What is tidy data?

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

Happy families are all alike, but every unhappy family is unhappy in its own way.

Leo Tolstoy

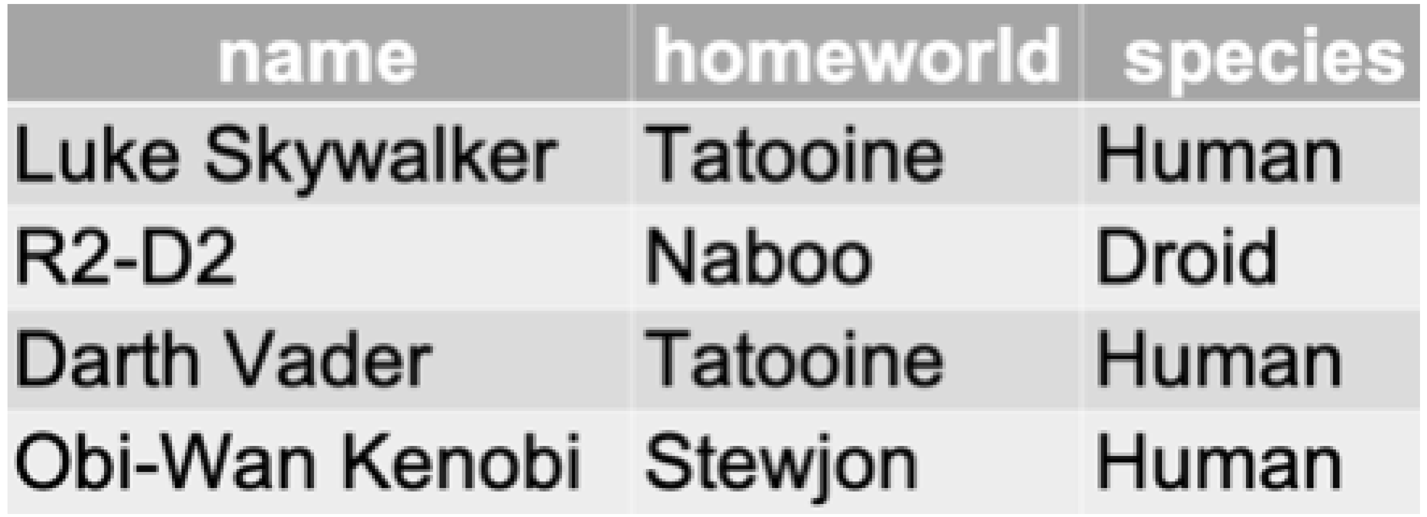
Tidy datasets are all alike, but every messy dataset is messy in its own way.

Hadley Wickham

Rectangular data

Structure

- Columns
- Rows
- Cells

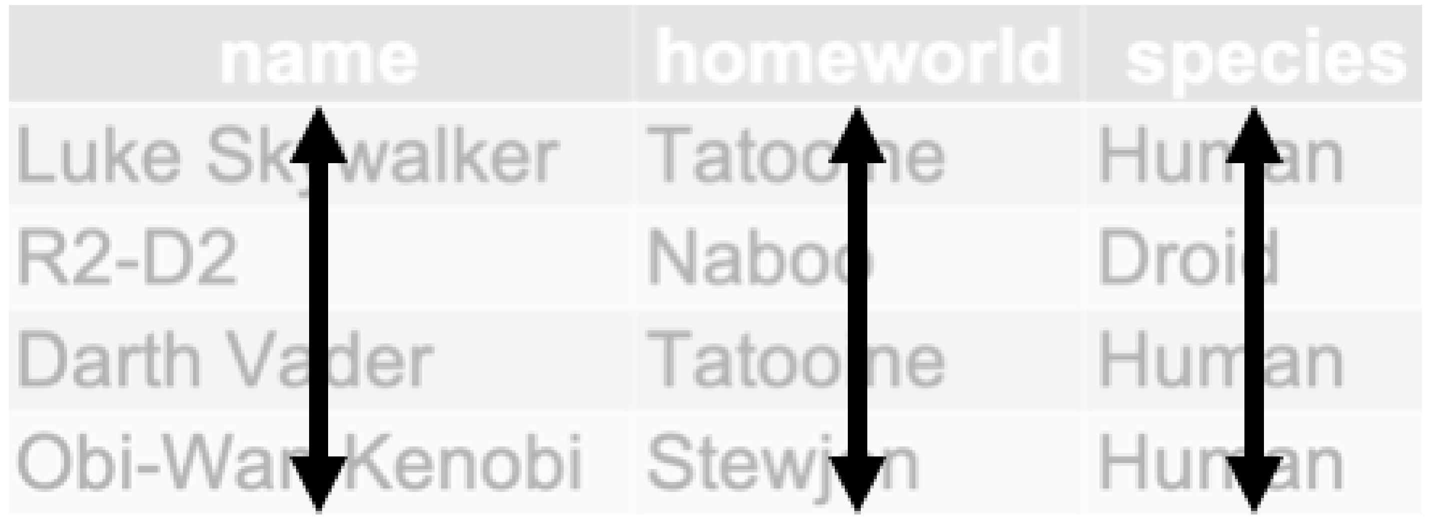


name	homeworld	species
Luke Skywalker	Tatooine	Human
R2-D2	Naboo	Droid
Darth Vader	Tatooine	Human
Obi-Wan Kenobi	Stewjon	Human

