

Lab 5：可视化图表进阶

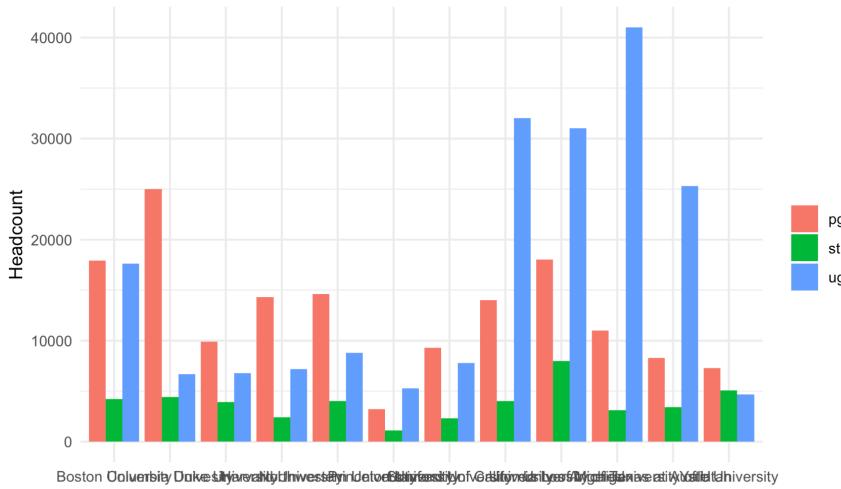
本次练习的主要内容为对可视化图表绘制的细节提升。

例 1：绘制多数据系列及堆积柱形图

1. 导入 `universities.csv` (美国部分大学数据)。该数据有 5 列，分别为 `university` (大学名称)、`ug` (本科生人数)、`pg` (研究生人数)、`st` (教职员人数) 和 `type` (公立或私立)。现在我们根据本科生、研究生和教职工的人数，绘制一幅多数据系列柱形图。绘制多数据系列柱形图时，最好的做法是先将二维表转化为一维表。

```
library(ggplot2)
library(readr)
library(dplyr)
library(tidyr)
#导入数据
univ <- read_csv("data/universities.csv")
# 把二维表降维成一维表
data.plot <- univ %>% pivot_longer(cols = c("ug", "pg", "st"),
                                         names_to = "Variable",
                                         values_to = "Values")
#画图
p <- ggplot(data.plot, aes(x=university, y=Values, fill=Variable)) +
  geom_bar(stat="identity", position="dodge") +
  theme_minimal() +
  labs(x="", y="Headcount", fill="")
p
```

输出效果：



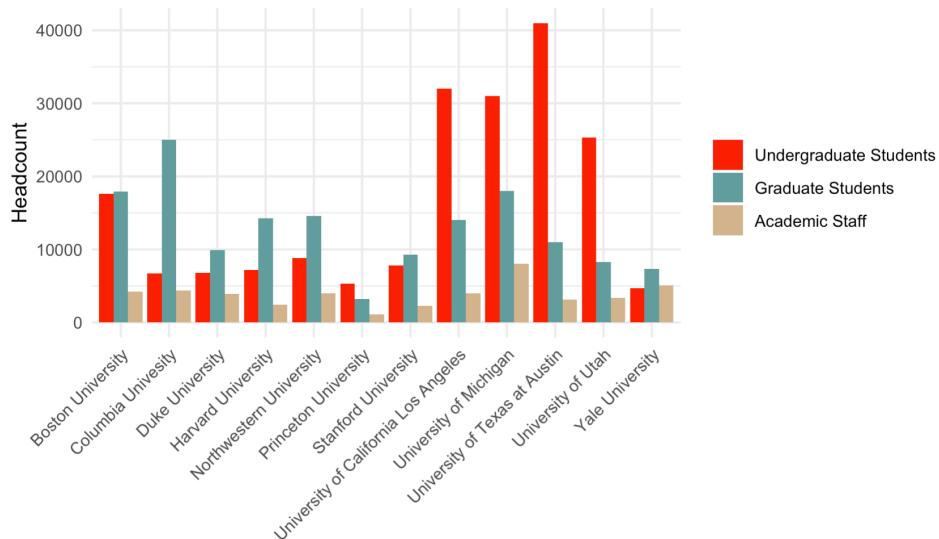
这幅图的主要问题有以下几个：

- 横轴的标注互相重叠，无法阅读。
- 每组内绘制顺序为研究生-教职员-本科生，不符合常规逻辑。
- 图例中的标注 (`pg`、`st`、`ug`) 意义不足，无法让不了解原数据的读者看懂。

2. 针对上一步里出现的三个主要问题进行修改：

```
#改变绘制顺序为本科生、研究生、教职工
data.plot$Variable <- factor(data.plot$Variable, levels=c("ug", "pg", "st"))
#画图
p <- ggplot(data.plot, aes(x=university, y=Values, fill=Variable)) +
  geom_bar(stat="identity", position="dodge") +
  theme_minimal() +
  labs(x="", y="Headcount", fill="") +
  #让x轴的标注倾斜
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  #改变图例中的标注
  scale_fill_manual(name = "",
                    breaks=c("ug", "pg", "st"),
                    values = c("ug"="red",
                               "pg"="cadetblue",
                               "st"="tan"),
                    labels=c("Undergraduate Students",
                            "Graduate Students",
                            "Academic Staff"))
p
```

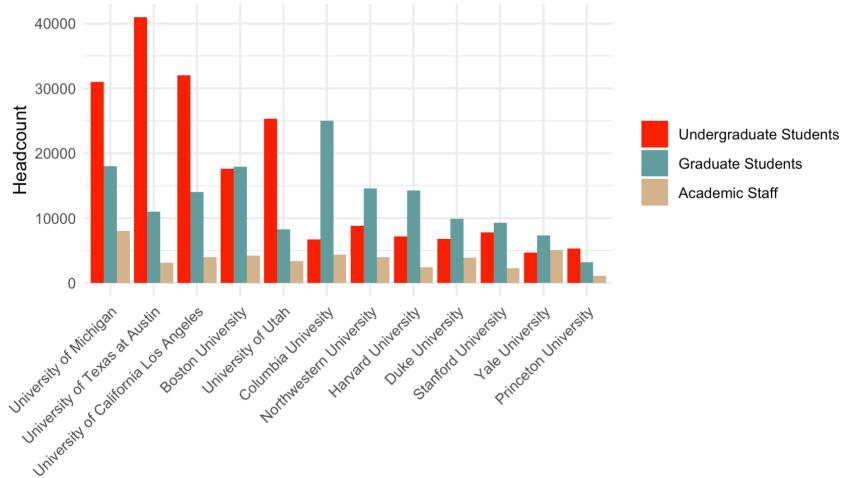
输出效果：



3. 现在，我们希望按照各校的总人数来安排大学在 x 轴上的位置。此时可以在 `aes()` 中调用 `reorder()` 函数进行排序。因为我们希望将人数最多的学校放在横轴左侧，因此要使用降序，在 `Values` 前加负号。

```
#按照学校的总人数排序
p <- ggplot(data.plot, aes(x=reorder(university,-Values), y=Values, fill=Variable)) +
  geom_bar(stat="identity", position="dodge") +
  theme_minimal() +
  labs(x="", y="Headcount", fill="") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)) +
  scale_fill_manual(name = "",
                    breaks=c("ug", "pg", "st"),
                    values = c("ug"="red",
                               "pg"="cadetblue",
                               "st"="tan"),
                    labels=c("Undergraduate Students",
                            "Graduate Students",
                            "Academic Staff"))
p
```

