

Tidying Data

tidyr

2019-08-16



Art by Allison Horst

tidyr

Functions for tidying data.

What is tidy data?

“Tidy datasets are all alike, but every messy dataset is messy in its own way.” -- Hadley Wickham



Tidy Data

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128062583

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128062583

observations

country	year	cases	population
Afghanistan	99	745	19987071
Afghanistan	00	2666	20095360
Brazil	99	37737	172006362
Brazil	00	80488	174504898
China	99	212258	1272915272
China	00	216766	128062583

values

Tidy Data

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128042583

variables

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20095360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128042583

observations

country	year	cases	population
Afghanistan	99	745	19987071
Afghanistan	00	2666	20095360
Brazil	99	37737	172006362
Brazil	00	80488	174504898
China	99	212258	1272915272
China	00	216766	128042583

values

Each **column** is a single **variable**

Tidy Data

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	216766	128042583

variables

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	216766	128042583

observations

country	year	cases	population
Afghanistan	99	745	19987071
Afghanistan	00	2666	20595360
Brazil	99	37737	172006362
Brazil	00	80488	174504898
China	99	212258	1272915272
China	00	216766	128042583

values

Each column is a single variable

Each **row** is a single **observation**

Tidy Data

country	year	cases	population
Afghanistan	1999	1845	19987071
Afghanistan	2000	2666	2059360
Brazil	1999	3737	17206362
Brazil	2000	8488	174504898
China	1999	21258	1272915272
China	2000	21766	128042583

variables

country	year	cases	population
Afghanistan	1999	1845	19987071
Afghanistan	2000	2666	2059360
Brazil	1999	3737	17206362
Brazil	2000	8488	174504898
China	1999	21258	1272915272
China	2000	21766	128042583

observations

country	year	cases	population
Afghanistan	99	75	987071
Afghanistan	00	66	59360
Brazil	99	737	006362
Brazil	00	488	504898
China	99	21258	1272915272
China	00	21766	128042583

values

Each column is a single variable

Each row is a single observation

Each **cell** is a **value**

gather()

```
gather(<DATA>, "<KEY>", "<VALUE>", <VARIABLES>)
```


Lord of the Rings

```
lotr <- tribble(  
  ~film, ~race, ~female, ~male,  
  "The Fellowship Of The Ring", "Elf", 1229L, 971L,  
  "The Fellowship Of The Ring", "Hobbit", 14L, 3644L,  
  "The Fellowship Of The Ring", "Man", 0L, 1995L,  
  "The Two Towers", "Elf", 331L, 513L,  
  "The Two Towers", "Hobbit", 0L, 2463L,  
  "The Two Towers", "Man", 401L, 3589L,  
  "The Return Of The King", "Elf", 183L, 510L,  
  "The Return Of The King", "Hobbit", 2L, 2673L,  
  "The Return Of The King", "Man", 268L, 2459L  
)
```

Lord of the Rings

```
lotr
```

```
## # A tibble: 9 x 4
```

##	film	race	female	male
##	<chr>	<chr>	<int>	<int>
## 1	The Fellowship Of The Ring	Elf	1229	971
## 2	The Fellowship Of The Ring	Hobbit	14	3644
## 3	The Fellowship Of The Ring	Man	0	1995
## 4	The Two Towers	Elf	331	513
## 5	The Two Towers	Hobbit	0	2463
## 6	The Two Towers	Man	401	3589
## 7	The Return Of The King	Elf	183	510
## 8	The Return Of The King	Hobbit	2	2673
## 9	The Return Of The King	Man	268	2459



new data alert!



lotr

	film	race	female	male
1	The Fellowship Of The Ring	Elf	1229	971
2	The Fellowship Of The Ring	Hobbit	14	3644
3	The Fellowship Of The Ring	Man	0	1995
4	The Two Towers	Elf	331	513
5	The Two Towers	Hobbit	0	2463
6	The Two Towers	Man	401	3589
7	The Return Of The King	Elf	183	510
8	The Return Of The King	Hobbit	2	2673
9	The Return Of The King	Man	268	2459

Where does it come from?

