---
title: "描述探索性数据分析"
output: ioslides_presentation

```{r setup, include = FALSE}
knitr::opts_chunk$set(echo    = T, collapse = T)
```

数据分析有 3 类, 一是描述探索性数据分析, 二是推断预测, 三是因果和机理性的分析
也可以分为 6 类, 描述, 探索, 推断, 预测, 因果, 机理

描述就是作图, 看数据的分布是什么样的. 探索就是在作图的基础上看数据之间有没有显而易见的相互关系. 一般见到的很漂亮的图表和词频都是描述探索性的.

推断预测, 推断是从样本来推断总体, 从 1000 个人向 10000 个人进行推断, 或从旧客户向新客户推断. 预测是从过去的数据去预测未来的数据. 一般推断和预测是合在一起使用的, 如从过去 2 年的旧客户预测未来 2 年的新客户. 推断和预测是看 xy 之间的关系, 关注的是相关性. 注意研究相关性时选择从逻辑上能够有关联的相关性. 不要随意. 如巧克力的销量与 GDP 的关系, 冰淇淋的销量和沙滩性犯罪的关系等.

因果和机理则是说明如果 x 发生变化, y 是否发生变化, 即 x 是 y 变化的原因. 因果关系比相关性更进一步, x 发生变化时 y 一定发生变化是因果, 机理性分析是 x 变化一定的量, y 也一定是变化一定的量, 是定量的因果关系. 更加难以得到, 一般在商业分析中不常用, 在科学研究中应用较多.

#分析性图表的基本原则

基本原则

来自爱德华.塔夫特(Edward Tufte)的书<美丽的证据(Beautiful Evidence)>
怎么通过作图来展现自己的证据或把故事讲明白.有以下 3 个原则.

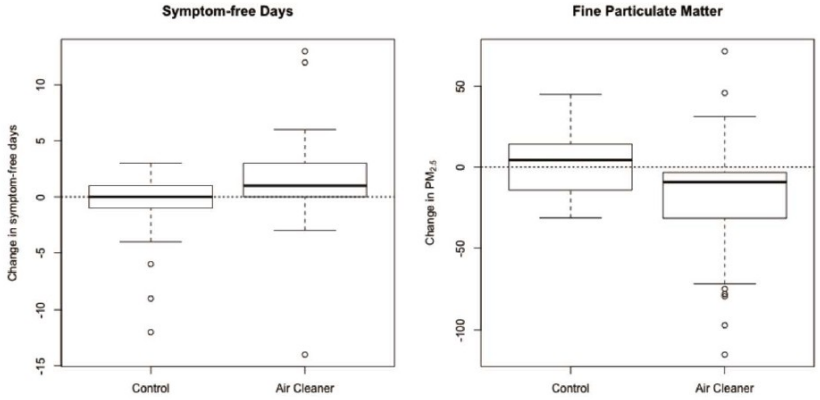
1. 明确的参照物:

- 支持某一个假设(思考)的证据一定与另一个假设是相对的, 假设和备择假设
- 经常问一个问题:以什么作为参照的

2. 体现(自己认为的)因果关系和机制, 做出合理的解释, 体现系统性的结构.(这里的因果关系不一定是真实的, 但一定是要自己能够解释清楚的因果关系)

- 你理解的因果关系, 需要解释你认为某个系统是如何运作的

实例分析

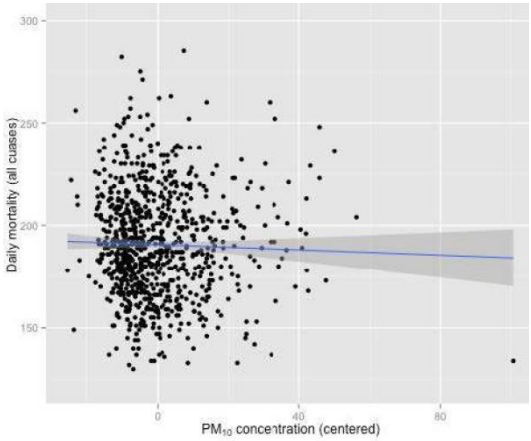


上面两图表示在患儿的家庭里是否安装空气净化器的对比图, 左图表示安装后哮喘症状平均多一天无症状天数, 右图反应了颗粒物的浓度变化

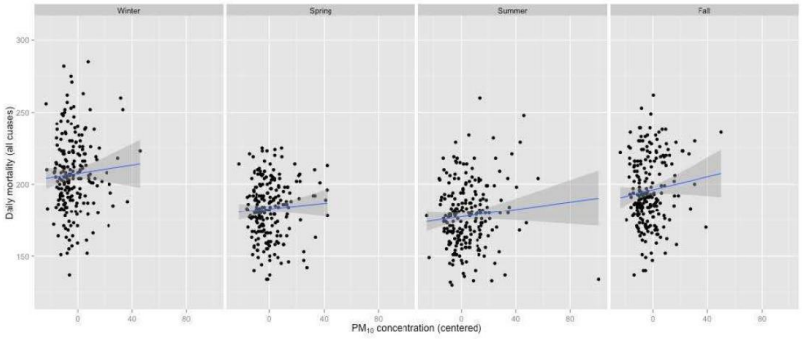
Reference: Butz AM, et al. JAMA Pediatrics, 2011.

3. 展示多元数据(multivariate data)

- 在一个图表中展示尽可能多的数据, 至少多于两个变量



x 轴表示每日中，气动直径(aerodynamic diameter)小于 10 微米的颗粒物浓度，y 每天纽约市死亡人数，是从 1987 年到 2000 年的数据



- 按照季节把不同的数据分开，每个图表中两者都是呈轻微的正相关，即季节会混淆 PM10 和死亡人数的真正关系.
- 所以尽给出尽可能多的合理变量是非常重要的
- 为什么不同季节的数据都是正相关的，把数据放在一起时就变成负相关了呢，因为不同季节数据的分布是不同的. 如果在实际案例中如果得到了不合逻辑的图形，也可以考虑能否根据一定的分类标准把数据分成多元的数据.

4. 整合你所拥有的证据

- 在一次演示中用尽可能多的方式(词语，数字，图形，表格)展现证据，呈现的信息越丰富越好
- 不要被工具限制，按你的想法来做图表

5. 对你呈现的证据进行描述和文档化，打上标签，标记好来源
   - 让你所展现的证据具有可信度，来源，过程都很重要

6. 内容至上
   - 描述的内容本身要独特有趣
   - 首先考虑内容是什么?故事是什么?有什么数据?然后再考虑展现它们的最好方式

最终的目的只有一个，能够最终说服别人接受自己的逻辑. 做数据分析可以看成是在讲一个故事或回答一个问题. 所以在一开始做数据分析时，作图时，作模型时，都要围绕着一条主线来进行，这个主线就是你想呈现给别人什么样的东西或回答一个什么样的问题.

做图的目的就是让别人接受自己的思想, 图形做的要让别人容易接受, 所以做出来的图形不能太复杂或太难懂. 而是要简单明了, 有对比的, 能够体现逻辑关系的. \

总结

1. 展示参照物
2. 体现因果关系或机制
3. 展现多元数据
4. 整合多种模式的证据
5. 描述和文档化证据
6. 内容至上

参考: Edward Tufte (2006). *Beautiful Evidence*, Graphics Press LLC. www.edwardtufte.com

安装包 lattice ggplot2

3 个绘图系统, base 绘图系统, lattice 绘图系统, ggplot2 绘图系统

#探索性图表

在数据分析中使用图表

意义:

- 理解数据的性质
- 寻找数据中的模式
- 提出一些建模策略
- 找出分析中的错误
- 传达结果

特点:

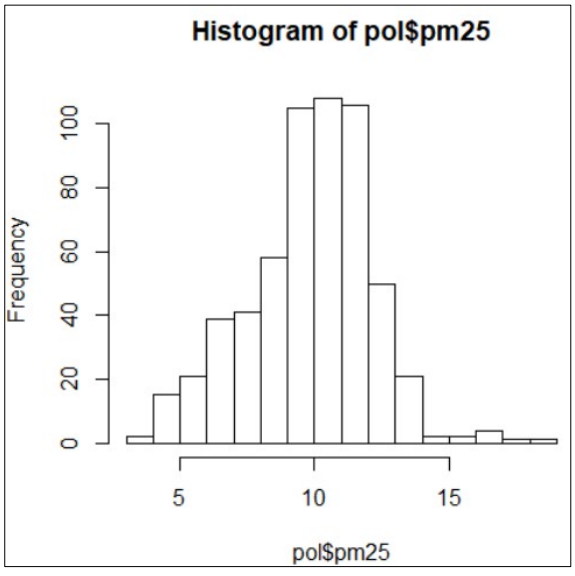
- 制图迅速, 数量巨大
- 目标是用于个人理解(对数据有感觉)
- 坐标和标签通常会在后期被清理掉
- 颜色和大小主要用于分离信息(需要细致斟酌)

数据集

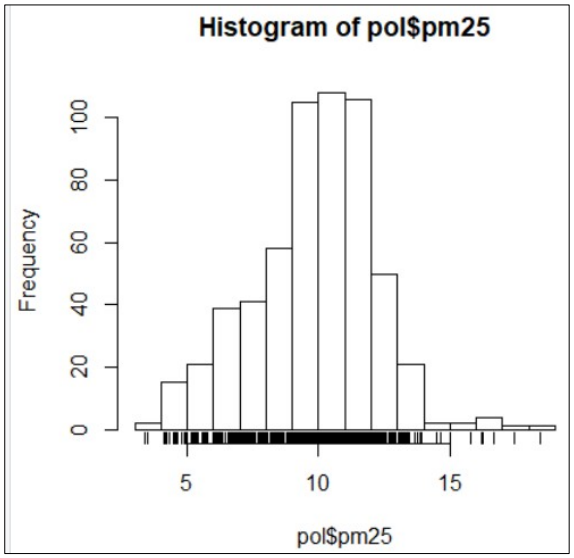
- 来自美国国家环境保护局(The U.S. Environmental Protection Agency, EPA) 的关于美国环境空气污染的数据集，EPA 制定了[美国国家环境空气质量标准](http://www.epa.gov/air/criteria.html)
- 微粒污染(fine particle pollution, PM2.5), 标准是:指定地点三年内的年平均不得超过$12\mu g/m^3$
- 在[EPA 空气质量系统](http://www.epa.gov/ttn/airs/airsaqs/detaildata/downloadaqsdata.htm)中可以得到相关数据

问题: 在美国有没有任何县的微粒污染超出了国家标准

```{r}
# flips 列要读成字符型的, 否则会自动把前面的 0 去掉.
pol <- read.csv("avgpm25.csv", colClasses = c("numeric", "character", "factor", "numeric", "numeric"))

> head(pol, 3)
      pm25  fips region longitude latitude
1  9.771185 01003   east -87.74826 30.59278
2  9.993817 01027   east -85.84286 33.26581
3 10.688618 01033   east -87.72596 34.73148

#直方图
hist(pol$pm25)
```




```
#添加密度图
rug(pol$pm25)
```




```

fips: 美国郡和县的数字代码 [Federal Information Processing Standards](https://en.wikipedia.org/wiki/Federal_Information_Processing_Standards)

## 简单的汇总

一维:
- 五数概括: 给出变量在分位数和均值上的汇总. 如果有缺失值, 会多出一行缺失值
- 箱线图: 箱子的上缘是 3/4 分位数 Q3, 下缘是 1/4 分位数 Q1, 四分位距*IQR = Q3-Q1*, 上下两条线段为异常值截断点, 分别是*Q1-1.5IQR, Q3+1.5IQR*
- 直方图: 横坐标将 x 等分成若干范围, 纵轴表示对应范围内的频数
- 核密度图: 与直方图配套, 估计密度分布
- 条形图: 汇总分类变量

# summary 五值概括

```
summary(pol$pm25)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  3.383   8.549  10.047   9.836  11.356  18.441 
```

```
#可以直接对整个数据框直接生成结果
summary(pol)
```

pm25	fips	region	longitude	latitude
Min. : 3.383	Length: 576	east: 442	Min. : -158.04	Min. : 19.68
1st Qu. : 8.549	Class : character	west: 134	1st Qu. : -97.38	1st Qu. : 35.30
Median : 10.047	Mode : character		Median : -87.37	Median : 39.09
Mean : 9.836			Mean : -91.65	Mean : 38.56
3rd Qu. : 11.356			3rd Qu. : -80.72	3rd Qu. : 41.75
Max. : 18.441			Max. : -68.26	Max. : 64.82
```

箱线图

boxplot {graphics} R Documentation
Box Plots

Description

Produce box-and-whisker plot(s) of the given (grouped) values.

Usage

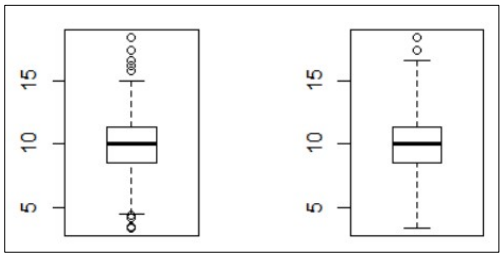
```
boxplot(x, ...)

## S3 method for class 'formula'
boxplot(formula, data = NULL, ..., subset, na.action = NULL,
        drop = FALSE, sep = ".", lex.order = FALSE)

## Default S3 method:
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

Arguments

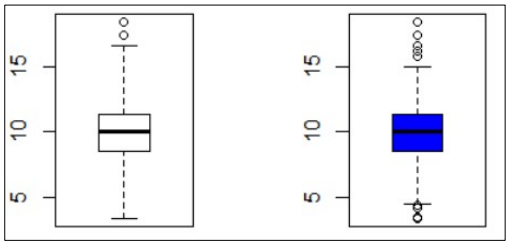
formula	a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variablegrp (usually a factor).
data	a data.frame (or list) from which the variables in formula should be taken.
subset	an optional vector specifying a subset of observations to be used for plotting.
na.action	a function which indicates what should happen when the data contain NAs. The default is to ignore missing values in either the response or the group.
drop, sep, lex.order	passed to split.default, see there.
x	for specifying data from which the boxplots are to be produced. Either a numeric vector, or a single list containing such vectors. Additional unnamed arguments specify further data as separate vectors (each corresponding to a component boxplot). NAs are allowed in the data.
...	For the formula method, named arguments to be passed to the default method. For the default method, unnamed arguments are additional data vectors (unless x is a list when they are ignored), and named arguments are arguments and graphical parameters to be passed to bxp in addition to the ones given by argument pars (and override those in pars). Note that bxp may or may not make use of graphical parameters it is passed: see its documentation.
range	this determines how far the plot whiskers extend out from the box. Ifrange is positive, the whiskers extend to the most extreme data point which is no more

	than range times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes.
width	a vector giving the relative widths of the boxes making up the plot.
varwidth	if varwidth is TRUE, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.
notch	if notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ (Chambers *et al*, 1983, p. 62). Seeboxplot.stats for the calculations used.
outline	if outline is not true, the outliers are not drawn (as points whereas S+ uses lines).
names	group labels which will be printed under each boxplot. Can be a character vector or an expression (see plotmath).
boxwex	a scale factor to be applied to all boxes. When there are only a few groups, the appearance of the plot can be improved by making the boxes narrower.
staplewex	staple line width expansion, proportional to box width.
outwex	outlier line width expansion, proportional to box width.
plot	if TRUE (the default) then a boxplot is produced. If not, the summaries which the boxplots are based on are returned.
border	an optional vector of colors for the outlines of the boxplots. The values in border are recycled if the length of border is less than the number of plots.
col	if col is non-null it is assumed to contain colors to be used to colour the bodies of the box plots. By default they are in the background colour.
log	character indicating if x or y or both coordinates should be plotted in log scale.
pars	a list of (potentially many) more graphical parameters, e.g.,boxwex or outpch; these are passed to bxp (if plot is true); for details, see there.
horizontal	logical indicating if the boxplots should be horizontal; defaultFALSE means vertical boxes.
add	logical, if true *add* boxplot to current plot.
at	numeric vector giving the locations where the boxplots should be drawn, particularly when add = TRUE; defaults to 1:n where nis the number of boxes.

```{r, fig.height = 3.5}
#是一个全局参数, 设置图形区有 1 行 2 列
par(mfrow = c(1, 2))

boxplot(pol$pm25)

# range 默认是 1.5, Q1-rangeIQR, Q3+rangeIQR
boxplot(pol$pm25, range = 2)
```

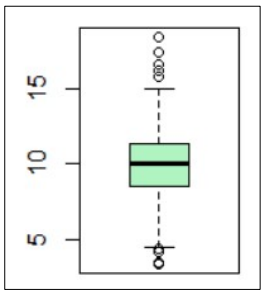


#箱线图. 作图的一个原则是对比, 所以一般会作多个箱线图
boxplot(pol$pm25, col = "blue")

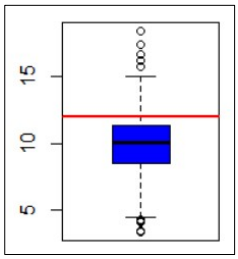


colors()　#显示 Rplot 时可选的颜色, 也可以使用数字. col = 1 表示黑色, col = 2 表示红色. 还可以使用 RGB 颜色. 可以在作图程序中选择一个颜色, 就会显示出 RGB 的值. 使用 col = rgb(98, 234, 132, 128, maxColorValue = 255), 前 3 个数字是 rgb 的值, 第 4 个数字是透明度, 数字越小透明度越大, 255 表示完全不透明, 0 表示 100%透明. 0-255 之间选取. 第 5 个参数说明前 4 个参数的最大值. 如果省略最后一个参数, 就要把 rgb 的颜色值除以 255, 把透明度设置为 0-1 之间. 如 rgb(98/255, 234/255, 132/255, 0.5). 颜色等于数值的好处是可以给一组曲线如散点图设置一组渐近的颜色. 颜色也可以等于某个指定颜色的函数.
rgb(98, 234, 132, 128, maxColorValue = 255)　#返回 16 进制颜色.
[1] "#62EA8480"

#箱线图显示数据的分布趋势, 主要展示数据的集中趋势. 直方图展示的是密度曲线, 显示数据的分布趋势, 是否均匀分布, 是否左偏右偏, 是否正态分布等.
boxplot(pol$pm25, col = rgb(98, 234, 132, 128, maxColorValue = 255))
boxplot(pol$pm25, col = "#62EA8480")
boxplot(pol$pm25, col = rgb(98/255, 234/255, 132/255, 0.5))



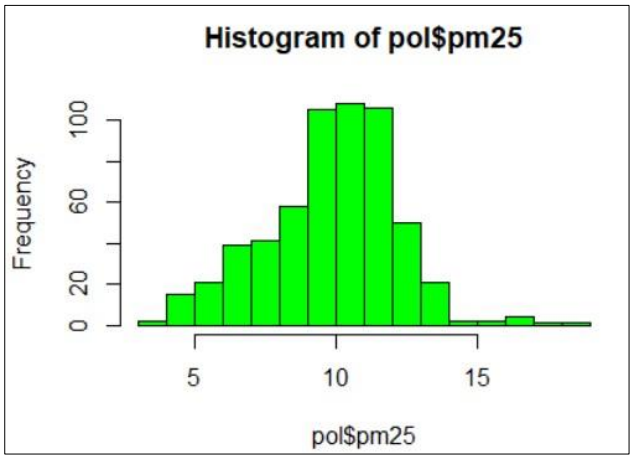
boxplot(pol$pm25, col = "blue")
# 添加水平线, 线宽 lwd 默认为 1, 设置为 2 表示是默认值的 2 倍.
abline(h = 12, col = "red", lwd = 2)



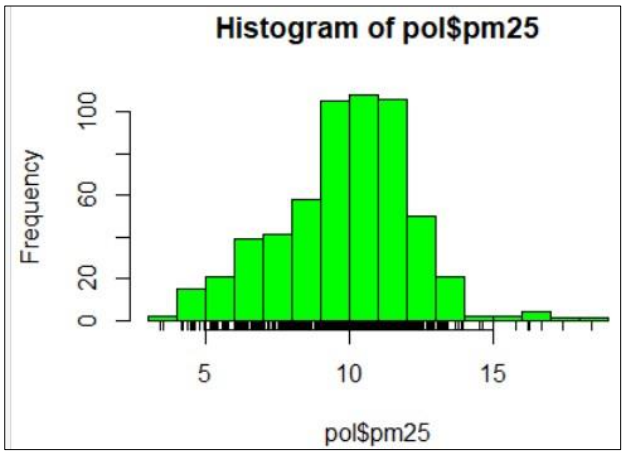
直方图

直方图. 直方图展示的是密度曲线, 显示数据的分布趋势, 是否均匀分布, 是否左偏右偏, 是否正态分布等.

# y 轴 frequency 表示指定范围内元素的数量
hist(pol$pm25, col = "green")



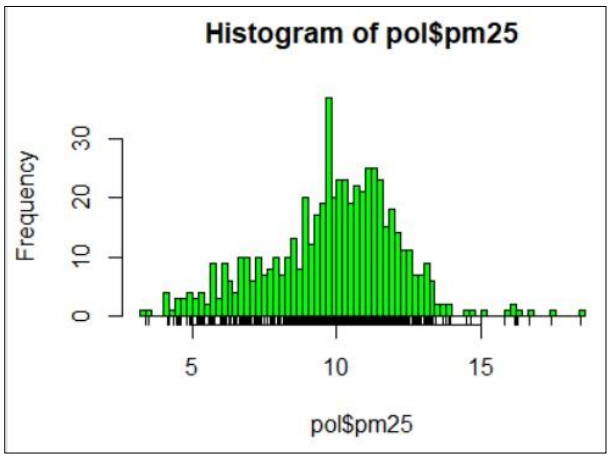
# 在直方图下面嫁一个地毯图. rug 是注释函数, 在原有的直方图的基础上添加的注释性的和密度图, 每一个数据是一条竖线, 也是展示数据的分布趋势.
rug(pol$pm25)



每次做图时都要指定数据框的名字, 可以使用 attach 来加载数据框, detach 用来取消加载. 如

attach(pol)
boxplot(pm25)

