THE UNIVERSITY *of* EDINBURGH

Tidy(ing) data in R

Bram van Bunnik

Core Scientist
Roslin Institute
University of Edinburgh

Tidy data - what is it & why do I need it?

- Huge amount of effort is spent cleaning data, estimates of up to 80% of analysis is spent on cleaning and preparing data
- Part of data cleaning is data tidying.
- Tidy datasets are easy to manipulate, model and visualise
- Tidy datasets have a specific structure:
- One table per dataset!!!

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

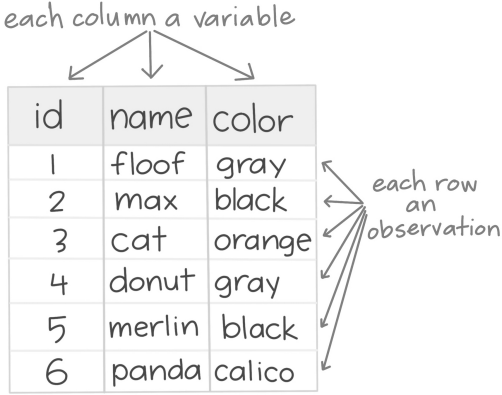
Tidy data

Tidy Data

Tidy data is a standard way of mapping the meaning of a dataset to its structure

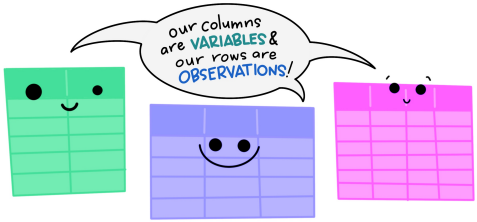
Hadley Wickham

- Tidy datasets have a specific structure:
  - Each column is a variable
  - Each row is an observation
  - Each cell is a single value



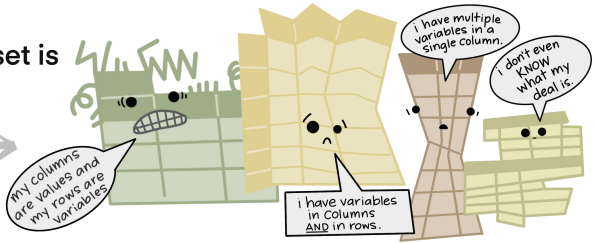
All tidy datasets are alike...

The standard structure of tidy data means that "tidy datasets are all alike..."



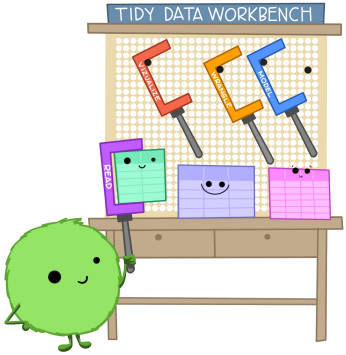
"...but every messy dataset is messy in its own way."

–HADLEY WICKHAM

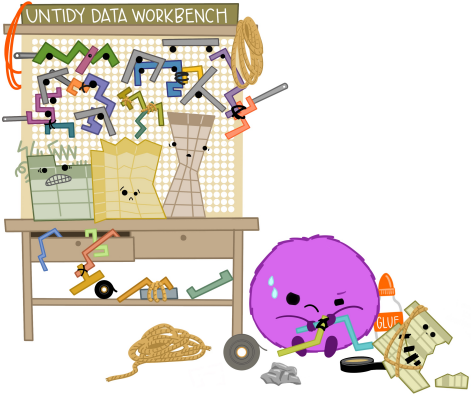


Tidy datasets are easier to work with...

When working with tidy data, we can use the same tools in similar ways for different datasets...

