

Lab 2 : 常用统计方法 & 常用统计图表的绘制

第一部分：常用统计方法

什么是统计检验

统计检验是一种基于数据驱动的决策过程，用于根据样本数据对总体做出推断。它涉及使用统计方法来评估观察到的数据是否支持或反对某个假设，或者比较不同组之间的差异是否显著。统计检验在科学、工程、社会科学和商业分析中都有广泛应用，它帮助研究者根据数据做出客观、可量化的决策。统计检验的基本步骤通常包括以下几点：

1. 确定研究问题和假设：

明确研究目的和问题，形成明确的假设。这通常包括原假设（null hypothesis, H_0 ）和备择假设（alternative hypothesis, H_1 ）。原假设通常表示没有效应或没有差异，而备择假设则表达了研究者期望发现的结果。

2. 选择适当的统计检验方法：

根据数据的类型（如连续变量、分类变量等）、研究设计（如独立样本、配对样本等）和假设的内容（如单尾检验、双尾检验等），选择合适的统计检验方法，比如 t 检验、卡方检验、方差分析（ANOVA）或非参数检验等。

3. 收集数据：

按照研究设计收集数据，确保数据的准确性和完整性。

4. 数据整理和预处理：

清洗数据，处理缺失值、异常值或重复值，确保数据质量。

5. 执行统计检验：

使用选定的统计方法对数据进行分析。这通常涉及计算检验统计量（如 t 值、 F 值、卡方值等）和对应的 p 值。

6. 解释结果：

根据得到的检验统计量和 p 值，判断原假设是否被拒绝。通常，如果 p 值小于预定的显著性水平（如 0.05），则拒绝原假设，认为观察到的效应或差异是显著的。

7. 得出结论：

基于统计检验的结果，得出关于研究问题的结论。注意结论应该客观、准确，并考虑到研究的局限性和可能的偏差。在整个过程中，还需要注意避免常见的统计错误，如第一类错误（误报）和第二类错误（漏报），以及确保样本的代表性、随机性和足够的样本量，以提高统计推断的准确性。

常用统计方法

- **T 检验 (Student's T-Test):** 主要用于样本含量较小（例如样本容量 $n < 30$ ），总体标准差 σ 未知的正态分布。
T 检验是用于小样本（样本容量小于 30）的两个平均值差异程度的检验方法。它是用 T 分布理论来推断差异发生的概率，从而判定两个平均数的差异是否显著。
- **Z 检验 (Z Test):** 是一般用于大样本（即样本容量 $n > 30$ ）平均值差异性检验的方法。它是用正态分布的理论来推断差异发生的概率，从而比较两个平均数的差异是否显著。
- **F 检验 (F Test):** 又叫方差齐性检验。在两样本 t 检验中要用到 F 检验。从两研究总体中随机抽取样本，要对这两个样本进行比较的时候，首先要判断两总体方差是否相同，即方差齐性。若两总体方差相等，则直接用 t 检验，若不等，可采用 t' 检验或变量变换或秩和检验等方法。其中要判断两总体方差是否相等，就可以用 F 检验。
- **方差分析 (ANOVA):** 视为特殊的 F 检验，用于检测两个及两个以上样本均数差别的显著性。方差分析的方法有

- 许多种，单因素方差分析主要用来研究一个控制变量的不同水平是否对观测变量产生了显著影响。
- **卡方检验 (Chi Square Test):** 主要是比较实际观测频数与期望频数之间是否存在显著差异，常用于检验分类变量。其基本思想是根据样本数据推断总体的分布与期望分布是否有显著性差异，或者推断两个分类变量是否相关或者独立。
 - **K-S 检验(Kolmogorov-Smirnov's D Score):** 是一种非参数检验方法，用于检验一个样本是否来自于某个特定的理论分布（如正态分布、均匀分布等），或者检验两个独立样本是否来自具有相同分布的总体。这种检验方法基于累积分布函数 (CDF)，通过比较样本数据的累积分布与理论分布之间的差异来进行判断。

T 检验

在 R 中执行 t 检验非常简单，可以直接使用内置的 `t.test()` 函数。具体来说，t 检验包括单样本、双样本（独立）和配对 t 检验三种常用类型。

单样本 t 检验： 测试单组数据的平均值是否与某个已知的值有所不同。

```
# 生成一组数据
data <- c(0.8, 1.3, 1.2, 0.9, 0.7, 1.1, 0.8)

# 查看这组数据的平均值是否和1显著不同
t_test_result <- t.test(data, mu = 1)
print(t_test_result)
```

输出的结果如下：

```
One Sample t-test

data: data
t = -0.33029, df = 6, p-value = 0.7524
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
0.7597602 1.1830970
sample estimates:
mean of x
0.9714286
```

可以看到，`p-value` 大于 `0.05`，即无法拒绝原假设（平均值和 `1` 没有区别），统计结果不显著。

如果使用另一组数据：

```
# 生成另一组数据
data <- c(4,2,5,3,7,1,6)

# 查看这组数据的平均值是否和1显著不同
t_test_result <- t.test(data, mu = 1)
print(t_test_result)
```

输出结果：

One Sample t-test

```
data: data
t = 3.6742, df = 6, p-value = 0.0104
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
2.002105 5.997895
sample estimates:
mean of x
4
```

这一次， p 值小于 0.05 ，可以拒绝原假设，统计结果显著，我们可以主张该组数据的平均值和 1 显著不同。

双样本 t 检验：测试单组数据的平均值是否与某个已知的值有所不同。

双样本 t 检验：测试 2 组数据的平均值是否明显不同。

```
# 生成两组数据
group1 <- c(5.1, 5.5, 4.9, 5.0, 5.3)
group2 <- c(6.1, 6.5, 6.2, 6.3, 6.4)

# 查看这2组数据的平均值是否显著不同
t_test_result <- t.test(group1, group2)
print(t_test_result)
```

输出结果：

```
Welch Two Sample t-test

data: group1 and group2
t = -8.8481, df = 6.908, p-value = 5.154e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-1.445485 -0.834515
sample estimates:
mean of x mean of y
5.16      6.30
```

p 值小于 0.05 ，统计结果显著。

配对 t 检验：配对比较数据，例如，同样一批患者，在用药前和用药后的某项数值是否有所区别，此时每一位患者被记录了两个数据，它们之间进行相互比较。

```
# 生成两组数据
before <- c(5.1, 5.5, 4.9, 5.0, 5.3)
after <- c(6.1, 6.5, 6.2, 6.3, 6.4)

# 配对比较
t_test_result <- t.test(before, after, paired = TRUE)
print(t_test_result)
```

输出结果：

Paired t-test

```
data: before and after
t = -16.808, df = 4, p-value = 7.343e-05
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
-1.3283077 -0.9516923
sample estimates:
mean difference
-1.14
```

p 值小于 0.05，统计结果显著。

Z 检验

Z 检验总体和 t 检验相似，但主要针对样本量较大的情况。

单样本 Z 检验：检验样本平均数与一个已知总体平均数是否存在差异（请自行查看输出结果）。

```
library(BSDA)

# 生成数据
data <- c(1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6)

# 需要对比的平均值
mu <- 5

# 需要对比的标准差
sigma <- 0.5

# Z 检验测试
z_test_result <- z.test(data, mu = mu, sigma.x = sigma)
print(z_test_result)
```

双样本 Z 检验：在群体方差已知的情况下，比较两个独立群体的均值。

```
# 生成两组数据
group1 <- c(5.1, 5.5, 4.9, 5.0, 5.3, 5.1, 5.5, 4.9, 5.0, 5.3, 5.1, 5.5, 4.9, 5.0, 5.3,
      5.1, 5.5, 4.9, 5.0, 5.3, 5.1, 5.5, 4.9, 5.0, 5.3, 5.1, 5.5, 4.9, 5.0, 5.3,
      5.1, 5.5, 4.9, 5.0, 5.3, 5.1, 5.5, 4.9, 5.0, 5.3, 5.1, 5.5, 4.9, 5.0, 5.3)
group2 <- c(6.1, 6.5, 6.2, 6.3, 6.4, 6.1, 6.5, 6.2, 6.3, 6.4, 6.1, 6.5, 6.2, 6.3, 6.4,
      6.1, 6.5, 6.2, 6.3, 6.4, 6.1, 6.5, 6.2, 6.3, 6.4, 6.1, 6.5, 6.2, 6.3, 6.4,
      6.1, 6.5, 6.2, 6.3, 6.4, 6.1, 6.5, 6.2, 6.3, 6.4, 6.1, 6.5, 6.2, 6.3, 6.4)

# 两组数据的标准差
sigma1 <- 0.5
sigma2 <- 0.6

# Z 检验测试
z_test_result <- z.test(x = group1, y = group2, sigma.x = sigma1, sigma.y = sigma2)
print(z_test_result)
```

F 检验和方差分析

在 R 中，你可以使用 `var.test()` 函数进行双样本 F 检验来比较两组数据的方差，或使用 `aov()` 函数进行单因子方差分析（ANOVA，本质上是多组数据的 F 检验）来检验多组数据的方差。

F 检验（请自行查看输出结果）

```
# 两组数据
group1 <- c(5.1, 5.5, 4.9, 5.0, 5.3)
group2 <- c(6.1, 6.5, 6.2, 6.3, 6.4)

# F-test
f_test_result <- var.test(group1, group2)
print(f_test_result)
```

ANOVA（请自行查看输出结果）

```
# 生成三组数据
group1 <- c(5.1, 5.5, 4.9, 5.0, 5.3)
group2 <- c(6.1, 6.5, 6.2, 6.3, 6.4)
group3 <- c(7.1, 7.5, 7.2, 7.3, 7.4)

# 将它们组合成一个数据框
data <- data.frame(
  value = c(group1, group2, group3),
  group = factor(rep(c("Group 1", "Group 2", "Group 3"), each = 5))
)

# ANOVA
anova_result <- aov(value ~ group, data = data)
summary(anova_result)
```

卡方检验

在 R 中，你可以使用 `chisq.test()` 函数执行卡方检验。该检验通常用于评估两个分类变量之间是否存在显著关联，或检验观测数据与理论分布的拟合程度。

独立卡方检验：要检验两个分类变量是否独立，通常要使用或然率表（contingency table）。

```
# 创造一个contingency table
data <- matrix(c(30, 10, 20, 40), nrow = 2)
colnames(data) <- c("Category 1", "Category 2")
rownames(data) <- c("Group A", "Group B")

# 独立卡方检验
chi_square_result <- chisq.test(data)
print(chi_square_result)
```

卡方拟合度检验：检验某个分类变量的观察频率是否与预期频率一致。

```
# 观察频率（假设有3个类别、100个样本，三个类别分别有50、30、20个样本）
observed <- c(50, 30, 20)

# 预期频率（以三等分频率为例，即预期中每个类别都是总样本的1/3）
expected <- c(33.33, 33.33, 33.33)

# 卡方拟合度检验
chi_square_gof_result <- chisq.test(observed, p = expected / sum(expected))
print(chi_square_gof_result)
```

