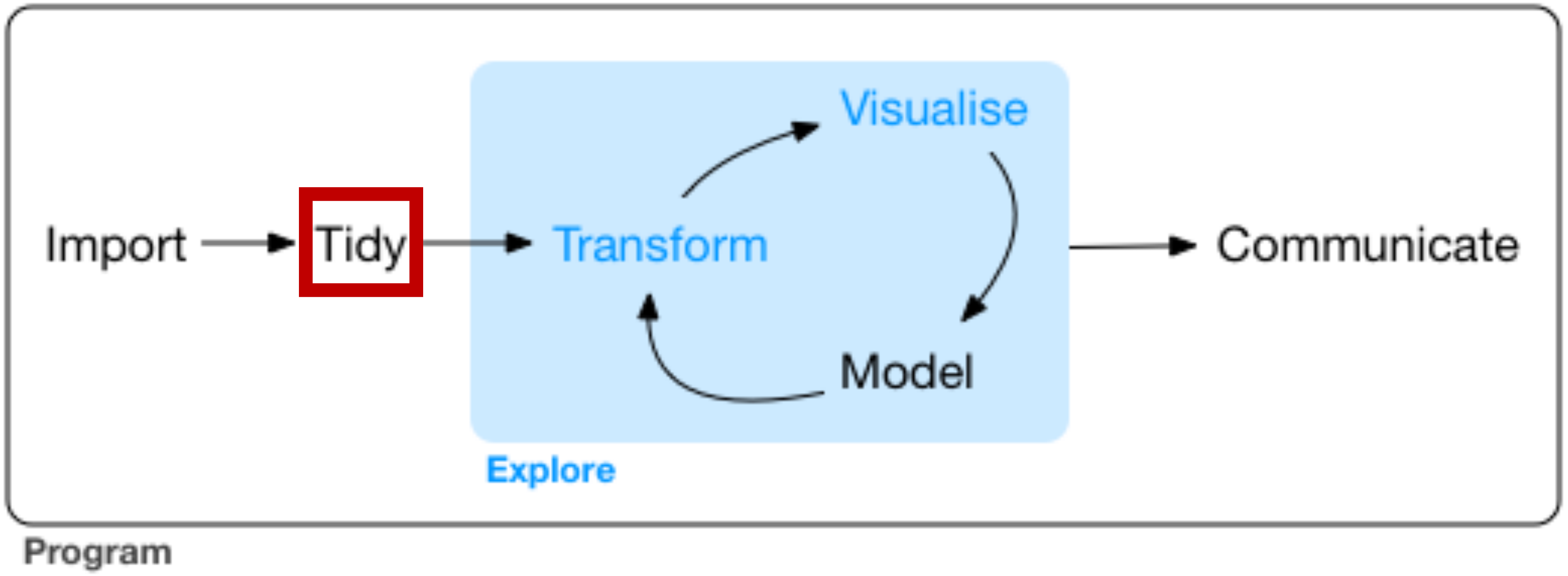
Tidy Data:
Consistency is the spice of life

McKinzie Garrison

R Ladies Baltimore

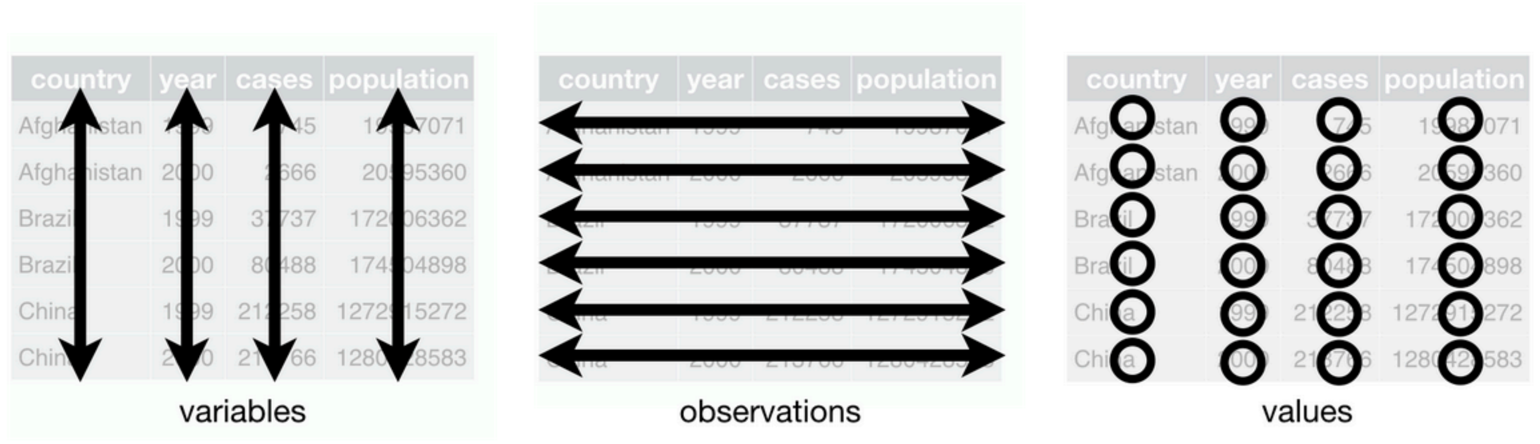
June 17, 2020

General Workflow



Grolemund, Wickham. R for Data Science. https://r4ds.had.co.nz/index.html

Tidy Data



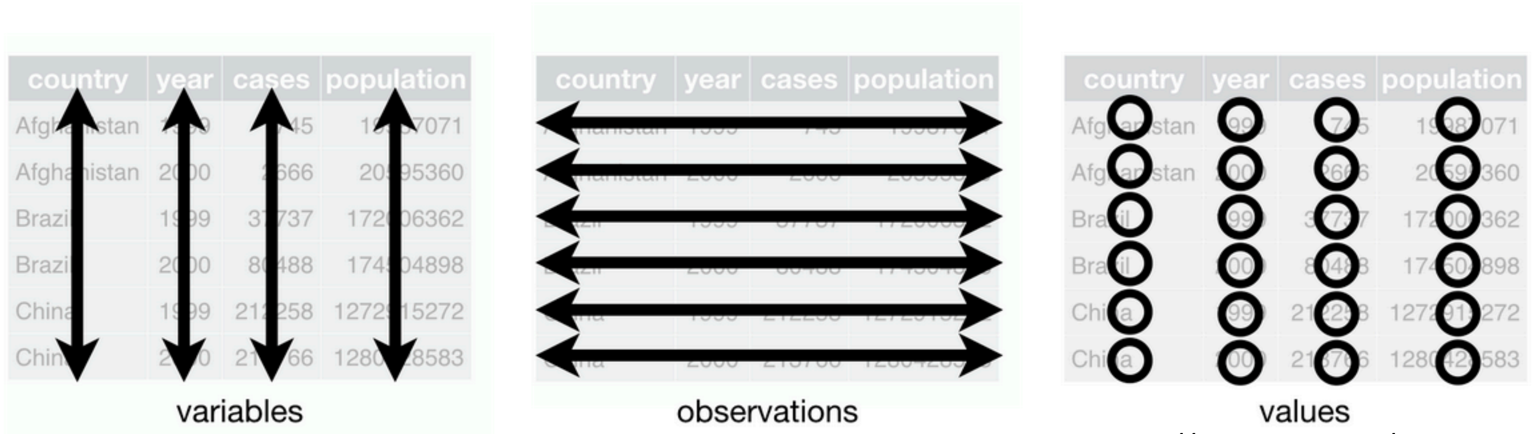
There are three interrelated rules which make a dataset tidy:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

Each table is a type of observational unit. – Hadley Wickham.

Grolemund, Wickham. R for Data Science. https://r4ds.had.co.nz/index.html

Tidy Data



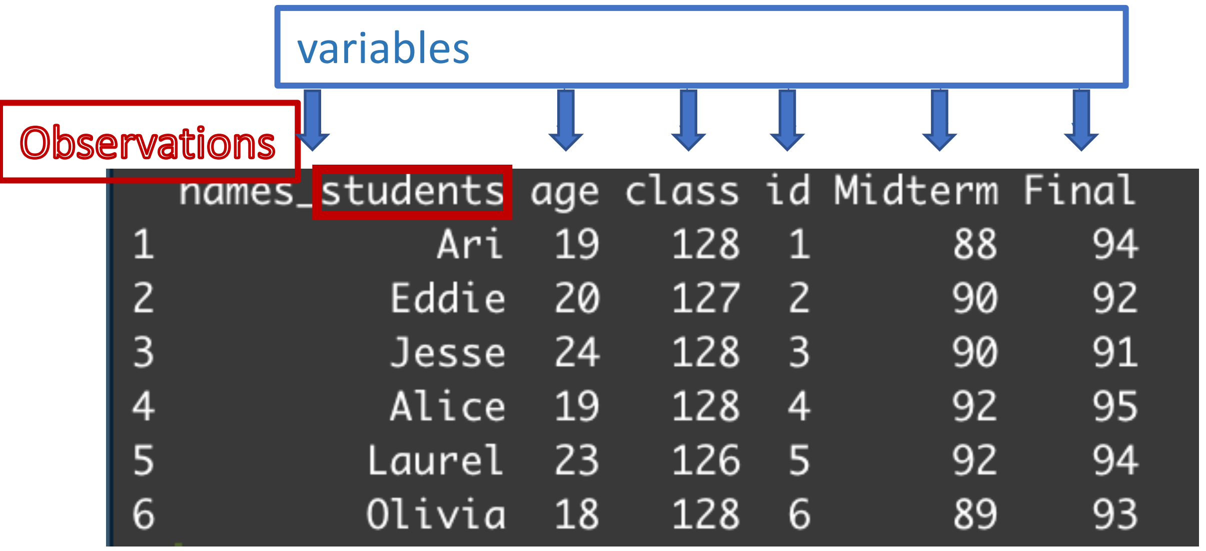
Grolemund, Wickham. R for Data Science. https://r4ds.had.co.nz/index.html

<u>Observation</u>: a case can be described by the variable values.
- Examples:
  - Sample
  - Patient
  - Region
  - Model type
  - Alignment read

<u>Variable</u>: values describing an attribute.
- Examples:
  - temperature
  - quality score
  - height
  - time
  - position

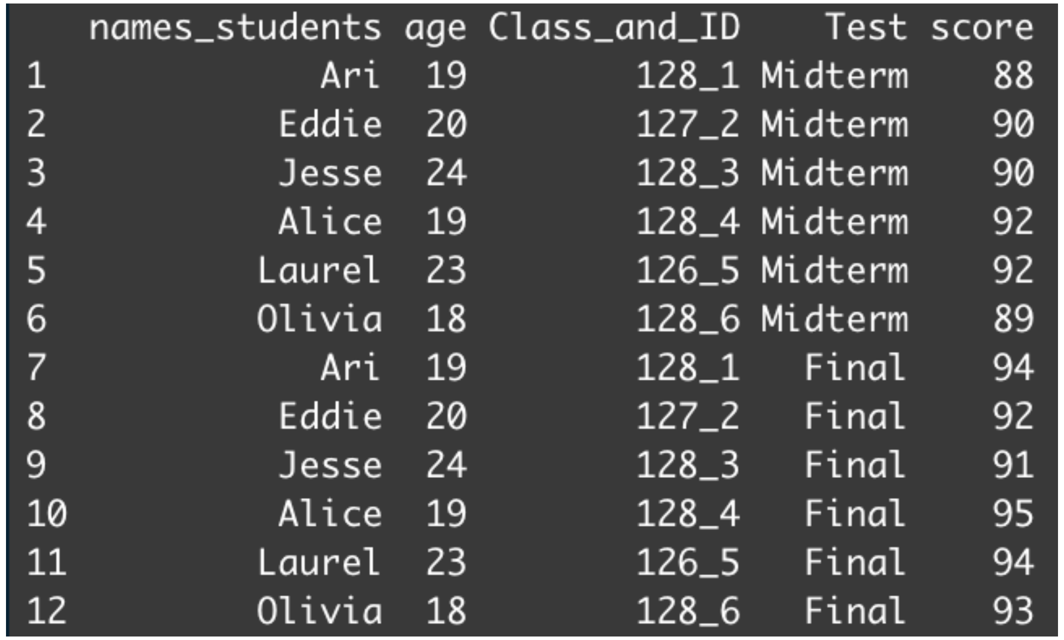
Example – Tidy vs Untidy

Tidy Table



```
  names_students age class id Midterm Final
1            Ari  19   128  1      88    94
2          Eddie  20   127  2      90    92
3          Jesse  24   128  3      90    91
4          Alice  19   128  4      92    95
5         Laurel  23   126  5      92    94
6         Olivia  18   128  6      89    93
```

Untidy Table

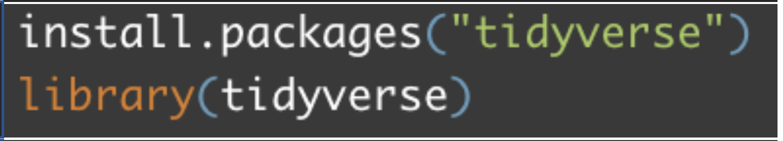


```
   names_students age Class_and_ID    Test score
1             Ari  19        128_1 Midterm    88
2           Eddie  20        127_2 Midterm    90
3           Jesse  24        128_3 Midterm    90
4           Alice  19        128_4 Midterm    92
5          Laurel  23        126_5 Midterm    92
6          Olivia  18        128_6 Midterm    89
7             Ari  19        128_1   Final    94
8           Eddie  20        127_2   Final    92
9           Jesse  24        128_3   Final    91
10          Alice  19        128_4   Final    95
11         Laurel  23        126_5   Final    94
12         Olivia  18        128_6   Final    93
```

Relevant Packages

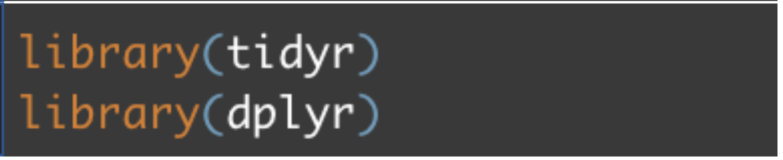
Load your relevant libraries:

A



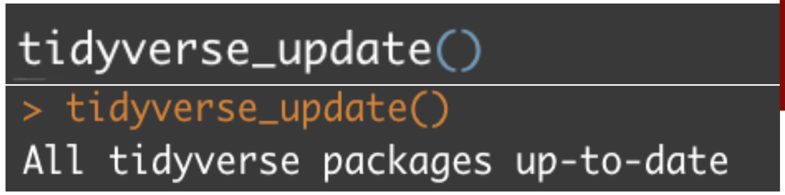
```
install.packages("tidyverse")
library(tidyverse)
```

B



```
library(tidyr)
library(dplyr)
```

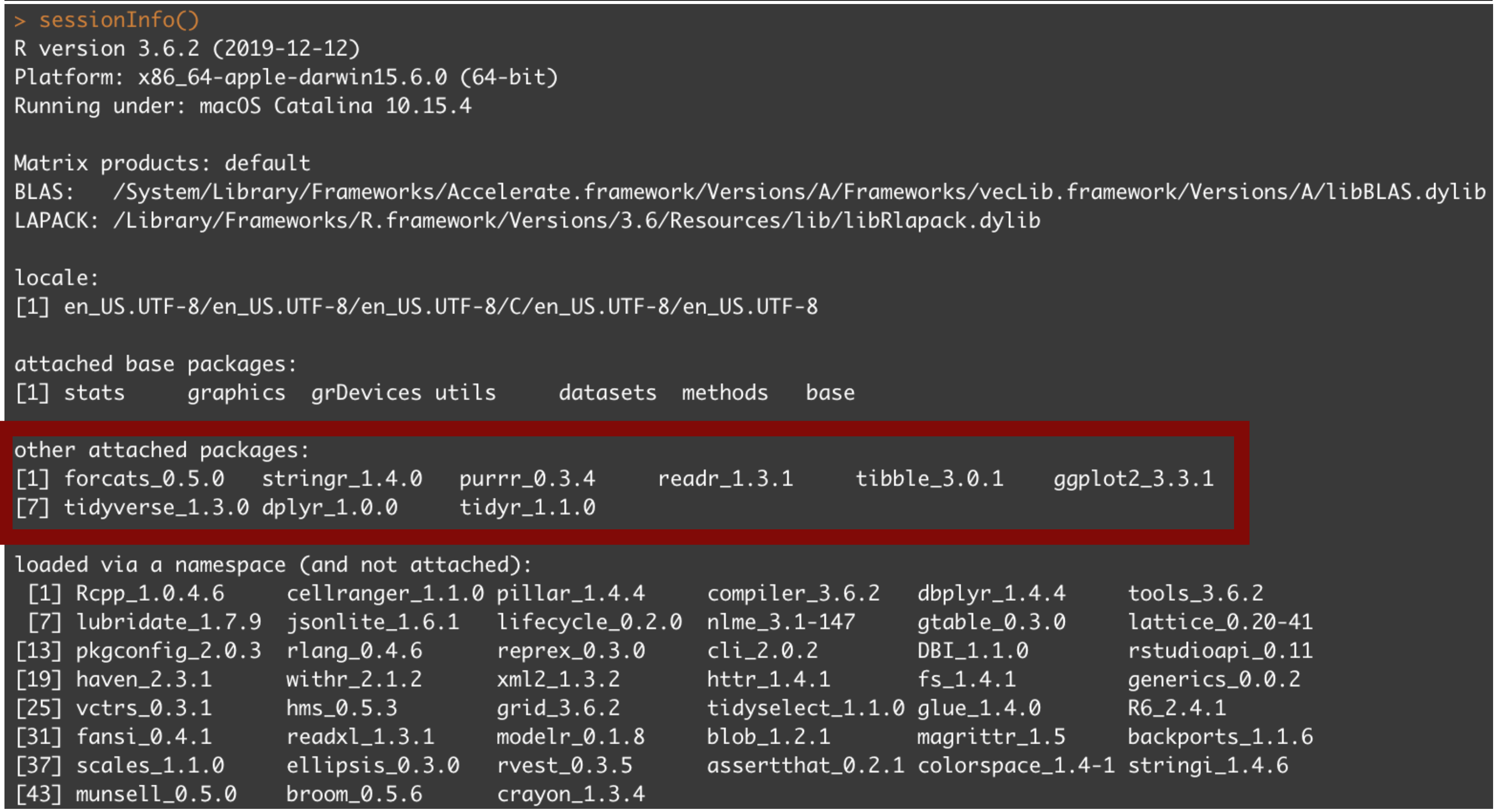
Check for updates:



```
tidyverse_update()
```

```
> tidyverse_update()
All tidyverse packages up-to-date
```

