# ggplot2画图

温灿红

中国科学技术大学管理学院

# ggplot2

- ggplot2是一个用来绘制统计图形的**R**软件包，图形是可迭代式地添加上去的；
- ggplot2 有着一套图形语法（该语法基于Grammar of Graphics一书），有一系列独立的图形部件组成，并能以各种不同的方式组合起来；
- 参考资料：
  - Hadley Wickham著的《ggplot2:数据分析与图形艺术》
  - http://ggplot2.org
  - https://www.r-graph-gallery.com/ggplot2-package.html

# 安装 ggplot2

1. 直接安装

```
install.packages("ggplot2")
```

2. 从 tidyverse 安装

```
install.packages("tidyverse")
```

- tidyverse 一个软件包的集合，包含了很多有用的软件包，如：
  - dplyr: 数据管理，如筛选、剔除、排序、分组处理等等
  - tidyr: 清洗并重构数据
  - haven: 从其他格式(如Stata, SAS, SPSS)中读取数据
  - stringr: 字符串数据处理

# 第一张ggplot2图：耗油量数据

- ggplot2内置的mpg数据集，记录了美国1999年到2008年间部分汽车的制造厂商、型号、类别、驱动系统和耗油量等信息。

```
library(ggplot2)
head(mpg, 6)
```

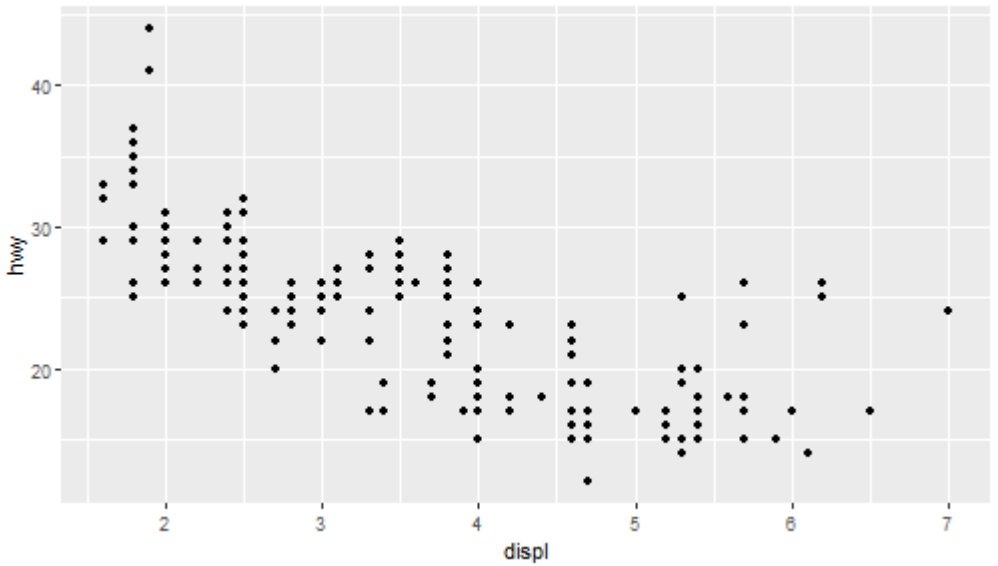
```
## # A tibble: 6 × 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa…
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
```

- cty和hwy分别记录城市和高速公路驾驶耗油量
- displ表示发动机排量
- drv表示驱动系统：前轮驱动(f)、后轮驱动(r)和四轮驱动(4)
- model表示车辆型号
- class表示车辆类型，如双座汽车、SUV，小型汽车等

# 第一张ggplot2图：耗油量数据

- 发动机排量和耗油量存在什么关系?

```
ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point()
```




- ggplot2图形的**三个基本构成**:
  - 数据： mpg
  - 图形属性映射：aes(), displ 映射到x轴, hwy 映射到y轴
  - 几何对象：geom_point()，散点图

# 图形的语法（Grammar of Graphics）

一张统计图形就是从数据到几何对象图形的一个映射，还包含了数据的统计变换、绘制的坐标系和是否分多个子图。

1. **数据(Data)**: 想要可视化的数据以及一系列将数据中的变量对应到图像属性的**映射(mapping)**
2. **图层(Layers)**: (1). **几何对象(Geoms)**: 在图中实际看到的图形元素 (2). **统计变换(Stats)**: 对数据的统计变换或汇总，如平均值、分组计数等
3. **标度(Scales)**: 将数据的取值映射到图形空间，如颜色、大小、形状。展示标度的常见做法是绘制图例和坐标轴
4. **坐标系(Coordiante systems)**: 描述了数据是如何映射到图形所在的平面的，也提供了坐标轴和网格线
5. **分面(Faceting)**: 描述了如何将数据分解为各个子集，并对子集作图后联合展示
6. **主题(Theme)**: 控制精细显示，如字体大小和背景颜色

# 数据

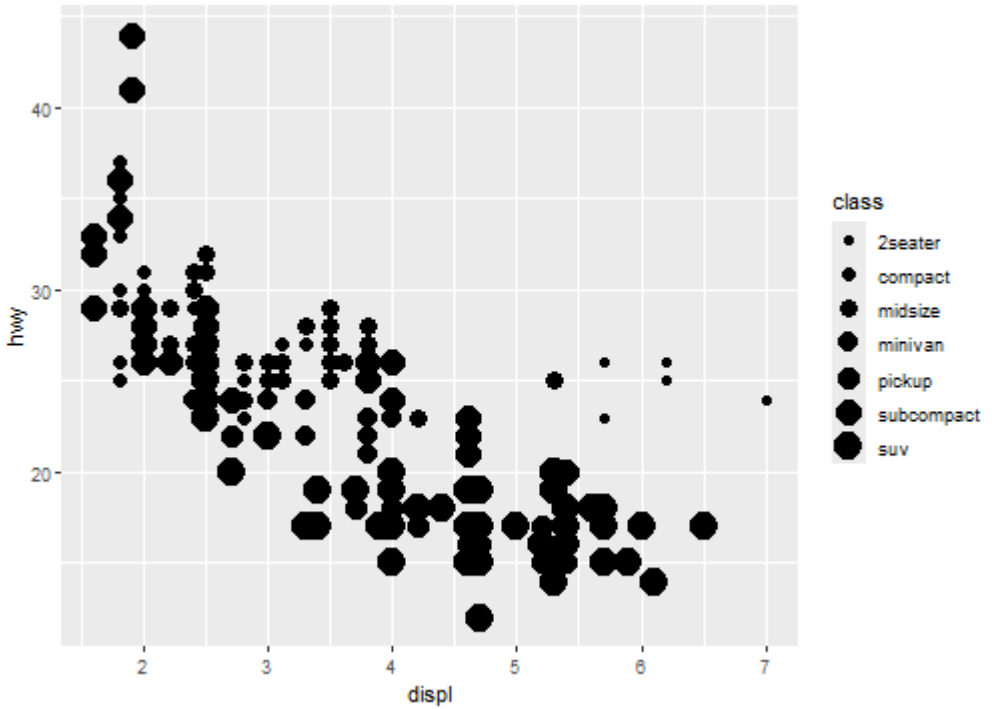
# 数据

- ggplot2中的数据必须存储在 data.frame格式中
- 通过 aes() 函数将数据映射到图形属性。如:
  - x: 映射到x轴的变量
  - y: 映射到y轴的变量
  - color: 图形属性的轮廓颜色
  - fill: 图形属性的内部填充颜色
  - alpha: 图形的透明程度（0-1之间，其中0代表全透明）
  - linetype: 线的类型，如实线solid, 虚线dashed, 点线dotted
  - shape: 点的形状
  - size: 图形属性的大小

# 例子（一）

```
ggplot(data = mpg, aes(x = displ, y = hwy, size = class)) + geom_point()
```

```
## Warning: Using size for a discrete variable is not advised.
```

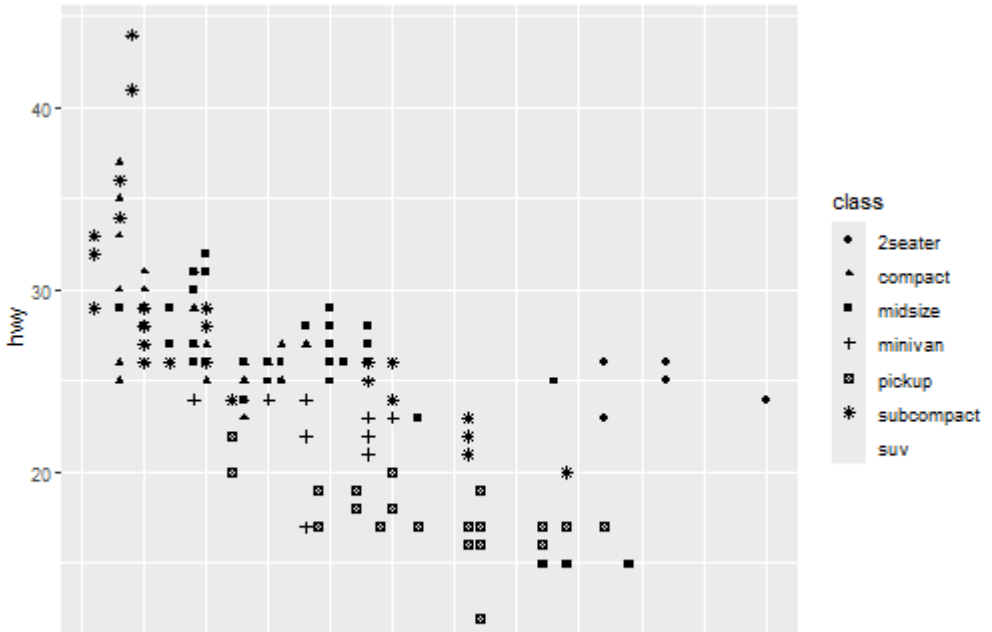


# 例子（二）

```
ggplot(data = mpg, aes(x = displ, y = hwy, shape = class)) + geom_point()
```

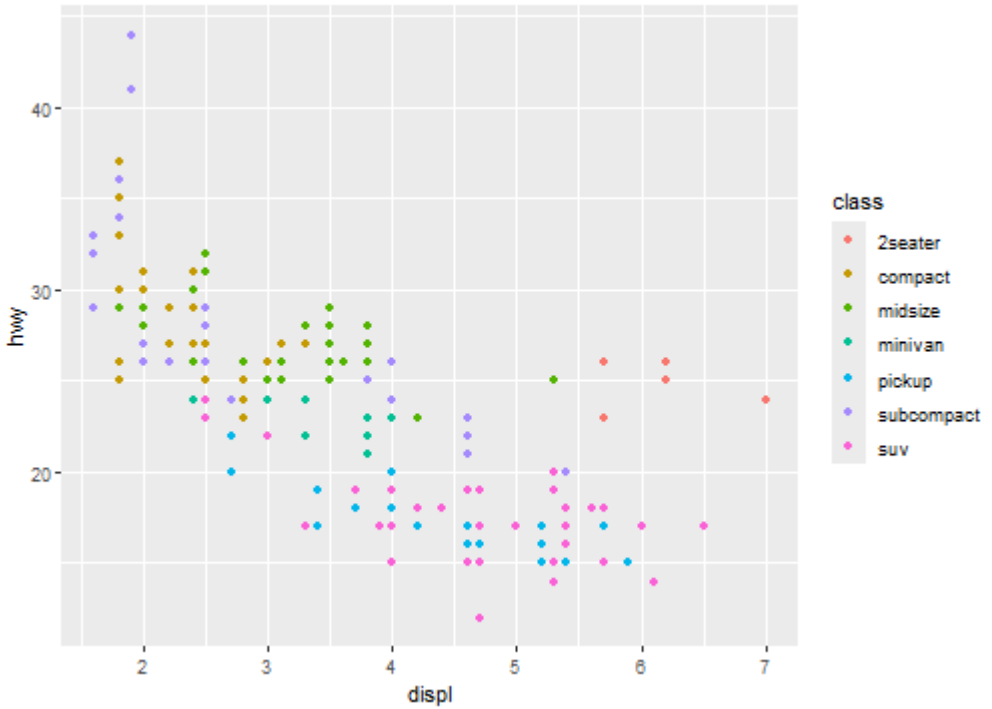
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more
## than 6 becomes difficult to discriminate
## ℹ you have requested 7 values. Consider specifying shapes manually if you need
##   that many have them.
```

```
## Warning: Removed 62 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



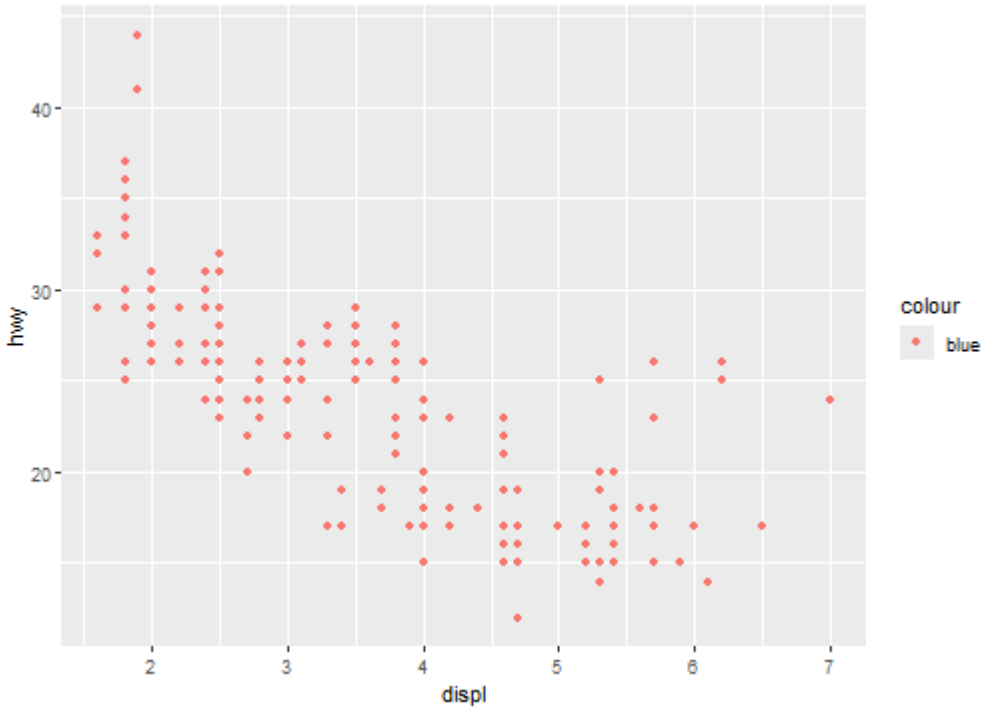
# 例子（三）

```
ggplot(data = mpg, aes(x = displ, y = hwy, color = class)) + geom_point()
```



# 例子（四）

```
ggplot(data = mpg, aes(x = displ, y = hwy, color = "blue")) + geom_point()
```

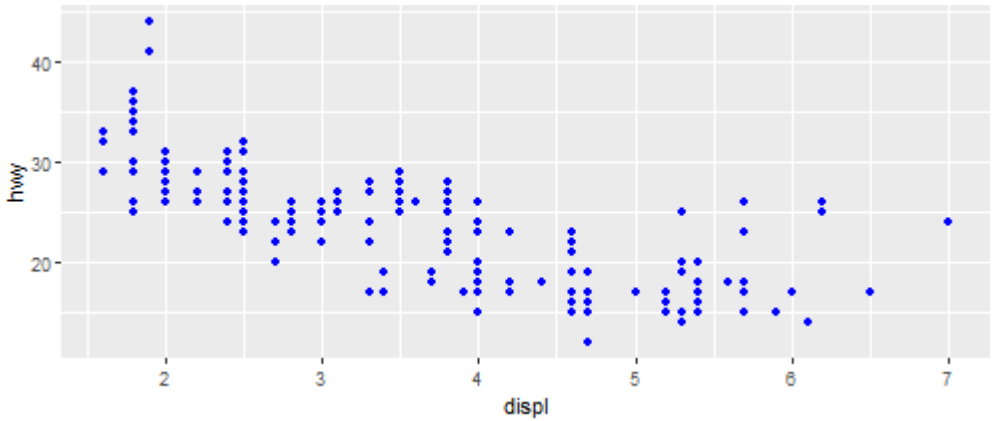



- 哪里出错了?