K-S 检验

Kolmogorov-Smirnov (K-S) 检验是一种非参数检验，用于确定两个样本是否来自同一分布，或将样本与参考概率分布进行比较。在 R 中，你可以使用 `ks.test()` 函数执行 K-S 检验。

单样本 K-S 检验：查看某个数据的分布是否与一个已知分布相同。

```
# 生成100个随机数据值，让其符合均值为0标准差为1的正态分布
data <- rnorm(100, mean = 0, sd = 1)

# 测试该数据是否为均值为0标准差为1的正态分布
ks_test_result <- ks.test(data, "pnorm", mean = 0, sd = 1)
print(ks_test_result)
```

注意（易错点），K-S 检验的原假设为“分布是相同的”，如果 `p` 值大于 `0.05`，则表示无法拒绝原假设，即分布是相同的；如果 `p` 值小于 `0.05`，则需要拒绝原假设，即分布显著不同。

双样本 K-S 检验：查看两组数据的分布是否相同。

```
#生成两组数据，但均值和标准差不同
data1 <- rnorm(100, mean = 0, sd = 1)
data2 <- rnorm(100, mean = 0.5, sd = 3)

#K-S检验
ks_test_result <- ks.test(data1, data2)
print(ks_test_result)
```

相关系数

相关系数是一种统计指标，用以衡量两个变量之间的线性关系密切程度。相关系数的值介于 -1 和 1 之间。当相关系数为 1 时，表示两个变量完全正相关，即一个变量增加时，另一个变量也相应增加。当相关系数为 -1 时，表示两个变量完全负相关，即一个变量增加时，另一个变量相应减少。当相关系数为 0 时，表示两个变量之间没有线性关系。需要注意的是，相关系数只能衡量线性关系的强度和方向，不能说明变量之间的因果关系。此外，即使相关系数很高，也不意味着一个变量是另一个变量变化的原因。

常见的相关系数：

- **皮尔逊相关系数 (Pearson correlation coefficient)**：通常用符号 r 表示，是一种常用来衡量两个变量之间线性关系密切程度的量。皮尔逊相关系数的取值范围在 -1 到 1 之间。值为 1 意味着完全的正线性相关性，值为 -1 意味着完全的负线性相关性，值为 0 意味着没有线性相关性。
- **斯皮尔曼等级相关系数 (Spearman's rank correlation coefficient)**，通常用符号 ρ/\rhoho 表示，是一种非参数性质的等级相关统计量。它用于评估两个变量之间的等级相关性，即当一个变量增加或减少时，另一个变量是否也会相应地增加或减少，而不考虑它们之间的具体数值关系。

Pearson's r

```
# 生成数据
x <- rnorm(100) # 100个数据点，符合正态分布
y <- 0.5 * x + rnorm(100) * 0.5 # 生成另一组数据点，和x是带有噪音的线性关系

# 计算Pearson's r
pearson_corr <- cor(x, y, method = "pearson")
print(paste("Pearson's r:", pearson_corr))

# 统计检验
pearson_test <- cor.test(x, y, method = "pearson")
print(pearson_test)
```

Spearman's ρ

```
# 生成数据
x <- rnorm(100) # 100个数据点，符合正态分布
y <- 0.5 * x + rnorm(100) * 0.5 # 生成另一组数据点，和x是带有噪音的线性关系

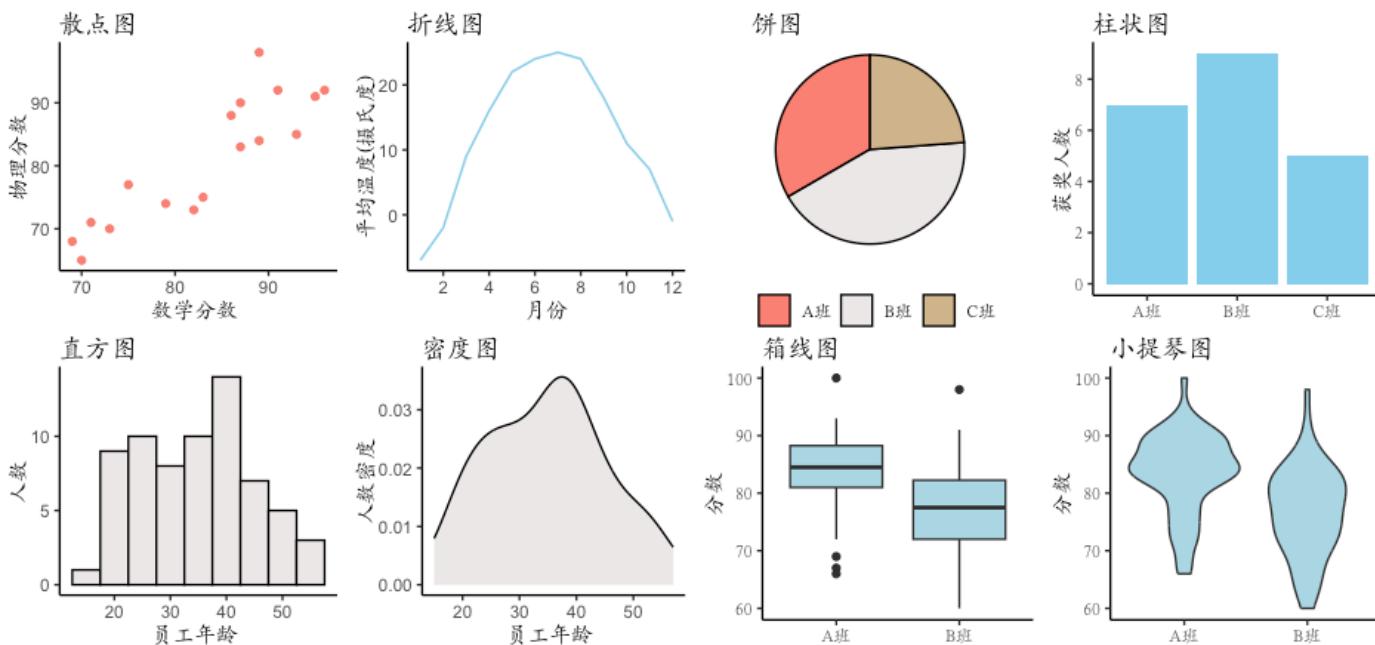
# 计算Spearman's rho
spearman_corr <- cor(x, y, method = "spearman")
print(paste("Spearman's rho:", spearman_corr))

# 统计检验
spearman_test <- cor.test(x, y, method = "spearman")
print(spearman_test)
```

第二部分：常用统计图表的绘制

常见统计图表类型及特征

- 散点图 (Scatter Plot)**: 主要用于显示两个数值变量之间的关系和分布模式。尤其是，我们通常使用散点图来确定两个变量是否存在相关性。例如，可以用散点图表示身高和体重之间的关系，或者气温与冰淇淋销量之间的关系。
- 折线图 (Line Chart)**: 通常用于表现趋势和时间序列数据。它可以很好地表示一段时间内某个指标的变化情况，如股票价格、气温走势、人口增长等。
- 饼图 (Pie Chart)**: 用于表示各部分对整体的占比，通常用于描绘类别性数据。例如，你饼图可以表示大学里不同学院的学生人数占学校学生总人数的比例，或一个公司的科研经费占年度总支出的比例。
- 柱状图 (Bar Chart)**: 可以用来比较不同组或类别之间的数量或频率。例如，商家可以通过柱状图直观地看出哪种产品销售最好，学校可以用柱状图来展示和对比每个学院发表论文的数量。
- 直方图 (Histogram)**: 通常被用来表现连续数值变量的分布情况。它通过把整个数值范围分割成一系列等宽的区间（也称为“bins”），并计算每个区间内样本数量来达到这一目的。例如，直方图可以用来显示一个城市各个年龄段人口的分布情况，或者一个班在期末考试中的各个分数段的人数。
- 密度图 (Density Plot)**: 用于描绘连续变量的分布情况。它可以看做直方图在不同条形间采用平滑线段进行连接后的结果。对比直方图，密度图更能清楚地展现数据的分布特征。
- 箱线图 (Box Plot)**: 也称为盒须图，它能提供有关数据集中心趋势和分布的信息。该图表会显示五个统计参数：最小值、第一四分位数、中位数、第三四分位数和最大值。它被用于展示数据的总体分布，同时能够辨识出数据中的离群值 (outliers)。箱线图通常被用来对比多组数据的分布，比如对比班上男生和女生的身高分布、对比两个班的考试成绩分布等。
- 小提琴图 (Violin Plot)**: 是箱线图和密度曲线的结合，不仅显示了数据的分布情况，还反映了数据的密度分布。这种类型的图表在比较多组数据的分布及其概率密度时特别有用。



ggplot2

`ggplot2` 是一个用于数据可视化的 R 语言包，它提供了一种高效、简洁和灵活的方式来生成高质量的图像。这个包基于“图形语法”（The Grammar of Graphics），也就是说，你可以使用 `ggplot2` 的函数构造各种复杂的图像。在 `ggplot2` 中，图形是由几个不同的部分组成，如数据集、坐标系、几何对象（多边形、线、点等）、视觉通道和映射（颜色、大小等）以及其他主题元素。通过将这些部分按照需要组合在一起，用户可以创建出广泛的静态或动态可视化效果。

ggplot2 的工作原理

在使用 `ggplot2` 时，你首先需要提供一个数据框架（`data frame`），其变量将会被映射到视觉通道。接下来，你需要选择一种或多种几何对象（`geometric objects`），如线条、点、柱状图等，以呈现数据。你还可以添加统计变换（`statistical transformations`）、坐标系统（`coordinate systems`）、主题元素（`theme elements`）等来进一步修改或美化图像。所有这些组件（数据、映射、几何对象、统计变换、坐标系统以及主题）都会被合并成一个“`ggplot`”对象（空白的对象可理解为一张“画布”）。然后，当 R 去 `print` 或者 `call` 这个“`ggplot`”对象时，`ggplot2` 库会将所有设置好的组件结合在一起，以图层的形式来生成最终的图像。

总的来说，`ggplot2` 中的每个函数都对应着图形生成过程中的一个图层。通过将各个函数相互叠加，可以构建出非常复杂和精细的图形。这也正是 `ggplot2` 的魅力所在：它既可以快速创建探索性数据分析所用的基本图表，又能经过微调实现出版物级别的复杂可视化需求。

例 1：绘制散点图

导入 `example1.csv`（两个班的学生在物理和化学考试中的成绩）。该数据有 5 列，第一列学号，第二列物理成绩，第三列化学成绩，第四列班级，第五列籍贯。

