

Lab 3：基础地图的绘制

Shapefile 格式

绘制地图时所用的矢量类地理空间数据主要基于 **Shapefile** 格式的文件。**Shapefile** 是地理信息领域最著名的 **Esri** 公司开发的一种矢量数据文件格式，通常用于地理空间分析。**Shapefile** 文件存储了地理要素的位置、几何形状和属性。**Shapefile** 中的地理要素(**features**)可表示为点、线或多边形三种图形类型。

点类数据可以用来表示面积或长度可以忽略不计的地址或地点，例如在世界地图尺度下的城市。线类数据通常用于描述道路、河流、断层、移动路径等线性地理要素。多边形类数据可以表示某个尺度下面积不能被忽略的地理要素，比如世界地图尺度下的国家和大型湖泊，或者城市地图尺度下的公园、街区、学校等。

Shapefile 通常以一组文件的形式出现。除了储存点、线或多边形三种图形数据的.**.shp** 以外，还必须包括用于储存图形索引和属性数据的.**.shx** 与.**.dbf** 文件，这些文件除了后缀以外，其名称应该相同，并储存在同一个路径下，否则难以被读取。此外，**Shapefile** 还可能包括指示地理参考系的.**.prj** 以及一些其他的非必要文件。本次练习及作业部分所用的 **Shapefile** 已将相应的文件汇总于同名的.**.zip** 压缩包里，使用前需要先解压缩。

Shapefile 文件可以用 **ArcGIS** 软件读取和处理，也可以被开源软件 **QGIS** 处理。在 **QGIS** 中，**Shapefile** 可以和其他矢量地理数据文件（比如 **geojson** 等）相互转化。在 **Python** 和 **R** 等计算机语言中也有相应的 **library** 可以处理 **Shapefile** 文件。本次练习中，我们将使用 **R** 中的 **sf** 库来处理 **Shapefile**。

地图的分类

根据不同的用途，常见的地图可以大致被分为通用地图（**general map**）和专题地图（**thematic map**）两大类。通用地图又分为地形图（**topographic map**）和概览图（**reference map**），其中地形图以展现某个地区的地形貌为主，通常会包括等高线，或根据高程来着色；概览图则主要反映某个地区的行政区划、居民点、水系和道路等地理知识。专题地图主要是展现某个领域的专业知识，比如人口地图、经济地图、地质图、气候图、洋流图、历史地图、候鸟迁徙图、航班线路图等。

例 1：概览图

- 安装并导入 **sf**。使用 **sf** 中的 **st_read()** 导入中国的版图数据 **china.shp**，并在 **RStudio** 中预览。

```
# 导入中国版图的shapefile数据
china <- sf::st_read("data/china/china.shp")
# 预览china数据
View(china)
# 查看china的数据结构
typeof(china)
# 查看china的基本信息
print(china)
```

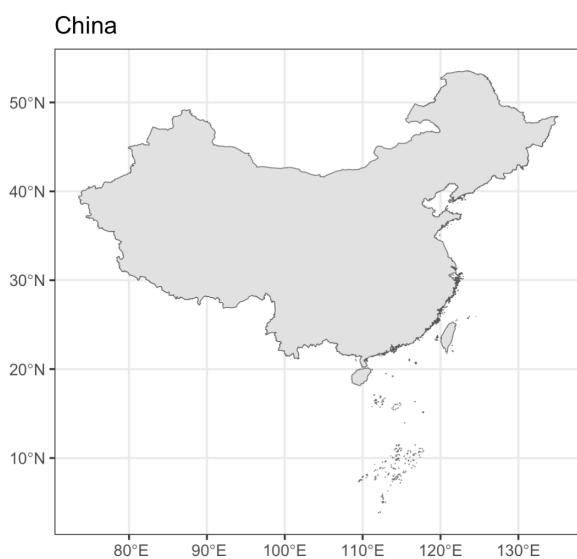
```
Simple feature collection with 1 feature and 2 fields
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:  xmin: 73.50235 ymin: 3.83703 xmax: 135.0957 ymax: 53.56362
Geodetic CRS:  WGS 84
  cn_adcode cn_name                      geometry
1    100000   China MULTIPOLYGON (((112.0607 3....
```

从基本信息中，我们可以看到该数据的种类（多边形-multipolygon）、数据及参数的多少（1 个多边形、2 个参数）、维度（2 维）、空间覆盖范围以及大地坐标系（WGS 84）。我们也能看到参数的具体情况：该数据集的两个参数为 `cn_adcode` 和 `cn_name`。第三列的 `geometry` 是多边形的图形数据，不是常规的属性。

2. 利用 `ggplot2` 内专门针对 `Shapefile` 数据的 `geom_sf()` 函数，绘制一幅最简单的中国地图。

```
#利用ggplot绘制中国地图
p <- ggplot() +
  geom_sf(data = china) +
  ggtitle("China") +
  theme_bw()
#绘制地图
p
```

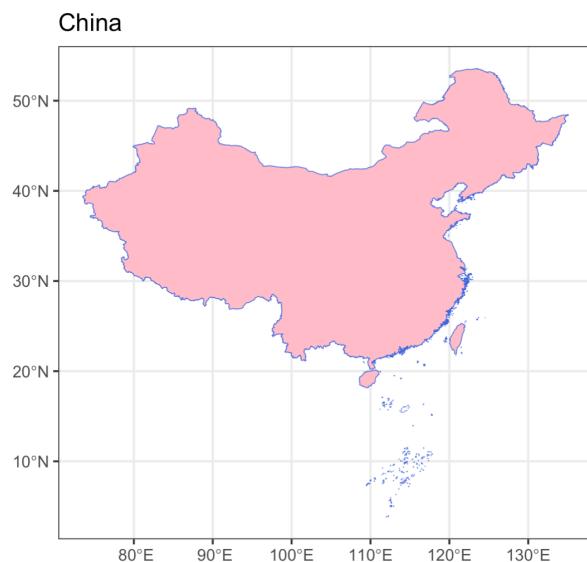
输出结果：



3. 和绘制常规的统计图表一样，我们可以在 `geom_sf()` 内调整图形的颜色等风格。

```
#改变地图的颜色
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  ggtitle("China") +
  theme_bw()
#绘制地图
p
```

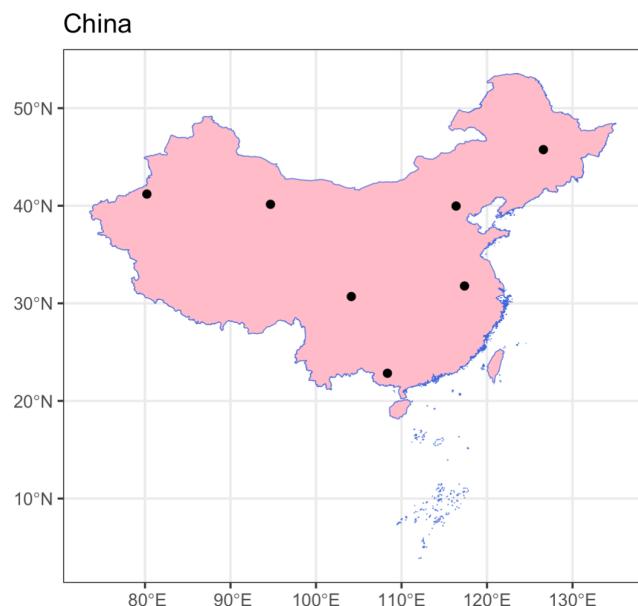
输出结果：



4. 目前，中国的轮廓是这幅地图的一个图层。在此基础上，我们可以增加图层，让地图的信息更丰富。例如，我们可以导入城市数据 `cities.csv`，该数据集包含了一些城市的经纬度坐标。我们可以按照常规散点图的方式，将这些城市按经纬度绘制到地图上，作为一个新的图层。

```
#导入城市数据
cities <- read_csv("data/china/chinese_cities.csv")
#将城市数据添加到地图
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  geom_point(data=cities,aes(x=lon,y=lat))+
  ggttitle("China") +
  theme_bw()+
  theme(axis.title = element_blank())
#绘制地图
p
```

输出结果：

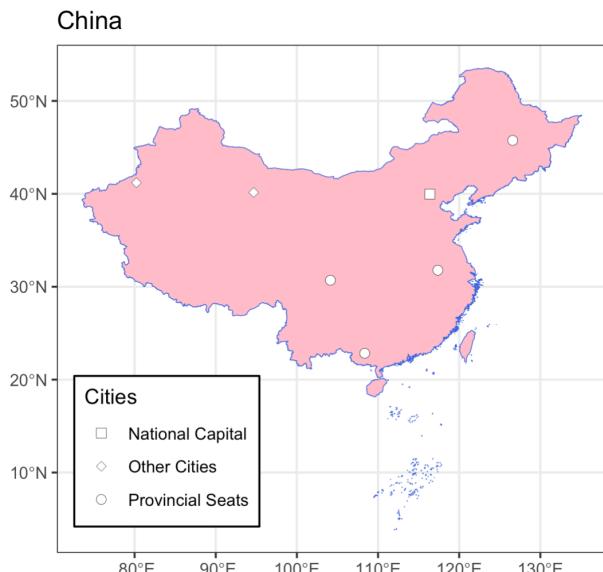


5. 当前，城市数据作为一个新的图层，被叠加到了地图上。我们可以添加视觉通道，让城市数据显示更加丰富的内容。比如，利用不同的形状显示不同的城市分类。我们也可以采用叠加两层点状图层、并设置略微不同的 `size`

