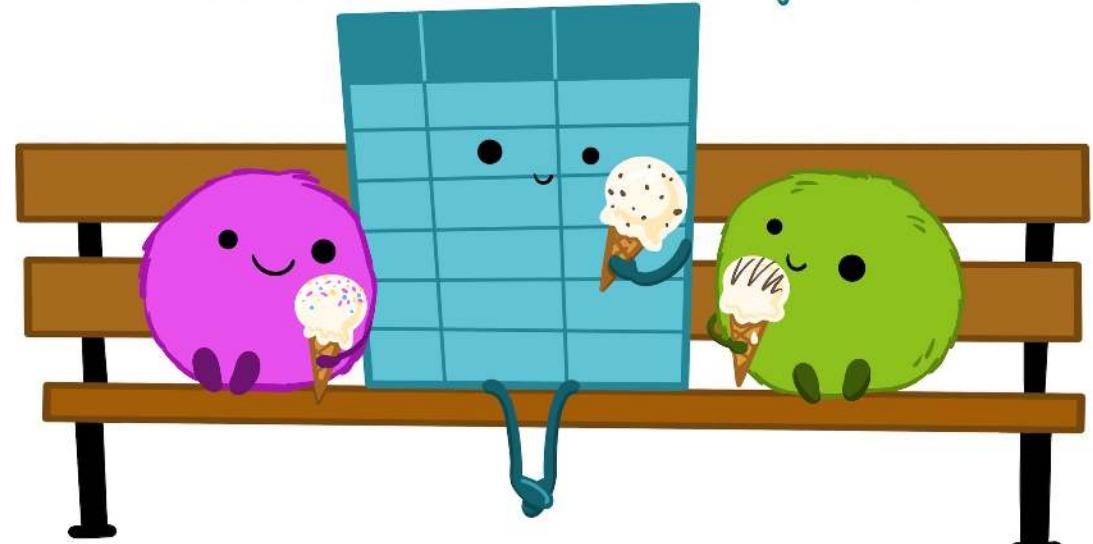


Summarizing & Transforming Data in R

Saving you time and sanity

make friends with tidy data.



[steffilazerte](https://steffilazerte@fosstodon.org)
 [@steffilazerte@fosstodon.org
 \[@steffilazerte
 \\[steffilazerte.ca\\]\\(https://steffilazerte.ca\\)\]\(https://twitter.com/steffilazerte\)](mailto:@steffilazerte@fosstodon.org)

First things first

 Save previous script

 Open New File

(make sure you're in the RStudio Project)

 Write `library(tidyverse)` at the top

 Save this new script

(consider names like `summarizing.R` or `4_sum_and_trans.R`)

Types of Modifications

1. Subset

- `filter()` observations (rows)
- `select()` variables (columns)

2. Joining data sets

- `left_join()`, `right_join()`, etc.

3. Creating new columns

- Creating categories
- Column calculations
- By group
- `mutate()` (and `group_by()`)

4. Summarize existing columns

- Summarizing by group
- `summarize()`

5. Transpose

- Going between **wide** and **long** data formats
 - `pivot_wider()` and `pivot_longer()`
- Transposing for analysis
- Transposing for visualizations

Getting ready

Check out the data:

```
1 library(tidyverse)
2 size <- read_csv("data/grain_size2.csv")
3 size
```

```
# A tibble: 114 × 9
  plot   depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay
  <chr> <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl> <dbl>
1 CSP01     4       13.0      17.4      19.7      14.1      11.2      8.17  16.3
2 CSP01    12       10.7      16.9      19.2      14.1      11.7      9.03  18.4
3 CSP01    35       12.1      17.8      16.1      10.3      9.51      7.47  26.7
4 CSP01    53       17.6      18.2      14.3       9.4      9.1       8.7   22.7
5 CSP01    83       21.0      18.4      14.3      9.79      8.79      7.29  20.4
6 CSP01   105       19.0      18.4      14.4      10.8      9.4       8.22  19.7
7 CSP08     10       11.6      17.1      20.8      16.3      9.55      6.23  18.4
8 CSP08     27       15.4      16.2      17.8      14.3      10.4      6.1   19.6
9 CSP08     90       14.9      15.8      18.6      15.1      11.5      7.56  16.5
10 CSP02    5        8.75      8.64      8.66      12.0      18.3      15.2  28.5
# i 104 more rows
```

Using data sets:

- [grain_size2.csv](#)
- [grain_meta.csv](#)

Subsetting

By rows and column

filter() observations

filter() is from dplyr*

```
1 filter(data, expression1, expression2, etc.)
```



- tidyverse functions always start with data
- Column expressions reference actual columns in data
- Here we use logical statements relating to column values

* part of the tidyverse

filter() observations

filter() by category

```
1 filter(size, plot %in% c("CSP11", "CSP13"))

# A tibble: 9 × 9
  plot    depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay
  <chr>   <dbl>      <dbl>       <dbl>     <dbl>      <dbl>       <dbl>     <dbl> <dbl>
1 CSP13     2        22.1       17.5     18.3      11.9       7.92     6.05  16.3
2 CSP13    10        12.1       14.9      18        13.1      10.4      7.92  23.6
3 CSP13    25        13.7       12.7     14.3      11.7       9.67     6.31  31.6
4 CSP13    60        27.1       9.74     11.1      9.69      9.79      7.82  24.8
5 CSP13   140        10.4      15.3      16.0      12.4      12.4     10.2   23.5
6 CSP11    20        6.67      3.94      5.52      23.7      23        14.8  22.3
7 CSP11    30        5.27      4.23      6.11      23.6      23.9     15.3   21.6
8 CSP11    47        4.34      4.03      6.62      24.5      25.5     13.8   21.3
9 CSP11   143        5.28      4.26      7.07      22.8      28.0     12.4   20.2
```



filter() observations

filter() by category

```
1 filter(size, plot %in% c("CSP11", "CSP13"))

# A tibble: 9 × 9
  plot    depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay
  <chr>   <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl> <dbl>
1 CSP13     2        22.1       17.5      18.3      11.9       7.92      6.05  16.3
2 CSP13    10        12.1       14.9       18        13.1      10.4       7.92  23.6
3 CSP13    25        13.7       12.7      14.3      11.7       9.67      6.31  31.6
4 CSP13    60        27.1       9.74      11.1      9.69       9.79      7.82  24.8
5 CSP13   140        10.4      15.3      16.0      12.4      12.4      10.2  23.5
6 CSP11    20        6.67       3.94      5.52      23.7       23        14.8  22.3
7 CSP11    30        5.27       4.23      6.11      23.6      23.9      15.3  21.6
8 CSP11    47        4.34       4.03      6.62      24.5      25.5      13.8  21.3
9 CSP11   143        5.28       4.26      7.07      22.8      28.0      12.4  20.2
```



Note: To save this as a separate object, don't forget assignments:

```
1 size_sub <- filter(size, plot %in% c("CSP11", "CSP13"))
```

filter() observations



filter() by measures

```
1 filter(size, depth > 140 | depth < 4)

# A tibble: 9 × 9
  plot   depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay
  <chr> <dbl>      <dbl>       <dbl>     <dbl>      <dbl>       <dbl>     <dbl> <dbl>
1 CSP13    2        22.1       17.5     18.3      11.9       7.92     6.05  16.3
2 CSP19   190       3.33       4.28     14.2      42.8      21.5      9.92   4
3 CSP11   143       5.28       4.26     7.07      22.8      28.0     12.4   20.2
4 CSP14    3        16.1       15.0     17.5      12.2      12        9.88  17.3
5 CSP15   146       13.6       12.3     12.5      12.0      18.1     10.4   21.1
6 CSP20    3        5.12       5.09     17.9      25.9      14.3     11.8   19.9
7 CSP20   150       22.7       12.9     12.7      17.7      14.9      7.59  11.5
8 CSP21    3        14.1       11.6     11.9      14.1      15.5     10.4   22.4
9 CSP22   182       17.9       13.6     13.1      13.5      12.6     8.39  20.9
```

filter() observations

filter() by a combination → use comma for AND

```
1 filter(size,
2       depth > 100,
3       plot %in% c("CSP11", "CSP13"))

# A tibble: 2 × 9
  plot   depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay
  <chr> <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl> <dbl>
1 CSP13    140      10.4     15.3     16.0     12.4     12.4     10.2    23.5
2 CSP11    143      5.28     4.26     7.07     22.8     28.0     12.4    20.2
```

filter() observations

filter() by a combination → use comma for AND

```
1 filter(size,
2       depth > 100,
3       plot %in% c("CSP11", "CSP13"))

# A tibble: 2 × 9
  plot   depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay
  <chr> <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 CSP13    140      10.4     15.3     16.0     12.4     12.4     10.2    23.5
2 CSP11    143      5.28     4.26     7.07     22.8     28.0     12.4    20.2
```

Equivalent to using &

```
1 filter(size,
2       depth > 100 &
3       plot %in% c("CSP11", "CSP13"))
```

Tangent

Logical Operators

Logical Operators

Possible options

Operator	Code
OR	
AND	&
EQUAL	==
NOT EQUAL	!=
NOT	!
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=
In	%in%

Your turn!

What do the following filters do?

```
1 size <- read_csv("data/grain_size2.csv")
2
3 filter(size, depth == 20)
4 filter(size, depth == 20, clay < 20)
5 filter(size, depth %in% c(20, 100) & clay == 6)
6 filter(size, !plot %in% c("CSP01", "CSP13"))
7 filter(size, depth >= 146)
8 filter(size, depth > 146)
9 filter(size, coarse_silt < 9 | clay < 5)
```

select () variables

select () is from **dplyr***

```
1 select(data, selection1, selection2, etc.)
```



- tidyverse functions always start with data
- Specify columns to keep or remove
- Column `selections` reference actual columns in data

* part of the tidyverse

select() variables

select() by name

```
1 select(size, coarse_sand, medium_sand, fine_sand)

# A tibble: 114 × 3
  coarse_sand medium_sand fine_sand
      <dbl>       <dbl>      <dbl>
1     13.0       17.4      19.7
2     10.7       16.9      19.2
3     12.1       17.8      16.1
4     17.6       18.2      14.3
# i 110 more rows
```

select() variables

select() by name

```
1 select(size, coarse_sand, medium_sand, fine_sand)

# A tibble: 114 × 3
  coarse_sand medium_sand fine_sand
     <dbl>      <dbl>      <dbl>
1     13.0      17.4      19.7
2     10.7      16.9      19.2
3     12.1      17.8      16.1
4     17.6      18.2      14.3
# i 110 more rows
```

Using helper functions

```
1 select(size, ends_with("sand"))

# A tibble: 114 × 3
  coarse_sand medium_sand fine_sand
     <dbl>      <dbl>      <dbl>
1     13.0      17.4      19.7
2     10.7      16.9      19.2
3     12.1      17.8      16.1
4     17.6      18.2      14.3
# i 110 more rows
```

select() variables

select() by name

```
1 select(size, coarse_sand, medium_sand, fine_sand)  
  
# A tibble: 114 × 3  
  coarse_sand medium_sand fine_sand  
        <dbl>       <dbl>      <dbl>  
1       13.0      17.4     19.7  
2       10.7      16.9     19.2  
3       12.1      17.8     16.1  
4       17.6      18.2     14.3  
# i 110 more rows
```

Using helper functions

```
1 select(size, ends_with("sand"))  
  
# A tibble: 114 × 3  
  coarse_sand medium_sand fine_sand  
        <dbl>       <dbl>      <dbl>  
1       13.0      17.4     19.7  
2       10.7      16.9     19.2  
3       12.1      17.8     16.1  
4       17.6      18.2     14.3  
# i 110 more rows
```

Some other helper functions ([?select_helpers](#)):

Function	Usage
starts_with()	starts_with("fine")
contains()	contains("sand")
everything()	Useful for rearranging
matches()	Uses regular expressions

select() variables

Put it all together

To explore the data

```
1 size |>
2   filter(depth > 100,
3         plot %in% c("CSP13", "CSP25")) |>
4   select(plot, depth, ends_with("sand"))

# A tibble: 2 × 5
  plot    depth coarse_sand medium_sand fine_sand
  <chr>   <dbl>      <dbl>       <dbl>      <dbl>
1 CSP13     140        10.4       15.3       16.0
2 CSP25     130        18.6       21.3       13.8
```

select () variables

Put it all together

To explore the data

```
1 size |>
2   filter(depth > 100,
3         plot %in% c("CSP13", "CSP25")) |>
4   select(plot, depth, ends_with("sand"))

# A tibble: 2 × 5
  plot    depth coarse_sand medium_sand fine_sand
  <chr>   <dbl>      <dbl>       <dbl>      <dbl>
1 CSP13     140        10.4       15.3       16.0
2 CSP25     130        18.6       21.3       13.8
```

To save as a separate object

```
1 size_sub_sand <- size |>
2   filter(depth > 100,
3         plot %in% c("CSP13", "CSP25")) |>
4   select(plot, depth, ends_with("sand"))
```

Your turn: Subsetting

- Subset the data to variables **plot**, **depth** and all measures of **sand**
- Keep only values where there is at least 30% **clay**

```
1 size <- read_csv("data/grain_size2.csv") |>  
2 filter(???) |>  
3 select(???)
```

Note:

All particle values
are percentages
(depth is cm)

Too Easy?

What happens if you `select()` before you `filter()`?
How many different ways can you select these columns?

Your turn: Subsetting

- Subset the data to variables **plot**, **depth** and all measures of **sand**
- Keep only values where there is at least 30% **clay**

```
1 size <- read_csv("data/grain_size2.csv") |>
2   filter(clay >= 30) |>
3   select(plot, depth, ends_with("sand"))
4
5 head(size)

# A tibble: 2 × 5
  plot    depth coarse_sand medium_sand fine_sand
  <chr>   <dbl>      <dbl>       <dbl>      <dbl>
1 CSP02     36        8.15       9.24      8.55
2 CSP13     25       13.7       12.7      14.3
```

Your turn: Subsetting

- Subset the data to variables **plot**, **depth** and all measures of **sand**
- Keep only values where there is at least 30% **clay**

```
1 size <- read_csv("data/grain_size2.csv") |>
2   filter(clay >= 30) |>
3   select(plot, depth, ends_with("sand"))
4
5 head(size)

# A tibble: 2 × 5
  plot    depth coarse_sand medium_sand fine_sand
  <chr>   <dbl>      <dbl>       <dbl>      <dbl>
1 CSP02     36        8.15       9.24       8.55
2 CSP13     25       13.7        12.7      14.3
```

Select equivalents:

- `select(plot, depth, ends_with("sand"))`
- `select(plot, depth, contains("sand"))`
- `select(plot, depth, coarse_sand, medium_sand, fine_sand)`
- `select(-coarse_silt, -medium_silt, -fine_silt, -clay)`

Your turn: Subsetting (Too Easy?)

