**数据的类型：离散和连续变量**

这个话题似乎离ggplot的内容有点远，但是实际上由于离散变量和连续变量在图形中的表现差别很大，所以不得不在这里稍微提出来说一下这两种变量的区别和联系。

在R中，数值变量被定义为连续变量，而factor和字符被定义为离散变量。这种差异在绘图中是本质的，在后面讲到**scale\_\***函数时会更加明显。在这里，读者只需要知道这两类变量的处理很相当不同的。如果你知道一个变量是离散变量，最好在绘图的时候将这个变量转换为factor。而对于希望作为离散变量处理的连续变量可以使用ggplot中提供的函数**cut\_interval**或者**cut\_number**，两者的区别是一个是按照变量的取值来切分，一个是按照变量的个数来切分。当然，R的base里也有cut函数，可以完成类似的功能，读者可以根据个人喜好选择。

下面用柱状图的例子来展示，为什么说使用ggplot时，必须注意变量的类型。 例子中首先，生成一些1到7之间的随机整数**int**，然后对应到字符A到G上**char**。然后，可以看到不同的类型组合可以有以下四种不同的图形。

**int <- floor(runif(1000,1,7))**

**ch <- data.frame(char=letters[int],int)**

**ggplot(ch,aes(int, fill=char)) + geom\_bar()**

**## stat\_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.**

stat\_bin: binwidth defaulted to range/30. Use ‘binwidth = x’ to adjust this.

这一句中，**int**是连续变量，所以直方图以前需要先确定binwidth，这也就是R弹出警告的原因，正确的使用方法是设置一个合理的binwidth。默认的binwidth就是图中看到range/30。同时注意到，右侧的色条是离散变量char。

**ggplot(ch,aes(factor(int), fill=char)) + geom\_bar()**

这一次，将**int**转换成了离散变量，所以不存在设置binwidth的问题了。R会直接按照数字出现的次数绘制直方图。同时注意到，右侧的色条也是离散变量char。

**ggplot(ch,aes(char, fill=int)) + geom\_bar()**

这一张图的色条变成了连续变量**int**，所以色条由原先的离散颜色变为了连续的过渡。由于x数据是离散变量所以也不会提示需要设置binwidth。

**ggplot(ch,aes(char, fill=factor(int))) + geom\_bar()**

这张图将前一张图中色条由连续变量换成了离散变量，由此色条也变成了离散颜色。

在本篇的后面我们将看到离散变量还有另外一个重要用途就是对数据进行分组。这也就是上一讲里**aes**放到不同的位置出现不同的图像的原因。不过在此之前我们需要先介绍一下ggplot中层的概念。

**层：灵活性的源泉**

层是一个非常有意思的概念，在无数的架构中都会引入层的概念。因为本质上讲，层的概念是将数据结构扩展了一个维度，所以相应的会给数据结构带来大量的灵活性，而且层的概念比较容易理解，所以在诸多架构中都得到使用就很不奇怪了。那么在ggplot里面什么是层呢？

在ggplot里，所有的图形构件都是绘制在层上的。这也是为什么如果单运行ggplot的命令，你会遇到**Error: No layers in plot**的错误。那层是什么呢？

【提示】：以下内容不是官方解释，是笔者的自我感觉，可能不甚准确，欢迎批评指正。

层是容纳一个**geom**的容器，每个**geom**需要一个层。为什么要把**geom**各放到一个层呢？最重要的原因是不同的层可以有不同的**aes**和不同的**data**。

大概的联系是这样的，一个ggplot的图形首先会从ggplot函数中继承默认的aes和data。如果层内没有指定这些性质的话，就会从默认的设置中提取对应值。而如果层内给定了值的话，在这一层里这些aes和data将覆盖默认值。

这里有一个性质是如果将离散变量定义给出了x,y数据的其他属性的话，相应的层会根据这个离散变量进行分组。比如前面的那个例子：

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

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

这一讲，由于载入了默认的mtcars，cyl和gear都是连续变量，所以在ggplot中设置的属性虽然传递到了其他层，但是连续变量不会分组，所以得到了一个回归曲线。