的方式，在点状图层中起到类似于描边的效果。

```
#按照城市类型分类
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  #点数据分两层绘制，可以起到描边效果
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2) +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2) +
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  )) +
  labs(shape="Cities") +
  theme_bw() +
  theme(legend.position = c(0.2,0.2),
        legend.background = element_rect(color="black"),
        axis.title = element_blank())
#绘制地图
p
```

输出结果：



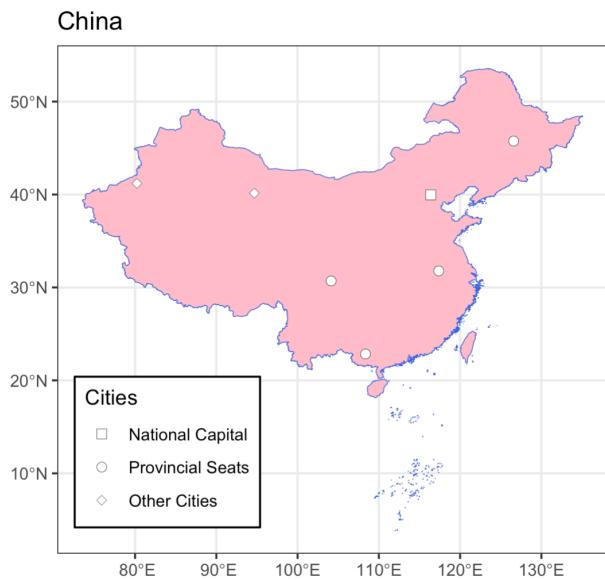
6. 仔细观察这幅地图，我们会发现一个问题：在图例中，从上到下分别是首都、其他城市、省会城市。这显然不符合通常的排列逻辑。我们希望图例按照首都、省会城市、其他城市的顺序排列。

这里，我们可以利用 `factor` 数据结构来处理城市数据的 `type` 一列，达到我们的目的。

在上述代码之前，插入以下代码处理 `cities` 数据，然后再次运行上述代码绘图，并观察结果。这是一种常见的处理方式，不仅可用于绘制地图，也可在绘制常规统计图表时使用。

```
#手动调整图例的排序(利用factor数据结构)
cities$type <- factor(cities$type, levels = c("National Capital",
                                             "Provincial Seats",
                                             "Other Cities"))
```

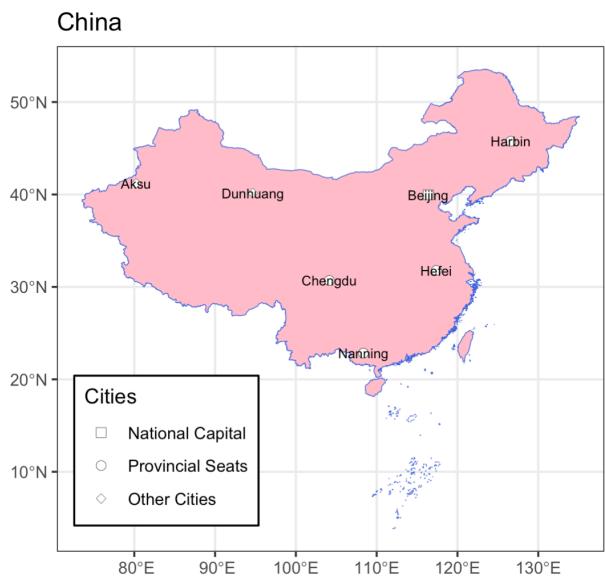
输出结果：



7. 接下来，我们需要告诉读者，这些点位分别是哪些城市。我们可以添加另一个图层，即城市名称的标注图层。

```
#给城市添加标注
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  #点数据分两层绘制，可以起到描边效果
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2) +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2)+ 
  geom_text(data = cities,aes(x=lon,y=lat,label=name),size=2.5) +
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  theme_bw()+
  theme(legend.position = c(0.2,0.2),
        legend.background = element_rect(color="black"),
        axis.title = element_blank())
#绘制地图
p
```

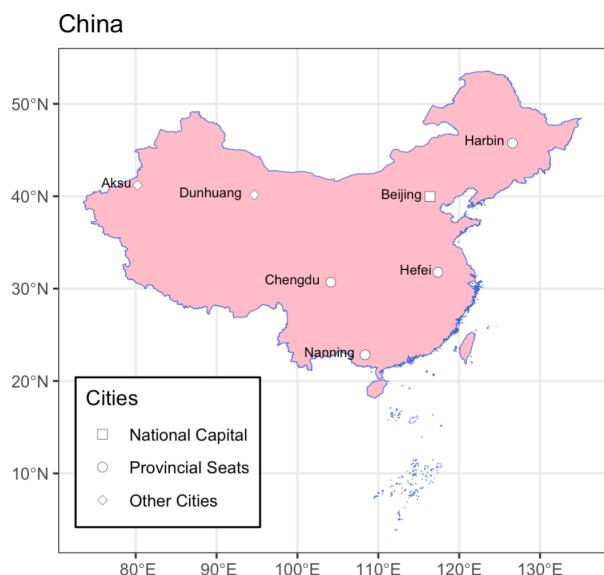
输出结果：



8. 地图上出现了城市的名称。然而，这些文字覆盖在了城市点位之上，影响了地图的可读性，因此我们需要对标注的具体位置进行调整。

```
#调整标注的位置
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  #点数据分两层绘制，可以起到描边效果
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2)+ 
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2)+ 
  geom_text(data =cities,aes(x=lon,y=lat,label=name),angle=0,
            vjust=0.2, hjust=1.2, size=2.5)+  
←
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  theme_bw()+
  theme(legend.position = c(0.2,0.2),
        legend.background = element_rect(color="black"),
        axis.title = element_blank())
#绘制地图
p
```

输出结果：

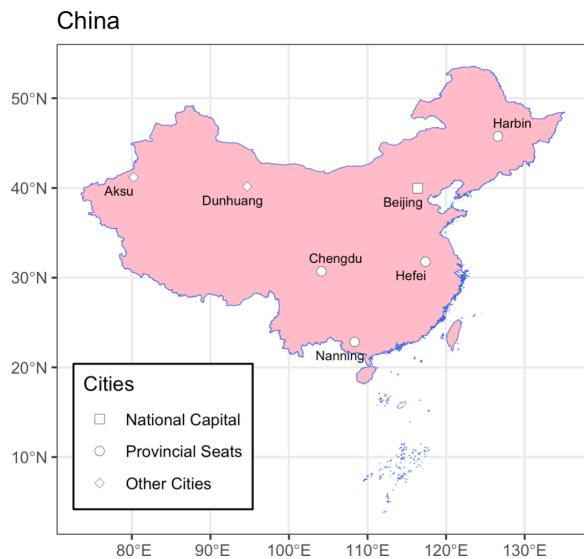


任务：尝试改变 angle、vjust、hjust 的值，观察并体会它们的具体作用。

除此以外，我们还可以安装并引入 `ggrepel` 库，在数据点较多且分布密集时，能更加清晰地展示文字标注。比如：

```
#利用repel功能自动调整
library(ggrepel)
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2)+ 
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2)+ 
  geom_text_repel(data =cities,aes(x=lon,y=lat,label=name),size=2.5)+  
←
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  theme_bw()+
  theme(legend.position = c(0.2,0.2),
        legend.background = element_rect(color="black"),
        axis.title = element_blank())
#绘制地图
p
```

输出结果：

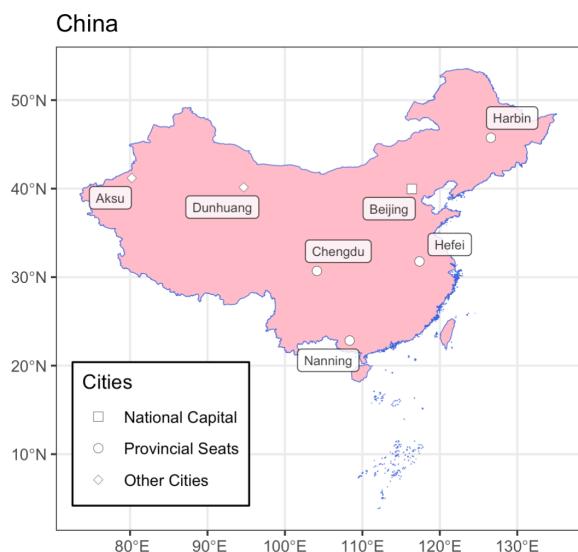


这个功能会让同一图层的标注自动找到最合适的位置，避免互相重叠或和数据点重叠。当数据点过于密集时，该功能会自动隐藏一部分标注。

如果地图图层比较复杂，直接将文字标注放于地图之上可能不利于阅读（比如上图中的“南宁”和海岸线重叠，导致清晰度下降）。此时可以考虑利用 `ggrepel` 安装包的功能，添加半透明的标注框。

```
#作为Label形式添加城市名
library(ggrepel)
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2) +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2) +
  geom_label_repel(data = cities,aes(x=lon,y=lat,label=name),size=2.5,alpha=0.8) +
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  )) +
  labs(shape="Cities") +
  theme_bw() +
  theme(legend.position = c(0.2,0.2),
        legend.background = element_rect(color="black"),
        axis.title = element_blank())
#绘制地图
p
```