What happens if you `select()` before you `filter()`?

```
1 size <- read_csv("data/grain_size2.csv") |>  
2   select(plot, depth, ends_with("sand")) |>  
3   filter(clay >= 30)
```

```
Error in `filter()`:  
i In argument: `clay >= 30`.  
Caused by error:  
! object 'clay' not found
```

- Lines are sequential
- First `select()` removes column `clay`
- Then `filter()` cannot find `clay`
 - (`object 'clay' not found`)

Joining or Merging data

Joining data sets

Measurements

Plot	Date	n_birds
A	2026-02-19	1
A	2026-03-15	11
A	2026-04-08	2
B	2026-05-02	4
B	2026-05-26	10
B	2026-06-19	21

Metadata

Plot	Vegetation Density
A	50
B	76

Joining data sets

Measurements

Plot	Date	n_birds
A	2026-02-19	1
A	2026-03-15	11
A	2026-04-08	2
B	2026-05-02	4
B	2026-05-26	10
B	2026-06-19	21

Metadata

Plot	Vegetation Density
A	50
B	76

Joining them together

Metadata is duplicated to line up with measurements

Plot	Date	n_birds	Vegetation Density
A	2026-02-19	1	50
A	2026-03-15	11	50
A	2026-04-08	2	50
B	2026-05-02	4	76
B	2026-05-26	10	76
B	2026-06-19	21	76

Joining data sets

Index or Metadata

```
1 meta <- read_csv("data/grain_meta.csv")
2 meta

# A tibble: 27 × 4
  plot habitat technician date
  <chr> <chr>     <chr>   <date>
1 CSP01 forest    Catharine 2009-04-23
2 CSP02 forest    Catharine 2009-05-06
3 CSP03 clearcut  Jason     2008-09-03
4 CSP04 forest    Catharine 2008-09-29
5 CSP05 grassland Catharine 2009-02-05
6 CSP06 grassland Jason     2008-07-01
7 CSP07 grassland Jason     2008-11-19
8 CSP08 grassland Catharine 2009-03-02
9 CSP09 forest    Catharine 2008-08-21
10 CSP10 grassland Jason    2009-02-17
11 CSP11 forest    Jason     2008-09-16
12 CSP12 grassland Catharine 2009-03-28
13 CSP13 grassland Catharine 2008-07-13
14 CSP14 clearcut  Jason     2009-06-01
15 CSP15 forest    Yasir     2008-12-02
# i 12 more rows
```

Measurements

```
1 size <- read_csv("data/grain_size2.csv")
2 size

# A tibble: 114 × 9
  plot depth coarse_sand medium_sand fine_sand coarse_silt
  <chr> <dbl>      <dbl>       <dbl>      <dbl>       <dbl>
1 CSP01     4        13.0       17.4      19.7       14.1
2 CSP01    12        10.7       16.9      19.2       14.1
3 CSP01    35        12.1       17.8      16.1       10.3
4 CSP01    53        17.6       18.2      14.3        9.4
5 CSP01    83        21.0       18.4      14.3       9.79
6 CSP01   105        19.0       18.4      14.4      10.8
7 CSP08     10        11.6       17.1      20.8      16.3
8 CSP08     27        15.4       16.2      17.8      14.3
9 CSP08     90        14.9       15.8      18.6      15.1
10 CSP02    5         8.75       8.64      8.66      12.0
11 CSP02    11        9.89       8.68      8.34      10.7
12 CSP02    36        8.15       9.24      8.55      10.7
13 CSP02    56        12.0       8.63      8.06      11.1
14 CSP02    70        17.5       10.5      8.45      11.2
15 CSP02    78        23.3       15.0      11.0      9.97
# i 99 more rows
# i 3 more variables: medium_silt <dbl>, fine_silt <dbl>, clay <dbl>
```

Joining data sets

Index or Metadata

```
1 meta <- read_csv("data/grain_meta.csv")
2 meta

# A tibble: 27 × 4
  plot habitat technician date
  <chr> <chr>     <chr>   <date>
1 CSP01 forest    Catharine 2009-04-23
2 CSP02 forest    Catharine 2009-05-06
3 CSP03 clearcut  Jason     2008-09-03
4 CSP04 forest    Catharine 2008-09-29
5 CSP05 grassland Catharine 2009-02-05
6 CSP06 grassland Jason     2008-07-01
7 CSP07 grassland Jason     2008-11-19
8 CSP08 grassland Catharine 2009-03-02
9 CSP09 forest    Catharine 2008-08-21
10 CSP10 grassland Jason    2009-02-17
11 CSP11 forest    Jason     2008-09-16
12 CSP12 grassland Catharine 2009-03-28
13 CSP13 grassland Catharine 2009-07-12
14 CSP14 clearcut  Jason     ...
15 CSP15 forest    Yasir     ...
# i 12 more rows
```

Measurements

```
1 size <- read_csv("data/grain_size2.csv")
2 size

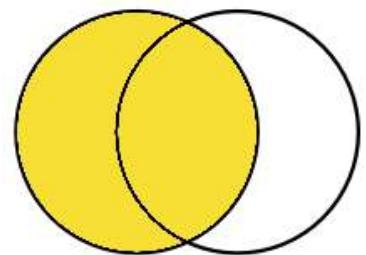
# A tibble: 114 × 9
  plot depth coarse_sand medium_sand fine_sand coarse_silt
  <chr> <dbl>      <dbl>       <dbl>      <dbl>       <dbl>
1 CSP01     4        13.0       17.4      19.7       14.1
2 CSP01    12        10.7       16.9      19.2       14.1
3 CSP01    35        12.1       17.8      16.1       10.3
4 CSP01    53        17.6       18.2      14.3        9.4
5 CSP01    83        21.0       18.4      14.3       9.79
6 CSP01   105        19.0       18.4      14.4       10.8
7 CSP08     10        11.6       17.1      20.8       16.3
8 CSP08     27        15.4       16.2      17.8       14.3
9 CSP08     90        14.9       15.8      18.6       15.1
10 CSP02    5         8.75       8.64      8.66       12.0
11 CSP02    11        9.89       8.68      8.34       10.7
12 CSP02    36        8.15       9.24      8.55       10.7
13 CSP02    56        10.0       9.63      8.06       11.1
14 CSP02    56        10.0       9.55      8.45       11.2
15 CSP02    56        10.0       9.00      11.0       9.97
# i 13 more variables: medium_silt <dbl>, fine_silt <dbl>, clay <dbl>
```

plot (CSP01, CSP02, etc.) identifies data in both

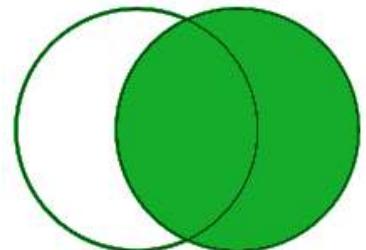
Types of Join: Which rows to keep?

`left_join(x, y)`

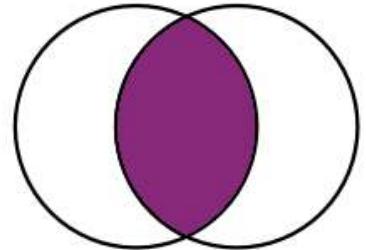
- Keep all rows in `x`
- Keep rows in `y` only if they're also in `x`



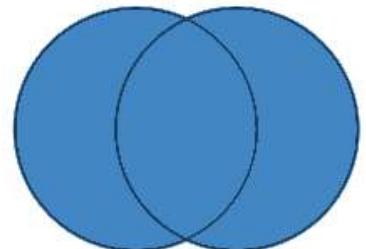
`left_join(x, y)`



`right_join(x, y)`



`inner_join(x, y)`



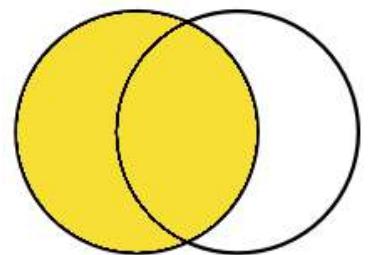
`full_join(x, y)`

Adapted from “R for Data Science”

Types of Join: Which rows to keep?

`left_join(x, y)`

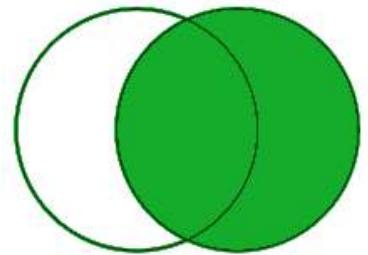
- Keep all rows in `x`
- Keep rows in `y` only if they're also in `x`



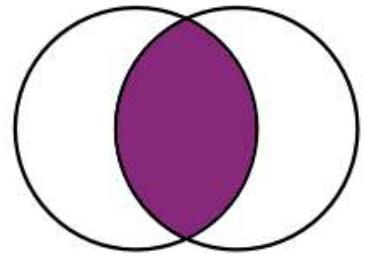
`left_join(x, y)`

`right_join(x, y)`

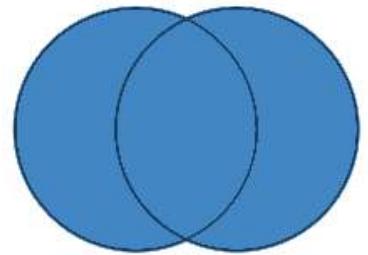
- Keep all rows in `y`
- Keep rows in `x` only if they're also in `y`



`right_join(x, y)`



`inner_join(x, y)`



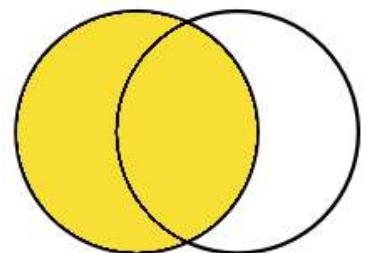
`full_join(x, y)`

Adapted from “R for Data Science”

Types of Join: Which rows to keep?

`left_join(x, y)`

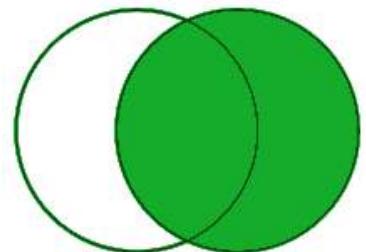
- Keep all rows in `x`
- Keep rows in `y` only if they're also in `x`



`left_join(x, y)`

`right_join(x, y)`

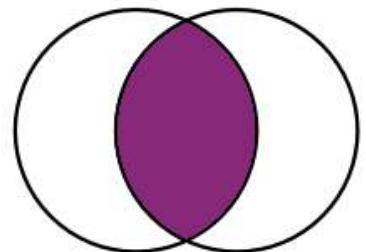
- Keep all rows in `y`
- Keep rows in `x` only if they're also in `y`



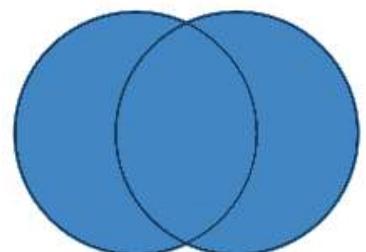
`right_join(x, y)`

`inner_join(x, y)`

- Keep **only** rows that exist in **both** data frames



`inner_join(x, y)`



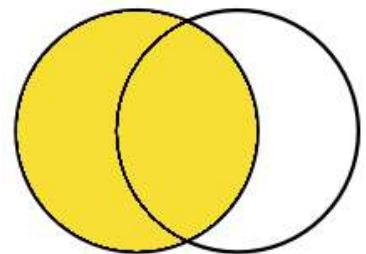
`full_join(x, y)`

Adapted from “R for Data Science”

Types of Join: Which rows to keep?

`left_join(x, y)`

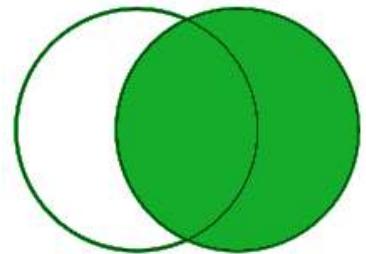
- Keep all rows in `x`
- Keep rows in `y` only if they're also in `x`



`left_join(x, y)`

`right_join(x, y)`

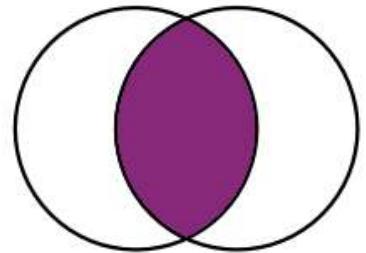
- Keep all rows in `y`
- Keep rows in `x` only if they're also in `y`



`right_join(x, y)`

`inner_join(x, y)`

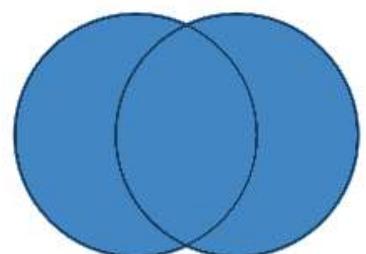
- Keep **only** rows that exist in **both** data frames



`inner_join(x, y)`

`full_join(x, y)`

- Keep **all** rows that exist in **either** `x` or `y`



`full_join(x, y)`

Adapted from “R for Data Science”

Joining data sets

`left_join()` is from `dplyr`*

```
1 left_join(x = data, y = data_to_join, by = c("column1", "column2"), ...)
```



- `tidyverse` functions always start with data (`x`)
- Here, also need second dataset (`y`)
- `by` refers columns in `x` and `y` used to join

* part of the `tidyverse`

Joining data sets

Keep all measurements (`size`), only keep `meta` if we have a measurement

```
1 size <- left_join(x = size, y = meta, by = "plot")
```

Joining data sets

Keep all measurements (`size`), only keep `meta` if we have a measurement

```
1 size <- left_join(x = size, y = meta, by = "plot")
```

```
1 # A tibble: 114 × 12
2   plot    depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay habitat technician
3   <chr>   <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl> <dbl> <chr>   <chr>
4   1 CSP01     4        13.0       17.4       19.7       14.1       11.2       8.17    16.3 forest Catharine
5   2 CSP01    12        10.7       16.9       19.2       14.1       11.7       9.03    18.4 forest Catharine
6   3 CSP01    35        12.1       17.8       16.1       10.3       9.51       7.47    26.7 forest Catharine
7   4 CSP01    53        17.6       18.2       14.3        9.4       9.1        8.7     22.7 forest Catharine
8   5 CSP01    83        21.0       18.4       14.3       9.79       8.79       7.29    20.4 forest Catharine
9   6 CSP01   105        19.0       18.4       14.4       10.8       9.4        8.22    19.7 forest Catharine
10  7 CSP08    10        11.6       17.1       20.8       16.3       9.55       6.23    18.4 grassland Catharine
11  8 CSP08    27        15.4       16.2       17.8       14.3       10.4       6.1     19.6 grassland Catharine
12  9 CSP08    90        14.9       15.8       18.6       15.1       11.5       7.56    16.5 grassland Catharine
13 10 CSP02     5        8.75       8.64       8.66       12.0       18.3       15.2    28.5 forest Catharine
14 # i 104 more rows
```

Joining data sets

Keep all measurements (`size`), only keep `meta` if we have a measurement

```
1 size <- left_join(x = size, y = meta, by = "plot")
```

```
1 # A tibble: 114 × 12
2   plot    depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay habitat technician
3   <chr>   <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl> <dbl> <chr>   <chr>
4   1 CSP01     4        13.0       17.4       19.7       14.1       11.2       8.17  16.3 forest Catharine
5   2 CSP01    12        10.7       16.9       19.2       14.1       11.7       9.03  18.4 forest Catharine
6   3 CSP01    35        12.1       17.8       16.1       10.3       9.51       7.47  26.7 forest Catharine
7   4 CSP01    53        17.6       18.2       14.3        9.4       9.1        8.7   22.7 forest Catharine
8   5 CSP01    83        21.0       18.4       14.3       9.79       8.79       7.29  20.4 forest Catharine
9   6 CSP01   105        19.0       18.4       14.4       10.8       9.4        8.22  19.7 forest Catharine
10  7 CSP08    10        11.6       17.1       20.8       16.3       9.55       6.23  18.4 grassland Catharine
11  8 CSP08    27        15.4       16.2       17.8       14.3       10.4       6.1   19.6 grassland Catharine
12  9 CSP08    90        14.9       15.8       18.6       15.1       11.5       7.56  16.5 grassland Catharine
13 10 CSP02     5        8.75       8.64       8.66       12.0       18.3       15.2   28.5 forest Catharine
14 # i 104 more rows
```

