

R 语言编程：基于 tidyverse

第 10 讲 数据重塑

张敬信

2022 年 3 月 24 日

哈尔滨商业大学

一. 什么是整洁数据？

- 采用 Hadley 的表述，脏的/不整洁的数据往往具有如下特点：
 - 首行（列名）是值，不是变量名
 - 多个变量放在一行
 - 变量既放在行也放在列
 - 多种类型的观测单元在同一个单元格
 - 一个观测单元放在多个表
- 而整洁数据具有如下特点：
 - 每个**变量**构成一行
 - 每个**观测**构成一行
 - 每个观测的每个变量**值**构成一个单元格

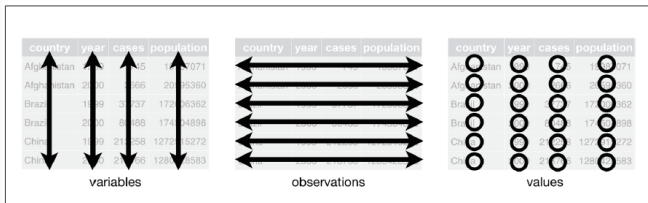


图 1: 整洁数据的 3 个特点

- tidyverse 系列包中的函数操作的都是这种整洁数据框，而不整洁数据，首先需要变成整洁数据，这就是**数据重塑**
- 数据重塑主要包括长宽表转化、拆分/合并列、方形化，用 tidyr 包实现。

- 先看一个不整洁的数据：

observation	A_count	B_count	A_dbh	B_dbh
Richmond(Sam)	7	2	100	110
Windsor(Ash)	10	5	80	87
Bilpin(Jules)	5	8	95	90

其不整洁表现在：

- observation 列有两个变量数据
- 列名中的 A/B 应是分类变量 species 的两个水平值
- 测量值列 count 和 dbh 应各占 1 列，而不是 2 列

- 用 `tidyr` 包重塑为整洁数据:

site	surveyor	speices	count	dbh
Richmond	Sam	A	7	100
Richmond	Sam	B	2	110
Windsor	Ash	A	10	80
Windsor	Ash	B	5	87
Bilpin	Jules	A	5	95
Bilpin	Jules	B	8	90

注：这里的关键是，要学会区分哪些是**变量**、**观测**、**值**。

二. 宽表变长表

- 宽表的特点是：表比较宽，本来该是“值”的，却出现在“变量（名）”中。这就需要给它变到“值”中，新起个列名存为一列，即宽表变长表：

```
pivot_longer(data, cols, names_to, values_to,  
              values_drop_na, ...)
```

- data: 要重塑的数据框
- cols: 用选择列语法选择要变形的列
- names_to: 为存放变形列的列名中的“值”，指定新列名
- values_to: 为存放变形列中的“值”，指定新列名
- values_drop_na: 是否忽略变形列中的 NA

注：若变形列的列名除了“值”外，还包含前缀、变量名 + 分隔符、正则表达式分组捕获模式，则可以借助参数 `names_prefix`, `names_sep`, `names_pattern` 来提取出“值”。

1. 每一行只有 1 个观测的情形

- 以分省年度 GDP 数据为例，每一行只有一个观测，关于一个省份的信息

```
df = read_csv("datas/分省年度 GDP.csv")
```

```
df
```

```
#> # A tibble: 4 x 4
```

```
#>   地区    `2019 年` `2018 年` `2017 年`
```

```
#>   <chr>      <dbl>      <dbl>      <dbl>
```

```
#> 1 北京市    35371.    33106.    28015.
```

```
#> 2 天津市    14104.    13363.    18549.
```

```
#> 3 河北省    35105.    32495.    34016.
```

```
#> # ... with 1 more row
```

- 要变形的列是除了地区列之外的列
- 变量 (名) 中的 2019 年、2018 年等是年份的值, 需要作为 1 列" 值" 来存放, 新起一个列名 年份
- 2019 年、2018 年等列中的值, 属于同一个变量 GDP, 新起一个列名 GDP 来存放:

```
df %>%
```

```
  pivot_longer(-地区, names_to = " 年份", values_to = "GDP")
```

```
#> # A tibble: 12 x 3  
#>   地区    年份      GDP  
#>   <chr> <chr>   <dbl>  
#> 1 北京市 2019 年 35371.  
#> 2 北京市 2018 年 33106.  
#> 3 北京市 2017 年 28015.  
#> # ... with 9 more rows
```


