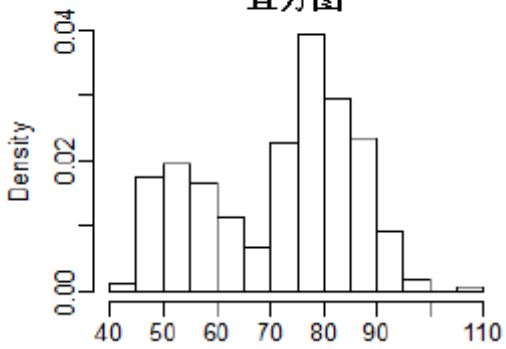
R语言数据可视化漫谈

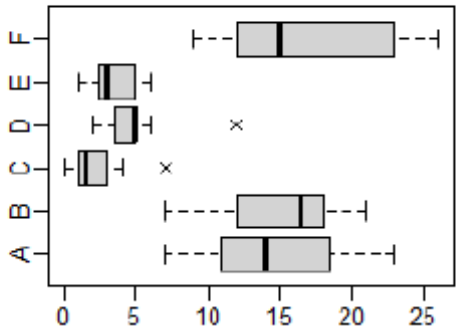
2017.11.19

数据可视化之美

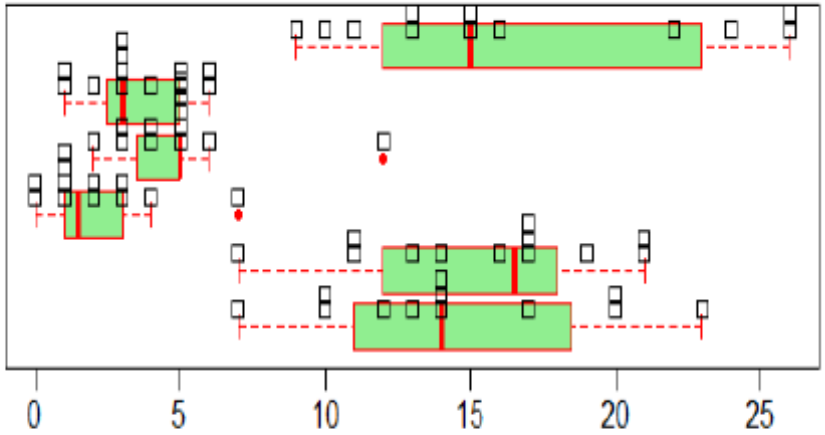
直方图



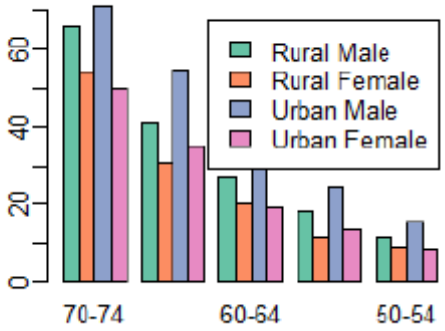
箱线图



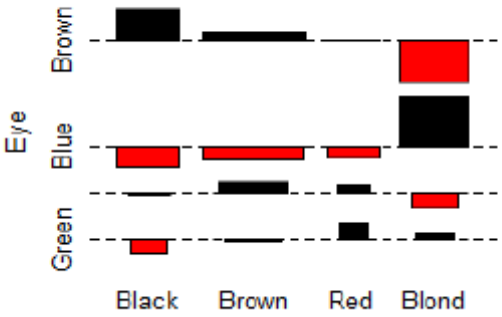
带状图



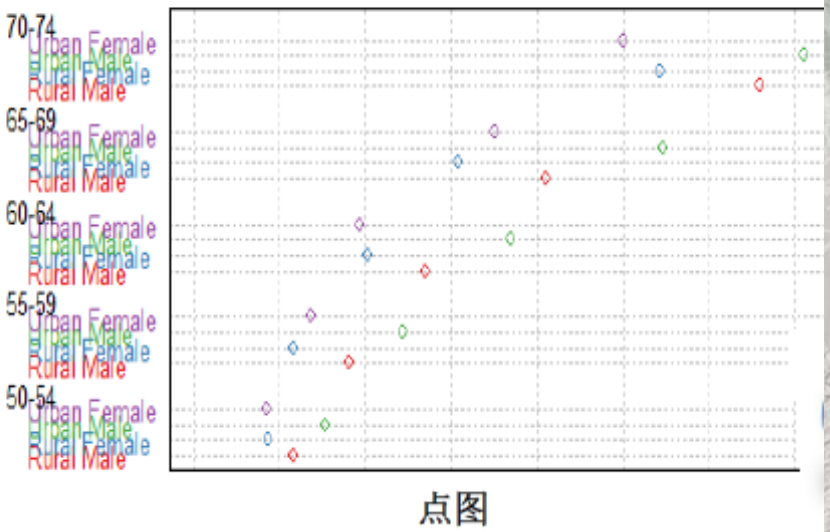
条形图



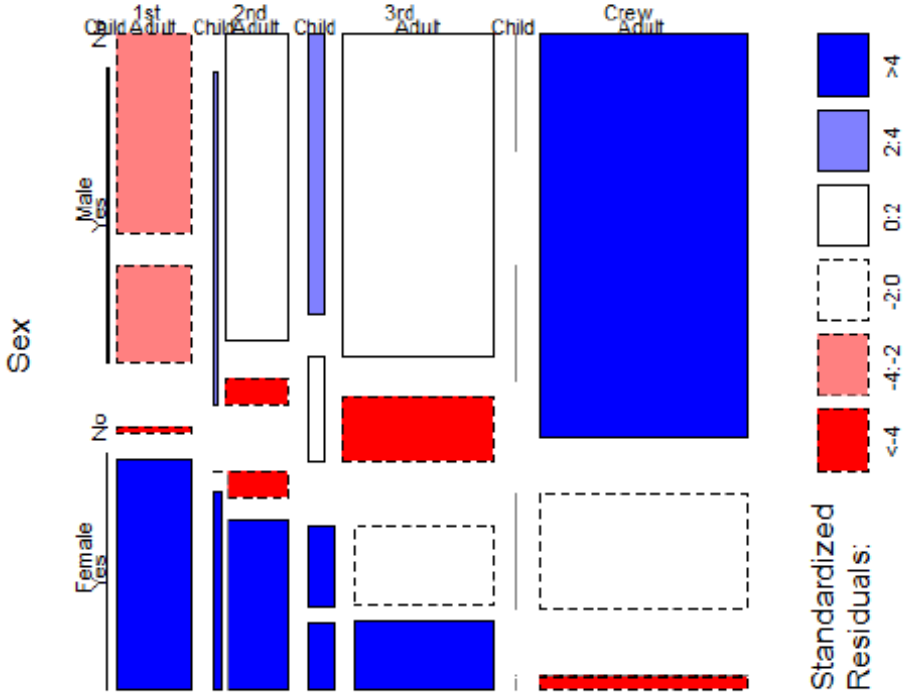
关联图

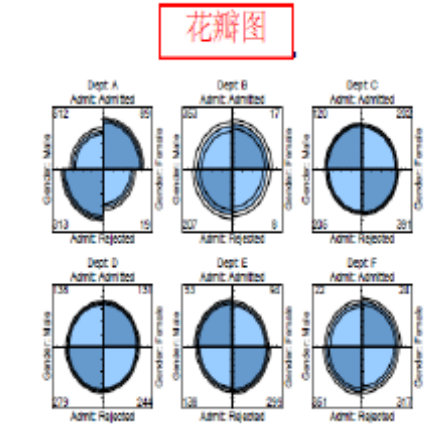


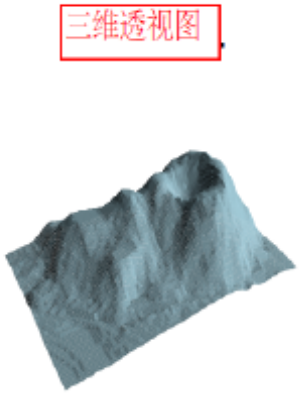
点图

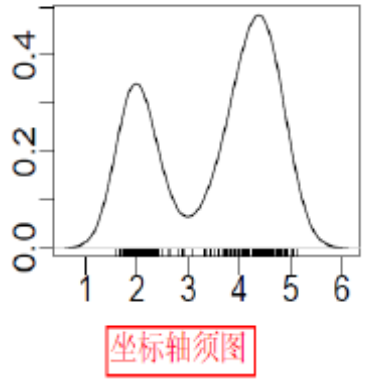


