第 5 章 ggplot2 增强包介绍

由于 ggplot2 语法统一且能够支持绝大部分日常图表，因此生态圈越来越蓬勃繁荣。为了实现更加丰富的功能，不断有 ggplot2 相关的增强包出现，如 ggforce、ggstream、cowplot、ggthemes 等。这些包极大地简化或增强了 ggplot2 包原有的功能，并和其语法进行了融合，强烈建议读者按照需求深入学习。这些包首先都需要安装，语法是 install.packages('包名称')，如安装 ggforce 包，则可用 install.packages('ggforce')。包安装部分的方法都是相同的，因此在介绍具体的包时为了简化，假设读者已经安装了对应的包，安装过程将不再提及。

5.1 ggforce 包介绍

通过 install.package('ggforce')可以安装 ggforce 包，使用时 library('ggforce')将其加载到 R 环境中。该包可以对数据局部增加子图来放大局部数据，也可以对某些区域添加标识，该包还可以绘制平行集合图、维诺图(Voronoi 图)，通过多个多边形切分空间，也叫沃罗诺伊图，俄国数学家格奥尔吉·沃罗诺伊发明了该空间分割算法。

前面介绍过如何使用 xlim、ylim 将散点图局部数据放大，这种方法有利于观察数据的局部，但是也丢失了范围之外的数据，导致图形不能反映整体。通过 ggforce 中的 facet_zoom 可以对局部放大，同时保留整体图表。当然 facet_zoom 可以实现更多放大功能，如可以筛选放大等。下面通过 iris 数据集进行相关操作，前面在介绍 dplyr 包时介绍过这个数据集。首先，以 Petal.Length 和 Petal.Width 分别为 x 轴、y 轴绘制散点图，其中 aes(x=Petal.Length,y=Petal.Width)可以简写为 aes(Petal.Length,Petal.Width)。通过 facet_zoom(x = Species == 'versicolor')实现在 Species == 'versicolor'的情况下对 x 轴放大，也就是筛选 Species 等于 versicolor 的情况下对 x 轴放大，代码如下：

```
#代码 5-1 facet_zoom 对 x 轴数据放大
library(ggplot2)
library(ggforce)
ggplot(iris, aes(Petal.Length, Petal.Width, colour = Species)) +
  geom_point() +
  facet_zoom(x = Species == 'versicolor')
```

图形结果自上到下分为两部分，分别是整体图形及被放大的子图。图形突出了子图，因此子图所占面积会更大一些。代码运行的结果如图 5-1 所示。

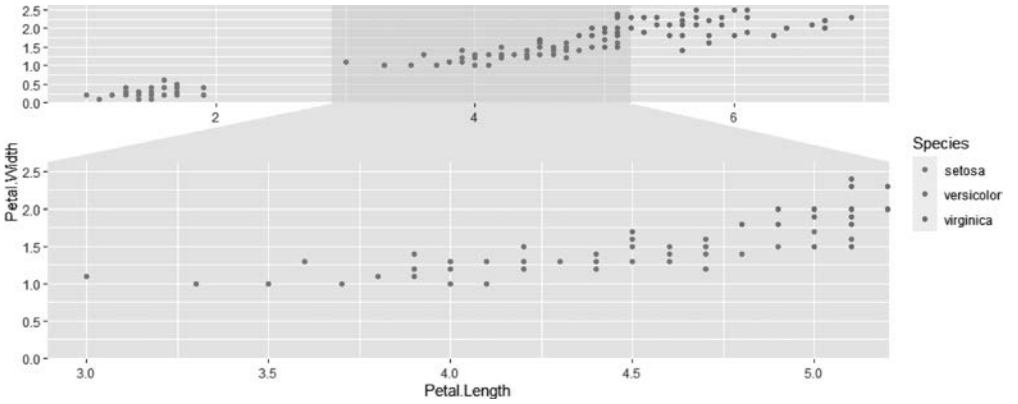

图 5-1　facet_zoom 对 x 轴数据放大

图 5-1 中上部的小图为原始图形，下部的大图为放大后的图形，这个显示效果用于强调被筛选出来的内容。显示效果与某些场景下需要“强调整体、弱化细节”是不同的，在实际使用场景中建议读者配合相关的文字说明。

也可以将 x 轴和 y 轴同时放大，facet_zoom(xy = Species == 'versicolor')将品类等于 versicolor 的点单独分离出来放大，代码如下：

```
#代码 5-2 facet_zoom 对 x 轴和 y 轴数据放大
library(ggplot2)
library(ggforce)
ggplot(iris, aes(Petal.Length, Petal.Width, colour = Species)) +
  geom_point() +
  facet_zoom(xy = Species == 'versicolor')
```

结果中左边是放大的局部图，右边是整体图，即右边包含了所有显示值，左边则只显示筛选的品种。代码运行的结果如图 5-2 所示。

将 x 轴和 y 轴同时放大的同时，在 facet_zoom 中可以增加参数 split = TRUE，将拆分后的数据显示为 x 轴放大、y 轴放大、x 轴和 y 轴同时放大 3 个子图，如图 5-3 所示，左上角为 y 轴放大图，右下角为 x 轴放大图，左下角为 x 轴和 y 轴放大图，代码如下：

```
#代码 5-3 facet_zoom 对 x 轴和 y 轴数据放大且分别显示
library(ggplot2)
library(ggforce)
ggplot(iris, aes(Petal.Length, Petal.Width, colour = Species)) +
  geom_point() +
  facet_zoom(xy = Species == 'versicolor', split = TRUE)
```

代码运行的结果如图 5-3 所示。

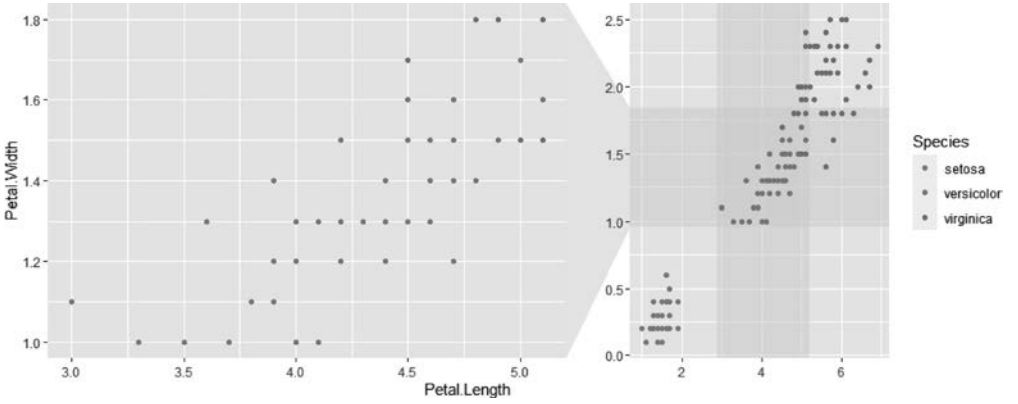

图 5-2　facet_zoom 对 x 轴和 y 轴数据放大

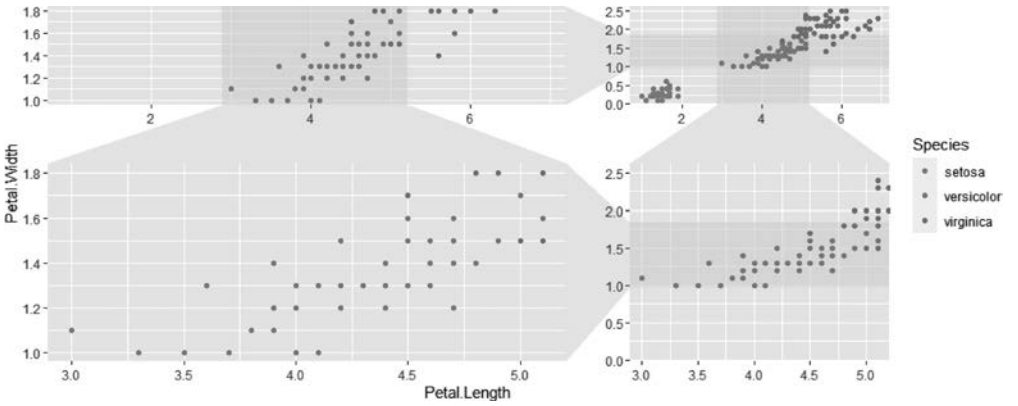

图 5-3　facet_zoom 对 x 轴和 y 轴数据放大且分图显示

前面的例子对特定筛选出的品类数据进行了放大，但是放大的是整个区域，因此也包含了其他 Species 的信息。如果只想显示筛选的 Species，则需要增加 zoom.data 参数，代码如下：

```
#代码 5-4 facet_zoom 指定放大数据
ggplot(iris, aes(Petal.Length, Petal.Width, colour = Species)) +
  geom_point() + facet_zoom(x = Species == 'versicolor', zoom.data = Species == 'versicolor')
```

facet_zoom 中设置了 zoom.data＝Species＝＝'versicolor'，即子图中只包含水平中为 versicolor 的数据观察点。代码运行的结果如图 5-4 所示。

当然，facet_zoom 也可设置参数 xlim＝c(最小值，最大值)或 ylim＝c(最小值，最大值)，将具体数值部分放大，当然也舍弃了筛选范围外的观测值。

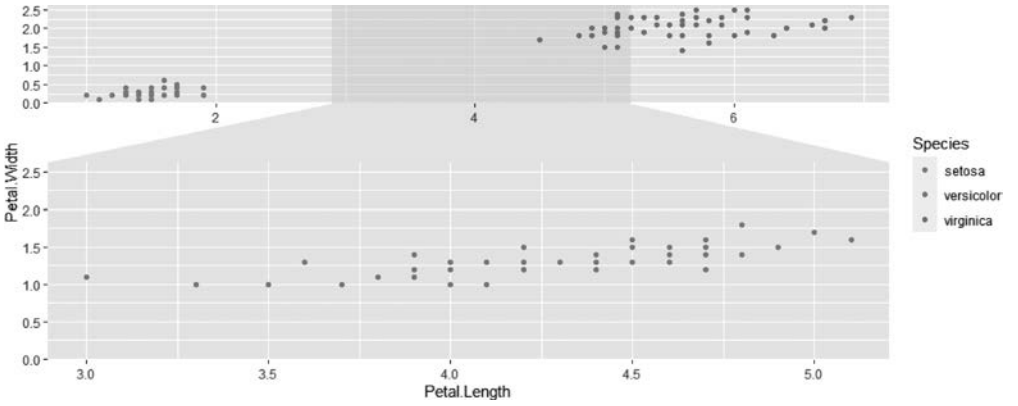

图 5-4　facet_zoom 指定放大数据

5.1.1　ggforce 中的分面

ggforce 中增加了几个分面的功能，如 facet_col 用于单列分面、facet_row 用于单行分面、facet_matrix 用于矩阵分面，下面分别举例介绍。先用 filter 函数筛选 US、CA、DE 3 个地区的数据，之后通过 facet_col 将数据在同一列分面展示，其中 scales='free'将 y 轴设置为依据数据自适应最优值。另外，space = 'free' 可以对每个子图的图片区域依据数据进行调整。facet_row 的使用方式与此类似，在此不举例说明，代码如下：

```
#代码 5-5 ggforce 分面
library(ggplot2)
library(ggforce)
library(readr)
library(dplyr)
library(magrittr)
data1 <- read_csv('D://Per//MB//bookfile//Mbook//data//salesdata.csv')
##Rows: 4425 Columns: 5
##Column specification
##Delimiter: ","
##chr (3): date, category, region
##dbl (2): quantity, sales
##
##i Use `spec()` to retrieve the full column specification for this data
##i Specify the column types or set `show_col_types = FALSE` to quiet this message
data1 %>% group_by(date,region) %>%
summarise(sales_total = round(sum(sales)/10000,0)) %>% filter(region %in% c('US','CA','DE') ) %>%
ggplot(aes(x = date,y = sales_total,color = region,group = region)) + geom_line() +
  facet_col(~region,scales = 'free') +
 theme_classic()
##`summarise()` has grouped output by 'date'. You can override using the `.groups` argument
```

facet_col 分面结果与 ggplot2 中的 facet_grid()与 facet_wrap()效果是一致的，作为另外一个技巧掌握即可。代码运行的结果如图 5-5 所示。

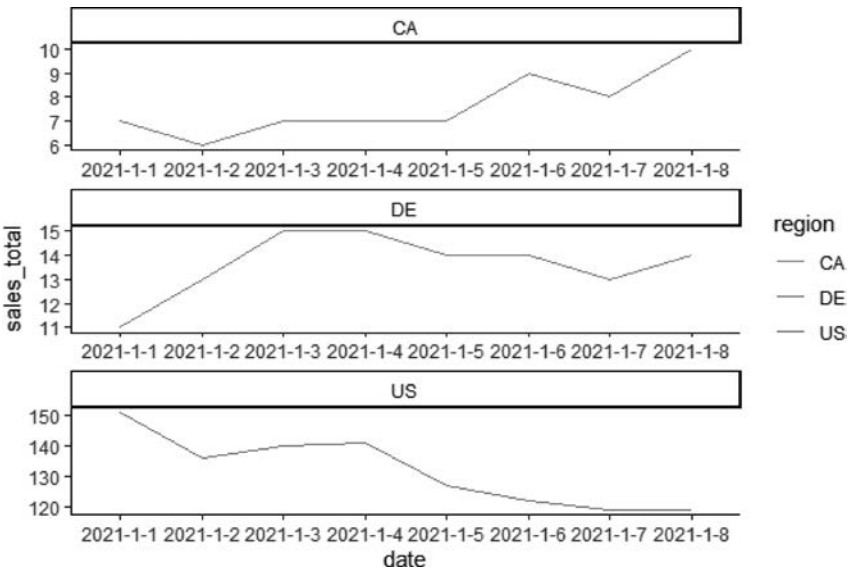

图 5-5　facet_col 分面

facet_matrix 可以依据选择的变量做矩阵分面，结果类似于散点图矩阵。下例中使用加载包自带的数据集 mpg，其实是一个描述不同生产商生产汽车的信息。选取其中的 3 个数值变量加入参数中，即 facet_matrix(vars(displ,cty,hwy))，最终得到一个散点矩阵图。该图便于观察数值变量间的大致关系，代码如下：

```
#代码 5-6 facet_matrix 分面
library(ggplot2)
library(ggforce)
ggplot(mpg) +
  geom_point(aes(x = .panel_x, y = .panel_y)) +
  facet_matrix(vars(displ, cty, hwy))
```

facet_matrix 分面散点图矩阵可以用来研究变量间的相关关系。粗略观察 hwy 与 cty 正相关性还是比较明显的，读者可以计算相关系数得到更加明确的结果。facet_matrix 分面散点图对线两边角显示的内容是重复的，观察一侧即可。示例结果如图 5-6 所示。

对于散点图矩阵，也可以直接将数据选取之后，使用基础绘图包 plot()绘制，其中 plot(data)也可以用 pairs(data)实现，代码如下：

```
#代码 5-7 散点图矩阵
data <- mpg %>% select(displ, cty, hwy)
plot(data)
```

代码运行的结果如图 5-7 所示。

如果使用 pairs()实现散点图矩阵，则可以选择保留对角线的上部或下部，希望深入研究的读者可以查看其帮助文件及网页，有更多自定义项可以选择。

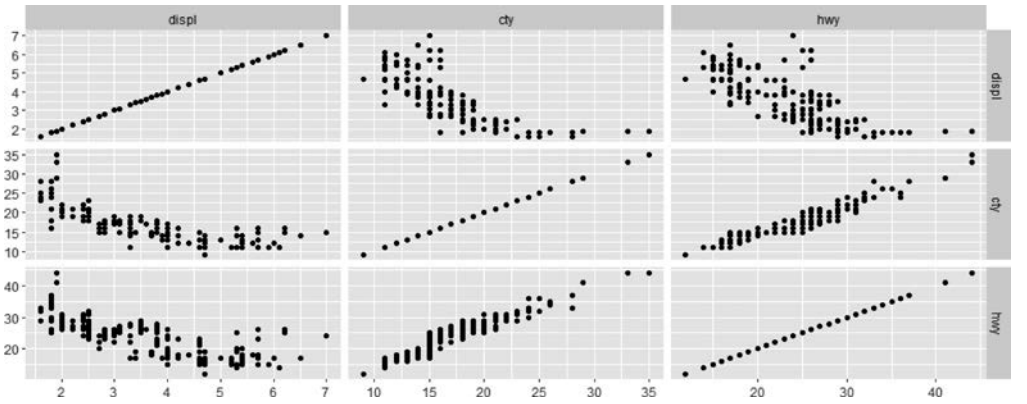

图 5-6　facet_matrix 分面

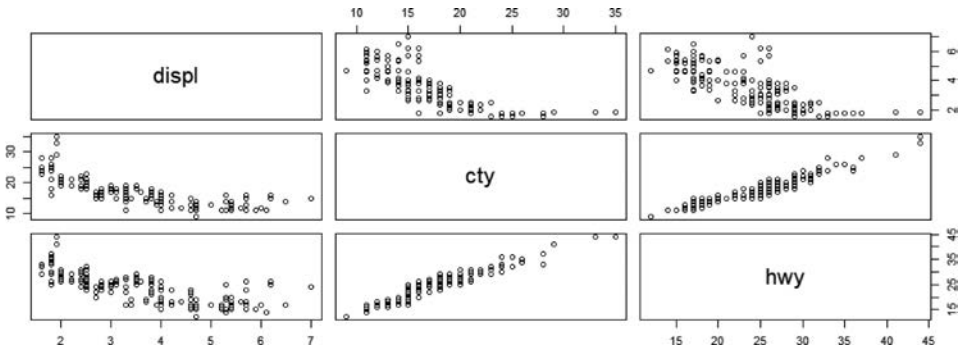

图 5-7　散点图矩阵

5.1.2　标注区域

在绘图中如果希望对某个区域进行标识，则可以使用 ggplot2 中的 annotate()设置参数 rect 实现，但只能使用矩形来标注并且需要输入确定值作为参数。ggforce 中的标识区域功能更加灵活，geom_mark_circle 使用圆形标识区域，geom_mark_ellipse 使用椭圆标识区域，geom_mark_hull 使用封闭曲线标识区域，geom_mark_rect 使用矩形标识区域。下面仍以鸢尾花数据集 iris 为数据源，首先绘制散点图，之后对某些品种的观测值增加批注信息。geom_mark_circle(aes(filter＝ Species ＝＝ 'versicolor'))用于实现对 Species 等于 versicolor 部分数据区域使用圆形进行标识，可以在 geom_mark_circle()中通过 color 来调整圆形边框颜色，通过 size 来调整圆形边框线条的粗细。需要注意在 R 环境中等号需要使用双等号，因此 Species 等于 versicolor 的代码为 Species ＝＝ 'versicolor'，代码如下：

```
#代码 5-8 geom_mark_circle 标识区域
library(ggplot2)
library(ggforce)
library(magrittr)
```

```
iris %>% ggplot(aes(Petal.Length, Petal.Width)) +
  geom_point(aes(color = Species)) +
  geom_mark_circle(aes(filter = Species == 'versicolor'))
```

黑色圆圈即是代码标识出来的区域，即 Species 等于 versicolor 的区域。当然，由于数据分布的原因，黑色圆圈内还包含了少量其他品种的观测值。代码运行的结果如图 5-8 所示。

