From wide to long data

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

datacamp

`separate()`

title	type	duration

title	type	value	unit

`separate_rows()`

drink	ingredients		
A	1	2	3
B	1	2	

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

Values in column headers

```
nuke_df
```

```
# A tibble: 2 x 6
  country             `1945` `1946` `1948` `1949` `1951`
  <chr>                <int>  <int>  <int>  <int>  <int>
1 United States            3      2      3     NA     16
2 Russian Federation      NA     NA     NA      1      2
```

Values in column headers

country	1945	1946
USA	3	2
USSR	NA	NA

country	year	n_bombs
USA	1945	3
USA	1946	2
USSR	1945	NA
USSR	1946	NA

The pivot_longer() function

```
nuke_df %>%
  pivot_longer(`1945`:`1951`)
```

```
# A tibble: 10 x 3
   country             name  value
   <chr>               <chr> <int>
 1 United States       1945      3
 2 United States       1946      2
 3 United States       1948      3
 4 United States       1949     NA
 5 United States       1951     16
 6 Russian Federation 1945     NA
 # ... with 4 more rows
```

The pivot_longer() function

```
nuke_df %>%
  pivot_longer(c(`1945`, `1946`, `1948`, `1949`, `1951`))
```

```
# A tibble: 10 x 3
   country             name  value
   <chr>               <chr> <int>
 1 United States       1945      3
 2 United States       1946      2
 3 United States       1948      3
 4 United States       1949     NA
 5 United States       1951     16
 6 Russian Federation 1945     NA
 # ... with 4 more rows
```

The pivot_longer() function

```
nuke_df %>%
  pivot_longer(-country)
```

```
# A tibble: 10 x 3
   country             name  value
   <chr>               <chr> <int>
 1 United States       1945      3
 2 United States       1946      2
 3 United States       1948      3
 4 United States       1949     NA
 5 United States       1951     16
 6 Russian Federation  1945     NA
 # ... with 4 more rows
```

pivot_longer() arguments

```
nuke_df %>%
  pivot_longer(-country, names_to = "year", values_to = "n_bombs")
```

```
# A tibble: 10 x 3
   country            year  n_bombs
   <chr>              <chr> <int>
 1 United States      1945      3
 2 United States      1946      2
 3 United States      1948      3
 4 United States      1949     NA
 5 United States      1951     16
 6 Russian Federation 1945     NA
 # ... with 4 more rows
```

pivot_longer() arguments

```
nuke_df %>%
  pivot_longer(
    -country,
    names_to = "year",
    values_to = "n_bombs",
    values_drop_na = TRUE
  )
```

```
# A tibble: 6 x 3
  country             year  n_bombs
  <chr>               <chr>   <int>
1 United States       1945        3
2 United States       1946        2
3 United States       1948        3
4 United States       1951       16
5 Russian Federation  1949        1
6 Russian Federation  1951        2
```

pivot_longer() arguments

```
nuke_df %>%
  pivot_longer(
    -country,
    names_to = "year",
    values_to = "n_bombs",
    values_drop_na = TRUE,
    names_transform = list(year = as.integer)
  )
```

```
# A tibble: 6 x 3
  country              year n_bombs
  <chr>               <int>   <int>
1 United States        1945       3
2 United States        1946       2
3 United States        1948       3
4 United States        1951      16
5 Russian Federation   1949       1
6 Russian Federation   1951       2
```

Let's practice!

RESHAPING DATA WITH TIDYR

Deriving variables from column headers

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

datacamp

Soviet space dogs

```
space_dogs_df
```

```
# A tibble: 42 x 4
   date       name_1    name_2   result
   <date>     <chr>     <chr>    <chr>
 1 1951-06-26 Lisa-2    Ryzhik-2 recovered safely
 2 1951-07-22 Dezik     Tsygan   recovered safely
 3 1951-07-29 Dezik     Lisa     parachute failed, both dogs died
 4 1951-08-15 Chizhik   Mishka   recovered safely
 5 1951-08-19 Ryzhik    Smeliy   recovered safely
 # ... with 37 more rows
```