输出效果：



4. 如果我们不按总人数排名，而只按其中的本科生人数排名，则需要先进行数据预处理，通过 R 中的“`factor`”数据结构，指定绘制的层级。

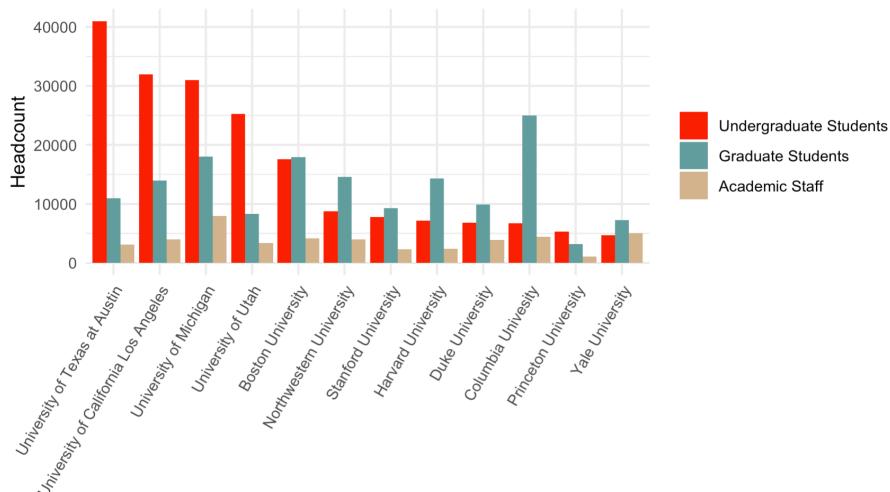
```
#将data.plot按本科生人数值降序排列，并按所需顺序获得大学的排序层级，组成临时数据集
ordered_data.plot <- data.plot %>%
  filter(Variable == "ug") %>%
  arrange(desc(Values)) %>% #如果要升序绘制，这里就不加desc()
  pull(university)

#根据上一步的临时数据集，改变data.plot的绘制级别
data.plot$university <- factor(data.plot$university, levels=ordered_data.plot)

#画图
p <- ggplot(data.plot, aes(x=university, y=Values, fill=Variable)) +
  geom_bar(stat="identity", position="dodge") +
  theme_minimal() +
  labs(x="", y="Headcount", fill="") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1)) +
  scale_fill_manual(name = "",
                    breaks=c("ug", "pg", "st"),
                    values = c("ug"="red",
                              "pg"="cadetblue",
                              "st"="tan"),
                    labels=c("Undergraduate Students",
                            "Graduate Students",
                            "Academic Staff"))

p
```

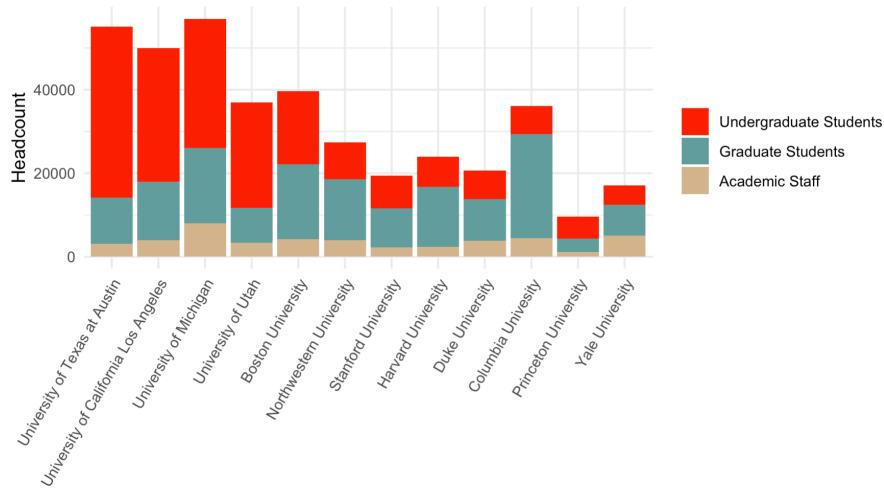
输出效果：



5. 将多数据系列柱形图改为堆积柱形图，只需要去掉 `position="dodge"`。

```
p <- ggplot(data.plot, aes(x=university, y=Values, fill=Variable)) +
  geom_bar(stat="identity") +
  theme_minimal() +
  labs(x="", y="Headcount", fill="") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1)) +
  scale_fill_manual(name = "",
                    breaks=c("ug", "pg", "st"),
                    values = c("ug"="red",
                               "pg"="cadetblue",
                               "st"="tan"),
                    labels=c("Undergraduate Students",
                            "Graduate Students",
                            "Academic Staff"))
p
```

