

## 探案工具

本案例使用了条形图（详见 4.1 小节）和玫瑰图（详见 1.3 和 5.5 小节），所使用的数据收集在 **MSG** 包中，前文所述的《卫报》数据表中的错误经过了订正。加载数据和绘制图形的代码如下：

```
# ggplot2 绘制 2010 年中美国力对比图
library(MSG)
library(ggplot2)
data(cn_vs_us, package = "MSG")
print(
  ggplot(cn_vs_us, aes(x = country, y = value, fill = country)) +
  geom_col() + facet_wrap(~metric, scales = "free_y", ncol = 4)
)
```

## 3.3 新冠疫情

### 3.3.1 案情回放

2020 年春节前后，新冠疫情爆发，亿万人的生活和工作都或多或少受到疫情的影响。笔者在关注疫情统计数据的过程中，遇到了一些有意思的问题。这里试举两例。

第一个例子是，笔者看到朋友分享的一幅疫情地图（如图 3.5），显示了 1 月 22 日全国各省确诊和疑似病例，只见沿海和中部地区一片火红（红色表示新冠肺炎确诊病例），唯有江苏省是白色。图上搭配文字：“在吗，救我”，在地图上格外抢眼。笔者即将赴江苏就职，对此格外关注的同时，不免产生疑问：江苏和周边省份差别真的有这么大吗？为什么？

第二个例子是，有一段时间，一种酷似鹦鹉螺的玫瑰图非常流行，如图 3.6 所示，显示了 2020 年 3 月 27 日除我国以外的世界各国新冠疫情。图中用不同颜色、相同圆心角的扇形来代表确诊 707 例以上的不同国家，各国新冠肺炎的确诊和死亡人数以数字标注。显然，图形经过了精心设计，令人过目不忘。然而，仔细观察却免不心生困惑：与 3.2 小节所举的例子类似，如果以半径代表数值，那么单从图形看，会给人一种印象，即美国确诊数是德国的 2 倍，而德国大约是英国的 2 倍；但一看数字，德国却大约是英国的 4 倍。那么，确诊数量究竟体现在扇形的哪个属性呢？从图上如何看出各国的实际差异呢？



图 3.5: 新冠疫情下的江苏: 2020 年 1 月 22 日, 除了江苏省, 东部和中部地区一片火红

## 探案过程

对于第一个关于疫情地图的问题, 笔者获取了当天(1月22日)以及前一天(1月21日)的疫情数据。这是为了避免数据统计时间的不同而出现与图3.6在统计数字上的差异。在这些数据的基础上, 笔者绘制了这两天的疫情地图, 如图3.7所示。

在这两张地图上, 我们同样是用红色来表示疫情的地理分布; 不同的是, 我们用红色的深浅来区分确诊病例的多少。同时, 由于各省确诊病例数量极差较大, 跨三个数量级, 我们使用了对数坐标, 以便显示低值之间的差异。

从图中可以看出, 江苏跟周边省份差别并不是很大: 1月21日, 除了浙江5例确诊(浅红色)外, 江苏和安徽无确诊, 山东只有1例, 在地图上都是白色或接近白色; 而1月22日, 浙江共10例, 而江苏、安徽、山东各增加1例, 均是极浅的浅红色。

其实, 我们可以看到, 在图3.5中, 很多省份其实只有1例确诊, 却涂上了跟几百例确诊的省份同样的颜色, 夸大了江苏和其它省份的差别。当然, 笔者作图的角度是比较各省确诊病例的数量级; 如果是为了区分“零确诊”与“有确诊”, 那就另当别论了。在这个案例里, 颜色的选择和图例的重要性可见一斑。

为了直观地看到疫情在地理上的变化, 笔者将每天各省的确诊病例



图 3.6: 2020 年 3 月 27 日世界各国新冠疫情的玫瑰图: 图中用不同颜色的扇形来代表不同国家

表 3.3: 全国省级行政区新冠肺炎确诊人数（未发现确诊病例的地区并未列出）

名称	01-21	01-22
湖北	270	444
广东	17	26
北京	10	14
浙江	5	10
上海	6	9
重庆	5	6
河南	1	5
四川	1	5
天津	2	4
湖南	1	4
海南	0	4
辽宁	0	2
江西	0	2
山东	1	2
广西	0	2
河北	0	1
山西	0	1
江苏	0	1
安徽	0	1
福建	0	1
贵州	0	1
云南	1	1
宁夏	0	1
台湾	1	1
澳门	0	1

