作者：黄宝臣  
链接：https://www.zhihu.com/question/24779017/answer/38750383  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

我也来占个坑吧。。。我主要讲理念的东西，希望大家有一定经验的再来看此答案。

总结来说有以下几点：

* ggplot2的核心理念是将绘图与数据分离，数据相关的绘图与数据无关的绘图分离
* ggplot2是按图层作图
* ggplot2保有命令式作图的调整函数，使其更具灵活性
* ggplot2将常见的统计变换融入到了绘图中。

**==================================================================  
  
1、ggplot2的逻辑。**

ggplot2的逻辑在我看来其实是真正实现了一个图层叠加的概念：一句语句代表一张图，然后再有最小的单元图层。这个与其他命令式的绘图完全不同，来做个比较：

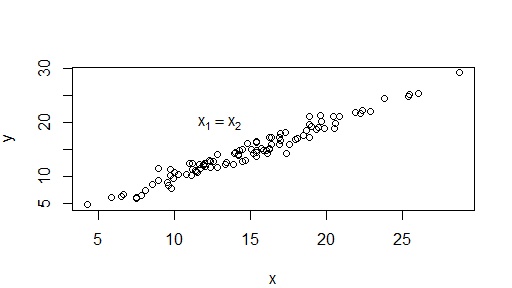
#这是基于graphic包里例子

x <- rnorm(100,14,5)

y <- x + rnorm(100,0,1)

plot(x,y)

text(13,20, expression(x[1] == x[2]))

输出的图是这样的：  
&lt;img src="https://pic4.zhimg.com/0d88322f1aeb67d53c1ab64859f1120b\_b.jpg" data-rawwidth="512" data-rawheight="288" class="origin\_image zh-lightbox-thumb" width="512" data-original="https://pic4.zhimg.com/0d88322f1aeb67d53c1ab64859f1120b\_r.jpg"&gt;我们可以看到这种绘图方式实际上是按命令添加的，以plot开始，可以以任何方式结束，每加上一个元素，实际上都是以一句单独的命令来实现的。这样做的缺点就是，其实不符合人对于画图的一般认识。其次，就是，我们没有一个停止绘图的标志，这使得有时候再处理的时候就会产生一些困惑。优势其实也有，在做参数修改的时候，我们往往可以很方便地直接用一句单独的命令修改，譬如对于x轴的调整，觉得不满意就可以写命令直接调整。而ggplot2则意味着要重新作图。

我们可以看到这种绘图方式实际上是按命令添加的，以plot开始，可以以任何方式结束，每加上一个元素，实际上都是以一句单独的命令来实现的。这样做的缺点就是，其实不符合人对于画图的一般认识。其次，就是，我们没有一个停止绘图的标志，这使得有时候再处理的时候就会产生一些困惑。优势其实也有，在做参数修改的时候，我们往往可以很方便地直接用一句单独的命令修改，譬如对于x轴的调整，觉得不满意就可以写命令直接调整。而ggplot2则意味着要重新作图。

再来看ggplot2的代码：

x <- rnorm(100,14,5)

y <- x + rnorm(100,0,1)

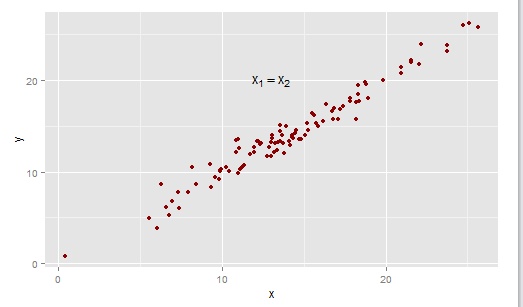
ggplot(data= NULL, aes(x = x, y = y)) + #开始绘图

geom\_point(color = "darkred") + #添加点

annotate("text",x =13 , y = 20,parse = T,

label = "x[1] == x[2]") #添加注释

画出的结果如下：

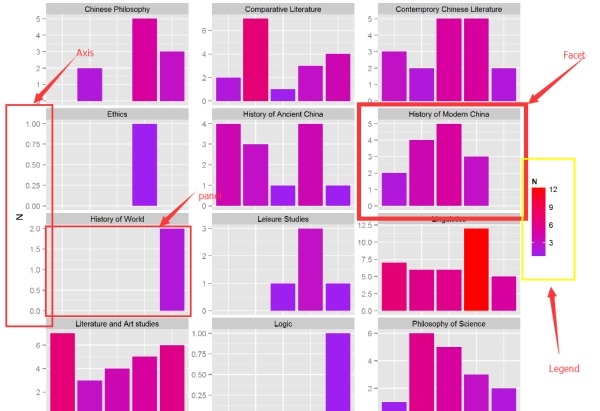
&lt;img src="https://pic1.zhimg.com/c0244d96538457f8d8615ab2606a9ca0\_b.jpg" data-rawwidth="523" data-rawheight="307" class="origin\_image zh-lightbox-thumb" width="523" data-original="https://pic1.zhimg.com/c0244d96538457f8d8615ab2606a9ca0\_r.jpg"&gt;我们可以发现，ggplot的绘图有以下几个特点：第一，有明确的起始（以ggplot函数开始）与终止（一句语句一幅图）；其二，图层之间的叠加是靠“+”号实现的，越后面其图层越高。