Session Info:

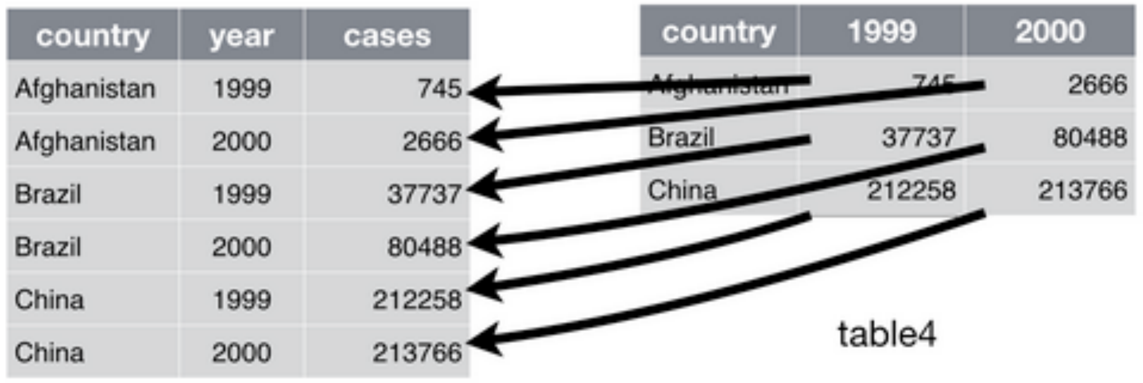


```
> sessionInfo()
R version 3.6.2 (2019-12-12)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Catalina 10.15.4

Matrix products: default
BLAS:   /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBLAS.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] forcats_0.5.0   stringr_1.4.0   purrr_0.3.4     readr_1.3.1     tibble_3.0.1    ggplot2_3.3.1
[7] tidyverse_1.3.0 dplyr_1.0.0     tidyr_1.1.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.4.6     cellranger_1.1.0 pillar_1.4.4     compiler_3.6.2   dbplyr_1.4.4     tools_3.6.2
 [7] lubridate_1.7.9  jsonlite_1.6.1   lifecycle_0.2.0  nlme_3.1-147     gtable_0.3.0     lattice_0.20-41
[13] pkgconfig_2.0.3  rlang_0.4.6      reprex_0.3.0     cli_2.0.2        DBI_1.1.0        rstudioapi_0.11
[19] haven_2.3.1      withr_2.1.2      xml2_1.3.2       httr_1.4.1       fs_1.4.1         generics_0.0.2
[25] vctrs_0.3.1      hms_0.5.3        grid_3.6.2       tidyselect_1.1.0 glue_1.4.0       R6_2.4.1
[31] fansi_0.4.1      readxl_1.3.1     modelr_0.1.8     blob_1.2.1       magrittr_1.5     backports_1.1.6
[37] scales_1.1.0     ellipsis_0.3.0   rvest_0.3.5      assertthat_0.2.1 colorspace_1.4-1 stringi_1.4.6
[43] munsell_0.5.0    broom_0.5.6      crayon_1.3.4
```

Tidying Functions

Data Wrangling Functions:	
pivot_longer()	pivot column names into new column and values of those columns into separate column
pivot_wider()	observations in multiple rows must be separated to new columns.
separate()	separate one column into multiple columns using a character for the split.
unite()	combine multiple columns into one column.
complete()	make implicit missing values into explicit NAs.
fill()	fill in NAs with prior non-NA value.

pivot_longer()

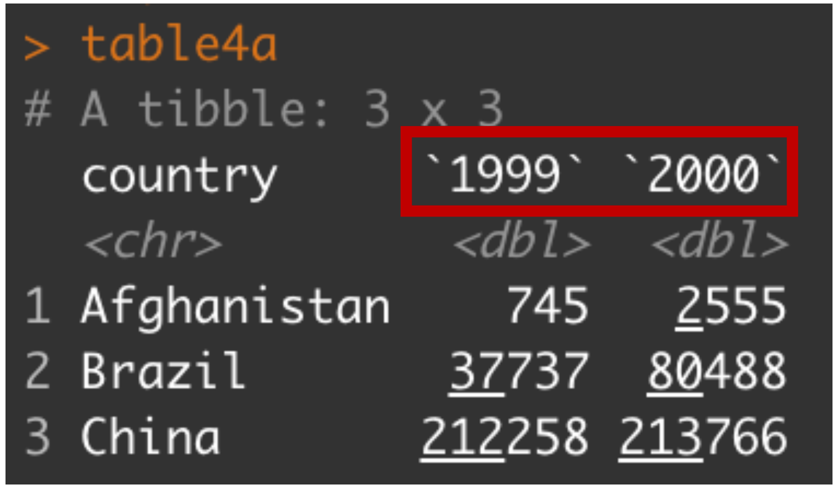


country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

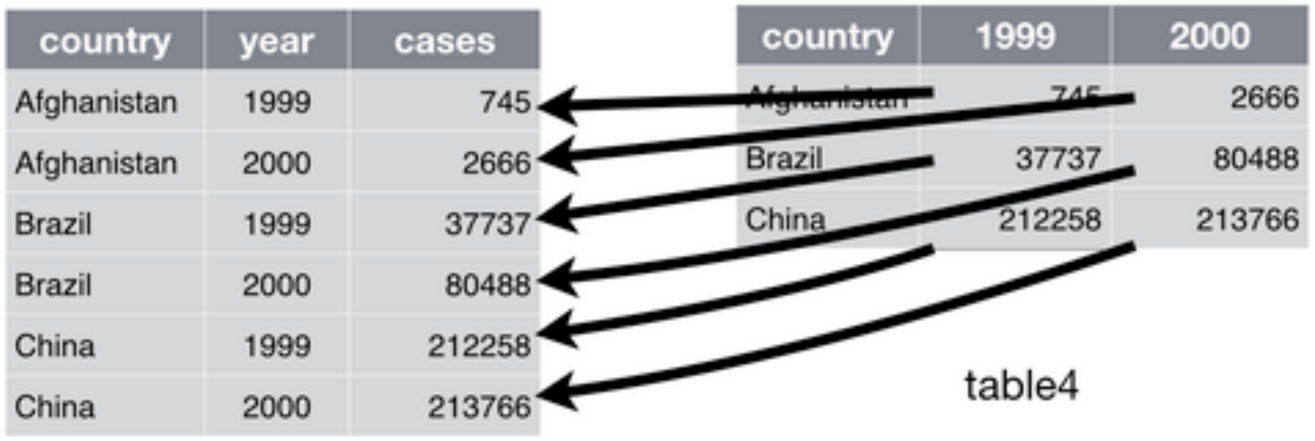
country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

pivot_longer()

Example case: column names are values, not variables.



```
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <dbl>  <dbl>
1 Afghanistan    745   2555
2 Brazil       37737  80488
3 China       212258 213766
```

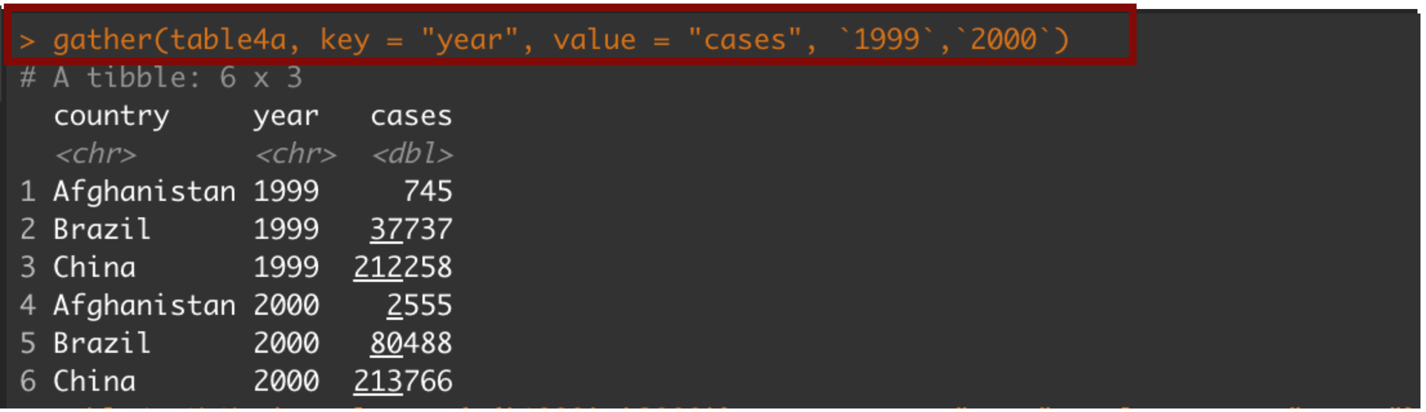


country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

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

Deprecated: `gather()` lifecycle retired

```
df %>% gather("key", "value", x, y, z)
```



```
> gather(table4a, key = "year", value = "cases", `1999`,`2000`)
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Brazil      1999   37737
3 China       1999  212258
4 Afghanistan 2000    2555
5 Brazil      2000   80488
6 China       2000  213766
```

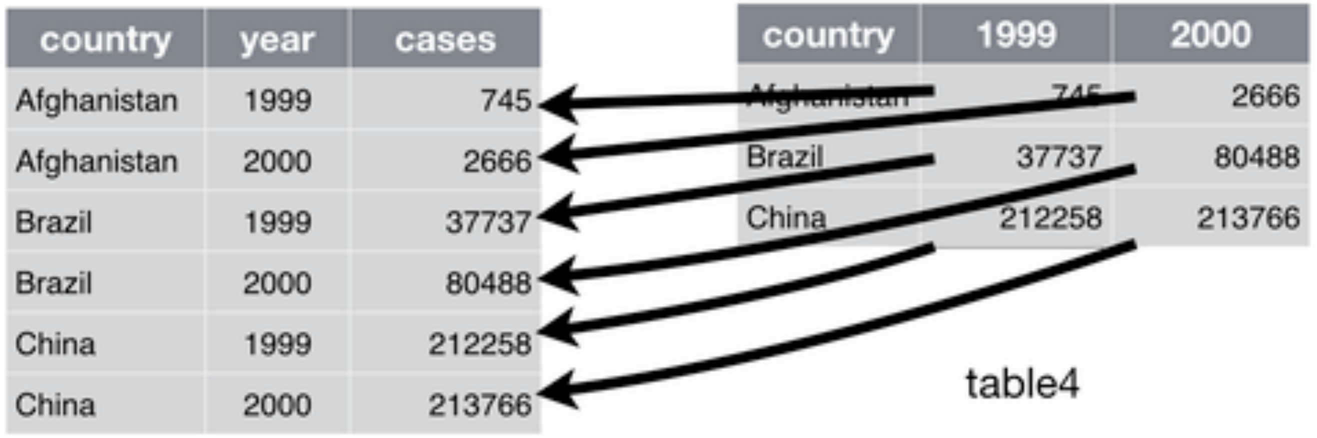
pivot_longer()

Example case: column names are values, not variables.

Deprecated: `gather()`

lifecycle retired

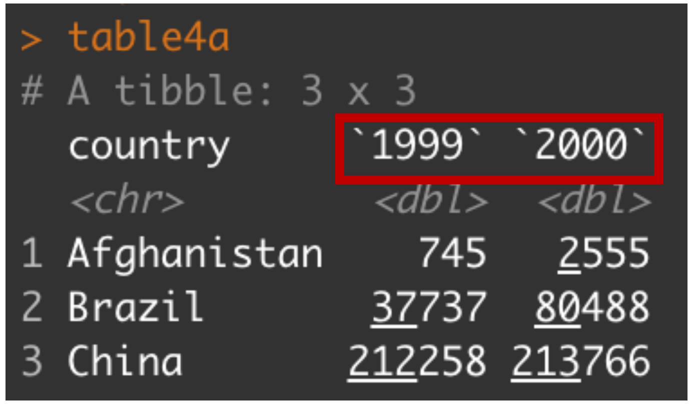
```
df %>% gather("key", "value", x, y, z)
```



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

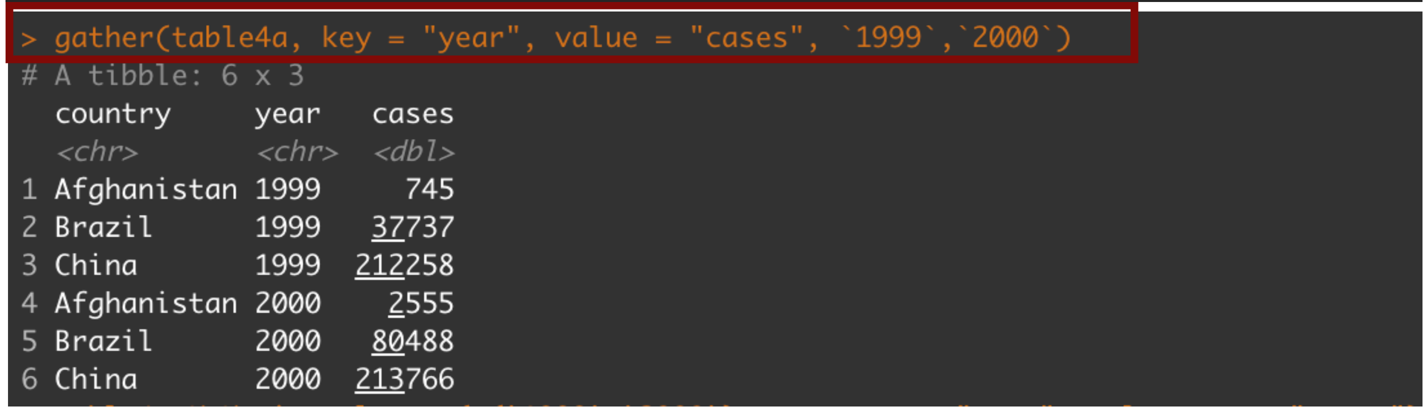
country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

Original tibble



```
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <dbl>  <dbl>
1 Afghanistan    745   2555
2 Brazil       37737  80488
3 China       212258 213766
```

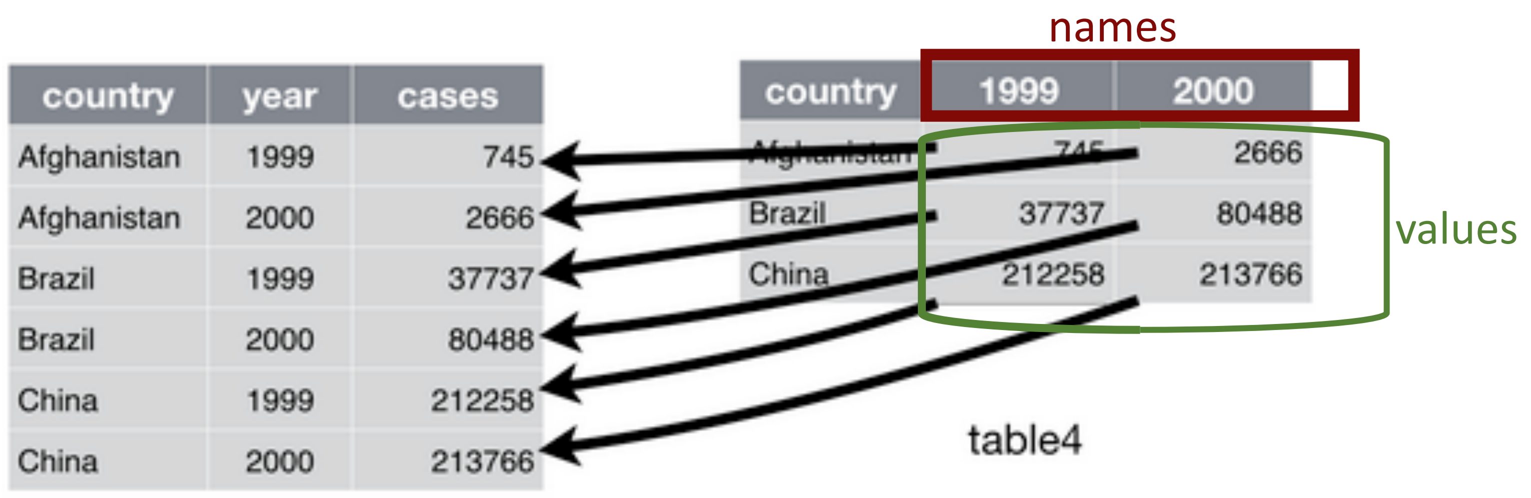
Tibble columns altered using gather() function



```
> gather(table4a, key = "year", value = "cases", `1999`,`2000`)
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Brazil      1999   37737
3 China       1999  212258
4 Afghanistan 2000    2555
5 Brazil      2000   80488
6 China       2000  213766
```

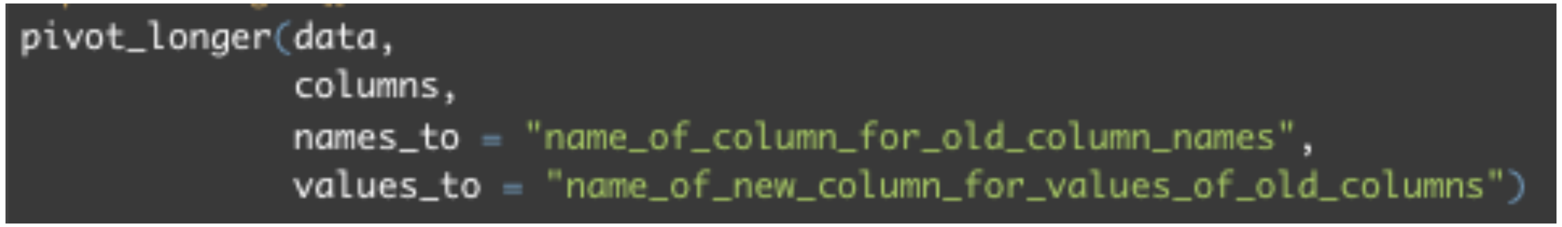
pivot_longer()

Example case: column names are values, not variables.



