

Summarizing and Transforming Data in R

Saving you time and sanity

country	year	cases	population
Afghanistan	1999	17745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	172006362
Brazil	2000	80488	174604898
China	1999	212258	127291272
China	2000	213766	128042583

variables

country	year	cases	population
Afghanistan	1999	17745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	172006362
Brazil	2000	80488	174604898
China	1999	212258	127291272
China	2000	213766	128042583

observations

country	year	cases	population
Afghanistan	99	7.5	19987071
Afghanistan	00	2666	20595360
Brazil	99	31737	172006362
Brazil	00	80488	174604898
China	99	212258	127291272
China	00	213766	128042583

values

Types of Modifications

1. Subset

- Subset by groups (i.e., rows)
- Subset by variables (i.e., columns)

2. Joining data sets

3. Creating new columns

- Creating categories
- Column calculations
- By group

4. Summarize existing columns

- Summarizing by group

5. Transpose

- Going between **wide** and **long** data formats
- Transposing for analysis
- Transposing for visualizations

Getting ready

Using packages:

```
library(tidyverse)
library(skimr)
```

Using data sets:

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

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

```
## # A tibble: 114 x 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
## # ... with 106 more rows
```

skim() our data

```
## — Data Summary —————
```

```
##                               Values
```

```
## Name                          size
```

```
## Number of rows                114
```

```
## Number of columns              9
```

```
## -----
```

```
## Column type frequency:
```

```
##   character                    1
```

```
##   numeric                      8
```

```
## -----
```

```
## Group variables                None
```

```
##
```

```
## — Variable type: character —————
```


```
##   skim_variable n_missing complete_rate   min   max empty n_unique whitespace
```


```
## 1 plot          0              1       5     5     0      27           0
```


```
##
```


```
## — Variable type: numeric —————
```


```
##   skim_variable n_missing complete_rate  mean    sd    p0    p25    p50    p75    p100 hist
```


```
## 1 depth          0              1  45.2  40.5    2    13    33    68.8  190  
```

```
## 2 coarse_sand     0              1  14.0   7.52   1.71   8.05  13.1   18.8   40.0  
```









```
## 3 medium_sand     0              1  12.4   4.83   2.7    8.44  12.7   16.2   22.2  
```

```
## 4 fine_sand       0              1  13.2   3.46   5.52  10.9   13.0   15.5   20.8  
```

```
## 5 coarse_silt     0              1  15.7   6.65   6.73  10.7   14.1   18.1   42.8  
```

```
## 6 medium_silt     0              1  14.1   4.09   7.85  11.2   13.0   16.4   28.0  
```

skim() our data

```
##
## -- Variable type: character -----
##   skim_variable n_missing complete_rate   min   max empty n_unique whitespace
## 1 plot           0             1       5     5     0      27           0
##
## -- Variable type: numeric -----
##   skim_variable n_missing complete_rate  mean    sd    p0    p25   p50    p75   p100 hist
## 1 depth         0             1  45.2  40.5   2    13    33    68.8  190  
## 2 coarse_sand   0             1  14.0   7.52  1.71  8.05  13.1   18.8  40.0  
## 3 medium_sand   0             1  12.4   4.83  2.7   8.44  12.7   16.2  22.2  
## 4 fine_sand     0             1  13.2   3.46  5.52  10.9  13.0   15.5  20.8  
## 5 coarse_silt   0             1  15.7   6.65  6.73  10.7  14.1   18.1  42.8  
## 6 medium_silt   0             1  14.1   4.09  7.85  11.2  13.0   16.4  28.0  
## 7 fine_silt     0             1   9.21  2.47  3.94  7.45  8.70   10.6  15.3  
## 8 clay          0             1  21.3   5.01  4     18.1  21.2   24.8  31.6  
```

Subsetting

By rows and column

Subsetting: By rows

filter() (**tidyverse** function, specifically from **dplyr** package)

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

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

Subsetting: By rows

Subset by group (i.e., by categorical value)

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

```
## # A tibble: 9 x 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
```


Subsetting: By rows

Subset by group (i.e., by categorical value)

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

```
## # A tibble: 9 x 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:

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

Subsetting: By rows

Subset by measures (i.e., by numerical value)

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

```
## # A tibble: 9 x 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

Tangent: 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%

Tangent: 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%

Single comparisons

```
1 < 2  
1 == 2  
1 != 2
```

Multiple comparisons

```
1 == c(1, 2, 1, "apple")  
"apple" %in% c(1, 2, 1, "apple")  
c(1, 2, 1, "apple") %in% "apple"  
  
fruit <- c("apple", "pear", "orange")  
fruit %in% c("apple", "pear")  
fruit == "apple" | fruit == "pear"
```

Your turn! Give it a try

Subsetting: By rows

Which values are greater than 100 OR less than 4?

```
size$depth > 140 | size$depth < 4
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [55] TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE
## [91] FALSE FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE
```

Return only rows with TRUE

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

Subsetting: By rows

Subset by combination

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

```
## # A tibble: 2 x 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

Subsetting: By rows

Subset by combination

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

```
## # A tibble: 2 x 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

Equivalent

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

Separate arguments in **filter** act like
AND (&)

Subsetting: By columns

select() (**tidyverse** function, specifically from **dplyr** package)

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

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

Subsetting: By columns

Subset by variable (i.e., column)

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

```
## # A tibble: 114 x 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
## # ... with 110 more rows
```

Subsetting: By columns

Subset by variable (i.e., column)

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

```
## # A tibble: 114 x 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
## # ... with 110 more rows
```

Using helper functions

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

```
## # A tibble: 114 x 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
## # ... with 110 more rows
```

Subsetting: By columns

Subset by variable (i.e., column)

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

```
## # A tibble: 114 x 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
## # ... with 110 more rows
```

Using helper functions

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

```
## # A tibble: 114 x 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
## # ... with 110 more rows
```

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

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

Subsetting: By columns

Put it all together

```
size %>%  
  filter(depth > 100,  
         plot %in% c("CSP13", "CSP25")) %>%  
  select(plot, depth, ends_with("sand"))
```

```
## # A tibble: 2 x 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
```

Subsetting: By columns

Put it all together

```
size %>%  
  filter(depth > 100,  
         plot %in% c("CSP13", "CSP25")) %>%  
  select(plot, depth, ends_with("sand"))
```

```
## # A tibble: 2 x 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

```
size_sub_sand <- size %>%  
  filter(depth > 100,  
         plot %in% c("CSP13", "CSP25")) %>%  
  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**

```
size <- read_csv("./data/grain_size2.csv") %>%  
  filter(???) %>%  
  select(???)
```

All particle values are percentages (depth is cm)

Extra Challenge

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

Joining/Merging

Joining data sets

Two data sets

- Measurements
- Metadata

Plot	Date	# birds
A	2018-05-01	1
A	2018-06-01	1
A	2018-07-01	2
B	2018-05-01	3
B	2018-06-01	4
B	2018-07-01	9