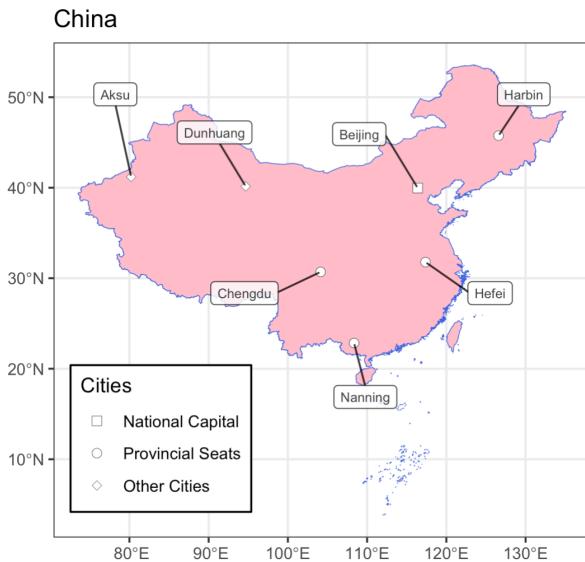
输出结果：



当数据密集时，我们也可以通过给标注框添加指针的方式缓解过于密集的标注情形。

```
#给label添加指针(适用于数据量大的情况)
library(ggrepel)
p <- ggplot() +
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2) +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2) +
  geom_label_repel(data = cities,aes(x=lon,y=lat,label=name),
    box.padding=1.5,size=2.5,alpha=0.8)+  
#箭头指向这里
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  theme_bw()+
  theme(legend.position = c(0.2,0.2),
    legend.background = element_rect(color="black"),
    axis.title = element_blank())
#绘制地图
p
```

输出结果：

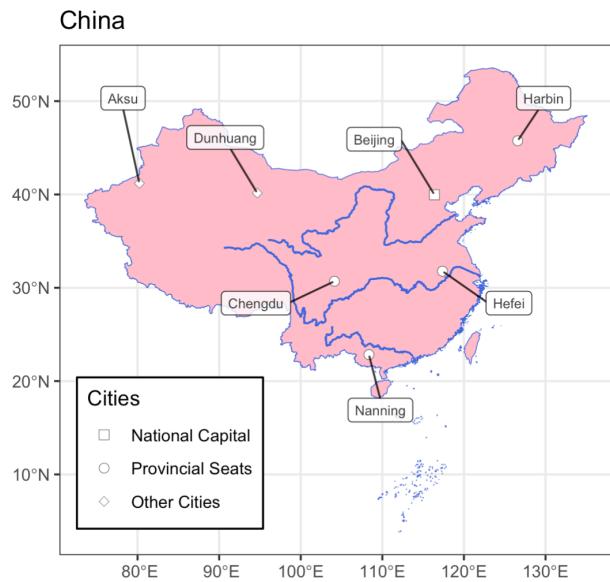


任务：改变 `box.padding` 的值，观察效果。

9. 除了叠加 csv 点数据图层以外，`shapefile` 图层还可以和其他 `shapefile` 图层叠加，比如多边形的中国轮廓图层和线形的河流图层叠加。

```
libraries <- sf::st_read("data/china/rivers.shp")
p <- ggplot()+
  geom_sf(data = china, color = "royalblue", fill = "pink") +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="black",size=2.2) +
  geom_point(data=cities,aes(x=lon,y=lat,shape=type),color="white",size=2) +
  geom_sf(data = rivers, color = "royalblue") +
  geom_label_repel(data = cities,aes(x=lon,y=lat,label=name),box.padding=1.5,size=2.5,alpha=0.8)+  
#箭头指向这里
  ggtitle("China") +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  theme_bw()+
  theme(legend.position = c(0.2,0.2),
    legend.background = element_rect(color="black"),
    axis.title = element_blank())
#绘制地图
p
```

输出结果：



例 2：改变地图投影

地图投影基础

在地图学中，地图投影（map projection）是一种将三维的地球表面展开到二维平面的方法。将球体投影到平面上，球面各区域的大小和形状（即方向）会有一定程度的变形。也就是说，平面地图中，地理要素形状的准确性与大小比例的准确性不可兼得，至少其中之一会发生扭曲变形。根据地图的类型，有些变形是可以接受的，有些是不可接受的。

例如，用于航海的海事地图需要保持方向的准确性，但可以在一定程度上牺牲地理要素大小的准确性；而用于展现各国人口密度的地图最好保持面积不变，但可以牺牲各国轮廓形状的准确性。还有些地图希望在大小和形状的变形之间取得折中方案。为了保留球面的某个性质，而牺牲其他性质的准确性，历史上的地图学家发明了不同的地图投影方法。

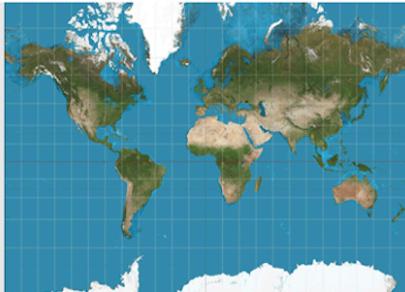
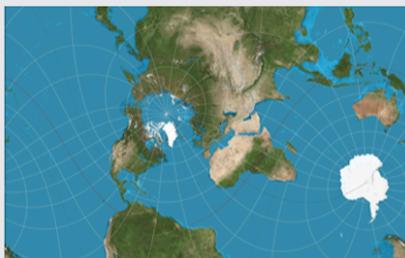
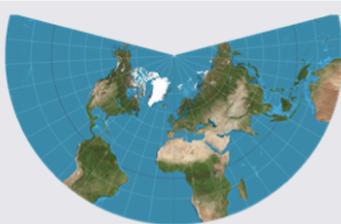
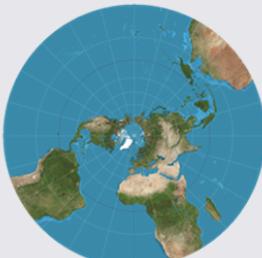
R 中绘制地图的默认投影为简单圆柱投影（等距投影的一种，见下文）。

常见的地图投影法

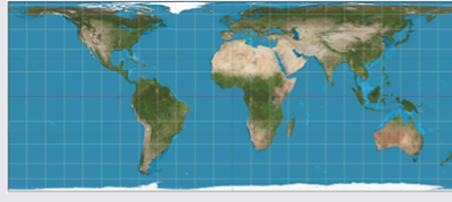
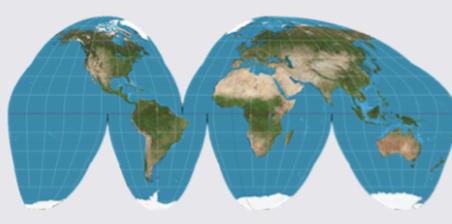
常见的投影法可以分为四大类：

1. 等角投影：牺牲面积大小，保证形状不扭曲。
2. 等积投影：允许轮廓形状发生扭曲，保证面积大小的准确性。
3. 等距投影：面积和形状都发生一定的扭曲，但能让沿特定方向的距离（长度）不变形。
4. 折中投影：根据需求，在面积和形状的扭曲程度之间找到一个合适的平衡点。

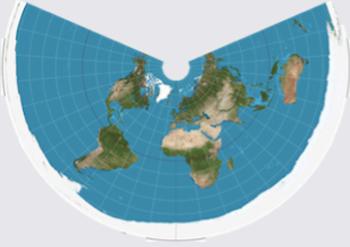
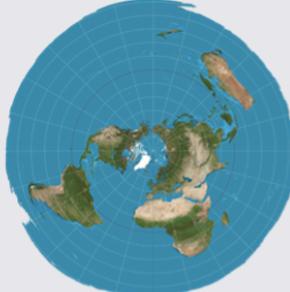
等角投影