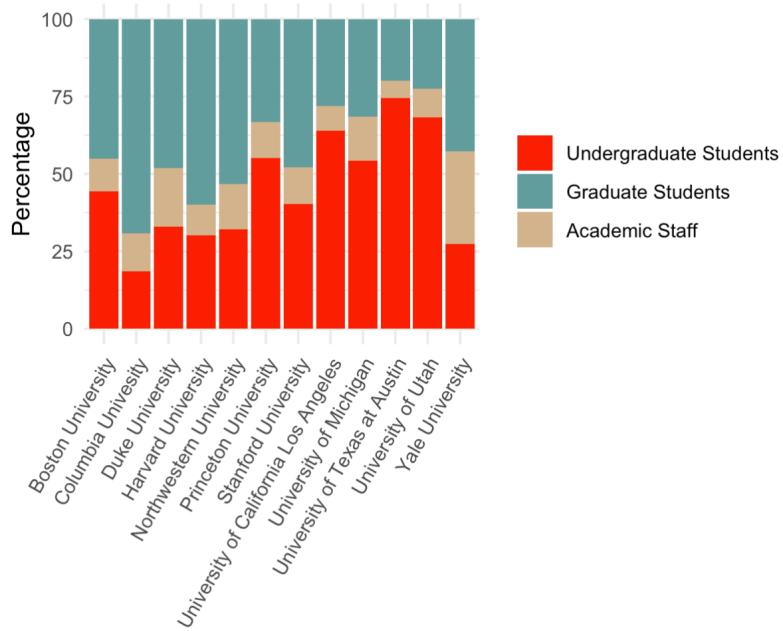
输出效果：



6. 如果绘制百分比堆积柱形图，需要在数据预处理阶段计算好比例，然后绘制。

```
univ <- read_csv("data/universities.csv")
data.plot <- univ %>% pivot_longer(cols = c("ug", "pg", "st"),
                                         names_to = "Variable",
                                         values_to = "Values")
total_values <- aggregate(Values ~ university, data.plot, sum) # 计算每个大学的总人数
names(total_values)[2] <- "Total"
data.plot <- merge(data.plot, total_values, by="university") # 和data.plot合并
data.plot$Proportion <- data.plot$Values / data.plot$Total * 100 # 计算百分比
#画图
p <- ggplot(data.plot, aes(x=university, y=Proportion, fill=Variable)) +
  geom_bar(stat="identity") +
  theme_minimal() +
  labs(x="", y="Percentage", fill="") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1, vjust = 1)) +
  scale_fill_manual(name = "",
                    breaks=c("ug", "pg", "st"),
                    values = c("ug"="red",
                               "pg"="cadetblue",
                               "st"="tan"),
                    labels=c("Undergraduate Students",
                            "Graduate Students",
                            "Academic Staff"))
p
```

输出结果：



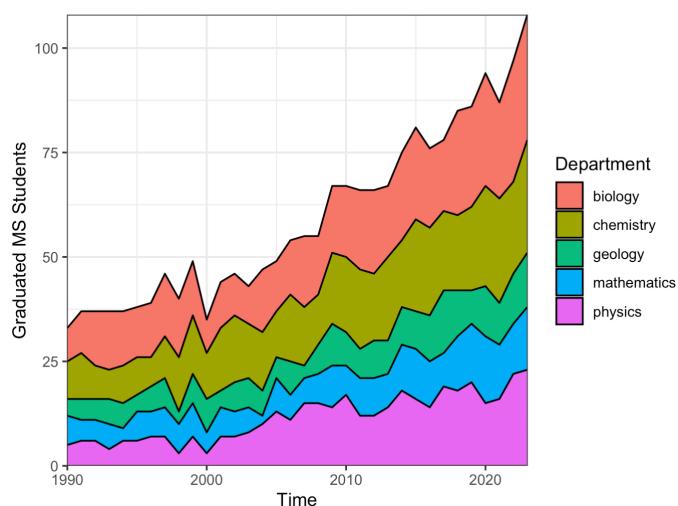
例 2：时间序列——堆积面积图

1. 导入 `grads.csv`, 这是某校自然科学学院历年的硕士生毕业人数数据, 分为 6 列, 分别为 `year` (年份) 以及五个学科 (`mathematics`, `physics`, `chemistry`, `biology` 和 `geology`)。绘制一幅堆积面积图。

```
#导入数据
grads <- read_csv("data/grads.csv")
#二维表降维成一维表
data <- grads %>% pivot_longer(cols = c("mathematics","physics","chemistry",
                                             "geology","biology"),
                                    names_to = "subject",
                                    values_to = "graduates")

#绘图
p <- ggplot() +
  geom_area(data=data, aes(x=year, y=graduates, fill=subject),color="black",position="stack")+
  labs(x="Time",y="Graduated MS Students",fill="Department")+
  theme_bw()+
  scale_x_continuous(expand=c(0,0))+ #expand参数设为0, 能让绘图区域抵达坐标轴
  scale_y_continuous(expand=c(0,0))
p
```

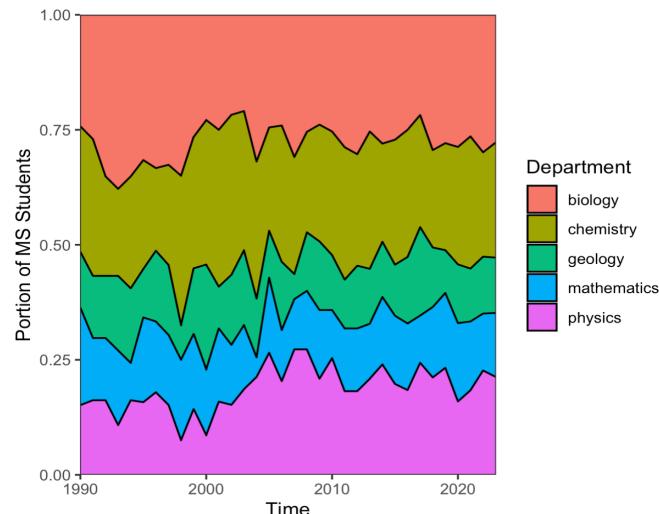
输出效果:



2. 绘制百分比堆积面积图，只需要把 position 的参数换为 “fill”。

```
p <- ggplot() +
  geom_area(data=data, aes(x=year, y=graduates, fill=subject), color="black", position="fill")+
  labs(x="Time", y="Portion of MS Students", fill="Department")+
  theme_bw()+
  scale_x_continuous(expand=c(0,0))+
  scale_y_continuous(expand=c(0,0))
p
```

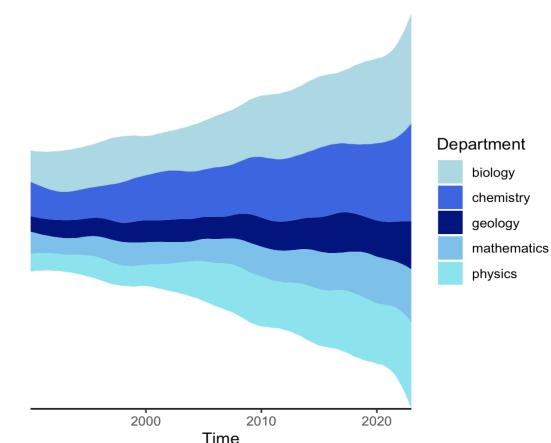
输出效果：



3. 绘制河流图，需要安装并导入扩展包 `ggstream`。

```
library(ggstream)
p <- ggplot() +
  geom_stream(data=data, aes(x=year, y=graduates, fill=subject)) +
  theme_classic()+
  scale_fill_manual(values = c("biology"="lightblue",
                               "chemistry"="royalblue",
                               "geology"="navy",
                               "mathematics"="skyblue2",
                               "physics"="cadetblue2"))+
  labs(x="Time", y="", fill="Department")+
  theme(panel.grid = element_blank(),
        axis.line.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank())+
  scale_x_continuous(expand=c(0,0))+
  scale_y_continuous(expand=c(0,0))
p
```

输出效果：



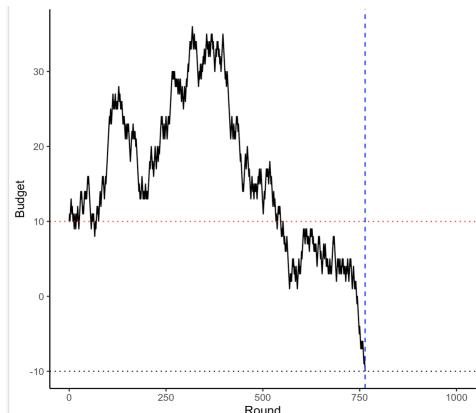
例 3：绘制蒙特卡洛模拟

时间序列的蒙特卡洛模拟需要进行大量的反复测试，每一次都能得到一条不同的时间序列图。要将这些测试结果绘制在图表中，需要将 `ggplot` 和循环语句相结合。现在我们模拟一个蒙特卡洛赌场的游戏：玩家和庄家对赌抛硬币，硬币是均匀的，正反面朝上的概率一样。如果正面朝上，则本轮玩家赢，庄家付给玩家 1 元；如果反面朝上，则本轮庄家赢，玩家付给庄家 1 元。假设玩家的资金一共有 10 元，且如果资金低于 -10，则玩家破产，游戏结束，并在图像上标出结束点。我们希望分析 1000 轮后玩家手中预期还有多少资金。

1. 先进行一次模拟并画图：

```
# 创建一个数据框来承接结果，每行代表1轮游戏
res <- data.frame(round = seq(1,1000,1))
# 玩家的资金
budget = 10
# 数据库新增列，来承接模拟结果
res$simu = NA
# 轮数
round=1
end=1000
# 开始模拟
while(round <= 1000){
  # 抛硬币（在0-1之间随机生成数据，0.5以上为正面，否则为反面）
  toss <- runif(1)
  if(toss < 0.5){
    budget <- budget + 1
  } else {
    budget <- budget - 1
  }
  # 判断游戏是否结束
  # 资金没归零，不结束：
  if(budget >-10){
    res$simu[round] <- budget # 更新数据框中该轮的资金
    round <- round + 1
  }
  # 如果资金归零，结束游戏
  if(budget <= -10){
    res$simu[round] <- -10
    end = round
    round <- 1001
  }
  cat("Tossing round",round,"\\n")
}
# 添加初始状态
res = rbind(c(0,10),res)
p <- ggplot()+
  geom_line(data=res, aes(x=round,y=simu))+ 
  theme_classic()+
  geom_vline(xintercept=end,linetype="dashed",color="blue")+
  geom_hline(yintercept=-10, linetype="dotted")+
  geom_hline(yintercept=10, linetype="dotted",color="red")+
  labs(x="Round",y="Budget")
```

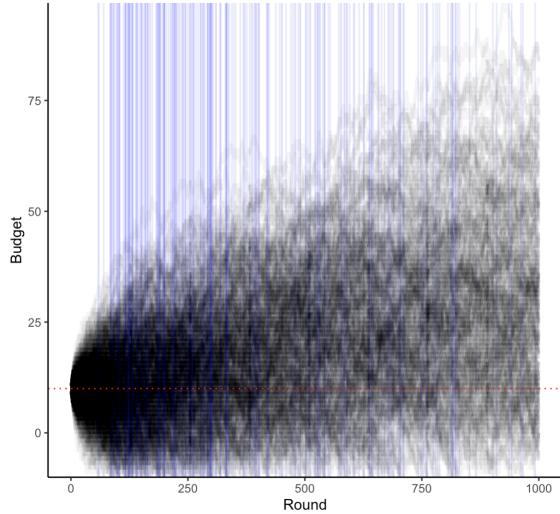
输出结果：



2. 加入蒙特卡洛模拟。我们将游戏进行 500 次，并记录下每一次的结果，绘制在图中。此时，我们需要把上一步中的 `while loop` 放入一个循环 500 次的 `for loop` 中。在 `for loop` 之外，首先要将 `ggplot` 的对象初始化。每进行一次游戏，就往 `ggplot` 对象上添加一个图层。

```
res <- data.frame(round = seq(1,1000,1))
p <- ggplot()
#500次蒙特卡洛模拟
for(i in 1:500){
  #每一次模拟，在res上新建一列，来承接结果。新建时可命名为“temp”即临时名称
  res$temp = NA
  #进行模拟
  budget = 10
  round = 1
  while(round <= 1000){
    toss <- runif(1)
    if(toss < 0.5){
      budget <- budget + 1
    } else {
      budget <- budget - 1
    }
    if(budget >= 10){
      res$temp[round] <- budget
      round <- round + 1
    }
    if(budget <= -10){
      res$temp[round] <- -10
      #添加游戏结束标记
      p <- p + geom_vline(xintercept=round,color="blue", alpha = 0.1, linewidth=0.5)
      round <- 1001
    }
  }
  #添加图层
  p <- p + geom_line(data=res, aes(x=round,y=temp), alpha = 0.05, linewidth=1)
  #将“temp”改名
  colnames(res)[ncol(res)] <- paste("simu-",toString(i),sep="")
  cat("Simulating episode",i,"\n")
}
#调整图像
p <- p + theme_classic()+
  geom_hline(yintercept=10, linetype="dotted",color="red")+
  labs(x="Round",y="Budget")+
  scale_y_continuous(expand=c(0,0))
p
```

输出效果：



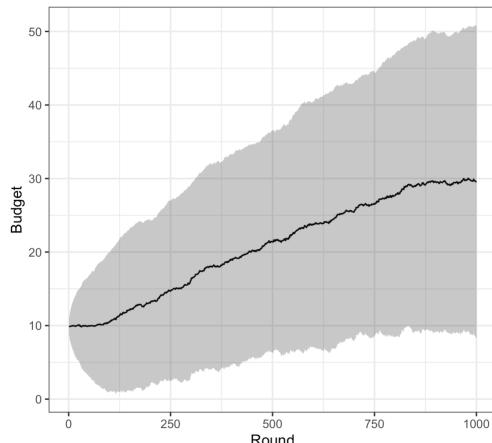
3. 接下来，我们利用上一步得到的 `res` 数据框，计算 500 次模拟中，每一轮后玩家剩下的平均资金和标准差，以及有多少次模拟中游戏尚未结束。

```
#计算第2-501列的平均值、标准差（第1列为轮次）
res$means <- apply(res[,2:501], 1, function(x) mean(x, na.rm = TRUE))
res$sd <- apply(res[,2:501], 1, function(x) sd(x, na.rm = TRUE))
#计算第2-501列的非NA次数（即游戏尚未结束的次数）
res$alive <- apply(res[,2:501], 1, function(x) sum(!is.na(x)))
```

4. 绘制 500 次模拟后，每轮的均值和标准差。标准差通常用 `geom_ribbon` 绘制。

```
p <- ggplot()+
  geom_line(data=res,aes(x=round,y=means))+ 
  geom_ribbon(data=res,aes(x=round,ymin=means-sd,ymax=means+sd),alpha=0.3)+ 
  theme_bw()+
  labs(x="Round", y="Budget")
```

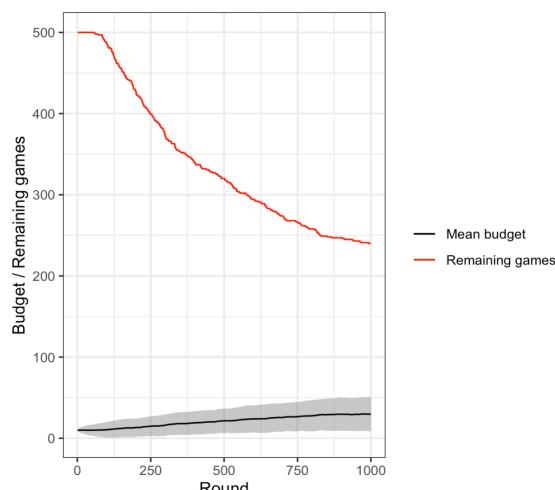
输出效果：



5. 看起来随着轮次的增加，玩家的资金是在不断增加的。但是实际上，这是博彩业误导性宣传的一种常见手段，因为在这张图里，玩家破产的情况并没有被纳入计算。我们可以把每轮后仍在进行的游戏数量绘制在同一张图上进行对比，容易发现轮数越多，剩余的游戏就越少，即玩家“输光”后退出游戏的情况越多。

```
p <- ggplot()+
  geom_line(data=res,aes(x=round,y=means,color="Mean budget"))+ 
  geom_ribbon(data=res,aes(x=round,ymin=means-sd,ymax=means+sd),alpha=0.3)+ 
  geom_line(data=res,aes(x=round,y=alive,color="Remaining games"))+ 
  theme_bw()+
  scale_color_manual(values=c(
    "Mean budget" = "black",
    "Remaining games" = "red"
  ))+
  labs(x="Round", y="Budget / Remaining games",color="")
```

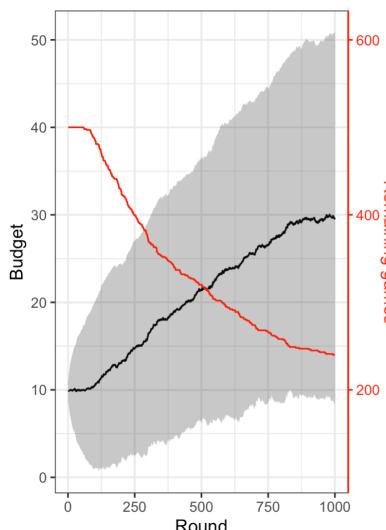
输出效果：



6. 我们发现，剩下的游戏次数和平均资金的数值之间差异过大，导致图像看起来十分割裂，且浪费了大量的空间。此时，我们可以引入第二个纵轴 (`sec.axis`)，并通过对数据进行缩放来完成绘制。比如，经过观察发现，可以将剩余游戏次数缩小 10 倍，再减去 10，这样能大致和平均资金的数值范围吻合，节约图像的空间。缩放之后，我们添加第二条 y 轴，并对其刻度进行逆向缩放（以下代码中相应位置的“.”代表刻度本身）：

```
p <- ggplot() +
  geom_line(data=res, aes(x=round, y=means, color="Mean budget")) +
  geom_ribbon(data=res, aes(x=round, ymin=means-sd, ymax=means+sd), alpha=0.3) +
  geom_line(data=res, aes(x=round, y=alive*0.1-10, color="Remaining games")) +
  theme_bw() +
  scale_color_manual(values=c(
    "Mean budget" = "black",
    "Remaining games" = "red"
  )) +
  labs(x="Round", color "") +
  scale_y_continuous(name="Budget",
    sec.axis=sec_axis(~(.+10)/0.1, name="Remaining games")) +
  theme(axis.line.y.right = element_line(color="red"),
        axis.ticks.y.right =element_line(color="red"),
        axis.text.y.right = element_text(color="red"),
        axis.title.y.right =element_text(color="red"))
p
```

输出效果：



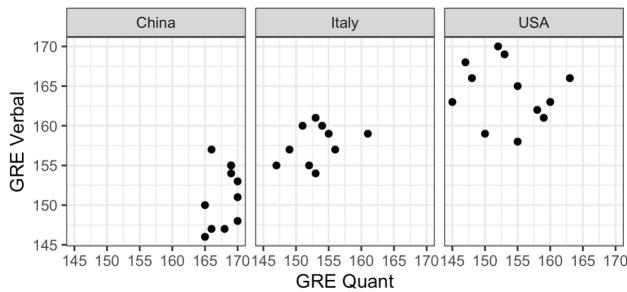
例 4：绘制多画幅图像 (multipanel figures)

由于版面的限制以及为了方便读者对比不同的数据，我们常常需要将不同的图表合并在同一幅图像里，形成多画幅图像。在 R 里，多画幅图像可以由不同的方式绘制，它们适用于不同的场景。

方法 1：直接在 `ggplot` 对象中添加 `facet_wrap` 函数。它适用于将同一组数据按不同的变量分配到不同画幅中，且每个画幅的图表是同类的。比如下面的例子，导入 `gre.csv`（美国某研究生院收到的硕士申请人的 GRE 分数，包括阅读分 `verbal`、数学分 `quant`、申请人的国籍 `country` 以及平时成绩 `gpa`），我们可以绘制三幅并列的图像来展示不同国家申请人的阅读分和数学分的分布情况。

```
gre <- read_csv("data/gre.csv")
p <- ggplot() +
  geom_point(data=gre, aes(x=quant, y=verbal)) +
  facet_wrap(~country) +
  labs(x="GRE Quant",
       y="GRE Verbal") +
  theme_bw()
p
```

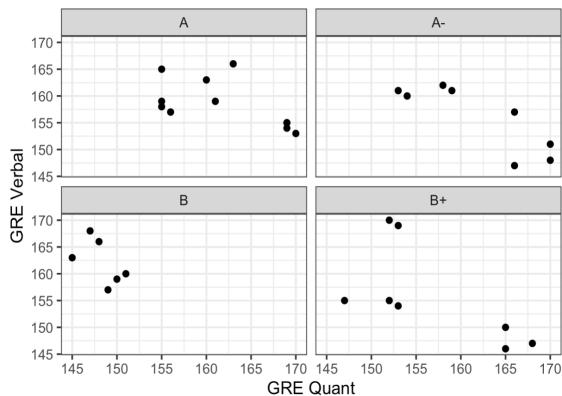
输出效果：



这种方法在无法使用颜色时（比如黑白印刷）尤其适用，可以通过拆分画幅来减少一个视觉通道的使用。当画幅比较多时，我们可以设置行数和列数来分配版面：

```
p <- ggplot() +
  geom_point(data=gre, aes(x=quant, y=verbal)) +
  facet_wrap(~gpa, ncol=2, nrow=2) +
  labs(x="GRE Quant",
       y="GRE Verbal") +
  theme_bw()
p
```

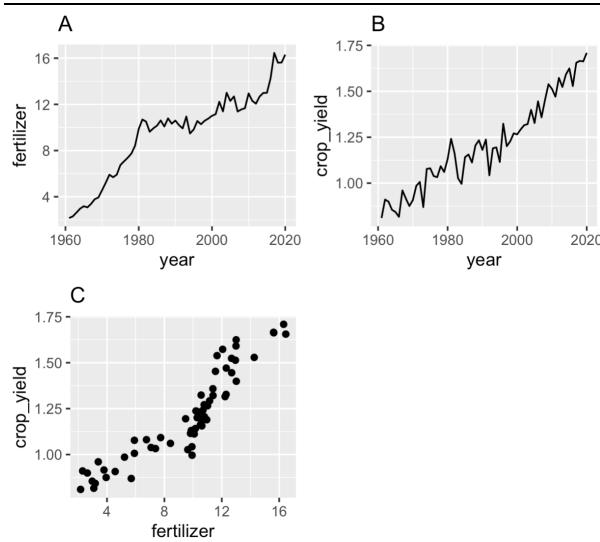
输出效果：



方法 2：使用 `gridExtra` 包。当多个画幅需要分别绘制时，可以安装并导入 `gridExtra` 包，将不同的画幅排布成网格状。例子如下，导入 `fertilizer.csv`，该数据集为非洲历年的农业产量 (`crop_yield`) 和肥料用量 (`fertilizer`) 的数据。

```
library(gridExtra)
data <- read_csv("data/fertilizer.csv")
p1 <- ggplot() + geom_line(data=data, aes(x=year, y=fertilizer))+ggtitle("A")
p2 <- ggplot() + geom_line(data=data, aes(x=year, y=crop_yield))+ggtitle("B")
p3 <- ggplot() + geom_point(data=data,aes(x=fertilizer,y=crop_yield))+ggtitle("C")
p4 <- ggplot()+ theme_void() #空白占位版面
#打开一个png窗口来保存图片
png("test.png",width=200,height=200,units="mm",res=200)
grid.arrange(p1,p2,p3,p4,nrow=2,ncol=2)
dev.off()
```

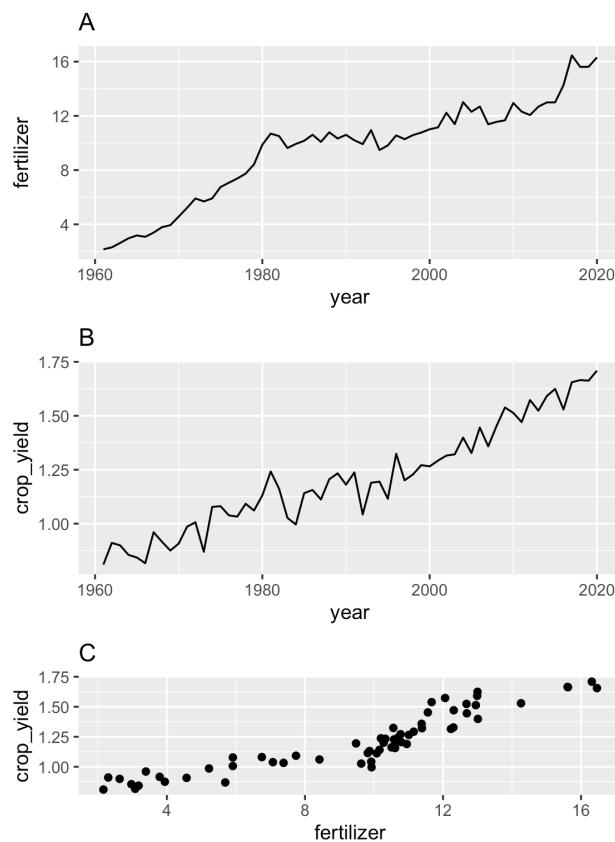
输出效果：



方法 3：使用 `cowplot` 包，可以改变行列的高度和宽度并自动对齐。比如，我们将前一个例子里的 p1~p3 竖向对齐，并让画幅 C 的高度相对变矮：

```
library(cowplot)
#将上一个例题中的p1~p3重新排版
gp <- plot_grid(p1,p2,p3, align = "v", nrow = 3, rel_heights = c(1,1,0.7))
gp
```

输出效果：

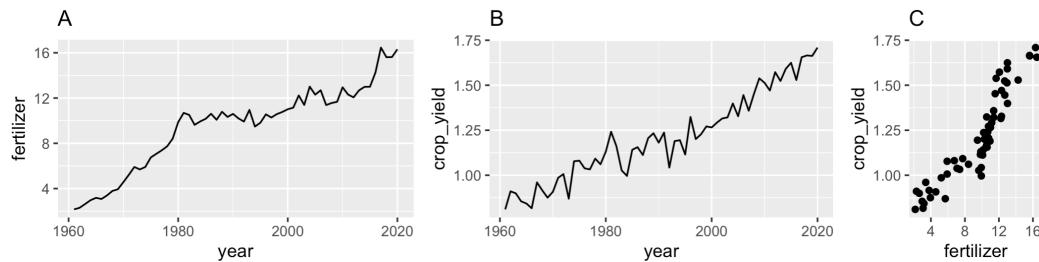


在使用 `cowplot` 时，最好用 `ggsave` 将图片导出后再查看效果。直接在 RStudio 里查看效果，可能会不准确。在横排的时候，也可以改变各画幅的相对宽度：

```
#将上一个例题中的p1~p3重新排版
```

```
gp <- plot_grid(p1,p2,p3, align = "h", ncol = 3, rel_widths = c(1,1,0.5))  
gp
```

输出效果：



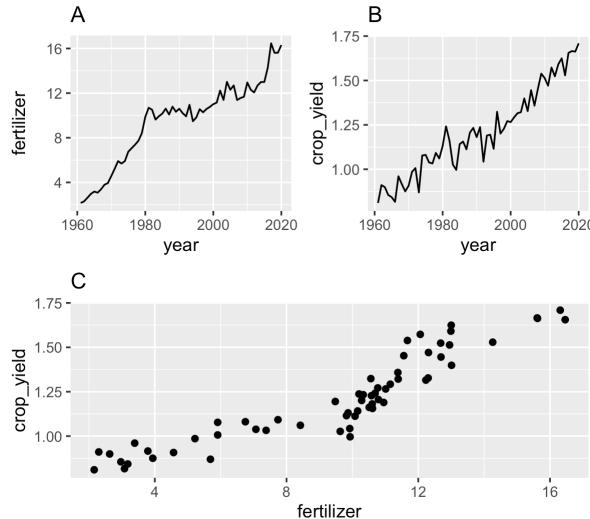
方法 4：使用 `patchwork` 包，可以任意调整画幅的具体格式。

```
library(patchwork)
```

```
#将上一个例题中的p1~p3重新排版
```

```
p <- ( p1 | p2 ) / p3  
p
```

输出效果：



例 5：平行坐标系图和雷达图（高维度数据）

1. 在 R 语言中，绘制平行坐标系图需要安装 `GGally` 包。如果安装时遇到报错（`Error in install.packages : Updating loaded packages`）请重启 RStudio 的 Session，这是因为 `GGally` 包是对 `ggplot2` 的扩展，如果 `ggplot2` 在 RStudio 中已被激活，则 `GGally` 无法安装。安装完成后，可用 `GGalley` 代替 `ggplot` 进行平行坐标系图的绘制。

