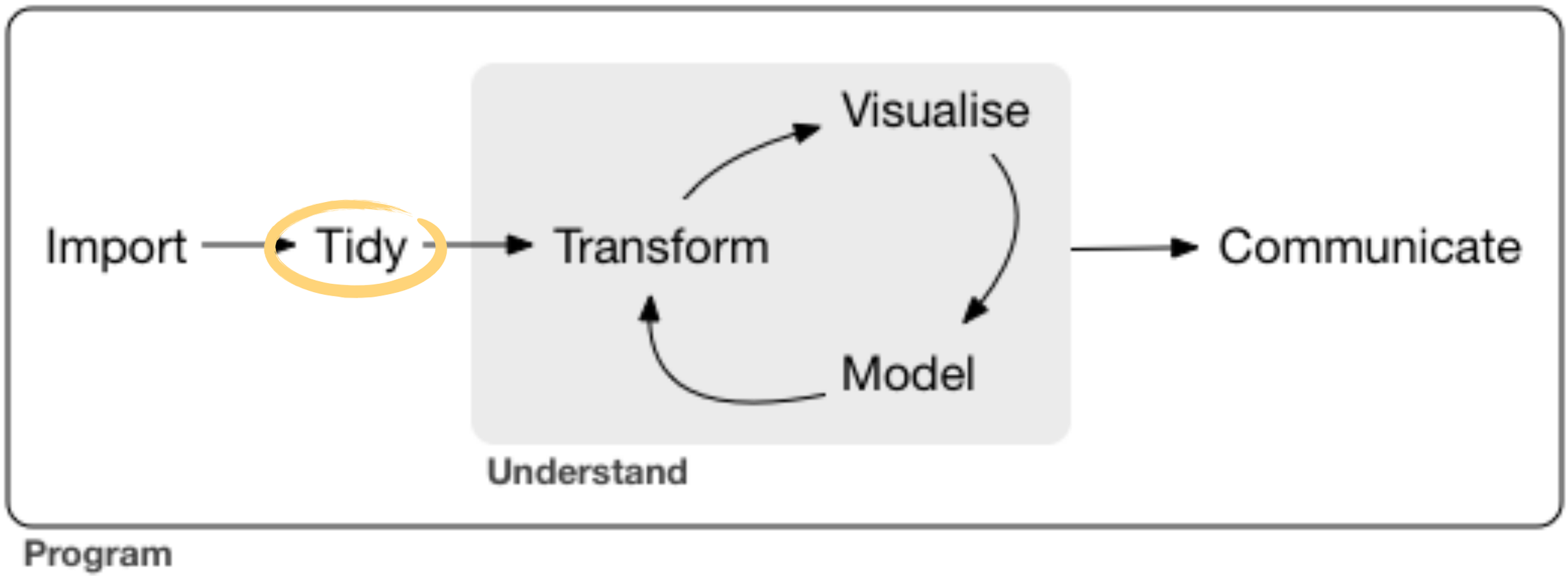
Wide and long data

Jonathan de Bruin and Barbara Vreede

Data science workflow



Tidy data ensures that further processing can be done efficiently, and reproducibly.

Tidy data is easy to manipulate, model, and visualize.

Tidy data

- Each **variable** is a column and contains **values**
- Each **observation** is a row
- Each type of **observational unit** forms a table

values in column names

Patient	BP	Med_A	BP_after_A	Med_B	BP_after_B
122030	120-82	300	119_85	NA	NA
122021	131-91	NA	NA	85	125_90
124500	118-86	300	119_70	NA	NA
126098	99-67			100	110_71

multiple data points in a single cell

Tidy data

- Each **variable** is a column and contains **values**
- Each **observation** is a row
- Each type of **observational unit** forms a table

Patient	Sys	Dia	Treatment	Sys_after	Dia_after
122030	120	82	A	119	85
122021	131	91	B	125	90
124500	118	86	A	119	70
126098	99	67	B	110	71

