Intro to non-rectangular data

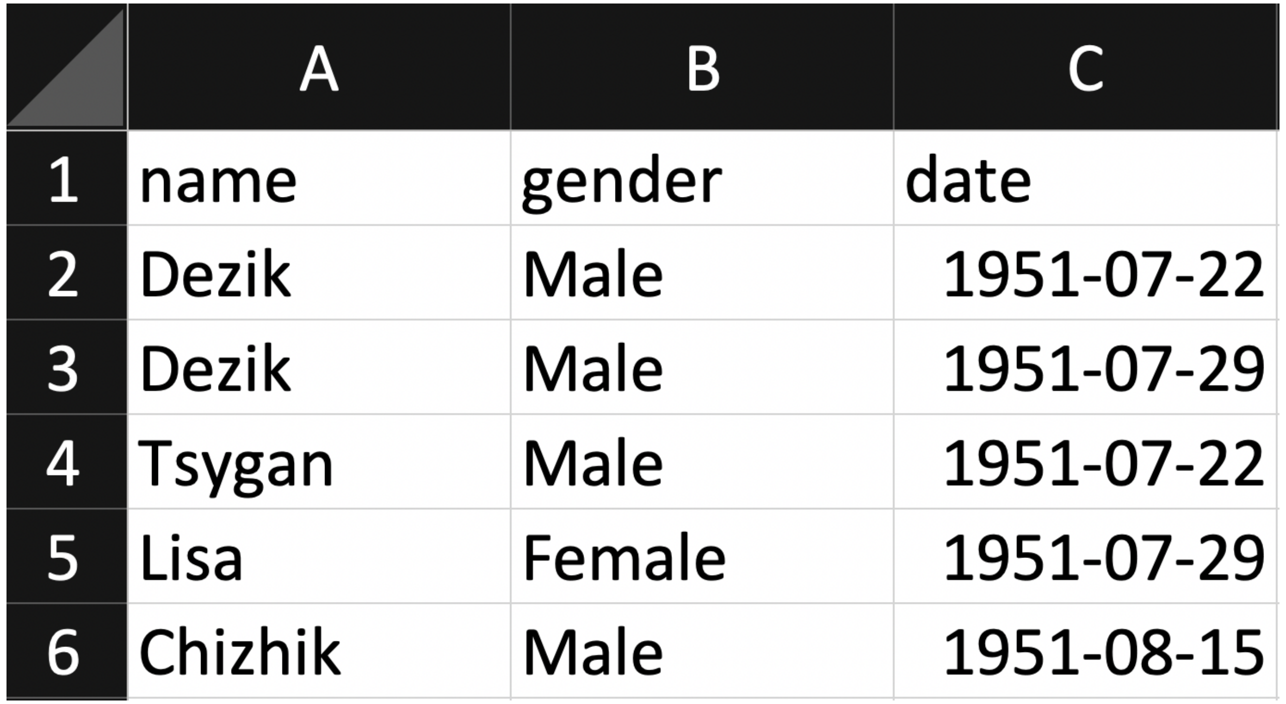
RESHAPING DATA WITH TIDYR

Jeroen Boeye

Head of Machine Learning, Faktion

Rectangular data

Spreadsheets



	A	B	C
1	name	gender	date
2	Dezik	Male	1951-07-22
3	Dezik	Male	1951-07-29
4	Tsygan	Male	1951-07-22
5	Lisa	Female	1951-07-29
6	Chizhik	Male	1951-08-15

CSV

```
name,gender,date
Dezik,Male,1951-07-22
Dezik,Male,1951-07-29
Tsygan,Male,1951-07-22
Lisa,Female,1951-07-29
Chizhik,Male,1951-08-15
```

Non-rectangular formats

JSON

```
{
    "name": "Darth Vader",
    "species": "Human",
    "homeworld": "Tatooine",
    "films": [
        "Revenge of the Sith",
        "Return of the Jedi",
        "The Empire Strikes Back",
        "A New Hope"
    ]
}
```

XML

```
<note>
  <from>Teacher</from>
  <to>Student</to>
  <heading>Almost there</heading>
  <body>It's the final chapter!</body>
</note>
```

[1] Star Wars data from the repurrrsive package.

A list of lists of lists

```
rjson::fromJSON(file = "star_wars.json")
```

```
[[1]]
[[1]]$name
[1] "Darth Vader"
[[1]]$films
[1] "Revenge of the Sith" "Return of the Jedi" "The Empire Strikes Back" "A New Hope"

[[2]]
[[2]]$name
[1] "Jar Jar Binks"
[[2]]$films
[1] "Attack of the Clones" "The Phantom Menace"
```

A first step to rectangling

```r
star_wars_list <- rjson::fromJSON(file = "star_wars.json")
tibble(character = star_wars_list)
```

```
# A tibble: 2 x 1
  character
  <list>
1 <named list [2]>
2 <named list [2]>
```

Unnesting lists to columns

```
tibble(character = star_wars_list) %>%
  unnest_wider(character)
```

```
# A tibble: 2 x 2
  name           films
  <chr>          <list>
1 Darth Vader    <chr [4]>
2 Jar Jar Binks <chr [2]>
```

Unnesting lists to columns

```
tibble(character = star_wars_list) %>%
  unnest_wider(character) %>%
  unnest_wider(films)
```

```
# A tibble: 2 x 5
  name           ...1                 ...2               ...3                    ...4
  <chr>          <chr>                <chr>              <chr>                   <chr>
1 Darth Vader    Revenge of the Sith  Return of the Jedi The Empire Strikes Back A New Hope
2 Jar Jar Binks Attack of the Clones The Phantom Menace NA                      NA
```

Let's practice!