而下面的代码,cyl被转换成了离散变量，所以回归曲线会按照cyl的值分组。

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

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

当然，也可以将gear也转换为离散变量，这样就会按照两个变量进行分组。所以下面的图中的右侧标识出现了相应的变化。但是奇怪的是为什么回归曲线还是三条呢？不是应该是九条吗？

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

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

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

如果你想到前面说的属性设置的覆盖性，你就知道因为回归线的设置中设置的size，所以在回归线中size的分组被取消了。真的是这样吗？

**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**

注意到有三个情况没有数据点来进行回归，所以只有六根回归线。

我想，到这里读者应该已经明白了每个geom会产生一个层，而层中的几何性质会根据属性的类型而不同。对于没有设置的属性，ggplot会到ggplot函数的定义中查找，如果没有的话就会使用默认值。

文章的最后，如果你想看看ggplot里面的层是什么样子的，可以用**str**函数去查看。

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

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

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

**print(str(p))**

**## List of 9**

**## $ data :'data.frame': 32 obs. of 11 variables:**

**## ..$ mpg : num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...**

**## ..$ cyl : num [1:32] 6 6 4 6 8 6 8 4 4 6 ...**

**## ..$ disp: num [1:32] 160 160 108 258 360 ...**

**## ..$ hp : num [1:32] 110 110 93 110 175 105 245 62 95 123 ...**

**## ..$ drat: num [1:32] 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...**

**## ..$ wt : num [1:32] 2.62 2.88 2.32 3.21 3.44 ...**

**## ..$ qsec: num [1:32] 16.5 17 18.6 19.4 17 ...**

**## ..$ vs : num [1:32] 0 0 1 1 0 1 0 1 1 1 ...**

**## ..$ am : num [1:32] 1 1 1 0 0 0 0 0 0 0 ...**

**## ..$ gear: num [1:32] 4 4 4 3 3 3 3 4 4 4 ...**

**## ..$ carb: num [1:32] 4 4 1 1 2 1 4 2 2 4 ...**

**## $ layers :List of 2**

**## ..$ :Classes 'proto', 'environment' <environment: 0x7f9fad1e1840>**

**## ..$ :Classes 'proto', 'environment' <environment: 0x7f9fad239808>**

**## $ scales :Reference class 'Scales' [package "ggplot2"] with 1 fields**

**## ..$ scales: list()**

**## ..and 21 methods, of which 9 are possibly relevant:**

**## .. add, clone, find, get\_scales, has\_scale, initialize, input, n,**

**## .. non\_position\_scales**

**## $ mapping :List of 4**

**## ..$ x : symbol wt**

**## ..$ y : symbol mpg**

**## ..$ size : language factor(gear)**

**## ..$ colour: language factor(cyl)**

**## $ theme : list()**

**## $ coordinates:List of 1**

**## ..$ limits:List of 2**

**## .. ..$ x: NULL**

**## .. ..$ y: NULL**

**## ..- attr(\*, "class")= chr [1:2] "cartesian" "coord"**

**## $ facet :List of 1**

**## ..$ shrink: logi TRUE**

**## ..- attr(\*, "class")= chr [1:2] "null" "facet"**

**## $ plot\_env :<environment: R\_GlobalEnv>**

**## $ labels :List of 4**

**## ..$ x : chr "wt"**

**## ..$ y : chr "mpg"**

**## ..$ size : chr "factor(gear)"**

**## ..$ colour: chr "factor(cyl)"**

**## - attr(\*, "class")= chr [1:2] "gg" "ggplot"**

**## NULL**

注意到**layers**变量是长度为2的列表，正好是这里定义的两层，一个**point**一个**smooth**。还可以看到**layers**下面有一个叫**scales**的东西，这个是干什么的呢？这个就是下一讲的内容，通过调整scales你就可以控制图形中点的大小范围，线的粗细范围等等。