`exercises.Rmd`

source:

github.com/jennybc/lotr-tidyr

How can I use it?

Run the code at the top of
`exercises.Rmd`

`View(lotr)`



*this saves it in your
global environment*

gather()

```
lotr %>%  
  gather("sex", "words", female:male)
```

gather()

```
lotr %>%  
  gather("sex", "words", female:male)
```

```
## # A tibble: 18 x 4
```

##	film	race	sex	words
##	<chr>	<chr>	<chr>	<int>
##	1 The Fellowship Of The Ring	Elf	female	1229
##	2 The Fellowship Of The Ring	Hobbit	female	14
##	3 The Fellowship Of The Ring	Man	female	0
##	4 The Two Towers	Elf	female	331
##	5 The Two Towers	Hobbit	female	0
##	6 The Two Towers	Man	female	401
##	7 The Return Of The King	Elf	female	183
##	8 The Return Of The King	Hobbit	female	2
##	9 The Return Of The King	Man	female	268
##	10 The Fellowship Of The Ring	Elf	male	971
##	# ... with 8 more rows			



new data alert!



table2, table4a, who

	country	iso2	iso3	year	new_sp_m014
1	Afghanistan	AF	AFG	1980	N/A
2	Afghanistan	AF	AFG	1981	N/A
3	Afghanistan	AF	AFG	1982	N/A
4	Afghanistan	AF	AFG	1983	N/A
5	Afghanistan	AF	AFG	1984	N/A
6	Afghanistan	AF	AFG	1985	N/A
7	Afghanistan	AF	AFG	1986	N/A
8	Afghanistan	AF	AFG	1987	N/A

	country	1999	2000
1	Afghanistan	745	2666
2	Brazil	37737	80488
3	China	212258	213766

	country	year	type	count
1	Afghanistan	1999	cases	745
2	Afghanistan	1999	population	19987071
3	Afghanistan	2000	cases	2666
4	Afghanistan	2000	population	20595360
5	Brazil	1999	cases	37737
6	Brazil	1999	population	172006362
7	Brazil	2000	cases	80488
8	Brazil	2000	population	174504898
9	China	1999	cases	212258
10	China	1999	population	1272915272
11	China	2000	cases	213766
12	China	2000	population	1280428583

Where does it come from?

The tidyr R package

How can I use it?

```
library(tidyr)
View(table2)
View(table4a)
View(who)
```



*they're
invisible!*

Your Turn 1

Use `gather()` to reorganize `table4a` into three columns: `country`, `year`, and `cases`.

Your Turn 1

Use `gather()` to reorganize `table4a` into three columns: **country, **year**, and **cases**.**

```
table4a %>%  
  gather("year", "cases", -country)
```

```
## # A tibble: 6 x 3  
##   country    year  cases  
##   <chr>      <chr> <int>  
## 1 Afghanistan 1999     745  
## 2 Brazil      1999   37737  
## 3 China       1999  212258  
## 4 Afghanistan 2000     2666  
## 5 Brazil      2000   80488  
## 6 China       2000  213766
```


spread()

```
spread(<DATA>, <KEY>, <VALUE>)
```

spread()

```
lotr %>%  
  gather("sex", "words", female:male) %>%  
  spread(race, words)
```

spread()

```
lotr %>%  
  gather("sex", "words", female:male) %>%  
  spread(race, words)
```

spread()

```
lotr %>%  
  gather("sex", "words", female:male) %>%  
  spread(race, words)
```

```
## # A tibble: 6 x 5
```

##	film	sex	Elf	Hobbit	Man
##	<chr>	<chr>	<int>	<int>	<int>
## 1	The Fellowship Of The Ring	female	1229	14	0
## 2	The Fellowship Of The Ring	male	971	3644	1995
## 3	The Return Of The King	female	183	2	268
## 4	The Return Of The King	male	510	2673	2459
## 5	The Two Towers	female	331	0	401
## 6	The Two Towers	male	513	2463	3589

Your Turn 2

Use `spread()` to reorganize `table2` into four columns: `country`, `year`, `cases`, and `population`.