导入数据创建 `ggplot` 对象：

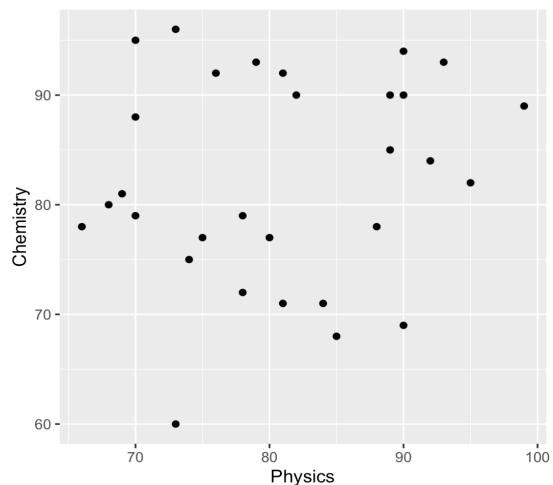
```
library(readr)
library(ggplot2)
# 导入数据
example1 <- read_csv("~/Documents/数据分析和可视化/data/lab2/example1.csv")
# 创建 ggplot 对象，命名为 plot
plot <- ggplot()
# 绘制所创建的对象
plot
```

RStudio 界面中会出现一张灰色的画布，即默认的 `ggplot` 空对象。

在对象上增添散点图函数：

```
# 添加散点图图层
plot <- plot +
  geom_point(data=example1, aes(x=Physics, y=Chemistry))
# 绘制所创建的对象
plot
```

绘制结果：



以上两个步骤可以合并代码:

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry))
#绘制所创建的对象
plot
```

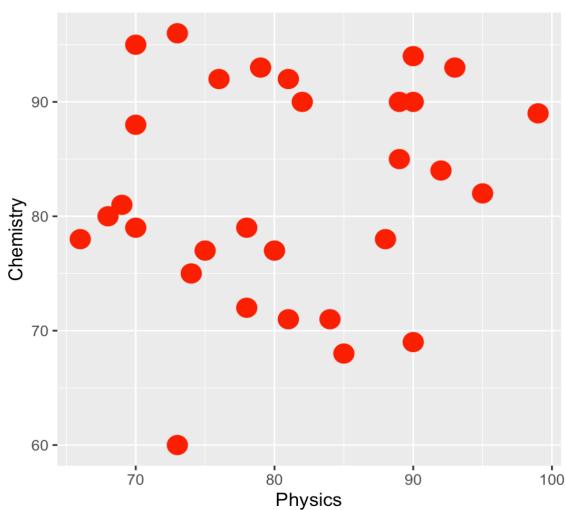
接下来，我们对这张图表做风格上的修改。

注意：这门课不以美观作为评分标准，而以对数据展现的良好程度作为评判标准。修改风格有利于我们更好地展现数据。

1. 把散点的颜色改成红色，并让点变大：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry), color = "red", size = 5)
#绘制所创建的对象
plot
```

输出结果：



R 的预设颜色名称：

coral3	deeppink4	gray27	gray87	grey39	grey99	lightpink1	mistyrose1	pink4	slategray1		
coral2	deeppink3	gray26	gray86	grey38	grey98	lightpink	mistyrose	pink3	slategray		
coral1	deeppink2	gray25	gray85	grey37	grey97	lightgrey	mintcream	pink2	slateblue4		
coral	deeppink1	gray24	gray84	grey36	grey96	lightgreen	midnightblue	pink1	slateblue3	yellowgreen	
chocolate4	deeppink	gray23	gray83	grey35	grey95	lightgray	mediumvioletred	pink	slateblue2	yellow4	
chocolate3	darkviolet	gray22	gray82	grey34	grey94	lightgoldenrod4	mediumspringgreen	peachpuff4	slateblue1	yellow3	
chocolate2	darkturquoise	gray21	gray81	grey33	grey93	lightgoldenrod3	mediumturquoise	peachpuff3	skyblue4	yellow2	
chocolate1	darkslategrey	gray20	gray80	grey32	grey92	lightgoldenrod3	mediumslateblue	peachpuff2	skyblue3	yellow1	
chocolate	darkslategray4	gray19	gray79	grey31	grey91	lightgoldenrod2	mediumseagreen	peachpuff1	skyblue2	yellow	
chartreuse4	darkslategray3	gray18	gray78	grey30	grey90	lightgoldenrod1	mediumpurple4	peachpuff	skyblue1	whitesmoke	
chartreuse3	darkslategray2	gray17	gray77	grey29	grey89	lightgoldenrod	mediumpurple3	peachpuff	skyblue	wheat4	
chartreuse2	darkslategray1	gray16	gray76	grey28	grey88	lightcyan4	mediumpurple2	papayawhip	skyblue	wheat3	
chartreuse1	darkslategray	gray15	gray75	grey27	grey87	lightcyan3	mediumpurple1	palevioletred4	sienna4	wheat2	
chartreuse	darkslateblue	gray14	gray74	grey26	grey86	lightcyan2	mediumpurple	palevioletred3	sienna3	wheat1	
cadetblue4	darkseagreen4	gray13	gray73	grey25	grey85	lightcyan1	mediumorchid4	palevioletred2	sienna2	wheat	
cadetblue3	darkseagreen3	gray12	gray72	grey24	grey84	lightcyan	mediumorchid3	palevioletred1	sienna1	violetred4	
cadetblue2	darkseagreen2	gray11	gray71	grey23	grey83	lightcoral	mediumorchid2	palevioletred	sienna	violetred3	
cadetblue1	darkseagreen1	gray10	gray70	grey22	grey82	lightblue4	mediumorchid1	paleturquoise4	seashell4	violetred2	
cadetblue	darkseagreen	gray9	gray69	grey21	grey81	lightblue3	mediumorchid	paleturquoise3	seashell3	violetred1	
burlywood4	darksalmon	gray8	gray68	grey20	grey80	lightblue2	mediumblue	paleturquoise2	seashell2	violetred	
burlywood3	darkred	gray7	gray67	grey19	grey79	lightblue1	mediumamarillo	paleturquoise1	seashell1	violet	
burlywood2	darkorchid4	gray6	gray66	grey18	grey78	lightblue	maroon4	palegreen4	seagreen4	turquoise4	
burlywood1	darkorchid3	gray5	gray65	grey17	grey77	lemonchiffon4	maroon3	palegreen3	seagreen3	turquoise3	
burlywood	darkorchid2	gray4	gray64	grey16	grey76	lemonchiffon3	maroon2	palegreen2	seagreen2	turquoise2	
brown4	darkorchid1	gray3	gray63	grey15	grey75	lemonchiffon2	maroon1	palegreen2	seagreen1	turquoise1	
brown3	darkorchid	gray2	gray62	grey14	grey74	lemonchiffon1	maroon	palegreen1	seagreen	tomato4	
brown2	darkorange4	gray1	gray61	grey13	grey73	lemonchiffon	magenta4	palegreen	seagreen	sandybrown	
brown1	darkorange3	gray0	gray60	grey12	grey72	lawngreen	magenta3	palegoldenrod	salmon4	tomato3	
brown	darkorange2	gray	gray59	grey11	grey71	lavenderblush4	magenta2	orchid4	salmon	tomato2	
blueviolet	darkorange1	goldenrod4	gray58	grey10	grey70	lavenderblush3	magenta1	orchid3	salmon3	tomato1	
blue4	darkorange	goldenrod3	gray57	grey9	grey69	lavenderblush2	magenta	orchid2	salmon2	tomato	
blue3	darkolivegreen4	goldenrod2	gray56	grey8	grey68	lavenderblush1	linen	orchid1	salmon1	thistle4	
blue2	darkolivegreen3	goldenrod1	gray55	grey7	grey67	lavenderblush	limegreen	orchid	salmon	thistle3	
blue1	darkolivegreen2	goldenrod	gray54	grey6	grey66	lavender	lightyellow4	orangered4	saddlebrown	thistle2	
blue	darkolivegreen1	gold4	gray53	grey5	grey65	khaki4	lightyellow3	orangered3	royalblue4	thistle1	
blanchedalmond	darkolivegreen	gold3	gray52	grey4	grey64	khaki3	lightyellow2	orangered2	royalblue3	thistle	
black	darkmagenta	gold2	gray51	grey3	grey63	khaki2	lightyellow1	orangered1	royalblue2	tan4	
bisque4	darkkhaki	gold1	gray50	grey2	grey62	khaki1	lightyellow	orangered	royalblue1	tan3	
bisque3	darkgrey	gold	gray49	grey1	grey61	khaki	lightsteelblue4	orange4	royalblue	tan2	
bisque2	darkgreen	ghostwhite	gray48	grey0	grey60	ivory4	lightsteelblue3	orange3	rosybrown4	tan1	
bisque1	darkgray	gainsboro	gray47	grey	grey59	ivory3	lightsteelblue2	orange2	rosybrown3	tan	
bisque	darkgoldenrod4	forestgreen	gray46	green46	greenyellow	grey58	ivory2	lightsteelblue1	orange1	rosybrown2	steelblue4
beige	darkgoldenrod3	floralwhite	gray45	green4	green57	ivory1	lightsteelblue	orange	rosybrown1	steelblue3	
azure4	darkgoldenrod2	firebrick4	gray44	green3	grey56	ivory	lightslategray	olivedrab4	rosybrown	steelblue2	
azure3	darkgoldenrod1	firebrick3	gray43	green2	grey55	indianred4	lightslategray	olivedrab3	red4	steelblue1	
azure2	darkgoldenrod	firebrick2	gray42	green1	grey54	indianred3	lightslateblue	olivedrab2	red3	steelblue	
azure1	darkcyan	firebrick1	gray41	green	grey53	indianred2	lightskyblue4	olivedrab1	red2	springgreen4	
azure	darkblue	firebrick	gray40	gray100	grey52	indianred1	lightskyblue3	olivedrab	red1	springgreen3	
aquamarine4	cyan4	dodgerblue4	gray39	gray99	grey51	indianred	lightskyblue2	oldlace	red	springgreen2	
aquamarine3	cyan3	dodgerblue3	gray38	gray98	grey50	hotpink4	lightskyblue1	navyblue	purple4	springgreen1	
aquamarine2	cyan2	dodgerblue2	gray37	gray97	grey49	hotpink3	lightskyblue	navy	purple3	springgreen	
aquamarine1	cyan1	dodgerblue1	gray36	gray96	grey48	hotpink2	lightseagreen	navajowhite4	purple2	snow4	
aquamarine	cyan	dodgerblue	gray35	gray95	grey47	hotpink1	lightsalmon4	navajowhite3	purple1	snow3	
antiquewhite4	cornsilk4	dimgray	gray34	gray94	grey46	hotpink	lightsalmon3	navajowhite2	purple	snow2	
antiquewhite3	cornsilk3	dimgray	gray33	gray93	grey45	honeydew4	lightsalmon2	navajowhite1	powderblue	snow1	
antiquewhite2	cornsilk2	deepskyblue4	gray32	gray92	grey44	honeydew3	lightsalmon1	navajowhite	plum4	snow	
antiquewhite1	cornsilk1	deepskyblue3	gray31	gray91	grey43	honeydew2	lightsalmon	moccasin	plum3	slategray	
antiquewhite	cornsilk	deepskyblue2	gray30	gray90	grey42	honeydew1	lightpink4	mistyrose4	plum2	slategray4	
aliceblue	cornflowerblue	deepskyblue1	gray29	gray89	grey41	honeydew	lightpink3	mistyrose3	plum1	slategray3	
white	coral4	deepskyblue	gray28	gray88	grey40	grey100	lightpink2	mistyrose2	plum	slategray2	