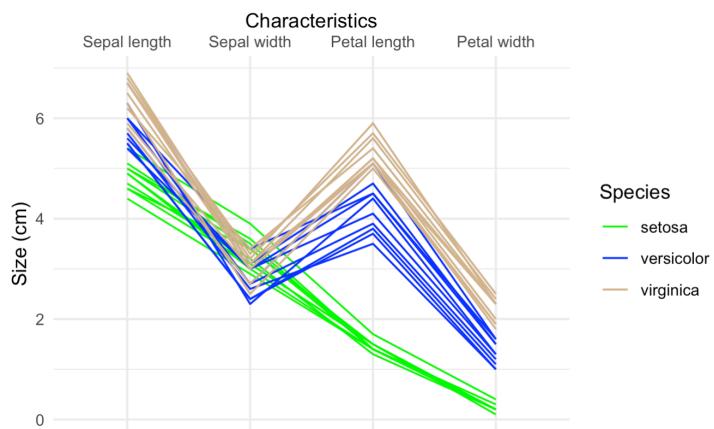
请导入 `iris.csv` 数据，该数据包含了部分鸢尾花的花瓣及花萼的长和宽(1-4列)以及具体的鸢尾花种类(`Species`)。我们可以利用 `GGally` 中的 `ggparcoord()` 函数绘制平行坐标系图来展示不同鸢尾花的花瓣及花萼的长和宽。

```

library(GGally)
p <- ggparcoord(data = iris,#所用数据框
                  columns = 1:4,#要绘制的维度在数据框的哪几列
                  mapping=aes(color=Species),#根据哪一列着色
                  groupColumn=5,#根据哪一列分类
                  scale="globalminmax" #如何对不同的维度进行比例缩放
                  #"globalminmax"为不进行缩放
                 #"std"为单变量减去平均数再除以标准差
                 #"robust"为单变量减去中位数再除以中位数绝对偏差
                  #"uniminmax"为针对各个变量的最小最大缩放 (min-max scaling)
                  )+
theme_minimal()+
scale_x_discrete(position = "top")+
labs(x="Characteristics",y="Size (cm)")+
scale_colour_manual(values=c("green","blue","tan"))
p

```

输出效果：



请尝试更改 `scale` 参数，观察绘图的效果有什么变化。

2. 我们可以用雷达图来对比这三种鸢尾花的花瓣及花萼长宽的平均值。雷达图需要统一各个轴的取值范围，因此通常来说，我们需要对不同的属性进行归一化或缩放。例如，我们可以对鸢尾花的四个属性进行 min-max 缩放，将其投射在 0 到 1 的区间内，然后再绘制雷达图。

绘制雷达图需要安装并导入 `fmsb`，然后先对数据进行预处理，计算出每一类鸢尾花四个属性的平均值并完成 min-max 缩放，并把数据框调整为 `fmsb` 认可的格式，即第一行为各属性取值范围的上限，第二行为各属性取值范围的下限，后面各行为不同类别的属性取值。如下图所示：

	Sepal length	Sepal width	Petal length	Petal width
1	1	1	1	1
2	0	0	0	0
3	0.184	0.63125	0.0326086956521739	0.05
4	0.58	0.2875	0.634782608695652	0.5041666666666667
5	0.82	0.45625	0.876086956521739	0.8625

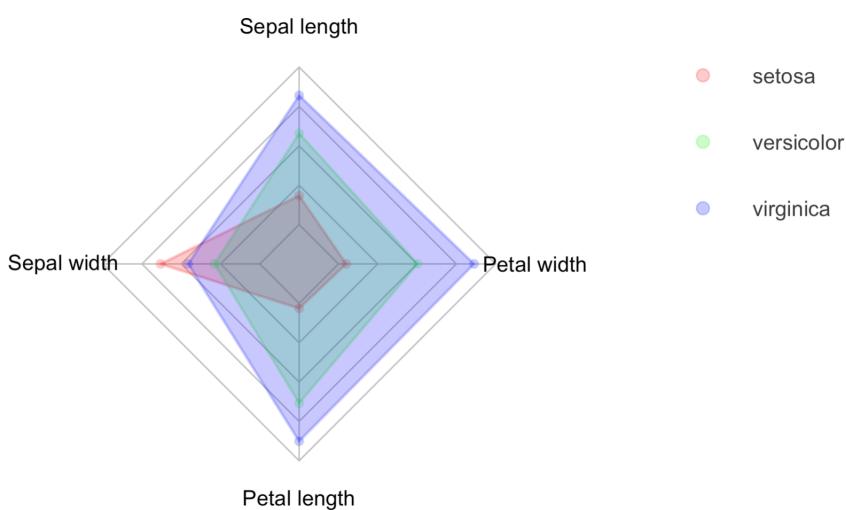
绘制雷达图的代码如下：

```

library(fmsb)
#建立min-max scaling缩放的函数
min_max_scale <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
# 对鸢尾花的数据进行缩放
iris[,1:4] <- lapply(iris[,1:4], min_max_scale)
# 对缩放后，每一类鸢尾花的四个属性取平均值
iris <- iris %>%
  group_by(Species) %>%
  summarise(across(c(`Sepal length`, `Sepal width`, `Petal length`, `Petal width`), mean))
#提取鸢尾花的种类名，为图例做准备
legend <- iris$Species
#在获得的数据框中，插入取值上下限的定义行，此处分别为0和1，4是属性列的数量
iris <- rbind(c("Min",rep(0,4)),iris)
iris <- rbind(c("Max",rep(1,4)),iris)
#将第一列改为行的名称
rownames(iris) <- iris$Species
iris <- iris[,-1]
#确保数据为numeric类型
iris[1:4] <- lapply(iris[1:4], as.numeric)
#绘制雷达图，因为需要单独定义透明度，因此这里的颜色需要用rgb()来定义
colors <- c(rgb(1, 0, 0, 0.25),
            rgb(0, 1, 0, 0.25),
            rgb(0, 0, 1, 0.25))
radarchart(iris, #数据
           cglty = 1, #坐标系样式
           cglcol = "gray", # 坐标系颜色
           pcol = colors, # 雷达图每一类的轮廓颜色
           plwd = 2, # 雷达图轮廓宽度
           plty = 1, # 雷达图轮廓线条种类
           pfcol = colors) # 雷达图每一类的颜色，如果不填充颜色则省略这一参数
legend("topright",
       legend = legend,
       bty = "n", pch = 20, col = colors,
       text.col = "grey25", pt.cex = 2)

```

输出效果：



作业部分

数据描述

soccer_schedule.csv

该数据集为某足球队在 2002 年的联赛赛程，以及赛前官方机构预测的每轮比赛的获胜几率。

- **round:** 联赛轮次
- **date:** 比赛日期
- **field:** 主场 (home) 或客场 (away)
- **opponent:** 比赛对手的队名
- **win_prob:** 获胜几率
- **draw_prob:** 打平几率
- **lose_prob:** 输球几率

roster.csv

该球队在 2002 年赛季的球员名单，前 8 列为球员在 8 个不同领域 (Dribbling 带球、Striking 射门、Passing 传球、Tackling 抢球、Header 头球、Speed 速度、Strong 强壮、Stamina 耐力) 的能力评分 (分值为 0-20)，第 9 列 (Position) 为球员在场上的位置，包括 Forward (前锋)、Midfielder (中场)、Defender (后卫) 和 Keeper (守门员)。第 10 列为球员的具体角色 (Role)。

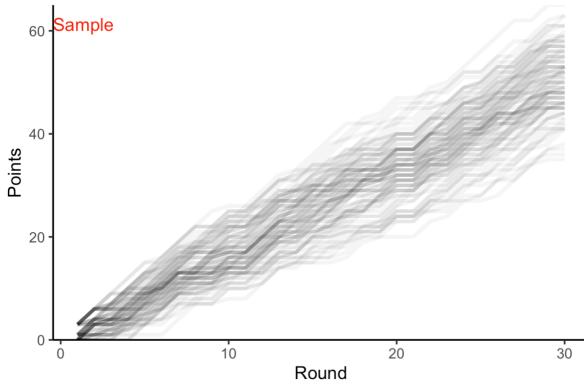
revenue.csv

该球队在 2002 年赛季每轮后的财政盈利状况 (\$).

- **round:** 联赛轮次
- **tv:** 电视转播分成
- **souvenir:** 周边商品出售
- **advertisement:** 广告收益

作业

1. 已知：足球联赛采用积分制，每轮比赛后球队有可能获得积分。如果球队获胜，则积分增加 3 分；如果打平则积分增加 1 分；如果输球则积分不会增加。根据 `soccer_schedule.csv` 中每轮比赛的胜率和打平几率，对该球队在 2002 赛季的每一轮后的积分累积情况进行蒙特卡洛模拟，并将结果绘制在同一张图表中。要求：横轴为轮次 (`round`)，纵轴为累计积分 (`points`)，蒙特卡洛模拟运行 100 次。请将这张图表的 `ggplot` 对象命名为 `p1`。
【提示 1：每轮比赛获胜、打平、输球的几率之和为 1，可以通过 `rrunif()` 并对 0-1 区间进行三段分割来进行判定；提示 2：导入的 `soccer_schedule.csv` 数据框可以对应于例题中的 `res` 数据框。效果如下图所示，但请绘制自己的模拟结果（模拟的结果有随机性）。】



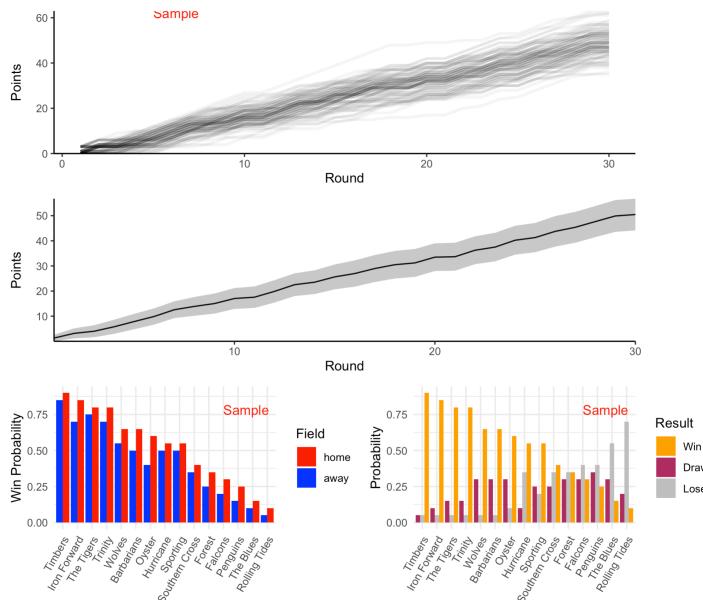
2. 计算每轮比赛后该球队累积积分的均值和标准差，并绘制在图表中。请将这张图表的 `ggplot` 对象命名为 `p2`。

【提示：计算均值和标准差时，需要注意数据框中存在的非积分属性（和蒙特卡洛模拟结果无关的属性）的列数。】

3. 在 2002 赛季中，该球队共面对 15 个不同的对手，和每个对手进行两轮比赛，其中主场、客场各进行一场比赛。绘制一幅多数据系列柱形图，展示该球队遇到各个对手时在主场和客场的获胜几率。要求：横轴为对手的队名，纵轴为获胜几率 (`win probability`)，主场和客场作为两个数据系列。横轴按照主场的获胜几率排序。请将这张图表的 `ggplot` 对象命名为 `p3`。【提示：如果只关注主客场 (`field`)、对手 (`opponent`) 和胜率 (`win_prob`) 三列，那么导入 `soccer_schedule.csv` 形成的数据框已经是一维表。】

4. 只考虑主场比赛的情况下，绘制一幅多数据系列柱形图展示该球队遇到各个对手时的胜、平、负几率，横轴按照获胜几率排序。请将这张图表的 `ggplot` 对象命名为 `p4`。

5. 将以上绘制的 `p1~p4` 组合成多画幅图像，该图像有 3 行 2 列，其中 `p1` 占据第一行的两列，`p2` 占据第二行的两列，`p3` 和 `p4` 分属第三行的两列。效果如下图所示，但请自己绘制。



6. 绘制一幅堆积面积图，展示该球队在各轮比赛后，电视转播分成、周边商品出售、广告收益等三个来源的营收情况。

7. 用百分比堆积面积图重新绘制第 6 问的图像，并将第 6、7 两问的图像组合成多画幅图像（一列、两行）。这两幅图像上下对齐。

8. 在第 7 题多画幅图像的基础上，在其下方再增添一个画幅，该画幅为该球队在各轮后电视转播分成、周边商品出售、广告收益等三个来源的营收情况的河流图。

9. 用平行坐标系图展示该球队不同位置（Position）球员的 8 项能力。

10. 筛选出属于 Striker (中锋)、Winger (边锋)、Central Back (中后卫) 和 Fullback (边后卫) 四种角色 (Role) 的球员，利用雷达图展示这几种角色 8 项能力的平均分。【提示：该数据集为对球员能力的评分，已经确认了 0-20 的分值，因此不需要做缩放，只需要调整雷达图的取值上下限。】

提交内容

将 10 幅图表分别插入学习通答题区的相应位置。将 R 代码汇总到一个文档里，作为附件上传到第一题。R 文档请命名为“Lab5+姓名+学号”，比如“Lab5 张三 1010101010.R”。

截止时间：2023 年 1 月 3 日北京时间 23:59。