示例	投影	特点介绍
	麦卡托(墨卡托)投影	地图学家麦卡托于1569年发明的投影法。经线和纬线都是直线，因此保持了地球上的方向，有助于导航，也保持了陆地的形状。但高纬度的地图面积会极度膨胀，格陵兰岛比非洲还大，甚至导致这种地图无法显示两极。这种地图在20世纪很火，因为冷战的美苏两国都爱用。苏联显示自己的强大，美国用来显示苏联的压迫感从而获得更多军费。
	麦卡托投影的变体 (图为霍汀投影)	横轴麦卡托：用于绘制南北狭长的区域（智利的福音）。美军发明的UTM制图坐标系统就基于它。 斜轴麦卡托（霍汀投影）：多用于绘制人造卫星斜向经过的狭窄区域。 网页麦卡托：针对现代网页优化后的麦卡托，常用于电子地图。
	兰伯特等角圆锥投影	兰伯特等角圆锥投影常被用于航空和航海图。它是地图投影研究界的大神乔纳·海因里奇·兰伯特于1772年发明的七种投影之一。美国部分州和中国部分省的地图多用这种投影。
	球极平面投影	这种投影诞生于古希腊时期，据说是喜帕恰斯（伊巴谷）发明的。它将所有的纬线映射为圆圈，保持中心部分角度和形状不变。通常被用来绘制北冰洋和南极洲地图，并且用于对其他行星的测绘，因为它可以保留陨石坑的形状。
	李氏四面体投影	这种投影利用迪克森椭圆函数把地球表面投到四面体上，然后展开成一个等边三角形地图。除了四面体的四个顶点以外，它在别的地方都保持了真实的形状，但面积会有很复杂的变形规律。由于四面体的性质，这种投影可以在平面上无限拼凑。
	皮尔斯五点投影	它的原理是把地球表面分别投影到8个等腰直角三角形上，再把它们拼成一个正方形。如果中心是北极点，那四个角就是展开后的南极点。这里的所谓“五点”说的是北极+四个角的南极组成了骰子里面5点的那个图案，而赤道在图中成了一个正方形。除了赤道那个正方形的四个角以外，别的地方形状被保持。这种投影的好处是，只要调整，肯定能让那个赤道上那四个有形变的地方落在海里。

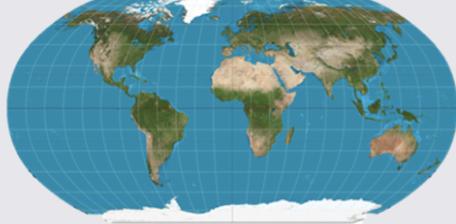
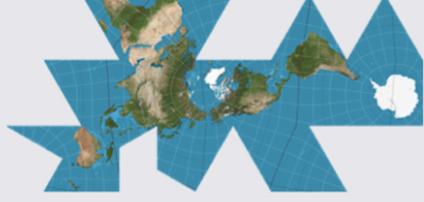
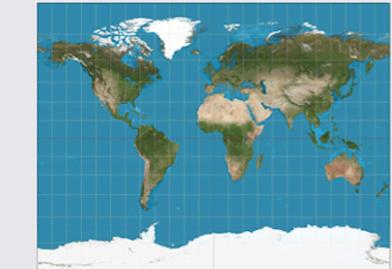
等积投影

示例	投影	特点介绍
	莫尔韦德投影	1805年由德国数学家卡尔·莫尔韦德发明的投影。赤道和中央经线是互相直线的直线，其他经线和赤道的交点间距相等。东西90度的经线形成一个正圆，而整个地球是一个长短轴2:1的椭圆。它常用于地球科学学术论文中的世界地图。
	正弦投影	正弦投影将两极表现为点，然后通过扭曲经线为正弦曲线，来达到保证面积准确的效果。赤道和中央经线是地图上最精确的部分，完全没有扭曲，但离它们越远的地方，形状扭曲就越大。
	贝赫曼投影	一种等面积圆柱投影特例，它的两条标准纬线是南北纬30°，两条标准纬线之间的地球表面被纵向拉伸，其余部分被纵向压扁，从而保留相对面积不变。这种投影很好认，俄罗斯加拿大被严重压扁。
	古迪投影	1923年由地图大神约翰·保罗·古迪（让制图学成为大学专业的关键人物之一）发明的等面积投影，是莫尔韦德投影和正弦投影的杂交体，它通过拆分海洋地区，让陆地在保证等面积的情况下，形变尽量小。很多世界地图册的大洲图使用这种投影，它整体看上去像瓣开的桔子皮。
	阿尔伯斯投影	一种等面积圆锥投影，有两条标准纬线，它们之间的形变比较小，做到了在保证面积准确的前提下，让地球上相对较大地区的形状不发生过分的失真，因此被美国、加拿大、阿根廷、印度和巴西等国土很大国家的测绘系统采用。
	平等地球投影	因西方政治正确而兴起于2018年左右，放大低纬度发展中国家、缩小高纬度发达国家的投影。它总体改造自妥协了面积和形状的罗宾逊投影，但强行保留了面积准确而扭曲了更多的形状。

等距投影

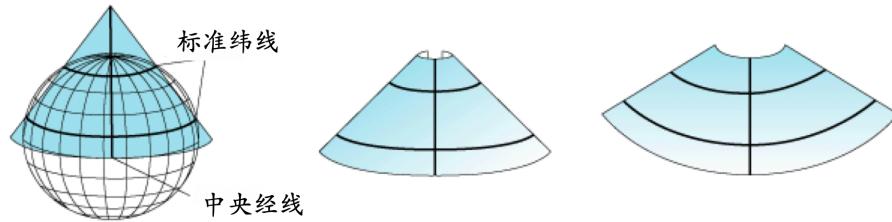
特点和场景	投影	特点介绍
	简易圆柱投影	这是一种以赤道为标准纬线的等距圆柱投影，从数学角度看，它是最简单的投影，就是把经纬度当做x和y展开到平面直角坐标系上，发明人是古希腊米利都的泰勒斯。地图上任意一点到赤道的垂直距离被保留，是许多GIS软件的默认投影，尤其适合低纬度地区。
	等距圆锥投影	这是公元150年左右由罗马帝国埃及行省的地理学家托勒密（也是绘制了第一幅真正意义上的世界地图、发明了“地理学”这个词的希腊文原始版本、提出了地心说的那个人）发明的投影法。所有经线沿线以及参考纬线沿线的距离被保留，常用于中纬度地区。
	两点等距投影	在地面上定义两个参考点。在这种投影中，地图上任何一点到这两个参考点的距离被真实保留。
	等距方位角投影	任意一点到地图中心所在点的距离被保持。最出名的一个等距方位角投影地图应该就是联合国旗帜上的那个地图了。不过联合国旗帜的地图在这基础上做了一些微调，让没有任何国家或大洲过于显眼。
	波恩投影	这种投影既等距又等面积，它保持了中央经线和标准纬线沿线区域的准确距离，很适合绘制呈“T”形的区域，因此被广泛用于绘制倒三角形的欧亚大陆。
	维尔纳投影	俗称桃心投影，是波恩投影的一种特例。如果波恩投影的那条标准纬线被定义为北纬或南纬90°（也就是极点），那它就把地球变成了桃心。这种投影很少用于实战，多数情况下只出现在广告和平面艺术中。

折中投影（任意投影）

示例	投影	特点介绍
	罗宾逊投影	由美国地图学家亚瑟·罗宾逊发明于1963年，后被国家地理协会采用。它是史上第一种非常成功的折中投影，在保留面积还是保留形状之间，找到了一个合适的平衡点。
	尼科洛西投影	尼科洛西投影是西亚伽色尼帝国的科学家比鲁尼（大地测量学之父）在11世纪发明的一种多锥体地图投影，它一次只能显示一个半球，17世纪被西西里王国的地理学家尼科洛西带到西方并深受喜爱。
	富勒地图投影	这是地球在二十面体表面的投影，它可以被展开并压扁为二维。为了尽量保留陆地的形状和尺寸，地图中的海洋部分被大量打断。
	张伯伦三角投影	在地球仪上固定三个点，然后通过三角测量法，把球体上的其他点映射到平面上。国家地理杂志的单一大洲地图喜欢使用这种投影。
	高尔立体投影	这是苏格兰神父詹姆斯·高尔对麦卡托投影的改良，标准纬线为南北纬45°。改良后，陆地轮廓有一定形变，但面积扭曲的幅度降低了许多。
	泰晤士报投影	国家公园之父约翰·缪尔在高尔立体投影的基础上进一步改良，让经线发生弯曲，进一步降低面积的扭曲。后来它被泰晤士报采纳为御用地图投影。

在 R 中改变地图投影

1. 例 1 中国地图的投影是 R 的默认投影——简单圆柱投影。现在，我们将它改为另一种在中国和美国尤为常见的投影——兰伯特等角圆锥投影（Lambert Conformal Conic Projection，简称 LCC）。如下图所示，我们假想把一张扇形的纸卷成圆锥状，让其切割地球，并假设地球的球心位置有一处光源。光源会把地球表面的地理要素（国家、湖泊、河流等）投影到扇形的纸面上，形成二维的地图。

