ggplot2

yphuang

*2015*年*10*月*30*日

R数据可视化之ggplot2

一、动机

基于shiny的ggplot2绘图演示

```
library(shiny)
source("Think_ggplot2_shinyDeom/helper.R")

shinyAppDir(appDir = "Think_ggplot2_shinyDeom", options = list(
  width="100%", height=550
))
```

ggplot2的优势

- 绘图便利而精美
  - ggplot2的默认设置已经可以使你的图形很精美，精美到甩plot几条大街。
  - 基础绘图系统中的plot()函数虽然已经很强大，但是还不够便利。因此，ggplot2有了模仿plot()设计的qplot()函数。
- 图层的概念利于结构化思维
  - 基础绘图系统中，有两类绘图函数，低级函数和高级绘图函数。高级绘图函数绘制的是一个完整的绘图函数，低级绘图函数在已有的图形对象中进行添加，相当于ggplot2中的图层叠加概念，但是肯定没有ggplot2的图层概念强大。假设你要用plot绘制一个类似分面的图形，那么你可能需要n个可重复的步骤，然后通过par()中的mfrow参数来整合到一块儿。
- 函数和参数设置方便记忆
  - 这一点非常重要。你不需要像SAS绘图系统那样牢记几十上百个参数就可以绘制比SAS绘图系统更漂亮的图形。在R绘图设备中，每一个不同的图都有一个独特的函数比如boxplot()，hist()，barplot()等；然而在qplot()函数中，用geom参数取不同的值即可画不同的图，而geom_+图形对象名即可绘制不同的图。结合Rstudio开发环境下的参数提示、函数提示和自动补全功能，再也不用担心复杂的参数和函数记忆问题了。
- 在更抽象的层面上控制图形
- ggplot的代码可重用性更强，并且可以很方便地定制化一些带有你自己特色的图形。

二、语法直观认识

几个重要的语法概念

- 数据(**data**)和映射(**mapping**)
  - 将数据中的变量值映射到图形属性中
- 图形属性(**aes** thetic)
  - 几何对象的视觉属性，设定**x**轴，**y**轴对应变量，线条颜色，点的形状等
- 标度(**scale**)
  - 对应坐标和刻度
- 几何对象(**geom** etric)
  - 实际看到的图形元素：点，线，面
  - 对应一个默认的统计变换
- 统计变换(**stat** istic)
  - 对原始数据进行统计计算
  - 对应一个默认的几何对象
- 引导元素(**guide**)
  - 如：刻度线和标签
- 坐标系统(**coord** inate)
  - 坐标轴位置变换，标度变换
- 图层(**layer**)
- 分面(**facet**)

简单说明

一个图形对象就是一个包含数据，映射，图层，标度，坐标和分面的列表，外加组件options。

每一个元素都是一个对象，可通过重载的‘**+**’叠加。这个特点的神奇之处，在于我们可以非常方便的逐步调整和修饰我们的图形。

先来个示例代码感受一下：

```
library(ggplot2)
  scatter<-ggplot(mtcars, aes(x = wt, y = mpg, color = as.factor(cyl), shape = as.factor(cy
l)))+
  geom_point()+
  geom_smooth(method = lm, se = FALSE, fullrange = TRUE)+
  labs(title="Miles per gallon \n according to the weight",
x="Weight (lb/1000)", y = "Miles/(US) gallon")+
  theme_classic()+
  scale_color_brewer(palette = "Accent")+
  theme_minimal()
```

以上的命令生成了示例中的散点图。其中，每一个“**+**”前面的部分都是独立的。

- 分面和图层将原数据切割称多个小数据集，即每个图层的每个分面面板都含有一个小数据集。

- 你可以把它想象成一个三维矩阵：分面面板形成了一个2维网格，图层在第三维的方向上叠加。
- 分面这个概念使得分组对比图的绘制非常方便，而图层的概念使得我们用R绘图跟艺术家绘图别无二致。
- gglot()函数如同开启了一块画布，通过“+”叠加的几何对象：简称geom，控制生成的图像类型和图层叠加，如geom_line(),geom_point(),geom_smooth()。
- 图层的叠加，也就相当于是层层绘制，把同一来源的数据通过不同的图形类型展示在同一幅图中，或者把不同来源的数据很方便的展示在同一个图形对象中。
- 也可以理解成每一块画布是透明的，通过画布的叠加来丰富我们的图形对象。

