

Summerizing relationships and tidying and joining data

Seung-Ho An, University of Arizona

- Housekeeping
- Z-scores and standardization
- Correlation
- Writing our own functions
- Causality review
- Pivoting data longer
- Joining data sets

SQL like functions to find a match (we will cover this during the lecture)

Easier tab functions: `prop.table()`; this is also a tidyway

`pull()` pulls a vector from an object. It works like a `$` under pipes (last week's example)

- You can also call a vector/variable/column with `pull()` (e.g., `pull(.data, var)`)

Here is the link for the another example from Gruber on the powerful DiD tool

- This link will expire in a couple of weeks

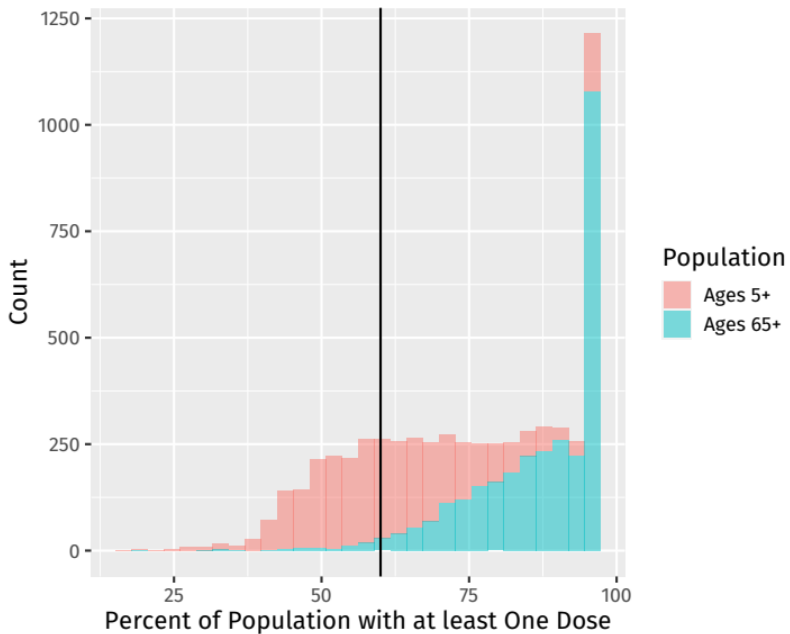
1. Z-scores and standardization

COVID vaccination rates and votes

```
library(tidyverse)
library(TPDDdata)
covid_votes
```

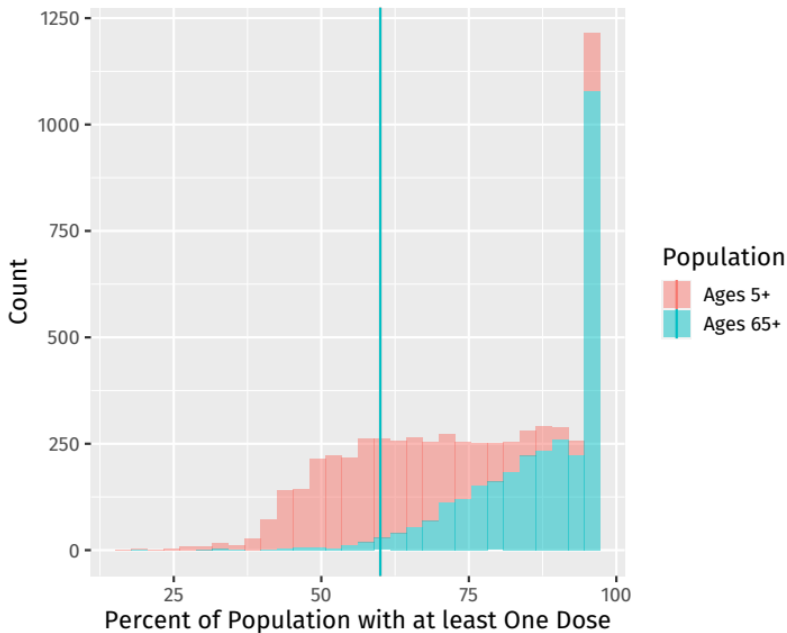
```
## # A tibble: 3,114 x 8
##   fips county      state one_dose_5plus-1 one_d-2 boost-3 dem_p-4 dem_p-5
##   <chr> <chr>      <chr>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 26039 Crawford County MI          55.7    77.3    31.2    43.8    34.0
## 2 40015 Caddo County OK           83.3     95     30.3    46.4    27.1
## 3 17007 Boone County IL           71.1    94.5    35.1    41.8    42.2
## 4 12055 Highlands County FL          68.9    93.7    24.7    40.3    32.5
## 5 34029 Ocean County NJ            71     95     32.1    47.2    35.0
## 6 01067 Henry County AL          58.5    85.5    18.2    40.1    28.0
## 7 27037 Dakota County MN           81     95     49.5    46.9    55.7
## 8 27115 Pine County MN          56.5     85    31.7    47.0    33.9
## 9 51750 Radford city VA           41.5    73.8     1.79    46.4    53.1
## 10 22009 Avoyelles Parish LA          59.7    80.1    21.9    45.7    28.8
## # ... with 3,104 more rows, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct, 3: booster_5plus_pct,
## #   4: dem_pct_2000, 5: dem_pct_2020
```