Plot	Vegetation Density
A	50
B	76

Joining data sets

Two data sets

- Measurements
- Metadata

Plot	Date	# birds
A	2018-05-01	1
A	2018-06-01	1
A	2018-07-01	2
B	2018-05-01	3
B	2018-06-01	4
B	2018-07-01	9

Plot	Vegetation Density
A	50
B	76

Joining them together

- Duplicate metadata to line up with measurements

Plot	Date	# birds	Vegetation Density
A	2018-05-01	1	50
A	2018-06-01	1	50
A	2018-07-01	2	50
B	2018-05-01	3	76
B	2018-06-01	4	76
B	2018-07-01	9	76

Joining data sets

Index or Metadata

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

```
## # A tibble: 6 x 4  
##   plot  habitat  technician date  
##   <chr> <chr>      <chr>      <date>  
## 1 CSP01 forest    Catharine 2008-12-28  
## 2 CSP02 clearcut Catharine 2008-11-19  
## 3 CSP03 grassland Jason      2009-04-10  
## 4 CSP04 grassland Catharine 2009-02-17  
## 5 CSP05 clearcut Catharine 2009-05-19  
## 6 CSP06 grassland Jason      2008-12-02
```

Measurements

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

```
## # A tibble: 6 x 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  
## # ... with 3 more variables: medium_silt <dbl>, fine_silt <dbl>,  
## #   clay <dbl>
```

Joining data sets

Index or Metadata

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

```
## # A tibble: 6 x 4  
##   plot  habitat  technician date  
##   <chr> <chr>      <chr>      <date>  
## 1 CSP01 forest    Catharine 2008-12-28  
## 2 CSP02 clearcut Catharine 2008-11-19  
## 3 CSP03 grassland Jason      2009-04-10  
## 4 CSP04 grassland Catharine 2009-02-17  
## 5 CSP05 clearcut Catharine 2009-05-19  
## 6 CSP06 grassland Jason      2008-12-02
```

Measurements

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

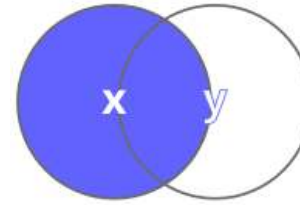
```
## # A tibble: 6 x 9  
##   plot  depth coarse_sand medium_sand fine_sand coarse_silt  
##   <chr> <dbl>      <dbl>      <dbl>      <dbl>      <dbl>  
## 1 CSP01     4        13.0        17.4        19.7        14.1  
## 2 CSP01    12        10.7        16.9        19.2        14.1  
## 3 CSP01    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  
## # ... with 3 more variables: medium_silt <dbl>, fine_silt <dbl>,  
## #   clay <dbl>
```

Types of Join: Which rows to keep?

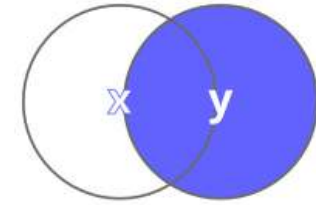
`left_join(x, y)`

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

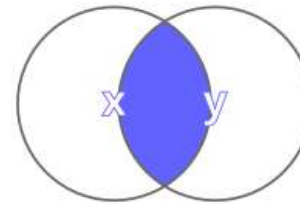
`left_join(x, y)`



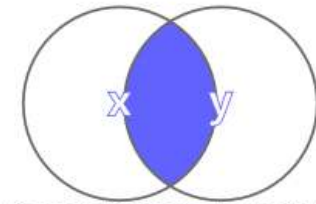
`right_join(x, y)`



`inner_join(x, y)`

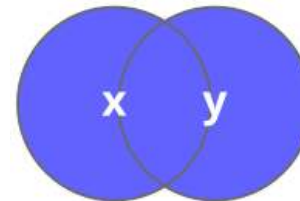


`semi_join(x, y)`

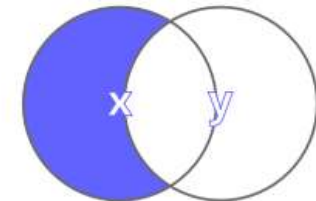


(never duplicate rows of x)

`full_join(x, y)`



`anti_join(x, y)`

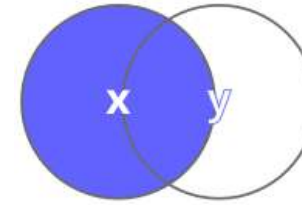


Types of Join: Which rows to keep?

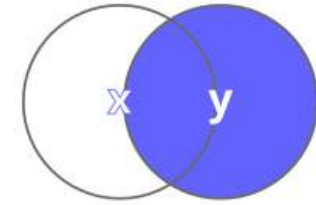
left_join(x, y)

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

left_join(x, y)



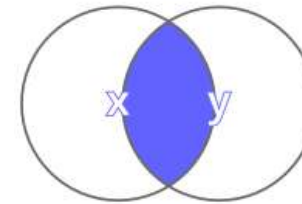
right_join(x, y)



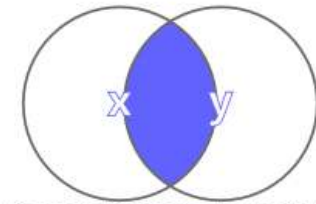
right_join(x, y)

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

inner_join(x, y)

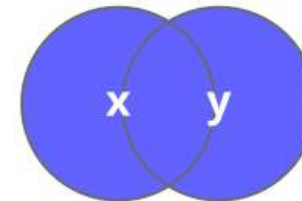


semi_join(x, y)

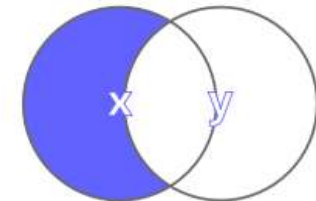


(never duplicate rows of x)

full_join(x, y)



anti_join(x, y)



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**

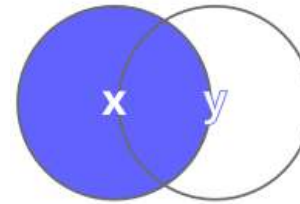
right_join(x, y)

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

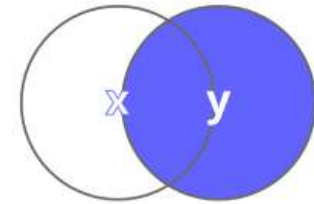
inner_join(x, y)

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

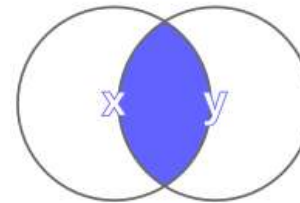
left_join(x, y)



