[R 数据科学]Tidy data 案例

庄亮亮

目录

1 Tidy data 介绍

在本章中，你将学习在 R 中组织数据的一种一致的方法，这种组织被称为 tidy data。将数据转换为这种格式需要一些前期工作，但这些工作从长远来看是值得的。一旦你有了整洁的数据和一些包提供的整洁工具，您将花费很少时间将数据从一种表示转换到另一种，从而可以将更多的时间花在分析问题上。

本文将为您提供整理数据的实用介绍以及 tidyr 包中附带的工具。如果你想了解更多的基本理论，你可能会喜欢发表在《统计软件杂志》上的整洁数据论文http://www.jstatsoft.org/v59/i10/paper

2 数据清洗案例

我们主要通过一个案例, 来了解如何整洁数据, 并将案例中的各个有用函数进行详细解读. 该案例数据来自 `tidyr::who`, 其包含按年份，国家，年龄，性别和诊断方法细分的结核病（TB）病例。数据来自 2014 年世界卫生组织《全球结核病报告》，网址为http://www.who.int/tb/country/data/download/en/。

```
library(tidyverse)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
## -- Attaching packages ---------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
## Warning: package 'tibble' was built under R version 3.6.3
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

```
## -- Conflicts ------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
who
```

```
## # A tibble: 7,240 x 60
##    country iso2  iso3   year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544
##    <chr>   <chr> <chr> <int>       <int>        <int>        <int>        <int>
##  1 Afghan~ AF    AFG    1980          NA           NA           NA           NA
##  2 Afghan~ AF    AFG    1981          NA           NA           NA           NA
##  3 Afghan~ AF    AFG    1982          NA           NA           NA           NA
##  4 Afghan~ AF    AFG    1983          NA           NA           NA           NA
##  5 Afghan~ AF    AFG    1984          NA           NA           NA           NA
##  6 Afghan~ AF    AFG    1985          NA           NA           NA           NA
##  7 Afghan~ AF    AFG    1986          NA           NA           NA           NA
##  8 Afghan~ AF    AFG    1987          NA           NA           NA           NA
##  9 Afghan~ AF    AFG    1988          NA           NA           NA           NA
## 10 Afghan~ AF    AFG    1989          NA           NA           NA           NA
## # ... with 7,230 more rows, and 52 more variables: new_sp_m4554 <int>,
## #   new_sp_m5564 <int>, new_sp_m65 <int>, new_sp_f014 <int>,
## #   new_sp_f1524 <int>, new_sp_f2534 <int>, new_sp_f3544 <int>,
## #   new_sp_f4554 <int>, new_sp_f5564 <int>, new_sp_f65 <int>,
## #   new_sn_m014 <int>, new_sn_m1524 <int>, new_sn_m2534 <int>,
## #   new_sn_m3544 <int>, new_sn_m4554 <int>, new_sn_m5564 <int>,
## #   new_sn_m65 <int>, new_sn_f014 <int>, new_sn_f1524 <int>,
## #   new_sn_f2534 <int>, new_sn_f3544 <int>, new_sn_f4554 <int>,
## #   new_sn_f5564 <int>, new_sn_f65 <int>, new_ep_m014 <int>,
## #   new_ep_m1524 <int>, new_ep_m2534 <int>, new_ep_m3544 <int>,
## #   new_ep_m4554 <int>, new_ep_m5564 <int>, new_ep_m65 <int>,
## #   new_ep_f014 <int>, new_ep_f1524 <int>, new_ep_f2534 <int>,
## #   new_ep_f3544 <int>, new_ep_f4554 <int>, new_ep_f5564 <int>,
## #   new_ep_f65 <int>, newrel_m014 <int>, newrel_m1524 <int>,
## #   newrel_m2534 <int>, newrel_m3544 <int>, newrel_m4554 <int>,
## #   newrel_m5564 <int>, newrel_m65 <int>, newrel_f014 <int>,
## #   newrel_f1524 <int>, newrel_f2534 <int>, newrel_f3544 <int>,
## #   newrel_f4554 <int>, newrel_f5564 <int>, newrel_f65 <int>
```

这是一个非常典型的现实示例数据集。它包含**冗余列，奇数变量代码和许多缺失值**。我们需要采取多个步骤来对其进行整理。

2.1 不是变量的列汇集在一起

首先将不是变量的列聚集在一起。所包含的列包括：

- country，iso2 和 iso3 是三个指定国家/地区的变量。
- year 是一个变量。
- 变量名中给出的结构（例如 new_sp_m014，new_ep_m014，new_ep_f014）可能是值，而不是变量。

因此，我们需要将从 `new_sp_m014` 到 `newrel_f65` 的所有列汇总在一起。我们用通用名称 “key” 来表示他们。我们知道单元格代表案件数，因此我们将变量数存储在 `cases` 中, 并用 `na.rm` 去除含有缺失值的行。

这里使用 `pivot_longer()` 将数据变长, 具体见后面函数详情.