三、快速入门——qplot

qplot(),是quick plot 的缩写,顾名思义，就是让你快速的绘制一幅图。Hadley建议我们，如果想学好R中的函数封装，应该认真阅读qplot的源码，理解它的工作机理。

先来看一下qplot()的参数。

```
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)
```

这个函数的参数可谓是，麻雀虽小，五脏俱全。

来个例子。

```
#lm regression
qplot(mpg, wt, data = mtcars, geom = c("point", "smooth"),
method = "lm")
```

要在基础绘图中实现这个图形，需要两步，而这里只需要一步。

再看一个例子。

```
qplot(cty,hwy,data = mpg)+facet_grid(.~cyl)
```

这个图，用基础绘图可能就不是一步两步就能搞定了。我后面推荐的几本书都有很多的使用示例，这里就不多赘述了。

四、绘图流程

4.1 数据准备

说到数据准备，就不得不提同为Hadley开发两个专用于数据清洗的包：dplyr和reshape2.这两个包也可以说是为ggplot2而生。三者结合起来，简直妙不可言。代码可以写得非常优雅，图形也会更有表现力。他们都是基于data.frame的数据结构之上。

dplyr理念与操作示例

- 理念
    - 时间是宝贵的。关键的部分由Rcpp写成。
    - 本地能做，远程也理应能做。远程数据库接口支持： PostgreSQL, MySQL, SQLite 。
    - 每个函数只做一件事，并把它做好。group_by,summarise,mutate,filter,select,arrange等等这些函数组合起来，几乎可以胜任所有日常的数据清洗工作.
    - 另外，我觉得最重要的是管道操作的功能。操作符'%>%'可以把前一步处理的结果当作第一个参数传给下一个处理函数。这个操作让你的代码更具有可读性，处理数据的思路过程也更清晰。
- 举个例子

```
library(dplyr)
library(nycflights13)
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarise(
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30)
```

由于这里的主角是ggplot2，其它的一些使用细节就不多谈了，详细使用请参考给出的链接。

- 参考：
    - 理念介绍http://blog.rstudio.org/2014/01/17/introducing-dplyr/ (http://blog.rstudio.org/2014/01/17/introducing-dplyr/)
    - 操作介绍https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html (https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html)

---

reshape2

如果接触过数据库的同学，对“长表”和“宽表”应该有所理解。而reshape2就是用于这二者的转换。reshape2目前我只用过两个函数：melt()和dcast().这两个函数用来“糅数据”简直无敌。随便你怎么在宽表和窄表之间转换。

- 举一个例子。

```
library(dplyr)
library(reshape2)
library(ggplot2)
View(anscombe)
anscombe.new<-mutate(anscombe, n = seq(1:nrow(anscombe)))
str(anscombe.new)
#melt
narrow.data<-melt(anscombe.new, id.vars = ("n"))
head(narrow.data)

narrow.data2<-mutate(narrow.data,
                     group = substr(variable,2,2),
                     variable = substr(variable,1,1))
#dcast
narrow.data2<-dcast(narrow.data2, n + group~variable)
View(narrow.data2)

#plot
ggplot(narrow.data2, aes(x = x, y = y, colour = group))+
  geom_point()+
  geom_smooth(method = "lm")+
  facet_grid(.~group)
```

- 深入理解：
  - 使用示例http://seananderson.ca/2013/10/19/reshape.html (http://seananderson.ca/2013/10/19/reshape.html)

4.2 创建图形对象

如下所示，用ggplot()函数创建一个图形对象。

```
library(ggplot2)
p<-ggplot(data = diamonds, aes(x = carat, y = price, colour = cut))
```

参数映射，只需要把需要映射到图形属性的变量名放到aes()函数内即可。

4.3 添加图层

可以使用layer()函数添加图层。如绘制以carat为x轴，price为y轴的散点图。

```
p1<-p+layer(geom = "point")
```

对于每一个图层，我们只需要设定stat或者geom参数即可。这是因为每一个几何对象都对应一个默认的统计变换和位置参数，而每一个统计变换都对应着一个默认的几何参数对象。

```
p2<-p+layer(stat = "identity")
```

也可以使用快捷函数来简化以上代码。

```
p3<-p+geom_point()
```

以上三种方式都可以生成同一幅图形。

4.4 数据

ggplot2需要传入一个data.frame格式的数据。数据的前期准备前面已有介绍。而数据在图形图像中是以副本的方式保存的，这说明我们可以保存图形对象（ggsave()）以待下次加载(load())。

4.5 图形属性映射aes()

aes()用来将数据变量映射到图形中。 如aes(x ,y,colour=z)； 又如aes(x,y,group = z)——将数据分成若干组，并用相同的方式对每个组进行渲染。值得一提的是，当现有的单个变量不能够正确的分组而两个变量的组合可以正确分组时，可以使用interaction()函数进行分组。

4.6 几何对象

几何对象：简称geom，控制生成的图像类型，如geom_line(),geom_point(),geom_bar().