做成一张地图，然后连起来做成一幅动画，可以在线观看<sup>4</sup>。

对于第二个关于玫瑰图的问题，玫瑰图的本质是条形图，只不过是把条形图的笛卡尔坐标转换成了极坐标：用极坐标的角度代表笛卡尔坐标的  $x$ ，而半径代表  $y$ 。从图 3.6 来看，各国确诊病例对应的肯定不是扇形半径，因为论确诊数量的话，排在第一位的美国是排在末位的菲律宾的一百多倍，而对应的扇形半径凭肉眼判断，肯定相差不到 100 倍。有没有可能是圆心并不代表 0 病例，而是某个其它数值呢？也不是，因为

---

<sup>4</sup>[https://ncov2020.org/animation/map\\_animation.gif](https://ncov2020.org/animation/map_animation.gif)

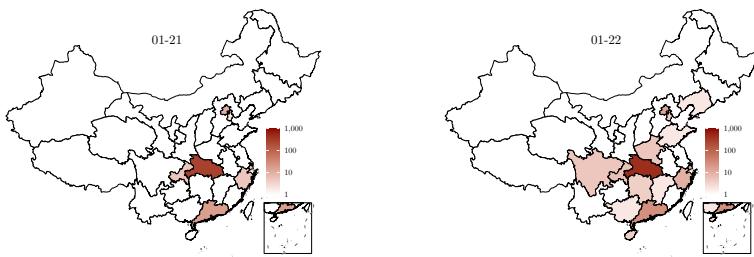


图 3.7: 新冠疫情中国地图。图中用红色的深浅来表示确诊病例的多少。由于各省确诊病例数量极差较大，数量跨三个数量级，使用对数坐标来显示低值之间的差异。

如果那样的话，两者半径相差的倍数只会更多。

会不会是用扇形面积表示确诊数量呢？也不是，因为美国确诊数量是德国的大约两倍，而其扇形面积显然超过这个比例。此外，选用扇形来对比两者大小的时候，要么是令圆心角相同而比较半径的不同（如 1.3 小节的极坐标面积图和 5.5 小节的风玫瑰图，此时面积与半径的平方成正比），要么是令半径相同而比较圆心角的不同（如 4.9 小节的饼图，此时面积与圆心角成正比）。由于人眼对扇形的角度和半径比较敏感，对面积不敏感，一般不会在半径不同的情况下用扇形面积来区分大小。

那么，有没有可能是将确诊数量取了对数之后作图呢？让我们来看一下：以美国、德国和英国为例，美国确诊病例是德国的大约 2 倍，而德国确诊病例是英国的大约 4 倍。假定是以扇形半径来表示确诊病例的对数，那么德英两国的扇形半径差应该是美德两国扇形半径差的 2 倍，但图 3.6 显然不是这样，甚至相反：德英两国的扇形半径差大约只有美德两国的一半。

为了一探究竟，笔者获取了当天的全球疫情数据，自行绘制了一张类似的玫瑰图，如图 3.8 的上图所示，以扇形的半径来表示确诊病例的对数。图中的统计数字略有出入，可能是数据更新的当天时刻有不同，这不影响我们的对比。半透明的环状带标示了病例为 10、100、1000 和 10000 的位置。

从图中可以看出，玫瑰图用对数来展示确诊病例之后，相邻两个扇形的半径差别就没有那么大了。虽然视觉冲击力小了很多，然而我们可以清晰地从图中看出哪些国家的确诊病例是属于同一个数量级（即处于同一环状带），以及任意两国的病例数对比关系。例如韩国在最外围的半