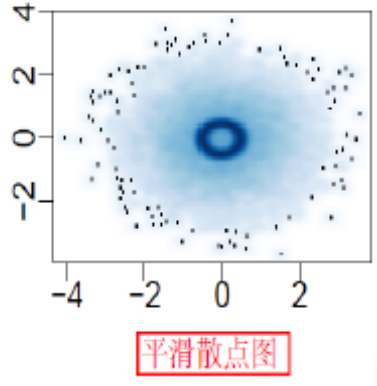
数据可视化 之美


马赛克图 of titanic
1st
2nd
3rd
Crew
Child
Adult
Sex
Male
Female
Yes
No
Standardized Residuals:
>4
2:4
0:2
-2:0
-4:-2
<-4


花瓣图
Dept A
Dept B
Dept C
Dept D
Dept E
Dept F
Admit: Admitted
Admit: Rejected
Gender: Male
Gender: Female


三维透视图


0.0
0.2
0.4
1
2
3
4
5
6
坐标轴须图


-4
-2
0
2
4
平滑散点图

低级绘图函数：

hist(x$x1)# 对科目1绘制直方图，横坐标表示分数区间，纵坐标表示频次。

plot(x$x1,x$x2)# 对科目1和科目2绘制散点图，大致看出两门课是否存在相关关系。

table(x$x2)#列联函数table，对每个数据进行统计。

barplot(table(x$x2))# 柱状图绘制函数barplot对统计结果进行绘制。barplot必须和table函数结合使用才有意义。