# 设定与映射

- **映射** 是指通过 aes() 函数将变量映射到图形属性上，通过变量控制图形
- **设定** 是指在函数 aes() 外对图形属性进行设置，一般设定为一个常数，不涉及到数据本身的任何变量

```
ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point(color = "blue")
```

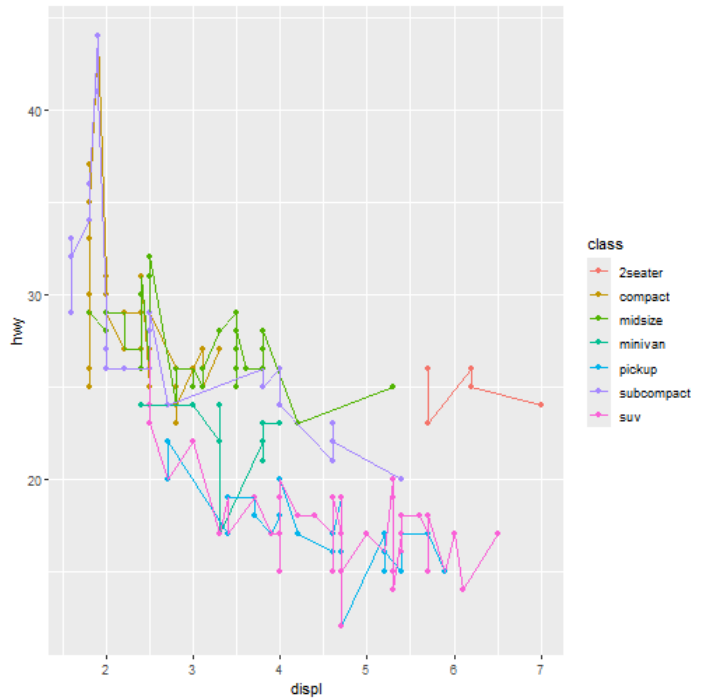



- 试试看如下命令是否正确?

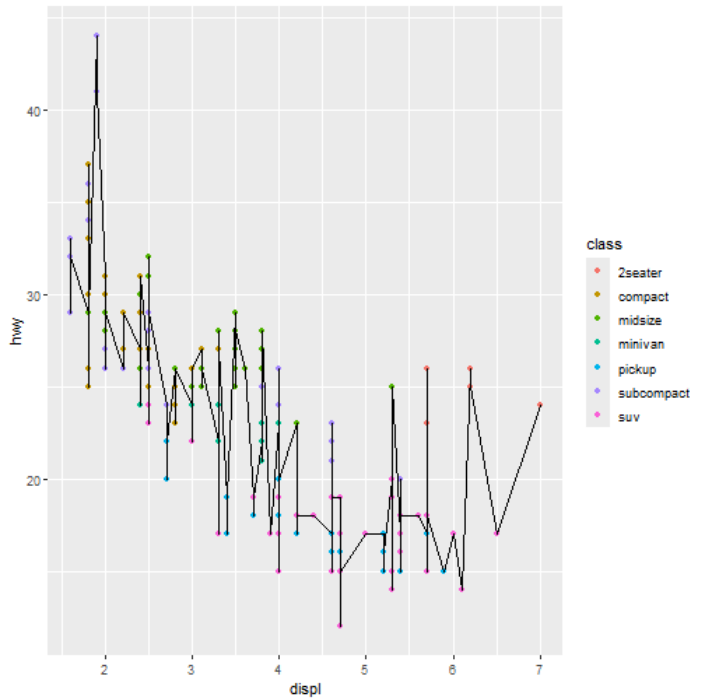
```
ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point(aes(color = "blue"))
```

# aes()的位置

```
ggplot(data = mpg, aes(x = displ, y =
  geom_point() + geom_line()
```

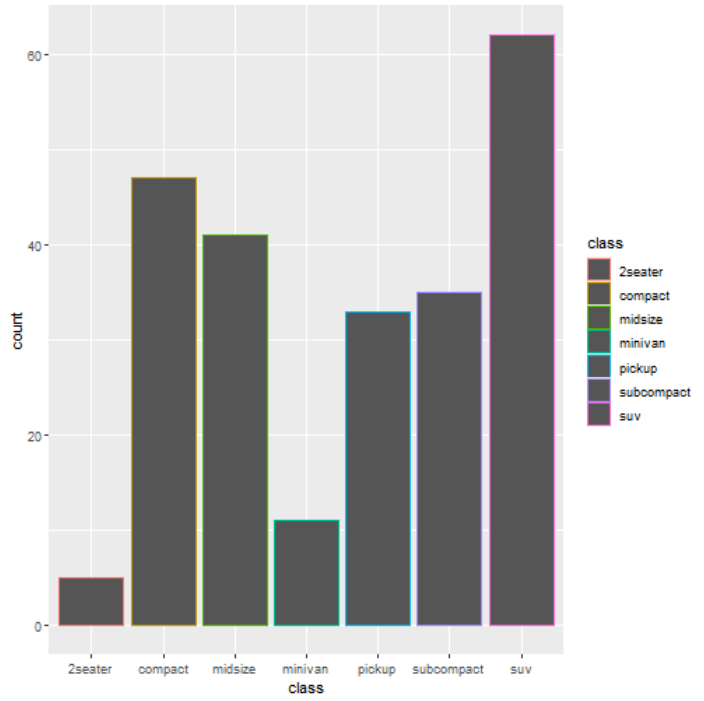



```
ggplot(data = mpg, aes(x = displ, y =
  geom_point(aes(color = class)) + geo
```

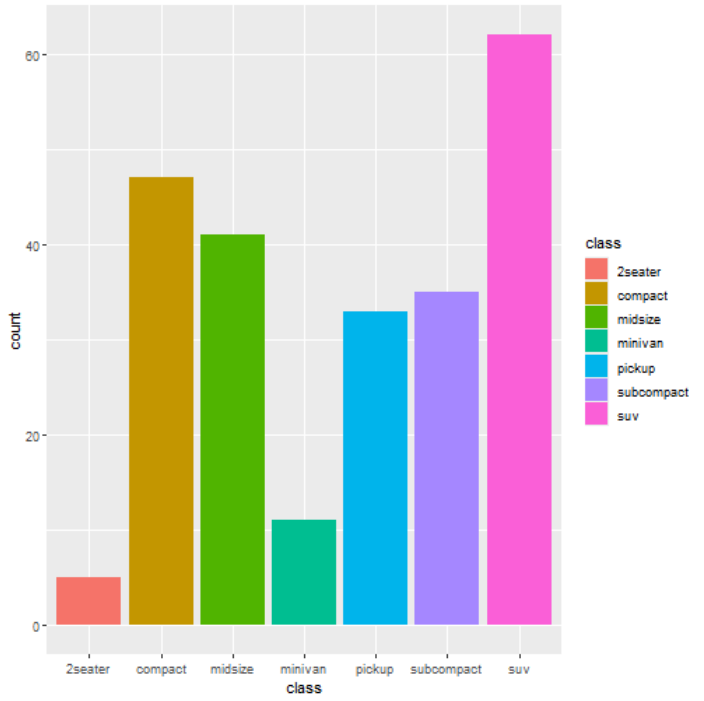


# color和fill的区别

```
ggplot(data = mpg, aes(x = class)) + g
```




```
ggplot(data = mpg, aes(x = class)) + g
```

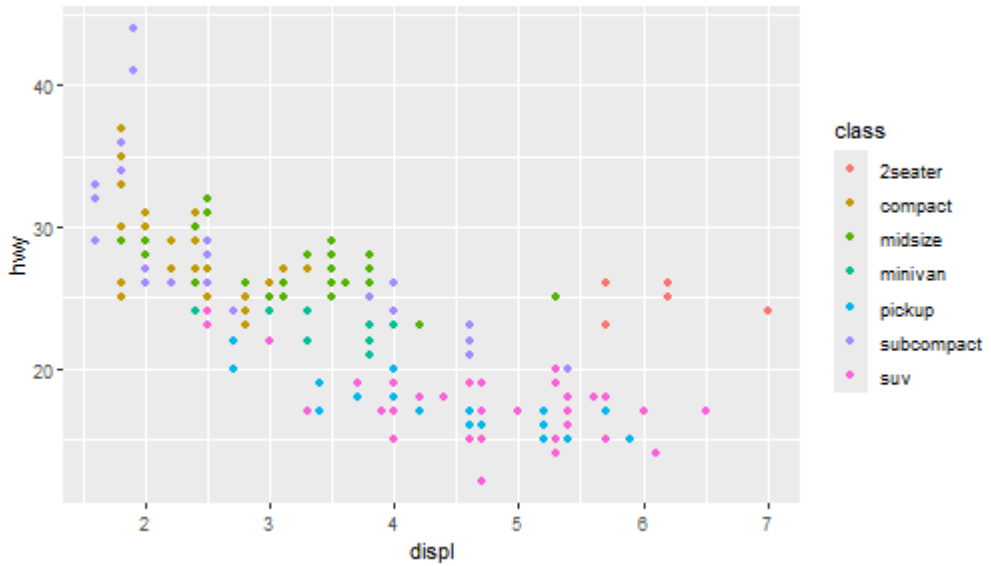


# 图层

# 图层的叠加： +

- ggplot2 中的图形是层层叠加上去的，每层之间通过+连接，而每一层的图形通过 geom 或 stat 产生;
- 每一层的图形的映射可从ggplot() 函数继承下来，也可以重新定义。
- 可把基础的图层保存下来，后面在此基础上添加不同的图层。

```
p <- ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point(aes(color = class))
p
```

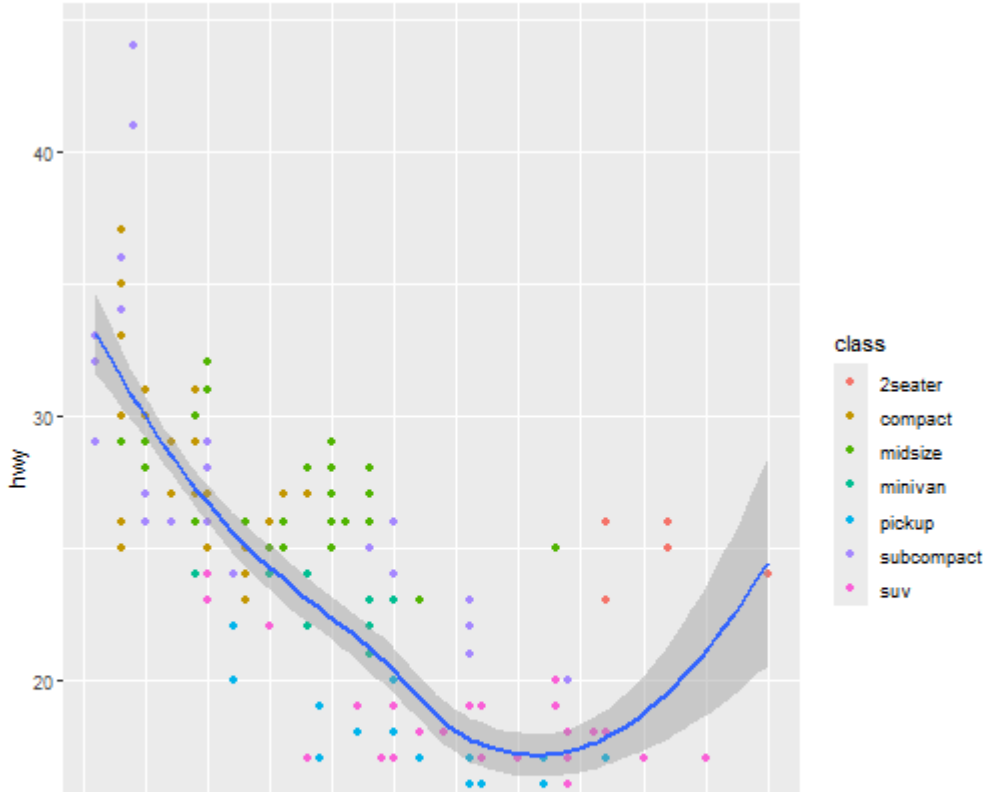


# 例子

- 增加一个光滑曲线拟合

```
p + geom_smooth()
```

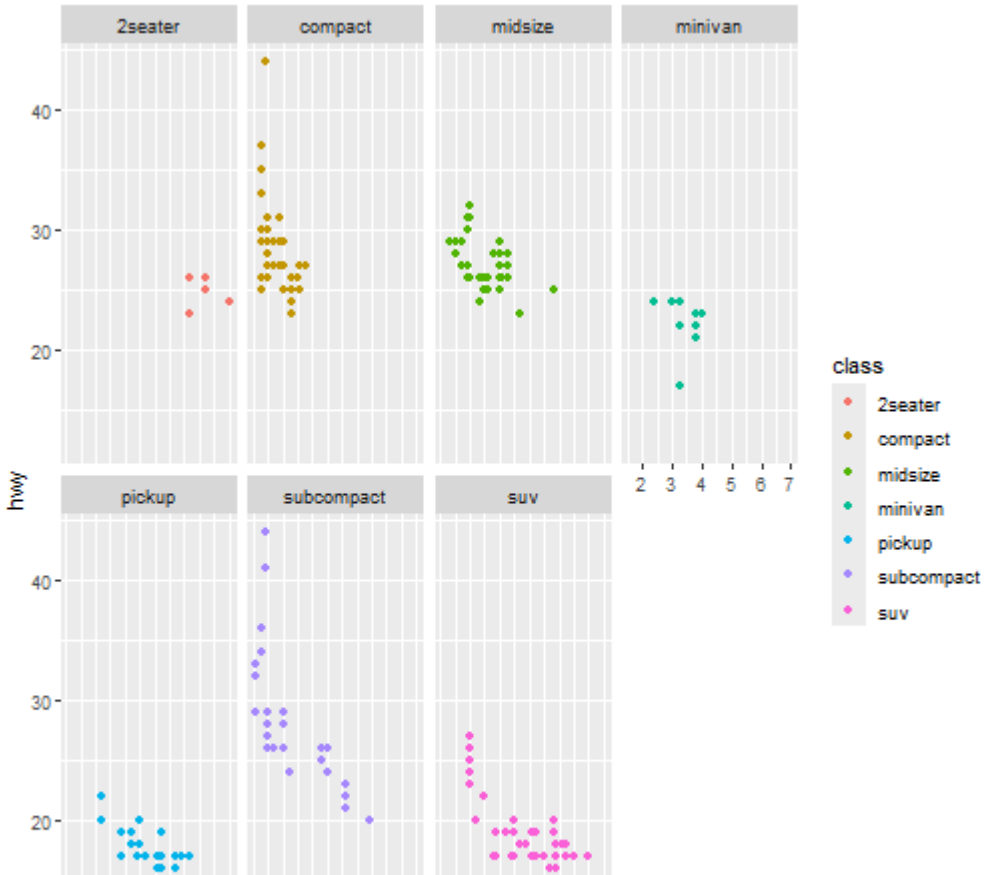
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



# 例子

- 将图形分面

```
p + facet_wrap(~ class, nrow = 2)
```



# 几何对象

# 几何对象：Geoms

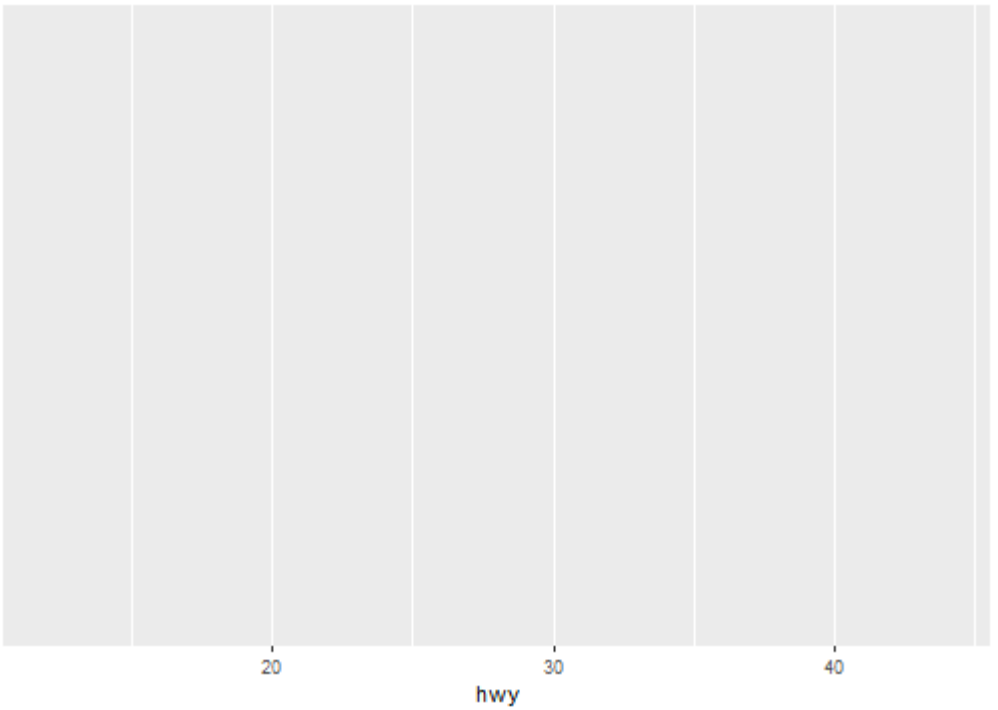
几何对象，简称为Geoms，实现图层的实际渲染，控制所创建图形的类型。 函数为geom_***形式，如：