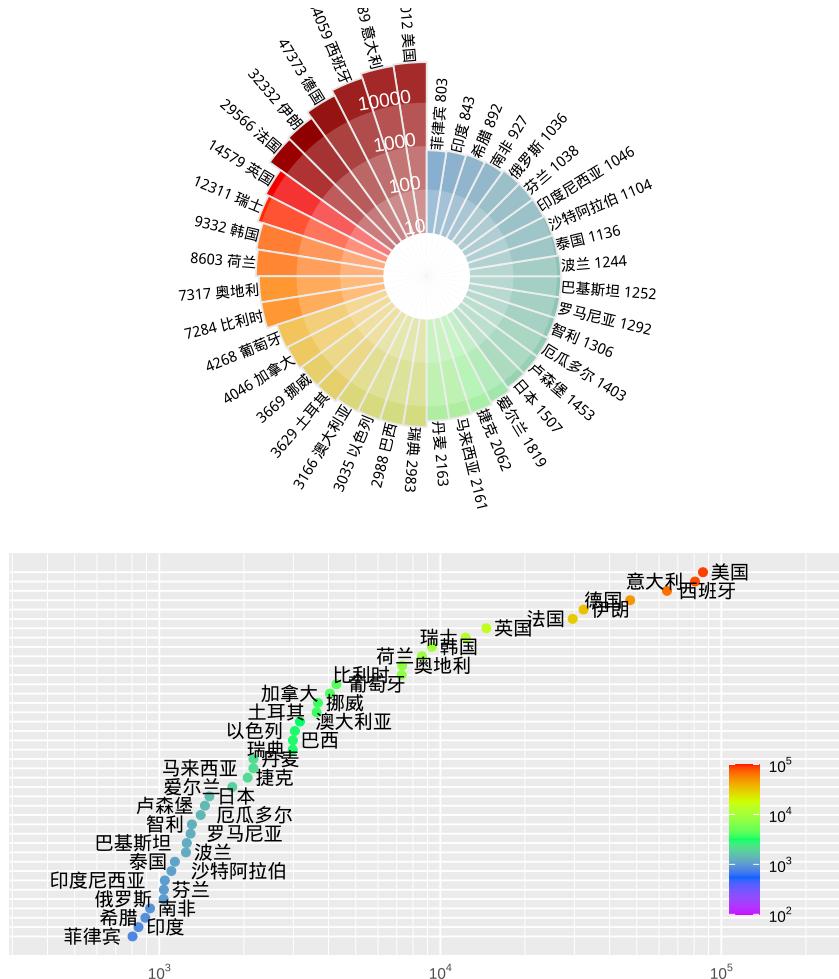


图 3.8: 对数坐标表示的世界各国新冠疫情的玫瑰图 (上) 和点图 (下):  
上图用不同颜色的扇形来代表不同国家

透明带边缘，而俄罗斯位于自外向内第二个半透明带的边缘，即使不看数字，也能知道两者是 10 倍的关系；类似的，美国跟韩国也是大约 10 倍的关系。而图 3.6 的扇形半径跟确诊病例数有何种换算关系，以及中心的白色半透明圆环表示的是什么意义，我们就不得而知了。

作为对比，笔者将同一组数据做了一张点图，见图 3.8 下图。与极坐标相比，在笛卡尔坐标系中，用不均匀的网格来表示对数坐标就会更容易理解。同时，读者可以很直观地从图中获取病例数的大概数值，以便做进一步的比较，这显然比玫瑰图更有意义。例如，前文所述，在对

数坐标下，德英两国确诊病例之差应该是美德两国之差的 2 倍，在点状图上一目了然。注意：这张点图里，点的颜色其实是多余的元素；因为点的  $x$  坐标已经表达了其确诊病例数的数值。笔者在这里仍然标识了颜色，仅仅是为了视觉上的美观而已。

关于笔者参与的有关新冠疫情的进一步研究工作，参见 Zou et al. [2020]。

## 探案工具

本案例使用了玫瑰图（详见 1.3 和 5.5 小节）、点图（详见 4.2 小节）和地图（详见 6.13 小节）。用于绘制图 3.7 的 R 语言代码为：

```
# 绘制中国新冠疫情地图
library(ncovr)
library(dplyr)
library(scales)
covidchina = readRDS(
  system.file("extdata", "covidchina.rds", package = "MSG"))
for (choose_date in c("01-21", "01-22")){
  print(plot_ggmap(covidchina,
    col_name = paste0("confirmed_", choose_date),
    show_capitals = FALSE,
    province_language = NA,
    add_title = choose_date))
}
```

