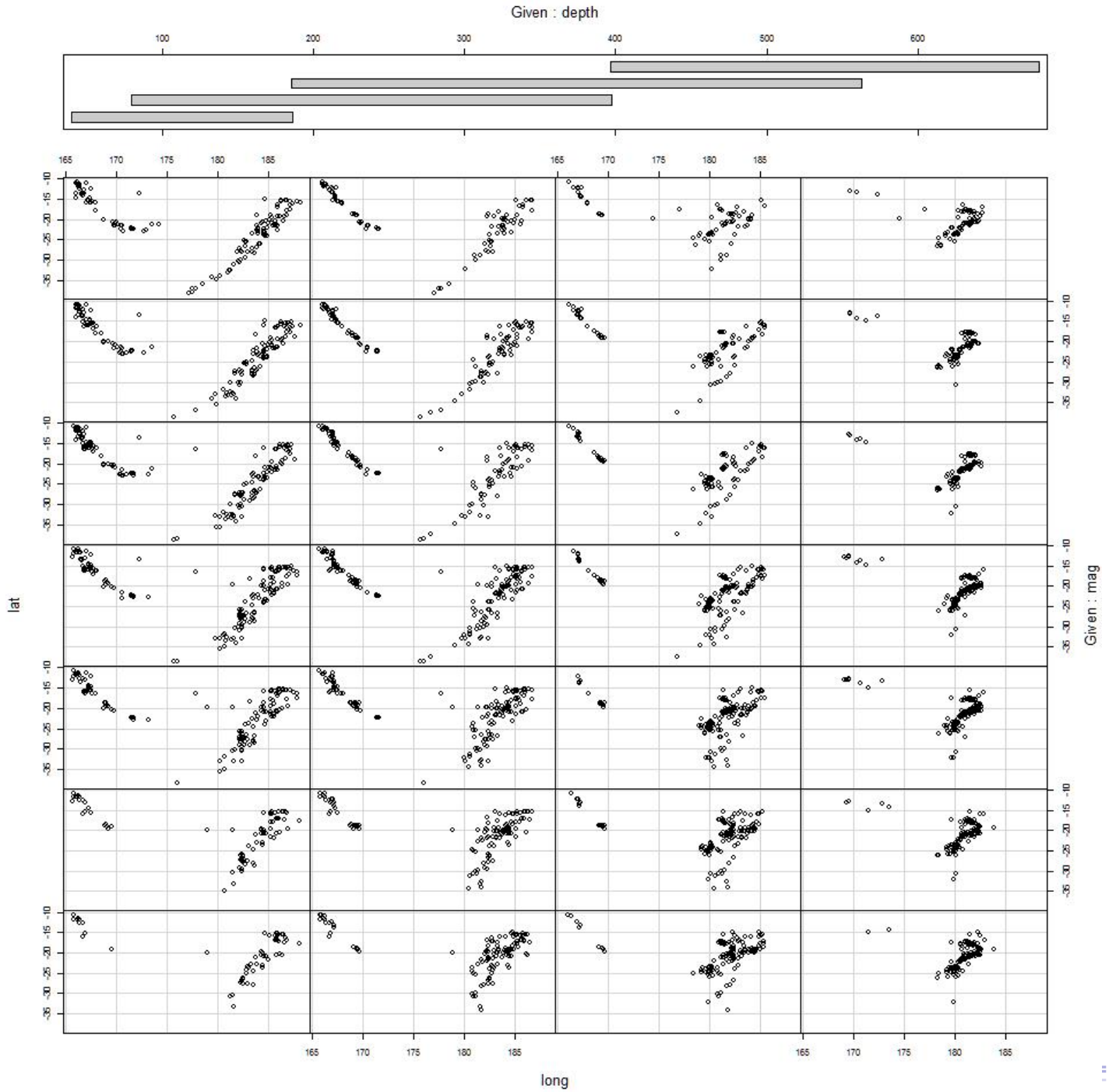
- 绘制给定条件下两个变量的条件图
- 使用方法: coplot(formula, data, xlab, ylab, main, xlim, ylim, show.given, ...)
  - formula: 是描述条件图的公式。形如$y \sim x|z$的表达形式，表示在条件变量z的前提下，y与x相关的图形。形如$y \sim x|u * v$ 表示在两个变量u和v的条件下，y与x的相关图形
  - data: 是数据框，包含了任何在formula参数中的变量值
  - xlab: 用于描述x轴
  - ylab: 用于描述y轴
  - main: 用于给图表标题
  - xlim: x轴的范围
  - ylim: y轴的范围
  - show.given: 是一个逻辑值(对2个条件变量长度可能为2)，对相应的条件变量(默认为TRUE) 条件图是否显示

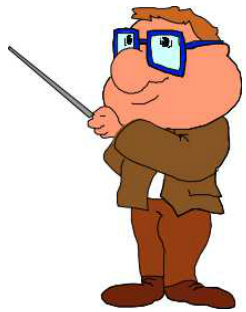
coplot函数例子

```
>  # Tonga Trench Earthquakes of Fiji since 1964
>  data(quakes)
>  win.graph(width=8, height=8, pointsize=8)
>  coplot(lat ~ long | depth, data = quakes)

>  # Conditioning on 2 variables:
>  ll.dm <- lat ~ long | depth * mag
>  win.graph(width=8, height=8, pointsize=8)
>  coplot(ll.dm, data = quakes, number = c(4, 7),
+         show.given = c(TRUE, FALSE))
```


Given : depth
100
200
300
400
500
600
165
170
175
180
185
-10
-15
-20
-25
-30
-35
lat
long

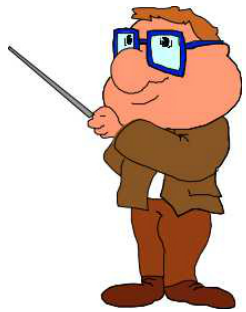



Given : depth
100
200
300
400
500
600
lat
long
Given : mag
165
170
175
180
185
-10
-15
-20
-25
-30
-35



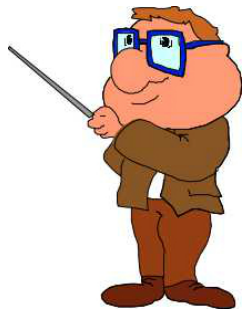
2.8 qqnorm函数

- 帮助我们鉴别样本的分布是否近似于正态分布
- 使用方法: qqnorm(x, main="Normal Q-Q Plot", xlab="Theoretical Quantiles", ylab="Sample Quantiles", plot.it=TRUE, datax=FALSE, ...)
  - x: 是包含QQ 图中使用数值的向量



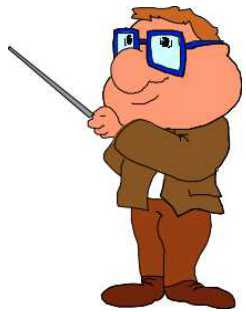
2.8 qqnorm函数

- 帮助我们鉴别样本的分布是否近似于正态分布
- 使用方法: qqnorm(x, main="Normal Q-Q Plot", xlab="Theoretical Quantiles", ylab="Sample Quantiles", plot.it=TRUE, datax=FALSE, ...)
  - x: 是包含QQ 图中使用数值的向量
- qqline(x, datax=FALSE, ...)绘制45度参考线
- 纵轴是数据的实际分位数，横轴是正态分布的理论分位数。如果数据是正态分布，则散点应该落在45度参考线上。即实际分位数与理论分位数偏差不大。反之，如果偏离较大，则证明数据并非来自正态总体



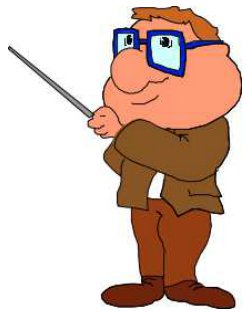
2.8 qqnorm函数

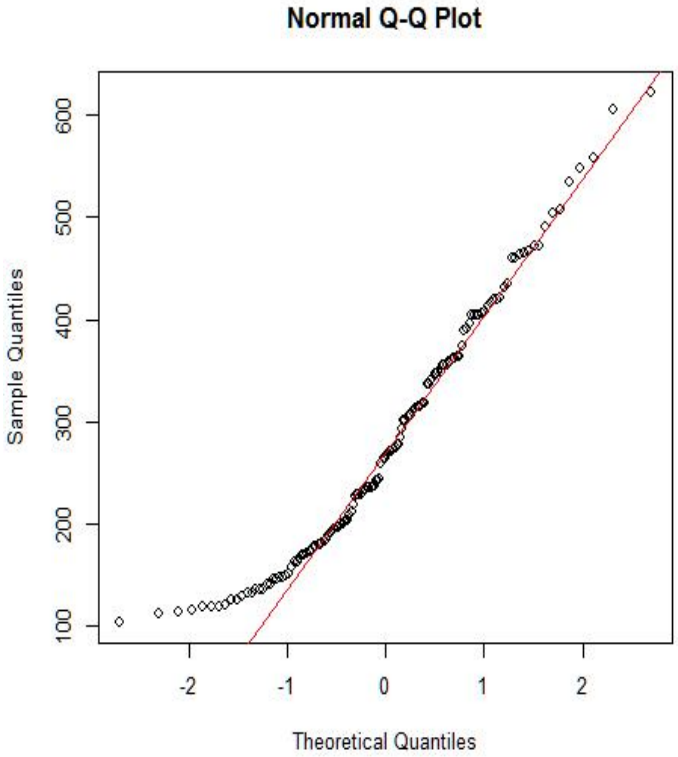
- 帮助我们鉴别样本的分布是否近似于正态分布
- 使用方法: qqnorm(x, main="Normal Q-Q Plot", xlab="Theoretical Quantiles", ylab="Sample Quantiles", plot.it=TRUE, datax=FALSE, ...)
  - x: 是包含QQ 图中使用数值的向量
- qqline(x, datax=FALSE, ...)绘制45度参考线
- 纵轴是数据的实际分位数，横轴是正态分布的理论分位数。如果数据是正态分布，则散点应该落在45度参考线上。即实际分位数与理论分位数偏差不大。反之，如果偏离较大，则证明数据并非来自正态总体
- qqplot(x,y)可以用来比较两个样本是否来自于同一分布

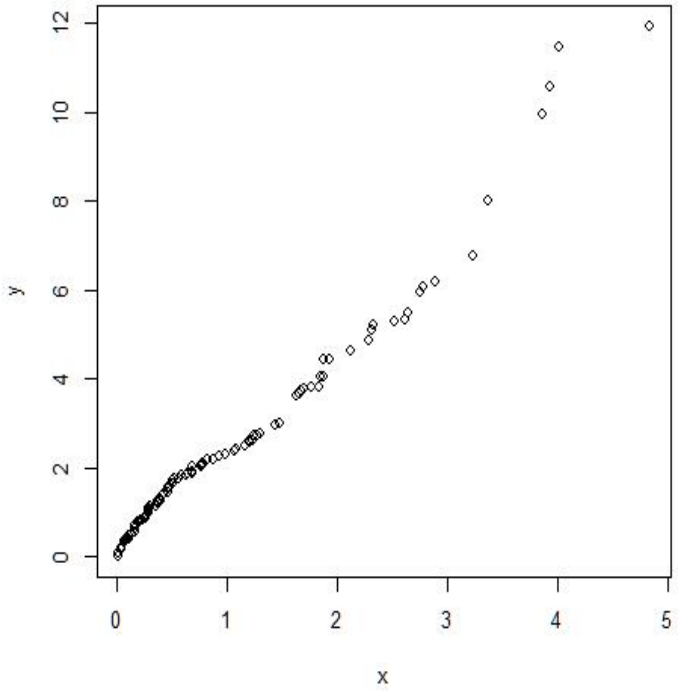


qqnorm函数例子

```
>  win.graph(width=8, height=4, pointsize=8)
>  par(mfrow=c(1,2))
>  data(AirPassengers)
>  qqnorm(AirPassengers)
>  qqline(AirPassengers,col="red")

>  x <- rexp(100)
>  y <- rchisq(100,3)
>  qqplot(x,y)
```

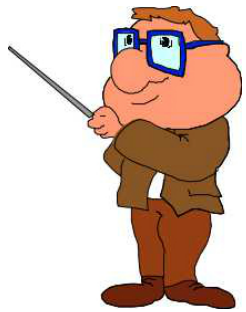



Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
100
200
300
400
500
600
-2
-1
0
1
2


y
x
0
2
4
6
8
10
12
0
1
2
3
4
5

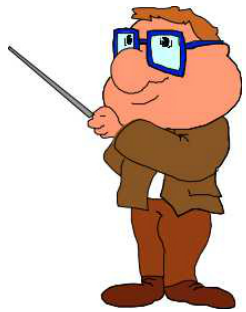
2.9 image函数

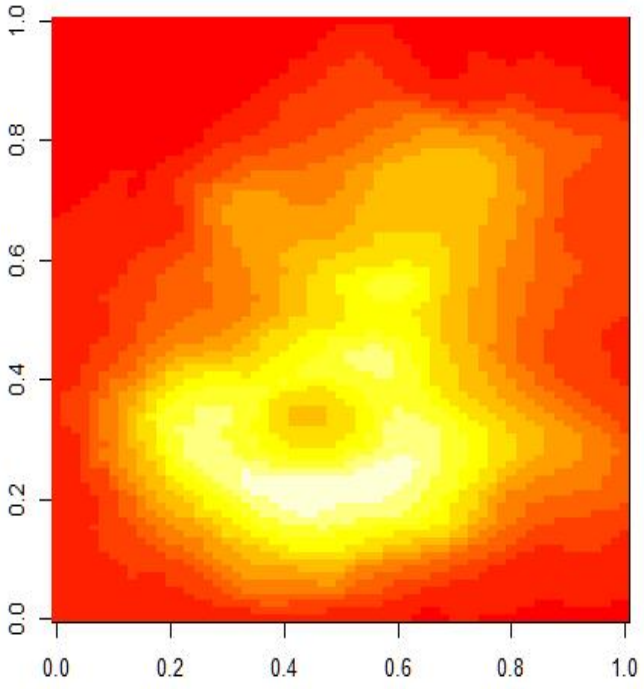
- 创建一个彩色或灰度矩形方格，颜色相当于z的值。从而能够用于绘制三维或空间数据图像。
- 使用方法: image(x, y, z, xlim, ylim, zlim, ...)
  - x,y: 网格线的位置，在该网格点上计算其z值。它们必须是有限，无缺失并且（严格来讲）是递增序列。默认情况下为0到1 之间的等距值
  - z: 一个数值或逻辑矩阵，它所含的值将被绘制(值为NAs是允许的)
  - xlim,ylim: 设置绘制的x和y值的范围，默认情况为x和y的范围
  - zlim: z的最小和最大值，并绘制其颜色，默认情况为z的有限值范围。每个给定的颜色被用于这个范围等距区间的上色。区间的中点代替区间范围，因而正好超出范围的值会被绘制

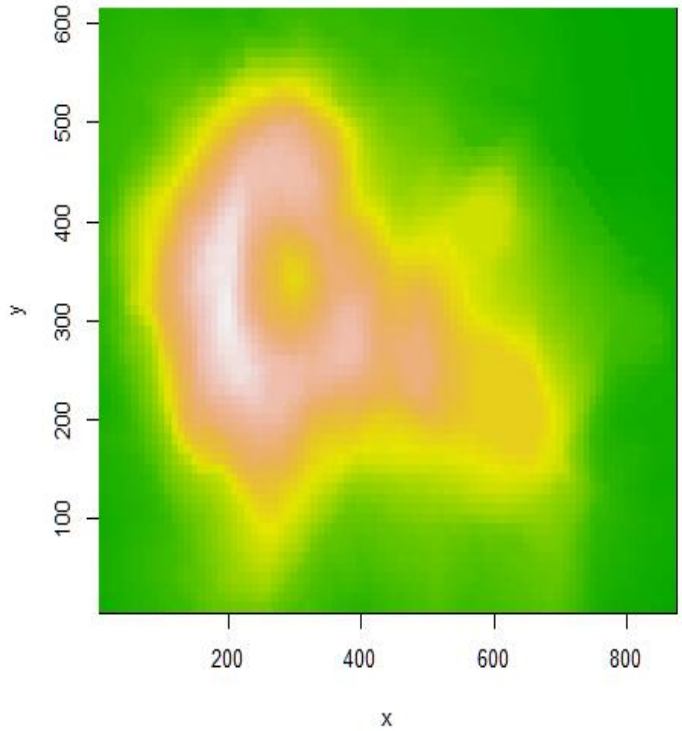


image函数例子

```
>  data(volcano)
>  win.graph(width=8, height=4, pointsize=8)
>  par(mfrow=c(1,2))
>  image(t(volcano)[ncol(volcano):1,])

>  x <- 10*(1:nrow(volcano))
>  y <- 10*(1:ncol(volcano))
>  image(x, y, volcano, col = terrain.colors(100))
```




0.0
0.2
0.4
0.6
0.8
1.0

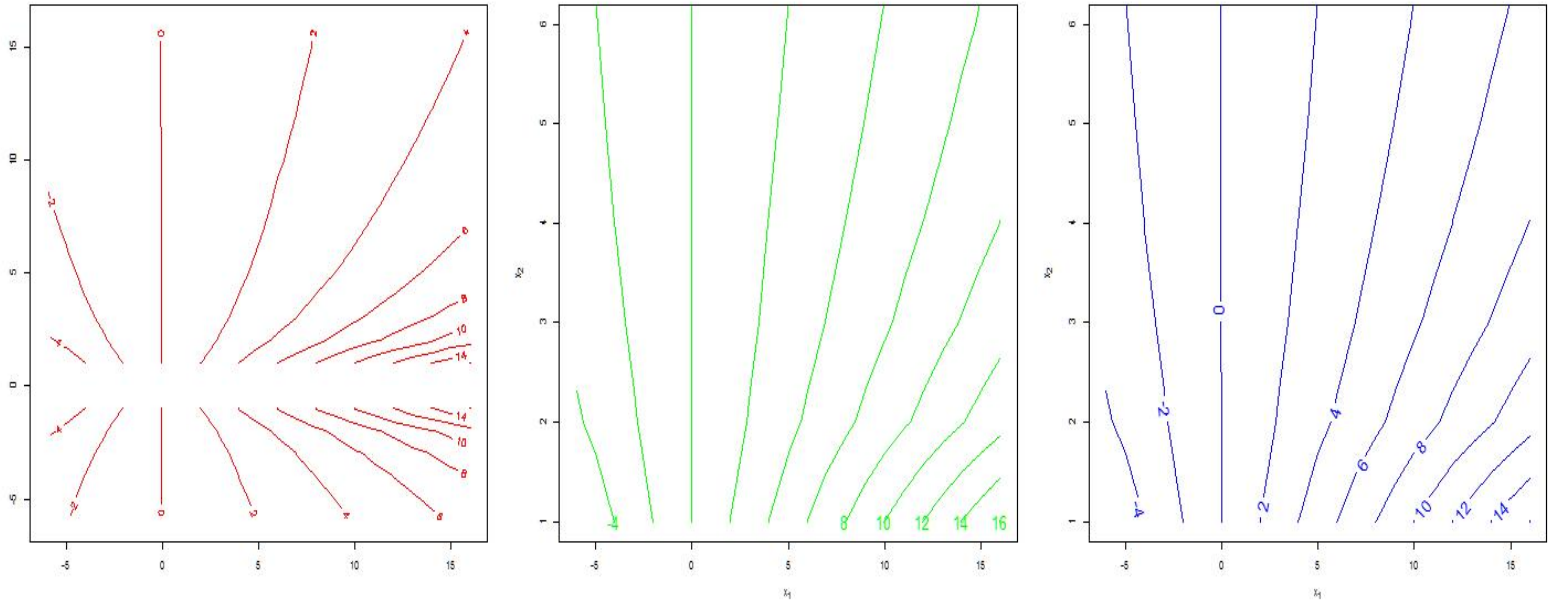

y
100
200
300
400
500
600
200
400
600
800
x

2.10 contour函数

- 绘制矩阵数据的等高线
- 使用方法: contour(x, y, z, xlim, ylim, zlim, method, ...)
  - x,y: 网格线的位置，在该网格点上计算其z 值。它们必须是有限，无缺失并且（严格来讲）是递增序列。默认情况下为0到1 之间的等距值
  - z: 一个数值或逻辑矩阵，它所含的值将被绘制(值为NAs是允许的)
  - xlim,ylim: 设置绘制的x和y 值的范围，默认情况为x和y的范围
  - zlim: z的最小和最大值，并绘制其颜色，默认情况为z的的有限值范围。每个给定的颜色被用于这个范围等距区间的上色。区间的中点代替区间范围，因而正好超出范围的值会被绘制
  - method: 设定等高线的画法，有三种取值：“simple”(在等高线的末端加标签、标签与等高线重叠),“edge”(在等高线的末端加标签、标签嵌在等高线内)和“flattest”(默认,在等高线最平缓的地方加标签、嵌在等高线内)

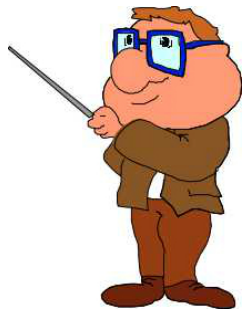
contour函数例子

```
>  x <- -6:16
>  z <- outer(x, sqrt(abs(x)), FUN = "/")
>  win.graph(width=12, height=4, pointsize=8)
>  par(mfrow=c(1,3))
>  contour(x, x, z, col = "pink", method = "edge",
+          vfont = c("sans serif", "plain"))

>  contour(x, x, z, ylim = c(1, 6), method = "simple", col="green",
+          labcex = 1, xlab = quote(x[1]), ylab = quote(x[2]))

>  contour(x, x, z, ylim = c(1, 6), method = "flattest", col="blue"
+          labcex = 1, xlab = quote(x[1]), ylab = quote(x[2]))
```