```{r, fig.height = 3.5}
par(mfrow = c(1, 2))
# breaks 要分多少个箱子.默认是 20, 即有 20 个分割线, 得到 19 个箱子.
hist(pol$pm25, col = "green", breaks = 100)
rug(pol$pm25)
```



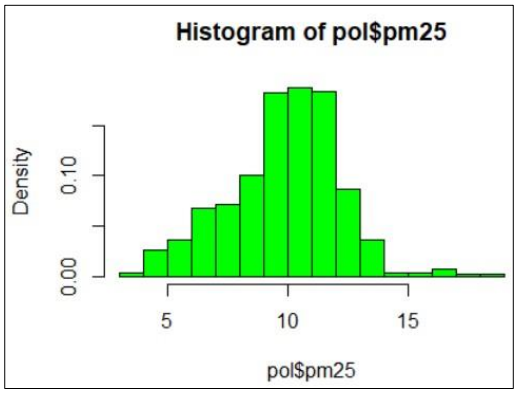

```

## 直方图添加特征

```{r, fig.height = 3.5}
par(mfrow = c(1, 2))  #是一行二列的意思, 同时显示 2 个图形

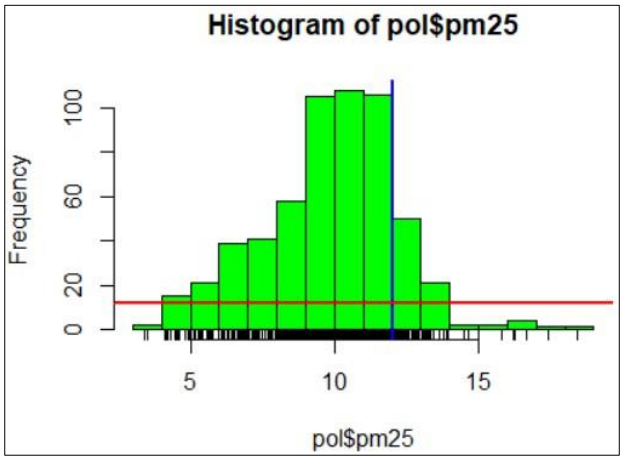
#freq 是纵轴的频数. 某个直方图对应的频数是在此区间内的数据个数. freq = F 时 y 轴得到的是频率, 即频数/总数. 正态分布的直方图的 y 轴就是频率. 沿着箱子顶部画一条曲线, 得到的就是概率密度曲线.
hist(pol$pm25, col = "green", freq = F)

hist(pol$pm25, col = "green", breaks = 20, freq = F)
```


```

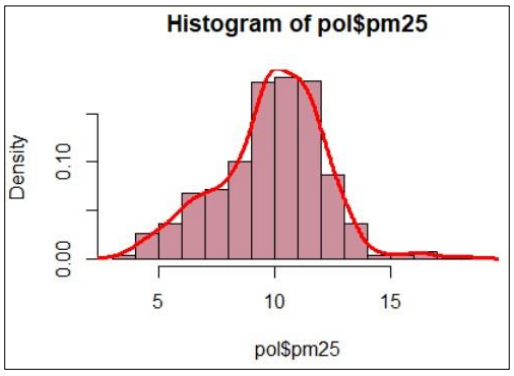
#添加辅助图. 微粒污染(fine particle pollution, PM2.5), 标准是:指定地点三年内的年平均不得超过 12 μ g/m3. 就要添加一条 12 的辅助线, 以显示空气是否超标. 可以画横线, 竖线和斜线(回归线). v:vertical h:horizontal. lwd 是线宽, line-width, 默认值是 1, 等于 2 时表示是默认值的倍数. 做图是一层层的进行覆盖的, 如果同时画两个位置相同的元素, 后面的会把前面的覆盖掉.

```
hist(pol$pm25, col = "green", breaks = 20)
rug(pol$pm25)
abline(v = 12, lwd = 2, col = "black")
abline(v = 12, lwd = 2, col = "blue")
abline(h = 12, lwd = 2, col = "red")
```

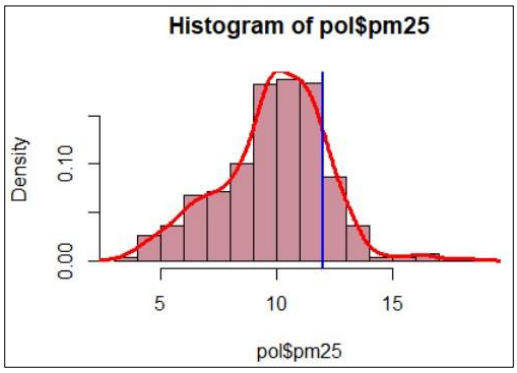


#添加概率密度曲线

```
hist(pol$pm25, col = "pink3", freq = F)
lines(density(pol$pm25), col = "red", lwd = 3)   #添加概率密度曲线, density()计算概率密度
```

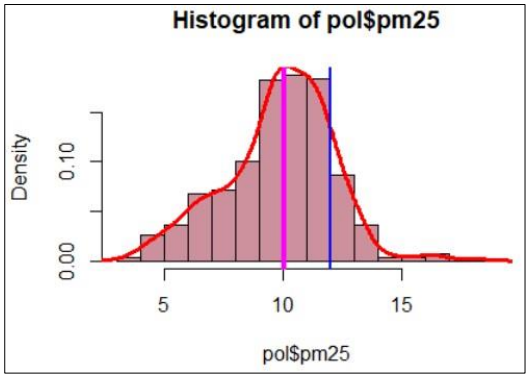



```
abline(v = 12, lwd = 2, col = "blue")
```



颜色是品红，添加中位数的曲线

abline(v = median(pol$pm25), col = "magenta", lwd = 4)



条形图

条形图用来展示分类型变量，给多少条数据，做多少条条形图

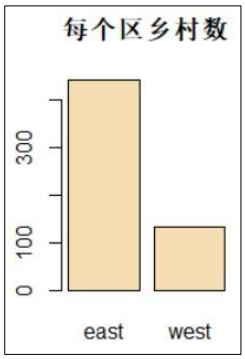
table 用来查看分类型数据每一类的频数

```
table(pol$region)
```

```
east west
 442  134
```

做条形图. main 为标题

barplot(table(pol$region), col = "wheat", main = "每个区乡村数")




```
```

Practices

我们使用 R 自带的数据 airquality 函数, 其中 Ozone 表示臭氧浓度, Solar.R 表示太阳辐射, Wind 表示风速, Temp 表示温度(华氏度).查看?airquality 了解更多

- 给出数据所有变量的总结(五数概括)

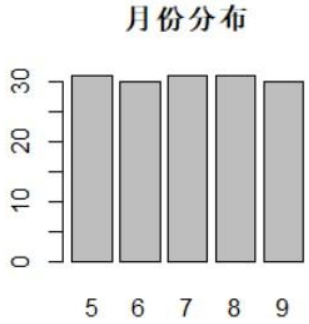
attach(airquality)

summary(airquality)

Ozone	Solar.R	Wind	Temp	Month	Day
Min. : 1.00	Min. : 7.0	Min. : 1.700	Min. : 56.00	Min. : 5.000	Min. : 1.0
1st Qu. : 18.00	1st Qu. : 115.8	1st Qu. : 7.400	1st Qu. : 72.00	1st Qu. : 6.000	1st Qu. : 8.0
Median : 31.50	Median : 205.0	Median : 9.700	Median : 79.00	Median : 7.000	Median : 16.0
Mean : 42.13	Mean : 185.9	Mean : 9.958	Mean : 77.88	Mean : 6.993	Mean : 15.8
3rd Qu. : 63.25	3rd Qu. : 258.8	3rd Qu. : 11.500	3rd Qu. : 85.00	3rd Qu. : 8.000	3rd Qu. : 23.0
Max. : 168.00	Max. : 334.0	Max. : 20.700	Max. : 97.00	Max. : 9.000	Max. : 31.0
NA's : 37	NA's : 7				

- 绘制月份的条形图, 定义标题为"月份分布"

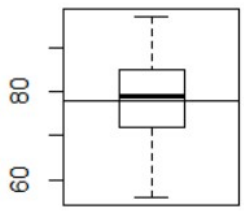
barplot(table(Month), main = "月份分布")



- 绘制温度的箱线图, 且在图形中添加均值的水平线

boxplot(Temp)

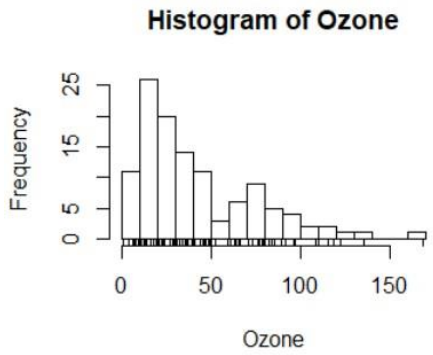
abline(h = mean(Temp))



- 绘制臭氧浓度的直方图, 纵轴是频数, 且给出图形的条数约为 20, 并且在直方图下添加地毯图

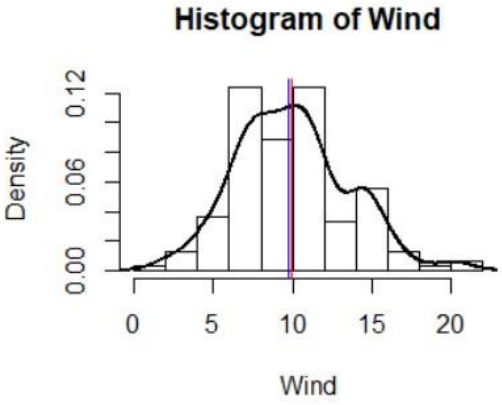
hist(Ozone, breaks = 21)

rug(Ozone)



- 绘制风速的直方图，纵轴是频率，添加垂直的均值线和中位数线，区分颜色，添加密度曲线

```
hist(Wind, freq = F)
lines(density(Wind), lwd = 2)
abline(v = mean(Wind), col = "red", na.rm = T) #去掉缺失值
abline(v = median(Wind), col = "blue")
# 一个数据包使用完后要及时的 detach，如果多次 attach，会在系统环境变量中有多个的数据包，占用内存，可以使用 search()来查看是否多次 attach 同一个包.
detach(airquality)
```



多个图形

二维:

- 多个/覆盖的一维图(Lattice/ggplot2)
- 散点图 Scatterplots
- 平滑散点图 Smooth scatterplot
- 饼图 pie

要不断学习新的包

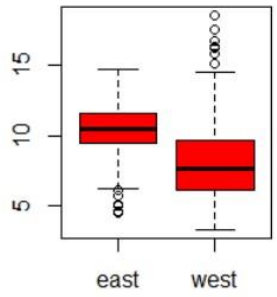
大于二维

- 多个/覆盖的二维图
- 使用颜色，大小，形状增加维度
- 可旋转的图(Spinning plot),
- 三维图(并不是很有用)

多个箱线图

```{r, fig.height = 3.5}
pol <- read.csv("avgpm25.csv", colClasses = c("numeric", "character", "factor", "numeric", "numeric"))

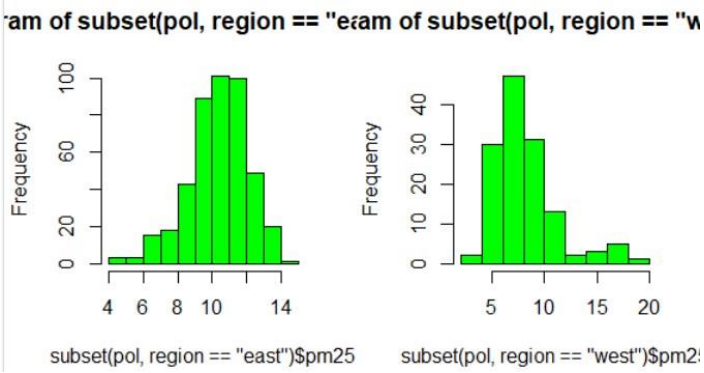
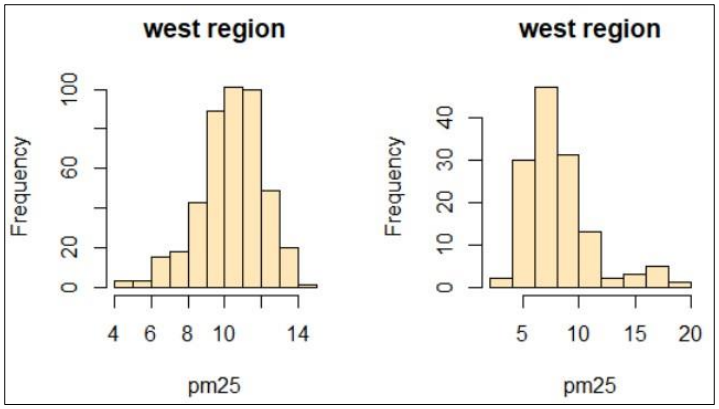
#~表示一个公式，一般情况下公式左边是 y，右边是 x，或者右边对左边预测，或者右边给左边划分，左边是数据，右边是划分依据或回归依据等. pm25 对 region 作多个箱线图. 对比两个图，大致有个直观的概念，看二者是否有显著的差异. 如果确定是否有显著的差异需要做两样本显著性 t 检验. regin 如果是字符型向量的话会自动转化为因子型向量，在有些作图函数中，即使是数值型向量，如 1, 2, 3，放在~的右边也会被转换成因子型向量. 所以如果数据中有分类变量的话，最好先手动转换成因子. 可以把所有分类变量的名都放在一个向量中，使用 as.factor 做一个循环即可.
# data: a data.frame (or list) from which the variables in formula should be taken. data 中要指定数据框.
boxplot(pm25~region, data = pol, col = "red")
```




```

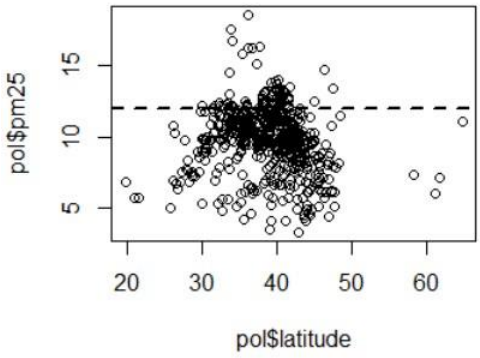
可以看到东部的中位数明显比西部高，也就是东部的环境污染明显比西部要严重一些，东部的异常值主要分布在下面，西部的异常值主要分布在上面.

