

Creating unique combinations of vectors

RESHAPING DATA WITH TIDYR



Jeroen Boeye

Head of Machine Learning, Faktion

The early atomic era: 1945 - 1954

nuke_df

```
# A tibble: 13 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 Russian Federation 1949     1
5 Russian Federation 1951     2
6 United States 1951    16
# ... with 7 more rows
```

The `expand_grid()` function

```
full_df <- expand_grid(  
  year = 1945:1954,  
  country = c(  
    "Russian Federation",  
    "United Kingdom",  
    "United States")  
)
```

full_df

```
# A tibble: 30 x 2  
  year country  
  <int> <chr>  
1  1945 Russian Federation  
2  1945 United Kingdom  
3  1945 United States  
4  1946 Russian Federation  
5  1946 United Kingdom  
6  1946 United States  
7  1947 Russian Federation  
8  1947 United Kingdom  
# ... with 22 more rows
```

right_join() with a tibble of unique combinations

```
nuke_df %>%  
  right_join(  
    full_df,  
    by = c("country", "year")  
  ) %>%  
  arrange(year)
```

```
# A tibble: 30 x 3  
  country      year n_bombs  
  <chr>      <int>   <int>  
1 United States  1945     3  
2 Russian Federation  1945    NA  
3 United Kingdom  1945    NA  
4 United States  1946     2  
5 Russian Federation  1946    NA  
6 United Kingdom  1946    NA  
7 Russian Federation  1947    NA  
8 United Kingdom  1947    NA  
# ... with 22 more rows
```

right_join() with a tibble of unique combinations

```
nuke_df %>%  
  right_join(  
    full_df,  
    by = c("country", "year")  
  ) %>%  
  arrange(year) %>%  
  replace_na(list(n_bombs = 0L))
```

```
# A tibble: 30 x 3  
  country      year n_bombs  
  <chr>      <int>   <int>  
1 United States  1945     3  
2 Russian Federation 1945     0  
3 United Kingdom  1945     0  
4 United States  1946     2  
5 Russian Federation 1946     0  
6 United Kingdom  1946     0  
7 Russian Federation 1947     0  
8 United Kingdom  1947     0  
# ... with 22 more rows
```

anti_join() to select missing observations

```
full_df %>%  
  anti_join(  
    nuke_df,  
    by = c("country", "year")  
  )
```

```
# A tibble: 17 x 2  
  year country  
  <int> <chr>  
1  1945 Russian Federation  
2  1945 United Kingdom  
3  1946 Russian Federation  
4  1946 United Kingdom  
5  1947 Russian Federation  
6  1947 United Kingdom  
7  1947 United States  
8  1948 Russian Federation  
# ... with 9 more rows
```

Let's practice!

RESHAPING DATA WITH TIDYR

Completing data with all value combinations

RESHAPING DATA WITH TIDYR



Jeroen Boeye

Head of Machine Learning, Faktion

Rolling Stones and Beatles

```
album_df
```

```
# A tibble: 3 x 3
  year artist      n_albums
  <int> <chr>         <int>
1  1977 Beatles             2
2  1977 Rolling Stones      1
3  1979 Beatles             1
```

Initial and target situation

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
1979	Beatles	1
1979	Rolling Stones	0

Initial and target situation

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

The complete() function

```
album_df %>%  
  complete(year, artist)
```

```
# A tibble: 4 x 3  
   year artist      n_albums  
  <int> <chr>      <int>  
1  1977 Beatles          2  
2  1977 Rolling Stones    1  
3  1979 Beatles          1  
4  1979 Rolling Stones    NA
```

The complete() function: overwriting NA values

```
album_df %>%  
  complete(year, artist, fill = list(n_albums = 0L))
```

```
# A tibble: 4 x 3  
  year artist      n_albums  
  <int> <chr>      <int>  
1  1977 Beatles          2  
2  1977 Rolling Stones    1  
3  1979 Beatles          1  
4  1979 Rolling Stones    0
```

