

Tidy data

Data Wrangling, Session 4

Kieran Healy

Code Horizons

January 2026

Tidy data with `tidyverse`

Load the packages, as always

```
library(here)      # manage file paths
library(socviz)    # data and some useful functions

library(tidyverse) # your friend and mine
library(gapminder) # gapminder data

## Quietens dplyr summarise chatter (with an 's')!
options(dplyr.summarise.inform = FALSE)
```

Tidy data
is data in
long format

The Tidyverse wants to be fed **tidy data**



Get your data into long format

Very, very often, the solution to some data-wrangling problem in Tidyverse-focused workflow is:

Get your data into long format

Very, very often, the solution to some data-wrangling problem in Tidyverse-focused workflow is:

Get the data in long format

Get your data into long format

Very, very often, the solution to some data-wrangling problem in Tidyverse-focused workflow is:

Get the data in long format

Then do the recoding thing that you want.

Get your data into long format

Very, very often, the solution to some data-wrangling problem in Tidyverse-focused workflow is:

Get the data in long format

Then do the recoding thing that you want.

Then transform it back to something wider if needed.

This isn't an *iron* rule

As we'll see later, `dplyr` is able to do “rowwise” operations if you need them.

It is a
pretty good
rule though

Tidy data

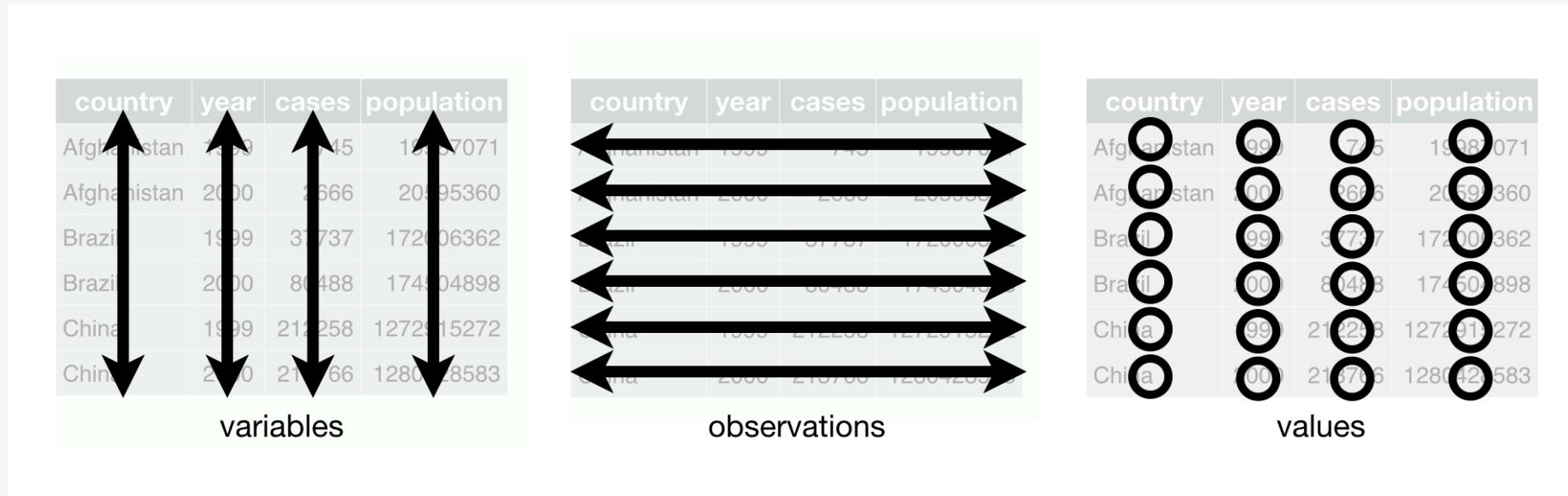
```
gapminder
```

```
# A tibble: 1,704 × 6
  country      continent  year lifeExp      pop gdpPercap
  <fct>        <fct>    <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia      1952    28.8  8425333     779.
2 Afghanistan Asia      1957    30.3  9240934     821.
3 Afghanistan Asia      1962    32.0 10267083     853.
4 Afghanistan Asia      1967    34.0 11537966     836.
5 Afghanistan Asia      1972    36.1 13079460     740.
6 Afghanistan Asia      1977    38.4 14880372     786.
7 Afghanistan Asia      1982    39.9 12881816     978.
8 Afghanistan Asia      1987    40.8 13867957     852.
9 Afghanistan Asia      1992    41.7 16317921     649.
10 Afghanistan Asia     1997    41.8 22227415     635.
# i 1,694 more rows
```

Tidy data

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

Tidy data



Tidy data

Tidy data

Each variable has its own column.

Tidy data

Each variable has its own column.

Each observation has its own row.

Tidy data

Each variable has its own column.

Each observation has its own row.

Each value has its own cell.

Tidy data

Each variable has its own column.

Each observation has its own row.

Each value has its own cell.

When data is tidy in this way, the vectorized character of R's way of doing things works best.

Untidy data: common for good reasons

Table A-1. Years of School Completed by People 25 Years and Over, by Age and Sex: Selected Years 1940 to 2016

(Numbers in thousands. Noninstitutionalized population except where otherwise specified.)

Age, sex, and years	Total	Years of School Completed						
		Elementary		High school		College		Median
		0 to 4 years	5 to 8 years	1 to 3 years	4 years	1 to 3 years	4 years or more	

25 YEARS AND OLDER

Male

2016	103,372	1,183	3,513	7,144	30,780	26,468	34,283	(NA)
2015	101,887	1,243	3,669	7,278	30,997	25,778	32,923	(NA)
2014	100,592	1,184	3,761	7,403	30,718	25,430	32,095	(NA)
2013	99,305	1,127	3,836	7,314	30,014	25,283	31,731	(NA)
2012	98,119	1,237	3,879	7,388	30,216	24,632	30,766	(NA)
2011	97,220	1,234	3,883	7,443	30,370	24,319	29,971	(NA)
2010	96,325	1,279	3,931	7,705	30,682	23,570	29,158	(NA)
2009	95,518	1,372	4,027	7,754	30,025	23,634	28,706	(NA)
2008	94,470	1,310	4,136	7,853	29,491	23,247	28,433	(NA)
2007	93,421	1,458	4,249	8,294	29,604	22,219	27,596	(NA)
2006	92,233	1,472	4,395	7,940	29,380	22,136	26,910	(NA)
2005	90,899	1,505	4,402	7,787	29,151	21,794	26,259	(NA)

Untidy data: common for good reasons

Storing data in long form is often *inefficient*

```
library(covdata)
covus %>
  filter(state == "NY") %>
  select(date:fips, measure:count)

# A tibble: 11,872 × 5
  date      state fips  measure           count
  <date>    <chr> <dbl> <chr>            <dbl>
1 2021-03-07 NY     36   positive        1681169
2 2021-03-07 NY     36   probable_cases    NA
3 2021-03-07 NY     36   negative         NA
4 2021-03-07 NY     36   pending          NA
5 2021-03-07 NY     36   hospitalized_currently 4789
6 2021-03-07 NY     36   hospitalized_cumulative NA
7 2021-03-07 NY     36   in_icu_currently   999
8 2021-03-07 NY     36   in_icu_cumulative  NA
9 2021-03-07 NY     36   on_ventilator_currently 682
10 2021-03-07 NY    36   on_ventilator_cumulative NA
# i 11,862 more rows
```

Untidy data: common for good reasons

Storing data in wide form is *easier to display* in a printed table

```
library(palmerpenguins)
penguins %>
  group_by(species, island, year) %>
  summarize(bill = round(mean(bill_length_mm, na.rm = TRUE), 2)) %>
  tinytable::tt()
```

species	island	year	bill
Adelie	Biscoe	2007	38.32
Adelie	Biscoe	2008	38.70
Adelie	Biscoe	2009	39.69
Adelie	Dream	2007	39.10
Adelie	Dream	2008	38.19
Adelie	Dream	2009	38.15
Adelie	Torgersen	2007	38.80
Adelie	Torgersen	2008	38.77
Adelie	Torgersen	2009	39.31
Chinstrap	Dream	2007	48.72
Chinstrap	Dream	2008	48.70
Chinstrap	Dream	2009	49.05
Gentoo	Biscoe	2007	47.01
Gentoo	Biscoe	2008	46.94
Gentoo	Biscoe	2009	48.50

Untidy data: common for good reasons

Storing data in wide form is *easier to display* in a printed table

```
penguins %>%  
  group_by(species, island, year) %>%  
  summarize(bill = round(mean(bill_length_mm, na.rm = TRUE), 2)) %>%  
  pivot_wider(names_from = year, values_from = bill) %>%  
  tinytable::tt()
```

species	island	2007	2008	2009
Adelie	Biscoe	38.32	38.70	39.69
Adelie	Dream	39.10	38.19	38.15
Adelie	Torgersen	38.80	38.77	39.31
Chinstrap	Dream	48.72	48.70	49.05
Gentoo	Biscoe	47.01	46.94	48.50

Again, these tables are made directly in R with the code you see here.

It's also common for *less* good reasons

State																		
A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	Q			
State	CD#	2018 Cook PVI Score	2018 Winner	Party	Dem Votes	GOP Votes	Other Votes	Dem %	GOP %	Other %	Dem Margin	2016 Clinton Margin	Swing vs. 2016 Prez	Raw Votes vs. 2016	Final?			
New House Breakdown: 235D, 199R, 1 Not Certified																		
Compiled by: David Wasserman & Ally Flinn, Cook Political Report. @Redistrict/@CookPolitical. <i>Italics</i> denotes freshman, Bold denotes party change.																		
Alabama	1	R+15	Bradley Byrne	R	89,226	153,228	163	36.8%	63.2%	0.1%	-26.4%	-29.2%	2.8%	79.3%	x			
Alabama	2	R+16	Martha Roby	R	86,931	138,879	420	38.4%	61.4%	0.2%	-23.0%	-31.7%	8.7%	78.7%	x			
Alabama	3	R+16	Mike Rogers	R	83,996	147,770	149	36.2%	63.7%	0.1%	-27.5%	-33.0%	5.5%	79.6%	x			
Alabama	4	R+30	Robert Aderholt	R	46,492	184,255	222	20.1%	79.8%	0.1%	-59.6%	-62.5%	2.9%	78.9%	x			
Alabama	5	R+18	Mo Brooks	R	101,388	159,063	222	38.9%	61.0%	0.1%	-22.1%	-32.9%	10.8%	82.8%	x			
Alabama	6	R+26	Gary Palmer	R	85,644	192,542	142	30.8%	69.2%	0.1%	-38.4%	-43.8%	5.4%	82.8%	x			
Alabama	7	D+20	Terri Sewell	D	185,010	0	4,153	97.8%	0.0%	2.2%	97.8%	41.2%	N/A	64.2%	x			
Alaska	AL	R+9	Don Young	R	131,199	149,779	1,188	46.5%	53.1%	0.4%	-6.6%	-14.7%	8.1%	88.6%	x			
Arizona	1	R+2	Tom O'Halleran	D	143,240	122,784	65	53.8%	46.1%	0.0%	7.7%	-1.1%	8.8%	92.0%	x			
Arizona	2	R+1	<i>Ann Kirkpatrick</i>	D	161,000	133,102	50	54.7%	45.2%	0.0%	9.5%	4.8%	4.7%	91.5%	x			
Arizona	3	D+13	Raul Grijalva	D	114,650	64,868	0	63.9%	36.1%	0.0%	27.7%	29.5%	-1.8%	84.8%	x			
Arizona	4	R+21	Paul Gosar	R	84,521	188,842	3,672	30.5%	68.2%	1.3%	-37.7%	-39.4%	1.7%	91.1%	x			
Arizona	5	R+15	Andy Biggs	R	127,027	186,037	0	40.6%	59.4%	0.0%	-18.8%	-20.5%	1.7%	91.7%	x			
Arizona	6	R+9	David Schweikert	R	140,559	173,140	0	44.8%	55.2%	0.0%	-10.4%	-9.8%	-0.6%	91.2%	x			
Arizona	7	D+23	Ruben Gallego	D	113,044	301	18,706	85.6%	0.2%	14.2%	85.4%	48.3%	N/A	79.0%	x			
Arizona	8	R+13	Debbie Lesko	R	135,569	168,835	13	44.5%	55.5%	0.0%	-10.9%	-20.8%	9.9%	91.5%	x			
Arizona	9	D+4	<i>Greg Stanton</i>	D	159,583	101,662	0	61.1%	38.9%	0.0%	22.2%	15.9%	6.3%	90.0%	x			
Arkansas	1	R+17	Rick Crawford	R	57,907	138,757	4,581	28.8%	68.9%	2.3%	-40.2%	-34.8%	-5.4%	77.2%	x			
Arkansas	2	R+7	French Hill	R	116,135	132,125	5,193	45.8%	52.1%	2.0%	-6.3%	-10.7%	4.4%	82.6%	x			
Arkansas	3	R+19	Steve Womack	R	74,952	148,717	6,039	32.6%	64.7%	2.6%	-32.1%	-31.4%	-0.7%	78.6%	x			
Arkansas	4	R+17	Bruce Westerman	R	63,984	136,740	4,168	31.2%	66.7%	2.0%	-35.5%	-32.8%	-2.7%	75.7%	x			
California	1	R+11	Doug LaMalfa	R	131,506	160,006	0	45.1%	54.9%	0.0%	-9.8%	-19.4%	9.6%	91.6%				
California	2	D+22	Jared Huffman	D	243,051	72,541	0	77.0%	23.0%	0.0%	54.0%	45.2%	8.8%	90.5%				
California	3	D+5	John Garamendi	D	132,983	96,106	0	58.0%	42.0%	0.0%	16.1%	12.5%	3.6%	86.8%				
California	4	R+10	Tom McClintock	R	156,253	184,401	0	45.9%	54.1%	0.0%	-8.3%	-14.5%	6.2%	94.6%				
California	5	D+21	Mike Thompson	D	203,012	0	53,836	79.0%	0.0%	21.0%	79.0%	44.6%	N/A	83.8%				
California	6	D+21	Doris Matsui	D	201,939	0	0	100.0%	0.0%	0.0%	100.0%	44.0%	N/A	81.4%				
California	7	D+3	Ami Bera	D	155,016	126,601	0	55.0%	45.0%	0.0%	10.1%	11.2%	-1.1%	91.0%				
California	8	R+9	Paul Cook	R	0	170,785	0	0.0%	100.0%	0.0%	-100.0%	-15.1%	N/A	73.3%				
California	9	D+8	Jerry McNerney	D	113,240	87,263	0	56.5%	43.5%	0.0%	13.0%	18.2%	-5.2%	82.4%				
California	10	EVEN	Ed Perlmutter	D	115,000	105,000	0	50.0%	47.0%	3.0%	1.0%	0.0%	1.0%	51.0%				