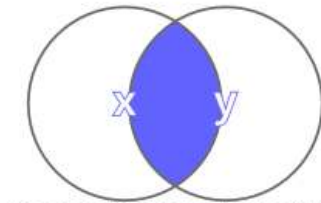
right_join(x, y)



inner_join(x, y)

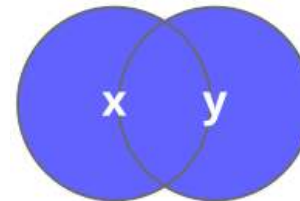


semi_join(x, y)

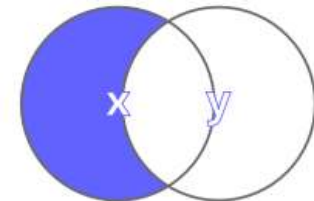


(never duplicate rows of x)

full_join(x, y)



anti_join(x, y)



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**

right_join(x, y)

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

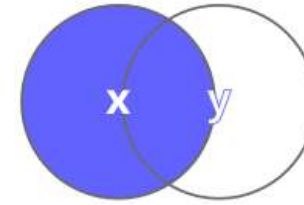
inner_join(x, y)

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

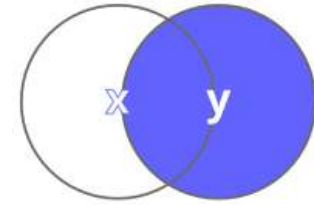
full_join(x, y)

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

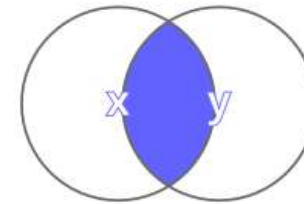
left_join(x, y)



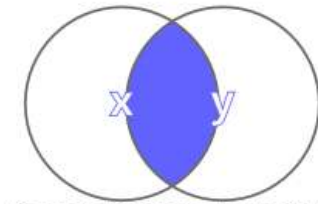
right_join(x, y)



inner_join(x, y)

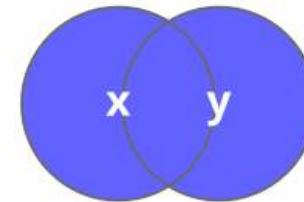


semi_join(x, y)

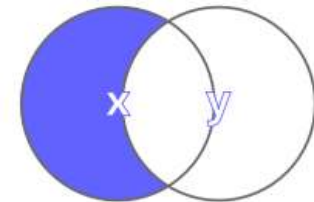


(never duplicate rows of x)

full_join(x, y)



anti_join(x, y)



Joining data sets

left_join() (**tidyverse** function, specifically from **dplyr** package)

(applies to other joins as well)

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

- **tidyverse** functions always start with **data**
- Here, also reference second **data_to_join**
- **by** refers **columns** in **data** and **data_to_join** used to join

Joining data sets

Keep all measurements, only keep meta if we have a measurement

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

Joining data sets

Keep all measurements, only keep meta if we have a measurement

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

OR

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

Joining data sets

Keep all measurements, only keep meta if we have a measurement

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

OR

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

```
## # A tibble: 6 x 12
##   plot  habitat technician date          depth coarse_sand medium_sand fine_sand coarse_silt medium_silt
##   <chr> <chr>    <chr>      <date>         <dbl>      <dbl>      <dbl>    <dbl>      <dbl>      <dbl>
## 1 CSP01 forest Catharine 2008-12-28      4        13.0       17.4     19.7       14.1       11.2
## 2 CSP01 forest Catharine 2008-12-28     12        10.7       16.9     19.2       14.1       11.7
## 3 CSP01 forest Catharine 2008-12-28     35        12.1       17.8     16.1       10.3        9.51
## 4 CSP01 forest Catharine 2008-12-28     53        17.6       18.2     14.3        9.4        9.1
## 5 CSP01 forest Catharine 2008-12-28     83        21.0       18.4     14.3        9.79       8.79
## 6 CSP01 forest Catharine 2008-12-28    105        19.0       18.4     14.4       10.8        9.4
## # ... with 2 more variables: fine_silt <dbl>, clay <dbl>
```

For more information see R for Data Science [Chapter 13.4 Mutating joins](#)

Creating columns with mutate()



Creating new columns

mutate() (**tidyverse** function, specifically from **dplyr** package)

```
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**

Creating new columns

R base

```
size <- read_csv("./data/grain_size2.csv")
size$total_sand <- size$coarse_sand +
  size$medium_sand +
  size$fine_sand
```

tidyverse

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

Creating new columns

R base

```
size <- read_csv("./data/grain_size2.csv")
size$total_sand <- size$coarse_sand +
  size$medium_sand +
  size$fine_sand
```

tidyverse

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

Either way

```
## # A tibble: 6 x 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
```

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

Tangent: Vectorized

Vectorized functions run in parallel across vectors

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

For example, try the following:

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

size$coarse_sand[1:5]
size$medium_sand[1:5]

size$coarse_sand[1:5] + size$medium_sand[1:5]
```

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, try the following:

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

size$coarse_sand[1:5]
size$medium_sand[1:5]

size$coarse_sand[1:5] + size$medium_sand[1:5]
```

For example

```
sum(a, a)
sum(size$coarse_sand[1:5],
    size$medium_sand[1:5])
mean(c(a, a))
mean(c(size$coarse_sand[1:5],
    size$medium_sand[1:5]))
```

Your turn: Creating new columns

- Add a calculation for **total silt**
- Check your work

```
meta <- read_csv("./data/grain_meta.csv")
size <- read_csv("./data/grain_size2.csv") %>%
  left_join(meta, by = "plot") %>%
  mutate(total_sand = coarse_sand + medium_sand + fine_sand,
         ???)
```

Extra Challenge

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

Mutating by group

group_by() and **ungroup()** (**tidyverse** functions, specifically from **dplyr** package)

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

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

Mutating by group

`mutate()` without grouping:

```
size <- size %>%  
  mutate(mean_sand_all = mean(total_sand))
```

```
## # A tibble: 114 x 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  
## # ... with 104 more rows
```

Mutating by group

`mutate()` without grouping:

```
size <- size %>%  
  mutate(mean_sand_all = mean(total_sand))
```

```
## # A tibble: 114 x 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  
## # ... with 104 more rows
```

Grouping via `group_by()`:

```
size <- size %>%  
  group_by(plot) %>%  
  mutate(mean_sand_plot = mean(total_sand)) %>%  
  ungroup()
```

```
## # A tibble: 114 x 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  
## # ... with 106 more rows
```

Mutating by group

`mutate()` without grouping:

```
size <- size %>%  
  mutate(mean_sand_all = mean(total_sand))
```

```
## # A tibble: 114 x 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  
## # ... with 104 more rows
```

Grouping via `group_by()`:

```
size <- size %>%  
  group_by(plot) %>%  
  mutate(mean_sand_plot = mean(total_sand)) %>%  
  ungroup()
```

