Tidy data with tidyr

Day 3 - Introduction to Data Analysis with R

Selina Baldauf
Freie Universität Berlin - Theoretical Ecology

October 11, 2025

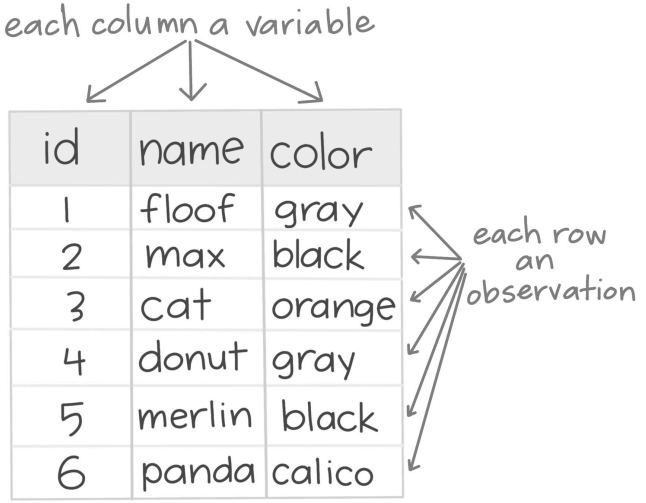
What is tidy data?

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



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.