## 多个直方图

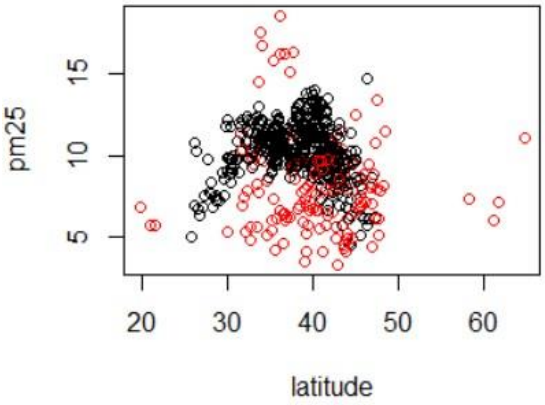
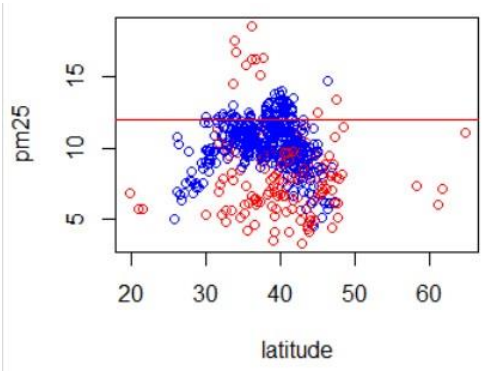
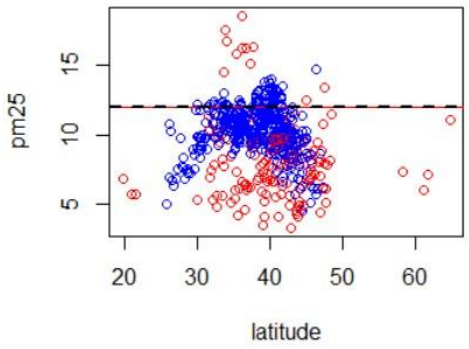
```{r, fig.height = 3.5}

# 显示出当前全局环境下所有的全局参数
par(no.readonly = T)
```
```

# 把当前全局环境下所有的全局参数或全局变量保存到一个变量中，然后修改全局参数，工作结束后再把这个变量读取到全局变量中即可．或者重启 R.
opar <- par(no.readonly = T)

# 全局参数．在 par()中设置全局参数即可.
# mfrow 设置图形的分布情况，默认是 c(1 ,1)，即 1 行 1 列. mfrow = c(2, 1)表示 2 行 1 列，如果是 2 行 2 列的，先填充行，再填充列. Rstudio 的绘图设备, png 图像设备．可以把图形输出到这个设备中.
# mar 表示内边距．即图形边框与画布之间的距离．内边界中放 x 轴即其标签，副标题, y 轴及其标签，主标题．c(4, 4, 2, 1)表示下左上右边距．默认是 c(5.1, 5.1, 4.1, 2.1)．可以使用 opar$mar 来查看.
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

几个变量的默认值
```
$mar
[1] 5.1 4.1 4.1 2.1
$mfcol
[1] 1 1
$mfrow
[1] 1 1
```

# subset 是使用函数取子集．第 2 个参数是筛选的依据. subset(pol, region = "east")是把 pol 中所有 region = "east"的行找出来，而$pm25 是把 pm25 列找出来.
hist(subset(pol, region == "east")$pm25, col = "green")
hist(subset(pol, region == "west")$pm25, col = "green")



使用[]来取子集. main 为主标题, xlab 为 x 轴标签.
hist(pol[pol$region == "west", "pm25"], col = "wheat1",
     main = "west region", xlab = "pm25")



#调用默认的全局变量
par(opar)
```

## 散点图

```{r}
par(mfrow = c(1, 1))
plot(pol$latitude, pol$pm25)

# 添加水平线, lty 是 line type 的缩写, 线的类型
abline(h = 12, lwd = 2, lty = 2)
```



#画纬度对应的散点图
with(pol, plot(latitude, pm25))

#region 会强制转换为因子, 是用整数进行存储的, east 存储的是 1, west 存储的是 2. col = region 相当于 col = 1 或 2, 1 对应于黑色, 2 对应的红色. region 有 576 个 east 或 west, 也就有 1 或 2. latitude 为 576 个, pm25 也有 576 个, 这样二者就一一对应了起来. 即一个 latitude(pm25) 的点对应于一个 region 对应的 1 或 2. 这样就把 col = 1 与 col = 2 的即 east 和 west 的点用不同的颜色区分开了. 黑色圆圈代表东部的县, 红色代表西部的县
with(pol, {plot(latitude, pm25, col = region)})
```



但这种方法并不是通用的设置颜色的方法，一般更改颜色的方法是先使用子集取出来 east 的点，使用一种颜色并画出来，同样用另一种颜色把 west 的点画出来. 代码如下. 也可以使用 RGB 写一个长度为 576 的过渡的颜色.

```
#pol$col 是颜色变量. 如果是 east 的，就设置为 red，不是 east 的设置为 blue
pol$col <- ifelse(pol$region == "east", "blue", "red")
with(pol, {plot(latitude, pm25, col = col)
      ;abline(h = 12, col = "red")})
```




```
abline(h = 12, lwd = 2, lty = 2)
```



多个散点图

```
```{r}
par(mfrow = c(1, 2), mar = c(5, 4, 2, 1))
```
```

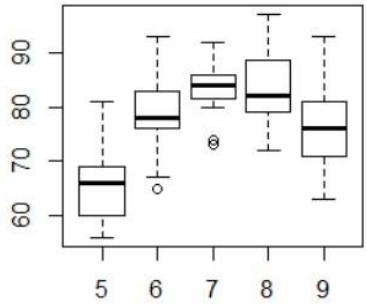
```
with(subset(pol, region == "east"), plot(latitude, pm25, main = "east"))
with(subset(pol, region == "west"), plot(latitude, pm25, main = "west"))
```

Practices

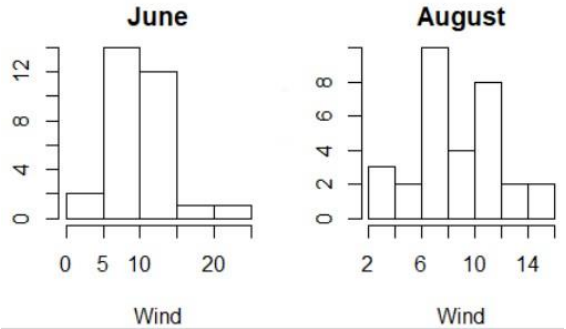
使用 airquality 的数据集

- 查看不同月份中温度的差异(箱线图展示)
- 查看 6 月份和 8 月份风速的分布情况(直方图), 绘制在一张图形中
- 查看 6 月份和 8 月份温度和风速的关系, 温度为 x, 散点图表示, 按月份区分颜色

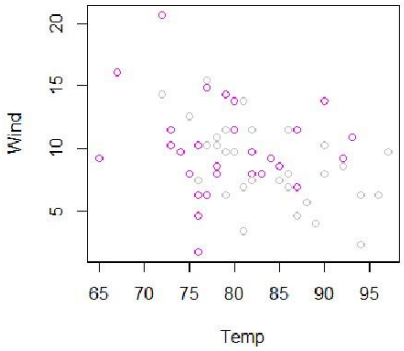
```
attach(airquality)
boxplot(Temp~Month)
```




```
opar <- par(no.readonly = T)
par(mfrow = c(2, 1), mar = c(4, 3, 2, 1))
hist(airquality[Month == 6, "Wind"], main = "June", xlab = "Wind")
hist(airquality[Month == 8, "Wind"], main = "August", xlab = "Wind")
par(opar)
```




```
with(airquality[Month == 6|Month == 8, ], {plot(Temp, Wind, col = Month)})
```



detach(airquality)

总结

- 探索性图表通常是"快速粗略"的
- 有点在于汇总数据，并且能突出显示某个特点
- 可以探索一些基本的问题和假设
- 为下一步的建模策略提供建议

资源

* [R Graph Gallery](http://gallery.r-enthusiasts.com/)
* [R Bloggers](http://www.r-bloggers.com/)

#R 绘图系统

基础绘图系统

基础绘图系统是 R 里最古老的绘图系统

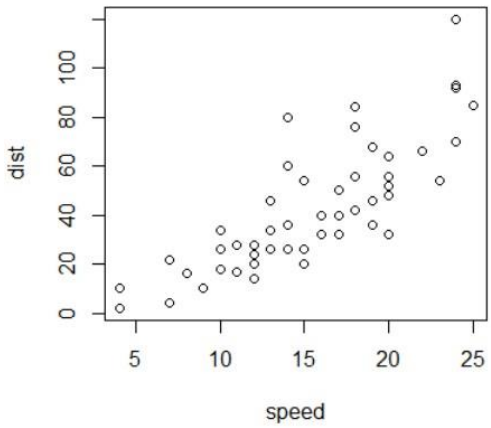
- 概念模型:"艺术家的调色板"
- 在一张空白的画布上逐项添加
- 最典型的方式就是首先使用 plot()函数
- 使用注释函数去添加或者修改(text, lines, points, axis)

- 优点
    - 方便直观，契合我们思考作图和分析数据的过程
- 缺点
    - 不能返回上一个操作(例如调整边界)，需要提前规划好
    - 代码解释性不强

- 参数大多需要人工设置

基本绘图

```{r, fig.height = 3.5}
library(datasets)
data(cars);with(cars, plot(speed, dist))
```



横轴表示汽车速度, 纵轴表示汽车制动长度

Lattice 绘图系统

依靠 lattice 包实现, 不是用一系列命令把一张图一点一点拼接起来, 而是每一张图都调用一个单独的函数画出来. 所有的设置都是在一个函数中完成的.

- 最常用的函数是 xyplot, bwplot 等, 要在调用函数是指定许多信息
- 常用于绘制所谓的 coplot 也就是条件散点图, 观察比如说随着 z 水平变化, x 和 y 之间关系的变化情况
- 有时被称作面板图, 每个面板上看到的都是相同的东西只是 z 不同
- 在基础绘图系统中需要指定的很多细节都会被自动计算出来, 例如边缘, 间距之类, 直接使用默认值就可以
- 缺点在于一旦创建图形不能添加任何东西

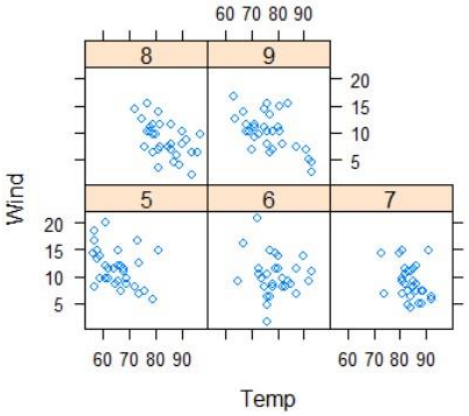
Lattice 图形

```
xyplot(x,
       data,
       allow.multiple = is.null(groups) || outer,
       outer = !is.null(groups),
```

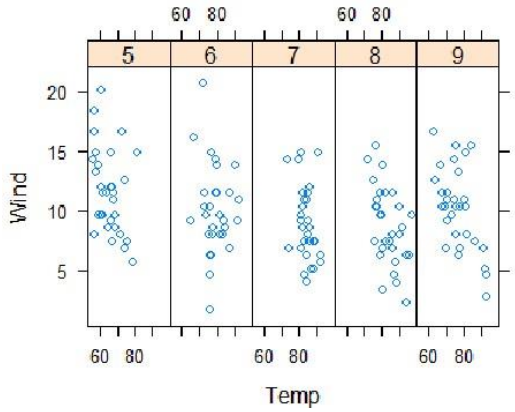
```
    auto.key = FALSE,
    aspect = "fill",
    panel = lattice.getOption("panel.xyplot"),
    prepanel = NULL,
    scales = list(),
    strip = TRUE,
    groups = NULL,
    xlab,
    xlim,
    ylab,
    ylim,
    drop.unused.levels = lattice.getOption("drop.unused.levels"),
    ...,
    lattice.options = NULL,
    default.scales,
    default.prepanel = lattice.getOption("prepanel.default.xyplot"),
    subscripts = !is.null(groups),
    subset = TRUE)
```

视频演示

```{r}
library(lattice)
attach(airquality)
# 画出来 Wind(Temp)的函数图像. 这里是画条件散点图, 在 | 后面指定条件.
xyplot(Wind~Temp|as.factor(Month))
```

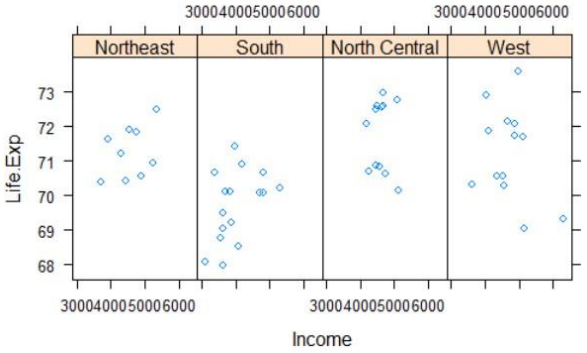



```
#最好给出一个 factor 因子作为散点图的条件, 否则可能会出错. layout = c(5, 1)表示 5 列 1 行.
xyplot(Wind~Temp|as.factor(Month), layout = c(5, 1))
```



detach(airquality)
```

```{r, fig.height = 3.2}
library(lattice)
state <- data.frame(state.x77, region = state.region)
xyplot(Life.Exp~Income|region, data = state, layout = c(4, 1))
```



Lattice 包里的数据，纵轴是一个州的平均预期寿命，横轴是这个周的人均收入，把这个国家每个州所在的地区作为条件变量

ggplot2 系统

结合了基础绘图表和 lattice 的优点

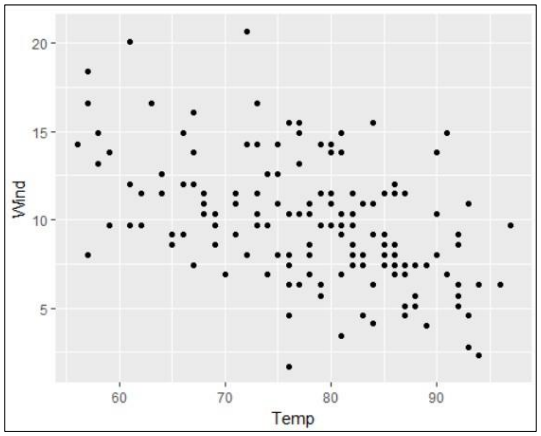
这个系统来自图形语法，图形语法为某类图像设定了一套规则，创造了一种用于描述图像的不同层面的语法，依靠 ggplot2 包实现

- 有点像 lattice 包和基础绘图包折中后的产物，既可以一项一项不断累加作图，也可以自动完成了很多美学方面的计算(间距，标签)
- 像 lattice 一样很方便，有很多默认参数，选择较多，自定义方便
-
- 不同绘图系统不能混合使用
```

ggplot2 图形

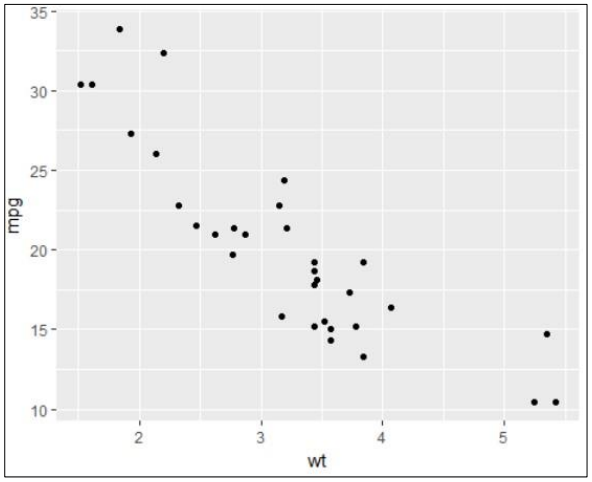
视频演示

```{r}
attach(airquality)
# x 为 Temp, y 为 Wind
qplot(Temp, Wind, data = airquality)
```




```
detach(airquality)
```

```{r, fig.height = 3.5}
library(ggplot2)
data(mtcars)
qplot(wt, mpg, data = mtcars)
```




```
```

横坐标是汽车重量，纵坐标是汽车每加仑英里数

#基础绘图系统

绘图系统

R 中所有核心的绘图和图像引擎都封装在如下的包中:
search()
"package:graphics" "package:grDevices"
- graphics:包含了最基本的绘图函数, 包括 plot, hist, boxplot 等
- grDevices:涵盖了实现绘图设备的全部代码, 包括屏幕设备, 比如 Windows 端口, 文件设备像 pdf, png 等

Lattice 绘图系统封装在如下的包中:

- lattice: 包含制作格子图形的代码, 独立于基础绘图系统, 包含函数 xyplot, bwplot, levelplot 等
- grid: 一种低级绘图系统, 通过 lattice 或 ggplot2 间接调用, lattice 包构建在此包之上

绘图过程

绘图的时候必须考虑如下几个问题(没有先后顺序)

- 图形绘制在哪里, 屏幕上?文件中?
- 图形如何使用
    - 在屏幕上临时显示?
    - 在网页浏览器中显示?
    - 最终被打印到纸上?要具有出版物质量么?
    - 用来做幻灯片展示?
- 绘制图形是否会用到大量的数据, 还是只有几个点?
- 是否需要动态调整图形的大小?
- 要使用哪种绘图系统:基础?lattice?ggplot2?这些不能混用

在把图形添加到出版物中的时候一定要注意保存图像到文件时的 width 和 height 是分辨率, 如果分辨率太高, x 轴和 y 轴的标签会变得非常小.

在做出版物时, 一定要注意图像的分辨率, 分辨率很大的话图像上的元素看起来就很小. 打印出来就会看不清楚.

基础绘图

基础绘图系统本质上画的都是二维图, 而且使用广泛

- 制图的两个步骤: 1. 新图的初始化, 2. 注释和补充
- 初始化图形常用的两个函数 plot(), hist(), 如果没有已经打开的图形设备, 调用这两个函数

会打开一个图形设备，一般会以弹窗的形式显示在屏幕上，接着会在新的图形设备上绘图
- plot()函数含有许多参数，没有指定这些参数的情况下会用缺省值
- 基础绘图系统中有很多参数来设置和优化图形，大部分参数在?par 的帮助页面里面都有说明

绘图参数

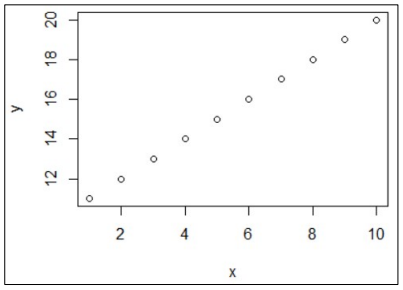
重要的基础绘图参数

需要掌握的的基本绘图参数，许多绘图函数都共用这些参数

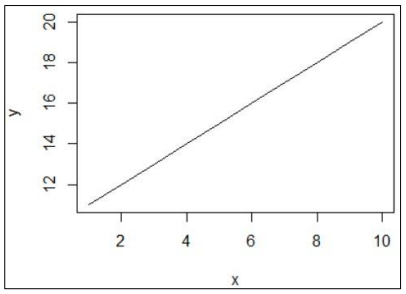
- type

- type: 常用于 plot 函数中，常见的有"p" :点; "l" :线; "b" :点线; "n"不制图

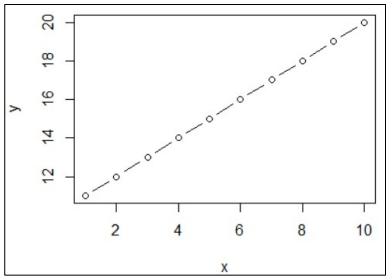
```
x <- 1:10; y <- 11:20
plot(x, y)    # 默认是散点图
```




```
plot(x, y, type = "l")    #1 表示 line，线图. type 默认值是 "p"，即是散点图
```



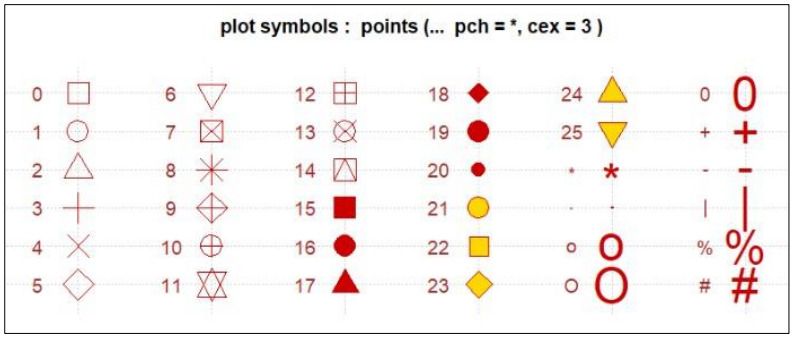