Tidy data, variables

Structure

- **Columns hold variables**
- Rows
- Cells

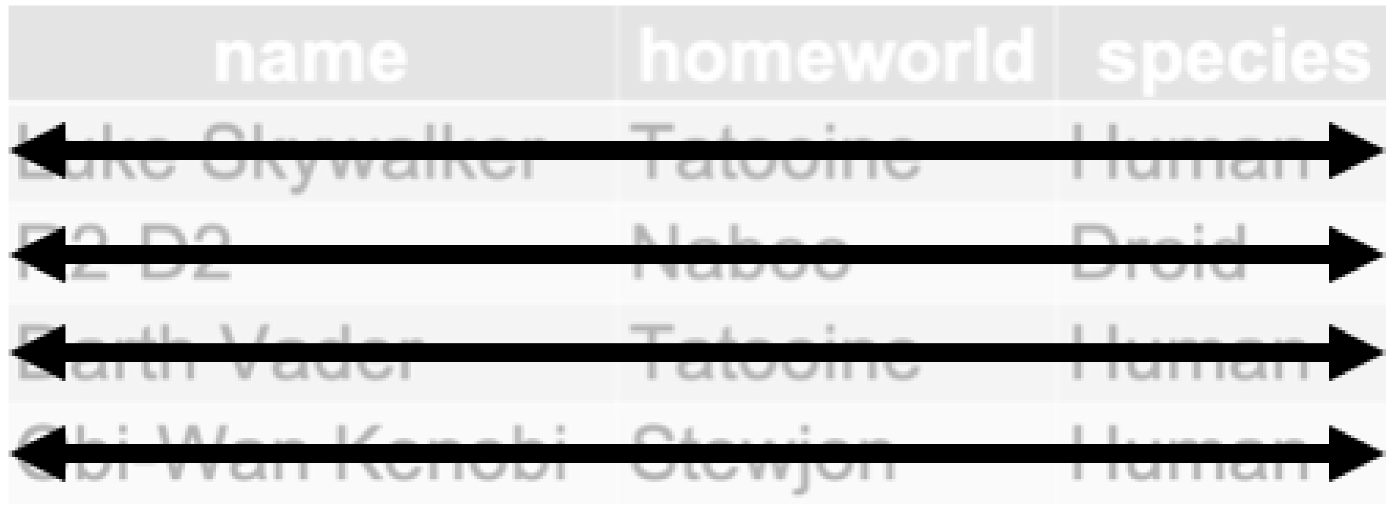


name	homeworld	species
Luke Skywalker	Tatooine	Human
R2-D2	Naboo	Droid
Darth Vader	Tatooine	Human
Obi-Wan Kenobi	Stewjon	Human