For more information see R for Data Science [Chapter 19.3 Basic joins](#)

Creating/modifying columns with `mutate()`



Artwork by @allison_horst

Creating new columns

`mutate()` is from `dplyr`*

```
1 mutate(data, column1 = expression1, column2 = expression2, ...)
```



- `tidyverse` functions always start with `data`
- Create new or modify existing `columns` in the `data`
- Columns filled according to `expression`

* part of the `tidyverse`

Creating new columns

```
1 size <- read_csv("data/grain_size2.csv") |>  
2   mutate(total_sand = coarse_sand + medium_sand + fine_sand)
```

Creating new columns

```
1 size <- read_csv("data/grain_size2.csv") |>
2   mutate(total_sand = coarse_sand + medium_sand + fine_sand)
```

Creates new column at the end, `total_sand`

```
# A tibble: 114 × 10
  plot    depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay total_sand
  <chr> <dbl>      <dbl>       <dbl>     <dbl>      <dbl>       <dbl>     <dbl> <dbl>      <dbl>
1 CSP01     4        13.0      17.4     19.7      14.1      11.2      8.17  16.3      50.1
2 CSP01    12        10.7      16.9     19.2      14.1      11.7      9.03  18.4      46.8
3 CSP01    35        12.1      17.8     16.1      10.3      9.51      7.47  26.7      46.0
4 CSP01    53        17.6      18.2     14.3       9.4      9.1       8.7   22.7      50.1
5 CSP01    83        21.0      18.4     14.3      9.79      8.79      7.29  20.4      53.8
6 CSP01   105        19.0      18.4     14.4      10.8      9.4       8.22  19.7      51.9
7 CSP08    10        11.6      17.1     20.8      16.3      9.55      6.23  18.4      49.6
8 CSP08    27        15.4      16.2     17.8      14.3      10.4      6.1   19.6      49.5
9 CSP08    90        14.9      15.8     18.6      15.1      11.5      7.56  16.5      49.2
10 CSP02   5         8.75      8.64     8.66      12.0      18.3      15.2  28.5      26.0
# i 104 more rows
```

Creating new columns

```
1 size <- read_csv("data/grain_size2.csv") |>  
2   mutate(total_sand = coarse_sand + medium_sand + fine_sand)
```

Creates new column at the end, `total_sand`

```
# A tibble: 114 × 10  
  plot    depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay total_sand  
  <chr> <dbl>      <dbl>       <dbl>     <dbl>      <dbl>       <dbl>     <dbl> <dbl>      <dbl>  
1 CSP01     4        13.0       17.4     19.7      14.1       11.2      8.17  16.3      50.1  
2 CSP01    12        10.7       16.9     19.2      14.1       11.7      9.03  18.4      46.8  
3 CSP01    35        12.1       17.8     16.1      10.3       9.51      7.47  26.7       46  
4 CSP01    53        17.6       18.2     14.3       9.4       9.1       8.7   22.7      50.1  
5 CSP01    83        21.0       18.4     14.3      9.79      8.79      7.29  20.4      53.8  
6 CSP01   105        19.0       18.4     14.4      10.8       9.4       8.22  19.7      51.9  
7 CSP08     10        11.6       17.1     20.8      16.3       9.55      6.23  18.4      49.6  
8 CSP08     27        15.4       16.2     17.8      14.3      10.4       6.1   19.6      49.5  
9 CSP08     90        14.9       15.8     18.6      15.1      11.5      7.56  16.5      49.2  
10 CSP02    5         8.75      8.64      8.66      12.0      18.3      15.2   28.5      26.0  
# i 104 more rows
```

Note: Column math is *vectorized* (i.e., row by row)

Tangent

Vectorizing

Tangent: Vectorized

Vectorized functions run in parallel across vectors

- Many functions in R are vectorized
- Makes them faster and easier

For example

```
1 a <- c(1, 2, 3)
2 a + a
3 a * a
```

Tangent: Vectorized

Vectorized functions run in parallel across vectors

- Many functions in R are vectorized
- Makes them faster and easier
- But not all functions are vectorized

For example

```
1 a <- c(1, 2, 3)
2 a + a
3 a * a
```

For example

```
1 sum(a)
2 sum(a, a)
3 mean(a)
4 mean(c(a, a))
```

Back to `mutate()` ...

Your turn: Creating new columns

- Add a calculation for **total silt**

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         ???)
```

Too Easy?

What happens if you add `total_sand` and `total_silt` together in the same `mutate()` function?

Your turn: Creating new columns

- Add a calculation for **total silt**

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt)
```

Your turn: Creating new columns (Too Easy?)

- What happens if you add `total_sand` and `total_silt` together in the same `mutate()`?

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt,
7         total = total_sand + total_silt)
```

- You get the sum!
- Lines within `mutate()` run sequentially
- You can create `total_sand` and `total_silt` in the first two lines then use them in the 3rd
- But you could not create `total_sand` and `total_silt` *after* using them

Your turn: Creating new columns

- Check your work

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt)
7
8 select(size, contains("silt"))

# A tibble: 114 × 4
  coarse_silt medium_silt fine_silt total_silt
  <dbl>       <dbl>      <dbl>      <dbl>
1     14.1        11.2      8.17      33.5
2     14.1        11.7      9.03      34.8
3     10.3        9.51      7.47      27.3
4      9.4        9.1       8.7       27.2
5      9.79       8.79      7.29      25.9
6     10.8        9.4       8.22      28.4
7     16.3        9.55      6.23      32.1
8     14.3        10.4      6.1       30.8
9     15.1        11.5      7.56      34.2
10    12.0        18.3      15.2      45.4
# i 104 more rows
```

Your turn: Creating new columns

- Check your work

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt)
7
8 select(size, contains("silt"))

# A tibble: 114 × 4
  coarse_silt medium_silt fine_silt total_silt
  <dbl>       <dbl>      <dbl>      <dbl>
1     14.1        11.2      8.17      33.5
2     14.1        11.7      9.03      34.8
3     10.3        9.51      7.47      27.3
4      9.4        9.1       8.7       27.2
5      9.79       8.79      7.29      25.9
6     10.8        9.4       8.22      28.4
7     16.3        9.55      6.23      32.1
8     14.3        10.4      6.1       30.8
9     15.1        11.5      7.56
10    12.0        18.3      15.2
# i 104 more rows
```

Wait... that doesn't add up!

Tangent

Tibbles

Where are...

... the decimal points?

- `tibble` rounds values for easy viewing

```
# A tibble: 114 × 14
  plot depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay habitat
  <chr> <dbl>      <dbl>       <dbl>     <dbl>      <dbl>       <dbl>     <dbl> <chr>
1 CSP01    4        13.0      17.4     19.7     14.1      11.2      8.17  16.3 forest
2 CSP01   12        10.7      16.9     19.2     14.1      11.7      9.03  18.4 forest
3 CSP01   35        12.1      17.8     16.1     10.3      9.51      7.47  26.7 forest
4 CSP01   53        17.6      18.2     14.3      9.4       9.1       8.7   22.7 forest
5 CSP01   83        21.0      18.4     14.3      9.79      8.79      7.29  20.4 forest
# i 109 more rows
# i 4 more variables: technician <chr>, date <date>, total_sand <dbl>, total_silt <dbl>
```

... my data?

```
# i 109 more rows
# i 5 more variables: technician <chr> ...
```

To see raw data

- Click on the name in the Environment pane
- Or use `as.data.frame()`

```
1 as.data.frame(size)
```

	plot	depth	coarse_sand	medium_sand	fine_sand	coarse_silt	medium_silt	fine_silt	clay	habitat
1	CSP01	4	13.04	17.37	19.71	14.12	11.25	8.17	16.30	forest
2	CSP01	12	10.74	16.90	19.15	14.13	11.68	9.03	18.40	forest
3	CSP01	35	12.11	17.75	16.14	10.33	9.51	7.47	26.70	forest
4	CSP01	53	17.61	18.16	14.32	9.40	9.10	8.70	22.70	forest
5	CSP01	83	21.05	18.38	14.34	9.79	8.79	7.29	20.40	forest
6	CSP01	105	19.02	18.43	14.44	10.79	9.40	8.22	19.70	forest
7	CSP08	10	11.60	17.14	20.81	16.30	9.55	6.23	18.40	grassland
8	CSP08	27	15.44	16.25	17.85	14.27	10.44	6.10	19.60	grassland
9	CSP08	90	14.88	15.79	18.57	15.13	11.54	7.56	16.50	grassland
10	CSP02	5	8.75	8.64	8.66	11.96	18.27	15.22	28.50	forest
11	CSP02	11	9.89	8.68	8.34	10.70	18.33	14.30	29.80	forest
12	CSP02	36	8.15	9.24	8.55	10.68	18.96	14.45	30.00	forest
13	CSP02	56	12.02	8.63	8.06	11.08	17.95	13.74	28.50	forest
14	CSP02	70	17.54	10.47	8.45	11.16	16.85	12.99	22.50	forest
15	CSP02	78	23.27	14.96	11.03	9.97	13.79	10.97	16.00	forest
16	CSP02	100	23.22	16.98	9.68	11.17	12.88	11.17	14.90	forest
17	CSP04	5	6.24	8.43	14.15	17.97	14.33	10.57	28.30	forest
18	CSP04	40	6.30	7.92	14.97	17.89	15.48	10.46	27.00	forest
19	CSP04	60	6.66	8.03	14.61	17.32	15.06	10.45	27.90	forest
20	CSP04	80	7.06	8.13	14.83	16.42	15.71	10.20	27.60	forest
21	CSP04	110	12.78	7.66	13.66	16.47	15.37	11.05	23.00	forest
22	CSP05	5	22.48	15.14	15.69	13.43	10.51	6.50	16.20	grassland

To see all rows

- Use `print()`

```
1 print(size, n = Inf)

# A tibble: 114 × 14
  plot depth coarse_sand medium_sand fine_sand coarse_silt medium_silt fine_silt clay habitat
  <chr> <dbl>     <dbl>      <dbl>     <dbl>      <dbl>      <dbl>      <dbl> <dbl> <chr>
1 CSP01    4       13.0      17.4      19.7      14.1      11.2      8.17   16.3 forest
2 CSP01   12       10.7      16.9      19.2      14.1      11.7      9.03   18.4 forest
3 CSP01   35       12.1      17.8      16.1      10.3      9.51     7.47   26.7 forest
4 CSP01   53       17.6      18.2      14.3       9.4      9.1      8.7    22.7 forest
5 CSP01   83       21.0      18.4      14.3      9.79     8.79     7.29   20.4 forest
6 CSP01  105       19.0      18.4      14.4      10.8      9.4      8.22   19.7 forest
7 CSP08   10       11.6      17.1      20.8      16.3      9.55     6.23   18.4 grassland
8 CSP08   27       15.4      16.2      17.8      14.3      10.4      6.1    19.6 grassland
9 CSP08   90       14.9      15.8      18.6      15.1      11.5     7.56   16.5 grassland
10 CSP02   5        8.75     8.64      8.66      12.0      18.3     15.2   28.5 forest
11 CSP02   11       9.89     8.68      8.34      10.7      18.3     14.3   29.8 forest
12 CSP02   36       8.15     9.24      8.55      10.7      19.0     14.4   30     forest
13 CSP02   56       12.0     8.63      8.06      11.1      18.0     13.7   28.5 forest
14 CSP02   70       17.5     10.5      8.45      11.2      16.8     13.0   22.5 forest
15 CSP02   78       23.3     15.0      11.0      9.97     13.8     11.0    16     forest
16 CSP02  100       23.2     17.0      9.68      11.2      12.9     11.2   14.9 forest
17 CSP04   5        6.24     8.43      14.2      18.0     14.3     10.6   28.3 forest
18 CSP04   40       6.3      7.92      15.0      17.9     15.5     10.5    27     forest
19 CSP04   60       6.66     8.03      14.6      17.3     15.1     10.4   27.9 forest
20 CSP04   80       7.06     8.13      14.8      16.4     15.7     10.2   27.6 forest
21 CSP04  110       12.8     7.66      13.7      16.5     15.4     11.0    23     forest
22 CSP05   5        22.5     15.1      15.7      13.4     10.5     6.5    16.2 grassland
```

Back to `mutate()` ...

Mutating by group

`mutate()` is from `dplyr`*



`mutate()` has a `.by` argument

```
1 mutate(data, column1 = expression1, column2 = expression2, ..., .by = c(group1, group2))
```

- Applies splits data frame by grouping columns *before* calculations

* part of the `tidyverse`

Mutating by group

`mutate()` without grouping

```
1 size <- size |>  
2   mutate(mean_sand_all = mean(total_sand))
```

```
# A tibble: 114 × 3  
  plot  total_sand mean_sand_all  
  <chr>    <dbl>        <dbl>  
1 CSP01      50.1        39.6  
2 CSP01      46.8        39.6  
3 CSP01       46          39.6  
4 CSP01      50.1        39.6  
5 CSP01      53.8        39.6  
6 CSP01      51.9        39.6  
7 CSP08      49.6        39.6  
8 CSP08      49.5        39.6  
9 CSP08      49.2        39.6  
10 CSP02     26.0        39.6  
# i 104 more rows
```

Overall mean calculated

Mutating by group

`mutate()` without grouping

```
1 size <- size |>
2   mutate(mean_sand_all = mean(total_sand))

# A tibble: 114 × 3
  plot  total_sand mean_sand_all
  <chr>     <dbl>        <dbl>
1 CSP01      50.1        39.6
2 CSP01      46.8        39.6
3 CSP01       46          39.6
4 CSP01      50.1        39.6
5 CSP01      53.8        39.6
6 CSP01      51.9        39.6
7 CSP08      49.6        39.6
8 CSP08      49.5        39.6
9 CSP08      49.2        39.6
10 CSP02     26.0        39.6
# i 104 more rows
```

Grouping via `.by = COLUMN:`

```
1 size <- size |>
2   mutate(mean_sand_plot = mean(total_sand),
3         .by = plot)
```

```
# A tibble: 114 × 3
  plot  total_sand mean_sand_plot
  <chr>     <dbl>        <dbl>
1 CSP01      50.1        49.8
2 CSP01      46.8        49.8
3 CSP01       46          49.8
4 CSP01      50.1        49.8
5 CSP01      53.8        49.8
6 CSP01      51.9        49.8
7 CSP08      49.6        49.4
8 CSP08      49.5        49.4
9 CSP08      49.2        49.4
# i 105 more rows
```

Overall mean calculated

Mean calculated for each group (i.e. plot)

Mutating by group (alternative)

`group_by()` and `ungroup()` are from `dplyr`*