pie(table(x$x2))# 饼图函数pie。

boxplot(x$x1,x$x2,x$x3)# 箱线图boxplot，对三门科目画箱线图。

boxplot(x[2:4],col=c('**red**','**green**','**blue**'))# 指定箱线图的颜色。

boxplot(x[2:4],col=c('**red**','**green**','**blue**'),horizontal=T)# 使用horizontal=T将箱线图水平放置。

stars(x[2:4])# 对每个人画雷达图。

stars(x[2:4],draw.segments=T)# 对雷达图进行颜色和样式的修改。使用draw.segments=T画扇形。

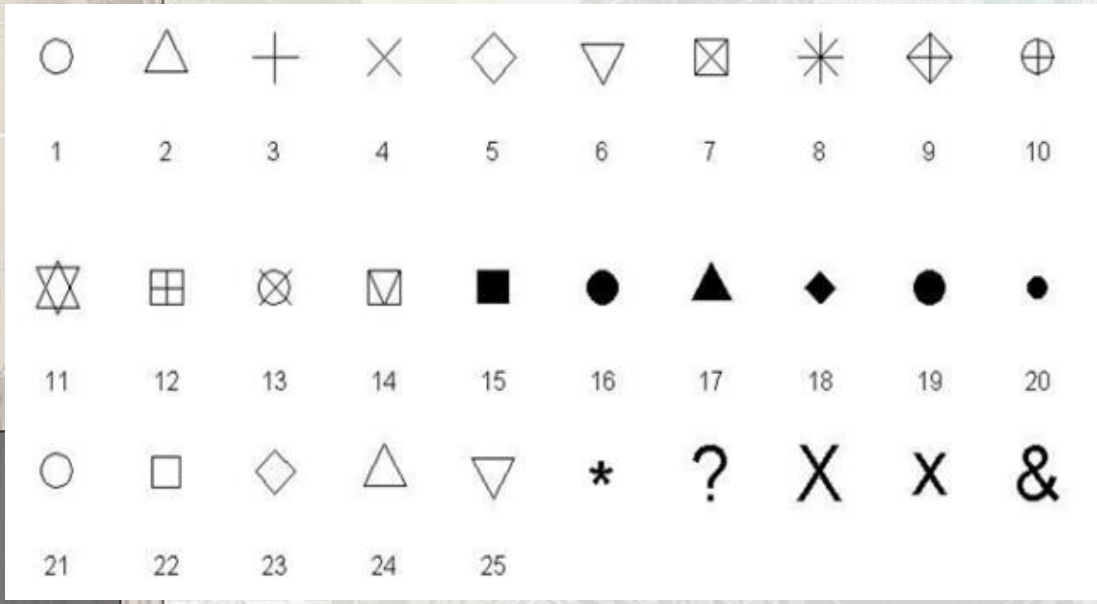
stem(x$x1)# 茎叶图

#此外，还有多种高级绘图函数points，lines，abline，title，text，axis，image，box，contour，rect，arrows，par等。