Tidy data, observations

Structure

- Columns hold variables
- **Rows hold observations**
- Cells

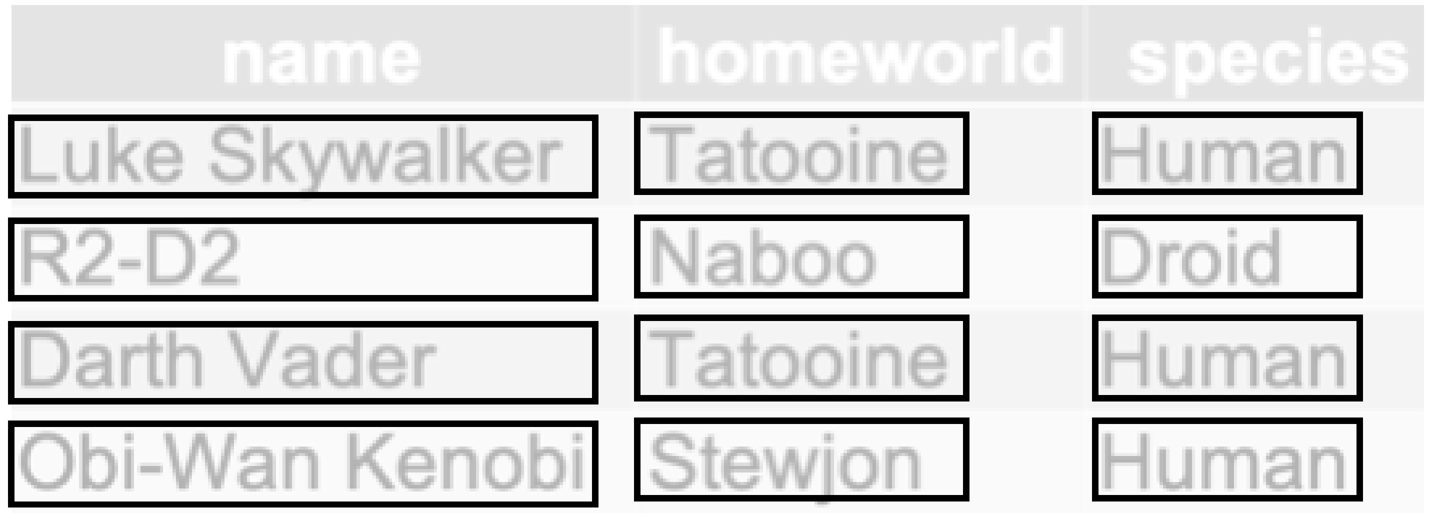


name	homeworld	species
Luke Skywalker	Tatooine	Human
R2-D2	Naboo	Droid
Darth Vader	Tatooine	Human
Obi-Wan Kenobi	Stewjon	Human

Tidy data, values

Structure

- Columns hold variables
- Rows hold observations
- **Cells hold values**



name	homeworld	species
Luke Skywalker	Tatooine	Human
R2-D2	Naboo	Droid
Darth Vader	Tatooine	Human
Obi-Wan Kenobi	Stewjon	Human

dplyr recap

```
character_df
```

```
# A tibble: 4 x 3
  name           homeworld species
  <chr>          <chr>     <chr>
1 Luke Skywalker Tatooine  Human
2 R2-D2          Naboo     Droid
3 Darth Vader    Tatooine  Human
4 Obi-Wan Kenobi Stewjon   Human
```

dplyr recap: select()

```
character_df %>%
  select(name, homeworld)
```

```
# A tibble: 4 x 2
  name           homeworld
  <chr>          <chr>
1 Luke Skywalker Tatooine
2 R2-D2          Naboo
3 Darth Vader    Tatooine
4 Obi-Wan Kenobi Stewjon
```

dplyr recap: filter()

```
character_df %>%
  filter(homeworld == "Tatooine")
```

```
# A tibble: 2 x 3
  name           homeworld species
  <chr>          <chr>     <chr>
1 Luke Skywalker Tatooine  Human
2 Darth Vader    Tatooine  Human
```

dplyr recap: mutate()

```r
character_df %>%
  mutate(is_human = species == "Human")
```

```
# A tibble: 4 x 4
  name           homeworld species is_human
  <chr>          <chr>     <chr>   <lgl>
1 Luke Skywalker Tatooine  Human   TRUE
2 R2-D2          Naboo     Droid   FALSE
3 Darth Vader    Tatooine  Human   TRUE
4 Obi-Wan Kenobi Stewjon   Human   TRUE
```

dplyr recap: group_by() and summarize()

```
character_df %>%
  group_by(homeworld) %>%
  summarize(n = n())
```

```
# A tibble: 3 x 2
  homeworld     n
  <chr>     <int>
1 Naboo         1
2 Stewjon       1
3 Tatooine      2
```

[1] magrittr.tidyverse.org

[1] www.tidyverse.org

Multiple variables in a single column

```
population_df
```

```
# A tibble: 4 x 2
  country                   population
  <chr>                          <dbl>
1 Brazil, South America        210.
2 Nepal, Asia                   28.1
3 Senegal, Africa               15.8
4 Australia, Oceania            25.0
```

Separating variables over two columns

```r
population_df %>%
  separate(country, into = c("country", "continent"), sep = ", ")
```

```
# A tibble: 4 x 3
  country   continent     population
  <chr>     <chr>              <dbl>
1 Brazil    South America      210.
2 Nepal     Asia                28.1
3 Senegal   Africa              15.8
4 Australia Oceania             25.0
```

Let's practice!

RESHAPING DATA WITH TIDYR

Columns with multiple values

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

Two variables in a single column

```r
netflix_df
```

```
# A tibble: 637 x 3
   title                  type    duration
   <chr>                  <chr>   <chr>
 1 Article 15             Movie   125 min
 2 Kill Me If You Dare    Movie   100 min
 3 The Spy                TV Show 1 Seasons
 4 The World We Make      Movie   108 min
 5 Watchman               Movie   93 min
```

Converting separated columns' data types

```r
netflix_df %>%
  separate(duration, into = c("value", "unit"), convert = TRUE)
```

