Tidying data

Filippo Chiarello, Ph.D.

We...

have *data organised in an unideal way for our analysis*

want *to reorganise the data to carry on with our analysis*

Data: Sales

We have...

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr> 
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>  
```

We want...

```
## # A tibble: 6 × 3
##   customer_id item_no item        
##         <dbl> <chr>   <chr>       
## 1           1 item_1  bread       
## 2           1 item_2  milk        
## 3           1 item_3  banana      
## 4           2 item_1  milk        
## 5           2 item_2  toilet paper
## 6           2 item_3  <NA>        
```

A grammar of data tidying

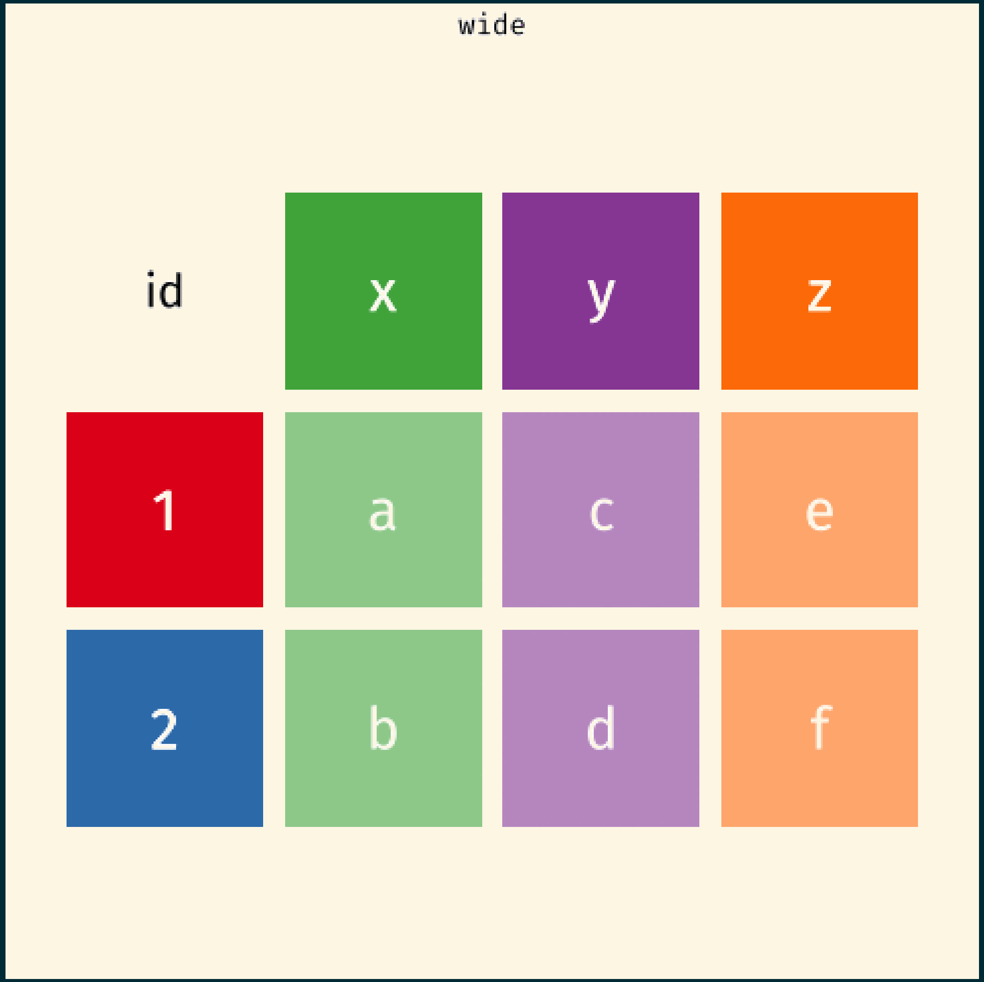
The goal of tidyr is to help you tidy your data via

- pivoting for going between wide and long data
- splitting and combining character columns
- nesting and unnesting columns
- clarifying how NAs should be treated

Pivoting data

Not this...

but this!


wide

id	x	y	z
1	a	c	e
2	b	d	f

Wider vs. longer

wider

more columns

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr>
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>
```

longer

more rows

```
## # A tibble: 6 × 3
##   customer_id item_no item
##         <dbl> <chr>   <chr>
## 1           1 item_1  bread
## 2           1 item_2  milk
## 3           1 item_3  banana
## 4           2 item_1  milk
## 5           2 item_2  toilet paper
## 6           2 item_3  <NA>
```

`pivot_longer()`

- `data` (as usual)

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

`pivot_longer()`

- `data` (as usual)
- `cols`: columns to pivot into longer format

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

`pivot_longer()`

- `data` (as usual)
- `cols`: columns to pivot into longer format
- `names_to`: name of the column where column names of pivoted variables go (character string)

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

`pivot_longer()`

- `data` (as usual)
- `cols`: columns to pivot into longer format
- `names_to`: name of the column where column names of pivoted variables go (character string)
- `values_to`: name of the column where data in pivoted variables go (character string)

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

Customers → purchases

```r
purchases <- customers %>%
  pivot_longer(
    cols = item_1:item_3,  # variables item_1 to item_3
    names_to = "item_no",  # column names -> new column called item_no
    values_to = "item"     # values in columns -> new column called item
    )

purchases
```

```
## # A tibble: 6 × 3
##   customer_id item_no item
##         <dbl> <chr>   <chr>
## 1           1 item_1  bread
## 2           1 item_2  milk
## 3           1 item_3  banana
## 4           2 item_1  milk
## 5           2 item_2  toilet paper
## 6           2 item_3  <NA>
```

Why pivot?

Most likely, because the next step of your analysis needs it

```r
prices
```

```
## # A tibble: 5 × 2
##   item         price
##   <chr>        <dbl>
## 1 avocado       0.5
## 2 banana        0.15
## 3 bread         1
## 4 milk          0.8
## 5 toilet paper  3
```

```r
purchases %>%
  left_join(prices)
```

```
## # A tibble: 6 × 4
##   customer_id item_no item         price
##         <dbl> <chr>   <chr>        <dbl>
## 1           1 item_1  bread         1
## 2           1 item_2  milk          0.8
## 3           1 item_3  banana        0.15
## 4           2 item_1  milk          0.8
## 5           2 item_2  toilet paper  3
## 6           2 item_3  <NA>         NA
```

Purchases → customers

- `data` (as usual)
- `names_from`: which column in the long format contains the what should be column names in the wide format
- `values_from`: which column in the long format contains the what should be values in the new columns in the wide format

```
purchases %>%
  pivot_wider(
    names_from = item_no,
    values_from = item
  )
```

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr>
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>
```