我们可以发现，ggplot的绘图有以下几个特点：第一，有明确的起始（以ggplot函数开始）与终止（一句语句一幅图）；其二，图层之间的叠加是靠“+”号实现的，越后面其图层越高。

其次就是对于分组数据的处理，其实这方面，lattice已经做得很好了，不过我会在后面更仔细地叙述ggplot2是怎么看分组数据的绘图的。

**2. ggplot2的要素**

我们这里不谈qplot（quickly plotting）方法，单纯谈ggplot方法。不谈底层的实现思想，我们简单地理解，ggplot图的元素可以主要可以概括如下：最大的是plot（指整张图，包括background和title），其次是axis（包括stick，text，title和stick）、legend（包括backgroud、text、title）、facet这是第二层次，其中facet可以分为外部strip部分（包括backgroud和text）和内部panel部分（包括backgroud、boder和网格线grid，其中粗的叫grid.major，细的叫grid.minor）。大致见下图，这部分内容的熟悉程度直接影响到对于theme的掌握，因此希望大家留心。

&lt;img src="https://pic3.zhimg.com/91b50b32d16552032b9867f4e7075a6a\_b.jpg" data-rawwidth="1002" data-rawheight="687" class="origin\_image zh-lightbox-thumb" width="1002" data-original="https://pic3.zhimg.com/91b50b32d16552032b9867f4e7075a6a\_r.jpg"&gt;

**3. ggplot2图层以及其他函数的分类**

好了，现在把这些理念的东西讲完了之后，下面来理解ggplot2里的绘图命令。

ggplot2里的所有函数可以分为以下几类：

* 用于运算（我们在此不讲，如fortify\_，mean\_等）
* 初始化、展示绘图等命令（ggplot，plot，print等）
* 按变量组图（facet\_等）
* 真正的绘图命令（stat\_，geom\_，annotate），这三类就是实现一个函数一个图层的核心函数。
* 微调图型：严格意义上说，这一类函数不是再实现图层，而是在做局部调整。
  + scale\_：直译为标尺，这就是与aes内的各种美学（shape、color、fill、alpha）调整有关的函数。
  + guides：调整所有的text。
  + coord\_：调整坐标。
  + theme：调整**不与数据有关**的图的元素的函数。

**4. 绘图**  
**第一步：初始化。**ggplot2风格的绘图的第一步就是初始化，说白了就是载入数据空间、选择数据以及选择默认aes。

p <- ggplot(data = , aes(x = , y = ))

data就是载入你要画的数据所在的数据框，指定为你的绘图环境，载入之后，就可以免去写大量的$来提取data.frame之中的向量。当然，如果你的数据都是向量，也可不指定，但是要在申明中标注**data = NULL**，不然就会得到不必要的报错。  
第二个是重头戏，即aes，是美学（aesthetic）的缩写。这是在ggplot2初学者眼里最不能理解的东西，甚至很多老手也会在犹豫，什么时候要把参数写在aes里，什么时候要写在aes外。我们做一个简单的，不非常恰当的解释：**任何与数据向量顺序相关，需要逐个指定的参数都必须写在aes里**。这之后我们会进一步解释，现在我们初始化的时候，最好只是把关于位置的x和y指定一下就好。

**第二部，绘制图层。**

很多人在解释ggplot2的时候喜欢说，ggplot2绘图有两种函数，一类是geom\_，绘图用的；一类是stat\_，统计变换用的。这样说不是不对，只是很不恰当，很多人就会问出一些问题，比如，统计变换竟然是做运算用的，为什么可以用来画图？为什么stat\_bin和geom\_histgram画出来的图是一样，竟然一样，为什么要重复？  
事实上，**任何一个ggplot2图层都包括stat和geom俩部分，或者说两个步骤（其实还包括position）。** 而stat\_identity则表示不做任何的统计变换。

我们来举个例子，还是上面的代码，为了更直观，我在此作了修改：

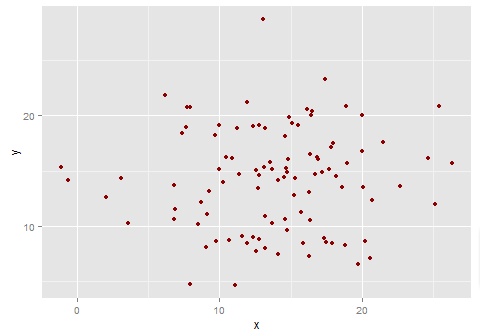
x <- c(rnorm(100,14,5),rep(20,20))

y <- c(rnorm(100,14,5) + rnorm(100,0,1),rep(20,20))

ggplot(data= NULL, aes(x = x, y = y)) + #开始绘图

geom\_point(color = "darkred")