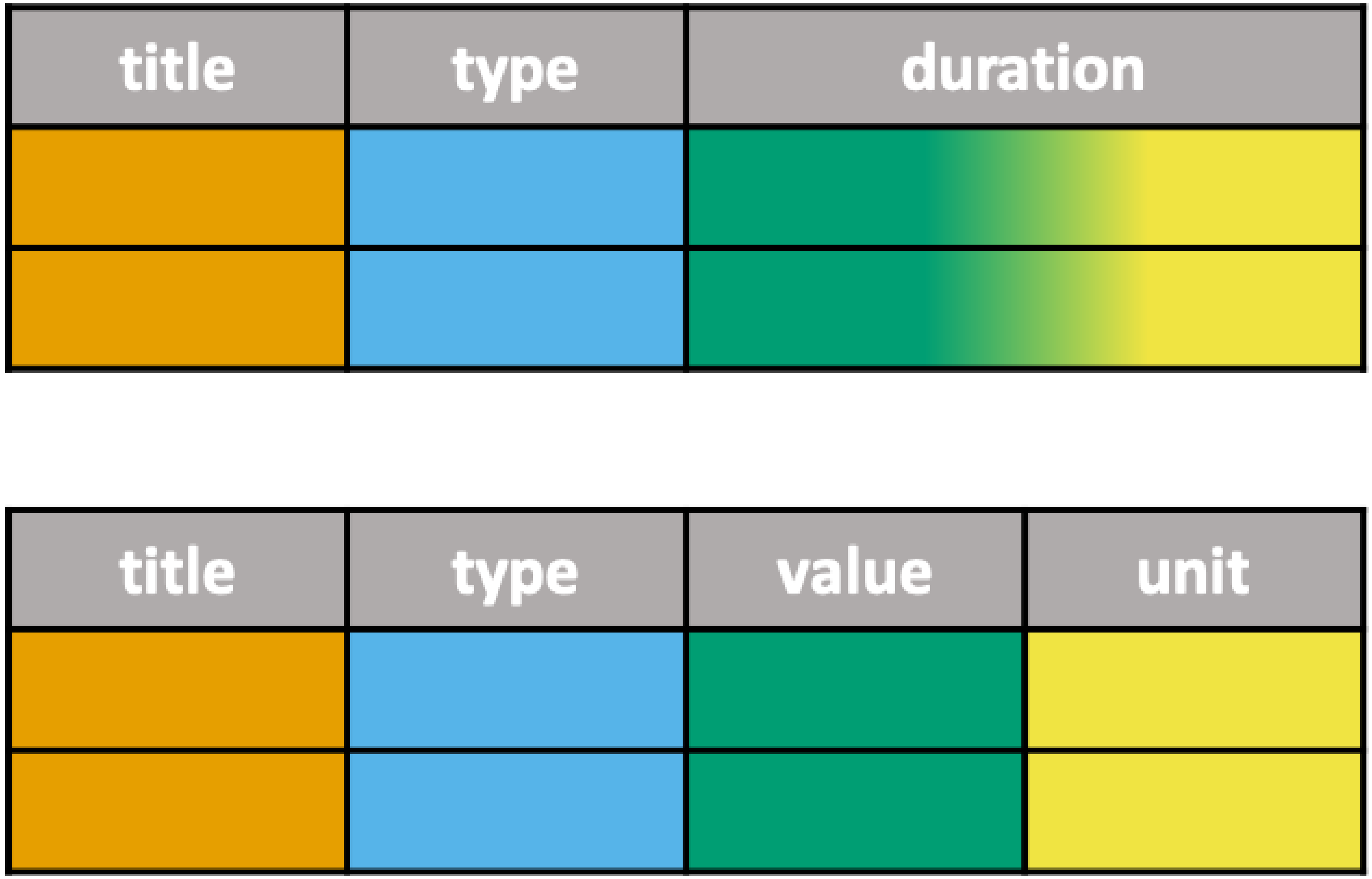
```
# A tibble: 5 x 4
  title                type     value unit
  <chr>                <chr>    <int> <chr>
1 Article 15           Movie      125 min
2 Kill Me If You Dare Movie      100 min
3 The Spy              TV Show      1 Seasons
4 The World We Make    Movie      108 min
5 Watchman             Movie       93 min
```

dplyr aggregation recap

```
netflix_df %>%
  separate(duration, into = c("value", "unit"), convert = TRUE) %>%
  group_by(type, unit) %>%
  summarize(mean_duration = mean(value))
```

```
# A tibble: 2 x 3
# Groups:   type [2]
  type     unit     mean_duration
  <chr>    <chr>            <dbl>
1 Movie    min              98.6
2 TV Show  Seasons           1.85
```

Separating variables over columns



Combining multiple columns into one

```
star_wars_df
```

```
# A tibble: 4 x 2
  given_name family_name
  <chr>      <chr>
1 Luke       Skywalker
2 Han        Solo
3 Leia       Organa
4 R2         D2
```

Combining multiple columns into one

```
star_wars_df %>%
  unite("name", given_name, family_name)
```

```
# A tibble: 4 x 1
  name
  <chr>
1 Luke_Skywalker
2 Han_Solo
3 Leia_Organa
4 R2_D2
```

Combining multiple columns into one

```r
star_wars_df %>%
  unite("name", given_name, family_name, sep = " ")
```

```
# A tibble: 4 x 1
  name
  <chr>
1 Luke Skywalker
2 Han Solo
3 Leia Organa
4 R2 D2
```