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

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

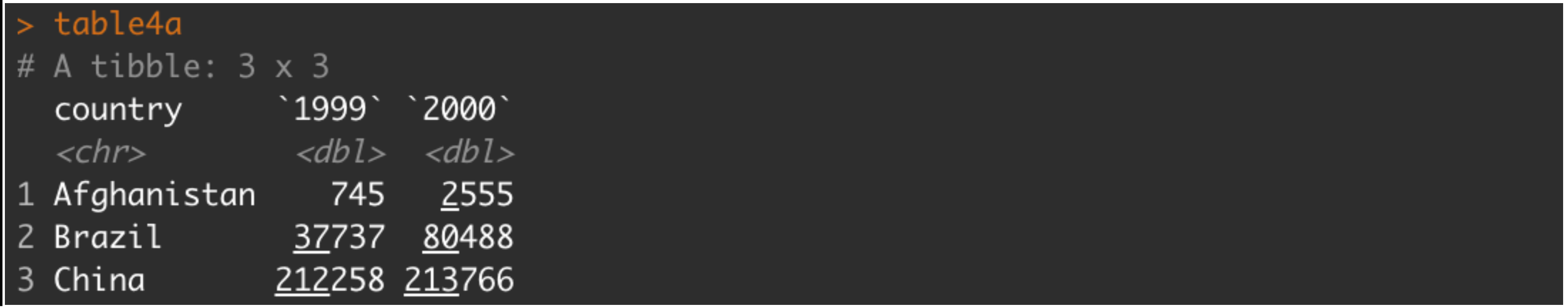


```
pivot_longer(data,
             columns,
             names_to = "name_of_column_for_old_column_names",
             values_to = "name_of_new_column_for_values_of_old_columns")
```

pivot_longer() - example

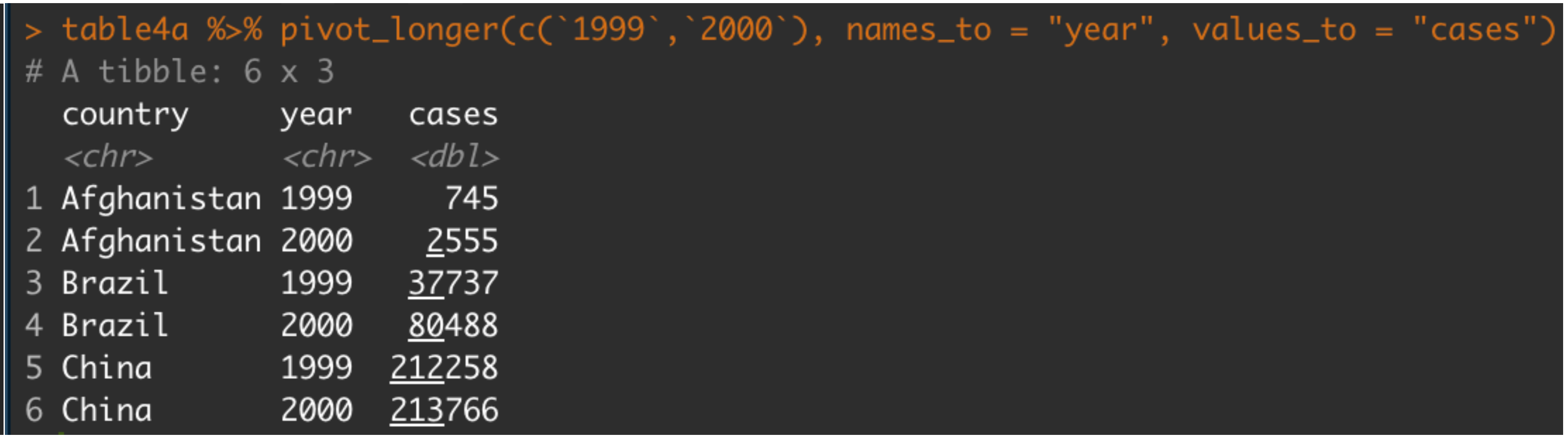
Example case: column names are values, not variables.

Original tibble



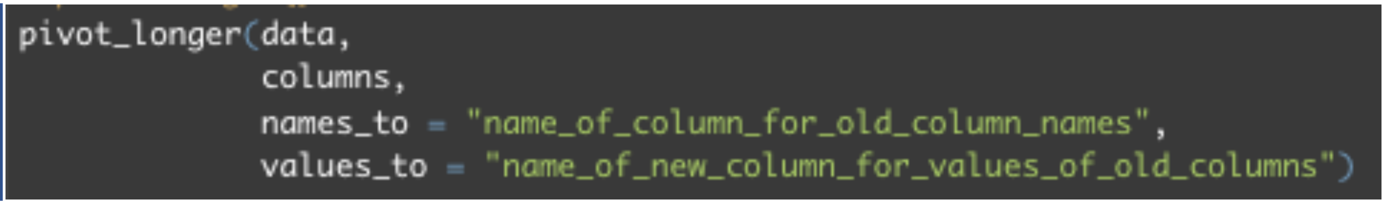
```
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <dbl>  <dbl>
1 Afghanistan    745   2555
2 Brazil       37737  80488
3 China       212258 213766
```

Pivot applied to tibble



```
> table4a %>% pivot_longer(c(`1999`,`2000`), names_to = "year", values_to = "cases")
# A tibble: 6 x 3
  country     year    cases
  <chr>       <chr>   <dbl>
1 Afghanistan 1999      745
2 Afghanistan 2000     2555
3 Brazil      1999    37737
4 Brazil      2000    80488
5 China       1999   212258
6 China       2000   213766
```

Function description

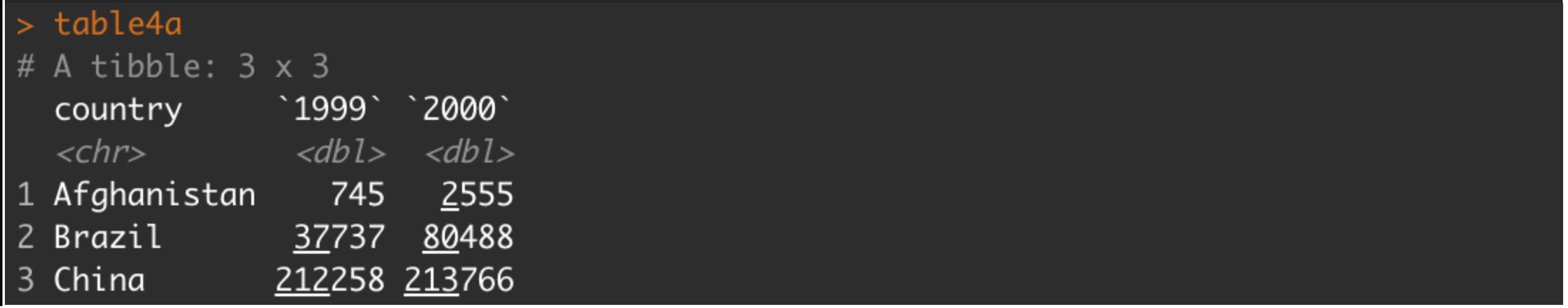


```
pivot_longer(data,
             columns,
             names_to = "name_of_column_for_old_column_names",
             values_to = "name_of_new_column_for_values_of_old_columns")
```

pivot_longer() - example

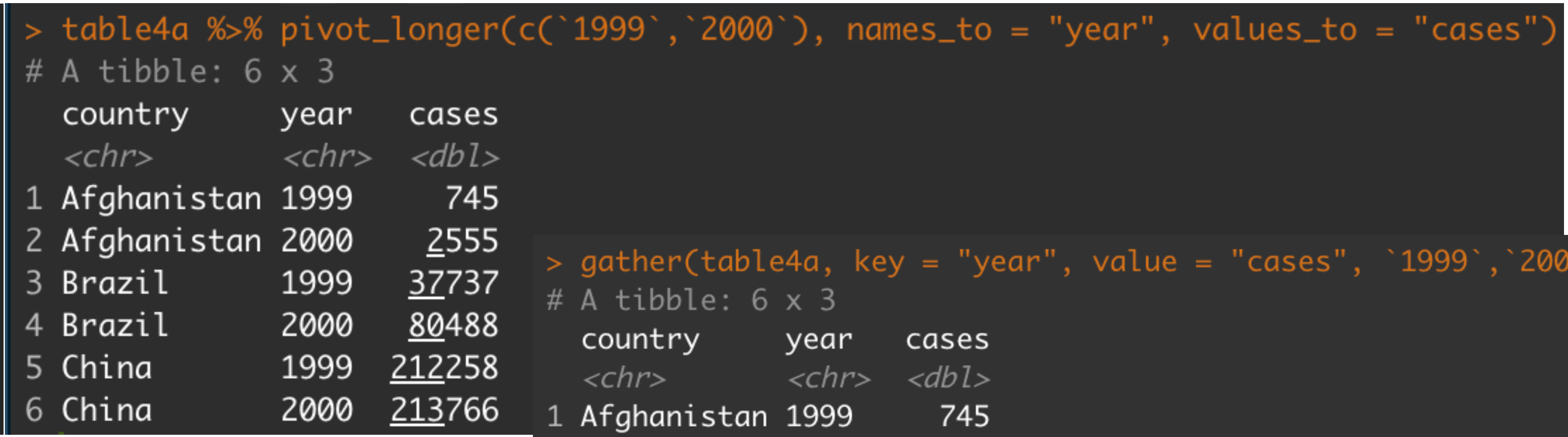
Example case: column names are values, not variables.

Original tibble

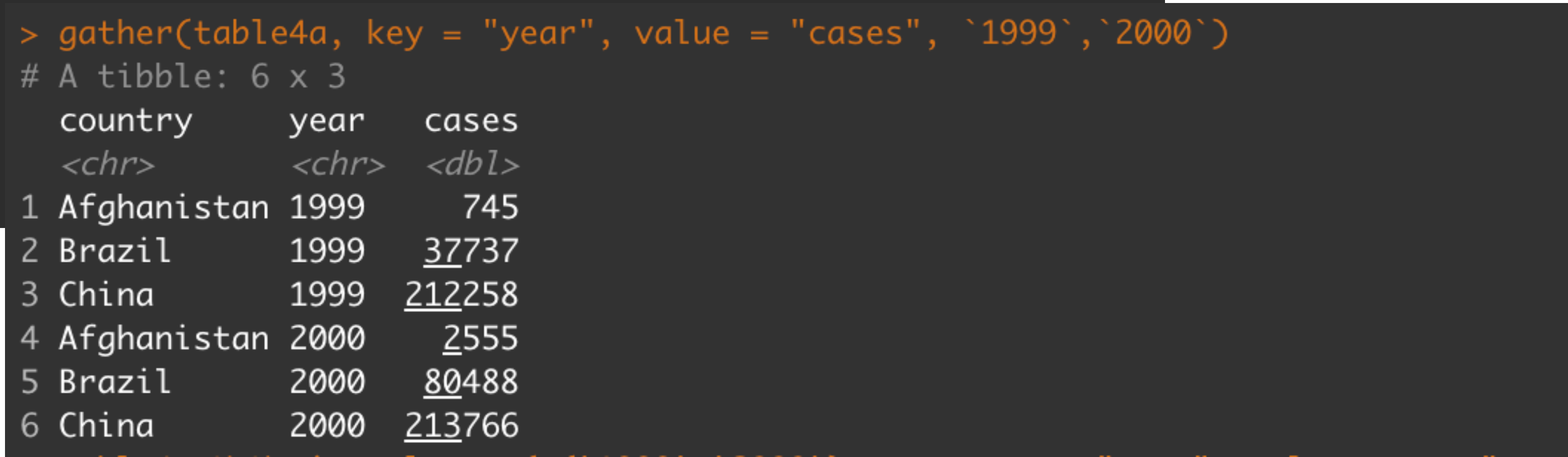


```
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <dbl>  <dbl>
1 Afghanistan    745   2555
2 Brazil       37737  80488
3 China       212258 213766
```

Pivot applied to tibble



```
> table4a %>% pivot_longer(c(`1999`,`2000`), names_to = "year", values_to = "cases")
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Afghanistan 2000    2555
3 Brazil      1999   37737
4 Brazil      2000   80488
5 China       1999  212258
6 China       2000  213766
```

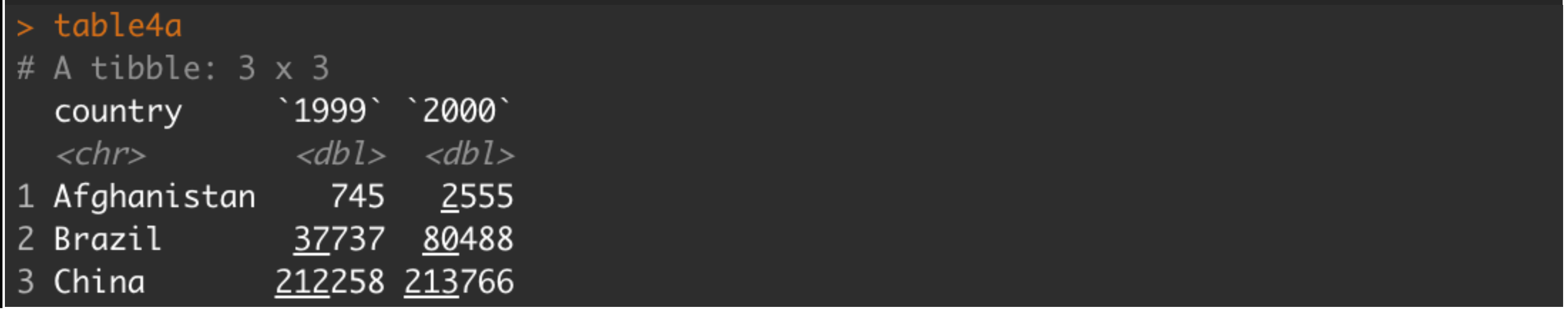


```
> gather(table4a, key = "year", value = "cases", `1999`,`2000`)
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Brazil      1999   37737
3 China       1999  212258
4 Afghanistan 2000    2555
5 Brazil      2000   80488
6 China       2000  213766
```

pivot_longer() - example

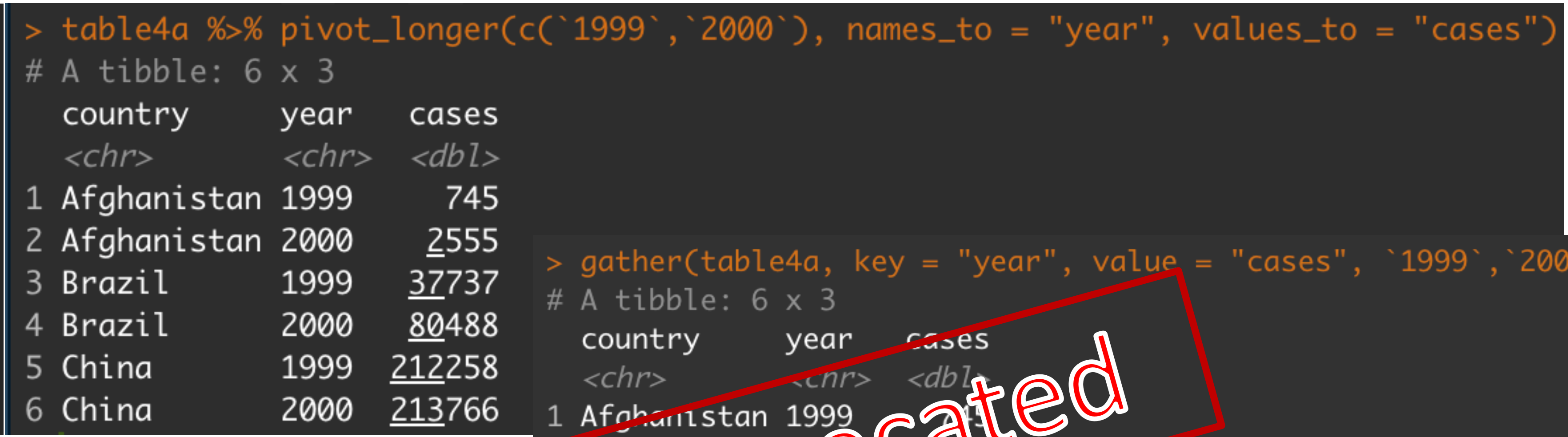
Example case: column names are values, not variables.