RESHAPING DATA WITH TIDYR

From nested values to observations

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

The unnest_wider() function recap

```
tibble(character = star_wars_list) %>%
  unnest_wider(character)
```

```
# A tibble: 2 x 2
  name           films
  <chr>          <list>
1 Darth Vader    <chr [4]>
2 Jar Jar Binks <chr [2]>
```

The unnest_wider() function recap

```
tibble(character = star_wars_list) %>%
  unnest_wider(character) %>%
  unnest_wider(films)
```

```
# A tibble: 2 x 5
  name          ...1                 ...2               ...3                    ...4
  <chr>         <chr>                <chr>              <chr>                   <chr>
1 Darth Vader   Revenge of the Sith  Return of the Jedi The Empire Strikes Back A New Hope
2 Jar Jar Binks Attack of the Clones The Phantom Menace NA                      NA
```

The unnest_longer() function

```
tibble(character = star_wars_list) %>%
  unnest_wider(character) %>%
  unnest_longer(films)
```

```
# A tibble: 45 x 2
   name        films
   <chr>       <chr>
 1 Chewbacca   Revenge of the Sith
 2 Chewbacca   Return of the Jedi
 3 Chewbacca   The Empire Strikes Back
 4 Chewbacca   A New Hope
 5 Chewbacca   The Force Awakens
 6 Darth Vader Revenge of the Sith
 7 Darth Vader Return of the Jedi
 8 Darth Vader The Empire Strikes Back
# ... with 37 more rows
```

Rectangling deeply nested data

```
course_df
```

```
# A tibble: 4 x 2
  ch_id metadata
  <chr> <list>
1 CH1   <named list [3]>
2 CH2   <named list [3]>
3 CH3   <named list [3]>
4 CH4   <named list [3]>
```

Rectangling deeply nested data

```
course_df %>%
  unnest_wider(metadata)
```

```
# A tibble: 4 x 4
  ch_id chapter_title               status      lessons
  <chr> <chr>                       <chr>       <list>
1 CH1   Tidy Data                   Complete    <list [3]>
2 CH2   From Wide to Long and Back Complete    <list [4]>
3 CH3   Expanding Data              Complete    <list [3]>
4 CH4   Rectangling Data            In progress <list [4]>
```

Combining unnest_wider() and unnest_longer()

```
course_df %>%
  unnest_wider(metadata) %>%
  unnest_longer(lessons)
```

```
# A tibble: 14 x 4
   ch_id chapter_title                 status      lessons
   <chr> <chr>                         <chr>       <list>
 1 CH1   Tidy Data                     Complete    <named list [3]>
 2 CH1   Tidy Data                     Complete    <named list [3]>
 3 CH1   Tidy Data                     Complete    <named list [3]>
 4 CH2   From Wide to Long and Back Complete    <named list [3]>
# ... with 10 more rows
```

Digging deeper

```
course_df %>%
  unnest_wider(metadata) %>%
  unnest_longer(lessons) %>%
  unnest_wider(lessons)
```

```
# A tibble: 14 x 6
  ch_id chapter_title              status   l_id  lesson_title                 exercises
  <chr> <chr>                      <chr>    <chr> <chr>                        <list>
1 CH1   Tidy Data                  Complete L1    What is tidy data?           <list [2]>
2 CH1   Tidy Data                  Complete L2    Columns with multiple values <list [3]>
3 CH1   Tidy Data                  Complete L3    Missing values               <list [3]>
4 CH2   From Wide to Long and Back Complete L1    From wide to long data       <list [3]>
# ... with 10 more rows
```

And deeper ...

```
course_df %>%
  unnest_wider(metadata) %>%
  unnest_longer(lessons) %>%
  unnest_wider(lessons) %>%
  select(ch_id, l_id, exercises) %>%
  unnest_longer(exercises)
```

```
# A tibble: 41 x 3
   ch_id l_id  exercises
   <chr> <chr> <list>
 1 CH1   L1    <named list [2]>
 2 CH1   L1    <named list [2]>
 3 CH1   L2    <named list [2]>
 4 CH1   L2    <named list [2]>
 5 CH1   L2    <named list [2]>
 6 CH1   L3    <named list [2]>
 7 CH1   L3    <named list [2]>
 8 CH1   L3    <named list [2]>
# ... with 33 more rows
```