- geom_bar()：展示离散变量的分布
- geom_boxplot()：箱线图
- geom_histogram()：直方图
- geom_line()：折线图
- geom_point()：散点图
- geom_ribbon()：条带，带有垂直厚度的路径
- geom_smooth()：最佳拟合平滑曲线
- 更多可见《ggplot2：数据分析与图形艺术（第2版）》第106-107页

# 例子：单变量

- 耗油量的分布?
- 默认是什么都不展示，只是展示背景

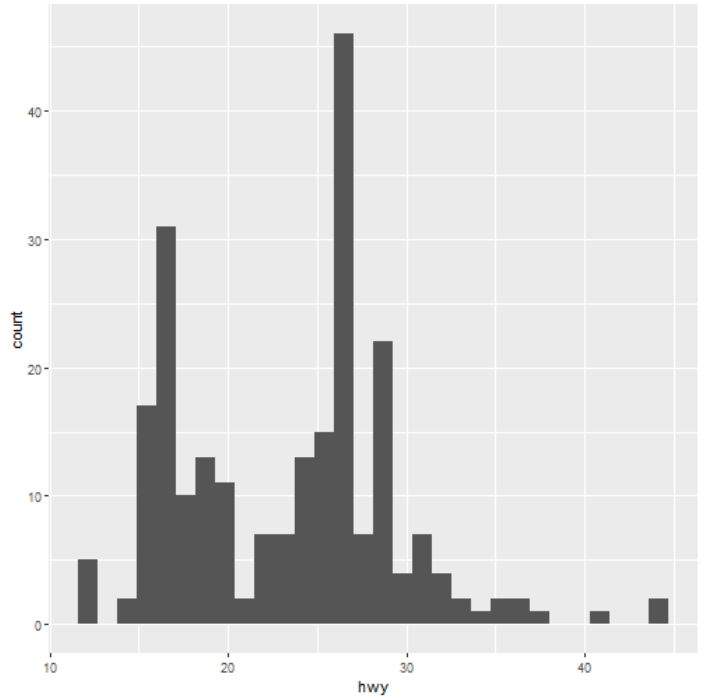
```
pro <- ggplot(data = mpg, aes(x = hwy))
pro
```



# 例子：单变量的直方图

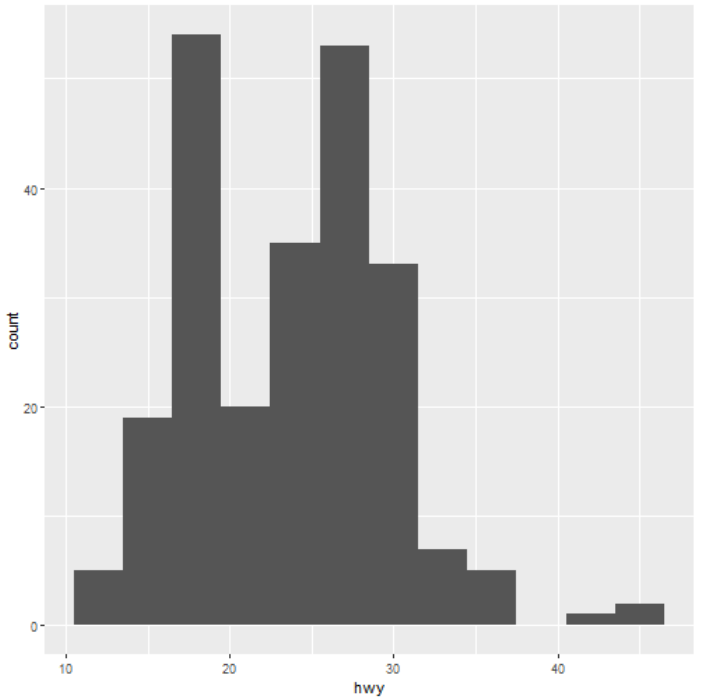
```
pro + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick bett
```



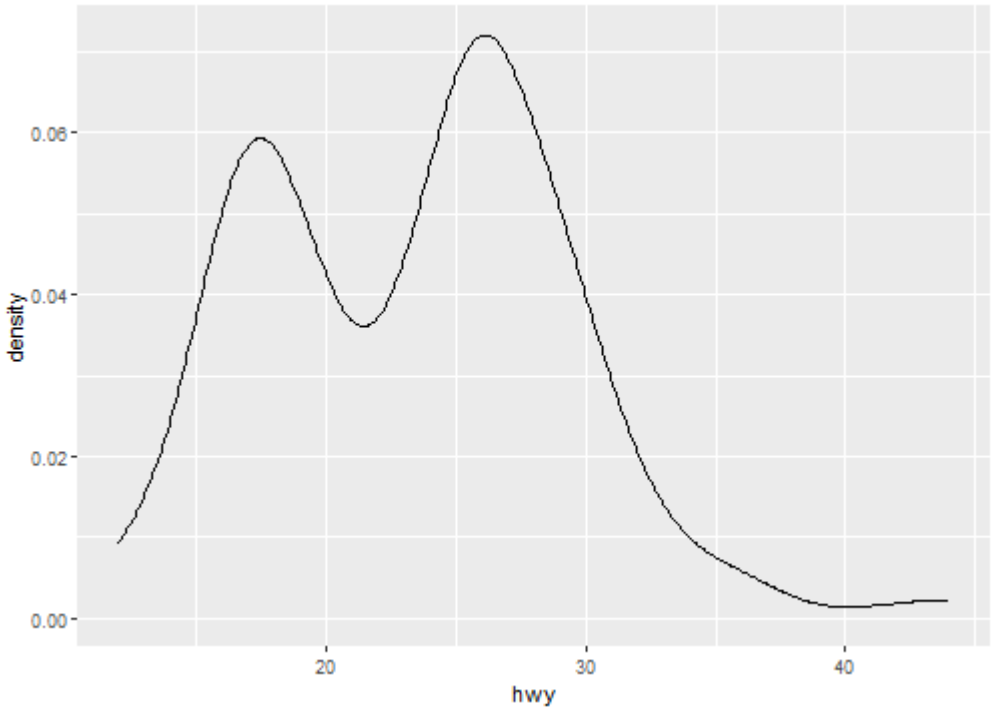

- 改变窗宽

```
pro + geom_histogram(binwidth = 3)
```



# 例子：单变量的密度估计图

```
pro + geom_density()
```

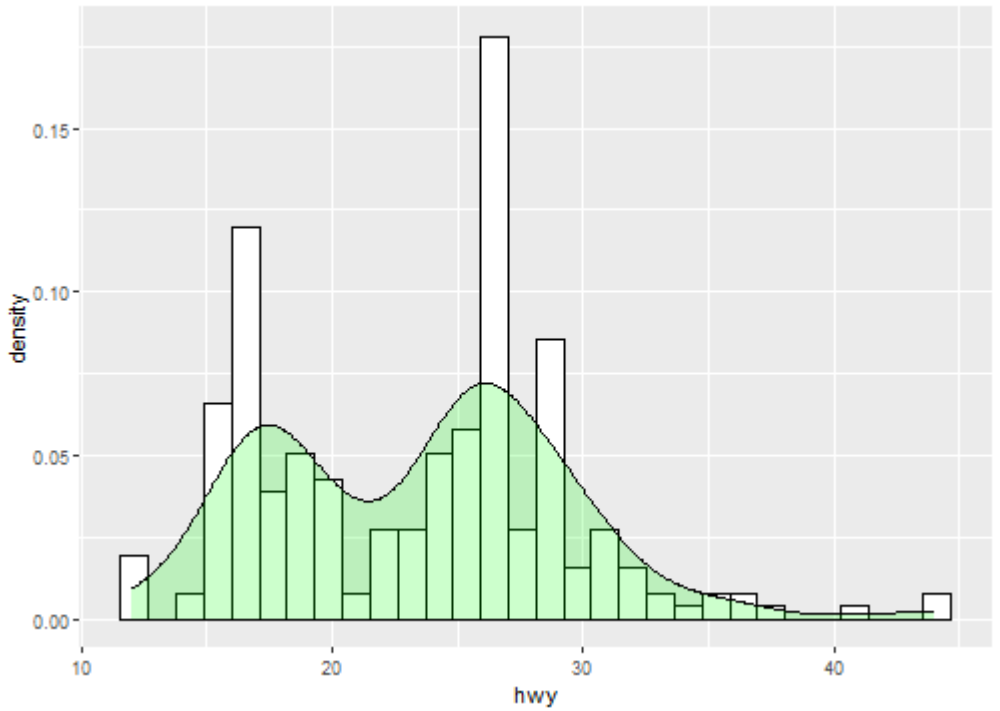


# 例子：单变量

- 直方图叠加密度估计图

```
pro + geom_histogram(aes(y=..density..), color = "black", fill="white") +
  geom_density(alpha=0.2, fill="green")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

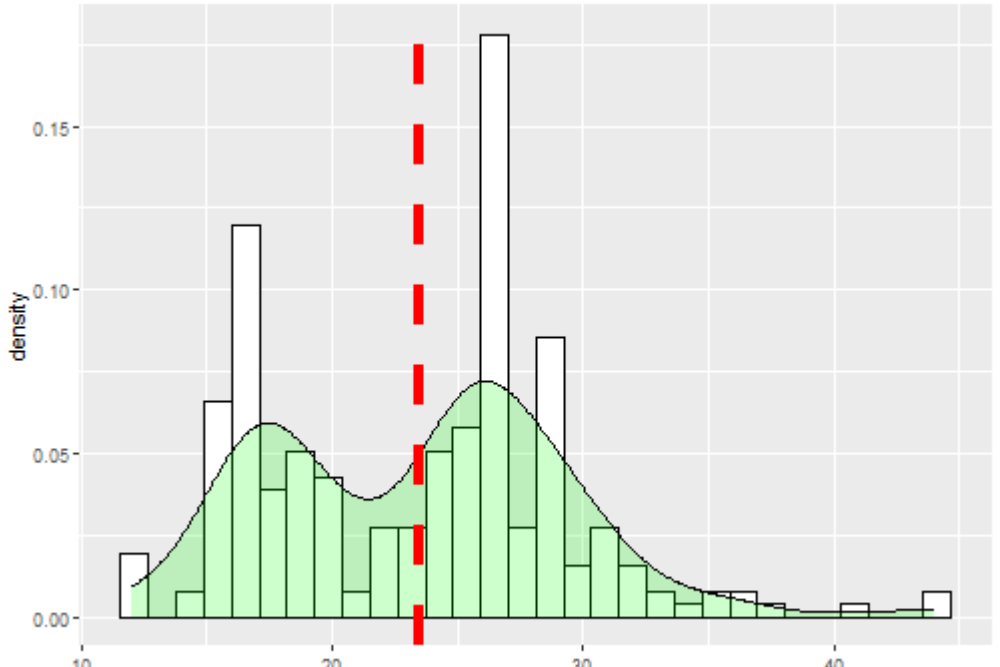


# 例子：单变量

- 直方图叠加密度估计图，额外加一个竖线显示平均耗油量

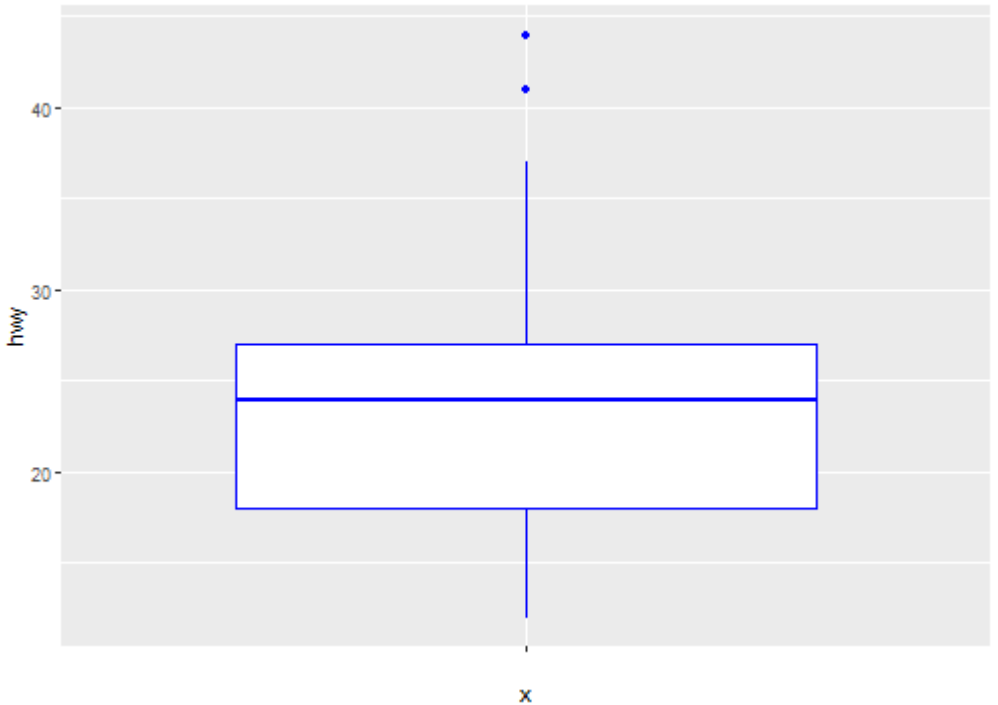
```
pro + geom_histogram(aes(y=..density..), color = "black", fill="white") +
  geom_density(alpha=0.2, fill="green")+
  geom_vline(aes(xintercept=mean(hwy)), color="red", linewidth=2, linetype="dashed
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



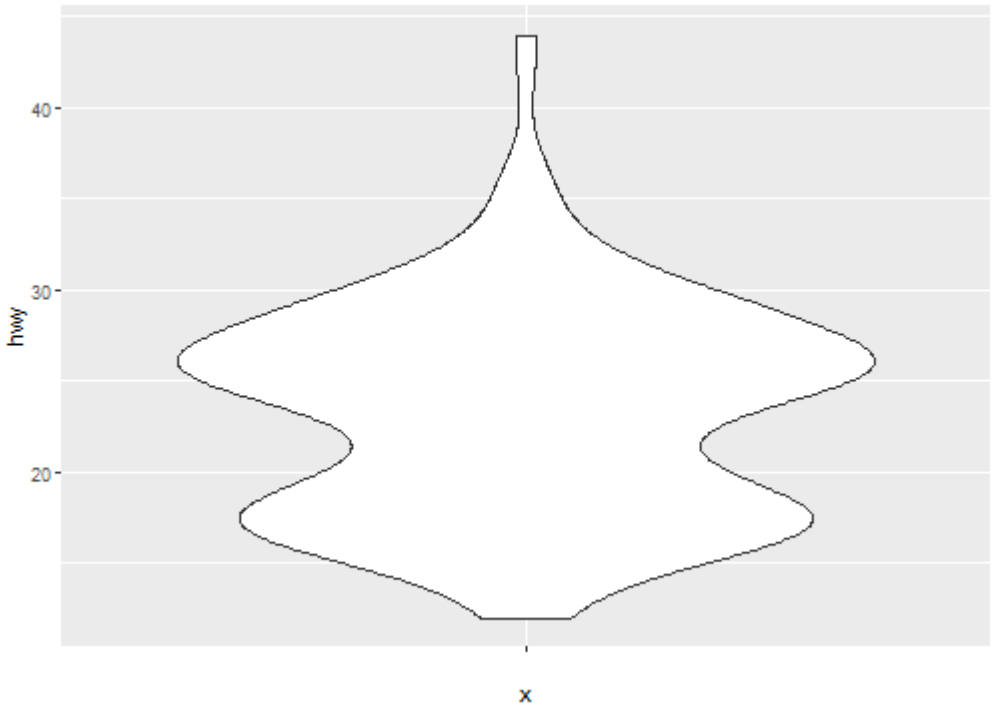
# 例子：单变量的箱线图

```
ggplot(mpg, aes(x="",y=hwy)) +  geom_boxplot(color="blue")
```



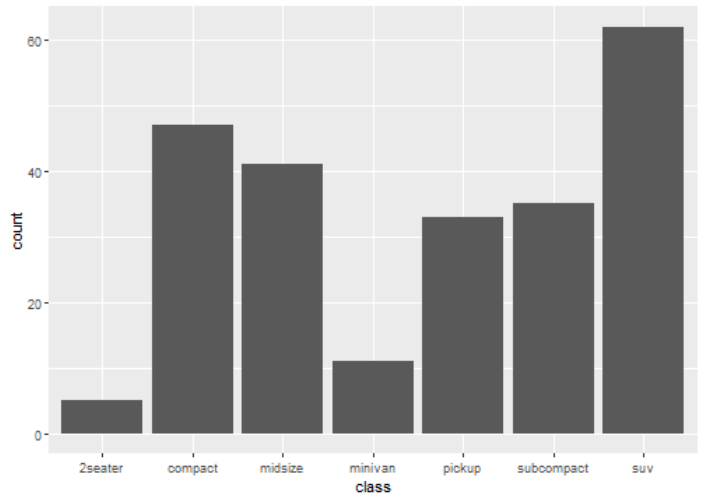
# 例子：单变量的小提琴图

```
ggplot(mpg, aes(x="",y=hwy)) +  geom_violin()
```