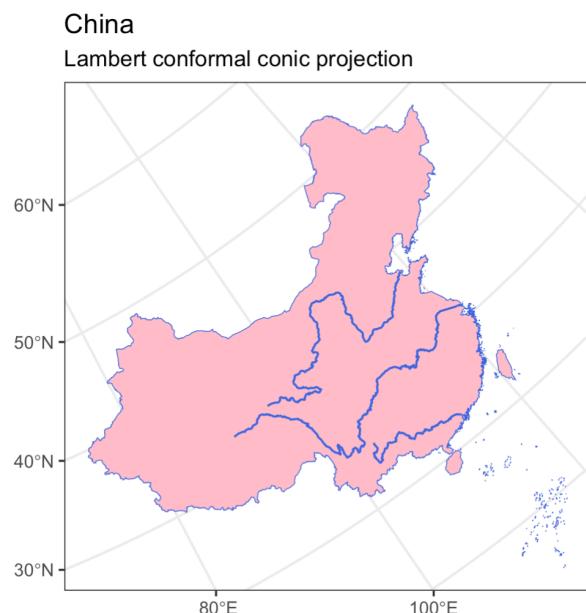


兰伯特等角圆锥投影是一种等角投影，保留地理要素的轮廓形状，但会牺牲面积比例的准确性。这种投影有两条标准纬线，通常位于同一半球。标准纬线的沿途没有面积扭曲，在两条标准纬线之间的区域，面积略有压缩（纬线距离越近，其间的面积扭曲幅度越小），在标准纬线以北和以南的区域，面积会被拉伸放大。因此，在使用兰伯特等角圆锥投影绘制地图时，我们需要根据地图所绘制区域的具体情况，合理地选择标准纬线。此外，兰伯特等角圆锥投影和许多其他投影一样，有一条中央经线，即当用这种投影绘制世界地图时，地图的正中间所对应的经线。

在 R 语言中，默认的中央经线为格林尼治线，如果需要使用其他经线作为中央经线，则需要手动定义。如果该投影有参考经线（比如兰伯特等角圆锥投影），也必须手动定义参考经线，否则系统很可能会报错。

```
#将中国地图改为兰伯特等角圆锥投影 (LCC)
#LCC投影的两条参考纬线设为北纬10°和北纬30°，中央经线设为默认的格林威治线 (0°)
lambert <- "+proj=lcc lat_1=10 lat_2=30 +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
china_lambert <- st_transform(china, lambert)
rivers_lambert <- st_transform(rivers, lambert)
#创建地图
p <- ggplot() +
  geom_sf(data = china_lambert, color = "royalblue", fill = "pink") +
  geom_sf(data = rivers_lambert, color = "royalblue") +
  ggtitle("China","Lambert conformal conic projection") +
  theme_bw()
#绘制地图
p
```

输出结果：



显然，中央经线设在格林威治不符合中国地图的需求。此外，由于标准纬线的度数设置得比较低，导致中国北方的面积增大比较明显，显得头重脚轻（比如，可以对比观察北纬 30° 到 40° 以及北纬 50° 到 60° 纬线之间的距离）。因此，我们需要根据中国的实际情况来重新设置标准纬线和中央经线。

```
#LCC 投影的两条参考纬线设为北纬20°和北纬50°，中央经纬线为东经100°
lambert <- "+proj=lcc lat_1=20 lat_2=50 +lon_0=100 +ellps=WGS84 +datum=WGS84 +units=m"
china_lambert <- st_transform(china, lambert)
rivers_lambert <- st_transform(rivers, lambert)
#创建地图
p <- ggplot() +
  geom_sf(data = china_lambert, color = "royalblue", fill = "pink") +
  geom_sf(data = rivers_lambert, color = "royalblue") +
  ggtile("China", "Lambert conformal conic projection") +
  theme_bw()
#绘制地图
p
```

输出结果：



2. 我们将例 1 中的城市数据也添加回兰伯特等角圆锥投影的地图里。其他图层转换投影后，城市数据不再能直接作为散点数据添加，因此我们首先需要将城市数据转换为 Shapefile，并改变其投影和参考系。

```
#将城市数据转换为Shapefile，并赋予其基本的WGS 84参考系（代码为4326）
cities_sf<- st_as_sf(cities, coords = c("lon", "lat"), crs=4326)
#将城市数据进行LCC投影变换
cities_lambert <- st_transform(cities_sf, lambert)
#创建地图，将城市数据添加到地图
p <- ggplot() +
  geom_sf(data = china_lambert, color="royalblue", fill="pink") +
  geom_sf(data = rivers_lambert, color="royalblue") +
  geom_sf(data = cities_lambert, aes(shape=type), color="black", size=2.2) +
  geom_sf(data = cities_lambert, aes(shape=type), color ="white", size=2) +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  ggtile("China", "Lambert conformal conic projection") +
  theme_bw()+
  theme(legend.position = c(0.25,0.18),
        legend.background = element_rect(color="black"),
        axis.title = element_blank())
#绘制地图
p
```

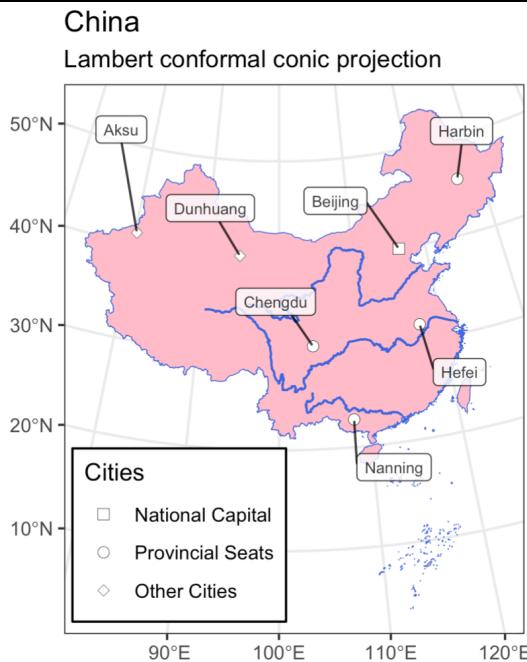
输出结果：



3. 此时如果要添加城市名，需在 `cities_lambert` 中添加新参考系下的经纬度坐标。通过 `sf` 添加到 R 环境中的 Shapefile 除了 `geometry` 形状数据以外，其他的属性可以按照常规的数据框（`data frame`）进行处理。如果希望获取 `geometry` 形状中的坐标信息，可以使用 `st_coordinates()` 函数来提取。

```
#获得经纬度坐标，为添加城市名做准备
cities_lambert$lon = 0
cities_lambert$lat = 0
#使用st_coordinates()函数提取geometry中的坐标
for(i in 1:nrow(cities_lambert)){
  cities_lambert$lon[i] = st_coordinates(cities_lambert$geometry[i])[1]
  cities_lambert$lat[i] = st_coordinates(cities_lambert$geometry[i])[2]
}
#创建地图并添加label和指针
p <- ggplot() +
  geom_sf(data = china_lambert, color="royalblue", fill="pink") +
  geom_sf(data = rivers_lambert, color="royalblue") +
  geom_sf(data = cities_lambert, aes(shape=type), color="black", size=2.2) +
  geom_sf(data = cities_lambert, aes(shape=type), color ="white", size=2) +
  geom_label_repel(data =cities_lambert,aes(x=lon,y=lat,label=name),
    box.padding=1,size=2.5,alpha=0.8) +
  scale_shape_manual(values=c(
    "National Capital" = 15,
    "Provincial Seats" = 16,
    "Other Cities" = 18
  ))+
  labs(shape="Cities")+
  ggtitle("China","Lambert conformal conic projection") +
  theme_bw()+
  theme(legend.position = c(0.25,0.18),
    legend.background = element_rect(color="black"),
    axis.title = element_blank())
#绘制地图
p
```

输出结果：



任务：将地图改为下面这些常见的投影类型，并尝试使用不同的中央经线（以及标准纬线，如果存在的话），观察它们的效果。

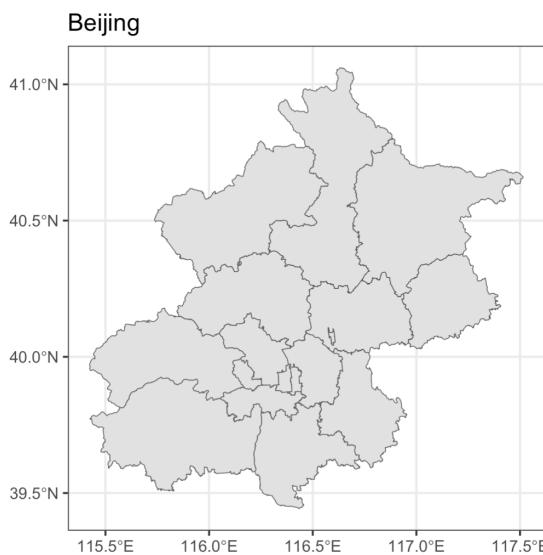
```
#罗宾逊投影
robinson <- "+proj=robin +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
#正弦投影
sinusoidal <- "+proj=sinu +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
#麦卡托（墨卡托）投影
mercator <- "+proj=merc +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
#米勒投影
miller <- "+proj=mill +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
#莫尔韦德投影
mollweide <- "+proj=moll +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
#阿尔伯斯投影
albers <- "+proj=aea +lat_1=30 +lat_2=50 +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
#古迪投影
goode <- "+proj=goode +lon_0=0 +ellps=WGS84 +datum=WGS84 +units=m"
```