Is 60% vaccinated a lot?

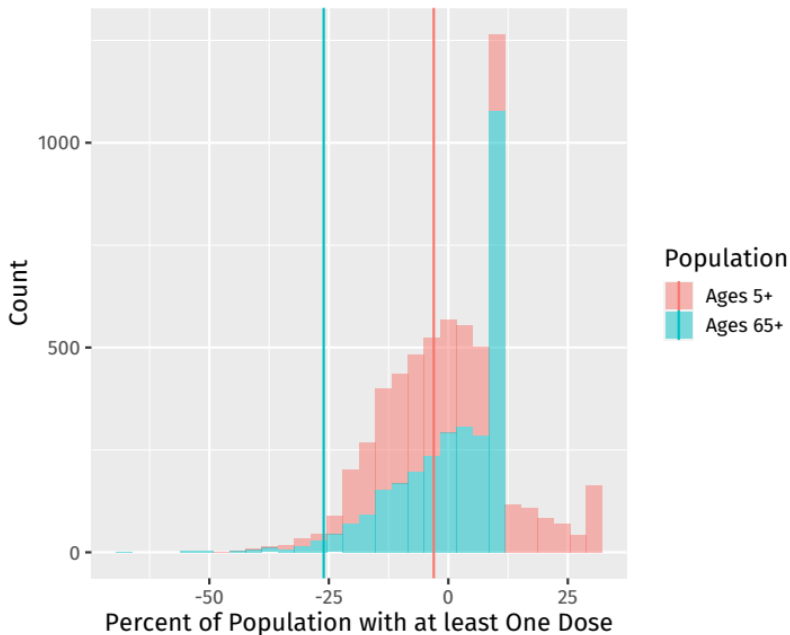


- How large 60% vaccinated is depends on the distribution!
 - Clear to see from the histogram
 - Middling for the 5+ group, but very low for the 65+ group
- Can we transform the values of our variables to be **common units**?
- Yes, with two transformations:
 - **Centering**: subtract the mean of the variable from each value
 - **Scaling**: dividing deviations from the mean by the standard deviation