It's also common for *less* good reasons

State	CD#	2018 Cook PVI Score	2018 Winner	Party	Dem Votes	GOP Votes	Other Votes	Dem %	GOP %	Other %	Dem Margin	2016 Clinton Margin	Swing vs. 2016 Prez	Raw Votes vs. 2016	Final?
				D	60,619,428	50,896,244	1,978,795	53.4%	44.8%	1.7%	8.6%	2.1%	6.5%	83.3%	
New House Breakdown: 235D, 199R, 1 Not Certified															
Compiled by: David Wasserman & Ally Flinn, Cook Political Report. @Redistrict@CookPolitical. <i>Italics</i> denotes freshman, Bold denotes party change.															
Alabama	1	R+15	Bradley Byrne	R	89,226	153,228	163	36.8%	63.2%	0.1%	-26.4%	-29.2%	2.8%	79.3%	x
Alabama	2	R+16	<i>Martha Roby</i>	R	86,931	138,879	420	38.4%	61.4%	0.2%	-23.0%	-31.7%	8.7%	78.7%	x
Alabama	3	R+16	<i>Mike Rogers</i>	R	83,996	147,770	149	36.2%	63.7%	0.1%	-27.5%	-33.0%	5.5%	79.6%	x
Alabama	4	R+30	Robert Aderholt	R	46,492	184,255	222	20.1%	79.8%	0.1%	-59.6%	-62.5%	2.9%	78.9%	x
Alabama	5	R+18	<i>Mo Brooks</i>	R	101,388	159,063	222	38.9%	61.0%	0.1%	-22.1%	-32.9%	10.8%	82.8%	x
Alabama	6	R+26	Gary Palmer	R	85,644	192,542	142	30.8%	69.2%	0.1%	-38.4%	-43.8%	5.4%	82.8%	x
Alabama	7	D+20	<i>Terri Sewell</i>	D	185,010	0	4,153	97.8%	0.0%	2.2%	97.8%	41.2%	N/A	64.2%	x
Alaska	AL	R+9	<i>Don Young</i>	R	131,199	149,779	1,188	46.5%	53.1%	0.4%	-6.6%	-14.7%	8.1%	88.6%	x
Arizona	1	R+2	<i>Tom O'Halleran</i>	D	143,240	122,784	65	53.8%	46.1%	0.0%	7.7%	-1.1%	8.8%	92.0%	x
Arizona	2	R+1	<i>Ann Kirkpatrick</i>	D	161,000	133,102	50	54.7%	45.2%	0.0%	9.5%	4.8%	4.7%	91.5%	x
Arizona	3	D+13	<i>Raul Grijalva</i>	D	114,650	64,868	0	63.9%	36.1%	0.0%	27.7%	29.5%	-1.8%	84.8%	x
Arizona	4	R+21	<i>Paul Gosar</i>	R	84,521	188,842	3,672	30.5%	68.2%	1.3%	-37.7%	-39.4%	1.7%	91.1%	x
Arizona	5	R+15	<i>Andy Biggs</i>	R	127,027	186,037	0	40.6%	59.4%	0.0%	-18.8%	-20.5%	1.7%	91.7%	x
Arizona	6	R+9	<i>David Schweikert</i>	R	140,559	173,140	0	44.8%	55.2%	0.0%	-10.4%	-9.8%	-0.6%	91.2%	x
Arizona	7	D+23	<i>Ruben Gallego</i>	D	113,044	301	18,706	85.6%	0.2%	14.2%	85.4%	48.3%	N/A	79.0%	x
Arizona	8	R+13	<i>Debbie Lesko</i>	R	135,569	168,835	13	44.5%	55.5%	0.0%	-10.9%	-20.8%	9.9%	91.5%	x
Arizona	9	D+4	<i>Greg Stanton</i>	D	159,583	101,662	0	61.1%	38.9%	0.0%	22.2%	15.9%	6.3%	90.0%	x
Arkansas	1	R+17	<i>Rick Crawford</i>	R	57,907	138,757	4,581	28.8%	68.9%	2.3%	-40.2%	-34.8%	-5.4%	77.2%	x
Arkansas	2	R+7	<i>French Hill</i>	R	116,135	132,125	5,193	45.8%	52.1%	2.0%	-6.3%	-10.7%	4.4%	82.6%	x
Arkansas	3	R+19	<i>Steve Womack</i>	R	74,952	148,717	6,039	32.6%	64.7%	2.6%	-32.1%	-31.4%	-0.7%	78.6%	x
Arkansas	4	R+17	<i>Bruce Westerman</i>	R	63,984	136,740	4,168	31.2%	66.7%	2.0%	-35.5%	-32.8%	-2.7%	75.7%	x
California	1	R+11	<i>Doug LaMalfa</i>	R	131,506	160,006	0	45.1%	54.9%	0.0%	-9.8%	-19.4%	9.6%	91.6%	
California	2	D+22	<i>Jared Huffman</i>	D	243,051	72,541	0	77.0%	23.0%	0.0%	54.0%	45.2%	8.8%	90.5%	
California	3	D+5	<i>John Garamendi</i>	D	132,983	96,106	0	58.0%	42.0%	0.0%	16.1%	12.5%	3.6%	86.8%	
California	4	R+10	<i>Tom McClintock</i>	R	156,253	184,401	0	45.9%	54.1%	0.0%	-8.3%	-14.5%	6.2%	94.6%	
California	5	D+21	<i>Mike Thompson</i>	D	203,012	0	53,836	79.0%	0.0%	21.0%	79.0%	44.6%	N/A	83.8%	
California	6	D+21	<i>Doris Matsui</i>	D	201,939	0	0	100.0%	0.0%	0.0%	100.0%	44.0%	N/A	81.4%	
California	7	D+3	<i>Ami Bera</i>	D	155,016	126,601	0	55.0%	45.0%	0.0%	10.1%	11.2%	-1.1%	91.0%	
California	8	R+9	<i>Paul Cook</i>	R	0	170,785	0	0.0%	100.0%	0.0%	-100.0%	-15.1%	N/A	73.3%	
California	9	D+8	<i>Jerry McNerney</i>	D	113,240	87,263	0	56.5%	43.5%	0.0%	13.0%	18.2%	-5.2%	82.4%	

More than one header row

Mixed data types in some columns

Color and typography used to encode variables and their values

Fix it **before** you import it

Prevention is better than cure!

Broman KW, Woo KH (2018) “Data organization in spreadsheets.” *The American Statistician* 78:2–10

THE AMERICAN STATISTICIAN
2018, VOL. 72, NO. 1, 2–10
<https://doi.org/10.1080/00031305.2017.1375989>



OPEN ACCESS Check for updates

Data Organization in Spreadsheets

Karl W. Broman^a and Kara H. Woo^b

^aDepartment of Biostatistics & Medical Informatics, University of Wisconsin-Madison, Madison, WI; ^bInformation School, University of Washington, Seattle, WA

ABSTRACT

Spreadsheets are widely used software tools for data entry, storage, analysis, and visualization. Focusing on the data entry and storage aspects, this article offers practical recommendations for organizing spreadsheet data to reduce errors and ease later analyses. The basic principles are: be consistent, write dates like YYYY-MM-DD, do not leave any cells empty, put just one thing in a cell, organize the data as a single rectangle (with subjects as rows and variables as columns, and with a single header row), create a data dictionary, do not include calculations in the raw data files, do not use font color or highlighting as data, choose good names for things, make backups, use data validation to avoid data entry errors, and save the data in plain text files.

ARTICLE HISTORY

Received June 2017
Revised August 2017

KEYWORDS

Data management; Data organization; Microsoft Excel; Spreadsheets

Key points from Broman & Woo

	A	B	C
1	Date	Assay date	Weight
2		12/9/05	54.9
3		12/9/05	45.3
4	12/6/2005	e	47
5		e	45.7
6		e	52.9
7		1/11/2006	46.1
8		1/11/2006	38.6

Figure 1. A spreadsheet with inconsistent date formats. This spreadsheet does not adhere to our recommendations for consistency of date format.

Use a consistent date format

ISO 8601

YYYY-MM-DD

The one true year-month-day format

Key points from Broman & Woo

A

	A	B	C
1	id	date	glucose
2	101	2015-06-14	149.3
3	102		95.3
4	103	2015-06-18	97.5
5	104		117.0
6	105		108.0
7	106	2015-06-20	149.0
8	107		169.4

B

	A	B	C	D	E	F	G	H	I
1		1 min				5 min			
2	strain	normal		mutant		normal		mutant	
3	A	147	139	166	179	334	354	451	474
4	B	246	240	178	172	514	611	412	447

No empty cells.

Use one row of headers only.

Key points from Broman & Woo

	A	B	C	D	E
1	strain	genotype	min	replicate	response
2	A	normal	1	1	147
3	A	normal	1	2	139
4	B	normal	1	1	246
5	B	normal	1	2	240
6	A	mutant	1	1	166
7	A	mutant	1	2	179
8	B	mutant	1	1	178
9	B	mutant	1	2	172
10	A	normal	5	1	334
11	A	normal	5	2	354
12	B	normal	5	1	514
13	B	normal	5	2	611
14	A	mutant	5	1	451
15	A	mutant	5	2	474
16	B	mutant	5	1	412
17	B	mutant	5	2	447

Tidied version

Key points from Bromman & Woo

A

	A	B	C	D	E	F
1						
2		101	102	103	104	105
3	sex	Male	Female	Male	Male	Male
4						
5		101	102	103	104	105
6	glucose	134.1	120.0	124.8	83.1	105.2
7						
8		101	102	103	104	105
9	insulin	0.60	1.18	1.23	1.16	0.73

B

	A	B	C	D	E	F	G
1	1MIN						
2			Normal			Mutant	
3	B6	146.6	138.6	155.6	166	179.3	186.9
4	BTBR	245.7	240	243.1	177.8	171.6	188.1
5							
6	5MIN						
7			Normal			Mutant	
8	B6	333.6	353.6	408.8	450.6	474.4	423.8
9	BTBR	514.4	610.6	597.9	412.1	447.4	446.5

C

	A	B	C	D	E	F	G
1							
2	Date	11/3/14					
3	Days on diet	126					
4	Mouse #	43					
5	sex	f					
6	experiment		values		mean	SD	
7	control		0.186	0.191	1.081	0.49	0.52
8	treatment A		7.414	1.468	2.254	3.71	3.23
9	treatment B		9.811	9.259	11.296	10.12	1.05
10							
11	fold change		values		mean	SD	
12	treatment A		15.26	3.02	4.64	7.64	6.65
13	treatment B		20.19	19.05	23.24	20.83	2.17

D

	A	B	C	D	E	F
1		GTT date	GTT weight	time	glucose mg/dl	insulin ng/ml
2	321	2/9/15	24.5	0	99.2	lo off curve
3				5	349.3	0.205
4				15	286.1	0.129
5				30	312	0.175
6				60	99.9	0.122
7				120	217.9	lo off curve
8	322	2/9/15	18.9	0	185.8	0.251
9				5	297.4	2.228
10				15	439	2.078
11				30	362.3	0.775
12				60	232.7	0.5
13				120	260.7	0.523
14	323	2/9/15	24.7	0	198.5	0.151
15				5	530.6	off curve lo

Rectangle your data

Key points from Broman & Woo

A

	A	B	C
1	id	GTT date	GTT weight
2	321	2/9/15	24.5
3	322	2/9/15	18.9
4	323	2/9/15	24.7

B

	A	B	C	D	E
1	id	GTT time	glucose mg/dl	insulin ng/ml	note
2	321	0	99.2	NA	insulin below curve
3	321	5	349.3	0.205	
4	321	15	286.1	0.129	
5	321	30	312	0.175	
6	321	60	99.9	0.122	
7	321	120	217.9	NA	insulin below curve
8	322	0	185.8	0.251	
9	322	5	297.4	2.228	
10	322	15	439	2.078	
11	322	30	362.3	0.775	
12	322	60	232.7	0.5	
13	322	120	260.7	0.523	
14	323	0	198.5	0.151	
15	323	5	530.6	NA	insulin below curve

Use more than one table if needed. We can join them later.