2. 每一行有多个观测的情形

- 以 family 数据集为例，每一行有两个观测，关于 child1 和 child2 的信息

```
load("datas/family.rda")  
knitr::kable(family, align = "c")
```

family	dob_child1	dob_child2	gender_child1	gender_child2
1	1998-11-26	2000-01-29	1	2
2	1996-06-22	NA	2	NA
3	2002-07-11	2004-04-05	2	2
4	2004-10-10	2009-08-27	1	1
5	2000-12-05	2005-02-28	2	1

- 要变形的列是除了 `family` 列之外的列;
- 变形列的列名以 “_” 分割为两部分, 用 `names_to` 指定这两部分的用途:
“`value`” 指定第一部分将继续留作列名用来存放值, 而第二部分, 即包含 “`child1`”、“`child2`”, 作为新变量 `child` 的 “值”
- 忽略变形列中的缺失值

```
family %>%
  pivot_longer(-family,
               names_to = c(".value", "child"),
               names_sep = "_",
               values_drop_na = TRUE)

#> # A tibble: 9 x 4
#>   family child  dob          gender
#>   <int> <chr>  <date>         <int>
#> 1      1  child1 1998-11-26         1
#> 2      1  child2 2000-01-29         2
#> 3      2  child1 1996-06-22         2
#> # ... with 6 more rows
```

- 学生报名信息：每一行有 3 个观测，关于 3 名队员的信息，变成每一行只有 1 名队员的信息。用到 `names_pattern` 参数和正则表达式分组捕获。

```
df = read_csv("datas/参赛队信息.csv")  
knitr::kable(df, align = "c")
```

队员 1 姓 名	队员 1 专 业	队员 2 姓 名	队员 2 专 业	队员 3 姓 名	队员 3 专 业
张三	数学	李四	英语	王五	统计学
赵六	经济学	钱七	数学	孙八	计算机

```
df %>%  
  pivot_longer(everything(),  
    names_to = c(" 队员", ".value"),  
    names_pattern = "(.*\\d)(.*)")  
#> # A tibble: 6 x 3  
#>   队员  姓名  专业  
#>   <chr> <chr> <chr>  
#> 1 队员 1 张三  数学  
#> 2 队员 2 李四  英语  
#> 3 队员 3 王五  统计学  
#> # ... with 3 more rows
```

三. 长表变宽表

- 长表的特点是：表比较长。有时候需要将分类变量的若干水平值，变成变量（列名）。这就是长表变宽表¹：

```
pivot_wider(data, id_cols, names_from, values_from,  
            values_fill, ...)
```

- data: 要重塑的数据框
- id_cols: 唯一识别观测的列，默认是除了 names_from 和 values_from 指定列之外的列
- names_from: 指定列名来自哪个变量列
- values_from: 指定列“值”来自哪个变量列
- values_fill: 若变宽后单元格值缺失，设置用何值填充

¹它与宽表变长表正好相反（二者互逆）。

- 只有一个列名列和一个值列, 比如 animals 数据集:

```
load("datas/animals.rda")
```

```
animals
```

```
#> # A tibble: 228 x 3
```

```
#>   Type      Year  Heads
```

```
#>   <chr>   <int>  <dbl>
```

```
#> 1 Sheep    2015 24943.
```

```
#> 2 Cattle   1972  2189.
```

```
#> 3 Camel    1985   559
```

```
#> # ... with 225 more rows
```

- 用 `names_from` 指定列名来自哪个变量; `values_from` 指定“值”来自哪个变量:

```
animals %>%
```

```
  pivot_wider(names_from = Type, values_from = Heads,  
              values_fill = 0)
```

```
#> # A tibble: 48 x 6
```

```
#>   Year  Sheep Cattle Camel  Goat Horse
```

```
#>   <int> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
#> 1  2015 24943.  3780.  368. 23593. 3295.
```

```
#> 2  1972 13716.  2189.  625.  4338. 2239.
```

```
#> 3  1985 13249.  2408.  559   4299. 1971
```

```
#> # ... with 45 more rows
```