And deeper ...

```
course_df %>%
  unnest_wider(metadata) %>%
  unnest_longer(lessons) %>%
  unnest_wider(lessons) %>%
  select(ch_id, l_id, exercises) %>%
  unnest_longer(exercises) %>%
  unnest_wider(exercises)
```

```
# A tibble: 41 x 4
   ch_id l_id  ex_id complete
   <chr> <chr> <chr> <lgl>
 1 CH1   L1    E1    TRUE
 2 CH1   L1    E2    TRUE
 3 CH1   L2    E1    TRUE
 4 CH1   L2    E2    TRUE
 5 CH1   L2    E3    TRUE
 6 CH1   L3    E1    TRUE
 7 CH1   L3    E2    TRUE
 8 CH1   L3    E3    TRUE
# ... with 33 more rows
```

Course status update

```
course_df %>%
  unnest_wider(metadata) %>%
  unnest_longer(lessons) %>%
  unnest_wider(lessons) %>%
  select(ch_id, l_id, exercises) %>%
  unnest_longer(exercises) %>%
  unnest_wider(exercises) %>%
  summarize(pct_complete = mean(complete))
```

```
# A tibble: 1 x 1
  pct_complete
         <dbl>
1        0.780
```

Let's practice!

RESHAPING DATA WITH TIDYR

Selecting nested variables

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

Unnesting list columns completely

```
planet_df %>%
  unnest_longer(moons) %>%
  unnest_wider(moons) %>%
  unnest_wider(moon_data)
```

```
# A tibble: 174 x 4
   planet  moon_name radius density
   <chr>   <chr>      <dbl>   <dbl>
 1 Mercury NA           NA    NA
 2 Venus   NA           NA    NA
 3 Earth   Moon       1738.    3.34
 4 Jupiter Io         1822.    3.53
 5 Jupiter Europa     1561.    3.01
 6 Jupiter Ganymede   2631.    1.94
 7 Jupiter Callisto   2410.    1.83
 8 Jupiter Amalthea     83.4   0.849
# ... with 166 more rows
```

Selective unnesting with hoist()

```
moons :List of 8
 $ :List of 67
  ..$ :List of 2
  .. ..$ moon_name: chr "Io"
  .. ..$ moon_data:List of 2
  .. .. ..$ radius : num 1822
  .. .. ..$ density: num 3.53
```

```
planet_df %>%
  hoist(
    moons,
    first_moon = list(1, "moon_name"),
    radius = list(1, "moon_data", "radius"))
```

```
# A tibble: 8 x 4
  planet  first_moon radius moons
  <chr>   <chr>       <dbl> <list>
1 Mercury NA           NA   <NULL>
2 Venus   NA           NA   <NULL>
3 Earth   Moon       1738.  <list [1]>
4 Jupiter Io         1822.  <list [67]>
5 Mars    Phobos       11.1 <list [2]>
6 Neptune Triton     1353.  <list [14]>
7 Saturn  Mimas       198.  <list [61]>
8 Uranus  Ariel       579.  <list [27]>
```

Selective unnesting with hoist()

```r
planet_df %>%
  unnest_longer(moons) %>%
  hoist(
    moons,
    moon_name = "moon_name",
    radius = list("moon_data", "radius")
  )
```

```
# A tibble: 174 x 4
   planet  moon_name radius moons
   <chr>   <chr>      <dbl> <list>
 1 Mercury NA           NA  <NULL>
 2 Venus   NA           NA  <NULL>
 3 Earth   Moon       1738. <named list [1]>
 4 Jupiter Io         1822. <named list [1]>
 5 Jupiter Europa     1561. <named list [1]>
 6 Jupiter Ganymede   2631. <named list [1]>
 7 Jupiter Callisto   2410. <named list [1]>
 8 Jupiter Amalthea     83.4 <named list [1]>
 9 Jupiter Himalia      85  <named list [1]>
10 Jupiter Elara        43  <named list [1]>
# ... with 164 more rows
```

Unnesting Google Maps data

```
city_df
```

```
# A tibble: 5 x 2
  city         json
  <chr>        <list>
1 Beijing      <named list [2]>
2 Buenos Aires <named list [2]>
3 New Delhi    <named list [2]>
4 New York     <named list [2]>
5 Paris        <named list [2]>
```

[1] Example from tidyr documentation: https://tidyr.tidyverse.org/articles/rectangle.html

Unnesting Google maps data

```
city_df %>%
  unnest_wider(json)
```

```
# A tibble: 5 x 3
  city         results     status
  <chr>        <list>      <chr>
1 Beijing      <list [1]> OK
2 Buenos Aires <list [1]> OK
3 New Delhi    <list [1]> OK
4 New York     <list [1]> OK
5 Paris        <list [1]> OK
```

[1] Example from tidyr documentation: https://tidyr.tidyverse.org/articles/rectangle.html

Unnesting Google maps data

```
city_df %>%
  unnest_wider(json) %>%
  unnest_longer(results) %>%
  unnest_wider(results)
```

```
  city         address_components formatted_address       geometry
  <chr>        <list>             <chr>                   <list>
1 Beijing      <list [3]>         Beijing, China          <named list [4]>
2 Buenos Aires <list [3]>         Buenos Aires, Argentina <named list [4]>
3 New Delhi    <list [3]>         New Delhi, Delhi, India <named list [4]>
4 New York     <list [3]>         New York, NY, USA       <named list [4]>
5 Paris        <list [4]>         Paris, France           <named list [4]>
# ... with 4 more variables: place_id <chr>, types <list>, partial_match <lgl>, status <chr>
```

