六 qplot: 快速构建图形

在这一讲，我们要看看ggplot官方和很多文档一开始就引入的**qplot**函数。之所以这个系列的文章没有从**qplot**的开头，是因为笔者并不认为**qplot**是ggplot的简化版本，而且自**qplot**的介绍使得ggplot的特色无法得到展示，而且显得比一般的绘图库繁琐。

这一点不难理解，前面几讲已经描述了ggplot的大部分内容，可以看到ggplot将绘图这一工作的抽象本身就较一般的绘图库抽象，而**qplot**试图在一个函数以内提供与一般绘图库类似的接口无疑困难的。而事实也证明，**qplot**虽然与一般的绘图语法比较相似，但是选项无疑复杂得多。同时，不用太深入的尝试，如果没有ggplot背后的设计结构的概念，qplot经常很难达到用户意图的效果。

当然，我的意思不是**qplot**是一个设计上失败的函数，而是它很大程度上并不是很多用户初始想象的ggplot的简化版本。ggplot的作者认为**qplot**提供了与base程序包中plot函数类似的接口，可以方便方便从R的base绘图库迁移过来的用户。因此，在官方文档中，为了让R的base绘图库用户觉得友好，从qplot开始讲解了。然而笔者已经在本系列第一个文章中已经说过，如果ggplot仅仅是另外一个绘图库，大家完全没有必要来学习。因此，笔者没有选择从qplot作为ggplot的入手点。那么，问题是为什么现在我还要介绍一下qplot呢？答案是qplot是ggplot背后语法的一个简化实现，在很多时候可以达到快速构建图形的作用。这可能就是qplot的q[uick]的意义了吧！

为什么那么说呢？前面几讲中的ggplot的例子是经典的ggplot的绘图架构，图形是通过一层一层的叠加起来的，再加上scale等的调整。这样的结构是清晰的，完整的，统一的；但是不可避免的需要多敲一些字符。而一般的绘图库都试图减少打字的量，比如base绘图库就试图用一句话完成一个绘图，然后再进行微调。lattice干脆就都放到一句话里。这样的函数结构紧凑，同时也能获得挺强大的功能，而这种需求在ggplot中的实现就是qplot。qplot的定义如下：

**qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE,**

**geom = "auto", stat = list(NULL), position = list(NULL), xlim = c(NA,**

**NA), ylim = c(NA, NA), log = "", main = NULL,**

**xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), asp = NA)**

**Arguments**

**x**

**x values**

**y**

**y values**

**...**

**other aesthetics passed for each layer**

**data**

**data frame to use (optional). If not specified, will create one, extracting vectors from the current environment.**

**facets**

**faceting formula to use. Picks facet\_wrap or facet\_grid depending on whether the formula is one sided or two-sided**

**margins**

**whether or not margins will be displayed**

**geom**

**character vector specifying geom to use. Defaults to "point" if x and y are specified, and "histogram" if only x is specified.**

**stat**

**character vector specifying statistics to use**

**position**

**character vector giving position adjustment to use**

**xlim**

**limits for x axis**

**ylim**

**limits for y axis**

**log**

**which variables to log transform ("x", "y", or "xy")**

**main**

**character vector or expression for plot title**

**xlab**

**character vector or expression for x axis label**

**ylab**

**character vector or expression for y axis label**

**asp**

**the y/x aspect ratio**

看到这个定义，你应该知道为什么我之前说qplot是个紧凑版的ggplot，可以看到函数中将aes，facets，geom等等的内容一次性的作为参数放到了函数里面。

下面就是一个例子，前面两句ggplot的绘图，可以用一句qplot来替代，但是最后一张图却不能只用qplot完成，因为qplot中不能管理层结构。

**library(ggplot2)**

**p <- ggplot(mtcars,aes(wt, mpg, size=factor(gear)**

**, color=factor(cyl)))**

**p + geom\_point() + geom\_smooth(method='lm')**

**## Warning in qt((1 - level)/2, df): NaNs produced**

**## Warning in qt((1 - level)/2, df): NaNs produced**

**## Warning in qt((1 - level)/2, df): NaNs produced**



**qplot(wt,mpg,size=factor(gear),color=factor(cyl),data=mtcars,**

**geom=c('point','smooth'), method='lm')**

**## Warning in qt((1 - level)/2, df): NaNs produced**

**## Warning in qt((1 - level)/2, df): NaNs produced**

**## Warning in qt((1 - level)/2, df): NaNs produced**



**p + geom\_point() + geom\_smooth(method='lm',size=1)**



不过，容易看到qplot虽然紧凑，但是由于缺乏层定义，灵活性不如ggplot。同时，如果将大量的自定义选项加入进去，那么函数的可读性会比较糟糕，而且层次也不够分明。更好的做法可能是将qplot用于尝试性绘图，正规的代码使用ggplot。或者可以混用qplot和ggplot，这样可以用qplot先大概产生出一个样子，再用ggplot的其他函数微调。比如下面的例子，先用qplot绘制点图，再添加其他内容：

**qplot(wt,mpg,size=factor(gear),color=factor(cyl),data=mtcars,**

**geom='point') + geom\_smooth(method='lm',size=1) +**

**scale\_color\_brewer(palette='Set1')**



qplot的内容就介绍到这里。下面我们去探索一下ggplot里将数据分组展示的另外一种技术——facet吧！