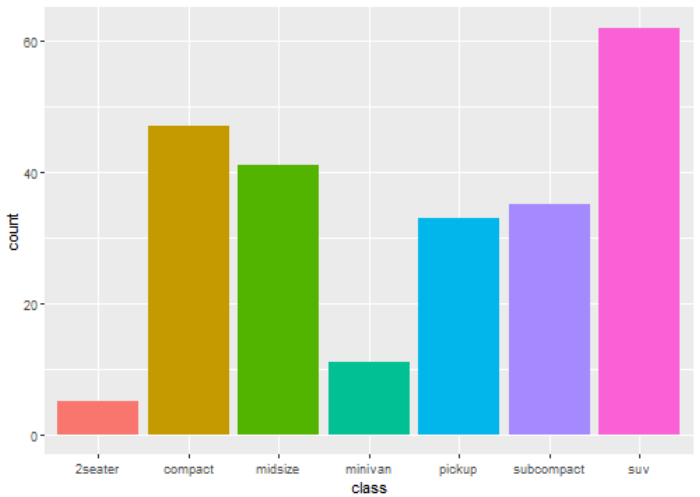


# 例子：单变量的条形图

```
p <- ggplot(data = mpg, aes(x = class))
p + geom_bar()
```




```
p + geom_bar(aes(fill = class)) +
  guides(fill=FALSE)
```



# 统计变换

# 统计变换：stat

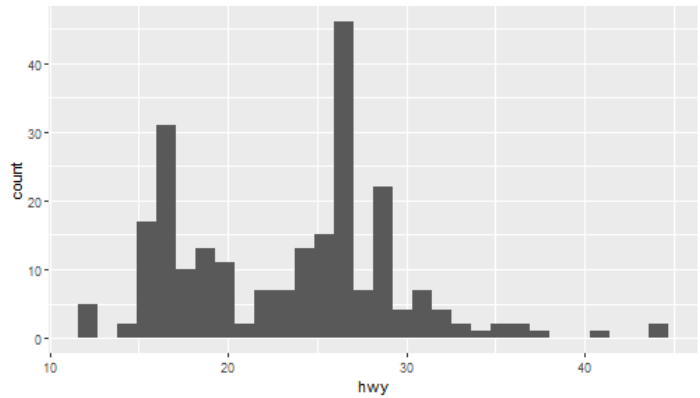
- 统计变换，简称stat，是通过某种形式的概括来转换数据的。函数为stat_*** 形式，一般是伴随着几何对象使用的，因此很少直接调用，如：
  - stat_bin(): geom_bar(), geom_histogram()
  - stat_boxplot(): geom_boxplot()
  - stat_contour(): geom_contour()
  - stat_sum(): geom_count()
- 一些不能用 geom_*** 函数创建的统计变换：
  - stat_ecdf()：计算经验累计分布函数图
  - stat_summary()：在不同的 x值上概述y
  - stat_qq()：Q-Q图的计算
  - stat_unique()：去掉重复的行

# 例子：直方图

- stat_bin() 等价于 geom_histogram()

```
ggplot(data = mpg, aes(x = hwy)) +
  stat_bin()
```

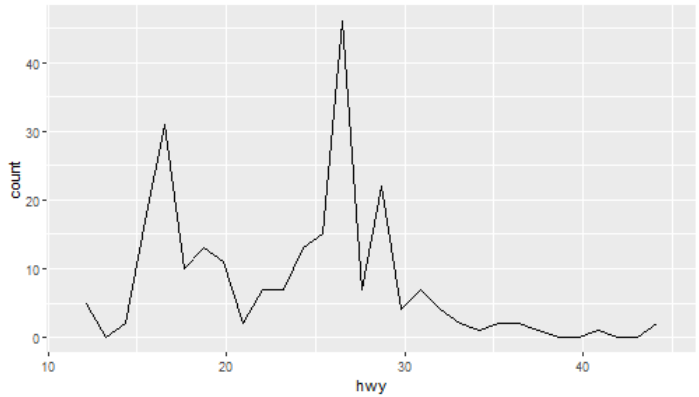
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




- 改变集合对象为折线：geom=line

```
ggplot(data = mpg, aes(x = hwy)) +
  stat_bin(geom = "line")
```

```
## `stat_bin()` using `bins = 30`. Pick better v
```

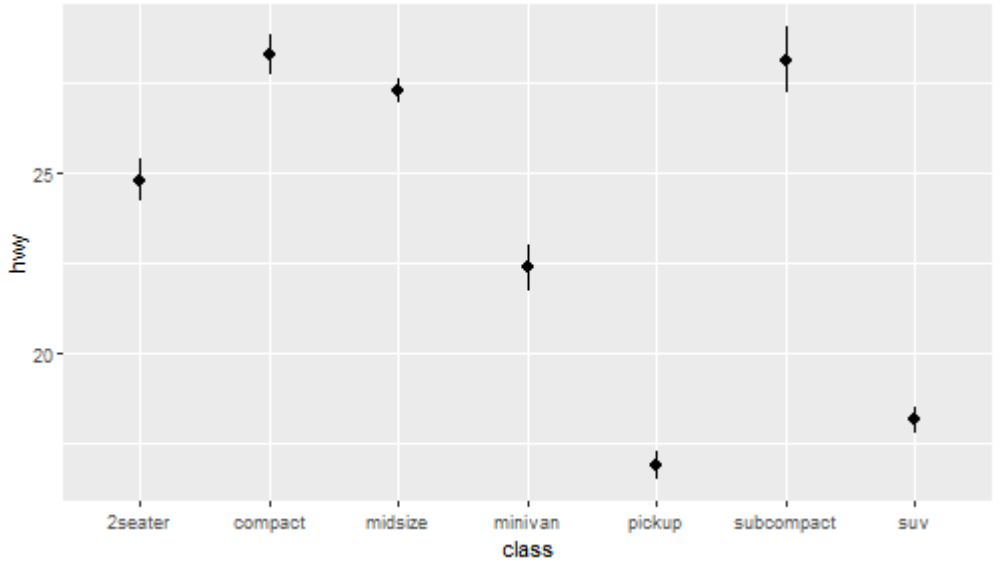


# 例子： stat_summary() 函数

- stat_summary() 函数在不同的 x值上概述y，默认的概括函数是mean_se，返回均值 ± 标准误。
- 对应的几何对象是“pointrange”，也就是在y的均值上画点，并延伸竖线至均值 ± 标准误位置。

```
ggplot(data = mpg, aes(x = class, y = hwy)) + stat_summary()
```

```
## No summary function supplied, defaulting to `mean_se()`
```

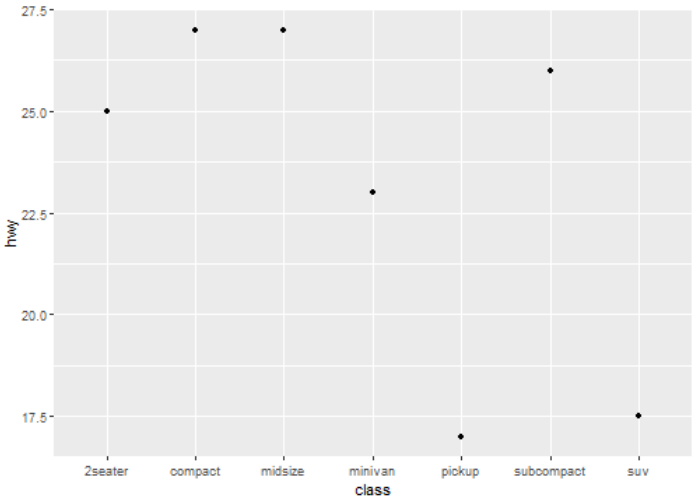


# 例子：`stat_summary()` 函数

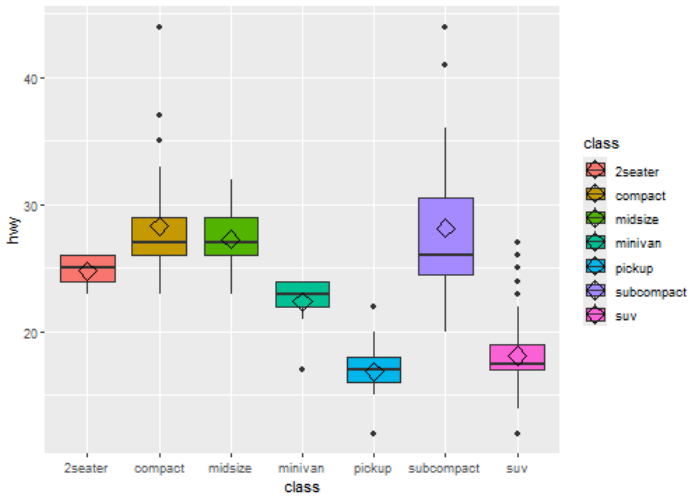
- 其他的概括函数有：
  - `median`: 中位数
  - `mean_cl_boot`: 均值和bootstrap得到的置信区间（默认为95%的置信区间)
  - `mean_cl_normal`: 均值和基于正态分布的置信区间（默认为95%的置信区间)
  - `mean_dsl`: 均值 ± 若干倍的标准差（默认为两倍)
  - `median_hilow`: 中位数和分位数（默认为0.025和 0.975分位数)

# 例子： stat_summary() 函数

```
ggplot(data=mpg, aes(x=class, y=hwy)) +
  stat_summary(fun.y="median", geom="p
```



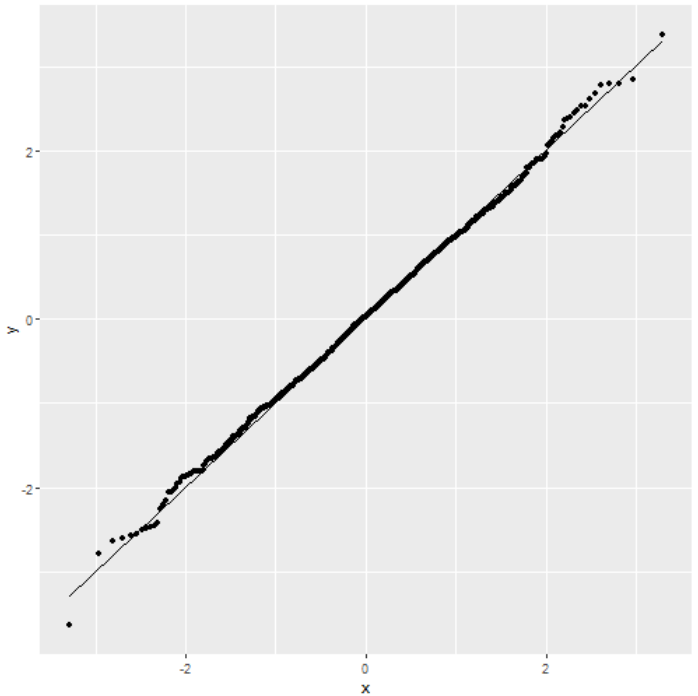

```
ggplot(mpg, aes(x=class,y=hwy, fill=cl
  geom_boxplot() +
  stat_summary(fun.y=mean,
               geom="point", shape=5,
```



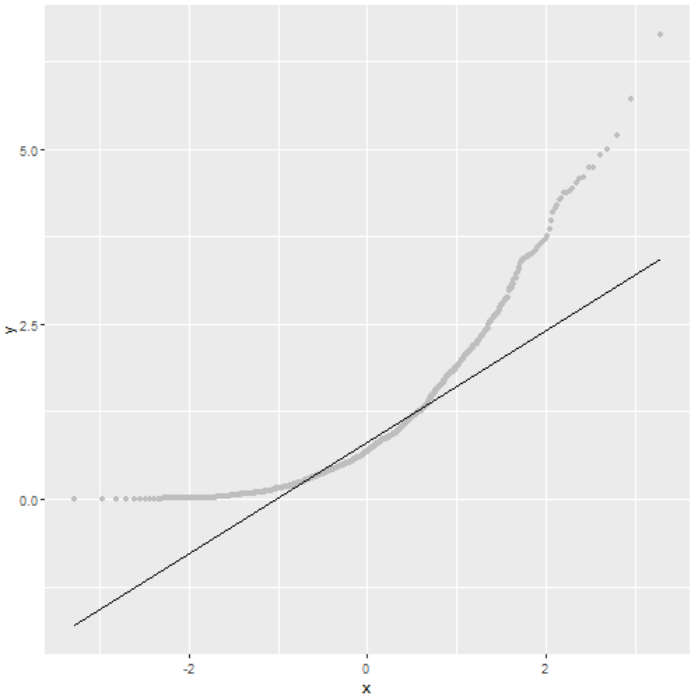
# 例子：Q-Q图

```
data <- data.frame(x = rnorm(1000), y = rexp(1000))
```

```
ggplot(data, aes(sample=x))+
  stat_qq()+stat_qq_line()
```



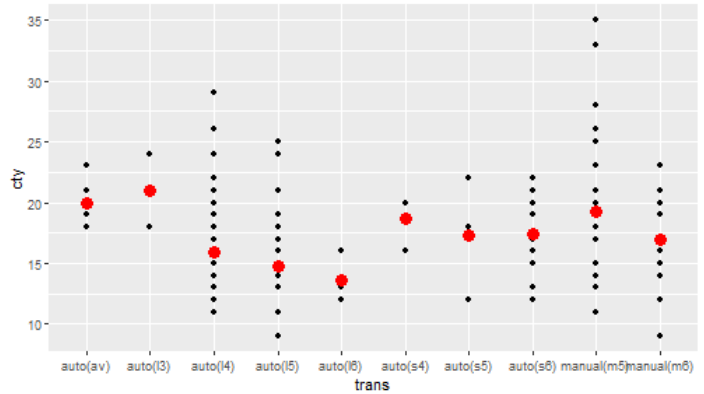

```
ggplot(data, aes(sample=y))+
  stat_qq(col="grey") + stat_qq_line()
```



# 统计变换和几何对象

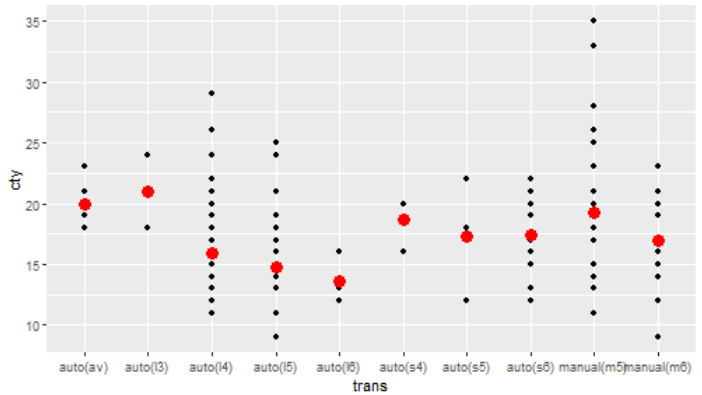
- 统计变换和几何对象有着默认的匹配关系，若要改变，可通过 stat_***()函数内修改默认的几何对象，或通过geom_***()函数内修改默认的统计变换。如：

```
ggplot(mpg, aes(trans, cty)) + geom_po
  stat_summary(geom = "point", fun.y =
```




```
ggplot(mpg, aes(trans, cty)) + geom_po
  geom_point(stat = "summary", fun.y =
```

```
## No summary function supplied, defaulting to `
```




- 《ggplot2：数据分析与图形艺术（第2版）》一书的作者更推荐右边的形式，因为它使得展示的概述---而不是原始数据---变得更清晰。

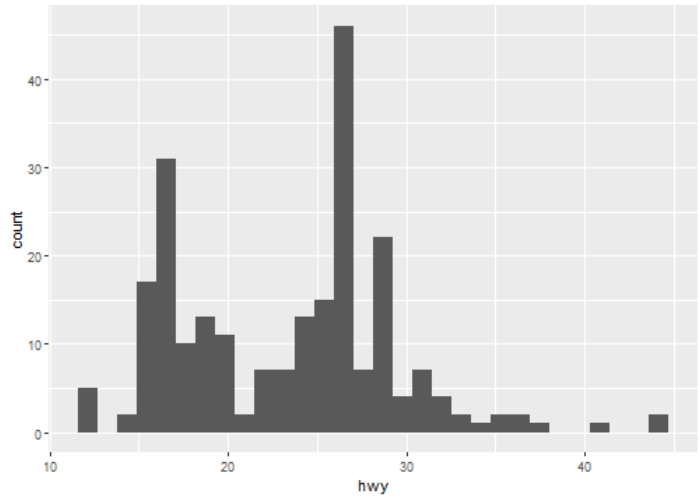
# 统计变换生成的变量

- 统计变换内部采用一种数据框的形式作为输入并返回一个数据框，因此可通过统计变换增加新的变量到原始数据集中。如
  - `stat_bin()` 直方图产生以下变量：
    - `count`：每个组里观测值的数目
    - `density`：每个组里观测值的密度
- 如要使用这些生成变量（generated variable），名字必须要用".."围起来，防止重名造成混淆，且能让你看出来哪些是由统计变换而来的。