[1] Example from tidyr documentation: https://tidyr.tidyverse.org/articles/rectangle.html

Unnesting Google maps data

```r
city_df %>%
  unnest_wider(json) %>%
  unnest_longer(results) %>%
  unnest_wider(results) %>%
  unnest_wider(geometry) %>%
  unnest_wider(location) %>%
  select(city, lat, lng)
```

```
# A tibble: 5 x 3
  city           lat    lng
  <chr>        <dbl>  <dbl>
1 Beijing       39.9 116.  
2 Buenos Aires -34.6 -58.4 
3 New Delhi     28.6  77.2 
4 New York      40.7 -74.0 
5 Paris         48.9   2.35
```

[1] Example from tidyr documentation: https://tidyr.tidyverse.org/articles/rectangle.html

Selecting Google maps data with hoist()

```
city_df %>%
  hoist(json,
        lat = list("results", 1, "geometry", "location", "lat"),
        lng = list("results", 1, "geometry", "location", "lng"))
```

```
# A tibble: 5 x 4
  city           lat    lng json
  <chr>        <dbl>  <dbl> <list>
1 Beijing       39.9 116.   <named list [2]>
2 Buenos Aires -34.6 -58.4  <named list [2]>
3 New Delhi     28.6  77.2  <named list [2]>
4 New York      40.7 -74.0  <named list [2]>
5 Paris         48.9   2.35 <named list [2]>
```

[1] Example from tidyr documentation: https://tidyr.tidyverse.org/articles/rectangle.html

Let's practice!

RESHAPING DATA WITH TIDYR

Nesting data for modeling

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

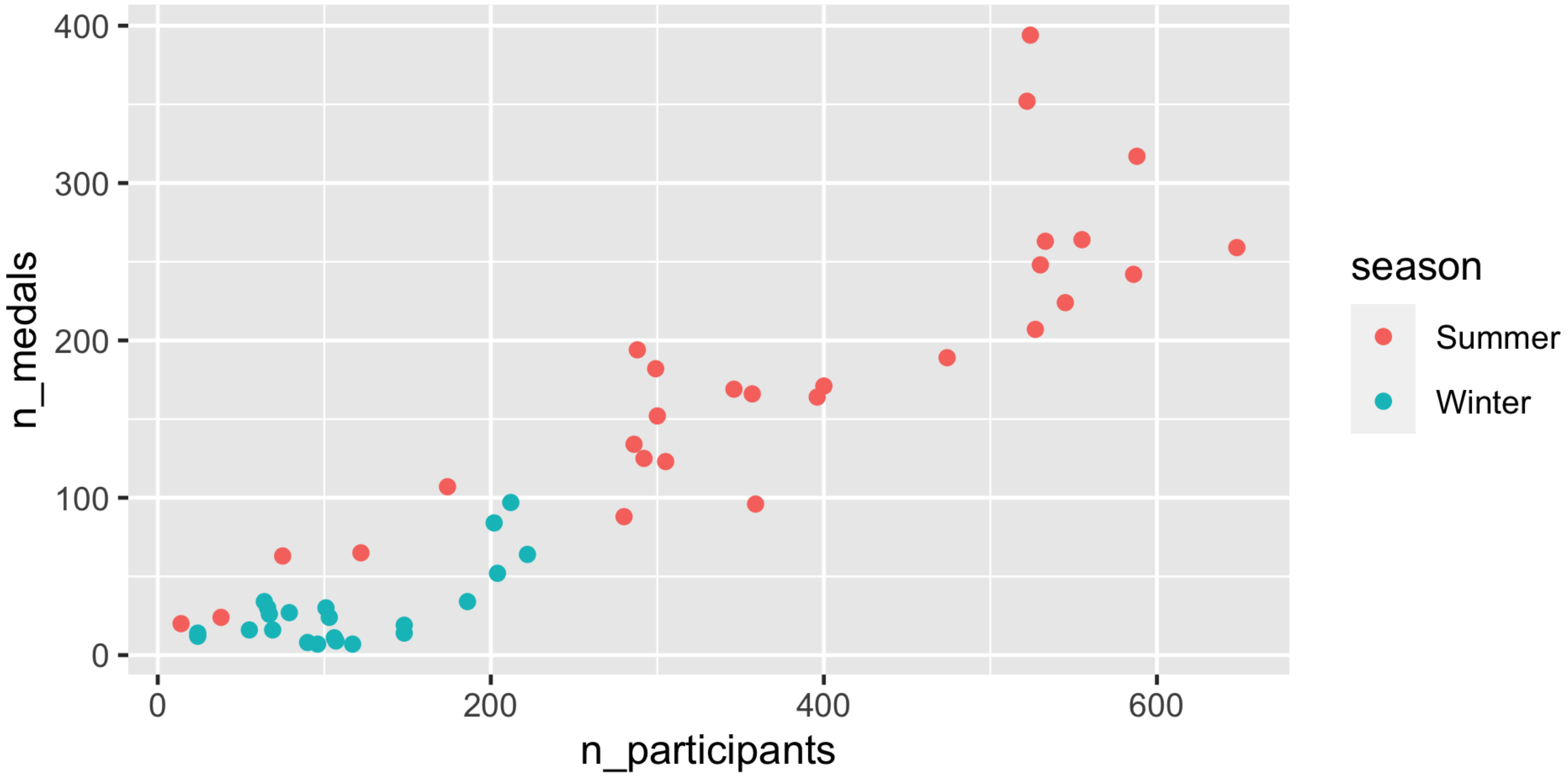
USA Olympic performance

```
usa_olympic_df
```

```
# A tibble: 50 x 5
   country  year season n_participants n_medals
   <chr>   <dbl> <chr>           <int>    <int>
 1 USA      1896 Summer             14       20
 2 USA      1900 Summer             75       63
 3 USA      1904 Summer            524      394
 4 USA      1906 Summer             38       24
 5 USA      1908 Summer            122       65
 6 USA      1912 Summer            174      107
# ... with 44 more rows
```