Key points from Broman & Woo

	A	B	C	D	E	F	G	H	I	J	K
1			week 4			week 6			week 8		
2	Mouse ID	SEX	date	weight	glucose	date	weight	glucose	date	weight	glucose
3	3005	M	3/30/2007	19.3	635	4/11/2007	31	460.7	4/27/2007	39.6	530.2
4	3017	M	10/6/2006	25.9	202.4	10/19/2006	45.1	384.7	11/3/2006	57.2	458.7
5	3434	F	11/22/2006	26.6	238.9	12/6/2006	45.9	378	12/22/2006	56.2	409.8
6	3449	M	1/5/2007	27.5	121	1/19/2007	42.9	191.3	2/2/2007	56.7	182.5
7	3499	F	1/5/2007	19.8	220.2	1/19/2007	36.6	556.9	2/2/2007	43.6	446

Needs a single header row and a consistent naming scheme.

Key points from Broman & Woo

	A	B	C	D	E	F
1	mouse_id	sex	week	date	glucose	weight
2	3005	M	4	3/30/2007	19.3	635
3	3005	M	6	4/11/2007	31	460.7
4	3005	M	8	4/27/2007	39.6	530.2
5	3017	M	4	10/6/2006	25.9	202.4
6	3017	M	6	10/19/2006	45.1	384.7
7	3017	M	8	11/3/2006	57.2	458.7
8	3434	F	4	11/22/2006	26.6	238.9
9	3434	F	6	12/6/2006	45.9	378
10	3434	F	8	12/22/2006	56.2	409.8
11	3449	M	4	1/5/2007	27.5	121
12	3449	M	6	1/19/2007	42.9	191.3
13	3449	M	8	2/2/2007	56.7	182.5
14	3499	F	4	1/5/2007	19.8	220.2
15	3499	F	6	1/19/2007	36.6	556.9
16	3499	F	8	2/2/2007	43.6	446

Tidied version.

The most common `tidyverse` operation

edu

```
# A tibble: 366 × 11
  age    sex   year total elem4 elem8   hs3   hs4 coll13 coll14 median
  <chr> <chr> <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
1 25-34 Male   2016 21845   116   468 1427  6386  6015  7432     NA
2 25-34 Male   2015 21427   166   488 1584  6198  5920  7071     NA
3 25-34 Male   2014 21217   151   512 1611  6323  5910  6710     NA
4 25-34 Male   2013 20816   161   582 1747  6058  5749  6519     NA
5 25-34 Male   2012 20464   161   579 1707  6127  5619  6270     NA
6 25-34 Male   2011 20985   190   657 1791  6444  5750  6151     NA
7 25-34 Male   2010 20689   186   641 1866  6458  5587  5951     NA
8 25-34 Male   2009 20440   184   695 1806  6495  5508  5752     NA
9 25-34 Male   2008 20210   172   714 1874  6356  5277  5816     NA
10 25-34 Male  2007 20024   246   757 1930  6361  5137  5593    NA
# i 356 more rows
```

The “Level of Schooling Attained” measure is spread across the columns, from `elem4` to `coll14`.

This is fine for a compact table, but for us it should be a single measure, say, “education”.

Wide to long with `pivot_longer()`

We're going to *pivot* the table. That is, we'll put the columns `elem4:coll4` into a new column, creating a new categorical measure named `education`. The numbers currently under each column will become a new `value` column corresponding to that level of education.

```
edu ▶
pivot_longer(elem4:coll4, names_to = "education")

# A tibble: 2,196 × 7
  age   sex   year total median education value
  <chr> <chr> <int>  <dbl>  <chr>      <dbl>
1 25-34 Male    2016 21845     NA elem4       116
2 25-34 Male    2016 21845     NA elem8       468
3 25-34 Male    2016 21845     NA hs3        1427
4 25-34 Male    2016 21845     NA hs4        6386
5 25-34 Male    2016 21845     NA coll3      6015
6 25-34 Male    2016 21845     NA coll4      7432
7 25-34 Male    2015 21427     NA elem4       166
8 25-34 Male    2015 21427     NA elem8       488
9 25-34 Male    2015 21427     NA hs3        1584
10 25-34 Male   2015 21427     NA hs4        6198
# i 2,186 more rows
```

Wide to long with `pivot_longer()`

We can name the “value” column to whatever we like. Here it’s a number of people, so let’s call it “`n`”.

```
edu ▶
pivot_longer(elem4:coll4, names_to = "education", values_to = "n")

# A tibble: 2,196 × 7
  age   sex   year total median education     n
  <chr> <chr> <int> <dbl> <chr>      <dbl>
1 25-34 Male    2016 21845    NA elem4        116
2 25-34 Male    2016 21845    NA elem8        468
3 25-34 Male    2016 21845    NA hs3         1427
4 25-34 Male    2016 21845    NA hs4         6386
5 25-34 Male    2016 21845    NA coll3       6015
6 25-34 Male    2016 21845    NA coll4       7432
7 25-34 Male    2015 21427    NA elem4       166
8 25-34 Male    2015 21427    NA elem8       488
9 25-34 Male    2015 21427    NA hs3         1584
10 25-34 Male   2015 21427    NA hs4         6198
# i 2,186 more rows
```

Let's recode it while we're here

We use `case_match()` for simple recoding:

```
edu >
  pivot_longer(elem4:coll4, names_to = "education", values_to = "n") >
  mutate(education = case_match(education,
                                "elem4" ~ "Elementary 4",
                                "elem8" ~ "Elementary 8",
                                "hs3" ~ "High School 3",
                                "hs4" ~ "High School 4",
                                "coll3" ~ "College 3",
                                "coll4" ~ "College 4"))
```

```
# A tibble: 2,196 × 7
  age    sex   year total median education      n
  <chr> <chr> <int> <int> <dbl> <chr>      <dbl>
1 25-34 Male   2016 21845     NA Elementary 4    116
2 25-34 Male   2016 21845     NA Elementary 8    468
3 25-34 Male   2016 21845     NA High School 3  1427
4 25-34 Male   2016 21845     NA High School 4  6386
5 25-34 Male   2016 21845     NA College 3     6015
6 25-34 Male   2016 21845     NA College 4     7432
7 25-34 Male   2015 21427     NA Elementary 4    166
8 25-34 Male   2015 21427     NA Elementary 8    488
9 25-34 Male   2015 21427     NA High School 3  1584
10 25-34 Male  2015 21427     NA High School 4   6198
# i 2,186 more rows
```

case_match() to collapse values

```
edu ▷  
pivot_longer(elem4:coll4, names_to = "education", values_to = "n") ▷  
mutate(education = case_match(education,  
                           c("elem4", "elem8") ~ "Elementary",  
                           c("hs3", "hs4") ~ "High School",  
                           c("coll3", "coll4") ~ "College"))
```

```
# A tibble: 2,196 × 7  
  age   sex   year total median education      n  
  <chr> <chr> <int> <int>    <dbl> <chr>      <dbl>  
1 25-34 Male   2016 21845     NA Elementary    116  
2 25-34 Male   2016 21845     NA Elementary    468  
3 25-34 Male   2016 21845     NA High School  1427  
4 25-34 Male   2016 21845     NA High School  6386  
5 25-34 Male   2016 21845     NA College     6015  
6 25-34 Male   2016 21845     NA College     7432  
7 25-34 Male   2015 21427     NA Elementary    166  
8 25-34 Male   2015 21427     NA Elementary    488  
9 25-34 Male   2015 21427     NA High School  1584  
10 25-34 Male  2015 21427     NA High School  6198  
# i 2,186 more rows
```

case_match() to collapse values

```
edu >
  pivot_longer(elem4:coll4, names_to = "education", values_to = "n") >
  mutate(education = case_match(education,
                                c("elem4", "elem8") ~ "Elementary",
                                c("hs3", "hs4") ~ "High School",
                                c("coll3", "coll4") ~ "College")) >
  group_by(age, sex, year, education) >
  summarize(n = sum(n))
```

```
# A tibble: 1,098 × 5
# Groups:   age, sex, year [366]
  age    sex    year education      n
  <chr> <chr> <int> <chr>     <dbl>
1 25-34 Female  1940 College     1406
2 25-34 Female  1940 Elementary  4342
3 25-34 Female  1940 High School 4986
4 25-34 Female  1947 College     1555
5 25-34 Female  1947 Elementary  3329
6 25-34 Female  1947 High School 6796
7 25-34 Female  1950 College     1795
8 25-34 Female  1950 Elementary  3119
9 25-34 Female  1950 High School 7034
10 25-34 Female  1952 College    1886
# i 1,088 more rows
```

case_match() and .default

```
edu ▷  
pivot_longer(elem4:coll4, names_to = "education", values_to = "n") ▷  
mutate(education = case_match(education,  
                           c("elem4", "elem8") ~ "Elementary",  
                           .default = "Other"))
```

```
# A tibble: 2,196 × 7  
  age   sex   year total median education      n  
  <chr> <chr> <int> <int>  <dbl> <chr>      <dbl>  
1 25-34 Male    2016 21845     NA Elementary    116  
2 25-34 Male    2016 21845     NA Elementary    468  
3 25-34 Male    2016 21845     NA Other       1427  
4 25-34 Male    2016 21845     NA Other       6386  
5 25-34 Male    2016 21845     NA Other       6015  
6 25-34 Male    2016 21845     NA Other       7432  
7 25-34 Male    2015 21427     NA Elementary   166  
8 25-34 Male    2015 21427     NA Elementary   488  
9 25-34 Male    2015 21427     NA Other       1584  
10 25-34 Male   2015 21427     NA Other       6198  
# i 2,186 more rows
```

case_when() for more complex criteria

case_match() replaces values with values. case_when() is more general and can create or recode variables based on criteria from other variables.

```
edu >  
pivot_longer(elem4:coll4, names_to = "education", values_to = "n") >  
mutate(  
  weird_group = case_when(  
    age == "25-34" & year > 1980 & (education %in% c("elem4", "elem8")) ~ "Group 1",  
    age == "25-34" & (year == 2016 | education == "hs3") ~ "Group 2",  
    .default = "Group 3"))
```

```
# A tibble: 2,196 × 8  
  age   sex   year total median education      n weird_group  
  <chr> <chr> <int> <int>  <dbl> <chr>     <dbl> <chr>  
1 25-34 Male    2016 21845     NA elem4       116 Group 1  
2 25-34 Male    2016 21845     NA elem8       468 Group 1  
3 25-34 Male    2016 21845     NA hs3        1427 Group 2  
4 25-34 Male    2016 21845     NA hs4        6386 Group 2  
5 25-34 Male    2016 21845     NA coll3      6015 Group 2  
6 25-34 Male    2016 21845     NA coll4      7432 Group 2  
7 25-34 Male    2015 21427     NA elem4       166 Group 1  
8 25-34 Male    2015 21427     NA elem8       488 Group 1  
9 25-34 Male    2015 21427     NA hs3        1584 Group 2  
10 25-34 Male   2015 21427     NA hs4        6198 Group 3  
# i 2,186 more rows
```

pivot_longer() implies pivot_wider()

```
gapminder
```

```
# A tibble: 1,704 × 6
  country   continent year lifeExp      pop gdpPercap
  <fct>     <fct>    <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia      1952    28.8  8425333     779.
2 Afghanistan Asia      1957    30.3  9240934     821.
3 Afghanistan Asia      1962    32.0 10267083     853.
4 Afghanistan Asia      1967    34.0 11537966     836.
5 Afghanistan Asia      1972    36.1 13079460     740.
6 Afghanistan Asia      1977    38.4 14880372     786.
7 Afghanistan Asia      1982    39.9 12881816     978.
8 Afghanistan Asia      1987    40.8 13867957     852.
9 Afghanistan Asia      1992    41.7 16317921     649.
10 Afghanistan Asia     1997    41.8 22227415     635.
# i 1,694 more rows
```

But they're not symmetric operations!

pivot_longer() implies pivot_wider()

```
gapminder >
  select(country, continent, year, lifeExp) >
  pivot_wider(names_from = year, values_from = lifeExp)

# A tibble: 142 × 14
  country   continent `1952` `1957` `1962` `1967` `1972` `1977` `1982` `1987` 
  <fct>     <fct>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  
1 Afghanistan Asia      28.8    30.3   32.0   34.0   36.1   38.4   39.9   40.8  
2 Albania      Europe    55.2    59.3   64.8   66.2   67.7   68.9   70.4   72      
3 Algeria       Africa    43.1    45.7   48.3   51.4   54.5   58.0   61.4   65.8  
4 Angola        Africa    30.0    32.0   34      36.0   37.9   39.5   39.9   39.9  
5 Argentina     Americas  62.5    64.4   65.1   65.6   67.1   68.5   69.9   70.8  
6 Australia     Oceania   69.1    70.3   70.9   71.1   71.9   73.5   74.7   76.3  
7 Austria       Europe    66.8    67.5   69.5   70.1   70.6   72.2   73.2   74.9  
8 Bahrain        Asia     50.9    53.8   56.9   59.9   63.3   65.6   69.1   70.8  
9 Bangladesh     Asia     37.5    39.3   41.2   43.5   45.3   46.9   50.0   52.8  
10 Belgium       Europe    68      69.2   70.2   70.9   71.4   72.8   73.9   75.4 
# i 132 more rows
# i 4 more variables: `1992` <dbl>, `1997` <dbl>, `2002` <dbl>, `2007` <dbl>
```

What about *multiple* columns?

This is a pretty common problem. A first thought (“Just don’t mention the other columns”) isn’t it:

```
gapminder >
  pivot_wider(names_from = year, values_from = lifeExp)

# A tibble: 1,704 × 16
  country continent    pop gdpPercap `1952` `1957` `1962` `1967` `1972` `1977`
  <fct>   <fct>     <int>   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
  1 Afghani... Asia      8.43e6    779.   28.8     NA     NA     NA     NA
  2 Afghani... Asia      9.24e6    821.   NA      30.3     NA     NA     NA
  3 Afghani... Asia      1.03e7    853.   NA      NA      32.0     NA     NA
  4 Afghani... Asia      1.15e7    836.   NA      NA      NA      34.0     NA
  5 Afghani... Asia      1.31e7    740.   NA      NA      NA      NA      36.1
  6 Afghani... Asia      1.49e7    786.   NA      NA      NA      NA      38.4
  7 Afghani... Asia      1.29e7    978.   NA      NA      NA      NA      NA
  8 Afghani... Asia      1.39e7    852.   NA      NA      NA      NA      NA
  9 Afghani... Asia      1.63e7    649.   NA      NA      NA      NA      NA
 10 Afghani... Asia      2.22e7    635.   NA      NA      NA      NA      NA
# i 1,694 more rows
# i 6 more variables: `1982` <dbl>, `1987` <dbl>, `1992` <dbl>, `1997` <dbl>,
#   `2002` <dbl>, `2007` <dbl>
```

`pop` and `gdpPercap` are still long, and now our table is really sparse.

What about *multiple* columns?

We need to specify that we want values from more than one column.

```
gapminder >
  select(country, continent, year, lifeExp, gdpPercap) >
  pivot_wider(names_from = year, values_from = c(lifeExp, gdpPercap))

# A tibble: 142 × 26
  country   continent lifeExp_1952 lifeExp_1957 lifeExp_1962 lifeExp_1967
  <fct>     <fct>      <dbl>       <dbl>       <dbl>       <dbl>
1 Afghanistan Asia        28.8       30.3       32.0       34.0
2 Albania      Europe      55.2       59.3       64.8       66.2
3 Algeria      Africa      43.1       45.7       48.3       51.4
4 Angola       Africa      30.0       32.0       34         36.0
5 Argentina    Americas     62.5       64.4       65.1       65.6
6 Australia    Oceania     69.1       70.3       70.9       71.1
7 Austria      Europe      66.8       67.5       69.5       70.1
8 Bahrain      Asia        50.9       53.8       56.9       59.9
9 Bangladesh   Asia        37.5       39.3       41.2       43.5
10 Belgium     Europe      68         69.2       70.2       70.9
# i 132 more rows
# i 20 more variables: lifeExp_1972 <dbl>, lifeExp_1977 <dbl>,
#   lifeExp_1982 <dbl>, lifeExp_1987 <dbl>, lifeExp_1992 <dbl>,
#   lifeExp_1997 <dbl>, lifeExp_2002 <dbl>, lifeExp_2007 <dbl>,
#   gdpPercap_1952 <dbl>, gdpPercap_1957 <dbl>, gdpPercap_1962 <dbl>,
#   gdpPercap_1967 <dbl>, gdpPercap_1972 <dbl>, gdpPercap_1977 <dbl>,
#   gdpPercap_1982 <dbl>, gdpPercap_1987 <dbl>, gdpPercap_1992 <dbl>, ...
```

This will give us a very wide table, but it's what we wanted.

Pivot wider while summarizing

```
# Some made-up data
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))
dfstrat
```

```
# A tibble: 1,000 × 5
  stratum sex   race  educ income
  <dbl> <chr> <chr> <chr> <dbl>
1     6 F     W     HS    83.7
2     5 F     W     BA   128.
3     3 F     B     HS    66.3
4     3 F     W     HS    111.
5     6 M     W     BA   116.
6     7 M     B     HS   159.
7     8 M     W     BA   131.
8     3 M     W     BA   94.4
9     7 F     B     HS   146.
10    2 F     W     BA   88.8
# i 990 more rows
```

Pivot wider while summarizing

```
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))
```

Pivot wider while summarizing

```
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))  
dfstrat
```

```
# A tibble: 1,000 × 5  
  stratum sex   race  educ income  
  <dbl> <chr> <chr> <chr> <dbl>  
1     6 F     W     HS    83.7  
2     5 F     W     BA   128.  
3     3 F     B     HS   66.3  
4     3 F     W     HS   111.  
5     6 M     W     BA   116.  
6     7 M     B     HS   159.  
7     8 M     W     BA   131.  
8     3 M     W     BA   94.4  
9     7 F     B     HS   146.  
10    2 F     W     BA   88.8  
# i 990 more rows
```

Pivot wider while summarizing

```
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))
dfstrat ⚡
  group_by(sex, race, stratum, educ)
```

```
# A tibble: 1,000 × 5
# Groups:   sex, race, stratum, educ [64]
  stratum sex   race  educ income
  <dbl>   <chr> <chr> <chr>  <dbl>
1       6 F     W    HS      83.7
2       5 F     W    BA     128.
3       3 F     B    HS      66.3
4       3 F     W    HS      111.
5       6 M     W    BA     116.
6       7 M     B    HS      159.
7       8 M     W    BA     131.
8       3 M     W    BA      94.4
9       7 F     B    HS     146.
10      2 F    W    BA      88.8
# i 990 more rows
```

Pivot wider while summarizing

```
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))
dfstrat >
  group_by(sex, race, stratum, educ) >
  summarize(mean_inc = mean(income),
            n = n())
```

```
# A tibble: 64 × 6
# Groups:   sex, race, stratum [32]
  sex    race   stratum educ mean_inc     n
  <chr>  <chr>   <dbl> <chr>    <dbl> <int>
1 F      B        1 BA    93.8    19
2 F      B        1 HS    99.3     6
3 F      B        2 BA    89.7    11
4 F      B        2 HS    93.0    16
5 F      B        3 BA   112.     13
6 F      B        3 HS    95.0    16
7 F      B        4 BA   108.     14
8 F      B        4 HS    96.1    15
9 F      B        5 BA    91.0    11
10 F     B        5 HS   92.6    15
# i 54 more rows
```

Pivot wider while summarizing

```
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))
dfstrat >
  group_by(sex, race, stratum, educ) >
  summarize(mean_inc = mean(income),
            n = n()) >
  pivot_wider(names_from = (educ),
              values_from = c(mean_inc, n))
```

```
# A tibble: 32 × 7
# Groups:   sex, race, stratum [32]
  sex   race  stratum mean_inc_BA mean_inc_HS n_BA n_HS
  <chr> <chr>    <dbl>      <dbl>     <dbl> <int> <int>
1 F     B        1       93.8      99.3     19    6
2 F     B        2       89.7      93.0     11    16
3 F     B        3       112.      95.0     13    16
4 F     B        4       108.      96.1     14    15
5 F     B        5       91.0      92.6     11    15
6 F     B        6       93.0      116.     15    15
7 F     B        7       102.      121.     13    13
8 F     B        8       105.      88.3     14    8 
9 F     W        1       92.6      110.     19    13
10 F    W       2       98.5      101.     15    19
# i 22 more rows
```

Pivot wider while summarizing

```
dfstrat ← read_csv(here::here("data", "dfstrat.csv"))
dfstrat >
  group_by(sex, race, stratum, educ) >
  summarize(mean_inc = mean(income),
            n = n()) >
  pivot_wider(names_from = (educ),
              values_from = c(mean_inc, n)) >
  ungroup()
```

```
# A tibble: 32 × 7
  sex   race stratum mean_inc_BA mean_inc_HS n_BA n_HS
  <chr> <chr>    <dbl>      <dbl>      <dbl> <int> <int>
  1 F     B        1       93.8      99.3     19     6
  2 F     B        2       89.7      93.0     11    16
  3 F     B        3      112.      95.0     13    16
  4 F     B        4      108.      96.1     14    15
  5 F     B        5      91.0      92.6     11    15
  6 F     B        6      93.0      116.     15    15
  7 F     B        7      102.      121.     13    13
  8 F     B        8      105.      88.3     14     8
  9 F     W        1      92.6      110.     19    13
 10 F    W        2      98.5      101.     15    19
# i 22 more rows
```

Here we end up with sex-by-race-by-stratum in the rows, and the income-by-education means, and income-by-education Ns, in their own columns.

Separate and Unite

separate() and unite() columns

```
## tribble() lets you make tibbles by hand
df ← tribble(
  ~name, ~occupation,
  "Nero.Wolfe", "Private Detective",
  "Archie.Goodwin", "Personal Assistant",
  "Fritz.Brenner", "Cook and Butler",
  "Theodore.Horstmann", "Orchid Expert"
)
```

```
df
```

```
# A tibble: 4 × 2
  name          occupation
  <chr>        <chr>
1 Nero.Wolfe  Private Detective
2 Archie.Goodwin Personal Assistant
3 Fritz.Brenner Cook and Butler
4 Theodore.Horstmann Orchid Expert
```

separate() and unite() columns

```
## tribble() lets you make tibbles by hand
df ← tribble(
  ~name, ~occupation,
  "Nero.Wolfe", "Private Detective",
  "Archie.Goodwin", "Personal Assistant",
  "Fritz.Brenner", "Cook and Butler",
  "Theodore.Horstmann", "Orchid Expert"
)
```

```
df
```

```
# A tibble: 4 × 2
  name          occupation
  <chr>        <chr>
1 Nero.Wolfe  Private Detective
2 Archie.Goodwin Personal Assistant
3 Fritz.Brenner Cook and Butler
4 Theodore.Horstmann Orchid Expert
```

Separate and unite

```
df
```

```
# A tibble: 4 × 2
  name          occupation
  <chr>         <chr>
1 Nero.Wolfe   Private Detective
2 Archie.Goodwin Personal Assistant
3 Fritz.Brenner Cook and Butler
4 Theodore.Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last"))
```

```
# A tibble: 4 × 3  
  first     last    occupation  
  <chr>    <chr>   <chr>  
1 Nero      Wolfe  Private Detective  
2 Archie    Goodwin Personal Assistant  
3 Fritz     Brenner Cook and Butler  
4 Theodore Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last")) %>  
  unite("full_name", first:last, sep = " ")
```

```
# A tibble: 4 × 2  
  full_name      occupation  
  <chr>          <chr>  
1 Nero Wolfe    Private Detective  
2 Archie Goodwin Personal Assistant  
3 Fritz Brenner Cook and Butler  
4 Theodore Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last")) %>  
  unite("full_name", first:last, sep = " ") %>  
  unite("both_together", full_name:occupation,  
        sep = ", ", remove = FALSE)
```

```
# A tibble: 4 × 3  
  both_together      full_name    occupation  
  <chr>            <chr>        <chr>  
1 Nero Wolfe, Private Detective Nero Wolfe Private Detective  
2 Archie Goodwin, Personal Assistant Archie Goodwin Personal Assistant  
3 Fritz Brenner, Cook and Butler  Fritz Brenner Cook and Butler  
4 Theodore Horstmann, Orchid Expert Theodore Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last")) %>  
  unite("full_name", first:last, sep = " ") %>  
  unite("both_together", full_name:occupation,  
        sep = ", ", remove = FALSE)
```

```
# A tibble: 4 × 3  
  both_together      full_name    occupation  
  <chr>            <chr>        <chr>  
1 Nero Wolfe, Private Detective Nero Wolfe Private Detective  
2 Archie Goodwin, Personal Assistant Archie Goodwin Personal Assistant  
3 Fritz Brenner, Cook and Butler   Fritz Brenner Cook and Butler  
4 Theodore Horstmann, Orchid Expert Theodore Horstmann Orchid Expert
```

Separate and unite

```
df
```

```
# A tibble: 4 × 2
  name          occupation
  <chr>         <chr>
1 Nero.Wolfe   Private Detective
2 Archie.Goodwin Personal Assistant
3 Fritz.Brenner Cook and Butler
4 Theodore.Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last"))
```

```
# A tibble: 4 × 3  
  first     last    occupation  
  <chr>    <chr>   <chr>  
1 Nero      Wolfe  Private Detective  
2 Archie    Goodwin Personal Assistant  
3 Fritz     Brenner Cook and Butler  
4 Theodore Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last")) %>  
  unite("full_name", first:last)
```

```
# A tibble: 4 × 2  
  full_name      occupation  
  <chr>          <chr>  
1 Nero_Wolfe   Private Detective  
2 Archie_Goodwin Personal Assistant  
3 Fritz_Brenner Cook and Butler  
4 Theodore_Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last")) %>  
  unite("full_name", first:last) %>  
  separate(full_name, into = c("first", "last"))
```

```
# A tibble: 4 × 3  
  first     last   occupation  
  <chr>    <chr>   <chr>  
1 Nero      Wolfe  Private Detective  
2 Archie    Goodwin Personal Assistant  
3 Fritz     Brenner Cook and Butler  
4 Theodore Horstmann Orchid Expert
```

Separate and unite

```
df %>  
  separate(name, into = c("first", "last")) %>  
  unite("full_name", first:last) %>  
  separate(full_name, into = c("first", "last"))
```

```
# A tibble: 4 × 3  
  first     last    occupation  
  <chr>    <chr>   <chr>  
1 Nero      Wolfe  Private Detective  
2 Archie    Goodwin Personal Assistant  
3 Fritz     Brenner Cook and Butler  
4 Theodore Horstmann Orchid Expert
```

The underscore, `_`, is the default uniting character.

Separate and unite

gss_sm

	year	id	ballot	age	child	sibs	degree	race	sex	region
income16	<dbl>	<dbl>	<labelled>	<dbl>	<dbl>	<labe>	<fct>	<fct>	<fct>	<fct>
	<fct>									
1	2016	1	1				47	3	2	Bache...
\$170000...								White	Male	New E...
2	2016	2	2				61	0	3	High ...
\$50000 ...								White	Male	New E...
3	2016	3	3				72	2	3	Bache...
\$75000 ...								White	Male	New E...
4	2016	4	1				43	4	3	High ...
\$170000...								White	Fema...	New E...
5	2016	5	3				55	2	2	Gradu...
\$170000...								White	Fema...	New E...
6	2016	6	2				53	2	2	Junio...
\$60000 ...								White	Fema...	New E...
7	2016	7	1				50	2	2	High ...
\$170000...								White	Male	New E...
8	2016	8	3				23	3	6	High ...
\$30000 ...								Other	Fema...	Middl...
9	2016	9	1				45	3	5	High ...
\$60000 ...								Black	Male	Middl...
10	2016	10	3				71	4	1	Junio...
\$60000 ...								White	Male	Middl...

Separate and unite

```
gss_sm %>  
  select(race, degree)
```

```
# A tibble: 2,867 × 2  
  race   degree  
  <fct> <fct>  
 1 White Bachelor  
 2 White High School  
 3 White Bachelor  
 4 White High School  
 5 White Graduate  
 6 White Junior College  
 7 White High School  
 8 Other High School  
 9 Black High School  
10 White Junior College  
# i 2,857 more rows
```

Separate and unite

```
gss_sm %>  
  select(race, degree) %>  
  mutate(racedeg = interaction(race, degree))  
  
# A tibble: 2,867 x 3  
  race   degree      racedeg  
  <fct>  <fct>       <fct>  
  1 White Bachelor  White.Bachelor  
  2 White High School  White.High School  
  3 White Bachelor  White.Bachelor  
  4 White High School  White.High School  
  5 White Graduate  White.Graduate  
  6 White Junior College  White.Junior College  
  7 White High School  White.High School  
  8 Other High School  Other.High School  
  9 Black High School  Black.High School  
 10 White Junior College  White.Junior College  
# i 2,857 more rows
```

Separate and unite

```
gss_sm %>  
  select(race, degree) %>  
  mutate(racedeg = interaction(race, degree)) %>  
  group_by(racedeg)
```

```
# A tibble: 2,867 × 3  
# Groups:   racedeg [16]  
  race    degree      racedeg  
  <fct>  <fct>       <fct>  
1 White Bachelor  White.Bachelor  
2 White High School  White.High School  
3 White Bachelor  White.Bachelor  
4 White High School  White.High School  
5 White Graduate  White.Graduate  
6 White Junior College  White.Junior College  
7 White High School  White.High School  
8 Other High School  Other.High School  
9 Black High School  Black.High School  
10 White Junior College  White.Junior College  
# i 2,857 more rows
```

Separate and unite

```
gss_sm %>  
  select(race, degree) %>  
  mutate(racedeg = interaction(race, degree)) %>  
  group_by(racedeg) %>  
  tally()
```

```
# A tibble: 16 × 2  
  racedeg          n  
  <fct>        <int>  
1 White.Lt High School    197  
2 Black.Lt High School     60  
3 Other.Lt High School     71  
4 White.High School    1057  
5 Black.High School     292  
6 Other.High School      112  
7 White.Junior College   166  
8 Black.Junior College    33  
9 Other.Junior College    17  
10 White.Bachelor       426  
11 Black.Bachelor        71  
12 Other.Bachelor        39  
13 White.Graduate        250  
14 Black.Graduate         31  
15 Other.Graduate        37  
16 <NA>                      8
```

Separate and unite

```
gss_sm %>  
  select(race, degree) %>  
  mutate(racedeg = interaction(race, degree)) %>  
  group_by(racedeg) %>  
  tally() %>  
  separate(racedeg, sep = "\\.", into = c("race", "degree"))
```

```
# A tibble: 16 × 3  
  race   degree      n  
  <chr>  <chr>     <int>  
1 White Lt High School    197  
2 Black Lt High School     60  
3 Other Lt High School    71  
4 White High School      1057  
5 Black High School       292  
6 Other High School       112  
7 White Junior College    166  
8 Black Junior College     33  
9 Other Junior College     17  
10 White Bachelor        426  
11 Black Bachelor         71  
12 Other Bachelor        39  
13 White Graduate        250  
14 Black Graduate         31  
15 Other Graduate         37  
16 <NA>    <NA>          8
```

This one is a bit trickier, and our first glimpse of a *regular expression*.
We have to tell **separate()** to split on the period, not the space.

More advanced pivots

Example: tidy selectors

```
billboard
```

```
# A tibble: 317 × 79
  artist      track date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
  <chr>      <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 2 Pac      Baby... 2000-02-26     87    82    72    77    87    94    99    NA
2 2Ge+her    The ... 2000-09-02     91    87    92    NA    NA    NA    NA    NA
3 3 Doors D... Kryp... 2000-04-08     81    70    68    67    66    57    54    53
4 3 Doors D... Loser  2000-10-21     76    76    72    69    67    65    55    59
5 504 Boyz   Wobb... 2000-04-15     57    34    25    17    17    31    36    49
6 98^0       Give... 2000-08-19     51    39    34    26    26    19    2     2
7 A*Teens    Danc... 2000-07-08     97    97    96    95    100   NA    NA    NA
8 Aaliyah    I Do... 2000-01-29     84    62    51    41    38    35    35    38
9 Aaliyah    Try ... 2000-03-18     59    53    38    28    21    18    16    14
10 Adams, Yo... Open... 2000-08-26    76    76    74    69    68    67    61    58
# i 307 more rows
# i 68 more variables: wk9 <dbl>, wk10 <dbl>, wk11 <dbl>, wk12 <dbl>,
# wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>,
# wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>,
# wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>,
# wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,
```

Example: tidy selectors

```
billboard >
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
  )

# A tibble: 5,307 × 5
  artist   track           date.entered week   rank
  <chr>   <chr>          <date>       <chr> <dbl>
1 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk1     87
2 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk2     82
3 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk3     72
4 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk4     77
5 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk5     87
6 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk6     94
7 2 Pac   Baby Don't Cry (Keep ... 2000-02-26 wk7     99
8 2Ge+her The Hardest Part Of ... 2000-09-02 wk1     91
9 2Ge+her The Hardest Part Of ... 2000-09-02 wk2     87
10 2Ge+her The Hardest Part Of ... 2000-09-02 wk3     92
# i 5,297 more rows
```

Example: parse functions

```
billboard >
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    names_transform = readr::parse_number,
    values_to = "rank",
    values_drop_na = TRUE,
  )

# A tibble: 5,307 × 5
  artist   track           date.entered week   rank
  <chr>   <chr>          <date>       <dbl> <dbl>
1 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     1     87
2 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     2     82
3 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     3     72
4 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     4     77
5 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     5     87
6 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     6     94
7 2 Pac   Baby Don't Cry (Keep ... 2000-02-26     7     99
8 2Ge+her The Hardest Part Of ... 2000-09-02     1     91
9 2Ge+her The Hardest Part Of ... 2000-09-02     2     87
10 2Ge+her The Hardest Part Of ... 2000-09-02    3     92
# i 5,297 more rows
```

Example: many vars in cols

```
who
```

```
# A tibble: 7,240 × 60
  country iso2 iso3   year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544
  <chr>   <chr> <chr> <dbl>       <dbl>       <dbl>       <dbl>
1 Afghani... AF    AFG  1980        NA         NA         NA         NA
2 Afghani... AF    AFG  1981        NA         NA         NA         NA
3 Afghani... AF    AFG  1982        NA         NA         NA         NA
4 Afghani... AF    AFG  1983        NA         NA         NA         NA
5 Afghani... AF    AFG  1984        NA         NA         NA         NA
6 Afghani... AF    AFG  1985        NA         NA         NA         NA
7 Afghani... AF    AFG  1986        NA         NA         NA         NA
8 Afghani... AF    AFG  1987        NA         NA         NA         NA
9 Afghani... AF    AFG  1988        NA         NA         NA         NA
10 Afghani... AF   AFG  1989        NA         NA         NA         NA
# i 7,230 more rows
# i 52 more variables: new_sp_m4554 <dbl>, new_sp_m5564 <dbl>,
#   new_sp_m65 <dbl>, new_sp_f014 <dbl>, new_sp_f1524 <dbl>,
#   new_sp_f2534 <dbl>, new_sp_f3544 <dbl>, new_sp_f4554 <dbl>,
#   new_sp_f5564 <dbl>, new_sp_f65 <dbl>, new_sn_m014 <dbl>,
#   new_sn_m1524 <dbl>, new_sn_m2534 <dbl>, new_sn_m3544 <dbl>,
```

Example: many vars in cols

```
who    >
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = c("diagnosis", "gender", "age"),
    names_pattern = "new_?(.*)_(.)(.*)",
    values_to = "count"
  )
# A tibble: 405,440 × 8
  country   iso2   iso3   year diagnosis gender age   count
  <chr>     <chr>   <chr> <dbl> <chr>     <chr> <dbl>
1 Afghanistan AF     AFG     1980 sp       m      014     NA
2 Afghanistan AF     AFG     1980 sp       m      1524    NA
3 Afghanistan AF     AFG     1980 sp       m      2534    NA
4 Afghanistan AF     AFG     1980 sp       m      3544    NA
5 Afghanistan AF     AFG     1980 sp       m      4554    NA
6 Afghanistan AF     AFG     1980 sp       m      5564    NA
7 Afghanistan AF     AFG     1980 sp       m      65      NA
8 Afghanistan AF     AFG     1980 sp       f      014     NA
9 Afghanistan AF     AFG     1980 sp       f      1524    NA
10 Afghanistan AF     AFG     1980 sp      f      2534   NA
# i 405,430 more rows
```

