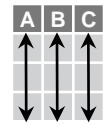


用 tidy 清理数据：速查表



清理数据 (Tidy data) 这种方法主要用于整理列表数据，使其具有跨包兼容的一致性数据结构。一个整洁的数据集应具备如下特征：



每个变量
占一列 (column)

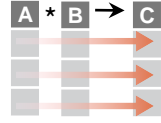
&



每组观测值，或每个案例，
占一行 (row)



以向量 (vectors)
的形式访问变量



通过向量化运算
保存案例

Tibbles

一种扩展型数据框



Tibbles 作为一种列表格式，由 **tibble** 数据包提供。其继承了 DataFrame 类型，但有如下改进：

- **取子集** `[]` 仍返回 tibble，`[]` 和 `$` 返回向量
- 返回列时，**不进行部分匹配** (不存在就不返回)
- **显示** 在屏幕上显示数据的简洁视图

`options(tibble.print_max = n, tibble.print_min = m, tibble.width = Inf)` 控制默认的显示设置

`View()` 或 `glimpse()` 观察整个数据集

创建一个 **TIBBLE** 数据框

`tibble(...)` 根据列(columns)创建
`tibble(x = 1:3, y = c("a", "b", "c"))`

`tribble(...)` 根据行(rows)创建
`tribble(~x, ~y,`

```
1, "a",  
2, "b",  
3, "c")
```

两种创建方法
结果一致

```
A tibble: 3 x 2  
  x     y  
  <int> <chr>  
1     1  a  
2     2  b  
3     3  c
```

`as_tibble(x, ...)` 将 DataFrame 类型转换为 tibble

`enframe(x, name = "name", value = "value")`
将向量转成 tibble，反之则为 `deframe()`。

`is_tibble(x)` 检测 x 是否为 tibble 类型

重组数据

- 数据透视，调整格局

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

`pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE)`

将多列折叠为两列，从而“拉长”数据。

原列名转移至新的 `names_to` 列，

原数值转移至新的 `values_to` 列。

`pivot_longer(table4a, cols = 2:3, names_to = "year", values_to = "cases")`

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

`pivot_wider(data, names_from = "name", values_from = "value")`

和 `pivot_longer()` 相反，将两列扩展为多列，从而“加宽”数据。

原来的两列数据，一列扩展为不同的新列名，另一列的数值也随之重新分布。

`pivot_wider(table2, names_from = type, values_from = count)`

拆分单元格

- 将单元格拆分或合并为独立数值

table5

country	century	year
A	19	99
A	20	00
B	19	99
B	20	00

→

country	year
A	1999
A	2000
B	1999
B	2000

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M

table3

country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M

→

country	year	rate
A	1999	0.7K
A	1999	19M
A	2000	2K
A	2000	20M
B	1999	37K
B	1999	172M
B	2000	80K
B	2000	174M

`unite(data, col, ..., sep = "_", remove = TRUE, na.rm = FALSE)` 多列合并为一列

`unite(table5, century, year, col = "year", sep = "")`

`separate(data, col, into, sep = "[^:alnum:]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...)` 一列拆分为多列

也可用 `extract()`。

`separate(table3, rate, sep = "/", into = c("cases", "pop"))`

`separate_rows(data, ..., sep = "[^:alnum:].+", convert = FALSE)` 一列拆分为多行

`separate_rows(table3, rate, sep = "/")`

扩展表格

创建新的变量组合，或识别隐式缺失值(即没有呈现在数据集中的变量组合)。

X

x1	x2	x3
A	1	3
B	1	4
B	2	3

→

x1	x2
A	1
A	2
B	1
B	2

`expand(data, ...)`

创建一个新 tibble，包含原数据 ... 列中所有可能的变量取值组合，同时舍去其他未选中的变量。

`expand(mtcars, cyl, gear, carb)`

X

x1	x2	x3
A	1	3
B	1	4
B	2	3

→

x1	x2	x3
A	1	3
A	2	NA
B	1	4
B	2	3

`complete(data, ..., fill = list())` 补全 ... 列中所有可能的变量取值组合，并在剩余的变量中相应填充为缺失值。

`complete(mtcars, cyl, gear, carb)`

处理缺失值

舍弃或替换明确的缺失值(NA)。

X

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
D	3

`drop_na(data, ...)`

舍弃 ... 列中含有缺失值的所有行

`drop_na(x, x2)`

X

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	1
C	1
D	3
E	3

`fill(data, ..., direction = "down")` 将前个或后个数据填充 ... 列中的缺失值 (默认自上而下填充)