The complete() function: adding unseen values

```
album_df %>%  
  complete(  
    year,  
    artist = c(  
      "Beatles",  
      "Rolling Stones",  
      "ABBA"),  
    fill = list(n_albums = 0L)  
  )
```

```
# A tibble: 6 x 3  
  year artist      n_albums  
  <int> <chr>      <int>  
1  1977 ABBA          0  
2  1977 Beatles       2  
3  1977 Rolling Stones 1  
4  1979 ABBA          0  
5  1979 Beatles       1  
6  1979 Rolling Stones 0
```

The complete() function: adding unseen values

```
album_df %>%  
  complete(  
    year = 1977:1979,  
    artist,  
    fill = list(n_albums = 0L)  
  )
```

```
# A tibble: 6 x 3  
  year artist      n_albums  
  <int> <chr>      <int>  
1  1977 Beatles          2  
2  1977 Rolling Stones    1  
3  1978 Beatles           0  
4  1978 Rolling Stones    0  
5  1979 Beatles           1  
6  1979 Rolling Stones    0
```

Generating a sequence with `full_seq()`

```
full_seq(c(1977, 1979), period = 1)
```

```
1977 1978 1979
```

```
full_seq(c(1977, 1979, 1980, 1980, 1980), period = 1)
```

```
1977 1978 1979 1980
```

```
full_seq(album_df$year, period = 1)
```

```
1977 1978 1979
```


Using `full_seq()` inside `complete()`

```
album_df %>%  
  complete(  
    year = full_seq(year, period = 1),  
    artist,  
    fill = list(n_albums = 0L)  
  )
```

```
# A tibble: 6 x 3  
  year artist      n_albums  
  <dbl> <chr>      <int>  
1  1977 Beatles          2  
2  1977 Rolling Stones    1  
3  1978 Beatles          0  
4  1978 Rolling Stones    0  
5  1979 Beatles          1  
6  1979 Rolling Stones    0
```

Generating a date sequence with `full_seq()`

```
full_seq(c(as.Date("2000-01-01"), as.Date("2000-01-10")), period = 1)
```

```
[1] "2000-01-01" "2000-01-02" "2000-01-03" "2000-01-04" "2000-01-05"  
[6] "2000-01-06" "2000-01-07" "2000-01-08" "2000-01-09" "2000-01-10"
```

Let's practice!

RESHAPING DATA WITH TIDYR

Advanced completions

RESHAPING DATA WITH TIDYR



Jeroen Boeye

Head of Machine Learning, Faktion

Nesting connected variables

```
nuke_df
```

```
# A tibble: 5 x 4
  continent      country n_bombs decade
  <chr>         <chr>    <int>   <int>
1 North America USA         8    1940
2 Europe        USSR         1    1940
3 North America USA       188    1950
4 Europe        USSR        82    1950
5 Europe        UK         21    1950
```

Nesting connected variables

```
nuke_df %>%  
  complete(  
    continent,  
    country,  
    decade,  
    fill = list(n_bombs = 0L)  
  )
```

```
# A tibble: 12 x 4  
  continent    country decade n_bombs  
  <chr>      <chr>    <int>   <int>  
1 Europe      UK       1940     0  
2 Europe      UK       1950    21  
3 Europe      USA      1940     0  
4 Europe      USA      1950     0  
5 Europe      USSR     1940     1  
6 Europe      USSR     1950    82  
7 North America UK       1940     0  
8 North America UK       1950     0  
# ... with 4 more rows
```

The nesting() function

```
nuke_df %>%  
  complete(  
    nesting(continent, country),  
    decade,  
    fill = list(n_bombs = 0L)  
  )
```

```
# A tibble: 6 x 4  
  continent      country decade n_bombs  
  <chr>         <chr>    <int>   <int>  
1 Europe        UK      1940     0  
2 Europe        UK      1950    21  
3 Europe        USSR     1940     1  
4 Europe        USSR     1950    82  
5 North America USA      1940     8  
6 North America USA      1950   188
```

Counting tropical storms

```
storm_df
```

```
# A tibble: 35 x 3
  name      start      end
  <chr>    <date>    <date>
1 ANDREA  2013-06-05 2013-06-08
2 ARTHUR  2014-06-28 2014-07-09
3 ANA     2015-05-06 2015-05-12
4 BARRY   2013-06-16 2013-06-21
5 TWO     2014-07-19 2014-07-23
6 BILL    2015-06-16 2015-06-21
# ... with 29 more rows
```