```
plot(x, y, type = "n")    #只是给出 xy，不画图

plot(x, y, type = "b")    #点线图
```



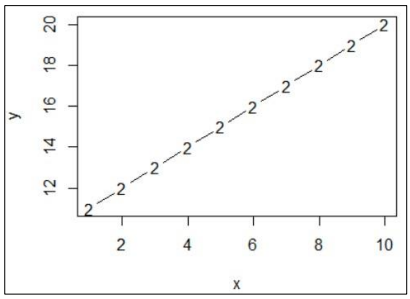
- pch

- pch: plotting character 绘图符号(默认是圆圈), 取数字会指向内置的符号表, 取字符会将该字符绘制在屏幕上, 例如 pch = "a"

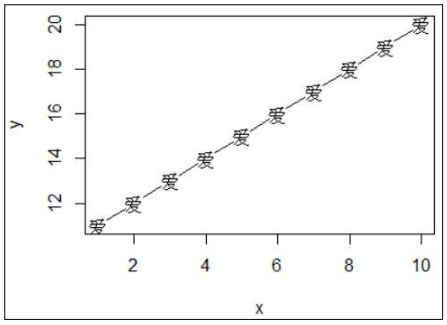
pch: 绘图符号样式




```
x <- 1:10; y <- 11:20
plot(x, y, type = "b", pch = "21")
```



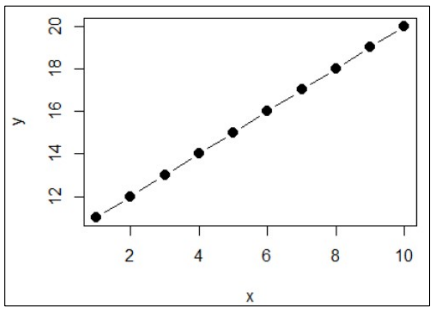

```
plot(x, y, type = "b", pch = "爱")  #点线图. pch, plotting character 绘图点的形状, 默认是 1, 即小圆圈.
```



- cex

- cex: 数值, 表符号的大小, 数值表示倍数

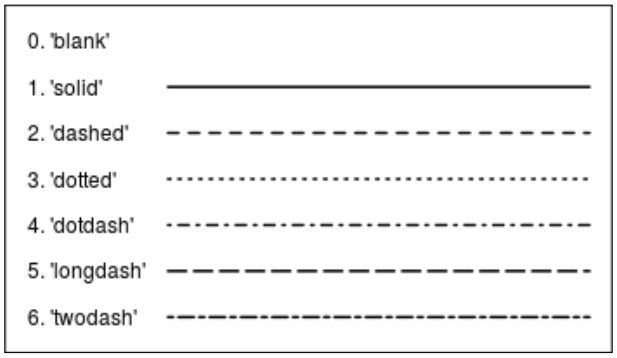
```
x <- 1:10; y <- 11:20
plot(x, y, type = "b", pch = 20, cex = 2)  # cex 为绘图点的大小, 为默认大小的倍数
```



- lty

- lty: line type 线型，默认是 1：实线

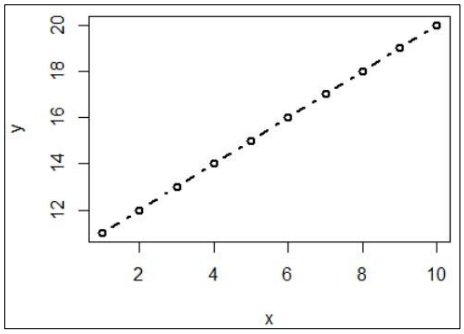
lty：线型



- lwd

- lwd: line width 线宽，指定一个整数，数值表示倍数

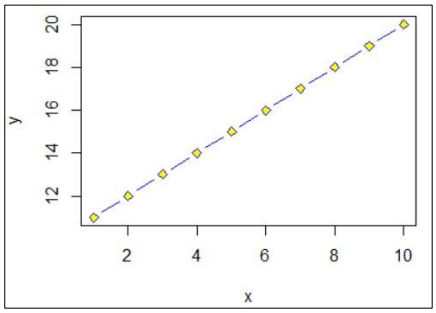
plot(x, y, type = "b", lty = 2, lwd = 2)　# lty 表示线型



- col

- col: color 图形的颜色，可以赋值为数字，字符串，字符格式的十六进制代码, colors()函数会给出一个可选颜色的名称列表，如果有轮廓颜色和填充颜色，轮廓用 col 指定，填充用 bg 指定

plot(x, y, type = "b", pch = 23, col = "blue", bg = "yellow")　# col 为绘图点轮廓的颜色, bg 为绘图点填充颜色



- xlab, ylab: x-axis label/y-axis label 指定 x, y 轴标签

全局参数

- par()函数

- par()函数用于指定全局图形参数，会影响 R 进程中的所有图形，一些参数可以在每次调用图形函数的时候在图形函数内覆写这些参数，而有些参数例如 mar, mfrow 只能用 par()设定.
?par #查看 plot 的默认参数.

- par(no.readonly = T) 用来得到当前所有的绘图参数
在修改全局参数之前先把默认的全局参数保存到一个变量中，修改全局参数完成作图后再从变量中恢复默认的全局参数.
# 1. 把全局参数保存到变量中
opar <- par(no.readonly1 = T)
# 2. 恢复默认的全局参数
par(opar);

- 全局参数默认值

```{r}
par("lty")  ##调用par 函数可以查看缺省值
## [1] "solid"
par("col")
## [1] "black"
par("pch")
## [1] 1
par("bg") ##默认透明
## [1] "white"
par("mar")
## [1] 5.1 4.1 4.1 2.1
par("mfrow")
## [1] 1 1
```

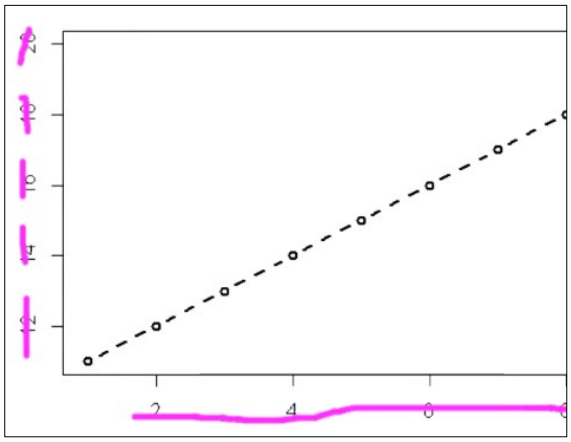
也可以直接在 opar 中查看

可以输入 example(points), 通过示例查看 R 绘图方面的功能

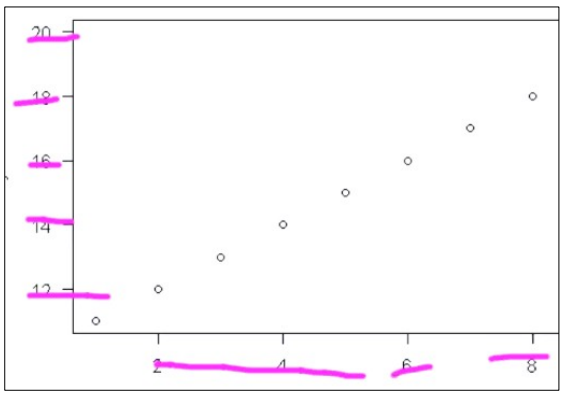
- las

- las: 设定刻度值的方向, 只能是 0, 1, 2, 3, 默认值 0 表示总是平行于坐标轴；1 表示总是水平方向；2 表示总是垂直于坐标轴；3 表示总是垂直方向. 坐标轴刻度的方向就是刻度值的方向, 沿刻度值的数字横切一刀的方向, 垂直于刻度值数字的正方向即刻度值的方向.

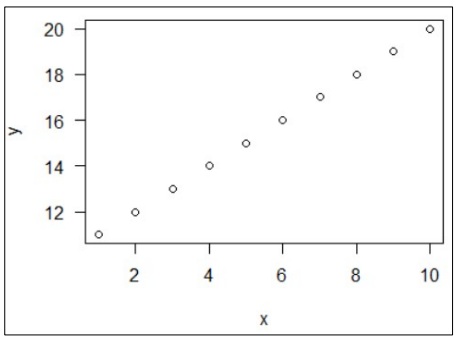
las = 0
plot(x, y, las = 1)



las = 1

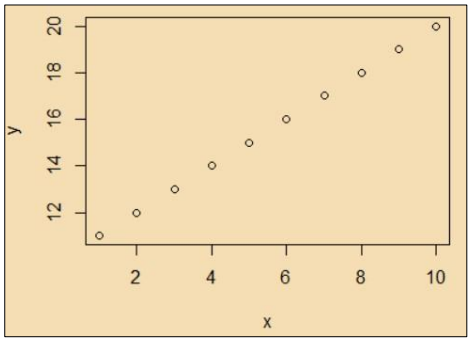


plot(x, y, las = 1)



- bg

- bg: background color 背景颜色
par(bg = "wheat") #bg 只能放在全局参数中，不能放在 plot 函数中.
plot(x, y, las = 0)

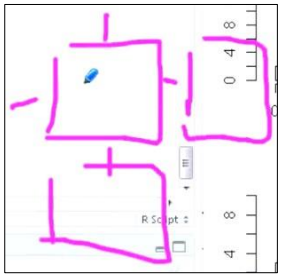


- mar

- mar: margin size 边距尺寸，当轴标非常复杂需要将边距调大些，四个数字，下左上右

- oma

- oma: 外边距尺寸(默认为 0)，一页中绘制多个图形，整张图的标签就可以放在外边距里，如设置为(0, 0, 2, 0)即可



- mfrow

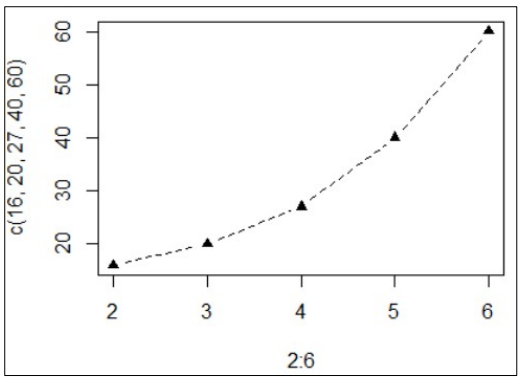
- mfrow: 控制多图布局，例如有一个图形矩阵，设置每行每列显示的图形数，按行填充

- mfcol

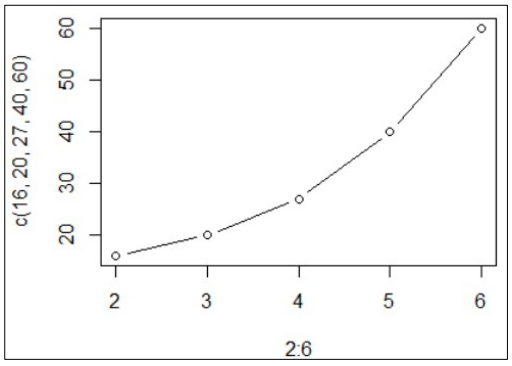
- mfcol: 同上，按列填充

- 全局参数修改演示

```{r, eval = F}
opar <- par(no.readonly = T)
par(lty = 2, pch = 17)
plot(2:6, c(16, 20, 27, 40, 60), type = "b")
```




```
par(opar)
plot(2:6, c(16, 20, 27, 40, 60), type = "b")
```

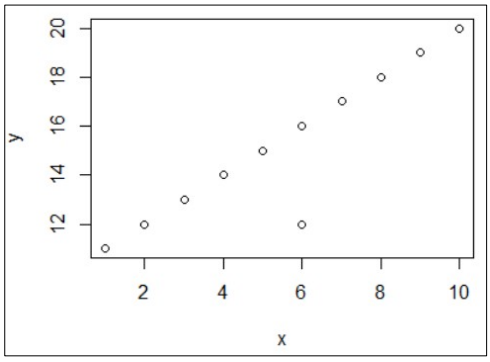



```

# ## 基础绘图函数

# - plot()

- plot(): 绘制散点图，根据绘制对象的特定类型可以绘制其他类型的图

# - lines()

- lines(): 在图上加线条，给出 x 向量和与之对应的 y 向量，或者是一个 2 列矩阵，此函数仅仅连接点
```

- points()

- points(): 在图上增加点

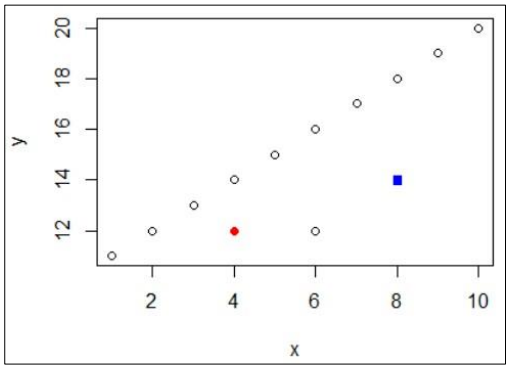
```
x <- 1:10;y <- 11:20
plot(x, y)
points(6, 12)  #添加一个点，也可以使用向量的方式添加多个点
```



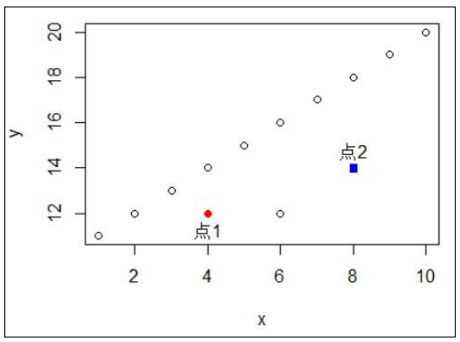
- text()

- text():在图形内增加文本标签，给定 x, y 坐标, *text(x, y, "label", pos)*, pos 文本相对于点的位置，取值:1234-下左上右

```
# 为了演示 text 的功能，先添加 2 个辅助的点
points(c(4, 8), c(12, 14), pch = c(21,22), col = c("red", "blue"), bg = c("red", "blue"))
```

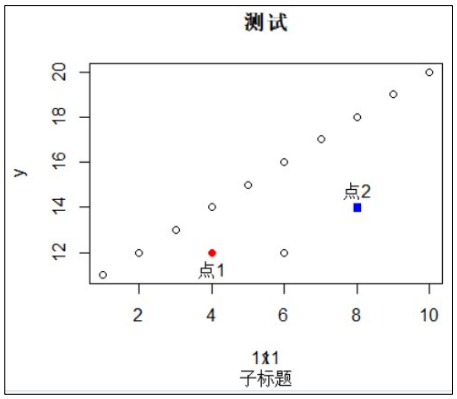



```
text(c(4, 8), c(12, 14), labels = c("点 1", "点 2"), pos = c(1, 3))  # 添加 2 个点(4, 12), (8, 14)，两个点的 x, y 坐标分开定义. 在某个点处添加标签，只是在某点处添加标签，不添加点.
```



- title()

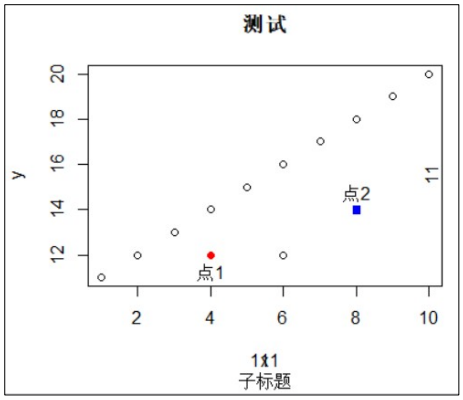
- title(): 用于在图形外添加注释, main 标题; sub 子标题; xlab, ylab: x, y 轴标签
title(main = "测试", sub = "子标题", xlab = "111") #主标题在图形的最上边, 副标题在 x 轴的下方, xlab 是 x 轴标签, 如果已存在 x 标签, 就会覆盖. 可以在之前的作图中使用 xlab = NA 来使 x 轴标签为空.



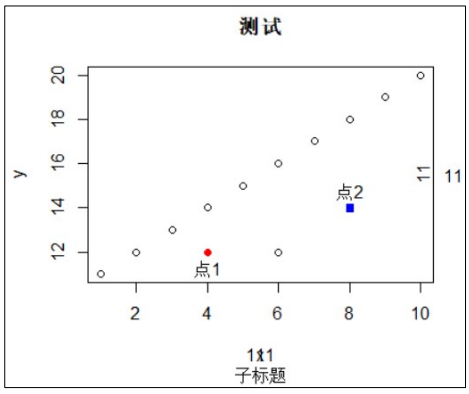
- mtext()

- mtext(): 表示将文本放在图形的页边距中, 可以是内边距或者外边距, *mtext("text", side, line)*, side 指定放置文本的边:1234-下左上右; line 来内移或者外移文本

mtext("11", side = 4, line = -1) #在图形右边添加文本, line 表示相对于坐标轴的位置, 这里表示相对于右边坐标轴是-1, 即在右坐标轴的内部.

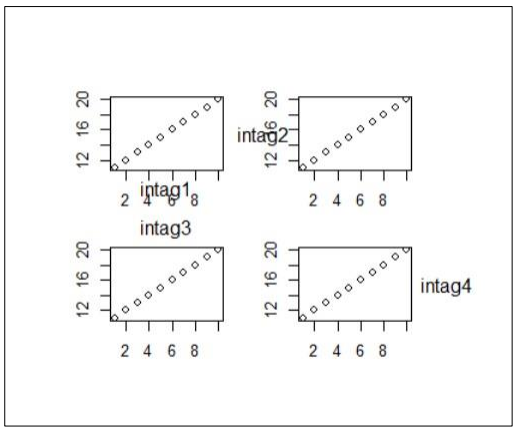


mtext("11", side = 4, line = 0.5, las = 1)　#line 大于 0 表示添加在坐标轴的外边, las 表示标签为水平方向.

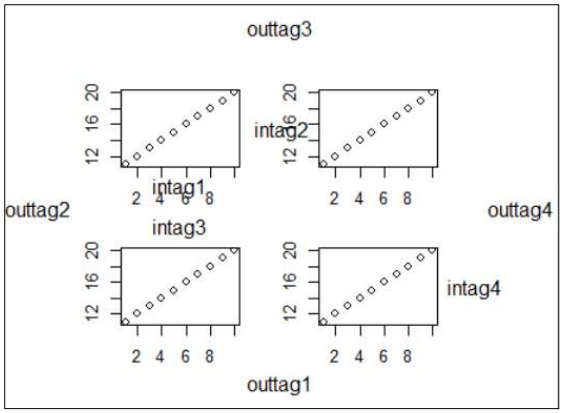


#如果是想要在外边距添加标签，必须要在多图的条件下，使用 outer = T 即可．如下例所示.

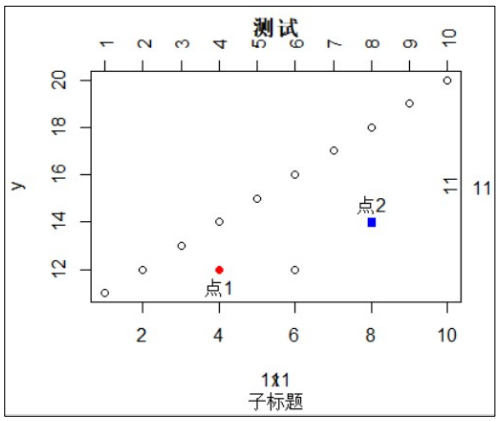
```{r}
x1 <- 1:10;y1 <- 11:20;

opar <- par(no.readonly1 = T);
par(mfrow = c(2, 2), oma = c(4, 4, 4, 4), mar = c(2, 2, 2, 2));

plot(x1, y1);mtext("intag1", side = 1, line = 0.5, las = 1);
plot(x1, y1);mtext("intag2", side = 2, line = 0.5, las = 1);
plot(x1, y1);mtext("intag3", side = 3, line = 0.5, las = 1);
plot(x1, y1);mtext("intag4", side = 4, line = 0.5, las = 1);
```



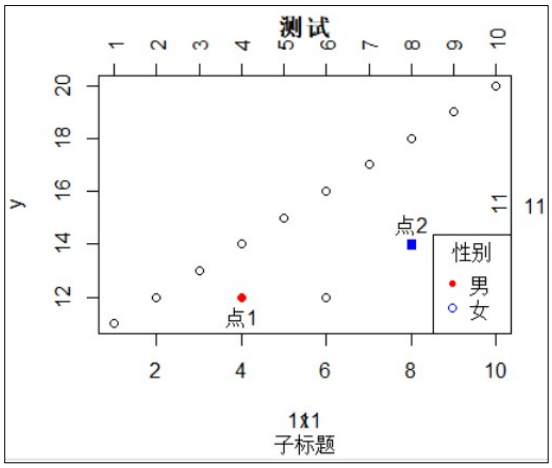