`fill(x, x2)`

X

x1	x2
A	1
B	NA
C	NA
D	3
E	NA

→

x1	x2
A	1
B	2
C	2
D	3
E	2

`replace_na(data, replace)` 用特定值替换指定列中的所有缺失值 NA

`replace_na(x, list(x2 = 2))`

嵌套数据

嵌套数据框 (**nested data frame**) 将不同的列表嵌套储存为一个更大数据框的 list-column。List-columns 内也可以是一系列的向量或不同的数据类型。嵌套数据框可用于:

- 保留观察值与数据各子集之间的关系。保留嵌套变量的类型 (比如因子型和日期时间型, 不会被强制转为字符型)。
- 一次性处理多个子表格, 通过 **purrr** 包中的 `map()`, `map2()`, 和 `pmap()` 等函数, 或 **dplyr** 包中的 `rowwise()` 分组。

创建嵌套数据

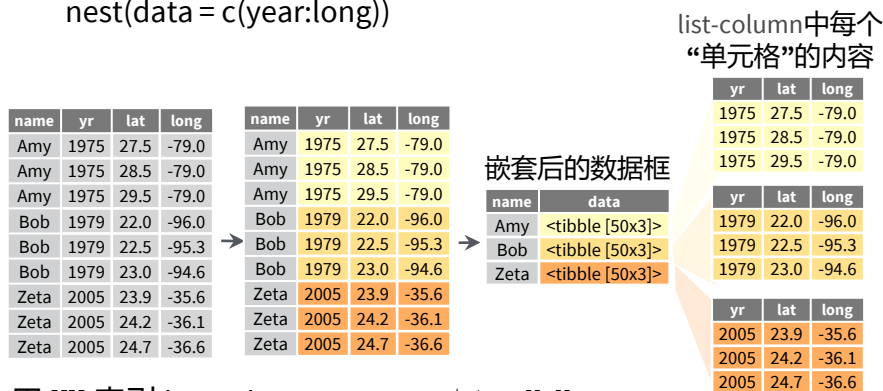
nest(data, ...)

将成组的单元格移至一个数据框的 list-column 中。可单独使用, 或与 **dplyr::group_by()** 函数配合使用。

- 先用 **group_by()** 分组, 然后用 **nest()** 将组移至 list-column 中。

```
n_storms <- storms %>%  
  group_by(name) %>%  
  nest()
```
- 用 **nest(new_col = c(x, y))** 指定需要分组的列
按照 **dplyr::select()** 语法

```
n_storms <- storms %>%  
  nest(data = c(year:long))
```



用 `[[]` 索引 list-column `n_storms$data[[1]]`

创建含有 list-columns 的 Tibbles

tibble::tribble(...) 按需要构造 list-columns

```
tribble( ~max, ~seq,  
         3, 1:3,  
         4, 1:4,  
         5, 1:5)
```

max	seq
3	<int [3]>
4	<int [4]>
5	<int [5]>

tibble::tibble(...) 把输入列表保存为 list-columns

```
tibble(max = c(3, 4, 5), seq = list(1:3, 1:4, 1:5))
```

tibble::enframe(x, name="name", value="value")

将多级列表转换为带有 list-columns 的 tibble

```
enframe(list('3'=1:3, '4'=1:4, '5'=1:5), 'max', 'seq')
```

通过其他函数输出 list-columns

dplyr::mutate(), **transmute()**, and **summarise()**

当这些函数返回列表时, 会输出 list-columns

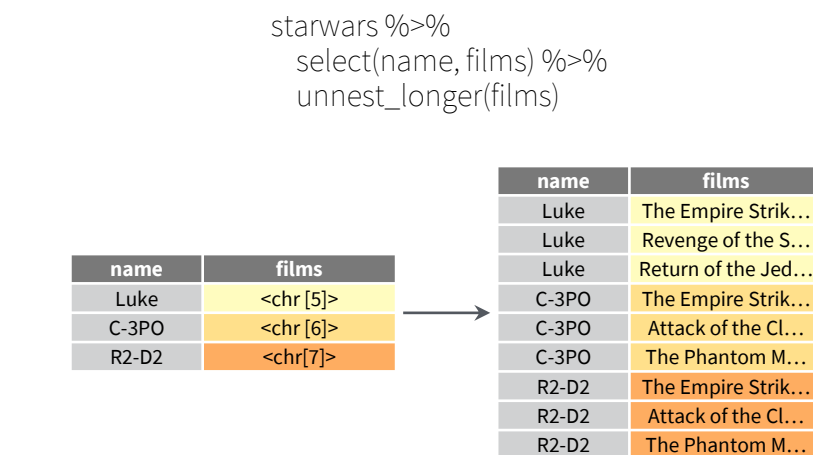
```
mtcars %>%  
  group_by(cyl) %>%  
  summarise(q = list(quantile(mpg)))
```

