Part I

Whole Game

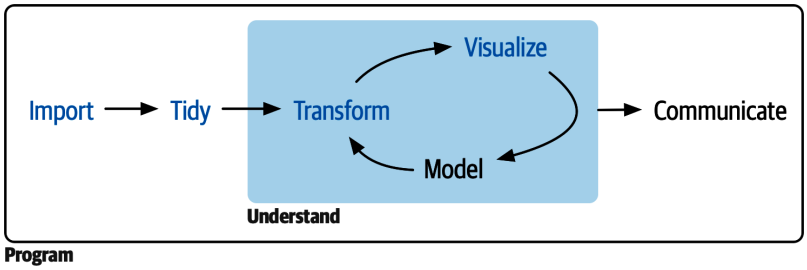

Figure 1: Here we will learn how to import, tidy, transform, and visualize data

Our goal in this part of the book is to give you a rapid overview of the main tools of data science: importing, tidying, transforming, and visualizing data, as shown in Figure 1.

Chapter 1

Data Visualization

1.1

1.1.1 数值型变量

数值变量分布的另一种可视化方法是密度图。密度图是直方图的平滑版本，也是一种实用的替代方案，特别是对于来自底层平滑分布的连续数据。它显示的细节比直方图少，但可以更轻松地快速收集分布的形状，特别是在众数和偏度方面。

1.2 可视化关系

1.2.1 数值变量和分类变量

为了可视化数值变量和分类变量之间的关系，我们可以使用并排箱线图。箱线图是描述分布的位置（百分位数）度量的一种视觉速记形式。它对于识别潜在的异常值也很有用。

请注意我们在这里使用的术语：

- 如果我们希望美学（aesthetic）所代表的视觉属性根据该变量的值而变化，那么我们将变量映射到 aesthetic。
- 否则，我们就设置 aesthetic 值

1.2.2 两个分类变量

我们可以使用堆积条形图来可视化两个分类变量之间的关系。

1.2.3 两个数值变量

可视化两个数值变量之间关系有散点图（使用创建 `geom_point()`）和平滑曲线（使用 `geom_smooth()` 创建）。散点图可能是最常用的用于可视化两个数值变量之间关系的图。

1.2.4 三个或更多变量

我们可以通过将更多变量映射到额外的美学效果来将它们合并到图中。然而，在图中添加太多美学映射会使其变得混乱且难以理解。另一种对于分类变量特别有用的方法是将图分割为分面（**facet**），即子图，每个子图显示数据的一个子集。

?? 中，你将了解许多其他几何图形，用于可视化变量的分布及其之间的关系。

1.3 保存绘图

绘制完绘图后，你可能希望将其保存为可在其他地方使用的图像，从而将其从 R 中取出。这就是 `ggsave()` 的工作，它将最近创建的绘图保存到磁盘。

如果你不指定 width，height 它们将从当前绘图设备的尺寸中获取。对于可重现的代码，你需要指定它们。ggsave()你可以在文档中了解更多信息。

不过，一般来说，我们建议你使用 Quarto 来组装最终报告，Quarto 是一种可重复的创作系统，允许你将代码和文档交错，并自动将图表包含在文章中。你将在 **??** 中了解有关 Quarto 的更多信息。

Chapter 2

数据转换

2.1 引言

tibble 是 tidyverse 使用的一种特殊类型的数据框，以避免一些常见的问题。tibbles 和数据框之间最重要的区别是 tibbles 的打印方式。它们是为大型数据集而设计的，因此它们仅显示前几行和适合一个屏幕的列。有几个选项可以查看所有内容。如果你使用 RStudio，最方便的可能是 `View(flights)`，它将打开交互式可滚动和可过滤视图。否则，你可以使用 `print(flights, width = Inf)` 显示所有列，或使用 `glimpse()`。

2.1.1 dplyr 基础知识

学习主要的 dplyr 动词（函数），这将使你能够解决绝大多数数据操作挑战。但在我们讨论它们的个体差异之前，有必要先说明一下它们的共同点：

- 第一个参数始终是数据框。
- 后续参数通常使用变量名称（不带引号）描述要操作的列。
- 输出始终是新的数据框。

dplyr 的动词根据其操作内容分为四组：rows、columns、groups 以及 table。

2.2 行

对数据集的行进行操作的最重要的动词是 filter()，它改变存在的行而不改变它们的顺序，以及 arrange() ，它改变存在的行的顺序而不改变存在的行。这两个函数都只影响行，列保持

不变。我们还将讨论 distinct()，用于查找具有唯一值。但与 arrange() 和 filter() 不同的是它还可以选择修改列。

2.2.1 filter()

filter() 允许你根据列的值保留行[1]。第一个参数是数据框，第二个和后续参数是保留该行必须满足的条件。

当使用 | 和 == 时有一个有用的快捷方式：`%in%`。它保留变量等于右侧值之一的行。

当你运行 filter() 时，会执行过滤操作，创建一个新的数据框，然后打印它。它不会修改现有数据集，因为 dplyr 函数永远不会修改其输入。要保存结果，你需要使用赋值运算符。