用plot绘制散点图的参数设置

```
plot(x$x1,x$x2,
    main='科目1与科目2的关系', #设置标题
    xlab='科目1',#设置横坐标名称，如果不写则默认为该变量的名称
    ylab='科目2',#设置纵坐标名称
    xlim=c(50,100),#设置横坐标的范围
    ylim=c(50,100),#设置纵坐标的范围
    xaxs='r',#xaxs='r', yaxs='i':分别设定 x 和y 轴的形式。 "i"(内部)与 "r"(预设值) 形式的刻度都会依照资料的范围而自动调整,但是 "r" 形式的刻度会在刻度范围两边留一些空隙。
    yaxs='r',
    col='red',#设置点的颜色
    pch=20 #设置画图的样式，20表示为圆点
    )
```

关于点的样式，上面显示用的是编号为20的样式，那么R其实提供了很多样式供使用，如下图。参考R语言绘图符号

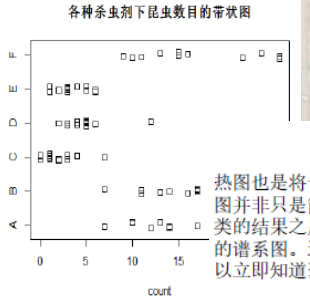


高级绘图函数：

带状图:stripchart( )

带状图（Strip Chart），又叫一维散点图（1-D Scatter Plot），是针对一维数据的“散点图”，它本质上是数据与固定值（固定x或固定y）之间的散点图

```
data(InsectSprays);head(InsectSprays);stripchart(count ~ spray, data = InsectSprays, method = "jitter",main="各种杀虫剂下昆虫数目的带状图")
```



条件分割图 : coplot( )

给定某一个（或几个）变量z之后看我们所关心的变量分布情况。在条件分割图中，这种“分布”主要指的两个变量之间的关系，通常以散点图表示

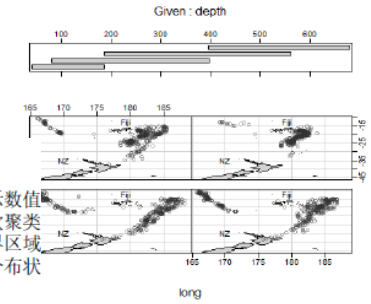
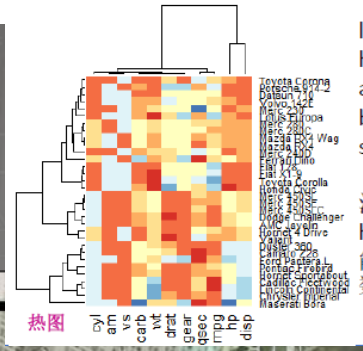


图 地震数据的条 斐济岛（Fiji）

热图: heatmap( )

热图也是将一个矩阵中单元格数值用颜色表达，如颜色深表示数值图并非只是简单表达数值大小，而是对矩阵的行或列进行层次聚类类的结果之后将行或列以聚类的顺序排列，并在颜色图的边界区域的谱系图。这样一来，我们不仅可以直接观察矩阵中的数值分布状以立即知道聚类的结果.

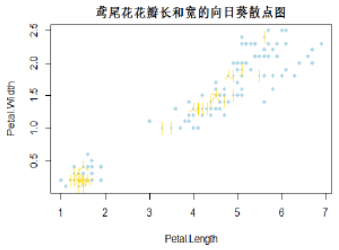



```
library(RColorBrewe
heatmap(as.matrix(
ars), col =
brewer.pal(9,"RdYlB
scale = "column",
      margins = c(4, 8
```

注意:as.matrix，heatmap()函数针对能是数据矩阵，不能数据框或其他

向日葵散点图：sunflowerplot( )

向日葵散点图在数据特别密集或者数据类型为分类数据时很有用，因为这两种情况下都容易产生重复的数据点，尤其是后一种情况下，数据几乎必然有重复（除非列联表单元格频数为1）。




```
par(mfrow=c(1,1), mar=c(4,4,2,2))
sunflowerplot(iris[, 3:4], col = "lightblue", seg.col = "gold",main="鸢尾花花瓣长和宽的向日葵散点图")#seg.col为花瓣的颜色
```