Example: many vars in cols

```
who %>
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = c("diagnosis", "gender", "age"),
    names_pattern = "new_?(.*)(.)(.*)",
    names_transform = list(
      gender = ~ readr::parse_factor(.x, levels = c("f", "m")),
      age = ~ readr::parse_factor(
        .x,
        levels = c("014", "1524", "2534", "3544", "4554", "5564", "65"),
        ordered = TRUE
      )
    ),
    values_to = "count"
  )

# A tibble: 405,440 × 8
  country   iso2   iso3   year diagnosis gender age   count
  <chr>     <chr>  <chr>  <dbl> <chr>    <fct> <ord> <dbl>
1 Afghanistan AF    AFG    1980 sp      m    014    NA
2 Afghanistan AF    AFG    1980 sp      m    1524   NA
3 Afghanistan AF    AFG    1980 sp      m    2534   NA
4 Afghanistan AF    AFG    1980 sp      m    3544   NA
5 Afghanistan AF    AFG    1980 sp      m    4554   NA
6 Afghanistan AF    AFG    1980 sp      m    5564   NA
7 Afghanistan AF    AFG    1980 sp      m    65     NA
8 Afghanistan AF    AFG    1980 sp      f    014    NA
9 Afghanistan AF    AFG    1980 sp      f    1524   NA
10 Afghanistan AF   AFG    1980 sp     f    2534   NA
# i 405,430 more rows
```

Example: long to wide

```
## Get the data and normalize the column names
df ← read_csv("http://kjhealy.co/MV0testdata.csv") ▷
  janitor::clean_names()

## Starting point
df

# A tibble: 26 × 11
      id date    wheelies right_shoe_color right_shoe_size left_shoe_color
      <dbl> <chr>    <chr>     <chr>           <dbl>    <chr>
1 8675309 2/1/2009   no       <NA>             NA       <NA>
2 8675309 2/4/2014   no       <NA>             NA       <NA>
3 8675309 2/15/2006  no       none            NA       none
4 8675309 3/1/2009   no       none            NA       <NA>
5 8675309 4/20/2013  no       white           NA       <NA>
6 8675309 4/30/2010  <NA>    white           3       white
7 8675309 5/5/2012   no       <NA>             NA       <NA>
8 8675309 7/31/2009  no       <NA>             NA       none
9 8675309 10/22/2008 no       <NA>            NA       none
10 9021033 1/11/2005 no       white          5       orange
# i 16 more rows
# i 5 more variables: left_shoe_size <dbl>, right_glove_color <chr>,
#   right_glove_size <dbl>, left_glove_color <chr>, left_glove_size <dbl>
```

Example: long to wide

```
colnames(df)
```

```
[1] "id"                  "date"                 "wheelies"  
[4] "right_shoe_color"   "right_shoe_size"    "left_shoe_color"  
[7] "left_shoe_size"     "right_glove_color" "right_glove_size"  
[10] "left_glove_color"   "left_glove_size"
```

Example: there & back

More fully lengthen by making side, item, color, and size into variables (ie columns)

```
df_lon ← df ▷  
pivot_longer(right_shoe_color:left_glove_size,  
            names_to = c("side", "item", ".value"),  
            names_pattern = "(.*)_(.*)_(.*)")  
  
df_lon  
  
# A tibble: 104 × 7  
      id date    wheelies  side   item  color  size  
      <dbl> <chr>     <chr> <chr> <chr> <dbl>  
1 8675309 2/1/2009 no     right shoe  <NA>    NA  
2 8675309 2/1/2009 no     left  shoe  <NA>    NA  
3 8675309 2/1/2009 no     right glove <NA>    3  
4 8675309 2/1/2009 no     left  glove <NA>    1  
5 8675309 2/4/2014 no     right shoe  <NA>    NA  
6 8675309 2/4/2014 no     left  shoe  <NA>    1  
7 8675309 2/4/2014 no     right glove <NA>    NA  
8 8675309 2/4/2014 no     left  glove <NA>    NA  
9 8675309 2/15/2006 no    right shoe none    NA  
10 8675309 2/15/2006 no    left  shoe none    NA  
# i 94 more rows
```

Example: there & back

```
df_superwide ← df_lon ▷  
  group_by(id, date) ▷  
  mutate(id = as.character(id),  
        date = lubridate::mdy(date),  
        seq_id = cur_group_rows()) ▷  
  relocate(seq_id, .after = id) ▷  
  ungroup() ▷  
  pivot_wider(names_from = seq_id, values_from = date:size)
```

Example: there & back

```
df_superwide
```

```
# A tibble: 3 × 625
  id      date_1     date_2     date_3     date_4     date_5     date_6
  <chr>   <date>    <date>    <date>    <date>    <date>    <date>
1 8675309 2009-02-01 2009-02-01 2009-02-01 2009-02-01 2014-02-04 2014-02-04
2 9021033 NA         NA         NA         NA         NA         NA
3 1234567 NA         NA         NA         NA         NA         NA
# i 618 more variables: date_7 <date>, date_8 <date>, date_9 <date>,
#   date_10 <date>, date_11 <date>, date_12 <date>, date_13 <date>,
#   date_14 <date>, date_15 <date>, date_16 <date>, date_17 <date>,
#   date_18 <date>, date_19 <date>, date_20 <date>, date_21 <date>,
#   date_22 <date>, date_23 <date>, date_24 <date>, date_25 <date>,
#   date_26 <date>, date_27 <date>, date_28 <date>, date_29 <date>,
#   date_30 <date>, date_31 <date>, date_32 <date>, date_33 <date>, ...
```

```
# Sheer madness
```

```
colnames(df_superwide)
```

```
[1] "id"      "date_1"   "date_2"   "date_3"   "date_4"   "date_5"   "date_6"
[8] "date_7"   "date_8"   "date_9"   "date_10"  "date_11"  "date_12"  "date_13"
[15] "date_14"  "date_15"  "date_16"  "date_17"  "date_18"  "date_19"  "date_20"
[22] "date_21"  "date_22"  "date_23"  "date_24"  "date_25"  "date_26"  "date_27"
[29] "date_28"  "date_29"  "date_30"  "date_31"  "date_32"  "date_33"  "date_34"
[36] "date_35"  "date_36"  "date_37"  "date_38"  "date_39"  "date_40"  "date_41"
[43] "date_42"  "date_43"  "date_44"  "date_45"  "date_46"  "date_47"  "date_48"
[50] "date_49"  "date_50"  "date_51"  "date_52"  "date_53"  "date_54"  "date_55"
[57] "date_56"  "date_57"  "date_58"  "date_59"  "date_60"  "date_61"  "date_62"
[64] "date_63"  "date_64"  "date_65"  "date_66"  "date_67"  "date_68"  "date_69"
[71] "date_70"  "date_71"  "date_72"  "date_73"  "date_74"  "date_75"  "date_76"
[78] "date_77"  "date_78"  "date_79"  "date_80"  "date_81"  "date_82"  "date_83"
[85] "date_84"  "date_85"  "date_86"  "date_87"  "date_88"  "date_89"  "date_90"
[92] "date_91"  "date_92"  "date_93"  "date_94"  "date_95"  "date_96"  "date_97"
[99] "date_98"  "date_99"  "date_100" "date_101" "date_102" "date_103" "date_104"
```

Nested Data

Example 1: from the `tidyr` Vignette

```
## Examples of recursive lists and nested/split data frames
# install.packages("repurrrsive")
library(repurrrsive)

chars ← tibble(char = got_chars)
chars

# A tibble: 30 × 1
  char
  <list>
  1 <named list [18]>
  2 <named list [18]>
  3 <named list [18]>
  4 <named list [18]>
  5 <named list [18]>
  6 <named list [18]>
  7 <named list [18]>
  8 <named list [18]>
  9 <named list [18]>
  10 <named list [18]>
# i 20 more rows
```

Example 1: from the `tidyr` Vignette

```
chars2 ← chars ▷  
unnest_wider(char)

chars2

# A tibble: 30 × 18
  url          id name gender culture born   died   alive titles aliases father
  <chr>     <int> <chr> <chr>  <chr> <chr> <lgl> <list> <list> <chr>
1 https://w... 1022 Theo... Male   "Ironb..." "In ... ""    TRUE  <chr>  <chr>  ""
2 https://w... 1052 Tyri... Male   ""      "In ... ""    TRUE  <chr>  <chr>  ""
3 https://w... 1074 Vict... Male   "Ironb..." "In ... ""    TRUE  <chr>  <chr>  ""
4 https://w... 1109 Will   Male   ""      ""      "In ... FALSE <chr>  <chr>  ""
5 https://w... 1166 Areo... Male   "Norvo..." "In ... ""    TRUE  <chr>  <chr>  ""
6 https://w... 1267 Chett  Male   ""      "At ... "In ... FALSE <chr>  <chr>  ""
7 https://w... 1295 Cres... Male   ""      ""      "In ... "In ... FALSE <chr>  <chr>  ""
8 https://w... 130 Aria... Female "Dorni..." "In ... ""    TRUE  <chr>  <chr>  ""
9 https://w... 1303 Daen... Female "Valyr..." "In ... ""    TRUE  <chr>  <chr>  ""
10 https://w... 1319 Davo... Male   "Weste..." "In ... ""   TRUE  <chr>  <chr>  ""

# i 20 more rows
# i 7 more variables: mother <chr>, spouse <chr>, allegiances <list>,
#   books <list>, povBooks <list>, tvSeries <list>, playedBy <list>
```

Example 1: from the `tidyr` Vignette

```
chars2 %>  
  select(where(is.list))  
  
# A tibble: 30 × 7  
  titles    aliases   allegiances books    povBooks  tvSeries  playedBy  
  <list>    <list>    <list>     <list>    <list>    <list>    <list>  
1 <chr [2]> <chr [4]> <chr [1]>  <chr [3]> <chr [2]> <chr [6]> <chr [1]>  
2 <chr [2]> <chr [11]> <chr [1]> <chr [2]> <chr [4]> <chr [6]> <chr [1]>  
3 <chr [2]> <chr [1]> <chr [1]>  <chr [3]> <chr [2]> <chr [1]> <chr [1]>  
4 <chr [1]> <chr [1]> <NULL>    <chr [1]> <chr [1]> <chr [1]> <chr [1]>  
5 <chr [1]> <chr [1]> <chr [1]>  <chr [3]> <chr [2]> <chr [2]> <chr [1]>  
6 <chr [1]> <chr [1]> <NULL>    <chr [2]> <chr [1]> <chr [1]> <chr [1]>  
7 <chr [1]> <chr [1]> <NULL>    <chr [2]> <chr [1]> <chr [1]> <chr [1]>  
8 <chr [1]> <chr [1]> <chr [1]>  <chr [4]> <chr [1]> <chr [1]> <chr [1]>  
9 <chr [5]> <chr [11]> <chr [1]> <chr [1]> <chr [4]> <chr [6]> <chr [1]>  
10 <chr [4]> <chr [5]> <chr [2]> <chr [1]> <chr [3]> <chr [5]> <chr [1]>  
# i 20 more rows
```

Example 1: from the `tidyr` Vignette

A row for every book and TV series that the character appears in:

```
chars2
```

```
# A tibble: 30 × 18
  url          id name  gender culture born   died   alive titles aliases father
  <chr>        <int> <chr> <chr>  <chr> <chr> <lgl> <list> <list>  <chr>
1 https://w... 1022 Theo... Male   "Ironb..." "In ... ""    TRUE  <chr>  <chr>  ""
2 https://w... 1052 Tyri... Male   ""      "In ... ""    TRUE  <chr>  <chr>  ""
3 https://w... 1074 Vict... Male   "Ironb..." "In ... ""    TRUE  <chr>  <chr>  ""
4 https://w... 1109 Will   Male   ""      ""      "In ... FALSE <chr>  <chr>  ""
5 https://w... 1166 Areo... Male   "Norvo..." "In ... ""    TRUE  <chr>  <chr>  ""
6 https://w... 1267 Chett  Male   ""      "At ... "In ... FALSE <chr>  <chr>  ""
7 https://w... 1295 Cres... Male   ""      "In ... "In ... FALSE <chr>  <chr>  ""
8 https://w... 130   Aria... Female "Dorni..." "In ... ""    TRUE  <chr>  <chr>  ""
9 https://w... 1303 Daen... Female "Valyr..." "In ... ""    TRUE  <chr>  <chr>  ""
10 https://w... 1319 Davo... Male   "Weste..." "In ... ""   TRUE  <chr>  <chr>  ""
# i 20 more rows
# i 7 more variables: mother <chr>, spouse <chr>, allegiances <list>,
#   books <list>, povBooks <list>, tvSeries <list>, playedBy <list>
```