```
who1 <- who %>%
    pivot_longer(
        cols = new_sp_m014:newrel_f65,
        names_to = 'key',
        values_to = 'cases',
        values_drop_na = T
)

who1
```

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key          cases
##    <chr>       <chr> <chr> <int> <chr>        <int>
##  1 Afghanistan AF    AFG    1997 new_sp_m014      0
##  2 Afghanistan AF    AFG    1997 new_sp_m1524    10
##  3 Afghanistan AF    AFG    1997 new_sp_m2534     6
##  4 Afghanistan AF    AFG    1997 new_sp_m3544     3
##  5 Afghanistan AF    AFG    1997 new_sp_m4554     5
```

```
##  6 Afghanistan AF    AFG    1997 new_sp_m5564     2
##  7 Afghanistan AF    AFG    1997 new_sp_m65       0
##  8 Afghanistan AF    AFG    1997 new_sp_f014      5
##  9 Afghanistan AF    AFG    1997 new_sp_f1524    38
## 10 Afghanistan AF    AFG    1997 new_sp_f2534    36
## # ... with 76,036 more rows
```

对 key 进行计数，我们可以得到一些有关值结构的提示：

```
who1 %>% count(key)
```

```
## # A tibble: 56 x 2
##    key              n
##    <chr>        <int>
##  1 new_ep_f014   1032
##  2 new_ep_f1524  1021
##  3 new_ep_f2534  1021
##  4 new_ep_f3544  1021
##  5 new_ep_f4554  1017
##  6 new_ep_f5564  1017
##  7 new_ep_f65    1014
##  8 new_ep_m014   1038
##  9 new_ep_m1524  1026
## 10 new_ep_m2534  1020
## # ... with 46 more rows
```

其中 key 的具体含义，查阅可得：
每列的前三个字母：新、旧病例。
之后两个字母：结核的类型。

- rel 代表复发病例
- ep 代表肺外结核病例
- sn 代表无法通过肺部涂片诊断（涂片阴性）的肺结核病例
- sp 代表可被诊断为肺部涂片（涂片阳性）的肺结核病例

第六字母：结核病患者的性别。男性（m）和女性（f）
其余数字给出了年龄段。数据集将案例分为七个年龄组：

- 014 = 0 – 14 岁
- 1524 = 15 – 24 岁
- 2534 = 25 – 34 岁
- 3544 = 35 – 44 岁
- 4554 = 45 – 54 岁
- 5564 = 55 – 64 岁
- 65 = 65 岁或以上

2.2 替换数据

我们需要对列名称的格式进行较小的修正：将 new_rel 替换为 newrel（很难在这里找到它，但是如果您不修正它，我们将在后续步骤中出错）。这里使用了 stringr 包中的 str_replace(), 将 `newrel` 替换 `new_rel`.

```
who2 <- who1 %>%
    mutate( names_from = stringr::str_replace(key,'newrel','new_rel')
    )
who2
```

```
## # A tibble: 76,046 x 7
##    country     iso2  iso3   year key          cases names_from
##    <chr>       <chr> <chr> <int> <chr>        <int> <chr>
##  1 Afghanistan AF    AFG    1997 new_sp_m014      0 new_sp_m014
##  2 Afghanistan AF    AFG    1997 new_sp_m1524    10 new_sp_m1524
##  3 Afghanistan AF    AFG    1997 new_sp_m2534     6 new_sp_m2534
##  4 Afghanistan AF    AFG    1997 new_sp_m3544     3 new_sp_m3544
##  5 Afghanistan AF    AFG    1997 new_sp_m4554     5 new_sp_m4554
##  6 Afghanistan AF    AFG    1997 new_sp_m5564     2 new_sp_m5564
##  7 Afghanistan AF    AFG    1997 new_sp_m65       0 new_sp_m65
```

```
##  8 Afghanistan AF    AFG    1997 new_sp_f014      5 new_sp_f014
##  9 Afghanistan AF    AFG    1997 new_sp_f1524    38 new_sp_f1524
## 10 Afghanistan AF    AFG    1997 new_sp_f2534    36 new_sp_f2534
## # ... with 76,036 more rows
```

2.3 字符分割

接下来就是将 key 中的字符进行分割, 我们使用 separate() 对字符进行两次分割.