做出的图如下：

&lt;img src="https://pic2.zhimg.com/371e71d4fff98d0b03968e45154fafa5\_b.jpg" data-rawwidth="480" data-rawheight="336" class="origin\_image zh-lightbox-thumb" width="480" data-original="https://pic2.zhimg.com/371e71d4fff98d0b03968e45154fafa5\_r.jpg"&gt;我们查看码源，就知道geom\_point的默认stat是identity，即不做任何统计变换：

我们查看码源，就知道geom\_point的默认stat是identity，即不做任何统计变换：

> geom\_point

function (mapping = NULL, data = NULL, stat = "identity", position = "identity",

na.rm = FALSE, ...)

{

GeomPoint$new(mapping = mapping, data = data, stat = stat,

position = position, na.rm = na.rm, ...)

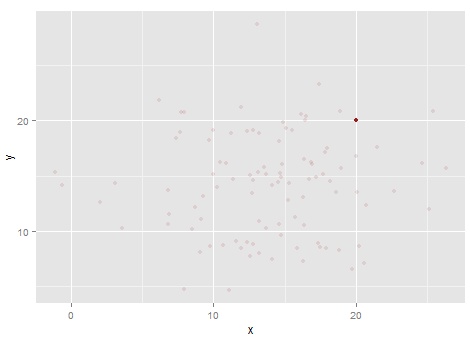
}

<environment: namespace:ggplot2>

大家可以发现，我在（20，20）这个点的数据事实上是有20个的，但由于没做统计转换(20,20)这个点被画了20次，因此我们理论上看到的点其实是最后一次画的那个点。可能这不够直观，没关系，我们调整一下透明度到10%：

ggplot(data= NULL, aes(x = x, y = y)) + #开始绘图

geom\_point(color = "darkred",alpha = 0.1)

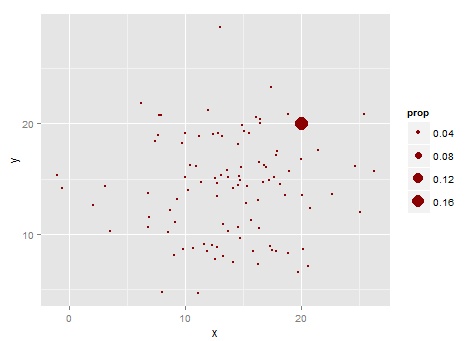
得到如下图：  
&lt;img src="https://pic3.zhimg.com/07c938d197cc884c80013e9738773e9a\_b.jpg" data-rawwidth="470" data-rawheight="340" class="origin\_image zh-lightbox-thumb" width="470" data-original="https://pic3.zhimg.com/07c938d197cc884c80013e9738773e9a\_r.jpg"&gt;这样应该就很明显了，由于(20,20)点被画了20次，所以透明度会叠加为20\*10% = 200%实际只展现100%。

这样应该就很明显了，由于(20,20)点被画了20次，所以透明度会叠加为20\*10% = 200%实际只展现100%。

我们现在就使用坐标转换来重新画这个图：

ggplot(data= NULL, aes(x = x, y = y)) + #开始绘图  
geom\_point(color = "darkred",stat = "sum")

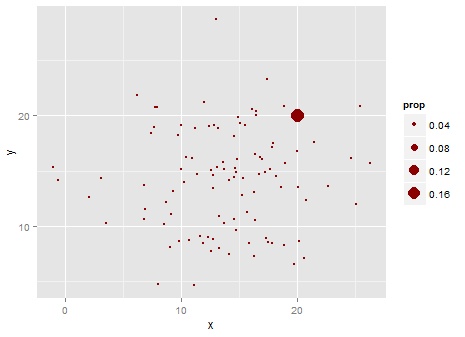
好了，解释一下，stat\_sum实际的意思就是按照某一点占所有点出现频率然后换算成大小来作图，因此，以上代码就可以得到下面这张图，因为(20,20)这个点出现频率为20/120=16.667%：

&lt;img src="https://pic3.zhimg.com/64601598881a6e192415c4819d272ed2\_b.jpg" data-rawwidth="470" data-rawheight="341" class="origin\_image zh-lightbox-thumb" width="470" data-original="https://pic3.zhimg.com/64601598881a6e192415c4819d272ed2\_r.jpg"&gt;好了，我们可以发现了，一个单纯的geom\_point里面也是带有stat\_的，因此，其实geom\_和stat\_实际上是一回事。可能你会问了，那照我的说法，以上这幅图用的是geom\_point里的一个参数，而不是再用stat\_sum，这是一回事吗？bingo！这个问题相当好，的确，按照以上的推理，应该存在一种以stat\_sum作为主函数的方法来绘制这幅图，搞不好，里面还有个参数geom，要设置成“point”。我们来实践一下吧：