Original tibble

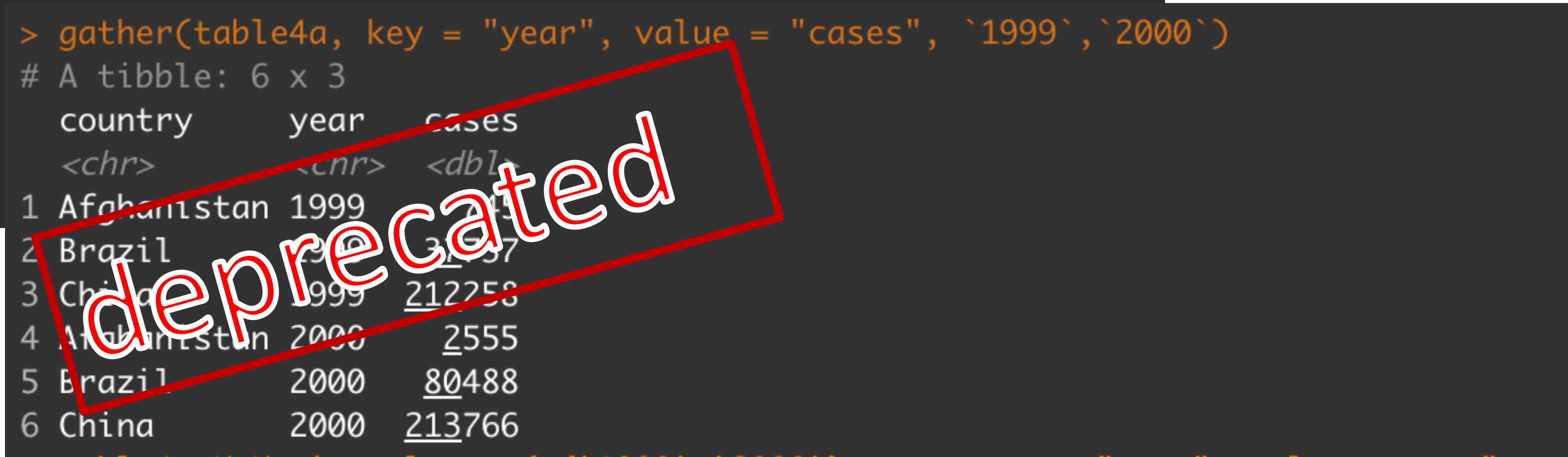


```
> table4a
# A tibble: 3 x 3
  country     `1999` `2000`
  <chr>        <dbl>  <dbl>
1 Afghanistan    745   2555
2 Brazil       37737  80488
3 China       212258 213766
```

Pivot applied to tibble



```
> table4a %>% pivot_longer(c(`1999`,`2000`), names_to = "year", values_to = "cases")
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Afghanistan 2000    2555
3 Brazil      1999   37737
4 Brazil      2000   80488
5 China       1999  212258
6 China       2000  213766
```



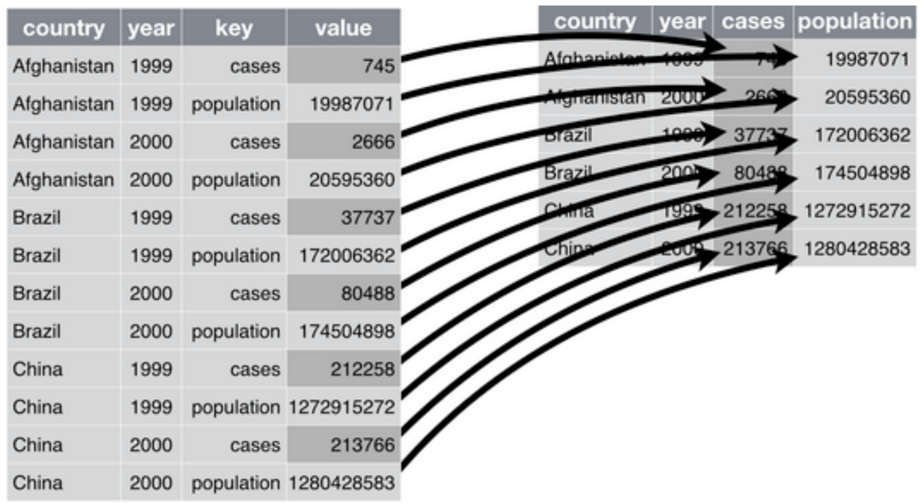
```
> gather(table4a, key = "year", value = "cases", `1999`,`2000`)
# A tibble: 6 x 3
  country     year   cases
  <chr>       <chr>  <dbl>
1 Afghanistan 1999     745
2 Brazil      1999   37737
3 China       1999  212258
4 Afghanistan 2000    2555
5 Brazil      2000   80488
6 China       2000  213766
```

deprecated

Tidying Functions

Data Wrangling Functions:	
pivot_longer()	pivot column names into new column and values of those columns into separate column
pivot_wider()	observations in multiple rows must be separated to new columns.
separate()	separate one column into multiple columns using a character for the split.
unite()	combine multiple columns into one column.
complete()	make implicit missing values into explicit NAs.
fill()	fill in NAs with prior non-NA value.

pivot_wider()



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

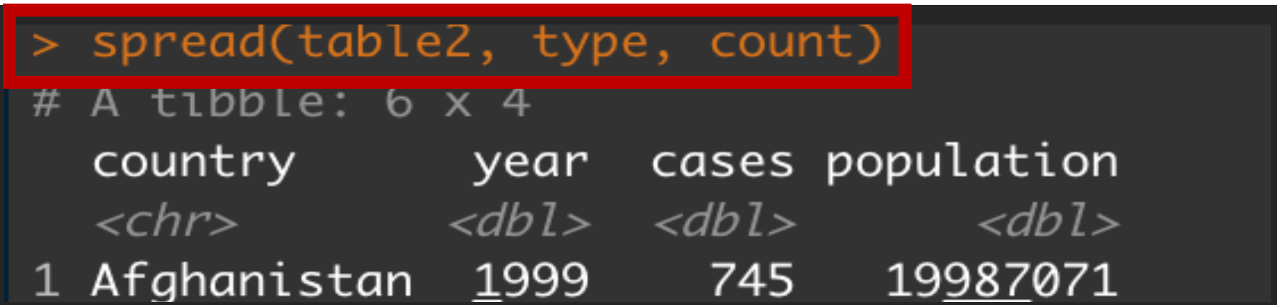
country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

pivot_wider()

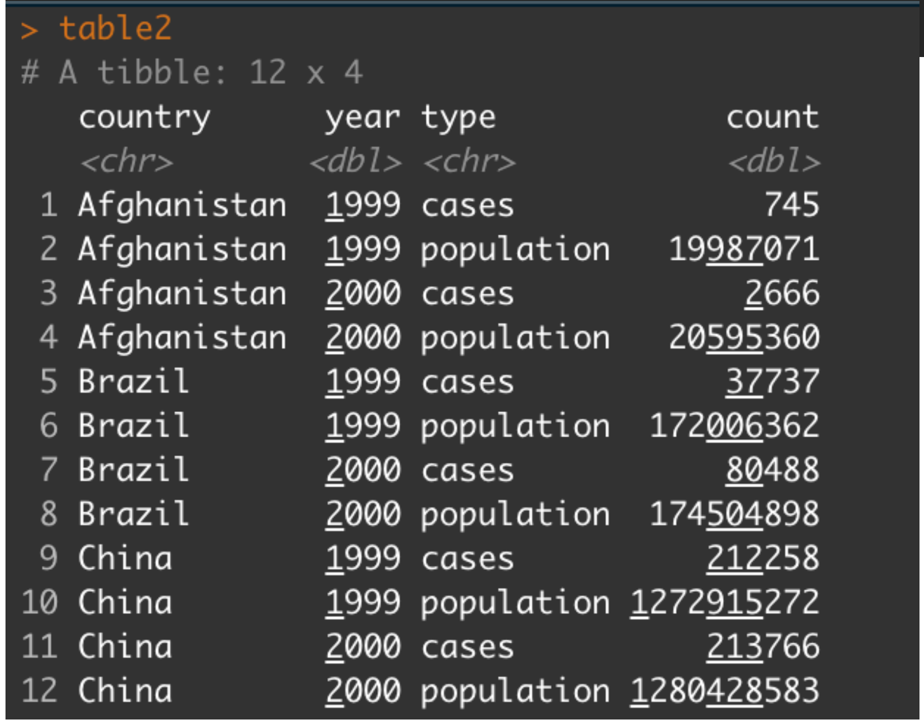
Example case: multiple observations in rows and want them in separate, new columns.

Deprecated: `spread()` lifecycle retired

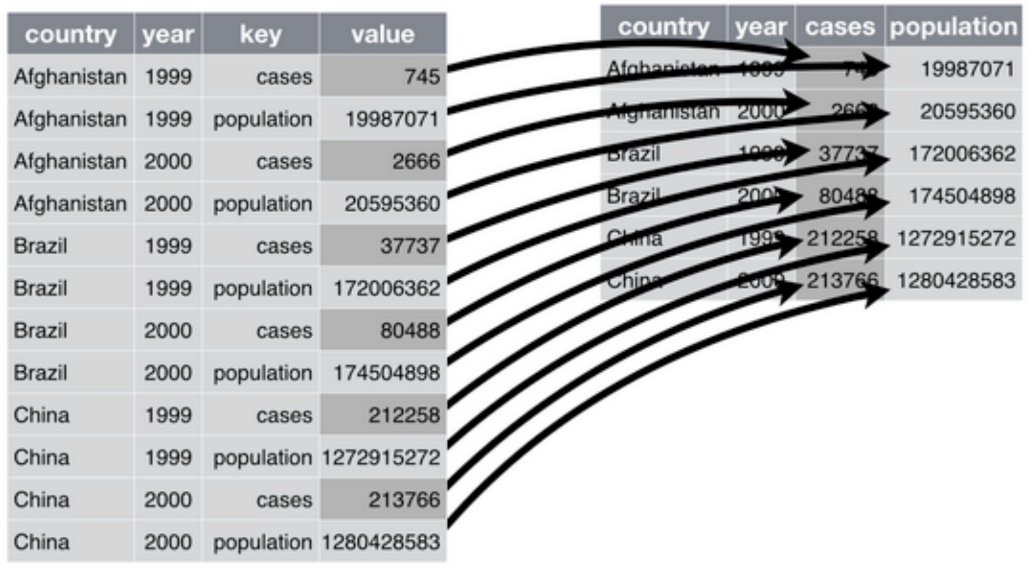
```
df %>% spread(key, value)
```



```
> spread(table2, type, count)
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
```



```
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <dbl> <chr>           <dbl>
 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
```



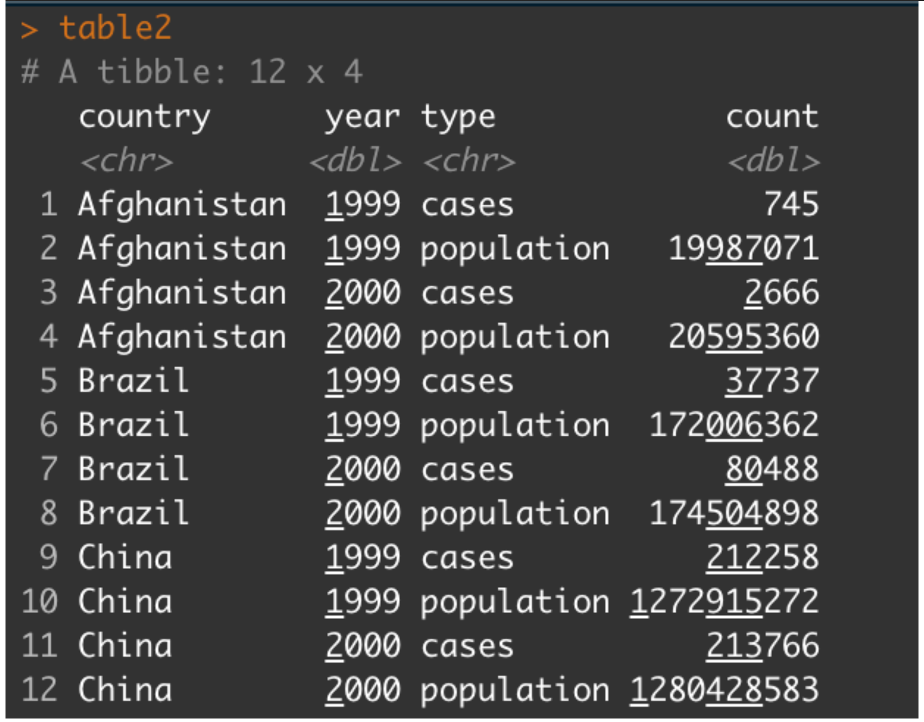
country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999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

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

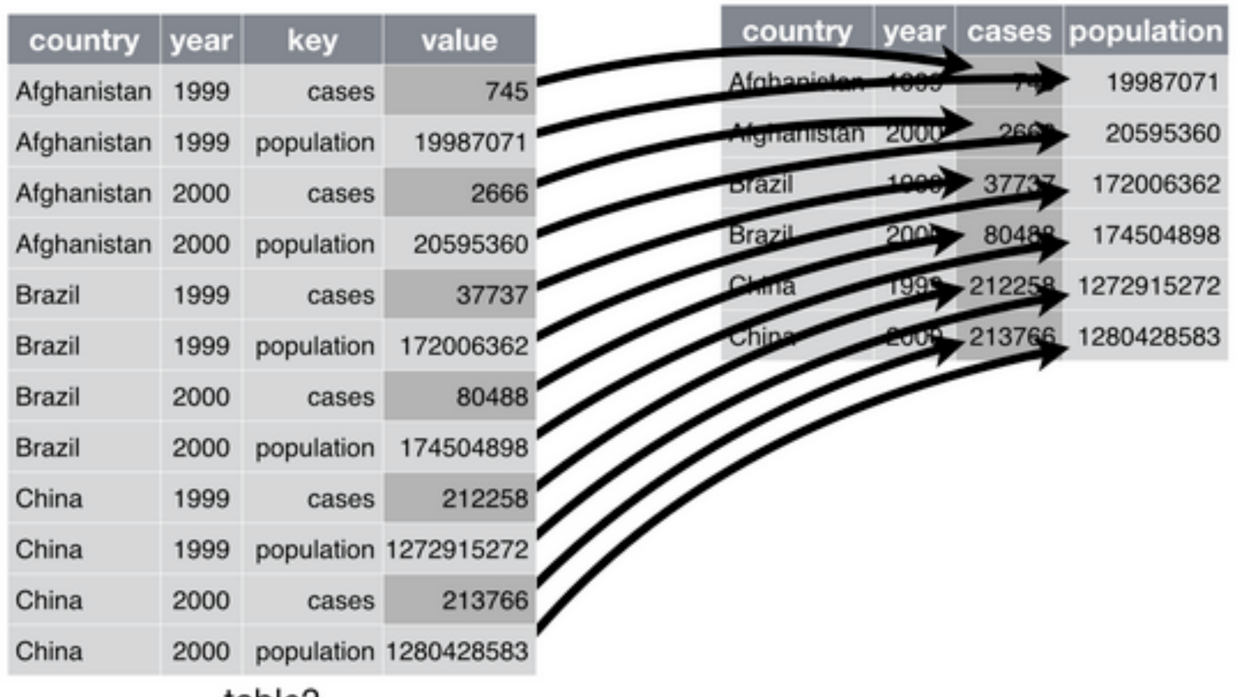
table2

pivot_wider() - example

Example case: multiple observations in rows.