Example of wide data

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1            5.1         3.5          1.4         0.2  setosa
2            4.9         3.0          1.4         0.2  setosa
3            4.7         3.2          1.3         0.2  setosa
4            4.6         3.1          1.5         0.2  setosa
5            5.0         3.6          1.4         0.2  setosa
6            5.4         3.9          1.7         0.4  setosa
7            4.6         3.4          1.4         0.3  setosa
8            5.0         3.4          1.5         0.2  setosa
9            4.4         2.9          1.4         0.2  setosa
10           4.9         3.1          1.5         0.1  setosa
11           5.4         3.7          1.5         0.2  setosa
12           4.8         3.4          1.6         0.2  setosa
13           4.8         3.0          1.4         0.1  setosa
14           4.3         3.0          1.1         0.1  setosa
15           5.8         4.0          1.2         0.2  setosa
16           5.7         4.4          1.5         0.4  setosa
17           5.4         3.9          1.3         0.4  setosa
18           5.1         3.5          1.4         0.3  setosa
19           5.7         3.8          1.7         0.3  setosa
20           5.1         3.8          1.5         0.3  setosa
21           5.4         3.4          1.7         0.2  setosa
22           5.1         3.7          1.5         0.4  setosa
23           4.6         3.6          1.0         0.2  setosa
```

Example of long data

```
> PlantGrowth
   weight group
1    4.17  ctrl
2    5.58  ctrl
3    5.18  ctrl
…
8    4.53  ctrl
9    5.33  ctrl
10   5.14  ctrl
11   4.81  trt1
12   4.17  trt1
13   4.41  trt1
…
18   4.89  trt1
19   4.32  trt1
20   4.69  trt1
21   6.31  trt2
22   5.12  trt2
23   5.54  trt2
….
28   6.15  trt2
29   5.80  trt2
30   5.26  trt2
```

Functions in R

Functions to transform long and wide data.

Functions in R regarding long and wide data

func	package	To long form	To wide form
stack/unstack	utils	stack	unstack
reshape	stats	reshape(direction="long", ...)	reshape(direction="wide", ...)
melt/dcast	reshape2	melt	dcast
gather/spread	tidyr	gather	spread

Functions in R regarding long and wide data

func	package	To long form	To wide form
stack/unstack	utils	stack	unstack
reshape	stats	reshape(direction="long", ...)	reshape(direction="wide", ...)
melt/dcast	reshape2	melt	dcast
gather/spread	tidyr	gather	spread

Very limited functionality

Functions in R regarding long and wide data

func	package	To long form	To wide form
stack/unstack	utils	stack	unstack
reshape	stats	reshape(direction="long", ...)	reshape(direction="wide", ...)
melt/dcast	reshape2	melt	dcast
gather/spread	tidyr	gather	spread

```
reshape(data, varying = NULL, v.names = NULL, timevar = "time",
        idvar = "id", ids = 1:NROW(data),
        times = seq_along(varying[[1]]),
        drop = NULL, direction, new.row.names = NULL,
        sep = ".",
        split = if (sep == "") {
            list(regexp = "[A-Za-z][0-9]", include = TRUE)
        } else {
            list(regexp = sep, include = FALSE, fixed = TRUE)}
        )
```

Good luck figuring out the arguments you need though.
-Hadley Wickham

Functions in R regarding long and wide data

func	package	To long form	To wide form
stack/unstack	utils	stack	unstack
reshape	stats	reshape(direction="long", ...)	reshape(direction="wide", ...)
melt/dcast	reshape2	melt	dcast
gather/spread	tidyr	gather	spread

Deprecated!

Functions in R regarding long and wide data

func	package	To long form	To wide form
stack/unstack	utils	stack	unstack
reshape	stats	reshape(direction="long", ...)	reshape(direction="wide", ...)
melt/dcast	reshape2	melt	dcast
gather/spread	tidyr	gather	spread

Very intuitive and flexible!

Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

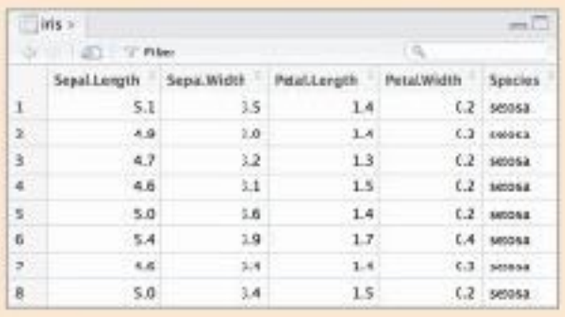
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).



dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` *is the same as* `f(x, y)`

`y %>% f(x, ., z)` *is the same as* `f(x, y, z )`

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - A foundation for wrangling in R

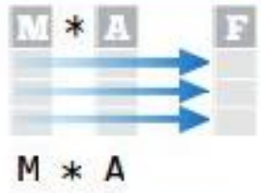
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

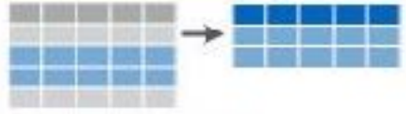
dplyr::**arrange(mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)



dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)

Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)

Select columns whose name ends with a character string.

select(iris, **everything()**)

Select every column.

select(iris, **matches("..")**)

Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)

Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)

Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)

Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)

Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)

Select all columns except Species.

Long and wide data

Explained in terms of keys and values.

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

keys

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

values

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

key-values

Example:

meas_A, 1.2

meas_B, 1.4

meas_C, 1.3

meas_A, 3.5

...

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

other columns

Wide and long patient dataset