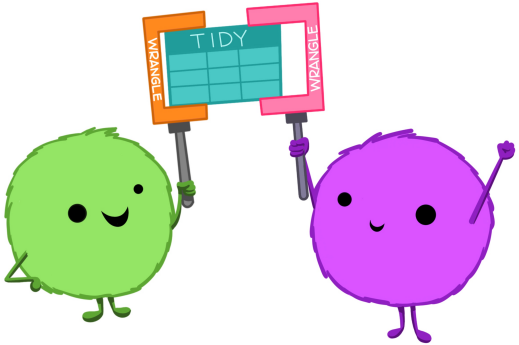


...but working with untidy data often means reinventing the wheel with one-time approaches that are hard to iterate or reuse.

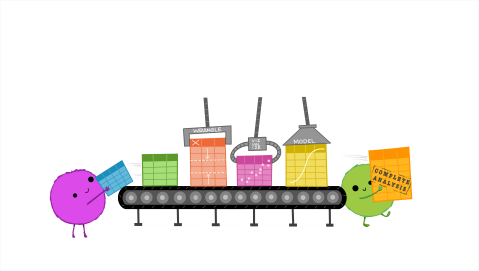


Tidy datasets make sharing and collaborating easier...

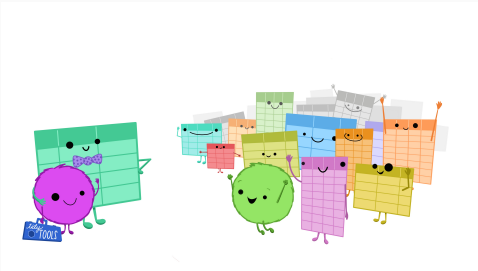
We all have the same tools we (can) use...



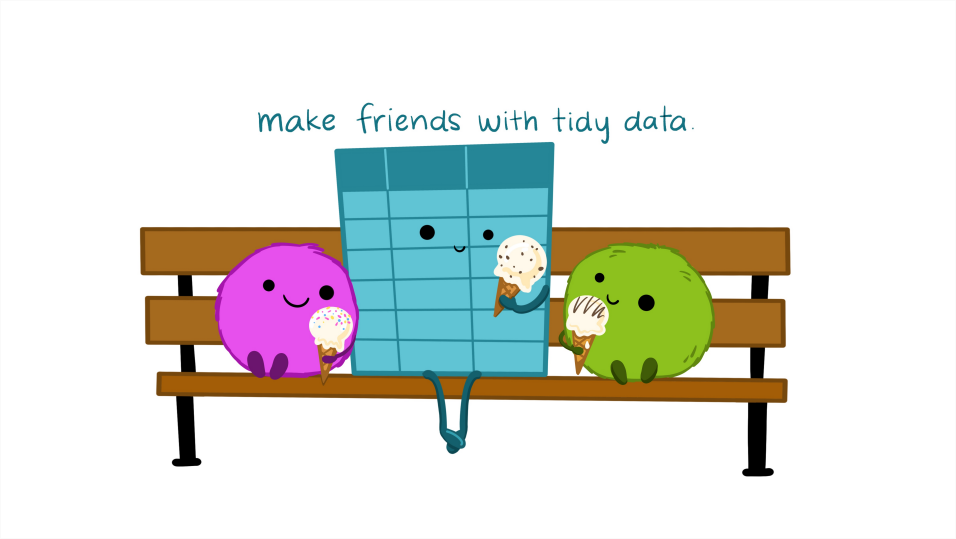
Tidy datasets are easier for automation and iteration



And it makes other (tidy) datasets seem more welcoming!



So...



Some examples...

We have some data about an imaginary classroom in a format commonly seen in the wild. The table has three columns and four rows, and both rows and columns are labeled.

```
classroom
#> # A tibble: 4 × 4
#>   name   quiz1 quiz2 test1
#>   <chr>  <chr> <chr> <chr>
#> 1 Billy  NA    D     C
#> 2 Suzy   F     NA    NA
#> 3 Lionel B     C     B
#> 4 Jenny  A     A     B
```

There are many ways to structure the same underlying data. We could for example transpose the dataset and it would still show the same data:

```
classroom
#> # A tibble: 3 × 5
#>   assessment Billy Suzy  Lionel Jenny
#>   <chr>      <chr> <chr> <chr>  <chr>
#> 1 quiz1      NA    F     B      A
#> 2 quiz2      D     NA    C      A
#> 3 test1      C     NA    B      B
```

The data is the same, but the layout is different. Our vocabulary of rows and columns is simply not rich enough to describe why the two tables represent the same data. In addition to appearance, we need a way to describe the underlying semantics, or meaning, of the values displayed in the table.