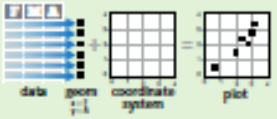
注意:左下方和中部都有一些重复数据

Data Visualization with ggplot2 Cheat Sheet

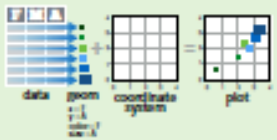
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

aesthetic mappings | data | geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes**(x = cty, y = hwy))

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method ="lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()

Returns the last plot

ggsave("plot.png", **width** = 5, **height** = 5)

Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

a + **geom_area**(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density**(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot**()
x, y, alpha, color, fill

a + **geom_freqpoly**()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram**(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar**()
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon**(aes(group = group))
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path**(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, linetype, size

d + **geom_ribbon**(aes(ymin=unemploy - 900, ymax=unemploy + 900))
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment**(aes(xend = long + delta_long, yend = lat + delta_lat))
x, xend, y, yend, alpha, color, linetype, size

e + **geom_rect**(aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank**()

f + **geom_jitter**()
x, y, alpha, color, fill, shape, size

f + **geom_point**()
x, y, alpha, color, fill, shape, size

f + **geom_quantile**()
x, y, alpha, color, linetype, size, weight

f + **geom_rug**(sides = "bl")
alpha, color, linetype, size

f + **geom_smooth**(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text**(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar**(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot**()
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot**(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill

g + **geom_violin**(scale = "area")
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter**()
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d**(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d**()
x, y, alpha, colour, linetype, size

i + **geom_hex**()
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area**()
x, y, alpha, color, fill, linetype, size

j + **geom_line**()
x, y, alpha, color, linetype, size

j + **geom_step**(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + **geom_crossbar**(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar**()
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange**()
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange**()
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + **geom_map**(aes(map_id = state), map = map) + **expand_limits**(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

m + **geom_contour**(aes(z = z))
x, y, z, alpha, colour, linetype, size, weight

m + **geom_raster**(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

m + **geom_tile**(aes(fill = z))
x, y, alpha, color, fill, linetype, size

Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. a + geom_bar(stat = "bin")

Each stat creates additional variables to map aesthetics to. These variables use a common **..name..** syntax.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. **stat_bin(geom="bar")** does the same as **geom_bar(stat="bin")**

stat function | layer specific mappings | variable created by transformation

```
i + stat_density2d(aes(fill = ..level..),
    geom = "polygon", n = 100)
```

geom for layer | parameters for stat

1D distributions

```
a + stat_bin(binwidth = 1, origin = 10)
  x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + stat_bindot(binwidth = 1, binaxis = "x")
  x, y, | ..count.., ..ncount..
a + stat_density(adjust = 1, kernel = "gaussian")
  x, y, | ..count.., ..density.., ..scaled..
```

2D distributions

```
f + stat_bin2d(bins = 30, drop = TRUE)
  x, y, fill | ..count.., ..density..
f + stat_binhex(bins = 30)
  x, y, fill | ..count.., ..density..
f + stat_density2d(contour = TRUE, n = 100)
  x, y, color, size | ..level..
```

3 Variables

```
m + stat_contour(aes(z = z))
  x, y, z, order | ..level..
m+ stat_spoke(aes(radius= z, angle = z))
  angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + stat_summary_hex(aes(z = z), bins = 30, fun = mean)
  x, y, z, fill | ..value..
m + stat_summary2d(aes(z = z), bins = 30, fun = mean)
  x, y, z, fill | ..value..
```

Comparisons

```
g + stat_boxplot(coef = 1.5)
  x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + stat_ydensity(adjust = 1, kernel = "gaussian", scale = "area")
  x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..
```

Functions

```
f + stat_ecdf(n = 40)
  x, y | ..x.., ..y..
f + stat_quantile(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x),
  method = "rq")
  x, y | ..quantile.., ..x.., ..y..
f + stat_smooth(method = "auto", formula = y ~ x, se = TRUE, n = 80,
  fullrange = FALSE, level = 0.95)
  x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..
```