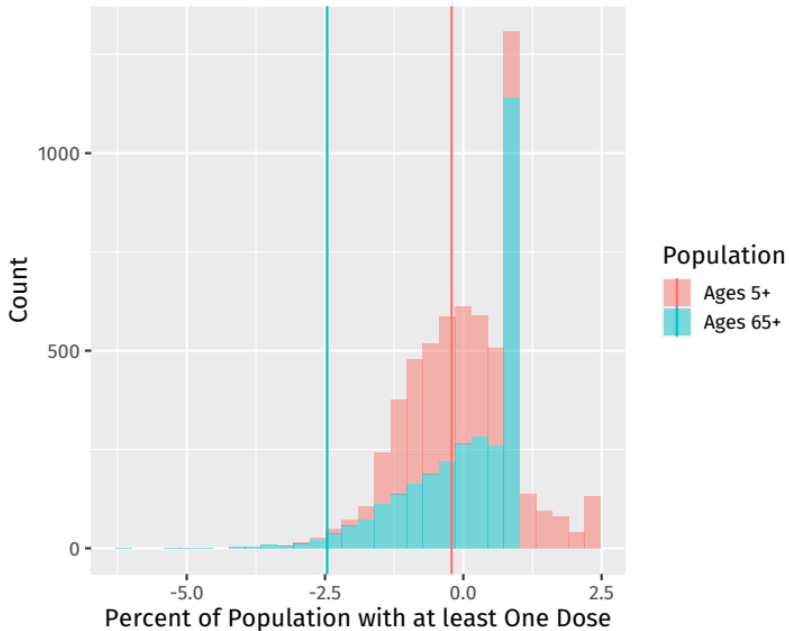
Original distributions



Centered distributions



Centered and scaled distributions



- Centering tells us immediately if a value is above or below the mean
- Scaling tells us how many standard deviations away from the mean it is
- Combine them with the **z-score** transformation:

$$\text{z-score of } x_i = \frac{x_i - \text{mean of } x}{\text{standard deviation of } x}$$

- Useful heuristic: data more than 3 SDs away from its mean are rare

```
covid_votes |>
  mutate(one_dose_centered = one_dose_5plus_pct -
         mean(one_dose_5plus_pct, na.rm=TRUE)) |>
  select(fips:state, one_dose_5plus_pct, one_dose_centered)
```

```
## # A tibble: 3,114 x 5
```

	fips	county	state	one_dose_5plus_pct	one_dose_centered
	<chr>	<chr>	<chr>	<dbl>	<dbl>
## 1	26039	Crawford County	MI	55.7	-7.35
## 2	40015	Caddo County	OK	83.3	20.2
## 3	17007	Boone County	IL	71.1	8.05
## 4	12055	Highlands County	FL	68.9	5.85
## 5	34029	Ocean County	NJ	71	7.95
## 6	01067	Henry County	AL	58.5	-4.55
## 7	27037	Dakota County	MN	81	17.9
## 8	27115	Pine County	MN	56.5	-6.55
## 9	51750	Radford city	VA	41.5	-21.6
## 10	22009	Avoyelles Parish	LA	59.7	-3.35

```
## # ... with 3,104 more rows
```

```
covid_votes |>
  mutate(
    one_dose_z =
      (one_dose_5plus_pct - mean(one_dose_5plus_pct, na.rm=TRUE)) /
      sd(one_dose_5plus_pct, na.rm=TRUE)) |>
  select(fips:state, one_dose_5plus_pct, one_dose_z)
```

```
## # A tibble: 3,114 x 5
```

	fips	county	state	one_dose_5plus_pct	one_dose_z
	<chr>	<chr>	<chr>	<dbl>	<dbl>
## 1	26039	Crawford County	MI	55.7	-0.508
## 2	40015	Caddo County	OK	83.3	1.40
## 3	17007	Boone County	IL	71.1	0.556
## 4	12055	Highlands County	FL	68.9	0.404
## 5	34029	Ocean County	NJ	71	0.549
## 6	01067	Henry County	AL	58.5	-0.314
## 7	27037	Dakota County	MN	81	1.24
## 8	27115	Pine County	MN	56.5	-0.452
## 9	51750	Radford city	VA	41.5	-1.49
## 10	22009	Avoyelles Parish	LA	59.7	-0.231
## #	... with 3,104 more rows				