例 3：专题地图

1. 使用 `st_read()` 导入北京区县数据 `beijing_districts.shp`，并绘制基本地图。

```
#导入北京行政区划shapefile
beijing <- sf::st_read("data/beijing/beijing_districts.shp")
#利用ggplot绘制北京行政区划的地图
p <- ggplot() +
  geom_sf(data = beijing) +
  ggtitle("Beijing") +
  theme_bw()
#绘制地图
p
```

输出结果：



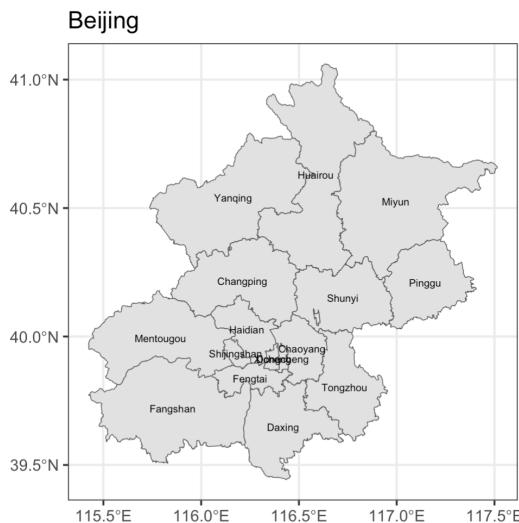
该数据集有三列，包括各区名称以及它们的几何图形：

	id	district	geometry
1	110101	Dongcheng	MULTIPOLYGON (((116.3877 39...
2	110102	Xicheng	MULTIPOLYGON (((116.393 39....
3	110105	Chaoyang	MULTIPOLYGON (((116.4439 39...
4	110106	Fengtai	MULTIPOLYGON (((116.4252 39...
5	110107	Shijingshan	MULTIPOLYGON (((116.1516 39...
6	110109	Haidian	MULTIPOLYGON (((116.0180 40...

2. 给每个区县添加标注

```
#给每个区添加标注
p <- ggplot() +
  geom_sf(data = beijing) +
  ggttitle("Beijing") +
  theme_bw()+
  geom_sf_text(data = beijing, aes(label = district), size = 2 ,
               fun.geometry = sf::st_centroid)
p
```

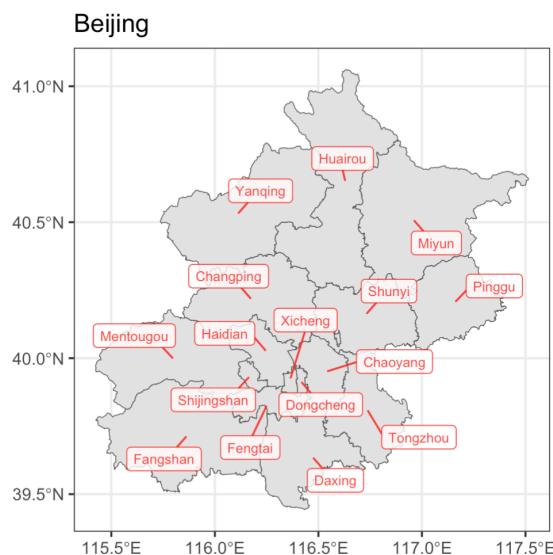
输出结果：



容易发现，区县密集区域的标注发生了重叠，不利于阅读，因此我们需要用到 `ggrepel` 的标注框和指针：

```
library(ggrepel)
p <- ggplot() +
  geom_sf(data = beijing) +
  ggttitle("Beijing") +
  theme_bw() +
  geom_label_repel(
    data = beijing,
    aes(label = district, geometry = geometry),
    size = 2.5,
    box.padding = 0.35,
    stat = "sf_coordinates",
    min.segment.length = 0,
    colour = "red", #标注的颜色
    segment.colour = "red", #指针的颜色
    alpha = 0.8
  )
p
```

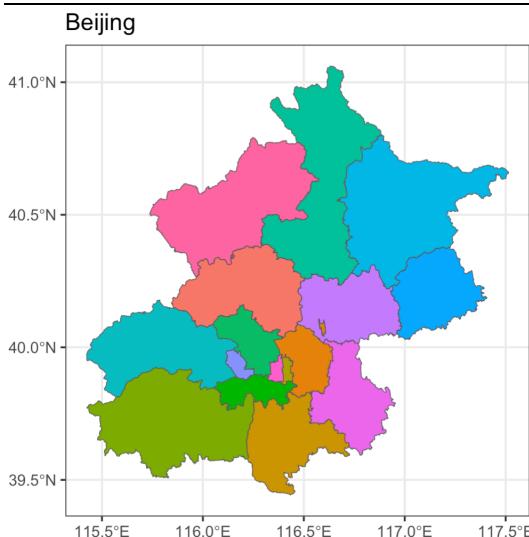
输出结果：



3. 用不同的颜色显示不同的区县。

```
#按照行政区着色
p <- ggplot() +
  geom_sf(data = beijing, aes(fill=district)) +
  labs(title = "Beijing") +
  theme_bw() +
  theme(
    legend.position = "none"
  )
#绘制地图
p
```

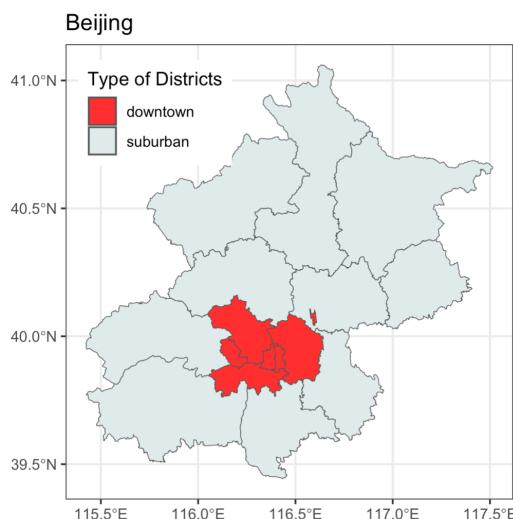
输出结果：



4. 前面提到，通过 `st_read()` 导入 R 环境的 Shapefile 可以当做数据框来进行处理。比如，我们要突出显示北京的六个主城区：

```
#新增一列，将各区类型设置为默认值suburban (郊区)
beijing$type = "suburban"
#定义北京主城区的范围，并改变相应的类型
downtown <- c("Dongcheng", "Xicheng", "Haidian", "Chaoyang", "Fengtai", "Shijingshan")
beijing$type[beijing$district %in% downtown] <- "downtown"
#按照各区类型着色
p <- ggplot() +
  geom_sf(data = beijing, aes(fill=type)) +
  labs(title = "Beijing") +
  theme_bw() +
  scale_fill_manual(values=c(
    "downtown" = "firebrick1",
    "suburban" = "azure2"
  )) +
  theme(
    legend.position = c(0.2, 0.85)
  ) +
  labs(fill="Type of Districts")
#绘制地图
p
```

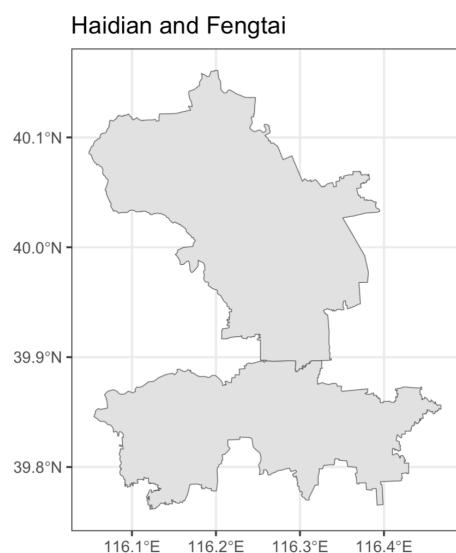
输出结果：



或者在导入数据后选择某个子集进行绘图。例如，我们只需要绘制海淀区和丰台区的地图：

```
#获得只包含海淀区和丰台区的子集，并绘制地图：
hf <- beijing[which(beijing$district %in% c("Haidian", "Fengtai")),]
p <- ggplot() +
  geom_sf(data = hf) +
  ggtitle("Haidian and Fengtai") +
  theme_bw()
#绘制地图
p
```