Create a new variable called `prevalence` that divides `cases` by `population` multiplied by `100000`.

Pass the data frame to a `ggplot`. Make a scatter plot with `year` on the x axis and `prevalence` on the y axis. Set the color aesthetic (`aes()`) to `country`. Use `size = 2` for the points. Add a line geom.

```
table2
```

Your Turn 2

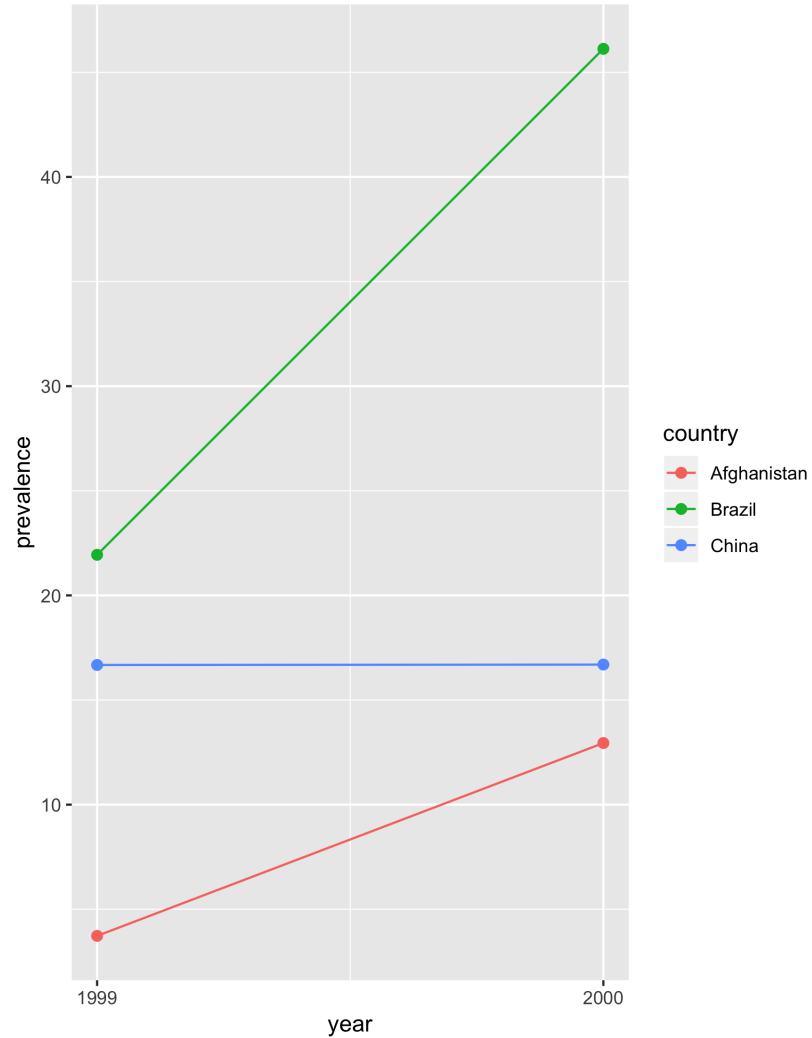
```
table2 %>%  
  spread(type, count) %>%  
  mutate(prevalence = (cases/population) * 100000)
```

```
## # A tibble: 6 x 5  
##   country      year  cases population prevalence  
##   <chr>      <int> <int>      <int>      <dbl>  
## 1 Afghanistan 1999     745   19987071      3.73  
## 2 Afghanistan 2000    2666   20595360     12.9  
## 3 Brazil      1999   37737   172006362     21.9  
## 4 Brazil      2000   80488   174504898     46.1  
## 5 China       1999  212258  1272915272     16.7  
## 6 China       2000  213766  1280428583     16.7
```