Data semantics

- A dataset is a collection of values, usually either numbers (if quantitative) or strings (if qualitative).
- Values are organised in two ways:
  Every value belongs to a variable and an observation.
- A variable contains all values that measure the same underlying attribute (like height, temperature, duration) across units.
- An observation contains all values measured on the same unit (like a person, or a day, or a race) across attributes.

Our example made tidy...

```
library(tidyverse)

classroom2 <- classroom |>
  pivot_longer(quiz1:test1,
               names_to = "assessment",
              values_to = "grade") |>
  arrange(name, assessment)

  classroom2
#> # A tibble: 12 × 3
#>    name   assessment grade
#>    <chr>  <chr>      <chr>
#>  1 Billy  quiz1      NA
#>  2 Billy  quiz2      D
#>  3 Billy  test1      C
#>  4 Jenny  quiz1      A
#>  5 Jenny  quiz2      A
#>  6 Jenny  test1      B
#>  7 Lionel quiz1      B
#>  8 Lionel quiz2      C
#>  9 Lionel test1      B
#> 10 Suzy   quiz1      F
#> # i 2 more rows
```

This makes the values, variables, and observations more clear.

The dataset contains 36 values representing 3 variables and 12 observations.

The variables are:

- "name", four possible values (Billy, Suzy, Lionel, and Jenny).
- "assessment", three possible values (quiz1, quiz2, and test1).
- "grade", five or six values (A, B, C, D, F, NA).

Messy mistakes

Five most common ”messy mistakes”

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

Surprisingly, most messy datasets, including types of messiness not explicitly described above, can be tidied with a small set of tools: pivoting (longer and wider) and separating.

Column headers being values...

```
> chick_wide
# A tibble: 50 x 14
   Chick  Diet     T0     T2     T4     T6     T8    T10    T12    T14    T16    T18    T20    T21
   <ord> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 1     1          42     51     59     64     76     93    106    125    149    171    199    205
 2 2     1          40     49     58     72     84    103    122    138    162    187    209    215
 3 3     1          43     39     55     67     84     99    115    138    163    187    198    202
 4 4     1          42     49     56     67     74     87    102    108    136    154    160    157
 5 5     1          41     42     48     60     79    106    141    164    197    199    220    223
 6 6     1          41     49     59     74     97    124    141    148    155    160    160    157
 7 7     1          41     49     57     71     89    112    146    174    218    250    288    305
 8 8     1          42     50     61     71     84     93    110    116    126    134    125     NA
 9 9     1          42     51     59     68     85     96     90     92     93    100    100     98
10 10    1          41     44     52     63     74     81     89     96    101    112    120    124
# i 40 more rows
# i Use `print(n = ...)` to see more rows
```

Data shows the weight of chicks (Chick) on different timepoints (T0...T21) for different diets (Diet).

In this dataset the columns starting with T are the timepoints at which a measerument was taken.

This data set is not tidy as each row has multiple observations...

How to tidy?

