

ISA 401: Business Intelligence & Data Visualization

09: Tidy Data in

Fadel M. Megahed, PhD

Raymond E. Glos Professor in Business
Farmer School of Business
Miami University

 @FadelMegahed

 fmegahed

 fmegahed@miamioh.edu

 Automated Scheduler for Office Hours

Fall 2025

Quick Refresher from Last Class

- Describe what is an API
- Download data using APIs

Learning Objectives for Today's Class

- Define tidy data
- Perform pivot and rectangling operations in 

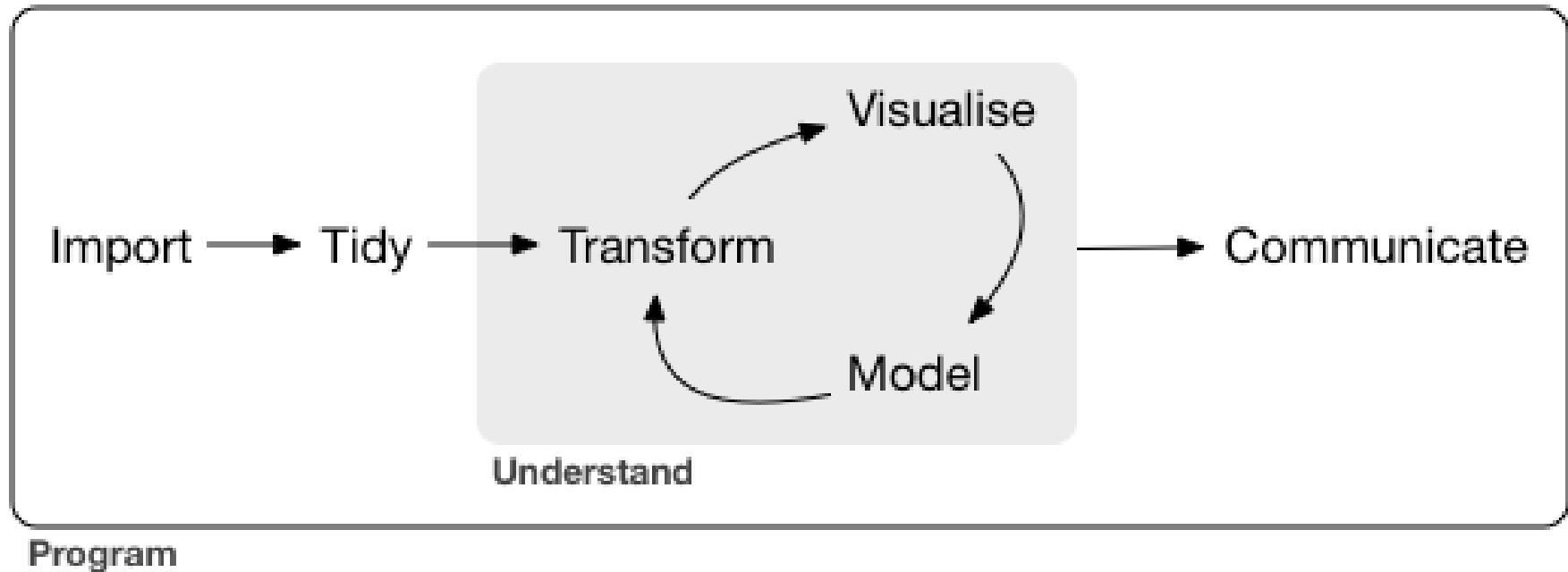
Tidy Data





Sources: All three images are obtained from Upsplash

The R for Data Science Workflow



Source: Image is from Wickham, H. Grolemund, G. (2017). "R for Data Science", O'Reilly.
<https://r4ds.had.co.nz/introduction.html>

The Rationale for Tidy Data

- The **tidy framework** provides a **consistent way to organize your data** in .
- Getting your data into this format requires some **upfront work, but that work pays off in the long term.**
- Once you have tidy data and the tidy tools provided by packages in the **tidyverse**, you will spend **much less time munging data from one representation to another, allowing you to spend more time on the analytic questions at hand.**