```
> data_patients
```

	patient	gender	meas_A	meas_B	meas_C
1	1	M	1.2	1.4	1.3
2	2	F	3.5	4.0	2.3
3	3	F	2.4	2.5	NA
4	4	<NA>	2.4	3.0	3.1

other columns-key-values

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
> data_patient_long
   patient gender measurement measurement_value
1        1      M      meas_A               1.2
2        2      F      meas_A               3.5
3        3      F      meas_A               2.4
4        4   <NA>      meas_A               2.4
5        1      M      meas_B               1.4
6        2      F      meas_B               4.0
7        3      F      meas_B               2.5
8        4   <NA>      meas_B               3.0
9        1      M      meas_C               1.3
10       2      F      meas_C               2.3
12       4   <NA>      meas_C               3.1
```

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
> data_patient_long
   patient gender measurement measurement_value
1        1      M      meas_A               1.2
2        2      F      meas_A               3.5
3        3      F      meas_A               2.4
4        4   <NA>      meas_A               2.4
5        1      M      meas_B               1.4
6        2      F      meas_B               4.0
7        3      F      meas_B               2.5
8        4   <NA>      meas_B               3.0
9        1      M      meas_C               1.3
10       2      F      meas_C               2.3
12       4   <NA>      meas_C               3.1
```

keys

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
> data_patient_long
   patient gender measurement measurement_value
1        1      M      meas_A               1.2
2        2      F      meas_A               3.5
3        3      F      meas_A               2.4
4        4   <NA>      meas_A               2.4
5        1      M      meas_B               1.4
6        2      F      meas_B               4.0
7        3      F      meas_B               2.5
8        4   <NA>      meas_B               3.0
9        1      M      meas_C               1.3
10       2      F      meas_C               2.3
12       4   <NA>      meas_C               3.1
```

values

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

```
> data_patient_long
   patient gender measurement measurement_value
1        1      M      meas_A               1.2
2        2      F      meas_A               3.5
3        3      F      meas_A               2.4
4        4   <NA>      meas_A               2.4
5        1      M      meas_B               1.4
6        2      F      meas_B               4.0
7        3      F      meas_B               2.5
8        4   <NA>      meas_B               3.0
9        1      M      meas_C               1.3
10       2      F      meas_C               2.3
12       4   <NA>      meas_C               3.1
```

key-values

Example:

meas_A, 1.2

meas_B, 1.4

meas_C, 1.3

meas_A, 3.5

...

Wide and long patient dataset

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
> data_patient_long
   patient gender measurement measurement_value
1        1      M      meas_A               1.2
2        2      F      meas_A               3.5
3        3      F      meas_A               2.4
4        4   <NA>      meas_A               2.4
5        1      M      meas_B               1.4
6        2      F      meas_B               4.0
7        3      F      meas_B               2.5
8        4   <NA>      meas_B               3.0
9        1      M      meas_C               1.3
10       2      F      meas_C               2.3
12       4   <NA>      meas_C               3.1
```

other columns

Wide and long patient dataset

