# Tidy Data

## 5 Most Common Problems With Messy Datasets

Pengfei Qin

12 April 2021

# Why we need data tidying?

## Data

- We use data for data analysis (regression or prediction), or do some larger projects (analyze pandemic trends).

## Problems

- Data may have many disadvantages (missing values, outliers, typo).

| color | director_name | duration | gross | movie_title | anguage | country | budget | title_year | imdb_score |
|---|---|---|---|---|---|---|---|---|---|
| Color | Martin Scorsese | 240 | 116866727 | The Wolf of Wall StreetÂ | English | USA | 100000000 | 2013 | 8.2 |
| Color | Shane Black | 195 | 408992272 | Iron Man 3Â | English | USA | 200000000 | 2013 | 7.2 |
| color | Quentin Tarantino | 187 | 54116191 | The Hateful EightÂ | English | USA | 44000000 | 2015 | 7.9 |
| Color | Kenneth Lonergan | 186 | 46495 | MargaretÂ | English | usa | 14000000 | 2011 | 6.5 |
| Color | Peter Jackson | 186 | 258355354 | The Hobbit: The Desolation of SmaugÂ | English | USA | 225000000 | 2013 | 7.9 |
| | N/A | 183 | 330249062 | Batman v Superman: Dawn of JusticeÂ | English | USA | 250000000 | 202 | 6.9 |
| Color | Peter Jackson | -50 | 303001229 | The Hobbit: An Unexpected JourneyÂ | English | USA | 180000000 | 2012 | 7.9 |
| Color | Edward Hall | 180 | | RestlessÂ | English | UK | | 2012 | 7.2 |
| Color | Joss Whedon | 173 | 623279547 | The AvengersÂ | English | USA | 220000000 | 2012 | 8.1 |
| Color | Joss Whedon | 173 | 623279547 | The AvengersÂ | English | USA | 220000000 | 2012 | 8.1 |
| | Tom Tykwer | 172 | 27098580 | Cloud AtlasA | English | Germany | 102000000 | 2012 | -7.5 |
| Color | Null | 158 | 102515793 | The Girl with the Dragon TattooÂ | English | USA | 90000000 | 2011 | 7.8 |
| Color | Christopher Spencer | 170 | 59696176 | Son of GodÂ | English | USA | 22000000 | 2014 | 5.6 |
| Color | Peter Jackson | 164 | 255108370 | The Hobbit: The Battle of the Five ArmiesÂ | English | New Zealand | 250000000 | 2014 | 7.5 |
| Color | Tom Hooper | 158 | 148775460 | Les MisÃ©rablesÂ | English | USA | 61000000 | 2012 | 7.6 |
| Color | Tom Hooper | 158 | 148775460 | Les MisÃ©rablesÂ | English | USA | 61000000 | 2012 | 7.6 |

- The structure of the data set is not uniform (various arrangements in variables and observations).

# Why we need data tidying?

## Solution

- Data Tidying: Structuring datasets to facilitate analysis.
- Tidy data provides us a standard method to organize the values in a dataset.

## Purpose

- Tidying the initial data can make subsequent data analysis easier.
- Consistent data structure will also make our data analysis easier.
- Make R happier.

# The principle of tidy data

## Tidy data

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

```
dataset4 = data.frame(Song = rep(c("Song_A", "Song_B", "Song_C")
  'Year' = rep(c("2019", "2020", "2021"), each = 3),
  'Time' = rep(c("241", "175", "239"), each = 3),
  'Date'= c("2021-04-01", "2021-04-02", "2021-04-03", "2021-04-0
  'Rank' = c(1,3,2,2,1,3,3,2,1))
dataset4
```

```
##     Song Year Time       Date Rank
## 1 Song_A 2019  241 2021-04-01    1
## 2 Song_A 2019  241 2021-04-02    3
## 3 Song_A 2019  241 2021-04-03    2
## 4 Song_B 2020  175 2021-04-01    2
## 5 Song_B 2020  175 2021-04-02    1
## 6 Song_B 2020  175 2021-04-03    3
## 7 Song_C 2021  239 2021-04-01    3
## 8 Song_C 2021  239 2021-04-02    2
## 9 Song_C 2021  239 2021-04-03    1
```

"5 Most Common Problems With Messy Datasets"

# 1. Column headers are values, not variable names.

```
dataset1 = data.frame(name = c("Company_A", "Company_B", "Compar
  'bachelor'= sample(1:10, 4),
  'master'= sample(1:10, 4),
  'PhD'= sample(1:10, 4))
dataset1
```

```
##        name bachelor master PhD
## 1 Company_A        4      4   5
## 2 Company_B        7      9   4
## 3 Company_C        5      5   1
## 4 Company_D       10      7   9
```

# 1. Column headers are values, not variable names.

```
dataset1 <- dataset1 %>%
  pivot_longer(-name, names_to = "degree", values_to = "frequenc
dataset1
```

```
## # A tibble: 12 x 3
##    name      degree    frequency
##    <chr>     <chr>         <int>
##  1 Company_A bachelor          4
##  2 Company_A master            4
##  3 Company_A PhD               5
##  4 Company_B bachelor          7
##  5 Company_B master            9
##  6 Company_B PhD               4
##  7 Company_C bachelor          5
##  8 Company_C master            5
##  9 Company_C PhD               1
## 10 Company_D bachelor         10
## 11 Company_D master            7
## 12 Company_D PhD               9
```

This form is tidy: there's one variable in each column, and each row represents one observation.