“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

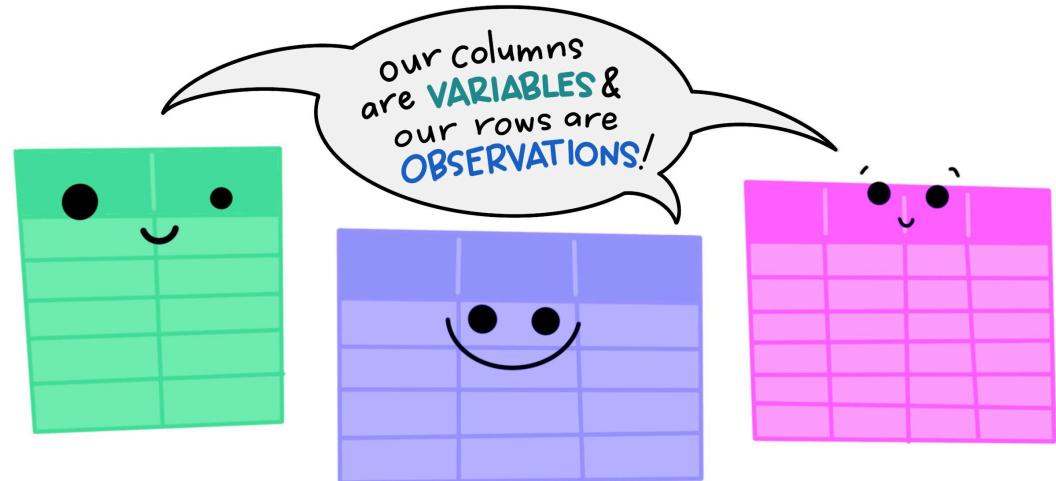
each column a variable

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

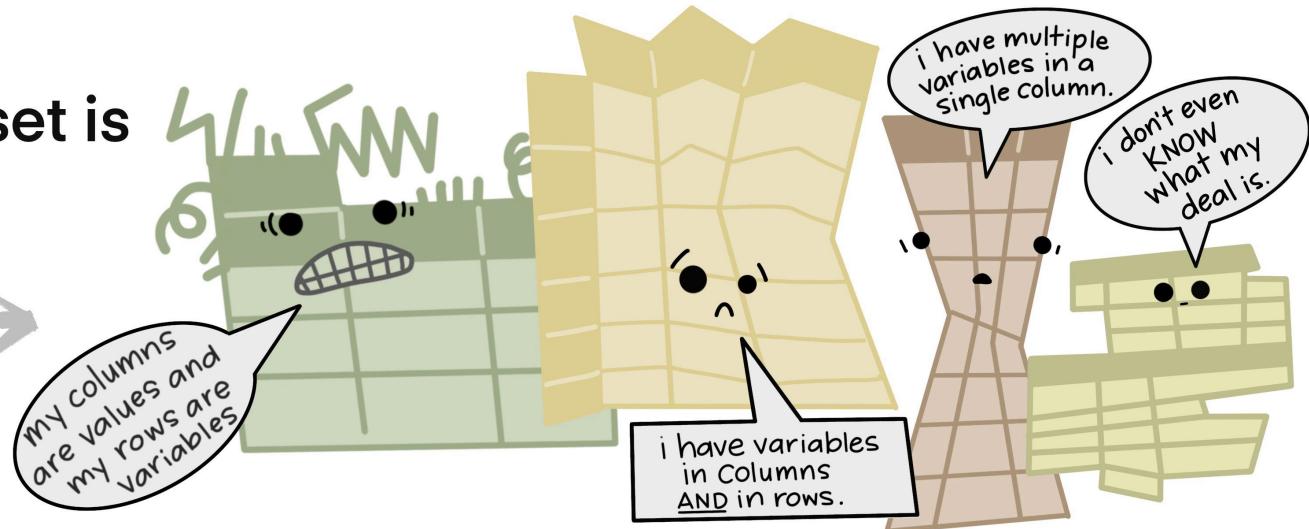
Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