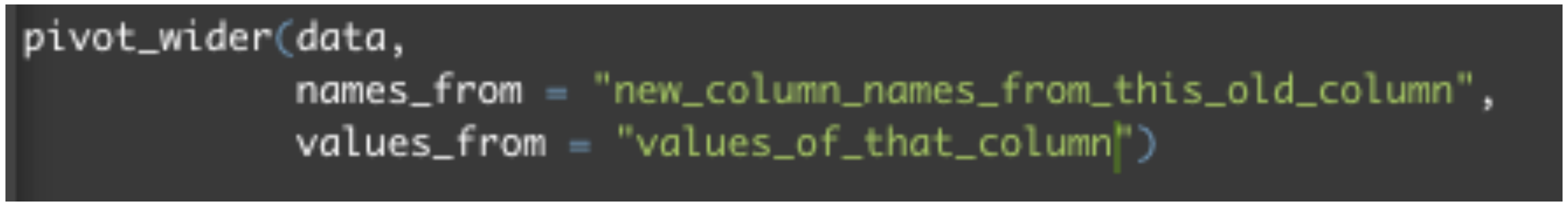
```
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <dbl> <chr>           <dbl>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       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
```



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

table2

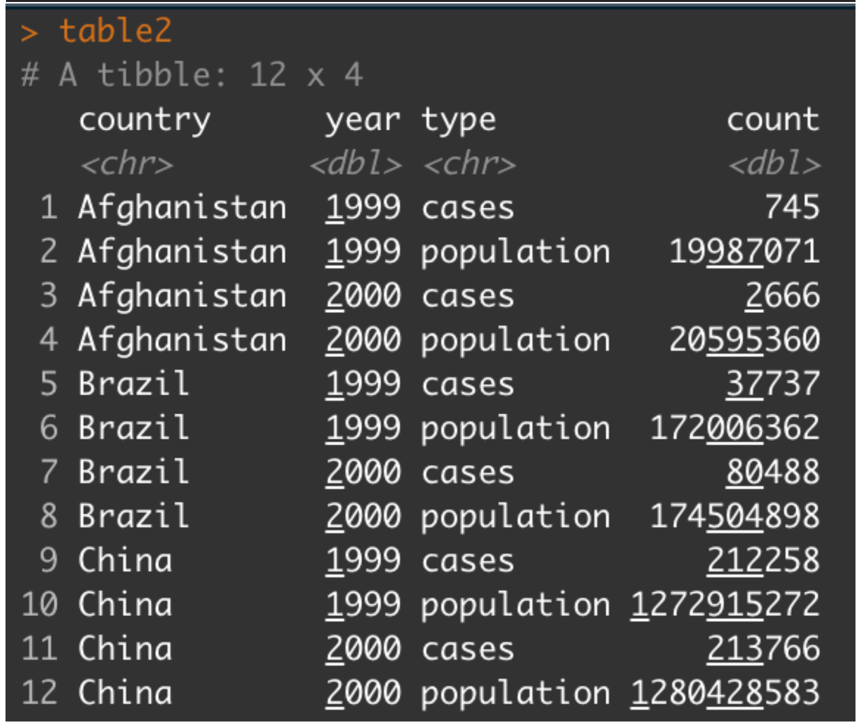
country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583



```
pivot_wider(data,
              names_from = "new_column_names_from_this_old_column",
              values_from = "values_of_that_column")
```

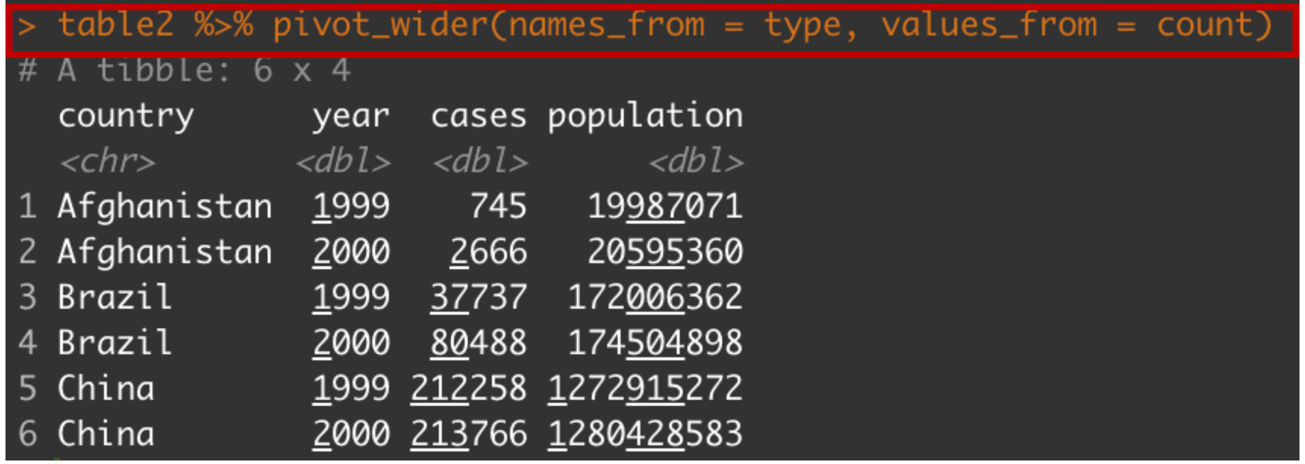
pivot_wider() - example

Original tibble



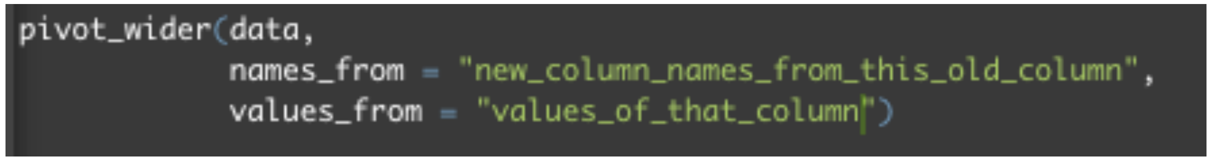
```
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <dbl> <chr>           <dbl>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       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
```

Pivotted tibble using pivot_wider()

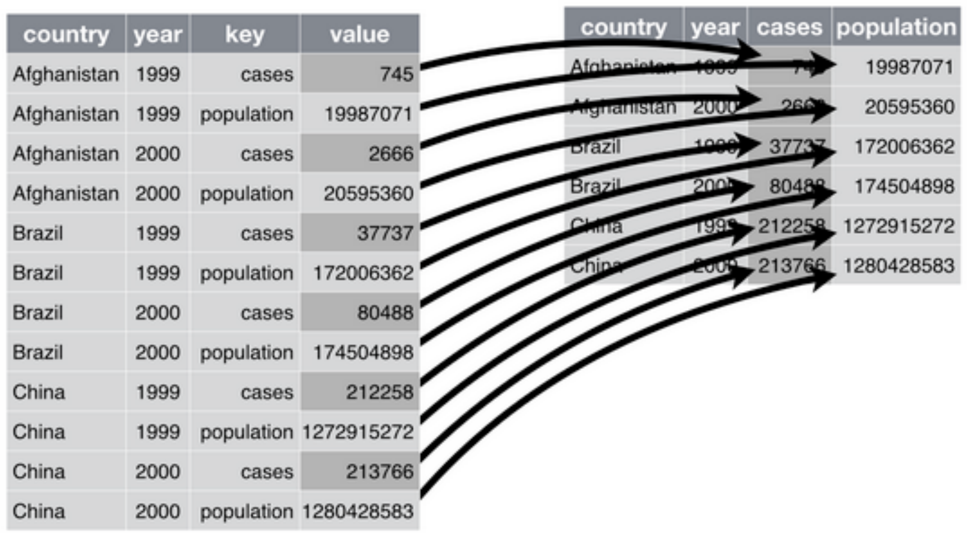


```
> table2 %>% pivot_wider(names_from = type, values_from = count)
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

Function info



```
pivot_wider(data,
            names_from = "new_column_names_from_this_old_column",
            values_from = "values_of_that_column")
```



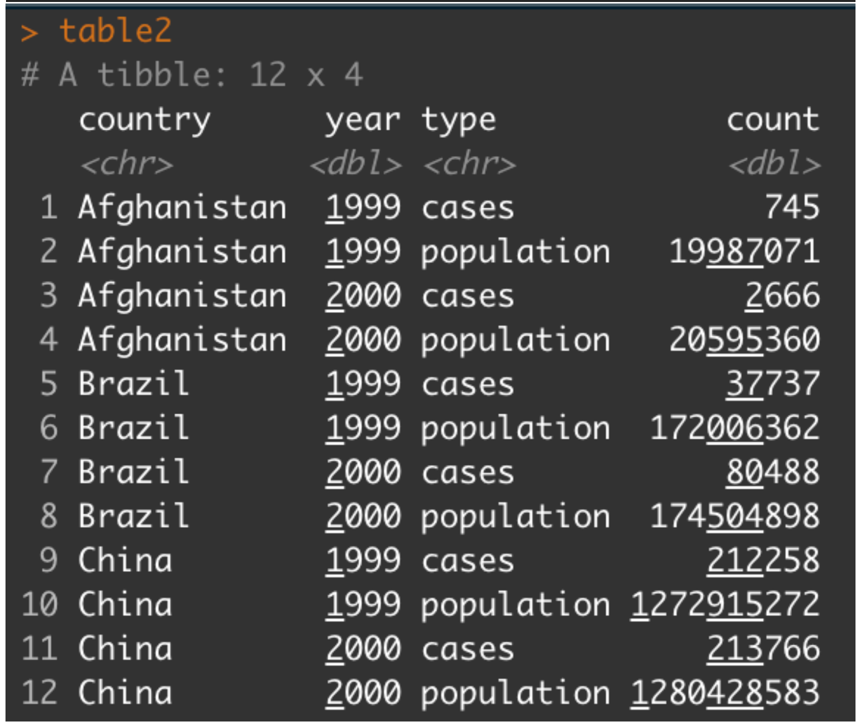
country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999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

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

table2

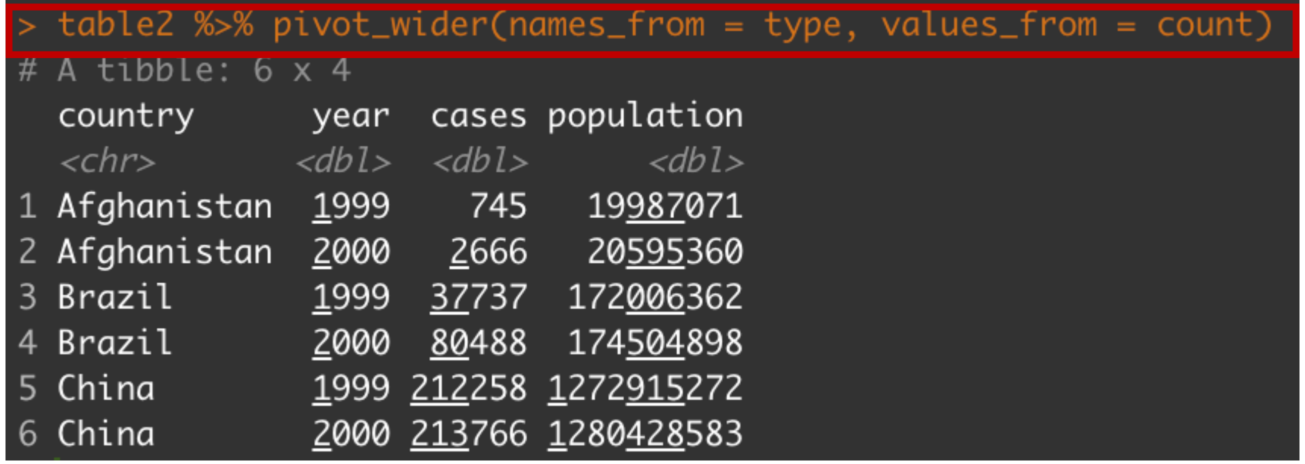
pivot_wider() - example

Original tibble

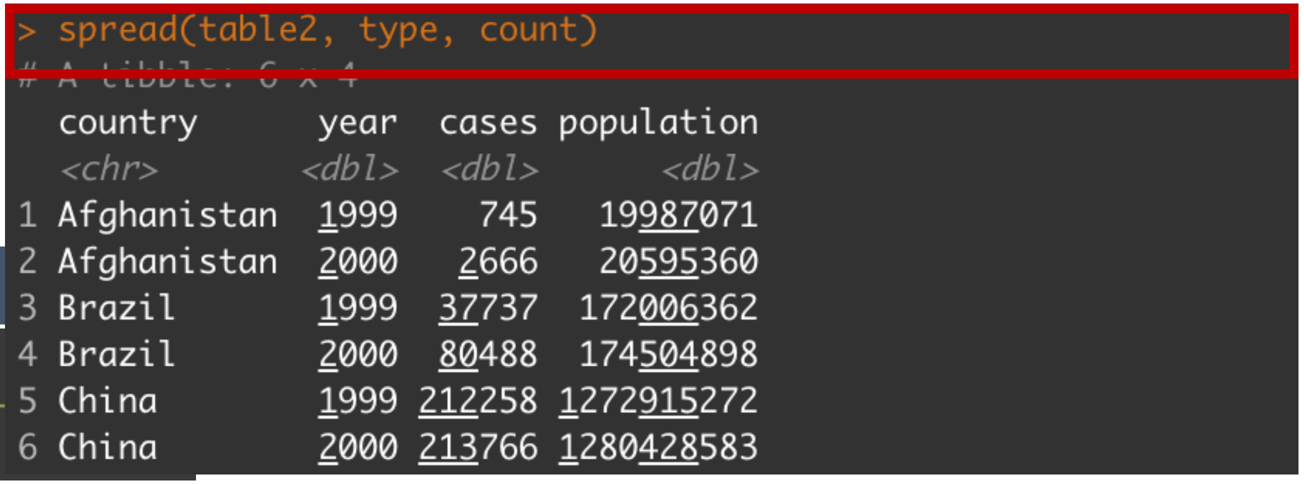


```
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <dbl> <chr>           <dbl>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       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
```

Pivotted tibble using pivot_wider()

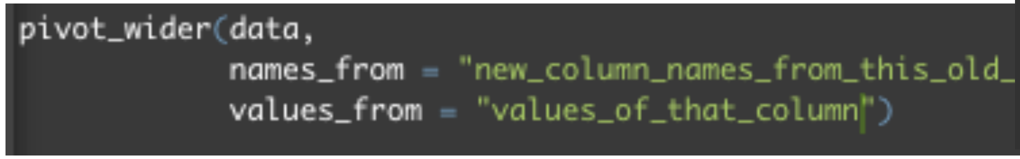


```
> table2 %>% pivot_wider(names_from = type, values_from = count)
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```



```
> spread(table2, type, count)
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

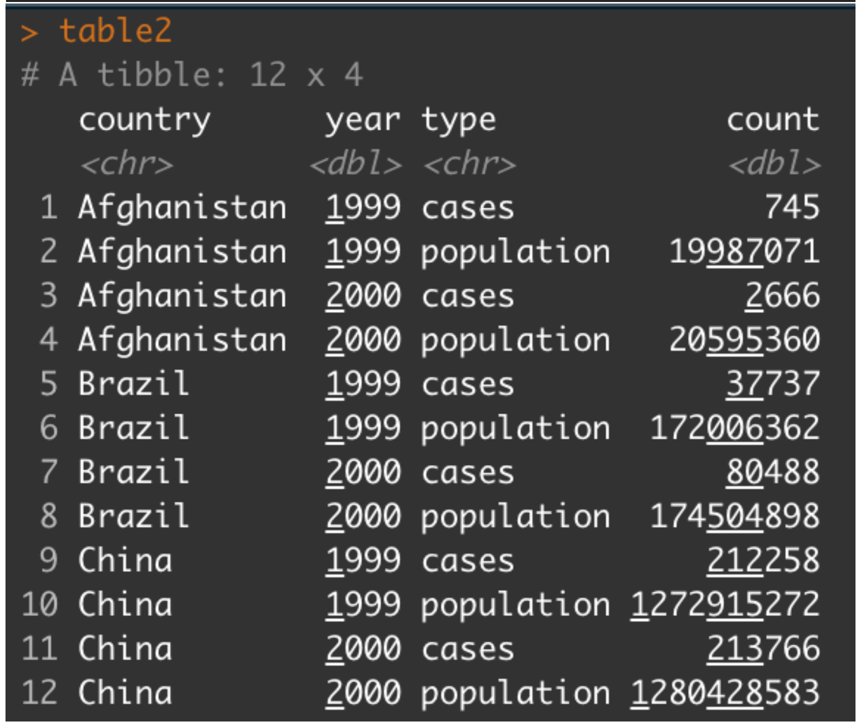
Function info



```
pivot_wider(data,
            names_from = "new_column_names_from_this_old_
            values_from = "values_of_that_column")
```

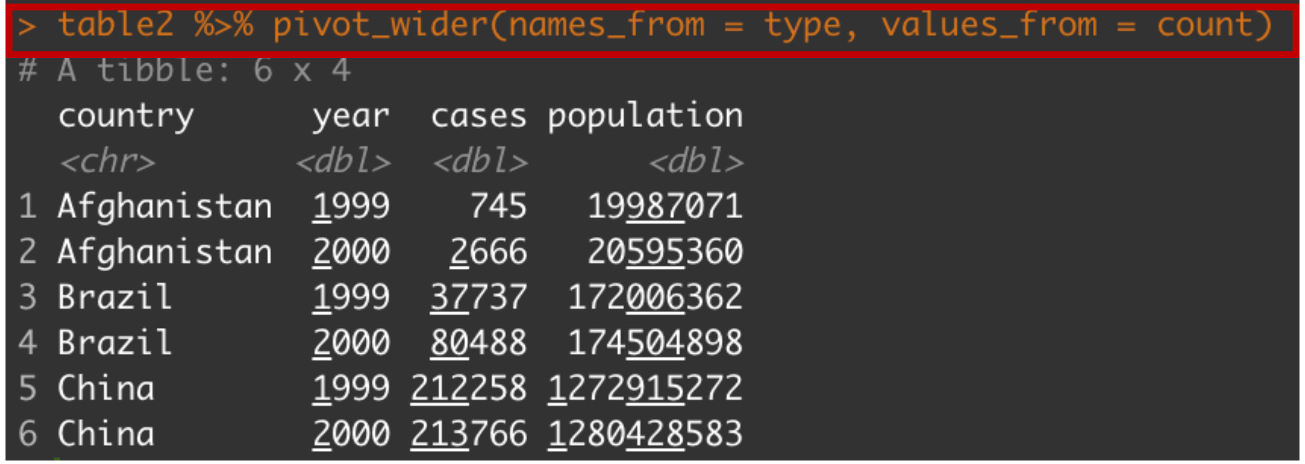
pivot_wider() - example

Original tibble

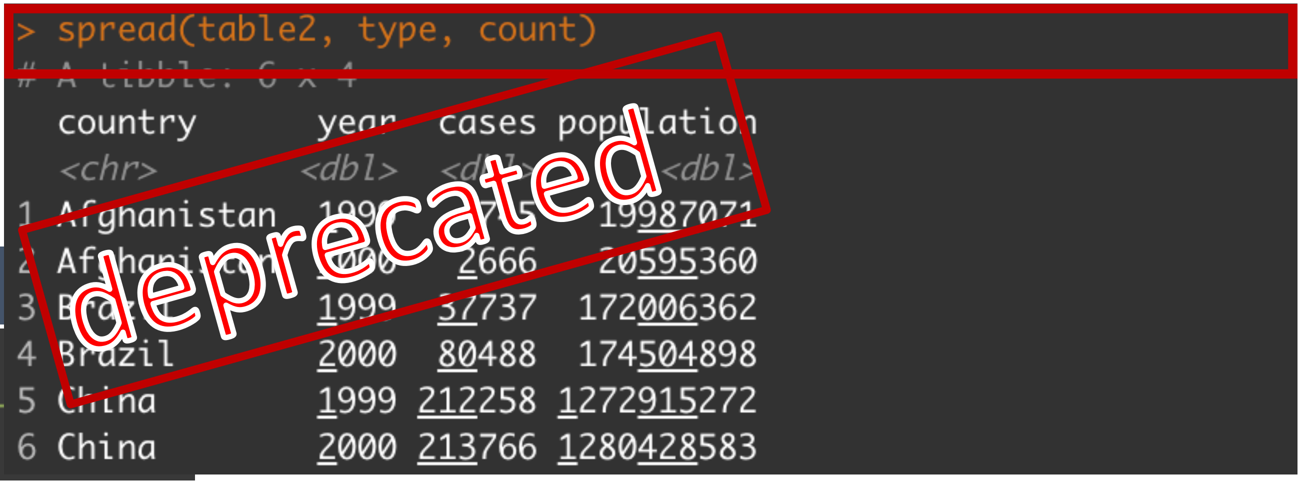