General Purpose

```
ggplot() + stat_function(aes(x = -3:3),
  fun = dnorm, n = 101, args = list(sd=0.5))
  x | ..y..
f + stat_identity()
ggplot() + stat_qq(aes(sample=1:100), distribution = qt,
  dparams = list(df=5))
  sample, x, y | ..x.., ..y..
f + stat_sum()
  x, y, size | ..size..
f + stat_summary(fun.data = "mean_cl_boot")
f + stat_unique()
```

Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

```
n <- b + geom_bar(aes(fill = fl))
n
```

scale_ | aesthetic to adjust | prepackaged scale to use | scale specific arguments

```
n + scale_fill_manual(
  values = c("skyblue", "royalblue", "blue", "navy"),
  limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
  name = "fuel", labels = c("D", "E", "P", "R"))
```

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

General Purpose scales

Use with any aesthetic:
alpha, color, fill, linetype, shape, size

scale_*_continuous() - map cont' values to visual values
scale_*_discrete() - map discrete values to visual values
scale_*_identity() - use data values **as** visual values
scale_*_manual(values = c()**)** - map discrete values to manually chosen visual values

X and Y location scales

Use with x or y aesthetics (x shown here)

scale_x_date(labels = date_format("%m/%d"), breaks = date_breaks("2 weeks")) - treat x values as dates. See ?strptime for label formats.
scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

Color and fill scales

Discrete | Continuous

```
n <- b + geom_bar(
  aes(fill = fl))
```

```
n + scale_fill_brewer(
  palette = "Blues")
```

For palette choices:
library(RColorBrewer)
display.brewer.all()

```
n + scale_fill_grey(
  start = 0.2, end = 0.8,
  na.value = "red")
```

```
o <- a + geom_dotplot(
  aes(fill = ..x..))
```

```
o + scale_fill_gradient(
  low = "red",
  high = "yellow")
```

```
o + scale_fill_gradient2(
  low = "red", high = "blue",
  mid = "white", midpoint = 25)
```

```
o + scale_fill_gradientn(
  colours = terrain.colors(6))
```

Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

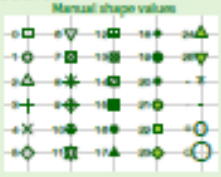
Shape scales

```
p <- f + geom_point(
  aes(shape = fl))
```

```
p + scale_shape(
  solid = FALSE)
```

```
p + scale_shape_manual(
  values = c(3:7))
```

Shape values shown in chart on right



Size scales

```
q <- f + geom_point(
  aes(size = cyl))
```

```
q + scale_size_area(max = 6)
```

Value mapped to area of circle (not radius)

Coordinate Systems

```
r <- b + geom_bar()
```

```
r + coord_cartesian(xlim = c(0, 5))
```

xlim, ylim
The default cartesian coordinate system

```
r + coord_fixed(ratio = 1/2)
```

ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

```
r + coord_flip()
```

xlim, ylim
Flipped Cartesian coordinates

```
r + coord_polar(theta = "x", direction=1 )
```

theta, start, direction
Polar coordinates

```
r + coord_trans(ytrans = "sqrt")
```

xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set extras and strains to the name of a window function.

```
z + coord_map(projection = "ortho",
orientation=c(41, -74, 0))
```

projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))
```

```
s + geom_bar(position = "dodge")
```

Arrange elements side by side

```
s + geom_bar(position = "fill")
```

Stack elements on top of one another, normalize height

```
s + geom_bar(position = "stack")
```

Stack elements on top of one another

```
f + geom_point(position = "jitter")
```

Add random noise to X and Y position of each element to avoid overplotting

Each position adjustment can be recast as a function with manual **width** and **height** arguments

```
s + geom_bar(position = position_dodge(width = 1))
```

Themes

```
r + theme_bw()
```

White background with grid lines

```
r + theme_classic()
```

White background no gridlines

```
r + theme_grey()
```

Grey background (default theme)