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 data \neq clean data

The `movies` data is tidy but not clean.

```
movies <- tibble::as_tibble(jsonlite::read_json(  
  "https://vega.github.io/vega-editor/app/data/movies.json",  
  simplifyVector = TRUE))  
  
movies |>  
  dplyr::relocate(Release_Date, US_DVD_Sales) |> # move cols to front  
  dplyr::slice(37:39, 268:269) |> # filter specific row numbers  
  print(width = 80) # print nicely
```

```
## # A tibble: 5 × 16  
##   Release_Date US_DVD_Sales Title      US_Gross Worldwide_Gross Production_Budget  
##   <chr>          <int> <chr>      <int>           <dbl>                  <int>  
## 1 9-Mar-94            NA Four Wed...  52700832        242895809        4500000  
## 2 18-Oct-06            NA 51 Birch...   84689         84689        350000  
## 3 1963-01-01            NA 55 Days ... 10000000 100000000       17000000  
## 4 <NA>                 NA Drei          0             0          7200000  
## 5 16-Jan-98            NA The Dress    16556        16556        2650000  
## # i 10 more variables: MPAA_Rating <chr>, Running_Time_min <int>,  
## #   Distributor <chr>, Source <chr>, Major_Genre <chr>, Creative_Type <chr>,  
## #   Director <chr>, Rotten_Tomatoes_Rating <int>, IMDB_Rating <dbl>,  
## #   IMDB_Votes <int>
```

05:00

Non-graded Activity: Tidy or Not?

Activity

table1

table2

table3

table4a

table4b

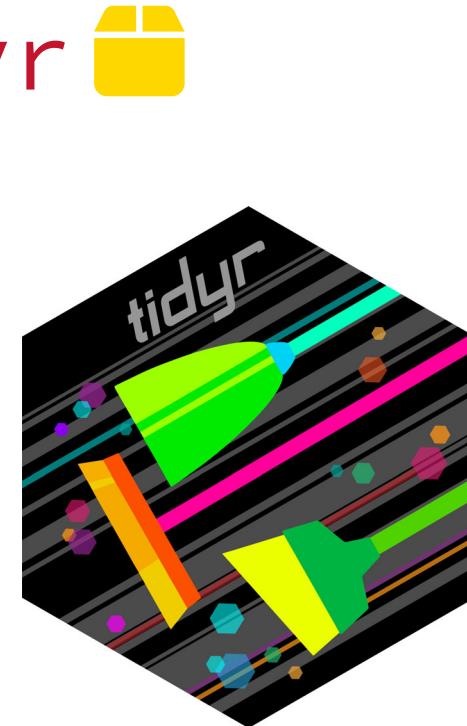
- In the next five panels, there are five tables all displaying the number of TB cases documented by the World Health Organization in Afghanistan, Brazil, and China between 1999 and 2000.
- The data contains values associated with four variables (country, year, cases, and population), but each table organizes the values in a different layout.
- **Based on the information in the previous slide, please document which of the table(s) is(are) tidy and if not, which rules are violated.**
- **Discuss your answer with your neighboring colleague.**

Note that you have a total of five minutes for this non-graded activity.

Getting Data into Tidy Format

Key Functions from the `tidyverse` 📁

type	function()	function()
pivoting	<code>pivot_longer()</code>	<code>pivot_wider()</code>
splitting/combing	<code>separate()</code>	<code>unite()</code>
nesting/unnesting	<code>nest()</code>	<code>unnest()</code>
missing	<code>complete()</code>	<code>fill()</code>



Wide Vs Long Data

wide				long		
id	x	y	z	id	key	val
1	a	c	e	1	x	a
2	b	d	f	2	x	b
				1	y	c
				2	y	d
				1	z	e
				2	z	f

pivot_() to Transform Wide from/to Long

wide

id	x	y	z
1	a	c	e
2	b	d	f

The pivot_longer() Function

```
pivot_longer(wide,  
            cols = x:z,  
            names_to = "key",  
            values_to = "val")
```

returns

	id	key	val
1	1	x	a
	1	y	c
	1	z	e
2	2	x	b
	2	y	d
	2	z	f

pivot_longer() for table4a [1]