```
1 group_by(data, column1, column2)
2 ungroup(data)
```



- `tidyverse` functions always start with `data`
- `group_by()` applies grouping according to specified `data` columns
- `ungroup()` removes grouping from `data`

* part of the `tidyverse`

Mutating by group (alternative)

`mutate()` without grouping

```
1 size <- size |>  
2   mutate(mean_sand_all = mean(total_sand))
```

```
# A tibble: 114 × 3  
  plot  total_sand mean_sand_all  
  <chr>    <dbl>        <dbl>  
1 CSP01      50.1        39.6  
2 CSP01      46.8        39.6  
3 CSP01       46          39.6  
4 CSP01      50.1        39.6  
5 CSP01      53.8        39.6  
6 CSP01      51.9        39.6  
7 CSP08      49.6        39.6  
8 CSP08      49.5        39.6  
9 CSP08      49.2        39.6  
10 CSP02     26.0        39.6  
# i 104 more rows
```

Overall mean calculated

Mutating by group (alternative)

mutate() without grouping

```
1 size <- size |>
2   mutate(mean_sand_all = mean(total_sand))
```

A tibble: 114 × 3
 plot total_sand mean_sand_all
 <chr> <dbl> <dbl>
1 CSP01 50.1 39.6
2 CSP01 46.8 39.6
3 CSP01 46 39.6
4 CSP01 50.1 39.6
5 CSP01 53.8 39.6
6 CSP01 51.9 39.6
7 CSP08 49.6 39.6
8 CSP08 49.5 39.6
9 CSP08 49.2 39.6
10 CSP02 26.0 39.6
i 104 more rows

Grouping via group_by():

```
1 size <- size |>
2   group_by(plot) |>
3   mutate(mean_sand_plot = mean(total_sand)) |>
4   ungroup()
```

A tibble: 114 × 3
 plot total_sand mean_sand_plot
 <chr> <dbl> <dbl>
1 CSP01 50.1 49.8
2 CSP01 46.8 49.8
3 CSP01 46 49.8
4 CSP01 50.1 49.8
5 CSP01 53.8 49.8
6 CSP01 51.9 49.8
7 CSP08 49.6 49.4
8 CSP08 49.5 49.4
i 106 more rows

Overall mean calculated

Mean calculated for each group (i.e. plot)

Mutating by group (alternative)

mutate() without grouping

```
1 size <- size |>
2   mutate(mean_sand_all = mean(total_sand))
```

A tibble: 114 × 3
 plot total_sand mean_sand_all
 <chr> <dbl> <dbl>
1 CSP01 50.1 39.6
2 CSP01 46.8 39.6
3 CSP01 46 39.6
4 CSP01 50.1 39.6
5 CSP01 53.8 39.6
6 CSP01 51.9 39.6
7 CSP08 49.6 39.6
8 CSP08 49.5 39.6
9 CSP08 49.2 39.6
10 CSP02 26.0 39.6
i 104 more rows

Overall mean calculated

Grouping via group_by():

```
1 size <- size |>
2   group_by(plot) |>
3   mutate(mean_sand_plot = mean(total_sand)) |>
4   ungroup()
```

A tibble: 114 × 3
 plot total_sand mean_sand_plot
 <chr> <dbl> <dbl>
1 CSP01 50.1 49.8
2 CSP01 46.8 49.8
3 CSP01 46 49.8
4 CSP01 50.1 49.8
5 CSP01 53.8 49.8
6 CSP01 51.9 49.8
7 CSP08 49.6 49.4
8 CSP08 49.5 49.4
9 CSP08 49.2 49.4
10 CSP02 26.0 49.4
i 106 more rows

Always remember to
ungroup() your data!

Mean calculated for each group (i.e. plot)



Artwork by @allison_horst

Your turn: Mutating by group

Add a column containing the mean amount of total silt *per plot*

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   ???
```

Too Easy?

Try both methods

Try grouping by multiple columns

Your turn: Mutating by group

Add a column containing the mean amount of total silt *per plot*

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   mutate(mean_silt = mean(total_silt), .by = plot)
```

```
# A tibble: 114 × 6
  plot  coarse_silt medium_silt fine_silt total_silt mean_silt
  <chr>     <dbl>      <dbl>     <dbl>      <dbl>      <dbl>
1 CSP01      14.1       11.2      8.17      33.5      29.5
2 CSP01      14.1       11.7      9.03      34.8      29.5
3 CSP01      10.3       9.51      7.47      27.3      29.5
4 CSP01       9.4        9.1       8.7       27.2      29.5
5 CSP01      9.79       8.79      7.29      25.9      29.5
6 CSP01      10.8       9.4       8.22      28.4      29.5
7 CSP08      16.3       9.55      6.23      32.1      32.4
8 CSP08      14.3       10.4      6.1       30.8      32.4
9 CSP08      15.1       11.5      7.56      34.2      32.4
10 CSP02     12.0       18.3      15.2      45.4      40.9
# i 104 more rows
```

Your turn: Mutating by group

Add a column containing the mean amount of total silt *per plot*

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   group_by(plot) |>
8   mutate(mean_silt = mean(total_silt)) |>
9   ungroup()
```

```
# A tibble: 114 × 6
  plot  coarse_silt medium_silt fine_silt total_silt mean_silt
  <chr>     <dbl>      <dbl>     <dbl>      <dbl>      <dbl>
1 CSP01      14.1       11.2      8.17      33.5      29.5
2 CSP01      14.1       11.7      9.03      34.8      29.5
3 CSP01      10.3       9.51      7.47      27.3      29.5
4 CSP01       9.4        9.1       8.7       27.2      29.5
5 CSP01      9.79       8.79      7.29      25.9      29.5
6 CSP01      10.8       9.4       8.22      28.4      29.5
7 CSP08      16.3       9.55      6.23      32.1      32.4
8 CSP08      14.3       10.4      6.1       30.8      32.4
9 CSP08      15.1       11.5      7.56      34.2      32.4
10 CSP02     12.0       18.3      15.2      45.4      40.9
# i 104 more rows
```

Your turn: Mutating by group

Group by multiple columns

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   mutate(mean_silt = mean(total_silt), .by = c(habitat, technician))
```

```
# A tibble: 114 × 7
  habitat technician coarse_silt medium_silt fine_silt total_silt mean_silt
  <chr>    <chr>        <dbl>       <dbl>      <dbl>      <dbl>      <dbl>
1 forest    Catharine     14.1        11.2      8.17      33.5      37.9
2 forest    Catharine     14.1        11.7      9.03      34.8      37.9
3 forest    Catharine     10.3        9.51      7.47      27.3      37.9
4 forest    Catharine      9.4        9.1       8.7       27.2      37.9
5 forest    Catharine     9.79        8.79      7.29      25.9      37.9
6 forest    Catharine     10.8        9.4       8.22      28.4      37.9
7 grassland Catharine     16.3        9.55      6.23      32.1      32.8
8 grassland Catharine     14.3        10.4      6.1       30.8      32.8
9 grassland Catharine     15.1        11.5      7.56      34.2      32.8
10 forest   Catharine     12.0        18.3      15.2      45.4      37.9
# i 104 more rows
```

Your turn: Mutating by group

Group by multiple columns

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   group_by(habitat, technician) |>
8   mutate(mean_silt = mean(total_silt)) |>
9   ungroup()
```

```
# A tibble: 114 × 7
  habitat technician coarse_silt medium_silt fine_silt total_silt mean_silt
  <chr>    <chr>        <dbl>       <dbl>      <dbl>      <dbl>      <dbl>
1 forest    Catharine     14.1        11.2      8.17      33.5      37.9
2 forest    Catharine     14.1        11.7      9.03      34.8      37.9
3 forest    Catharine     10.3        9.51      7.47      27.3      37.9
4 forest    Catharine      9.4        9.1       8.7       27.2      37.9
5 forest    Catharine     9.79        8.79      7.29      25.9      37.9
6 forest    Catharine     10.8        9.4       8.22      28.4      37.9
7 grassland Catharine     16.3        9.55      6.23      32.1      32.8
8 grassland Catharine     14.3        10.4      6.1       30.8      32.8
9 grassland Catharine     15.1        11.5      7.56      34.2      32.8
10 forest   Catharine     12.0        18.3      15.2      45.4      37.9
# i 104 more rows
```

Put it all together

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   mutate(mean_sand = mean(total_sand),
8         mean_silt = mean(total_silt),
9         .by = plot)
```

Put it all together

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt) |>
7   mutate(mean_sand = mean(total_sand),
8         mean_silt = mean(total_silt),
9         .by = plot)
```

Check it out

```
1 select(size, plot, depth, total_sand, total_silt, mean_sand, mean_silt)
# A tibble: 114 × 6
  plot    depth total_sand total_silt mean_sand mean_silt
  <chr> <dbl>     <dbl>      <dbl>     <dbl>     <dbl>
1 CSP01     4       50.1      33.5      49.8      29.5
2 CSP01    12       46.8      34.8      49.8      29.5
3 CSP01    35       46        27.3      49.8      29.5
4 CSP01    53       50.1      27.2      49.8      29.5
5 CSP01    83       53.8      25.9      49.8      29.5
6 CSP01   105       51.9      28.4      49.8      29.5
7 CSP08     10       49.6      32.1      49.4      32.4
8 CSP08     27       49.5      30.8      49.4      32.4
9 CSP08     90       49.2      34.2      49.4      32.4
10 CSP02    5        26.0      45.4      34.7      40.9
# i 104 more rows
```

Summarizing

Summarizing by group

`summarize()` is from `dplyr`*

```
1 summarize(data,  
2             column1 = expression1, column2 = expression2, ...,  
3             .by = c(group1, group2))
```



- `tidyverse` functions always start with `data`
- `summarize()` collapses `data`
- Creates new `columns`
- Columns filled according to `expression`

* part of the `tidyverse`

Summarizing by group

- Similar to `mutate()`, but **collapses** rows whereas `mutate()` repeats data

`mutate()`

```
1 size <- size |>
2     mutate(mean_sand = mean(total_sand), .by = plot)
3
4 select(size, plot, contains("sand"))

# A tibble: 114 × 6
  plot   coarse_sand medium_sand fine_sand total_sand mean_sand
  <chr>      <dbl>       <dbl>      <dbl>      <dbl>      <dbl>
1 CSP01       13.0        17.4      19.7      50.1      49.8
2 CSP01       10.7        16.9      19.2      46.8      49.8
3 CSP01       12.1        17.8      16.1       46        49.8
4 CSP01       17.6        18.2      14.3      50.1      49.8
5 CSP01       21.0        18.4      14.3      53.8      49.8
6 CSP01       19.0        18.4      14.4      51.9      49.8
7 CSP08       11.6        17.1      20.8      49.6      49.4
8 CSP08       15.4        16.2      17.8      49.5      49.4
9 CSP08       14.9        15.8      18.6      49.2      49.4
10 CSP02      8.75       8.64      8.66      26.0      34.7
# i 104 more rows
```

Repeated values

Summarizing by group

- Similar to `mutate()`, but **collapses** rows whereas `mutate()` repeats data

`summarize()`

```
1 size <- size |>
2   summarize(mean_sand = mean(total_sand), .by = plot)
3
4 size

# A tibble: 27 × 2
  plot    mean_sand
  <chr>     <dbl>
 1 CSP01     49.8
 2 CSP08     49.4
 3 CSP02     34.7
 4 CSP04     30.3
 5 CSP05     44.6
 6 CSP09     37.9
 7 CSP12     43.1
 8 CSP13     46.6
 9 CSP16     43.9
10 CSP17     41.2
# i 17 more rows
```

No repeated values **and**
drops unused columns

Summarizing by group

- Keep other id columns by adding them to `.by`
- Beware: think carefully about grouping variables!

```
1 size |>
2   summarize(mean_sand = mean(total_sand), .by = c(plot, depth))  
  
# A tibble: 114 × 3  
  plot    depth mean_sand  
  <chr>   <dbl>     <dbl>  
1 CSP01      4     50.1  
2 CSP01     12     46.8  
3 CSP01     35      46  
4 CSP01     53     50.1  
5 CSP01     83     53.8  
6 CSP01    105     51.9  
7 CSP08      10     49.6  
8 CSP08      27     49.5  
9 CSP08      90     49.2  
10 CSP02     5      26.0  
# i 104 more rows
```

`depth` is not a category, therefore not an appropriate grouping factor

Summarizing by group

- Use true groups of interest (e.g., Sex, Age)
- Or use factors which are on the same level (e.g., ID columns)

```
1 size |>
2   summarize(mean_sand = mean(total_sand), .by = c(plot, habitat))

# A tibble: 27 × 3
  plot    habitat  mean_sand
  <chr>   <chr>     <dbl>
1 CSP01  forest      49.8
2 CSP08  grassland   49.4
3 CSP02  forest      34.7
4 CSP04  forest      30.3
5 CSP05  grassland   44.6
6 CSP09  forest      37.9
7 CSP12  grassland   43.1
8 CSP13  grassland   46.6
9 CSP16  forest      43.9
10 CSP17 forest      41.2
# i 17 more rows
```

Better: `habitat` varies with `plot` (alternatively could have Joined later)

Summarizing by group

Summarizing is an excellent way to calculate statistics to describe your data

Statistic	Function(s)
sample sizes / total counts	<code>n()</code> *
means	<code>mean(x)</code>
standard deviations	<code>sd(x)</code>
standard errors	<code>sd(x) / sqrt(n())</code> **
total values	<code>sum(x)</code>

Where `x` is the column you want to calculate a summary statistic for
And no, `n()` is not missing an `x` 😊

* Special function, can only be used *inside* of `mutate()` or `summarize()`

** There is no standard error function in R, but we can calculate it by hand

Summarizing by group

`n()` is from `dplyr`*

```
1 n()
```

- Helper `tidyverse` function which **does NOT** start with data
- Returns row counts according to groups (if present)
- Can only be used *inside* `mutate()` or `summarize()`

For example...

```
1 size |>
2   summarize(samples_total = n(), .by = plot)
```

```
# A tibble: 27 × 2
  plot    samples_total
  <chr>      <int>
1 CSP01        6
2 CSP08        3
3 CSP02        7
4 CSP04        5
5 CSP05        5
6 CSP09        5
7 CSP12        5
8 CSP13        5
# i 19 more rows
```



Your Turn: Calculate summary statistics

For each plot and habitat, calculate

- standard error for `total_sand`
- mean for `total_silt`
- standard deviation for `total_silt`
- standard error for `total_silt`

Hint

Use `sd(x)/sqrt(n())` for standard error calculations

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt)
7
8 size_sum <- size |>
9   summarize(sample_size = n(),
10             mean_sand = mean(total_sand),
11             sd_sand = sd(total_sand),
12             se_sand = ???,
13             ???,
14             .by = c(plot, habitat))
```

Too Easy?

Can you recycle some of the calculated values into the next statistic?

Challenging

What if you only wanted to calculate the mean of each column... can you use the `across()` function to do this elegantly?

Your Turn: Calculate summary statistics

For each plot and habitat, calculate

- standard error for **total_sand**
- mean for **total_silt**
- standard deviation for **total_silt**
- standard error for **total_silt**

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt)
7
8 size_sum <- size |>
9   summarize(sample_size = n(),
10             mean_sand = mean(total_sand),
11             sd_sand = sd(total_sand),
12             se_sand = sd_sand / sqrt(sample_size),
13             mean_silt = mean(total_silt),
14             sd_silt = sd(total_silt),
15             se_silt = sd_silt / sqrt(sample_size),
16             .by = c(plot, habitat))
```

Your Turn: Calculate summary statistics

Check your work

```
1 size_sum

# A tibble: 27 × 9
  plot habitat sample_size mean_sand sd_sand se_sand mean_silt sd_silt se_silt
  <chr> <chr>     <int>    <dbl>   <dbl>    <dbl>    <dbl>   <dbl>
1 CSP01 forest       6     49.8    2.96    1.21    29.5    3.72    1.52
2 CSP08 grassland    3     49.4    0.176   0.102    32.4    1.73    0.998
3 CSP02 forest       7     34.7    10.8     4.06    40.9    4.29    1.62
4 CSP04 forest       5     30.3    2.18     0.973   43.0    0.544   0.243
5 CSP05 grassland    5     44.6    5.52     2.47    31.8    1.81    0.811
6 CSP09 forest       5     37.9    2.98     1.33    38.4    1.17    0.524
7 CSP12 grassland    5     43.1    3.24     1.45    37.5    2.03    0.908
8 CSP13 grassland    5     46.6    6.91     3.09    29.4    3.67    1.64
9 CSP16 forest       5     43.9    14.9     6.67    38.3    17.4    7.78
10 CSP17 forest      3     41.2    3.46     2.00    32.4    1.62    0.937
# i 17 more rows
```

Transposing

Let's talk about tidy data

Upcoming illustrations from the [Openscapes](#) blog *Tidy Data for reproducibility, efficiency, and collaboration*
by Julia Lowndes and Allison Horst

TIDY DATA

“TIDY DATA is a standard way of mapping the meaning of a dataset to its structure.”

—HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

each row an observation

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

Tidy Data

id	name	colour	age	mass (lb)
1	floof	grey	10	7
1	floof	grey	12	7.5
2	max	black	1	5
2	max	black	2	6
3	cat	orange	5	10
3	cat	orange	7	12

Long data

One measurement per row

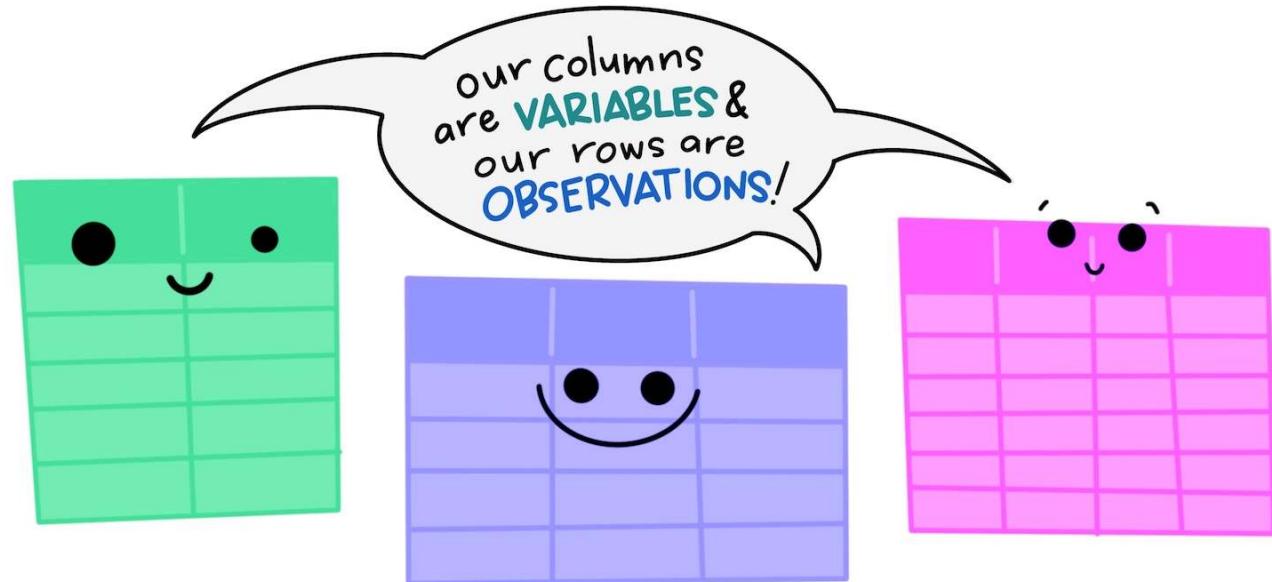
Un-tidy

id	name	colour	age1	mass1	age2	mass2
1	floof	grey	10	7	12	7.5
2	max	black	1	5	2	6
3	cat	orange	5	10	7	12

Wide data

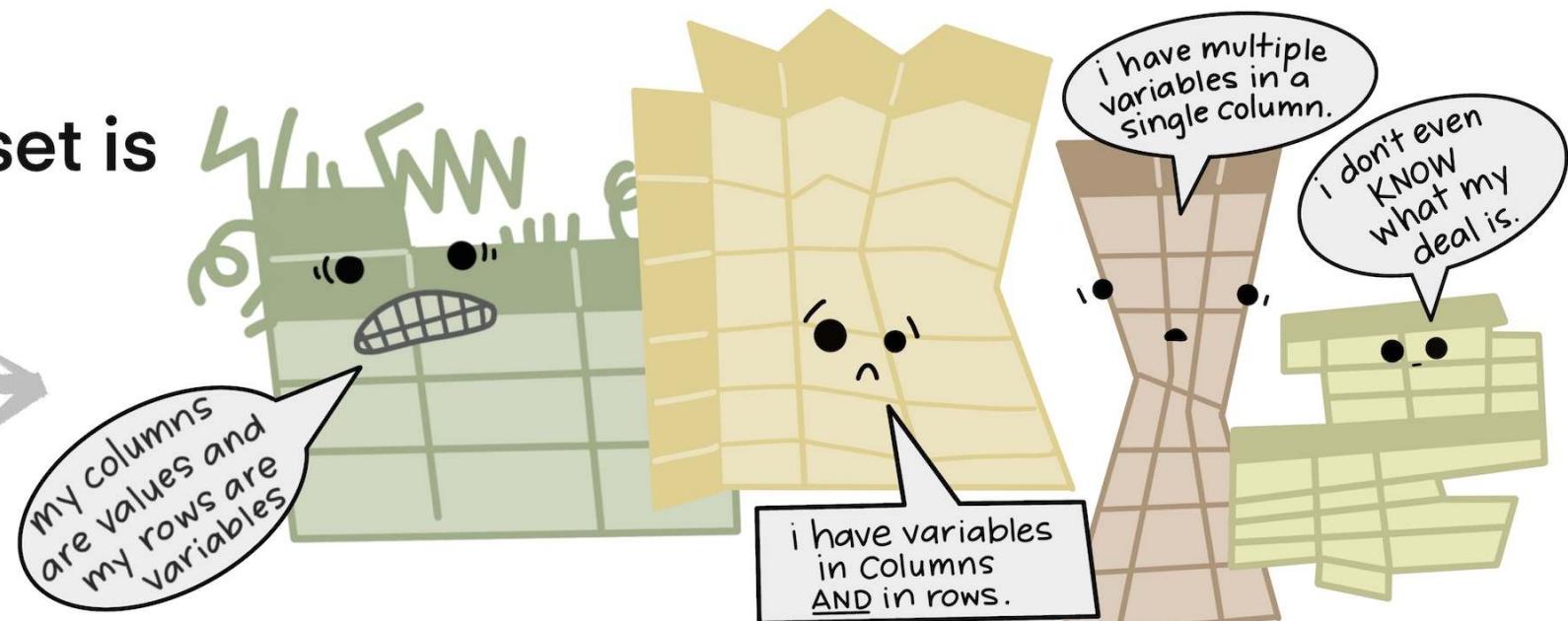
Several measurements
per row

The standard structure of
tidy data means that
“tidy datasets are all alike...”

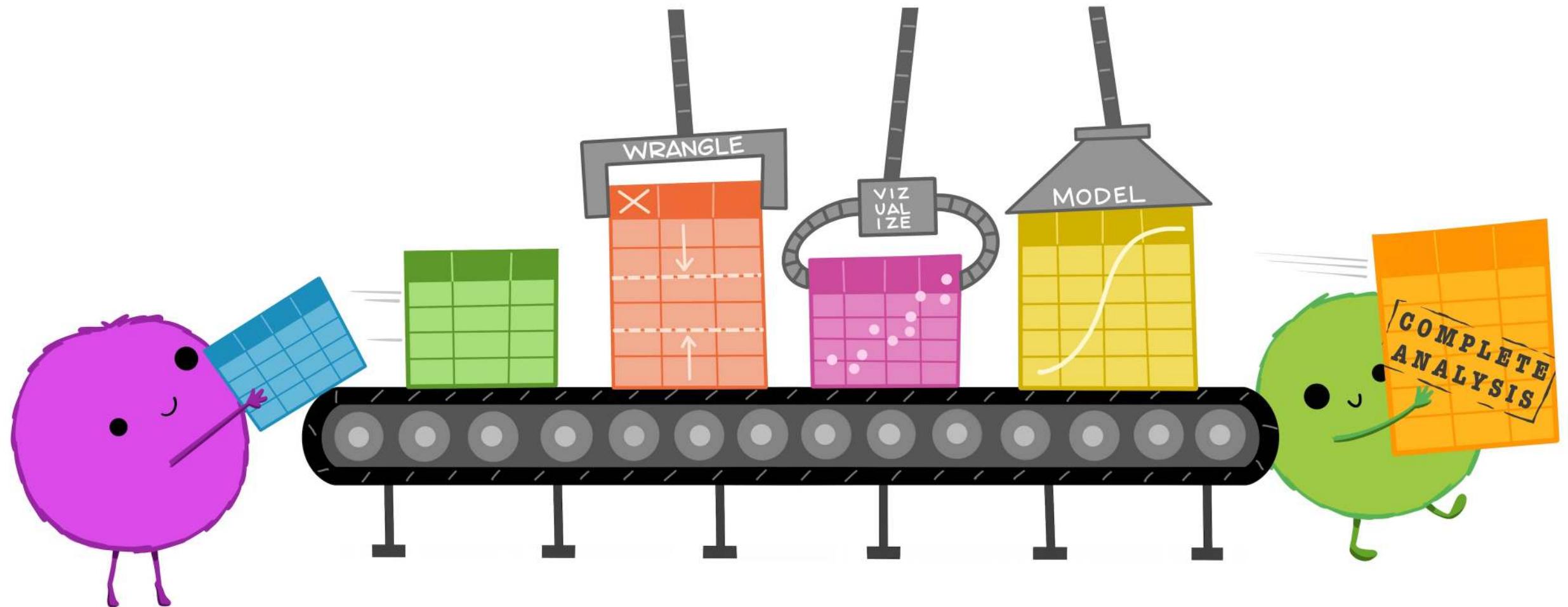


“...but every messy dataset is
messy in its own way.”

—HADLEY WICKHAM



Why do we care?



Why do we care?

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766
Tuberculosis cases per year per country		

How would you plot this untidy data as the number of cases by country for each year?

```
1 ggplot(data = table4a, aes(x = ???, y = ???)) +  
2 ???
```

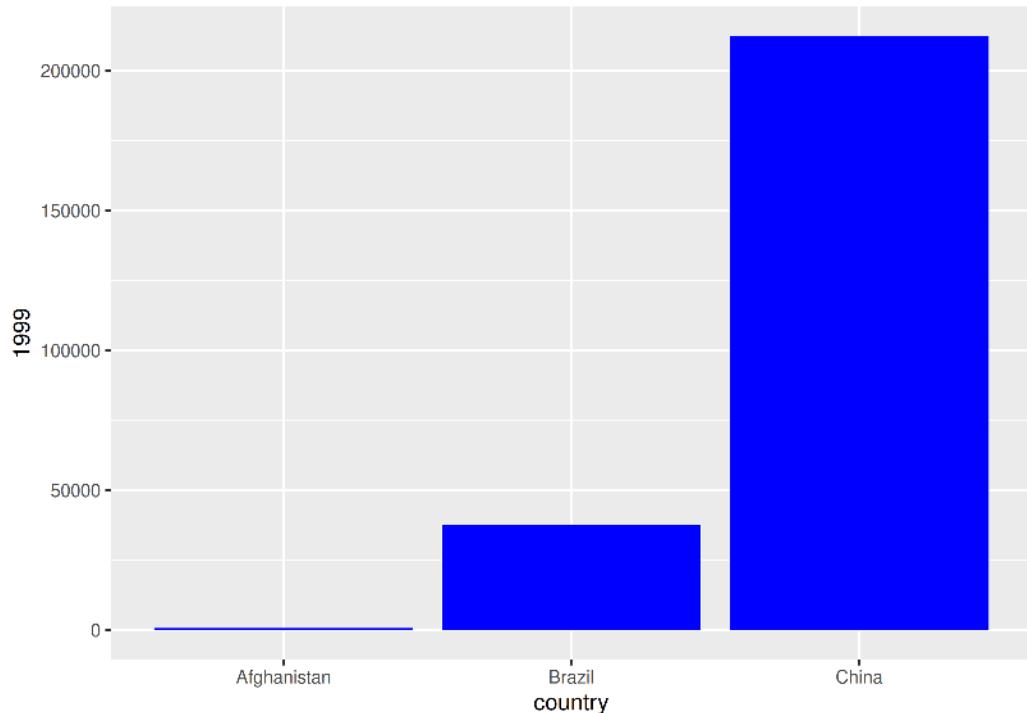
Note

- `table4a` is a built-in data frame
- Type `table4a` in the console to take a look
- Type `?table4a` to pull up the help file with information

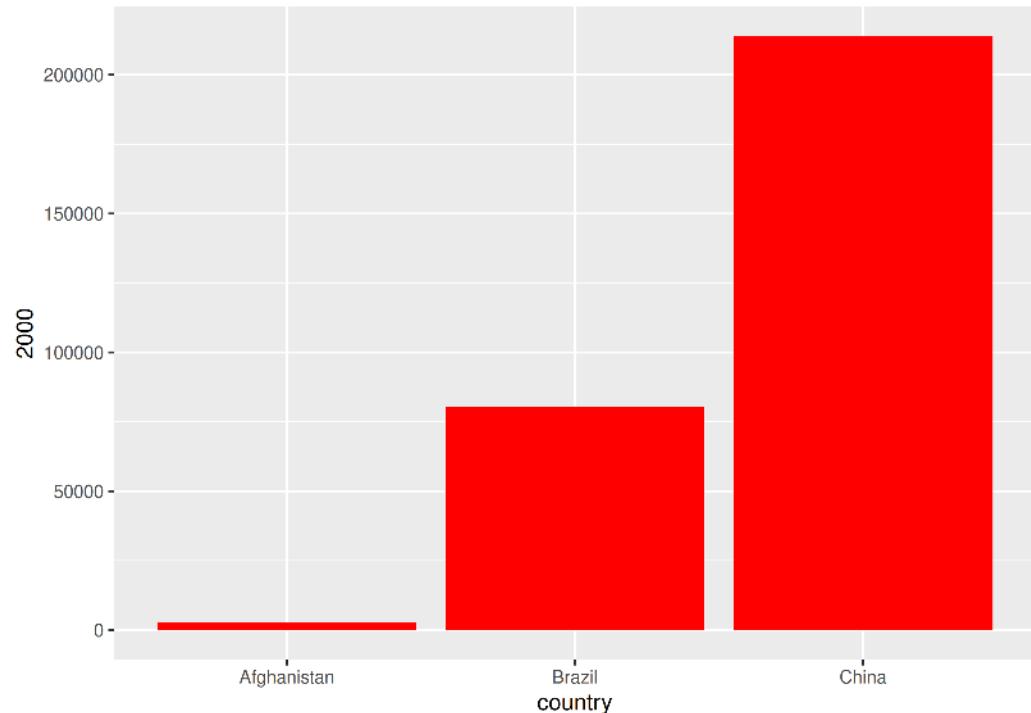
Why do we care?