Tidying this dataset is relatively straightforward:

```
chick_wide |>
pivot_longer(cols = -c(Chick, Diet),
        names_to = "time",
        values_to = "weight") |>
        separate(time, into=c(NA, "time"), sep=1)
```

```
# A tibble: 600 x 4
Chick Diet   time  weight
<ord> <fct>  <chr>  <dbl>
 1 1     1          0     42
 2 1     1          2     51
 3 1     1          4     59
 4 1     1          6     64
 5 1     1          8     76
 6 1     1         10     93
 7 1     1         12    106
 8 1     1         14    125
 9 1     1         16    149
10 1     1         18    171
# i 590 more rows
# i Use `print(n = ...)` to see more rows
```

Variables are stored in both rows and columns

```
#> # A tibble: 5,769 × 22
#>    iso2   year    m04   m514   m014 m1524 m2534 m3544 m4554 m5564   m65    mu
#>    <chr> <int>  <int>  <int>  <int> <int> <int> <int> <int> <int> <int> <int>
#>  1 AD     1989     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>  2 AD     1990     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>  3 AD     1991     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>  4 AD     1992     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>  5 AD     1993     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>  6 AD     1994     NA     NA     NA     NA     NA     NA     NA     NA     NA
#>  7 AD     1996     NA     NA      0      0      0      4      1      0      0    NA
#>  8 AD     1997     NA     NA      0      0      1      2      2      1      6    NA
#>  9 AD     1998     NA     NA      0      0      0      1      0      0      0    NA
#> 10 AD     1999     NA     NA      0      0      0      1      1      0      0    NA
#> # ℹ 5,759 more rows
#> # ℹ 10 more variables: f04 <int>, f514 <int>, f014 <int>, f1524 <int>,
#> #   f2534 <int>, f3544 <int>, f4554 <int>, f5564 <int>, f65 <int>,
#> #   fu <int>
```

- Dataset records the counts of confirmed tuberculosis cases by country, year, and demographic group.
- The demographic groups are broken down by sex (m, f) and age (0-14, 15-25, 25-34, 35-44, 45-54, 55-64, 65 and older, unknown).

How would you go about tidying this dataset?

Multiple variables are stored in one column

Original dataset (kindly provided by Erhan Yalcindag):

```
A tibble: 386 x 83
...1                ASV1  ASV2  ASV3  ASV4  ASV5  ASV6  ASV7  ASV8  ASV9 ASV10
<chr>               <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Querylength        233   233   292   281   281   281   283   292   292   292
 2 1504-Jan20-Fec_1456_Calf   100     0     0     0     0     0     0     0     0     0
 3 1504-Jan20-Fec_1868_Calf     0   100     0     0     0     0     0     0     0     0
 4 1504-Jan20-Fec_216_Calf      0     0     0     0     0     0     0     0     0     0
 5 1504-Jan20-Fec_396_Calf      0     0   100     0     0     0     0     0     0     0
 6 1504-Nov19-Fec_1107_Calf     0   100     0     0     0     0     0     0     0     0
 7 1504-Nov19-Fec_1172_Calf     0     0     0   100     0     0     0     0     0     0
 8 1504-Nov19-Fec_1232_Calf     0     0     0     0     0     0     0     0     0     0
 9 1504-Nov19-Fec_2434_Calf     0     0     0     0   100     0     0     0     0     0
10 1504-Nov19-Fec_278_Calf    100     0     0     0     0     0     0     0     0     0
# i 376 more rows
# i 72 more variables: ASV11 <dbl>, ASV12 <dbl>, ASV13 <dbl>,
```

Two (main) issues here:

- Multiple variables in one column.
- Multiple measurements in one row.

(Which other issue(s) can you spot?)