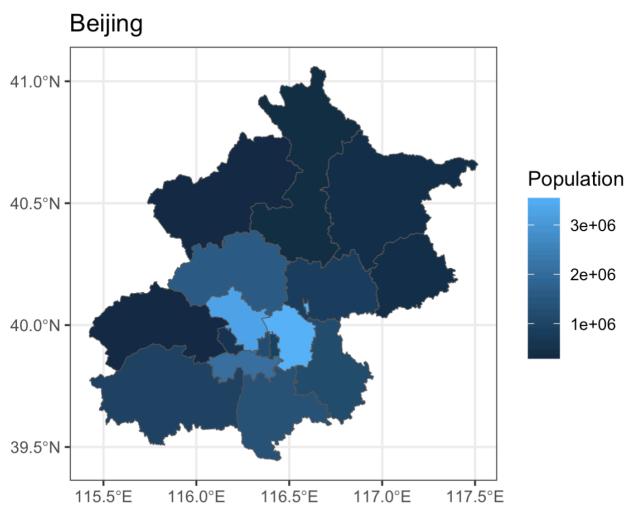
输出结果：



5. R 中的 Shapefile 还可以和其他数据合并。比如，添加各区的人口数据 (beijing_population.csv) 并绘图：

```
#导入北京各区人口数据
pop <- read_csv("data/beijing/beijing_population.csv")
#将人口数据与北京行政区划地图数据合并
beijing = merge(x=beijing, y=pop, by="district")
#利用ggplot绘制北京各区人口地图
p <- ggplot() +
  geom_sf(data = beijing, aes(fill=population)) +
  labs(title = "Beijing") +
  theme_bw()+
  theme(
    legend.position = "right"
  )+
  labs(fill = "Population")
#绘制地图
p
```

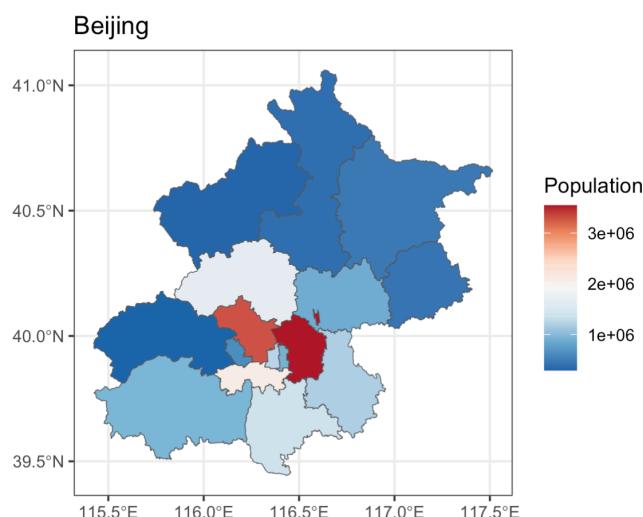
输出结果：



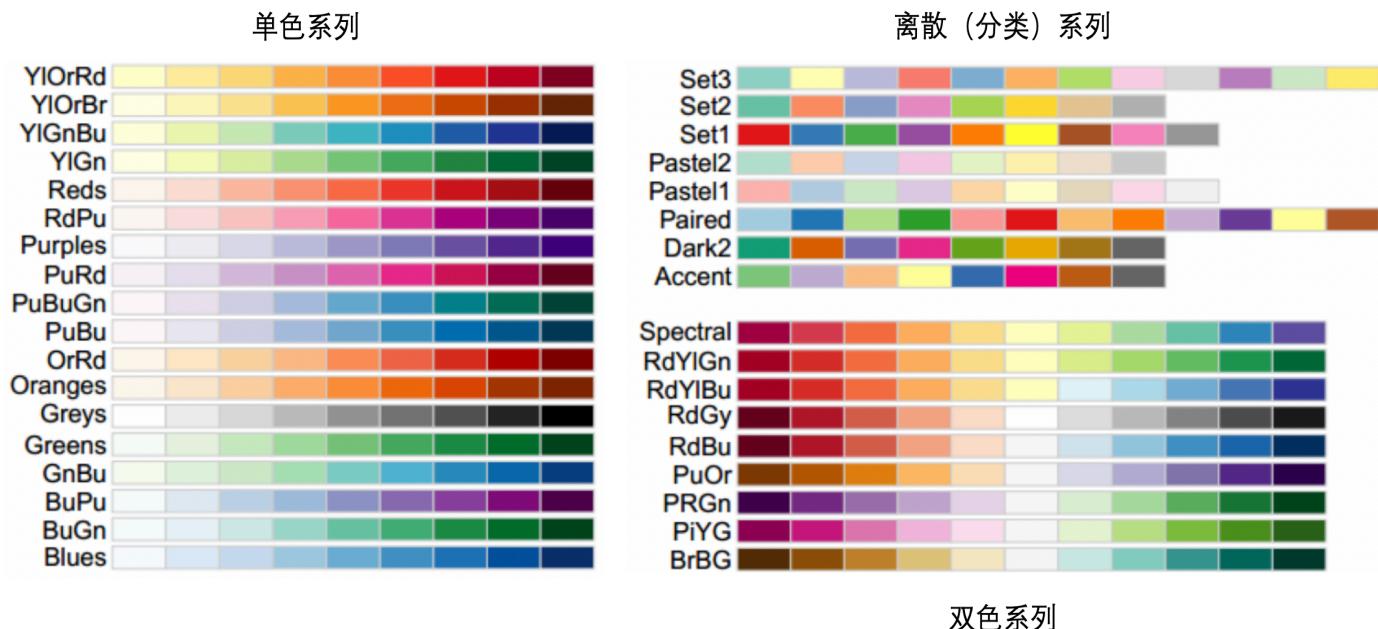
数据的分布情况导致默认的单色系列配色方案中暗色区域太多，不利于地图的阅读。我们可以改变配色方案，让地图更清晰。比如，使用 ColorBrewer 中的双色系列色带（已预置在 R 里的 `palette`）。

```
#改变配色方案
p <- ggplot() +
  geom_sf(data = beijing, aes(fill=population)) +
  labs(title = "Beijing") +
  theme_bw() +
  theme(
    legend.position = "right"
  ) +
  labs(fill = "Population") +
  scale_fill_distiller(palette = "RdBu")
#绘制地图
p
```

输出结果：



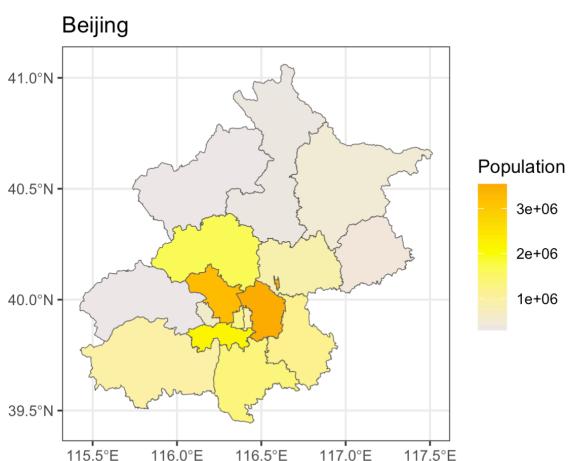
以下为 R 中预置的 ColorBrewer 色带名：



除此之外，还可以自己定义色带：

```
#自定义色带
p <- ggplot() +
  geom_sf(data = beijing, aes(fill=population)) +
  labs(title = "Beijing") +
  theme_bw()+
  theme(
    legend.position = "right"
  )+
  labs(fill = "Population")+
  scale_fill_gradient2(midpoint = 2000000,
                       low="snow2",
                       mid="yellow",
                       high="orange",
                       na.value="gray50")
#绘制地图
p
```

输出结果：



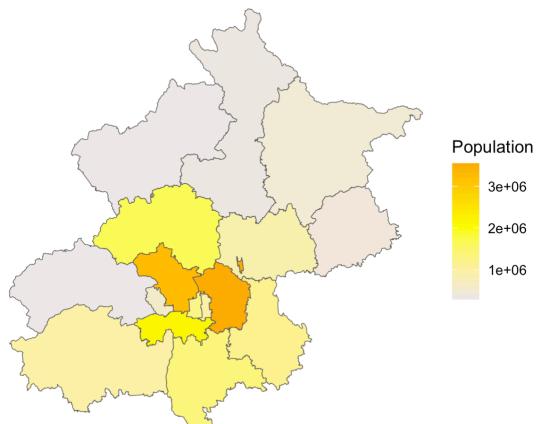
6. 通常来说，专题地图要尽量简洁，突出主题。我们可以去掉多余的要素，比如经纬度等。

```

p <- ggplot() +
  geom_sf(data = beijing, aes(fill=population)) +
  theme_void() +
  theme(
    legend.position = "right"
  ) +
  labs(fill = "Population") +
  scale_fill_gradient2(midpoint = 2000000,
                       low="snow2",
                       mid="yellow",
                       high="orange",
                       na.value="gray50")
#绘制地图
p

```

输出结果:



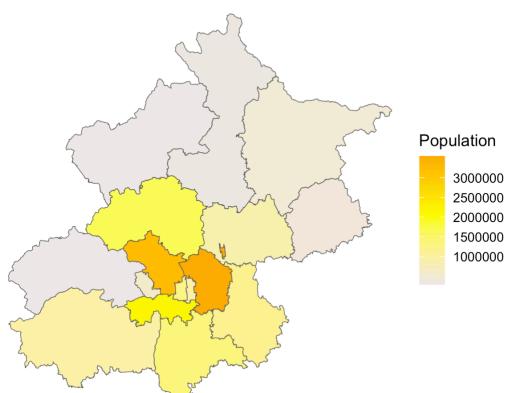
自定义图例中的色带标注分割点:

```

p <- ggplot() +
  geom_sf(data = beijing, aes(fill=population)) +
  theme_void() +
  theme(
    legend.position = "right"
  ) +
  labs(fill = "Population") +
  scale_fill_gradient2(midpoint = 2000000,
                       low="snow2",
                       mid="yellow",
                       high="orange",
                       na.value="gray50",
                       labels = c("1000000", "1500000", "2000000", "2500000", "3000000"),
                       breaks = c(1000000, 1500000, 2000000, 2500000, 3000000))
#绘制地图
p