Your Turn 2

```
table2 %>%  
  spread(type, count) %>%  
  mutate(prevalence = (cases/population) * 100000) %>%  
  ggplot(aes(x = year, y = prevalence, color = country)) +  
    geom_point(size = 2) +  
    geom_line() +  
    scale_x_continuous(breaks = c(1999L, 2000L))
```

Your Turn 2



gather() and spread()

wide

id	wide		
	x	y	z
1	a	c	e
2	b	d	f

Your Turn 3

Gather the 5th through 60th columns of who into a key column: value column pair named `codes` and `n`. Then select just the county, year, codes and n variables.

```
who
```

Your Turn 3

```
who %>%  
gather("codes", "n", 5:60) %>%  
select(country, year, codes, n)
```

Your Turn 3

```
who %>%  
  gather("codes", "n", 5:60) %>%  
  select(country, year, codes, n)
```

```
## # A tibble: 405,440 x 4  
##   country      year codes      n  
##   <chr>      <int> <chr>    <int>  
## 1 Afghanistan  1980 new_sp_m014 NA  
## 2 Afghanistan  1981 new_sp_m014 NA  
## 3 Afghanistan  1982 new_sp_m014 NA  
## 4 Afghanistan  1983 new_sp_m014 NA  
## 5 Afghanistan  1984 new_sp_m014 NA  
## 6 Afghanistan  1985 new_sp_m014 NA  
## 7 Afghanistan  1986 new_sp_m014 NA  
## 8 Afghanistan  1987 new_sp_m014 NA  
## 9 Afghanistan  1988 new_sp_m014 NA  
## 10 Afghanistan 1989 new_sp_m014 NA  
## # ... with 405,430 more rows
```

separate()/unite()

```
separate(<DATA>, <VARIABLE>, into = c("<VARIABLE1>", "<VARIABLE2>"))  
unite(<DATA>, <VARIABLES>)
```

Your Turn 4

Use the cases data below. Separate the sex_age column into **sex** and **age** columns.

```
cases <- tribble(
  ~id,      ~sex_age,
  "1",      "male_56",
  "2",      "female_77",
  "3",      "female_49"
)
separate(_____, _____, into = c("_____", "_____"))
```

Your Turn 4

```
cases <- tribble(  
  ~id,      ~sex_age,  
  "1",      "male_56",  
  "2",      "female_77",  
  "3",      "female_49"  
)  
separate(cases, sex_age, into = c("sex", "age"))
```

Your Turn 4

```
cases <- tribble(
  ~id,      ~sex_age,
  "1",      "male_56",
  "2",      "female_77",
  "3",      "female_49"
)
separate(cases, sex_age, into = c("sex", "age"))
```

```
## # A tibble: 3 x 3
##   id    sex    age
##   <chr> <chr>  <chr>
## 1 1    male    56
## 2 2    female  77
## 3 3    female  49
```


Your Turn 4

```
cases <- tribble(
  ~id,      ~sex_age,
  "1",      "male_56",
  "2",      "female_77",
  "3",      "female_49"
)
separate(cases, sex_age, into = c("sex", "age"))
```

```
## # A tibble: 3 x 3
##   id    sex    age
##   <chr> <chr> <chr>
## 1 1    male   56
## 2 2    female 77
## 3 3    female 49
```

Your Turn 4

```
cases <- tribble(
  ~id,      ~sex_age,
  "1",      "male_56",
  "2",      "female_77",
  "3",      "female_49"
)
separate(cases, sex_age, into = c("sex", "age"), convert = TRUE)
```

```
## # A tibble: 3 x 3
##   id    sex    age
##   <chr> <chr> <int>
## 1 1    male    56
## 2 2    female  77
## 3 3    female  49
```

Your Turn 5: Challenge!