To tidy a dataset like this, we need to pivot the **offending columns into a new pair of variables**. To describe that operation we need **three parameters**:

- The set of columns whose names are values, not variables. In this example, those are the columns **1999** and **2000**.
- The name of the variable to move the column names to. Here it is **year**.
- The name of the variable to move the column values to. Here it's **cases**.

pivot_longer() for table4a [2]

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

table4

The diagram illustrates the transformation of the original data table (table4) into a longer, tidier form. The original table has three columns: country, year, and cases. The pivoted table has three columns: country, 1999, and 2000. Arrows point from each row in the original table to its corresponding column in the pivoted table. For example, the first row in the original table (Afghanistan, 1999, 745) points to the 1999 column in the pivoted table. The second row (Afghanistan, 2000, 2666) points to the 2000 column. The third row (Brazil, 1999, 37737) points to the 1999 column. The fourth row (Brazil, 2000, 80488) points to the 2000 column. The fifth row (China, 1999, 212258) points to the 1999 column. The sixth row (China, 2000, 213766) points to the 2000 column.

Pivoting table4a into a longer, tidy form

pivot_longer() for table4a [3]

```
tidyr::pivot_longer(table4a, c(`1999`, `2000`), names_to = "year", values_to = "cases")
```

```
## # A tibble: 6 × 3
##   country     year   cases
##   <chr>       <chr>  <dbl>
## 1 Afghanistan 1999    745
## 2 Afghanistan 2000   2666
## 3 Brazil      1999  37737
## 4 Brazil      2000  80488
## 5 China       1999 212258
## 6 China       2000 213766
```

The pivot_wider() Function

```
pivot_wider(long,  
           names_from = key,  
           values_from = val)
```

returns

id	x	y	z
1	a	c	e
2	b	d	f

pivot_wider() for table2 [1]

- `pivot_wider()` is the opposite of `pivot_longer()`.
- You use it when an observation is scattered across multiple rows.
- For example, take `table2`: an observation is a country in a year, but each observation is spread across two rows.

```
head(table2, n = 3)
```

```
## # A tibble: 3 × 4
##   country     year type      count
##   <chr>       <dbl> <chr>     <dbl>
## 1 Afghanistan 1999 cases     745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases     2666
```

pivot_wider() for table2 [2]

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

pivot_wider() for table2 [3]

```
tidyr::pivot_wider(table2, names_from = type, values_from = count)
```

```
## # A tibble: 6 × 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
```

separate() for table3 [1]

```
## # A tibble: 6 × 3
##   country     year    rate
##   <chr>       <dbl>  <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
```

table3 has a different problem:

- we have one column (**rate**) that contains two variables (**cases** and **population**).
- To fix this problem, we'll need the **separate()** function.

separate() for table3 [2]

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

table3

separate() for table3 [3]

```
tidyr::separate(table3, rate, into = c("cases", "population"), convert = TRUE)
```

```
## # A tibble: 6 × 4
##   country     year  cases population
##   <chr>       <dbl>  <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
```

05:00

Non-graded Class Activity

Activity

Your Solution

In this five minute non-graded activity, please do the following

- Go to <https://usafacts.org/visualizations/coronavirus-covid-19-spread-map/>
- Download the data for **Deaths** by clicking on the tab to the right of the page.
- **Tidy this data based on the information you have learned in today's class.**