除此之外，也可以用 HEX 颜色代码来定义颜色，比如：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry),
             color = "#6B7CD4", size = 5)
plot
```

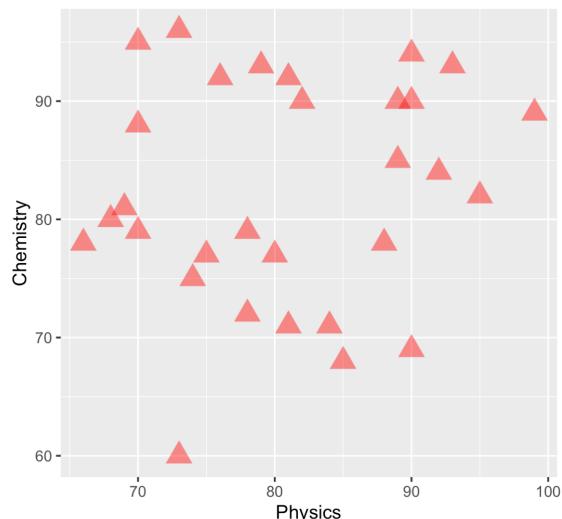
获取 HEX 颜色代码: <https://htmlcolorcodes.com/color-picker/>

另外，ggplot 中的 color 和 colour 两个拼写是通用的。

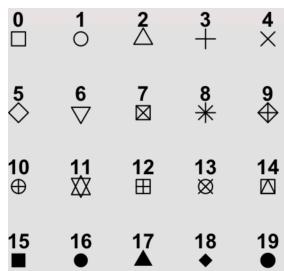
将散点改为三角形，并设置 50% 透明度：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry),
             color = "red", size = 5,
             shape = 17, alpha = 0.5)
#绘制所创建的对象
plot
```

输出结果：



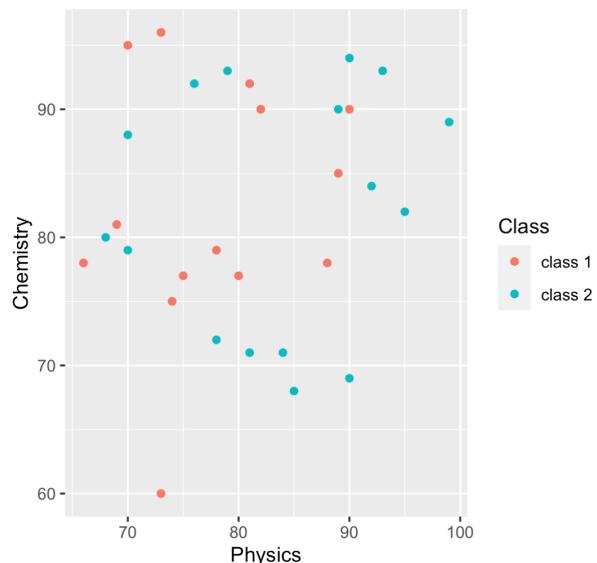
参考：ggplot2 的默认形状参数：



2. 用不同的颜色来显示学生所在的班

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class))
#绘制所创建的对象
plot
```

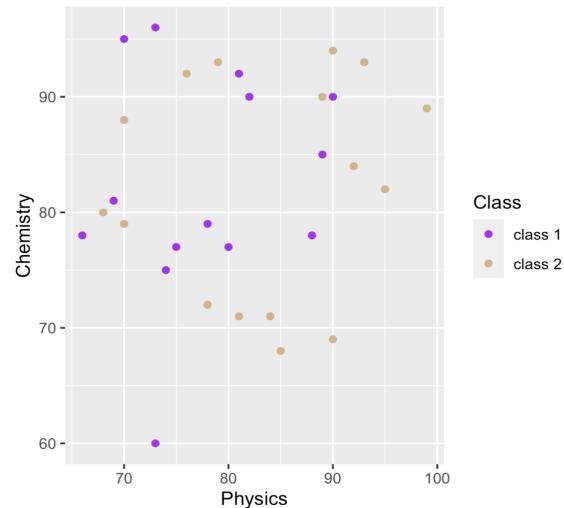
输出结果：



自定义分班的颜色：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class))+
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
  ))
#绘制所创建的对象
plot
```

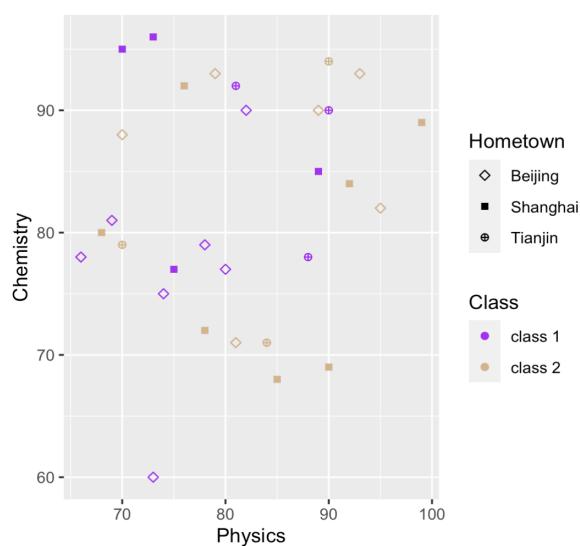
输出结果:



3. 在此基础上，用不同的形状来区分学生的籍贯，并自定义形状：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
  ))+
  scale_shape_manual(values=c(
    "Beijing" = 5,
    "Tianjin" = 10,
    "Shanghai" = 15
  ))
#绘制所创建的对象
plot
```

输出结果:



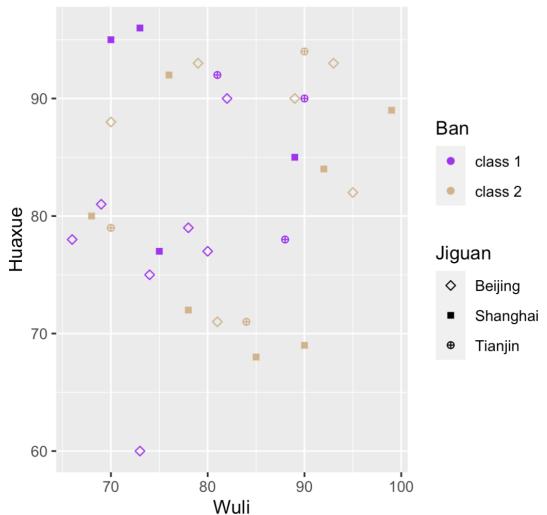
4. 将坐标轴和图例的标记改为拼音:

```

plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+ 
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
  ))+
  scale_shape_manual(values=c(
    "Beijing" = 5,
    "Tianjin" = 10,
    "Shanghai" = 15
  ))+
  labs(
    x = "Wuli",
    y = "Huaxue",
    shape = "Jiguan",
    color = "Ban"
  )
#绘制所创建的对象
plot

```

输出结果：



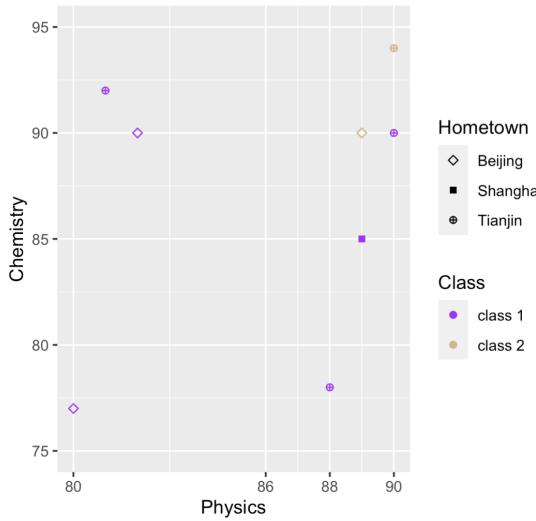
5. 仅显示物理 (x 轴) 80~90 分、化学 (y 轴) 75~95 分。并且，x 轴自定义刻度，y 轴自动绘制 5 个等距刻度。

```

plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+ 
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
  ))+
  scale_shape_manual(values=c(
    "Beijing" = 5,
    "Tianjin" = 10,
    "Shanghai" = 15
  ))+
  scale_x_continuous(limits = c(80,90),breaks = c(80,86,88,90))+ 
  scale_y_continuous(limits = c(75,95), breaks = scales::pretty_breaks(n = 5))
#绘制所创建的对象
plot

```

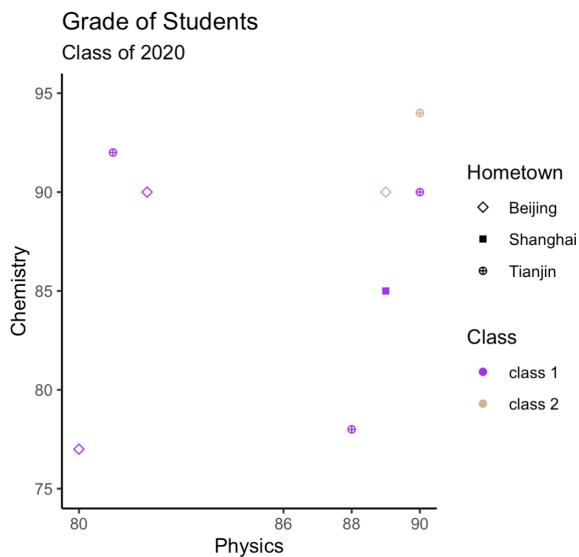
输出结果：



6. 添加标题、副标题，并且使用预设主题风格“classic”：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
  ))+
  scale_shape_manual(values=c(
    "Beijing" = 5,
    "Tianjin" = 10,
    "Shanghai" = 15
  ))+
  scale_x_continuous(limits = c(80,90),breaks = c(80,86,88,90))+ 
  scale_y_continuous(limits = c(75,95), breaks = scales::pretty_breaks(n = 5))+ 
  ggtitle("Grade of Students","Class of 2020")+
  theme_classic()
#绘制所创建的对象
plot
```