好了，我们可以发现了，一个单纯的geom\_point里面也是带有stat\_的，因此，其实geom\_和stat\_实际上是一回事。可能你会问了，那照我的说法，以上这幅图用的是geom\_point里的一个参数，而不是再用stat\_sum，这是一回事吗？bingo！这个问题相当好，的确，按照以上的推理，应该存在一种以stat\_sum作为主函数的方法来绘制这幅图，搞不好，里面还有个参数geom，要设置成“point”。我们来实践一下吧：

ggplot(data= NULL, aes(x = x, y = y)) + #开始绘图

stat\_sum(color = "darkred",geom = "point")

&lt;img src="https://pic3.zhimg.com/a9382345aa56d6d0d6391c80b8c300fa\_b.jpg" data-rawwidth="465" data-rawheight="337" class="origin\_image zh-lightbox-thumb" width="465" data-original="https://pic3.zhimg.com/a9382345aa56d6d0d6391c80b8c300fa\_r.jpg"&gt;尼玛，还真可以，还长得一模一样。

尼玛，还真可以，还长得一模一样。

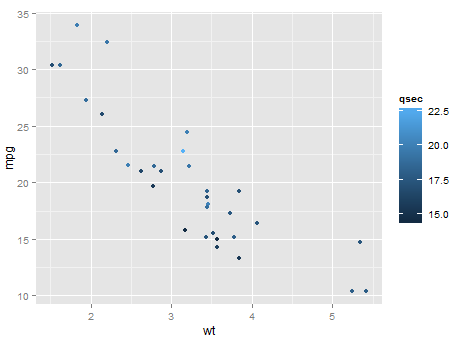
现在就讲通了，对于有过经验的同学现在应该重新修正这个观点——stat\_和geom\_是两种绘图方法。这是错的，其实它们是ggplot2每一个图层绘制都必须有的，是一个图层的**一体两面**。

在这一步之中，我们也要回到我们在第一步时出现的问题，aes到底是什么？为什么说**任何与数据向量顺序相关，需要逐个指定的参数都必须写在aes里？**什么时候color、shape、size、fill写外面，什么时候写里面？

aes实际上做的是将aes里的向量的顺序逐个地绘制。譬如以下代码（转自geom\_point帮助文档中的实例）：

p <- ggplot(mtcars, aes(wt, mpg)) #<---- code 1

p + geom\_point(aes(colour = qsec)) #<---- code 2

结果是：  
&lt;img src="https://pic1.zhimg.com/385e9ae6cbff7baa71b52a2fc009f894\_b.jpg" data-rawwidth="467" data-rawheight="348" class="origin\_image zh-lightbox-thumb" width="467" data-original="https://pic1.zhimg.com/385e9ae6cbff7baa71b52a2fc009f894\_r.jpg"&gt;  
我们来分析一下ggplot2是怎么作图的。首先，我们来看一下mtcars这个数据集长什么样：

> head(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb

Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1

Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2

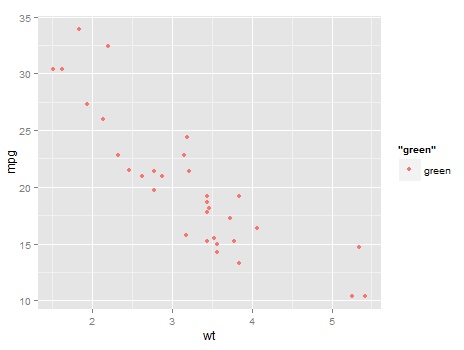
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

code 1: ggplot首先载入了这个mtcars的集合，然后指定给了mpg作为其x坐标位置，wt为y坐标位置。  
code 2: 指定了qsec作为其染色的标准（分组），qsec为numeric变量，因此，应该选择连续型的标尺，而不是分组染色。然后开始绘制，读取mtcars$mpg[1]、mtcars$wt[1]，确定位置，然后为其染成mtcars$qsec[1]颜色；再绘制第二点。。。

因此，aes里的美学特征其实就是按照向量顺序指定每个位置的美学特征，大家可以比较tapply函数的写法。

好了，现在问题就来了。我想为所有点的颜色都染成绿色，怎么办？其实很简单，如果不需要指定这么一个染色的顺序，而选择将整个图层染成一种颜色，则只需要将color写在aes外：

p + geom\_point(color = "green")

哦，怪不得写在aes里染出来的颜色不是绿色，但为什么写到里面就不可以了，为了写到里面，然出来的是粉色？  
&lt;img src="https://pic1.zhimg.com/66d541f62b634fb6166b3fa4d89d7048\_b.jpg" data-rawwidth="464" data-rawheight="350" class="origin\_image zh-lightbox-thumb" width="464" data-original="https://pic1.zhimg.com/66d541f62b634fb6166b3fa4d89d7048\_r.jpg"&gt;好了，我们再来分析一下把color = "green"写到了aes里，到底发生了什么。

好了，我们再来分析一下把color = "green"写到了aes里，到底发生了什么。

p + geom\_point(aes(colour = "green"))