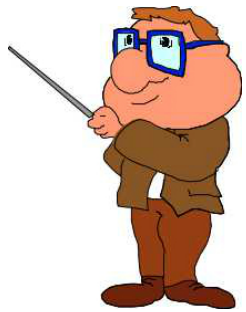
2.11 persp 函数

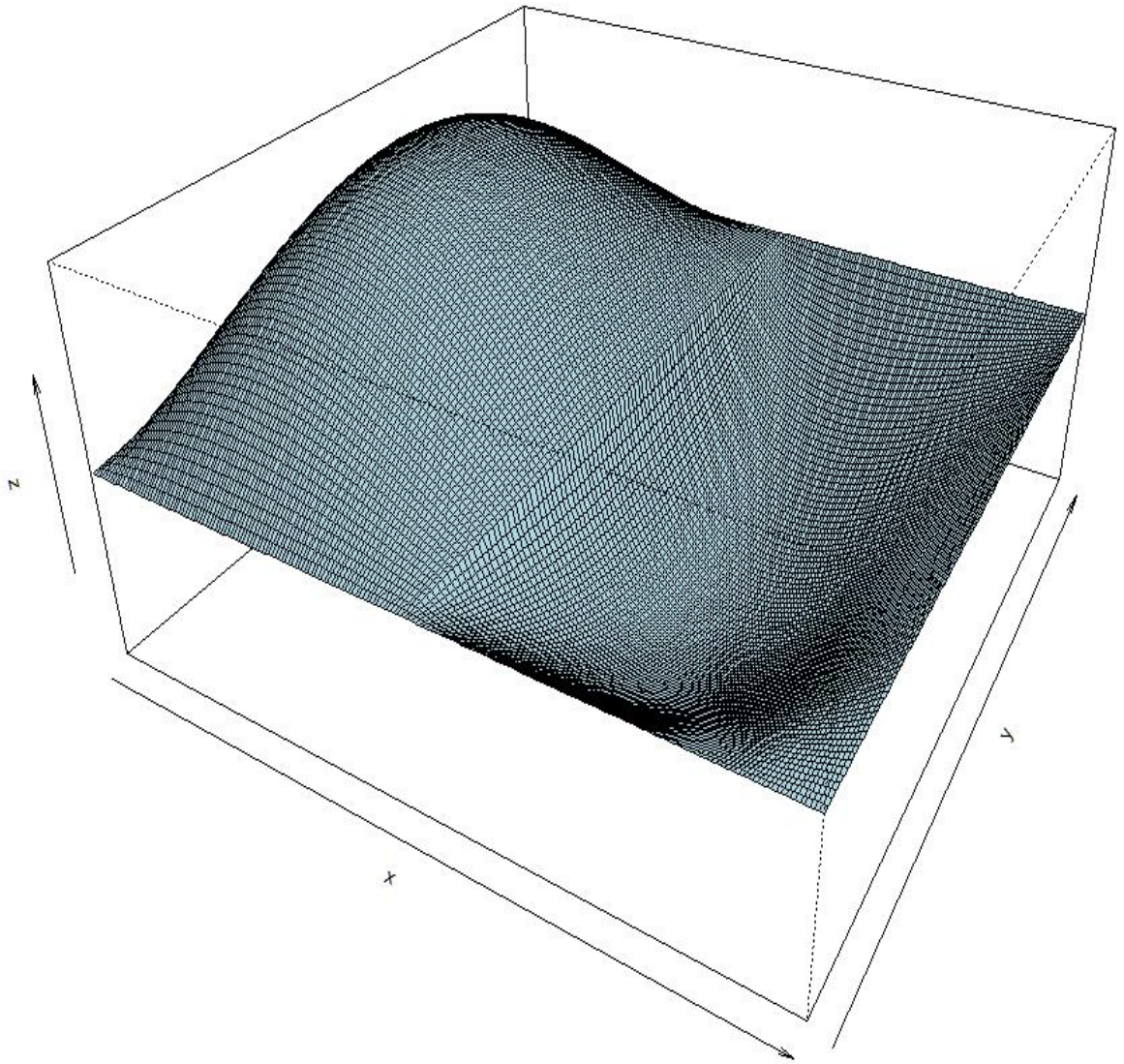
- 绘制三维曲面图
- 使用方法: persp(x, y, z, theta, phi, expand, col, ...)
  - x,y: 网格线的位置，在该网格点上计算其z 值。它们必须是有限，无缺失并且（严格来讲）是递增序列。默认情况下为0到1之间的等距值
  - z: 一个数值或逻辑矩阵，它所含的值将被绘制(值为NAs是允许的）
  - theta: 控制三维图的左右
  - phi: 控制三维图的上下
  - expand: 是一个在z坐标的扩张系数。通常在(0,1)内取，能在z方向上扩大或缩小图象的形状。控制三维图的立体性
  - col: 用于设置曲面的颜色

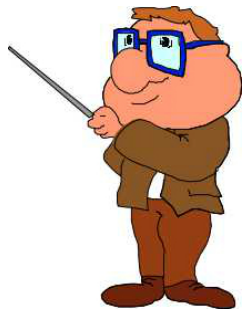


persp函数例子

```
>   x <- seq(0,2,0.01)
>   y <- seq(0,1,0.01)
>   f <- function(x,y)
+   {
+      ifelse(x<=1, 10*x*log10(x)*y*(y-1),
+             -10*(x-1)*log10(abs(x-1))*y*(y-1))
+   }
>   z <- outer(x,y,f)
>   win.graph(width=8, height=8, pointsize=8)
>   persp(x,y,z,theta=30,phi=30,expand=0.5,col="lightblue")
```

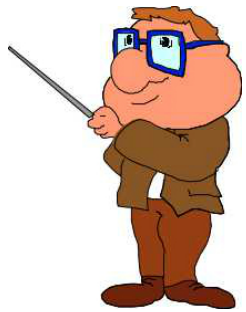



z
x
y



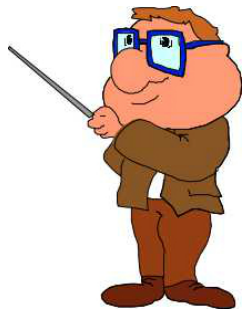
2.12 heatmap函数

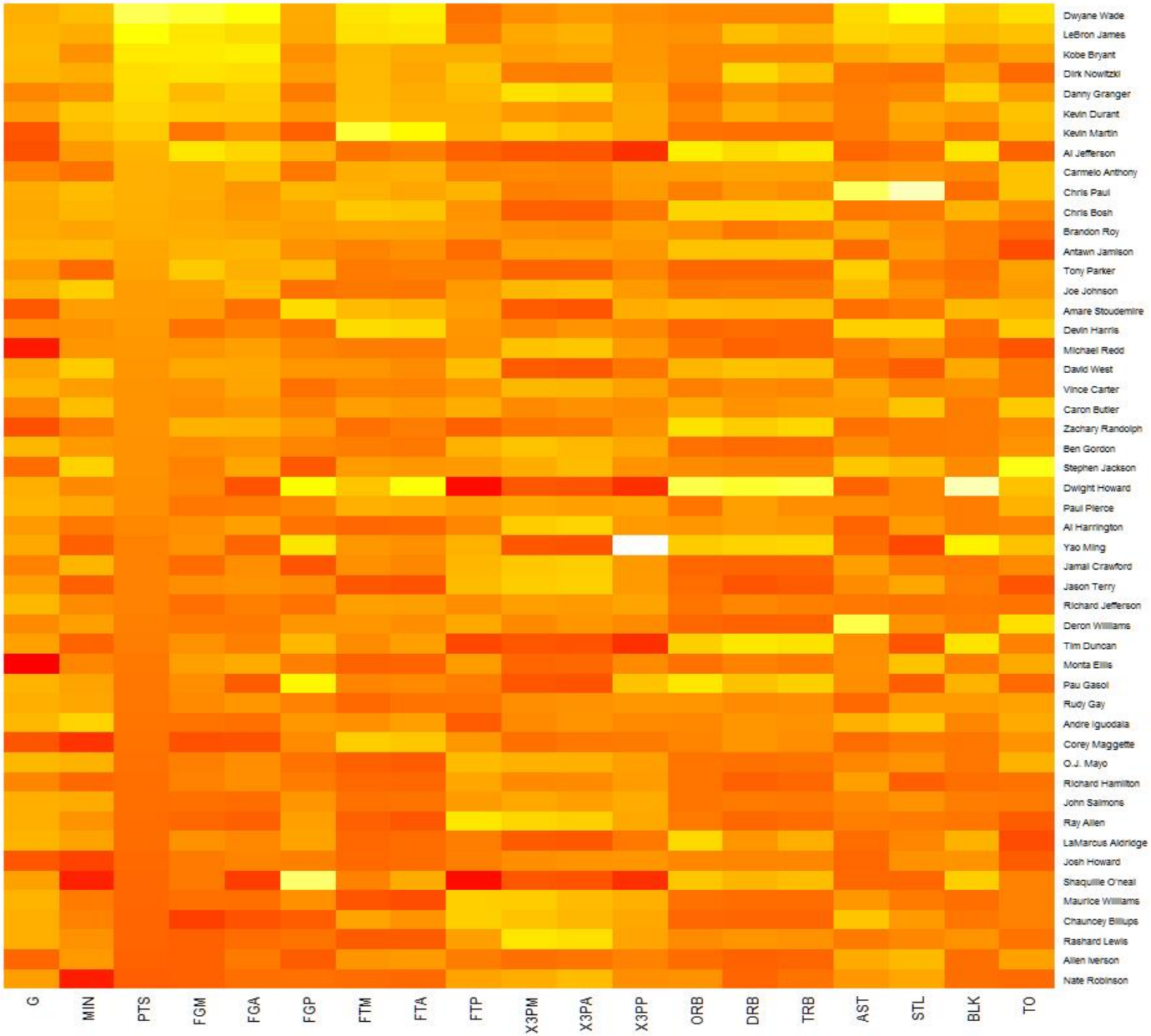
- 绘制热图
- 使用方法: heatmap(x, Rowv=NULL, Colv=if(symm)“Rowv” else NULL, scale, margins, col, ...)
  - x: 是包含热图中使用数值的矩阵
  - Rowv: 决定是否和如何计算和重新排列行聚类分析
  - Colv: 决定是否和如何计算和重新排序列聚类分析
  - scale: 是一个字符，按照行的方向或列的方向对数值进行标准化处理
  - margins: 是一个长度为2的数值型向量，设置列名和行名长度
  - col: 用于设置颜色



heatmap函数例子

```
>  nba <- read.csv("E:\\R软件介绍48学时\\lecture 4\\ppg2008.csv", se
>  nba <- nba[order(nba$PTS),] # 按每场得分从大到小排序
>  row.names(nba) <- nba$Name # 以球员名字命名
>  nba <- nba[,2:20]
>  nba_matrix <- data.matrix(nba)
>  win.graph(width=8, height=8, pointsize=8)
>  heatmap(nba_matrix, Rowv=NA, Colv=NA, col = heat.colors(256),
+          scale="column", margins=c(5,10))
```




Dwyane Wade
LeBron James
Kobe Bryant
Dirk Nowitzki
Danny Granger
Kevin Durant
Kevin Martin
Al Jefferson
Carmelo Anthony
Chris Paul
Chris Bosh
Brandon Roy
Antawn Jamison
Tony Parker
Joe Johnson
Amare Stoudemire
Devin Harris
Michael Redd
David West
Vince Carter
Caron Butler
Zachary Randolph
Ben Gordon
Stephen Jackson
Dwight Howard
Paul Pierce
Al Harrington
Yao Ming
Jamal Crawford
Jason Terry
Richard Jefferson
Deron Williams
Tim Duncan
Monta Ellis
Pau Gasol
Rudy Gay
Andre Iguodala
Corey Maggette
O.J. Mayo
Richard Hamilton
John Salmons
Ray Allen
LaMarcus Aldridge
Josh Howard
Shaquille O'neal
Maurice Williams
Chauncey Billups
Rashard Lewis
Allen Iverson
Nate Robinson
G
MIN
PTS
FGM
FGA
FGP
FTM
FTA
FTP
X3PM
X3PA
X3PP
ORB
DRB
TRB
AST
STL
BLK
TO

3. 低级绘图函数
常见低阶函数及其用法

- points(x,y)：添加点
- lines(x,y)：添加线
- text(x,y,labels,...)：在(x,y)处添加用labels指定的文字
- mtext(text,side=3,line=0,...)：在边空处添加用text指定的文字
- segments(x_0, y_0, x_1, y_1)：从(x_0, y_0)到(x_1, y_1)画线段
- arrows(x_0, y_0, x_1, y_1)：从(x_0, y_0)到(x_1, y_1)画带箭头的线段
- abline(a,b)：绘制斜率为b和截距为a的直线
- abline(h=y)：在纵坐标y处画水平直线
- abline(v=x)：在横坐标x处画垂直线
- abline(lm.obj)：画出lm.obj确定的回归线
- rect(x_1, y_1, x_2, y_2)：绘制长方形，左下角为(x_1, y_1)，右上角为(x_2, y_2)
- legend(x,y,legend)：在点(x,y)处添加图例，说明内容由legend给出

4. 图形参数
4.1 图形参数和par函数

- 图形的显示可以用绘图参数来改良。
- 部分绘图参数可以作为图形函数的选项
- 也可以用函数**par**来永久地改变绘图参数，也就是说后来的图形都将按照**par**指定的参数来绘制。
- 例如：使用命令

  ```
  > par(bg="yellow")
  ```

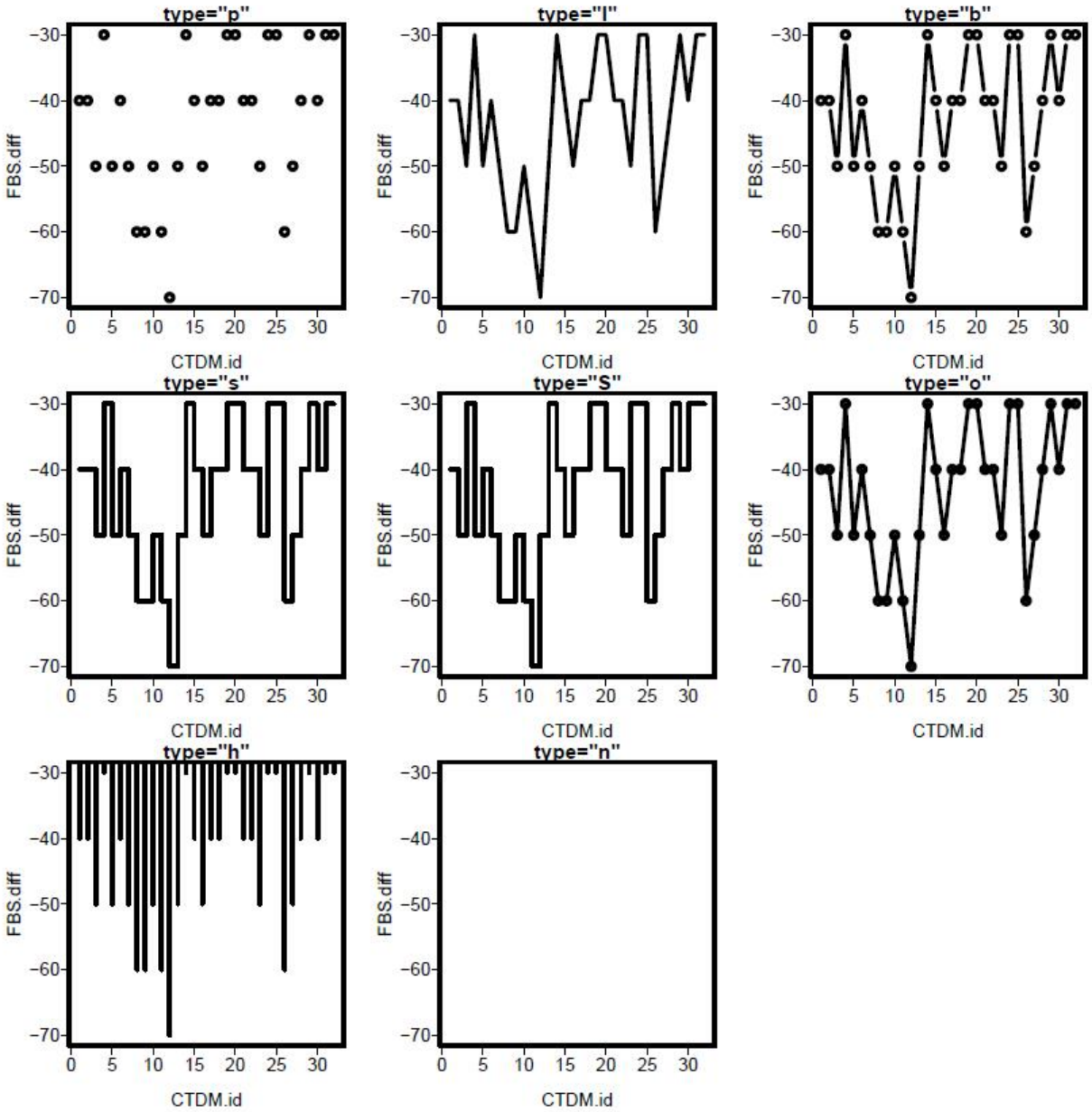
  将导致后面所有的图形都以黄色背景来绘制。
- 一共有70多个绘图参数，可以使用?par 来查看这些参数

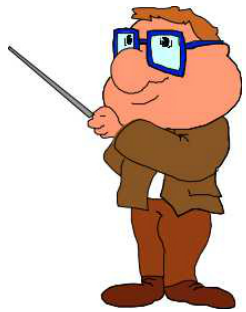
4.2 常用图形参数

Table: 常用的图形参数

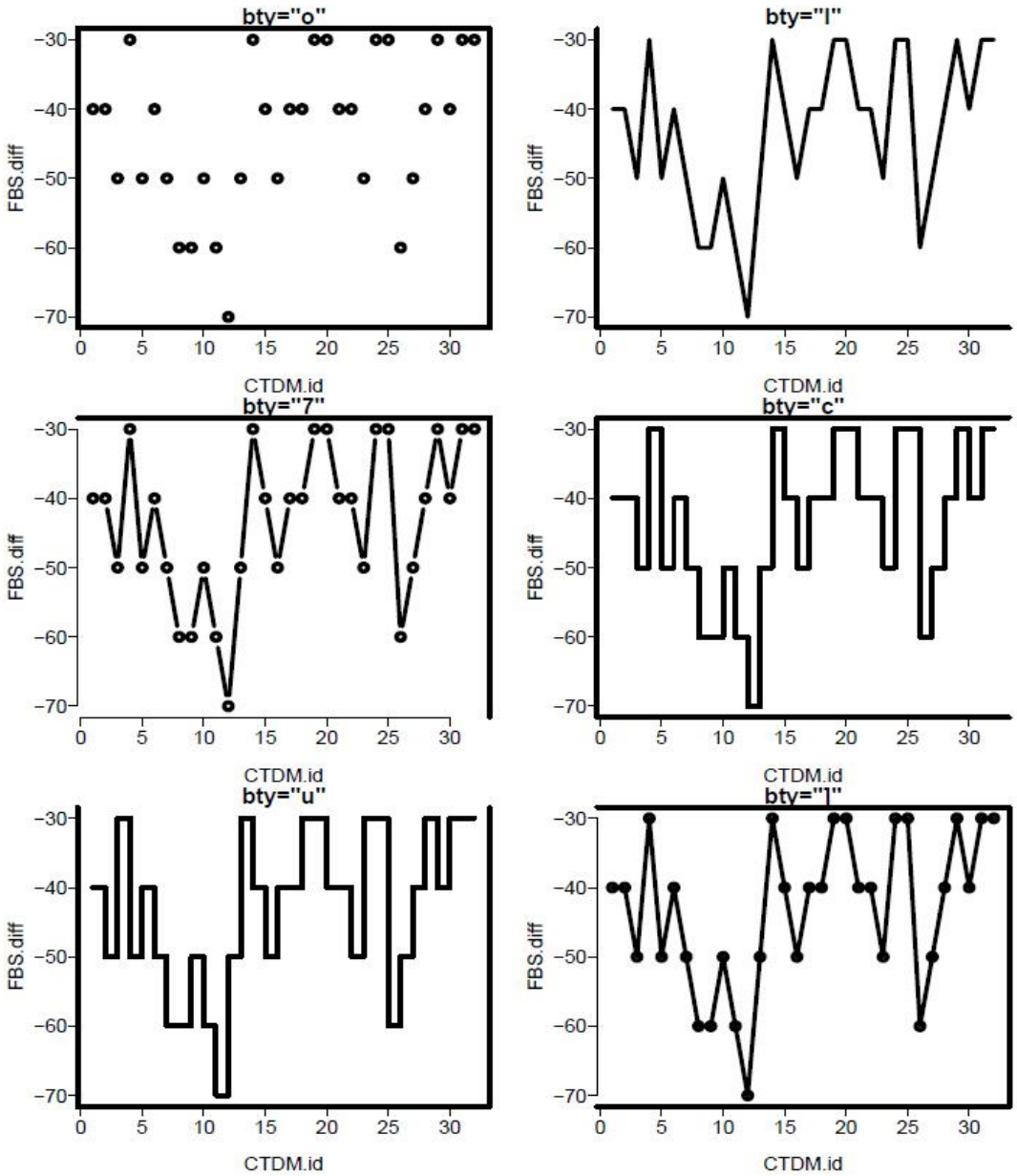
参数	用途
type	设置图形显示方式，可用的值为："p"，"l"，"b"，"o"，"s"，"S"，"h"，"n"
bg	设置图形背景颜色
bty	控制图形边框形状，可用的值为："o"，"l"，"7"，"c"，"u" 和"]"（边框和字符的外表相像）；如果bty="n"则不绘制边框
cex	控制默认状态下符号和文字大小的值
col	控制符号的颜色，颜色名称可以通过colors()函数查看
font	控制文字字体的整数（1: 正常，2: 粗体，3: 斜体，4: 粗斜体）
lty	控制连线的线型，可以是整数（1:实线，2:虚线，3:点线，4:点虚线，5:长虚线，6:双虚线）
lwd	控制连线宽度的数字
mar	控制图形边空的有4个值的向量c(bottom, left, top, right)，默认值为c(5.1, 4.1, 4.1, 2.1)
mfcol	c(nr,nc)的向量，分割绘图窗口为nr行nc列的矩阵布局，按列次序使用各子窗口
mfrow	同上，但是按行次序使用各子窗口
pch	控制符号的类型，可以是0到25的整数，也可以是具体字符

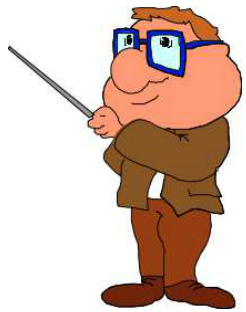
类型


type="p"
type="l"
type="b"
type="s"
type="S"
type="o"
type="h"
type="n"
FBS.diff
CTDM.id
−30
−40
−50
−60
−70
0
5
10
15
20
25
30