输出结果：



任务：请自行尝试其他预设主题风格，并观察各自的特色、思考适用的场景。

预设主题风格列表：

- `theme_gray()`

- theme_bw()
- theme_linedraw()
- theme_light()
- theme_dark()
- theme_minimal()
- theme_classic()
- theme_void()

7. 在预设主题风格的基础上，进行自定义调整。

比如，改变图例的位置，并添加网格线：

```
plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+  

  scale_color_manual(values=c(  

    "class 1" = "purple",  

    "class 2" = "tan"  

  ))+  

  scale_shape_manual(values=c(  

    "Beijing" = 5,  

    "Tianjin" = 10,  

    "Shanghai" = 15  

  ))+  

  scale_x_continuous(limits = c(80,90),breaks = seq(80,90,2))+  

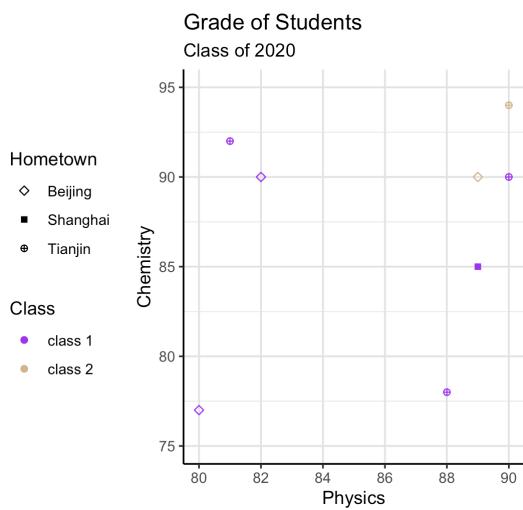
  scale_y_continuous(limits = c(75,95), breaks = scales::pretty_breaks(n = 5))+  

  ggtitle("Grade of Students","Class of 2020")+
  theme_classic()+
  theme(legend.position = "left",
        panel.grid.major.x = element_line(color = "gray90", linewidth = 0.5),
        panel.grid.major.y = element_line(color = "gray90", linewidth = 0.5),
        panel.grid.minor.y = element_line(color = "gray90", linewidth = 0.2))  

#绘制所创建的对象  

plot
```

输出结果：



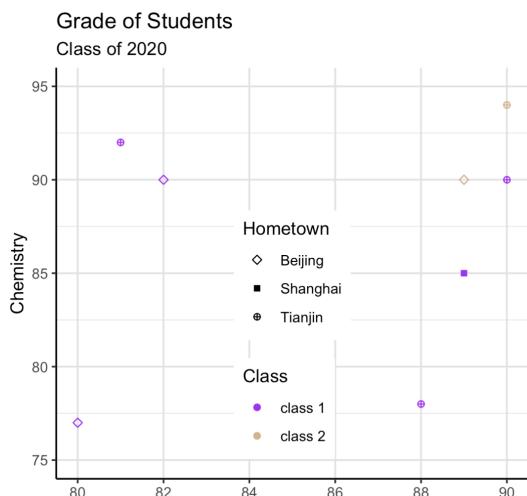
将图例放入图表内部，并去掉 x 轴的标题：

```

plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+ 
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
))+
  scale_shape_manual(values=c(
    "Beijing" = 5,
    "Tianjin" = 10,
    "Shanghai" = 15
))+
  scale_x_continuous(limits = c(80,90),breaks = seq(80,90,2))+ 
  scale_y_continuous(limits = c(75,95), breaks = scales::pretty_breaks(n = 5))+ 
  ggtitle("Grade of Students","Class of 2020")+
  theme_classic()+
  theme(legend.position = c(0.5,0.35),
        panel.grid.major.x = element_line(color = "gray90", linewidth = 0.5),
        panel.grid.major.y = element_line(color = "gray90", linewidth = 0.5),
        panel.grid.minor.y = element_line(color = "gray90", linewidth = 0.2),
        axis.title.x = element_blank())
#绘制所创建的对象
plot

```

输出结果：



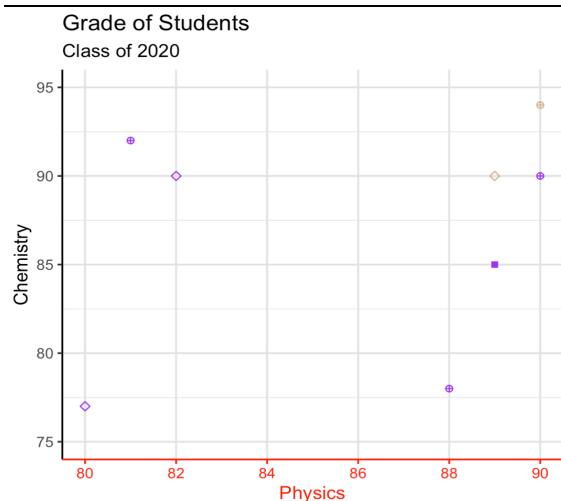
隐藏图例，并将 x 轴及其对应的标题和数字改为红色：

```

plot <- ggplot()+
  geom_point(data=example1, aes(x=Physics, y=Chemistry, color=Class, shape=Hometown))+ 
  scale_color_manual(values=c(
    "class 1" = "purple",
    "class 2" = "tan"
))+
  scale_shape_manual(values=c(
    "Beijing" = 5,
    "Tianjin" = 10,
    "Shanghai" = 15
))+
  scale_x_continuous(limits = c(80,90),breaks = seq(80,90,2))+ 
  scale_y_continuous(limits = c(75,95), breaks = scales::pretty_breaks(n = 5))+ 
  ggtitle("Grade of Students","Class of 2020")+
  theme_classic()+
  theme(legend.position = "none",
        panel.grid.major.x = element_line(color = "red", linewidth = 0.5),
        panel.grid.major.y = element_line(color = "red", linewidth = 0.5),
        panel.grid.minor.y = element_line(color = "red", linewidth = 0.2),
        axis.title.x = element_text(color="red"),
        axis.line.x = element_line(color="red"),
        axis.text.x = element_text(color="red"),
        axis.ticks.x = element_line(color="red"))
#绘制所创建的对象
plot

```

输出结果：



例 2：绘制折线图

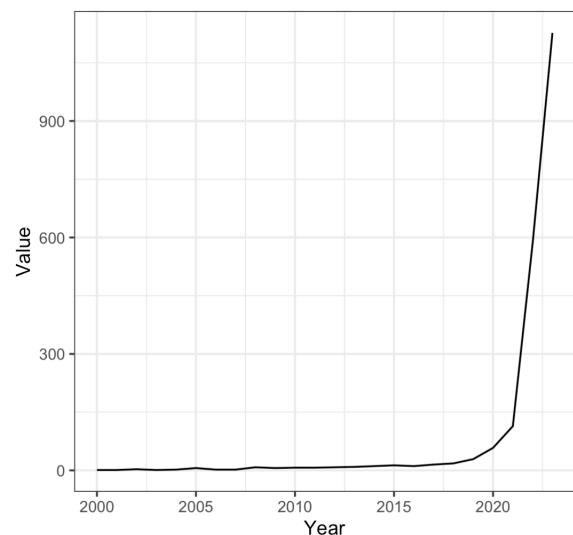
导入数据 `example2.csv`, 该数据有 3 列, 第一列为年份, 第二列 (`ts1`) 和第三列 (`ts2`) 分别为一组时间序列数据。

1. 首先, 绘制 `ts1` 的变化趋势:

```
library(readr)
# 导入数据
example2 <- read_csv("~/Documents/数据分析和可视化/data/lab2/example2.csv")

plot <- ggplot() +
  geom_line(data=example2, aes(x=year, y=ts1)) +
  theme_bw() +
  labs(
    x = "Year",
    y = "Value"
  )
# 绘制所创建的对象
plot
```

输出结果:



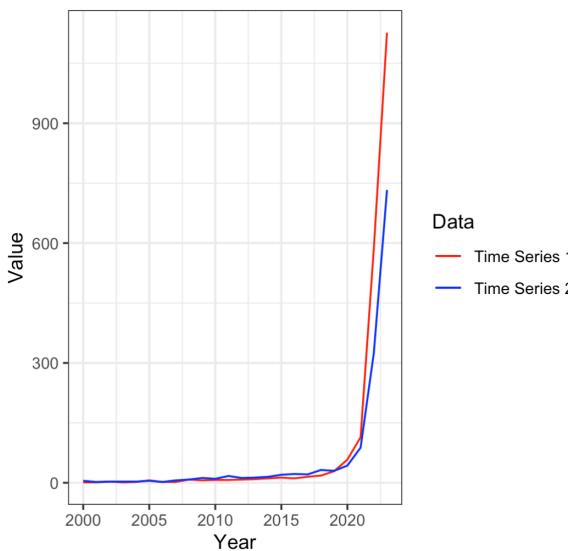
2. 添加 `ts2` 数据, 并定义它们的颜色:

```

plot <- ggplot()+
  geom_line(data=example2, aes(x=year, y=ts1,
                                color = "Time Series 1"))+
  geom_line(data=example2, aes(x=year, y=ts2,
                                color = "Time Series 2"))+
  scale_color_manual(values = c(
    "Time Series 1" = "red",
    "Time Series 2" = "blue"
  ))+
  theme_bw()+
  labs(
    x = "Year",
    y = "Value",
    color = "Data"
  )
#绘制所创建的对象
plot

```

输出结果：



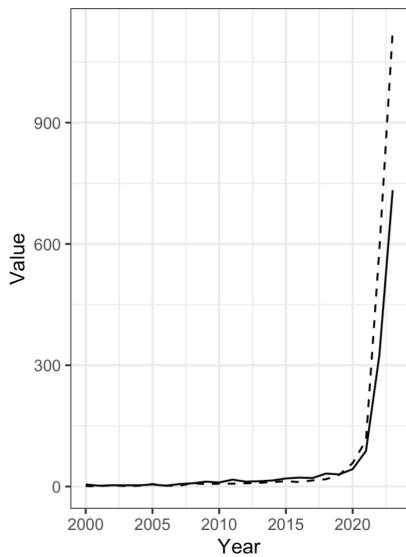
3. 或者，我们也可以用不同的线条样式来表示不同的时间序列：

```

plot <- ggplot()+
  geom_line(data=example2, aes(x=year, y=ts1,
                               linetype = "Time Series 1"))+
  geom_line(data=example2, aes(x=year, y=ts2,
                               linetype = "Time Series 2"))+
  scale_linetype_manual(values = c(
    "Time Series 1" = "dashed",
    "Time Series 2" = "solid"
  ))+
  theme_bw()+
  labs(
    x = "Year",
    y = "Value",
    linetype = "Data"
  )
#绘制所创建的对象
plot