```
r + theme_minimal()
```

Minimal theme

ggthemes - Package with additional ggplot2 themes

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
```

```
t + facet_grid(. ~ fl)
```

facet into columns based on fl

```
t + facet_grid(year ~ .)
```

facet into rows based on year

```
t + facet_grid(year ~ fl)
```

facet into both rows and columns

```
t + facet_wrap(~ fl)
```

wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

```
t + facet_grid(y ~ x, scales = "free")
```

x and y axis limits adjust to individual facets

- **"free_x"** - x axis limits adjust
- **"free_y"** - y axis limits adjust

Set **labeller** to adjust facet labels

```
t + facet_grid(. ~ fl, labeller = label_both)
```

fl: c	fl: d	fl: e	fl: p	fl: r

```
t + facet_grid(. ~ fl, labeller = label_bquote(alpha ^ .(x)))
```

α^{c}	α^{d}	α^{e}	α^{p}	α^{r}

```
t + facet_grid(. ~ fl, labeller = label_parsed)
```

c	d	e	p	r

Labels

```
t + ggtitle("New Plot Title")
```

Add a main title above the plot

```
t + xlab("New X label")
```

Change the label on the X axis

```
t + ylab("New Y label")
```

Change the label on the Y axis

```
t + labs(title ="New title", x = "New x", y = "New y")
```

All of the above

Use scale functions to update legend labels

Legends

```
t + theme(legend.position = "bottom")
```

Place legend at "bottom", "top", "left", or "right"

```
t + guides(color = "none")
```

Set legend type for each aesthetic: colorbar, legend, or none (no legend)

```
t + scale_fill_discrete(name = "Title",
labels = c("A", "B", "C"))
```

Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)

```
t + coord_cartesian(
  xlim = c(0, 100), ylim = c(10, 20))
```

With clipping (removes unseen data points)

```
t + xlim(0, 100) + ylim(10, 20)
t + scale_x_continuous(limits = c(0, 100)) +
  scale_y_continuous(limits = c(0, 100))
```

ggplot2是一个用来绘制图形的R软件包。与其他大多数的图形软件包不同，ggplot是由其背后的一套图形语法所支持。ggplot2可以绘制出很多美观的图形，同时能避免诸多繁琐的细节。采用了图层的设计方式，从原始图层开始，首先绘制原始数据，然后不断添加图形注释和统计汇总结果。ggplot2可以帮助学生锻炼结构化的分析思维，进而达到专业的水准。在ggplot2中，用于创建新图形的表达式是由高级的图形元素组成的，例如对原始数据的展现，以及某些统计变换等。