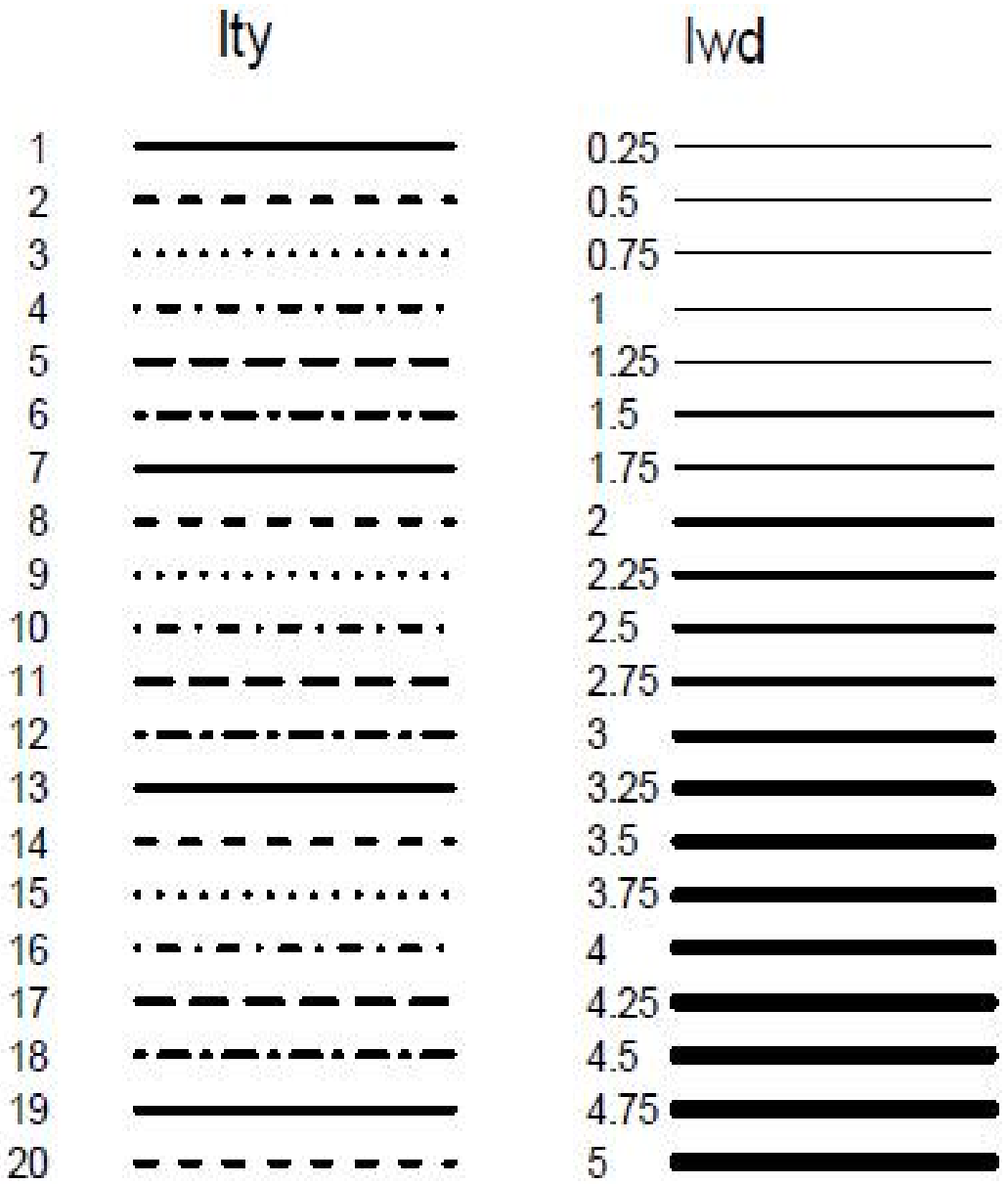


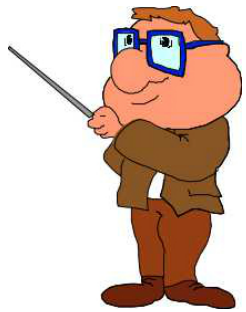
边框


btw="o"
btw="l"
CTDM.id
btw="7"
CTDM.id
btw="c"
CTDM.id
btw="u"
CTDM.id
btw="]"
CTDM.id
FBS.diff
−30
−40
−50
−60
−70
0
5
10
15
20
25
30

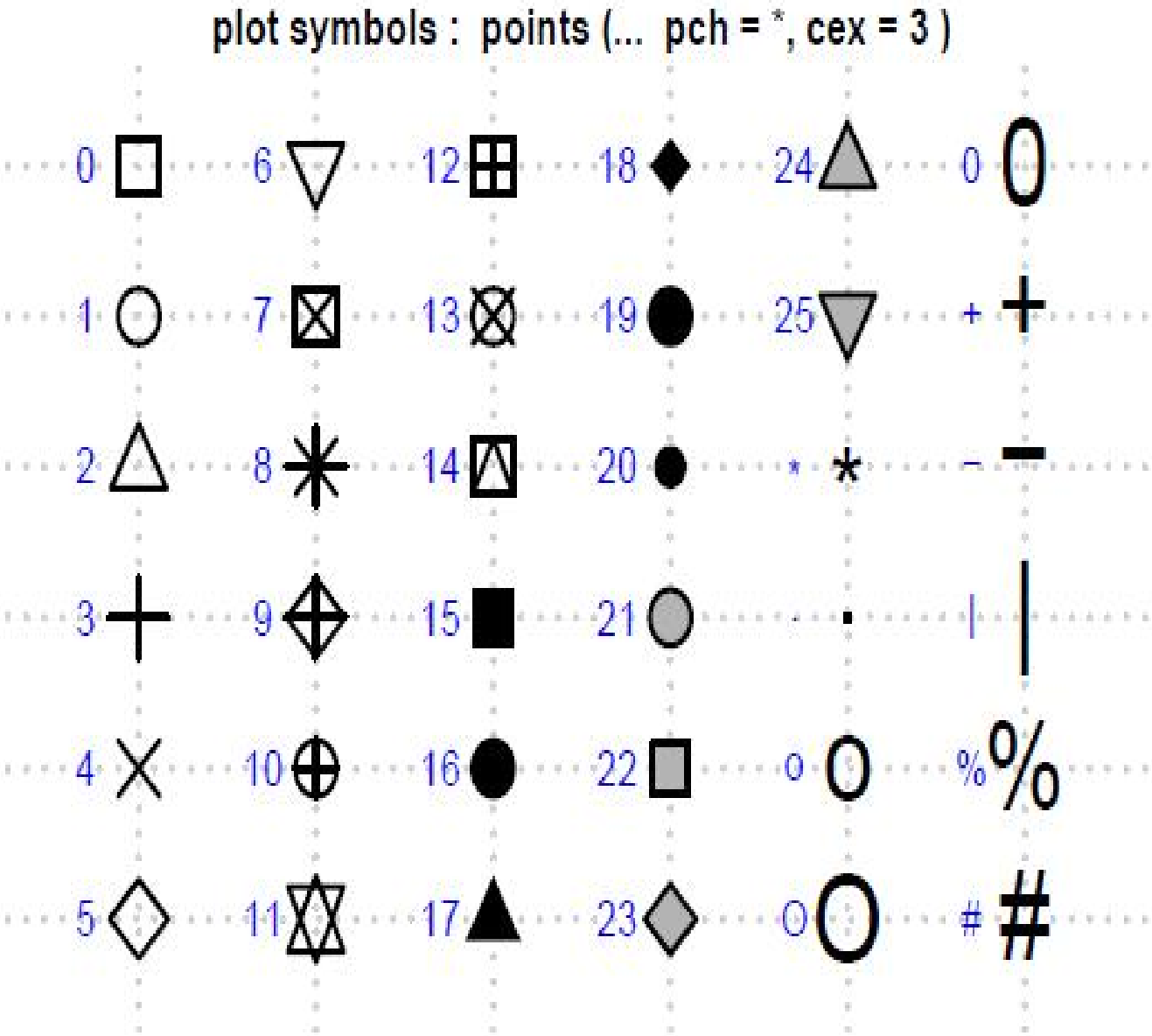


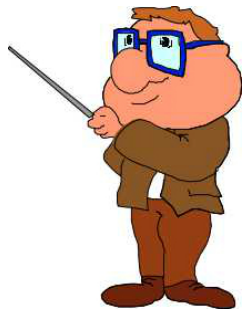
线条


lty
lwd
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
0.25
0.5
0.75
1
1.25
1.5
1.75
2
2.25
2.5
2.75
3
3.25
3.5
3.75
4
4.25
4.5
4.75
5

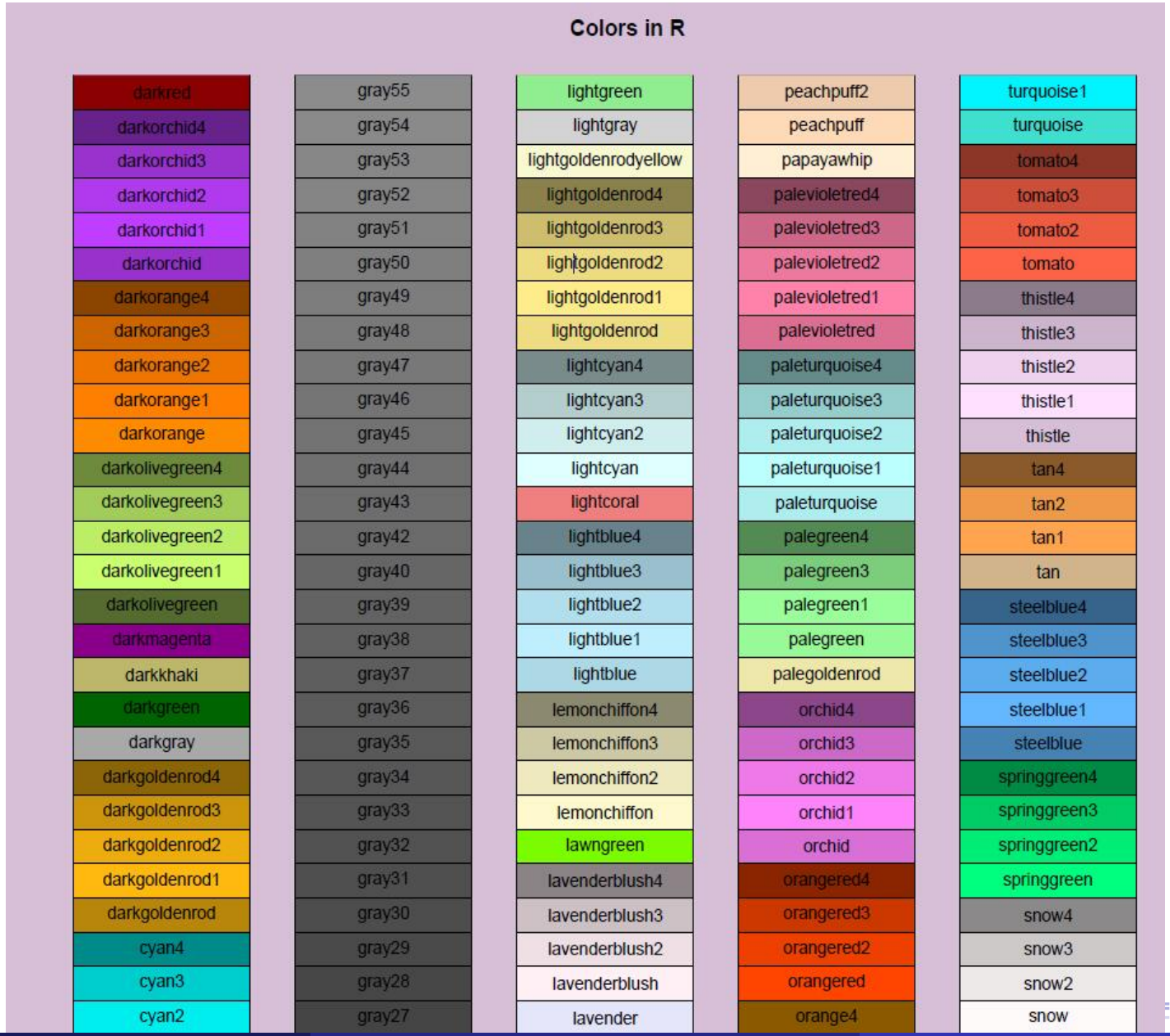


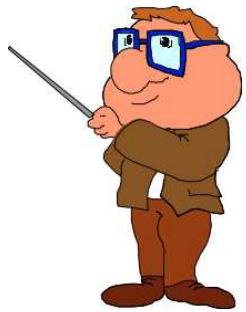
点


plot symbols : points (... pch = *, cex = 3)
0
6
12
18
24
0
1
7
13
19
25
+
2
8
14
20
*
–
3
9
15
21
.
|
4
10
16
22
o
%
5
11
17
23
O
#



颜色


Colors in R
darkred
darkorchid4
darkorchid3
darkorchid2
darkorchid1
darkorchid
darkorange4
darkorange3
darkorange2
darkorange1
darkorange
darkolivegreen4
darkolivegreen3
darkolivegreen2
darkolivegreen1
darkolivegreen
darkmagenta
darkkhaki
darkgreen
darkgray
darkgoldenrod4
darkgoldenrod3
darkgoldenrod2
darkgoldenrod1
darkgoldenrod
cyan4
cyan3
cyan2
gray55
gray54
gray53
gray52
gray51
gray50
gray49
gray48
gray47
gray46
gray45
gray44
gray43
gray42
gray40
gray39
gray38
gray37
gray36
gray35
gray34
gray33
gray32
gray31
gray30
gray29
gray28
gray27
lightgreen
lightgray
lightgoldenrodyellow
lightgoldenrod4
lightgoldenrod3
lightgoldenrod2
lightgoldenrod1
lightgoldenrod
lightcyan4
lightcyan3
lightcyan2
lightcyan
lightcoral
lightblue4
lightblue3
lightblue2
lightblue1
lightblue
lemonchiffon4
lemonchiffon3
lemonchiffon2
lemonchiffon
lawngreen
lavenderblush4
lavenderblush3
lavenderblush2
lavenderblush
lavender
peachpuff2
peachpuff
papayawhip
palevioletred4
palevioletred3
palevioletred2
palevioletred1
palevioletred
paleturquoise4
paleturquoise3
paleturquoise2
paleturquoise1
paleturquoise
palegreen4
palegreen3
palegreen1
palegreen
palegoldenrod
orchid4
orchid3
orchid2
orchid1
orchid
orangered4
orangered3
orangered2
orangered
orange4
turquoise1
turquoise
tomato4
tomato3
tomato2
tomato
thistle4
thistle3
thistle2
thistle1
thistle
tan4
tan2
tan1
tan
steelblue4
steelblue3
steelblue2
steelblue1
steelblue
springgreen4
springgreen3
springgreen2
springgreen
snow4
snow3
snow2
snow



4.3 数学文字与符号

- 图形显示有时需添加数学文字与符号或公式。
- 可以用函数**expression()** 来添加数学文字与符号或公式。
- 例如：使用命令

```
>  text(x, y, expression(paste(bgroup("(", atop(n, x), ")"),
+        p^x, q^{n-x})))
```

在二项概率函数加上数学符号。

- 更多详尽的符号，可以通过下列方式查看：

```
help(plotmath)
example(plotmath)
demo(plotmath)
```

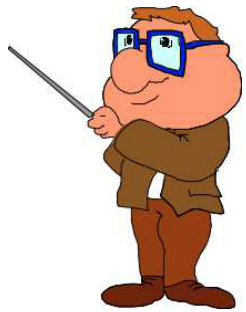
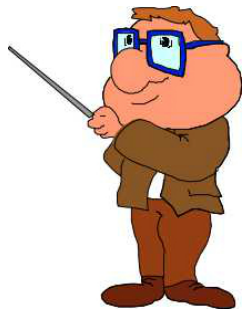


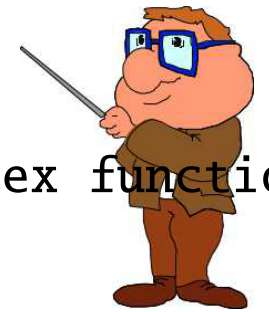
Table: 常用绘图用数学符号

代数运算	
$x + y$	$x + y$
$x - y$	$x - y$
$x \star y$	xy
x / y	x/y
$x \%+-\% y$	$x \pm y$
$x \%/\% y$	$x \div y$
$x \%*\% y$	$x \times y$
$- x$	$-x$
$+ y$	$+y$
上标、下标	
$x[i]$	x_i
$x\hat{}2$	x^2
...	

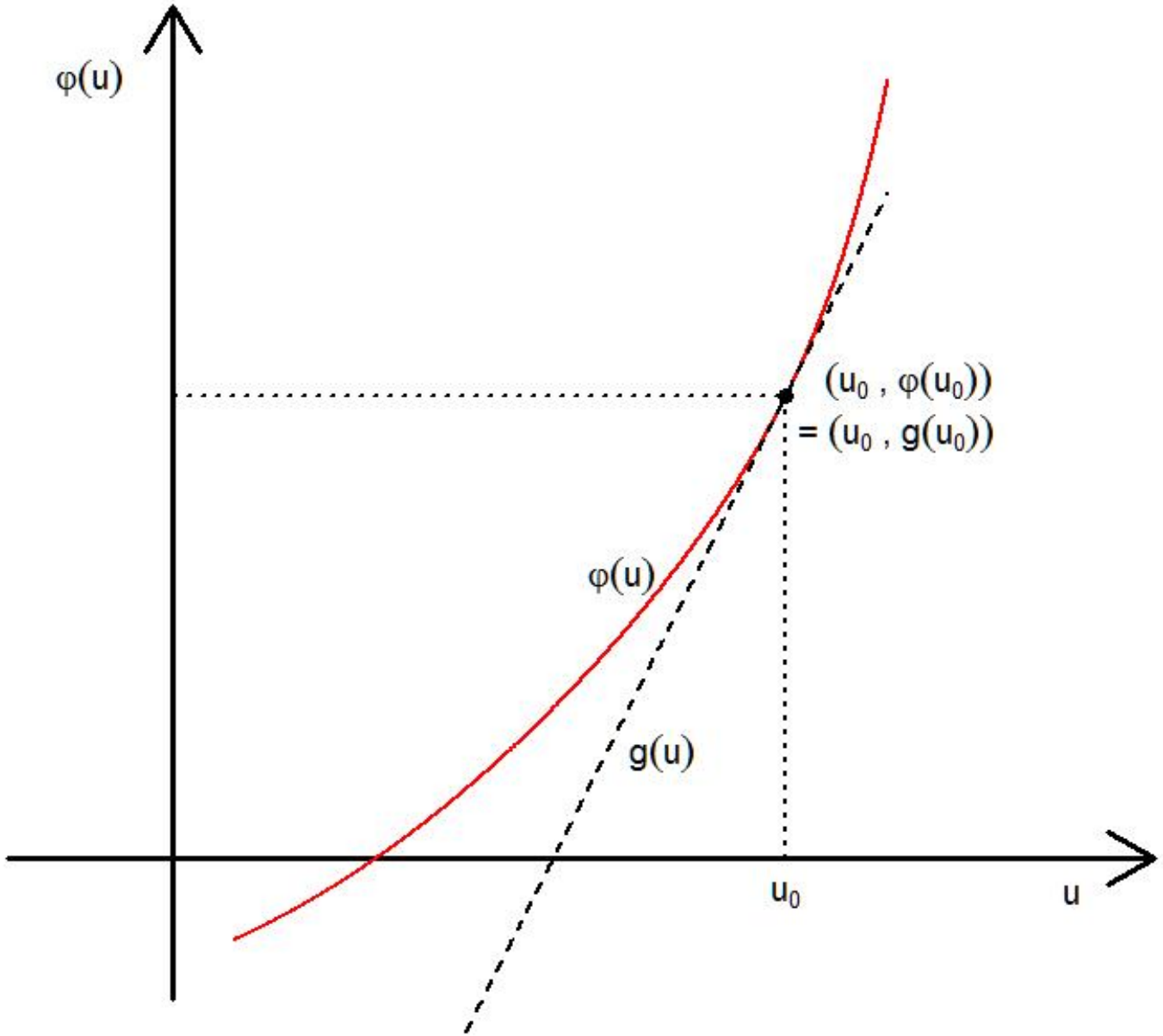


凸函数的一阶泰勒多项式

```
>  x<-seq(-14,14,0.01)
>  y<-seq(-14,14,0.01)
>  plot(x,y,type="n",axes=F,xlab="",ylab="")
>  #----------------------axis----------------------------
>  arrows(-14,-10,14,-10,lty=19,col=1,lwd=2.5)
>  arrows(-10,-14,-10,14,lty=19,col=1,lwd=2.5)
...
>  #----------------------------points----------------------
>  points(x,y,type="p",pch=19)
>  #----------------------------text------------------------
>  text(12, -11, expression(u))
...
>  text(8, y+0.5, expression((u[0]~","~varphi(u[0]))))
>  text(7.7, y-1, expression("="~(u[0]~","~g(u[0]))))
>  title(main="The first-order Taylor polynomial of a convex functi
```

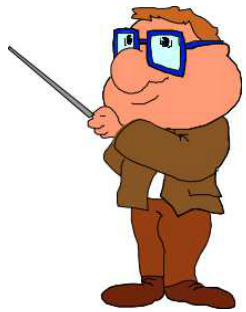


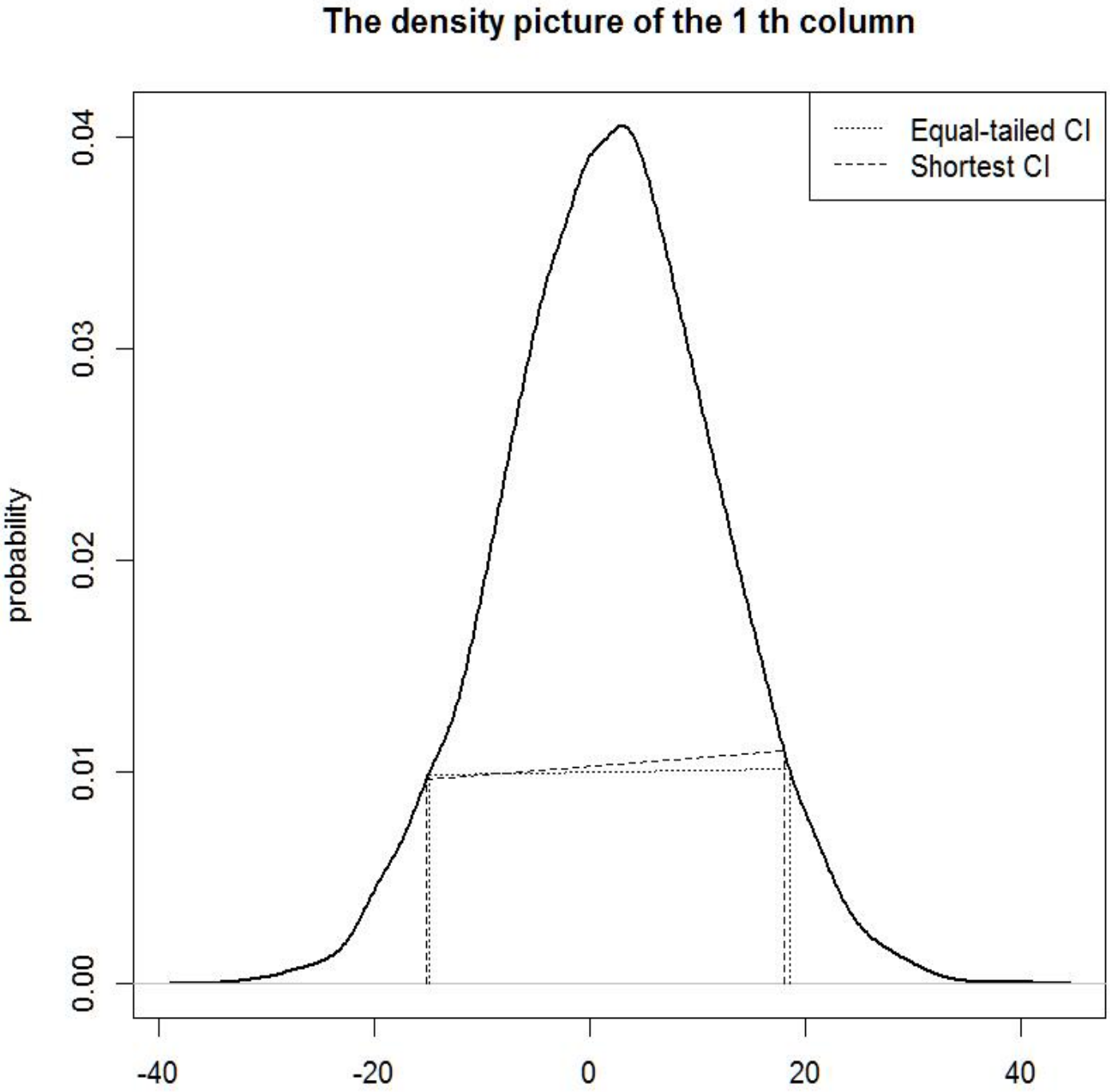
The first-order Taylor polynomial of a convex function