这里用到了笔者开发的 R 附加包 **ncovr** [Zhao and Cao, 2020] 中的 `plot_ggmap()` 函数。这个函数将中国地图的数据进行了封装，包含了九段线区域（即中国地图右下角的矩形区域）。**ncovr** 包发布在 GitHub 上，安装之前需要先安装 **remotes** 包，再使用其中的 `install_github()` 函数进行远程安装（或者使用 `MSG::msg("0")` 来一次性安装复现本书图形的所有附加包）：

```
install.packages("remotes")
remotes::install_github("pzhaonet/ncovr")
```

用于绘制图 3.8 的代码比较长，除了准备数据的代码外，大部分指令都是用作装饰和细微调整的。例如绘制其中的玫瑰图的核心代码其实只有以下四行：

```
# 数据映射:  
ggplot(covid, aes(country, cum_confirm, fill=cum_confirm)) +  
# 条形图:  
  geom_col(width=1, color='grey90') +  
# 对数坐标:  
  scale_y_log10() +  
# 极坐标:  
  coord_polar()
```

其它代码用来添加参考线和文字、修改主题和配色方案、隐藏图例等。完整代码如下：

```
# 绘制世界各国疫情形势图  
library(ggplot2)  
library(patchwork)  
library(scales)  
covid = readRDS(  
  system.file("extdata", "covidcountries.rds", package = "MSG"))  
n_countries = nrow(covid)  
covid = transform(  
  covid, hjust = 1, label = paste(cum_confirm, country),  
  angle = 1: n_countries * 360/n_countries - 90 - 180/n_countries)  
second_half = (n_countries %% 2):n_countries  
covid$angle[second_half] = covid$angle[second_half] + 180  
covid$hjust[second_half] = 0  
covid$label[second_half] =  
  paste(covid$country, covid$cum_confirm)[second_half]  
  
p_polar =  
  ggplot(covid, aes(country, cum_confirm, fill=cum_confirm)) +  
  geom_col(width=1, color='grey90') +  
  geom_col(aes(y=I(10000)), width=1, fill='white', alpha = .2) +  
  geom_col(aes(y=I(1000)), width=1, fill='white', alpha = .2) +  
  geom_col(aes(y=I(100)), width=1, fill='white', alpha = .2) +  
  geom_col(aes(y=I(10)), width=1, fill = "white") +  
  scale_y_log10() +  
  scale_fill_gradientn(  
    colors=c("steelblue", "lightgreen", "orange",  
            "red", "darkred", "brown"),  
    trans="log") +
```

```

geom_text(aes(label=label, y = cum_confirm,
              angle=angle, hjust = hjust), vjust= 0.5, size = 3) +
annotate(
  "text", x = 39, y = c(10, 100, 1000, 10000)*1.5, color = "white",
  label=c(10, 100, 1000, 10000), angle = 360 / 40) +
theme_void() +
theme(legend.position="none") +
coord_polar()

p_point =
ggplot(covid, aes(country, cum_confirm)) +
geom_point(aes(color=cum_confirm), size = 2) +
scale_color_gradientn(
  colours = rev(rainbow(5)), trans="log",
  limits = 10^c(2, 5),
  breaks = 10^(2:5),
  labels = trans_format("log10", math_format(10^.x)),
  minor_breaks = as.vector(sapply(2:10, function(x) x * 10^(2:6))))+
geom_text(aes(label=country), hjust=rep(c(-0.2,1.2), 20), vjust=0.5) +
scale_y_log10(
  breaks = 10^(2:5),
  limits = c(400, 200000),
  labels = trans_format("log10", math_format(10^.x)),
  minor_breaks = as.vector(sapply(2:10, function(x) x * 10^(2:6))))+
scale_x_discrete(expand = c(0.05, 0.05)) +
labs(x = NULL, y = NULL) +
coord_flip() +
theme(legend.title = element_blank(), legend.position = c(0.9,0.3),
      axis.ticks = element_blank(), axis.text.y = element_blank(),
      legend.background = element_blank())
print(p_polar / p_point + plot_layout(heights = c(4,3)))

```

### 3.4 末日狂奔

#### 案情回放

“末日狂奔”（Canabalt）是一款速度躲避游戏，操控很简单：点击屏幕控制主角跳起，控制好跳跃力度尽量避开一路上的障碍，不要掉下楼，看最后能跑多远。这款游戏的简单与刺激吸引了很多玩家。此外，游戏中还有一定的随机成分，玩家无法预测下一步的场景是什么。