- 多个列名列或多个值列，比如 `us_rent_income` 数据集有两个值列：

```
us_rent_income
```

```
#> # A tibble: 104 x 5
```

```
#>   GEOID NAME      variable estimate   moe
```

```
#>   <chr> <chr>    <chr>         <dbl> <dbl>
```

```
#> 1 01      Alabama income      24476   136
```

```
#> 2 01      Alabama rent         747     3
```

```
#> 3 02      Alaska  income      32940   508
```

```
#> # ... with 101 more rows
```

```

us_rent_income %>%
  pivot_wider(names_from = variable,
              values_from = c(estimate, moe))
#> # A tibble: 52 x 6
#>   GEOID NAME      estimate_income estimate_rent moe_income m
#>   <chr> <chr>          <dbl>          <dbl>      <dbl>
#> 1 01     Alabama      24476           747        136
#> 2 02     Alaska       32940          1200        508
#> 3 04     Arizona      27517           972        148
#> # ... with 49 more rows

```

长变宽时，经常会遇到两个问题：

- 长变宽正常会压缩行，为什么行数没变呢？
- 值不能被唯一识别，输出将包含列表列

```
df = tibble(  
  x = 1:6, y = c("A", "A", "B", "B", "C", "C"),  
  z = c(2.13, 3.65, 1.88, 2.30, 6.55, 4.21))
```

```
df
```

```
#> # A tibble: 6 x 3  
#>       x y       z  
#>   <int> <chr> <dbl>  
#> 1     1 A     2.13  
#> 2     2 A     3.65  
#> 3     3 B     1.88  
#> # ... with 3 more rows
```

- 想让 y 列提供变量名, z 列提供值, 做长变宽, 但是

```
df %>%
```

```
  pivot_wider(names_from = y, values_from = z)
```

```
#> # A tibble: 6 x 4
```

```
#>       x     A     B     C
```

```
#>   <int> <dbl> <dbl> <dbl>
```

```
#> 1     1     1 2.13 NA     NA
```

```
#> 2     2     2 3.65 NA     NA
```

```
#> 3     3     3 NA     1.88  NA
```

```
#> # ... with 3 more rows
```

这就是前面说到的第一个问题, 本来该压缩成 2 行, 但是由于 x 列的存在, 无法压缩, 只能填充 NA, 这不是想要的效果。所以, 在长变宽时要注意, 是不能带着类似 x 列这种唯一识别各行的 ID 列的。

- 那去掉 x 列，重新做长变宽，但是又遇到了前面说的第二个问题：

```
df = df[-1]
df %>%
  pivot_wider(names_from = y, values_from = z)
#> # A tibble: 1 x 3
#>   A           B           C
#>   <list>      <list>      <list>
#> 1 <dbl [2]> <dbl [2]> <dbl [2]>
```

值不能唯一识别²，结果变成了列表列，同样不是想要的结果。

²值唯一识别，是指各分组（A 组 B 组 C 组）组内元素必须要能唯一识别，否则不能区分行的先后，只能打包到列表。

- 增加一个各组的唯一识别列:

```
df = df %>%  
  group_by(y) %>%  
  mutate(n = row_number())  
df  
#> # A tibble: 6 x 3  
#> # Groups:   y [3]  
#>   y           z     n  
#>   <chr> <dbl> <int>  
#> 1 A       2.13     1  
#> 2 A       3.65     2  
#> 3 B       1.88     1  
#> # ... with 3 more rows
```

- 这才是能够长变宽的标准数据，再来做长变宽：

```
df %>%  
  pivot_wider(names_from = y, values_from = z)  
#> # A tibble: 2 x 4  
#>       n      A      B      C  
#>   <int> <dbl> <dbl> <dbl>  
#> 1     1  2.13  1.88  6.55  
#> 2     2  3.65  2.3   4.21
```

这回是想要的结果，新增加的列 n 若不想要，删除列即可。

四. 拆分列与合并列

- 拆分列与合并列也是正好相反（二者互逆）。
- `separate(data, col, into, sep, ...)`: 按分隔符 `sep` 将一列拆分为多列

```
table3
```

```
#> # A tibble: 6 x 3  
#>   country      year rate  
#> * <chr>      <int> <chr>  
#> 1 Afghanistan 1999 745/19987071  
#> 2 Afghanistan 2000 2666/20595360  
#> 3 Brazil       1999 37737/172006362  
#> # ... with 3 more rows
```



```

table3 %>%
  separate(rate, into = c("cases", "population"), sep = "/",
            convert = TRUE)      # 同时转化为数值型

#> # A tibble: 6 x 4
#>   country      year cases population
#>   <chr>      <int> <int>      <int>
#> 1 Afghanistan 1999    745    19987071
#> 2 Afghanistan 2000   2666    20595360
#> 3 Brazil      1999  37737    172006362
#> # ... with 3 more rows