首先，数据的初始化跟上面那个例子是相同的。然后，因为color放到了aes里，于是ggplot开始搜索mtcars里面的向量了，发现没有叫"green"的，然后又找了global，也没有。于是，ggplot就开始把它认作了一个新的向量。等等，有个问题，我要按照这个向量来分别染色，而事实上，这个向量长度为1，怎么办？ggplot就先把他展开成了factor(rep("green",nrow(mtcars)),levels = unique("green"))，bingo！现在开始染色了。啊第一个数据mtcars$mpg[1]、mtcars$wt[1]，其颜色变量是"green",因子水平是1，染成默认调色第一种，哦，就是这个蛋蛋的粉红色；再染第二个，还是"green"，因子水平也是1，染成蛋蛋的粉红色；... 终于完成了，咦？怎么都是蛋蛋的粉红色。

通过举了这个染色的例子大家应该都弄懂了，aes到底在干什么了。其他的美学特征其实也是完全一致的。只是需要解释group=1的意思就是说不做分组来进行绘图。什么？还是搞不清该放aes里面还是外面？那就记着想统一整个图层时就放到aes外，想分成不同组调整，并且已经有一个与x、y长度一致的分组变量了，那就放到aes里。

在这一步里，还要要说的就是我们要讲的是ggplot2大致内置了哪些图：

* 点（point, text）：往往只有x、y指定位置，有shape但没有fill
* 线(line,vline,abline,hline,stat\_function等)：一般是基于函数来处理位置
* 射(segment)：特征是指定位置有xend和yend，表示射线方向
* 面(tile, rect)：这类一般有xmax,xmin,ymax,ymin指定位置
* 棒(boxplot,bin,bar,histogram)：往往是二维或一维变量，具有width属性
* 带(ribbon,smooth):透明是特征是透明的fill
* 补：包括rug图，误差棒(errorbar,errorbarh)

然后，就是按照你的需要一步步加图层了（使用“+”）。

**第三部，加注释。**所有注释的实现都是通过annotate函数实现的，其实annotate就是一个最简单的geom\_单元，它一次只添加一个位置上的图形（可以通过设置向量来实现同时绘制多个图形，但这个理念和注释的理念有所偏差）。annotate的geom就是指定注释的类型，其属性按照geom的不同而发生变化。

**第四步，调整。**这里的调整主要是使用微调图形这大类的函数做美学特征、坐标轴、标题、绘图主题的调整。这部分也就是继承了命令式作图的思想，使ggplot2的灵活性增加。

如何搜索你要用什么美学特征调整函数，其实就是按照美学特征的名字来，例如，你要调整的是fill，就找scale\_fill\_之后就有一些不同的染色方法（关于色彩，如果有时间还会添加相关知识）；调整的是横坐标标尺，就找scale\_x\_然后后面跟上你的横坐标类型；其他雷同。

在调整主题这方面，值得褒奖的是，theme函数其实最妙的地方是将对于数据相关的美学调整和与数据无关的美学调整分离了。譬如说，我们要改变x轴的颜色，或者panel的底色，这个其实与数据处理无关，这种分离就会使得我们可以如此流程化地操作作图，而不需要在考虑数据的时候还要关注到与数据无关的美学参数。有人有时候会觉得ggplot2很奇怪的地方就是为什么调整legend的时候，有时要用scale\_，有时又要用theme，其实这都是对于ggplot2这个设计理念的不理解，作者的设计思路是要将数据处理与数据美学分开，数据美学与数据无关的调整分开。  
其次，theme函数采用了四个简单地函数来调整所有的主题特征：element\_text调整字体，element\_line调整主题内的所有线，element\_rect调整所有的块，element\_blank清空。这种设计相当地棒。