```
> data_patients
```

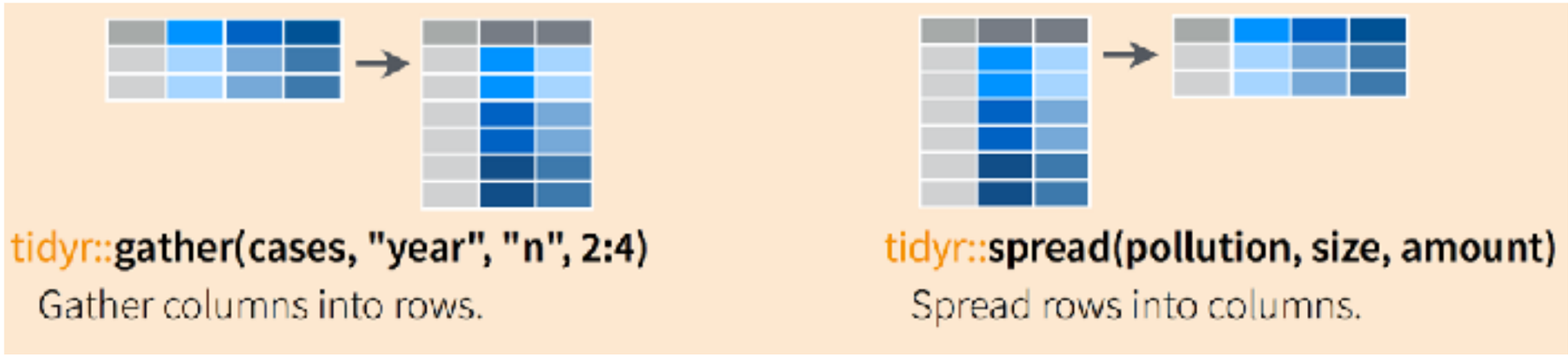
	patient	gender	meas_A	meas_B	meas_C
1	1	M	1.2	1.4	1.3
2	2	F	3.5	4.0	2.3
3	3	F	2.4	2.5	NA
4	4	<NA>	2.4	3.0	3.1

other columns-key-values

```
> data_patient_long
```

	patient	gender	measurement	measurement_value
1	1	M	meas_A	1.2
2	2	F	meas_A	3.5
3	3	F	meas_A	2.4
4	4	<NA>	meas_A	2.4
5	1	M	meas_B	1.4
6	2	F	meas_B	4.0
7	3	F	meas_B	2.5
8	4	<NA>	meas_B	3.0
9	1	M	meas_C	1.3
10	2	F	meas_C	2.3
12	4	<NA>	meas_C	3.1

Wide to long -> gather()



gather()

Gather columns into key-value pairs.

Description

Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed. You use `gather()` when you notice that you have columns that are not variables.

Usage

```
gather(data, key = "key", value = "value", ..., na.rm =
FALSE, convert = FALSE, factor_key = FALSE)
```

The data to convert from wide into long.

gather()

Gather columns into key-value pairs.

Description

Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed. You use `gather()` when you notice that you have columns that are not variables.

Usage

```
gather(data, key = "key", value = "value", ..., na.rm =
FALSE, convert = FALSE, factor_key = FALSE)
```

Name of the key column in the new (long) data frame

gather()

Gather columns into key-value pairs.

Description

Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed. You use `gather()` when you notice that you have columns that are not variables.

Usage

```
gather(data, key = "key", value = "value", ..., na.rm =
FALSE, convert = FALSE, factor_key = FALSE)
```

Name of the value column in the new (long) data frame

gather()

Gather columns into key-value pairs.

Description

Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed. You use `gather()` when you notice that you have columns that are not variables.

Usage

```
gather(data, key = "key", value = "value", ..., na.rm =
FALSE, convert = FALSE, factor_key = FALSE)
```

Names of columns to turn into key-value pairs.

gather()

Gather columns into key-value pairs.

Description

Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed. You use `gather()` when you notice that you have columns that are not variables.

Usage

```
gather(data, key = "key", value = "value", ..., na.rm =
FALSE, convert = FALSE, factor_key = FALSE)
```

Drop NA values. Default False.

gather()

```
> data_patients
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
> gather(data_patients, measurement, measurement_value,
    meas_A, meas_B, meas_C)
```

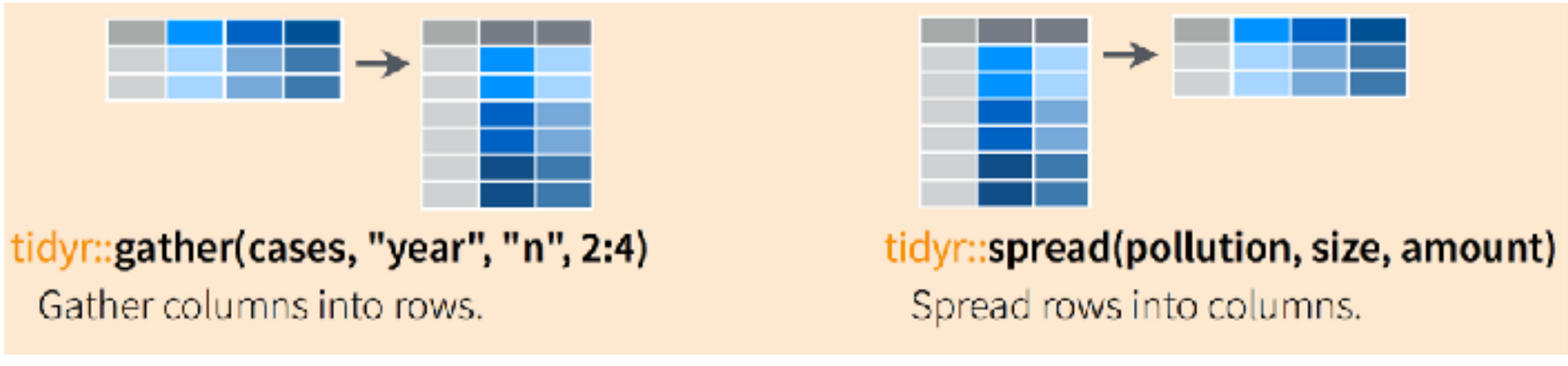
gather()