Counting tropical storms: pivot to long format

```
storm_df %>%  
  pivot_longer(  
    -name,  
    names_to = "status",  
    values_to = "date"  
  )
```

```
# A tibble: 70 x 3  
  name    status date  
  <chr>   <chr>   <date>  
1 ANDREA start 2013-06-05  
2 ANDREA end 2013-06-08  
3 ARTHUR start 2014-06-28  
4 ARTHUR end 2014-07-09  
5 ANA start 2015-05-06  
6 ANA end 2015-05-12  
7 BARRY start 2013-06-16  
8 BARRY end 2013-06-21  
9 TWO start 2014-07-19  
10 TWO end 2014-07-23  
# ... with 60 more rows
```

Counting tropical storms: grouped completion

```
storm_df %>%  
  pivot_longer(  
    -name,  
    names_to = "status",  
    values_to = "date"  
  ) %>%  
  group_by(name) %>%  
  complete(date = full_seq(date, 1)) %>%  
  ungroup()
```

```
# A tibble: 263 x 3  
  name    date    status  
  <chr>  <date>    <chr>  
1 ANA    2015-05-06 start  
2 ANA    2015-05-07 NA  
3 ANA    2015-05-08 NA  
4 ANA    2015-05-09 NA  
5 ANA    2015-05-10 NA  
6 ANA    2015-05-11 NA  
7 ANA    2015-05-12 end  
8 ANDREA 2013-06-05 start  
9 ANDREA 2013-06-06 NA  
10 ANDREA 2013-06-07 NA  
# ... with 253 more rows
```

Counting tropical storms: the actual count

```
storm_df %>%  
  pivot_longer(  
    -name,  
    names_to = "status",  
    values_to = "date"  
  ) %>%  
  group_by(name) %>%  
  complete(date = full_seq(date, 1)) %>%  
  ungroup() %>%  
  count(date, name = "n_storms")
```

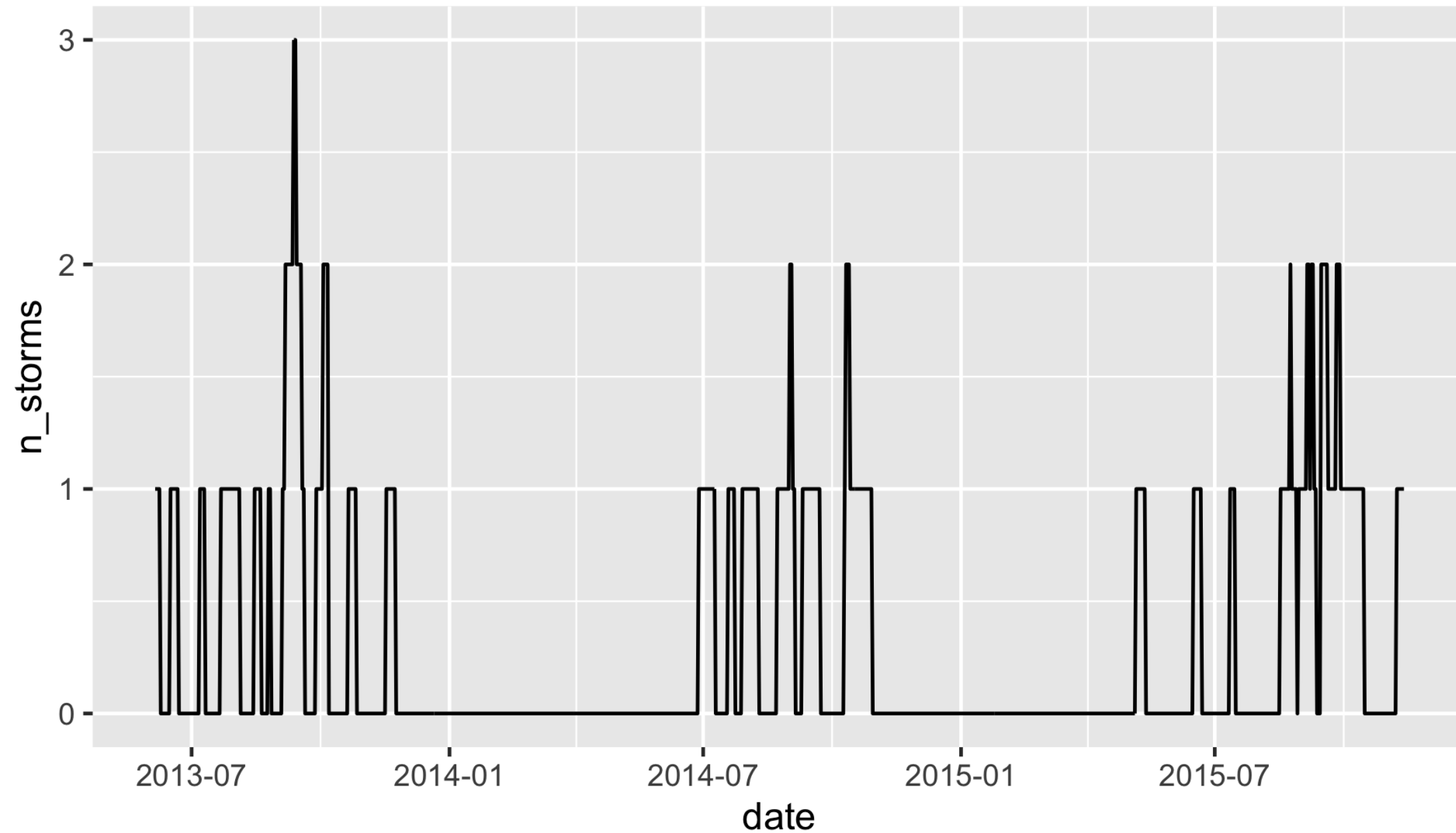
```
# A tibble: 227 x 2  
  date      n_storms  
  <date>      <int>  
1 2013-06-05         1  
2 2013-06-06         1  
3 2013-06-07         1  
4 2013-06-08         1  
5 2013-06-16         1  
6 2013-06-17         1  
7 2013-06-18         1  
8 2013-06-19         1  
9 2013-06-20         1  
10 2013-06-21         1  
# ... with 217 more rows
```