由此，一个极具诚意的作图应该长成下面这个样子：

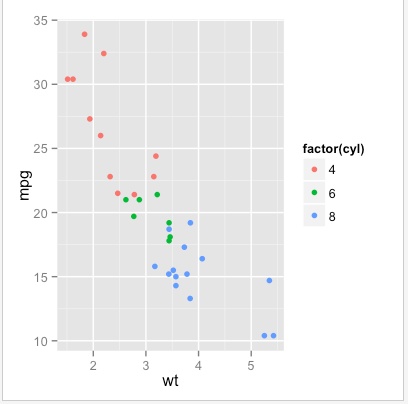
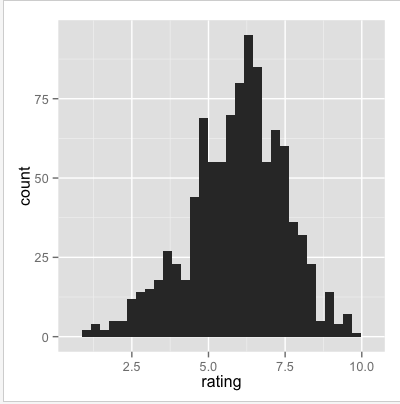
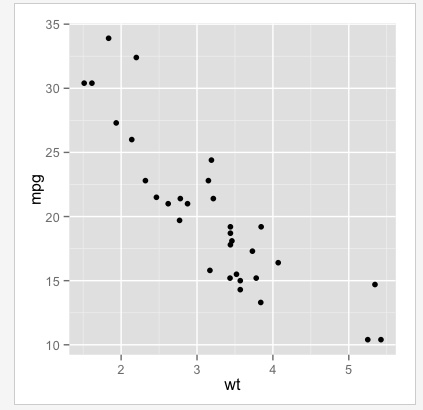
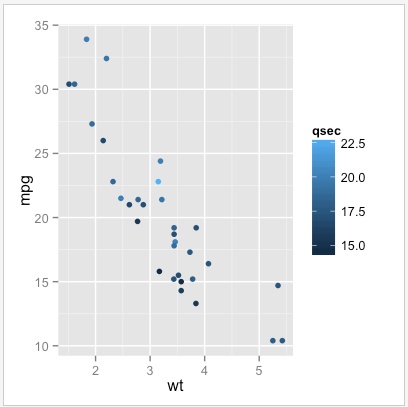
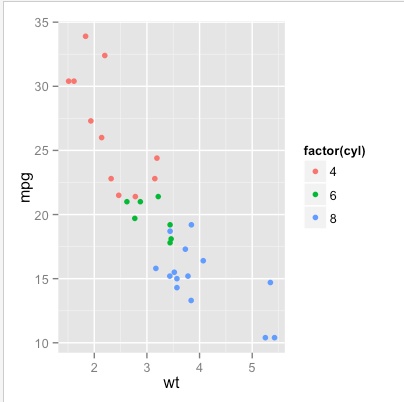
ggplot(data = , aes(x = , y = )) +

geom\_XXX(...) + ... + stat\_XXX(...) + ... +

annotate(...) + ... +

scale\_XXX(...) + coord\_XXX(...) + guides(...) + theme(...)

5. ggplot2的一些缺点

* 公式支持不好，自带的plotmath公式无法满足很多需求
* 无法针对多个legends进行调整
* 效率不高，绘图速度较慢，这也表示二次开发的可能性不高
* 作者：顾志耐  
  链接：https://www.zhihu.com/question/24779017/answer/28942972  
  来源：知乎  
  著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。
* ggplot2基于Leland Wilkinson在Grammar of Graphics（图形的语法）中提出的理论，取首字母缩写再加上plot，于是得名ggplot。按照《图形的语法》一书中的观点，一张统计图形就是从数据到点、线或方块等**几何对象**的颜色、形状或大小等**图形属性**的一个映射，其中还可能包含对数据进行**统计变换**（如求均值或方差），最后将这个映射绘制在一定的**坐标系**中就得到了我们需要的图形。图中可能还有**分面**，就是生成关于数据的不同子集的图形。使用ggplot2绘图的过程就是选择合适的几何对象、图形属性和统计变换来充分暴露数据中所含有的信息的过程。ggplot2需要一定的时间去入门学习，但是当你掌握了ggplot2中图形的语法的时候，我相信你会感受到这套语法的优雅。
* 在接下来的部分，我假定读者已经对R语言有了基本的了解，我将不会介绍DataFrame等基本概念。
* 安装ggplot2和安装其他的R包没有差别，在R的console里面运行install.packages("ggplot2") 一句就可以了，如果你使用RStudio，也可以在Package列表那里用鼠标去安装。
* 先来介绍一些ggplot2中的基本概念，括号里面对应的是ggplot2中为这种属性赋值的时候需要使用的参数名  
  **图形属性（aes）** 横纵坐标、点的大小、颜色，填充色等  
  **几何对象（geom\_）** 上面指定的图形属性需要呈现在一定的几何对象上才能被我们看到，这些承载图形属性的对象可能是点，可能是线，可能是bar  
  **统计变换 （stat\_）** 比如求均值，求方差等，当我们需要展示出某个变量的某种统计特征的时候，需要用到统计变换
* 我们来看下面这张图  
  &lt;img src="https://pic2.zhimg.com/0b075d86628a173b5bf5ada116f1fdad\_b.jpg" data-rawwidth="408" data-rawheight="404" class="content\_image" width="408"&gt;  
  这张图里面用到的图形属性是，横坐标x，纵坐标y，点的颜色，使用的几何对象是点，没有进行统计变换  
  &lt;img src="https://pic3.zhimg.com/3a235ee6c1e54143b57935a4e9b4f5be\_b.jpg" data-rawwidth="400" data-rawheight="404" class="content\_image" width="400"&gt;
* 下面这张图使用的图形属性是横坐标x，使用的几何对象是bar，使用的统计变换是封箱（bin），也就是分一点间距计数。 大家都知道，这张图其实就是直方图（histogram）。这种传统的统计图形也都是可以被容纳在ggplot2的图形语法框架之内的，当你熟练掌握ggplot2之后，可以根据图形语法自己构建新的统计图形。
* ggplot2对于ggplot()函数生成的ggplot对象重载了+这个运算符号，我们可以从一个不含有任何内容而只是指定了数据的ggplot2图形开始，逐步添加新的元素，最终得到一张内容丰富的图形。  
  假设我们这里已经有了一个名为mtcars的数据库，其中有两列wt和mpg  
  p <- ggplot(mtcars)+aes(x=wt,y=mpg)  
  我们首先指定要用mtcars这个dataframe来绘图，然后我们又指定了图中的横纵坐标分别表示什么，现在图上还是看不到东西的，只能看到坐标轴，我们要继续指定几何对象。p + geom\_point()  
  &lt;img src="https://pic1.zhimg.com/9a37cb3ba183f3ed60d4c0d441d9a084\_b.jpg" data-rawwidth="423" data-rawheight="410" class="origin\_image zh-lightbox-thumb" width="423" data-original="https://pic1.zhimg.com/9a37cb3ba183f3ed60d4c0d441d9a084\_r.jpg"&gt;这样我们就得到了一张最基本的散点图。这样我们就得到了一张最基本的散点图。  
  我们还可以按照另外一个变量对点上色  
  p + geom\_point(aes(colour = qsec))  
  &lt;img src="https://pic1.zhimg.com/6e1661937d0c34bc068a3b99f140b8d0\_b.jpg" data-rawwidth="408" data-rawheight="407" class="content\_image" width="408"&gt;这里使用的qsec是一个连续变量，我们还可以指定一个因子这里使用的qsec是一个连续变量，我们还可以指定一个因子  
  p + geom\_point(aes(colour = factor(cyl)))  
  &lt;img src="https://pic4.zhimg.com/03531241f457bc4baec3b9f5bea24403\_b.jpg" data-rawwidth="404" data-rawheight="402" class="content\_image" width="404"&gt;这样就生成了我们之前的那个例子
* 这样就生成了我们之前的那个例子
* 先写到这里 之后再慢慢更新吧
* 推荐两本书  
  一本是Hadley自己写的  
  ggplot2: Elegant Graphics for Data Analysis  
  另一本是 R Graphics Cookbook