1. 将在每个下划线处拆分代码。

```
who3 <- who2 %>%
    separate(key,c('new','type','sexage'),sep = '_')
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 2580 rows [243,
## 244, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 903,
## 904, 905, 906, ...].
```

```
who3
```

```
## # A tibble: 76,046 x 9
##    country     iso2  iso3   year new   type  sexage cases names_from
##    <chr>       <chr> <chr> <int> <chr> <chr> <chr>  <int> <chr>
##  1 Afghanistan AF    AFG    1997 new   sp    m014       0 new_sp_m014
##  2 Afghanistan AF    AFG    1997 new   sp    m1524     10 new_sp_m1524
##  3 Afghanistan AF    AFG    1997 new   sp    m2534      6 new_sp_m2534
##  4 Afghanistan AF    AFG    1997 new   sp    m3544      3 new_sp_m3544
##  5 Afghanistan AF    AFG    1997 new   sp    m4554      5 new_sp_m4554
##  6 Afghanistan AF    AFG    1997 new   sp    m5564      2 new_sp_m5564
##  7 Afghanistan AF    AFG    1997 new   sp    m65        0 new_sp_m65
##  8 Afghanistan AF    AFG    1997 new   sp    f014       5 new_sp_f014
##  9 Afghanistan AF    AFG    1997 new   sp    f1524     38 new_sp_f1524
## 10 Afghanistan AF    AFG    1997 new   sp    f2534     36 new_sp_f2534
## # ... with 76,036 more rows
```

利用 select() 删除没用的列:new,iso2，iso3.

```
who3 %>% count(new)
```

```
## # A tibble: 2 x 2
##   new        n
##   <chr>  <int>
## 1 new    73466
## 2 newrel  2580
```

```
who4 <- who3 %>% select(-new,-iso2,-iso3)
who4
```

```
## # A tibble: 76,046 x 6
##    country      year type  sexage cases names_from
##    <chr>       <int> <chr> <chr>  <int> <chr>
##  1 Afghanistan  1997 sp    m014       0 new_sp_m014
##  2 Afghanistan  1997 sp    m1524     10 new_sp_m1524
##  3 Afghanistan  1997 sp    m2534      6 new_sp_m2534
##  4 Afghanistan  1997 sp    m3544      3 new_sp_m3544
##  5 Afghanistan  1997 sp    m4554      5 new_sp_m4554
##  6 Afghanistan  1997 sp    m5564      2 new_sp_m5564
##  7 Afghanistan  1997 sp    m65        0 new_sp_m65
##  8 Afghanistan  1997 sp    f014       5 new_sp_f014
##  9 Afghanistan  1997 sp    f1524     38 new_sp_f1524
## 10 Afghanistan  1997 sp    f2534     36 new_sp_f2534
## # ... with 76,036 more rows
```

2. 将分离 sexage 到 sex 和 age 通过的第一个字符后拆分：

```
who5 <- who4 %>%
    separate(sexage,c('sex','age'),sep=1)
who5
```

```
## # A tibble: 76,046 x 7
##    country      year type  sex   age   cases names_from
##    <chr>       <int> <chr> <chr> <chr> <int> <chr>
```