4.7 统计变换

统计变换：stat。它通常以某种方式对数据信息进行汇总。ggplot2中包含的统计变换有bin,density等。

统计变换可以向原数据集中插入新的变量。示例。如

```
#geom
hist1<-ggplot(diamonds, aes(carat))+
  geom_histogram(aes(y = ..density..),binwidth = 0.1)

#或者stat
hist2<-ggplot(diamonds,aes(carat))+
  layer(stat = "bin", binwidth = 0.1)
```

以上统计变换生成了一些新的变量count,density,x。我们可以在aes()中调用这些变量。

4.8 位置调整

位置调整。位置调整一般多见于处理离散性数据。如条形图中的：堆叠(stack)，填充(fill)，并列(dodge)。 此外，还有jitter:给点添加扰动避免重合；identity:不做任何调整。

如下所示：

```
library(ggplot2)
library(gridExtra)
hist1<-ggplot(data = diamonds, aes(x = clarity, fill = cut))+
  geom_histogram(position = "dodge")

hist2<-ggplot(data = diamonds, aes(x = clarity, fill = cut))+
  geom_histogram(position = "fill")

hist3<-ggplot(data = diamonds, aes(x = clarity, fill = cut))+
  geom_histogram(position = "stack")

grid.arrange(hist1,hist2,hist3, ncol = 3, nrow = 1)
```

4.9 对象整合

> 几何对象+统计变换.

示例：直方图的几种变体.

```
d<-ggplot(data = diamonds, aes(x = carat))
d1<-d+
  stat_bin(aes(ymax = ..count..), binwidth = 0.1,geom = "area")

d2<-d+
  stat_bin(aes(size = ..density..),binwidth = 0.1, geom = "point",position = "identity")

grid.arrange(d1, d2, ncol = 2, nrow = 1)
```

> 将不同来源的数据画在同一张图

举例:

```
#例子来源于stackoverflow
df1<-data.frame(x=1:10,y=rnorm(10))
df2<-data.frame(x=1:10,y=rnorm(10))

ggplot(df1,aes(x,y))+geom_line(aes(color="First line"))+
  geom_line(data=df2,aes(color="Second line"))+
  labs(color="Legend text")
```

4.10 图片保存

> 输出图片类型：矢量型 or 光栅型

- 矢量型：存储指令，图形可无限缩放而没有细节的损失；

- 光栅型：以像素阵列存储，具有固定的最优观测大小。

两种操作方式

- 通过打开图形设备：与基础绘图系统的图片保存相同。
- 通过ggsave()。

示例1：输出PDF矢量文件

```
#方式1:

##打开PDF图形设备
pdf(file = "testPlot.pdf", width = 4, height = 4)

##绘制图形
print(ggplot(mtcars,aes(x = wt, y = mpg))+geom_point())
plot(mtcars$wt, mtcars$mpg)
## 关闭图形设备
dev.off()
```

```
#方式2:

##绘图
testPlot2<-ggplot(mtcars,aes(x = wt, y = mpg))+geom_point()
##保存图片
ggsave(filename = "testPlot2.pdf", plot = testPlot2, width = 8, height = 8, units = "cm")
```

示例2：输出点阵(PNG)文件

```
#方式1:
##注：width和height为像素大小
png(file = "img/testPlot-%d.png",width = 400,height = 400)
##绘图
print(ggplot(mtcars,aes(x = wt, y = mpg))+geom_point())
plot(mtcars$wt, mtcars$mpg)
##
dev.off()

#方式2:

##绘图
testPlot2<-ggplot(mtcars,aes(x = wt, y = mpg))+geom_point()
##保存图片
###注：ppi参数为分辨率
ggsave(filename = "img/testPlot2.png", plot = testPlot2, width = 8, height = 8, units = "cm",dp
i = 300)
```