USA Olympic performance

```
usa_olympic_df %>%
  ggplot(aes(x = n_participants, y = n_medals, color = season))+
  geom_point()
```



Modeling the pattern

```
model <- lm(n_medals ~ n_participants + 0, data = usa_olympics_df)


model
```

```
Call:
lm(formula = n_medals ~ n_participants + 0, data = usa_olympics_df)


Coefficients:
n_participants
         0.463
```

Untidy model statistics

```r
summary(model)
```

```
Call:
lm(formula = n_medals ~ n_participants + 0, data = usa_olympics_df)
Residuals:
    Min      1Q  Median      3Q     Max
-70.222 -36.175  -9.554   6.871 151.380
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
n_participants  0.46302    0.01791   25.86   <2e-16 ***
--
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 40.17 on 49 degrees of freedom
Multiple R-squared:  0.9317,    Adjusted R-squared:  0.9303
F-statistic: 668.5 on 1 and 49 DF,  p-value: < 2.2e-16
```

The broom package

```
broom::glance(model)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC   BIC deviance df.residual
      <dbl>         <dbl> <dbl>     <dbl>     <dbl> <int>  <dbl> <dbl> <dbl>    <dbl>       <int>
1     0.932         0.930  40.2      668. 3.25e-30     1  -255.  514.  518.   79079.          49
```

```
broom::tidy(model)
```

```
# A tibble: 1 x 5
  term           estimate std.error statistic  p.value
  <chr>             <dbl>     <dbl>     <dbl>    <dbl>
1 n_participants    0.463    0.0179      25.9 3.25e-30
```

broom + dplyr + tidyr

```
usa_olympics_df %>%
  group_by(country) %>%
  nest()
```

```
# A tibble: 1 x 2
# Groups:   country [1]
  country data
  <chr>   <list>
1 USA     <tibble [50 × 4]>
```

Nested tibble & purrr::map()

```
usa_olympics_df %>%
  group_by(country) %>%
  nest() %>%
  mutate(fit = purrr::map(data, function(df) lm(n_medals ~ n_participants + 0, data = df)))
```

```
 # A tibble: 1 x 3
# Groups:   country [1]
  country data              fit
  <chr>   <list>            <list>
1 USA     <tibble [50 × 4]> <lm>
```

Working with nested tibbles

```r
usa_olympics_df %>%
  group_by(country) %>%
  nest() %>%
  mutate(fit = purrr::map(data, function(df) lm(n_medals ~ n_participants + 0, data = df)),
         glanced = purrr::map(fit, broom::glance))
```

```
 # A tibble: 1 x 4
# Groups:   country [1]
  country data                fit    glanced
  <chr>   <list>              <list> <list>
1 USA     <tibble [50 × 4]> <lm>   <tibble [1 × 11]>
```

Unnesting model results

```
usa_olympics_df %>%
  group_by(country) %>%
  nest() %>%
  mutate(fit = purrr::map(data, function(df) lm(n_medals ~ n_participants + 0, data = df)),
         glanced = purrr::map(fit, broom::glance)) %>%
  unnest(glanced)
```

```
# A tibble: 1 x 14
# Groups:   country [1]
  country data              fit    r.squared adj.r.squared sigma statistic  p.value    df
  <chr>   <list>            <list>     <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>
1 USA     <tibble [50 × 4]> <lm>       0.932         0.930  40.2      668. 3.25e-30     1
# with 5 more variables: logLik <dbl>, AIC <dbl>, BIC <dbl> deviance <dbl>, df.residual <int>
```

Unnesting model results

```r
usa_olympics_df %>%
  group_by(country) %>%
  nest() %>%
  mutate(fit = purrr::map(data, function(df) lm(n_medals ~ n_participants + 0, data = df)),
         tidied = purrr::map(fit, broom::tidy)) %>%
  unnest(tidied)
```

```
# A tibble: 1 x 8
# Groups:   country [1]
  country data              fit    term           estimate std.error statistic  p.value
  <chr>   <list>            <list> <chr>             <dbl>     <dbl>     <dbl>    <dbl>
1 USA     <tibble [50 × 4]> <lm>   n_participants    0.463    0.0179      25.9 3.25e-30
```