最短置信区间

```
>  find.shortest.CI.iidsample <- function(X, alpha, IND=1, ind=1, column.number=1)
+  {
+    # Function name: find.shortest.CI.iidsample(X, alpha, IND=1, ind=1, column.number=1)
+    # Input: X is the dataset
...
+       if (ind ==1)
+       {
+          plot(al, lower, type="l", lwd=2, xlim=c(0,alpha),ylim = c(min(lower)-0.5,
+               max(upper)+0.5), xlab = "", ylab="Confidence intervals")
+          title(paste(a,"% equal-tailed and shortest CIs of",column.number,"th column"))
+          lines(al, upper, type="l", lwd=2)
+          segments(al[indexxx],lower[indexxx],al[indexxx],upper[indexxx],lty=2,lwd=2)
+          points(al[indexxx],lower[indexxx],type="p", pch=19)
+          points(al[indexxx],upper[indexxx],type="p", pch=19)
+          segments(alpha/2,Xvsort[indexx2],alpha/2,Xvsort[G-indexx2+1],lty=3,lwd=2)
+          points(alpha/2,Xvsort[indexx2],type="p", pch=19)
+          points(alpha/2,Xvsort[G-indexx2+1],type="p", pch=19)
+          legend("topleft",legend=c("Equal-tailed CI","Shortest CI"),lty=c(3,2),lwd=c(2,2))
+       }
...
+    G <- apply(X, 2, shortest.CI)
+    rownames(G) <- c("Shortest.Lower","Shortest.Upper","Shortest.Width",
+                         "Equal-tailed.Lower","Equal-tailed.Upper","Equal-tailed.Width")
+    return(G)
+  }
>  X <- replicate(2,rnorm(1e4,2,10))
>  find.shortest.CI.iidsample(X, 0.10, IND=1, ind=2, column.number=1)
```



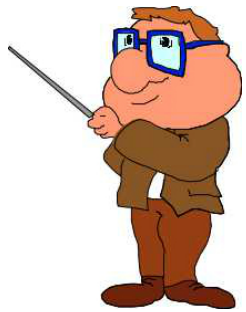

The density picture of the 1 th column
Equal-tailed CI
Shortest CI
probability
0.00
0.01
0.02
0.03
0.04
-40
-20
0
20
40

5. 网格作图
5.1 网格作图简介

- 什么是网格作图：在一个大图形中包含多个子图形
- 为什么要进行网格作图：显示更多的信息，对比图形等
- 一些高级作图命令得到的图形就是网格图，其中包含多个小图
- 例如前面的**pairs**和**coplot**函数的图形
- 但我们需要更灵活的方法进行网格作图

例

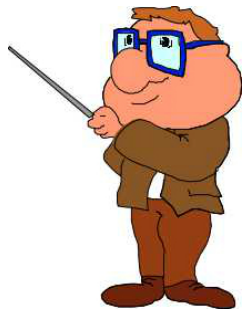
```
> data(mtcars)  # 使用 mtcars 数据集展示下面的例子

> plot(mtcars$wt, mtcars$mpg) # 所有数据的散点图
> plot(mtcars[mtcars$cyl==4, ]$wt, mtcars[mtcars$cyl==4,
+      ]$mpg) # 4缸汽车的散点图
> plot(mtcars[mtcars$cyl==6, ]$wt, mtcars[mtcars$cyl==6,
+      ]$mpg) # 6缸汽车的散点图
> plot(mtcars[mtcars$cyl==8, ]$wt, mtcars[mtcars$cyl==8,
+      ]$mpg) # 8缸汽车的散点图

> coplot(mpg~wt|as.factor(cyl), data=mtcars)
> # 同时画出4, 6, 8缸发动机的散点图
```



例

```
> data(mtcars)  # 使用 mtcars 数据集展示下面的例子

> plot(mtcars$wt, mtcars$mpg) # 所有数据的散点图
> plot(mtcars[mtcars$cyl==4, ]$wt, mtcars[mtcars$cyl==4,
+      ]$mpg) # 4缸汽车的散点图
> plot(mtcars[mtcars$cyl==6, ]$wt, mtcars[mtcars$cyl==6,
+      ]$mpg) # 6缸汽车的散点图
> plot(mtcars[mtcars$cyl==8, ]$wt, mtcars[mtcars$cyl==8,
+      ]$mpg) # 8缸汽车的散点图

> coplot(mpg~wt|as.factor(cyl), data=mtcars)
> # 同时画出4, 6, 8缸发动机的散点图
```

- 问题：我们能不能把4个子图形放入同一个大图形中？



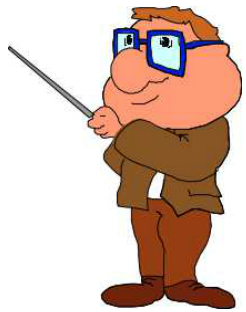
网格作图的一般方法：图形分割

对图形分割有三类方法：

- 使用作图参数设置
- 使用分割函数
- 使用网格作图包

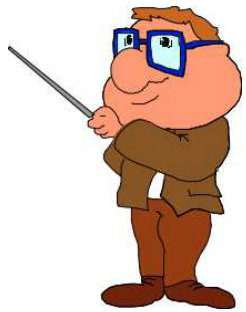
5.2 使用图形参数进行图形分割

- **mfrow**参数按照c(nr,nc)的向量，分割绘图窗口为*nr* 行*nc* 列的矩阵布局，按行次序使用各子窗
- **mfcol**参数按照c(nr,nc)的向量，分割绘图窗口为*nr* 行*nc* 列的矩阵布局，按列次序使用各子窗口



例

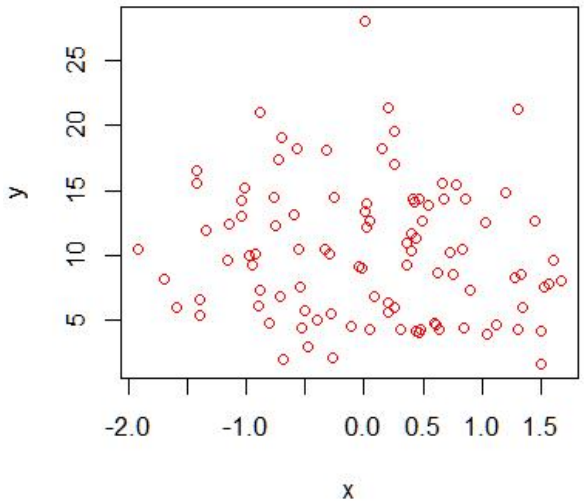
```
> x <- rnorm(100)
> y <- rchisq(100, df=10)
> par(mfrow=c(2,2))  # 把整个图形画板分割为2 行2列
> plot(x,y, col="red")
> boxplot(x)
> boxplot(y)
> plot(x,log(y), col="blue")
> par(mfrow=c(1,1)) # 还原设置
```

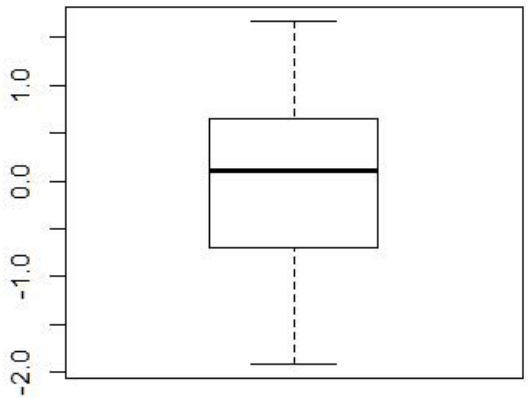


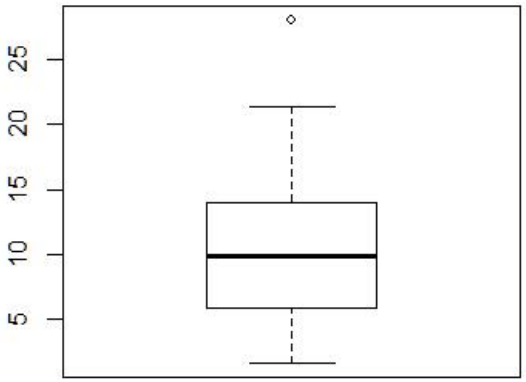
例

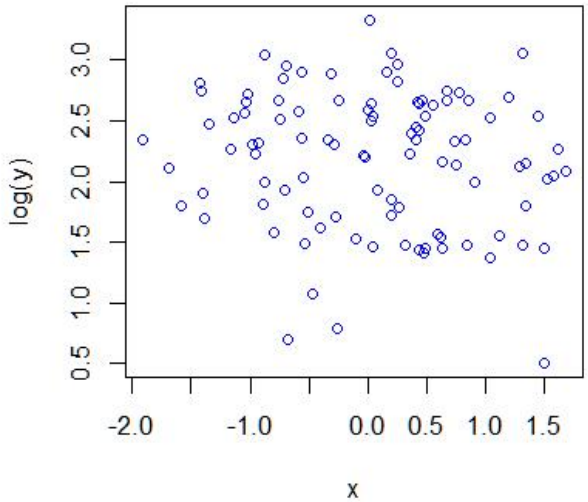
```
> x <- rnorm(100)
> y <- rchisq(100, df=10)
> par(mfrow=c(2,2))  # 把整个图形画板分割为2 行2列
> plot(x,y, col="red")
> boxplot(x)
> boxplot(y)
> plot(x,log(y), col="blue")
> par(mfrow=c(1,1)) # 还原设置
```

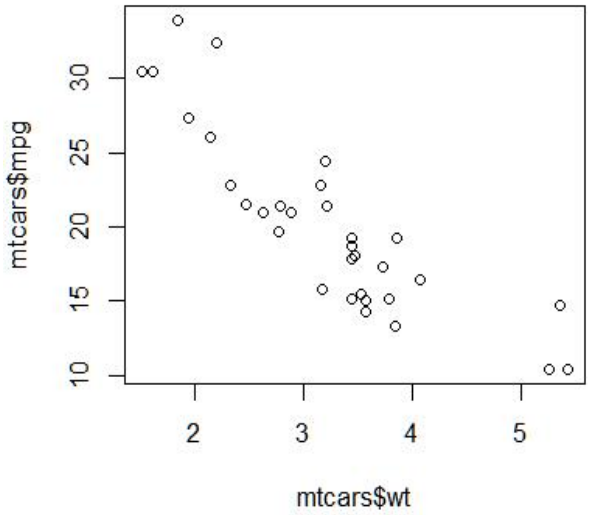
- **mfrow** 参数解决前面提出的问题：把4个子图形放入同一个大图形中

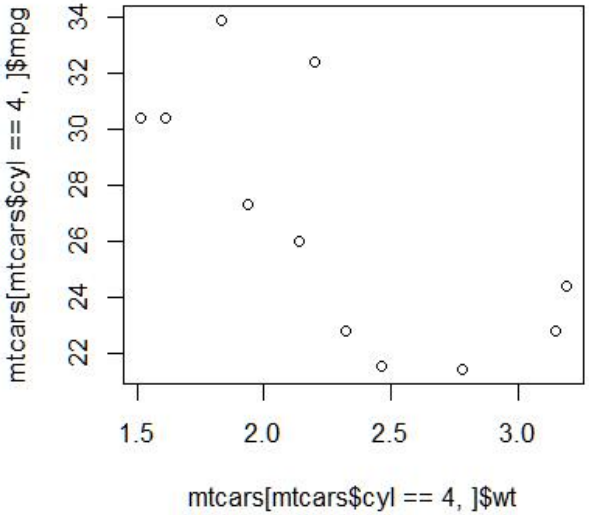

y
x
5
10
15
20
25
-2.0
-1.0
0.0
0.5
1.0
1.5

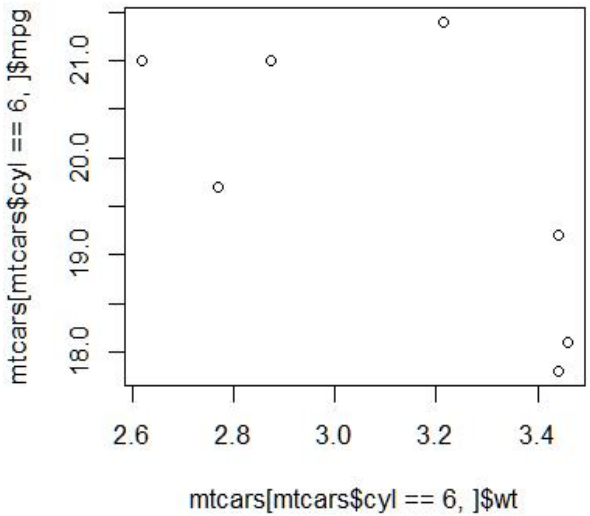

-2.0
-1.0
0.0
1.0

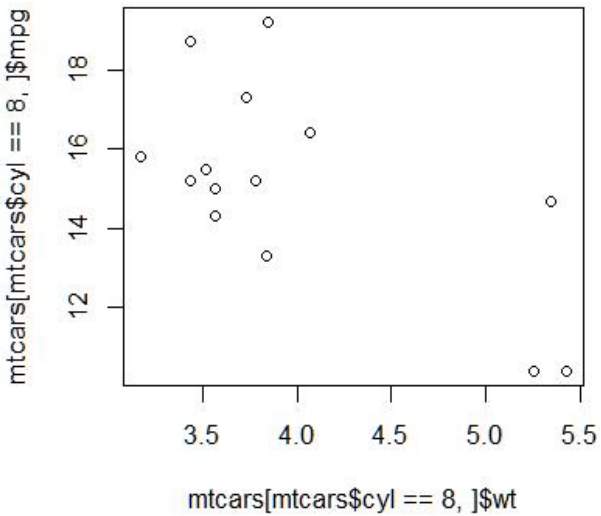

5
10
15
20
25

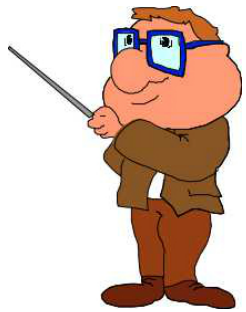

log(y)
x
0.5
1.0
1.5
2.0
2.5
3.0
-2.0
-1.0
0.0
0.5
1.0
1.5


mtcars$mpg
mtcars$wt
10
15
20
25
30
2
3
4
5


mtcars[mtcars$cyl == 4,]$mpg
mtcars[mtcars$cyl == 4,]$wt
22
24
26
28
30
32
34
1.5
2.0
2.5
3.0


mtcars[mtcars$cyl == 6,]$mpg
mtcars[mtcars$cyl == 6,]$wt
18.0
19.0
20.0
21.0
2.6
2.8
3.0
3.2
3.4


mtcars[mtcars$cyl == 8,]$mpg
mtcars[mtcars$cyl == 8,]$wt
12
14
16
18
3.5
4.0
4.5
5.0
5.5



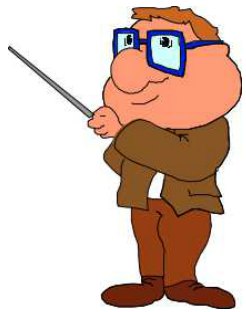
5.3 使用layout函数进行图形分割

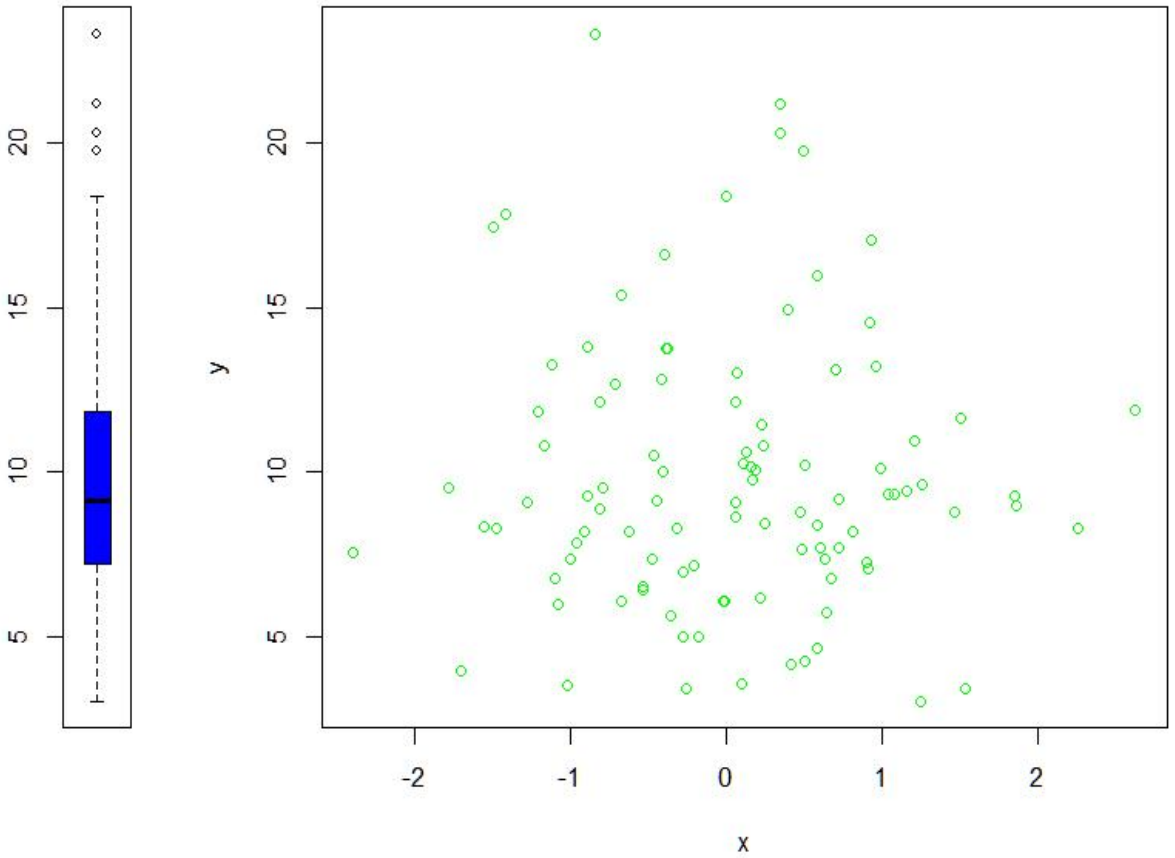
layout 函数：

- 参数是元素为整数的矩阵
- 把绘图页面按矩阵的形状进行分割
- 元素为0的地方不可用
- 元素相同的地方合并
- 可以使用**layout.show(n)**来查看当前页面分割情况
- 可以用选项**widths**和**heights**修改分割的宽和高

例

```
> layout(matrix(1:4, nrow=2))
> # 把画板平均分为2行2列, 等同于mfrow参数
> layout.show(4)
> layout(matrix(0:3, nrow=2))
> # 把画板分成3块
> layout.show(3)
> layout(matrix(c(1:3,3), nrow=2))
> # 画板分成大小不等的3块
> layout.show(3)

> layout(matrix(c(1,0,2,3), nrow=2), widths=c(1,4),
+        heights=c(3,1)) # 构建复合图形
> layout.show(3)

> x <- rnorm(100)
> y <- rchisq(100, df=10)
> boxplot(y, col="blue") # 变量y的垂直箱线图
> plot(x,y, col="green")  # x与y的散点图
> boxplot(x, horizontal=T, col="red") # 变量x的水平箱线图
```




5
10
15
20
y
-2
-1
0
1
2
x

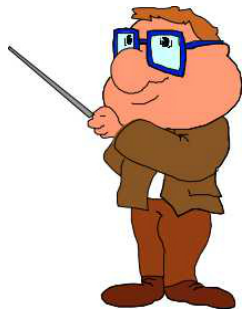

-2
-1
0
1
2

6. 网格图形系统

- 在R语言中，除了基础图形系统之外，R语言还提供了grid、lattice和ggplot2这三种图形系统，它们克服了R基础图形系统的低效性，大大扩展了的R语言的绘图能力。
- grid图形系统可以很轻松地控制图形基础单元，给予编程者创作图形极大的灵活性。
- lattice包通过一维、二维或三维条件绘图来对多元变量关系进行直观展示。
- ggplot2包则基于一种全面的图形语法，提供了一种全新的图形创建方式。
- grid只提供低层次的绘图函数，不提供生成统计图形以及完整绘图的函数，这里重点介绍Deepayan Sarkar的lattice和Hadley Wickham的ggplot2图形系统。

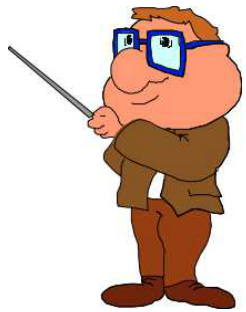
6.1 grid图形系统

- grid中有用来画基础输出—诸如线、文本和多边形—的函数，也有稍微高级的图形组件(如坐标轴)的函数
- 颜色、线型、字体以及其它图形输出可通过一组图形参数控制
- grid绘图系统并没有为图形输出提供预定的区域，但可基于视图概念定义区域，所有视图都带有一组坐标系，使得它可以用物理单位或相对于坐标轴尺寸的方法以及其他方法来确定输出的位置和大小
- 所有grid的输出都与当前页面视图有关，如需开启新的输出页面，需调用grid.newpage()函数。