2.2.2 arrange()

arrange() 根据列的值更改行的顺序。它需要一个数据框和一组列名（或更复杂的表达式）来排序。如果你提供多个列名称，则每个附加列将用于打破前一列值中的联系。

在 arrange() 内部可以对某列使用 desc() 然后来根据该列按降序（从大到小）顺序对数据框重新排序。

2.2.3 distinct()

distinct() 查找数据集中的所有唯一行，因此从技术意义上来说，它主要对行进行操作。然而，大多数时候，你需要某些变量的不同组合，因此你还可以选择提供列名称。

如果你想在过滤唯一行时保留其他列，则可以使用该 `.keep_all = TRUE` 选项。distinct() 将找到数据集中第一次出现的唯一行并丢弃其余行。

如果你想查找出现的次数，最好替换 distinct() 为 count()，并且使用 `sort = TRUE` 参数可以按出现次数的降序排列它们。

2.3 列

有四个重要的动词会影响列而不更改行：mutate() 创建从现有列派生的新列、select() 更改存在的列、rename() 更改列的名称以及 relocate() 更改列的位置。

[1]稍后，你将了解系列 slice_*()，它允许你根据位置选择行。

2.3.1 mutate()

mutate() 的工作是添加根据现有列计算的新列。默认情况下，mutate() 在数据集的右侧添加新列，这使得很难看到此处发生的情况。我们可以使用 `.before` 参数将变量添加到左侧。

该 . 符号表明 .before 是函数的参数，而不是我们正在创建的第三个新变量的名称。你还可以 .after 在变量后面添加，并且在两者中 .before 都 .after 可以使用变量名称而不是位置。

或者，你可以使用 .keep 参数控制哪些变量与参数保留。一个特别有用的选项是 "used" 来指定我们只保留步骤中涉及或创建的列 mutate()。

2.3.2 select()

获取包含数百甚至数千个变量的数据集并不罕见。在这种情况下，第一个挑战通常只是关注你感兴趣的变量。select() 允许你使用基于变量名称的操作快速放大有用的子集。

通过 =，你可以 select() 使用来重命名变量。新名称出现在 = 的左侧，旧变量出现在右侧。

2.3.3 rename()

如果你想保留所有现有变量并且只想重命名一些变量，你可以使用 rename() 替换 select()。

如果你有一堆命名不一致的列，并且手动修复它们会很痛苦，请检查 janitor::clean_names() 它是否提供了一些有用的自动清理。

2.3.4 relocate()

relocate() 用于移动变量。你可能希望将相关变量收集在一起或将重要变量移到前面。默认情况下将变量移到前面。你还可以使用 .before 和 .after 参数指定放置它们的位置，与 mutate() 类似。

2.4 管道

2.5 组

当你添加与群组合作的功能时，dplyr 会变得更加强大。在本节中，我们将重点关注最重要的函数：`group_by()`、`summarize()` 和 slice 函数系列。

2.5.1 group_by()

`group_by()` 用于将数据集分为对分析有意义的组。group_by() 不会更改数据，但是，如果你仔细查看输出，你会注意到输出表明它是按分组变量“分组”的。这意味着后续操作现在将按组进行。group_by() 将此分组功能（称为类）添加到数据框中，这会更改应用于数据的后续动词的行为。

2.5.2 summarize()

最重要的分组操作是摘要，如果用于计算单个摘要统计量，则会将数据框减少为每个组只有一行[2]。

你可以在一次调用中创建任意数量的 summarize()。有各种有用的摘要，但一个非常有用的摘要是 n()，它返回每组中的行数。

2.5.3 slice_ 函数

有五个方便的函数允许你提取每个组中的特定行：

- df | > slice_head(n = 1) 取出每组的第一行。
- df | > slice_tail(n = 1) 占据每组的最后一行。
- df | > slice_min(x, n = 1) 取列 x 具有最小值的行。
- df | > slice_max(x, n = 1) 取列 x 具有最大列值的行。
- df | > slice_sample(n = 1) 随机取一行。

你可以 `n` 选择多行也可以使用 `prop = 0.1` 选择（例如）每组中 10% 的行。

2.5.4 按多个变量分组

你可以使用多个变量创建组。当你对按多个变量分组的小标题进行汇总时，每个汇总都会剥离最后一组。事后看来，这并不是让这个函数发挥作用的好方法，但是在不破坏现有代码的情况下很难进行更改。为了让发生的情况一目了然，dplyr 会显示一条消息，告诉你如何更改此行为。

如果您对此行为感到满意，您可以明确请求它以抑制该消息：`.groups = "drop_last"`，或者，通过设置不同的值来更改默认行为，例如，”drop” 删除所有分组或 ”keep” 保留相同的组。

[2] summarise()，如果你更喜欢英式英语。

这里主要是说 grouped_df 类型的对象是有一个 groups 值会自动删除最后一个组。

2.5.5 取消分组

您可能还想从数据框中删除分组而没有使用 summarize(). 您可以使用来执行此操作 ungroup()。

2.5.6 .by

dplyr 1.1.0 包含用于按操作分组的新的实验性语法，即参数 .by。`group_by()` 并且 `ungroup()` 不会消失，但您现在还可以使用 .by 参数在单个操作中进行分组。

.by 适用于所有动词，其优点是您不需要使用参数 .groups 来抑制分组消息或者分组后再使用 `ungroup()`。

2.6 案例研究：总量和样本量

How Not To Sort By Average Rating
警告：仅按均值排序可能会导致错误