With un-tidy data

```
1 ggplot(data = table4a, aes(x = country, y = `1999`)) +  
2   geom_bar(stat = "identity", fill = "blue")
```



```
1 ggplot(data = table4a, aes(x = country, y = `2000`)) +  
2   geom_bar(stat = "identity", fill = "red")
```



We have to plot it twice!

Why do we care?

With tidy data

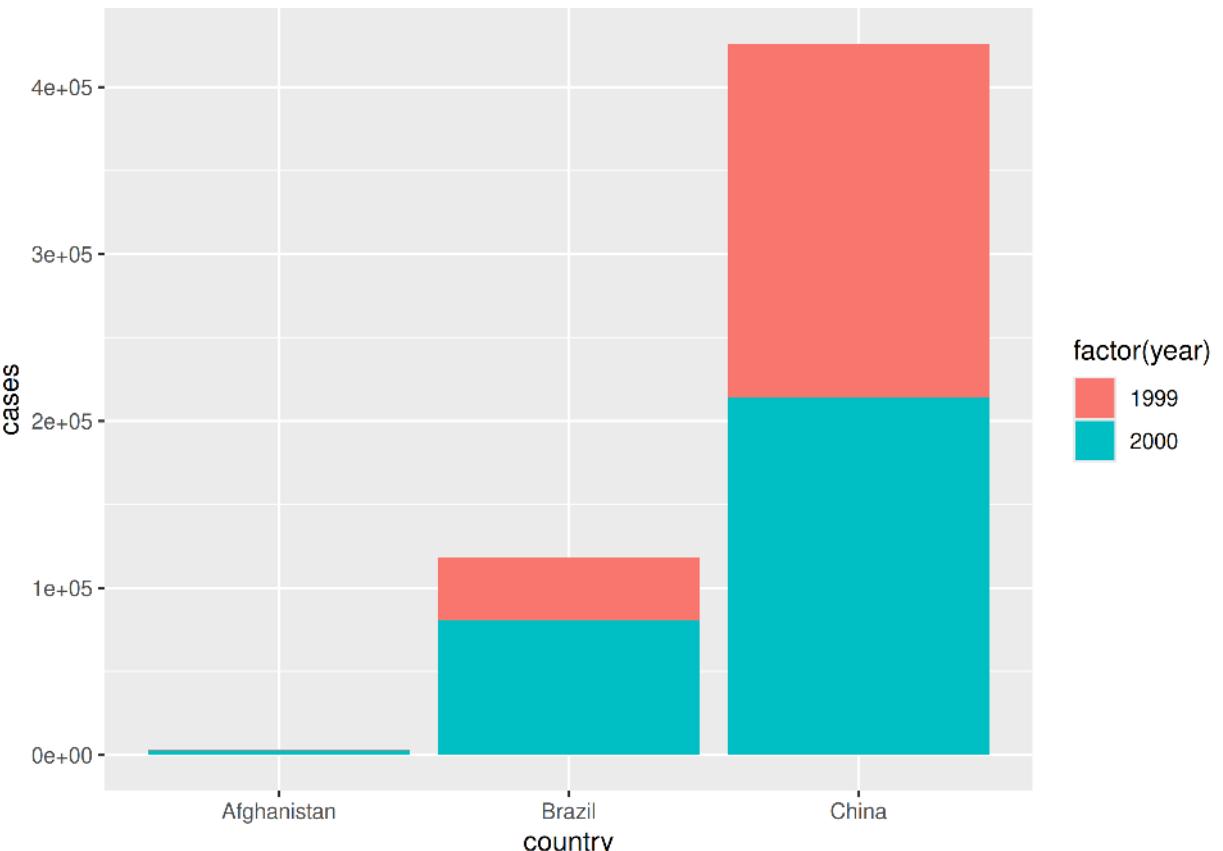
country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Why do we care?

With tidy data

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

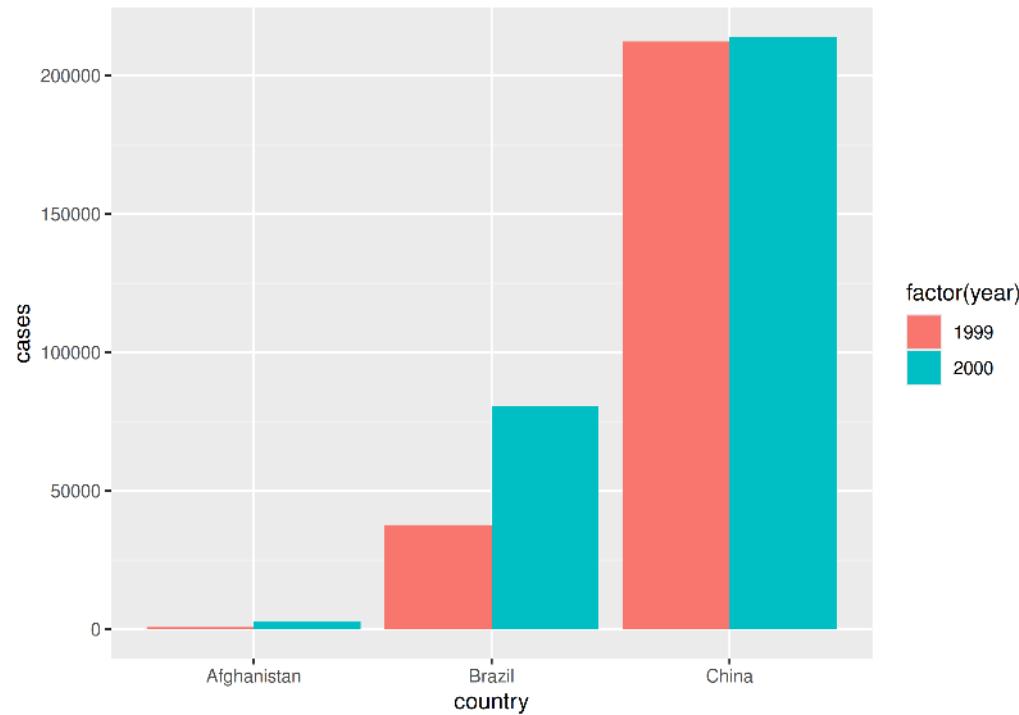
```
1 ggplot(data = table1, aes(x = country, y = cases, fill = factor(year))) +  
2   geom_bar(stat = "identity")
```



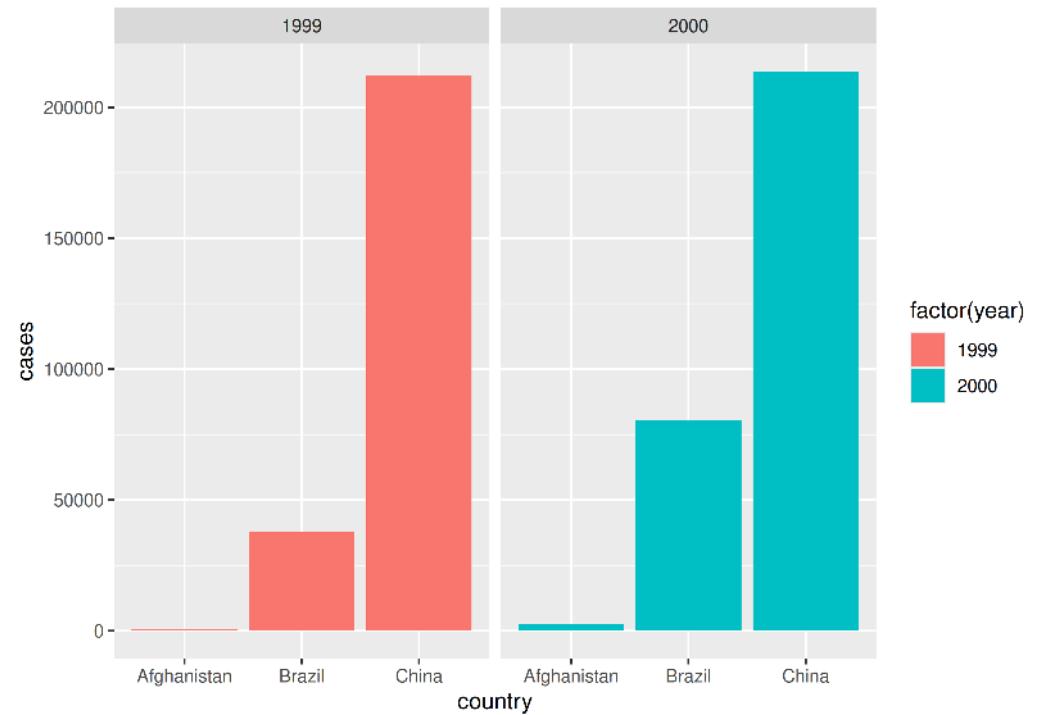
Why do we care?

With tidy data

```
1 ggplot(data = table1,
2         aes(x = country, y = cases, fill = factor(year))) +
3   geom_bar(stat = "identity", position = "dodge")
```



```
1 ggplot(data = table1,
2         aes(x = country, y = cases, fill = factor(year))) +
3   geom_bar(stat = "identity") + facet_wrap(~year)
```



Going long

`pivot_longer()`

country	year	cases	country	1999	2000
Afghanistan	1999	745	Afghanistan	745	2666
Afghanistan	2000	2666	Brazil	37737	80488
Brazil	1999	37737	China	212258	213766
Brazil	2000	80488			
China	1999	212258			
China	2000	213766			

table4

The diagram illustrates the transformation of a wide table into a long table using the `pivot_longer()` function. It shows two tables side-by-side. The left table, labeled "table4", has columns for country, year, and cases. The right table has columns for country, 1999, and 2000. Arrows point from the "cases" values in the left table to the "1999" and "2000" columns in the right table, indicating that the "cases" values are being mapped to the corresponding years in the pivoted table.

Going long

From wide ...

```
# A tibble: 114 × 6
  plot   depth coarse_silt medium_silt fine_silt total_silt
  <chr> <dbl>      <dbl>       <dbl>     <dbl>      <dbl>
1 CSP01    4        14.1       11.2     8.17     33.5
2 CSP01   12        14.1       11.7     9.03     34.8
3 CSP01   35        10.3       9.51     7.47     27.3
4 CSP01   53        9.4        9.1      8.7      27.2
5 CSP01   83        9.79      8.79     7.29     25.9
6 CSP01  105        10.8       9.4      8.22     28.4
7 CSP08   10        16.3       9.55     6.23     32.1
8 CSP08   27        14.3       10.4      6.1      30.8
9 CSP08   90        15.1       11.5     7.56     34.2
10 CSP02   5        12.0       18.3     15.2     45.4
11 CSP02   11        10.7       18.3     14.3     43.3
12 CSP02   36        10.7       19.0     14.4     44.1
13 CSP02   56        11.1       18.0     13.7     42.8
14 CSP02   70        11.2       16.8     13.0      41
15 CSP02   78        9.97      13.8     11.0     34.7
# i 99 more rows
```

Going long

From wide ...

```
# A tibble: 114 × 6
  plot   depth coarse_silt medium_silt fine_silt total_silt
  <chr> <dbl>      <dbl>       <dbl>     <dbl>      <dbl>
1 CSP01    4        14.1       11.2      8.17     33.5
2 CSP01   12        14.1       11.7      9.03     34.8
3 CSP01   35       10.3       9.51      7.47     27.3
4 CSP01   53        9.4        9.1       8.7      27.2
5 CSP01   83       9.79       8.79      7.29     25.9
6 CSP01  105       10.8       9.4       8.22     28.4
7 CSP08   10       16.3       9.55      6.23     32.1
8 CSP08   27       14.3       10.4       6.1      30.8
9 CSP08   90       15.1       11.5      7.56     34.2
10 CSP02   5        12.0       18.3      15.2     45.4
11 CSP02   11       10.7       18.3      14.3     43.3
12 CSP02   36       10.7       19.0      14.4     44.1
13 CSP02   56       11.1       18.0      13.7     42.8
14 CSP02   70       11.2       16.8      13.0      41
15 CSP02   78       9.97      13.8      11.0     34.7
# i 99 more rows
```

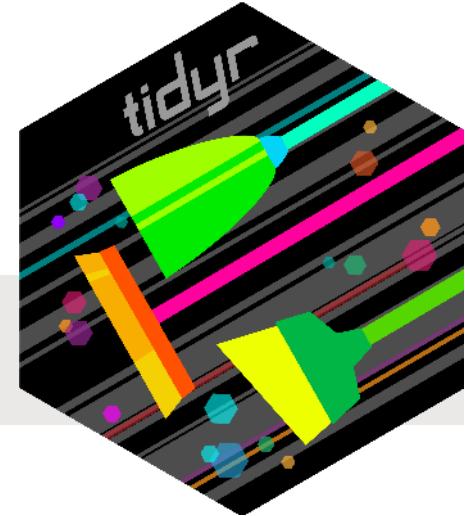
... to long

```
# A tibble: 456 × 4
  plot   depth type      amount
  <chr> <dbl> <chr>     <dbl>
1 CSP01    4 coarse_silt 14.1
2 CSP01    4 medium_silt 11.2
3 CSP01    4 fine_silt   8.17
4 CSP01    4 total_silt  33.5
5 CSP01   12 coarse_silt 14.1
6 CSP01   12 medium_silt 11.7
7 CSP01   12 fine_silt   9.03
8 CSP01   12 total_silt  34.8
9 CSP01   35 coarse_silt 10.3
10 CSP01   35 medium_silt 9.51
11 CSP01   35 fine_silt   7.47
12 CSP01   35 total_silt  27.3
13 CSP01   53 coarse_silt 9.4
14 CSP01   53 medium_silt 9.1
15 CSP01   53 fine_silt   8.7
16 CSP01   53 total_silt  27.2
17 CSP01   83 coarse_silt 9.79
18 CSP01   83 medium_silt 8.79
19 CSP01   83 fine_silt   7.29
20 CSP01   83 total_silt  25.9
# i 436 more rows
```

Going long

`pivot_longer()` is from `tidyverse`*

```
1 pivot_longer(data, cols = c(column1, column2),  
2                 names_to = "new_categorical_column",  
3                 values_to = "new_numerical_column")
```



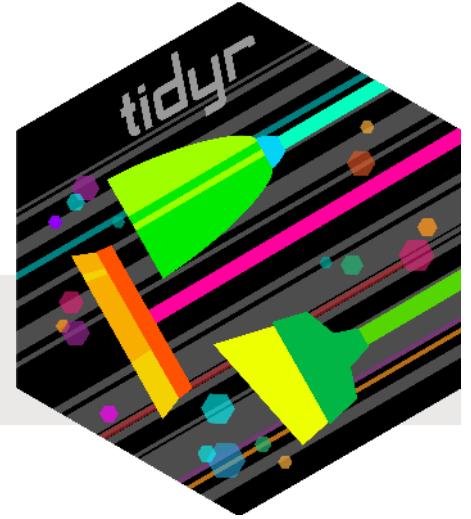
- `tidyverse` functions always start with `data`
- Takes columns and converts to long data
- Column names (`column1` and `column2`) go into “`new_categorical_column`”
- Column values (*contents* of `column1` and `column2`) go into “`new_numerical_column`”