```

输出结果:



作业部分

数据描述

europe.zip

压缩包中有 `europe.shp` 及相关文件，是欧洲各国矢量轮廓数据（包括传统上属于亚洲的土耳其、塞浦路斯以及高加索地区各国，俄罗斯只有欧洲部分），包含了欧洲各国的人口数据（2005 年）。其中，`country_id` 属性为各国的标准双字母缩写，在本次作业的所有相关数据文件里作为唯一识别符使用。

germany_railways.zip

压缩包中有 `germany_railways.shp` 及相关文件，为德国境内的铁路矢量地理数据。其中 `type` 属性为铁路的分类，该列中的“rail”类为仍在运行的客运和货运铁路，而“abandoned”类为废弃铁路。

eu.csv

欧盟（European Union）相关数据，包括以下属性：

- `country_id`: 各国的标准双字母缩写。
- `eu_member`: 是否为欧盟成员，包括 `Current`（现成员）、`Former`（已退出成员）和 `No`（其他国家）三类。
- `eurozone_member`: 是否为欧元区，包括 `Yes`（欧元区正式成员）、`Future`（已确认将加入欧元区的未来成员）、`Adopted`（使用欧元的非成员国）、`ERM II`（采用欧洲汇率机制的非成员国）和 `No`（其他国家）五类。
- `schengen_member`: 是否为《申根协议》签约国（互相之间取消边境检查，虚化国界线），包括 `Current`（现有签约国）、`Future`（即将签约）、`No`（其他国家）三类。
- `union_date`: 正式加入欧盟的年份。
- `eurozone_date`: 正式加入欧元区的年份。
- `schengen_date`: 正式签约《申根协议》的年份。
- `exit_date`: 退出欧盟的年份。

europe_gdp.csv

欧洲各国的 2022 年国内生产总值（GDP）数据，包括以下属性：

- `country_id`: 各国的标准双字母缩写。
- `gdp`: 2022 年该国的 GDP（单位：10 亿美元）。

germany_cities.csv

德国城市数据集，包括以下属性：

- `name`: 城市名称
- `type`: 城市种类，包括 `capital`（首都）、`major city`（大城市）、`small city`（小城市）三类。
- `lat/lon`: 城市的 WGS 84 地理坐标。

nordic_capitals.csv

北欧国家的首都，包括以下属性：

- `name`: 城市名称
- `lat/lon`: 城市的 WGS 84 地理坐标。

第一部分：按要求绘图

请严格按照描述的要求绘图（要求以外的部分可适当发挥，但不能影响所要求内容的呈现）。将地图填进答题区的相应位置（可直接粘贴，也可作为附件），不需要配文字。

1. 荷兰(Netherlands)、比利时(Belgium)和卢森堡(Luxembourg)常被合称为低地国家(Low Countries)。绘制一幅低地国家的地图，使其满足如下要求：**a**)该图只包含这三个国家；**b**)轮廓线用蓝色(blue)；**c**)在国土上标注各自的 2022 年 GDP 数值(10 亿美元)；**d**)找出三国中 2022 年 GDP 最高的一个，填充为粉红色(pink)，剩余两个国家填充为 80% 的灰色(gray80)；**e**)不显示经纬线坐标和地图边框，放置图例到底部。[每个要求 2 分]
2. 北欧国家(Nordic Countries)包括瑞典(Sweden)、芬兰(Finland)、挪威(Norway)、丹麦(Denmark)和冰岛(Iceland)，以及丹麦的自治领(附属国)法罗群岛(Faroe Islands)。绘制一幅北欧各国的简单示意图，使其符合以下要求：**a**)显示以上各国(包括自治领)的领土范围；**b**)标注以上各国(包括自治领)的**双字母缩写**；**c**)显示以上各国(包括自治领)的首都位置；**d**)标注首都的名称；**e**)该地图采用阿尔伯斯投影(见练习部分地图投影章节的最后一个任务)，并选用合理的标准纬线和中央经线(提示：北欧国家纬度较高，阿尔伯斯投影两条标准纬线之间的形变较小)。[每个要求 2 分]
3. 绘制一幅欧洲各国的 2005 年人口地图。要求：**a**)去除包含于数据中的传统意义上的亚洲国家：土耳其(Turkey)、塞浦路斯(Cyprus)、阿塞拜疆(Azerbaijan)、亚美尼亚(Armenia)和格鲁吉亚(Georgia)；**b**)使用罗宾逊投影，并妥善选择中央经线；**c**)自定义色带，最低值用蓝色(blue)，最高值用红色(red)，中间值用白色(white)；**d**)找到合适的色带中间值，并在图例中妥善选择标注分割点；**e**)用标注框加指针的方式标出人口最多的三个国家。[每个要求 2 分]
4. 绘制一幅德国铁路示意图。该图必须包含的要素有：**a**)德国的国境范围；**b**)德国境内仍在运行的客运和货运铁路；**c**)提供的数据集中包含的德国城市(用不同的形状体现首都、大城市和小城市)；**d**)标注城市名称；**e**)完善相关的图例。[每个要求 2 分]

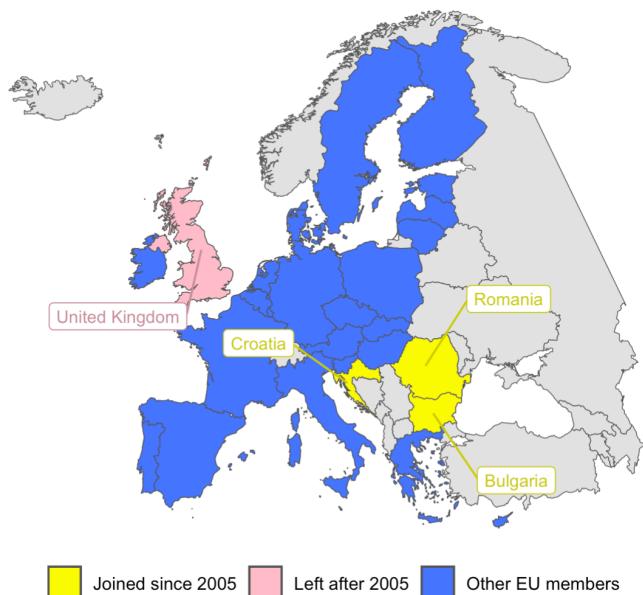
第二部分：围绕问题自由绘图

根据题目中的问题画地图，每道题只画一张地图。总体要求：没有背景知识的读者能通过看你的地图找到每个问题的答案。具体的地图样式设计没有对错之分，但需要简单易懂，标注清晰，从视觉上突出主题。将地图填进答题区的相应位置（可直接粘贴，也可作为附件）。

此外，在答题区用一两句话简述地图的设计思路（你的地图如何帮助读者得到问题的答案，参见样例）。[地图切题程度 2 分，能清晰且完整地传达所需信息 3 分，地图呈现的数据符合提供的数据集(即数据没有在处理和画图的过程中出错) 3 分，简述设计思路 2 分]

样例题目：欧盟成员在 2005 年以后有哪些变化？

地图：



简述：图中用不同的颜色填充了欧盟相关国家，从而向读者展现欧盟的整体变化。其中，黄色填充及相应的标注展现了 2005 年以后新加入欧盟的国家；粉红色填充及相应的标注展现了 2005 年以后离开欧盟的国家；其余 2005 年以后未发生变化的欧盟国家用蓝色填充。

题目如下：

5. 前南斯拉夫 (Yugoslavia) 存在于 1945 年到 1992 年之间，它的范围包括了今天的波黑 (Bosnia and Herzegovina)、克罗地亚 (Croatia)、北马其顿 (North Macedonia)、黑山 (Montenegro)、塞尔维亚 (Serbia) 和斯洛文尼亚 (Slovenia) 六个国家。请问：前南斯拉夫在欧洲的什么位置？
6. 欧元区有哪些现有成员和将要加入的成员？另有哪些使用欧元的非成员国？
7. 从 2008 年起，《申根协议》签约国发生了怎样的变化？有哪些国家分别于哪年成为了新的申根国家？
8. 在欧盟范围内（不包括已退出成员），2022 年 GDP 排前 5 的国家是哪些？
9. 欧元区的未来成员国各自有多少人口（以万人为单位）？
10. 德国境内的废弃铁路（type 为 abandoned）分布情况是怎样的？

提交内容

将 10 幅地图分别填入学习通答题区的相应位置。其中，5–10 题需要配上设计思路的概述。

将所有用到的 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“Lab3+姓名+学号”，比如“Lab3 张三 1010101010.R”。

截止时间：2023 年 12 月 18 日北京时间 23:59。