```
## # A tibble: 114 x 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  
## # ... with 106 more rows
```

Always remember to
`ungroup()` your data



Your turn: Mutating by group

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

```
meta <- read_csv("./data/grain_meta.csv")
size <- read_csv("./data/grain_size2.csv") %>%
  left_join(meta, by = "plot") %>%
  mutate(total_sand = coarse_sand + medium_sand + fine_sand,
         total_silt = coarse_silt + medium_silt + fine_silt) %>%
```

```
??? %>%
```

```
??? %>%
```

```
???
```

Put it all together

```
meta <- read_csv("./data/grain_meta.csv")
size <- read_csv("./data/grain_size2.csv") %>%
  left_join(meta, by = "plot") %>%
  mutate(total_sand = coarse_sand + medium_sand + fine_sand,
         total_silt = coarse_silt + medium_silt + fine_silt) %>%
  group_by(plot) %>%
  mutate(mean_sand = mean(total_sand),
         mean_silt = mean(total_silt)) %>%
  ungroup()

select(size, plot, depth, total_sand, total_silt, mean_sand, mean_silt)
```

```
## # A tibble: 114 x 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
```

Summarizing

Summarizing by group

summarize() (tidyverse functions, specifically from **dplyr** package)

```
summarize(data, column1 = expression1, column2 = expression2)
```

- **tidyverse** functions always start with **data**
- Collapse **data**
- Create new **columns**
- **Columns** filled according to **expression**

Summarizing by group

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

`mutate()`

```
size <- size %>%  
  group_by(plot) %>%  
  mutate(mean_sand = mean(total_sand))  
  select(size, plot, contains("sand"))
```

```
## # A tibble: 114 x 6  
## # Groups:   plot [27]  
##   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  
## # ... with 110 more rows
```

Summarizing by group

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

`summarize()`

```
size <- size %>%  
  group_by(plot) %>%  
  summarize(mean_sand = mean(total_sand), .groups = "drop") #Ungroup data  
size
```

```
## # A tibble: 27 x 2  
##   plot    mean_sand  
##   <chr>      <dbl>  
## 1 CSP01      49.8  
## 2 CSP02      34.7  
## 3 CSP03      29.9  
## 4 CSP04      30.3  
## 5 CSP05      44.6  
## # ... with 22 more rows
```

Summarizing by group

- Keep other id columns by adding them to **group_by()**
- Beware: think carefully about grouping factors!

```
size %>%  
  group_by(plot, depth) %>%  
  summarize(mean_sand = mean(total_sand), .groups = "drop")
```

```
## # A tibble: 114 x 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  
## # ... with 108 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)

```
size %>%  
  group_by(plot, habitat) %>%  
  summarize(mean_sand = mean(total_sand), .groups = "drop")
```

```
## # A tibble: 27 x 3  
##   plot habitat mean_sand  
##   <chr> <chr>      <dbl>  
## 1 CSP01 forest      49.8  
## 2 CSP02 clearcut    34.7  
## 3 CSP03 grassland   29.9  
## 4 CSP04 grassland   30.3  
## 5 CSP05 clearcut    44.6  
## 6 CSP06 grassland   37.8  
## # ... with 21 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

- sample sizes (`n()`)
- means (`mean()`)
- standard deviations (`sd()`)
- standard errors (`sd() / sqrt(n())`)
- total values (`sum()`)
- total counts (`n()`)

Summarizing by group

n() (**tidyverse** functions, specifically from **dplyr** package)

```
n()
```

- *Internal* **tidyverse** function which **does NOT** start with data
- Returns row counts of a data frame according to groups (if present)
- Special function, can only be used *inside* **mutate()** or **summarize()**

For example...

```
size %>%  
  group_by(plot) %>%  
  summarize(samples_total = n(),  
            .groups = "drop")
```

```
## # A tibble: 27 x 2  
##   plot    samples_total  
##   <chr>         <int>  
## 1 CSP01             6  
## 2 CSP02             7  
## 3 CSP03             4  
## 4 CSP04             5  
## 5 CSP05             5  
## 6 CSP06             5  
## # ... with 21 more rows
```

Your Turn: Calculate summary statistics

For each plot and habitat, calculate

- sample sizes with `n()`
- means (`mean()`) for `total_sand` and `total_silt`
- standard deviations (`sd()`) for `total_sand` and `total_silt`
- standard errors (`sd()/sqrt(n())`) for `total_sand` and `total_silt`

```
meta <- read_csv("./data/grain_meta.csv")
size <- read_csv("./data/grain_size2.csv") %>%
  left_join(meta, by = "plot") %>%
  mutate(total_sand = coarse_sand + medium_sand + fine_sand,
         total_silt = coarse_silt + medium_silt + fine_silt)