五、策略调整——让图形更有表现力

5.1 图层叠加策略

目的

1. 展示数据本身；
2. 展示数据的统计摘要；
3. 添加额外的元数据、上下文信息和注解，或者背景图层，如地图。

示例1：添加回归模型拟合线

```
##基本绘图对象
lm.plot<-ggplot(mtcars,aes(x = wt, y = mpg))

##叠加图层。level为置信水平，se为是否显示置信域
lm.plot + geom_point() + stat_smooth(method = lm, level = 0.99, se = TRUE)
```

示例2：用geom_text()向条形图添加数值标签

```
##为注解产生数据
library(dplyr)
my_diamonds<-group_by(diamonds,color) %>%
  summarise(label_y = n())

##绘图
mybar<-ggplot(data = diamonds, aes(x = color))+
  geom_bar(fill = "cornsilk",colour = "black")

##添加数值标签
mybar + geom_text(data = my_diamonds,aes(x = color,y = label_y,label = label_y), colour = "re
d",vjust = -0.2)
```

示例3：用annotate()添加说明

```
##绘制正态分布密度函数
p<-ggplot(data.frame(x = c(-3,3)),aes(x = x)) +
  stat_function(fun = dnorm)

##添加注解
p+annotate("text",x = 2, y = 0.3, parse = TRUE,
           label = "frac(1, sqrt(2*pi))^e^{-x^2/2}",size = 6)
```

5.2 处理遮盖

策略

1. 绘制更小的点(shape =)
2. 调整透明度(alpha =)
3. 增加扰动(geom_jitter)
4. 使用二维核密度估计(geom_density2d)
5. 使用数据分箱(bin),调整填充色

示例

```
##简单绘图
sp<-ggplot(diamonds,aes(x = carat, y = price))
sp1<-sp+geom_point()
##方式1：调整透明度
sp2<-sp+geom_point(alpha = 0.01)
##方式2：数据分箱
sp + stat_bin2d()
##调整填充色
sp + stat_bin2d(bins = 50)+
  scale_fill_gradient(low = "lightblue", high = "red", limits = c(0,6000))
```

5.3 统计摘要

对于每个x的取值,计算并展示对应y值的统计摘要，如均值，中位数，最小值，最大值等。

```
##简单绘图
p1<-ggplot(data = mtcars,aes(x = cyl, y = mpg))+geom_point()
##绘制每组的均值
p1 + stat_summary(fun.y = mean, geom = "point",colour = "red")
```

5.4 标度

类别

- 位置标度：将变量映射到绘图区域，以及构造对应的坐标轴
- 颜色标度：将变量映射至颜色
- 手动离散性标度：将离散型变量映射至我们选择的符号大小，线条类型，形状等
- 同一型标度：直接将变量值映射至图形属性

示例1：位置标度调整

```
qplot(log10(carat),log10(price),data = diamonds)
##对比
qplot(carat,price,data = diamonds)+
  scale_x_log10()+scale_y_log10()
```

示例2：手动离散型标度调整

```
ggplot(data = diamonds, aes(x = color,group = cut))+
  geom_bar(aes(linetype = cut,colour = cut),size = 1,position = "dodge",fill = "white")+
  facet_grid(.~cut)+
  theme_classic()+
  scale_linetype_manual(values=c("twodash", "solid","F1","dotted","dotdash"))
```

5.5 分面

在一个页面上摆放多幅图形。

两个常用基本参数：

- 分面变量的设置；
- “scales =”。

网格分面(facet_grid)

```
qplot(cty,data = mpg,geom = "histogram", binwidth = 2)+
  facet_grid(cyl~.)

##例2
ggplot(data = diamonds, aes(x = carat, y = price))+
  geom_point(aes(colour = color))+
  facet_grid(color~.)
```

封装分面(facet_wrap)

先生成一个长的面板条块，然后将它封装在2维中。适合处理单个多水平变量。

```
library(reshape2)
library(ggplot2)
sp <- ggplot(tips, aes(x=total_bill, y=tip/total_bill)) + geom_point(shape=1)
sp + facet_wrap( ~ day, ncol=2)
##对比
sp + facet_grid(.~day)
```

六、深度加工——专业化主题定制

6.1 标题

- 添加标题：ggtitle()
- 标题加粗，更换字体：theme(plot.title =)

示例

```
##基本绘图
library(gcookbook)
p<-ggplot(heightweight,aes(x = ageYear,y = heightIn))+
  geom_point()

##添加、修饰标题
p+ggtitle("Age and Height")+
  theme(plot.title = element_text(size = rel(1.5),lineheight = .9,family = "Times",
                                  face = "bold.italic",colour = "red"))
```