Example 1: Vignette

```
chars2
```

```
# A tibble: 30 × 18
  url          id name gender culture born   died alive titles aliases father
  <chr>        <int> <chr> <chr>  <chr> <chr> <lgl> <list> <list> <chr>
1 https://w... 1022 Theo... Male  "Ironb..." "In ..." TRUE  <chr> <chr>  ""
2 https://w... 1052 Tyri... Male  ""      "In ..." TRUE  <chr> <chr>  ""
3 https://w... 1074 Vict... Male  "Ironb..." "In ..." TRUE  <chr> <chr>  ""
4 https://w... 1109 Will  Male  ""      ""     "In ..." FALSE <chr> <chr>  ""
5 https://w... 1166 Areo... Male  "Norvo..." "In ..." TRUE  <chr> <chr>  ""
6 https://w... 1267 Chett Male  ""      "At ..." "In ..." FALSE <chr> <chr>  ""
7 https://w... 1295 Cres... Male  ""      "In ..." "In ..." FALSE <chr> <chr>  ""
8 https://w... 130 Aria... Female "Dorni..." "In ..." TRUE  <chr> <chr>  ""
9 https://w... 1303 Daen... Female "Valyr..." "In ..." TRUE  <chr> <chr>  ""
10 https://w... 1319 Davo... Male  "Weste..." "In ..." TRUE  <chr> <chr>  ""
# i 20 more rows
# i 7 more variables: mother <chr>, spouse <chr>, allegiances <list>,
#   books <list>, povBooks <list>, tvSeries <list>, playedBy <list>
```

Example 1: Vignette

```
chars2 %>  
  select(name, books, tvSeries)
```

```
# A tibble: 30 × 3  
  name          books      tvSeries  
  <chr>        <list>     <list>  
1 Theon Greyjoy <chr [3]> <chr [6]>  
2 Tyrion Lannister <chr [2]> <chr [6]>  
3 Victarion Greyjoy <chr [3]> <chr [1]>  
4 Will          <chr [1]> <chr [1]>  
5 Aroo Hotah   <chr [3]> <chr [2]>  
6 Chett         <chr [2]> <chr [1]>  
7 Cressen       <chr [2]> <chr [1]>  
8 Arianne Martell <chr [4]> <chr [1]>  
9 Daenerys Targaryen <chr [1]> <chr [6]>  
10 Davos Seaworth <chr [1]> <chr [5]>  
# i 20 more rows
```

Example 1: Vignette

```
chars2 %>  
  select(name, books, tvSeries) %>  
  pivot_longer(c(books, tvSeries),  
               names_to = "media",  
               values_to = "value")
```

```
# A tibble: 60 × 3  
  name      media    value  
  <chr>     <chr>   <list>  
1 Theon Greyjoy books   <chr [3]>  
2 Theon Greyjoy tvSeries <chr [6]>  
3 Tyrion Lannister books   <chr [2]>  
4 Tyrion Lannister tvSeries <chr [6]>  
5 Victarion Greyjoy books   <chr [3]>  
6 Victarion Greyjoy tvSeries <chr [1]>  
7 Will      books   <chr [1]>  
8 Will      tvSeries <chr [1]>  
9 Areo Hotah books   <chr [3]>  
10 Areo Hotah tvSeries <chr [2]>  
# i 50 more rows
```

Example 1: Vignette

```
chars2 %>  
  select(name, books, tvSeries) %>  
  pivot_longer(c(books, tvSeries),  
               names_to = "media",  
               values_to = "value") %>  
  unnest_longer(value)
```

```
# A tibble: 179 × 3  
  name          media    value  
  <chr>        <chr>    <chr>  
1 Theon Greyjoy books   A Game of Thrones  
2 Theon Greyjoy books   A Storm of Swords  
3 Theon Greyjoy books   A Feast for Crows  
4 Theon Greyjoy tvSeries Season 1  
5 Theon Greyjoy tvSeries Season 2  
6 Theon Greyjoy tvSeries Season 3  
7 Theon Greyjoy tvSeries Season 4  
8 Theon Greyjoy tvSeries Season 5  
9 Theon Greyjoy tvSeries Season 6  
10 Tyrion Lannister books   A Feast for Crows  
# i 169 more rows
```

Example 2: GitHub

Example 2: GitHub

The `fromJSON()` function in `{jsonlite}` does its best to simplify what the API returns into a table, which you can convert to a tibble.

```
# install.packages("jsonlite")
jsonlite::fromJSON("https://api.github.com/users/kjhealy/repos") %>
  as_tibble()

# A tibble: 30 × 79
   id node_id name  full_name private owner$login html_url description fork
   <int> <chr>  <chr> <lgl>    <chr>      <chr>      <chr>      <lgl>
1 4.37e8 R_kgDO... .doo... kjhealy/... FALSE    kjhealy    https://... Doom Emacs... FALSE
2 5.37e7 MDEwOl... 2016... kjhealy/... FALSE    kjhealy    https://... <NA>        TRUE
3 5.10e6 MDEwOl... 5by5... kjhealy/... FALSE    kjhealy    https://... Data and R... FALSE
4 7.63e8 R_kgDO... abor... kjhealy/... FALSE    kjhealy    https://... <NA>        FALSE
5 8.97e8 R_kgDO... adve... kjhealy/... FALSE    kjhealy    https://... <NA>        FALSE
6 1.11e9 R_kgDO... adve... kjhealy/... FALSE    kjhealy    https://... <NA>        FALSE
7 3.48e7 MDEwOl... apple kjhealy/... FALSE    kjhealy    https://... Trend plot... FALSE
8 2.59e8 MDEwOl... appl... kjhealy/... FALSE    kjhealy    https://... <NA>        FALSE
9 1.56e6 MDEwOl... asa... kjhealy/... FALSE    kjhealy    https://... Comparativ... FALSE
10 4.65e7 MDEwOl... asa... kjhealy/... FALSE   kjhealy    https://... Some plots... FALSE
# i 20 more rows
# i 88 more variables: owner$id <int>, $node_id <chr>, $avatar_url <chr>,
#   $gravatar_id <chr>, $url <chr>, $html_url <chr>, $followers_url <chr>,
#   $following_url <chr>, $gists_url <chr>, $starred_url <chr>,
#   $subscriptions_url <chr>, $organizations_url <chr>, $repos_url <chr>,
#   $events_url <chr>, $received_events_url <chr>, $type <chr>,
```

Example 2: GitHub

The `read_json()` function in `{jsonlite}` gives you a list of the JSON the API returns, which you won't be able to immediately convert.

```
gh_raw ← jsonlite::read_json("https://api.github.com/users/kjhealy/repos")

gh_tb ← tibble(gh = gh_raw)

gh_tb

# A tibble: 30 × 1
  gh
  <list>
  1 <named list [79]>
  2 <named list [79]>
  3 <named list [79]>
  4 <named list [79]>
  5 <named list [79]>
  6 <named list [79]>
  7 <named list [79]>
  8 <named list [79]>
  9 <named list [79]>
  10 <named list [79]>
# i 20 more rows
```

Example 2: GitHub

This is what the `unnest_wider()` function is for:

```
gh_tb >
  unnest_wider(gh)

# A tibble: 30 × 79
      id node_id   name full_name private owner          html_url description
      <int> <chr>    <chr> <chr>     <lgl>   <list>        <chr>    <chr>
1 436805268 R_kgDOG... .doo... kjhealy/... FALSE  <named list> https://... Doom Emacs...
2 53698061 MDEwOlJ... 2016... kjhealy/... FALSE  <named list> https://... <NA>
3 5103336 MDEwOlJ... 5by5... kjhealy/... FALSE  <named list> https://... Data and R...
4 762813356 R_kgDOL... abor... kjhealy/... FALSE  <named list> https://... <NA>
5 897474513 R_kgDON... adve... kjhealy/... FALSE  <named list> https://... <NA>
6 1108578379 R_kgDOQ... adve... kjhealy/... FALSE <named list> https://... <NA>
7 34824505 MDEwOlJ... apple kjhealy/... FALSE  <named list> https://... Trend plot...
8 259012888 MDEwOlJ... appl... kjhealy/... FALSE <named list> https://... <NA>
9 1555513 MDEwOlJ... asa-... kjhealy/... FALSE  <named list> https://... Comparativ...
10 46535044 MDEwOlJ... asa-... kjhealy/... FALSE <named list> https://... Some plots...
# i 20 more rows
# i 71 more variables: fork <lgl>, url <chr>, forks_url <chr>, keys_url <chr>,
#   collaborators_url <chr>, teams_url <chr>, hooks_url <chr>,
#   issue_events_url <chr>, events_url <chr>, assignees_url <chr>,
#   branches_url <chr>, tags_url <chr>, blobs_url <chr>, git_tags_url <chr>,
#   git_refs_url <chr>, trees_url <chr>, statuses_url <chr>,
```

Example 2: GitHub

By default we only get the first 30 items back. (The API is paginated.)

```
gh_tb >
  unnest_wider(gh) >
  pull(name)

[1] ".doom.d"          "2016-03-wapo-uber" "5by5-figures"
[4] "abortion_gss"     "advent24"           "advent25"
[7] "apple"             "apple_covid_post"  "asa-dues"
[10] "asa-sections"    "asa_sections17"   "asdfree"
[13] "assault-2023"    "assault-deaths"   "atpfm"
[16] "babcock-ratings" "bass_graphs"      "beepsays.com"
[19] "bib"               "bookdown"          "bookdown-demo"
[22] "boom"              "boomers"          "bootstrap"
[25] "canmap"            "cavax"            "cbofigure"
[28] "cdccovidview"     "congress"         "corr_example"
```

GitHub

```
gh_tb >
  unnest_wider(gh) >
  select(id, name, ends_with("count")) >
  arrange(desc(watchers_count))

# A tibble: 30 × 6
    id      name stargazers_count watchers_count forks_count open_issues_count
  <int>    <chr>        <int>        <int>        <int>        <int>
1 34824505 apple            27           27          25           0
2 259012888 appl...          13           13           4           1
3 5249003 assa...          11           11           4           0
4 128972396 boom            9            9           1           0
5 114724 bib               7            7           2           0
6 621306299 assa...          6            6           0           0
7 160963411 canm...          5            5           2           0
8 216038719 cong...          4            4           0           0
9 148529869 asa_...          2            2           1           0
10 35690047 atpfm           2            2           2           0
# i 20 more rows
```

Example 3: the NY Citibike API

NYC CitiBikes

Many cities have Bike Share programs. Many of those provide data according to the [GBFS specification](#). A spec like this is a set of rules that says “If you adhere to this spec, you will provide data in the following consistent way”, where this includes rules about the data format and the endpoints or URLs where that data can be found.

NYC CitiBikes

The spec has a rule saying “Provide a feed specifying the data available”.

```
gbfsurl ← "https://gbfs.citibikenyc.com/gbfs/2.3/gbfs.json"
feeds ← jsonlite::fromJSON(gbfsurl)
str(feeds)

List of 4
$ data      :List of 3
..$ en:List of 1
...$ feeds:'data.frame': 12 obs. of  2 variables:
... ..$ url : chr [1:12] "https://gbfs.lyft.com/gbfs/2.3/bkn/gbfs.json" "https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_information.json"
"https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_information.json" "https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_status.json" ...
... ..$ name: chr [1:12] "gbfs" "system_information" "station_information" "station_status" ...
..$ fr:List of 1
...$ feeds:'data.frame': 12 obs. of  2 variables:
... ..$ url : chr [1:12] "https://gbfs.lyft.com/gbfs/2.3/bkn/gbfs.json" "https://gbfs.lyft.com/gbfs/2.3/bkn/fr/system_information.json"
"https://gbfs.lyft.com/gbfs/2.3/bkn/fr/station_information.json" "https://gbfs.lyft.com/gbfs/2.3/bkn/fr/station_status.json" ...
... ..$ name: chr [1:12] "gbfs" "system_information" "station_information" "station_status" ...
..$ es:List of 1
...$ feeds:'data.frame': 12 obs. of  2 variables:
... ..$ url : chr [1:12] "https://gbfs.lyft.com/gbfs/2.3/bkn/gbfs.json" "https://gbfs.lyft.com/gbfs/2.3/bkn/es/system_information.json"
"https://gbfs.lyft.com/gbfs/2.3/bkn/es/station_information.json" "https://gbfs.lyft.com/gbfs/2.3/bkn/es/station_status.json" ...
... ..$ name: chr [1:12] "gbfs" "system_information" "station_information" "station_status" ...
$ last_updated: int 1767724749
$ ttl        : int 60
$ version    : chr "2.3"
```

This comes in to us a *nested list*.