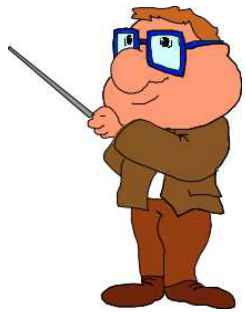
grid基本图形函数

生成输出的函数	描述
grid.move.to()	设置当前位置
grid.line.to()	从当前位置到新位置画一条直线，然后重置当前位置
grid.lines()	按顺序通过多个位置画出一条线
grid.polyline()	按顺序通过多个位置画出多条线
grid.segments()	在多对位置之间画出多条线
grid.xspline()	画出关于控制点的光滑曲线
grid.rect()	根据给定的位置和尺寸画出矩形
grid.roundrect()	根据给定的位置和尺寸画出圆角矩形
grid.circle()	根据给定的位置和半径画出圆
grid.polygon()	根据给定的顶点画出多边形
grid.path()	画出含有多个路径的多边形
grid.text()	根据给定的字符串、位置和方向画出文本
grid.raster()	画出位图
grid.curve()	画出两个端点之间的光滑曲线
grid.points()	根据给定的位置画出数据符号
grid.xaxis()	画出x轴
grid.yaxis()	画出y轴



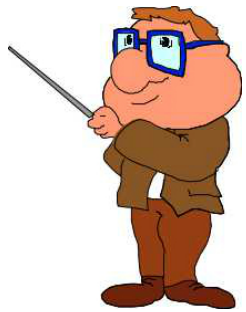
viewport简介

- viewport简单说就是图形中一块矩形区域，是在这个区域中进一步绘图的基础
- 新建一个空白的图形调用函数grid.newpage()
- 新建一个viewport调用函数viewport(x, y, width, height, angle, ...)
- 通过pushViewport()将对象push为当前viewport

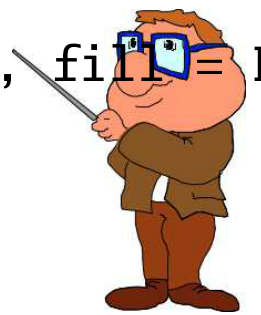


viewport例子

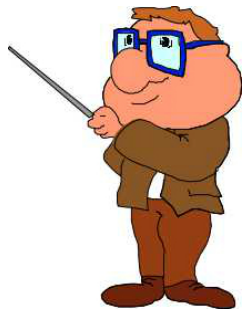
```
> library(grid)
>
> # viewport说明例子
> # 新建一个空白的图形
> grid.newpage()
> # 新建一个viewport
> vp <- viewport(x = 0.5, y = 0.5, width = 0.5,
+                height = 0.25, angle=45)
> # 现在图形中什么都没有，我们需要将对象vp push到图形中
> pushViewport(vp)
> # 此时图形仍是空白，我们可以画个矩形在vp中
> grid.rect()
> # 画出一个viewport的示例，带有一些辅助的说明。
> grid.show.viewport(viewport(x = 0.6, y = 0.6,
+                     w = unit(1, "inches"),
+                     h = unit(1, "inches")))
```



```
> # 连续push三个viewport到一个图形中
> grid.newpage()
> colvec <- c('red', 'green', 'blue')
> xvec <- c(0.3, 0.5, 0.7)
> for (i in 1 : 3) {
+     vp <- viewport(x = xvec[i], y = 0.5, width = 0.4,
+                    height = 0.8, gp = gpar(col = colvec[i]))
+     pushViewport(vp)
+     grid.rect()
+ }
> # 使用upViewport回到当前viewport的上一级viewport
> grid.newpage()
> colvec <- c('red', 'green', 'blue')
> xvec <- c(0.3, 0.5, 0.7)
> for (i in 1 : 3) {
+     vp <- viewport(x = xvec[i], y = 0.5, width = 0.4,
+                    height = 0.8,gp = gpar(col = colvec[i], fill = I
+     pushViewport(vp)
+     grid.rect()
+     upViewport()
+ }
```



```
> library(grid)
> roofvp <- viewport(x=0.5, y=5/12, width=1/3, height=1/6,
+ just=c("centre","bottom"), clip="on")
> pushViewport(roofvp)
> roof <- grid.circle(x=0.5, y=0, r=0.8, gp=gpar(fill="brown"),name="roof")
> grid.draw(roof); upViewport()
> grid.polygon(x=c(1/3,2/3,15/24,9/24), y=c(0,0,1/2-1/12,5/12), gp=gpar(fill="grey"))
> vp <- viewport(x=0.5, y=2/3, width=1/6, height=2/6, just= c("right","bottom"))
> blade1 <- gTree (children= gList(rectGrob(x=c(rep(0,6), rep(0.5,6)),
+ y=c(rep(0:5/6,2)), width=1/2, height=1/6, just=c("left","bottom"),
+ gp=gpar(col="grey",lwd=3, fill="orange"),vp=vp),
+ rectGrob(gp=gpar(col="white", lwd=3), vp=vp)), name="blade1")
> blade2 <- editGrob(blade1, vp= viewport(angle=90), name="blade2")
> blade3 <- editGrob(blade1, vp= viewport(angle=180), name="blade3")
> blade4 <- editGrob(blade1, vp= viewport(angle=270), name="blade4")
> segments <- segmentsGrob(x0=c(0,0.5), y0=c(0.5,0), x1=c(1,0.5),
+ y1=c(0.5,1), gp=gpar(col="brown",lwd=10))
> fengche <- gTree(children= gList(blade1,blade2,blade3,blade4,segments),
+ vp=viewport(angle=45), name="fengche")
> grid.draw(fengche)
> # We can rotate the blades as follows:
> for (i in 1:10000) grid.edit("fengche", vp=viewport(angle=i/1))
```



6.2 lattice图形系统

- lattice图的默认外观在一些情况下更好看
- lattice中的绘图元素排布更自动化
- 图例可以由lattice自动生成，无需用户自己确认图中图例和颜色以及数据符号一致
- lattice绘图函数有几种高效的方法用于扩展
- lattice函数的输出时grid输出，具备很多grid的特性，用于注释、编辑以及保存图像输出

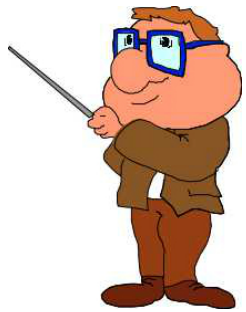


Table: lattice 包中常见的图形类型函数

图形类型	函数	表达式
三维等高线图	contourplot()	$z \sim x * y$
三维水平图	levelplot()	$z \sim y * x$
三维散点图	cloud()	$z \sim x * y \mid A$
三维线框图	wireframe()	$z \sim y * x$
条形图	barchart()	$x \sim A$或$A \sim x$
箱线图	bwplot()	$x \sim A$或$A \sim x$
点图	dotplot()	$\sim x \mid A$
直方图	histogram()	$\sim x$
核密度图	densityplot()	$\sim x \mid A * B$
平行坐标图	parallel()	dataframe
散点图	xyplot()	$y \sim x \mid A$
散点图矩阵	splom()	dataframe

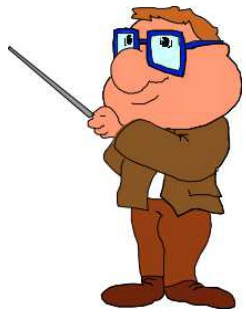
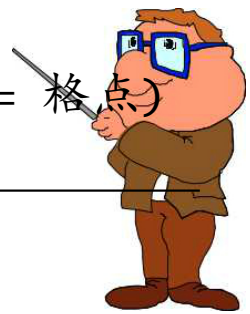


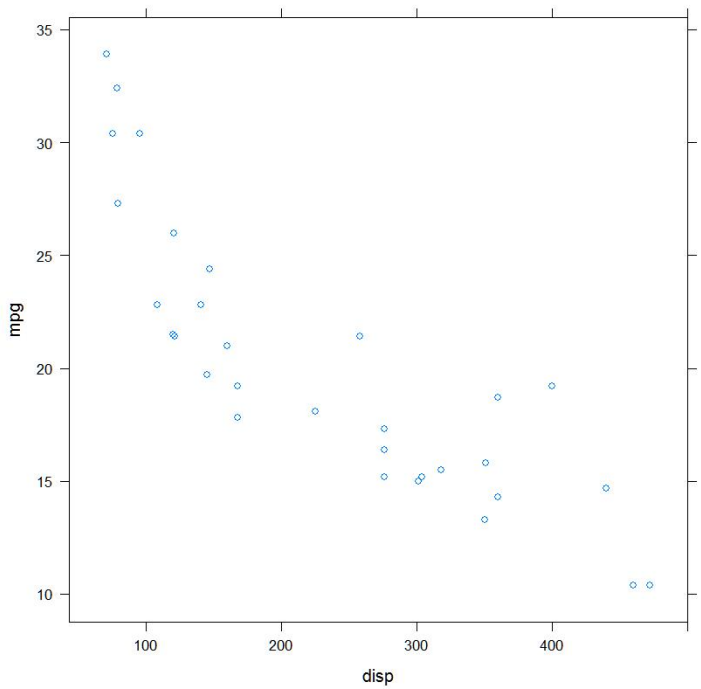
Table: lattice 包中图形函数常见的配置选项

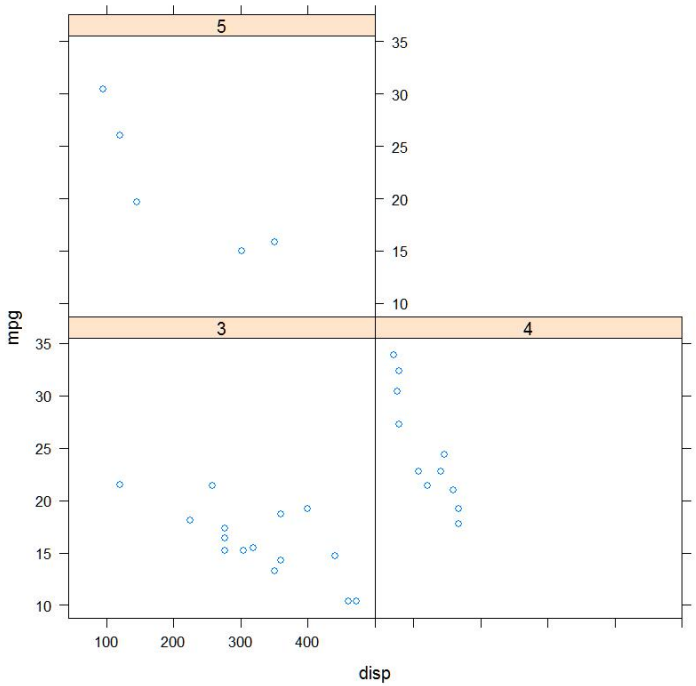
选项	描述
aspect	数值，设定每个面板中图形的宽高比
col、pch、lty、lwd	向量，分别设定图形中的颜色、符号、线条类型和线宽
groups	用来分组的变量(因子)
index.cond	列表，设定面板的展式顺序
key(或auto.key)	函数，添加分组变量的图例符号
layout	两元素数值型向量，设定面板的摆放方式(行数和列数)； 如有需要，可添加第三个元素，以指定页数
main、sub	字符型向量，设定主标题和副标题
panel	函数，设定每个面板要生成的图形
scales	列表，添加坐标轴标注信息
strip	函数，设定面板条带区域
split、position	数值型向量，在一页上绘制多幅图形
type	字符型向量，设定一个或多个散点图的绘图参数 (如p=点、l= 线、r= 回归、smooth= 平滑曲线、g= 格点)
xlab、ylab	字符型向量，设定横轴和纵轴标签



6.2.1 formula参数和条件多框图

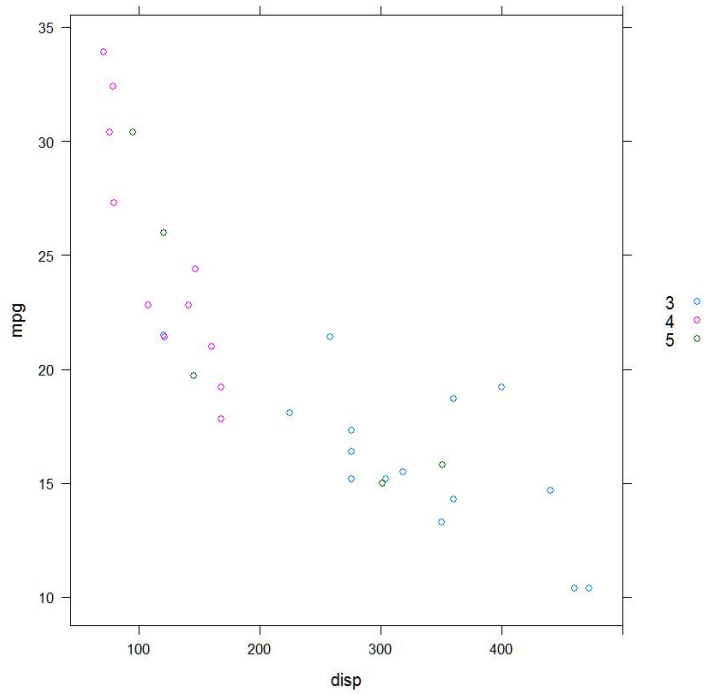
```
> library(lattice)
> data(mtcars)
> # 散点图
> xyplot(mpg~disp, data=mtcars)
> xyplot(mpg~disp|factor(gear), data=mtcars)
```

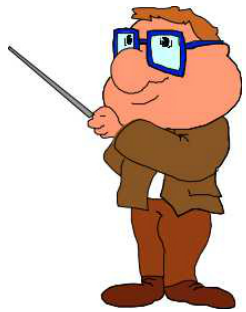




6.2.2 group参数和图例

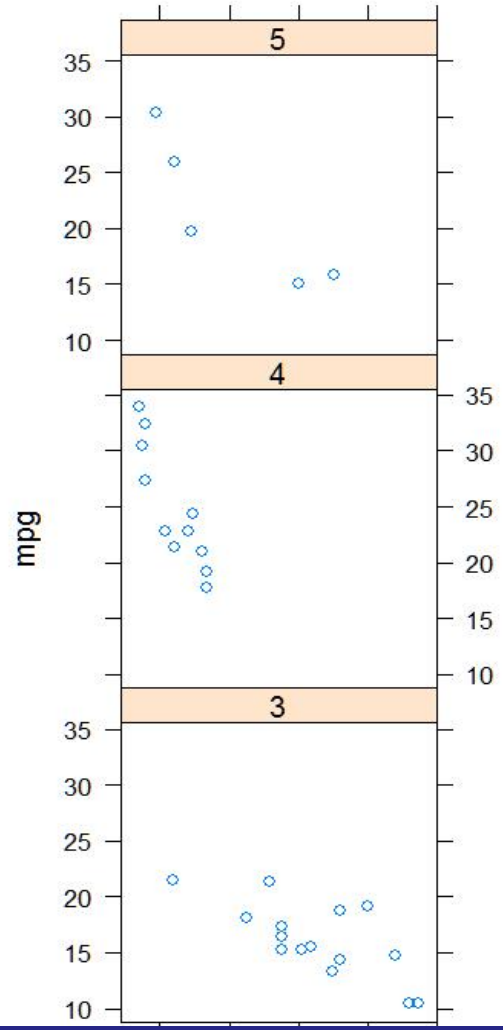
```
>   # group参数和图例
>   xyplot(mpg~disp, data=mtcars, group=gear,
+          auto.key=list(space="right"))
```

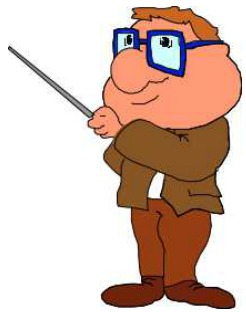




6.2.3 layout参数和排列绘图

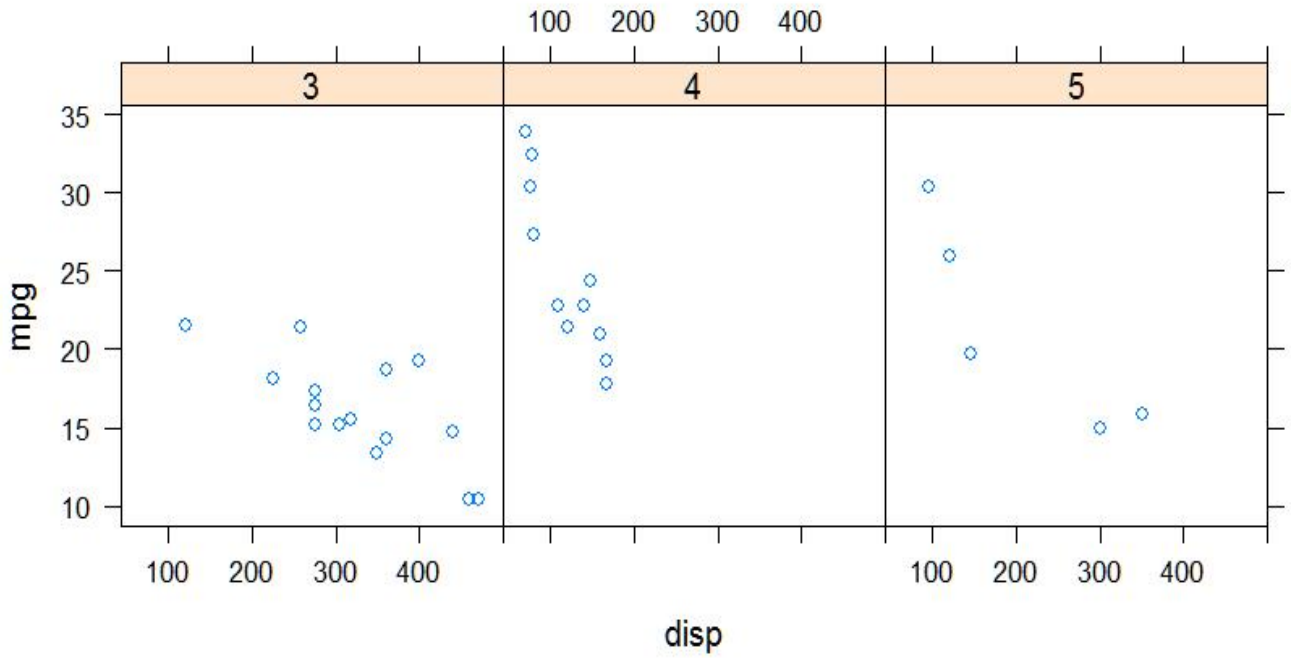
```
>   # layout参数和排列绘图
>   xyplot(mpg~disp|factor(gear), data=mtcars,
+          layout=c(1,3), aspect=1)
```

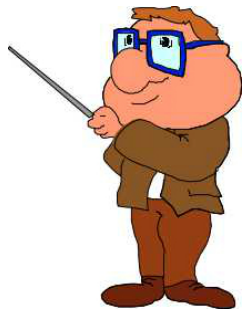




6.2.4 scales参数以及为坐标轴添加标签

```
>  # scales参数以及为坐标轴添加标签
>  xyplot(mpg~disp|factor(gear), data=mtcars,
+         layout=c(3,1), aspect=1,
+         scales=list(y=list(at=seq(10,30,5))),
+         ylab="miles per gallon",
+         xlab=expression(paste("displacement (", inch^3, ")")))
```





6.2.5 panel参数和图注释

```
>  # panel参数和图注释
>  xyplot(mpg~disp|factor(gear), data=mtcars,
+         layout=c(3,1), aspect=1,
+         panel=function(x,y,...) {
+               panel.lmline(x,y)
+               panel.xyplot(x,y,...)
+               panel.abline(h=29, lty="dashed")
+               panel.text(470, 29.5, "efficiency criterion",
+                          adj=c(1,0), cex=0.7)
+         })
```