```

- `separate_rows()`: 可对不定长的列进行分列，并按行堆叠放置

```
df = tibble(Class = c("1 班", "2 班"),  
             Name = c(" 张三, 李四, 王五", " 赵六, 钱七"))  
df  
#> # A tibble: 2 x 2  
#>   Class Name  
#>   <chr> <chr>  
#> 1 1 班   张三, 李四, 王五  
#> 2 2 班   赵六, 钱七
```

```
df1 = df %>%  
  separate_rows(Name, sep = ", ")  
df1  
#> # A tibble: 5 x 2  
#>   Class Name  
#>   <chr> <chr>  
#> 1 1 班    张三  
#> 2 1 班    李四  
#> 3 1 班    王五  
#> # ... with 2 more rows
```

- 若要逆操作还原回去:

```
df1 %>%  
  group_by(Class) %>%  
  summarise(Name = str_c(Name, collapse = ", "))
```

- `unite(data, col, sep, ...)`: 用分隔符 `sep` 将多列合并为一列

table5

```
#> # A tibble: 6 x 4
```

```
#>   country      century year  rate
```

```
#> * <chr>      <chr>    <chr> <chr>
```

```
#> 1 Afghanistan 19      99    745/19987071
```

```
#> 2 Afghanistan 20      00    2666/20595360
```

```
#> 3 Brazil       19      99    37737/172006362
```

```
#> # ... with 3 more rows
```

```
table5 %>%  
  unite(new, century, year, sep = "")  
#> # A tibble: 6 x 3  
#>   country      new    rate  
#>   <chr>      <chr> <chr>  
#> 1 Afghanistan 1999  745/19987071  
#> 2 Afghanistan 2000  2666/20595360  
#> 3 Brazil      1999  37737/172006362  
#> # ... with 3 more rows
```

- 综合示例：重塑世界银行人口数据。

```
world_bank_pop
```

```
#> # A tibble: 1,056 x 20
```

```
#>   country indicator   `2000`   `2001`   `2002`   `2003`   `2004`
```

```
#>   <chr>    <chr>          <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
```

```
#> 1 ABW      SP.URB.TOTL 42444     43048     43670     4.42e4 4.44e4
```

```
#> 2 ABW      SP.URB.GROW    1.18      1.41      1.43    1.31e0 9.70e-1
```

```
#> 3 ABW      SP.POP.TOTL 90853     92898     94992     9.70e4 9.73e4
```

```
#> # ... with 1,053 more rows, and 11 more variables: 2007 <dbl>, 2008 <dbl>
```

```
#> #   2009 <dbl>, 2010 <dbl>, 2011 <dbl>, 2012 <dbl>, 2013 <dbl>, 2014 <dbl>
```

```
#> #   2015 <dbl>, 2016 <dbl>, 2017 <dbl>
```

- 先从最显然的入手：年份跨过了多个列，应该宽表变长表：

```
pop2 = world_bank_pop %>%  
  pivot_longer(`2000`:`2017`, names_to = "year",  
               values_to = "value")
```

```
pop2
```

```
#> # A tibble: 19,008 x 4  
#>   country indicator   year  value  
#>   <chr>    <chr>      <chr> <dbl>  
#> 1 ABW      SP.URB.TOTL 2000  42444  
#> 2 ABW      SP.URB.TOTL 2001  43048  
#> 3 ABW      SP.URB.TOTL 2002  43670  
#> # ... with 19,005 more rows
```

- 再来考察 indicator 变量:

```
pop2 %>%  
  count(indicator)  
#> # A tibble: 4 x 2  
#>   indicator      n  
#>   <chr>      <int>  
#> 1 SP.POP.GROW  4752  
#> 2 SP.POP.TOTL  4752  
#> 3 SP.URB.GROW  4752  
#> # ... with 1 more row
```

这里, SP.POP.GROW 为人口增长率, SP.POP.TOTL 为总人口, SP.URB.* 也类似, 只是城市的。将该列值拆分为两个变量: area (URB, POP) 和 variable (GROW, TOTL)


```
pop3 = pop2 %>%  
  separate(indicator, c(NA, "area", "variable"))  
pop3  
#> # A tibble: 19,008 x 5  
#>   country area  variable year  value  
#>   <chr>   <chr> <chr>    <chr> <dbl>  
#> 1 ABW     URB    TOTL    2000  42444  
#> 2 ABW     URB    TOTL    2001  43048  
#> 3 ABW     URB    TOTL    2002  43670  
#> # ... with 19,005 more rows
```