ggplot2图形的基本组成部分：

1）geom_area()用于绘制面积图，即在普通线图的基础上，依y轴方向填充了下方面积的图形。

2）geom_bar(stat = “identity”)用于绘制条形图，需要指定stat = “identity”，因为默认的统计变换将自动对值进行计数。

3）geom_line()绘制线条图，图形属性group决定了哪些观测是连接在一起的。

4）geom_point()绘制散点图。

5）geom_polygon()绘制多边形，即填充后的路径。

6）geom_text()可在指定点处添加标签。

7）geom_tile()用来绘制色深图或水平图。

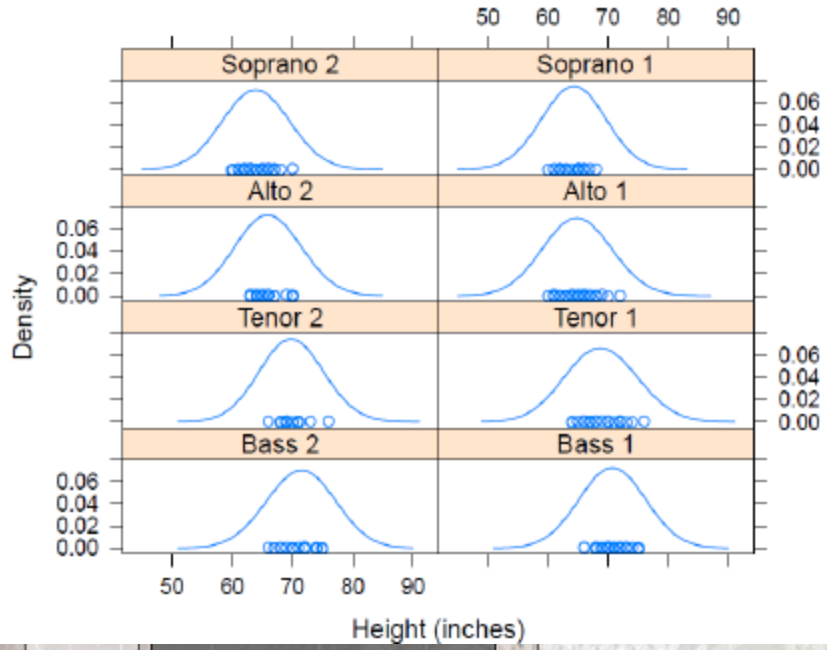
lattice

lattice是由Deepayan Sarkar基于grid包的一套统计图形系统，它的图形设计理念来自于Cleveland的Trellis图形，其主要特征是根据特定变量（往往是分类变量）将数据分解为若干子集，并对每个子集画图。就像数理统计中的条件期望、条件概率一样，lattice的图形也是一种“条件作图”。

格包中的高级函数如下：

函数	默认显示	函数	默认显示
histogram	直方图	xyplot	散点图
densityplot	核密度图	splom	散点图阵列
qqmath	理论分位数图	contourplot	表面等高线图
qq	双样本分位数图	levelplot	表面伪色彩图
stripplot	带形图	wireframe	三维表面透视图
bwplot	盒图	cloud	三维散点图
dotplot	克里夫兰点图	parallel	平行坐标图
barchart	条形图		

lattice




```
#以lattice包中的densityplot()函数为例

require(stats)
histogram( ~ height | voice.part, data = singer, nint = 17,
        endpoints = c(59.5, 76.5), layout = c(2,4), aspect = 1,
        xlab = "Height (inches)")
histogram( ~ height | voice.part, data = singer,
        xlab = "Height (inches)", type = "density",
        panel = function(x, ...) {
          panel.histogram(x, ...)
          panel.mathdensity(dmath = dnorm, col = "black",
                    args = list(mean=mean(x),sd=sd(x)))
        } )
densityplot( ~ height | voice.part, data = singer, layout = c(2
4),
          xlab = "Height (inches)", bw = 5)
```

附录1 数据生成

ID = seq(1,100)# 用seq函数产生100个数字，表示不同人的ID。

x1 = round(runif(100,min=50,max=100)) #用runif函数产生100个随机数（随机数是小数），代表科目1的成绩，该结果是**均匀分布**，用round函数对其取整。

x2 = round(rnorm(100,mean=80,sd=7))#用rnorm函数产生**正态分布数字**，代表科目2的成绩，再用round函数取整。

x3 = round(rnorm(100,mean=90,sd=14))#用rnorm函数产生**正态分布数字**，代表科目3的成绩，再用round函数取整。

x3[which(x3>100)]=100#将超过100的修改为100。

x = data.frame(ID,x1,x2,x3)# 将上面4个向量放到组合为数据框data.frame。

附录2 ggplot

```
library(ggplot2)
p <- ggplot(mtcars)
summary(p)
p <- p + aes(wt, hp)
summary(p)
p <- ggplot(mtcars, aes(x = mpg, y = wt))
p + geom_point()
p + geom_point(aes(colour = factor(cyl)))
p + geom_point(aes(y = disp))
p <- ggplot(mtcars, aes(mpg, wt))
p + geom_point(colour = "darkblue")
p + geom_point(aes(colour = "darkblue"))
```

```
library(nlme)
#ggplot2用数据集：nlme包里有一个简单的纵向数据集，26名男孩（Subject）在9个不同时期（Occasion）所测定的身高（height）和中心化后的年龄（age）
p <- ggplot(Oxboys, aes(age, height, group = Subject)) + geom_line()
p
p + geom_smooth(aes(group = Subject), method=”lm”, se=F)
p
p + geom_smooth(aes(group = 1), method=”lm”, size=2, se=F)
p
boysbox <- ggplot(Oxboys, aes(Occasion, height)) + geom_boxplot()
boysbox
boysbox+ geom_line(aes(group = Subject), colour=”#3366FF”)
boysbox

d <- ggplot(diamonds, aes(carat)) + xlim(0, 3)
d + stat_bin(aes(ymax = ..count..), binwidth = 0.1, geom = "area")
d + stat_bin(aes(size = ..density..), binwidth = 0.1, geom = "point", position="identity" )
d + stat_bin2d(aes(y = 1, fill = ..count..), binwidth = 0.1, geom = "tile",
position="identity")
```

谢谢