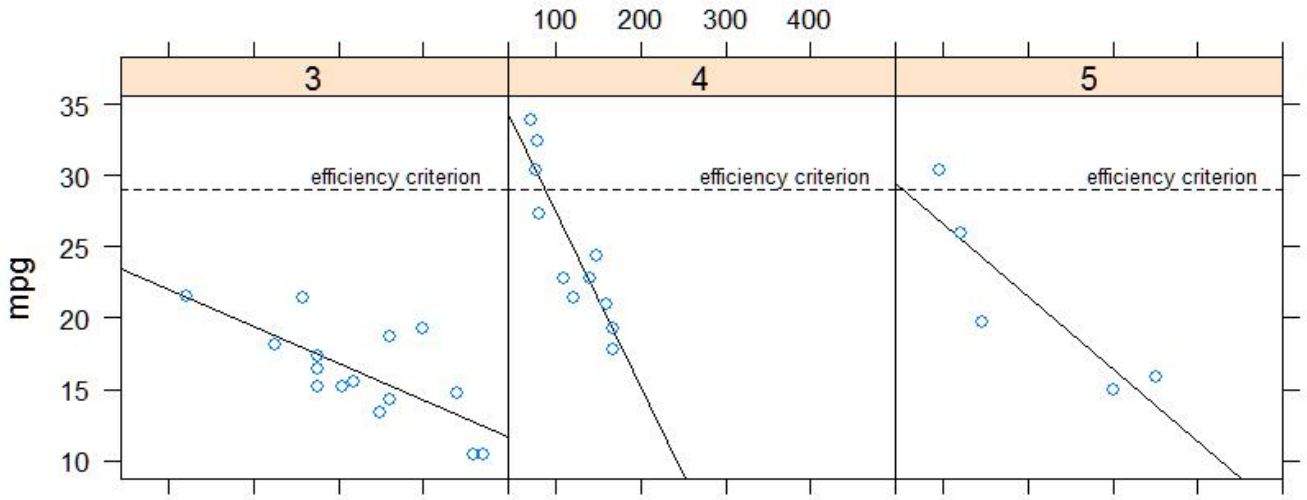


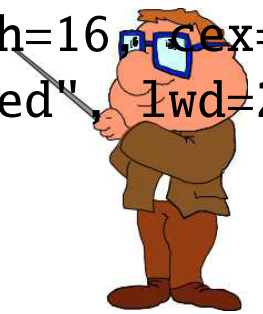
Table: 部分用来将图形输出添加到lattice图中面板的预设面板函数

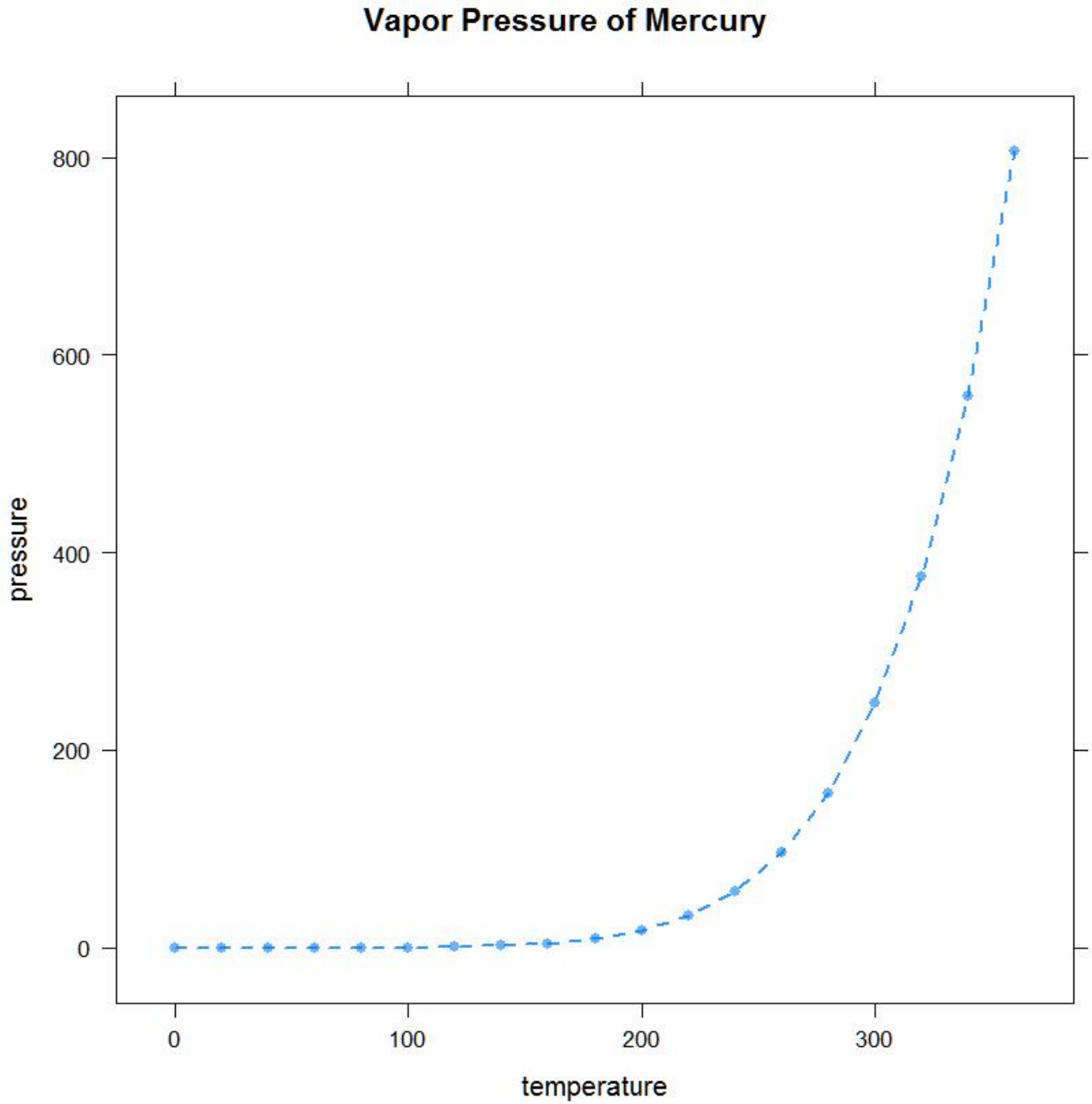
函数	描述
panel.points()	在位置(x, y)上画出数据符号
panel.line()	在多个位置(x, y)之间画线
panel.segments()	在位置(x_0, y_0)和(x_1, y_1)之间画线段
panel.arrows()	画出线段并在端点添加箭头
panel.rect()	在左下角(x_l, y_l) 和右下角(x_r, y_r)之间画矩形
panel.polygon()	根据向量(x,y)画出一个或多个多边形
panel.text()	在位置(x, y)画出文本
panel.abline()	以截断a和斜率b 画出直线
panel.curve()	画出expr给出的函数
panel.rug()	在x坐标或y坐标上画出刻度
panel.grid()	画出一个(灰色的)参考网格
panel.loess()	画出通过(x, y)的loess光滑曲线
panel.violin()	画出一个或多个提琴图
panel.smoothScatter()	画出(x, y)的2D光滑密度

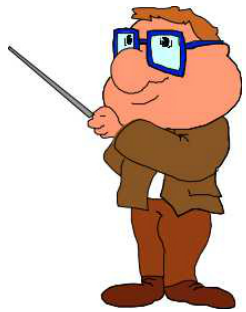
6.2.6 par.settings和图形参数

- 数据符号和颜色(例如col、lty和lwd)通过par.settings参数来设置
- 参数组plot.line包含了alpha、col、lty和lwd来控制不同数据位置上的线的透明度、颜色、线型和线宽
- 参数组plot.symbol包含了alpha、cex、col、pch和fill来控制数据符号的大小、形状和颜色
- 当前图形参数设定的值可以用trellis.par.get()函数来获得

```
>  # par.settings和图形参数
>  xyplot(pressure~temperature, data=pressure, type="o",
+         main="Vapor Pressure of Mercury",
+         par.settings=list(plot.symbol=list(alpha=0.6, pch=16, cex=
+                           plot.line=list(alpha=0.8, lty="dashed", lwd=
```

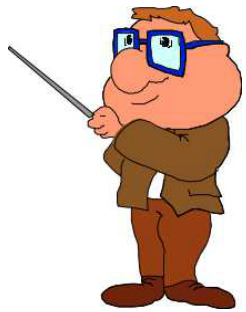



Vapor Pressure of Mercury
pressure
temperature
0
200
400
600
800
0
100
200
300



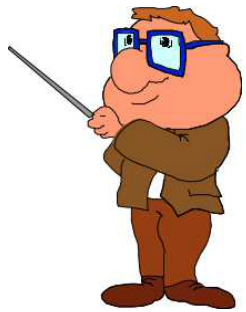
6.3 ggplot2简介

- ggplot2是由Hadley Wickham创建的一个十分强大的可视化R包



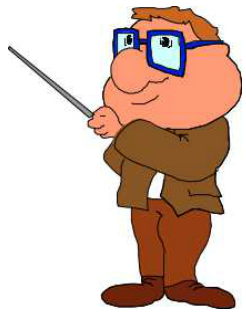
6.3 ggplot2简介

- ggplot2是由Hadley Wickham创建的一个十分强大的可视化R包
- 按照ggplot2的绘图理念，Plot(图)= data(数据集)+ Aesthetics(美学映射)+ Geometry(几何对象)：
  - data：数据集，主要是data frame
  - Aesthetics：美学映射，比如将变量映射给x，y坐标轴，或者映射给颜色、大小、形状等图形属性
  - Geometry：几何对象，比如柱形图、直方图、散点图、线图、密度图等



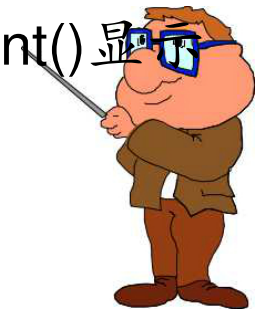
6.3 ggplot2简介

- ggplot2是由Hadley Wickham创建的一个十分强大的可视化R包
- 按照ggplot2的绘图理念，Plot(图)= data(数据集)+ Aesthetics(美学映射)+ Geometry(几何对象)：
  - data：数据集，主要是data frame
  - Aesthetics：美学映射，比如将变量映射给x，y坐标轴，或者映射给颜色、大小、形状等图形属性
  - Geometry：几何对象，比如柱形图、直方图、散点图、线图、密度图等
- 在ggplot2中有两个主要绘图函数：qplot()以及ggplot()
  - qplot()：快速绘图
  - ggplot()：此函数才是ggplot2的精髓，远比qplot()强大，可以一步步绘制十分复杂的图形



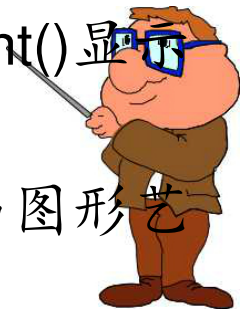
6.3 ggplot2简介

- ggplot2是由Hadley Wickham创建的一个十分强大的可视化R包
- 按照ggplot2的绘图理念，Plot(图)= data(数据集)+ Aesthetics(美学映射)+ Geometry(几何对象)：
  - data：数据集，主要是data frame
  - Aesthetics：美学映射，比如将变量映射给x，y坐标轴，或者映射给颜色、大小、形状等图形属性
  - Geometry：几何对象，比如柱形图、直方图、散点图、线图、密度图等
- 在ggplot2中有两个主要绘图函数：qplot()以及ggplot()
  - qplot()：快速绘图
  - ggplot()：此函数才是ggplot2的精髓，远比qplot()强大，可以一步步绘制十分复杂的图形
- 由ggplot2绘制出来的ggplot图可以作为一个变量，然后由print()显示出来



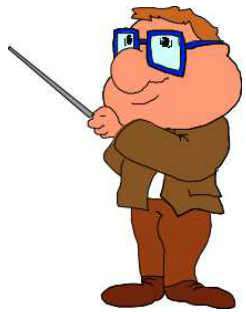
6.3 ggplot2简介

- ggplot2是由Hadley Wickham创建的一个十分强大的可视化R包
- 按照ggplot2的绘图理念，Plot(图)= data(数据集)+ Aesthetics(美学映射)+ Geometry(几何对象)：
  - data：数据集，主要是data frame
  - Aesthetics：美学映射，比如将变量映射给x，y坐标轴，或者映射给颜色、大小、形状等图形属性
  - Geometry：几何对象，比如柱形图、直方图、散点图、线图、密度图等
- 在ggplot2中有两个主要绘图函数：qplot()以及ggplot()
  - qplot()：快速绘图
  - ggplot()：此函数才是ggplot2的精髓，远比qplot()强大，可以一步步绘制十分复杂的图形
- 由ggplot2绘制出来的ggplot图可以作为一个变量，然后由print()显示出来
- 更多细节，参考Hadley Wickham的书《ggplot2：数据分析与图形艺术》



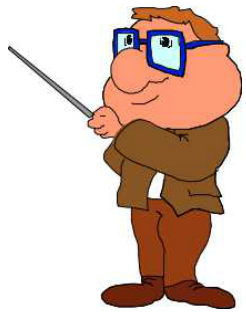
6.3.1 为什么需要ggplot2

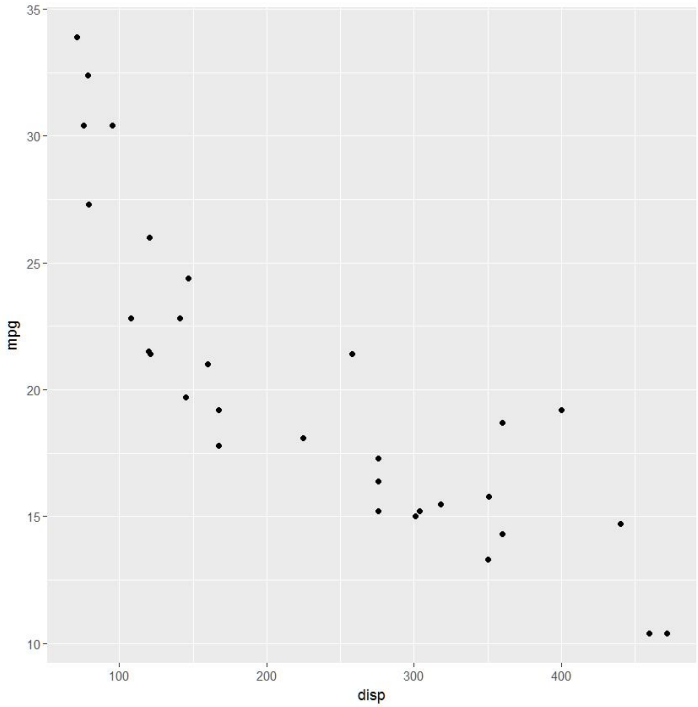
- 许多ggplot2生成的图和传统绘图以及lattice图形系统的图很相似，但ggplot2具有以下优点：
  - 图的默认外形已经根据视觉感知仔细挑选过，对一些人来说，ggplot2的风格或许会比lattice风格更具有吸引力
  - 绘图组件的排布和图例的选取都是自动化的额，相比lattice，ggplot2的灵活性更容易理解，但更复杂
  - ggplot2提供了一门强大的语言，能够简洁地表达多种图形
  - ggplot2使用grid来做渲染，使得注释、编辑和嵌入ggplot2 输出更具灵活性

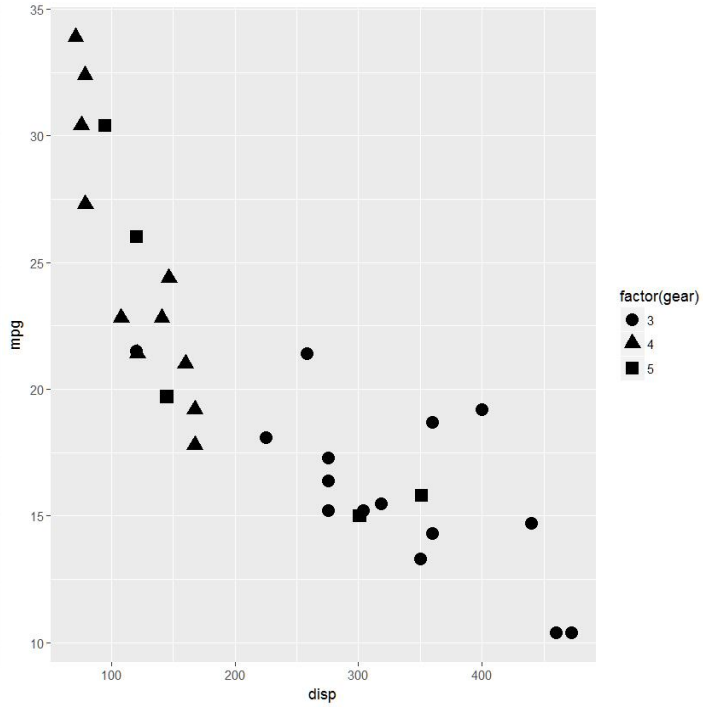


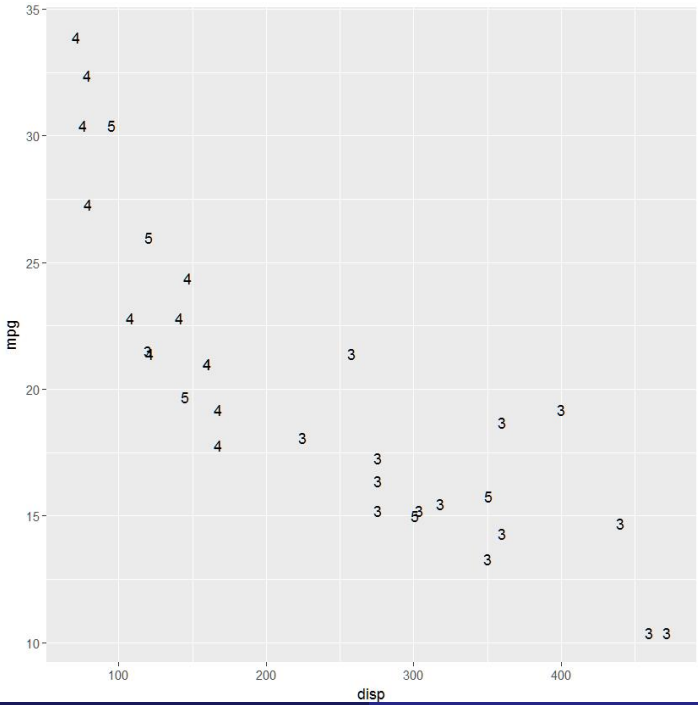
6.3.2 几何对象和图形属性

```
> library(rlang)
> library(ggplot2)
> # 生成一个新的“ggplot”对象
> p <- ggplot(mtcars)
> # 几何对象和图形属性
> p + geom_point(aes(x=disp, y=mpg))
> # gear为数值型变量，需转化为分类变量
> p + geom_point(aes(x=disp, y=mpg, shape=factor(gear)), size=4)
> p + geom_text(aes(x=disp, y=mpg, label=gear))
> lmcoef <- coef(lm(mpg~disp, mtcars))
> p + geom_point(aes(x=disp, y=mpg)) +
+     geom_abline(intercept=lmcoef[1], slope=lmcoef[2])
```




mpg
disp


mpg
disp
factor(gear)
3
4
5


mpg
disp

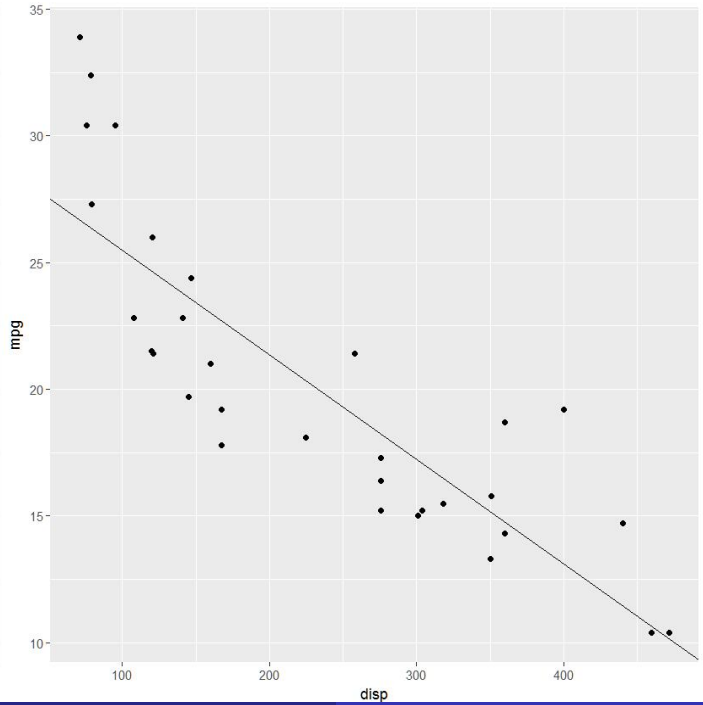

mpg
disp

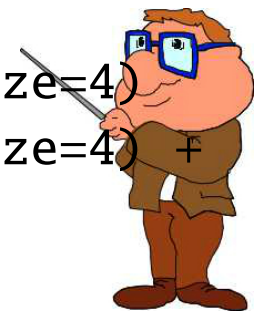
Table: ggplot 中部分常用的集合对象和对应的图形属性

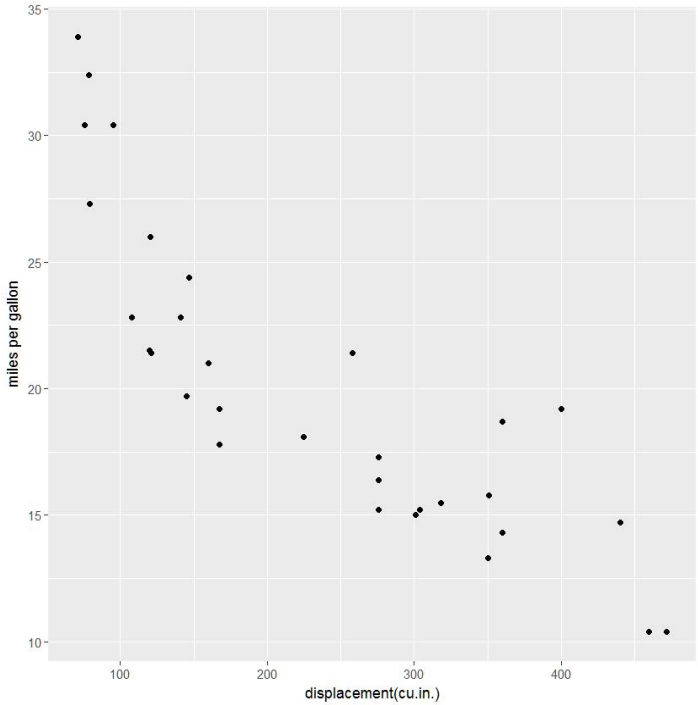
几何对象	描述	图形属性
geom_point()	数学符号	x,y,shape,fill
geom_line()	直线(按x排序)	x,y,linetype
geom_path()	直线(按原始顺序)	x,y,linetype
geom_text()	文本标签	x,y,label,angle,hjust,vjust
geom_rect()	矩形	xmin,xmax,ymin,ymax,fill,linetype
geom_polygon()	多边形	x,y,fill,linetype
geom_segment()	线段	x,y,xend,yend,linetype
geom_bar()	条状图	x,fill,linetype,weight
geom_histogram()	直方图	x,fill,linetype,weight
geom_boxplot()	箱线图	x,y,fill,weight
geom_density()	密度图	x,y,fill,linetype
geom_contour()	等高线图	x,y,fill,linetype
geom_smooth()	光滑曲线	x,y,fill,linetype
ALL		color,size,group

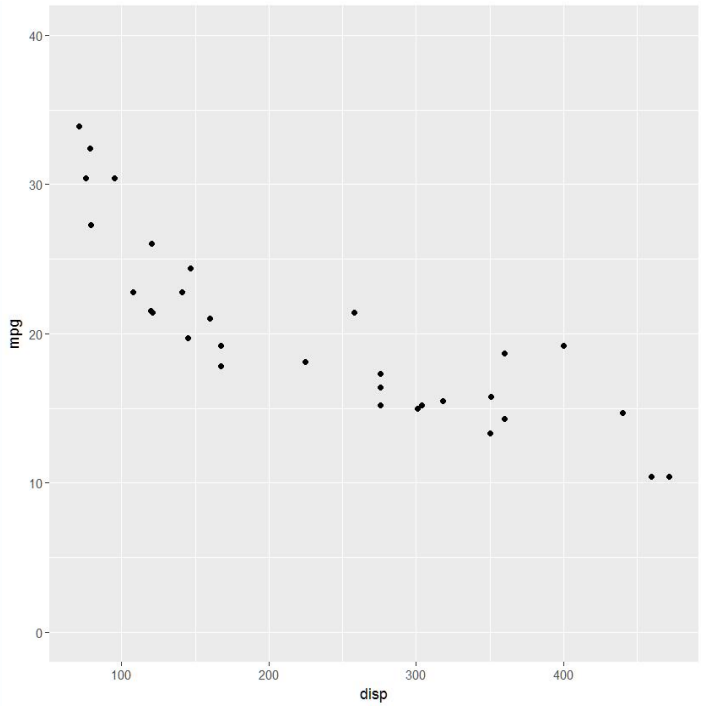
6.3.3 标度

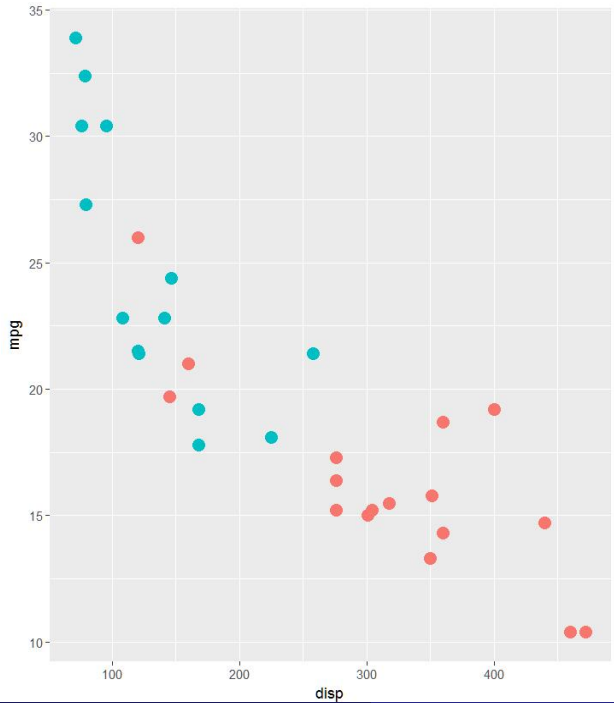
- 在ggplot2中，标度与图中的坐标和图例紧密结合在一起
- ggplot2可以自动生成标度元素—x轴和y轴
- 使用scale_x_continuous() 和scale_y_continuous() 可以显示设置坐标轴，覆盖原坐标轴
- scale_color_manual() 可以指定对应变量的颜色