```
> gather(data_patients, measurement, measurement_value,
         meas_A, meas_B, meas_C)
   patient gender measurement measurement_value
1        1      M      meas_A                1.2
2        2      F      meas_A                3.5
3        3      F      meas_A                2.4
4        4   <NA>      meas_A                2.4
5        1      M      meas_B                1.4
6        2      F      meas_B                4.0
7        3      F      meas_B                2.5
8        4   <NA>      meas_B                3.0
9        1      M      meas_C                1.3
10       2      F      meas_C                2.3
11       3      F      meas_C                 NA
12       4   <NA>      meas_C                3.1
```

gather() - remove NA values

```
> gather(data_patients, measurement, measurement_value,
         meas_A, meas_B, meas_C, na.rm=TRUE)
   patient gender measurement measurement_value
1        1      M      meas_A                1.2
2        2      F      meas_A                3.5
3        3      F      meas_A                2.4
4        4   <NA>      meas_A                2.4
5        1      M      meas_B                1.4
6        2      F      meas_B                4.0
7        3      F      meas_B                2.5
8        4   <NA>      meas_B                3.0
9        1      M      meas_C                1.3
10       2      F      meas_C                2.3
12       4   <NA>      meas_C                3.1
```

Long to wide -> spread()



spread()

Spread a key-value pair across multiple columns.

Description

Spread a key-value pair across multiple columns.

Usage

```
spread(data, key, value, fill = NA, convert = FALSE,
       drop = TRUE, sep = NULL)
```

The data to convert from long into wide.

spread()

Spread a key-value pair across multiple columns.

Description

Spread a key-value pair across multiple columns.

Usage

```
spread(data, key, value, fill = NA, convert = FALSE,
       drop = TRUE, sep = NULL)
```

Name of the key column in the long data frame

spread()

Spread a key-value pair across multiple columns.

Description

Spread a key-value pair across multiple columns.

Usage

```
spread(data, key, value, fill = NA, convert = FALSE,
       drop = TRUE, sep = NULL)
```

Name of the value column in the long data frame

spread()

Spread a key-value pair across multiple columns.

Description

Spread a key-value pair across multiple columns.

Usage

```
spread(data, key, value, fill = NA, convert = FALSE,
       drop = TRUE, sep = NULL)
```

Fill missing positions with this value.

spread()

```
> data_patient_long
   patient gender measurement measurement_value
1        1      M      meas_A                1.2
2        2      F      meas_A                3.5
3        3      F      meas_A                2.4
4        4   <NA>      meas_A                2.4
5        1      M      meas_B                1.4
6        2      F      meas_B                4.0
7        3      F      meas_B                2.5
8        4   <NA>      meas_B                3.0
9        1      M      meas_C                1.3
10       2      F      meas_C                2.3
12       4   <NA>      meas_C                3.1
> spread(data_patient_long, measurement, measurement_value)
```

spread()

```
> spread(data_patient_long, measurement, measurement_value)
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
```

spread() - fill NA values

```
> spread(data_patient_long, measurement, measurement_value)
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5     NA
4       4   <NA>    2.4    3.0    3.1
> spread(data_patient_long, measurement, measurement_value, fill=0)
  patient gender meas_A meas_B meas_C
1       1      M    1.2    1.4    1.3
2       2      F    3.5    4.0    2.3
3       3      F    2.4    2.5      0
4       4   <NA>    2.4    3.0    3.1
```

Plotting

Plots with long data

```
ggplot(data_patient_long, aes(measurement, measurement_value))
+ geom_boxplot()
```