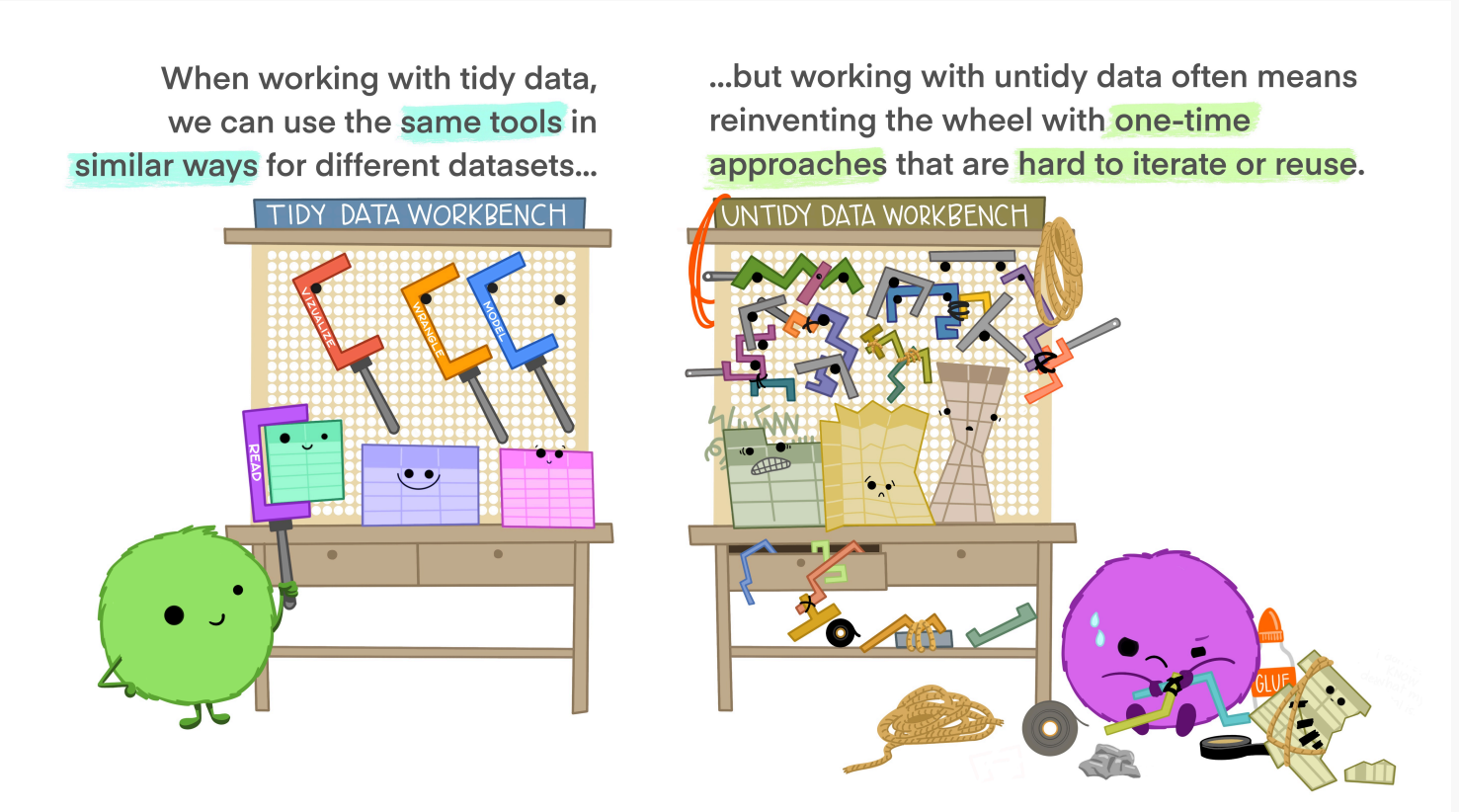


Illustration from the Openscapes blog *Tidy Data for reproducibility, efficiency, and collaboration* by Julia Lowndes and Allison Horst

Example

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

- Expand to reproduce the data

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

This already looks pretty tidy.

Same data different format

▶ Expand to reproduce the data

```
cities_untidy
```

```
#> # A tibble: 2 × 11
#>   type            Turkey_Istanbul Russia_Moscow UK_London Russia_Saint Petersb…¹
#>   <chr>                     <dbl>         <dbl>     <dbl>                  <dbl>
#> 1 population_size        15100000      12500000   9000000                5400000
#> 2 city_area                 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 abbreviated name: ¹`Russia_Saint Petersburg`
#> # ℹ 1 more variable: France_Paris <dbl>
```

What's not tidy here?

- Each row has multiple observation
- 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

The `tidyr` pacakge

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

First load the package with either

```
library(tidyr)
```

or

```
library(tidyverse)
```

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 Petersb…¹
#>   <chr>                     <dbl>         <dbl>     <dbl>                  <dbl>
#> 1 population_size        15100000      12500000   9000000                5400000
#> 2 city_area                 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 abbreviated name: ¹`Russia_Saint Petersburg`
#> # ℹ 1 more variable: France_Paris <dbl>
```

pivot_longer()

One variable split into multiple columns can be solved with pivot_longer

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

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

pivot_longer()

One variable split into multiple columns can be solved with `pivot_longer`

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

Another way to select the columns to pivot:

```
step1 <- pivot_longer(
  cities_untidy, # the tibble
  cols = !type, # All columns except type#<<
  names_to = "location", # name of the new column
  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_size Turkey_Istanbul 15100000
#> 2 population_size Russia_Moscow   12500000
#> # ℹ 18 more rows
```

```
step2 <- separate_wider_delim(
  step1, # the tibble
  location, # the column to separate
  delim = "_", # the separator
  names = c("country", "city_name") # names of new columns
)
```

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

```
step3 <- pivot_wider(
  step2, # the tibble
  names_from = type, # the variables
  values_from = value # the values
)
```

```
#> # A tibble: 10 × 4
#>   country city_name        population_size city_area
#>   <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:

```
cities_tidy <- cities_untidy |>
  pivot_longer(
    Turkey_Istanbul:France_Paris,
    names_to = "location",
    values_to = "values"
  ) |>
  separate_wider_delim(
    location,
    delim = "_",
    names = c("country", "city_name")
  ) |>
  pivot_wider(
    names_from = type,
    values_from = values
  )
```

Remove missing values with `drop_na()`

Drop rows with missing values:

```
# drop rows with missing values in any column
drop_na(and_vertebrates)
# drop rows with missing values in weight column
drop_na(and_vertebrates, weight_g)
# drop rows with missing values in weight and species columns
drop_na(and_vertebrates, weight_g, species)
```

This is an easier and more intuitive alternative to `filter(!is.na(...))`.

Now you

Task (30 min)

Tidy data with tidyr

Find the task description here