Recap

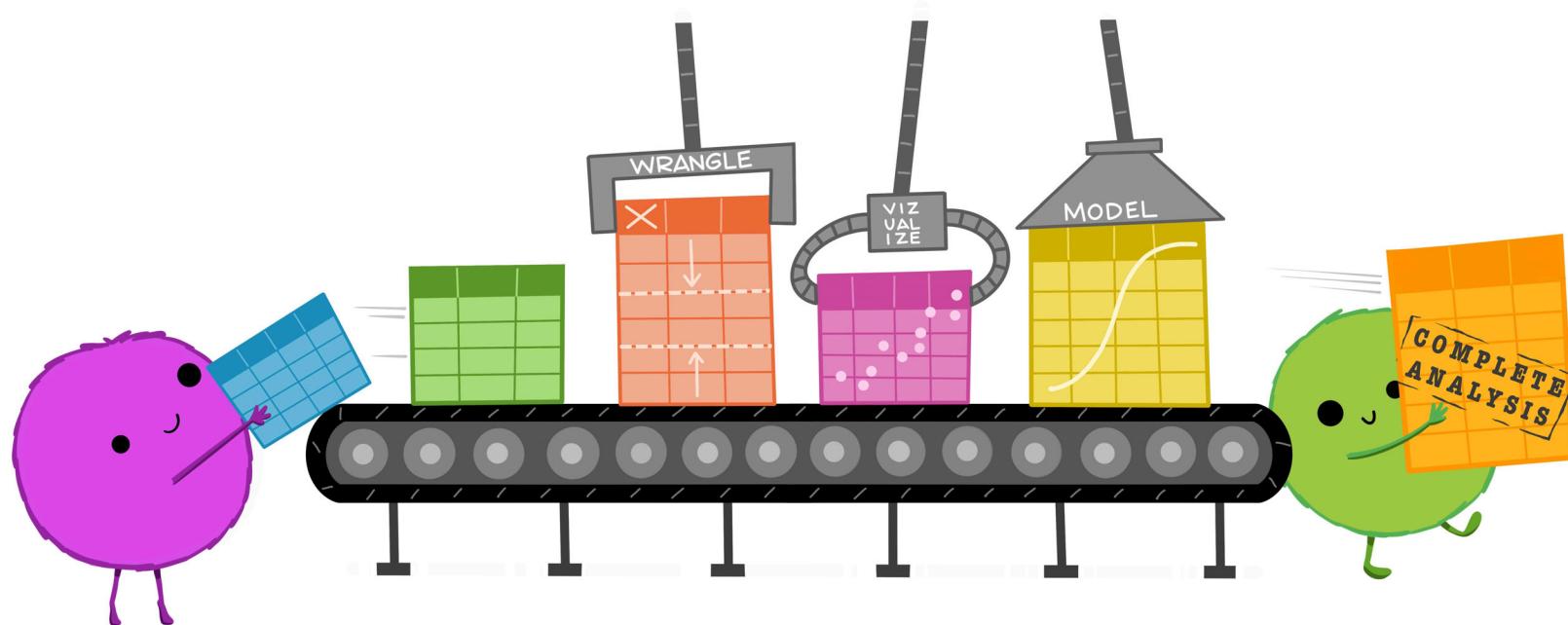
Summary of Main Points

By now, you should be able to do the following:

- Define tidy data
- Perform pivot and rectangling operations in 

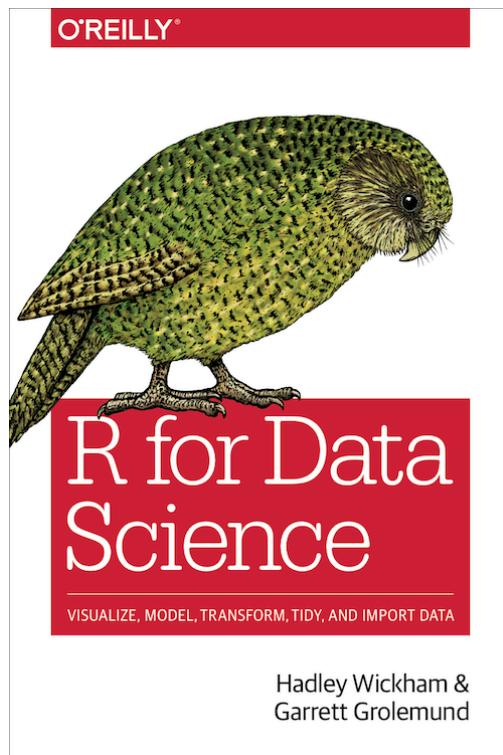
Advantages of Tidy Data

- one set of consistent tools for different datasets
- easier for automation and iteration



Things to Do Prior to Next Class

Please go through the following two supplementary readings and complete [assignment 06: tidy data](#).



- [Tidy data](#)
- [{tidyverse} cheatsheet](#)