# 例子：直方图和密度图

```
ggplot(data = mpg, aes(x = hwy)) +
  stat_bin()
```

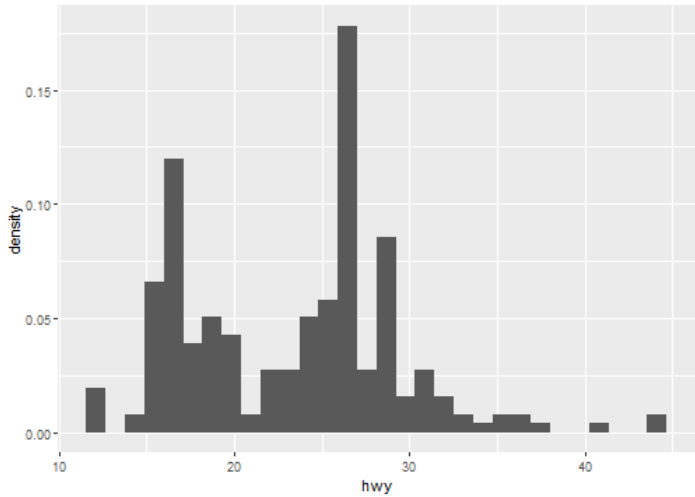
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = mpg, aes(x = hwy)) +

  stat_bin(aes(y = ..density..))
```

```
## `stat_bin()` using `bins = 30`. Pick better v
```



# 标度

# 标度：scale

- 标度控制着数据到图形属性的映射，将数据转化为视觉上可以看得到的东西，如大小，颜色，形状等。
- 标度提供了读图时所用的工具：坐标轴和图例。
- 标度是由三个分离的下划线”_“组成：
    1. scale
    2. 图形属性的名字，如colour, shape, x, y, fill
    3. 标度的名字，如
        - 连续时间标度：用于映射整数、数值、时间等到x轴或y轴。如continuous
        - 颜色标度：用来映射连续或离散数据到颜色上，如brewer, grey
        - 人工标度：用来映射离散数据到所选的尺寸、线性或颜色上，如discrete, manual
- 更多的可见《ggplot2：数据分析与图形艺术（第2版）》的第六章

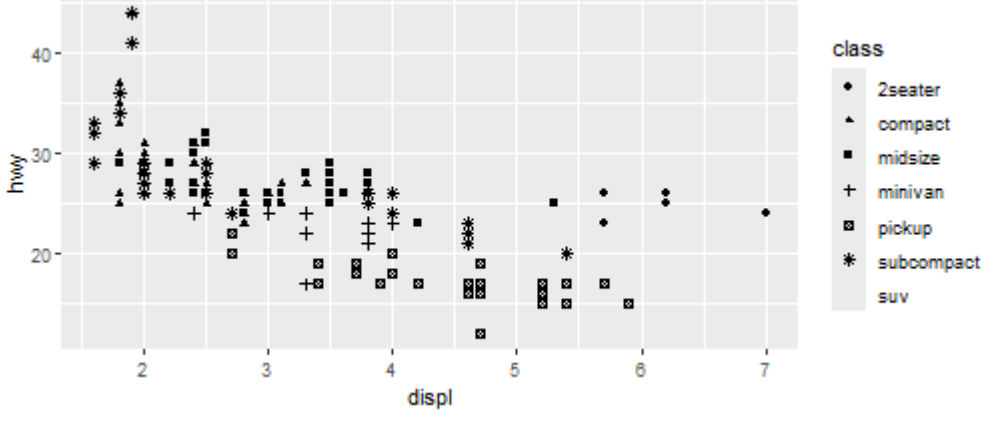
# 例子：散点图（一）

- 回顾之前画的一个散点图：""

```
g <- ggplot(data = mpg, aes(x = displ, y = hwy, shape = class)) + geom_point(); g
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more
## than 6 becomes difficult to discriminate
## ℹ you have requested 7 values. Consider specifying shapes manually if you need
##   that many have them.
```

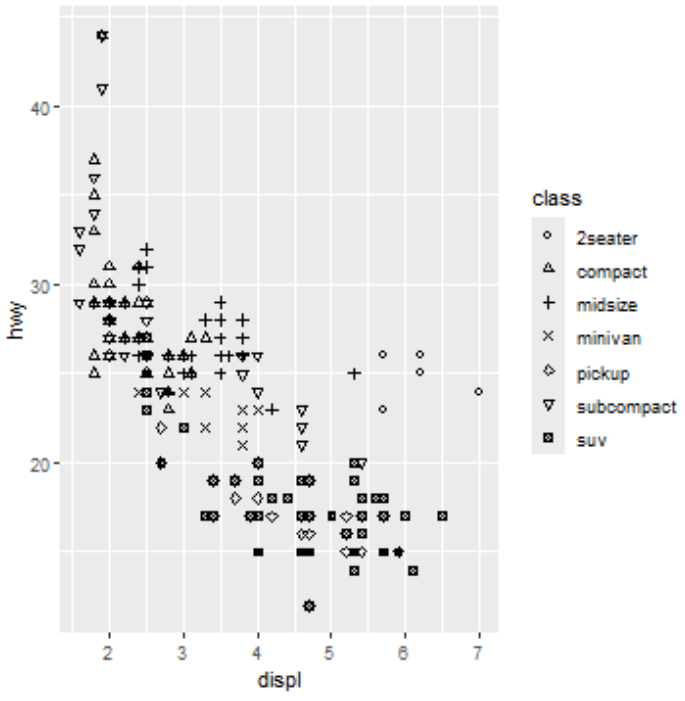
```
## Warning: Removed 62 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



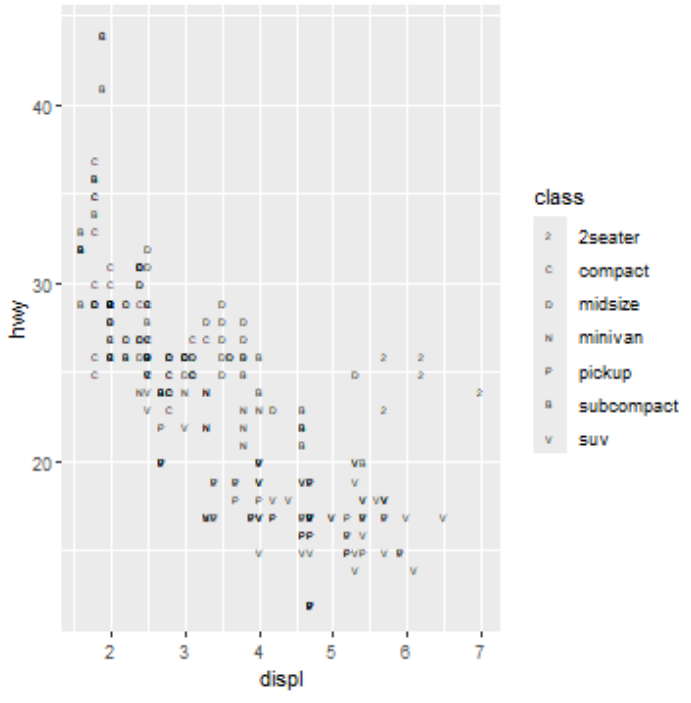

- suv类的点并没有画出来，因为默认的最多只能有6类。

# 例子：散点图（二）

```
g + scale_shape_manual(values = 1:7)
```



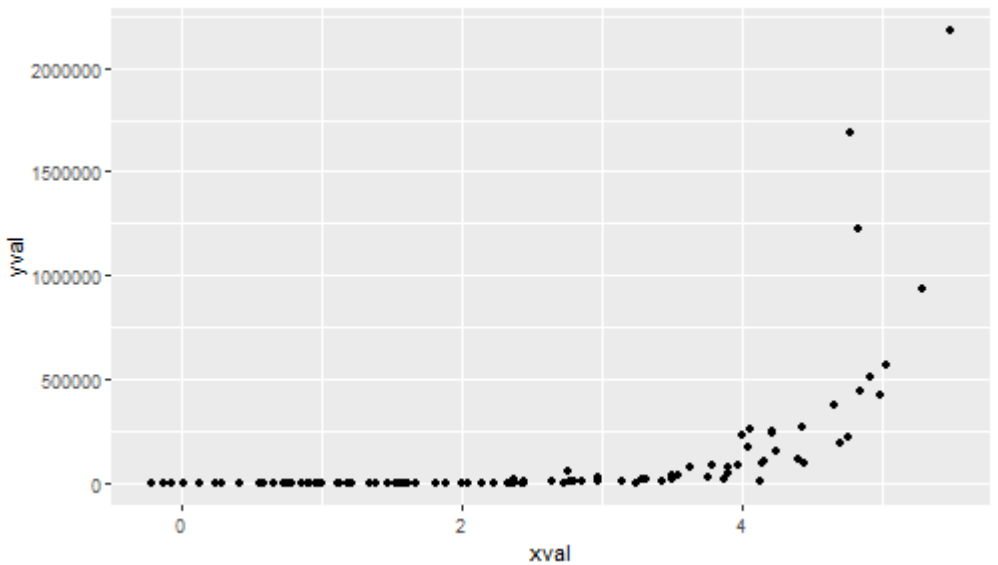

```
g + scale_shape_manual(values = c("2",
```



# 连续位置标度

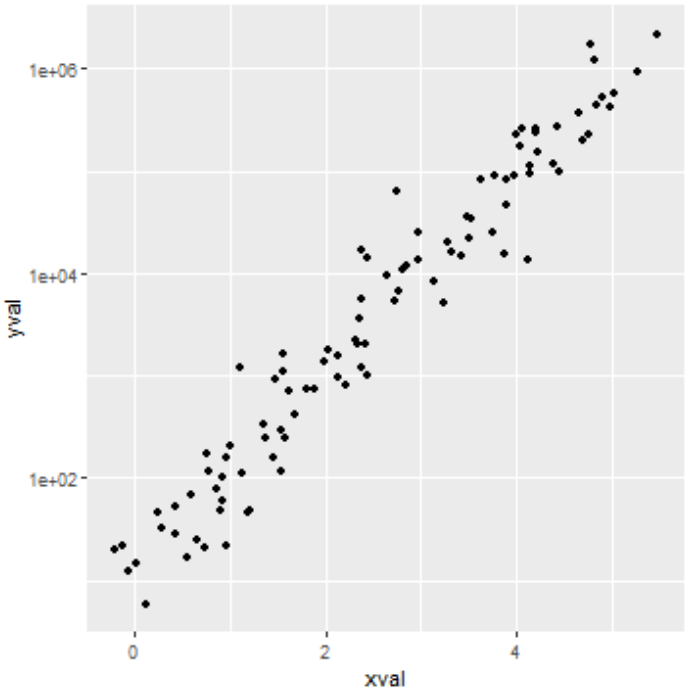
- scale_x_continuous()和scale_y_continuous()默认是线性地将数据映射到x轴和y轴上，我们可以通过trans =对其进行变换，如以下幂指数函数

```
set.seed(201)
n <- 100
dat <- data.frame(xval = (1:n+rnorm(n,sd=5))/20, yval = 10*10^((1:n+rnorm(n,sd=5))
sp <- ggplot(dat, aes(xval, yval)) + geom_point()
sp
```

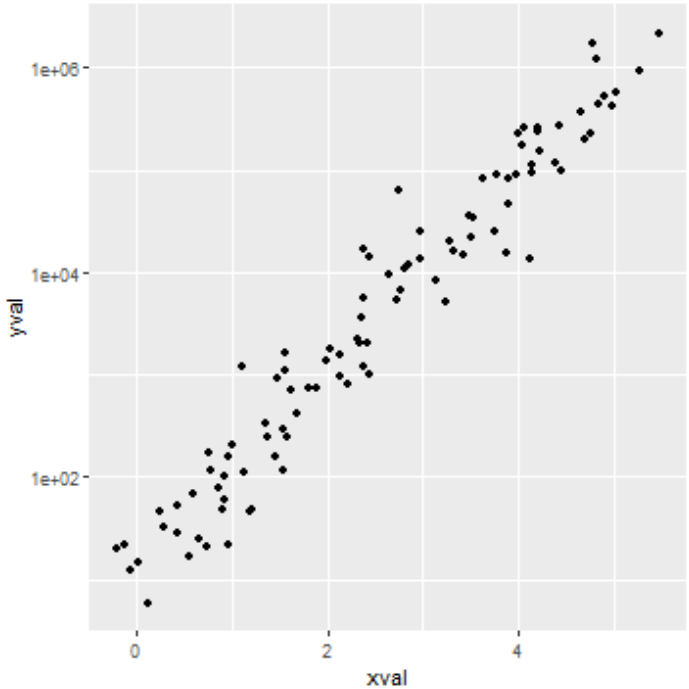


# 例子：连续位置标度

```
sp + scale_y_continuous(trans = "log10
```



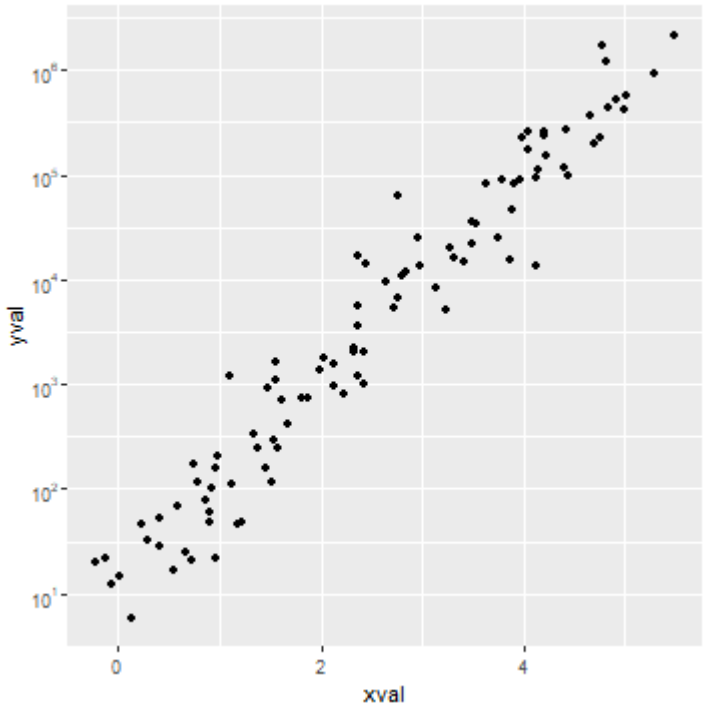

```
sp + scale_y_log10()
```



# 例子：连续位置标度

- 对坐标也进行变换

```
library(scales)      # Need the scales package
sp + scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                   labels = trans_format("log10", math_format(10^.x)))
```



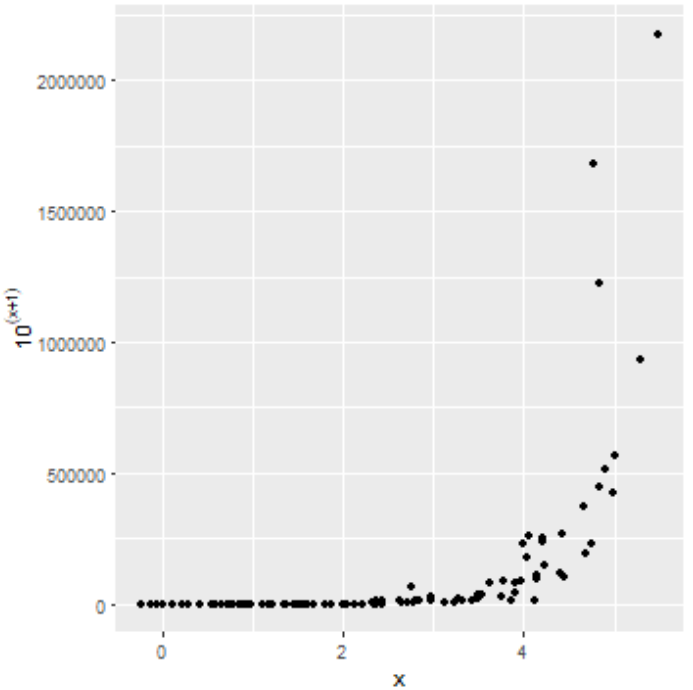
# 标度标签和限度

- 可通过 scale_x_continuous()和scale_y_continuous()来改变标度标签和限度，但推荐使用更简化的函数：
  - lims, xlim, ylim: 设定x轴和y轴的画图范围
  - xlab, ylab, labs: 设定x轴和y轴的标签
  - ggtitle: 设定图的标题