Tidied data:

```
data |>
  # remove first row
  filter(!row_number() %in% c(1)) |>
  rename("sample" = "...1") |>
  # split sample into different columns
  separate(col = sample,
       into = c("sample", "date", "stage", "reads", "age")) |>
  # re-order meaningfully
  mutate(date = fct(date,
       levels = c("Nov19", "Jan20", "May20", "July20"))) |>
  mutate(stage = fct(stage)) |>
  mutate(reads = as.numeric(reads)) |>
  mutate(age = fct(age))
```

```
# A tibble: 385 x 87
sample date   stage reads age   ASV1  ASV2  ASV3  ASV4
<chr>  <fct>  <fct> <dbl> <fct> <dbl> <dbl> <dbl> <dbl>
1 1504   Jan20 Fec    1456 Calf    100     0     0     0
2 1504   Jan20 Fec    1868 Calf      0   100     0     0
3 1504   Jan20 Fec     216 Calf      0     0     0     0
4 1504   Jan20 Fec     396 Calf      0     0   100     0
5 1504   Nov19 Fec    1107 Calf      0   100     0     0
6 1504   Nov19 Fec    1172 Calf      0     0     0   100
7 1504   Nov19 Fec    1232 Calf      0     0     0     0
8 1504   Nov19 Fec    2434 Calf      0     0     0     0
9 1504   Nov19 Fec     278 Calf    100     0     0     0
10 1504   Nov19 Fec    7201 Calf      0     0     0     0
# i 375 more rows
# i 60 more variables: ASV5 <dbl>, ASV6 <dbl>,...
```

Tidy further by creating a ”long” dataset

```
data |>
  pivot_longer(cols=starts_with("ASV"),
       names_to = "ASV", values_to = "identity") |>
  mutate(ASV = fct(ASV))
```

```
# A tibble: 31,570 x 7
sample date   stage reads age   ASV   identity
<chr>  <fct>  <fct> <dbl> <fct> <fct>    <dbl>
1 1504   Jan20 Fec    1456 Calf  ASV1       100
2 1504   Jan20 Fec    1456 Calf  ASV2         0
3 1504   Jan20 Fec    1456 Calf  ASV3         0
4 1504   Jan20 Fec    1456 Calf  ASV4         0
5 1504   Jan20 Fec    1456 Calf  ASV5         0
6 1504   Jan20 Fec    1456 Calf  ASV6         0
7 1504   Jan20 Fec    1456 Calf  ASV7         0
8 1504   Jan20 Fec    1456 Calf  ASV8         0
9 1504   Jan20 Fec    1456 Calf  ASV9         0
10 1504   Jan20 Fec    1456 Calf  ASV10        0
# i 31,560 more rows
# i Use `print(n = ...)` to see more rows
```

Misc data issues

- Excel... It has many functions that don't play nicely with data analyis.
- Colour coding/formatting cells to denote information... This should be a variable
- Merging cells... Breaks the rules of variables / observations
- Top row should always be header names... PLEASE, not blank rows for "aesthetics"
- Tidy data is rarely aesthetically pleasing to humans (and it is not intended to be!)
- Tidy data is specific to your research question

Useful functions for tidying data

- read_csv(), read_tsv(), read_∗(): read in text file in a specific format, returns a tibble
- View() or glimpse(): view the entire dataset
- pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE): Lengthen data by collapsing several columns into two
- pivot_wider(): the opposite of pivot_longer
- separate(), separate_wider_delim(), separate_wider_position(): turns a single character column into multiple columns
- separate_longer_∗(): turn into multiple rows
- mutate(): create new column(s) that are functions of existing variables. It can also modify or delete columns

Useful functions for tidying data, cont.

- select(): select variables in a dataframe
- filter(): subset dataframe to only include rows that satisfy condition
- bind_rows(), bind_cols(): bind any number of df's by row / column
- {inner, left, right, full}_join(): Mutating joins add columns from y to x, matching observations based on the keys. There are four mutating joins: the inner join, and the three outer joins.
- clean_names() (from janitor package): Cleans names of a df. Resulting names are unique and consist only of the "_" character, numbers, and letters
- Any others??

Resources

Various cheatsheets from R-Studio (and others):
https://github.com/rstudio/cheatsheets/

Some noteworthy ones:

- https://github.com/rstudio/cheatsheets/blob/main/pngs/tidyr.png
- https://github.com/rstudio/cheatsheets/blob/main/pngs/rstudio-ide.png
- https://github.com/rstudio/cheatsheets/blob/main/pngs/gtsummary.png
- https://github.com/rstudio/cheatsheets/blob/main/pngs/factors.png

And remember: