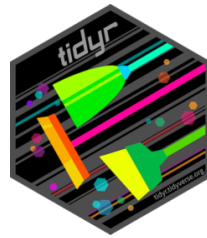


Principles of tidy data

An introduction to tabular data structures



Roland Krause | rworkshop | 2021-09-08

Session set-up

Learning objectives

- Principles of *tidy data* to structure data in tables
- Find errors in existing data sets
- Structure data programmatically



Comments

- Cleaning data also requires `dplyr`
- `tidyr` and `dplyr` are intertwined
- Focus on only *tidy data*
- Introduction of `tidyr` ways
- Other `tidyr` functionality will be covered in `dplyr` and `broom` lectures.

Typical flow of data

Source data →

- Experimental data
- External data sets
- Manually collected data and meta data

Intermediate →

- Derived data
- Computation
- Manual curation
- **Tidy data**



Analysis →

- Exploratory analysis
- Statistical models
- Hypothesis testing



Manuscript →

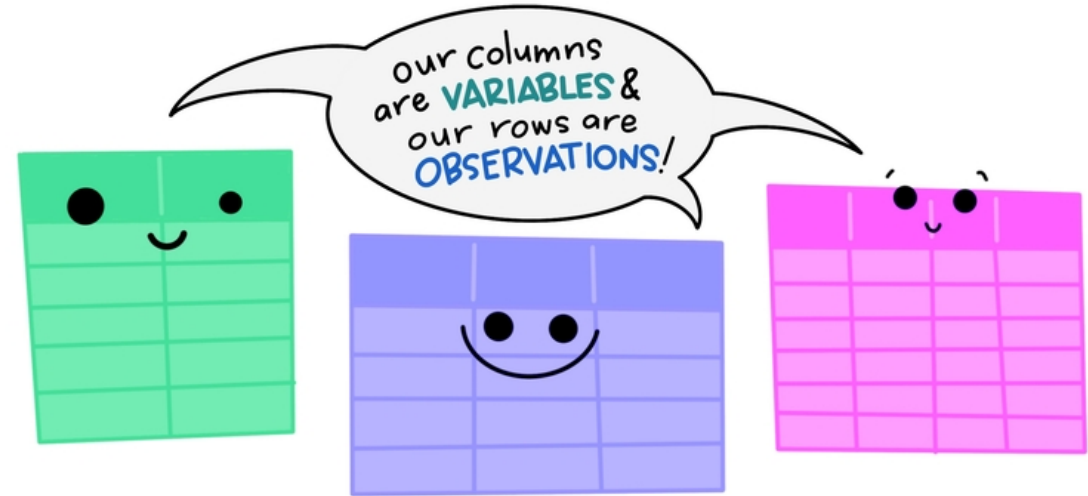
- Can you reproduce your work?
 - All numbers
 - Summaries
 - Images



One workflow

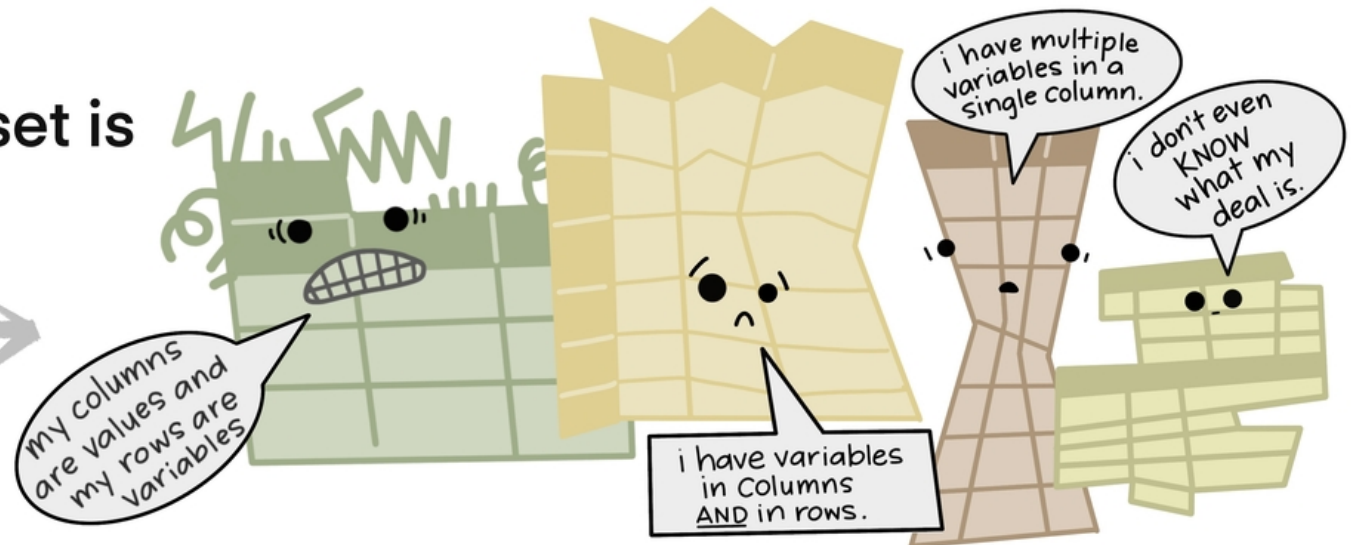
- No editing of data at any step
- All code needed to reproduce from one ingestion to manuscript coded and repeatable

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



Credit: Artwork by [Allison Horst](#)

A definition of tidy data

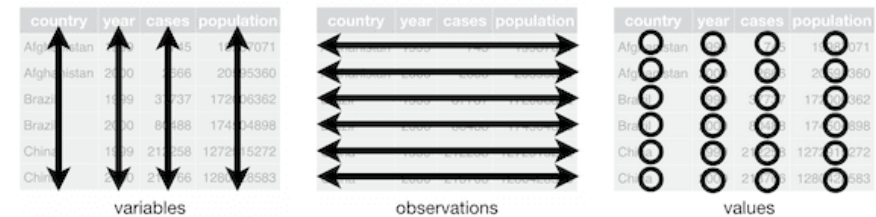
Basic consideration

- **Variable:** A quantity, quality, or property that you can measure.
- **Observation:** A set of values that display the relationship between variables. To be an observation, values need to be measured under similar conditions, usually measured on the same observational unit at the same time.
- **Value:** The state of a variable that you observe when you measure it.

source: [Garret Grolemund](#) and `vignette("tidy-data")`

What is tidy data

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.



Tidying messy data

For clean data ensure that

- Column names are easy to use and informative
- Row names are easy to use and informative
- Obvious mistakes in the data have been removed
- Variable values are internally consistent
- Appropriate transformed variables have been added

x Bad data exercise

- The following table lists data from two questionnaires --[stai](#) and [rec](#)-- recorded in different languages.
- What's wrong with [the Excel](#) sheet?
- Which problems are [tidy](#) issues?

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																	
2				Native					Scale 1-5				REC Scale 1-4				
3	N	ID	Gender	Language	Birthday		morale (by STAI test)	STAI		Language		REC-1	Rec-2	REC-3	Language		Comment
4	R	1	male	German	01/12/2005			92	1=5,2=4, 3=3	d		1	1	4	D		Was lookir
5	L	2	m	German	26/2/1992			122	1=5,2=4, 3=3	d		4	2	1	F		
6		2	m	GErman	26/2/1992			123	1=5,2=4, 3=2	d		2	3	5	F		
7		2	male	German	26/2/1992			122	1=5,2=4, 3=3	d		3	3	2	D		
8	M	4	f	German	12-03-2002			112	1=5,2=4, 3=4	d		3	3	3	Deutsch		Looks nice
9	H	5	m	French	12/11/70			99	1=3,2=3, 3=3,4=2			4	3	4	French		
10	H	5			11-12-70							4	2	12	Deutsch		
11	L	7	f	Französisch				101	1=3,2=3, 3=3	f		4	2	1	French		
12	D	8	f	Französisch	2005-12-10			140	1=5,2=4, 3=3	f		1	2	1	F		
13	H	9	male	Deutsch	1972-10-01			138	1=5,2=4, 3=3	d		3	3	3	Deutsch		
14	K	10	männlich	Deutsch	5.4.1970			142	1=5,2=4, 3=3	d		4	3	4	Deutsch		
15	Ir	11	weiblich	Deutsch	1/1/1990			99	1=2,2=4,3=3,	d		4	2	2	Deutsch		
16								shorter time = better				37	29	42	Sum		
17												3.0833333	2.4166667	3.5	average (mean)		
18																	
19																	
20																	

Tidy errors

Error	Tidy violation	Comment
Person name	No	Data protection violation
Identical column names	Yes	Variable error
Inconsistent variables names	No	Bad practice
Non-English columns names	No	Bad practice
Color coding	No	The horror, the horror
Inconsistent dates	No	Use ISO8601
Multiple columns for one item	Yes	One observation per line
Redundant information	Yes	Each variable is in its own column
Repeated rows	Yes	Each observation is in its own row
Missing coding	Yes/No	Each value in its own cell
Unnecessary information (Birthdate, comments)	No	Bad practice
Name of the table	No	Bad practice

Data cleaning exercise

Clean the "bad table"

- Bring data into shape such that it conforms to tidy data requirements
- Pay attention to details of format, less to actual data
- Do not use R manipulaiont for doing the manipulations, just capture the results.

tribble()

A handy way to jot down tibbles.

```
clean_data <- tribble(  
  ~family, ~patient,  
  "F1", 101,  
  "F1", 102,  
  "F2", 201  
)
```

clean_data

```
# A tibble: 3 × 2  
  family patient  
  <chr>    <dbl>  
1 F1      101  
2 F1      102  
3 F2      201
```


A possible solution

```
readxl::read_excel("data/bad-table-psych.xlsx",
  skip = 2,
  n_max = 12) %>%
  select(-Name, -Birthday, Comment) %>%
  select(where(~ !(all(is.na(.)) | all(. == "")))) %>% #
  janitor::remove_empty("cols") %>%
  janitor::clean_names() %>%
  rename( "test-time" = x9,
    "rec-lang" = language_11) %>%
  separate(`test_reaction`, into = c("test", "reaction"))
mutate(gender = str_extract(gender, '^.'),
  language_native =
    case_when(str_detect(language_4, "^[gG]") ~ "de",
      str_detect(language_4, "^[fF]") ~ "fr")
  rowid_to_column() %>%
  separate_rows(stai, sep = ",") %>%
  select(rowid, starts_with("stai"), everything()) %>%
  separate(stai, into = c("stai_key", "stai_val"), sep = ",")
mutate(stai_val = as.integer(stai_val),
  stai_key = str_c("stai_", trimws(stai_key))) %>%
  pivot_wider( names_from = stai_key, values_from = stai_val)
select(rowid, id, starts_with("stai"), everything())
```

```
# A tibble: 12 × 21
  rowid   id stai_1 stai_2 stai_3 stai_6 stai_5 stai_4 gender langua
  <int> <dbl> <int> <int> <int> <int> <int> <int> <chr> <chr>
1     1     1     5     4     3     7    NA    NA m    German
2     2     2     5     4     3    NA     7    NA m    German
3     3     2     5     4     2    NA    NA     1 m    German
4     4     2     5     4     3    NA    NA     3 m    German
5     5     4     5     4     4     7    NA    NA f    German
6     6     5     3     3     3    NA    NA     2 m    French
7     7     5    NA    NA    NA    NA    NA    NA <NA> <NA>
8     8     7     3     3     3    NA    NA     4 f    Franzö
9     9     8     5     4     3     7    NA    NA f    Franzö
10    10     9     5     4     3    NA    NA     0 m    Deutsc
11    11    10     5     4     3    NA     1     1 m    Deutsc
12    12    11     2     4     3    NA     3     4 w    Deutsc
# ... with 10 more variables: reaction <chr>, test-time <dbl>, rec-lang
#   rec_1 <dbl>, rec_2 <dbl>, rec_3 <dbl>, language_16 <chr>, comment
#   language_native <chr>, NA <int>
```

Learning objectives

Common *tidy* data violations

- Inconsistent data from manual entry (`stringr`, `tidyr`)
- Multiple variables stored in one column (e.g. `tidyr::separate`)
- Multiple types in one table (`dplyr`, data transformation)
- One type in multiple tables (`dplyr`, combine into single table)
- Column headers are values, not variable names (`tidyr::pivot`)
- Variables are stored in both rows and columns (`tidyr::pivot`)
- Repeated observations (`nest` or separate tables)

Tidy data solutions

- Nicer column names with `janitor`
- Basic column-wise manipulation
 - now with `tidyr` and `stringr`
- Reshaping operations
 - after `dplyr`
- Nesting and advanced collapsing operations
 - after `broom`

Side note - janitor

```
readxl::read_excel("data/bad-table-psych.xlsx",  
  skip = 2)
```

```
# A tibble: 23 × 18  
  Name          ID Gender Language...4 Birthda  
  <chr>        <dbl> <chr>  <chr>      <chr>  
1 Roland Krause      1 male   German    38687  
2 Lars Juhl Krause    2 m      German   26/2/19  
3 <NA>                2 m      GERman   26/2/19  
4 <NA>                2 male   German   26/2/19  
5 Martina Krause      4 f      German   12-03-2  
6 Horst-Sergio Krause  5 m      French   12/11/7  
7 Horst-Sergio Krause  5 <NA>    <NA>      11-12-7  
8 LJK                 7 f      Französisch <NA>  
9 DFK                 8 f      Französisch 2005-12  
10 Horst Mond          9 male   Deutsch   1972-10  
# ... with 13 more rows, and 11 more variables: morale (by  
#   ...9 <chr>, STAI <chr>, Language...11 <chr>, ...12 <  
#   Rec-2 <dbl>, REC-3 <dbl>, Language...16 <chr>, ...17
```

Janitor

```
readxl::read_excel("data/bad-table-psych.xlsx", skip = 2  
  janitor::clean_names())
```

```
# A tibble: 12 × 18  
  name          id gender language_4 birthday x6      test_r  
  <chr>        <dbl> <chr>  <chr>      <chr>    <lgl> <chr>  
1 Roland...      1 male   German    38687    NA    dog=3  
2 Lars J...      2 m      German   26/2/19... NA    cat=3  
3 <NA>            2 m      GERman   26/2/19... NA    cat=3  
4 <NA>            2 male   German   26/2/19... NA    cat=3  
5 Martin...      4 f      German   12-03-2... NA    dog=2  
6 Horst-...      5 m      French   12/11/70 NA    trolle  
7 Horst-...      5 <NA>    <NA>      11-12-70 NA    <NA>  
8 LJK             7 f      Französis... <NA>    NA    trolle  
9 DFK             8 f      Französis... 2005-12... NA    trolle  
10 Horst ...      9 male   Deutsch   1972-10... NA    dog=1  
11 Kevin ...     10 männl... Deutsch    5.4.1970 NA    dog=3  
12 Inga M...     11 weibl... Deutsch    1/1/1990 NA    dog=5  
# ... with 10 more variables: x9 <dbl>, stai <chr>, langua  
#   rec_1 <dbl>, rec_2 <dbl>, rec_3 <dbl>, language_16 <  
#   comment <chr>
```

The data still sucks but at least you have nice column names.

Basic rearrangements

Splitting values - `tidyr::separate()`

Key-value pairs

```
# A tibble: 6 × 2
  subject_id gender_age
    <int> <chr>
1      1001 m-34
2      1002 f-24
3      1003 m-53
4      1004 f-44
5      1005 m-24
6      1006 f-30
```

```
patient %>%
  separate(gender_age,
           c("sex", "age"),
           convert = TRUE)
```

```
# A tibble: 6 × 3
  subject_id sex    age
    <int> <chr> <int>
1      1001 m      34
2      1002 f      24
3      1003 m      53
4      1004 f      44
5      1005 m      24
6      1006 f      30
```

No separator

```
# A tibble: 6 × 2
  subject_id gender_age
    <int> <glue>
1      1001 f50
2      1002 m34
3      1003 m63
4      1004 m57
5      1005 f42
6      1006 m58
```

```
patient %>%
  separate(gender_age, sep = 1,
           c("sex", "age"),
           convert = TRUE)
```

```
# A tibble: 6 × 3
  subject_id sex    age
    <int> <chr> <int>
1      1001 f      50
2      1002 m      34
3      1003 m      63
4      1004 m      57
5      1005 f      42
6      1006 m      58
```

Splitting and collating - separate() and unite()

Input tibble

```
data_value <-  
  tibble(  
    year = c(2015, 2014, 2014),  
    month = c(11, 2, 4),  
    day = c(23, 1, 30),  
    value = c("high", "low", "low"))
```

data_value

```
# A tibble: 3 × 4  
  year month   day value  
<dbl> <dbl> <dbl> <chr>  
1  2015     11    23 high  
2  2014      2     1 low  
3  2014      4    30 low
```



Tip: Demonstration only. Use the package `lubridate` for actually working with dates.

unite()

```
date_unite <- unite(data_value,  
                    date, year, month, day,  
                    sep = "-")
```

date_unite

```
# A tibble: 3 × 2  
  date      value  
<chr>    <chr>  
1 2015-11-23 high  
2 2014-2-1   low  
3 2014-4-30  low
```

No need to clean up old columns.

