

Tidy data with `tidyverse`

Day 3 - Introduction to Data Analysis with R

Selina Baldauf

Freie Universität Berlin - Theoretical Ecology

October 2, 2023



What is tidy data?

What is tidy data?

“**TIDY DATA** is a standard way of mapping the meaning of a dataset to its structure.”

—HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

each row an observation

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

Illustration from the [Openscapes blog](#) *Tidy Data for reproducibility, efficiency, and collaboration* by Julia Lowndes and Allison Horst

What is tidy data?

Let's look at some examples

Tidy

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

Non-tidy

floof	max	cat	donut	merlin	panda
gray	black	orange	gray	black	calico
gray	black	orange	calico		
floof	max	cat		panda	
donut	merlin				

Sometimes *raw data* is non-tidy because its structure is optimized for data entry or viewing rather than analysis.

Why tidy data?

The main advantages of tidy data is that the `tidyverse` packages are built to work with it.

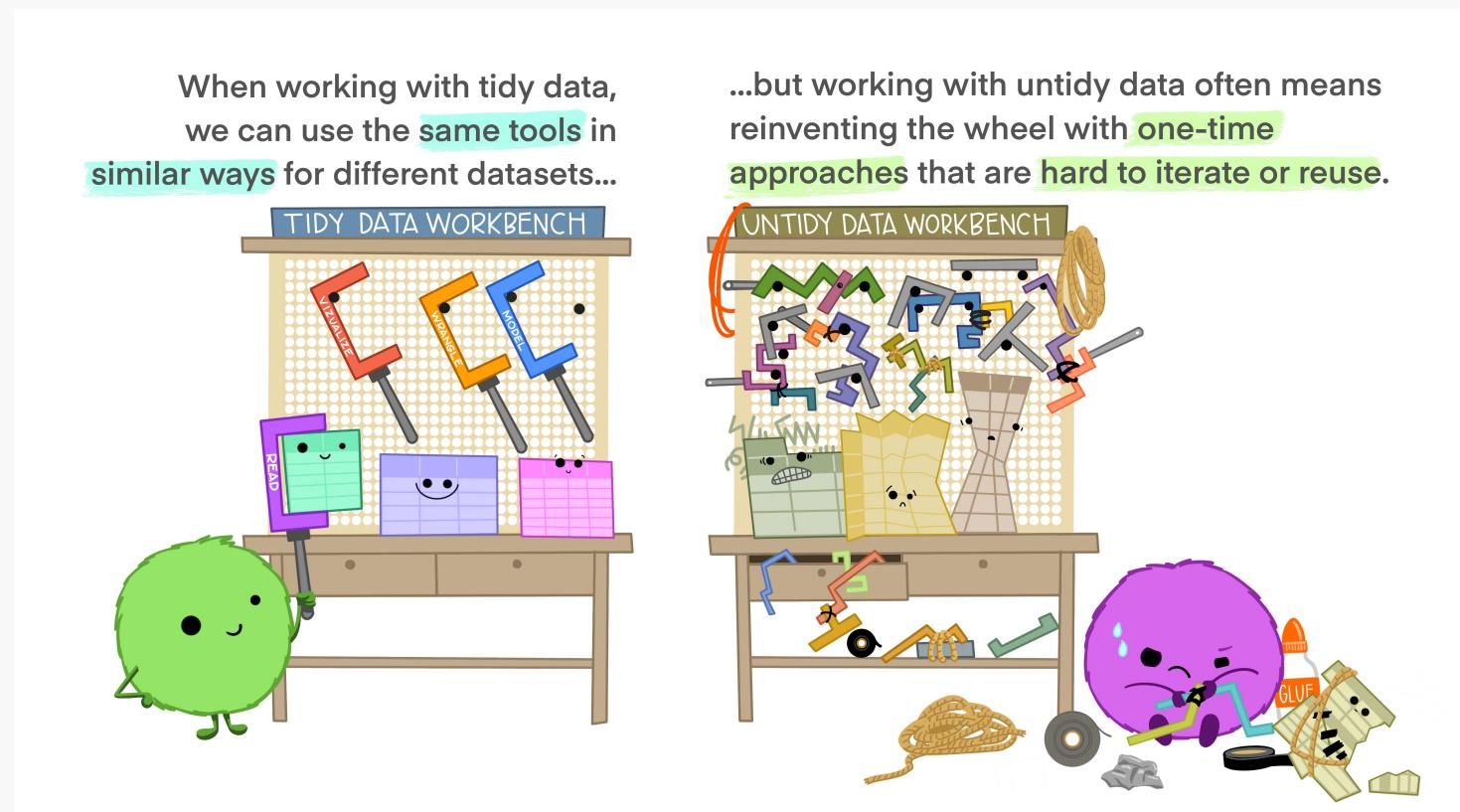


Illustration from the [Openscapes blog](#) *Tidy Data for reproducibility, efficiency, and collaboration* by Julia Lowndes and Allison Horst

Example

Let's go back to the city data set from earlier:

```
1 cities_tbl
2 #> # A tibble: 10 × 4
3 #>   city           population area_km2 country
4 #>   <chr>        <dbl>     <dbl> <chr>
5 #>   1 Istanbul      15100000    2576 Turkey
6 #>   2 Moscow        12500000    2561 Russia
7 #>   3 London         9000000    1572 UK
8 #>   4 Saint Petersburg 5400000    1439 Russia
9 #>   5 Berlin         3800000     891 Germany
10 #>  6 Madrid          3200000     604 Spain
11 #>  7 Kyiv            3000000     839 Ukraine
12 #>  8 Rome            2800000    1285 Italy
13 #>  9 Bucharest       2200000     228 Romania
14 #> 10 Paris           2100000     105 France
```

This already looks pretty tidy.