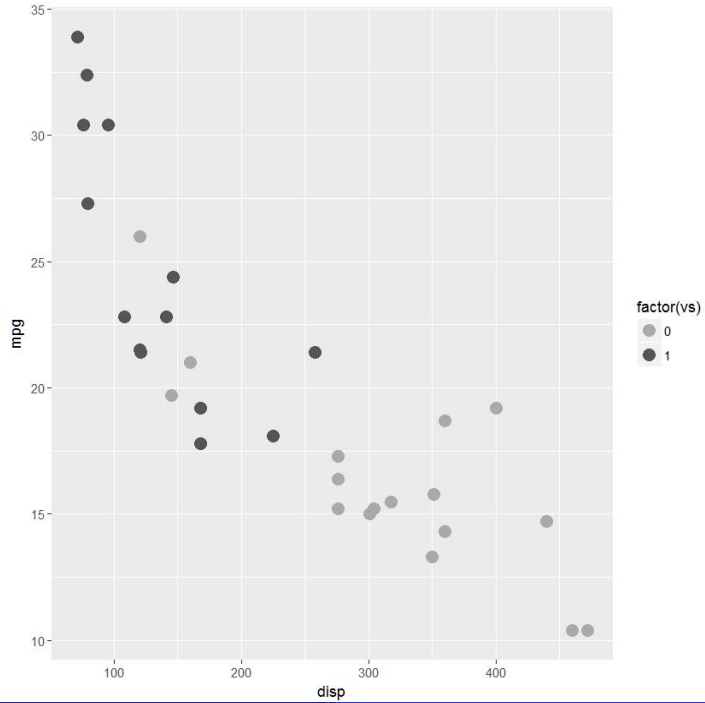
```
>  # 标度
>  p + geom_point(aes(x=disp, y=mpg)) +
+      scale_x_continuous(name="displacement(cu.in.)") +
+      scale_y_continuous(name="miles per gallon")
>  p + geom_point(aes(x=disp, y=mpg)) +
+      scale_y_continuous(limits=c(0, 40))
>  p + geom_point(aes(x=disp, y=mpg, color=factor(vs)), size=4)
>  p + geom_point(aes(x=disp, y=mpg, color=factor(vs)), size=4) +
+      scale_color_manual(values=c(gray(2/3), gray(1/3)))
```




miles per gallon
displacement(cu.in.)


mpg
disp


mpg
disp
factor(vs)
0
1


mpg
disp
factor(vs)
0
1

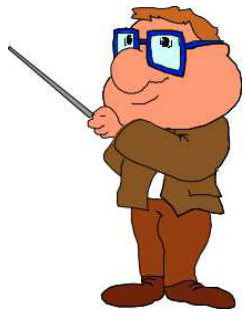
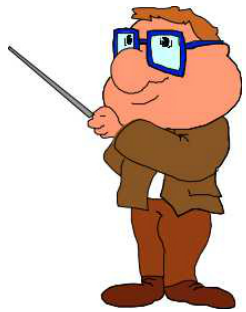


Table: ggplot 中部分常用标度

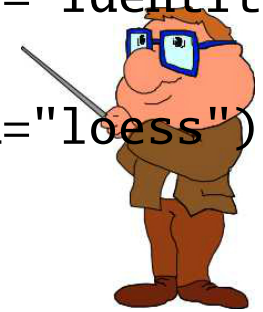
标度	描述	参数
scale_x_continuous()	连续坐标轴	expand,trans
scale_x_discrete()	分类坐标轴	
scale_x_date()	日期坐标轴	
scale_shape()	符号形状图例	
scale_linetype()	线型模式图例	
scale_color_manual()	符号/线颜色图例	values
scale_fill_manual()	符号/条带填充图例	values
scale_size()	符号尺寸图例	trans
ALL		name,breaks,labels,limits

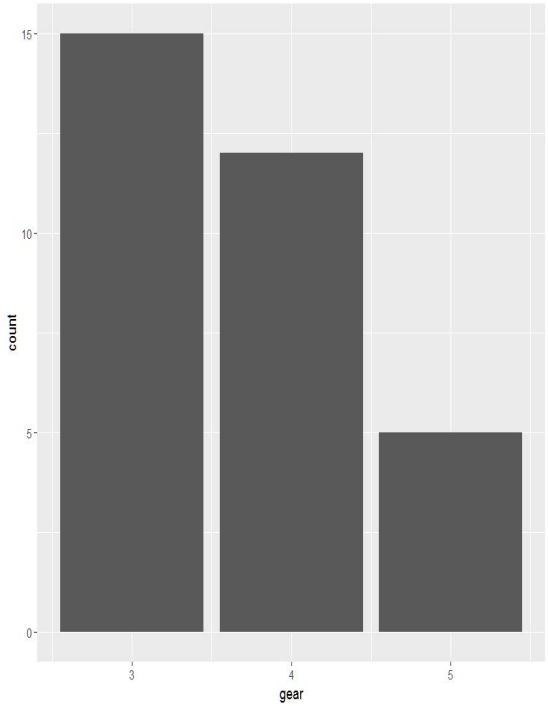


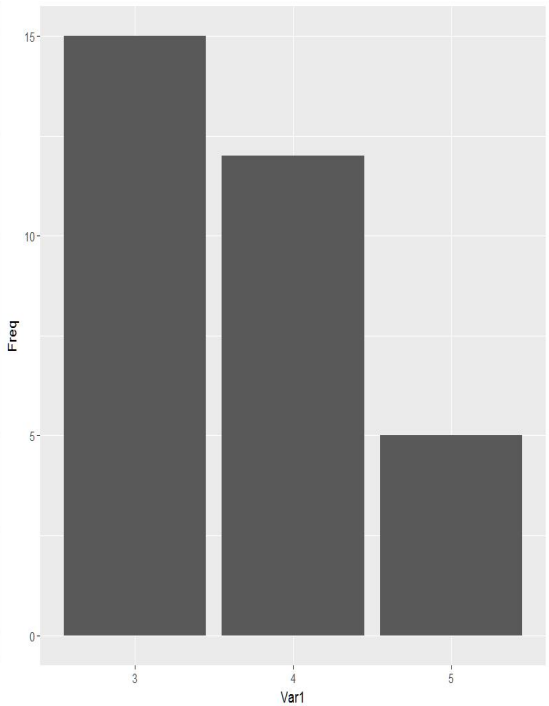
6.3.4 统计变换

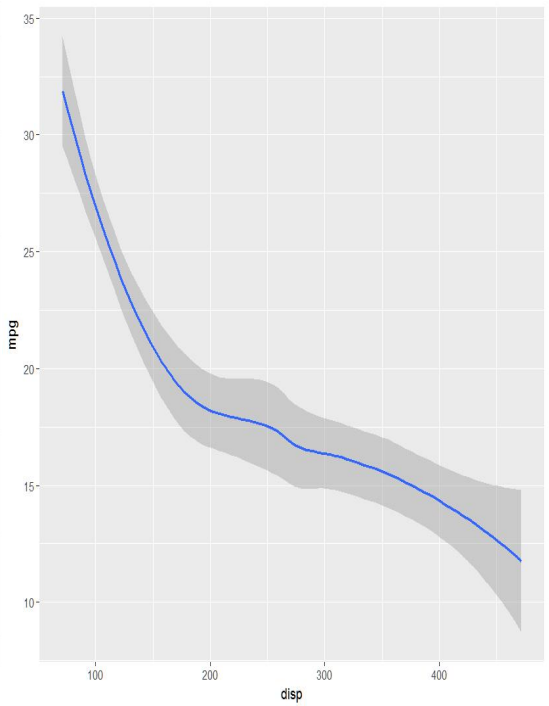
- 在ggplot2中，数据值通常被直接映射到图形属性设置中
- 但某些几何对象不能直接使用数据值，相反，数据值先经过某些统计变换再被映射称图形属性

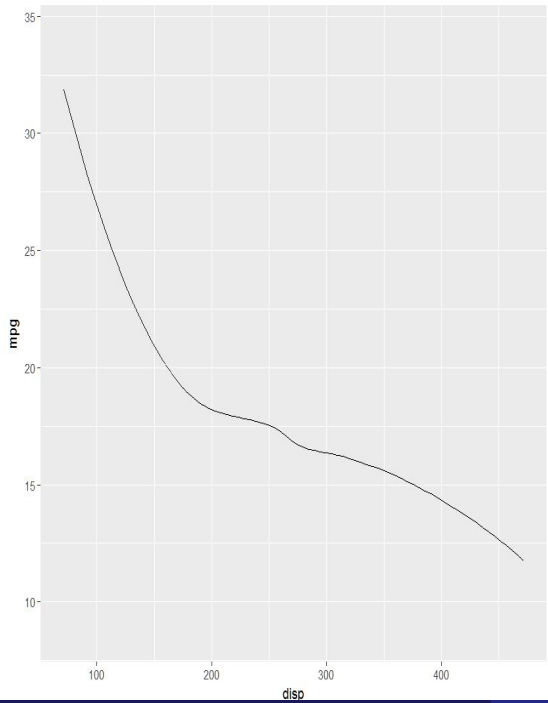
```
>  # 统计变换
>  p + geom_bar(aes(x=gear))
>  transCount <- as.data.frame(table(mtcars$gear))
>  head(transCount)
  Var1 Freq
1    3   15
2    4   12
3    5    5
>  ggplot(transCount) + geom_bar(aes(x=Var1, y=Freq), stat="identity
>  p + geom_smooth(aes(x=disp, y=mpg))
>  p + geom_line(aes(x=disp, y=mpg), stat="smooth", method="loess")
>  p + stat_smooth(aes(x=disp, y=mpg))
>  p + stat_smooth(aes(x=disp, y=mpg), method="lm")
```

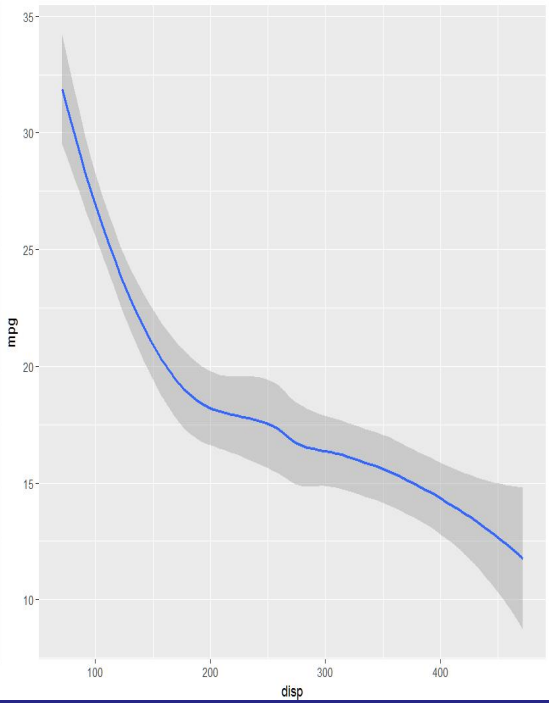



count
gear


Freq
Var1


mpg
disp


mpg
disp


mpg
disp

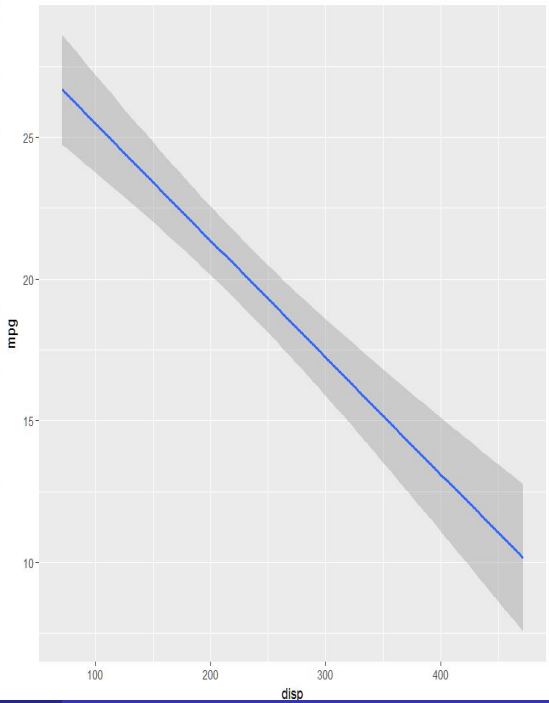
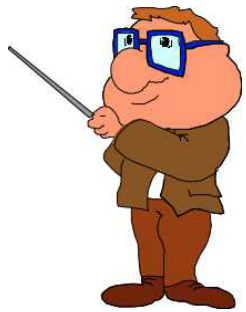

mpg
disp

Table: ggplot 中常用的统计变换

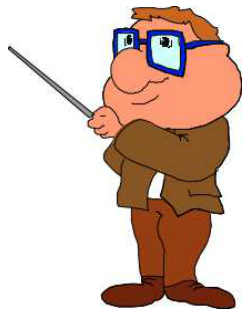
标度	描述	参数
stat_identity()	不作变换	--
stat_bin()	分组	binwidth
stat_smooth()	光滑化	method,se,n
stat_boxplot()	箱线图统计量	varwidth
stat_contours()	等高线	bins

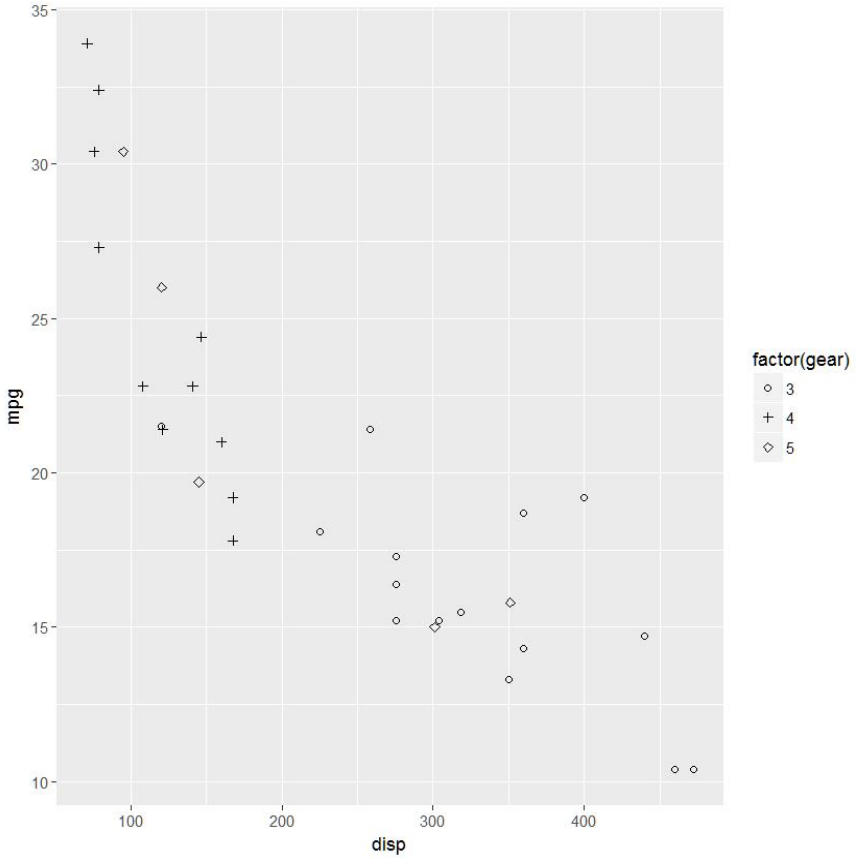


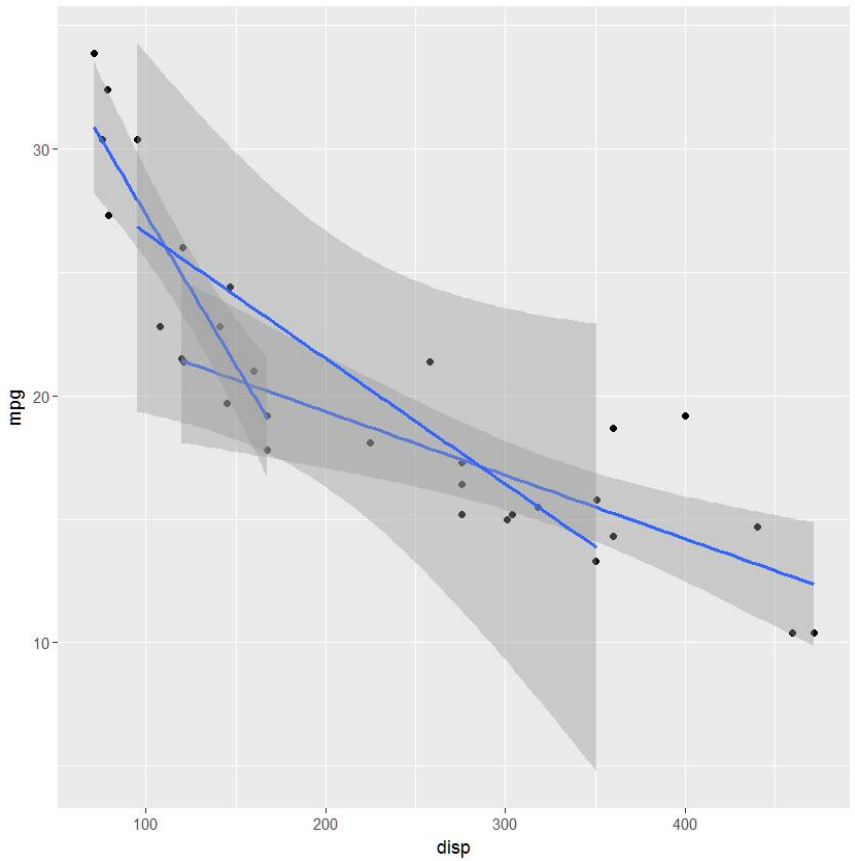
6.3.5 图形属性group

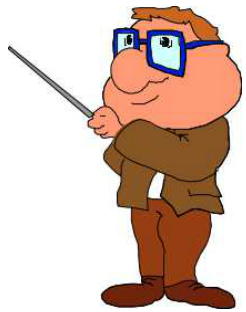
- ggplot2可以自动处理画多组数据的情况，并用scale_shape_manual() 控制数据符号的映射
- 图形属性group 可以达到强制分组的作用

```
>  # 图形属性group
>  p + geom_point(aes(x=disp, y=mpg, shape=factor(gear))) +
+      scale_shape_manual(values=c(1,3,5))
>  ggplot(mtcars, aes(x=disp, y=mpg)) + geom_point() +
+  stat_smooth(aes(group=factor(gear)), method="lm")
```



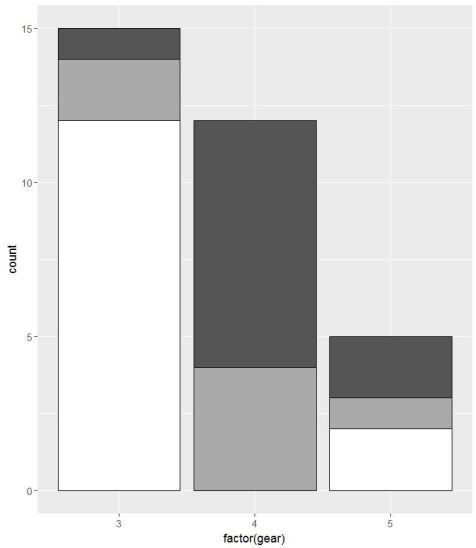

mpg
disp
factor(gear)
3
4
5

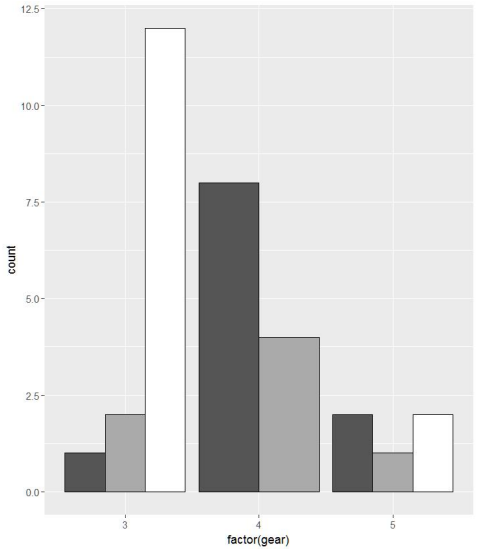

mpg
disp

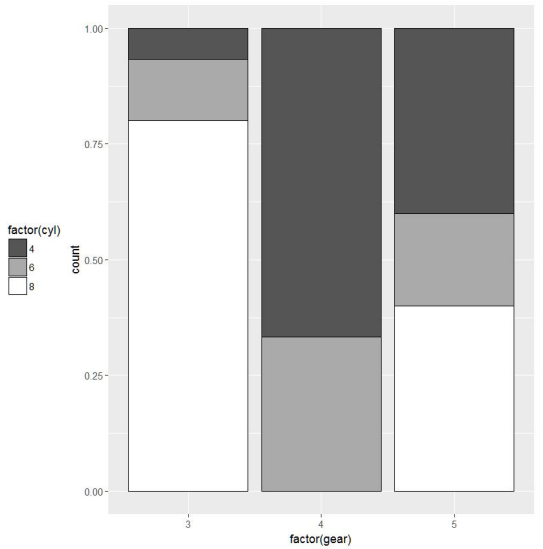


6.3.6 位置调整

```
>  # 位置调整
>  p + geom_bar(aes(x=factor(gear), fill=factor(cyl)), color="black"
+      scale_fill_manual(values=gray(1:3/3))
>  p + geom_bar(aes(x=factor(gear), fill=factor(cyl)),
+                color="black", position="dodge") +
+      scale_fill_manual(values=gray(1:3/3))
>  p + geom_bar(aes(x=factor(gear), fill=factor(cyl)),
+                color="black", position="fill") +
+      scale_fill_manual(values=gray(1:3/3))
```