```
> table2
# A tibble: 12 x 4
   country      year type            count
   <chr>       <dbl> <chr>           <dbl>
 1 Afghanistan  1999 cases             745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000 cases            2666
 4 Afghanistan  2000 population   20595360
 5 Brazil       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
```

Pivotted tibble using pivot_wider()



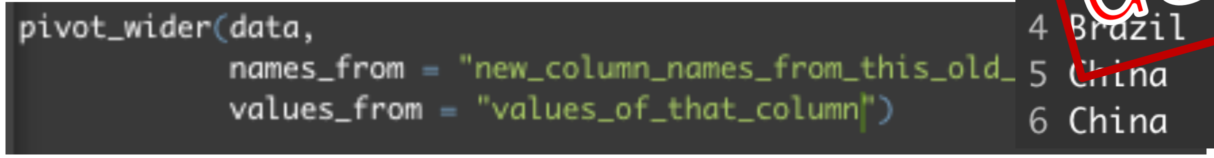
```
> table2 %>% pivot_wider(names_from = type, values_from = count)
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```



```
> spread(table2, type, count)
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <dbl>  <dbl>      <dbl>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

deprecated

Function info

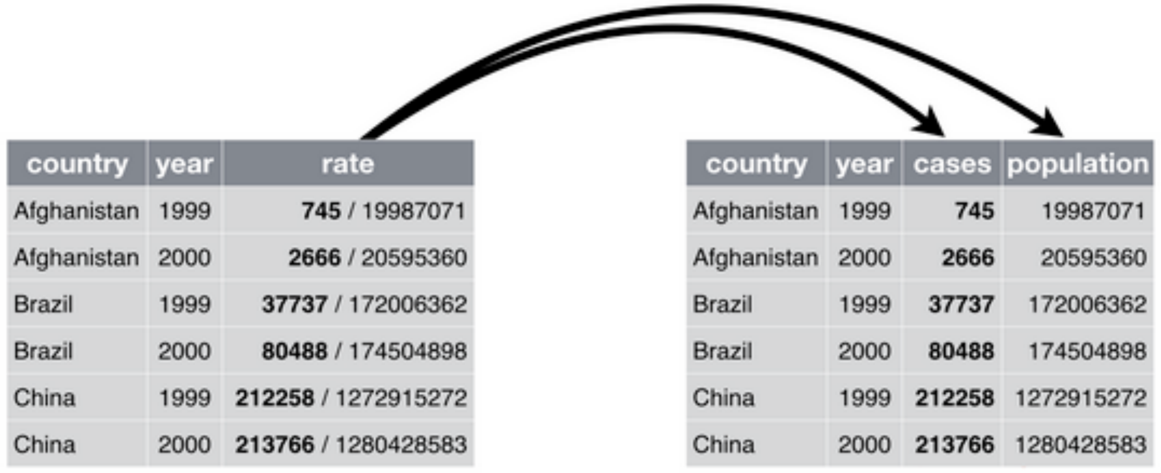


```
pivot_wider(data,
            names_from = "new_column_names_from_this_old_
            values_from = "values_of_that_column")
```

Tidying Functions

Data Wrangling Functions:	
pivot_longer()	pivot column names into new column and values of those columns into separate column
pivot_wider()	observations in multiple rows must be separated to new columns.
separate()	separate one column into multiple columns using a character for the split.
unite()	combine multiple columns into one column.
complete()	make implicit missing values into explicit NAs.
fill()	fill in NAs with prior non-NA value.

separate()

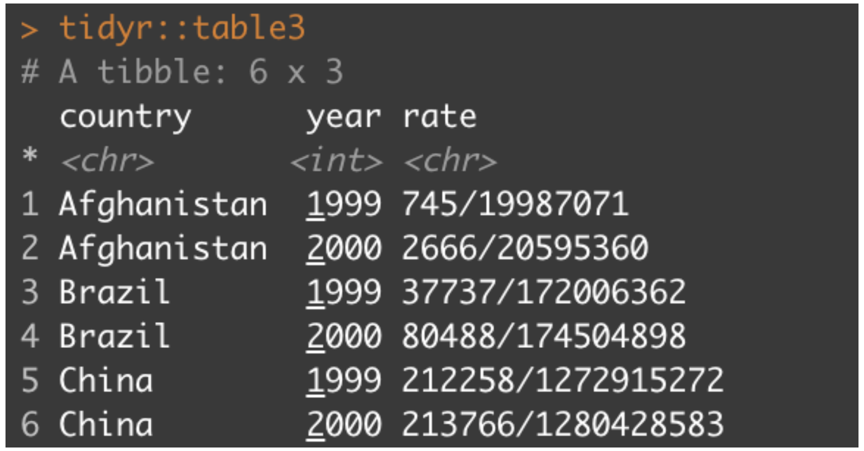


country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

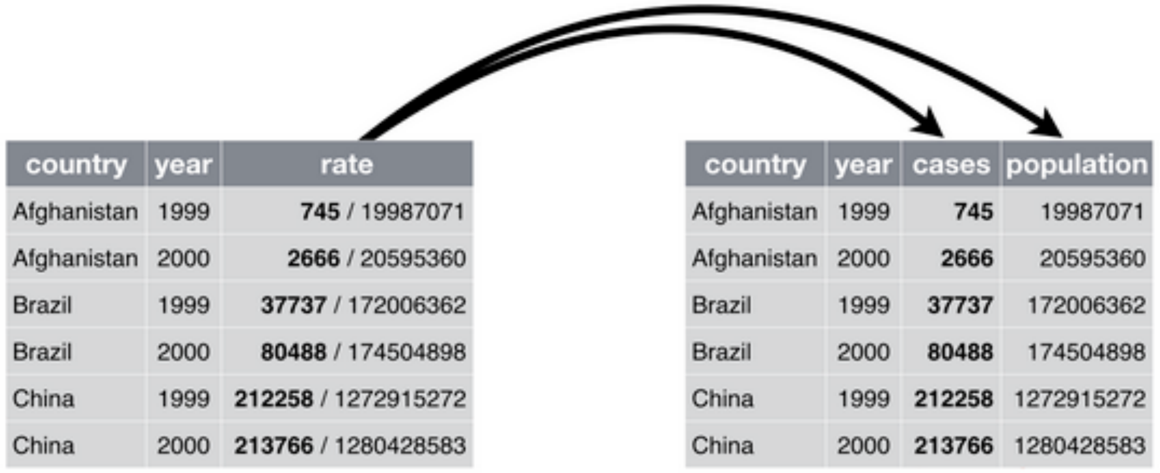
country	year	cases	population
Afghanistan	1999	**745**	19987071
Afghanistan	2000	**2666**	20595360
Brazil	1999	**37737**	172006362
Brazil	2000	**80488**	174504898
China	1999	**212258**	1272915272
China	2000	**213766**	1280428583

separate()

Example case: separate two values per cell by a character.

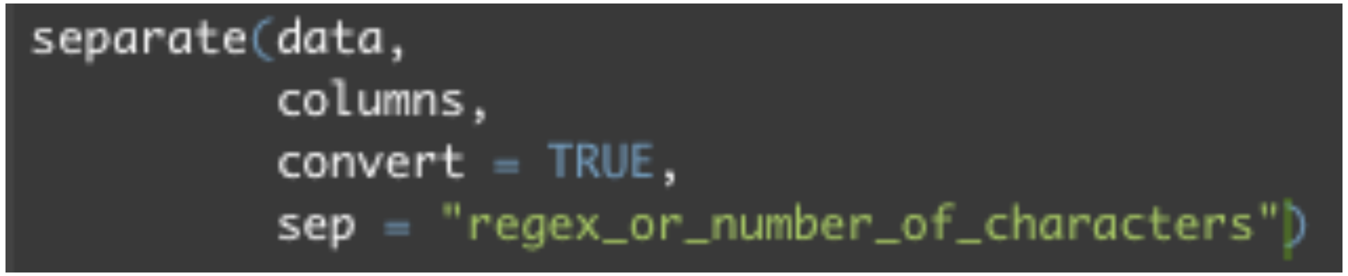


```
> tidyr::table3
# A tibble: 6 x 3
  country      year rate
* <chr>       <int> <chr>
1 Afghanistan  1999 745/19987071
2 Afghanistan  2000 2666/20595360
3 Brazil       1999 37737/172006362
4 Brazil       2000 80488/174504898
5 China        1999 212258/1272915272
6 China        2000 213766/1280428583
```



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

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

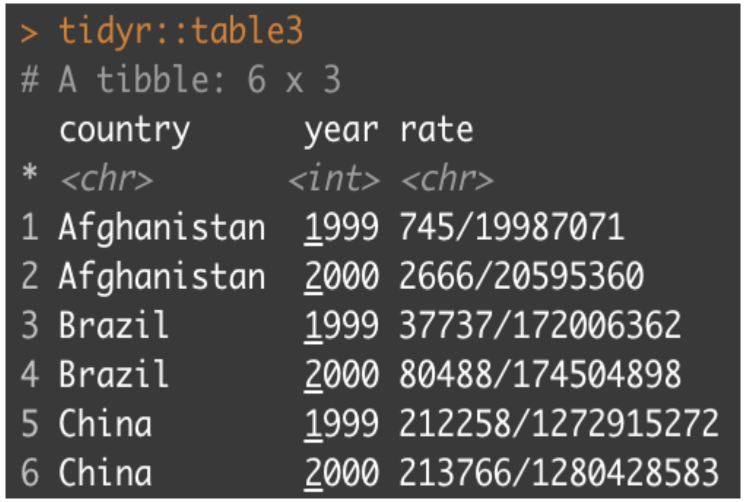


```
separate(data,
         columns,
         convert = TRUE,
         sep = "regex_or_number_of_characters")
```

separate()

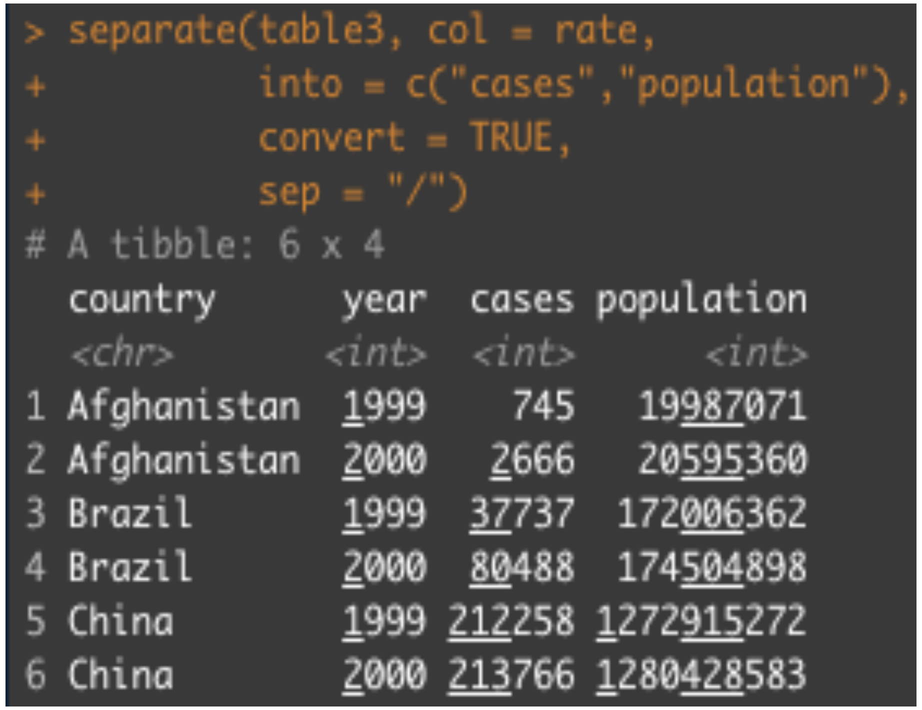
Example case: separate two values per cell by a character.

Original table



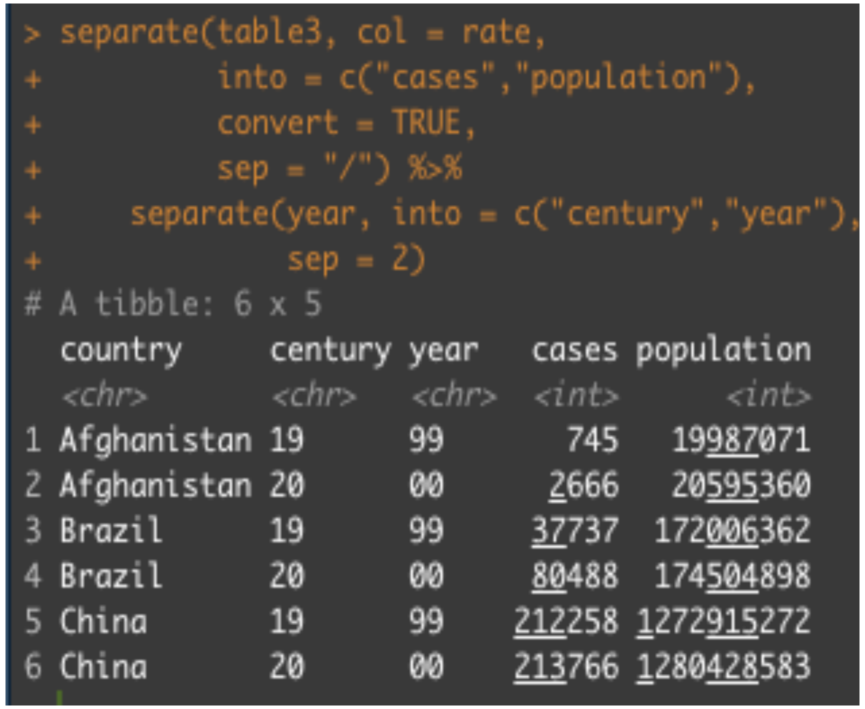
```
> tidyr::table3
# A tibble: 6 x 3
  country      year rate
* <chr>       <int> <chr>
1 Afghanistan  1999 745/19987071
2 Afghanistan  2000 2666/20595360
3 Brazil       1999 37737/172006362
4 Brazil       2000 80488/174504898
5 China        1999 212258/1272915272
6 China        2000 213766/1280428583
```

Separate “rate” column.



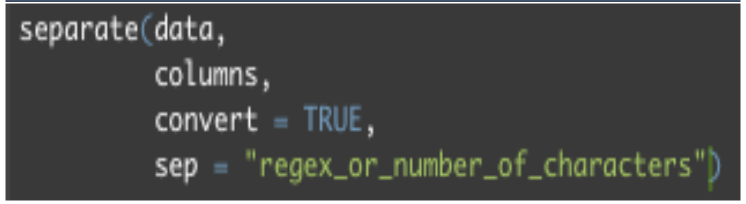
```
> separate(table3, col = rate,
+          into = c("cases","population"),
+          convert = TRUE,
+          sep = "/")
# A tibble: 6 x 4
  country      year  cases population
  <chr>       <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999 212258 1272915272
6 China        2000 213766 1280428583
```

Separate in two different ways.



```
> separate(table3, col = rate,
+          into = c("cases","population"),
+          convert = TRUE,
+          sep = "/") %>%
+     separate(year, into = c("century","year"),
+          sep = 2)
# A tibble: 6 x 5
  country     century year   cases population
  <chr>       <chr>   <chr>  <int>      <int>
1 Afghanistan 19      99       745   19987071
2 Afghanistan 20      00      2666   20595360
3 Brazil      19      99     37737  172006362
4 Brazil      20      00     80488  174504898
5 China       19      99    212258 1272915272
6 China       20      00    213766 1280428583
```

Function info

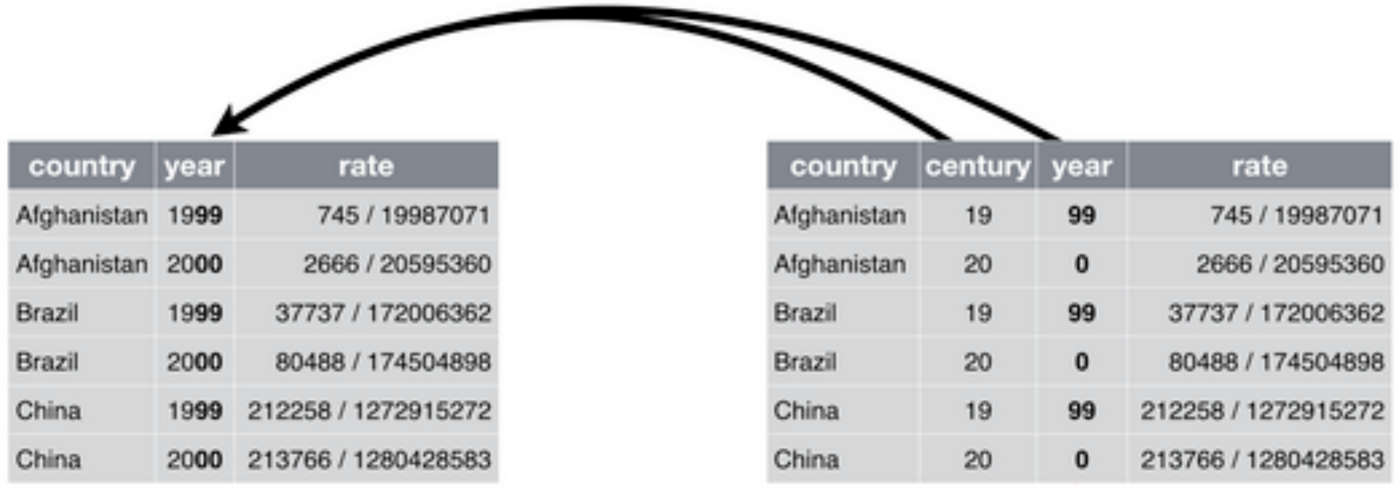


```
separate(data,
         columns,
         convert = TRUE,
         sep = "regex_or_number_of_characters")
```

Tidying Functions

Data Wrangling Functions:	
pivot_longer()	pivot column names into new column and values of those columns into separate column
pivot_wider()	observations in multiple rows must be separated to new columns.
separate()	separate one column into multiple columns using a character for the split.
unite()	combine multiple columns into one column.
complete()	make implicit missing values into explicit NAs.
fill()	fill in NAs with prior non-NA value.

unite()

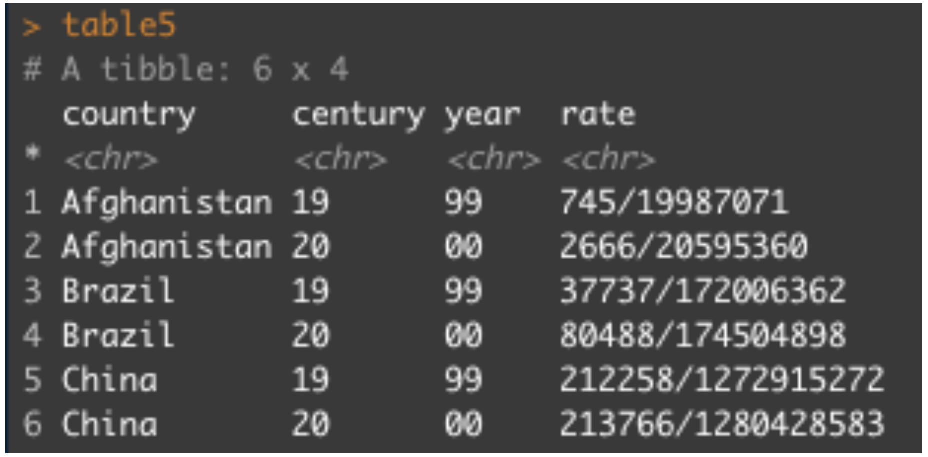


country	year	rate
Afghanistan	1999	745 / 19987071
Afghanistan	2000	2666 / 20595360
Brazil	1999	37737 / 172006362
Brazil	2000	80488 / 174504898
China	1999	212258 / 1272915272
China	2000	213766 / 1280428583

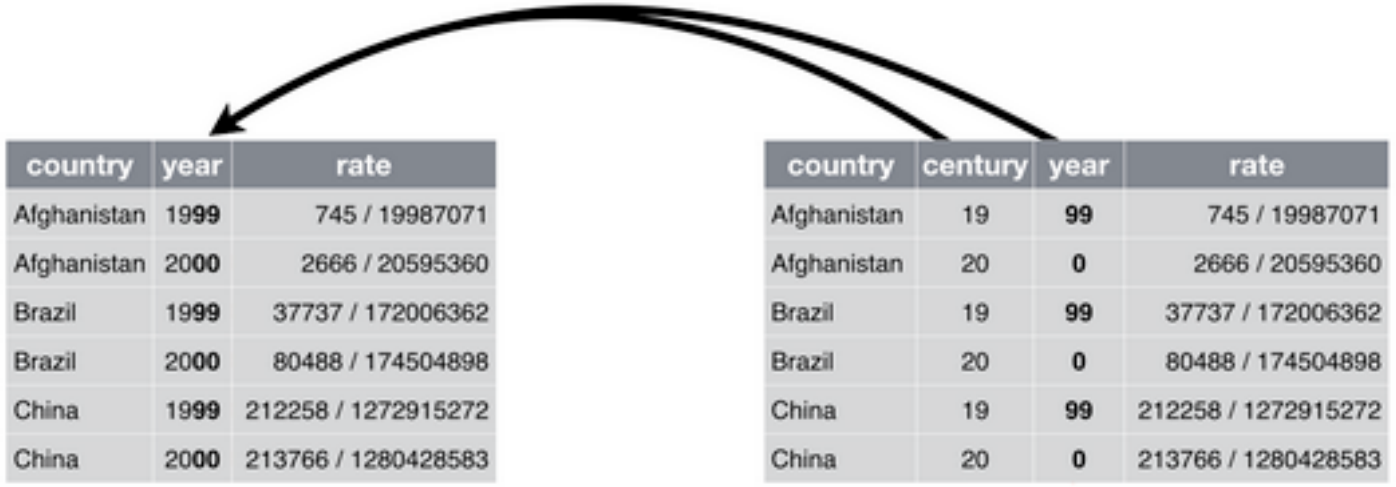
country	century	year	rate
Afghanistan	19	99	745 / 19987071
Afghanistan	20	0	2666 / 20595360
Brazil	19	99	37737 / 172006362
Brazil	20	0	80488 / 174504898
China	19	99	212258 / 1272915272
China	20	0	213766 / 1280428583

unite()

Example case: combine multiple columns into a single column.

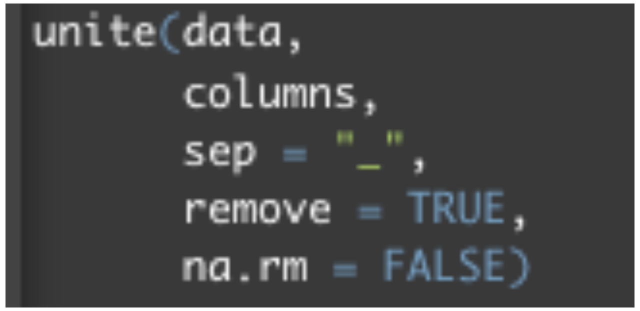


```
> table5
# A tibble: 6 x 4
  country     century year  rate
* <chr>       <chr>   <chr> <chr>
1 Afghanistan 19      99    745/19987071
2 Afghanistan 20      00    2666/20595360
3 Brazil      19      99    37737/172006362
4 Brazil      20      00    80488/174504898
5 China       19      99    212258/1272915272
6 China       20      00    213766/1280428583
```



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

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

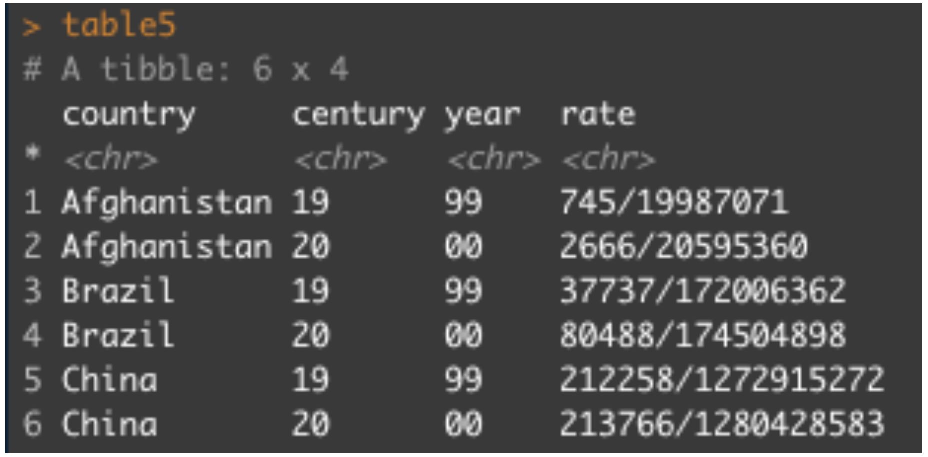


```
unite(data,
      columns,
      sep = "_",
      remove = TRUE,
      na.rm = FALSE)
```

unite()

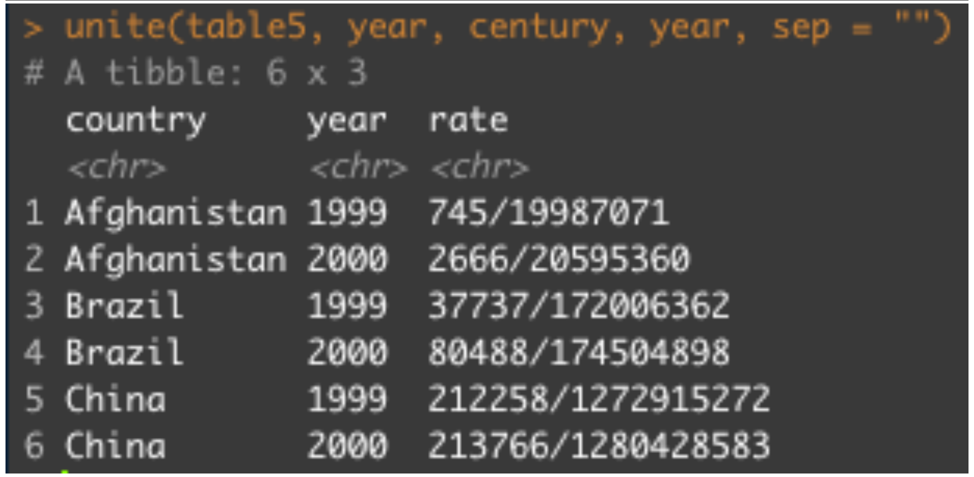
Example case: combine multiple columns into a single column.

Original table



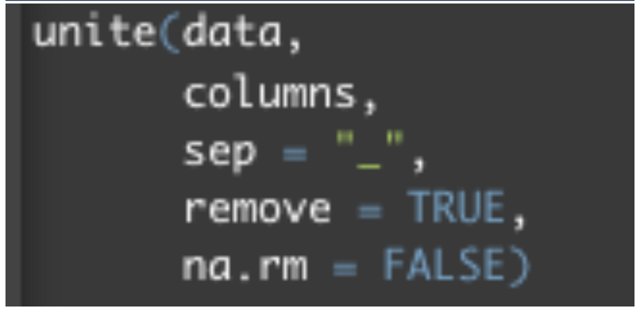
```
> table5
# A tibble: 6 x 4
  country     century year  rate
* <chr>       <chr>   <chr> <chr>
1 Afghanistan 19      99    745/19987071
2 Afghanistan 20      00    2666/20595360
3 Brazil      19      99    37737/172006362
4 Brazil      20      00    80488/174504898
5 China       19      99    212258/1272915272
6 China       20      00    213766/1280428583
```

United table



```
> unite(table5, year, century, year, sep = "")
# A tibble: 6 x 3
  country     year  rate
  <chr>       <chr> <chr>
1 Afghanistan 1999  745/19987071
2 Afghanistan 2000  2666/20595360
3 Brazil      1999  37737/172006362
4 Brazil      2000  80488/174504898
5 China       1999  212258/1272915272
6 China       2000  213766/1280428583
```

Function info



```
unite(data,
      columns,
      sep = "_",
      remove = TRUE,
      na.rm = FALSE)
```

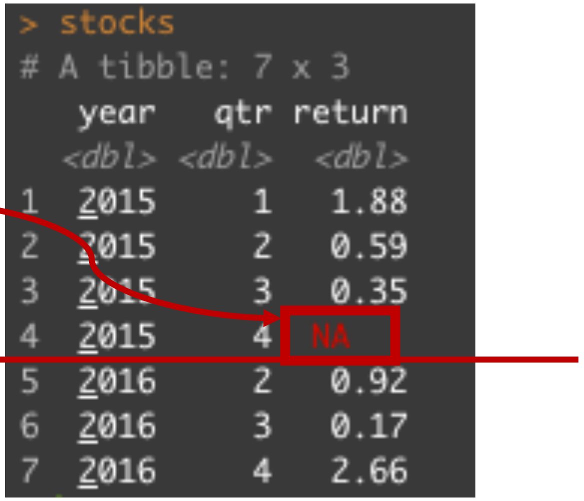
Tidying Functions

Data Wrangling Functions:	
pivot_longer()	pivot column names into new column and values of those columns into separate column
pivot_wider()	observations in multiple rows must be separated to new columns.
separate()	separate one column into multiple columns using a character for the split.
unite()	combine multiple columns into one column.
complete()	make implicit missing values into explicit NAs.
fill()	fill in NAs with prior non-NA value.

Missing values – two types

Explicit missing values: NA declared.

Implicit missing values: data not there.



```
> stocks
# A tibble: 7 x 3
   year   qtr return
  <dbl> <dbl>  <dbl>
1  2015     1   1.88
2  2015     2   0.59
3  2015     3   0.35
4  2015     4  NA
5  2016     2   0.92
6  2016     3   0.17
7  2016     4   2.66
```

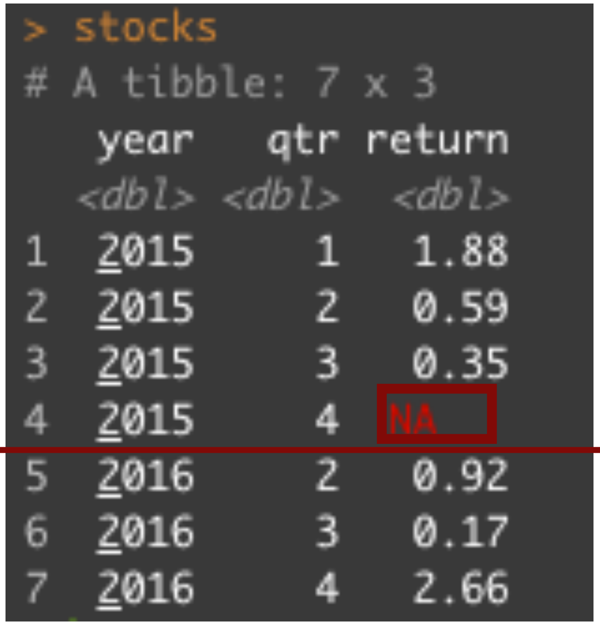
Missing Values

Example case: handling missing values.

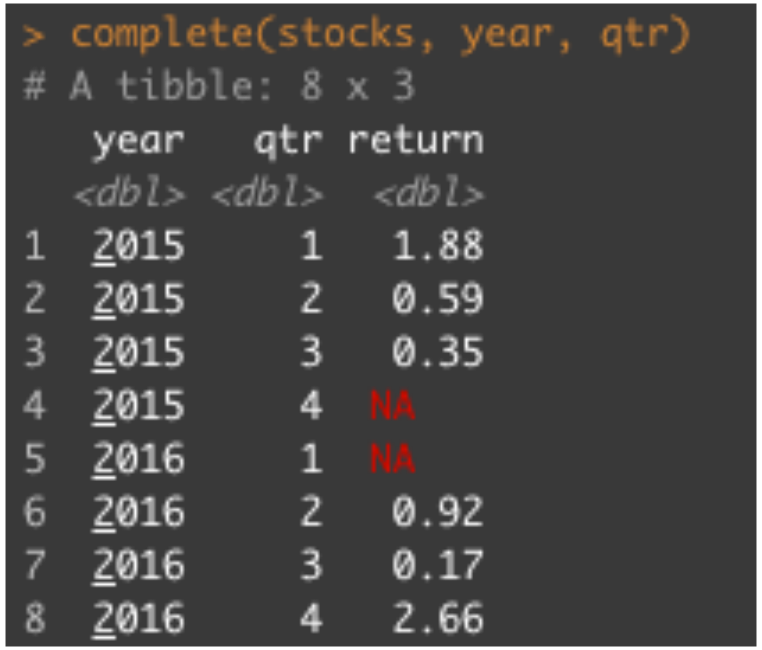
Explicit missing values: NA declared.

Implicit missing values: data not there.

complete(): make implicit missing values into explicit missing values.