NYC CitiBikes

JSON data is typically nested. It can look really complex at first glance. This is one reason to read the spec. We can put the feed into a tibble if we like:

```
feeds_df ← as_tibble(feeds)
feeds_df

# A tibble: 3 × 4
  data      last_updated    ttl version
  <named list>     <int> <int> <chr>
1 <named list [1]> 1767724749    60 2.3
2 <named list [1]> 1767724749    60 2.3
3 <named list [1]> 1767724749    60 2.3
```

RStudio's object viewer is a good way to explore the hierarchical structure of unfamiliar lists.

NYC CitiBikes

If we explore or look at the spec we see that each row provides the same feed information in English, Spanish, or French. We can slice out the English feed and look at what's in it:

```
feeds_df ▷  
  slice(1) ▷  
  unnest(data) ▷ # It's two levels down  
  unnest(data)  
  
# A tibble: 12 × 5  
  url                               name last_updated   ttl version  
  <chr>                            <chr>      <int>    <int> <chr>  
1 https://gbfs.lyft.com/gbfs/2.3/bkn/gbfs.json gbfs     1767724749  60  2.3  
2 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system/gbfs.json syst... 1767724749  60  2.3  
3 https://gbfs.lyft.com/gbfs/2.3/bkn/en/station/gbfs.json stat... 1767724749  60  2.3  
4 https://gbfs.lyft.com/gbfs/2.3/bkn/en/stationinfo/gbfs.json stat... 1767724749  60  2.3  
5 https://gbfs.lyft.com/gbfs/2.3/bkn/en/free_bike_station/gbfs.json free... 1767724749  60  2.3  
6 https://gbfs.lyft.com/gbfs/2.3/bkn/en/systemstatus/gbfs.json syst... 1767724749  60  2.3  
7 https://gbfs.lyft.com/gbfs/2.3/bkn/en/systemstatus/gbfs.json syst... 1767724749  60  2.3  
8 https://gbfs.lyft.com/gbfs/2.3/bkn/en/systemstatus/gbfs.json syst... 1767724749  60  2.3  
9 https://gbfs.lyft.com/gbfs/2.3/bkn/en/systemstatus/gbfs.json syst... 1767724749  60  2.3  
10 https://gbfs.lyft.com/gbfs/2.3/bkn/en/systemstatus/gbfs.json syst... 1767724749  60  2.3  
11 https://gbfs.lyft.com/gbfs/2.3/bkn/en/gbfs_stations/gbfs.json gbfs... 1767724749  60  2.3  
12 https://gbfs.lyft.com/gbfs/2.3/bkn/en/vehicle/gbfs.json vehi... 1767724749  60  2.3
```

More urls! Let's extract the station data feed.

NYC CitiBikes

```
nyc_stations_url ← feeds_df ▷  
  slice(1) ▷ unnest(data) ▷ unnest(data) ▷  
  filter(name == "station_information") ▷ pull(url)  
  
nyc_stations_url  
  
[1] "https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_information.json"
```

NYC CitiBikes

That was tedious. If we know *precisely* what we're after we can do it faster by plumbing down through the list:

```
# Base R style  
feeds_df[[1]]$en$feeds$url[3]  
  
[1] "https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_information.json"
```

Or with `pluck()`:

```
# Pluck by element number or name  
feeds_df %> pluck(1, "en", "feeds", "url", 3)  
  
[1] "https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_information.json"
```

You can find these index locations via the Object Inspector.

NYC CitiBikes

Or with some combination of methods:

```
feeds_df >
  pluck(1, "en", "feeds") >
  as_tibble()

# A tibble: 12 × 2
  url                                     name
  <chr>
  1 https://gbfs.lyft.com/gbfs/2.3/bkn/gbfs.json      gbfs
  2 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_information.json system_infor...
  3 https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_information.json station_info...
  4 https://gbfs.lyft.com/gbfs/2.3/bkn/en/station_status.json station_stat...
  5 https://gbfs.lyft.com/gbfs/2.3/bkn/en/free_bike_status.json free_bike_st...
  6 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_hours.json system_hours
  7 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_calendar.json system_calen...
  8 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_regions.json system_regio...
  9 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_pricing_plans.json system_prici...
 10 https://gbfs.lyft.com/gbfs/2.3/bkn/en/system_alerts.json system_alerts
 11 https://gbfs.lyft.com/gbfs/2.3/bkn/en/gbfs_versions.json gbfs_versions
 12 https://gbfs.lyft.com/gbfs/2.3/bkn/en/vehicle_types.json vehicle_types
```

Then work as you would with a tibble.

NYC CitiBikes

```
station_status_url ← feeds_df %> pluck(1, "en", "feeds") %>
  filter(name = "station_status") %> pull(url)

station_status_df ← jsonlite::fromJSON(station_status_url)

str(station_status_df) # Still a list

List of 4
$ data      :List of 1
..$ stations:'data.frame': 2321 obs. of 13 variables:
.. ..$ num_docks_available    : int [1:2321] 0 0 0 0 0 0 0 0 7 5 ...
.. ..$ is_returning          : int [1:2321] 0 0 0 0 0 0 0 0 0 1 ...
.. ..$ num_docks_disabled    : int [1:2321] 0 0 0 0 0 0 0 0 0 1 ...
.. ..$ last_reported         : int [1:2321] 1767032075 86400 86400 86400 86400 86400 86400 86400 1759761514
1767611347 1767724493 ...
.. ..$ is_installed          : int [1:2321] 0 0 0 0 0 0 0 0 1 1 ...
.. ..$ vehicle_types_available :List of 2321
.. ... ..$ : 'data.frame': 2 obs. of 2 variables:
.. ... ... ..$ count        : int [1:2] 0 0
.. ... ... ..$ vehicle_type_id: chr [1:2] "1" "2"
.. ... ..$ : 'data.frame': 2 obs. of 2 variables:
.. ... ... ..$ count        : int [1:2] 0 0
.. ... ... ..$ vehicle_type_id: chr [1:2] "1" "2"
.. ... ..$ : 'data.frame': 2 obs. of 2 variables:
.. ... ... ..$ count        : int [1:2] 0 0
.. ... ... ..$ vehicle_type_id: chr [1:2] "1" "2"
```

Like the overall feeds data, the station status data is a nested list with information on when it was last updated and the software version as well as the

NYC CitiBikes

```
station_status_df ← station_status_df$data$stations ▷  
  as_tibble()  
  
station_status_df  
  
# A tibble: 2,321 × 13  
#>   num_docks_available is_returning num_docks_disabled last_reported  
#>   <int>            <int>           <int>           <int>  
#> 1             0            0              0      1767032075  
#> 2             0            0              0       86400  
#> 3             0            0              0       86400  
#> 4             0            0              0       86400  
#> 5             0            0              0       86400  
#> 6             0            0              0       86400  
#> 7             0            0              0       86400  
#> 8             0            0              0      1759761514  
#> 9             7            0              0      1767611347  
#> 10            5            1              1      1767724493  
#> # i 2,311 more rows  
#> # i 9 more variables: is_installed <int>, vehicle_types_available <list>,  
#> #   num_bikes_available <int>, num_ebikes_available <int>, station_id <chr>,  
#> #   is_renting <int>, num_bikes_disabled <int>, num_scooters_available <int>,  
#> #   num_scooters_unavailable <int>
```

As you can see there is more nested data in here too, in the column about the types of bike and scooter models available.

NYC CitiBikes

Let's do the same for the feed of static information about stations.

```
station_info_url ← feeds_df ▷ pluck(1, "en", "feeds") ▷  
  filter(name = "station_information") ▷ pull(url)  
  
station_info_df ← station_info_url ▷  
  jsonlite::fromJSON() ▷  
  pluck("data", "stations") ▷  
  as_tibble()  
  
station_info_df  
  
# A tibble: 2,321 × 8  
  lon short_name rental_uris$android    $ios      lat name capacity station_id  
  <dbl> <chr>          <chr>      <dbl> <chr>   <int> <chr>  
1 -73.9 5043.06 https://bkn.lft.to/la... http... 40.7 Suyd...     17 498d7e8e-...  
2 -74.0 5736.09 https://bkn.lft.to/la... http... 40.7 E 14...     43 66dc7ba2-...  
3 -73.9 6789.20 https://bkn.lft.to/la... http... 40.8 31 S...     35 190583724...  
4 -74.0 6569.08 https://bkn.lft.to/la... http... 40.8 W 35...     41 a47f77a2-...  
5 -73.9 6325.01 https://bkn.lft.to/la... http... 40.7 43 S...     19 41495491-...  
6 -74.0 7323.09 https://bkn.lft.to/la... http... 40.8 5 Av...     39 b9720921-...  
7 -74.0 4829.01 https://bkn.lft.to/la... http... 40.7 Colu...     37 66db3687-...  
8 -74.0 6708.04 https://bkn.lft.to/la... http... 40.8 Broa...    103 64f0f28c-...  
9 -74.0 3034.02 https://bkn.lft.to/la... http... 40.6 Matt...     21 182266303...  
10 -74.0 2009.04 https://bkn.lft.to/la... http... 40.6 Shor...    12 212442630...  
# i 2,311 more rows  
# i 1 more variable: region_id <chr>
```

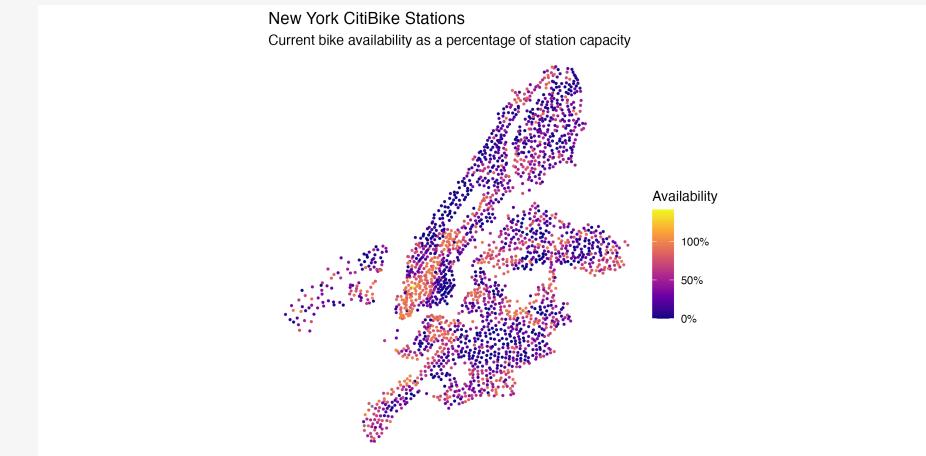
NYC CityBikes

We can join these by `station_id`

```
stations_df ← station_status_df ▷  
  left_join(station_info_df, by = "station_id") ▷  
  relocate(name, capacity, num_bikes_available, lat, lon)  
  
stations_df  
  
# A tibble: 2,321 × 20  
  name           capacity num_bikes_available   lat    lon num_docks_available  
  <chr>          <int>            <int> <dbl> <dbl>             <int>  
1 Broadway & W 48...     103                 0  40.8 -74.0                  0  
2 43 St & Skillma...      19                  0  40.7 -73.9                  0  
3 31 St & Broadway      35                  0  40.8 -73.9                  0  
4 W 35 St & Dyer ...     41                  0  40.8 -74.0                  0  
5 E 14 St & Ave B       43                  0  40.7 -74.0                  0  
6 5 Ave & E 87 St      39                  0  40.8 -74.0                  0  
7 Columbia Height...     37                  0  40.7 -74.0                  0  
8 Suydam St & St ...     17                  0  40.7 -73.9                  0  
9 Matthews Ct & C...     21                  0  40.6 -74.0                  7  
10 Shore Rd & 4 Ave     12                  5  40.6 -74.0                  5  
# i 2,311 more rows  
# i 14 more variables: is_returning <int>, num_docks_disabled <int>,  
#   last_reported <int>, is_installed <int>, vehicle_types_available <list>,  
#   num_ebikes_available <int>, station_id <chr>, is_renting <int>,  
#   num_bikes_disabled <int>, num_scooters_available <int>,  
#   num_scooters_unavailable <int>, short_name <chr>, rental_uris <df[,2]>,
```

NYC CityBikes

```
stations_df >
  ggplot(aes(x = lon,
             y = lat,
             color = num_bikes_available/capacity)) +
  geom_point(size = 0.5) +
  scale_color_viridis_c(option = "plasma",
                        labels = scales::percent_format()) +
  coord_equal() +
  labs(color = "Availability",
       title = "New York CitiBike Stations",
       subtitle = "Current bike availability as a percentage of station
theme_void()
```



NYC CitiBikes

Finally, the “last updated” tag is a good old Unix time number expressed in seconds since 1970:

```
feeds_df$last_updated[1]
```

```
[1] 1767724749
```

```
as_datetime(feeds_df$last_updated)[1]
```

```
[1] "2026-01-06 18:39:09 UTC"
```