# 2. Multiple variables stored in one column

```
dataset2 = data.frame(name = c("Company_A", "Company_B", "Compar
  'm2035'= sample(1:10, 4),
  'm3550'= sample(1:10, 4),
  'm5065'= sample(1:10, 4),
  'f2035'= sample(1:10, 4),
  'f3550'= sample(1:10, 4),
  'f5060'= sample(1:10, 4))
dataset2
```

```
##        name m2035 m3550 m5065 f2035 f3550 f5060
## 1 Company_A     8    10     1     8     8     8
## 2 Company_B    10     4     8     7     6     2
## 3 Company_C     3     9     5     3     9     5
## 4 Company_D     6     2     3     2     1     7
```

# 2. Multiple variables stored in one column

```
dataset2 <- dataset2 %>%
  pivot_longer(-name, names_to = "combination", values_to = "fre
dataset2
```

```
## # A tibble: 24 x 3
##    name      combination frequency
##    <chr>     <chr>           <int>
##  1 Company_A m2035               8
##  2 Company_A m3550              10
##  3 Company_A m5065               1
##  4 Company_A f2035               8
##  5 Company_A f3550               8
##  6 Company_A f5060               8
##  7 Company_B m2035              10
##  8 Company_B m3550               4
##  9 Company_B m5065               8
## 10 Company_B f2035               7
## # … with 14 more rows
```

# 2. Multiple variables stored in one column

```
dataset2 <- dataset2 %>%
  separate(combination, c("sex", "age"),1)
dataset2
```

```
## # A tibble: 24 x 4
##    name      sex   age   frequency
##    <chr>     <chr> <chr>     <int>
##  1 Company_A m     2035          8
##  2 Company_A m     3550         10
##  3 Company_A m     5065          1
##  4 Company_A f     2035          8
##  5 Company_A f     3550          8
##  6 Company_A f     5060          8
##  7 Company_B m     2035         10
##  8 Company_B m     3550          4
##  9 Company_B m     5065          8
## 10 Company_B f     2035          7
## # … with 14 more rows
```

This form is tidy: there's one variable in each column, and each row represents one observation.

# 3. Variables are stored in both columns and rows.

```
dataset3 = data.frame(city = rep(c("Beijing", "Hong Kong", "Los
  'month'= c("January"),
  'element'= c("avg_environmental_quality", "avg_air_quality"),
  'y2019'= sample(c("high", "median", "low"),8, replace = TRUE),
  'y2020'= sample(c("high", "median", "low"),8, replace = TRUE),
  'y2021'= sample(c("high", "median", "low"),8, replace = TRUE))
dataset3
```

```
##          city   month                   element  y2019  y2020  y20
## 1     Beijing January avg_environmental_quality   high    low   hi
## 2     Beijing January           avg_air_quality   high   high   hi
## 3   Hong Kong January avg_environmental_quality   high median    l
## 4   Hong Kong January           avg_air_quality median    low   hi
## 5 Los Angeles January avg_environmental_quality median   high   hi
## 6 Los Angeles January           avg_air_quality   high median medi
## 7    New York January avg_environmental_quality   high    low   hi
## 8    New York January           avg_air_quality   high    low    l
```

11 / 17

# 3. Variables are stored in both columns and rows.

```
dataset3 <- dataset3 %>%
  pivot_longer(y2019:y2021, names_to = "year", values_to = "valu

dataset3 <- dataset3 %>%
  mutate(year = as.integer(gsub("y", "", year))) %>%
  select(city, month, element, year, value)

dataset3
```

```
## # A tibble: 24 x 5
##    city      month   element                      year value
##    <chr>     <chr>   <chr>                       <int> <chr>
##  1 Beijing   January avg_environmental_quality    2019 high
##  2 Beijing   January avg_environmental_quality    2020 low
##  3 Beijing   January avg_environmental_quality    2021 high
##  4 Beijing   January avg_air_quality              2019 high
##  5 Beijing   January avg_air_quality              2020 high
##  6 Beijing   January avg_air_quality              2021 high
##  7 Hong Kong January avg_environmental_quality    2019 high
##  8 Hong Kong January avg_environmental_quality    2020 median
##  9 Hong Kong January avg_environmental_quality    2021 low
## 10 Hong Kong January avg_air_quality              2019 median
## # … with 14 more rows
```