- 最后, 再将分类变量 `variable` 的水平值变为列名 (长表变宽表), 就完成重塑:

```
pop3 %>%  
  pivot_wider(names_from = variable, values_from = value)  
#> # A tibble: 9,504 x 5  
#>   country area  year  TOTL  GROW  
#>   <chr>   <chr> <chr> <dbl> <dbl>  
#> 1 ABW     URB    2000  42444  1.18  
#> 2 ABW     URB    2001  43048  1.41  
#> 3 ABW     URB    2002  43670  1.43  
#> # ... with 9,501 more rows
```

五. 方形化

方形化 (Rectangling) 是将一个深度嵌套的列表 (通常来自 JSON 或 XML) 驯服成一个整齐的行和列的数据集。主要通过组合使用以下函数实现:

- `unnest_longer()`: 提取列表列的每个元, 再按行存放 (横向展开)
- `unnest_wider()`: 提取列表列的每个元, 再按列存放 (纵向展开)
- `unnest_auto()`: 提取列表列的每个元, 猜测按行或按列存放
- `hoist()`: 类似 `unnest_wider()`, 但只取出选择的组件, 且可以深入多个层

- 以权力游戏角色数据集 `got_chars` 为例，它是个长度为 30 的列表，里面又嵌套很多列表。一种技巧是，先把它创建成 `tibble` 方便后续操作：

```
library(repurrrsive)    # 使用 got_chars 数据集
chars = tibble(char = got_chars)
chars

#> # A tibble: 30 x 1
#>   char
#>   <list>
#> 1 <named list [18]>
#> 2 <named list [18]>
#> 3 <named list [18]>
#> # ... with 27 more rows
```

- char 是嵌套列表列，每个元素又是长度为 18 的列表，先横向展开它们：

```
chars1 = chars %>%
```

```
  unnest_wider(char)
```

```
chars1
```

```
#> # A tibble: 30 x 18
```

```
#>   url          id name  gender culture born   died alive
```

```
#>   <chr>      <int> <chr>  <chr>   <chr>   <chr> <chr> <lglt>
```

```
#> 1 https://~  1022 Theon~ Male    "Ironb~ In 27~ ""    TRUE
```

```
#> 2 https://~  1052 Tyrio~ Male    ""      In 27~ ""    TRUE
```

```
#> 3 https://~  1074 Victa~ Male    "Ironb~ In 26~ ""    TRUE
```

```
#> # ... with 27 more rows, and 7 more variables: mother <chr>
```

```
#> #   allegiances <list>, books <list>, povBooks <list>, tvS
```

```
#> #   playedBy <list>
```

- 生成一个表，以匹配人物角色和他们的外号，name 直接选择列，外号来自列表列 titles, 纵向展开它：

```
chars1 %>%  
  select(name, title = titles) %>%  
  unnest_longer(title)  
#> # A tibble: 60 x 2  
#>   name          title  
#>   <chr>         <chr>  
#> 1 Theon Greyjoy Prince of Winterfell  
#> 2 Theon Greyjoy Captain of Sea Bitch  
#> 3 Theon Greyjoy Lord of the Iron Islands (by law of the gr  
#> # ... with 57 more rows
```

- 或者用 `hoist()` 直接从内层提取想要的列，再对列表列 `title` 做纵向展开：

```
chars %>%
  hoist(char, name = "name", title = "titles") %>%
  unnest_longer(title)

#> # A tibble: 60 x 3
#>   name          title
#>   <chr>         <chr>
#> 1 Theon Greyjoy Prince of Winterfell
#> 2 Theon Greyjoy Captain of Sea Bitch
#> 3 Theon Greyjoy Lord of the Iron Islands (by law of the gr
#> # ... with 57 more rows
```

注：还有 `tibblify` 包专门做嵌套列表转化为 `tibble` 数据框。

本篇主要参阅 (张敬信, 2022), (Hadley Wickham, 2017), (Desi Quintans, 2019), 以及包文档, 模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

Desi Quintans, J. P. (2019). *Working in the Tidyverse*. HIE Advanced R workshop.

Hadley Wickham, G. G. (2017). *R for Data Science*. O' Reilly, 1 edition. ISBN 978-1491910399.

张敬信 (2022). *R 语言编程：基于 tidyverse*. 人民邮电出版社, 北京.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.