```
mtext("outtag1", side = 1, line = 0.5, las = 1, outer = T);
mtext("outtag2", side = 2, line = 0.5, las = 1, outer = T);
mtext("outtag3", side = 3, line = 0.5, las = 1, outer = T);
mtext("outtag4", side = 4, line = 0.5, las = 1, outer = T);
```



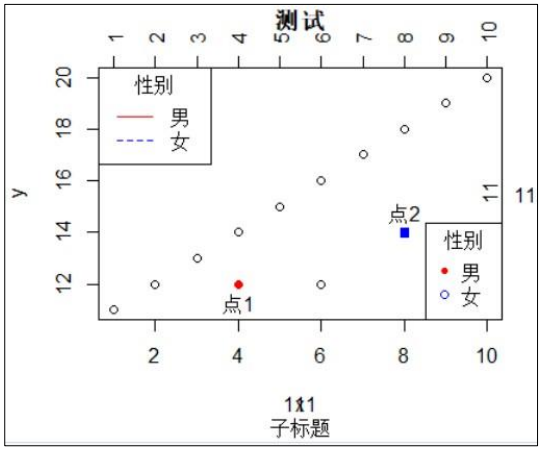
par(opar);

```

# - axis()

- axis():用于设定坐标轴, side 哪边绘制坐标轴(1 下 2 左 3 上 4 右); at 绘制刻度线的位置;labels 刻度线的文本;las 表示 labels 的方向.

axis(side = 3, at = 1:10, labels = 1:10, las = 3)  #at 坐标轴刻度的值




# - abline()

- abline():  加直线或者回归线, h 水平线;v 垂直线

# - legend()

- legend():图例, 四个角 topleft, topright, bottomleft, bottomright.例 legend("topleft", pch, col, legend, title), 其中 legend 表标签, title 表标题. legend 是对图中不同曲线的说明. 如红色表示男, 绿色表示女

# pch = 20 表示实心点, pch = 1 表示空心圆.pch = c(20, 1), col = c("red", "blue") 上面的图例是红色实心点, 下面的图例是蓝色空心圆. legend 为图例名称. 'arg' 可以取 "bottomright",
```

"bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"
legend("bottomright", pch = c(20, 1), col = c("red", "blue"), legend = c("男", "女"), title = "性别")



#想要使用直线和点线来表示男女，把 pch 改为 lty 即可.
legend("topleft", lty = c(1, 2), col = c("red", "blue"), legend = c("男", "女"), title = "性别")



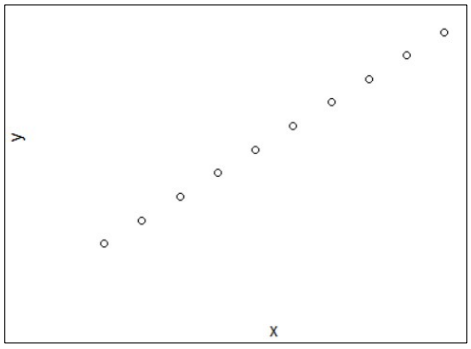
特殊参数

- type = "n" 不绘制图形

- axes = F 禁用全部坐标轴

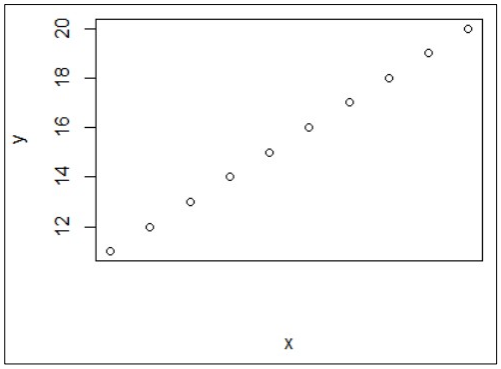
x <- 1:10; y <- 11:20

plot(x, y, axes = F)

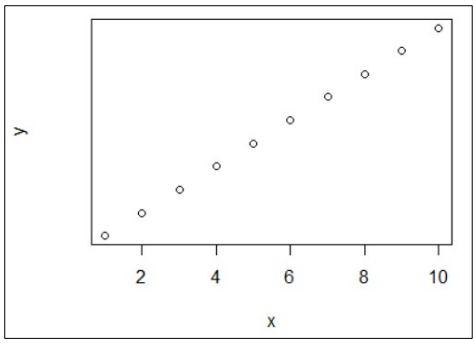


- xaxt = "n", yaxt = "n" 分别禁用 x, y 轴

plot(x, y, xaxt = "n")

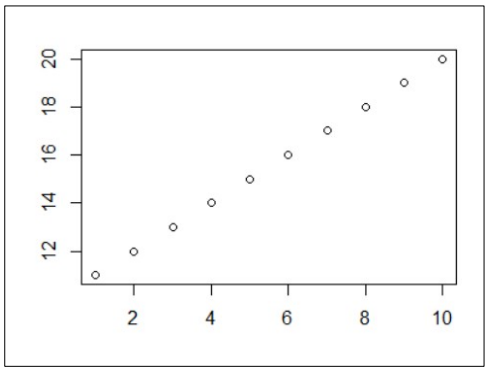


plot(x, y, yaxt = "n")



- ann = F 移除全部注释

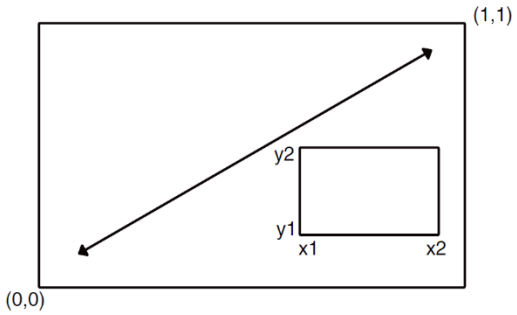
plot(x, y, ann = F) # annotation



图形组合及布局

不好用，一般不使用.

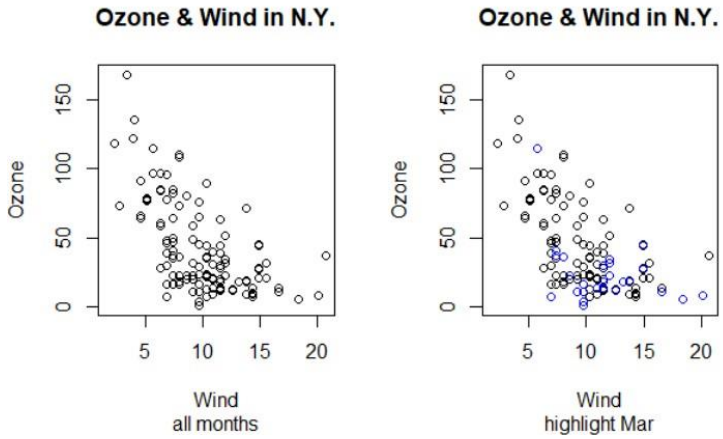
- 回顾 mfrow, mfcol
- fig = c(x1, x2, y1, y2):指明图形所在的绘图区域，在 par()里设置，添加新图时同时添加 new = T



基本图形注释

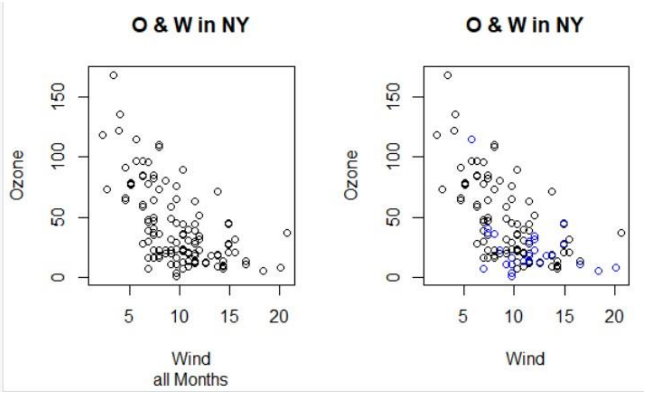
- title 标题

```{r, fig.height = 3.5}
par(mfrow = c(1, 2)) #一行两列
library(datasets)
#画第一个图形
with(airquality, plot(Wind, Ozone)) #Ozone(Wind)图
title(main = "Ozone and Wind in New York City", sub = "all months") ## 添加主标题和副标题
#画第二个图形

with(airquality, plot(Wind, Ozone, main = "Ozone and Wind highlight Mar"))
with(subset(airquality, Month == 5), points(Wind, Ozone, col = "blue"))
```



视频演示

```{r}
attach(airquality)
par(mfrow = c(1, 2))

plot(Wind, Ozone)
title(main = "O & W in NY", sub = "all Months")

plot(Wind, Ozone, main = "O & W in NY") #先画出所有的图，再把 5 月份的图颜色设置为蓝色，实际上是单独把 5 月份的图抽取出来，覆盖掉原来图中对应的 5 月份的点.
with(airquality[Month == 5, ], points(Wind, Ozone, col = "blue"))
par(opar)
```



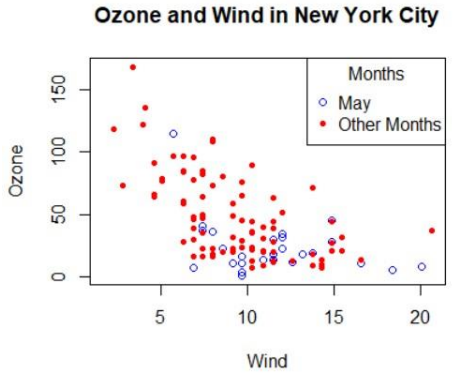
- legend 图例

```{r, fig.height = 3.5}

#画一个空图，然后再添加点. 也可以先画好 5 月份，再画其它月份的.
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City", type = "n"))
```

with(subset(airquality, Month == 5), points(Wind, Ozone, col = "blue")) #5 月份蓝色空心点
with(subset(airquality, Month != 5), points(Wind, Ozone, col = "red", pch = 20)) #其它月份为红色实心点

#添加图例
legend("topright", pch = c(1, 20), col = c("blue", "red"), legend = c("May", "Other Months"), title = "Months")



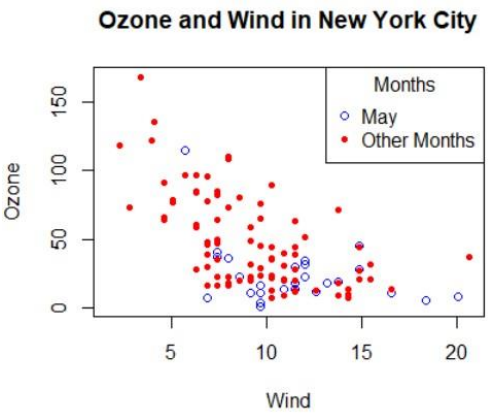

```

视频演示，先画 5 月份的图，再添加其它月份的

```{r}
with(airquality[Month == 5, ], plot(Wind, Ozone, col = "blue", main = "O&W in NY"))

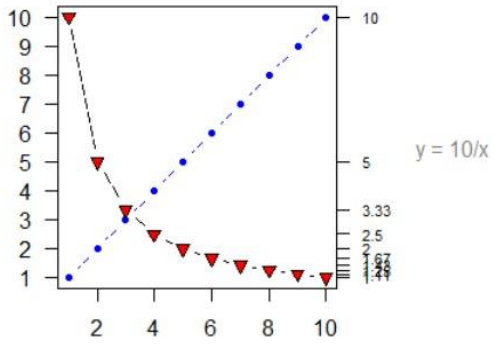
with(airquality[Month != 5, ], points(Wind, Ozone, pch = 20, col = "red"))

legend("topright", pch = c(1, 20), col = c("blue", "red"), legend = c("May", "Other Months"), title = "Months")
```


```

- mtext 标签

```{r, fig.height = 3.5}
x <- 1:10; y <- x; z <- 10/x
opar <- par(no.readonly = T)
par(mar = c(5, 4, 4, 8))
plot(x, y, type = "b", pch = 20, col = "blue", yaxt = "n", lty = 4, ann = F)
lines(x, z, type = "b", pch = 25, bg = "red", lty = 5)
axis(2, at = x, labels = x, las = 2);axis(4, at = z, labels = round(z, 2), las = 1, cex.axis = 0.7)
mtext("y = 10/x", side = 4, line = 3, las = 2, col = rgb(0.5, 0.5, 0.5))
par(opar)
```



视频演示

```{r}
x <- 1:10;y <- x;z <- 10/x

opar <- par(no.readonly = T)
par(mar = c(5, 4, 4, 8)) #内边界宽度

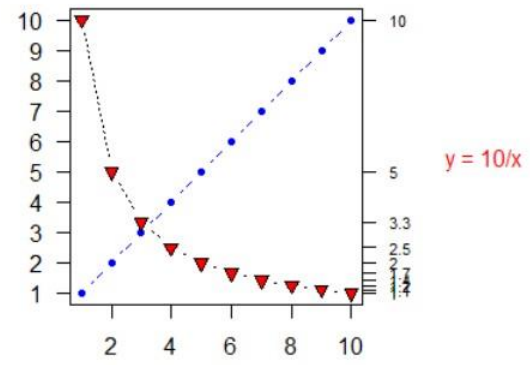
plot(x, y, type = "b", pch = 20, lty = 4, ann = F, col = "blue", yaxt = "n") #type = "b"点线图, pch = 20 实心点, lty = 4 线型. ann = F 去掉所有的注释, yaxt 去掉 y 轴坐标轴.

lines(x, z, type = "b", pch = 25, lty = 3, bg = "red")  #在原有的图形的添加线. pch = 25 是倒三角, 线型 3, 图形填充为红色.

axis(2, at = y, labels = y, las = 2) #2 表示在左边画坐标轴, at 在哪些位置画坐标轴, at = y 个表示在 1, 2, 3, ... 10 画坐标轴. 坐标轴标签为 y, las 表示标签为垂直于坐标轴.

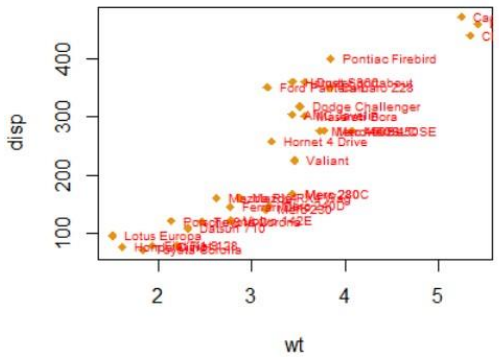
axis(4, at = z, labels = round(z, digits = 1), cex.axis = 0.7, las = 1) #右边坐标轴, 在 z 上画坐标轴, 坐标轴标签也为 z, 由于 z 的小数位数太多了, 要四舍五入一下, 只保留 1 位小数. cex.axis 表示坐标轴刻度的的大小. las = 1 坐标轴标签垂直于坐标轴.
```

mtext("y = 10/x", side = 4, line = 3, las = 1, col = "red") #在右边页边距添加标签, line 表示距离坐标轴的位置. las 表示标签文字的方向
par(opar)
```




# - text 文本标注

```{r, fig.height = 4}
with(mtcars, {plot(wt, disp, pch = 18, col = rgb(229/255, 147/255, 17/255))
text(wt, disp, rownames(mtcars), cex = 0.6, pos = 4, col = "red")})
```

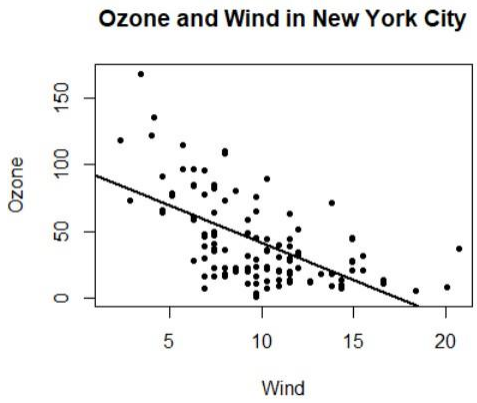


视频演示

```{r}
View(mtcars)
?mtcars
with(mtcars, plot(wt, disp, pch = 18, col = "orange"))  #画 disp(wt)的散点图, 点形状为 18.
#text(x, y, "label", pos)
text(mtcars$wt, mtcars$disp, rownames(mtcars), pos = 4, cex = 0.7, col = "red")  #给每一个点添加一个名字, 在图内添加标签. pos 为点标签的位置, 4 表示点的右边. cex 为图内符号的大小. 因为之前没有 attach(mtcars), 所以这里要加上 mtcars$wt
```
```

```

## 添加回归线

```{r, fig.height = 3.5}
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City", pch = 20))
model <- lm(Ozone ~ Wind, airquality)
abline(model, lwd = 2)
```



视频演示

```{r}
plot(Wind, Ozone, pch = 20, main = "标题")
mylm <- lm(Ozone~Wind, data = airquality); mylm    #lm 做线性回归分析, linear model.
```

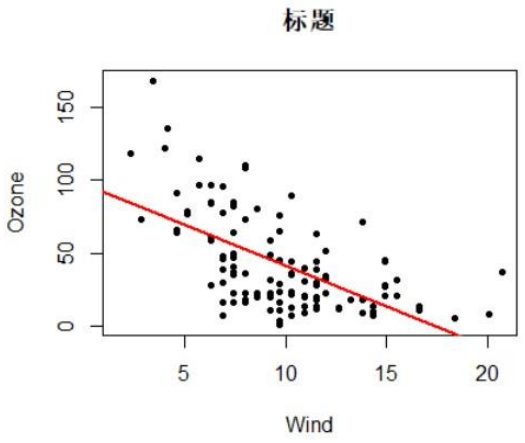
```
Call:
lm(formula = Ozone ~ Wind, data = airquality)

Coefficients:
(Intercept)         Wind
     96.873       -5.551
```

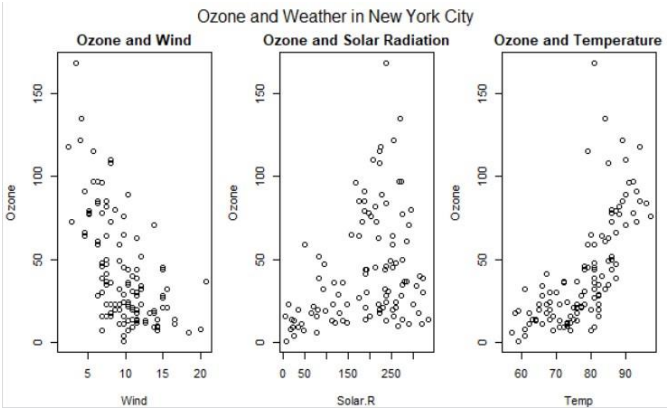
```
summary(mylm)
```

Call:
lm(formula = Ozone ~ Wind, data = airquality)