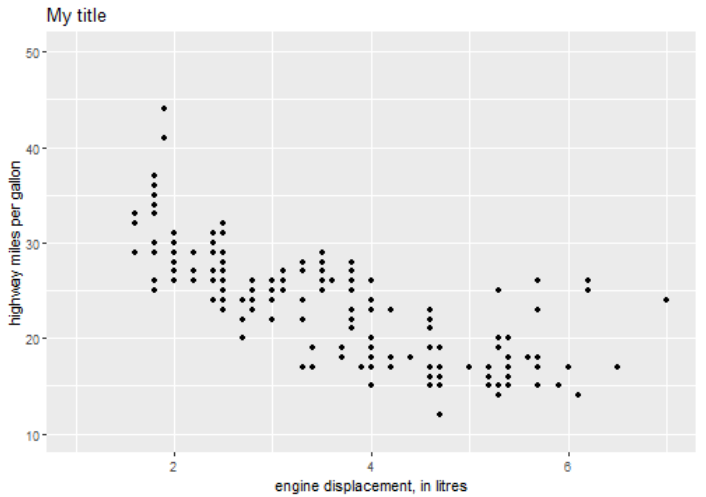
```
p <- ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point()
```

# 例子：标度标签和限度

```
sp + scale_x_continuous("x") + scale_y
```




```
p + lims(x=c(1,7), y=c(10,50)) +
  labs(x= "engine displacement, in lit
  ggtitle("My title")
```



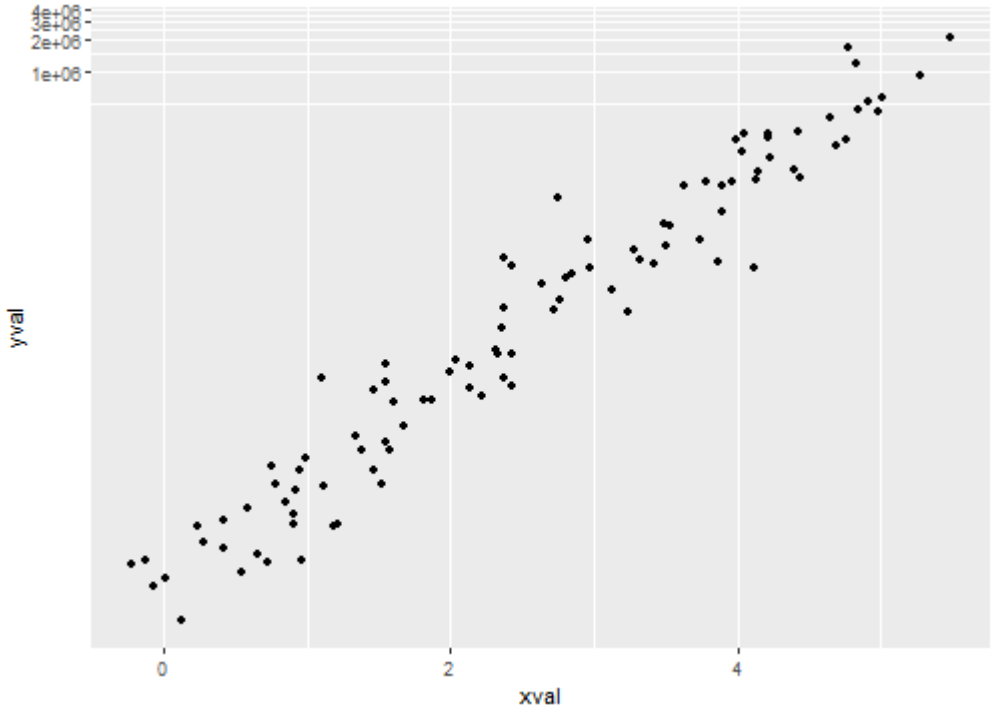
# 坐标系

# 坐标系

- 坐标系有两种：
  - 线性坐标系：保持了几何对象的形状。
    - coord_cartesian()：默认的笛卡尔坐标系
    - coord_fixed()：宽高比固定的直角坐标系
    - coord_flip()：x轴和y轴反转了的笛卡尔坐标系
  - 非线性坐标系：可以改变形状，如直线不再是直线，非欧空间。
    - coord_map()/coord_quickmap()：地图投影
    - coord_polar：极坐标系
    - coord_trans()：对数据进行统计变换之后，对x和y位置进行任意变换。

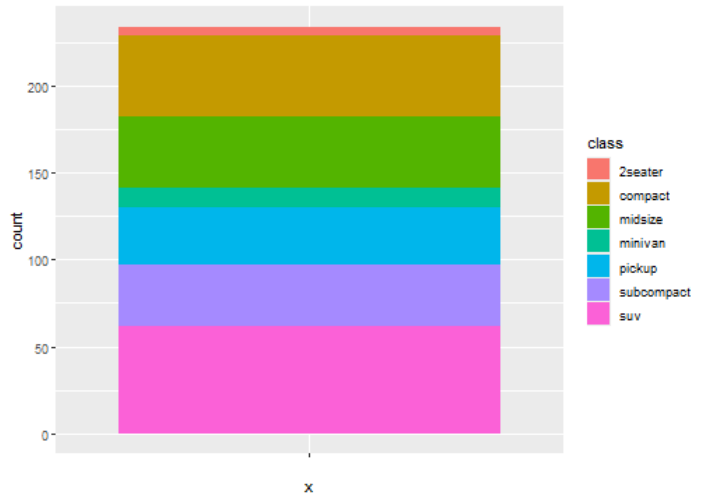
# 例子：对y轴取log变换

```
sp + coord_trans(y="log10")
```

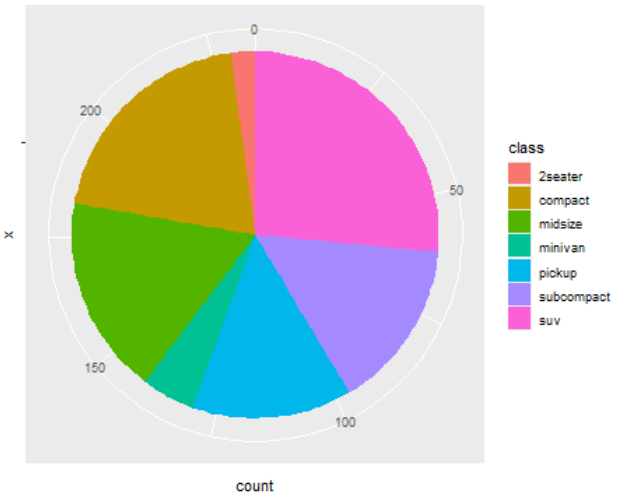


# 例子：柱状图的极坐标系化——饼图

```
p <- ggplot(mpg, aes(x="", fill=class)
p + geom_bar()
```

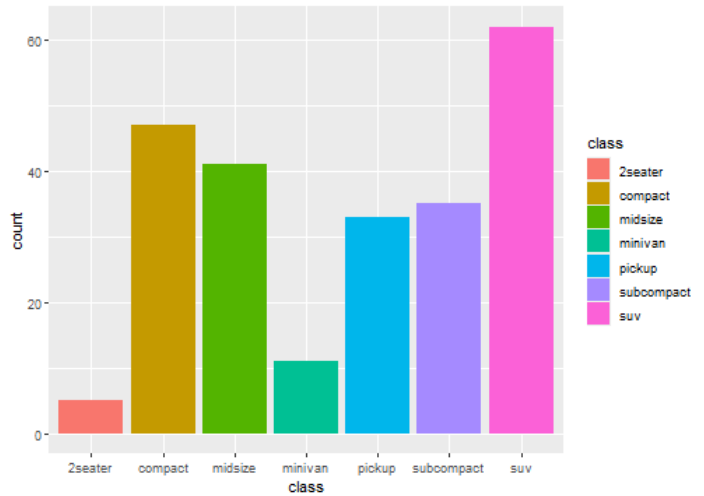



```
p + geom_bar() + coord_polar(theta="y"
```

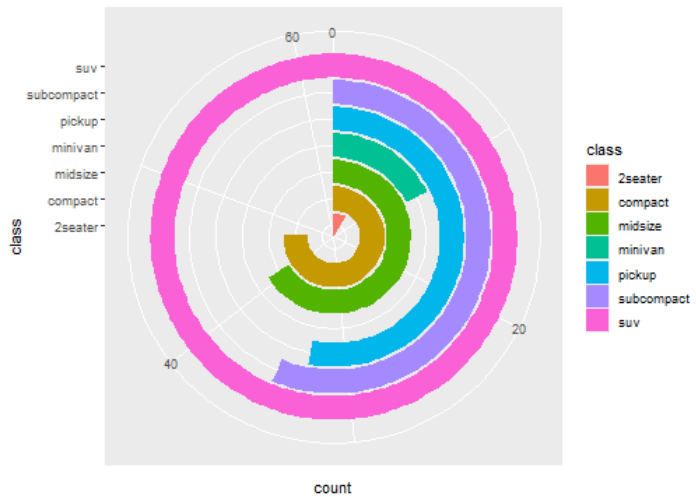


# 例子：柱状图的极坐标系化——牛眼图

```
p <- ggplot(mpg, aes(x=class, fill=cla
p + geom_bar()
```

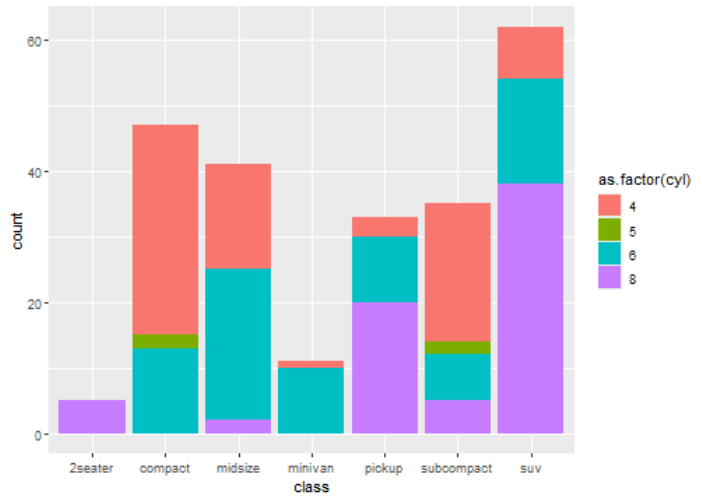



```
p + geom_bar() + coord_polar(theta="y"
```

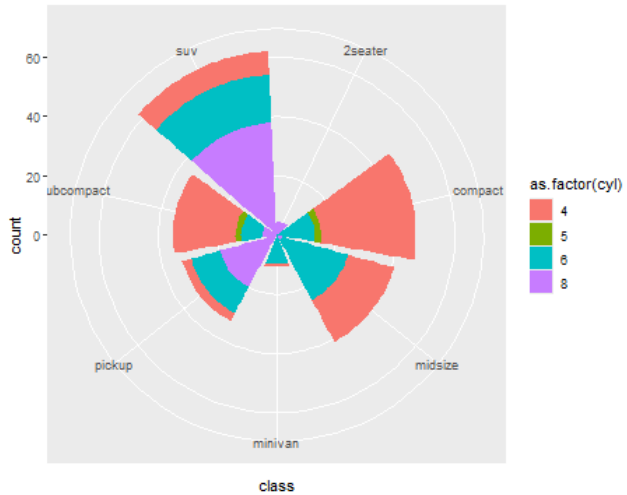


# 例子：柱状图的极坐标系化——玫瑰图

```
p <- ggplot(mpg, aes(x=class, fill=as.
p + geom_bar()
```



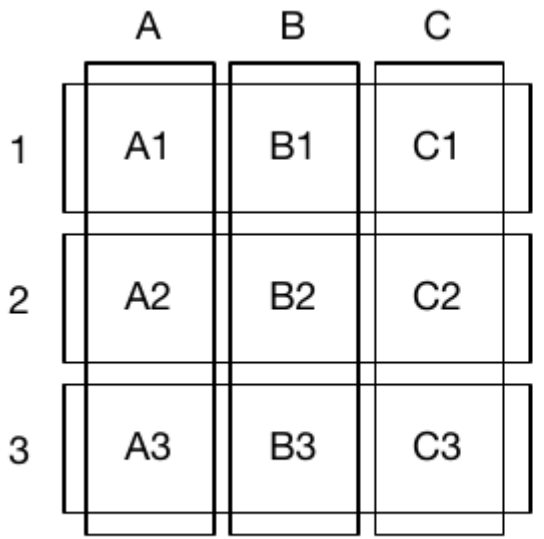

```
p + geom_bar() + coord_polar()
```



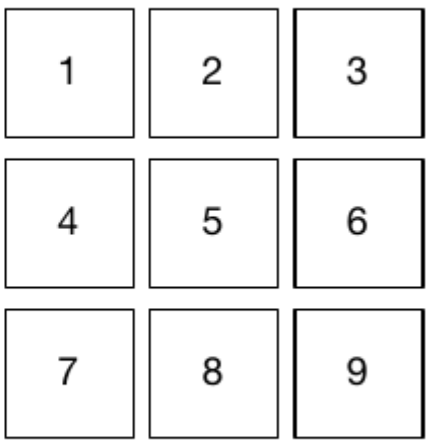
# 分面

# 分面

- 分面通过切割数据生成一系列小连号图，有三种分面类型：
  - `facet_null()`：单个图像，默认情况。
  - `facet_wrap()`：把1维面板条状封装在2维中。
  - `facet_grid()`：生成一个2维的面板网格，其中行和列由变量组成。




**facet_grid**




**facet_wrap**

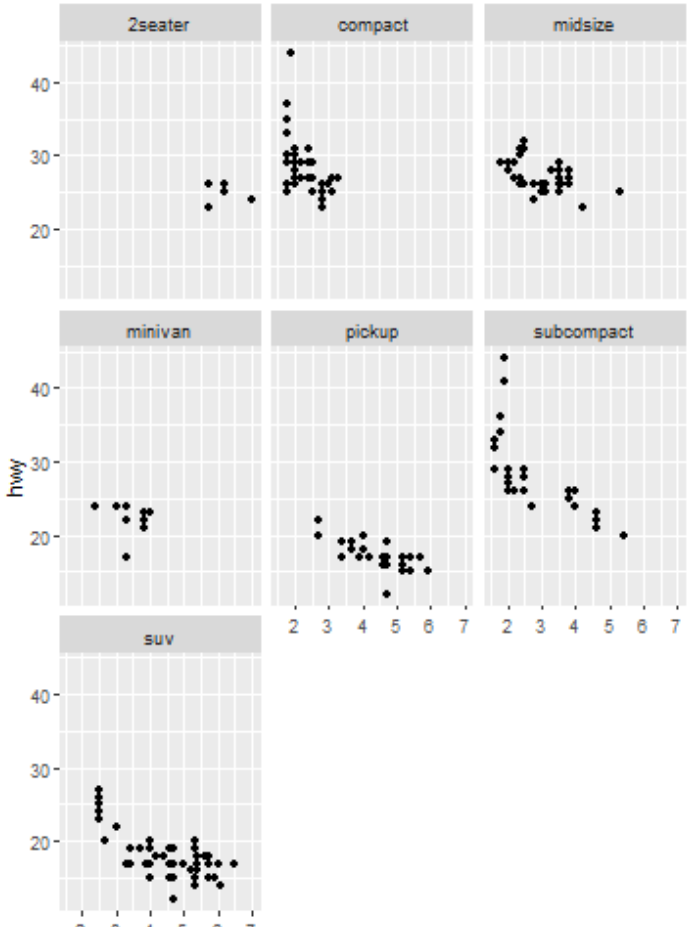
# 封装分面facet_wrap()

- facet_wrap()函数把1维面板条状封装在2维中。在处理单个多水平变量时，特别有用。
- 通过 nrow, ncol, as.table, dir来控制网格如何封装条块。
  - nrow, ncol控制有多少行/列（只要设置其中一个即可）
  - as.table控制分面的布局， TRUE最高值显示在右下角，反之则显示在右上角
  - dir控制封装的方向， h 表示横向， v 表示纵向