size_sum <- size %>%
  group_by(plot, habitat) %>%
  ???
```

Extra Challenge
Calculate summary
statistics for your own
data

Transposing

Tidy Data

country	year	cases	population
Afghanistan	1999	37745	19987071
Afghanistan	2000	42666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128042583

variables

country	year	cases	population
Afghanistan	1999	37745	19987071
Afghanistan	2000	42666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	216766	128042583

observations

country	year	cases	population
Afghanistan	99	745	987071
Afghanistan	00	2666	0595360
Brazil	99	7737	72006362
Brazil	00	0488	74504898
China	99	2258	272915272
China	00	6766	28042583

values

Tidy Data

Not Tidy

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

(wide data)

Tidy Data

Not Tidy

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

(wide data)

Tidy

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

(long data)

Why do we care?

How would you plot the untidy data?

(No. of cases by country for each year)

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

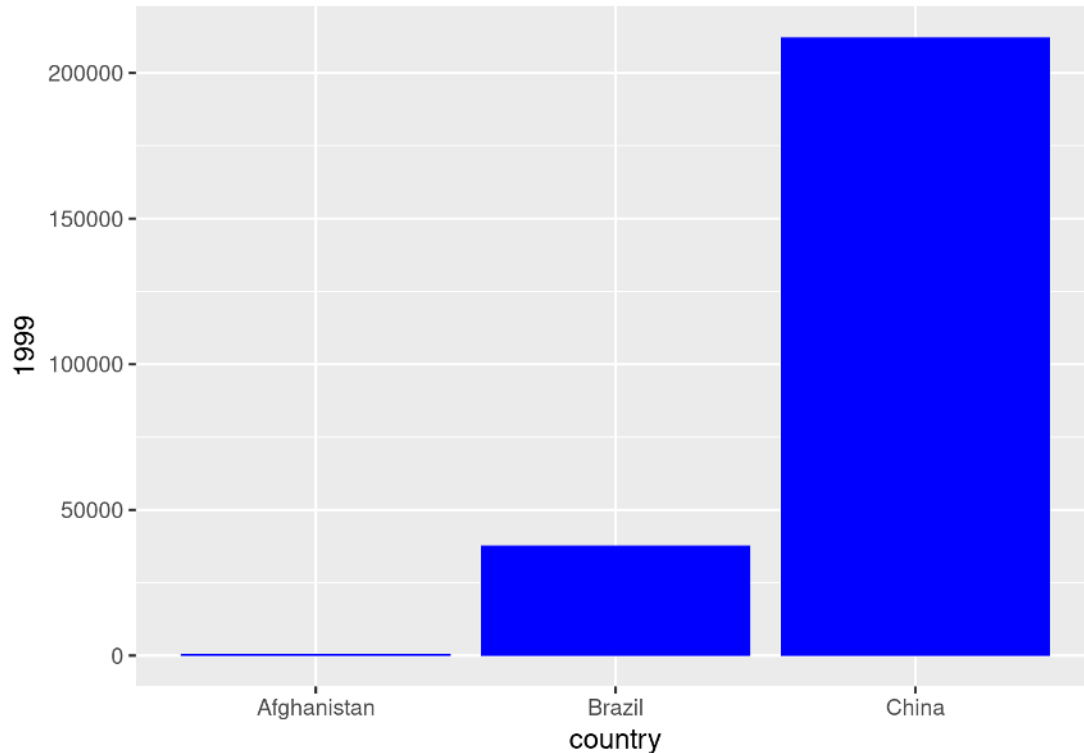
Note

- **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

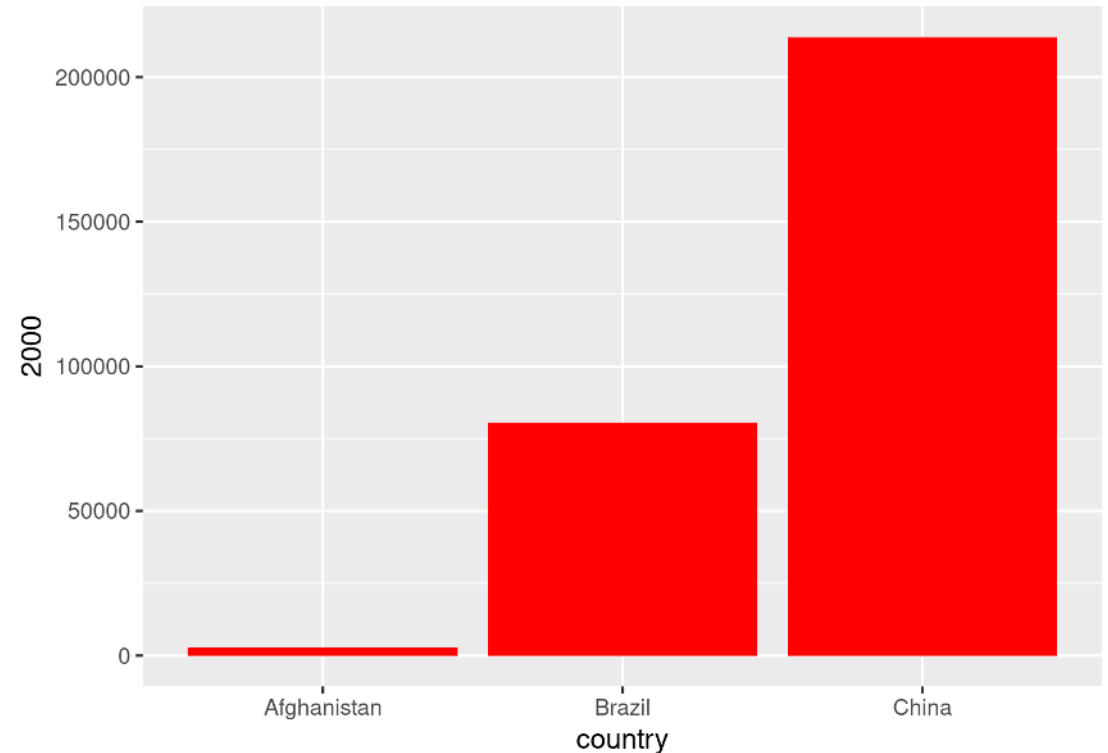
Why do we care?

With un-tidy data

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



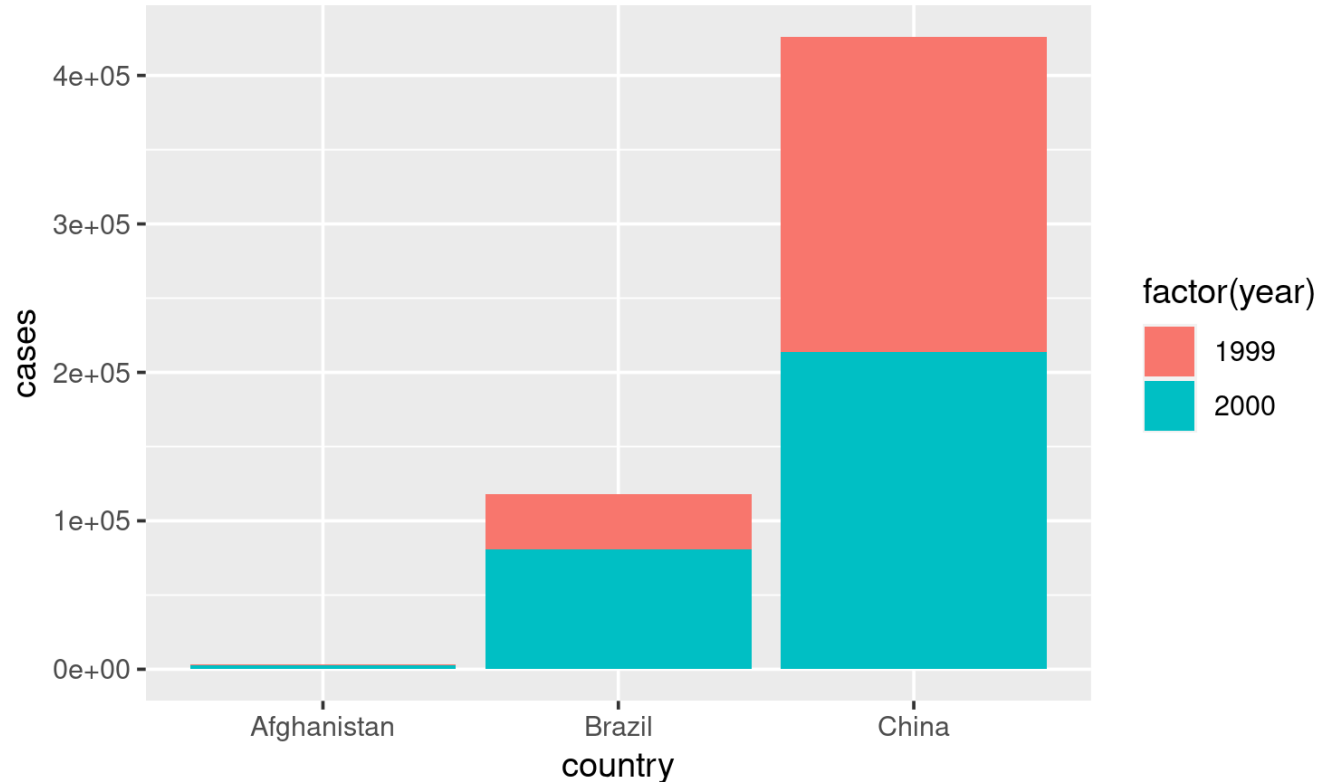
```
ggplot(data = table4a, aes(x = country, y = `2000`)) +  
  geom_bar(stat = "identity", fill = "red")
```



Why do we care?

With tidy data

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

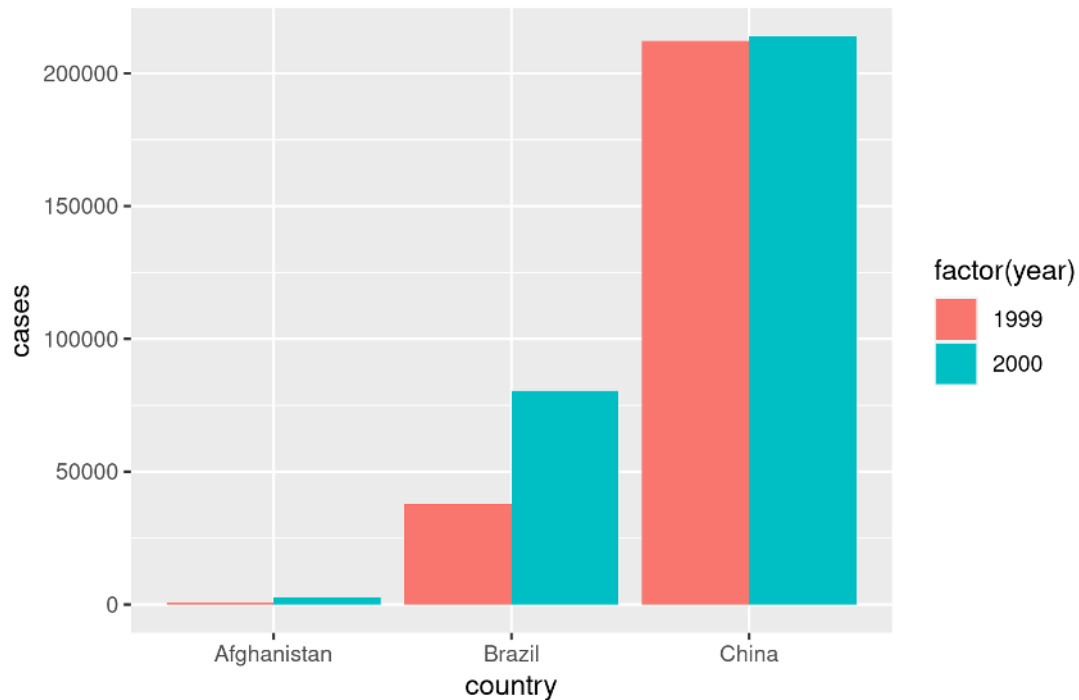


Why do we care?

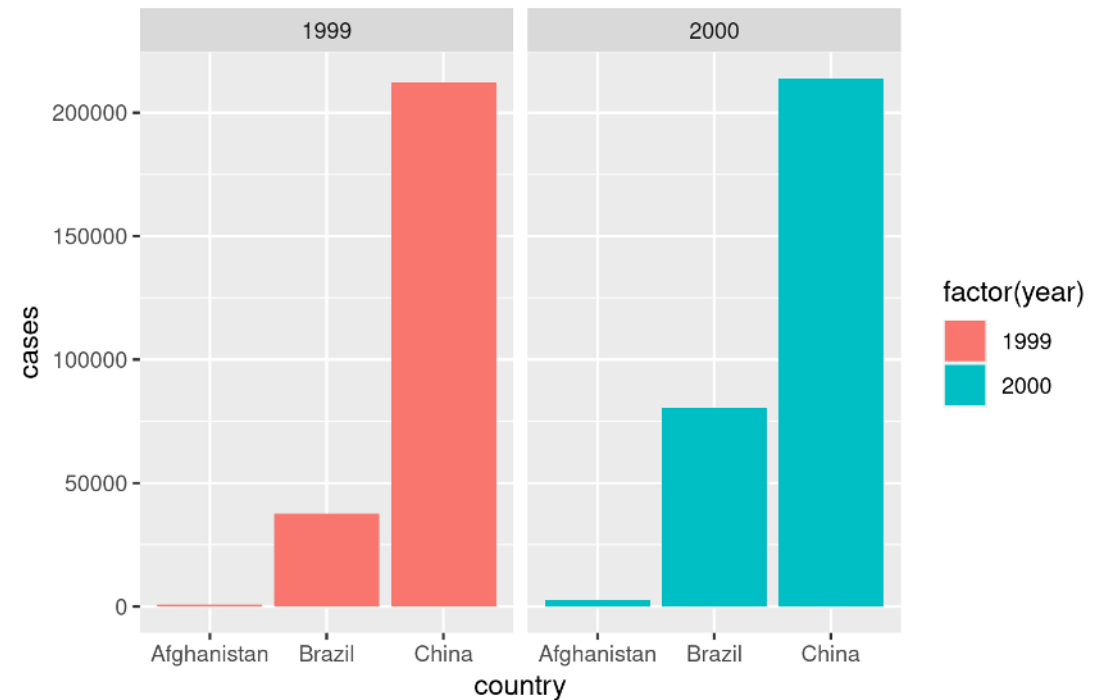
With tidy data