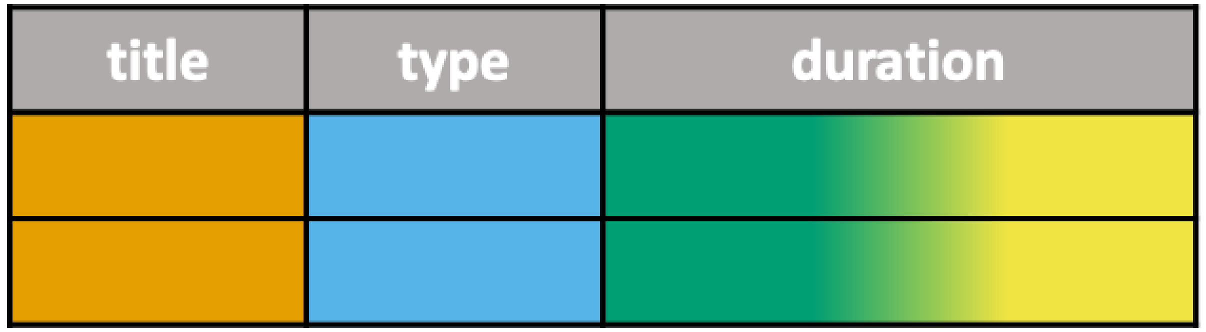
Multiple values in a single cell

```r
drink_df
```

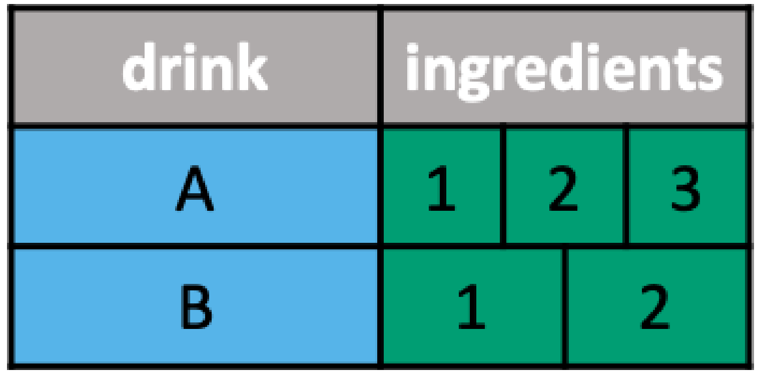
```
# A tibble: 2 x 2
  drink           ingredients
  <chr>           <chr>
1 Chocolate milk milk, chocolate, sugar
2 Orange juice   oranges, sugar
```