Residuals:
Min	1Q	Median	3Q	Max
-51.572	-18.854	-4.868	15.234	90.000

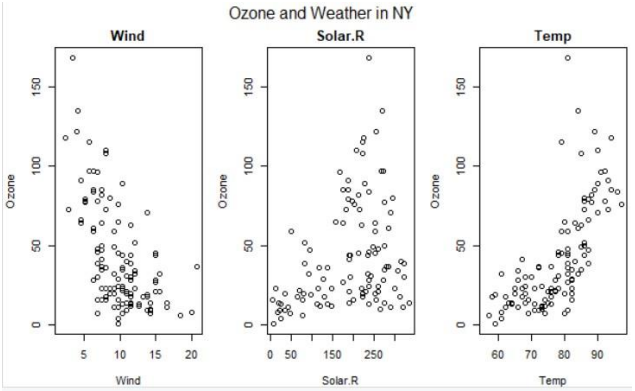
Coefficients:
```

```
                Estimate  Std.  Error  t  value  Pr(>|t|)
(Intercept)    96.8729          7.2387      13.38    <  2e-16  ***
Wind                -5.5509          0.6904      -8.04  9.27e-13  ***
---
Signif.  codes:    0  '***'  0.001  '**'  0.01  '*'  0.05  '.'  0.1  '  '  1

Residual  standard  error:  26.47  on  114  degrees  of  freedom
   (37  observations  deleted  due  to  missingness)
Multiple  R-squared:    0.3619,    Adjusted  R-squared:    0.3563
F-statistic:  64.64  on  1  and  114  DF,    p-value:  9.272e-13
```

```
abline(lm(Ozone~Wind), lwd = 2, col = "red") #添加回归线
#abline(mylm, col = "red") #方法二
```




```
```

绘制多个图形

```
```{r, fig.height = 2.8, fig.width = 8}
par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))
with(airquality, {
      plot(Wind, Ozone, main = "Ozone and Wind")
      plot(Solar.R, Ozone, main = "Ozone and Solar Radiation")
      plot(Temp, Ozone, main = "Ozone and Temperature")
      mtext("Ozone and Weather in New York City", outer = TRUE)
      ## 为三个图形添加一个总标题
})
```
```



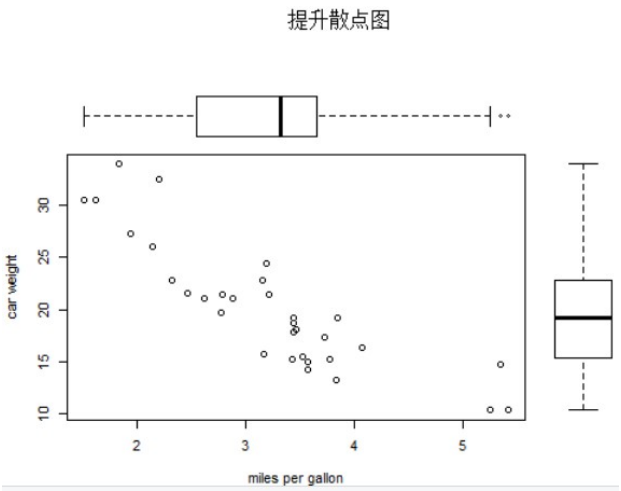
视频演示

```{r}
opar <- par(no.readonly = T)
par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))  #mfrow = c(1, 3)表示 1 行 3 列, mar 为内边距, oma 为外边距

plot(Wind, Ozone, main = "Wind")
plot(Solar.R, Ozone, main = "Solar.R")
plot(Temp, Ozone, main = "Temp")

mtext("Ozone and Weather in NY", outer = T) #添加主标题 outer = T, 表示在外边距上添加标题. 否则标题默认是添加最后一个图形的内边距上的.
par(opar)
```



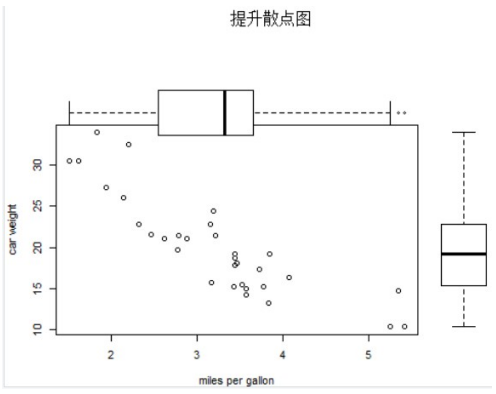

```{r, eval = F}
opar <- par(no.readonly = T)
par(fig = c(0, 0.8, 0, 0.8))
with(mtcars, plot(wt, mpg, xlab = "miles per gallon", ylab = "car weight"))
par(fig = c(0, 0.8, 0.55, 1), new = T)
with(mtcars, boxplot(wt, horizontal = T, axes = F))
par(fig = c(0.65, 1, 0, 0.8), new = T)
with(mtcars, boxplot(mpg, axes = F))
```

mtext("提升散点图", outer = T, side = 3)
par(opar)




```

```{r, echo = F}
opar <- par(no.readonly = T)
par(fig = c(0, 0.8, 0, 0.8))
with(mtcars, plot(wt, mpg, xlab = "miles per gallon", ylab = "car weight"))
par(fig = c(0, 0.8, 0.45, 1), new = T)
with(mtcars, boxplot(wt, horizontal = T, axes = F))
par(fig = c(0.65, 1, 0, 0.8), new = T)
with(mtcars, boxplot(mpg, axes = F))
mtext("提升散点图", outer = T, side = 3)
par(opar)
```



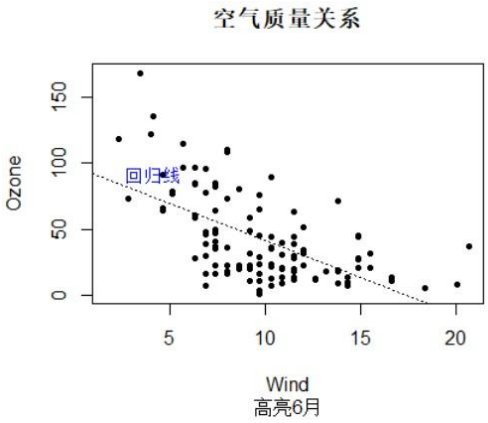

```

# ## Practices

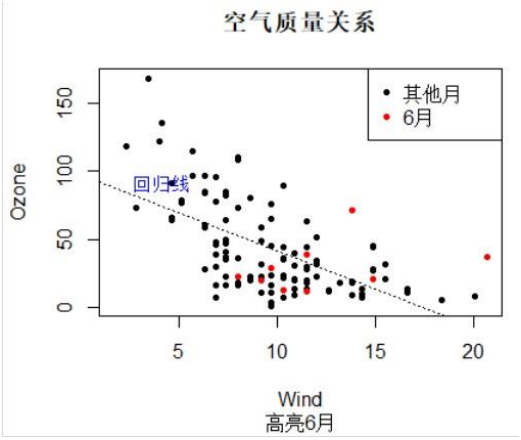
准确的绘制图形，用到 airquality 数据集，红色高亮 6 月的数据，做所有月份数据回归线，在坐标(2, 90)处添加 0.8 倍大小的文本"回归线"，添加 legend 图例，细节参考下图

```{r, echo = F, fig.height = 4}
```

```
with(airquality, {plot(Wind, Ozone, pch = 20, main = ("空气质量关系"), sub = "高亮 6 月")
    abline(lm(Ozone~Wind), lty = 3)
    text(2, 90, "回归线", cex = 0.8, pos = 4, col = "blue")
    })
```



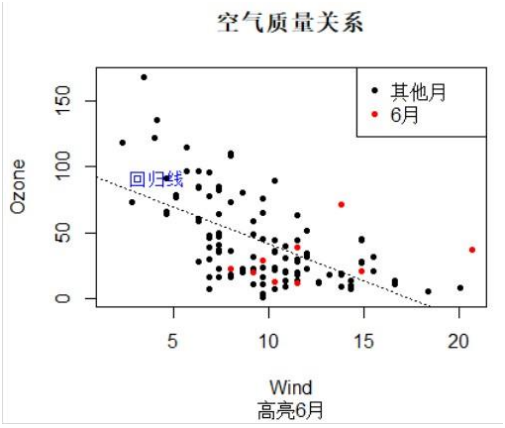

```
with(subset(airquality, Month == 6), {points(Wind, Ozone, pch = 20, col = "red")
    legend("topright", pch = 20, col = c("black", "red"), legend = c("其他月", "6 月"))
    })
```




```
```

视频演示

```
attach(airquality)
plot(Wind, Ozone, pch = 20, main = "空气质量关系", sub = "高亮 6 月")
with(airquality[Month == 6, ], points(Wind, Ozone, pch = 20, col = "red"))
legend("topright", pch = 20, col = c("black", "red"), legend = c("其他月", "6 月"))
abline(lm(Ozone~Wind), lty = 3)
text(2, 90, "回归线", col = "blue", cex = 0.8, pos = 4)
detach(airquality)
```



R 中的图形设备

图形设备

- 图形设备是用来显示图形的
    - 电脑上的窗口(屏幕设备)
    - 文件设备:PDF, PNG, JPEG, SVG(可缩放矢量图)格式等
- 当你在 R 中作图时，图形必须被传到特定的图形设备中，否则图形无法生成
- 图形最常发送的目的地是屏幕设备
    - 在 Mac 上屏幕设备由 quartz()启动
    - 在 Windows 上屏幕设备由 windows()启动
    - 在 Unix/Linux 上屏幕设备由 x11()启动

- 作图时首先要决定在哪里显示图像
    - 在设备帮助页面?Devices 中，可以看到完整的设备清单，也有一些图形设备是 CRAN 上的用户创建的，通过相应的扩展包可以使用
- 进行可视化和探索性分析时，通常使用屏幕设备
    - 常用函数 plot(), xyplot(), ggplot()会默认将图像发送到屏幕设备上
    - 在一个给定的操作系统中只有一个屏幕设备
- 当图形需要打印或者配合文档使用的时候(论文，报告，展示)，这时候就需要用到文件设备
    - 有各种各样可选择的文件设备
- 注意:并不是所有的图形设备都适用于所有的平台(例如屏幕设备只能只能对应特定的系统)

生成图像

有两种最基本的制图方法，最常见的是

1. 调用绘图函数 plot(), xyplot, qplot()等

2. 图像显示在屏幕上
3. 添加注释
4. Enjoy

第二种常用于文件设备上

1. 明确的打开一个文件设备
2. 调用作图函数绘图(注意如果利用文件设备，屏幕上不显示图形)
3. 添加注释
4. 明确的关闭设备，通过 dev.off()函数(必须执行)

示例

```{r, eval = F}
pdf(file = "myplot.pdf") ## 打开 pdf 设备，在工作目录下创建"myplot.pdf"文件
with(faithful, plot(eruptions, waiting)) ## 创建图像并且发送到文件，屏幕上不显示
title(main = "Old Faithful Geyser data") ## 添加注释，屏幕上不显示
dev.off() ## 关闭 pdf 设备，在工作目录下可以找到创建的 pdf 文件
```

视频演示，把图形输出到设备上

```{r}
png(file = "myplot.png", width = 720, height = 720) #打开一个 png 设备，图形默认大小是 480
plot(Wind, Ozone, pch = 20, main = "标题") #在打开的设备中画图
abline(lm(Ozone~Wind), lwd = 2)
dev.off() #必须要关闭设备才会把图像写入到设备中.

```

?png #查看设备帮助
?pdf

图形设备

可以把图形设备分为两大类，一类是矢量格式(vector format)，另一类是位图(bitmap)

矢量格式

- pdf:用于线状图形，大小可调，移植性好，点多了不合适(文件很大)

- svg:可缩放矢量图形格式，支持动画和交互，几乎适用所有浏览器
- win.metafile:图元文件格式，只能在 win 系统中使用的矢量格式
- PostScript:老旧的格式, pdf 的前身，不常用

位图格式

- png:便携网络图形(portable network graphics)，图形和图片都由一系列像素表征，适合含有大量点的图形以及线条和纯色图像之类的图形，使用一种无损压缩算法(lossless compression algorithm)，文件体积通常比较小适用于几乎所有浏览器.缩放性不强
- jpeg:适合照片或者自然场景类的图像(有渐变色，非单色)，采用有损压缩算法(lossy compression algorithm)，文件体积可以有效的压缩到很小，也适用于大量点，缩放性不强，最好不要缩放，适用于几乎所有浏览器，不太适用于线条图，一些线条可能会失真
- tiff:较为老旧的位图格式文件，支持无损压缩
- bmp:windows 使用的位图文件格式，常用在图表图形上

打开多个图形设备

- 可以同时打开多个图形设备(屏幕，文件或者一起)，比如说同时生成多个图像，还要观察，就要同时打开多个屏幕设备
- 每次只能在一个设备上进行绘制
- 这在绘制的图形设备就是当前活动的图形设备，可以通过 dev.cur()来查看当前的活动设备(current)
- 每个打开的图形设备会被赋值一个大于等于 2 的整数，所以以上函数返回的是一个整数
- 可以通过 dev.set(整数)来更改当前的图形设备，输入图形设备对应的整数值

复制图形

当需要更改设备的时候，直接复制图形到另一个设备就会显得很有用，比如在屏幕上产生了图形，然后想写入文件中，可以复制粘贴代码，也可以用如下函数直接将图形从屏幕设备复制到文件设备中

- dev.copy:将图形从一个设备复制到另一个设备
- dev.copy2pdf:图形从屏幕设备复制进 pdf 文件

注意:文件中得到的图形可能和屏幕上看到的图形有细微的差距

```{r, eval = F}
library(datasets)
with(faithful, plot(eruptions, waiting)) ## 在屏幕上创建图形
title(main = "old faithful geyser data") ## 添加注释
dev.copy(png, file = "geyserplot.png") ## 第一个参数是打开文件设备的函数，会显示 png 为第 4 个设备
```

dev.off() ## 一定要关闭设备，关闭后显示当前设备为 RstudioGD 为第 2 个设备
```

视频演示，把 Rstudio 当前显示的图形复制到设备中.
```{r}
dev.copy(png, file = "myplot1.png", width = 480, height = 360)
dev.off()
```

总结

- 所有图形都要通过图形设备才能生成
- 默认图形设备是屏幕设备，用于探索性分析
- 想要保存图像或发送给他人时需要在文件设备中生成
- 对于文件设备，有矢量格式和位图格式
    - 矢量格式用于线条绘制，中等数量的点，固定的颜色
    - 位图格式适用于大量点，自然图形和网络图形

Practices

- 将上一个练习绘制好的图形分别输出到名为"lala.png"的 png 文件上和"lala.pdf"的 pdf 文件上
attach(airquality)
plot(Wind,Ozone,pch=20,main = "空气质量关系",sub="高亮 6 月")
with(airquality[Month==6,],points(Wind,Ozone,pch=20,col="red"))
legend("topright",pch=20,col=c("black","red"),legend=c("其他月","6 月"))
abline(lm(Ozone~Wind),lty=3)
text(2,90,"回归线",col="blue",cex=0.8,pos=4)
detach(airquality)

Lattice 绘图系统

lattice 绘图系统

和基础绘图系统运行原理不同，通常用于高维数组制图和一次性绘制多图，可以优化的创建非常高密度的图

lattice 函数
```

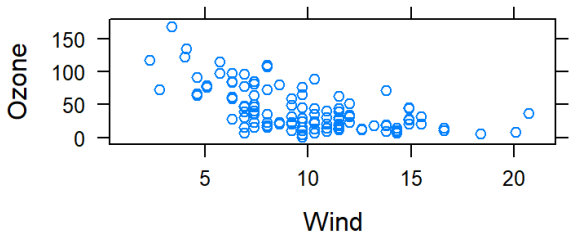
- xyplot:最重要的函数，用来做散点图
- bwplot:用来做箱线图(box-and-whiskers plots: boxplots)
- histogram:直方图
- stripplot:和 bwplot 很像，以点画图
- dotplot:看起来像是在小提琴琴弦上画点
- splom():散点图矩阵，和基础绘图系统中的 pairs()有点像
- levelplot(), contourplot():用来绘制"图形"数据

lattice 函数一般用公式作为第一个参数

```r
xyplot(y~x|f*g, data)
```

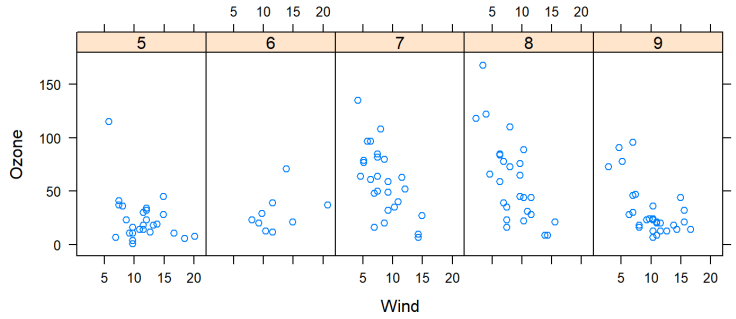
- 公式中~左边是 y 右边是 x, |右边 f*g 是作为条件的分类变量(可选)
    - f*g 中的*表示两个变量的交互(interaction)
- 第二个参数是数据，或者是包含前面公式中变量的列表
    - 如果没有传递参数，会在工作空间中寻找变量

```{r, fig.height = 2, fig.width = 4}
library(lattice);library(datasets);xyplot(Ozone~Wind, data = airquality)
```



简单的 lattice 图

```{r, fig.height = 3.5}
library(lattice);library(datasets)
airquality <- transform(airquality, Month = factor(Month))
xyplot(Ozone ~ Wind | Month, data = airquality, layout = c(5, 1))
```



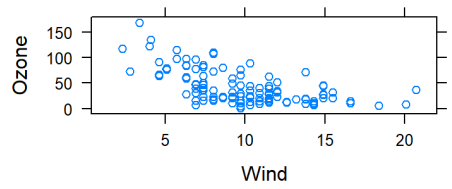
风和臭氧之间的关系，可见非常方便的创建了多维面板
```

# ## lattice 运行方式

lattice 绘图系统的运行方式和基础绘图系统相比不太一样

- 基础绘图函数直接将数据绘制到图形设备中
- lattice 绘图系统并没有确切的画出任何图，而是返回一个栅栏对象(trellis)，然后自动输出这个对象，这样图形的数据才被传递到图形设备中去，知道原理即可

```{r, fig.height = 2, fig.width = 4}
p <- xyplot(Ozone ~ Wind, data = airquality) ## 什么都没发生
print(p)  ## 输出图形
```