```
g <- ggplot(data = table1, aes(x = country, y = cases, fill = factor(year)))
```

```
g + geom_bar(stat = "identity", position = "dodge")
```



```
g + geom_bar(stat = "identity") + facet_wrap(~year)
```



Going long

pivot_longer()

The diagram illustrates the transformation of a wide table into a long table using the `pivot_longer()` function. The wide table on the right has columns for country, 1999, and 2000. The long table on the left has columns for country, year, and cases. Arrows show how the data is reshaped: the 'country' column remains the same, the '1999' column is mapped to the 'year' column, and the '2000' column is mapped to the 'cases' column. The label 'table4' is placed below the wide table.

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

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

table4

Going long

Wide

```
## # A tibble: 15 x 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
```

Going long

Wide

```
## # A tibble: 15 x 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
```

Long

```
## # A tibble: 15 x 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
```

Going long

pivot_longer() (**tidyverse** function, specifically from **tidyr** package)

```
pivot_longer(data, cols = c(column1, column2),  
             names_to = "categorical_column",  
             values_to = "numerical_column")
```

- **tidyverse** functions always start with **data**
- Takes columns and converts to long **data**
- Column names ('**column1**' and '**column2**') go into "**categorical_column**"
- Column values (*values* of **column1** and **column2**) go into "**numerical_column**"

Going long

pivot_longer() (**tidyverse** function, specifically from **tidyr** package)

```
pivot_longer(data, cols = c(column1, column2),  
             names_to = "categorical_column",  
             values_to = "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"**

Going long

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

```
## # A tibble: 1,026 x 7  
##   plot  depth habitat technician date       type      amount  
##   <chr> <dbl> <chr>    <chr>    <date>    <chr>    <dbl>  
## 1 CSP01     4 forest Catharine 2008-12-28 coarse_sand 13.0  
## 2 CSP01     4 forest Catharine 2008-12-28 medium_sand 17.4  
## 3 CSP01     4 forest Catharine 2008-12-28 fine_sand   19.7  
## 4 CSP01     4 forest Catharine 2008-12-28 coarse_silt 14.1  
## 5 CSP01     4 forest Catharine 2008-12-28 medium_silt 11.2  
## 6 CSP01     4 forest Catharine 2008-12-28 fine_silt   8.17  
## 7 CSP01     4 forest Catharine 2008-12-28 clay        16.3  
## 8 CSP01     4 forest Catharine 2008-12-28 total_sand  50.1  
## 9 CSP01     4 forest Catharine 2008-12-28 total_silt  33.5  
## 10 CSP01    12 forest Catharine 2008-12-28 coarse_sand 10.7  
## 11 CSP01    12 forest Catharine 2008-12-28 medium_sand 16.9  
## 12 CSP01    12 forest Catharine 2008-12-28 fine_sand   19.2  
## # ... with 1,014 more rows
```

Your turn: Lengthen data

- Practice transforming a summarized sand data
- Gather all variables except plot and sample_size into a long format

```
sand_sum <- read_csv("./data/grain_size2.csv") %>%  
  mutate(total_sand = coarse_sand + medium_sand + fine_sand) %>%  
  group_by(plot) %>%  
  summarize(sample_size = n(),  
            mean_sand = mean(total_sand),  
            sd_sand = sd(total_sand),  
            se_sand = sd_sand / sqrt(sample_size))
```

```
sand_long <- pivot_longer(sand_sum, ???)
```

Going wide

pivot_wider()

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

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

table2

Going wide

Long

```
## # A tibble: 15 x 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
```

Going wide

Long

```
## # A tibble: 15 x 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
```

Wide

```
## # A tibble: 15 x 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
```

Going wide

pivot_wider() (**tidyverse** function, specifically from **tidyr** package)

```
pivot_wider(data,  
            names_from = categorical_column,  
            values_from = numerical_column)
```

- **tidyverse** functions always start with **data**
- Takes columns and converts to wide **data**
- Values in **categorical_column** become column names
- Values in **numerical_column** become column values

Going wide

pivot_wider() (**tidyverse** function, specifically from **tidyr** package)

```
pivot_wider(data,  
            names_from = categorical_column,  
            values_from = numerical_column)
```

In our example:

- **data** = **size**
- **names_from** = **type**
- **values_from** = **amount**

Going wide

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

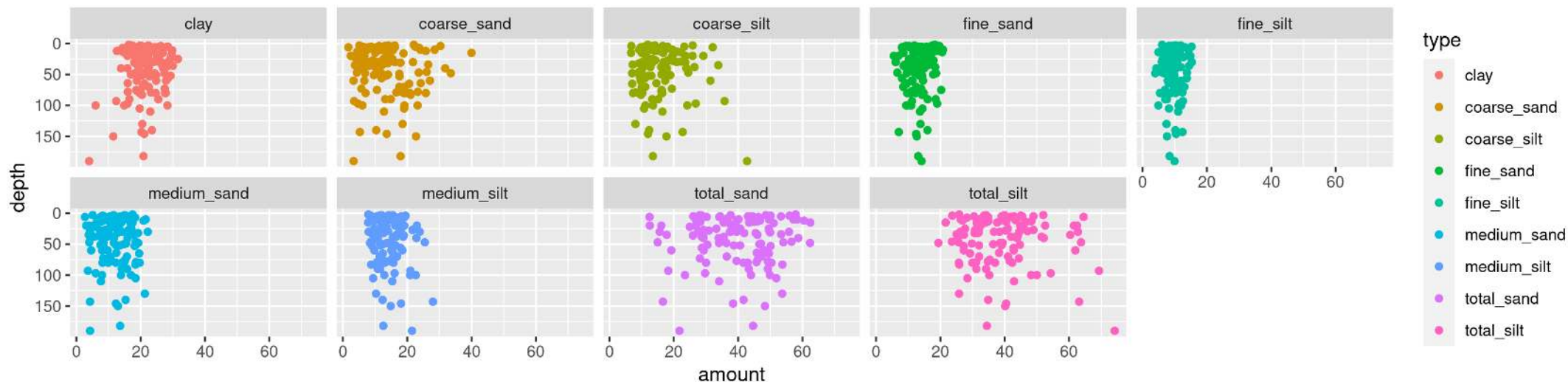
```
## # A tibble: 114 x 14  
##   plot depth habitat technician date coarse_sand medium_sand fine_sand coarse_silt medium_silt  
##   <chr> <dbl> <chr>    <chr>      <date>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>  
## 1 CSP01     4 forest Catharine 2008-12-28 13.0      17.4      19.7      14.1      11.2  
## 2 CSP01    12 forest Catharine 2008-12-28 10.7      16.9      19.2      14.1      11.7  
## 3 CSP01    35 forest Catharine 2008-12-28 12.1      17.8      16.1      10.3      9.51  
## 4 CSP01    53 forest Catharine 2008-12-28 17.6      18.2      14.3       9.4       9.1  
## 5 CSP01    83 forest Catharine 2008-12-28 21.0      18.4      14.3       9.79      8.79  
## 6 CSP01   105 forest Catharine 2008-12-28 19.0      18.4      14.4      10.8       9.4  
## 7 CSP08    10 grassl... Catharine 2008-07-26 11.6      17.1      20.8      16.3       9.55  
## 8 CSP08    27 grassl... Catharine 2008-07-26 15.4      16.2      17.8      14.3      10.4  
## 9 CSP08    90 grassl... Catharine 2008-07-26 14.9      15.8      18.6      15.1      11.5  
## 10 CSP02     5 clearc... Catharine 2008-11-19 8.75      8.64      8.66      12.0      18.3  
## # ... with 104 more rows, and 4 more variables: fine_silt <dbl>, clay <dbl>, total_sand <dbl>, total_silt <dbl>
```


Again: Why transpose?

Figures: Long data are great for graphing

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

```
ggplot(data = size_long, aes(y = depth, x = amount, colour = type)) +  
  geom_point() +  
  scale_y_reverse() +  
  facet_wrap(~ type, nrow = 2)
```



Again: Why transpose?