There are two CSV files in this folder containing SEER data in breast cancer incidence in white and black women. For both sets of data:

Import the data

Gather the columns into 2 new columns called `year` and `incidence`

Add a new variable called `race`. Remember that each data set corresponds to a single race.

Bind the data sets together using `bind_rows()` from the `dplyr` package. Either save it as a new object or pipe the result directly into the `ggplot2` code.

Plot the data using the code below. Fill in the blanks to have `year` on the x-axis, `incidence` on the y-axis, and `race` as the color aesthetic.

Other neat tidyr tools

Uncounting frequency tables

```
lotr %>%  
  gather("sex", "count", female, male) %>%  
  uncount(count)
```

Other neat tidyr tools

Uncounting frequency tables

```
## # A tibble: 21,245 x 3
##   film                                race sex
##   <chr>                                <chr> <chr>
## 1 The Fellowship Of The Ring Elf      female
## 2 The Fellowship Of The Ring Elf      female
## 3 The Fellowship Of The Ring Elf      female
## 4 The Fellowship Of The Ring Elf      female
## 5 The Fellowship Of The Ring Elf      female
## 6 The Fellowship Of The Ring Elf      female
## 7 The Fellowship Of The Ring Elf      female
## 8 The Fellowship Of The Ring Elf      female
## 9 The Fellowship Of The Ring Elf      female
## 10 The Fellowship Of The Ring Elf      female
## # ... with 21,235 more rows
```

Other neat tidyr tools

Work with data frames

`crossing()` and `expand()`

`nest()` and `unnest()`


Other neat tidyr tools

Work with missing data

`complete()`

`drop_na()` and `replace_na()`

In development: `pivot_longer()`, `pivot_wider()`

 **tidyr** part of the tidyverse
0.8.99.9000

Tidy dataReferenceArticles▼Changelog

Pivoting

Source: `vignettes/pivot.Rmd`

Introduction

This vignette describes the use of the new `pivot_longer()` and `pivot_wider()` functions. Their goal is to improve the usability of `gather()` and `spread()`, and incorporate state-of-the-art features found in other packages.

For some time, it's been obvious that there is something fundamentally wrong with the design of `spread()` and `gather()`. Many people don't find the names intuitive and find it hard to remember which direction corresponds to spreading and which to gathering. It also seems surprisingly hard to remember the arguments to these functions, meaning that many people (including me!) have to consult the documentation every time.

There are two important new features inspired by other R packages that have been advancing reshaping in R:

- `pivot_longer()` can work with multiple value variables that may have different types, inspired by the enhanced `melt()` and `dcast()` functions provided by the `data.table` package by Matt Dowle and Arun Srinivasan.
- `pivot_longer()` and `pivot_wider()` can take a data frame that specifies precisely how metadata stored in column names becomes data variables (and vice versa), inspired by the `cdata` package by John Mount and Nina Zumel.

In this vignette, you'll learn the key ideas behind `pivot_longer()` and `pivot_wider()` as you see them used to solve a variety of data reshaping challenges ranging from simple to complex.

To begin we'll load some needed packages. In real analysis code, I'd imagine you'd do with the `library(tidyverse)`, but I can't do that here since this vignette is embedded in a package.

Contents

- Introduction
- Longer
 - String data in column names
 - Numeric data in column names
 - Many variables in column names
 - Multiple observations per row
 - Duplicated column names
- Wider
 - Capture-recapture data
 - Aggregation
 - Generate column name from multiple variables
 - Tidy census
 - Contact list
- Longer, then wider
 - World bank
 - Multi-choice
- Manual specs
 - Longer
 - Wider
 - By hand
 - Theory

<https://tidyr.tidyverse.org/dev/>

Resources

R for Data Science: A comprehensive but friendly introduction to the tidyverse. Free online.

RStudio Primers: Free interactive courses in the Tidyverse