6.2 坐标轴

- 更改默认坐标轴文字标签及数值范围： labs(),xlab(),ylab(),xlim(),ylim()
- 坐标轴加粗，更换字体：theme(axis.title.x =)

示例

```
p <- qplot(mpg, wt, data = mtcars)
p + labs(title = "New plot title")
p + labs(x = "New x label")
p + xlab("New x label")
p + ylab("New y label")+ylim(2, 4)
```

6.3 主题模板

使用主题

```
##基本绘图
library(gcookbook)
p<-ggplot(heightweight,aes(x = ageYear,y = heightIn))+
  geom_point()

##默认主题
p+theme_gray()

##黑白主题
p+theme_bw()

##经典主题
p+theme_classic()
```

创建自定义主题

```
##创建自定义主题
mytheme<-theme_bw()+
  theme(text = element_text(colour = "red"),
        axis.title = element_text(size = rel(1.25)),
        panel.grid.major.x = element_blank())

library(gcookbook)
p<-ggplot(heightweight,aes(x = ageYear,y = heightIn))+
  geom_point()

##使用自定义注主题
p+mytheme
```

使用拓展包:library(ggthemes)

```
##示例1
p <- ggplot(data=mpg, mapping=aes(x=cty,y=hwy))
p + geom_point(aes(color=factor(year)),
               alpha=0.5,position = "jitter")+
  stat_smooth(method = "lm")+
  theme_economist() + scale_color_economist()+
  labs(title = '汽车型号与油耗',
       y = '每加仑高速公路行驶距离',
       x = '每加仑城市公路行驶距离',
       colour = '年份')

##示例2
ggplot(mtcars,aes(x=mpg,y=wt,size=cyl,colour=factor(gear)))+
  geom_point()+
  scale_size_area()+
  theme_solarized(light=FALSE) +
  scale_colour_solarized("red")+
  ggtitle("Motor Trend Car Road Tests")
```

6.4 着色调整

- 离散性变量：scale_color_brewer()，scale_colour_manual(),scale_fill_manual();
- 连续型变量：scale_color_gradient(),scale_color_gradient2()

示例1：针对离散型变量

```
##对离散型变量使用不同的调色板
library(gcookbook)
p<-ggplot(uspopage, aes(x = Year, y = Thousands, fill = AgeGroup))+geom_area()
p

##使用ColorBrewer调色板
p+scale_fill_brewer()

##进一步调整颜色
p+scale_fill_brewer(palette = "Oranges")
```

示例2：针对连续型变量

```
library(gcookbook)
p<-ggplot(heightweight,aes(x = ageYear,y = heightIn,colour = weightLb))+
  geom_point()
p
##使用两种颜色的渐变色
p+scale_colour_gradient(low = "black", high = "white")

##使用几个颜色的渐变色
p+scale_colour_gradientn(colours = c("darkred","orange","yellow","white"))
```

6.5 图例调整

- 移除图例
- 修改图例位置
- 移除图例标题
- 修改图例标签

```
####图例调整
library(gcookbook)
p<-ggplot(PlantGrowth,aes(x = group,y = weight,fill = group))+
  geom_boxplot()
p
##使用guides移除图例
p+guides(fill = FALSE)

##修改图例位置
p+theme(legend.position = "top")

##移除图例标题
p+guides(fill = guide_legend(title = NULL))

## 修改图例标签
p+scale_fill_discrete(labels = c("control 1","treatment 1","treatment 2"))
```

七、Next Generation——ggvis

ggvis介绍：

- http://blog.rstudio.org/2014/06/23/introducing-ggvis/ (http://blog.rstudio.org/2014/06/23/introducing-ggvis/)
- http://ggvis.rstudio.com/ (http://ggvis.rstudio.com/)

八、学习资料推荐

- 三本书：
    - ggplot2-数据分析与图形艺术

    - ggplot2_The_Elements_for_Elegant_Data_Visulization_in_R
    - R数据可视化手册
- 两篇cheatsheet总结：
    - ggplot2-cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf)
    - Beautiful plotting in R_ A ggplot2 cheatsheet _ Technical Tidbits From Spatial Analysis & Data Science (http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/)
    -
- 一个博客系列地址：
    - http://xccds1977.blogspot.jp/search/label/ggplot2 (http://xccds1977.blogspot.jp/search/label/ggplot2)