Figures: Take it to the next step

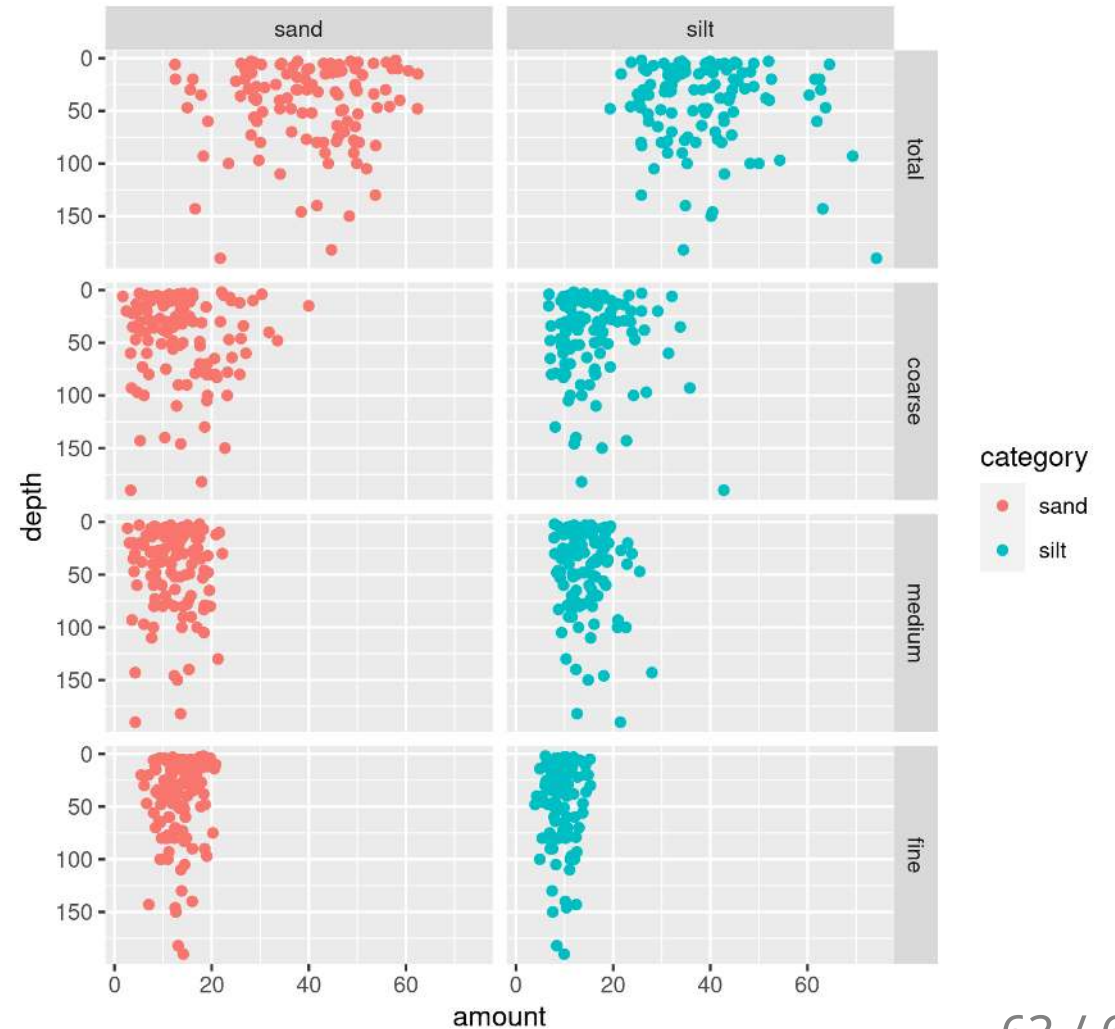
```
size <- read_csv("./data/grain_size2.csv") %>%  
  left_join(meta, by = "plot") %>%  
  mutate(total_sand = coarse_sand + medium_sand + fine_sand,  
         total_silt = coarse_silt + medium_silt + fine_silt)  
  
size_long <- pivot_longer(size, cols = c(-plot, -depth, -technician, -habitat, -date,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

```
## # A tibble: 912 x 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 2008-12-28 coarse sand     13.0  
## 2 CSP01     4   16.3 forest Catharine 2008-12-28 medium sand     17.4  
## 3 CSP01     4   16.3 forest Catharine 2008-12-28 fine  sand     19.7  
## 4 CSP01     4   16.3 forest Catharine 2008-12-28 coarse silt     14.1  
## 5 CSP01     4   16.3 forest Catharine 2008-12-28 medium silt     11.2  
## 6 CSP01     4   16.3 forest Catharine 2008-12-28 fine  silt       8.17  
## # ... with 906 more rows
```

Again: Why transpose?

Figures

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



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)

```
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)

```
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)

```
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)

```
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 (**cases**) vs. the **population** in data frame **table2**

```
temp <- table2 %>%
```

```
  ???(???)
```

```
ggplot(data = temp, ???) +
```

```
  ???
```

Put it all together

```
meta <- read_csv("./data/grain_meta.csv")
size <- read_csv("./data/grain_size2.csv") %>%
  left_join(meta, by = "plot") %>%
  mutate(total_sand = coarse_sand + medium_sand + fine_sand,
         total_silt = coarse_silt + medium_silt + fine_silt)

size_sum <- size %>%
  group_by(plot, habitat) %>%
  summarize(sample_size = n(),
            total_sand = sum(total_sand),
            mean_sand = mean(total_sand),
            sd_sand = sd(total_sand),
            se_sand = sd_sand / sqrt(sample_size),
            total_silt = sum(total_silt),
            mean_silt = mean(total_silt),
            sd_silt = sd(total_silt),
            se_silt = sd_silt / sqrt(sample_size))

size_long <- size %>%
  pivot_longer(cols = c(-plot, -depth, -technician, -habitat, -date, -clay),
              values_to = "amount", names_to = c("size", "category"), names_sep = "_") %>%
  mutate(size = factor(size, levels = c("total", "coarse", "medium", "fine")))
```

Put it all together: Save your data

```
write_csv(size, "./Datasets/size_total.csv")  
write_csv(size_sum, "./Datasets/size_summary.csv")  
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 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
 - 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()** your data before you can widen it
- **mutate()** is giving me weird results
 - Is your data grouped when it shouldn't be?
 - Try using **ungroup()** first
- I get a warning when I join data sets
 - Often, this refers to mismatched factor levels
 - This happens if the factor levels in one data frame do not match the factor levels in the other
 - 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 5: Transforming data](#)
 - [Chapter 12: Tidy data](#)
 - [Chapter 13: Relational data](#)
- [RStudio Data Manipulation with **dplyr**, tidyr](#)
 - Or Help > Cheatsheets > Data Manipulation with dplyr, tidyr