```

输出结果：



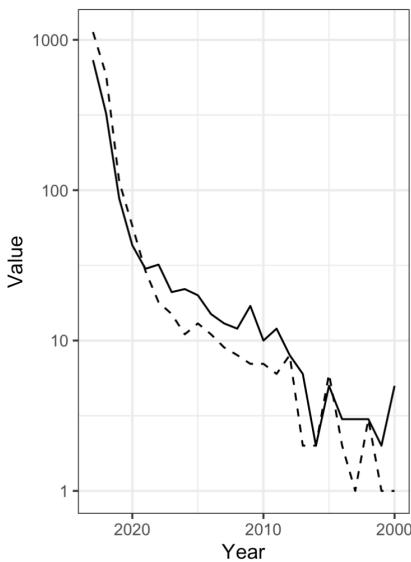
R 的预设线条样式:

'solid'	—————
'dashed'	- - - - -
'dotted'	· · · · ·
'dotdash'	- · - - -
'longdash'	- - - - -
'twodash'	- - - - -

4. 该数据中，在 2020 年以前，有大量的数据集中在较低的水平，目前的图表不利于展现它们的变化。我们可以把 y 轴改为对数 (\log_{10}) 比例。此外，在一些领域，我们习惯于把距今较近的时间放在时间轴的左侧，因此需要翻转 x 轴:

```
plot <- ggplot()+
  geom_line(data=example2, aes(x=year, y=ts1,
                                linetype = "Time Series 1"))+
  geom_line(data=example2, aes(x=year, y=ts2,
                                linetype = "Time Series 2"))+
  scale_linetype_manual(values = c(
    "Time Series 1" = "dashed",
    "Time Series 2" = "solid"
))+
  theme_bw()+
  labs(
    x = "Year",
    y = "Value",
    linetype = "Data"
) +
  scale_y_continuous(trans = "log10")+
  scale_x_reverse(limits = c(2023, 2000), breaks = scales::pretty_breaks(n = 3))
#绘制所创建的对象
plot
```

输出结果:



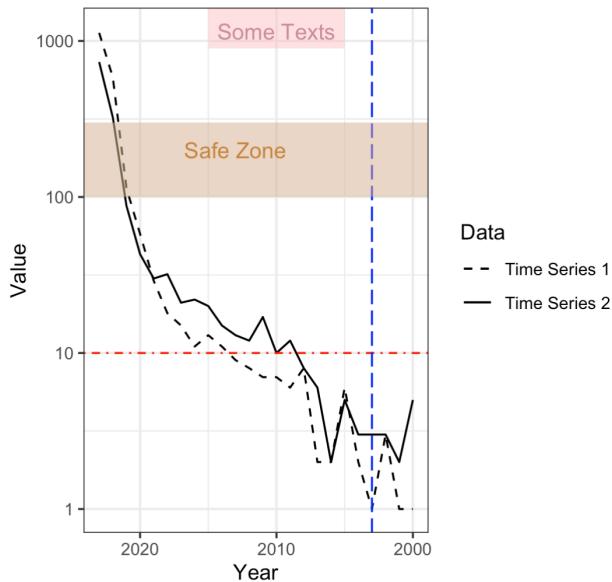
5. 添加辅助线和其他辅助标志:

```

plot <- ggplot()+
  geom_line(data=example2, aes(x=year, y=ts1,
                                linetype = "Time Series 1"))+
  geom_line(data=example2, aes(x=year, y=ts2,
                                linetype = "Time Series 2"))+
  scale_linetype_manual(values = c(
    "Time Series 1" = "dashed",
    "Time Series 2" = "solid"
))+
  theme_bw()+
  labs(
    x = "Year",
    y = "Value",
    linetype = "Data"
) +
  geom_hline(yintercept = 10, linetype = "dotdash", color = "red")+
  geom_vline(xintercept = 2003, linetype = "longdash", color = "blue")+
  geom_rect(aes(xmax = 2015, xmin = 2005, ymax = Inf, ymin = 900),
            color = NA, fill = "pink", alpha = 0.5)+ 
  geom_rect(aes(xmax = Inf, xmin = -Inf, ymax = 100, ymin = 300),
            color = NA, fill = "tan", alpha = 0.5)+ 
  geom_text(aes(x = 2010,y = 1150,label="Some Texts"), size=4, color = "pink3") +
  geom_text(aes(x = 2013,y = 200,label="Safe Zone"), size=4, color = "tan3") +
  scale_y_continuous(trans = "log10")+
  scale_x_reverse(limits = c(2023,2000), breaks = scales::pretty_breaks(n = 3))
#绘制所创建的对象
plot

```

输出结果:



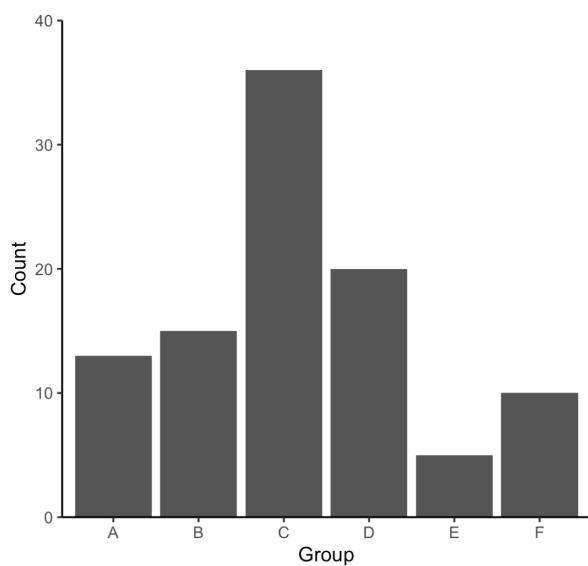
例 3：绘制柱状图和饼图

导入 example3.csv，该数据有两列，第一列为分组，第二列为每组的数值。

1. 绘制柱状图：

```
plot <- ggplot() +
  geom_bar(data=example3, aes(x=Group, y=Count), stat = "identity")+
  theme_classic()+
  scale_y_continuous(limits = c(0,40), expand = c(0,0))
#绘制所创建的对象
plot
```

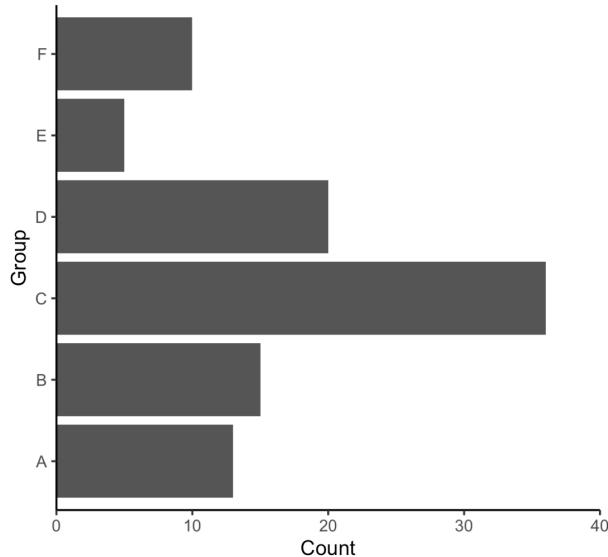
输出结果：



2. 改为条形图：

```
plot <- ggplot() +
  geom_bar(data=example3, aes(x=Group, y=Count), stat = "identity")+
  theme_classic()+
  scale_y_continuous(limits = c(0,40), expand = c(0,0))+ 
  coord_flip()
#绘制所创建的对象
plot
```

输出结果:

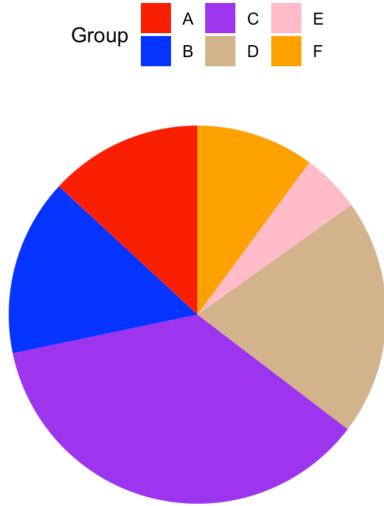


3. 绘制饼图:

```
plot <- ggplot() +
  geom_bar(data=example3, aes(x="", y=Count, fill=Group),
           stat="identity", width=1) +
  coord_polar("y", start=0)+ 
  scale_fill_manual(values=c(
    "A" = "red",
    "B" = "blue",
    "C" = "purple",
    "D" = "tan",
    "E" = "pink",
    "F" = "orange"
))+
  ggtitle("Pie Chart")+
  theme_void()+
  theme(legend.position = "top")
#绘制所创建的对象
plot
```

输出结果:

Pie Chart



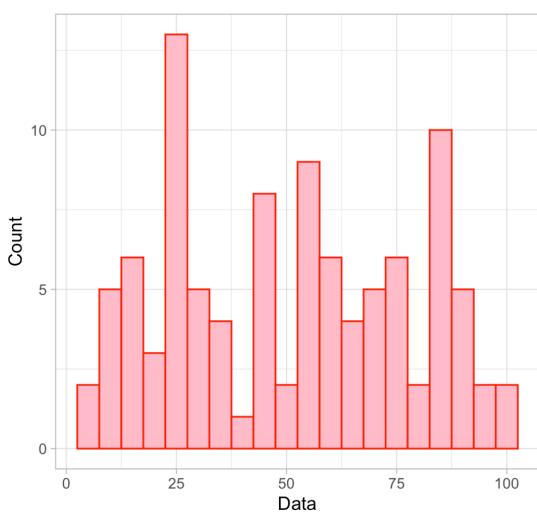
例 4：绘制直方图、密度图、箱线图、小提琴图

导入 example4.csv，该数据有 3 列，第一列为数据点的 ID，第二列为数据值，第三列为数据点的分组。

1. 绘制所有数据的直方图：

```
plot <- ggplot() +
  geom_histogram(data=example4,aes(x=Data),
                 color="red", fill="pink", binwidth=5) +
  labs(x = "Data",
       y = "Count") +
  theme_light()
#绘制所创建的对象
plot
```

输出结果：



尝试改变 binwidth 的大小：有什么效果？

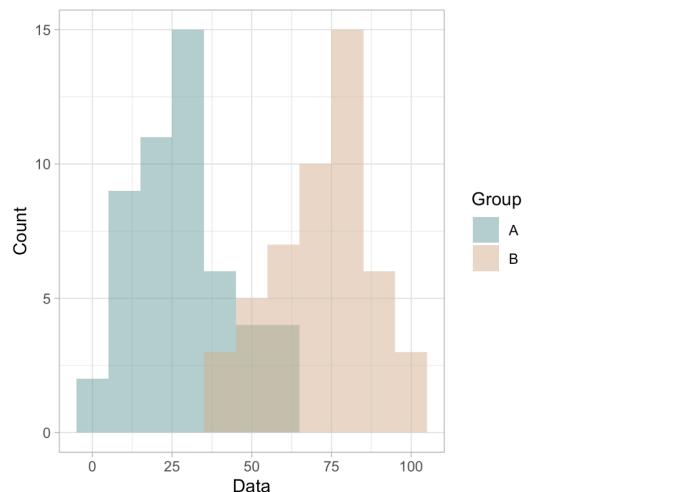
2. 分组的直方图：

```

plot <- ggplot() +
  geom_histogram(data=example4, aes(x=Data, fill=Group),
                 color=NA, binwidth=10, alpha=0.5, position = 'identity')+
  labs(x = "Data",
       y = "Count",
       fill = "Group")+
  scale_fill_manual(values=c(
    "A" = "#cadetblue",
    "B" = "#tan"
  ))+
  theme_light()
#绘制所创建的对象
plot