Multiple values in a single cell

Netflix data

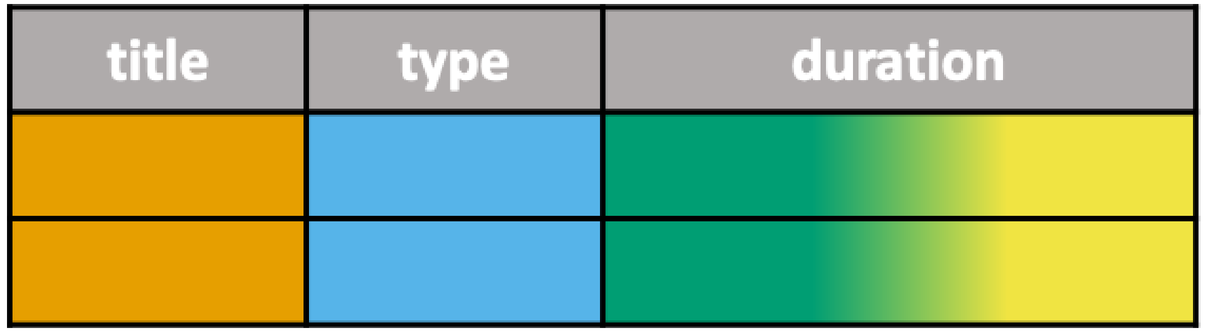


Drinks data

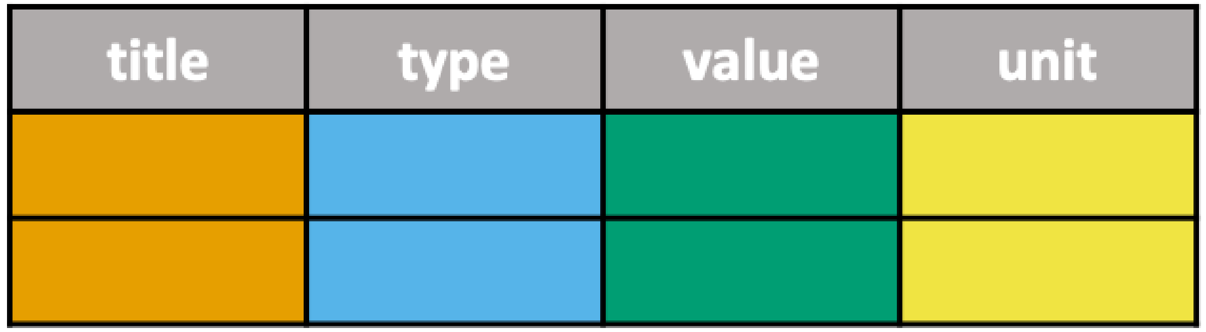


Multiple values in a single cell

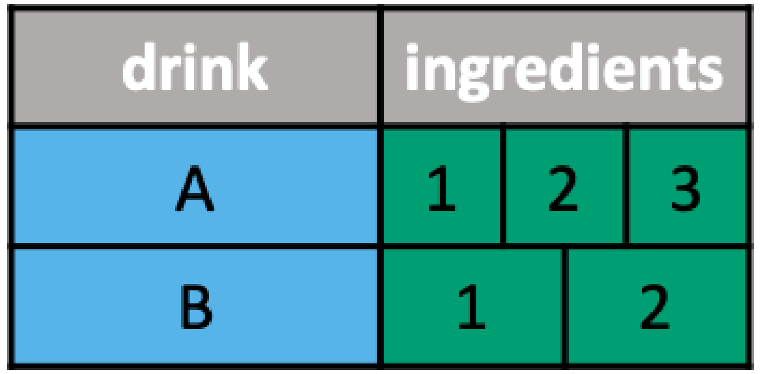
Netflix data



Values to variables

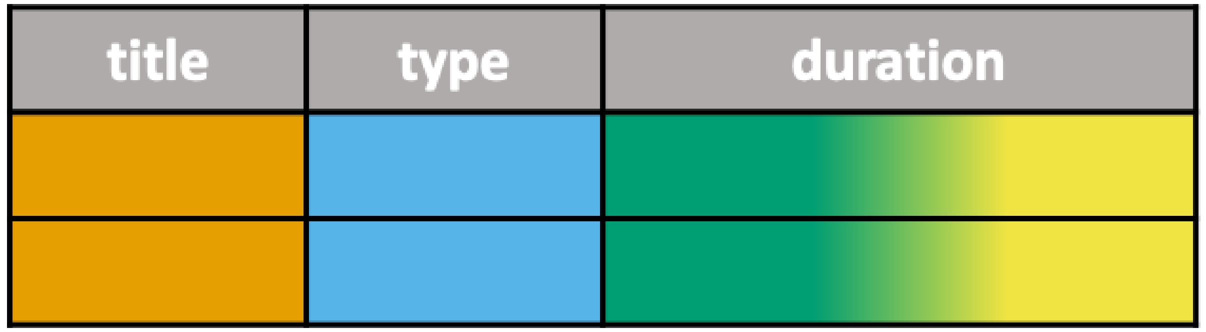


Drinks data



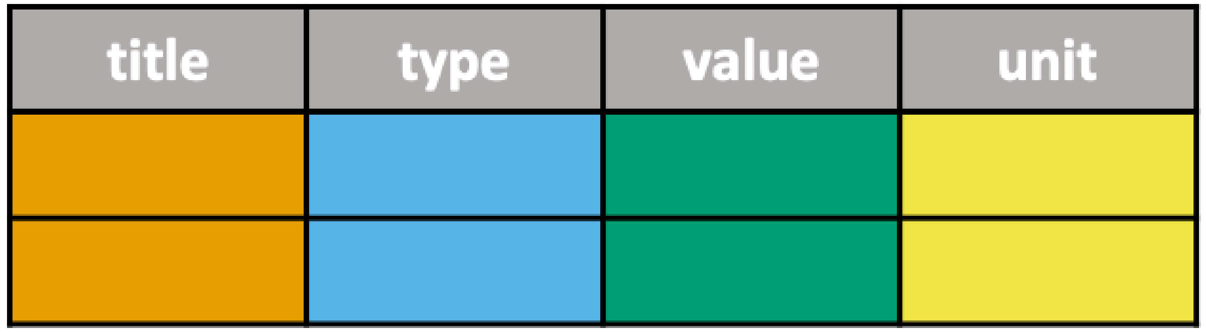
Multiple values in a single cell

Netflix data



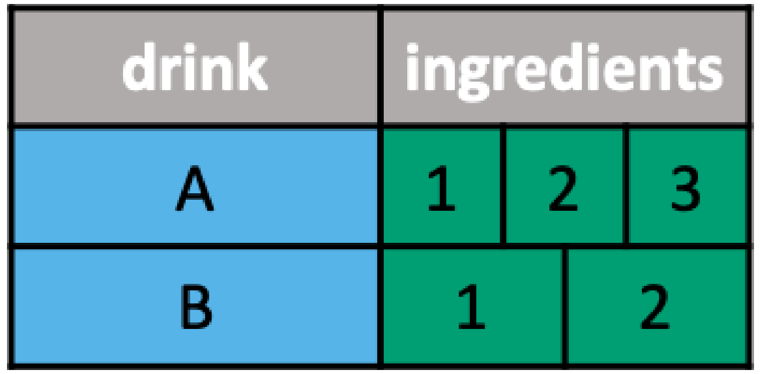
title	type	duration
🟧	🟦	🟩🟨
🟧	🟦	🟩🟨

Values to variables



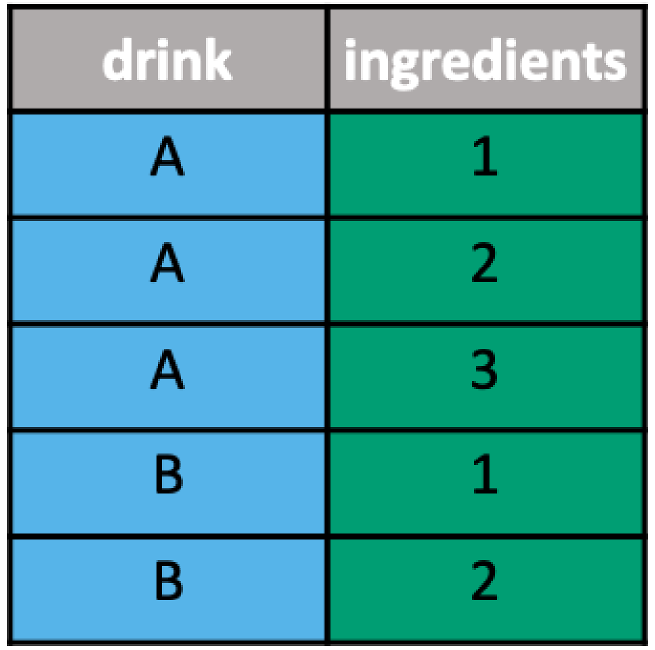
title	type	value	unit
🟧	🟦	🟩	🟨
🟧	🟦	🟩	🟨

Drinks data



drink	ingredients		
A	1	2	3
B	1	2	

Values to observations



drink	ingredients
A	1
A	2
A	3
B	1
B	2

Separating values over rows

```r
drink_df %>%