Splitting columns continued

Split with `separate()`

- Default split on non-alphanumeric characters
- Use **quotes** to designate columns since we are not referring to objects.

```
date_unite %>%  
  separate(date, c("year", "month", "day"))
```

```
# A tibble: 3 × 4  
  year month day value  
  <chr> <chr> <chr> <chr>  
1 2015  11   23  high  
2 2014   2    1   low  
3 2014   4   30   low
```

Separate rows with multiple entries

Multiple values per cell

```
patient_df <- tibble(  
  subject_id = 1001:1003,  
  visit_id = c("1,2, 3", "1|2", "1"),  
  measured = c("9,0, 11", "11, 3", "12"))  
patient_df
```

```
# A tibble: 3 × 3  
  subject_id visit_id measured  
    <int> <chr>    <chr>  
1     1001 1,2, 3    9,0, 11  
2     1002 1|2      11, 3  
3     1003 1        12
```

Note the incoherent white space and separators.

Combinations of variables

```
patient_df %>%  
  separate_rows(visit_id, measured,  
                convert = TRUE) -> patient_separate  
patient_separate
```

```
# A tibble: 6 × 3  
  subject_id visit_id measured  
    <int>    <int>    <int>  
1     1001         1         9  
2     1001         2         0  
3     1001         3        11  
4     1002         1        11  
5     1002         2         3  
6     1003         1        12
```

Fill all combinations with complete()

Combinations of variables

```
patient_separate %>%  
  complete(subject_id,  
            nesting(visit_id))
```

```
# A tibble: 9 × 3  
  subject_id visit_id measured  
    <int>    <int>    <int>  
1     1001         1         9  
2     1001         2         0  
3     1001         3        11  
4     1002         1        11  
5     1002         2         3  
6     1002         3        NA  
7     1003         1        12  
8     1003         2        NA  
9     1003         3        NA
```



Tip: Use `<NA>`, not `0` for missing data in real life application.

Determine filling element as list

```
patient_separate %>%  
  complete(subject_id,  
            nesting(visit_id), fill = list(measured = 0))
```

```
# A tibble: 9 × 3  
  subject_id visit_id measured  
    <int>    <int>    <dbl>  
1     1001         1         9  
2     1001         2         0  
3     1001         3        11  
4     1002         1        11  
5     1002         2         3  
6     1002         3         0  
7     1003         1        12  
8     1003         2         0  
9     1003         3         0
```

Data cleaning

"Manual" Curation

```
my_col <- c("F", "M", "female",  
           "male", "male",  
           "female", "female")  
  
write.csv(as.data.frame("a" <- my_col),  
          "~/Downloads/important.csv")
```

Open Excel



```
my_clean_data <- read.csv("~/Downloads/important.csv")
```

Scripted Curation

```
my_col <- c("F", "M", "female",  
           "male", "male", "female",  
           "female", "männlich")  
  
convert_gender <- function(x) {  
  case_when(  
    str_detect(x, "^[Ff]") ~ "Female",  
    str_detect(x, "^[Mm]") ~ "Male",  
    TRUE ~ x  
  )  
}  
  
convert_gender(my_col)
```

```
[1] "Female" "Male"   "Female" "Male"   "Male"   "Female" "Fema
```

Before we stop

Tidy data core

- <http://tidyr.tidyverse.org/>
- `vignette("tidy-data")`

Further reading ☐

- **Rectangling** - Taming JSON and other data into rectangles
 - <https://tidyr.tidyverse.org/dev/articles/rectangle.html>

Acknowledgments ☐ ☐

- Hadley Wickham (Tidy concepts)
- Alison Hill (**tidyr**)
- Artwork by [Allison Horst](#)
- Initial development: Roland Krause

Thank you for your attention!