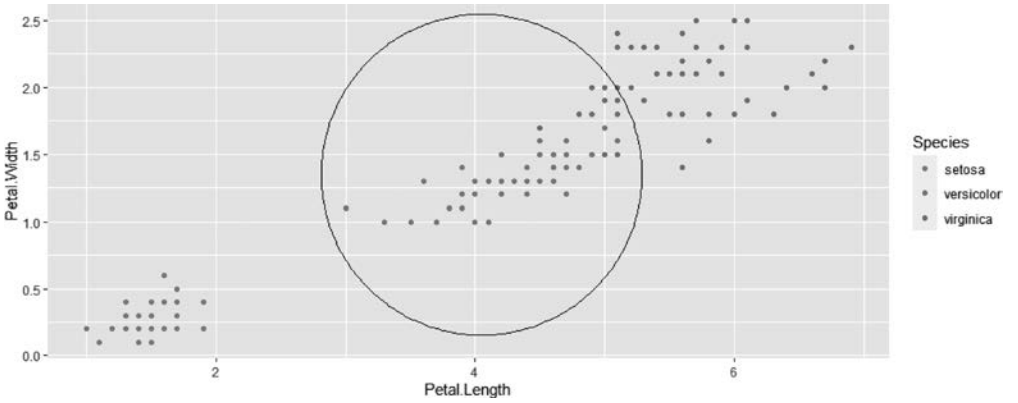

图 5-8　geom_mark_circle 标识区域

在 geom_mark_circle()函数中增加了 fill，可以设置圆形标识区域的填充色，如果将 fill 放入 aes()的内部，则可以将变量映射到填充色，同理边框颜色 color 也可以，代码如下：

```
#代码 5-9 geom_mark_circle 设置标识区域底色
library(ggplot2)
library(ggforce)
library(magrittr)
iris %>% ggplot(aes(Petal.Length, Petal.Width)) +
  geom_point(aes(color = Species)) +
  geom_mark_circle(aes(fill = Species,filter = Species == 'versicolor'))
```

增加了标识区域填充色后，突出效果更加明显。代码运行的结果如图 5-9 所示。

为了使图形更加清晰，一般可以在 geom_mark_circle 中通过 alpha 设置填充色的透明度，或者将 geom_point 放到绘图代码的最后。

geom_mark_ellipse 可以实现对数据使用椭圆形进行标识，代码与 geom_mark_circle 类似，代码如下：

```
#代码 5-10 geom_mark_ellipse 标识区域
library(ggforce)
library(magrittr)
iris %>% ggplot(aes(Petal.Length, Petal.Width)) +
  geom_point(aes(color = Species)) +
  geom_mark_ellipse(aes(fill = Species,filter = Species == 'versicolor'))
```

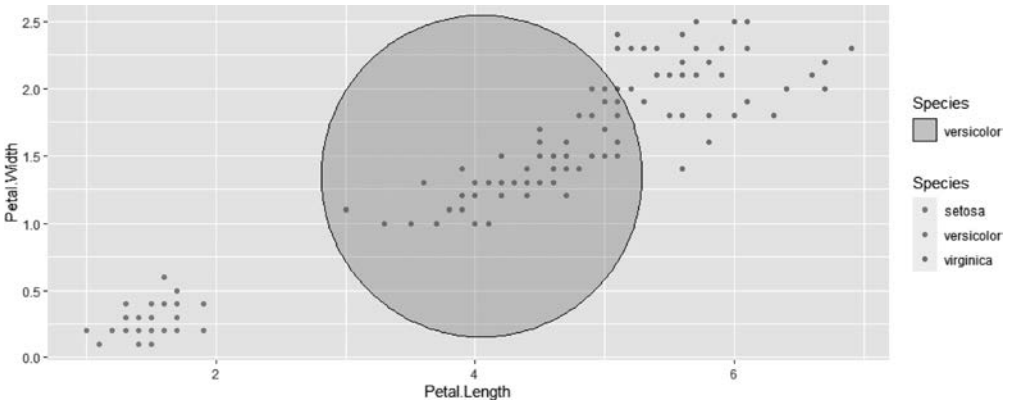

图 5-9　geom_mark_circle 设置标识区域底色

geom_mark_ellipse 圈识的区域比起使用圆更加聚焦，包含的非观测点空白区域会少些。代码运行的结果如图 5-10 所示。

图 5-10　geom_mark_ellipse 标识区域

geom_mark_hull 可以使用多边形对指点区域进行标识。由于该函数需要调用 concaveman 包的功能，所以需要通过 install.packages('concaveman')先安装该包，但是不需要手动单独将该包加载到 R 环境中，代码如下：

```
#代码 5-11 geom_mark_hull 设置标识区域填充色
library(ggforce)
library(magrittr)
iris %>% ggplot(aes(Petal.Length, Petal.Width)) +
  geom_point(aes(color = Species)) +
  geom_mark_hull(aes(fill = Species,filter = Species == 'versicolor'))
##Warning: The concaveman package is required for geom_mark_hull
```

代码运行的结果如图 5-11 所示。

通过 geom_mark_circle()、geom_mark_ellipse()、geom_mark_hull()等设置标识区域

图 5-11　geom_mark_hull 设置标识区域填充色

非常方便，并且可以与颜色、线条颜色、线条类型等配合使用，可以更好地对需要突出的内容突出显示。

5.1.3　平行集合图

平行集合图和桑基图比较类似，主要用来对数据的多个维度观察结构或者流量总分关系演变。在 ggforce 中使用 gather_set_data 函数对数据进行处理，之后使用 ggplot2 中的 ggplot()函数结合 ggforce 中的 geom_parallel_sets、geom_parallel_sets_axes、geom_parallel_sets_axes 来绘制平行集合图。由于该图形比较复杂、抽象，因此下面的例子从原始数据开始介绍。强烈建议读者在绘图环境中分步骤输入代码，分步查看运行结果，以便有一个直观的理解。

下面以 datasets 包中的 Titanic 为数据源，该数据源反映了泰坦尼克号事故中不同维度下乘客存活的状态。维度有船舱等级 Class、性别 Sex、年龄段 Age 、是否存活 Survived，Freq 表示频次。先使用 as.data.frame()函数将 Titanic 数据集由列联表转换为数据框，将结果赋值给对象 data0，在代码的最外边加上括号表示运行代码并显示结果，为了节约显示空间，使用 head()函数选取前 10 行数据，代码如下：

```
#代码 5-12 Titanic 数据源
library(ggforce)
library(magrittr)
(data0 <- Titanic %>% as.data.frame()) %>% head(10)
##  Class    Sex   Age      Survived  Freq
##1   1st   Male Child         No        0
##2   2nd   Male Child         No        0
##3   3rd   Male Child         No       35
##4  Crew   Male Child         No        0
```

```
##5    1st   Female Child       No    0
##6    2nd   Female Child       No    0
##7    3rd   Female Child       No   17
##8   Crew   Female Child       No    0
##9    1st     Male Adult       No  118
##10   2nd     Male Adult       No  154
```

之后，将 data0 使用 gather_set_data 转换为长格式。gather_set_data()中的第 1 个参数是待处理数据集，第 2 个参数是原始数据中需要将哪几列做长格式处理，可以类比 gather()函数理解。本例中对前 4 列进行处理，因此代码为 gather_set_data(data0,1：4)，最终将结果赋值给 plot_data。

最后，通过 ggplot2 中的函数 ggplot()和 ggforce 包中的函数 geom_parallel_sets()绘图，使用 geom_parallel_sets_axes()对图形中的垂直柱子宽度等进行设置，使用 geom_parallel_sets_labels()对图形中的垂直柱中的标签进行设置，代码如下：

```
#码 5-13 gather_set_data 重塑数据源并绘制平行坐标轴图
library(ggforce)
library(magrittr)
 plot_data <- gather_set_data(data0,1:4)

 ggplot(plot_data, aes(x, id = id, split = y, value = Freq)) +
  geom_parallel_sets(aes(fill = Sex), alpha = 0.3, axis.width = 0.1) +
  geom_parallel_sets_axes(axis.width = 0.2) +
  geom_parallel_sets_labels(colour = 'white',size=8,vjust=2)
```

代码中 geom_parallel_sets(aes(fill = Sex), alpha = 0.3, axis.width = 0.1)中的 aes(fill = Sex)用于将性别 Sex 映射给柱子间连接曲线带的填充色，alpha = 0.3 用于设置曲面带填充色透明度，axis.width = 0.1 用于设置曲面带水平宽度，可以理解为曲面带与黑色柱子中间的间隔，当该值为 1 时曲面带消失，当该值为 0 时曲面带与黑色柱子之间完全连接。

geom_parallel_sets_axes(axis.width = 0.2)用于设置柱子的宽度，geom_parallel_sets_labels(colour = 'white',size=6,vjust=2)用于设置柱子左侧标签文字的颜色及字体，其中默认柱子标签是在柱子中间的，vjust=2 用于将柱子标签左移。这里需要注意，vjust 实际上是对标签进行垂直移动，在本例中受 geom_parallel_sets_labels 原始代码影响最终体现的是左右移动。

上面介绍了图形的绘制方法，现对其表达意义给予说明，从图形 Age 列可以提炼出信息：乘客成年人 Adult 占绝大多数，从 Age 列右侧的连接曲面带可以看出男性 Male 占大多数；图形 Class 列代表客舱等级，从其中可以得出：船员 Crew 占比最大，其中男性占比非常高，其次 3 等舱 3nd 占比也明显。其他内容也可以类似理解。平行集合图可以对数据多维度做一个直观的理解，特别针对结构数据占比情况。代码运行的结果如图 5-12 所示。

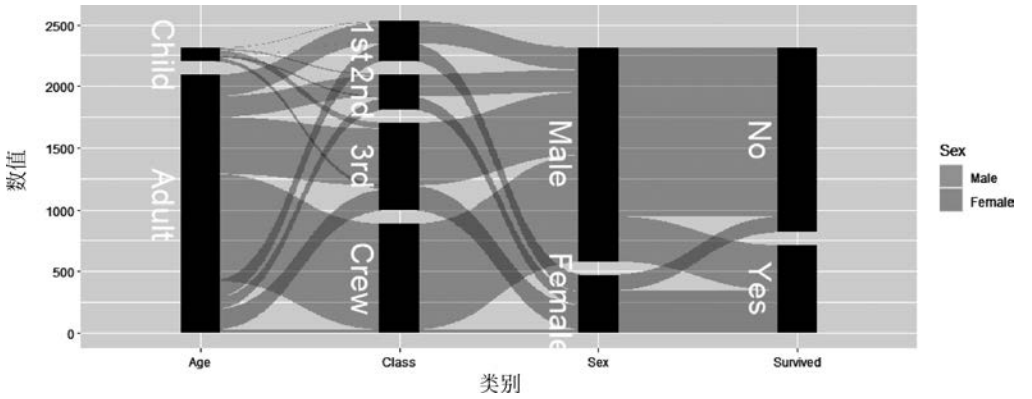

图 5-12　平行坐标轴图

平行坐标轴图用于表达数据的多个维度是非常直观的：如销售数据分析可能涉及品类间的占比、地区间的占比、销售部门间的占比等。通常的展示方法是将它们分开，逐个处理，但是相对烦琐，并且也不能用某个维度贯穿在这些分类中。

5.1.4　沃罗诺伊图

沃罗诺伊图采用多个多边形对空间切割，通过面积大小描述原始数据大小。该包依赖于 deldir 包，需要提前通过 install.packages('deldir')安装。下面以鸢尾花数据集的绘制举例。geom_voronoi_tile 中 aes(fill = Species)将按照 Species 分组填充颜色，geom_voronoi_segment()可以设置多边形框线，geom_text()中的参数比较复杂，stat(nsides)将每个分割点内的观测值计数后映射给标签，stat(vorarea)按照面积大小映射给标签，代码如下：

```
#代码 5-14 沃罗诺伊图
library(ggforce)
ggplot(iris, aes(Sepal.Length, Sepal.Width,group=1)) +
  geom_voronoi_tile(aes(fill = Species)) +
  geom_voronoi_segment() +
  geom_text(aes(label = stat(nsides), size = stat(vorarea)),
    stat = 'delvor_summary', switch.centroid = TRUE
  )
##Warning: stat_voronoi_tile: dropping duplicated points
##Warning: stat_voronoi_segment: dropping duplicated points
##Warning: stat_delvor_summary: dropping duplicated points
```

通过多边形将数据切分为不同大小的区域，并配合填充颜色，对数据结构进行直观展示并且节约了图形空间。代码运行的结果如图 5-13 所示。

另外，沃罗诺伊图可以对数据结构进行展示的同时，以小块数据区域数量显示数据分布聚集度。

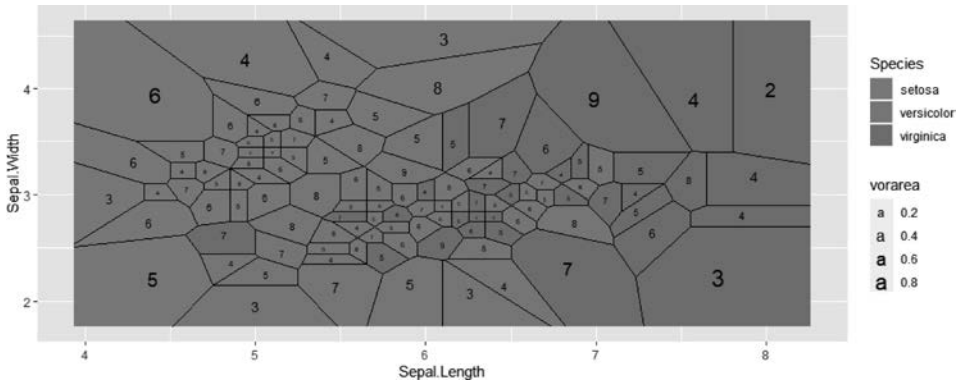

图 5-13　沃罗诺伊图

5.2　cowplot 包介绍

5.2.1　添加脚注

cowplot 包也是 ggplot2 增强包，提供了增加副标题、拼图、复合坐标轴图、组合边际图等内容，是提升 ggplot2 能力的重要补充包。下面介绍如何添加图表底部备注的功能，其中分为 3 个步骤：首先使用 ggplot2()绘制相关图形，其次使用 cowplot 包中的函数 add_sub()添加脚注，最后使用 ggdraw()函数生成最终的图形，代码如下：

```
#代码 5-15 添加脚注
library(ggplot2)
library(cowplot)
p1 <- ggplot(mtcars, aes(x = mpg, y = disp)) + geom_line(colour = "blue") + theme_minimal()
ggdraw(add_sub(plot = p1, label = "绘制折线图\n并添加底部备注的例子"))
```

add_sub()函数中参数 plot 是 ggplot2 生成的对象，参数 label 是文本标签，其中可以使用\n 实现换行显示，在此列中 plot 及 label 关键子都可以省略。代码运行的结果如图 5-14 所示。

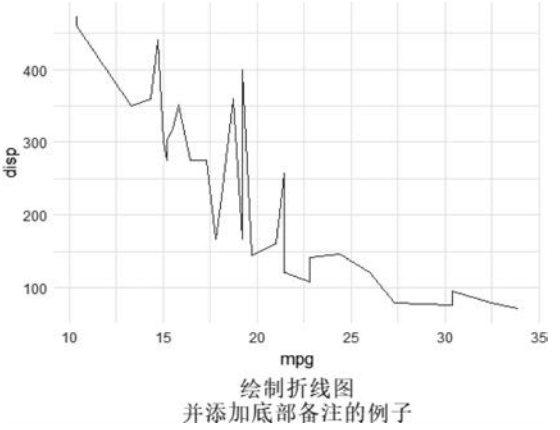

图 5-14　添加脚注

add_sub()中还可以设置文本位置坐标 x、y，以及设置文字大小、行间距等内容。具体可以参考 R 语言自带的帮助文档。

5.2.2 双坐标轴图

下面介绍 cowplot 绘制双 y 轴图，类似于 Excel 中添加次坐标轴序列。绘制的步骤为首先使用自带的数据集 mpg 中的前 100 行绘制两个 ggplot2 图形对象，接下来通过 cowplot 中的 align_plots()函数拼接 ggplot2 对象，最后使用 ggdraw()函数将拼接好的图形绘制出基础图，并使用 draw_plot()函数添加另一幅图形，代码如下：

```
#代码 5-16 双坐标轴图
library(ggplot2)
library(cowplot)
library(magrittr)
p1 <- ggplot(mpg %>% head(100), aes(x = manufacturer, y = hwy)) + stat_summary(fun.y =
"median",geom = "bar", fill = 'lightblue') + theme_half_open() +
  theme(axis.title.x = element_blank(), axis.text.x = element_blank())
p2 <- ggplot(mpg %>% head(100), aes(manufacturer, displ)) + geom_point(color = "red") +
scale_y_continuous(position = "right") + theme_half_open()
combine_plots <- align_plots(p1,p2,align = "hv",axis = "tblr")
ggdraw(combine_plots[[1]]) + draw_plot(combine_plots[[2]])
```

代码中使用 theme_half_open()主题，这个主题会将图形背景填充色变为透明，以便互相重叠后可以正常使用，手动自行设置也可以，不过相对烦琐，如图 5-15 所示。

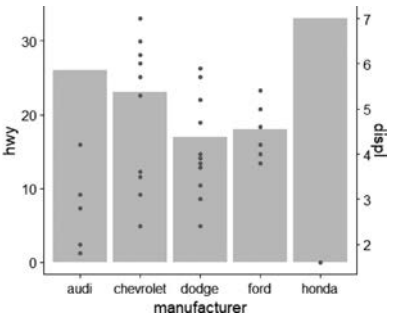

图 5-15 双坐标轴图

由于 p1 及 p2 中 x 轴标签是重复的，因此 p1 中使用 theme 中的 axis.title.x=element_blank()去除 x 轴标题，使用 axis.text.x=element_blank()去除 x 轴刻度标签，若要求不高，则这个去除代码可以不添加。align_plots(p1,p2,align = "hv",axis="tblr")中 align 用于设置图形的呈现方向，axis 用于设置两个图形坐标轴的对齐方式，具体参数可以参考帮助文档。

5.2.3 图形添边际密度图

ggplot2 绘制的图形中在 x 轴、y 轴或者同时在 x 轴和 y 轴增加绘图区域，之后在区域增加密度图。在新增加的区域也可以添加文字等其他内容。下面的例子以鸢尾花 iris 数据集为数据，第 1 步，先绘制主图 main_plot，在主图的基础上通过 axis_canvas()新建 x 轴边际画布，随后添加密度曲线，得到 x 轴子图 x_sub_plot，以同样的逻辑建立 y 轴子图 y_sub_plot。第 2 步，使用 insert_xaxis_grob 将 x_sub_plot 插入主图 main_plot 中，接着插入 y_sub_plot。第 3 步，使用 ggdraw()函数绘制图形，代码如下：

```
#代码 5-17 边际图
library(ggplot2)
library(cowplot)
library(magrittr)
#使用 ggplot2 新建散点图,作为主图
main_plot <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) + geom_
point()

#在主图上增加 x 轴边际画布,并增加密度曲线
x_sub_plot <- axis_canvas(main_plot, axis = "x") +
  geom_density(data = iris, aes(x = Sepal.Length, fill = Species), alpha = 0.7, size = 0.2)

#在主图上增加 y 轴边际画布,并增加密度曲线
y_sub_plot <- axis_canvas(main_plot, axis = "y", coord_flip = TRUE) +
  geom_density(data = iris, aes(x = Sepal.Width, fill = Species), alpha = 0.7, size = 0.2) +
  coord_flip()

#使用 insert_xaxis_grob()数据,将子图 x_sub_plot 插入主图 main_plot 中
p1 <- insert_xaxis_grob(main_plot, x_sub_plot, grid::unit(0.1, "null"), position = "top")
#接下来,使用 insert_yaxis_grob()函数插入 y_sub_plot,最终储存在对象 p2 中
p2 <- insert_yaxis_grob(p1, y_sub_plot, grid::unit(.2, "null"), position = "right")
#最终使用 ggdraw()将图绘制出来
ggdraw(p2)
```

使用上述代码在 x 轴及 y 轴添加了数据分布,并且通过填充色对品种 Species 进行了区分。结合中间的散点图,阅读者可以快速获得更多数据信息。代码运行的结果如图 5-16 所示。

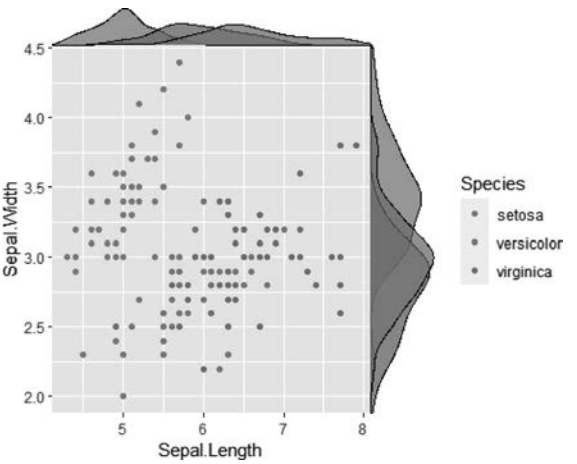

图 5-16 边际图

该图形的绘制过程比较复杂,建议读者分步执行代码,以便有一个直观的感受,最终理解绘制过程。y 轴的边际图反映了 3 个品类 Species 在 Sepal.Width 上的分布情况,同理可以理解 x 轴边际图。cowplot 包还有其他一些功能,感兴趣的读者可以参考帮助文档。

5.3 ggstream 包介绍

ggstream 包可以绘制河流图，在主要的绘制过程中将组别映射到 fill 参数，整体还是非常简单的。当然，该图不能精确地反映数据的值，另外数据量稍大时渲染时间会稍长。下面首先新建数据源 df，x = rep(1：10，3)表示 1～10 重复 3 次，即 1～10 每个值重复 3 次组成了变量 x。rpois(30，2)用于生成 30 个泊松随机数，其均值被设置为 2。sort(rep(c('A'，'B'，'C')，10))表示先用 rep()函数将向量 c('A','B','C')重复 10 次，之后用 sort()函数排序。最后使用 ggplot2()绘图函数并增加 geom_stream()即可绘制河流图，其中 fill 参数是必选参数，代码如下：

```
#代码 5-18 河流图
library(ggplot2)
library(ggstream)
#首先建立数据集:
#rep(1:10, 3)表示 1～10 重复 3 次,即 1～10 每个值重复 3 次组成了变量 x
#rpois(30, 2)用于生成 30 个泊松随机数,其均值被设置为 2
 df <- data.frame(x = rep(1:10, 3),
                  y = rpois(30, 2),
                  group = sort(rep(c("A", "B", "C"), 10)))
ggplot(df, aes(x=x, y=y, fill = group)) +
  geom_stream()+theme_classic()
```

河流图对于数据趋势的整体展示非常直观，无须额外说明，但是由于数据是堆叠的，因此不适合展示精确的数据趋势或者序列间的对比。代码运行的结果如图 5-17 所示。

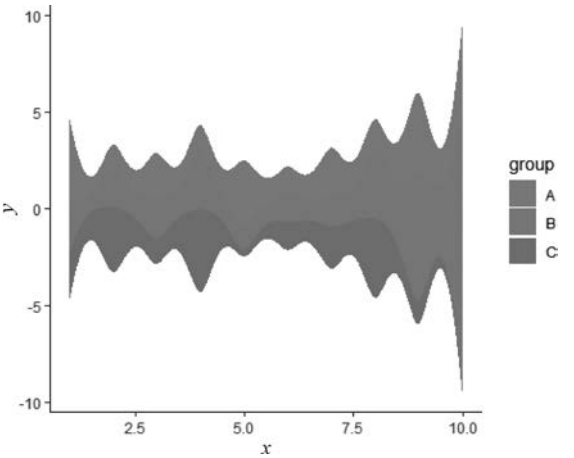

图 5-17 河流图

图 5-17 展示出值 7.5 前数据的整体波动趋势，之后增幅明显，这个变化主要受 B 组数据明显增加的影响。

5.4 ggrepel 包介绍

ggrepel 包主要包含 geom_text_repel()和 geom_label_repel()函数，用来处理标签，其功能与 ggplot2 包中的 geom_text()类似，不同点在于这两个函数可以自动地对每个标签的位置进行优化，尽量避免互相遮盖。geom_text_repel()和 geom_label_repel()的区别在于后者在添加标签文本的同时会在标签周围添加一个矩形框。下面以数据源 mtcars 为例介绍 geom_text_repel()的用法，代码如下：

```
#代码 5-19 geom_text_repel()处理文本遮盖问题
library(ggplot2)
library(ggrepel)
ggplot(mtcars,
  aes(wt, mpg, label = rownames(mtcars), colour = factor(cyl))) +
  geom_point() + geom_text_repel() + theme_classic()
```

通过 geom_text_repel()对文本标签的位置自动进行了优化，并为几个点与标签距离较远的添加了与点中间引导线。代码运行的结果如图 5-18 所示。

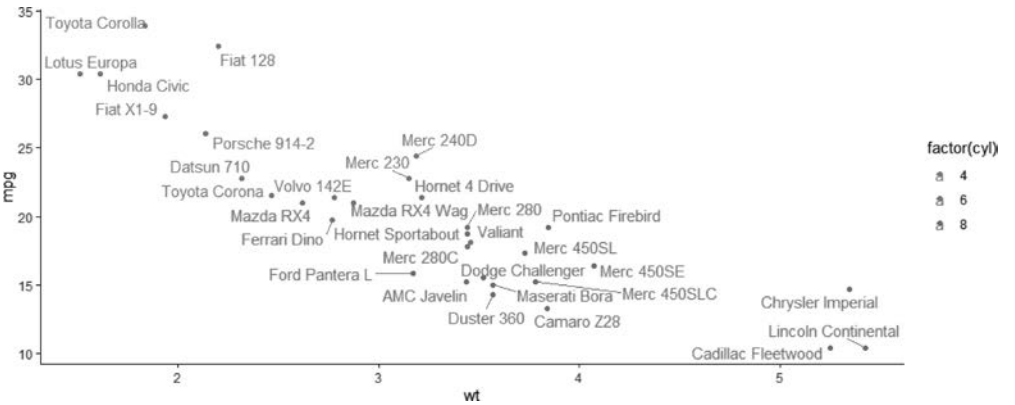

图 5-18 geom_text_repel()处理文本遮盖问题

geom_label_repel()在文本周围添加了文本框，除此之外功能和 geom_text_repel()一致，代码如下：

```
#代码 5-20 geom_label_repel()处理文本遮盖问题
library(ggplot2)
library(ggrepel)
ggplot(mtcars,
  aes(wt, mpg, label = rownames(mtcars), colour = factor(cyl))) +
  geom_point() + geom_label_repel() +
  theme_classic()
## Warning: ggrepel: 7 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

添加文本框标签后，标签文字更加清晰，但是标签占据了空间，图形相对显得拥挤。代码运行的结果如图 5-19 所示。

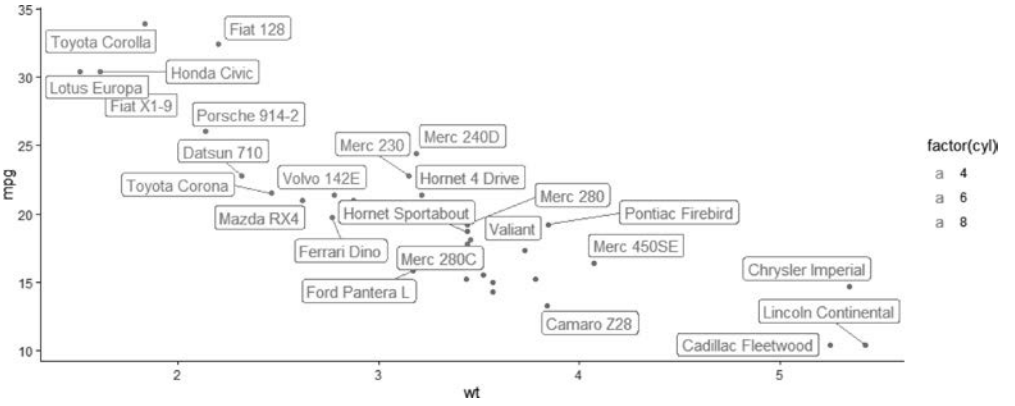

图 5-19　geom_label_repel()处理文本遮盖问题

图 5-19 中 label 可以设置边框颜色、填充底色等，在某些情况下可以展示另外的数据维度。

笔者常在 x 轴及 y 轴均是分类变量的情况下使用 geom_tile()展示数据，同时使用 geom_label()将某个变量映射到填充色中，类似于 Excel 中的条件格式显示效果。感兴趣的读者可以进一步研究。

5.5　treemapify 包介绍

treemapify 包可以绘制树形图，可以视同为面积分割图，即将总面积分割为不同的长方形，以此表示数值大小。也有人将谱系图称为树形图(像树根一样表达不同的层级关系)，后面章节会有提及。treemapify 包绘制的图其实不太像“树”，更像是迷宫图。treemapify 包是 ggplot2()增强包，因此遵守其语法，需要注意的是新增参数 area 将切割出来的面积大小与原始数据对应，ggplot()之后直接使用 geom_treemap()即可出图，代码如下：

```
#代码 5-21 树图
library(ggplot2)
library(treemapify)
library(magrittr)
treemap_data <- data.frame(category = c('A','B','C','D'),
                           sales = c(70,40,60,12))
treemap_data %>% ggplot(aes(fill = category,area = sales)) +
  geom_treemap()
```

当序列较多时 geom_treemap()展示各序列占比的效果优于饼图，并且会由大到小突出重点序列。代码运行的结果如图 5-20 所示。

如果希望将标签添加到图形中，则不能使用 geom_text()，而是需要使用 geom_treemap_text()函数，接下来在图 5-20 中添加品类及对应的金额作为标签，代码如下：

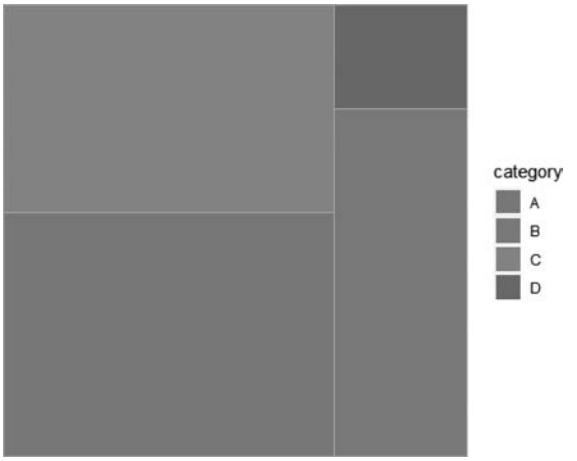

图 5-20　树图

```
#代码 5-22 向树图添加标签
library(ggplot2)
library(treemapify)
library(magrittr)
treemap_data <- data.frame(category = c('A','B','C','D'),
                           sales = c(70,40,60,12))
treemap_data %>% ggplot(aes(fill = category,area = sales)) +
  geom_treemap() + geom_treemap_text(aes(label = paste0(category,'\n',sales))) +
  theme(legend.position = 'none')
```

通常情况下建议数图都添加标签，增加其精确的数据信息，最终在快速展示数据值大小的同时，提供给读者更多信息。代码运行的结果如图 5-21 所示。

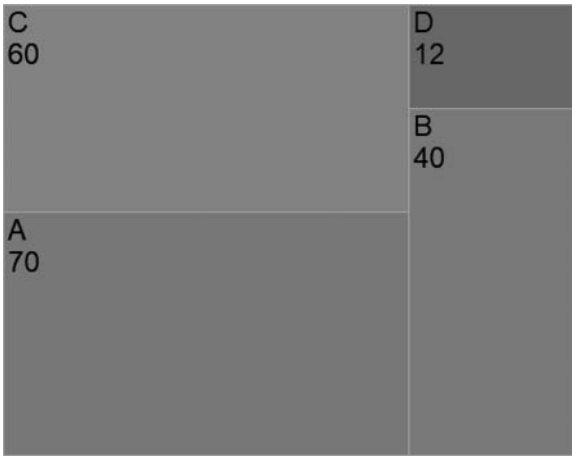

图 5-21　向树图添加标签

treemapify 包可以添加明细层级，分别使用 geom_treemap_subgroup_border()表达第 2 层级，使用 geom_treemap_subgroup2_border()表达第 3 层级。

树图由于使用长方形面积的大小表达每个部门的占比，整体空间使用率会高于饼图。另外，多层次结构也可以表达更多内容。

5.6　waterfalls 包介绍

waterfalls 包用来绘制瀑布图，展示数据演变的过程。该包的语法和 ggplot2 中的不同：fill_by_sign = TRUE 表示按照正负号分组填充颜色，如果参数值为 FALSE，则每项视同向 1 个离散变量填充颜色；calc_total = TRUE 表示需要计算总列数；total_rect_text 用于设置总列数柱标签的显示值；rect_border 用于设置是否显示柱子边缘线；rect_text_size 用于设置柱子标签文字大小的倍数；total_axis_text 用于设置总计列 x 轴标签的显示内容。如果还需要修饰其他内容，则可以设置包中的其他参数或者通过 ggplot2 中的 theme() 函数来完成，代码如下：

```
#代码 5-23 瀑布图
library(ggplot2)
library(waterfalls)
waterfall_data <- data.frame(item = c('收入','成本','销售费用','管理费用','财务费用'),
                        amount = c(1000, -550, -200, -90, -38))
waterfall_data %>% waterfall(fill_by_sign = TRUE,
                             calc_total = TRUE,
                             total_rect_text = sum(waterfall_data$amount),
                             rect_border = FALSE,
                             rect_text_size = 1.5,
total_axis_text = "利润") + theme_minimal() +
theme(axis.title = element_blank(),
      axis.text.x = element_text(size = 15))
```

本列中展示了收入到利润的关系，利润等于收入扣减成本费用后的结果，有一定财经背景的读者相信对此会更加熟悉。代码运行的结果如图 5-22 所示。

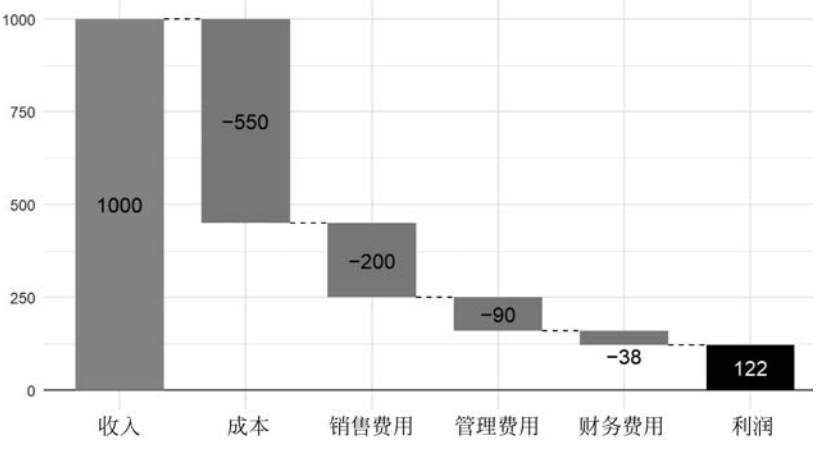

图 5-22　瀑布图

从操作角度考虑：Excel 中绘制瀑布图操作更加简单，设置标签、填充色等更加自由灵活，如果只是一次性制图，则可以考虑使用 Excel。

但是，如果是和计算等结合，希望后续能高效复用，则建议使用 R 语言。笔者在实际工作中经常使用循环语句绘制多幅瀑布图来展示不同角度的变化，效率提升还是非常明显的。

waterfalls 包不支持直接分面，但是如果需要，则可以通过循环生成图形之后拼接在一起。

5.7 geomtextpath 包介绍

这个包主要对 ggplot2 中文字标签的呈现进行了优化：将文字按照曲线或线条方向做出有弧度的文字，类似于 Office 中的有弧度艺术字。首先通过 install.packages('geomtextpath') 安装该包。也可以安装最新的开发版 remotes::install_GitHub("AllanCameron/geomtextpath")，如果用此方法安装，则需要先安装 remotes 包。另外，该包对英文支持较好，直接使用即可。如果使用中文，则由于不是所有类型的字体都可以转换角度，因此需要下载特定的字体，如下载 noto-sans 字体，之后通过 family 参数指定该字体。

5.7.1 geom_textpath 函数

geom_textpath()可以依据图形的形状添加对应的有弧度的文本标签，可以配合 coord_polar 极坐标使用。下例中的圆环图，如果使用 ggplot2 包中的 geom_text()，则可以添加文本，不过文本都是水平方向的，使用 geom_textpath()即可添加显示效果更佳的有弧度文本，代码如下：

```
#代码 5-24 向圆环图添加弧形标签
library(tidyverse)
library(geomtextpath)
library(ggplot2)
plot_data_2 <- data.frame(category = c('category_A','category_B','category_C','category_D'),
                          amount = c(1,6,4,7))

plot_data_2 %>% ggplot(aes(x = 1,y = amount,fill = category)) +
  geom_col() + geom_textpath(position = position_stack(vjust = 0.5),
                             aes(label = category)) +
  coord_polar() + theme_void()
```

添加有弧度的标签后，整个图形更加柔和，增加了高级感。代码运行的结果如图 5-23 所示。

向圆环图添加弧形标签这一技巧不会改变图形呈现的内容，不过对于希望对图形精修或有更高要求的读者是一项不错的选择。

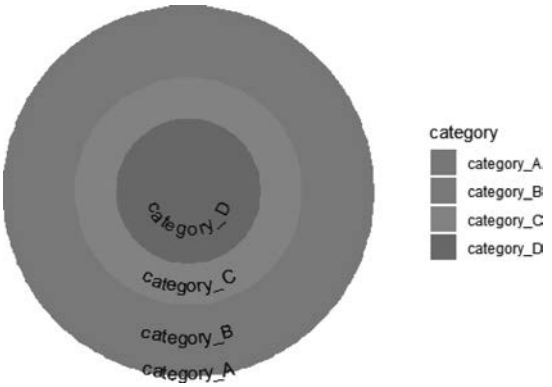

图 5-23　向圆环图添加弧形标签

5.7.2　geom_textline 函数

geom_textline()函数可以添加曲线及曲线标签，当然这个标签也是有弧度的。下例以datasets 包中的 pressure 数据集为例，使用 geom_textline()在绘制曲线的同时使用弧度标签。geom_textline()参数有 label 显示的文本标签、size 文字大小、vjust 垂直调整量、linewidth 线条粗细、linecolor 线条颜色、linetype 表示线型(实线或虚线等)、color 标签颜色。geom_textline()类似于 ggplot2()中 geom_line()和 geom_text()的组合，只是文字有了弧度，代码如下：

```
#代码 5-25 geom_textline()添加弧形文本标签
library(datasets)
library(geomtextpath)
library(ggplot2)
ggplot(pressure, aes(x = temperature, y = pressure)) +
  geom_textline(label = "curved pressure line", size = 6, vjust = -0.5,
                linewidth = 1, linecolor = "red4", linetype = 2,
                color = "deepskyblue4")
```

geom_textline()添加与曲线弧度相同的文本标签，增强了图形的趋势展示能力。代码运行的结果如图 5-24 所示。

在 geom_textline()中当数据序列比较多呈现锯齿状时，可以通过 text_smoothing 参数调整字体的平滑度，代码如下：

```
#代码 5-26 在 geom_textline()中调整标签弧度
library(ggplot2)
library(geomtextpath)
ggplot(economics, aes(date, unemploy)) +
  geom_textline(linecolour = "grey", size = 4, vjust = -1,hjust = 0.35,
                label = "1990s Decline", text_smoothing = 30)
## Warning: The text offset exceeds the curvature in one or more paths. This will result in
## displaced letters. Consider reducing the vjust or text size, or use the hjust parameter to
## move the string to a different point on the path
```

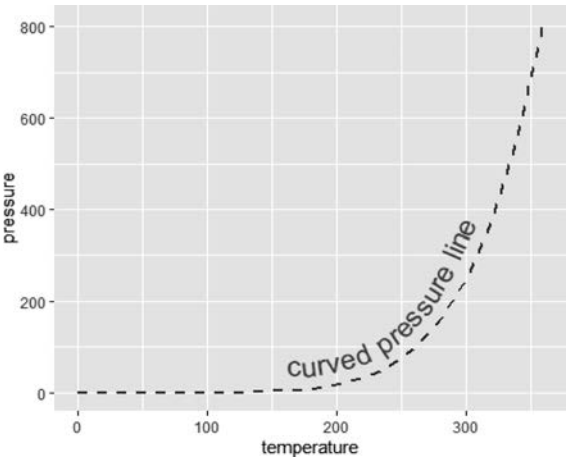

图 5-24　geom_textline()添加弧形文本标签

在 geom_textline()中调整标签弧度效果可能需要多次调整后才可以得到最优效果，当然使用 R 语言绘图的优势就是调整参数后可以快速绘制新条件下的图形。代码运行的结果如图 5-25 所示。

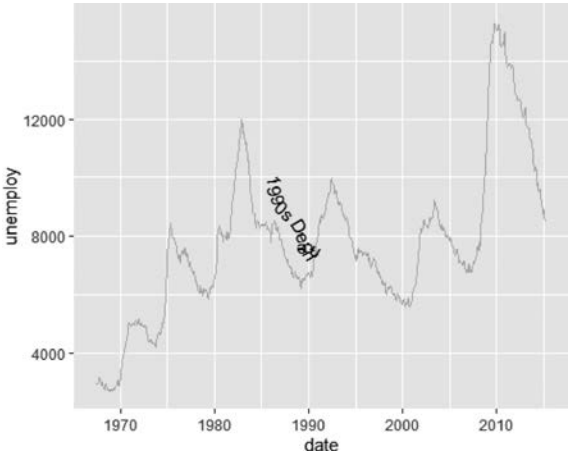

图 5-25　在 geom_textline()中调整标签弧度

5.7.3　geom_textdensity 函数

geom_textdensity()可以在密度曲线中添加弧度文字，图形线条部分和 geom_density()是一致的。下例中 fontface＝2 用于设置标签文字加粗，如参数是 1，则无加粗效果，其中的 2 也可以使用 bold。通过 hjust 及 vjust 调整文字在坐标轴上的位置，代码如下：

```
#代码 5-27 geom_textdensity()绘制密度曲线
library(geomtextpath)
```

```
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, colour = Species, label = Species)) +
  geom_textdensity(size = 6, fontface = 2, hjust = 0.2, vjust = 0.3) +
  theme(legend.position = "none")
```

标签位置为居于线条的中间，即线条会穿过文字的横向中心位置。当然这里所指的横向是一个直观的大概描述，不是精确描述，因为曲线是有斜率的。代码运行的结果如图 5-26 所示。

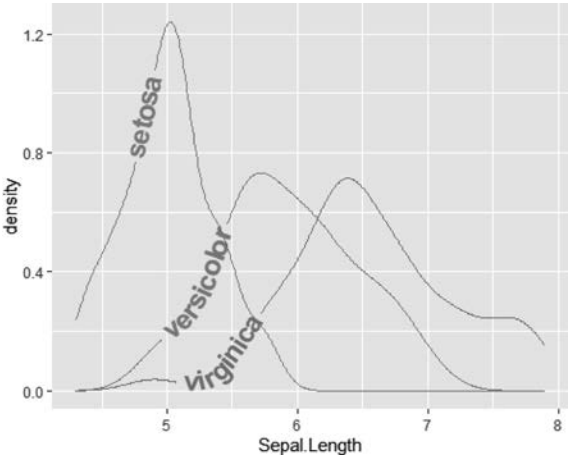

图 5-26　geom_textdensity()绘制密度曲线

图 5-26 中的标签位置通过 hjust、vjust 在水平及垂直方向给予了调整，如果希望标签在每个序列的最高点显示，则可以将 hjust 参数值调整为 ymax，代码如下：

```
#代码 5-28 在 geom_textdensity()中使用参数值 ymax
library(geomtextpath)
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, colour = Species, label = Species)) +
  geom_textdensity(size = 4, fontface = 2,spacing = 40, hjust = 'ymax', vjust = 0.3) +
  theme(legend.position = "none")
```

使用 ymax 后标签会位于每个序列的最高点，但是如果标签文字太长且曲线顶点较为陡峭，标签则会显得拥挤。代码运行的结果如图 5-27 所示。

代码中 spacing 参数用于设置文字占据位置的大小。当文字大小一定时，如果调小 spacing 参数，则文字间距会被压缩，更加紧凑，反之亦然。

geom_textdensity()添加的标签会位于线条的正中，会覆盖对应区域的线条，这一点与 geom_textline()添加的标签效果是不同的。

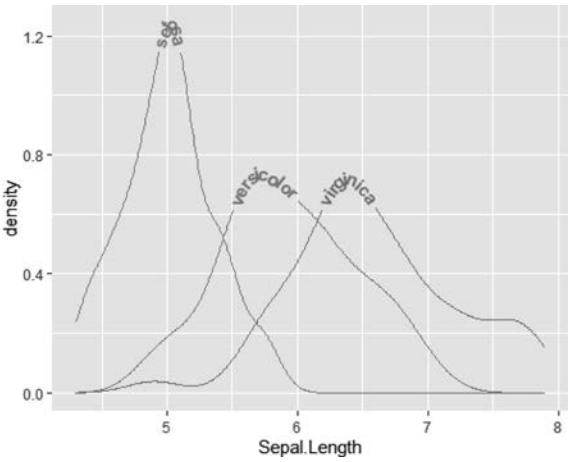

图 5-27　在 geom_textdensity()中使用参数值 ymax

5.7.4　geom_textsmooth 和 geom_labelsmooth

先前介绍过 geom_smooth 在散点图或折线图中添加拟合曲线。如果需要给拟合曲线添加弧度标签，则可以使用 geomtextpath 包中的 geom_textsmooth 和 geom_labelsmooth，后者会在文本下面添加一个带底色的文本框，二者的区别类似于 geom_text()和 geom_label()的区别。下例以 iris 为绘图数据源，其中 text_smoothing 用于设置标签文本的平滑度，fill 用于设置标签文本框的底色，method 用于设置拟合方法，boxlinewidth 用于设置标签文本框边线的粗细。scale_colour_manual 用于自定义图形中的线条及边框颜色。因为已经有文字标签，因此使用 theme(legend.position = 'none')将图例删除，以免信息重复，代码如下：

```
#代码 5-29 geom_labelsmooth 的用法
library(datasets)
library(geomtextpath)
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(alpha = 0.3) +
  geom_labelsmooth(aes(label = Species), text_smoothing = 30, fill = "#F6F6FF",
                   method = "loess", formula = y ~ x,
                   size = 4, linewidth = 1, boxlinewidth = 0.1) +
  scale_colour_manual(values = c("forestgreen", "deepskyblue4", "tomato4")) +
  theme(legend.position = "none")
```

geom_label_smooth 的左右与 geom_label 实现的效果类似，只是将标签框和标签文字给予了弧度，尽可能让更多图形元素表达数据的趋势，在呈现效果上更加精致。代码运行的结果如图 5-28 所示。

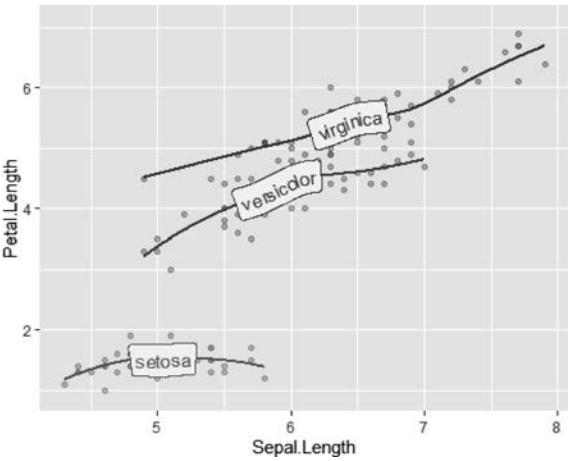

图 5-28　geom_labelsmooth 的用法

5.7.5　geom_contour_filled 和 geom_textcontour

geom_contour_filled()结合 geom_textcontour()可以绘制等高线密度图，geom_contour_filled()负责绘制二维密度图，geom_textcontour()负责绘制等高线及数值标签。下例中使用 volcano 数据集作为数据源。首先，使用 volcano 由 matrix 存储样式转换为 data.frame()，expand.grid()函数将原数据生成新数据框中的 x 和 y 变量，col_seq 将 volcano 中的值赋值给新数据框中的 z 变量。

在 geom_contour_filled()中 bins 用于设置数据分箱值，alpha 将填充色透明度设置为 60%。geom_textcontour()中的 straight=TRUE 用于设置等高线数值标签无弧度(但是数据的方向还是依随曲线有角度的)。在 scale_fill_manual(values = terrain.colors(11))中使用 terrain.colors 调色板中的 11 颜色填充。theme(legend.position ='none')用于去除图例，代码如下：

```
#代码 5-30 geom_textcontour 的用法
library(datasets)
library(geomtextpath)
library(ggplot2)
df <- expand.grid(x = seq(nrow(volcano)), y = seq(ncol(volcano)))
df$z <- as.vector(volcano)
ggplot(df, aes(x=x, y=y, z = z)) +
  geom_contour_filled(bins = 6, alpha = 0.6) +
  geom_textcontour(bins = 6, size = 4, straight = TRUE) +
  scale_fill_manual(values = terrain.colors(11)) +
  theme(legend.position = "none")
```

在实际工作中对大部分读者来讲应该使用机会较少，不过如果涉及地理等信息的展示，则是不错的选择。代码运行的结果如图 5-29 所示。

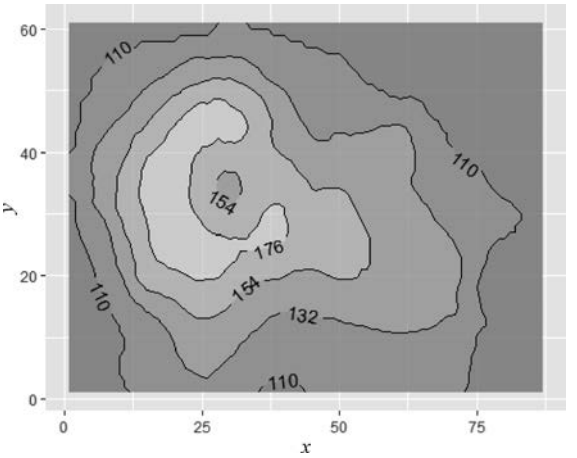

图 5-29　geom_textcontour 的用法

5.7.6　添加带标签的参考线

ggplot2 中可以使用 geom_hline()、geom_vline()、geom_abline()分别添加水平参考线、垂直参考线和有角度的直线。在 geomtextpath 包中对应的集合对象是 geom_texthline、geom_textvline、geom_textabline，只是能同时添加文本标签，ggplot2 中需要在绘制线条之后使用 geom_text()添加标签。geom_textabline()中的 slope 参数用于设置直线的角度，这 3 个几何对象中的其他参数都与上面例子中的参数类似，代码如下：

```
#代码 5-31 geom_textvline 的用法
library(geomtextpath)
library(ggplot2)
ggplot(mtcars, aes(mpg, disp)) +
  geom_point() +
  geom texthline(yintercept = 200, label = "horizon line",
                 hjust = 0.8, color = "red4") +
  geom_textvline(xintercept = 20, label = "vertical line", hjust = 0.8,
                 linetype = 2, vjust = 1.3, color = "blue4") +
  geom_textabline(slope = 15, intercept = -100, label = "slope line",
                  color = "green4", hjust = 0.6, vjust = -0.2)
```

可以调整斜率的参考线，是对 ggplot2 中 geom_vline()和 geom_hline()的重要补充。代码运行的结果如图 5-30 所示。

除了图 5-30 中的 3 类参考线外，时常需要标识序列间的差异值，如月份间差异、预算实际差异等，geom_textcurve()可以实现上述效果，代码如下：

```
#代码 5-32 geom_textcurve()添加参考线
library(ggplot2)
library(geomtextpath)
df <- data.frame(sales_type = c("actual", "budget"), sales = c(200, 150))
```

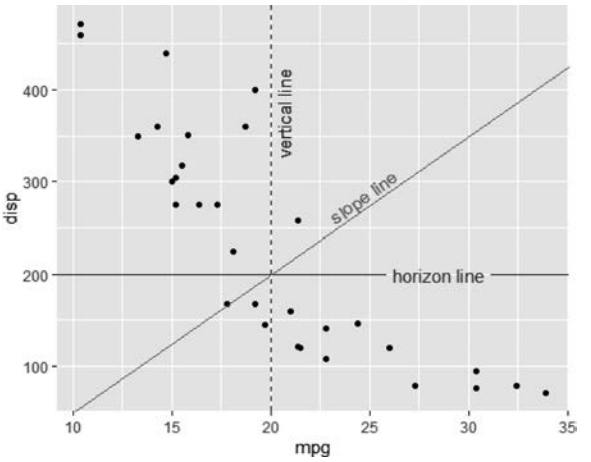

图 5-30　geom_textvline 的用法

```
gap <- paste0('actual vs budget : ',df $ sales[1] - df $ sales[2])
ggplot(df, aes(sales_type, sales)) +
  geom_col(fill = "gold", color = "gray50") +
  geom_textcurve(data = data.frame(x = 1, xend = 2,
                                   y = 200 + 20,
                                   yend = 150 + 20),
                 aes(x, y, xend = xend, yend = yend), hjust = 0.4,
                 curvature = -0.4, label = gap) +
  geom_point(aes(y = sales + 20)) +
  scale_y_continuous(limits = c(0, 300))
```

geom_textcurve()添加连接参考线后使图标更加直观，也符合大多数阅读者的理解方式，唯一不太方便的地方是需要构建标签数据源。代码运行的结果如图 5-31 所示。

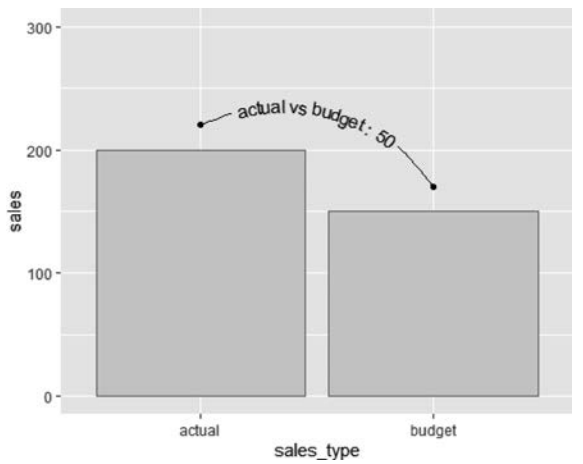

图 5-31　geom_textcurve()添加参考线

5.8 ggfittext 包介绍

geomtextpath 包用于解决随曲线方向显示有弧度标签的问题，ggfittext 包用于解决在特定区域里显示文字的问题：随区域大小改变文字字号或换行显示，当然弧形文字也可以实现。ggfittext 包更为重要的一项特性是配合 geom_tile()瓦片图中的文字可以反色显示，即填充色为深色时标签文字为浅色。下例中先使用 geom_tile()绘制瓦片图，然后使用 ggfittext 包中的 geom_fit_text()添加文字，代码如下：

```
#代码 5-33 ggfittext 的用法
library("ggplot2")
library("ggfittext")
ggplot(animals, aes(x = type, y = flies, label = animal)) +
  geom_tile(fill = "white", colour = "black") +
  geom_fit_text()
```

geom_fit_text()让标签文字以速效文字大小的方式显示在一行，结果类似于 Excel 单元格格式中的“缩小字体填充”。代码运行的结果如图 5-32 所示。

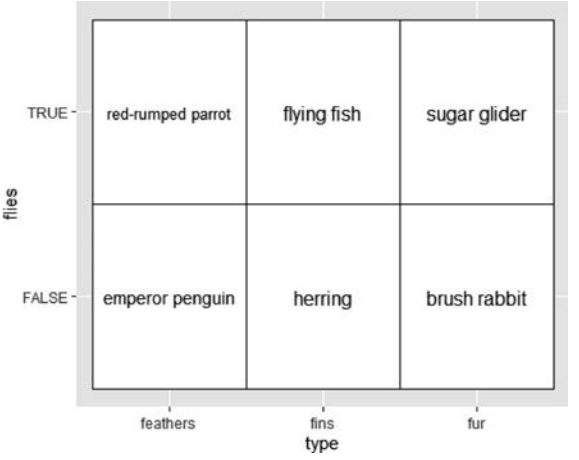

图 5-32 ggfittext 的用法

通过上面的例子，将标签文字设置为自动换行，在 geom_fit_text()中添加 reflow = TRUE 即可，代码如下：

```
#代码 5-34 在 ggfittext 中设置自动换行
ggplot(animals, aes(x = type, y = flies, label = animal)) +
  geom_tile(fill = "white", colour = "black") +
  geom_fit_text(reflow = TRUE)
```

在 ggfittext 中设置自动换行的结果类似于 Excel 单元格格式设置中的“自动换行”。代码运行的结果如图 5-33 所示。

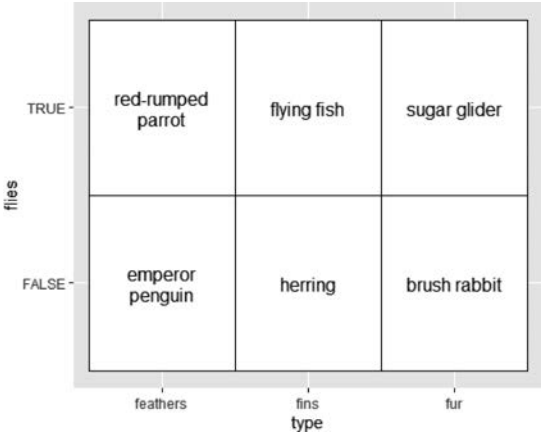

图 5-33　在 ggfittext 中设置自动换行

由于文字大小不同，下面将文字占用面积调整为一样，即将文字多的标签字号调小，将文字少的字号调整大，在 geom_fit_text()中设置 grow = TRUE 即可实现上述效果，代码如下：

```
#代码 5-35 在 ggfittext 中 grow 参数的用法
ggplot(animals, aes(x = type, y = flies, label = animal)) +
  geom_tile(fill = "white", colour = "black") +
  geom_fit_text(reflow = TRUE, grow = TRUE)
```

将文字自适应调整大小显示，无须对标签提前处理或逐个设置文字大小，非常方便。代码运行的结果如图 5-34 所示。

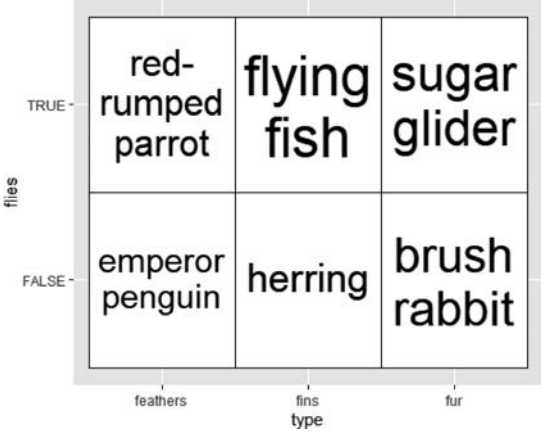

图 5-34　ggfittext 中 grow 参数的用法

下例中将文字调整为左上对齐，在 geom_fit_text()中设置 place ="topleft"即可实现上述效果，代码如下：

```
#代码 5-36 在 ggfittext 中设置文字对齐
ggplot(animals, aes(x = type, y = flies, label = animal)) +
```

```
  geom_tile(fill = "white", colour = "black") +
  geom_fit_text(place = "topleft", reflow = TRUE)
```

在实际应用中,统一调整标签位置是常见的需求。代码运行的结果如图 5-35 所示。

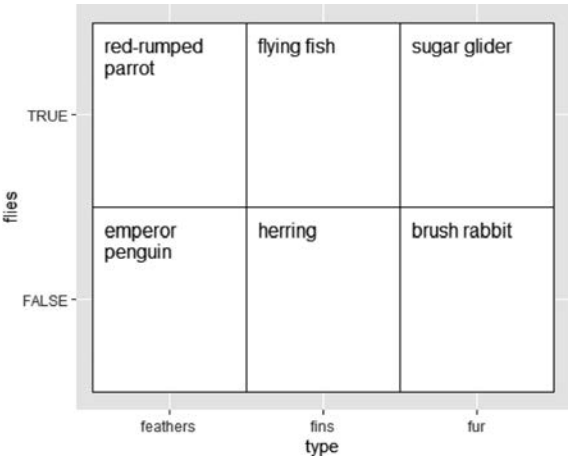

图 5-35　在 ggfittext 中设置文字对齐

在 ggplot2 包绘制柱状图的过程中文本位置与柱子有交错,填充色与字体色之间的色彩对比度弱,导致可视化效果差,需要输入参数调整才能有更好的显示效果,代码如下:

```
#代码 5-37 在柱状图中添加标签
ggplot(altitudes, aes(x = craft, y = altitude, label = altitude)) +
  geom_col() + geom_text()
```

使用 geom_text()不做特殊处理,标签文字有 50%是和图形重叠的。代码运行的结果如图 5-36 所示。

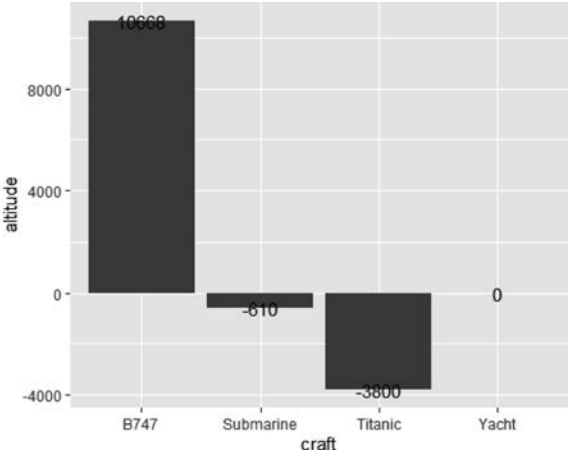

图 5-36　在柱状图中添加标签

图 5-36 中采用 ggplot2 中通常的方式添加标签，标签一般会与图形柱子重叠并且重叠部分都是黑色系颜色，显示效果差。在 ggplot2 中可以通过 vjust 参数调整文字的显示位置，通过 aes 中 colour 参数调整文字颜色，在此不进行介绍。

上述问题可以使用 ggfittext 包中的 geom_bar_text()替代 geom_text()来解决，代码如下：

```
#代码 5-38 geom_bar_text 的用法
ggplot(altitudes, aes(x = craft, y = altitude, label = altitude)) +
  geom_col() +geom_bar_text()
```

直接使用 geom_bar_text()即可对标签进行优化，代码运行的结果如图 5-37 所示。

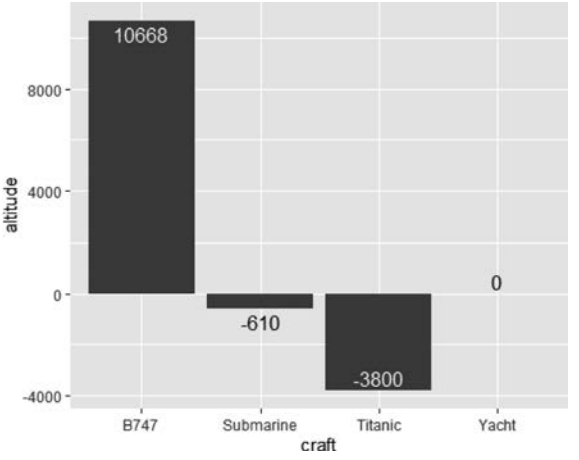

图 5-37　geom_bar_text 的用法

geom_bar_text 调整了文字颜色、文字位置，相对使用 ggplot2 中的方法更加简单。

下例中使用 geom_bar_text()给堆积柱状图添加标签，其中将对齐方式设置为 position = "stack"即可堆积显示，代码如下：

```
#代码 5-39 geom_bar_text 设置对齐方式
ggplot(beverages, aes(x = beverage, y = proportion, label = ingredient,
                      fill = ingredient)) +
  geom_col(position = "stack") +
  geom_bar_text(position = "stack", reflow = TRUE, place = 'center')
```

使用 ggplot2 中的 geom_text()可以增加堆积柱状图标签，geom_bar_text()可以方便地设置对齐效果，并且可以设置自动换行等效果。代码运行的结果如图 5-38 所示。

在簇状柱状图中如果通过 geom_text()添加标签，则需要在 position 中设置 position_dodge 参数，但是标签会有遮盖问题，代码如下：

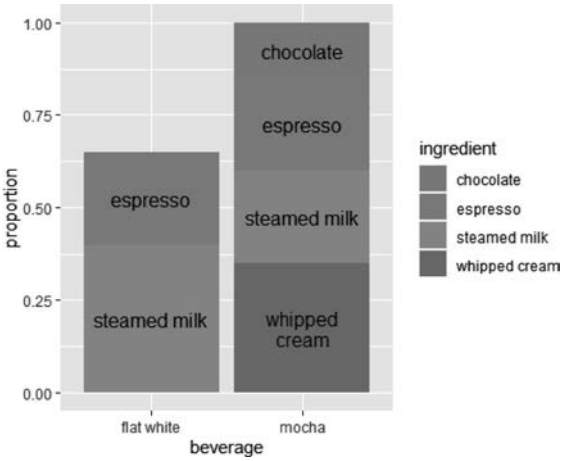

图 5-38　geom_bar_text 设置对齐方式

```
#代码 5-40 geom_text()添加标签
ggplot(beverages, aes(x = beverage, y = proportion, label = ingredient,
                      fill = ingredient)) +
  geom_col(position = "dodge") +
  geom_text(position = position_dodge(1))
```

使用 geom_text()添加标签的结果如图 5-39 所示。

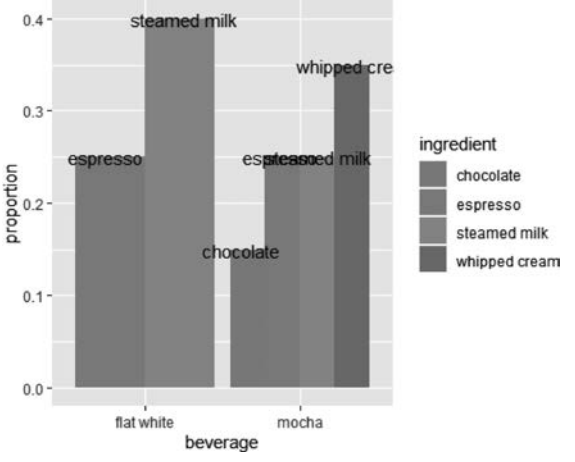

图 5-39　geom_text 添加标签

使用 geom_bar_text()增加标签，设置 reflow＝TRUE 后可实现自动换行。geom_bar_text()增加的标签不会重复出现，如序列 espresso 标签在第一簇中出现了，在第二簇中就没有显示，代码如下：

```
#代码 5-41 geom_bar_text 设置参数值 dodge
library(magrittr)
```

```
library(dplyr)
beverages %>% arrange(match(beverage,c("mocha","flat white"))) %>% mutate() %>%
ggplot(aes(x = beverage, y = proportion, label = ingredient,
                    fill = ingredient)) +
  geom_col(position = "dodge") +
  geom_bar_text(position = "dodge",reflow = TRUE)
```

上述代码的显示效果如图 5-40 所示。

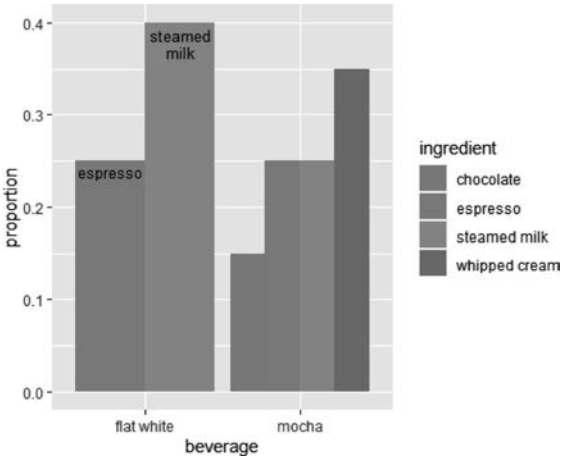

图 5-40 geom_bar_text 设置参数值 dodge

在上面的例子中如果对坐标轴使用 coord_flip()转置，效果则会更好，代码如下：

```
#代码 5-42 geom_bar_text 与 coor_flip
ggplot(beverages,aes(x = beverage, y = proportion, label = ingredient,
                    fill = ingredient)) +
  geom_col(position = "dodge") +
  geom_bar_text(position = "dodge",reflow = TRUE) +
  coord_flip()
```

在有长文本标签的情况下，建议做转置，阅读效果会更好。堆积柱状图添加标签并做转置后，结果如图 5-41 所示。

下例中使用 geom_fit_text()配合 geom_rect()和 coord_polar()实现极坐标下的弧形文字显示。绘图的步骤为先使用 geom_rect()绘制矩形图。矩形图参数分别是 4 个角的坐标，通过 x 轴最小值及最大值、y 轴最小值及最大值确定。geom_fit_text()用于添加文本。scale_fill_gradient()对矩形填充色基于最小值颜色值和最大值颜色值计算出一组颜色组合，代码如下：

```
#代码 5-43 geom_fit_text 的用法
ggplot(gold, aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax,
fill = linenumber, label = line)) +

  geom_rect() +
```

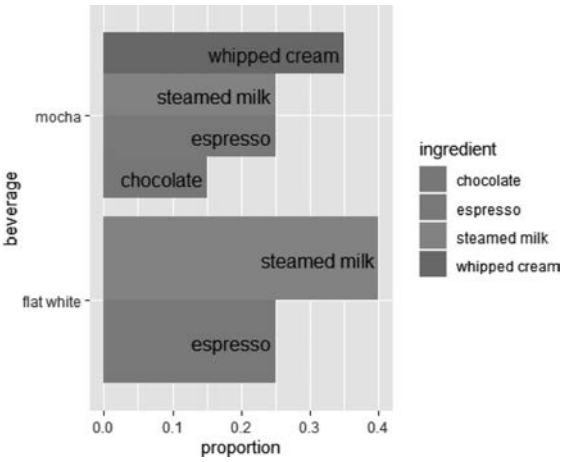

图 5-41 geom_bar_text 与 coor_flip

```
coord_polar() +
geom_fit_text(min.size = 0, grow = TRUE) +
scale_fill_gradient(low = "#fee391", high = "#238443")
```

上述代码用于展示层次结构应该是一个不错的选择，特别是不同级别间序列有交错的场景下。代码运行的结果如图 5-42 所示。

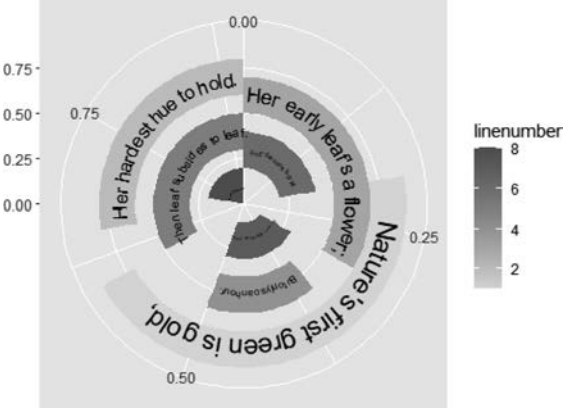

图 5-42 geom_fit_text()的用法

瓦片图中常见填充色和文字颜色值相近，造成标签不清晰。通过观察颜色值和背景色的关系，使用判断函数等对标签文本给予不同的颜色，可以解决该问题，不过略显烦琐。在 geom_fit_text()中设置 contrast = TRUE 即可解决这个问题，代码如下：

```
#代码 5-44 geom_fit_text 设置字体与背景色对比
library(RColorBrewer)
library(ggfittext)
```

```
ggplot(animals, aes(x = type, y = files, fill = mass, label = animal)) +
  geom_tile() +
  geom_fit_text(reflow = TRUE, grow = TRUE, contrast = TRUE)
```

在 geom_fit_text()中设置 contrast = TRUE 使文字颜色和背景色的对比增加了，显示更加清晰。代码运行的结果如图 5-43 所示。

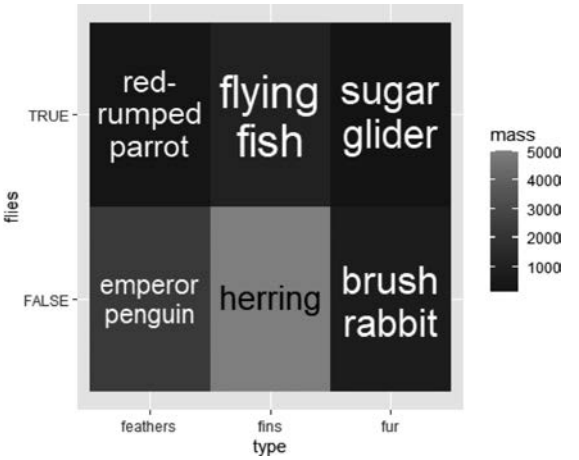

图 5-43　geom_fit_text()设置字体与背景色对比

在 geom_fit_text()中使用 contrast = TRUE 参数让字体颜色与背景色对比更加突出。使用 ggplot2 中的原有方法将相当费劲，需要多次尝试修改参数。

5.9　ggtext 包介绍

ggplot2 中可以使用 labs()、annotation()、geom_text()等添加标题注释等内容，在大多数情况下可以满足运用的需要。当需要在文本中显示多样格式，甚至显示图标或图片时可以使用 ggtext 包中的内容。该包的内容对 geom_text()等做了非常有益的补充，非常值得读者学习。该包文本部分使用 HTML 语法，这个对没有接触过的读者会是一个挑战，当然简单的语法还是比较好理解的。

5.9.1　在 theme()函数中使用 element_markdown()

在下面的例子中首先加载 tidyverse 包，该包集合了管道操作包、dplyr 包等，本例中需要使用其中的多个包，也可以分别加载各自的包。glue 包是胶水函数包，主要将数据框中的内容组合拼接为 HTML 语句，当然使用 paste0()或 paste()等函数也可以实现。使用 tibble()函数新建一个数据框，之后使用 dplyr 包中的 mutate()函数新建列，用于存储二进制颜色值及标签颜色值，使用 fct_reorder()函数对数值进行排序并转换为因子，便于在绘图中按照大小顺序显示。随后的绘图语法大多数和 ggplot2 中的语法类似，在 theme()函数中

调用了 element_markdown()参数，该参数将前面的文本解析为 HTML，并最终产生效果，代码如下：

```
#代码 5-45 element_markdown()的用法
library(tidyverse)
library(ggtext)
library(glue)

plot_data <- tibble(
  category = c("category_A", "category_B", "category_C", "category_D"),
  sales = c(9, 12, 7, 3),
)

plot_data %>% mutate(
  color = c("#009E73", "#D55E00", "#0072B2", "#000000"),
  label_text = glue("<i style='color:{color}'>{category}</i>")
) %>%
  ggplot(aes(x = sales, y = label_text, fill = color)) +
  geom_col(alpha = 0.5) +
  scale_fill_identity() +
  theme(axis.text.y = element_markdown())
```

通过上述代码可将标签颜色、填充颜色调整为对应的同一种颜色，虽然代码稍微烦琐，但是显示结果的友好性得到了提升。代码运行的结果如图 5-44 所示。

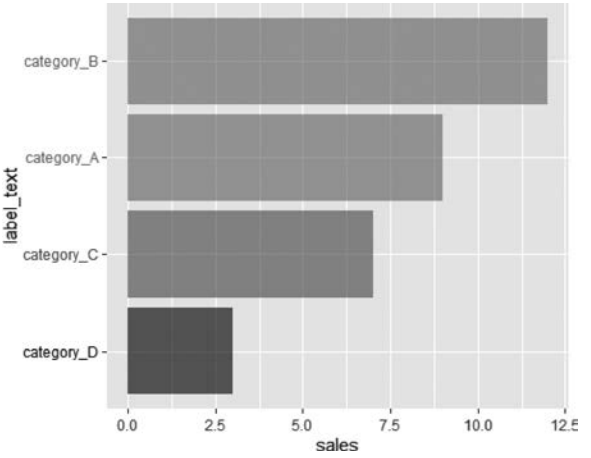

图 5-44 element_markdown()的用法

5.9.2 在 theme()函数中使用 element_textbox()

element_textbox()类似于在 ggplot2 中输入文本框，之后编辑文本框内容，适合在段落文字内部以不同的格式显示。下例中以自带数据集为绘图数据源绘制散点图，之后在图形中添加图标题 title、x 轴及 y 轴标签。标签文字采用 HTML 语法，之后在 theme()函数中使用 element_textbox_simple()函数对上述标签进行设置，代码如下：

```
#代码 5-46 element_textbox()的用法 1
library(ggplot2)
library(ggtext)
ggplot(mtcars,aes(disp,mpg)) +
  geom_point() +
  labs(
    title = "汽车燃油效率与引擎关系",
    x = "汽车引擎(disp)",
    y = "每加仑千米数(mpg)<br><span style = 'font-size:8pt'>衡量汽车燃油效率的一个指
标.</span>"
  ) +
  theme(
    plot.title.position = "plot",
    plot.title = element_textbox_simple(
      size = 13,
      lineheight = 1,
      padding = margin(5.5,5.5,5.5,5.5),
      margin = margin(0,0,0,0),
      fill = "cornsilk"),
    axis.title.x = element_textbox_simple(
      width = NULL,
      padding = margin(4,4,4,4),
      margin = margin(4,0,0,0),
      linetype = 1,
      r = grid::unit(8, "pt"),
      fill = "azure1"),
    axis.title.y = element_textbox_simple(
      hjust = 0,
      orientation = "left-rotated",
      minwidth = unit(1, "in"),
      maxwidth = unit(2,"in"),
      padding = margin(4,4,2,4),
      margin = margin(0,0,2,0),
      fill = "lightsteelblue"))
```

element_textbox_simple 优化了标签的样式,对图表细节比较关注的读者可以学习并掌握此技巧。代码的运行结果如图 5-45 所示。

在下面的例子中 element_blank()将分面标签替换为 element_textbox()并在其中设置标签文本的样式等,其中加载来自 cowplot 包中的 theme_half_open()样式,代码如下:

```
#代码 5-47 element_textbox()的用法 2
library(cowplot)
library(ggplot2)
library(ggtext)
ggplot(mpg, aes(cty, hwy)) +
  geom_point() +
```

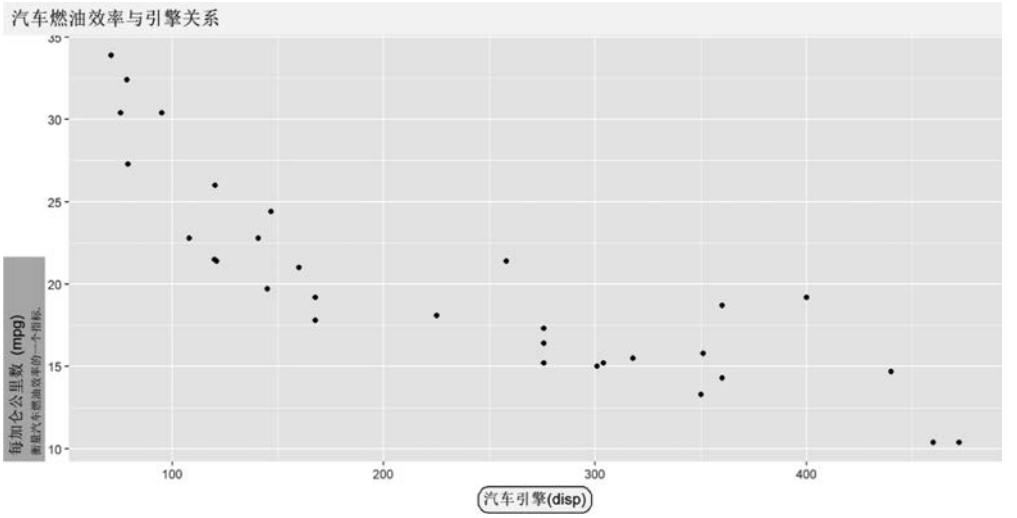

图 5-45　element_textbox()的用法 1

```
facet_wrap(~class) +
theme_half_open(12) +
background_grid() +
theme(
  strip.background = element_blank(),
  strip.text = element_textbox(
    size = 12,
    color = "white", fill = "#5D729D", box.color = "#4A618C",
    halign = 0.5, linetype = 1, r = unit(5, "pt"), width = unit(1, "npc"),
    padding = margin(2, 0, 1, 0), margin = margin(3, 3, 3, 3)
  )
)
```

对于分面标签优化的结果如图 5-46 所示。

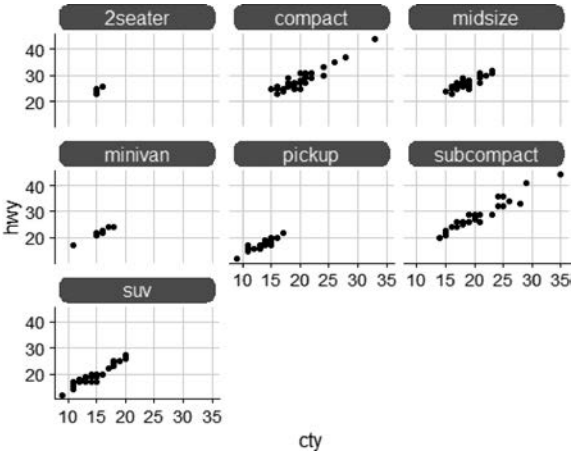

图 5-46　element_textbox()的用法 2

5.10 ggbreak包介绍

在绘制柱状图或条形图时，当数据间的差异比较大时，较小的数值容易显示不全。解决此问题可以使用对数刻度，或者对数据截断。使用前面学习的方法，即使用 xlim 或 ylim 参数截断数据，分别绘图后拼接在一起。当然也可以使用其他方法绘制多幅图，之后拼接在一起。简便的方式就是使用现成的包，例如 ggbreak。首先使用 ggplot2 绘制默认的柱状图，代码如下：

```
#代码 5-48 geom_col()绘制柱状图
library(ggplot2)
library(magrittr)
plot_data <- data.frame(category = c('a','b','c','d'),
                        sales = c(10,3,1000,1500))

plot_data %>% ggplot(aes(x = category,y = sales)) + geom_col()
```

在默认情况下前两个较小的数值对应的柱子非常低矮，基本无法区分其数值的大小。结果如图 5-47 所示。

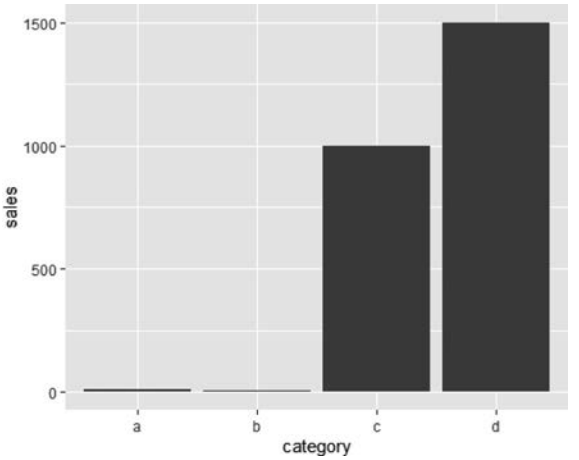

图 5-47 geom_col()绘制柱状图

针对图 5-47 中的不足之处，下例中对 y 轴使用对数刻度 scale_y_log10()，图形显示效果有明显改善，代码如下：

```
#代码 5-49 使用对数刻度 scale_y_log10()
library(ggplot2)
library(magrittr)
```

```
plot_data <- data.frame(category = c('a','b','c','d'),
                        sales = c(10,3,1000,1500))

plot_data %>% ggplot(aes(x = category,y = sales)) + geom_col() +
  scale_y_log10()
```

优化后的图形显示结果如图 5-48 所示。

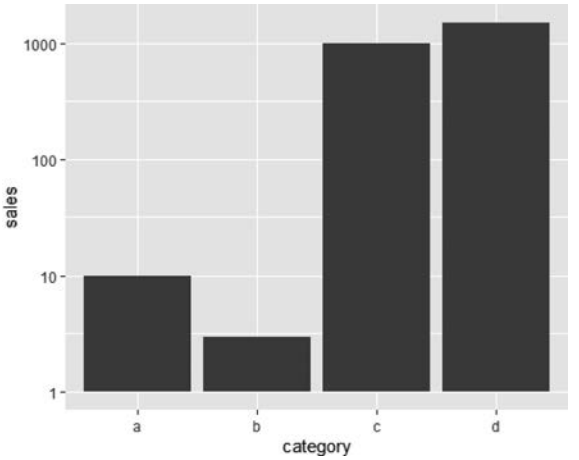

图 5-48　使用对数刻度 scale_y_log10()

图 5-48 能在一定程度上改善序列间差异较大的问题，但是需要给予备注解释信息，因为通常读者在理解柱状图时将 y 轴刻度理解为相同的，这样会夸大序列较小的值。笔者建议首选如下介绍的坐标轴截断方式。

最终使用 ggtext 包的截断功能对图形进行处理。在 scale_y_break()中通过 c(15,990)设置极端区域，scales ='free'设置为截断的两部分图各自按照不同的刻度大小显示，以便得到最优显示效果，代码如下：

```
#代码 5-50 绘制截断图
library(ggplot2)
library(magrittr)
library(ggbreak)
plot_data <- data.frame(category = c('a','b','c','d'),
                        sales = c(10,3,1000,1500))

plot_data %>% ggplot(aes(x = category,y = sales)) + geom_col() +
  scale_y_break(c(15,990),scales = 'free')
```

图形 y 轴截断后分为两部分，分别采用不用的数据量纲和刻度，并且图形的各部分以白色间隔区别。代码运行的结果如图 5-49 所示。

图 5-49 中可以在 scale_y_break()中设置参数 space 的值来调整截断空白的高度。其他优化技巧和普通的柱状图没有区别，读者可以尝试对上图进行进一步优化。

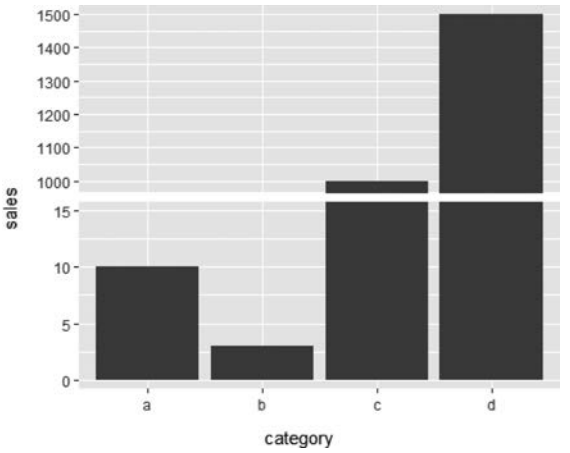

图 5-49 绘制截断图

5.11 ggpointdensity 包介绍

当点图比较密集时，常用的处理遮盖的方式就是分箱或者绘制二维密度图，使用热力颜色表达密度大小值。ggpointdensity 包中 geom_pointdensity()能够同时绘制点图和密度热力图。在下面的例子中使用了 viridis 包中的 scale_color_viridis()对颜色进行调整，该包包含非常好的颜色搭配，有兴趣的读者可以进一步研究，代码如下：

```
#代码 5-51 点图和密度热力图
library(ggplot2)
library(dplyr)
library(viridis)
##Loading required package: viridisLite
##
##Attaching package: 'viridis'
##The following object is masked from 'package:scales'
##
##viridis_pal
library(ggpointdensity)

dat <- bind_rows(
  tibble(x = rnorm(7000, sd = 1),
         y = rnorm(7000, sd = 10),
         group = "foo"),
  tibble(x = rnorm(3000, mean = 1, sd = .5),
         y = rnorm(3000, mean = 7, sd = 5),
         group = "bar"))

ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity() +
  scale_color_viridis()
```

代码运行的结果如图 5-50 所示。

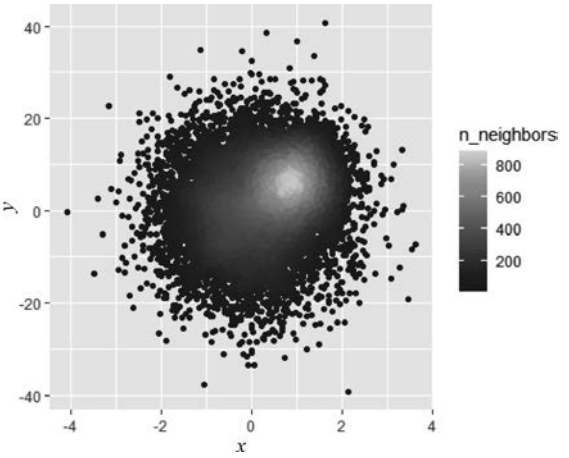

图 5-50　点图和密度热力图

在 geom_pointdensity()函数中可以使用 adjust 参数对二维密度图中的颜色聚类点数的大小进行调整，类似于在 geom_smooth()中可以调整线条的平滑程度，值越小聚类越细腻。首先使用参数值 0.01，代码如下：

```
#代码 5-52 参数 adjust 的使用(1)
library(ggplot2)
library(dplyr)
library(viridis)
library(ggpointdensity)

dat <- bind_rows(
  tibble(x = rnorm(7000, sd = 1),
         y = rnorm(7000, sd = 10),
         group = "foo"),
  tibble(x = rnorm(3000, mean = 1, sd = .5),
         y = rnorm(3000, mean = 7, sd = 5),
         group = "bar"))

ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity(adjust = .01) +
  scale_color_viridis()
```

调整后的图形如图 5-51 所示。

之后使用参数值 0.8，读者可以对比其中的明显差异：高亮黄色区域增加了，代码如下：

```
#代码 5-53 参数 adjust 的使用(2)
library(ggplot2)
library(dplyr)
library(viridis)
```

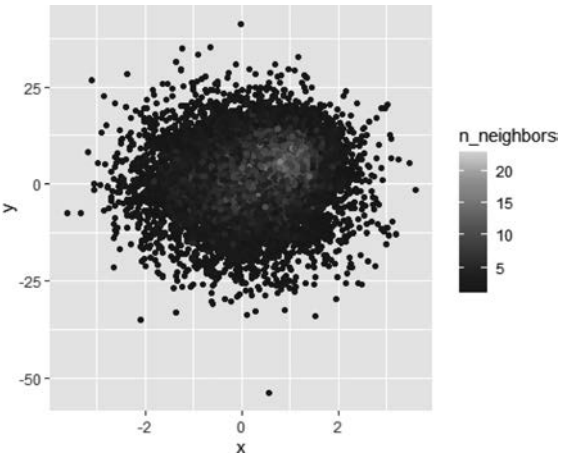

图 5-51　参数 adjust 的使用(1)

```
library(ggpointdensity)

dat <- bind_rows(
  tibble(x = rnorm(7000, sd = 1),
         y = rnorm(7000, sd = 10),
         group = "foo"),
  tibble(x = rnorm(3000, mean = 1, sd = .5),
         y = rnorm(3000, mean = 7, sd = 5),
         group = "bar"))

ggplot(data = dat, mapping = aes(x = x, y = y)) +
  geom_pointdensity(adjust = .8) +
  scale_color_viridis()
```

调整后的图形如图 5-52 所示。

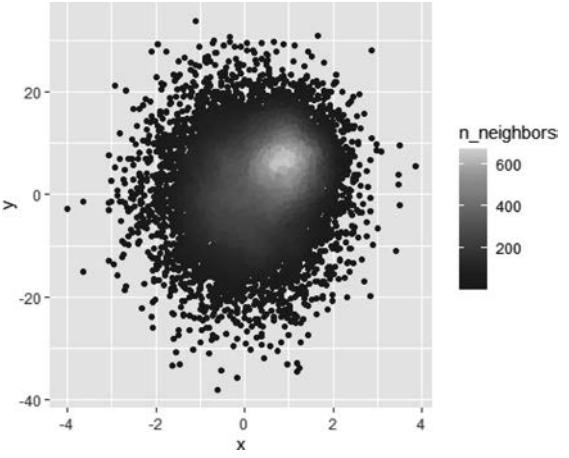

图 5-52　参数 adjust 的使用(2)

5.12 ggridges 包介绍

ggridges 包中的 geom_density_ridges()可以绘制峰峦图。峰峦图其实就是多系列密度图在 y 轴给予一个错位量，将不同序列错位显示。与 ggplot2 中通过颜色映射等区别不同序列密度图相比较，峰峦图更加清晰。与 ggplot2 中通过分面显示不同序列密度图的效果相似，但峰峦图更加紧凑、节省 y 轴空间，代码如下：

```
#代码 5-54 基础峰峦图
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species)) + geom_density_ridges()
```

代码运行的结果如图 5-53 所示。

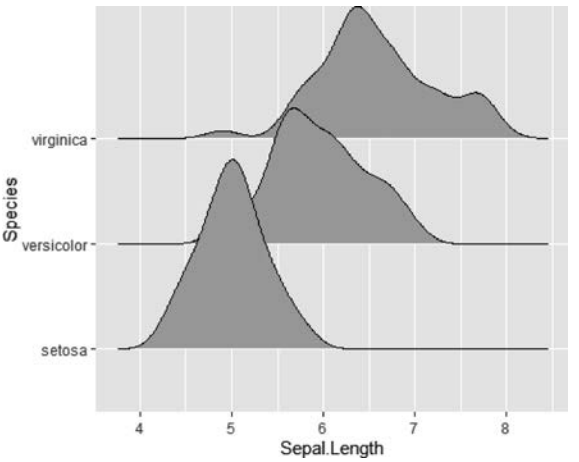

图 5-53 基础峰峦图

图 5-53 中各序列间有重叠，可以在 geom_density_ridges()中设置 scale 参数值的大小给予改变：值越大，序列间重叠部分就越大，值越小，重叠部分就越小。下面将 scale 参数设置为 1，序列间将无重叠，代码如下：

```
#代码 5-55 scale 参数的使用(1)
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species)) + geom_density_ridges(scale = 1)
##Picking joint bandwidth of 0.181
```

代码运行结果如图 5-54 所示。

下例中将 scale 参数增大，整个图形中序列重叠部分将增加，代码如下：

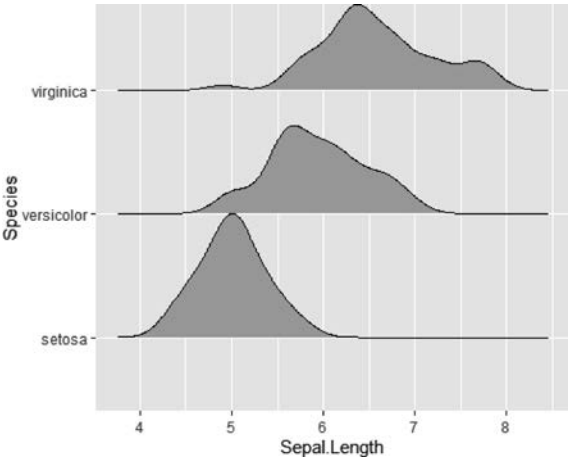

图 5-54 scale 参数的使用(1)

```
#代码 5-56 scale 参数的使用(2)
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species)) + geom_density_ridges(scale = 2)
##Picking joint bandwidth of 0.181
```

调整 scale 参数后,代码运行的结果如图 5-55 所示。

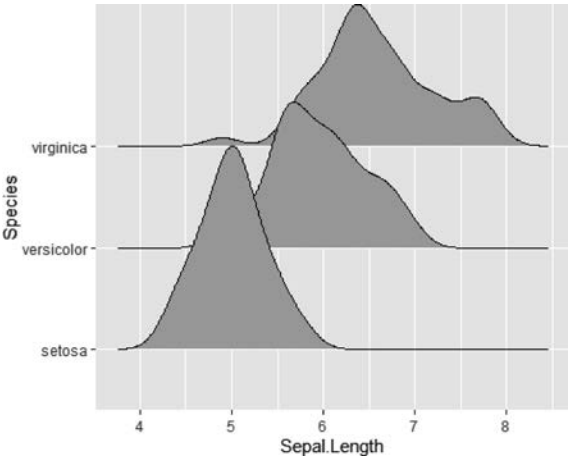

图 5-55 scale 参数的使用(2)

下例使用包自带的数据集 lincoln_weather 为数据源,将颜色映射到密度,即密度大小通过颜色给予区别。stat(x)对 x 做统计计算,并赋值给填充参数 fill,之后使用 geom_density_ridges_gradient()将颜色映射给密度,代码如下:

```
#代码 5-57 geom_density_ridges_gradient()的使用
library(ggplot2)
```

```
library(ggridges)
library(viridis)
ggplot(lincoln_weather, aes(x = `Mean Temperature [F]`, y = Month, fill = stat(x))) +
  geom_density_ridges_gradient(scale = 3, rel_min_height = 0.01) +
  scale_fill_viridis_c(name = "温度", option = "C") +
  labs(title = '2022 年每月温度',x = "温度",y = "月份")
# # Picking joint bandwidth of 3.37
```

代码运行的结果如图 5-56 所示。

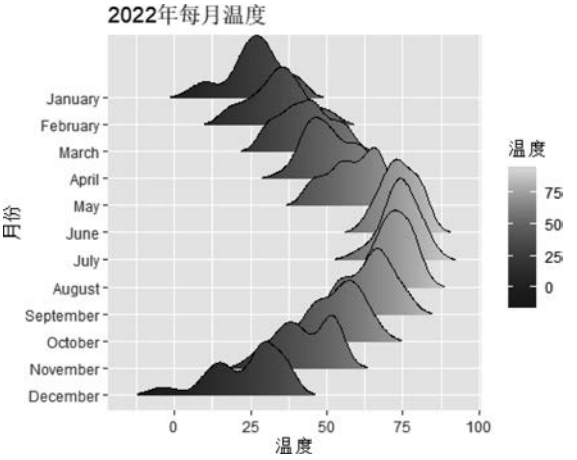

图 5-56　geom_density_ridges_gradient()的使用

在 stat_density_ridges()中可以使用 quantile_lines 添加分位数竖线，读者可以结合 quantile()函数理解，代码如下：

```
# 代码 5-58 添加分位数竖线
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species)) +
  stat_density_ridges(quantile_lines = TRUE)
# # Picking joint bandwidth of 0.181
```

代码运行的结果如图 5-57 所示。

上例中显示 3 根分位数竖线，也可以控制只显示其中某条分位数竖线。如下例中显示第 2 根分位数竖线，使用参数 quantiles = 2 即可，代码如下：

```
# 代码 5-59 添加多条分位数竖线
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species)) +
  stat_density_ridges(quantile_lines = TRUE, quantiles = 2)
# # Picking joint bandwidth of 0.181
```

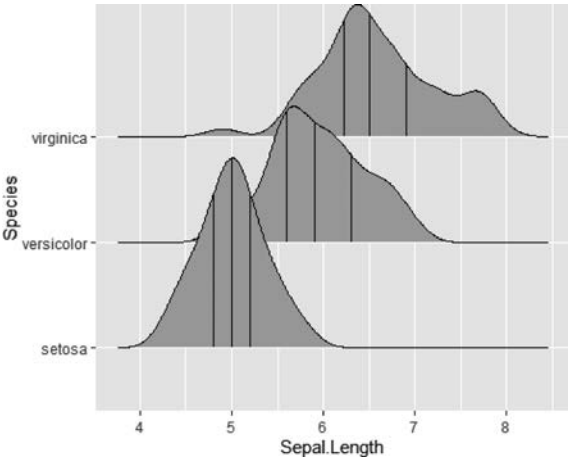

图 5-57　添加分位数竖线

使用参数 quantiles = 2 后运行代码，结果如图 5-58 所示。

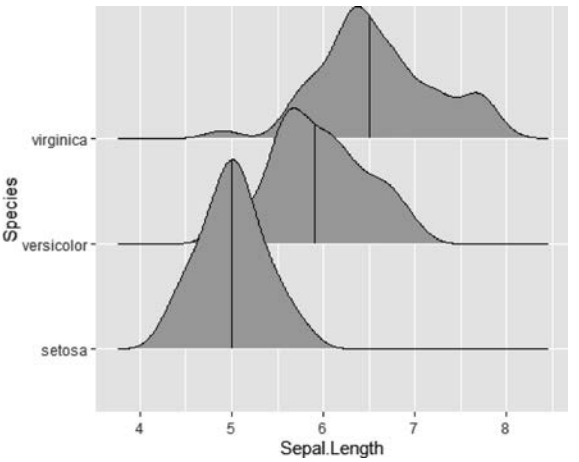

图 5-58　添加多条分位数竖线

quantiles 参数也可以输入向量，在向量中确定具体的分位数数值点。

如下例中显示 0.025 及 0.975 分位点，为了防止序列间遮盖，将填充的透明度使用 alpha 参数调整为 0.7，代码如下：

```
#代码 5-60 添加指定分位数的分位数竖线
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species)) +
  stat_density_ridges(quantile_lines = TRUE, quantiles = c(0.025, 0.975), alpha = 0.7)
##Picking joint bandwidth of 0.181
```

将 alpha 参数设置为 0.7，调整填充的透明度，改善序列间的遮盖问题，代码运行的结果如图 5-59 所示。

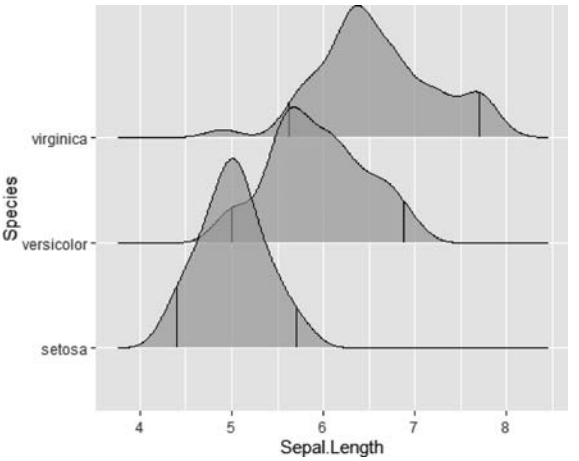

图 5-59　添加指定分位数的分位数竖线

上面用竖线标明了分位数，下面将不同分位区域填充为不同颜色。首先，在 aes()中确定 fill 参数为 factor(stat(quantile))，表示使用 stat 将分位数作为计算输入参数，factor()将其计算结果转换为因子。stat_density_ridges()中的 geom 参数表示几何对象，calc_ecdf 表示是否计算累计分布，quantiles 表示分位数个数，quantile_lines 表示是否显示分位数竖线，代码如下：

```
#代码 5-61 按照分位区域填充颜色
library(ggplot2)
library(ggridges)
ggplot(iris, aes(x = Sepal.Length, y = Species, fill = factor(stat(quantile)))) +
  stat_density_ridges(
    geom = "density_ridges_gradient", calc_ecdf = TRUE,
    quantiles = 4, quantile_lines = TRUE
  ) +
  scale_fill_viridis_d(name = "分位数")
##Picking joint bandwidth of 0.181
```

代码运行的结果如图 5-60 所示。

ggridges 包中还有 scale_fill_cyclical()可以将输入的颜色循环填充到序列。下例中使用自带的数据集 diamonds 绘制峰峦密度图，scale_fill_cyclical()将'blue'和 'green'两种颜色循环作为填充色间隔轮换填充到每个序列，代码如下：

```
#代码 5-62 使用 scale_fill_cyclical()
library(ggplot2)
library(ggridges)
ggplot(diamonds, aes(x = price, y = cut, fill = cut)) +
   geom_density_ridges(scale = 4, size = 1) +
   scale_fill_cyclical(
     name = "颜色",
     values = c("blue", "green"), guide = "legend")
##Picking joint bandwidth of 458
```

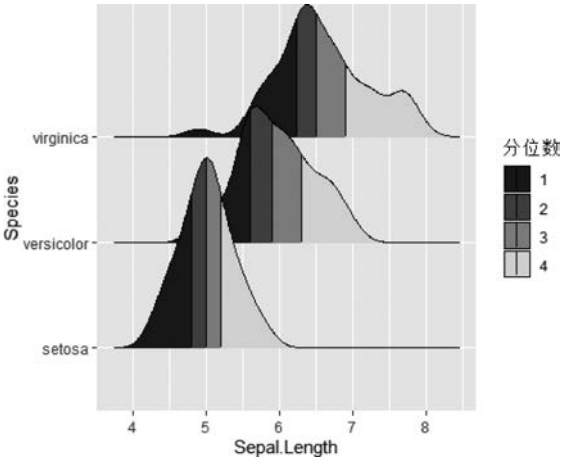

图 5-60　按照分位区域填充颜色

代码运行的结果如图 5-61 所示。

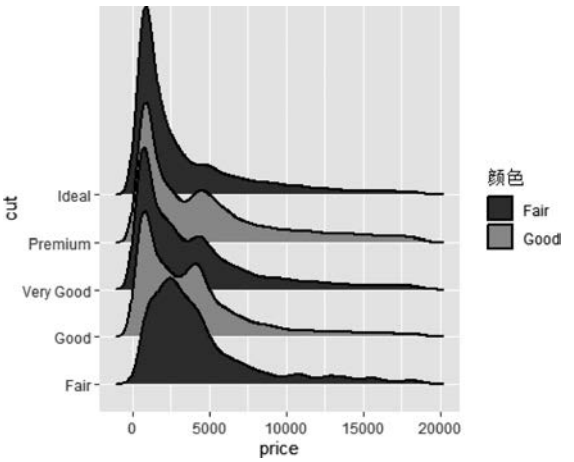

图 5-61　使用 scale_fill_cyclical()

5.13　ggmosaic 包介绍

ggmosaic 包可以绘制马赛克图，也就是不等宽柱状图。ggplot2 中通过 geom_segment() 函数也可以绘制马赛克图，不过首先需要构建数据源，通过 ggmosaic 包绘制则不需要。下面以泰坦尼克号(titanic)数据集为绘图数据，该数据集反映了在不同等级船舱、不同性别、成人或小孩最终是否存活的情况。product 参数中可以输入多个变量参数，以便反映这几个参数的交互情况，代码如下：

```
#代码 5-63 马赛克图
library(ggplot2)
```

```
library(ggmosaic)

ggplot(data = titanic) +
  geom_mosaic(aes(x = product(Class), fill = Survived)) +
  theme_mosaic()
## Warning: `unite_()` was deprecated in tidyr 1.2.0
## Please use `unite()` instead
## This warning is displayed once every 8 hours
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated
```

代码运行的结果如图 5-62 所示。

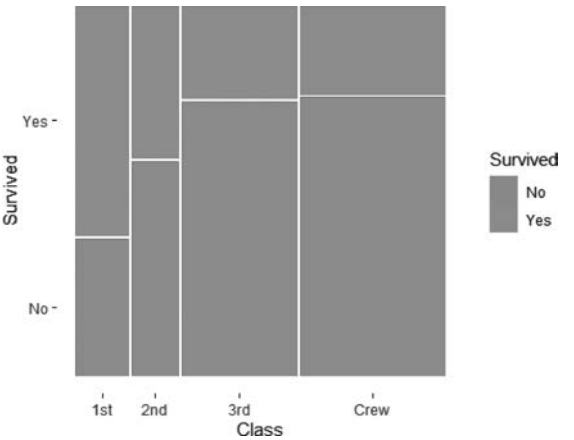

图 5-62 马赛克图

如果希望添加标签，则可以使用 geom_mosaic_text()实现，不过添加的标签其实就是将 x 轴和 y 轴标签合并显示，似乎会让图表变得凌乱。5.14 节将介绍的 ggcharts 包对某些 ggplot2 运用场景进行了简化封装，如棒棒糖图、哑铃图、分面条形图、金字塔图等。

5.14 ggcharts 包介绍

5.14.1 ggcharts 包对分面优化

下面是传统的 ggplot2 绘制分面条形图的数据准备过程：使用 filter 筛选出 3 年数据，按照 year 变量分组，使用 top_n()函数提取每组收入前 10 的数据，tidytext::reorder_within 对 company 变量排序以便图形按照从大到小的顺序显示，代码如下：

```
#代码 5-64 ggplot2 分面图
library(dplyr)
library(ggplot2)
library(ggcharts)
data("biomedicalrevenue")

biomedicalrevenue %>%
```

```
  filter(year %in% c(2012, 2015, 2018)) %>%
  group_by(year) %>%
  top_n(10, revenue) %>%
  ungroup() %>%
  mutate(company = tidytext::reorder_within(company, revenue, year)) %>%
  ggplot(aes(company, revenue)) +
  geom_col() +
  coord_flip() +
  tidytext::scale_x_reordered() +
  facet_wrap(vars(year), scales = "free_y")
```

使用 ggplot2 中的分面技术，结果如图 5-63 所示。

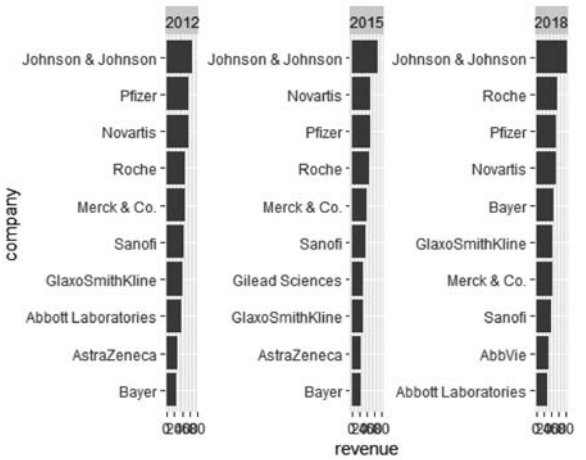

图 5-63　ggplot2 分面图

下面使用 ggcharts 包实现上例中的效果，大部分内容在 bar_chart 中实现，如 facet = year 实现按年分面，top_n = 10 表示显示前 10 的数据，与图 5-63 比较相当简约。唯一需要读者留意的是 x、y 还是按照 ggplot2 中的输入顺序来输入，这个有点让初次学习的读者感到疑惑，代码如下：

```
#代码 5-65 bar_chart 分面图
library(ggplot2)
library(ggcharts)
biomedicalrevenue %>%
  filter(year %in% c(2012, 2015, 2018)) %>%
  bar_chart(x = company, y = revenue, facet = year, top_n = 10)
```

使用 ggcharts 中的 bar_chart 效果比 ggplot2 的效果改善明显，本身由于排盘等截图效果不是太好，读者可以自行运行代码查看结果。代码运行的结果如图 5-64 所示。

图 5-64 由于导出后有些变形，看起来不理想，但在实际 R 环境下还是比较美观的。使用 bar_chart 的优点是可以省略坐标轴旋转等语句，并且可以筛选显示每组的 Top 值范围；缺点是语法上和 ggplot2 中的分面有所区别。

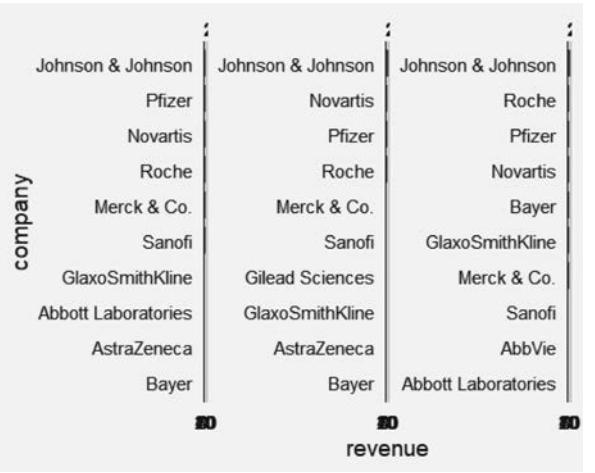

图 5-64　bar_chart 分面图

5.14.2　棒棒糖图

ggcharts 包中 lollipop_chart 可以直接绘制棒棒糖图。相比 ggplot2 中需要 geom_point()和 geom_segment()组合绘制棒棒糖图，lollipop_chart()简单许多。下例中展示不同公司收入金额的大小，x 轴是公司名称，y 轴是收入值大小，threshold 用于对收入值设定门槛筛选值。当然这里和上例一样，ggcharts 坐标默认做了转置，类似于 coord_flip()，因此 x、y 输入是按照转置前的输入方式。scale_y_continuous()中对标签做了自定义样式，expand 对坐标轴的最大值和最小值做了调整：从 0 点开始，最大值放大 0.05，即 5%，代码如下：

```
#代码 5-66 棒棒糖图
library(ggplot2)
library(ggcharts)
biomedicalrevenue %>%
  filter(year == 2018) %>%
  lollipop_chart(x = company, y = revenue, threshold = 30) +
  labs(
    x = NULL,
    y = "Revenue",
    title = "Biomedical Companies with Revenue > $30Bn."
  ) +
  scale_y_continuous(
    labels = function(x) paste0("$", x, "Bn."),
    expand = expansion(mult = c(0, 0.05))
  )
##Scale for 'y' is already present. Adding another scale for 'y', which will replace the
##existing scale
```

代码运行的结果如图 5-65 所示。

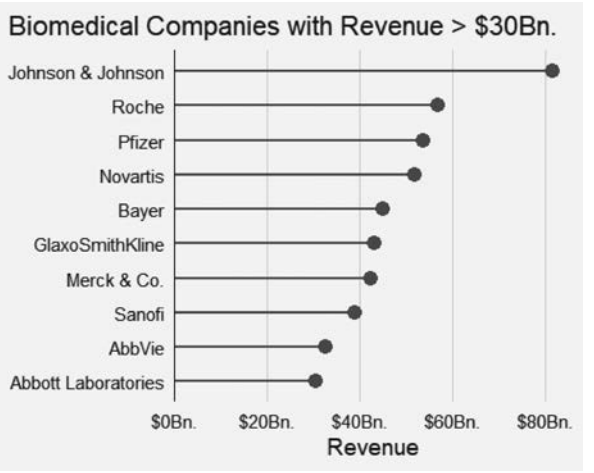

图 5-65 棒棒糖图

图 5-65 中通过 scale_y_continuous 对 x 轴格式做了设置：labels 后面使用了公式 expand = expansion(mult = c(0, 0.05))将 x 轴调整为从 0 开始并将最大值放大 5%。

5.14.3 哑铃图

使用 ggcharts 包中的 dumbbell_chart 可以绘制哑铃图。若要比较多序列 2 期数据的变化，则该图比其他图形更加简洁。下例中使用 dumbbell_chart()描绘了欧洲 1952 年与 2007 年人口的变化情况，通过 data 参数输入绘图数据集，x 轴输入国家，y1 和 y2 输入对比的两年的人数，top_n 用于筛选数据中的前 10，point_colors 对两个数据序列给予不同的颜色，代码如下：

```
#代码 5-67 哑铃图
library(ggplot2)
library(ggcharts)
data("popeurope")
dumbbell_chart(
  data = popeurope,
  x = country,
  y1 = pop1952,
  y2 = pop2007,
  top_n = 10,
  point_colors = c("lightgray", "#494F5C")
) +
  labs(
    x = NULL,
    y = "Population",
```

```
    title = "Europe's Largest Countries by Population in 2007"
  ) +
  scale_y_continuous(
    limits = c(0, NA),
    labels = function(x) paste(x, "Mn.")
  )
```

代码运行的结果如图 5-66 所示。

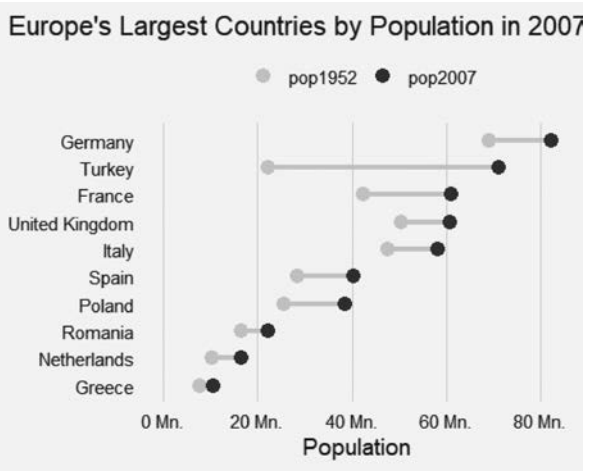

图 5-66 哑铃图

哑铃图使用 ggplot2 基础的绘图方式也可以实现，也就是使用 geom_point()结合 geome_segment()。基础绘图代码量会多些，但是语法统一，通过与颜色等其他图形属性结合，表现能力非常丰富。

5.14.4 正负值条形图

在 ggplot2 中绘制条形图时 y 轴标签均在左侧，当数值有正负数时坐标文本标签与柱子会重叠。diverging_bar_chart()将正负数对应的标签错位显示，解决了上述遮盖问题。下例中使用 mtcars 数据集，先将 mtcars 数据集中行名称赋值给 model 变量，将变量 hp 使用函数 scale()标准化，最后得到新的数据集 mtcars_z。在 diverging_bar_chart()中分别输入数据集 mtcars_z、x 及 y 对应的变量即可，代码如下：

```
#代码 5-68 正负值条形图
data(mtcars)
library(ggcharts)
mtcars_z <- dplyr::transmute(
  .data = mtcars,
  model = row.names(mtcars),
```

```
  hpz = scale(hp)
)

diverging_bar_chart(data = mtcars_z, x = model, y = hpz)
```

代码运行的结果如图 5-67 所示。

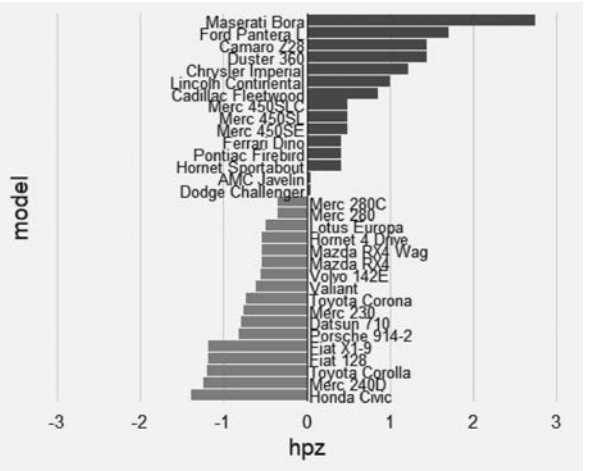

图 5-67 正负值条形图

图 5-67 优化了正负值条形图标签遮盖的问题，使用基础包中的 geome_text()结合计算位置点也可以对标签位置进行优化，有兴趣的读者可以自行尝试。

5.14.5 正负值棒棒糖图

除正负值条形图，ggcharts 包中的 diverging_lollipop_chart 可以绘制正负值棒棒糖图。仍使用上例中的 mtcars_z 数据集，diverging_lollipop_chart 的用法与 diverging_bar_chart 类似，lollipop_colors 用于设定棒棒糖图形颜色，text_color 用于设定标签颜色，代码如下：

```
#代码 5-69 正负值棒棒糖图
library(ggcharts)
diverging_lollipop_chart(
  data = mtcars_z,
  x = model,
  y = hpz,
  lollipop_colors = c("#006400", "#b32134"),
  text_color = c("#006400", "#b32134")
)
```

代码运行的结果如图 5-68 所示。

类似正负值条形图，正负值棒棒糖图标签也可以通过 geome_text()计算位置点做部分模拟，请读者思考具体的代码。

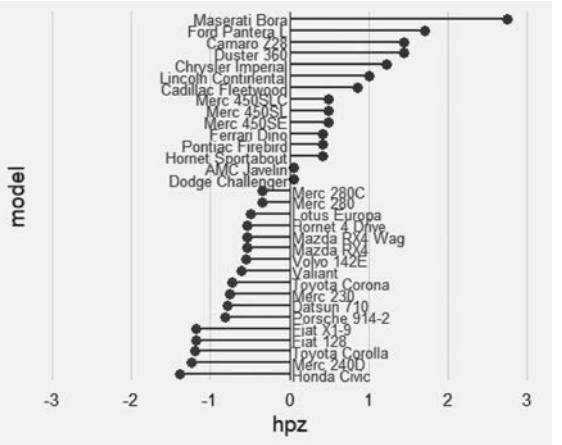

图 5-68　正负值棒棒糖图

5.14.6　金字塔图

pyramid_chart 的用法也非常简单，分别是数据集、x、y 及分组 group 参数，代码如下：