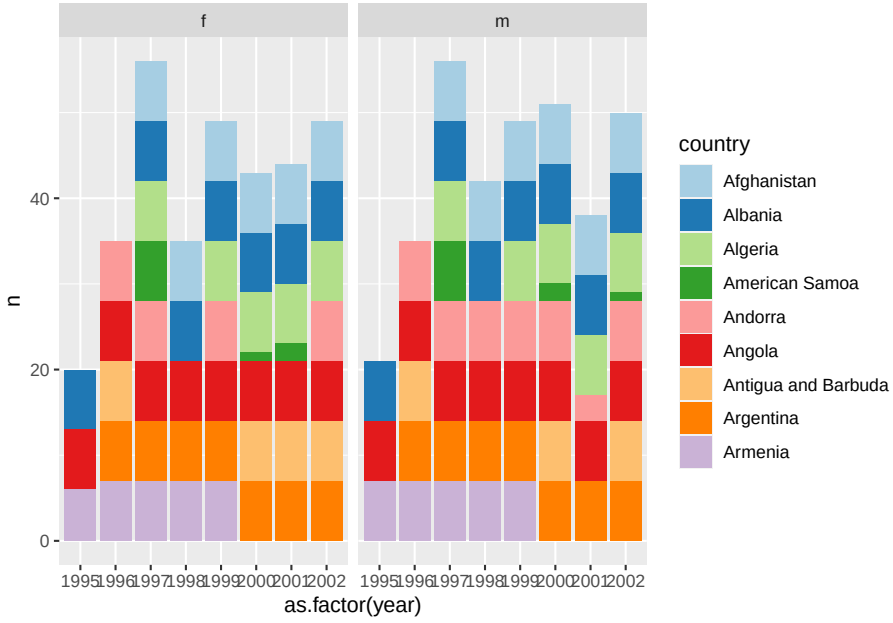
```
##  1 Afghanistan  1997 sp    m     014       0 new_sp_m014
##  2 Afghanistan  1997 sp    m     1524     10 new_sp_m1524
##  3 Afghanistan  1997 sp    m     2534      6 new_sp_m2534
##  4 Afghanistan  1997 sp    m     3544      3 new_sp_m3544
##  5 Afghanistan  1997 sp    m     4554      5 new_sp_m4554
##  6 Afghanistan  1997 sp    m     5564      2 new_sp_m5564
##  7 Afghanistan  1997 sp    m     65        0 new_sp_m65
##  8 Afghanistan  1997 sp    f     014       5 new_sp_f014
##  9 Afghanistan  1997 sp    f     1524     38 new_sp_f1524
## 10 Afghanistan  1997 sp    f     2534     36 new_sp_f2534
## # ... with 76,036 more rows
```

这时,who 数据集是目前整洁!

2.4 可视化

数据清洗完毕, 就可以做一些初步的可视化, 探索性分析. 这里简单绘制了前几个国家不同年份, 不同性别的结核病病例总数。

```
who5 %>% group_by(country,year,sex)  %>% filter(year<2003) %>%
    count() %>%
    head(100) %>%
    ggplot(aes(x=as.factor(year),y=n,fill=country))+geom_col() +facet_wrap(~sex,nrow =
      scale_fill_brewer(palette = "Paired")
```



2.5 复杂的管道函数

事实上你可以直接只用管道函数构建一个复杂的函数：

```
who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  mutate(
    key = stringr::str_replace(key, "newrel", "new_rel")
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

Figure 12.2: Pivoting `table4` into a longer, tidy form.

图 1: pivot_longer，从左到右

3 所用函数详细解释

3.1 pivot_longer()、poivot_wider()

`pivot_longer()` 将在列中列名（数值）转换到一列上。具体可见下图, 将列变量转化为数据存在 year 列名中, 相当于把数据变长 (longer).

函数主要参数:

> cols 选取的列 names_to 字符串，指定要从数据的列名中存储的数据创建的列的名称。values_to 字符串，指定要从存储在单元格值中的数据创建的列的名称。values_drop_na 如果为真，将删除 value_to 列中只包含 NAs 的行。这有效地将显式缺失值转换为隐式缺失值，通常只在由其结构创建的数据中缺失值时使用.

例子如上面例子: 将 new_sp_m014 到 newrel_f65 之间的列选取, 汇总到 `key` 列名中, 值存在 `cases` 列名中, 并将含有缺失值的行进行删除.

```
who1 <- who %>%
    pivot_longer(
        cols = new_sp_m014:newrel_f65,
        names_to = 'key',
        values_to = 'cases',
```

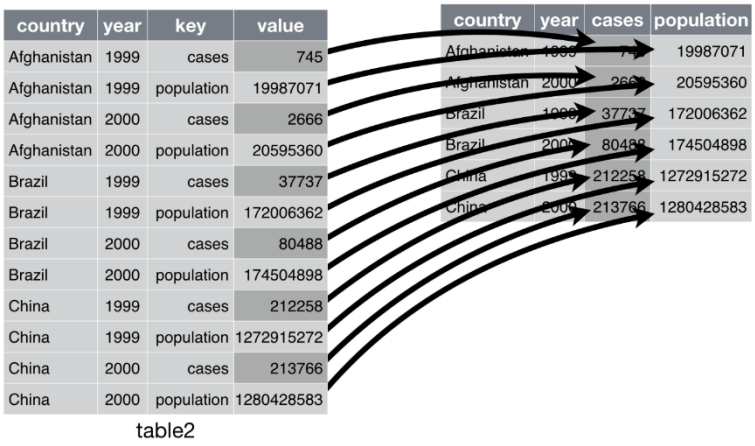


Figure 12.3: Pivoting `table2` into a "wider", tidy form.

图 2: pivot_wider，从右到左

