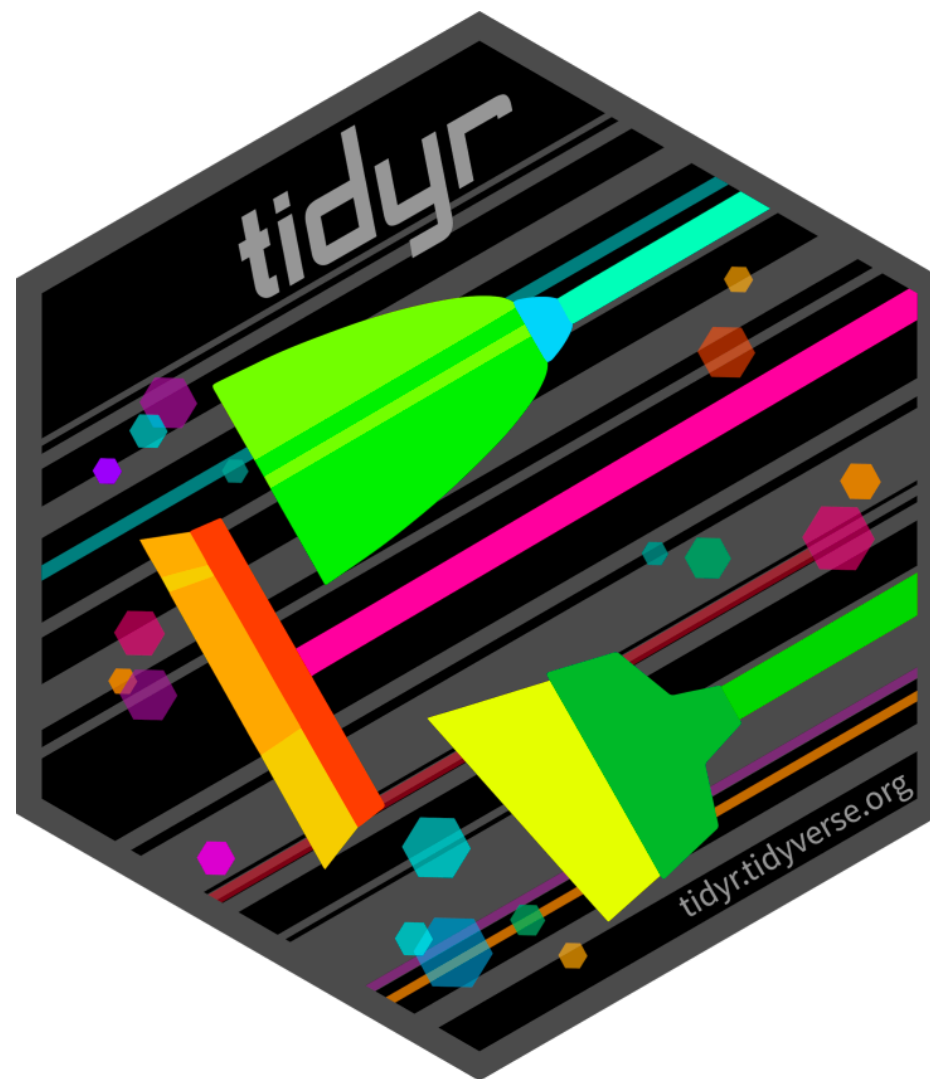


Tidy data with tidyr (basics)



Outline

The tidyr package and tidy data review

Functions

Important ones: `pivot_wider()`, `pivot_longer()`, and friends

Others that might be of some use!

Data tidying with tidyr :: CHEAT SHEET



Tidy data is a way to organize tabular data in a consistent data structure across packages.

A table is tidy if:



Each **variable** is in its own **column**

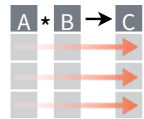
&



Each **observation**, or **case**, is in its own row



Access **variables** as **vectors**



Preserve **cases** in vectorized operations

Tibbles

AN ENHANCED DATA FRAME

Tibbles are a table format provided by the **tibble** package. They inherit the data frame class, but have improved behaviors:

- **Subset** a new tibble with `[],` a vector with `[[` and `$.`
- **No partial matching** when subsetting columns.
- **Display** concise views of the data on one screen.

options(`tibble.print_max = n, tibble.print_min = m, tibble.width = Inf`) Control default display settings.

View() or **glimpse()** View the entire data set.

CONSTRUCT A TIBBLE

tibble(...) Construct by columns.

`tibble(x = 1:3, y = c("a", "b", "c"))`

tribble(...) Construct by rows.

`tribble(~x, ~y,`
1, "a",
2, "b",
3, "c")

Both make this tibble

A tibble: 3 × 2
 <int> <chr>
1 1 a
2 2 b
3 3 c

as_tibble(x, ...) Convert a data frame to a tibble.

enframe(x, name = "name", value = "value")
Convert a named vector to a tibble. Also **deframe()**.

is_tibble(x) Test whether x is a tibble.

Reshape Data - Pivot data to reorganize values into a new layout.

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)

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

`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")

The inverse of pivot_longer(). "Widen" data by expanding two columns into several. One column provides the new column names, the other the values.

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

Split Cells - Use these functions to split or combine cells into individual, isolated values.