* part of the `tidyverse`

Going long

`pivot_longer()` is from `tidyverse`*

```
1 pivot_longer(data, cols = c(column1, column2),  
2                 names_to = "new_categorical_column",  
3                 values_to = "new_numerical_column")
```



In our example:

- `data = size`
- `cols = c(-plot, -depth, -habitat, -technician, -date)`
 - Here, easiest to exclude columns
- `names_to = "type"`
- `values_to = "amount"`

* part of the `tidyverse`

Going long

```
1 size_long <- pivot_longer(size, cols = c(-plot, -depth, -habitat, -technician, -date),  
2                               names_to = "type", values_to = "amount")
```

```
# A tibble: 1,026 × 7  
  plot   depth habitat technician date       type      amount  
  <chr>  <dbl> <chr>    <chr>     <date>    <chr>      <dbl>  
1 CSP01      4 forest   Catharine 2009-04-23 coarse_sand 13.0  
2 CSP01      4 forest   Catharine 2009-04-23 medium_sand 17.4  
3 CSP01      4 forest   Catharine 2009-04-23 fine_sand  19.7  
4 CSP01      4 forest   Catharine 2009-04-23 coarse_silt 14.1  
5 CSP01      4 forest   Catharine 2009-04-23 medium_silt 11.2  
6 CSP01      4 forest   Catharine 2009-04-23 fine_silt   8.17  
7 CSP01      4 forest   Catharine 2009-04-23 clay        16.3  
8 CSP01      4 forest   Catharine 2009-04-23 total_sand 50.1  
9 CSP01      4 forest   Catharine 2009-04-23 total_silt 33.5  
10 CSP01     12 forest  Catharine 2009-04-23 coarse_sand 10.7  
11 CSP01     12 forest  Catharine 2009-04-23 medium_sand 16.9  
12 CSP01     12 forest  Catharine 2009-04-23 fine_sand  19.2  
# i 1,014 more rows
```

Your turn: Lengthen data

- We'll first create a summary dataset for sand variables

```
1 sand_sum <- read_csv("data/grain_size2.csv") |>
2   mutate(total_sand = coarse_sand + medium_sand + fine_sand) |>
3   summarize(sample_size = n(),
4             mean_sand = mean(total_sand),
5             sd_sand = sd(total_sand),
6             se_sand = sd_sand / sqrt(sample_size),
7             .by = plot)
8
9 sand_sum
```

```
# A tibble: 27 × 5
  plot sample_size mean_sand sd_sand se_sand
  <chr>     <int>     <dbl>    <dbl>    <dbl>
1 CSP01       6      49.8     2.96    1.21
2 CSP08       3      49.4     0.176   0.102
3 CSP02       7      34.7     10.8    4.06
4 CSP04       5      30.3     2.18    0.973
5 CSP05       5      44.6     5.52    2.47
6 CSP09       5      37.9     2.98    1.33
7 CSP12       5      43.1     3.24    1.45
8 CSP13       5      46.6     6.91    3.09
9 CSP16       5      43.9     14.9    6.67
10 CSP17      3      41.2     3.46    2.00
# i 17 more rows
```

Your turn: Lengthen data

- Gather all variables except `plot` and `sample_size` into a long format

```
1 sand_long <- pivot_longer(sand_sum,  
2                               cols = ???,  
3                               names_to = ???,  
4                               values_to = ???)
```

Your turn: Lengthen data

- Gather all variables except `plot` and `sample_size` into a long format

```
1 sand_long <- pivot_longer(sand_sum,
2                               cols = contains("sand"),
3                               names_to = "type",
4                               values_to = "amount")
```

```
# A tibble: 81 × 4
  plot    sample_size type      amount
  <chr>      <int> <chr>     <dbl>
1 CSP01        6 mean_sand 49.8
2 CSP01        6 sd_sand   2.96
3 CSP01        6 se_sand   1.21
4 CSP08        3 mean_sand 49.4
5 CSP08        3 sd_sand   0.176
6 CSP08        3 se_sand   0.102
7 CSP02        7 mean_sand 34.7
8 CSP02        7 sd_sand   10.8
9 CSP02        7 se_sand   4.06
10 CSP04       5 mean_sand 30.3
11 CSP04       5 sd_sand   2.18
12 CSP04       5 se_sand   0.973
13 CSP05       5 mean_sand 44.6
14 CSP05       5 sd_sand   5.52
15 CSP05       5 se_sand   2.47
# i 66 more rows
```

Your turn: Lengthen data

- Gather all variables except `plot` and `sample_size` into a long format

```
1 sand_long <- pivot_longer(sand_sum,
2                               cols = contains("sand"),
3                               names_to = "type",
4                               values_to = "amount")
```

```
# A tibble: 81 × 4
  plot sample_size type      amount
  <chr>     <int> <chr>    <dbl>
1 CSP01        6 mean_sand 49.8
2 CSP01        6 sd_sand   2.96
3 CSP01        6 se_sand   1.21
4 CSP08        3 mean_sand 49.4
5 CSP08        3 sd_sand   0.176
6 CSP08        3 se_sand   0.102
7 CSP02        7 mean_sand 34.7
8 CSP02        7 sd_sand   10.8
9 CSP02        7 se_sand   4.06
10 CSP04       5 mean_sand 30.3
11 CSP04       5 sd_sand   2.18
12 CSP04       5 se_sand   0.973
13 CSP05       5 mean_sand 44.6
14 CSP05       5 sd_sand   5.52
15 CSP05       5 se_sand   2.47
# i 66 more rows
```

Remember tidy selectors!

Could also use...

- `cols = ends_with("sand")`
- `cols = c(mean_sand, sd_sand, se_sand)`
- `cols = c(-plot, -sample_size)`

Going wide

`pivot_wider()`

The diagram illustrates the transformation of a long-format data frame (table2) into a wide-format data frame. The source data frame (table2) has columns: country, year, key, and value. The target data frame has columns: country, year, cases, and population. Arrows show the mapping from the source columns to the target columns.

country	year	key	value	country	year	cases	population
Afghanistan	1999	cases	745	Afghanistan	1999	745	19987071
Afghanistan	1999	population	19987071		2000	2666	20595360
Afghanistan	2000	cases	2666	Brazil	1999	37737	172006362
Afghanistan	2000	population	20595360		2000	80488	174504898
Brazil	1999	cases	37737	China	1999	212258	1272915272
Brazil	1999	population	172006362		2000	213766	1280428583
Brazil	2000	cases	80488				
Brazil	2000	population	174504898				
China	1999	cases	212258				
China	1999	population	1272915272				
China	2000	cases	213766				
China	2000	population	1280428583				

table2

Going wide

From long ...

```
# A tibble: 456 × 4
  plot   depth type      amount
  <chr> <dbl> <chr>     <dbl>
1 CSP01     4 coarse_silt 14.1
2 CSP01     4 medium_silt 11.2
3 CSP01     4 fine_silt   8.17
4 CSP01     4 total_silt  33.5
5 CSP01    12 coarse_silt 14.1
6 CSP01    12 medium_silt 11.7
7 CSP01    12 fine_silt   9.03
8 CSP01    12 total_silt  34.8
9 CSP01    35 coarse_silt 10.3
10 CSP01   35 medium_silt  9.51
11 CSP01   35 fine_silt   7.47
12 CSP01   35 total_silt  27.3
13 CSP01   53 coarse_silt  9.4
14 CSP01   53 medium_silt  9.1
15 CSP01   53 fine_silt   8.7
16 CSP01   53 total_silt  27.2
17 CSP01   83 coarse_silt  9.79
18 CSP01   83 medium_silt  8.79
19 CSP01   83 fine_silt   7.29
20 CSP01   83 total_silt  25.9
# i 436 more rows
```

Going wide

From long ...

```
# A tibble: 456 × 4
  plot   depth type      amount
  <chr>  <dbl> <chr>     <dbl>
1 CSP01    4 coarse_silt 14.1
2 CSP01    4 medium_silt 11.2
3 CSP01    4 fine_silt   8.17
4 CSP01    4 total_silt  33.5
5 CSP01   12 coarse_silt 14.1
6 CSP01   12 medium_silt 11.7
7 CSP01   12 fine_silt   9.03
8 CSP01   12 total_silt  34.8
9 CSP01   35 coarse_silt 10.3
10 CSP01   35 medium_silt 9.51
11 CSP01   35 fine_silt  7.47
12 CSP01   35 total_silt 27.3
13 CSP01   53 coarse_silt 9.4
14 CSP01   53 medium_silt 9.1
15 CSP01   53 fine_silt  8.7
16 CSP01   53 total_silt  27.2
17 CSP01   83 coarse_silt 9.79
18 CSP01   83 medium_silt 8.79
19 CSP01   83 fine_silt  7.29
20 CSP01   83 total_silt  25.9
# i 436 more rows
```

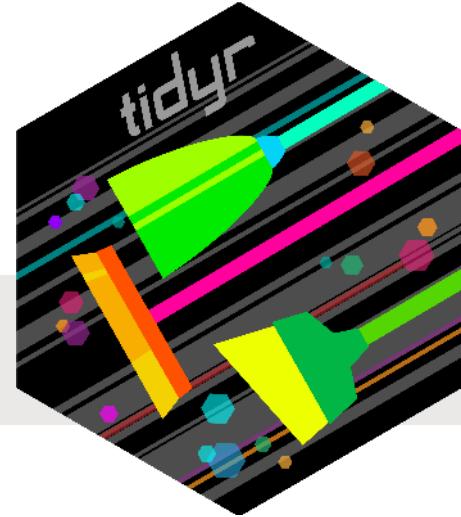
... to wide

```
# A tibble: 114 × 6
  plot   depth coarse_silt medium_silt fine_silt total_silt
  <chr>  <dbl>     <dbl>       <dbl>     <dbl>       <dbl>
1 CSP01    4        14.1      11.2      8.17      33.5
2 CSP01   12        14.1      11.7      9.03      34.8
3 CSP01   35        10.3      9.51      7.47      27.3
4 CSP01   53        9.4       9.1       8.7       27.2
5 CSP01   83        9.79      8.79      7.29      25.9
6 CSP01  105        10.8      9.4       8.22      28.4
7 CSP08    10        16.3      9.55      6.23      32.1
8 CSP08    27        14.3      10.4      6.1       30.8
9 CSP08    90        15.1      11.5      7.56      34.2
10 CSP02   5         12.0      18.3      15.2      45.4
11 CSP02   11        10.7      18.3      14.3      43.3
12 CSP02   36        10.7      19.0      14.4      44.1
13 CSP02   56        11.1      18.0      13.7      42.8
14 CSP02   70        11.2      16.8      13.0      41
15 CSP02   78        9.97     13.8      11.0      34.7
# i 99 more rows
```

Going wide

`pivot_wider()` is from `tidyverse`*

```
1 pivot_wider(data,
2   names_from = existing_categorical_column,
3   values_from = existing_numerical_column)
```



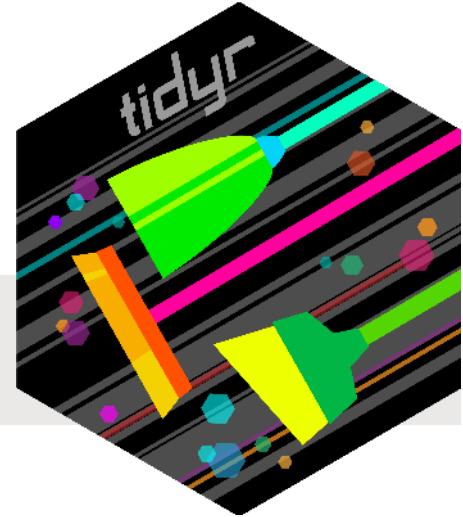
- `tidyverse` functions always start with `data`
- Takes columns and converts to wide data
- Values in `existing_categorical_column` become column names
- Values in `existing_numerical_column` become column contents

* part of the `tidyverse`

Going wide

`pivot_wider()` is from `tidyverse`*

```
1 pivot_wider(data,
2   names_from = existing_categorical_column,
3   values_from = existing_numerical_column)
```



- `tidyverse` functions always start with `data`
- Takes columns and converts to wide data
- Values in `existing_categorical_column` become column names
- Values in `existing_numerical_column` become column contents

In our example:

- `data = size_long`
- `names_from = type`
- `values_from = amount`

* part of the `tidyverse`

Going wide

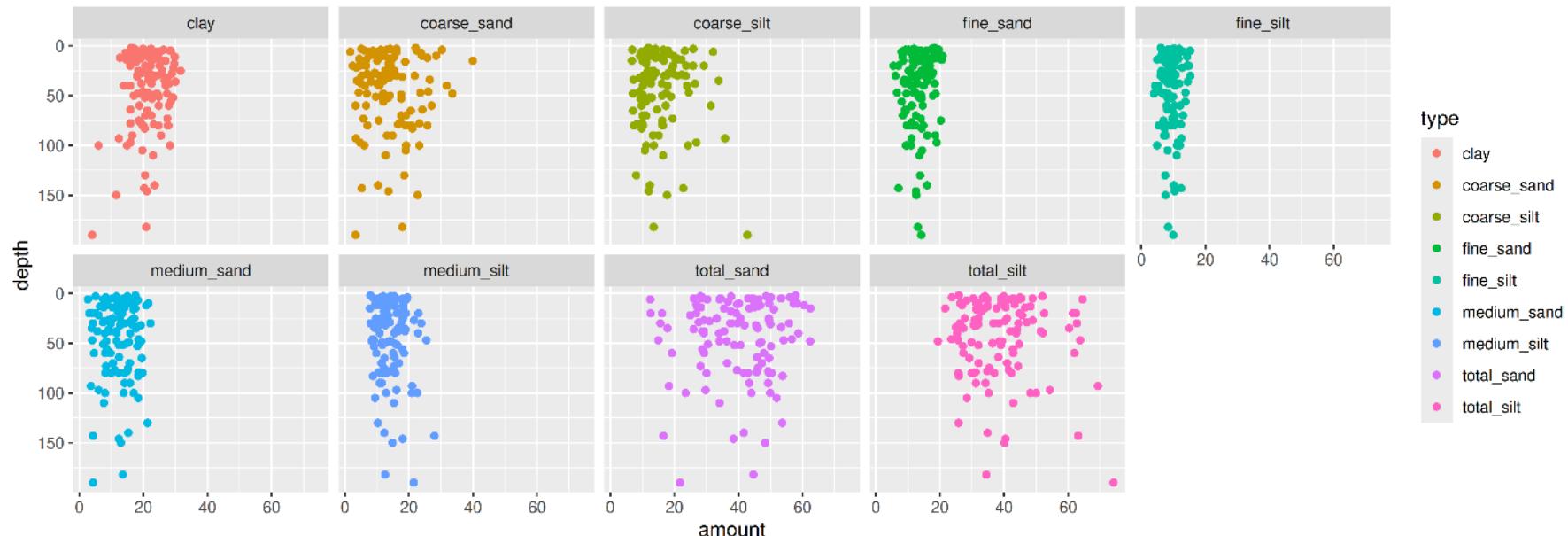
```
1 size_wide <- pivot_wider(size_long, names_from = type, values_from = amount)
```

```
1 # A tibble: 114 × 14
2   plot  depth habitat technician date    coarse_sand medium_sand fine_sand coarse_silt medium_silt fine
3   <chr> <dbl> <chr>     <chr>     <date>      <dbl>       <dbl>       <dbl>       <dbl>       <dbl>
4   1 CSP01     4 forest Catharine 2009-04-23    13.0       17.4       19.7       14.1       11.2
5   2 CSP01    12 forest Catharine 2009-04-23    10.7       16.9       19.2       14.1       11.7
6   3 CSP01    35 forest Catharine 2009-04-23    12.1       17.8       16.1       10.3       9.51
7   4 CSP01    53 forest Catharine 2009-04-23    17.6       18.2       14.3        9.4       9.1
8   5 CSP01    83 forest Catharine 2009-04-23    21.0       18.4       14.3       9.79       8.79
9   6 CSP01   105 forest Catharine 2009-04-23    19.0       18.4       14.4       10.8       9.4
10  7 CSP08     10 grassland Catharine 2009-03-02   11.6       17.1       20.8       16.3       9.55
11  8 CSP08     27 grassland Catharine 2009-03-02   15.4       16.2       17.8       14.3       10.4
12  9 CSP08    90 grassland Catharine 2009-03-02   14.9       15.8       18.6       15.1       11.5
13 10 CSP02      5 forest Catharine 2009-05-06    8.75       8.64       8.66      12.0      18.3
14 # i 104 more rows
```

Again: Why transpose?