Multiple model pipeline

```
usa_olympics_df %>%
  group_by(country, season) %>%
  nest() %>%
  mutate(fit = purrr::map(data, function(df) lm(n_medals ~ n_participants + 0, data = df)),
         tidied = purrr::map(fit, broom::tidy)) %>%
  unnest(tidied)
```

```
# A tibble: 2 x 9
# Groups:   country, season [2]
  country season data            fit    term           estimate std.error statistic  p.value
  <chr>   <chr>  <list>          <list> <chr>             <dbl>     <dbl>     <dbl>    <dbl>
1 USA     Summer <tibble [28×3]> <lm>   n_participants    0.478    0.0213      22.5  5.29e-19
2 USA     Winter <tibble [22×3]> <lm>   n_participants    0.263    0.0292       9.00 1.18e- 8
```

Let's practice!

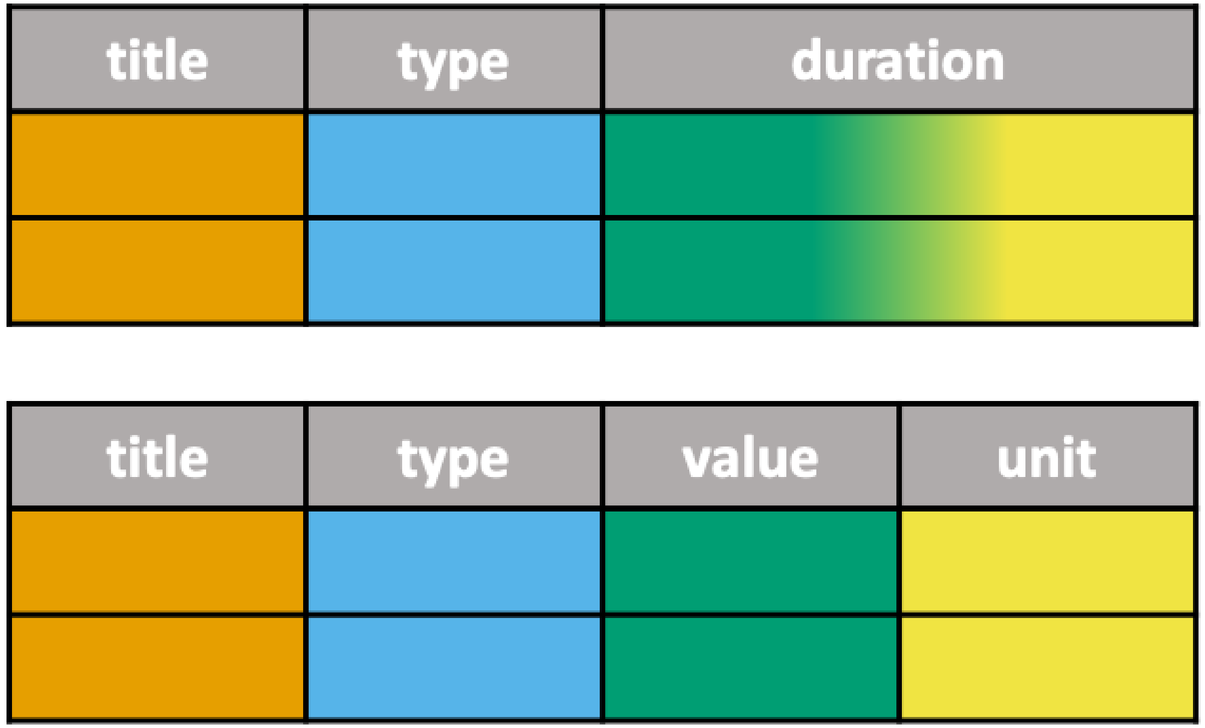
Congratulations!