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

unite(data, col, ..., sep = "_", remove = TRUE, na.rm = FALSE) Collapse cells across several columns into a single column.

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

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	172
B	2000	80K	174

separate(data, col, into, sep = "[^:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn", ...) Separate each cell in a column into several columns. Also **extract()**.

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

separate_rows(data, ..., sep = "[^:alnum:].]+", convert = FALSE) Separate each cell in a column into several rows.

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

Expand Tables

Create new combinations of variables or identify implicit missing values (combinations of variables not present in the data).

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, ...) Create a new tibble with all possible combinations of the values of the variables listed in ... Drop other variables.
`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()) Add missing possible combinations of values of variables listed in ... Fill remaining variables with NA.
`complete(mtcars, cyl, gear, carb)`

Handle Missing Values

Drop or replace explicit missing values (NA).

x

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

x1	x2
A	1
D	3

drop_na(data, ...) Drop rows containing NA's in ... columns.
`drop_na(x, x2)`

x

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

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

fill(data, ..., .direction = "down") Fill in NA's in ... columns using the next or previous value.
`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) Specify a value to replace NA in selected columns.
`replace_na(x, list(x2 = 2))`

tidyr

The *tidyr* package is a part of the *tidyverse* and is the main function for tidying data

Tidy data principles:

1. Every column is a variable
2. Every row is an observation
3. Every cell is a single value

There are five main actions/categories that *tidyr* addresses

1. Pivoting
2. Rectangling
3. Nesting
4. Splitting
5. Implicit/Explicit

Pivotting Data

There are many circumstances in which you need to change the shape of the data

I've encountered it most often in plotting, but it happens elsewhere!

Example:

name	hw1	hw2	hw3	test1	test2
"John"	60	89	93	85	89
"Mary"	89	93	75	90	82
"Ben"	76	98	83	87	76
"Steph"	88	81	87	90	95

I want to plot how the students did over time/assignment with one point per student. How do I do that?

pivot_longer()

`pivot_longer()` takes data in a “wide” format and gathers the data into a long format.

Now the third iteration of the same function: `melt()`, `gather()`, `pivot_longer()`

pivot_longer()

```
pivot_longer(  
  data,
```

```
  cols,
```

```
  names_to = "name",
```

```
  names_prefix = NULL,
```

```
  names_sep = NULL,
```

```
  names_pattern = NULL,
```

```
  names_ptypes = list(),
```

```
  names_repair = "check_unique",
```

```
  values_to = "value",
```

```
  values_drop_na = FALSE,
```

```
  values_ptypes = list()  
)
```

The data to be pivoted

The columns to pivot
into longer format

Name of the column
where variable names go

Name of the column
where values go

pivot_longer()

```
tb %>%
```

```
  pivot_longer(-name,
```

```
    names_to = "assignment",
```

```
    values_to = "grade")
```

tb

name	hw1	hw2	hw3	test1	test2
"John"	60	89	93	85	89
"Mary"	89	93	75	90	82
"Ben"	76	98	83	87	76
"Steph"	88	81	87	90	95

```
# A tibble: 20 x 3
```

	name	assignment	grade
	<chr>	<chr>	<dbl>
1	John	hw1	60
2	John	hw2	89
3	John	hw3	93
4	John	test1	85
5	John	test2	89
6	Mary	hw1	89
7	Mary	hw2	93
8	Mary	hw3	75
9	Mary	test1	90
10	Mary	test2	82
11	Ben	hw1	76
12	Ben	hw2	98
13	Ben	hw3	83
14	Ben	test1	87
15	Ben	test2	76
16	Steph	hw1	88
17	Steph	hw2	81
18	Steph	hw3	87
19	Steph	test1	90
20	Steph	test2	95

pivot_wider()

`pivot_wider()` takes data in a “long” format and spreads the data into a wide format.

Now the third iteration of the same function: `dcast()`, `spread()`, `pivot_wider()`

It is the inverse of `pivot_longer()`

To see how it works, let's take the data back to its original form!

pivot_wider()

```
tb2 %>%
```

```
tb2
```

```
  pivot_wider(names_from = assignment,  
              values_from = grade)
```

```
# A tibble: 20 x 3
```

	name <chr>	assignment <chr>	grade <dbl>
1	John	hw1	60
2	John	hw2	89
3	John	hw3	93
4	John	test1	85
5	John	test2	89
6	Mary	hw1	89
7	Mary	hw2	93
8	Mary	hw3	75
9	Mary	test1	90
10	Mary	test2	82
11	Ben	hw1	76
12	Ben	hw2	98
13	Ben	hw3	83
14	Ben	test1	87
15	Ben	test2	76
16	Steph	hw1	88
17	Steph	hw2	81
18	Steph	hw3	87
19	Steph	test1	90
20	Steph	test2	95

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

	name <chr>	hw1 <dbl>	hw2 <dbl>	hw3 <dbl>	test1 <dbl>	test2 <dbl>
1	John	60	89	93	85	89
2	Mary	89	93	75	90	82
3	Ben	76	98	83	87	76
4	Steph	88	81	87	90	95

hoist() and unnest() variants

hoist(), unnest_longer(), and unnest_wider() are variants that provide tools for rectangling or collapsing deeply nested lists into tidy tibbles

Examples to come!

More Examples!

Let's look at other specifications and datasets [here](#)