Figures: Long data are great for graphing

```
1 size_long <- pivot_longer(size, cols = c(-plot, -depth, -technician, -habitat, -date),  
2                               names_to = "type", values_to = "amount")  
3  
4 ggplot(data = size_long, aes(y = depth, x = amount, colour = type)) +  
5   geom_point() +  
6   scale_y_reverse() +  
7   facet_wrap(~ type, nrow = 2)
```



Again: Why transpose?

Figures: Take it to the next step

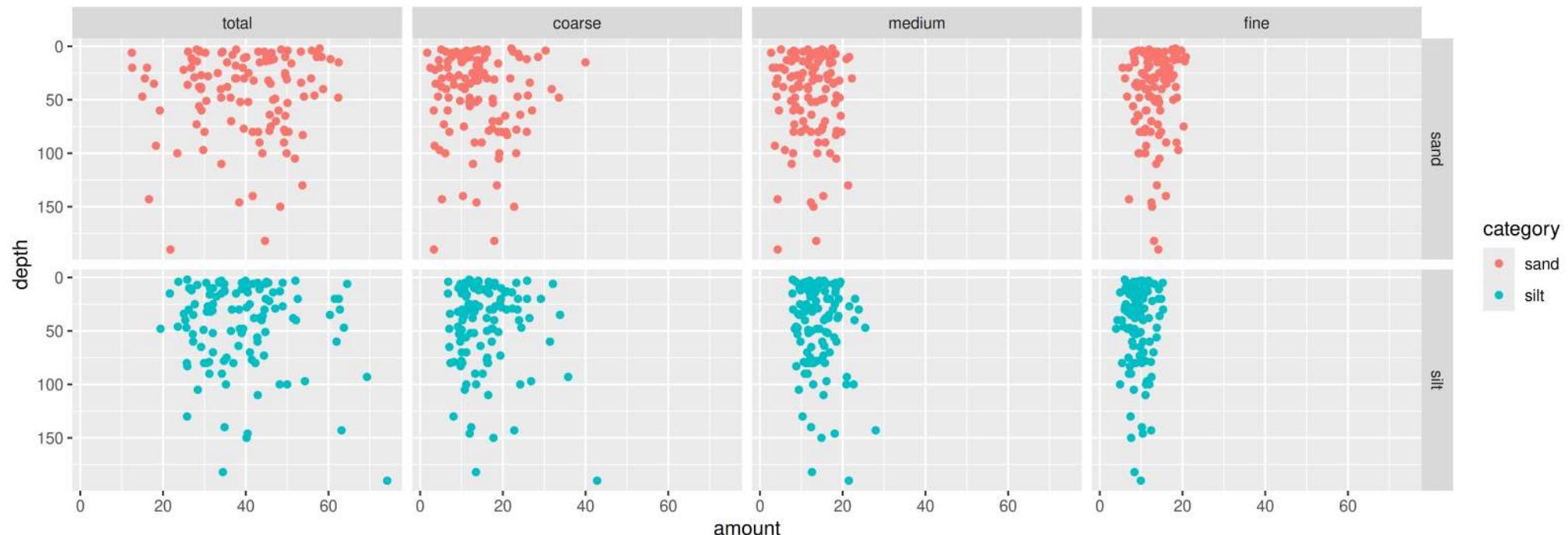
```
1 size <- read_csv("data/grain_size2.csv") |>
2   left_join(meta, by = "plot") |>
3   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
4         total_silt = coarse_silt + medium_silt + fine_silt)
5
6 size_long <- pivot_longer(size, cols = c(-plot, -depth, -technician, -habitat, -date, -clay),
7                           names_to = c("size", "category"), values_to = "amount",
8                           names_sep = "_") |>
9   mutate(size = factor(size, levels = c("total", "coarse", "medium", "fine")))
```

```
# A tibble: 912 × 9
  plot    depth    clay habitat technician date      size category amount
  <chr>   <dbl>  <dbl> <chr>     <chr>    <date>    <fct>   <chr>    <dbl>
1 CSP01      4    16.3 forest Catharine 2009-04-23 coarse sand  13.0
2 CSP01      4    16.3 forest Catharine 2009-04-23 medium sand 17.4
3 CSP01      4    16.3 forest Catharine 2009-04-23 fine   sand 19.7
4 CSP01      4    16.3 forest Catharine 2009-04-23 coarse silt 14.1
5 CSP01      4    16.3 forest Catharine 2009-04-23 medium silt 11.2
6 CSP01      4    16.3 forest Catharine 2009-04-23 fine   silt  8.17
7 CSP01      4    16.3 forest Catharine 2009-04-23 total  sand 50.1
8 CSP01      4    16.3 forest Catharine 2009-04-23 total  silt 33.5
9 CSP01     12    18.4 forest Catharine 2009-04-23 coarse sand 10.7
10 CSP01     12    18.4 forest Catharine 2009-04-23 medium sand 16.9
# i 902 more rows
```

Again: Why transpose?

Figures: Take it to the next step

```
1 ggplot(data = size_long,
2         aes(y = depth, x = amount, colour = category)) +
3   geom_point() +
4   scale_y_reverse() +
5   facet_grid(category ~ size)
```



Again: Why transpose?

Anlyses: Linear models `lm(y ~ x, data)`

Use `pivot_longer()` in analysis where grouping variables are important

- i.e., do amounts of different size classes differ with depth? (need size classes in “type” column)

```
1 lm(amount ~ type + depth, data = size_long)
```

Use `pivot_wider()` in analyses where each variable must be in it's own column

- i.e., does the amount of sand differ with depth? (need size classes in separate columns)

```
1 lm(total_sand ~ depth, data = size_wide)
```

Again: Why transpose?

Analyses: Linear models `lm(y ~ x, data)`

Use `pivot_longer()` in analysis where grouping variables are important

- i.e., do amounts of different size classes differ with depth? (need size classes in “type” column)

```
1 lm(amount ~ type + depth, data = size_long)
```

Use `pivot_wider()` in analyses where each variable must be in its own column

- i.e., does the amount of sand differ with depth? (need size classes in separate columns)

```
1 lm(total_sand ~ depth, data = size_wide)
```

If you can't figure out how to plot or analyse your data, they probably need to be transposed

Your Turn: Transpose for plotting

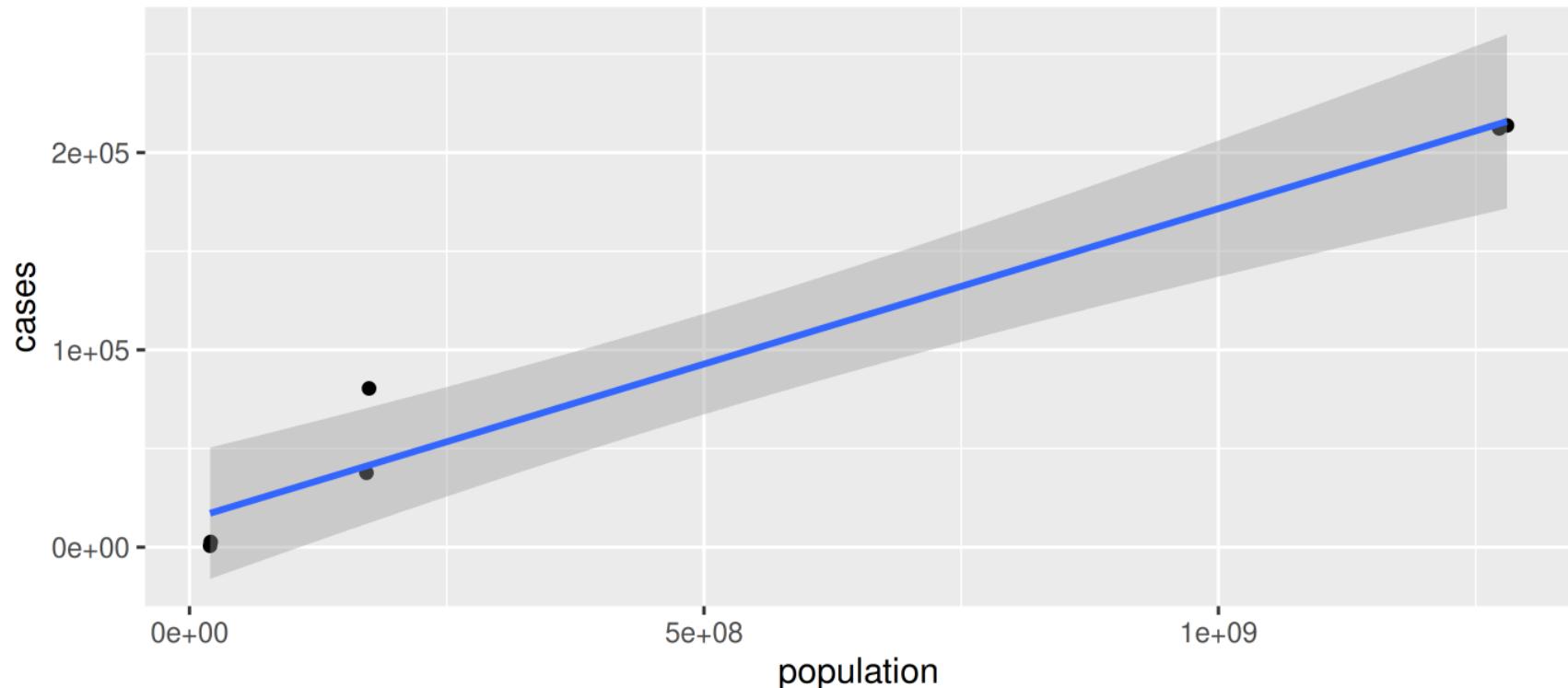
Plot the number of Tuberculosis `cases` vs. the `population` in data frame `table2`

```
1 temp <- pivot_???(table2, ???)
2
3 ggplot(data = temp, ???) +
4 ???
```

Your Turn: Transpose for plotting

Plot the number of Tuberculosis cases (`cases`) vs. the `population` in data frame `table2`

```
1 temp <- pivot_wider(table2, names_from = "type", values_from = "count")
2
3 ggplot(data = temp, aes(x = population, y = cases)) +
4   geom_point() +
5   stat_smooth(method = "lm")
```



Put it all together

```
1 meta <- read_csv("data/grain_meta.csv")
2
3 size <- read_csv("data/grain_size2.csv") |>
4   left_join(meta, by = "plot") |>
5   mutate(total_sand = coarse_sand + medium_sand + fine_sand,
6         total_silt = coarse_silt + medium_silt + fine_silt)
7
8 size_sum <- size |>
9   summarize(sample_size = n(),
10             total_sand = sum(total_sand),
11             mean_sand = mean(total_sand),
12             sd_sand = sd(total_sand),
13             se_sand = sd_sand / sqrt(sample_size),
14             total_silt = sum(total_silt),
15             mean_silt = mean(total_silt),
16             sd_silt = sd(total_silt),
17             se_silt = sd_silt / sqrt(sample_size),
18             .by = c(plot, habitat))
19
20 size_long <- size |>
21   pivot_longer(cols = c(-plot, -depth, -technician, -habitat, -date, -clay),
22                 values_to = "amount", names_to = c("size", "category"), names_sep = "_") |>
23   mutate(size = factor(size, levels = c("total", "coarse", "medium", "fine")))
```

Put it all together

Save your data

```
1 write_csv(size, "datasets/size_total.csv")
2 write_csv(size_sum, "datasets/size_summary.csv")
3 write_csv(size_long, "datasets/size_long.csv")
```

Keep yourself organized

- Keep your R-created data in a **different** folder from your ‘raw’ data
- If you have a lot going on, split your work into several scripts, and number the both the scripts AND the data sets produced:
 - 1_cleaned.csv
 - 2_summarized.csv
 - 3_graphing.csv

Wrapping up: Common mistakes

`select()` doesn't work

- You may have the `MASS` package loaded, it also has a `select()` function
- Make sure you loaded `tidyverse` or `dplyr` packages
- Try using `dplyr::select()`

Wrapping up: Common mistakes

`select()` doesn't work

- You may have the `MASS` package loaded, it also has a `select()` function
- Make sure you loaded `tidyverse` or `dplyr` packages
- Try using `dplyr::select()`

I can't figure out how to `pivot_wider()` my data in the way I want it

- Sometimes you need to `pivot_longer()` before you can widen it

Wrapping up: Common mistakes

`select()` doesn't work

- You may have the `MASS` package loaded, it also has a `select()` function
- Make sure you loaded `tidyverse` or `dplyr` packages
- Try using `dplyr::select()`

I can't figure out how to `pivot_wider()` my data in the way I want it

- Sometimes you need to `pivot_longer()` before you can widen it

`mutate()` is giving me weird results

- Is your data grouped when it shouldn't be? (try using `ungroup()`)

Wrapping up: Common mistakes

`select()` doesn't work

- You may have the `MASS` package loaded, it also has a `select()` function
- Make sure you loaded `tidyverse` or `dplyr` packages
- Try using `dplyr::select()`

I can't figure out how to `pivot_wider()` my data in the way I want it

- Sometimes you need to `pivot_longer()` before you can widen it

`mutate()` is giving me weird results

- Is your data grouped when it shouldn't be? (try using `ungroup()`)

Joins use `by`, `mutate/summarizes` use `.by`

Wrapping up: Common mistakes

I get a warning when I join data sets

- Can be because multiple joins
- Can be because mismatched factor levels
 - If the category levels in one data frame do not match the other data frame
 - They will be transformed to character
 - If that's a problem, use `as.factor()` to turn them back

Wrapping up: Further reading

- R for Data Science
 - [Chapter 3: Data transformation](#)
 - [Chapter 5: Data tidying](#)
 - [Chapter 19: Joins](#)
- RStudio Data Manipulation with `dplyr`
 - Or Help > Cheatsheets > Data Transformation with dplyr