# 3. Variables are stored in both columns and rows.

```
dataset3 <- dataset3 %>%
  pivot_wider(names_from = element, values_from = value)
dataset3
```

```
## # A tibble: 12 x 5
##    city        month    year avg_environmental_quality avg_air_qua
##    <chr>       <chr>   <int> <chr>                     <chr>
##  1 Beijing     January  2019 high                      high
##  2 Beijing     January  2020 low                       high
##  3 Beijing     January  2021 high                      high
##  4 Hong Kong   January  2019 high                      median
##  5 Hong Kong   January  2020 median                    low
##  6 Hong Kong   January  2021 low                       high
##  7 Los Angeles January  2019 median                    high
##  8 Los Angeles January  2020 high                      median
##  9 Los Angeles January  2021 high                      median
## 10 New York    January  2019 high                      high
## 11 New York    January  2020 low                       low
## 12 New York    January  2021 high                      low
```

This form is tidy: there's one variable in each column, and each row represents one observation.

13 / 17

# 4. Multiple types of observational units are stored in the same table

```
##      Song Year Time       Date Rank
## 1 Song_A 2019  241 2021-04-01    1
## 2 Song_A 2019  241 2021-04-02    3
## 3 Song_A 2019  241 2021-04-03    2
## 4 Song_B 2020  175 2021-04-01    2
## 5 Song_B 2020  175 2021-04-02    1
## 6 Song_B 2020  175 2021-04-03    3
## 7 Song_C 2021  239 2021-04-01    3
## 8 Song_C 2021  239 2021-04-02    2
## 9 Song_C 2021  239 2021-04-03    1
```

```
song <- dataset4 %>%
  distinct(Song, Year, Time)
song
```

```
##      Song Year Time
## 1 Song_A 2019  241
## 2 Song_B 2020  175
## 3 Song_C 2021  239
```

# 4. Multiple types of observational units are stored in the same table

```
rank <- dataset4 %>%
  left_join(song, c("Song", "Year", "Time")) %>%
  select(Song, Date, Rank)
rank
```

```
##     Song       Date Rank
## 1 Song_A 2021-04-01    1
## 2 Song_A 2021-04-02    3
## 3 Song_A 2021-04-03    2
## 4 Song_B 2021-04-01    2
## 5 Song_B 2021-04-02    1
## 6 Song_B 2021-04-03    3
## 7 Song_C 2021-04-01    3
## 8 Song_C 2021-04-02    2
## 9 Song_C 2021-04-03    1
```

# 5. A single observational unit is stored in multiple tables

```
GDP_and_Tax = data.frame(City = rep(c("Beijing", "Hong Kong", "N
  'GDP'= runif(6, 100, 200),
  'Tax_Revenue' = runif(6, 15, 25))
GDP_and_Tax
```

```
##        City      GDP Tax_Revenue
## 1   Beijing 180.6807    22.73900
## 2   Beijing 181.1368    21.24779
## 3 Hong Kong 108.6159    21.25485
## 4 Hong Kong 156.1439    18.90633
## 5  New York 111.7318    19.66630
## 6  New York 112.4898    24.78881
```

```
Energy_and_Industry = data.frame(City = rep(c("Beijing", "Hong K
  'Energy_Consumption'= runif(6, 1900, 2000),
  'Industrial_Output' = runif(6, 250, 300))
Energy_and_Industry
```

```
##        City Energy_Consumption Industrial_Output
## 1   Beijing           1909.612          265.5763
## 2   Beijing           1913.279          280.9248
## 3 Hong Kong           1925.124          279.3627
## 4 Hong Kong           1932.035          263.0203
## 5  New York           1985.700          286.1806
## 6  New York           1942.070          285.3541
```

16 / 17

# 5. A single observational unit is stored in multiple tables

```
dataset5 <- inner_join(GDP_and_Tax, Energy_and_Industry)
```

```
## Joining, by = "City"
```

```
dataset5
```

```
##         City      GDP Tax_Revenue Energy_Consumption Industrial_Ou
## 1    Beijing 180.6807    22.73900           1909.612          265.
## 2    Beijing 180.6807    22.73900           1913.279          280.
## 3    Beijing 181.1368    21.24779           1909.612          265.
## 4    Beijing 181.1368    21.24779           1913.279          280.
## 5  Hong Kong 108.6159    21.25485           1925.124          279.
## 6  Hong Kong 108.6159    21.25485           1932.035          263.
## 7  Hong Kong 156.1439    18.90633           1925.124          279.
## 8  Hong Kong 156.1439    18.90633           1932.035          263.
## 9   New York 111.7318    19.66630           1985.700          286.
## 10  New York 111.7318    19.66630           1942.070          285.
## 11  New York 112.4898    24.78881           1985.700          286.
## 12  New York 112.4898    24.78881           1942.070          285.
```