```

输出结果:



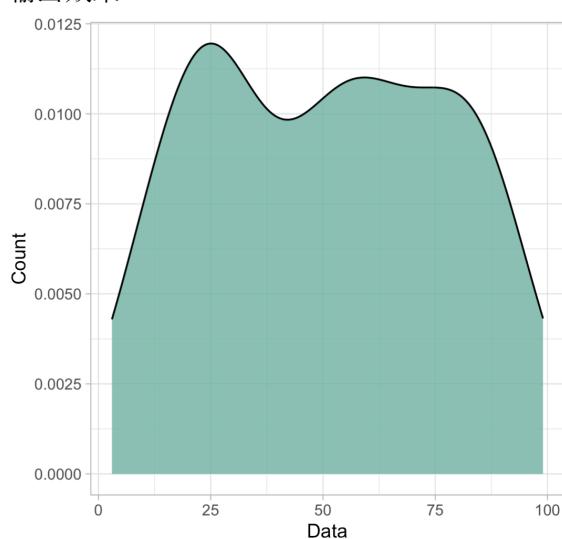
3. 所有数据的密度图:

```

plot <- ggplot() +
  geom_density(data=example4, aes(x=Data),
               fill="#69b3a2", color="black", alpha=0.8)+ 
  labs(x = "Data",
       y = "Count")+
  theme_light()
#绘制所创建的对象
plot

```

输出效果:



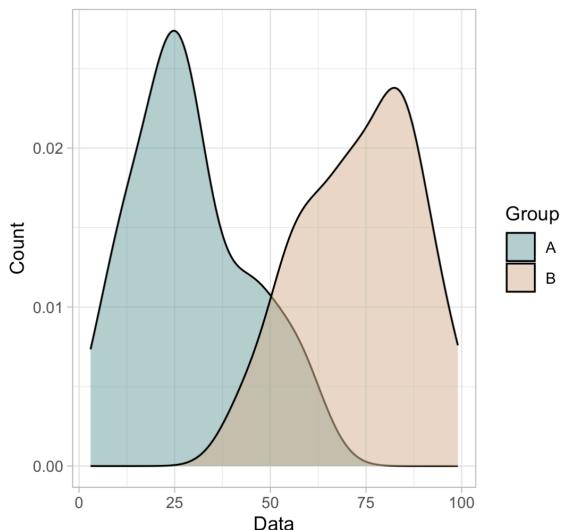
4. 分组的密度图:

```

plot <- ggplot() +
  geom_density(data=example4, aes(x=Data, fill=Group, group=Group),
               color="black", alpha=0.5) +
  labs(x = "Data",
       y = "Count",
       fill = "Group") +
  scale_fill_manual(values=c(
    "A" = "cadetblue",
    "B" = "tan"
  )) +
  theme_light()
#绘制所创建的对象
plot

```

输出结果:



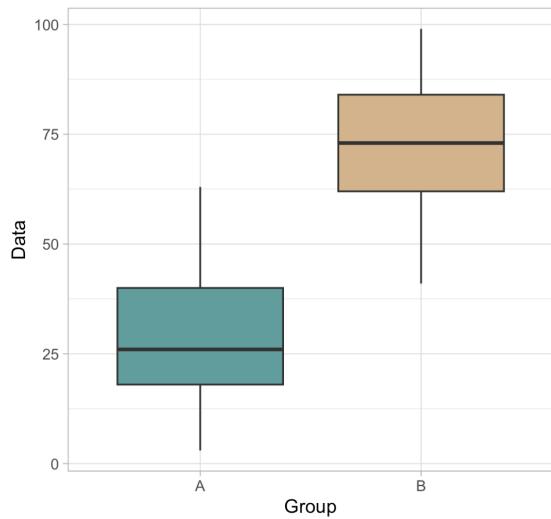
5. 绘制箱线图

```

plot <- ggplot() +
  geom_boxplot(data=example4, aes(x=Group, y=Data, fill=Group)) +
  labs(x = "Group",
       y = "Data",
       fill = "Group") +
  scale_fill_manual(values=c(
    "A" = "cadetblue",
    "B" = "tan"
  )) +
  theme_light() +
  theme(legend.position = "none")
#绘制所创建的对象
plot

```

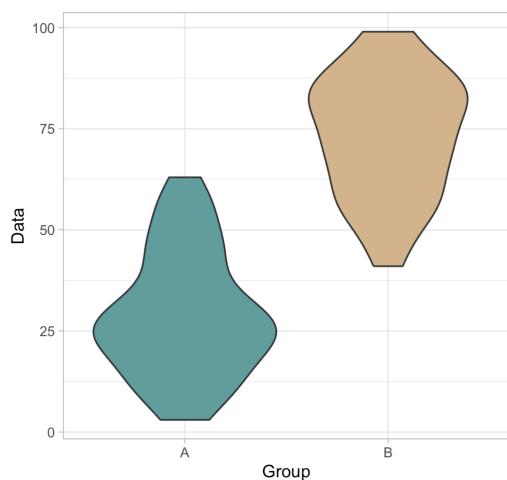
输出结果:



6. 绘制小提琴图:

```
plot <- ggplot() +  
  geom_violin(data=example4, aes(x=Group, y=Data, fill=Group))+  
  labs(x = "Group",  
       y = "Data",  
       fill = "Group") +  
  scale_fill_manual(values=c(  
    "A" = "cadetblue",  
    "B" = "tan"  
) +  
  theme_light() +  
  theme(legend.position = "none")  
#绘制所创建的对象  
plot
```

输出结果:



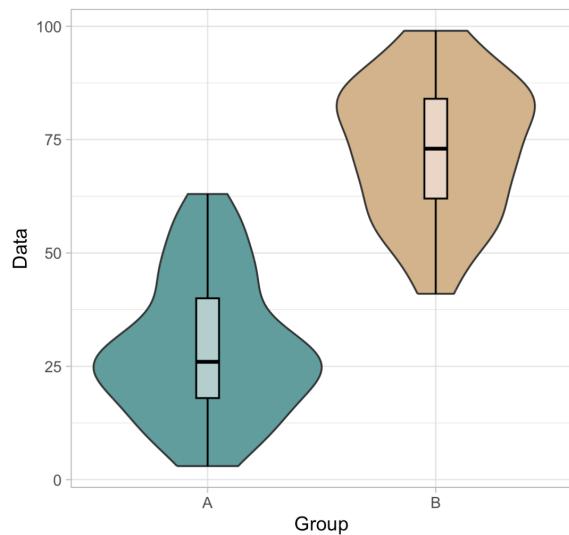
7. 小提琴图叠加箱线图:

```

plot <- ggplot() +
  geom_violin(data=example4, aes(x=Group, y=Data, fill=Group),
              width=1)+
  geom_boxplot(data=example4, aes(x=Group, y=Data), fill="white",
               color = "black",alpha=0.5, width=0.1)+
  labs(x = "Group",
       y = "Data",
       fill = "Group")+
  scale_fill_manual(values=c(
    "A" = "cadetblue",
    "B" = "tan"
))+
  theme_light()+
  theme(legend.position = "none")
#绘制所创建的对象
plot

```

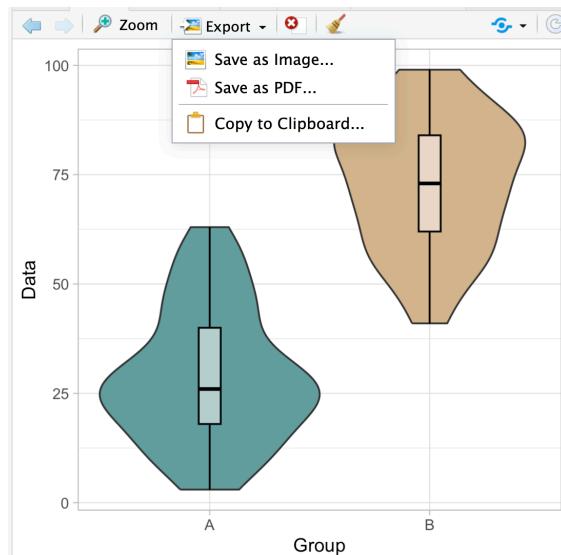
输出结果：



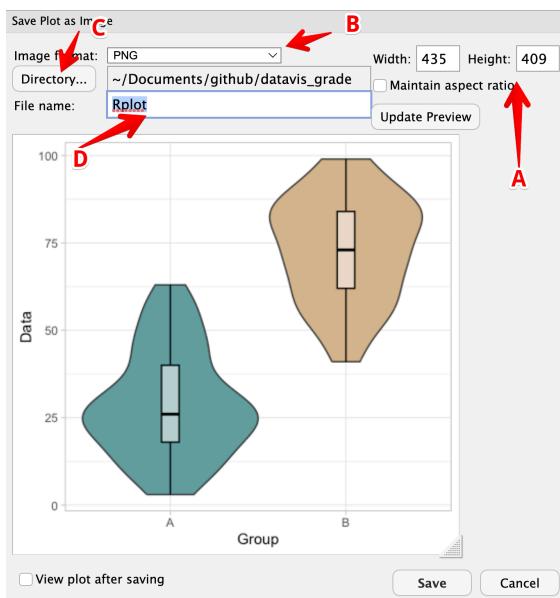
储存图像

方法一：使用 RStudio 的图形界面

1. 点击 Export，选择 Save as Image (或 PDF)



2.A 设置图片的长宽大小（单位为像素）；B 选择图片格式；C 选择储存路径；D 输入图片文件名。



缺点：无法输出高清图片。

方法二：运行 `ggsave` 函数自动输出高清图片

```
ggsave(
  "new_plot.png", #图片文件名
  plot = plot, #要输出的对象，如果需要输出最近一张图，则可以省略这一参数
  path = "xxx/data/lab2/img", #储存路径
  width = 100, #宽度
  height = 100, #高度
  units = "mm", #尺寸单位，可选单位包括英寸"in"、厘米"cm"、毫米"mm"、像素"px"
  dpi = 300, #清晰度（每英寸多少个像素点），印刷出版物通常要求300以上，顶级学术期刊可能要求600以上
  bg = "transparent" #设定透明背景，如果不需要透明背景，则省略这一参数
)
```