```
#代码 5-70 金字塔图
library(ggcharts)
data("popch")
pyramid_chart(data = popch, x = age, y = pop, group = sex)
##Warning: `expand_scale()` is deprecated; use `expansion()` instead

##Warning: `expand_scale()` is deprecated; use `expansion()` instead

##Warning: `expand_scale()` is deprecated; use `expansion()` instead

##Warning: `expand_scale()` is deprecated; use `expansion()` instead
```

代码运行的结果如图 5-69 所示。

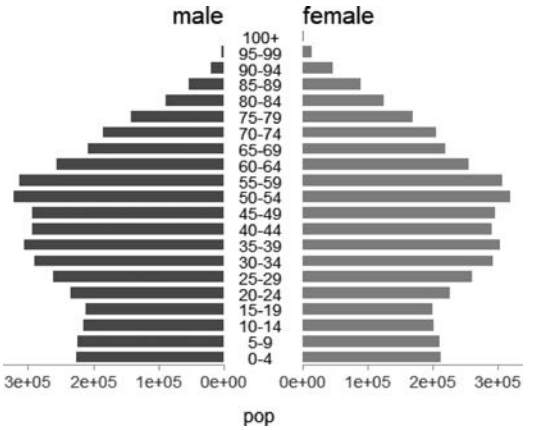

图 5-69　金字塔图

图 5-69 金字塔图表现两组数据的对比关系比较直观，唯一不足是比较的精确度不易观察出来，特别是两组数据差异不大的情况下。

5.15 patchwork 包介绍

对于需要一页多图的情况，首选的解决方案是通过原始数据计算，分组或离散化之后以分面的形式来完成，也就是使用前面 ggplot2 绘图包中的 facet_grid()或 facet_wrap()。分面快捷、简单，但是当需要对个性化图表进行拼接时就不太适合了，如想对散点图、条形图进行拼接，或者对某个子图占有空间进行灵活调整。patchwork 包最大的特点是代码简洁明了，使用加号、除号来表示横向及竖向拼接，通过 plot_layout()函数具体设置子图的位置等。下面先绘制散点图 p1 及箱线图 p2，通过加号直接水平拼接在一起，代码如下：

```
#代码 5-71 横向拼图
library(ggplot2)
library(patchwork)
p1 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))

p1 + p2
```

将散点图和箱线图横向拼接后的结果如图 5-70 所示。

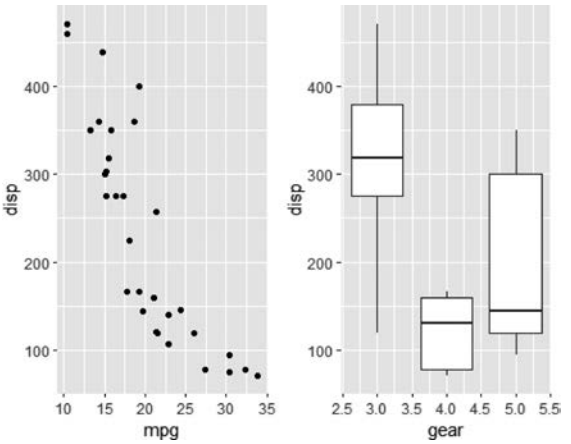

图 5-70 横向拼图

上面的加号也可以使用“|”替换。接下来将图 p1 和 p2 竖向拼接在一起，方法也特别简单，使用除号连接两个图形对象即可，代码如下：

```
#代码 5-72 竖向拼图
library(ggplot2)
library(patchwork)
```

```
p1 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))

p1/p2
```

竖向拼图的结果如图 5-71 所示。

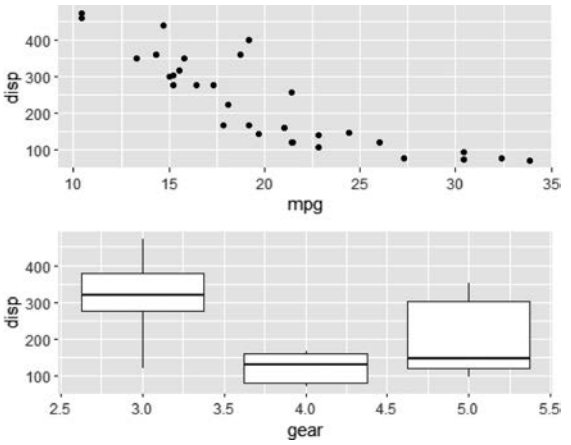

图 5-71　竖向拼图

更复杂的拼图可以结合分组实现，patchwork 包中使用小括号将括号内的内容视作一组，类似于一个图形对象的使用。下面先绘制 4 幅图，将其中的 1～3 幅图水平拼接，之后与第 4 幅图垂直拼接，代码如下：

```
#代码 5-73 复杂拼图
library(ggplot2)
library(patchwork)
p1 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))
p3 <- ggplot(mtcars) + geom_bar(aes(gear)) + facet_wrap(~cyl)
p4 <- ggplot(mtcars) + geom_bar(aes(carb))
(p1 | p2 | p3) /
      p4
```

复杂拼图的结果如图 5-72 所示。

上面使用加号和除号拼接图形，也可以在使用加号之后通过 plot_layout()中的 design 参数的设置来控制图形的位置，代码如下：

```
#代码 5-74 plot_layout 设置拼图
library(ggplot2)
library(patchwork)
p1 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))
```

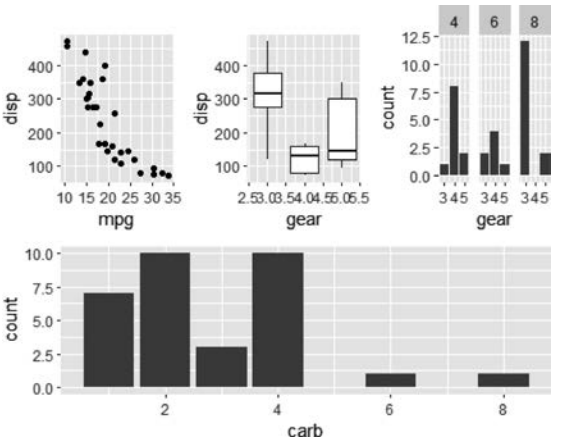

图 5-72　复杂拼图

```
p3 <- ggplot(mtcars) + geom_bar(aes(gear)) + facet_wrap(~cyl)
p4 <- ggplot(mtcars) + geom_bar(aes(carb))
layout <- '123
           444'
p1 + p2 + p3 + p4 + plot_layout(design = layout)
```

使用 plot_layout()实现复杂拼图的结果如图 5-73 所示。

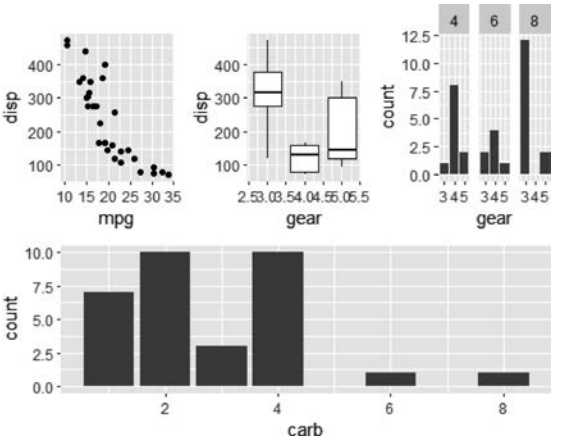

图 5-73　plot_layout 设置拼图

上例中 plot_layout(design＝layout)的 design 参数值为自定义的输入值 layout，1234 分别代表待拼接的图形 p1～p4，其中 123 在第 1 行，表示 p1～p3 水平横向依次排在一起，444 代表 p4 在第 2 行且占满全行。这输入方式非常直观，也非常符合人类的直观理解。当然如果想更加专业，则可以将输入值都放到一行里，即用'123\n444 '替代。在 plot_layout()中可以通过 ncol 参数设定列数，通过 nrow 参数设定行数。接下来介绍插入式拼图，结果类似于现在许多 App 在主页上可以增加一个小浮窗来呈现不同的内容或者类似于熟悉的中

国地图在右下角会增加一个子图以展示南海地区。在 patchwork 包中使用 inset_element() 函数实现此功能，代码如下：

```
#代码 5-75 子母图
library(ggplot2)
library(patchwork)
p1 <- ggplot(mtcars) + geom_point(aes(mpg, disp))
p2 <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))
p1 + inset_element(p2,left = 0.5,bottom = 0.5,right = 1,top = 1)
```

代码运行的结果如图 5-74 所示。

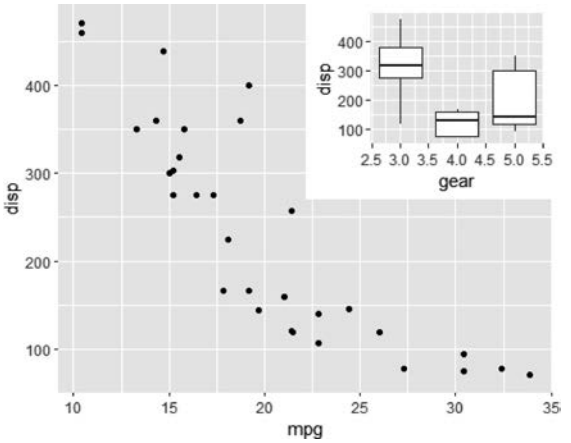

图 5-74　子母图

上面的方法可以归纳为主图＋inset_element(子图，子图左边缘位置，子图下边缘位置，子图右边缘位置、子图上边缘位置)。边缘位置采用的是相对于主图的相对值，即 0 代表主图左下角的位置，1 代表主图右上角的位置，本例中的值代表子图左下角从主图的中间位置开始，在主图的最右上角结束。可以直观类比理解 Windows 系统中许多软件的操作：将子图左下角放到中间点，之后拖动右上角将其放大到窗口右上角。

5.16　绘图相关的其他包介绍

前面介绍的 ggplot2 系列包绘制的图形属于静态图形，随着互联网上各种动态网页的发展，动态可视化库不断涌现。D3.js 就是现在非常流行的可视化库，可以绘制各种动态网页图形，如果读者有这方面的基础，则可以直接在 rmarkdown 中调用 D3.js。对于没有 JavaScript 及 D3.js 库知识的 R 语言使用者来讲，networkD3 包是一个不错的选择。R 语言中 networkD3 包基于 D3.js 库构建，虽然不可能涵盖 D3 库的所有内容，但可以通过它在 R 环境下绘制动态的网络图、树图、谱系图、桑基图等。下面介绍 radialNetwork 图形，使用基础包 datasets 中的 USArrests 为原始数据绘图，该数据反映了美国各州特定犯罪指标情况。

首先使用 dist()函数将数据间的距离计算出来，之后使用 hclust()函数对数据进行聚类。as.radialNetwork()函数对聚类结果格式进行转换，最终将数据传递给 radialNetwork()函数，详细用法见下面的例子，代码如下：

```
#代码 5-76 放射状网络图
library(networkD3)
find("USArrests")
hc <- hclust(dist(USArrests), "ave")
radialNetwork(as.radialNetwork(hc))
```

代码运行的结果如图 5-75 所示。

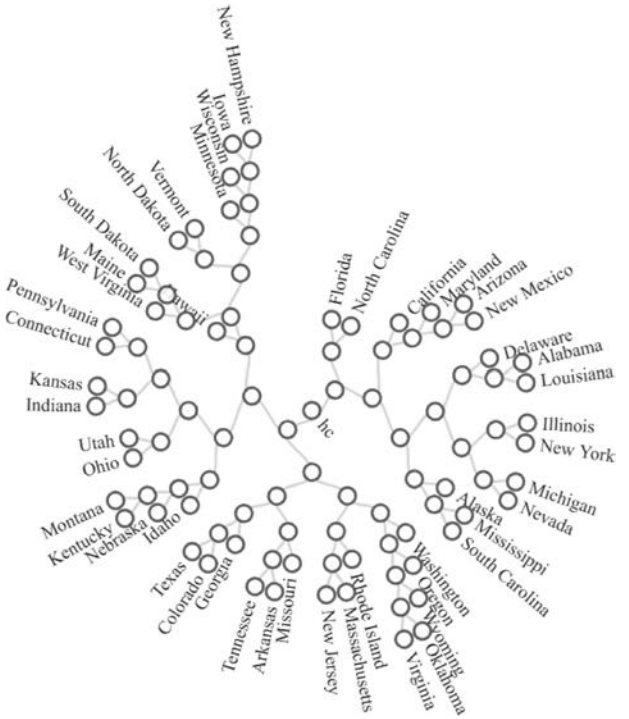

图 5-75　放射状网络图

chordNetwork()函数用于绘制弦图，dendroNetwork()函数用于绘制谱系图，diagonalNetwork()函数用于绘制水平谱系图，sankeyNetwork()函数用于绘制桑基图。前面几种类型图相对简单，但实用性不是特别强，下面介绍桑基图的绘制，详细用法见下面的例子，代码如下：

```
#代码 5-77 基础桑基图
library(networkD3)
library(tidyverse,quietly = TRUE)
source <- c("电视","欧洲","欧洲","欧洲","欧洲","电视","美国","美国","美国","美国","计
算机","计算机","计算机")
```

```
target <- c("欧洲","德国","法国","英国","意大利","美国","加州","纽约","佛罗里达","其他
州","德国","美国","法国")
value <- c(100,52,32,11,5,230,120,43,46,21,56,32,67)
sankey_data <- data.frame(source,target,value)
node <- data.frame(name = c(sankey_data $ source,sankey_data $ target)) %>% unique()
sankey_data $ sourceID <- match(sankey_data $ source,node $ name) - 1
sankey_data $ targetID <- match(sankey_data $ target,node $ name) - 1
sankeyNetwork(Links = sankey_data,Source = 'sourceID',
              Target = 'targetID',Nodes = node,
              NodeID = 'name',LinkGroup = 'source',Value = 'value',fontSize = 12)
```

构建数据源及节点等图形所需参数。代码运行的结果如图 5-76 所示。

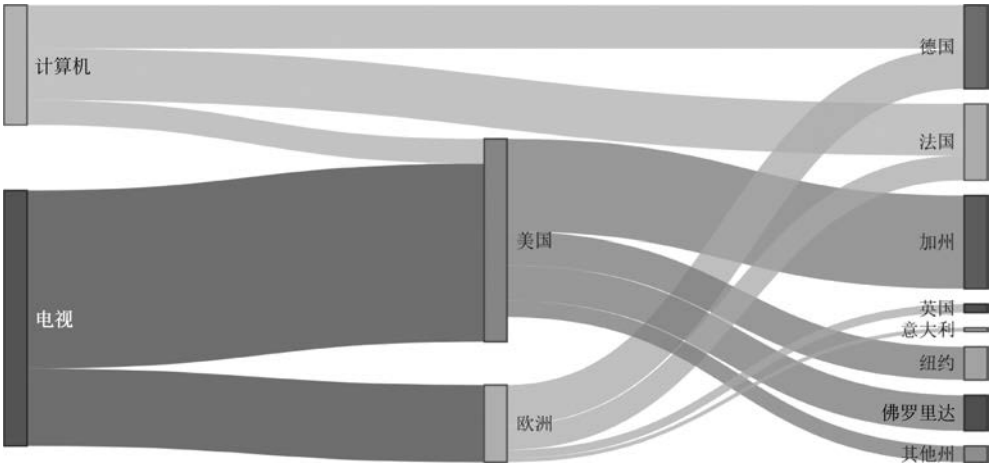

图 5-76　基础桑基图

首先准备数据源，图 5-76 中首先生成 source、target、value 3 个向量，之后生成数据框 sankey_data，也可以从外部导入。该数据源代表电视、计算机各地域销售结构情况，数据是一个混合结构的。第 1 行是电视在欧洲的销售值，2～4 行是电视在欧洲各个国家的销售情况，逻辑上 2～4 行数据是第 1 行的下一层级数据，但是这里都放到了一起。电视在美国的销售数据结构与此类似，首先是电视在美国的总销售情况，接下来是电视在美国各州的销售情况。绘制桑基图的重要基础就是构建数据结构，本例中省略了这个步骤，实际遇到的数据结构不会这样，需要读者自行处理。sankey_data 详细数据结构参考下面的例子，代码如下：

```
#代码 5 - 78 桑基图数据源
library(networkD3)
library(tidyverse,quietly = TRUE)
source <- c("电视","欧洲","欧洲","欧洲","欧洲","电视","美国","美国","美国","美国","计
算机","计算机","计算机")
target <- c("欧洲","德国","法国","英国","意大利","美国","加州","纽约","佛罗里达","其他
州","德国","美国","法国")
value <- c(100,52,32,11,5,230,120,43,46,21,56,32,67)
(sankey_data <- data.frame(source,target,value))
```

随后通过 sankey_data 将其中的起点 source 和终点 target 文字提取出来，unique()函数去除重复项，生成数据框 node 作为桑基图的节点标签。在数据框 sankey_data 中添加 sourceID、targetID，这两个变量分别以 source、target 在数据框 node 中的顺序号减 1 得到，相当于 sankey_data 与 node 连接在一起。最终调用 sankeyNetwork()函数绘制桑基图。

桑基图可以进一步美化，colourScale 用于设置填充颜色，nodeWidth 用于设置节点柱子的宽度，nodePadding 用于设置节点柱子间空白区域的大小，详细用法见下面的例子，代码如下：

```
#代码 5-79 桑基图格式调整
library(networkD3)
library(tidyverse,quietly = TRUE)
source <- c("电视","欧洲","欧洲","欧洲","欧洲","电视","美国","美国","美国","美国","计
算机","计算机","计算机")
target <- c("欧洲","德国","法国","英国","意大利","美国","加州","纽约","佛罗里达","其他
州","德国","美国","法国")
value <- c(100,52,32,11,5,230,120,43,46,21,56,32,67)
sankey_data <- data.frame(source,target,value)
node <- data.frame(name = c(sankey_data$source,sankey_data$target)) %>% unique()
sankey_data$sourceID <- match(sankey_data$source,node$name) - 1
sankey_data$targetID <- match(sankey_data$target,node$name) - 1
sankeyNetwork(Links = sankey_data,Source = 'sourceID',
              Target = 'targetID',Nodes = node,
              NodeID = 'name',LinkGroup = 'source',Value = 'value',fontSize = 16,
              colourScale = JS("d3.scaleOrdinal(d3.schemeCategory20);"),
              nodeWidth = 40,
              nodePadding = 15)
```

优化后的桑基图如图 5-77 所示。

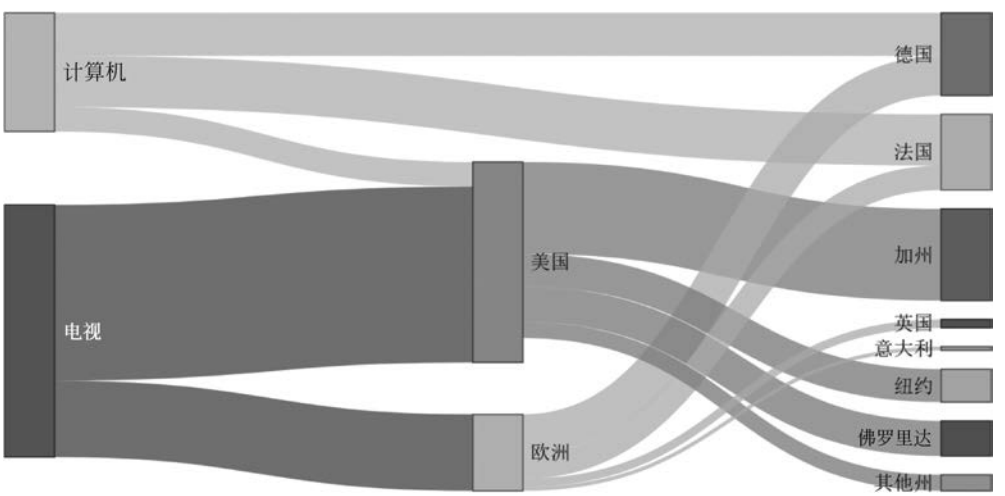

图 5-77 桑基图格式调整