Soviet space dogs: a basic pivot operation

```
dog_df %>%
  pivot_longer(
    c(name_1, name_2),
    names_to = "id",
    values_to = "name",
    values_drop_na = TRUE
  ) %>%
  select(-result)
```

```
# A tibble: 81 x 3
   date       id     name
   <date>     <chr>  <chr>
 1 1951-06-26 name_1 Lisa-2
 2 1951-06-26 name_2 Ryzhik-2
 3 1951-07-22 name_1 Dezik
 4 1951-07-22 name_2 Tsygan
 5 1951-07-29 name_1 Dezik
 6 1951-07-29 name_2 Lisa
 7 1951-08-15 name_1 Chizhik
 8 1951-08-15 name_2 Mishka
 9 1951-08-19 name_1 Ryzhik
 # ... with 72 more rows
```

Soviet space dogs: removing a prefix

```
dog_df %>%
  pivot_longer(
    c(name_1, name_2),
    names_to = "id",
    values_to = "name",
    values_drop_na = TRUE,
    names_prefix = "name_"
  ) %>%
  select(-result)
```

```
# A tibble: 81 x 3
   date       id    name
   <date>     <chr> <chr>
 1 1951-06-26 1     Lisa-2
 2 1951-06-26 2     Ryzhik-2
 3 1951-07-22 1     Dezik
 4 1951-07-22 2     Tsygan
 5 1951-07-29 1     Dezik
 6 1951-07-29 2     Lisa
 7 1951-08-15 1     Chizhik
 8 1951-08-15 2     Mishka
 9 1951-08-19 1     Ryzhik
 # ... with 72 more rows
```

Soviet space dogs: transforming data types

```
dog_df %>%
  pivot_longer(
    c(name_1, name_2),
    names_to = "id",
    values_to = "name",
    values_drop_na = TRUE,
    names_prefix = "name_",
    names_transform = list(id = as.integer)
  ) %>%
  select(-result)
```

```
# A tibble: 81 x 3
   date           id name
   <date>      <int> <chr>
 1 1951-06-26     1 Lisa-2
 2 1951-06-26     2 Ryzhik-2
 3 1951-07-22     1 Dezik
 4 1951-07-22     2 Tsygan
 5 1951-07-29     1 Dezik
 6 1951-07-29     2 Lisa
 7 1951-08-15     1 Chizhik
 8 1951-08-15     2 Mishka
 9 1951-08-19     1 Ryzhik
 # ... with 72 more rows
```

Soviet space dogs: the starts_with() function

```
dog_df %>%
  pivot_longer(
    starts_with("name_"),
    names_to = "id",
    values_to = "name",
    values_drop_na = TRUE,
    names_prefix = "name_",
    names_transform = list(id = as.integer)
  ) %>%
  select(-result)
```

```
# A tibble: 81 x 3
   date           id name
   <date>      <int> <chr>
 1 1951-06-26     1 Lisa-2
 2 1951-06-26     2 Ryzhik-2
 3 1951-07-22     1 Dezik
 4 1951-07-22     2 Tsygan
 5 1951-07-29     1 Dezik
 6 1951-07-29     2 Lisa
 7 1951-08-15     1 Chizhik
 8 1951-08-15     2 Mishka
 9 1951-08-19     1 Ryzhik
 # ... with 72 more rows
```

Apple revenue: two variables per column name

```
apple_revenue_df
```

```
# A tibble: 4 x 5
  segment `2019_Q1` `2019_Q2` `2019_Q3` `2019_Q4`
  <chr>       <dbl>     <dbl>     <dbl>     <dbl>
1 iPhone      52.0      31.0      26.0      33.4
2 Mac          7.42      5.51      5.82      6.99
3 iPad         6.73      4.87      5.02      4.66
4 Other       18.2      16.6      17.0      19.0
```

Apple revenue: visualizing issue and solution

segment	2019_Q1	2019_Q2
iPhone	52.0	31.0
Mac	7.42	5.51

segment	year	quarter	revenue
iPhone	2019	1	52.0
iPhone	2019	2	31.0
Mac	2019	1	7.42
Mac	2019	2	5.51