```
        values_drop_na = T
)
```

当然还有一个和他相反功能的函数 `poivot_wider()`。具体见下图, 相当于把 key 中的值变为列名, 对应的 values 数据转化到 population 中. 下面是简单的例子.

```
library(tidyverse)
stocks <- tibble(
  year   = c(2015, 2015, 2016, 2016),
  half   = c(   1,    2,     1,    2),
  return = c(1.88, 0.59, 0.92, 0.17)
)
stocks
```

```
## # A tibble: 4 x 3
##    year  half return
##   <dbl> <dbl>  <dbl>
```

```
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2016     1   0.92
## 4  2016     2   0.17
```

我们将数据变宽, 将 year 变为列名, 对应在 return 中的数据进行填充.

```
stocks %>%
    pivot_wider(names_from = year,values_from = return)
```

```
## # A tibble: 2 x 3
##    half `2015` `2016`
##   <dbl>  <dbl>  <dbl>
## 1     1   1.88   0.92
## 2     2   0.59   0.17
```

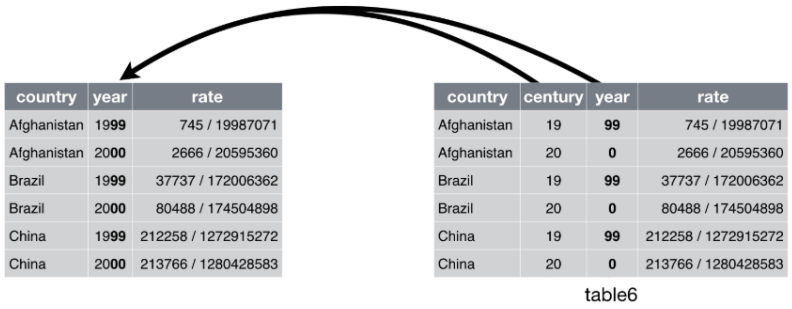
3.2 separate()

该函数可将字符进行分割, 具体案例如上.

默认情况下，当 separate() 看到非字母数字字符 (即不是数字或字母的字符) 时，它将分割值。可以用里面的参数 `sep`。比如：`sep='_'`。他还有一个功能, 当 `sep=2` 时, 可通过第二个位置进行分割, 使用在省份市级, 等数据上. 例如以下函数, 其中 into = c("century", "year") 将原始分割后的数据导入两个新列上, 分别叫 century 和 year.

```
table3 %>%
  separate(year, into = c("century", "year"), sep = 2)
```

注意: 默认情况下，会转化成字符形式，你可以用参数 `convert=T`，将数据转化最佳结构.



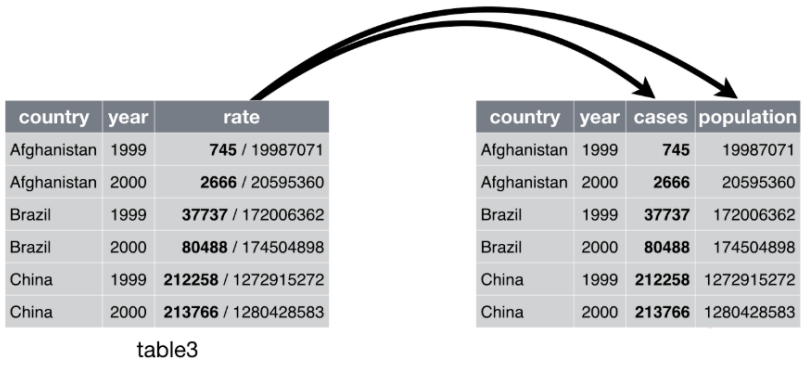
country	year	rate
Afghanistan	1999	745 / 19987071
Afghanistan	2000	2666 / 20595360
Brazil	1999	37737 / 172006362
Brazil	2000	80488 / 174504898
China	1999	212258 / 1272915272
China	2000	213766 / 1280428583

country	century	year	rate
Afghanistan	19	99	745 / 19987071
Afghanistan	20	0	2666 / 20595360
Brazil	19	99	37737 / 172006362
Brazil	20	0	80488 / 174504898
China	19	99	212258 / 1272915272
China	20	0	213766 / 1280428583

table6

Figure 12.5: Uniting `table5` makes it tidy

图 3: unite



country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

country	year	cases	population
Afghanistan	1999	**745**	19987071
Afghanistan	2000	**2666**	20595360
Brazil	1999	**37737**	172006362
Brazil	2000	**80488**	174504898
China	1999	**212258**	1272915272
China	2000	**213766**	1280428583

table3

Figure 12.4: Separating `table3` makes it tidy

3.3 unite

是 `separate()` 的反函数, 这里做个补充
默认情况下，`sep='_'` 如果我们不需要任何分隔符可以使用 `sep=''`.

4 缺失值处理

两种情况会出现缺失值:

1. **显式**: 标记为 NA.
2. **隐式**: 没出在书中的.

以下为例子