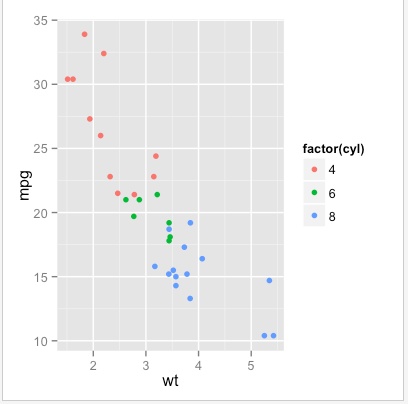
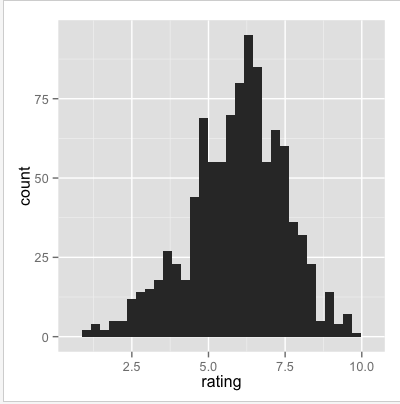
作者：顾志耐  
链接：https://www.zhihu.com/question/24779017/answer/28942972  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

ggplot2基于Leland Wilkinson在Grammar of Graphics（图形的语法）中提出的理论，取首字母缩写再加上plot，于是得名ggplot。按照《图形的语法》一书中的观点，一张统计图形就是从数据到点、线或方块等**几何对象**的颜色、形状或大小等**图形属性**的一个映射，其中还可能包含对数据进行**统计变换**（如求均值或方差），最后将这个映射绘制在一定的**坐标系**中就得到了我们需要的图形。图中可能还有**分面**，就是生成关于数据的不同子集的图形。使用ggplot2绘图的过程就是选择合适的几何对象、图形属性和统计变换来充分暴露数据中所含有的信息的过程。ggplot2需要一定的时间去入门学习，但是当你掌握了ggplot2中图形的语法的时候，我相信你会感受到这套语法的优雅。