Counting tropical storms: adding zero counts

```
storm_df %>%
  pivot_longer(
    -name,
    names_to = "status",
    values_to = "date"
  ) %>%
  group_by(name) %>%
  complete(date = full_seq(date, 1)) %>%
  ungroup() %>%
  count(date, name = "n_storms") %>%
  complete(
    date = full_seq(date, 1),
    fill = list(n_storms = 0L)
  )
```

```
# A tibble: 892 x 2
   date      n_storms
<date>      <int>
1 2013-06-05         1
2 2013-06-06         1
3 2013-06-07         1
4 2013-06-08         1
5 2013-06-09         0
6 2013-06-10         0
7 2013-06-11         0
8 2013-06-12         0
9 2013-06-13         0
10 2013-06-14         0
# ... with 882 more rows
```

Counting tropical storms: visualizing the result



Timestamp completions

```
sensor_df
```

```
# A tibble: 3 x 2
  time                temperature
  <dtm>                <int>
1 2020-01-01 11:00:00         25
2 2020-01-01 11:40:00         26
3 2020-01-01 12:20:00         25
```

Timestamp completions

```
sensor_df %>%  
  complete(time = seq(from = min(time), to = max(time), by = "20 min"))
```

```
# A tibble: 5 x 2  
  time                temperature  
  <dtm>              <int>  
1 2020-01-01 11:00:00          25  
2 2020-01-01 11:20:00         NA  
3 2020-01-01 11:40:00          26  
4 2020-01-01 12:00:00         NA  
5 2020-01-01 12:20:00          25
```

Timestamp completions

```
sensor_df %>%  
  complete(time = seq(from = min(time), to = max(time), by = "20 min")) %>%  
  fill(temperature)
```

```
# A tibble: 5 x 2  
  time                temperature  
  <dtm>                <int>  
1 2020-01-01 11:00:00          25  
2 2020-01-01 11:20:00          25  
3 2020-01-01 11:40:00          26  
4 2020-01-01 12:00:00          26  
5 2020-01-01 12:20:00          25
```


Let's practice!

RESHAPING DATA WITH TIDYR