```
stocks <- tibble(
  year   = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
  qtr    = c(   1,    2,    3,    4,    2,    3,    4),
  return = c(1.88, 0.59, 0.35,   NA, 0.92, 0.17, 2.66)
)
```

在这个数据集中有两个缺失的值:

1. 2015 年第四季度出现显式丢失，因为它的值应该在的单元格中包含 NA。
2. 2016 年第一季度的收益是隐式缺失的，因为它根本没有出现在数据集中。

对于隐式缺失值, 我们可以将数据进行变化, 显示隐性缺失.

```
stocks %>%
    pivot_wider(names_from = year,values_from = return)
```

```
## # A tibble: 4 x 3
##     qtr `2015` `2016`
##   <dbl>  <dbl>  <dbl>
## 1     1   1.88  NA
## 2     2   0.59   0.92
## 3     3   0.35   0.17
## 4     4  NA      2.66
```

这样就可以看到原来隐型的缺失值了.

这些显式缺失的值在数据的其他表示中可能并不重要，你可以在 pivot_longer() 中设置 values_drop_na = TRUE 来将显式缺失的值变成隐式缺失值:

```
stocks %>%
  pivot_wider(names_from = year, values_from = return) %>%
  pivot_longer(
    cols = c(`2015`, `2016`),
    names_to = "year",
    values_to = "return",
    values_drop_na = TRUE
  )
```

```
## # A tibble: 6 x 3
##     qtr year  return
##   <dbl> <chr>  <dbl>
## 1     1 2015    1.88
## 2     2 2015    0.59
## 3     2 2016    0.92
## 4     3 2015    0.35
## 5     3 2016    0.17
## 6     4 2016    2.66
```

4.0.0.1 complete()

在整洁数据中显式显示缺失值的另一个重要工具是 complete().

complete() 获取一组列，并查找所有唯一的组合。然后，它确保原始数据集包含所有这些值，并在必要时显式填充 NAs。

```
stocks %>%
    complete(year,qtr)
```

```
## # A tibble: 8 x 3
##    year   qtr return
##   <dbl> <dbl>  <dbl>
## 1  2015     1   1.88
## 2  2015     2   0.59
## 3  2015     3   0.35
## 4  2015     4  NA
```

```
## 5  2016     1  NA
## 6  2016     2   0.92
## 7  2016     3   0.17
## 8  2016     4   2.66
```

有时，当一个数据源主要用于数据输入时，丢失的值表明以前的值应该结转:

```
treatment <- tribble(
    ~person,~treatment,~response,
    'A',1,7,
    NA,2,10,
    NA,3,9,
    'B',1,4
)
```

4.0.0.2 fill()

您可以使用 fill() 填充这些缺失的值。它采用一组列，在这些列中，您希望用最近的非缺失值 (有时称为最后一次观察结转) 替换缺失值。

```
treatment %>% fill(person)
```

```
## # A tibble: 4 x 3
##   person treatment response
##   <chr>      <dbl>    <dbl>
## 1 A              1        7
## 2 A              2       10
## 3 A              3        9
## 4 B              1        4
```

直接将缺失值填写为前面的字符 ('A').