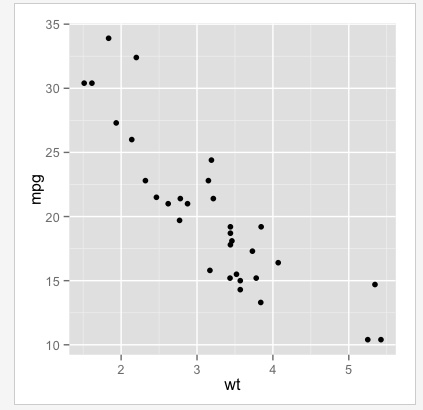
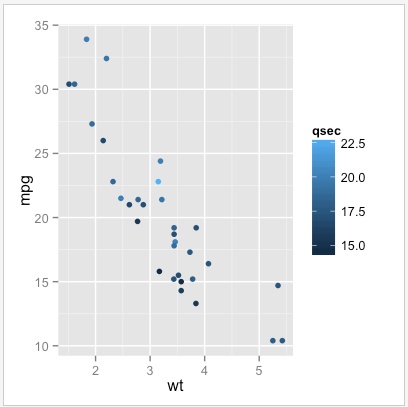
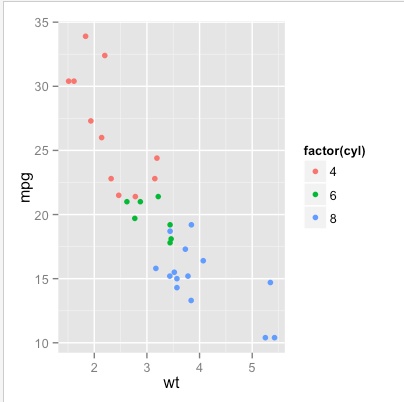
在接下来的部分，我假定读者已经对R语言有了基本的了解，我将不会介绍DataFrame等基本概念。

安装ggplot2和安装其他的R包没有差别，在R的console里面运行install.packages("ggplot2") 一句就可以了，如果你使用RStudio，也可以在Package列表那里用鼠标去安装。

先来介绍一些ggplot2中的基本概念，括号里面对应的是ggplot2中为这种属性赋值的时候需要使用的参数名  
**图形属性（aes）** 横纵坐标、点的大小、颜色，填充色等  
**几何对象（geom\_）** 上面指定的图形属性需要呈现在一定的几何对象上才能被我们看到，这些承载图形属性的对象可能是点，可能是线，可能是bar  
**统计变换 （stat\_）** 比如求均值，求方差等，当我们需要展示出某个变量的某种统计特征的时候，需要用到统计变换

我们来看下面这张图  
&lt;img src="https://pic2.zhimg.com/0b075d86628a173b5bf5ada116f1fdad\_b.jpg" data-rawwidth="408" data-rawheight="404" class="content\_image" width="408"&gt;  
这张图里面用到的图形属性是，横坐标x，纵坐标y，点的颜色，使用的几何对象是点，没有进行统计变换  
&lt;img src="https://pic3.zhimg.com/3a235ee6c1e54143b57935a4e9b4f5be\_b.jpg" data-rawwidth="400" data-rawheight="404" class="content\_image" width="400"&gt;

下面这张图使用的图形属性是横坐标x，使用的几何对象是bar，使用的统计变换是封箱（bin），也就是分一点间距计数。 大家都知道，这张图其实就是直方图（histogram）。这种传统的统计图形也都是可以被容纳在ggplot2的图形语法框架之内的，当你熟练掌握ggplot2之后，可以根据图形语法自己构建新的统计图形。

ggplot2对于ggplot()函数生成的ggplot对象重载了+这个运算符号，我们可以从一个不含有任何内容而只是指定了数据的ggplot2图形开始，逐步添加新的元素，最终得到一张内容丰富的图形。  
假设我们这里已经有了一个名为mtcars的数据库，其中有两列wt和mpg  
p <- ggplot(mtcars)+aes(x=wt,y=mpg)  
我们首先指定要用mtcars这个dataframe来绘图，然后我们又指定了图中的横纵坐标分别表示什么，现在图上还是看不到东西的，只能看到坐标轴，我们要继续指定几何对象。p + geom\_point()  
&lt;img src="https://pic1.zhimg.com/9a37cb3ba183f3ed60d4c0d441d9a084\_b.jpg" data-rawwidth="423" data-rawheight="410" class="origin\_image zh-lightbox-thumb" width="423" data-original="https://pic1.zhimg.com/9a37cb3ba183f3ed60d4c0d441d9a084\_r.jpg"&gt;这样我们就得到了一张最基本的散点图。这样我们就得到了一张最基本的散点图。  
我们还可以按照另外一个变量对点上色  
p + geom\_point(aes(colour = qsec))  
&lt;img src="https://pic1.zhimg.com/6e1661937d0c34bc068a3b99f140b8d0\_b.jpg" data-rawwidth="408" data-rawheight="407" class="content\_image" width="408"&gt;这里使用的qsec是一个连续变量，我们还可以指定一个因子这里使用的qsec是一个连续变量，我们还可以指定一个因子  
p + geom\_point(aes(colour = factor(cyl)))  
&lt;img src="https://pic4.zhimg.com/03531241f457bc4baec3b9f5bea24403\_b.jpg" data-rawwidth="404" data-rawheight="402" class="content\_image" width="404"&gt;这样就生成了我们之前的那个例子

这样就生成了我们之前的那个例子

先写到这里 之后再慢慢更新吧

推荐两本书  
一本是Hadley自己写的  
ggplot2: Elegant Graphics for Data Analysis  
另一本是 R Graphics Cookbook