Apple revenue: Advanced pivoting

```
apple_df %>%
  pivot_longer(
    -segment,
    names_to = c("year", "quarter"),
    values_to = "revenue",
    names_sep = "_Q",
    names_transform = list(
      year = as.integer,
      quarter = as.integer
      )
  )
```

```
# A tibble: 16 x 4
   segment  year quarter revenue
   <chr>   <int>   <int>   <dbl>
 1 iPhone   2019       1   52.0
 2 iPhone   2019       2   31.0
 3 iPhone   2019       3   26.0
 4 iPhone   2019       4   33.4
 5 Mac      2019       1    7.42
 6 Mac      2019       2    5.51
 7 Mac      2019       3    5.82
 8 Mac      2019       4    6.99
 # ... with 8 more rows
```

Let's practice!

RESHAPING DATA WITH TIDYR

Deriving variables from complex column headers

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

datacamp

Separating column headers into variables

segment	2019_Q1	2019_Q2
iPhone	52.0	31.0
Mac	7.42	5.51

segment	year	quarter	revenue
iPhone	2019	1	52.0
iPhone	2019	2	31.0
Mac	2019	1	7.42
Mac	2019	2	5.51

Multiple variable combinations in column headers

```
who_df
```

```
# A tibble: 181 x 5
   country             female_pct.obese male_pct.obese female_life.exp male_life.exp
   <chr>                          <dbl>          <dbl>           <dbl>         <dbl>
 1 Afghanistan                      7.6            3.2            64.5          61
 2 Albania                         21.8           21.6            78.6          74.3
 3 Algeria                         34.9           19.9            77.4          75.4
 4 Angola                          12.1            4              64.9          60.3
 5 Antigua and Barbuda             25.9           11.6            77.5          72.5
 6 Argentina                       29             27.3            80.3          73.5
 7 Armenia                         23             17.1            78.1          71.2
 8 Australia                       28.4           29.6            84.8          81
```

Multiple variable combinations in column headers

country	female_pct.obese	male_pct.obese	female_life.exp	male_life.exp
Afghanistan	7.6	3.2	64.5	61
Albania	21.8	21.6	78.6	74.3

country	sex	pct.obese	life.exp
Afghanistan	female	7.6	52.0
Afghanistan	male	3.2	31.0
Albania	female	21.8	7.42
Albania	male	21.6	5.51

The special .value name

```
who_df %>%
  # Example input column name = male_obesity.pct
  pivot_longer(-country,
               names_to = c("sex", ".value"),
               names_sep = "_")
```

```
# A tibble: 362 x 4
   country             sex    pct.obese life.exp
   <chr>               <chr>      <dbl>    <dbl>
 1 Afghanistan         female       7.6     64.5
 2 Afghanistan         male         3.2     61
 3 Albania             female      21.8     78.6
```

pivot_longer() recap

country	1945	1946
USA	3	2
USSR	NA	NA

country	year	n_bombs
USA	1945	3
USA	1946	2
USSR	1945	NA
USSR	1946	NA

segment	2019_Q1	2019_Q2
iPhone	52.0	31.0
Mac	7.42	5.51

segment	year	quarter	revenue
iPhone	2019	1	52.0
iPhone	2019	2	31.0
Mac	2019	1	7.42
Mac	2019	2	5.51

country	female_pct.obese	male_pct.obese	female_life.exp	male_life.exp
Afghanistan	7.6	3.2	64.5	61
Albania	21.8	21.6	78.6	74.3

country	sex	pct.obese	life.exp
Afghanistan	female	7.6	52.0
Afghanistan	male	3.2	31.0
Albania	female	21.8	7.42
Albania	male	21.6	5.51

Uncounting data

```
nuke_df
```

```
# A tibble: 8 x 2
  country           n_bombs
  <chr>               <int>
1 Pakistan                2
2 India                   6
3 North Korea             6
4 United Kingdom         21
5 China                  45
6 France                200
7 Russian Federation    726
8 United States        1150
```