重组嵌套数据

unnest(data, cols, ..., keep_empty = FALSE)

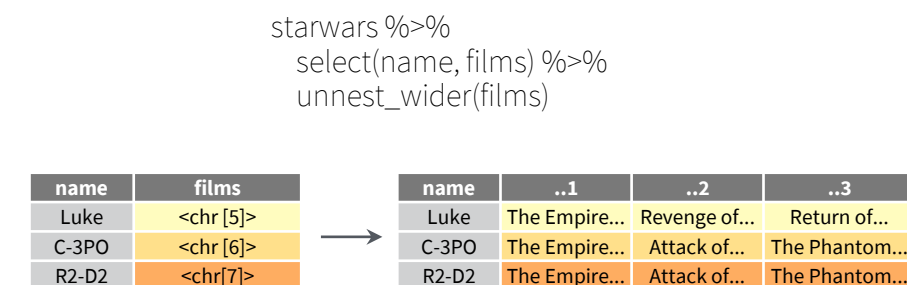
与 **nest()** 相反, 将嵌套的列重新展开为常规列,
`n_storms %>% unnest(data)`

unnest_longer(data, col, values_to = NULL, indices_to = NULL)
将一个 list-column 按行展开



unnest_wider(data, col)

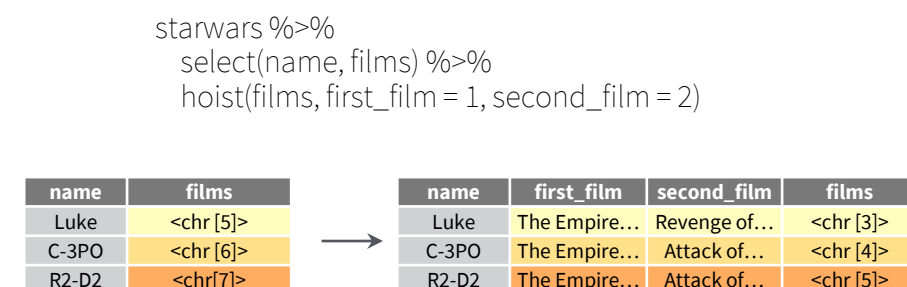
将一个 list-column 按列展开



hoist(.data, .col, ..., .remove = TRUE)

选择性地按列展开 list-column

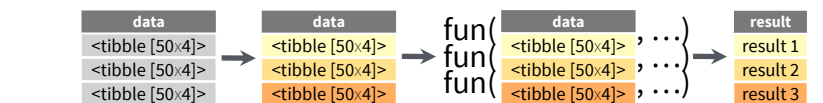
用 **purrr::pluck()** 语法选择 list-column 中需要展开的列



变换嵌套数据

一个向量化函数通常要取一个向量, 平行变换每一个元素, 然后返回一个相同长度的向量。因此向量函数本身无法应用于列表, 比如嵌套形成的 list-columns。

经 **dplyr::rowwise(.data, ...)** 这一函数分组处理后, 每行为一组, 每组中的 list-columns 元素直接出现 (通过 `[[]` 访问), 而非长度为1的列表。当你使用 **rowwise()** 函数时, **dplyr** 包中的函数就能以向量化处理的方式应用于 list-columns 了。



将一个函数应用到一个 list-column, 并创建一个新的 list-column。

```
n_storms %>%  
  rowwise() %>%  
  mutate(n = list(dim(data)))
```

dim() 每行返回两个值

加一层列表函数, 令 mutate 函数创建一个 list-column

将一个函数应用到一个 list-column, 并创建一个新的常规列。

```
n_storms %>%  
  rowwise() %>%  
  mutate(n = nrow(data))
```

nrow() 每行返回一个整型

将多个 list-columns 折叠为一个。

```
starwars %>%  
  rowwise() %>%  
  mutate(transport = list(append(vehicles, starships)))
```

append() 每行返回一个列表, 所以列的类型必须为列表类型

将一个函数用于多个 list-columns

```
starwars %>%  
  rowwise() %>%  
  mutate(n_transports = length(c(vehicles, starships)))
```

length() 每行返回一个整型

参见 **purrr** 包中的更多列表函数