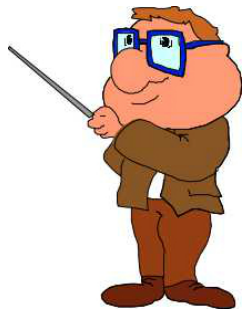


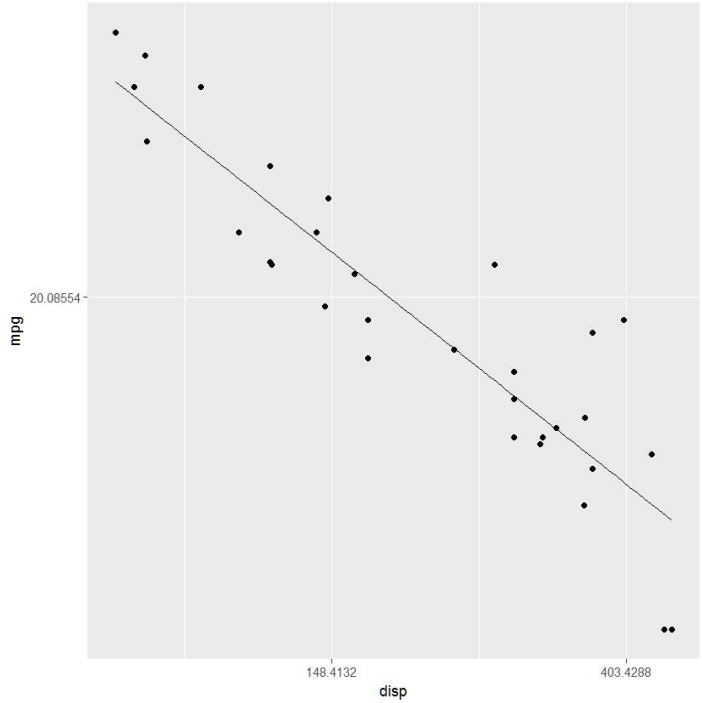


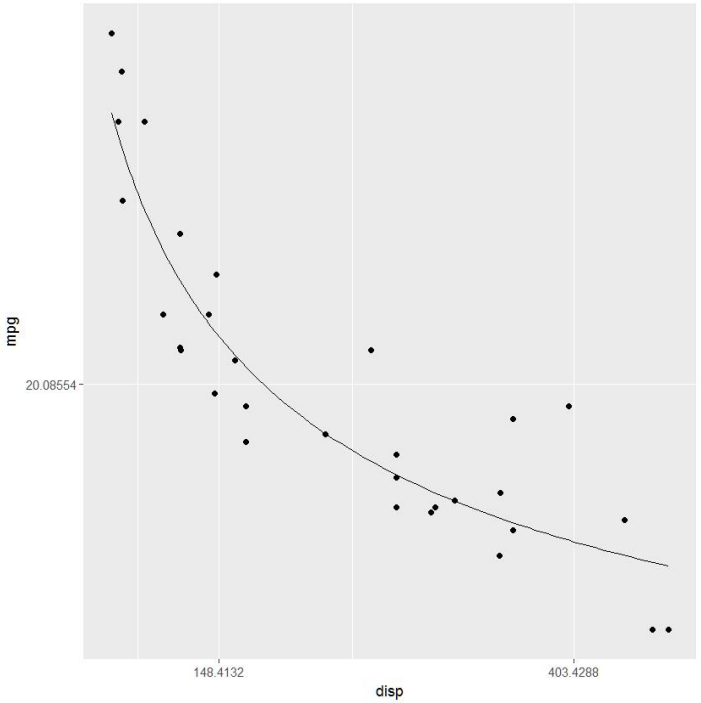


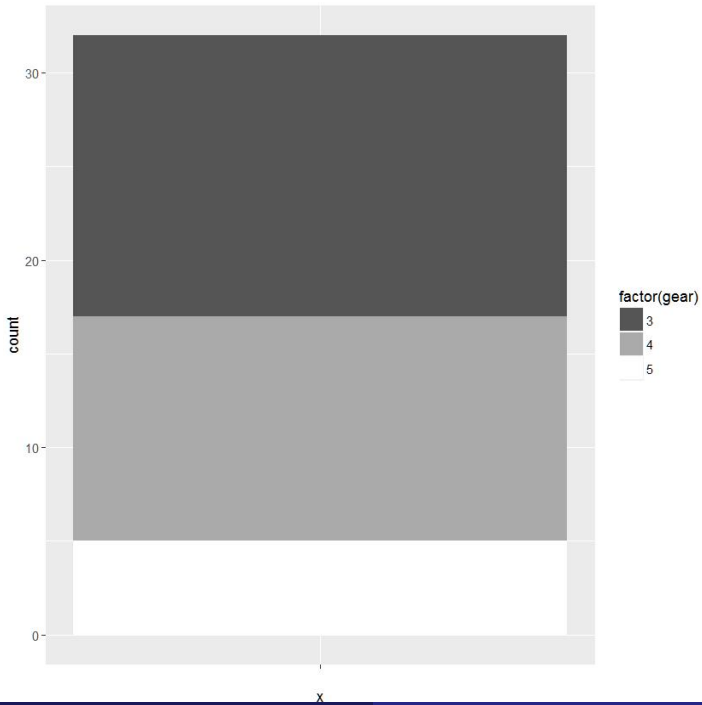
6.3.7 坐标变换

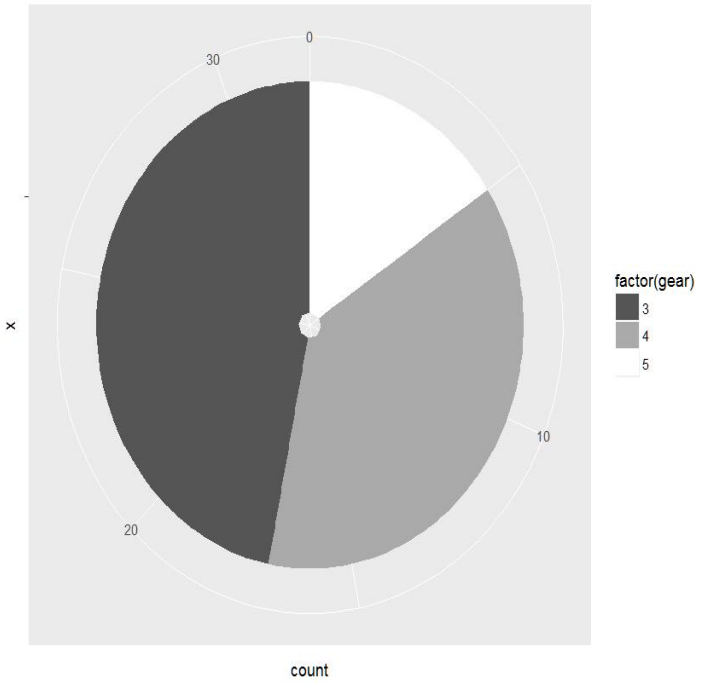
```
>  # 坐标变换
>  p + geom_point(aes(x=disp, y=mpg)) +
+      scale_x_continuous(trans="log") +
+      scale_y_continuous(trans="log") +
+      geom_line(aes(x=disp, y=mpg), stat="smooth", method="lm")
>  p + geom_point(aes(x=disp, y=mpg)) +
+      scale_x_continuous(trans="log") +
+      scale_y_continuous(trans="log") +
+      geom_line(aes(x=disp, y=mpg), stat="smooth", method="lm") +
+      coord_trans(x="exp", y="exp")
>  p + geom_bar(aes(x="", fill=factor(gear))) +
+      scale_fill_manual(values=gray(1:3/3))
>  p + geom_bar(aes(x="", fill=factor(gear))) +
+      scale_fill_manual(values=gray(1:3/3)) +
+      coord_polar(theta="y")
```




mpg
20.08554
148.4132
403.4288
disp


mpg
20.08554
148.4132
403.4288
disp

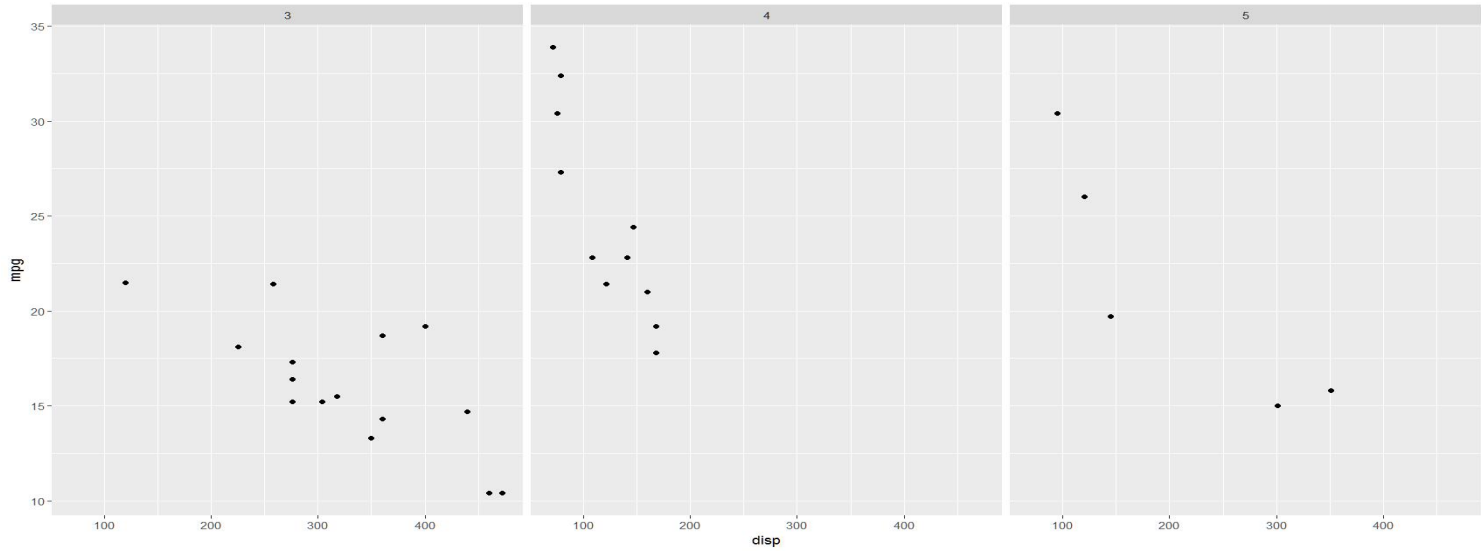

30
20
10
0
count
x
factor(gear)
3
4
5

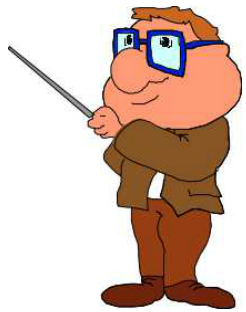

0
30
10
20
x
count
factor(gear)
3
4
5

6.3.8 分面

- 多面化意味着将数据分成几个子集，在同一页面中为每一个子集生成一个独立的图，通过facet_wrap()来为一个图添加小平面

```
>   # 分面
>   p + geom_point(aes(x=disp, y=mpg)) +
+       facet_wrap(~gear, nrow=1)
```

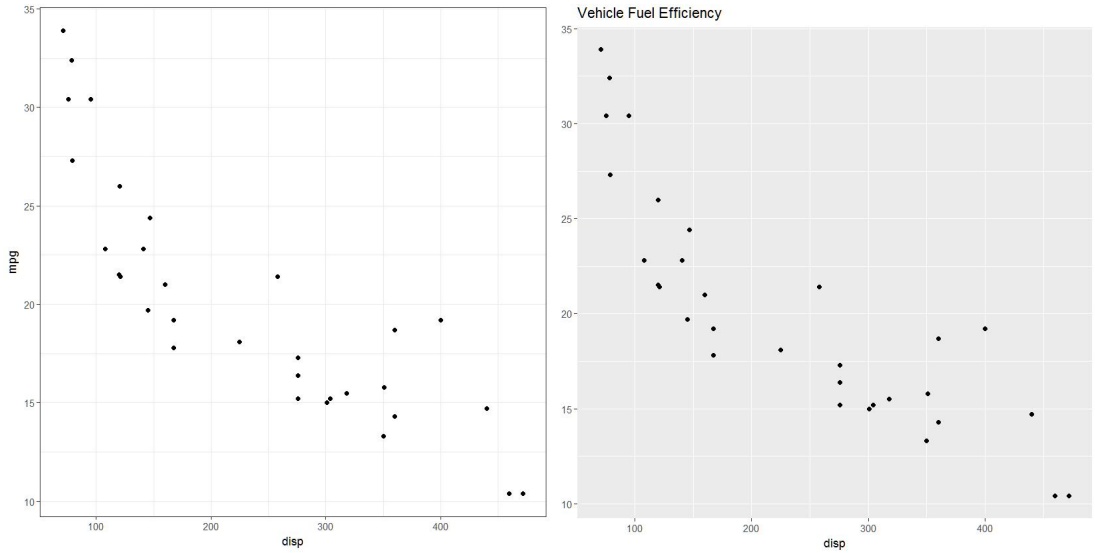


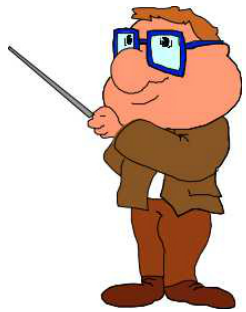


6.3.9 主题

- 在ggplot2中，数据元素与非数据元素的输出是分开的，控制非数据元素的图形参数集合被称为ggplot2的主题

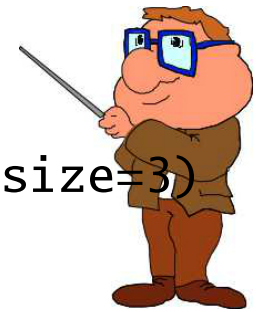
```
>  # 主题
>  p + geom_point(aes(x=disp, y=mpg)) +
+      theme_bw()
>  p + geom_point(aes(x=disp, y=mpg)) +
+      labs(y="", title="Vehicle Fuel Efficiency")
>  # 应该是版本问题暂时不能使标题居中
```

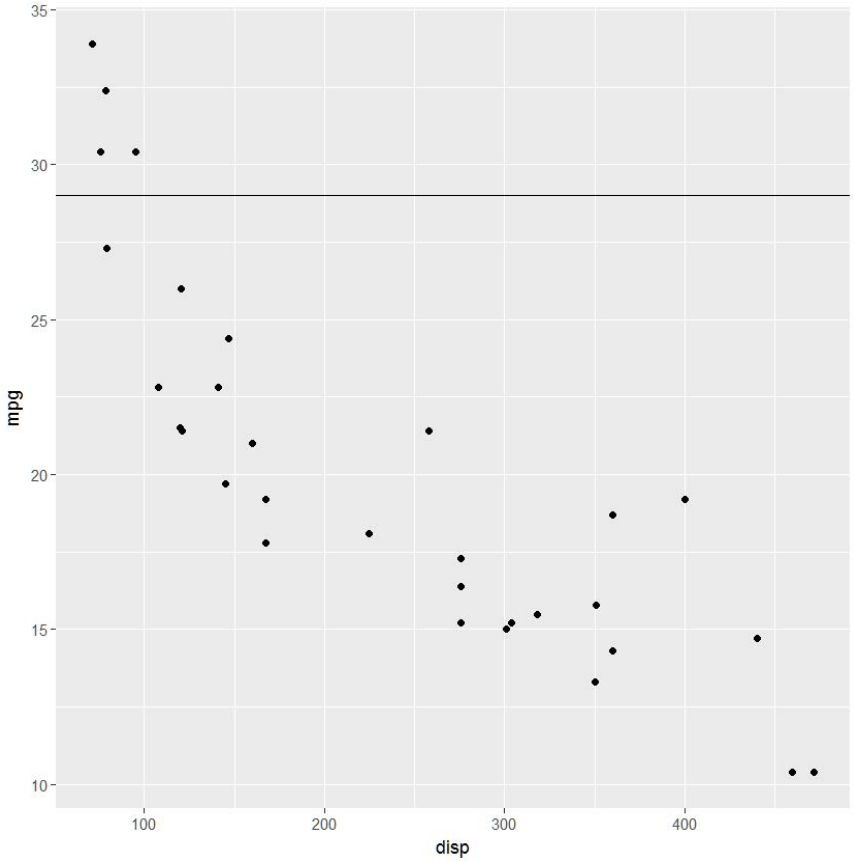


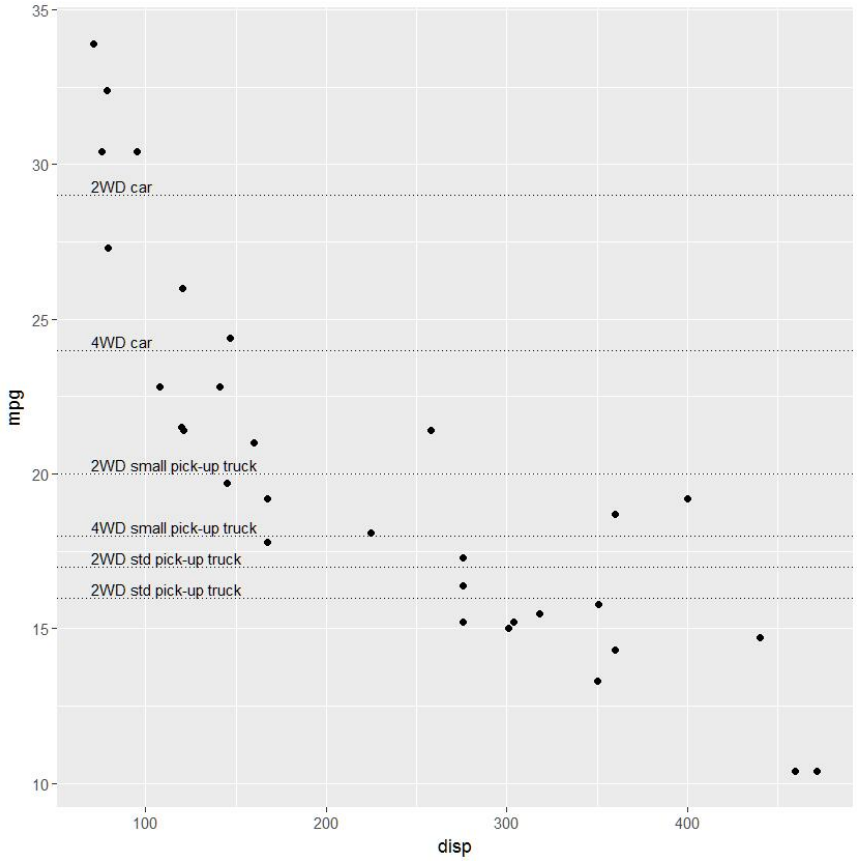


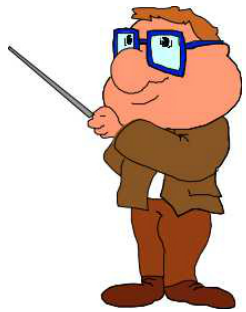
6.3.10 注释

```
>  # 注释
>  p + geom_point(aes(x=disp, y=mpg)) +
+      geom_hline(yintercept=29)
>  gcLimits <- data.frame(
+      category=c("2WD car", "4WD car",
+                 "2WD small pick-up truck",
+                 "4WD small pick-up truck",
+                 "2WD std pick-up truck",
+                 "2WD std pick-up truck"),
+      limits=c(29, 24, 20, 18, 17, 16))
>  p + geom_point(aes(x=disp, y=mpg)) +
+      geom_hline(data=gcLimits, aes(yintercept=limits),
+                 linetype="dotted") +
+      geom_text(data=gcLimits, aes(y=limits + 0.1,
+                label=category), x=70, hjust=0, vjust=0, size=3)
```



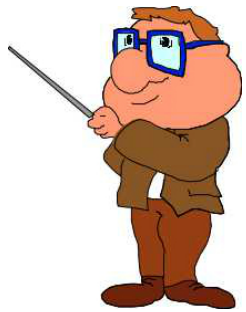

35
30
25
20
15
10
mpg
100
200
300
400
disp


35
30
25
20
15
10
mpg
2WD car
4WD car
2WD small pick-up truck
4WD small pick-up truck
2WD std pick-up truck
2WD std pick-up truck
100
200
300
400
disp



7. 图形管理
7.1 R图形设备概述

- R画图默认打开一个画图窗口
- 还可以把图形画在其他文件上
- 把展示或保存图形的窗口或文件称为设备驱动(device driver)
- 可以调用设备驱动函数来启动设备驱动。
- 每一种设备驱动都有对应的函数：输入**help(Devices)**可以得到它们的列表。



7.2 R图形设备种类

- 常见的图形设备有：

设备函数	说明
windows()	打开绘图窗口(默认设备)
x11()	Unix下的绘图窗口(在Windows下等价于windows())
jpeg()	创建JPEG格式图形文件(.jpg,.jpeg)
png()	创建PNG格式图形文件(.png)
bmp()	创建BMP格式图形文件(.bmp)
postscript()	创建ps格式文件保存图形
pdf()	创建pdf格式文件保存图形

7.3 图形设备管理

- R作图时有时需要同时使用几个图形设备。
- 在一个时间点，只有一个图形设备可以接受图形命令，这个设备称为当前设备。
- 最后打开的设备将成为当前的绘图设备，随后的所有图形都在这上面显示。
- 当多个设备同时启动时，它们将形成一个以名字作为标识符的有限任务序列。
- 可以使用图形设备管理命令来对图形设备进行管理。

7.4 R图形设备管理命令

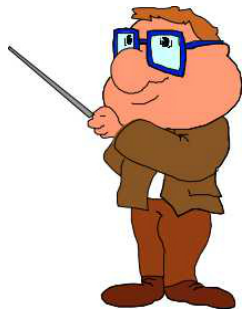
- 常见的图形设备管理命令有：

命令	说明
dev.list()	显示当前所有的设备列表
dev.cur()	显示当前设备
dev.new()	打开图形设备窗口
dev.set(*k*)	设置当前设备
dev.next()	显示当前设备的下一个设备
dev.prev()	显示当前设备的前一个设备
dev.off()	关闭当前设备
graphics.off()	关闭当前所以设备

7.5 利用图形设备管理命令保存R图形

- 创建图形设备时可以指定保存图形的文件
- 还可以指定图形和文件的其他属性

```
> getwd()
> jpeg()
> plot(1:10, ylab=" 变量y")
> dev.off()

> jpeg(file="散点图.jpg", width=800, height=800)
> par(mfrow=c(2,2))
> plot(1:10, ylab=" 变量y")
> plot(1:10, type="l")
> plot(1:10, type="o")
> boxplot(1:10)
> dev.off()

> pdf(family="GB1", width=9, height=6)
> plot(1:10, ylab=" 变量y")
> dev.off()
```



第四章结束了！

THANKS