RESHAPING DATA WITH TIDYR

Jeroen Boeye
Head of Machine Learning, Faktion

Separating messy string columns

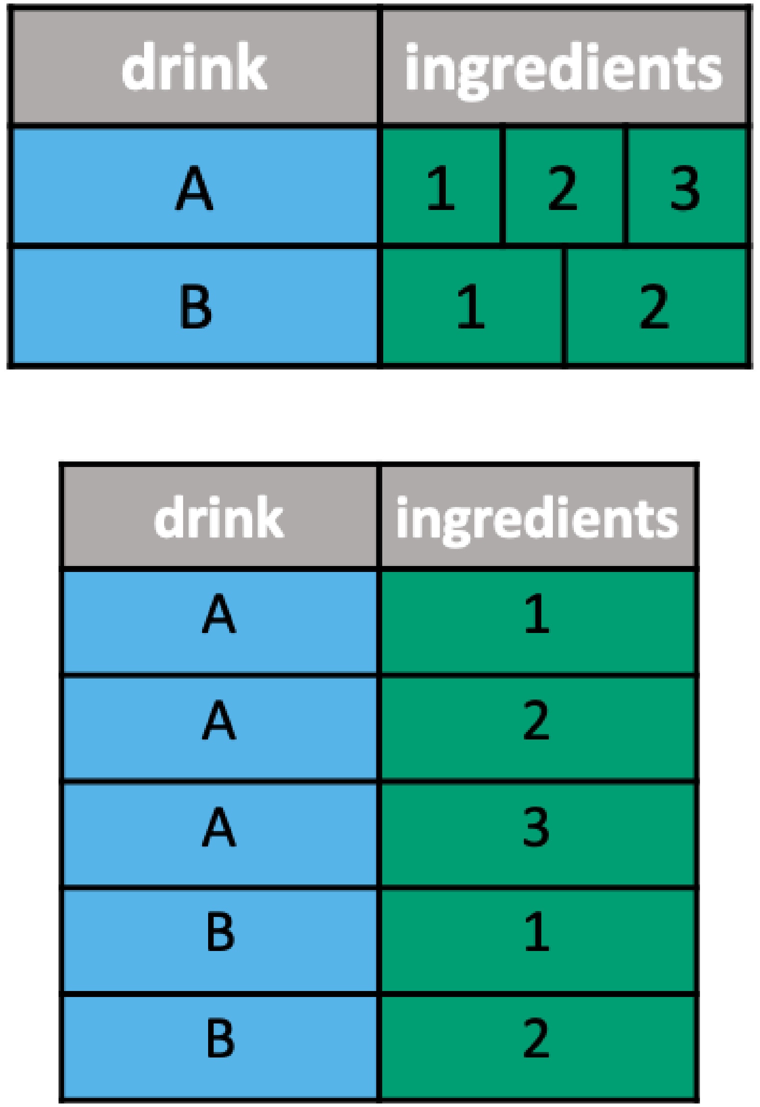
`separate()`



title	type	duration
🟧	🟦	🟩🟨
🟧	🟦	🟩🟨

title	type	value	unit
🟧	🟦	🟩	🟨
🟧	🟦	🟩	🟨

`separate_rows()`

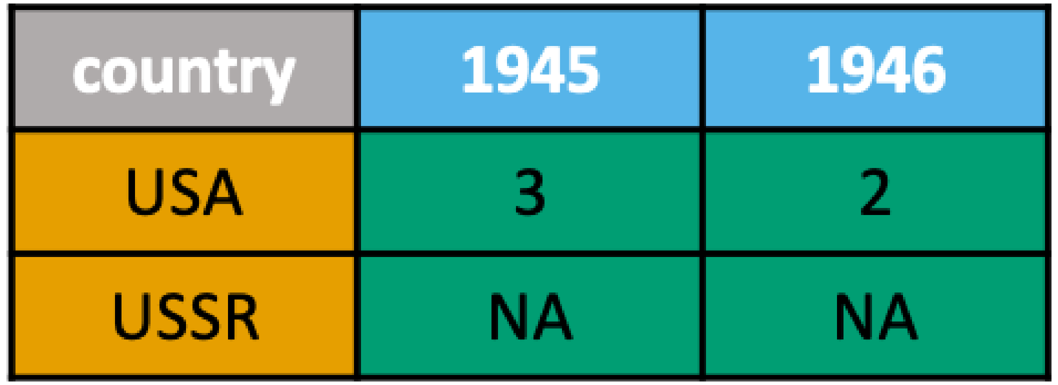


drink	ingredients
A	1 2 3
B	1 2

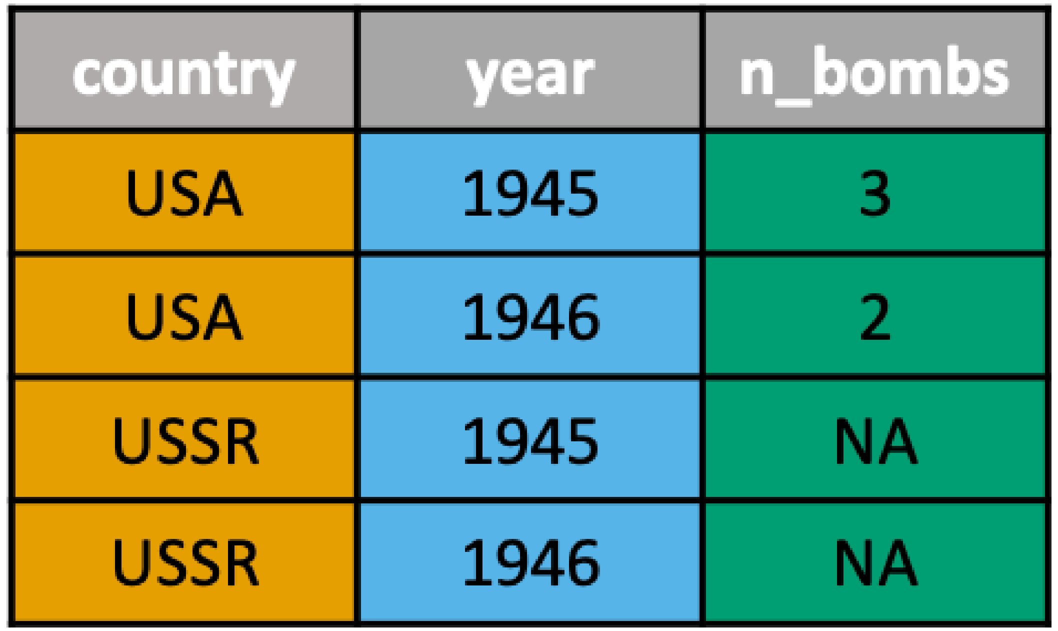
drink	ingredients
A	1
A	2
A	3
B	1
B	2

Pivoting data

`pivot_longer()`

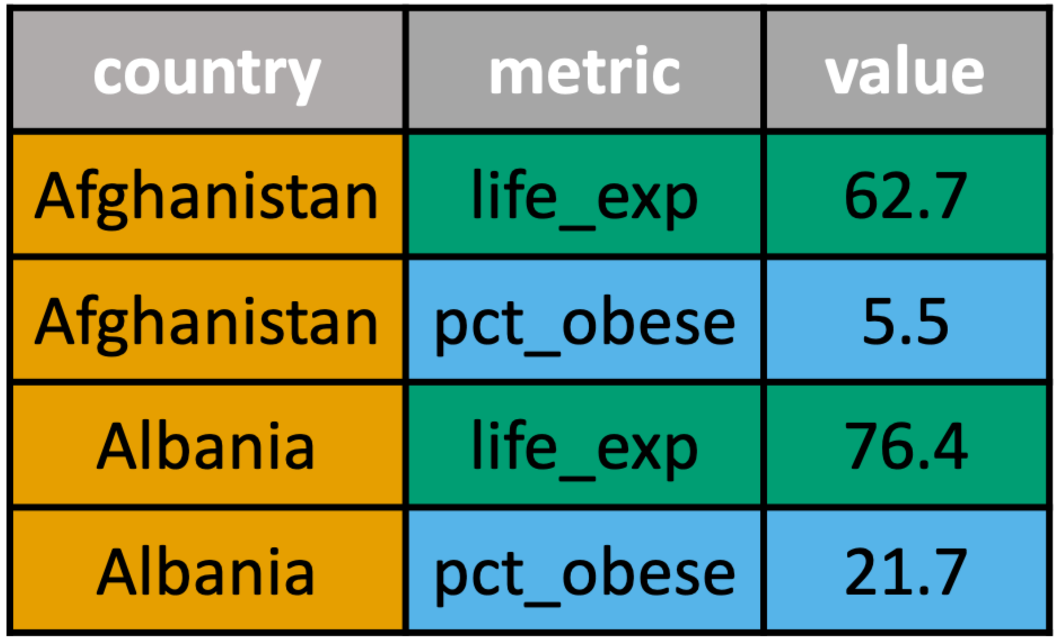


country	1945	1946
USA	3	2
USSR	NA	NA

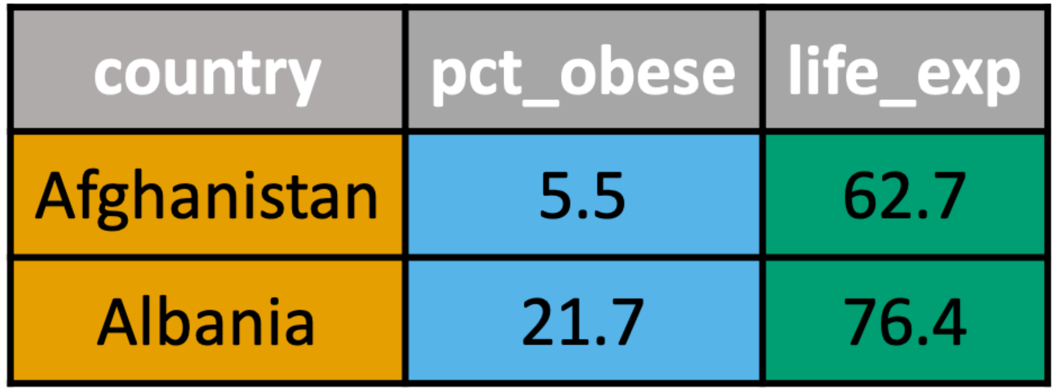


country	year	n_bombs
USA	1945	3
USA	1946	2
USSR	1945	NA
USSR	1946	NA

`pivot_wider()`



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

Expanding data

`complete()`

year	artist	n_albums
1977	Beatles	2
1977	Rolling Stones	1
1979	Beatles	1

year	artist	n_albums
1977	Beatles	2
1977	Rolling Stones	1
1978	Beatles	0
1978	Rolling Stones	0
1979	Beatles	1
1979	Rolling Stones	0

Unnesting data

```
tibble(character = star_wars_list) %>%
  unnest_wider(character) %>%
  unnest_longer(films)
```

```
# A tibble: 45 x 2
   name        films
   <chr>       <chr>
 1 Chewbacca   Revenge of the Sith
 2 Chewbacca   Return of the Jedi
 3 Chewbacca   The Empire Strikes Back
 4 Chewbacca   A New Hope
 5 Chewbacca   The Force Awakens
 6 Darth Vader Revenge of the Sith
 7 Darth Vader Return of the Jedi
 8 Darth Vader The Empire Strikes Back
# ... with 37 more rows
```

The end

RESHAPING DATA WITH TIDYR