```
> stocks
# A tibble: 7 x 3
   year   qtr return
  <dbl> <dbl>  <dbl>
1  2015     1   1.88
2  2015     2   0.59
3  2015     3   0.35
4  2015     4  NA
5  2016     2   0.92
6  2016     3   0.17
7  2016     4   2.66
```



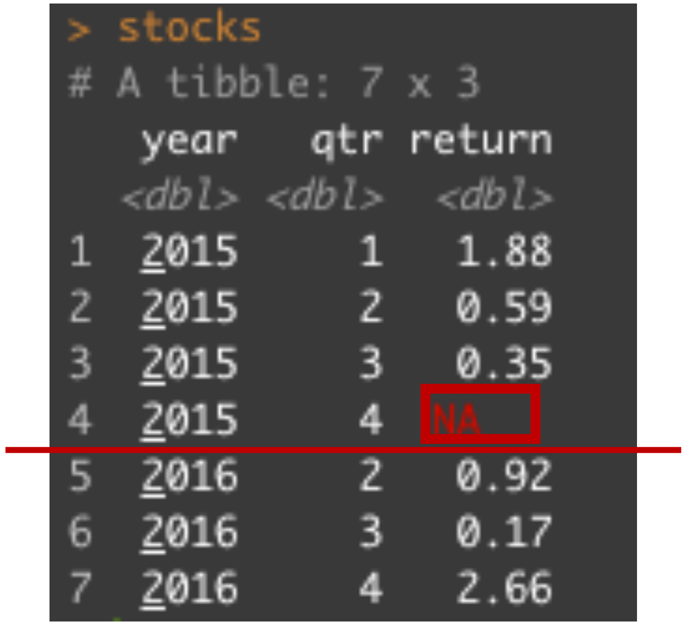
```
> complete(stocks, year, qtr)
# A tibble: 8 x 3
   year   qtr return
  <dbl> <dbl>  <dbl>
1  2015     1   1.88
2  2015     2   0.59
3  2015     3   0.35
4  2015     4  NA
5  2016     1  NA
6  2016     2   0.92
7  2016     3   0.17
8  2016     4   2.66
```

Missing Values

Example case: handling missing values.

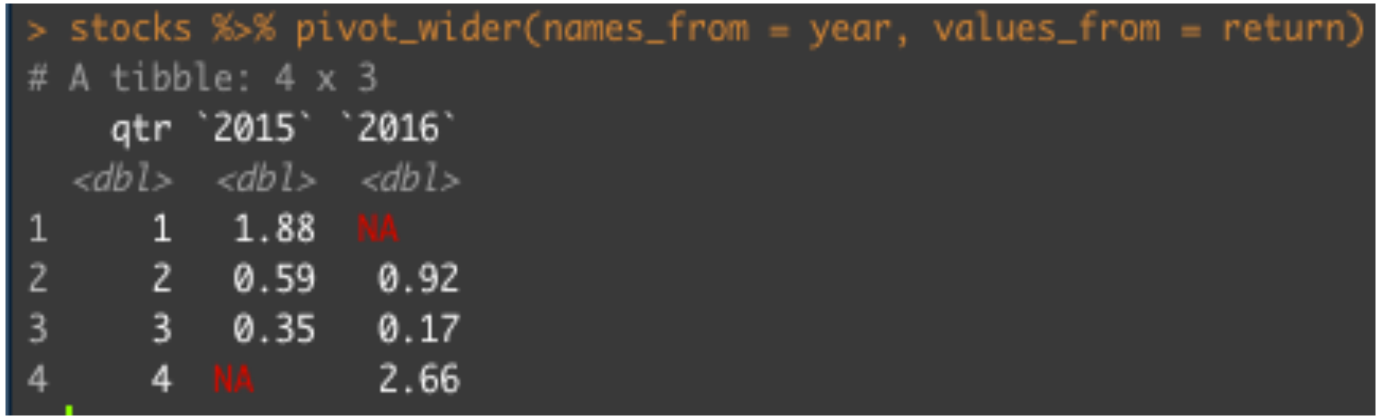
Explicit missing values: NA declared.

Implicit missing values: data not there.



```
> stocks
# A tibble: 7 x 3
   year   qtr return
  <dbl> <dbl>  <dbl>
1  2015     1   1.88
2  2015     2   0.59
3  2015     3   0.35
4  2015     4  NA
5  2016     2   0.92
6  2016     3   0.17
7  2016     4   2.66
```

Pivoting can reveal implicit missing values.



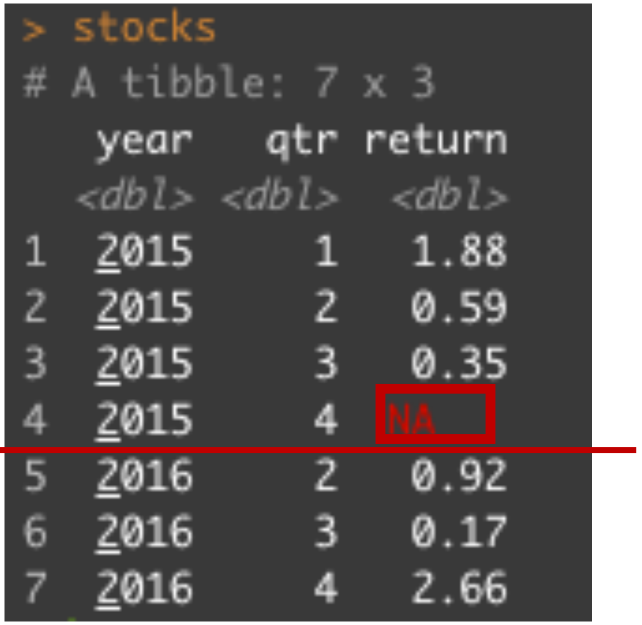
```
> stocks %>% pivot_wider(names_from = year, values_from = return)
# A tibble: 4 x 3
    qtr `2015` `2016`
  <dbl>  <dbl>  <dbl>
1     1   1.88  NA
2     2   0.59   0.92
3     3   0.35   0.17
4     4  NA      2.66
```

Missing Values

Example case: handling missing values.

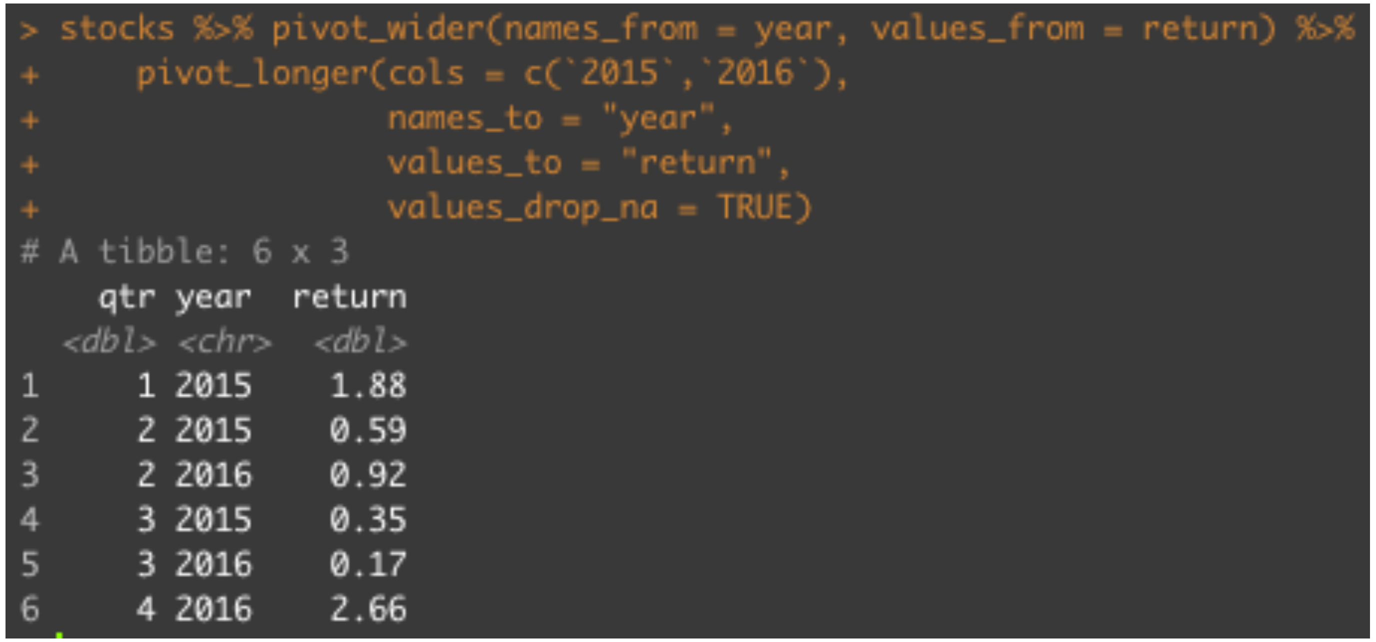
Explicit missing values: NA declared.

Implicit missing values: data not there.



```
> stocks
# A tibble: 7 x 3
   year   qtr return
  <dbl> <dbl>  <dbl>
1  2015     1   1.88
2  2015     2   0.59
3  2015     3   0.35
4  2015     4  NA
5  2016     2   0.92
6  2016     3   0.17
7  2016     4   2.66
```

Pivoting can remove explicit and implicit missing values.

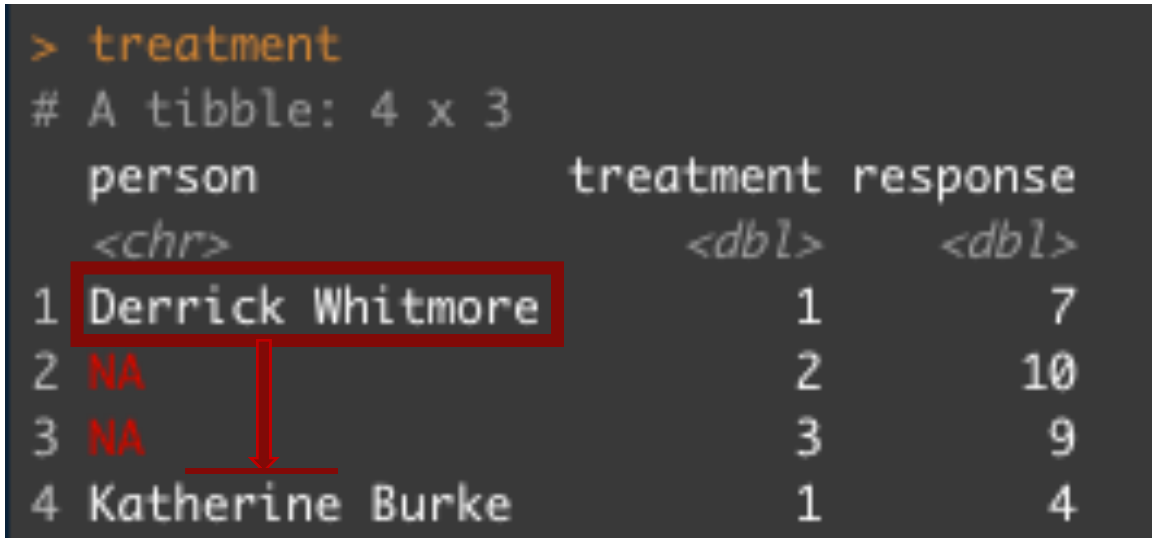


```
> stocks %>% pivot_wider(names_from = year, values_from = return) %>%
+     pivot_longer(cols = c(`2015`,`2016`),
+                  names_to = "year",
+                  values_to = "return",
+                  values_drop_na = TRUE)
# A tibble: 6 x 3
    qtr year  return
  <dbl> <chr>  <dbl>
1     1 2015    1.88
2     2 2015    0.59
3     2 2016    0.92
4     3 2015    0.35
5     3 2016    0.17
6     4 2016    2.66
```

Missing Values

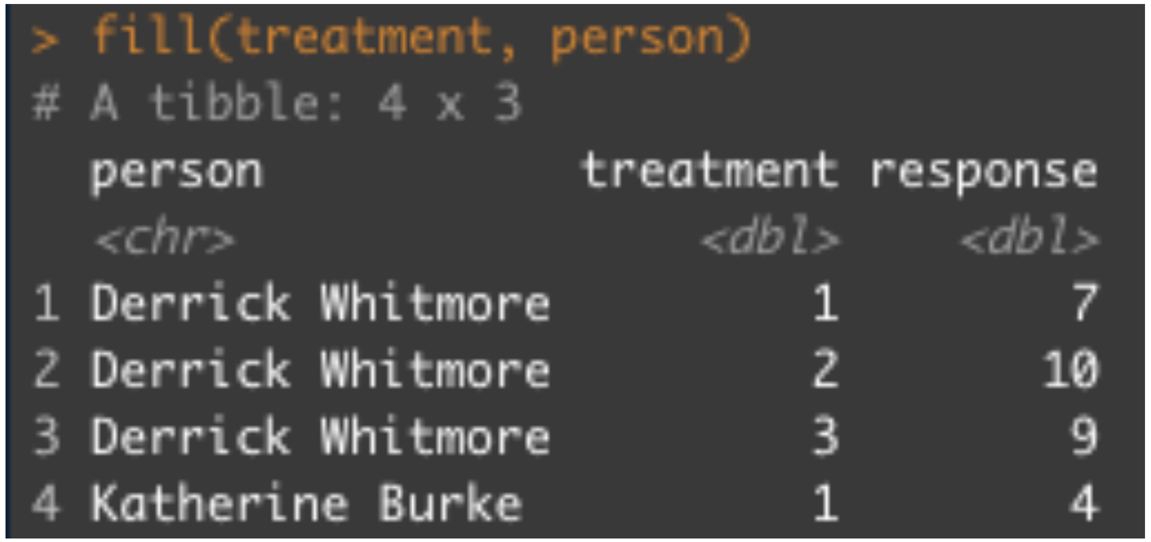
Example case: handling missing values.

Original table.



```
> treatment
# A tibble: 4 x 3
  person           treatment response
  <chr>                <dbl>    <dbl>
1 Derrick Whitmore         1        7
2 NA                       2       10
3 NA                       3        9
4 Katherine Burke          1        4
```

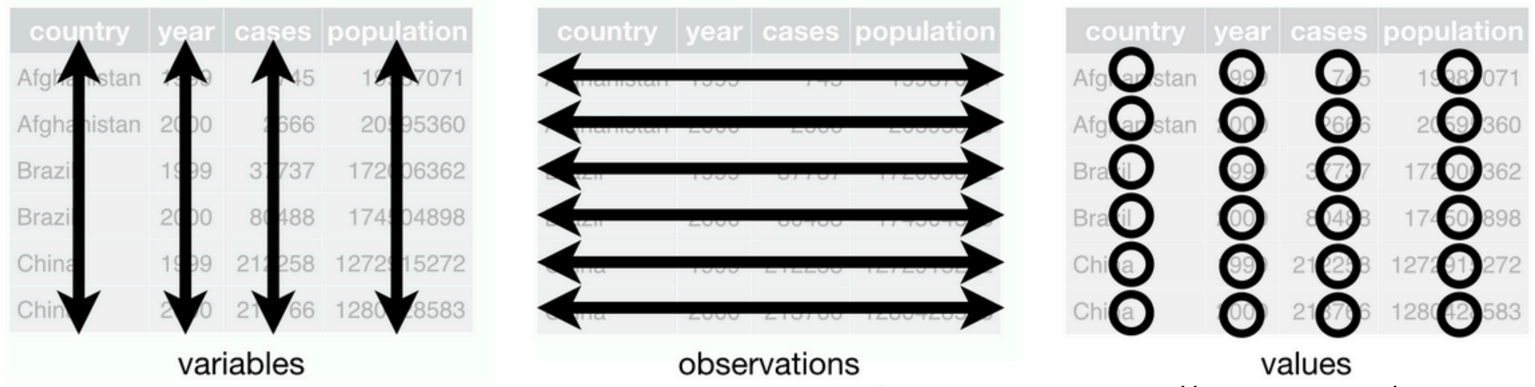
Applied fill() function.



```
> fill(treatment, person)
# A tibble: 4 x 3
  person           treatment response
  <chr>                <dbl>    <dbl>
1 Derrick Whitmore         1        7
2 Derrick Whitmore         2       10
3 Derrick Whitmore         3        9
4 Katherine Burke          1        4
```

Tidy Data Format & Data Wrangling

Tidy format:



Grolemund, Wickham. R for Data Science. https://r4ds.had.co.nz/index.html

<u>Simpler Rules:</u>

1. Put the data into a tibble.
2. Put each variable into a column.

Untidy data can be wrangled into an easier, tidy format:

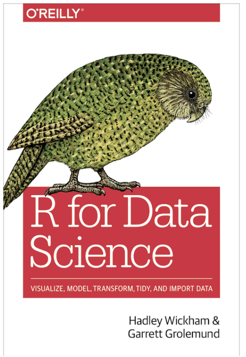
Data Wrangling Functions:	
pivot_longer()	pivot column names into new column and values of those columns into separate column
pivot_wider()	observations in multiple rows must be separated to new columns.
separate()	separate one column into multiple columns using a character for the split.
unite()	combine multiple columns into one column.
complete()	make implicit missing values into explicit NAs.
fill()	fill in NAs with prior non-NA value.

Acknowledgements

Sarah Wheelan
Wheelan Lab

Marchionni Lab
Claudio Zanettini

Stephanie Hicks
Margaret Taub
Rachael Workman
Frederick Tan
R Ladies Community


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund

Grolemund, Wickham. **R For Data Science**: Ch 12 Tidy Data. 2017 – present. https://r4ds.had.co.nz/tidy-data.html