  separate_rows(ingredients, sep = ", ")
```

```
# A tibble: 5 x 2
  drink          ingredients
  <chr>          <chr>
1 Chocolate milk milk
2 Chocolate milk chocolate
3 Chocolate milk sugar
4 Orange juice   oranges
5 Orange juice   sugar
```

Counting ingredients

```
drink_df %>%
  separate_rows(ingredients, sep = ", ") %>%
  count(drink)
```

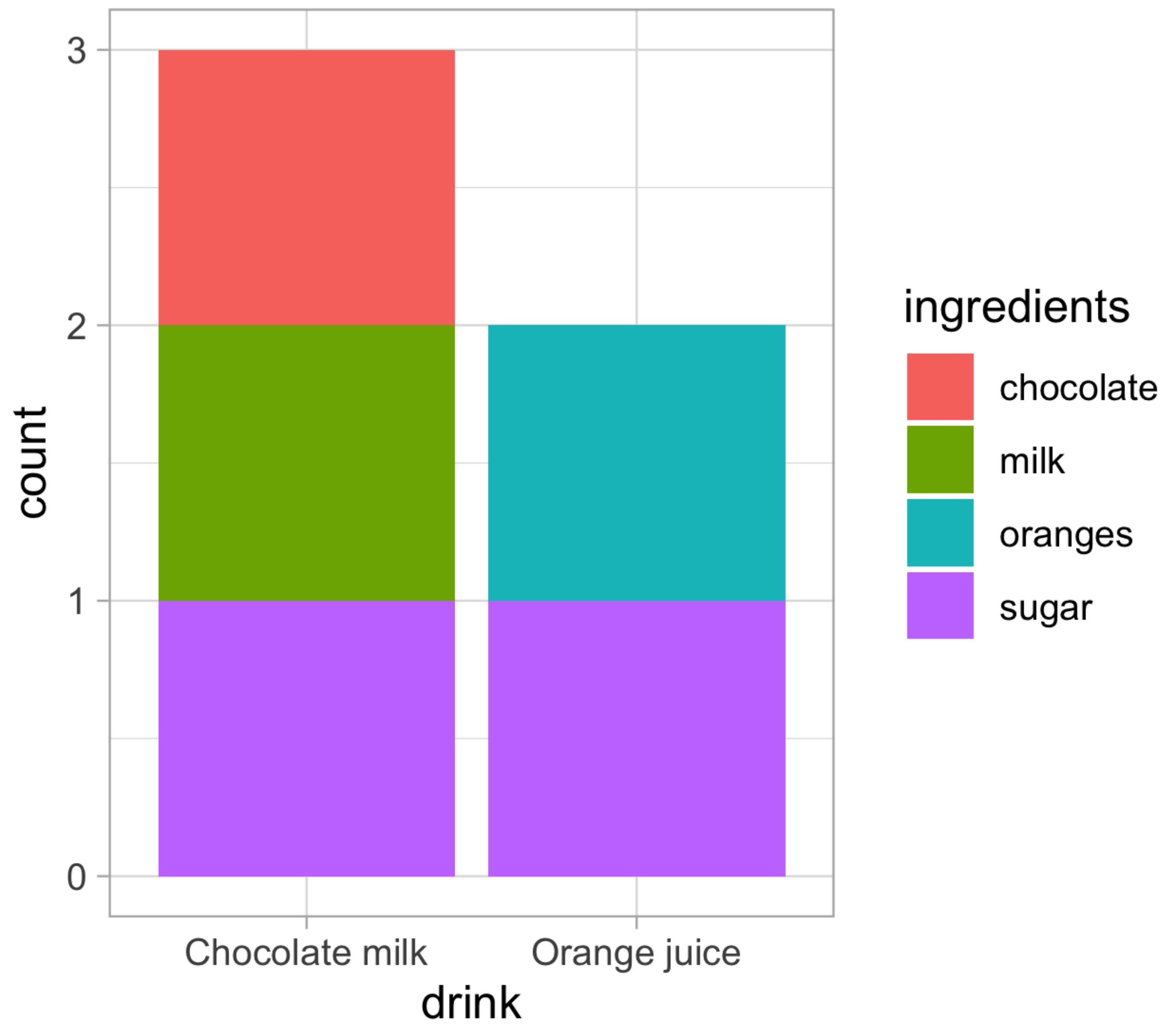
```
# A tibble: 2 x 2
  drink              n
  <chr>          <int>
1 Chocolate milk     3
2 Orange juice       2
```

```
drink_df %>%
  separate_rows(ingredients, sep = ", ") %>%
  count(ingredients)
```

```
# A tibble: 4 x 2
  ingredients     n
  <chr>       <int>
1 chocolate       1
2 milk            1
3 oranges         1
4 sugar           2
```