中文字体

R 语言中的中文处理涉及到 UTF，以及中文字体的安装和针对系统的设置调试。

查看现有字体，可引入 `systemfonts` 安装包，然后用 `system_fonts()` 函数：

```
ft <- data.frame(system_fonts())
```

在未改变预设的 MacOS 系统中，可以在 `theme` 函数里把文字类元素指定为系统预设的中文字体 `family`：

```
theme(axis.title.x = element_text(family = "Kai"))
```

MacOS 可选的预设中文字体：楷体（Kai 或 STKaiti 或 Kaiti SC），黑体（Hei 或 STHeiti），宋体（Songti SC 或 STSong），细黑（STXihei），行楷（Xingkai SC），苹方（Pingfang SC），圆体（Yuanti SC），仿宋（STFangsong），娃娃体（Wawati SC），兰亭黑（Lantinghei SC），报隶（Baoli SC），凌慧体（LingWai SC），钢笔手写体（HanziPen SC），魏碑体（Weibei SC），隶变（Libian SC），雅痞（Yuppy SC）。以上字体 SC 为简体中文的意思，如果改为 TC 则为相应的繁体中文字体。 ST 是简繁体通用的字体。

同样，该方法也可以用于改变英文字体或使用其他语言的字体。常见的预设英文字体包括 Arial, Times New Roman, Georgia, Mali, Chalkboard, Helvetica, KoHo, Gill Sans, Avenir, American Typewriter, Courier New, Krub, Lava Kannada, Phosphate 等。

Windows 系统中使用中文字体通常需要先对字体进行安装和加载，或引入一些额外的 library (比如 showtext)。如果 MacOS 系统为旧版或改变过预设，或需要使用其他罕见的中文字体，也需要安装和加载字体。

本课程不要求掌握 ggplot2 的中文标注，如果有兴趣或有相关需求，可以参考：

https://blog.csdn.net/Bruce_Kong/article/details/113819950
<https://r-lover.com/tutorial/visualization/chinese-in-ggplot2/>
<https://cloud.tencent.com/developer/article/1036396>

补充：数据预处理

补充一些本次作业可能会用到的数据预处理方法。

对齐合并数据框：

当两个数据框的行数相同时，可以利用 cbind (column bind) 函数直接合并。比如：

```
# 创建一个3行2列的数据框
df1 <- data.frame(names = c("A", "B", "C"),
                   data1 = c(1, 2, 3))

# 创建一个3行3列的数据框
df2 <- data.frame(var = c("F", "G", "H"),
                   num1 = c(NA, NA, 3),
                   num2 = c(0, 0, 0))

# 合并以上两个数据库
df <- cbind(df1, df2)
print(df)
```

数据框也可以和 array 合并，前提是 array 的长度等于数据框的行数。比如：

```
# 创建一个长度为3的array
char <- c("Test 1", "Test 2", "Test 3")
# 将array与之前的数据框df合并
df <- cbind(char, df)
print(df)

# 也可以三个以上的数据框或array直接合并
df <- cbind(df1, char, df2)
print(df)
```

相似地，当两个数据框的列数相同，且每列的名称相同时，可以用 rbind (row bind) 函数进行合并。比如：

```
# 创建一个2列2行的数据框，两列的名称分别为names和data
df1 <- data.frame(names = c("A", "B"),
                   data = c(0, 2))

# 创建一个2列3行的数据框，两列的名称分别为names和data
df2 <- data.frame(names = c("A", "C", "D"),
                   data = c(0, 3, 5))

# 合并以上两个数据库
df <- rbind(df1, df2)
print(df)
```

同样的道理，array 也可以作为新的一行，与数据框合并，前提是 array 的长度等于数据框的列数。比如：

```
#创建一个长度为2的array  
char = c("X", 1)  
#将array与之前的数据框df合并  
df <- rbind(df, char)  
print(df)
```

提取数据框中的某些行和列：

```
#创建一个示例数据框  
df <- data.frame(var1 = c(1,2,3,4,5),  
                   var2 = c(2,3,4,5,6),  
                   var3 = c(0,0,0,0,0),  
                   var4 = c(1,1,1,1,1))  
#提取第2-3列，组成一个新的数据框  
df2 = df[,2:3]  
print(df2)  
#提取第2-4行，组成一个新的数据框  
df3 = df[2:4,]  
print(df3)
```

反转数据框的行和列：

通常可以使用 `data.frame` 包中的 `transpose()` 函数。

```
library(data.table)  
#创建一个示例数据框  
df <- data.frame(year = c(2001, 2002, 2003, 2004),  
                   data = c(100, 200, 300, 400),  
                   note = c(NA, NA, "Note", NA))  
#提取各列的名称，以备后用  
cols <- colnames(df)  
#反转df的行和列  
df = transpose(df)  
#将原本各列的名称设为反转后各行的名称  
rownames(df) <- cols  
#或者，将原本各列的名称作为新的一列，插入反转后的数据框  
df <- cbind(cols, df)
```

作业部分

根据数据集回答以下问题，并附上合适的可视化图表。图表不以“美观”作为评判标准，因此主题和色彩等可以自由发挥。但是，可视化图表必须做到有效地传达数据和信息，需要重视图例、图表标题、坐标轴及其刻度、坐标轴的标题（包括单位，如果涉及的话）等要素（可参考样例题）。每一道题只画一张图。

数据集 1：Edu_Expenditure_GDP.csv

该数据集来源于 UNESCO（联合国教科文组织）和世界银行，其中含有 12 个国家在 2010-2020 年期间的教育支出占全国 GDP 的百分比。该数据集分 3 列：

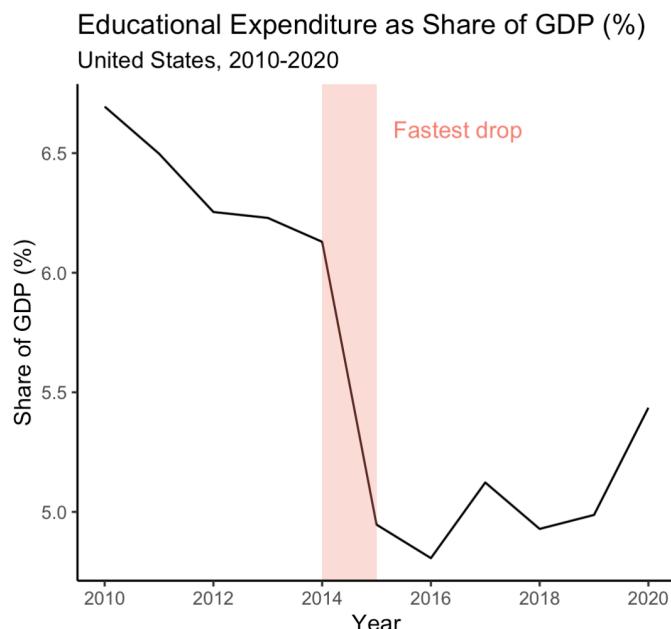
- **Country:** 国家
- **Year:** 年份
- **Expenditure:** 教育支出占 GDP 的比重（单位为%）

样例：

问题：从 2010 年到 2020 年，美国（United States）的教育支出占 GDP 的比重增加了还是减少了？其中最明显的变化出现在什么时候？

回答：如图所示，图中的黑色曲线显示了美国教育支出占 GDP 比重和年份之间的关系。从 2010 年到 2020 年期间，这条曲线总体呈下降趋势，因此美国教育支出占 GDP 的比重在此期间减少了。其中，最明显的变化发生在 2014-2015 年期间（如图中红色阴影区域所示）。

附图：



问题：

1. 2020 年时，这 12 个国家中，哪个国家的教育支出占全国 GDP 的比重最大？
2. 总体而言，中国（China）在 2010-2020 年期间的教育支出占 GDP 的比重，与日本（Japan）更接近，还是与法国（France）更接近？
3. 在哪一年里，泰国（Thailand）的教育支出占 GDP 的比重高于意大利（Italy）？

数据集 2：Public_School_Revenue_USA.csv

该数据集来源于美国人口普查局（US Bureau of the Census）和美国国家教育统计中心（NCES），包含美国在 1990-2010 年期间不同层级的政府对公立学校的经费支持情况。数据集有 4 列：

- Year: 年份
- Local: 地方政府（县政府和市政府）拨给公立学校的经费占美国当年 GDP 总量的百分比
- State: 州级政府拨给公立学校的经费占美国当年 GDP 总量的百分比
- Federal: 联邦政府拨给公立学校的经费占美国当年 GDP 总量的百分比

后三列相加后，可视为历年美国公立教育的总拨款占美国当年 GDP 总量的百分比。

问题：

4. 在 2000 年，美国的三级政府中，哪一级拨给公立学校的经费占公立教育总拨款的比例最大？
5. 在哪些年份里，地方政府对公立学校的拨款少于州级政府？

数据集 3：WDI_Education.csv

该数据集为世界银行的世界发展指数（World Development Indicators）提供的部分 2015 年初等（小学）教育相关数据。该数据集有 8 列：

- country: 国家
- primary_school_completion_rate: 25 岁以上人口中的小学毕业率（有小学文凭的人数占比）
- primary_school_enroll_rate: 适龄人口中的小学入学率
- trained_teacher: 小学教师中，受过系统培训人数的比例
- compulsory_education: 义务教育年限
- poverty_ratio: 极端贫困率（每日收入在 2.75 美元以下人群占总人口的比例）
- region: 国家所属区域
- income_class: 国家收入水平

问题：

6. 该数据集中收录的国家被归为了哪几个区域？每个区域各有多少个国家？
7. 哪个区域的国家，义务教育年数的中位数最大？
8. 25 岁以上人口的小学毕业率是否和国家收入水平相关？
9. 适龄人口小学入学率和极端贫困率之间是否有相关性？
10. 是否有某个区域的国家，其极端贫困率普遍较高，且 25 岁以上人口小学毕业率普遍较低？

提交内容

在学习通提交对以上 **10** 个问题的回答及附图。回答需简洁并和图表对应，具体格式不限。图必须用 `ggplot2` 绘制，图的种类不限、样式不限，但必须要适用于相关的问题，并和文字相互呼应，有效地传达信息（参见样例）。

图表中的标注内容可用中文，但不做强制要求。系统没有预设中文字体的可以写英文或拼音（如果写拼音，请按词语分割，比如“中国的教育支出”：Zhongguo De Jiaoyu Zhichu）。

每个问题满分 **10** 分，共计 **100** 分。请将所有 R 代码（包括数据预处理和绘图）汇总在同一个 R 文档里，并写好注释，作为问题 1的附件上传到学习通。R 文档请命名为“**Lab2+姓名+学号**”，比如“**Lab2 张三 1010101010.R**”。

截止时间：

2023 年 12 月 10 日（周日）北京时间 23:59。