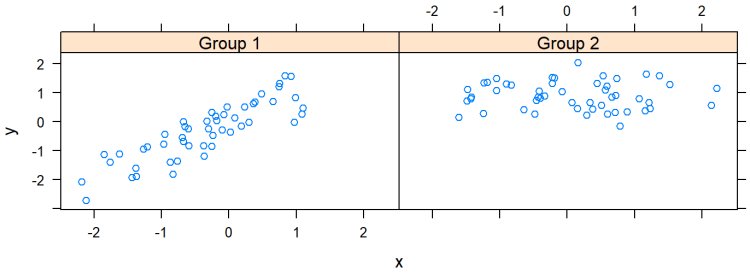
lattice 面板函数

- lattice 绘图系统包含所谓的**面板函数**，当存在多个面板时，这个函数可以让你在各个面板中分别进行绘图
- lattice 包带有许多默认的面板函数，也可以根据需要自定义各个面板中的行为
- 每个面板函数都会接受特定面板中数据点的 x/y 值，每个面板代表数据的一个子集

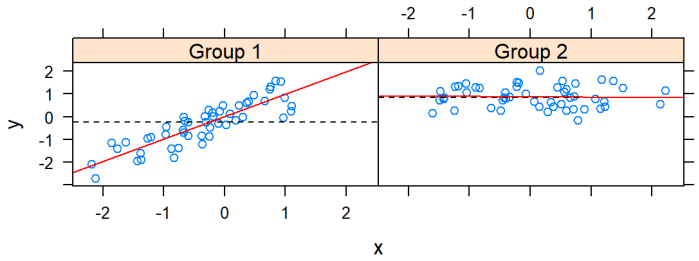
下面的例子生成一些遵循线性模型的随机数据

```{r, fig.height = 3}

```
set.seed(10)
x <- rnorm(100);f <- rep(0:1, each = 50)
y <- x + f - f * x+ rnorm(100, sd = 0.5)
f <- factor(f, labels = c("Group 1", "Group 2"))
xyplot(y ~ x | f, layout = c(2, 1))   ## 绘制两个面板
```




```{r, fig.height = 2.5, fig.width = 6}
## 自定义面板函数
xyplot(y~x|f, panel = function(x, y, ...){
        panel.xyplot(x, y, ...)    ## 调用默认的 xyplot 面板函数使数据点出现
        panel.abline(h = median(y), lty = 2)    ## 在每个面板加了一条中位数的水平线
        panel.lmline(x, y, col = 2) ## 加一条回归线
})
```



不能使用基础绘图系统中的任何函数，不能混用绘图系统中的函数

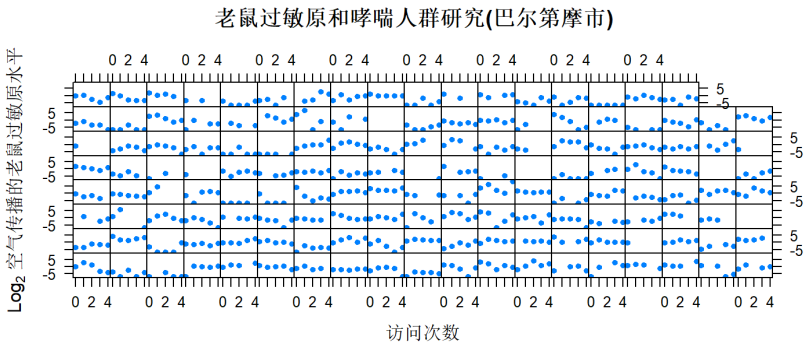
MAACS 例子

- 研究:一项对老鼠过敏原和哮喘人群的研究(Mouse Allergen and Asthma Cohort Study, MAACS)
- 研究对象:观察了巴尔第摩市哮喘儿童的室内居住环境，许多儿童对老鼠过敏
- 设计:观察性研究，研究包括一次最初的基准家访，之后一年中每三个月再进行一次家访，共五次
- 问题:对于每个受试者来说室内空气中的过敏原数量是如何随时间变化的
- 150 个受试者，每个 5 次家访，这里利用多面板 lattice 绘图

[Ahluwalia et al., *Journal of Allergy and Clinical Immunology*, 2013](http://www.ncbi.nlm.nih.gov/pubmed/23810154)

多面板图

```{r, fig.height = 3.2}
library(lattice);env <- readRDS("data/maacs_env.rds")
env <- transform(env, MxNum = factor(MxNum))
xyplot(log2(airmus)~VisitNum|MxNum, data = env, strip = FALSE, pch = 20, xlab = "访问次数",
       ylab = expression(Log[2] * "  空气传播的老鼠过敏原水平"),
       main = "老鼠过敏原和哮喘人群研究(巴尔第摩市)")
```



这里 strip 表示每个水平上面的条形标注，见?xyplot

总结

- Lattice 绘图是通过调用一个核心 lattice 函数(例如 xyplot())
- 边界，间距，标签等都是自动设置的而且通常满足使用
- 当需要控制某些条件变量来观测数据时, lattice 绘图系统是理想的选择
- 面板函数通过修改参数用来自定义每个面板中的图形
- lattice 绘图系统在查看大量数据的时候非常快速，大家可以尝试下其他的 lattice 函数

ggplot2 绘图系统

ggplot2 是什么

- 兰.威尔金森(Leland Wilkinson)提出的*绘图语法*(Grammar of Graphics)，绘图语法是如何把图形分解为抽象概念的描述，可以把这些抽象概念组合出新的图形类型(就像语言语法中的动词名词等)

- 爱荷华州立大学的哈德利.韦翰(Hadley Wickham)在其研究生阶段将绘图语法变成了 ggplot 包, 这个包的最新版本叫做 ggplot2
- 官网:http://ggplot2.org
- 基本思想是:*缩短从思维到纸张的距离*

来自书籍:ggplot2

"简单来说, 绘图语法告诉我们统计图形就是一种从数据到**几何形状(点, 线, 柱)**的**美学属性(aesthetic)**(颜色, 形状, 大小)的**映射**, 绘图也许还包括了数据的统计转换, 并绘制在特定的坐标系上"

先给定数据, 然后再把数据映射到它的美学属性上. 即先给出数据做出一个基础的图形, 再逐个向图形上添加点, 线, 条件, 主题, 标签等内容. 只是在添加美学属性之前已经指定了数据是什么.
ggplot2 中其实有两套绘图系统, 一套类似于 lattice, 可以直接出图, 另一套类似于基础绘图系统, 一步步添加元素完成作图.

基本函数:qplot()

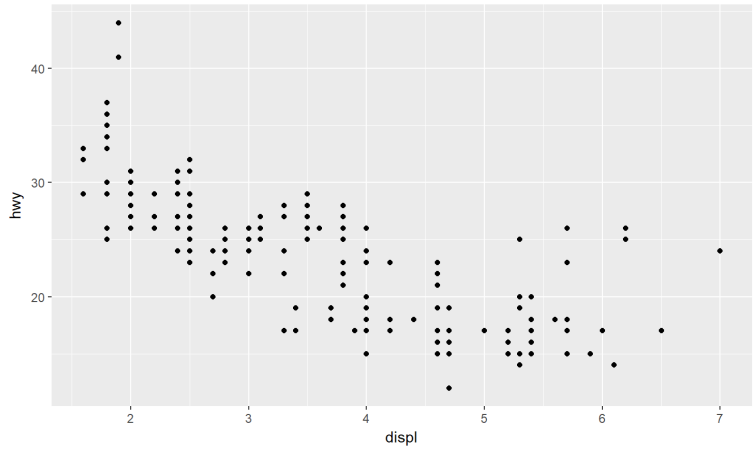
- 和基础绘图系统中的 plot 函数很像
- 关键的不同点在于数据需要被整理成一个数据框, 绘制变量的时候, 变量都将来自数据框, 如果不指定数据框, 会在当前工作环境中查找数据(和 lattice 很像)
- 图形由美学 aesthetics 和几何 geoms 组成
- 使用因子变量的思想, 因子能指出数据的子集, 因子需要有正确的可解释的标签
- qplot()隐藏了许多 ggplot 在后台操作的细节
- ggplot()才是系统的核心函数, 非常灵活, 可以和许多东西组合使用, 这是 qplot()无法实现的

案例

```{r}
library(ggplot2)
str(mpg) ##  不同车型的每加仑公里数数据
```
```
## Classes 'tbl_df', 'tbl' and 'data.frame':	234 obs. of  11 variables:
##  $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
##  $ model       : chr  "a4" "a4" "a4" "a4" ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
```

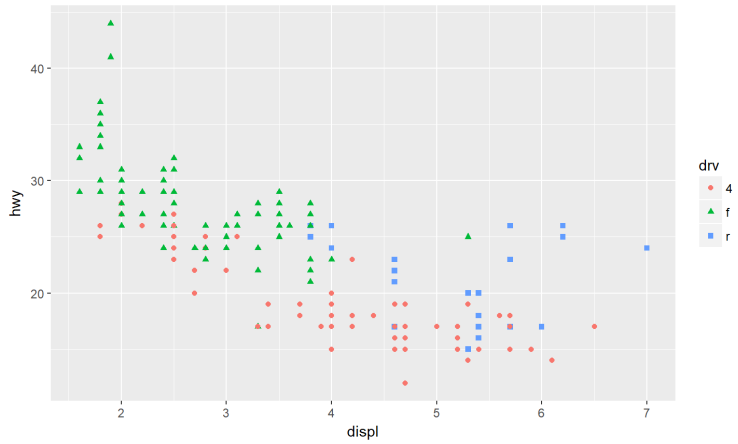
```
##  $ drv          : chr  "f" "f" "f" "f" ...
##  $ cty          : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy          : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl           : chr  "p" "p" "p" "p" ...
##  $ class        : chr  "compact" "compact" "compact" "compact" ...
```
displ:排量;hwy:每年公里数;drv:4 四轮驱动, f 前轮驱动, r 后轮驱动

```{r}
qplot(displ, hwy, data = mpg)
```



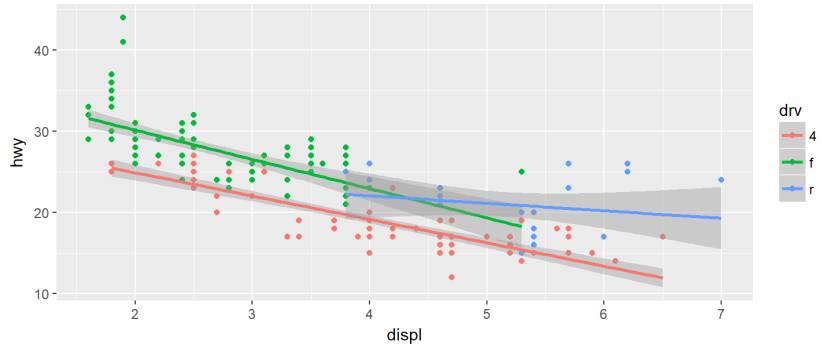
美学属性修改

```{r}
qplot(displ, hwy, data = mpg, color = drv, shape = drv)
```



添加几何属性

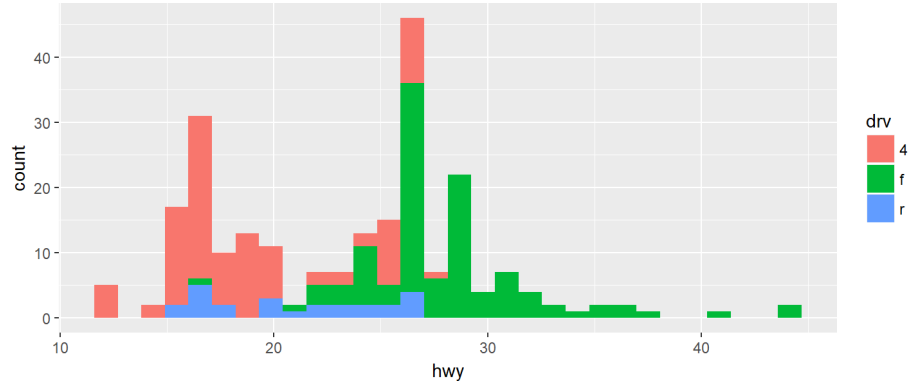
```{r, fig.height = 3.2, warning = F}
qplot(displ, hwy, data = mpg, color = drv, geom = c("point", "smooth"), method = "lm")
```



增加了 geom(几何属性)参数，想在图中添加两种几何属性:1.点;2.光滑(smooth)属性(使用的模型是线性回归模型 lm，不指定 method 是光滑的拟合曲线):线，灰色区间是 95%置信区间

直方图

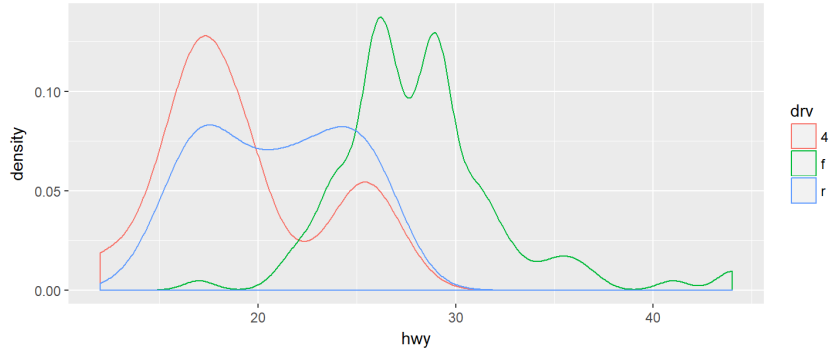
```{r, fig.height = 3.2, warning = F}
qplot(hwy, data = mpg, fill = drv) ## 通过指定单个变量绘制直方图
```



fill 参数表示依据不同的驱动填充不同的颜色

密度图

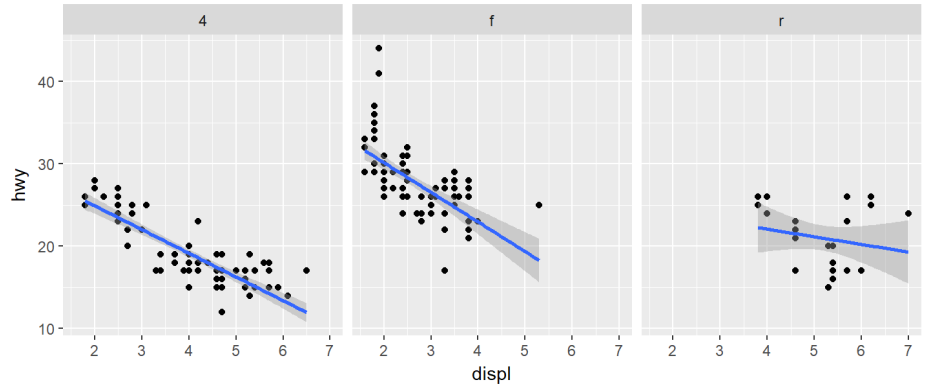
```{r, fig.height = 3.2}
qplot(hwy, data = mpg, geom = "density", color = drv)
```



面 Facets

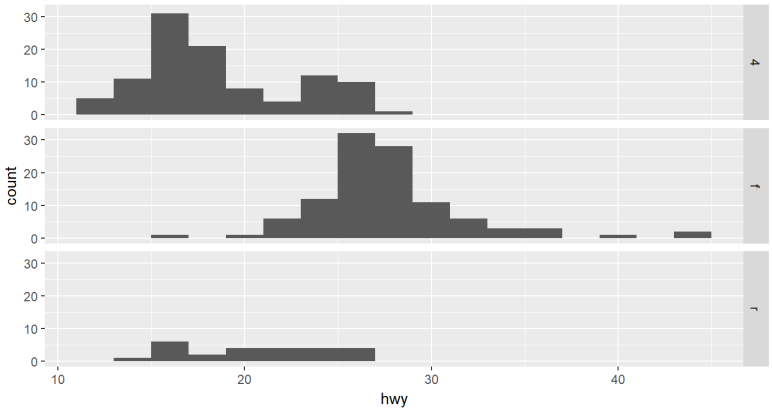
- 面 facets 等同于 lattice 中的面板 panels，可以根据因子创造多个数据子集的图形
- facets 后面是由~连接的式子，左边决定面板的行，右边决定面板的列，如果没有用.表示

```{r, fig.height = 3.2, warning = F}
qplot(displ, hwy, data = mpg, geom = c("point", "smooth"), method = "lm", facets = .~drv)
```



面 Facets

```{r, fig.height = 4}
qplot(hwy, data = mpg, facets = drv ~ ., binwidth = 2)
```



总结

- qplot()函数和 plot()函数类似，但是多了很多的特征
- 语法介于基础绘图系统和 lattice 系统之间
- 得到的图形很美观而且适用于出版等(如果设计风格和你口味)
- 订制化程度低，自定义个人风格比较困难

ggplot2 绘图的基本组成部分

- 数据框
- 美学映射(aesthetic mappings):将数据与点的颜色和大小对应
- 几何形状 geoms:放在页面上的特定对象，点，线，形状等
- 面 facets:用于条件绘图
- 统计转换 stats:比如柱形分析(bining)，分位数(quantiles)，平滑(smoothing)，回归(regression)等等
- 刻画 scale:绘图的时候不同的变量是如何刻画的(男 = 红，女 = 蓝)
- 坐标系 coordinate system:具体的数字如何转化成一副图的

- 用 ggplot2 制图的时候可以想成是艺术家的调色板模型(artist's palette)，与基础绘图系统的模型相似
- 图是一层一层往上创建的:1.绘制数据以及基本的美学因子;2.加上一层概括(统计 stats);3.元数据或额外的标注

例子

- 老鼠过敏原和哮喘的群体研究
- 关于巴尔第摩市 5-17 岁的孩子的研究

- 都患有持续性哮喘，过去一年内病情加重
- 关注的是诱发哮喘发病的一些环境因子:室内环境污染，细微颗粒物 PM2.5;二氧化氮 NO2
- 问题:体重指数 BMI(Body Mass Index)是否会影响 PM2.5 和哮喘之间的关系?

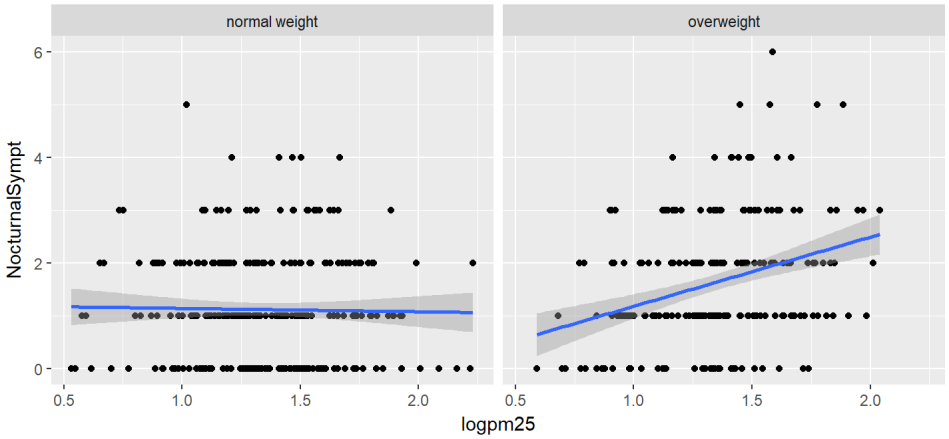
读取数据