Visualizing ingredients

```
drink_df %>%
  separate_rows(ingredients, sep = ", ") %>%
  ggplot(aes(x=drink, fill=ingredients)) +
  geom_bar()
```



Let's practice!

Missing values

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

Missing values in R

NA = Not Available

```
# A tibble: 5 x 4
  drink          ingredient quantity unit
  <chr>          <chr>         <int> <chr>
1 Chocolate milk milk              1 L
2 Chocolate milk chocolate       100 g
3 Chocolate milk sugar            20 g
4 Orange juice   oranges           3 NA
5 Orange juice   sugar            20 g
```

Imputing with a default value: replace_na()

```
moon_df
```

```
# A tibble: 4 x 2
   year people_on_moon
  <int>          <int>
1  1969              4
2  1970             NA
3  1971              4
4  1972              4
5  1973             NA
```

Imputing with a default value: replace_na()

```r
moon_df %>%
  replace_na(list(people_on_moon = 0L))
```

```
# A tibble: 4 x 2
   year people_on_moon
  <int>          <int>
1  1969              4
2  1970              0
3  1971              4
4  1972              4
5  1973              0
```

```r
typeof(0L)
```

```
[1] "integer"
```

```r
typeof(0)
```

```
[1] "double"
```

Imputing with the most recent value: fill()

```
cumul_moon_df
```

```
# A tibble: 5 x 3
   year people_on_moon total_people_on_moon
  <int>          <int>                <int>
1  1969              4                    4
2  1970             NA                   NA
3  1971              4                    8
4  1972              4                   12
5  1973             NA                   NA
```

Imputing with the most recent value: fill()

```
cumul_moon_df %>%
  fill(total_people_on_moon)
```

```
# A tibble: 5 x 3
   year people_on_moon total_people_on_moon
  <int>          <int>                <int>
1  1969              4                    4
2  1970             NA                    4
3  1971              4                    8
4  1972              4                   12
5  1973             NA                   12
```

fill() imputation options

```
cumul_moon_df %>%
  fill(total_people_on_moon, .direction = "down")
```

```
# A tibble: 5 x 3
   year people_on_moon total_people_on_moon
  <int>          <int>                <int>
1  1969              4                    4
2  1970             NA                    4
3  1971              4                    8
4  1972              4                   12
5  1973             NA                   12
```

fill() imputation options

```r
cumul_moon_df %>%
  fill(total_people_on_moon, .direction = "up")
```

```
# A tibble: 5 x 3
   year people_on_moon total_people_on_moon
  <int>          <int>                <int>
1  1969              4                    4
2  1970             NA                    8
3  1971              4                    8
4  1972              4                   12
5  1973             NA                   NA
```

Removing rows with missing values: drop_na()

```
moon_df %>%
  drop_na()
```

```
# A tibble: 3 x 2
   year people_on_moon
  <int>          <int>
1  1969              4
2  1971              4
3  1972              4
```

drop_na() caveats

```
mars_df
```

```
# A tibble: 5 x 3
   year people_on_moon people_on_mars
  <int>          <int> <int>
1  1969              4 NA
2  1970             NA NA
3  1971              4 NA
4  1972              4 NA
5  1973             NA NA
```

drop_na() caveats

```r
mars_df %>%
  drop_na()
```

```
# A tibble: 0 x 3
# ... with 3 variables: year <int>, people_on_moon <int>, people_on_mars <int>
```

drop_na() caveats

```
mars_df %>%
  drop_na(people_on_moon)
```

```
# A tibble: 3 x 3
   year people_on_moon people_on_mars
  <int>          <int> <int>
1  1969              4 NA
2  1971              4 NA
3  1972              4 NA
```

Let's practice!

RESHAPING DATA WITH TIDYR