Same data different format

```
1 cities_untidy

#> # A tibble: 2 × 11
#>   type      Turkey_Istanbul Russia_Moscow UK_London `Russia_Saint Petersburg` 
#>   <chr>          <dbl>       <dbl>       <dbl>                      <dbl>
#> 1 population    15100000    12500000   9000000                  5400000
#> 2 area_km2      2576        2561        1572                      1439
#>   Germany_Berlin Spain_Madrid Ukraine_Kyiv Italy_Rome Romania_Bucharest
#>   <dbl>        <dbl>       <dbl>       <dbl>                      <dbl>
#> 1 3800000     3200000    3000000    2800000                  2200000
#> 2 891          604         839        1285                      228
#> # i 1 more variable: France_Paris <dbl>
```

What's not tidy here?

- Each row has multiple observations
- At the same time, each observation is split across multiple rows
- Country and city variable are split into multiple columns
- Country and city variable values are united to one value

Let's tidy this data using functions from the `tidyverse` package!

pivot_longer()

One variable split into multiple columns can be solved with pivot_longer

```
#> # A tibble: 2 × 11
#>   type      Turkey_Istanbul Russia_Moscow UK_London `Russia_Saint Petersburg` 
#>   <chr>          <dbl>       <dbl>       <dbl>                      <dbl>
#> 1 population    15100000  12500000  9000000                  5400000
#> 2 area_km2      2576        2561       1572                     1439
#> Germany_Berlin Spain_Madrid Ukraine_Kyiv Italy_Rome Romania_Bucharest
#>   <dbl>       <dbl>       <dbl>       <dbl>                      <dbl>
#> 1 3800000     3200000    3000000   2800000                  2200000
#> 2 891          604         839        1285                     228
#> # i 1 more variable: France_Paris <dbl>
```

```
1 step1 <- pivot_longer(
2   cities_untidy,                      # the tibble
3   cols = Turkey_Istanbul:France_Paris, # the columns to pivot from:to
4   names_to = "location",               # name of the new column
5   values_to = "value")                # name of the value column
```



```
#> # A tibble: 20 × 3
#>   type      location           value
#>   <chr>     <chr>            <dbl>
#> 1 population Turkey_Istanbul 15100000
#> 2 population Russia_Moscow  12500000
#> 3 population UK_London      9000000
#> 4 population Russia_Saint Petersburg 5400000
#> # i 16 more rows
```

pivot_longer()

One variable split into multiple columns can be solved with pivot_longer

```
1 step1 <- pivot_longer(  
2   cities_untidy,                      # the tibble  
3   cols = Turkey_Istanbul:France_Paris,  # the columns to pivot from:to  
4   names_to = "location",                # name of the new column  
5   values_to = "value")                 # name of the value column
```

Another way to select the columns to pivot:

```
1 step1 <- pivot_longer(  
2   cities_untidy,                      # the tibble  
3   cols = !type,                      # All columns except type#<<  
4   names_to = "location",                # name of the new column  
5   values_to = "value")                 # name of the value column
```

`separate_wider_delim()`

Multiple variable values that are united into one can be separated using `separate_wider_delim`

```
#> # A tibble: 20 × 3
#>   type      location       value
#>   <chr>     <chr>        <dbl>
#> 1 population Turkey_Istanbul 15100000
#> 2 population Russia_Moscow   12500000
#> # i 18 more rows

1 step2 <- separate_wider_delim(
2   step1,                               # the tibble
3   location,                            # the column to separate
4   delim = "_",                         # the separator
5   names = c("country", "city")) # names of new columns

#> # A tibble: 20 × 4
#>   type      country city       value
#>   <chr>     <chr>   <chr>     <dbl>
#> 1 population Turkey  Istanbul 15100000
#> 2 population Russia  Moscow   12500000
#> # i 18 more rows
```

The opposite function exists as well and is called `unite`. Check out `?unite` for details.

pivot_wider()

One observation split into multiple rows can solved with `pivot_wider`

```
#> # A tibble: 20 × 4
#>   type      country city        value
#>   <chr>     <chr>   <chr>      <dbl>
#> 1 population Turkey  Istanbul  15100000
#> 2 population Russia Moscow   12500000
#> # i 18 more rows

1 step3 <- pivot_wider(
2   step2,                               # the tibble
3   names_from = type,                  # the variables
4   values_from = value)               # the values

#> # A tibble: 10 × 4
#>   country city        population area_km2
#>   <chr>   <chr>      <dbl>      <dbl>
#> 1 Turkey  Istanbul    15100000    2576
#> 2 Russia  Moscow     12500000    2561
#> 3 UK      London     9000000     1572
#> 4 Russia  Saint Petersburg 5400000    1439
#> 5 Germany Berlin    3800000     891
#> # i 5 more rows
```

All steps in 1

We can also use a pipe to do all these steps in one:

```
1 cities_tidy <- cities_untidy |>
2   pivot_longer(Turkey_Istanbul:France_Paris,
3     names_to = "location",
4     values_to = "values"
5   ) |>
6   separate(location,
7     sep = "_",
8     into = c("country", "city")
9   ) |>
10  pivot_wider(
11    names_from = type,
12    values_from = values
13  )
```

Now you

Task (30 min)

Tidy data with `tidyverse`

Find the task description [here](#)