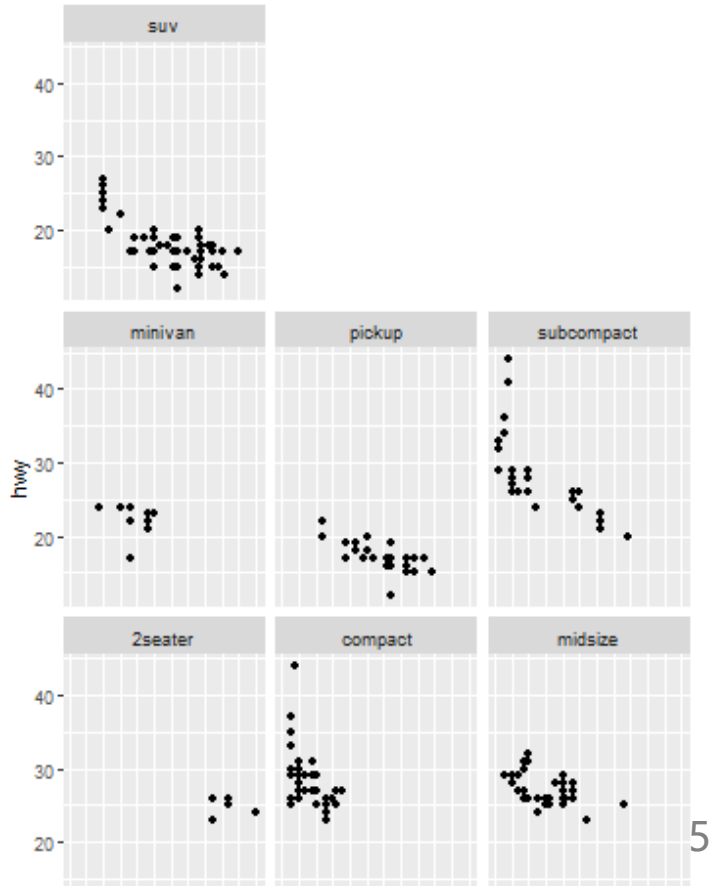
```
g <- ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point()
```

# 例子：封装分面

```
g + facet_wrap(~ class, nrow = 3)
```




```
g + facet_wrap(~ class, nrow = 3, as.t
```

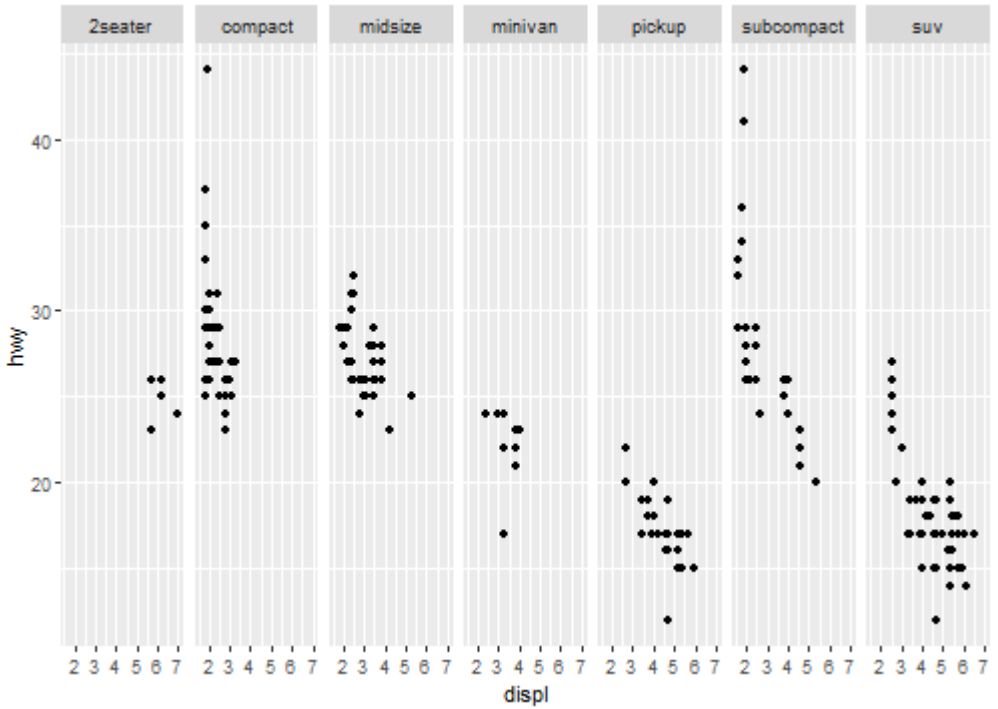


# 网格分面facet_grid()

- facet_grid()函数在2维网格中展示图像，用以下表达式定义：
  - . ~ a 把a的值按列展开
  - b ~ . 把b的值按行展开
  - b ~ a 把a的值按列展开， 把b的值按行展开。可以是多个的，如 a + b ~ c + d

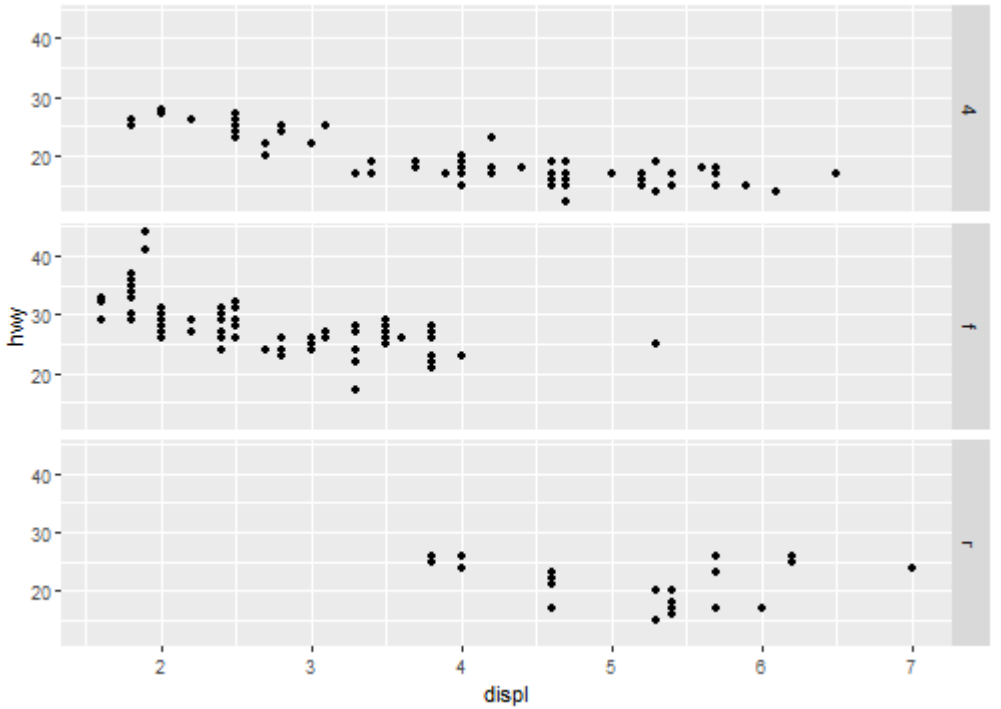
# 例子： 网格分面（一）

```
g + facet_grid(~ class)
```



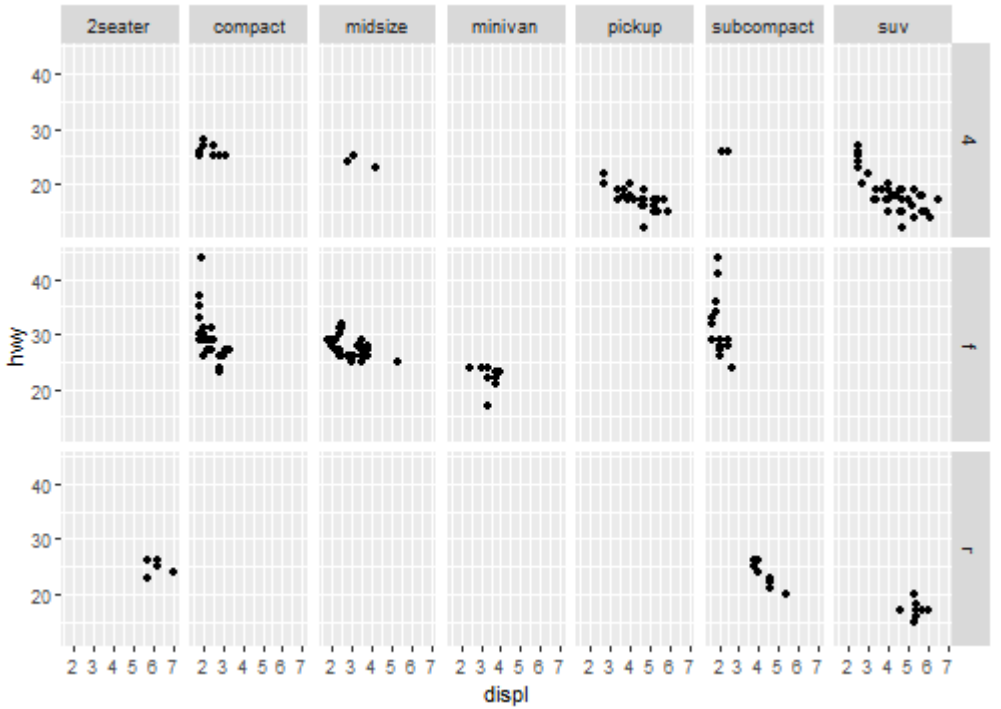
# 例子： 网格分面（二）

```
g + facet_grid(drv ~ .)
```



# 例子： 网格分面（三）

```
g + facet_grid(drv ~ class)
```

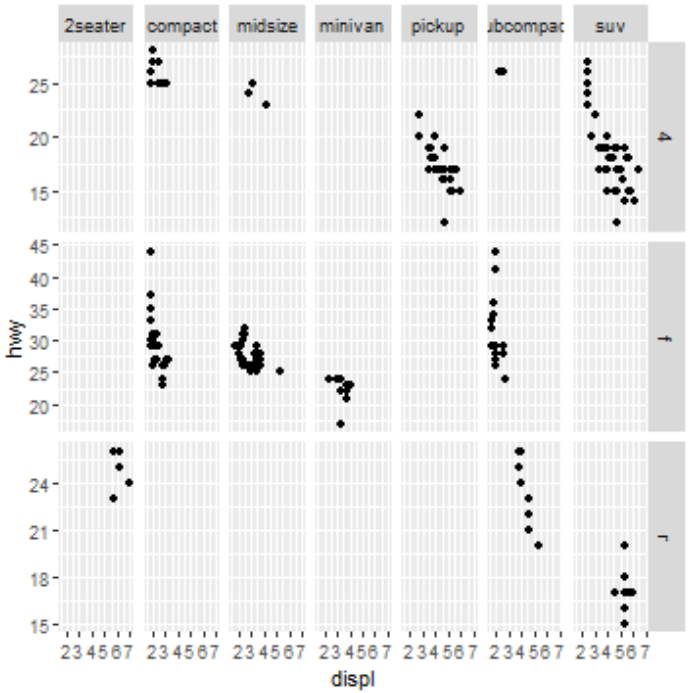


# 标度控制

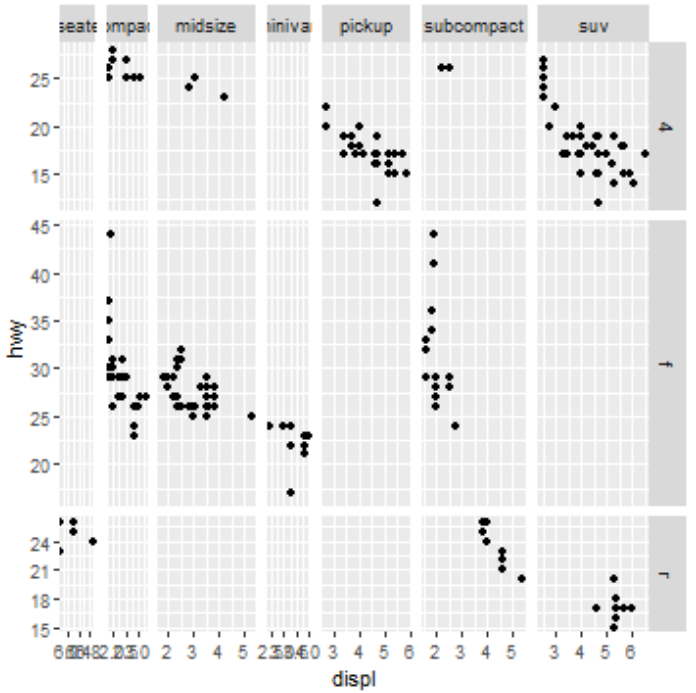
- 通过调整参数 scales 来控制面板的位置标度是相同的还是允许变化的。
  - scales = "fixed": x和y的标度在所有面板中都固定
  - scales = "free_x": x的标度可变，y的标度固定
  - scales = "free_y": y的标度可变，x的标度固定
  - scales = "free": x和y的标度在每个面板都可以变化

# 例子：标度控制

```
g + facet_grid(drv ~ class, scales = "
```




```
g + facet_grid(drv ~ class, scales = "
```

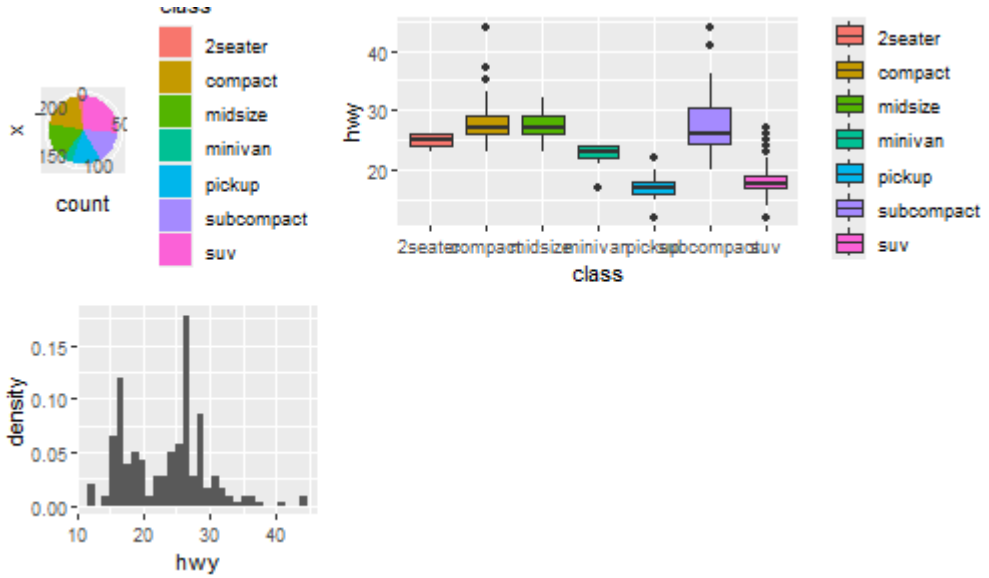


# 多个子图

```
library("gridExtra")
p1 <- ggplot(mpg, aes(x="", fill=class))+geom_bar()+coord_polar(theta="y")
p2 <- ggplot(mpg, aes(x=class, y=hwy, fill=class))+geom_boxplot()
p3 <- ggplot(mpg, aes(x=hwy,y=..density..)) + geom_histogram()

grid.arrange(p1, p2, p3, ncol=2, widths=c(1,2))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

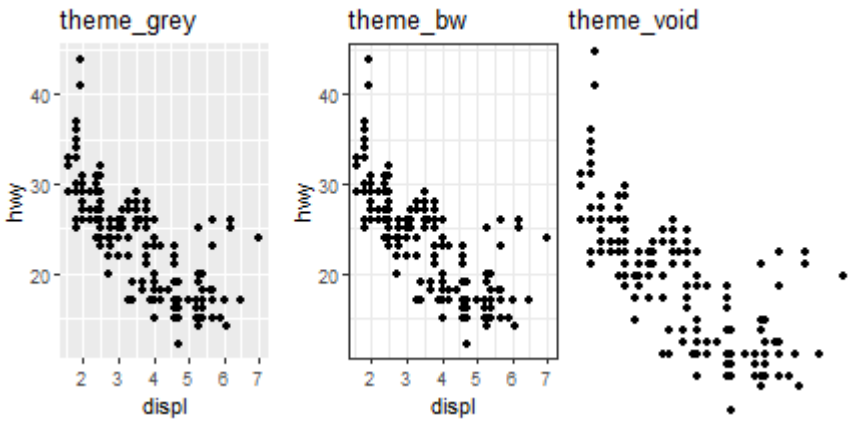


# 主题

# 主题

- 主题对图像中的非数据元素进行精细调整，不影响几何对象和标度等数据元素。
- ggplot2自带了八种内置主题，分别为theme_grey(), theme_bw(), theme_linedraw(), theme_light(), theme_dark(), theme_minimal(), theme_classic(), theme_void()