```{r}
maacs <- read.csv("data/bmi_pm25_no2_sim.csv")
head(maacs, 2)
```

```
##       logpm25 logno2_new              bmicat NocturnalSympt
## 1 1.247700      1.183799 normal weight                    1
## 2 1.121648      1.551536       overweight                 0
```

基础图形

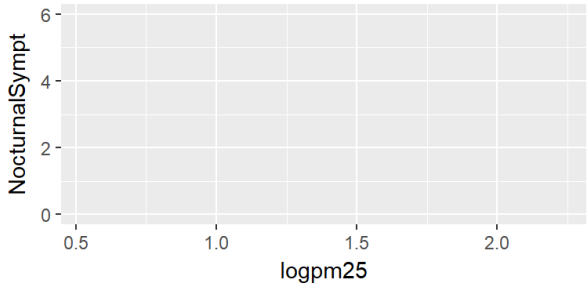
```{r, fig.height = 3.5, warning = F}
qplot(logpm25, NocturnalSympt, data = maacs, facets = .~bmicat,
      geom = c("point", "smooth"), method = "lm")
```



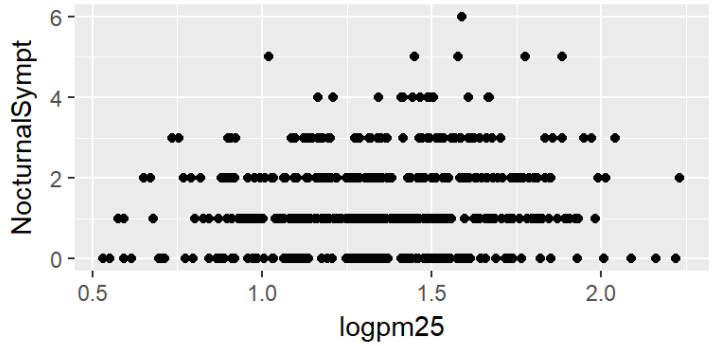
其中纵坐标 NocturnalSympt 表示过去两星期中夜间出现症状的天数

创建图层

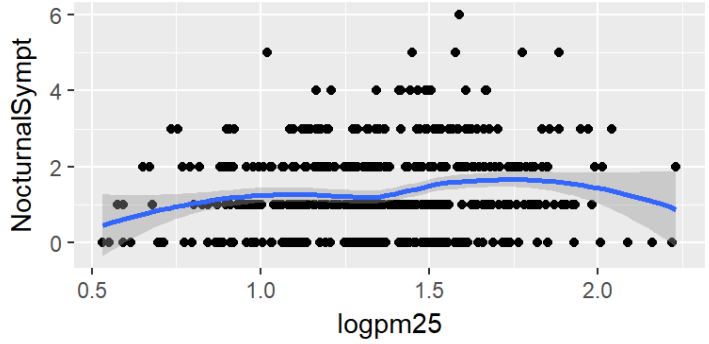
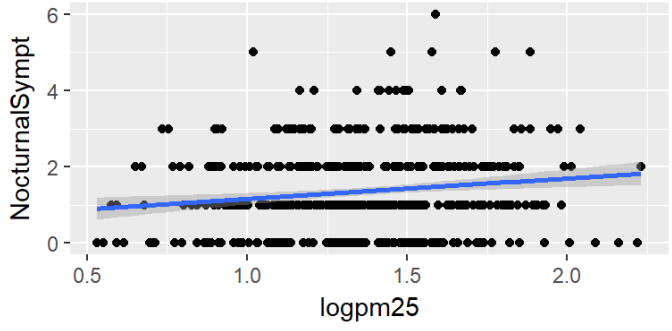
```{r, error = F, fig.height = 2, fig.width = 4}
g <- ggplot(maacs, aes(logpm25, NocturnalSympt));print(g)
```



p <- g+geom_point();print(p)
```

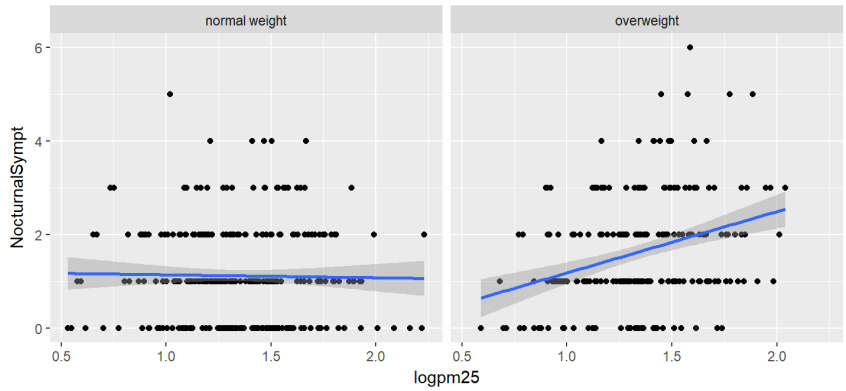



## 增加图层:平滑器

```{r, fig.height = 2, fig.width = 4, warning = F}
g + geom_point() + geom_smooth()
## `geom_smooth()` using method = 'loess'




g + geom_point() + geom_smooth(method = "lm")



```

增加图层:facets

```{r, fig.height = 3.5}
g+geom_point()+facet_grid(.~bmicat)+geom_smooth(method    = "lm")
```



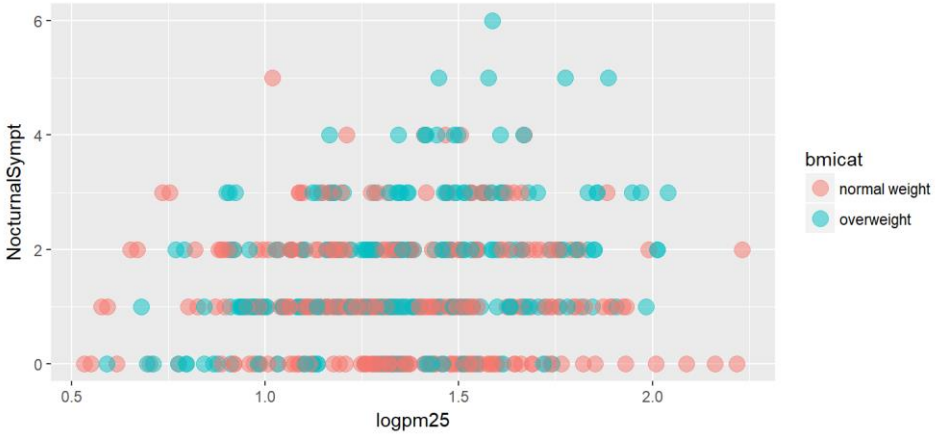
- 因子的标签会自动标注成图形的标签，故提早设定好因子标签
- 当有多行多列是可使用 facet_wrap()

注释

- 标签:x 轴标签 xlab();y 轴标签 ylab();标题 ggtitle();综合函数 labs()用来指定 x, y 轴坐标或者标题
- 每个 geom 函数例如 geom_point()和 geom_smooth()都有用于改变图形的参数
- 有些函数只有在全局状态下使用才有意义，例如 theme(legend.position = "none")，也可以是背景颜色等
- 内置的综合主题，控制图中的不同元素
    - theme_gray():是默认的主题，灰色背景白色格线
    - theme_bw():黑白色，对比度更强

修改美学特性

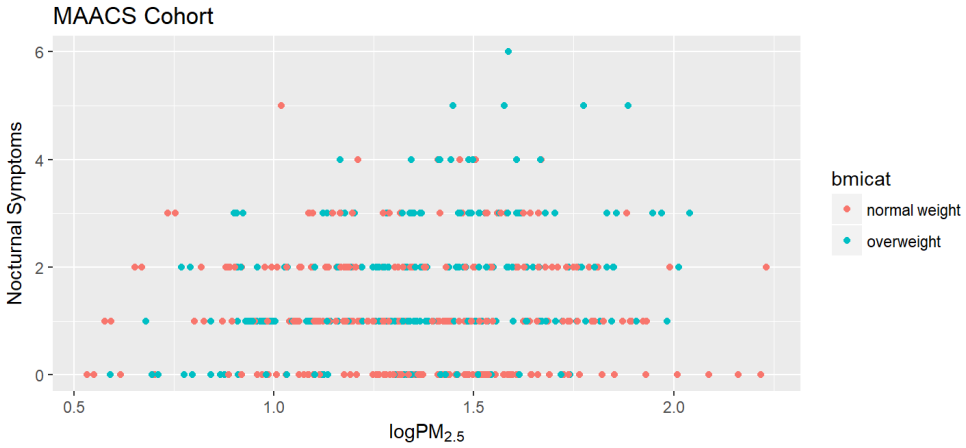
```{r, fig.height = 3.5}
g + geom_point(aes(color = bmicat), size = 4, alpha = 1/2)
```
```



- 可以设定 color 为固定颜色如 color = "steelblue"", alpha 表透明度
- 注意:当给 color 赋值变量时，要将整个表达式用 aes()括起来

修改标签

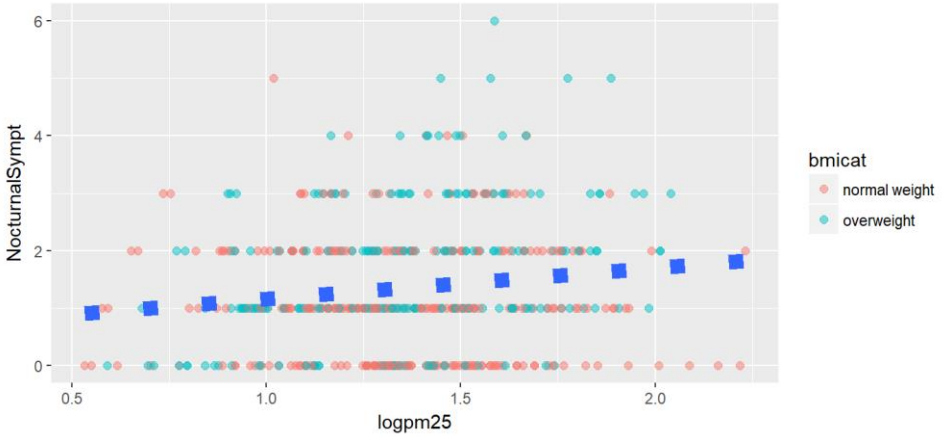
```{r, fig.height = 3.5}
g+geom_point(aes(color = bmicat))+labs(title = "MAACS Cohort")+
      labs(x = expression("log"*PM[2.5]), y = "Nocturnal Symptoms")
```



注表达式 expression("log"*PM[2.5])中中括号会显示为下标

自定义平滑

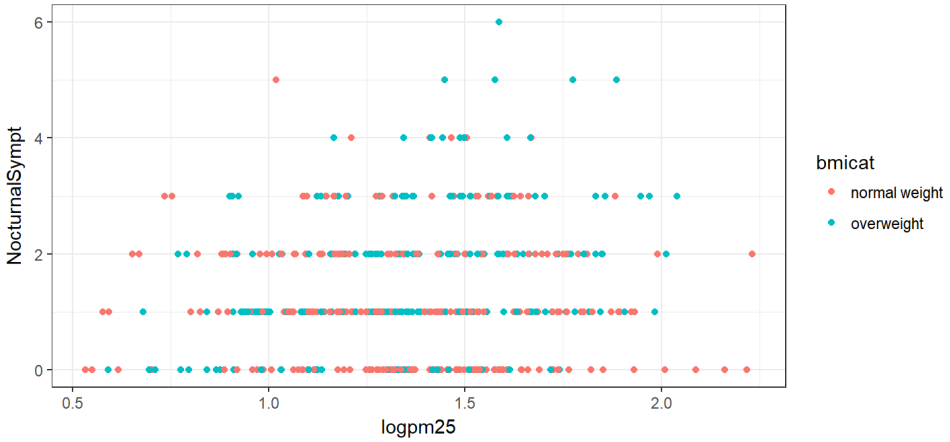
```{r, fig.height = 3.5}
g+geom_point(aes(color = bmicat), size = 2, alpha = 1/2)+
      geom_smooth(size = 4, linetype = 3, method = "lm", se = FALSE)
```



size 表示线的粗细, linetype 表示线型, se = F 不显示置信区间

改变主题

```{r, fig.height = 3.5, warning = F}
g+geom_point(aes(color = bmicat))+theme_bw(base_family = "Times")
```



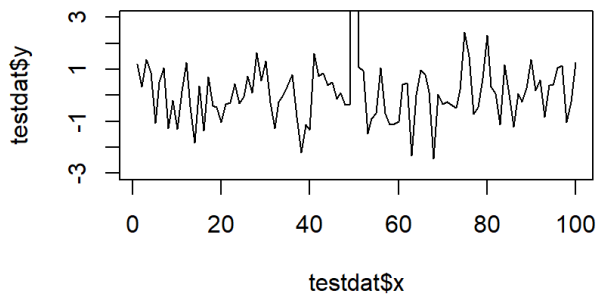
主题改变为基本的黑白主题，字体改成了 Times 字体()

坐标轴范围

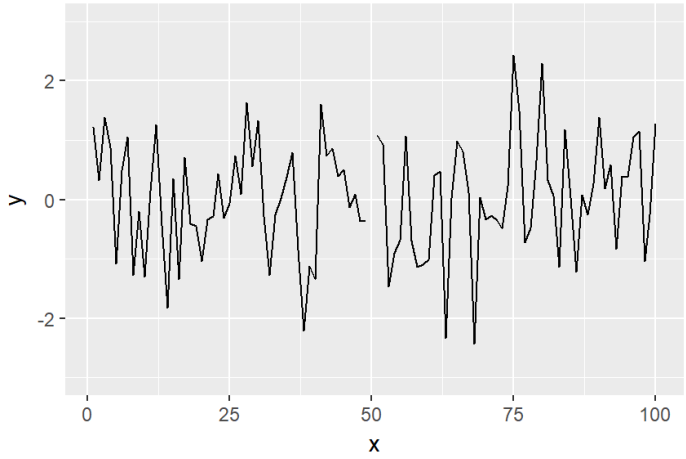
ggplot 和基础绘图不一样的地方是，如果一副图中的某个数据超出了图的范围，基础绘图会看到有一个异常值，但是 ggplot 可能不会

```{r, fig.width = 4.5, fig.height = 3}
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50, 2] <- 100  ## Outlier!
plot(testdat$x, testdat$y, type = "l", ylim = c(-3, 3))
```

```

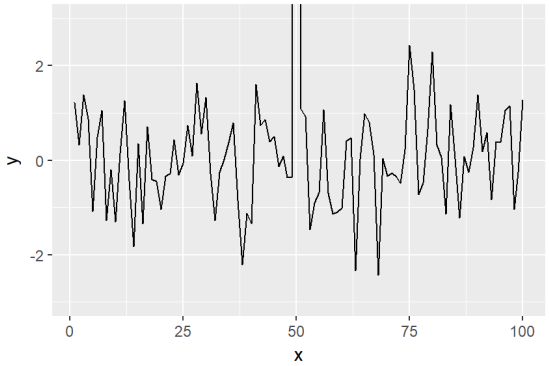



```{r, fig.width = 4.5, fig.height = 3}
g <- ggplot(testdat, aes(x = x, y = y))
g + geom_line()+ ylim(-3, 3)
```



ggplot 会去数据的子集，只包含-3 到 3 之间的值，故异常值丢失

```{r, fig.width = 4.5, fig.height = 3}
g + geom_line()+ coord_cartesian(ylim = c(-3, 3))
```



coord_cartesian()函数不会丢失点，这里不会取子集
```

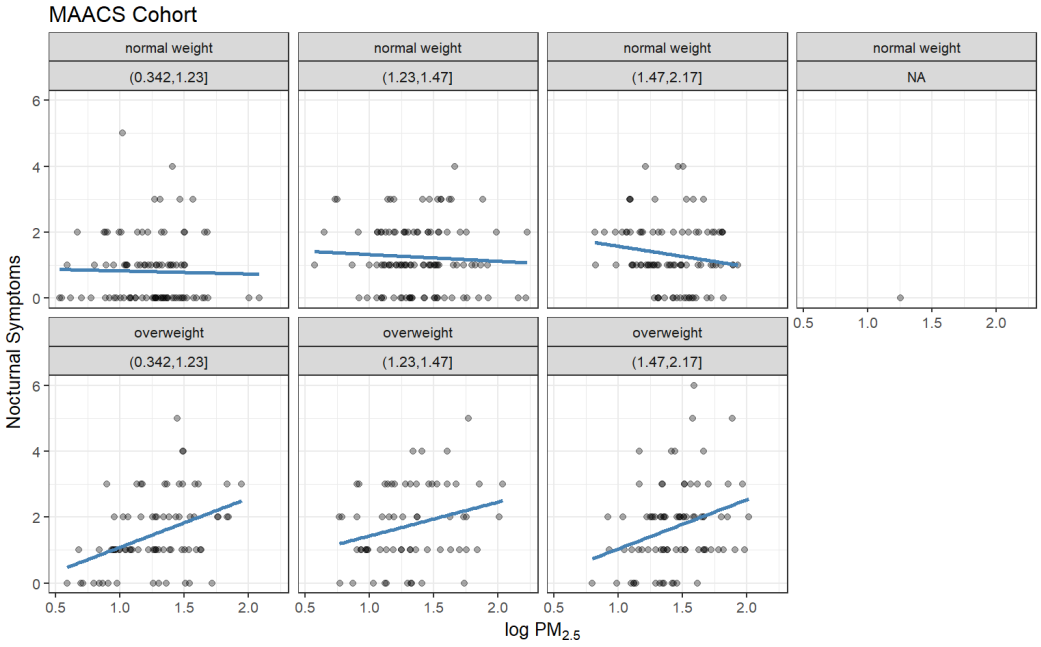
更复杂的例子

- 想知道 PM2.5 和夜间症状之间的关系是如何随着 BMI 和 NO2 变化的
- 麻烦在于 NO2 是连续的，我们没有办法将条件作用于一个连续变量
- 需要把这个变量归类到一系列合理的分布范围使之成为分类变量
  -使用 cut()函数

将 NO2 分成三份

```{r}
cutpoints <- quantile(maacs$logno2_new, seq(0, 1, length = 4), na.rm = TRUE) ## 找出分割点
maacs$no2tert <- cut(maacs$logno2_new, cutpoints) ## 分成三份
levels(maacs$no2tert) ## 查看生成的三个等级
## [1] "(0.342,1.23]" "(1.23,1.47]"  "(1.47,2.17]"
```

最终图形




```{r, echo = FALSE, fig.width = 8, fig.height = 5, warning = F}
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
g + geom_point(alpha = 1/3) +
    facet_wrap(bmicat~no2tert, nrow = 2, ncol = 4)+
    geom_smooth(method = "lm", se = FALSE, col = "steelblue") +
    theme_bw(base_size = 10) +
```

```
    labs(x = expression("log " * PM[2.5])) +
    labs(y = "Nocturnal Symptoms") +
    labs(title = "MAACS Cohort")
```

对应代码

```{r, eval = FALSE}
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
g + geom_point(alpha = 1/3) +
    facet_wrap(bmicat~no2tert, nrow = 2, ncol = 4)+
    geom_smooth(method = "lm", se = FALSE, col = "steelblue") +
    theme_bw(base_size = 10) +
    labs(x = expression("log " * PM[2.5])) +
    labs(y = "Nocturnal Symptoms") +
    labs(title = "MAACS Cohort")
```

#总结

- ggplot2 非常强大和灵活，如果掌握了其中的"语法"和各种设置
- 可以绘制各种类型的图形，详见相关书籍