The uncount() function

```
nuke_df %>%
  uncount(n_bombs)
```

```
# A tibble: 2,156 x 1
   country
   <chr>
 1 Pakistan
 2 Pakistan
 3 India
 4 India
 5 India
 6 India
# ... with 2,150 more rows
```

The uncount() function

```
nuke_df %>%
  uncount(2)
```

```
# A tibble: 16 x 2
   country             n_bombs
   <chr>                 <int>
 1 Pakistan                  2
 2 Pakistan                  2
 3 India                     6
 4 India                     6
 5 North Korea               6
 6 North Korea               6
# ... with 10 more rows
```

The uncount() function

```
nuke_df %>%
  uncount(n_bombs, .id = "bomb_id")
```

```
# A tibble: 2,156 x 2
   country     bomb_id
   <chr>         <int>
 1 Pakistan          1
 2 Pakistan          2
 3 India             1
 4 India             2
 5 India             3
 6 India             4
# ... with 2,150 more rows
```

Let's practice!

RESHAPING DATA WITH TIDYR

From long to wide data

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

datacamp

Variable names in a column

```
who_df
```

```
# A tibble: 362 x 3
   country             metric    value
   <chr>               <chr>     <dbl>
 1 Afghanistan         life_exp   62.7
 2 Afghanistan         pct_obese   5.5
 3 Albania             life_exp   76.4
 4 Albania             pct_obese  21.7
# ... with 358 more rows
```

Variable names in a column

country	metric	value
Afghanistan	life_exp	62.7
Afghanistan	pct_obese	5.5
Albania	life_exp	76.4
Albania	pct_obese	21.7

country	pct_obese	life_exp
Afghanistan	5.5	62.7
Albania	21.7	76.4

The pivot_wider() function

```
who_df %>%
  pivot_wider(names_from = metric, values_from = value)
```

```
# A tibble: 181 x 3
   country             life_exp pct_obese
   <chr>                  <dbl>     <dbl>
 1 Afghanistan             62.7       5.5
 2 Albania                 76.4      21.7
 3 Algeria                 76.4      27.4
 4 Angola                  62.6       8.2
# ... with 177 more rows
```

The pivot_wider() function

```
who_long_df %>%
  pivot_wider(names_from = metric, values_from = value, names_prefix = "national_")
```

```
# A tibble: 181 x 3
   country             national_life_exp national_pct_obese
   <chr>                           <dbl>               <dbl>
 1 Afghanistan                      62.7                 5.5
 2 Albania                          76.4                21.7
 3 Algeria                          76.4                27.4
 4 Angola                           62.6                 8.2
# ... with 177 more rows
```

Transposing a data frame

```
sideways_df
```

```
# A tibble: 2 x 5
  variable       `1969` `1970` `1971` `1972`
  <chr>           <int>  <int>  <int>  <int>
1 people_on_moon      4      0      4      4
2 nuclear_bombs      82     85     59     62
```

Transposing a data frame

variable	`1969`	`1970`	`1971`	`1972`
people_on_moon	4	0	4	4
nuclear_bombs	82	85	59	62

year	people_on_moon	nuclear_bombs
1969	4	82
1970	0	85
1971	4	59
1972	4	62

Transposing a data frame: step 1

```
sideways_df %>%
  pivot_longer(-variable, names_to = "year", names_transform = list(year = as.integer))
```

```
# A tibble: 8 x 3
  variable       year  value
  <chr>          <int> <int>
1 people_on_moon 1969      4
2 people_on_moon 1970      0
3 people_on_moon 1971      4
4 people_on_moon 1972      4
5 nuclear_bombs  1969     82
6 nuclear_bombs  1970     85
7 nuclear_bombs  1971     59
8 nuclear_bombs  1972     62
```

Transposing a data frame: step 2

```
sideways_df %>%
  pivot_longer(-variable, names_to = "year", names_transform = list(year = as.integer)) %>%
  pivot_wider(names_from = variable, values_from = value)
```

```
# A tibble: 4 x 3
  year  people_on_moon nuclear_bombs
  <int>          <int>         <int>
1 1969               4            82
2 1970               0            85
3 1971               4            59
4 1972               4            62
```

Let's practice!

RESHAPING DATA WITH TIDYR