```
p1 <- g + theme_grey() + ggtitle("theme_grey")
p2 <- g + theme_bw() + ggtitle("theme_bw")
p3 <- g + theme_void() + ggtitle("theme_void")
grid.arrange(p1, p2, p3, ncol=3)
```

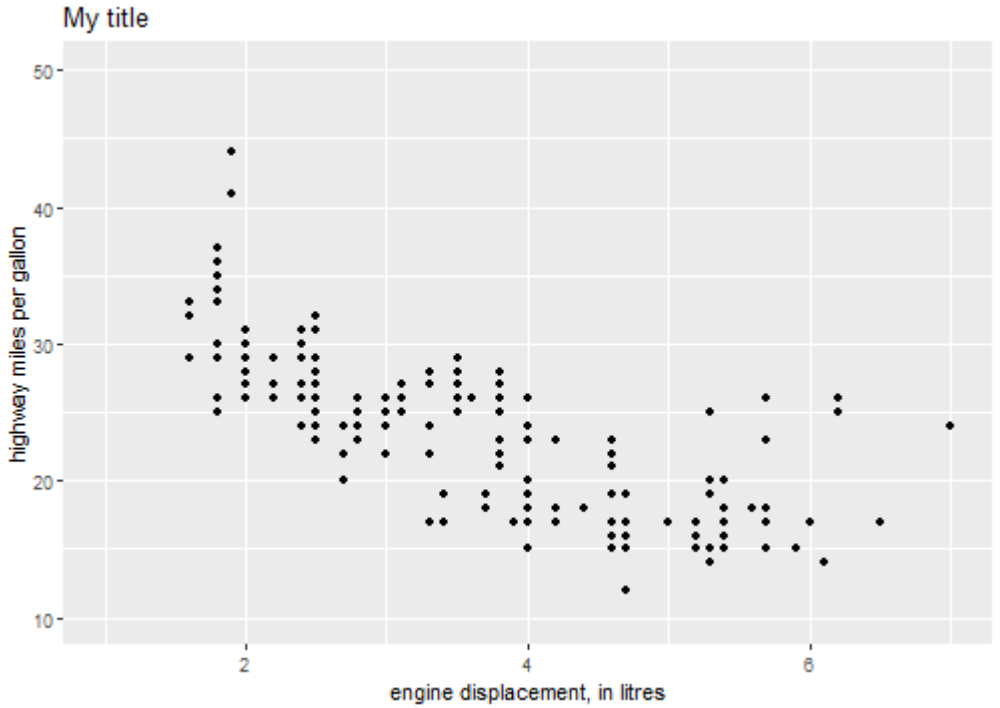


# 主题之单个主题组件

- 修改单个主题组件，则需要使用形如 `plot + theme(element.name = element_function())`
- 主题元素（`element.name`）制定了能控制的非数据元素，如 `plot.title`控制了图像标题的外观， `axis.ticks.x`控制了x轴上的标签。
- 元素函数（`element_function`）描述的是元素的视觉属性， 如`element_text()`设定了字体的大小、颜色。内置的元素函数有四种基本类型：
  - 文字（`element_text`）：绘制标签和标题，如控制字体的family(字体族)、face(字型)、colour(颜色)、size(大小)、hjust(横向对齐)等
  - 线条（`element_line`）：绘制线条，参数有colour(颜色)、size(大小)和linetype(线条类型)
  - 矩形（`element_rect`）：绘制（背景的）矩形，参数有fill(填充)的颜色、边缘的colour(颜色)、size(大小)和linetype(线条类型)
  - 空白（`element_blank`）：不绘制任何东西。

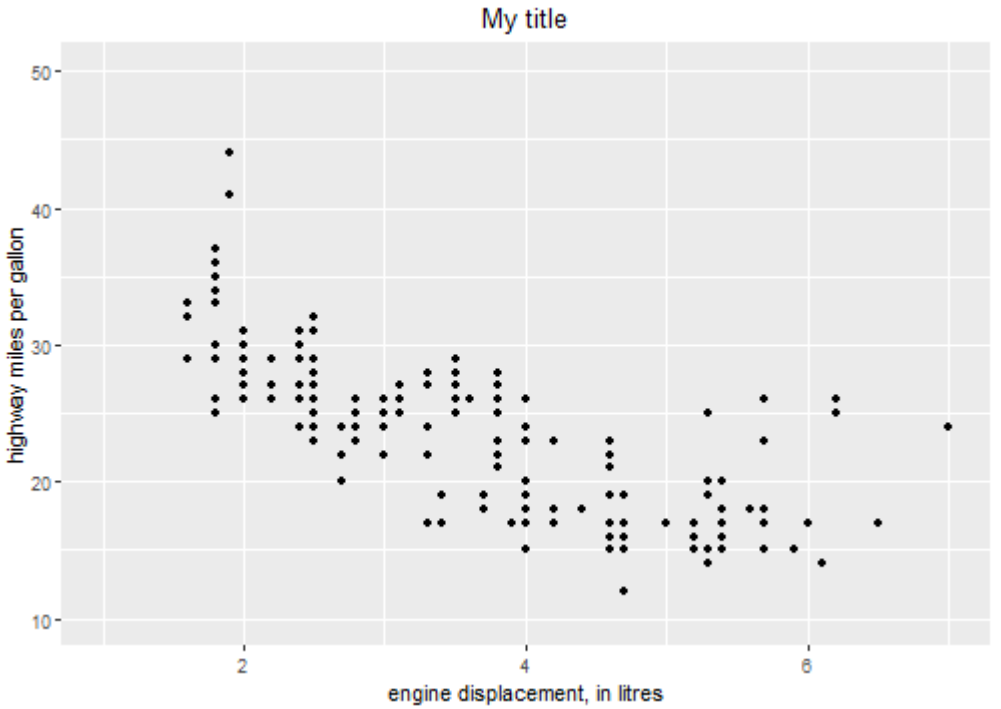
# 例子

```
pt <- ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point() +
  lims(x=c(1,7), y=c(10,50)) +
  labs(x= "engine displacement, in litres", y = "highway miles per gallon")
pt + ggtitle("My title")
```



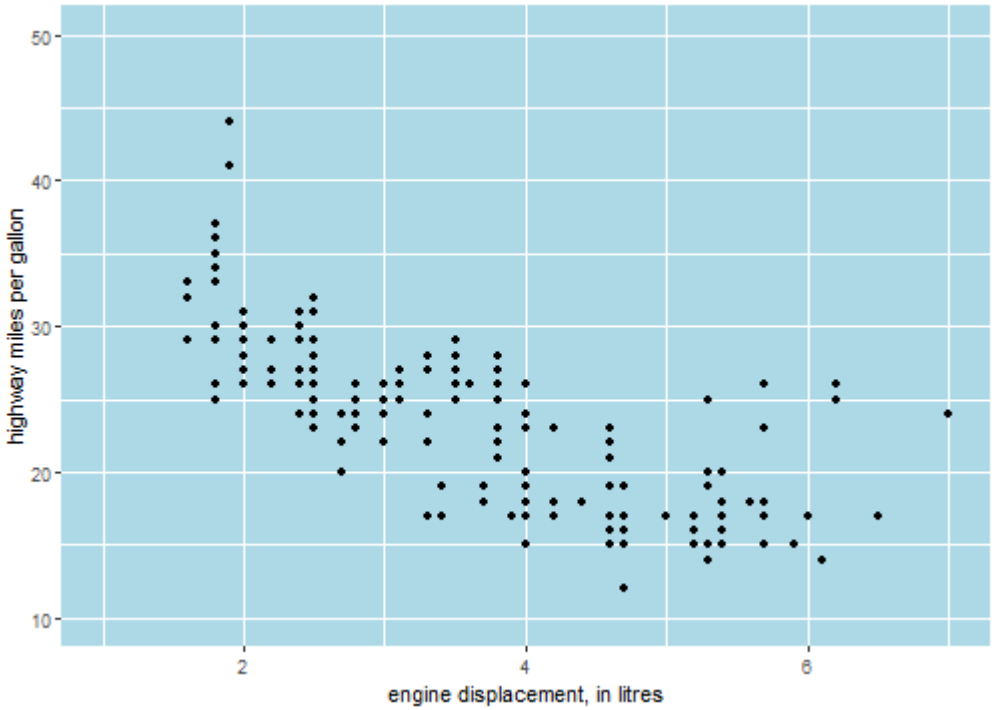
# 例子：标题居中

```
pt + ggtitle("My title") + theme(plot.title = element_text(hjust = 0.5))
```



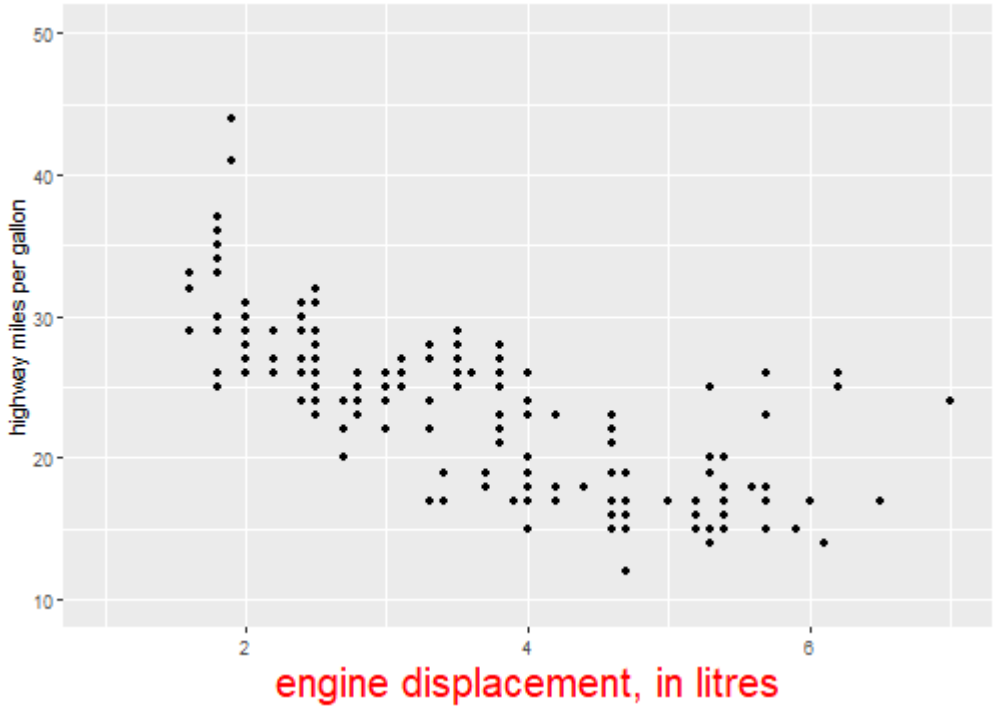
# 例子：改变背景颜色

```
pt + theme(panel.background = element_rect(fill = "lightblue"))
```



# 例子：在x轴标签改变文字格式

```
pt + theme(axis.title.x = element_text(size = 20, color = "red"))
```



# 例子：去掉背景

```
pt + theme(panel.background = element_blank(), axis.title.x = element_blank())
```

# 保存图片

- 可以使用标准做法来保存图片，也就是先打开一个图形设备，生成图像，然后关闭设备。
- 更推荐ggsave()，这是为了图像交互而优化过的函数，可将图像存储到外面文件中。

```
ggsave("plot.pdf")
```

```
## Saving 7 x 7 in image
```

```
p <- ggplot(data = mpg, aes(x = displ, y = hwy)) + geom_point()
ggsave("myplot.png", plot=p, width=7, height=5, units = "in")
```

# 案例分析：南丁格尔玫瑰图

```
# Load some R packages:
library(tidyverse) # for data manipulation and visualization.
```

```
## ── Attaching core tidyverse packages ──────────────────────────────────────── t
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1
## ── Conflicts ──────────────────────────────────────────────────────────────────
## ✖ readr::col_factor() masks scales::col_factor()
## ✖ dplyr::combine()    masks gridExtra::combine()
## ✖ purrr::discard()    masks scales::discard()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ dplyr::lag()        masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
```

```
library(stringr) # for text prcessing.
library(lubridate) # for date-time processing.
library(hrbrthemes) # For using black theme.
```

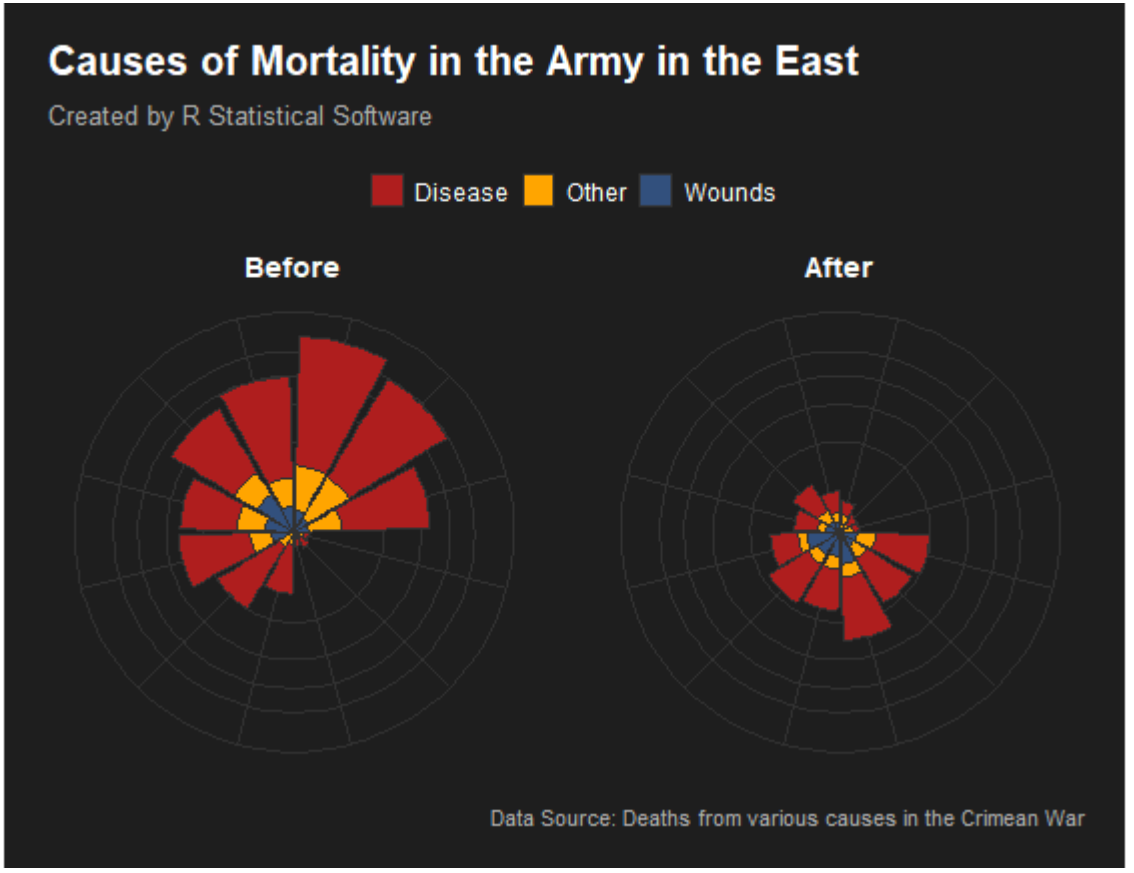
# 案例分析：南丁格尔玫瑰图

```
# Load Nightingale data set:
data("Nightingale", package = "HistData")

# Convert to long form and processing for some columns:
Nightingale %>%
  select(Date, Disease.rate, Wounds.rate, Other.rate) %>%
  gather(Cause, Deaths, -Date) %>%
  mutate(Cause = str_replace_all(Cause, "\\.rate", ""), Regime = rep(c("Before", "
  mutate(Regime = factor(Regime, levels = c("Before", "After"))) %>%
  mutate(mo = month(Date, label = TRUE, abbr = TRUE)) -> longForm_data
```

# 案例分析：南丁格尔玫瑰图

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```



# 案例分析：南丁格尔玫瑰图

```
longForm_data %>%
  ggplot(aes(x = mo, y = Deaths, fill = Cause)) +
  geom_col(color = "grey20") +
  theme_modern_rc() +
  scale_fill_manual(values = c("firebrick", "orange", "#365181"), name = "") +
  scale_y_sqrt() +
  facet_wrap(~ Regime) +
  coord_equal(ratio = 1) +
  coord_polar() +
  labs(title = "Causes of Mortality in the Army in the East",
       subtitle = "Created by R Statistical Software",
       caption = "Data Source: Deaths from various causes in the Crimean War") +
  theme(legend.position = "top") +
  theme(text = element_text(size = 14)) +
  theme(axis.title.y = element_blank()) +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.y = element_blank()) +
  theme(axis.ticks = element_blank()) +
  theme(plot.margin = unit(rep(0.7, 4), "cm")) +
  theme(plot.title = element_text(color = "white",size = 20)) +
  theme(plot.caption = element_text(color = "grey70", size = 11)) +
  theme(plot.subtitle = element_text(color = "grey70", size = 13)) +
  theme(legend.text = element_text(color = "white", size = 12)) +
  theme(strip.text = element_text(color = "white", size = 14, face = "bold",hjust
```

# 谢 谢