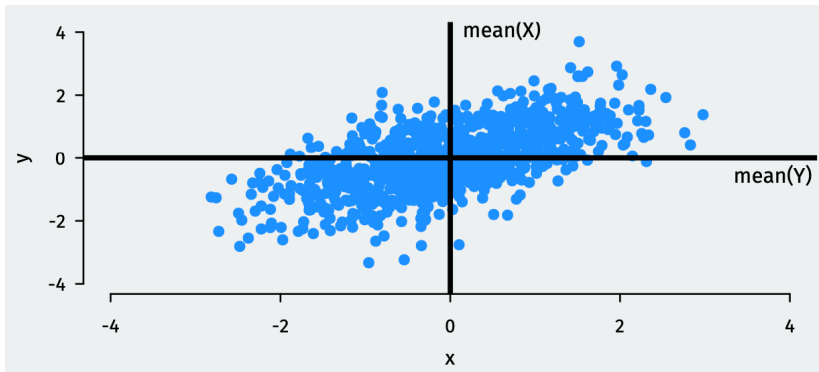
2. Correlation

- How do variables move together on average?
- When x_i is big, what is y_i likely to be?
 - Positive correlation: When x_i is big, y_i is also big
 - Negative correlation: When x_i is big, y_i is small
 - High magnitude of correlation: data cluster tightly around a line
- The technical definition of the **correlation coefficient**:

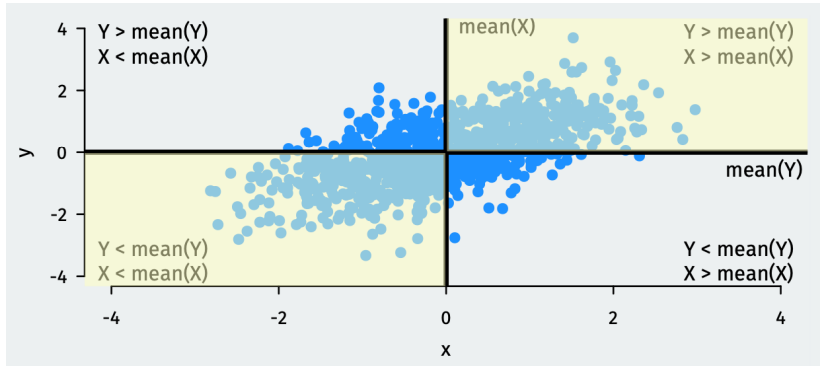
$$\frac{1}{n-1} \sum_{i=1}^n [(\text{z-score for } x_i) \times (\text{z-score for } y_i)]$$

- Interpretation:
 - Correlation is between -1 and 1
 - Correlation of 0 means no linear association
 - Positive correlations \approx positive associations
 - Negative correlations \approx negative associations
 - Closer to -1 or 1 means stronger association

Correlation intuition



Correlation intuition



Large values of X tend to occur with large values of Y:

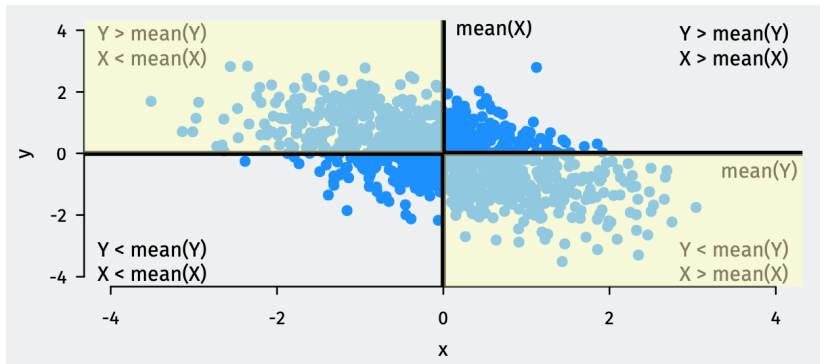
$$(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{pos. num.}) \times (\text{pos. num}) = +$$

Small values of X tend to occur with small values of Y:

$$(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{neg. num.}) \times (\text{neg. num}) = +$$

If these dominate \approx positive correlation

Correlation intuition



Large values of X tend to occur with small values of Y:

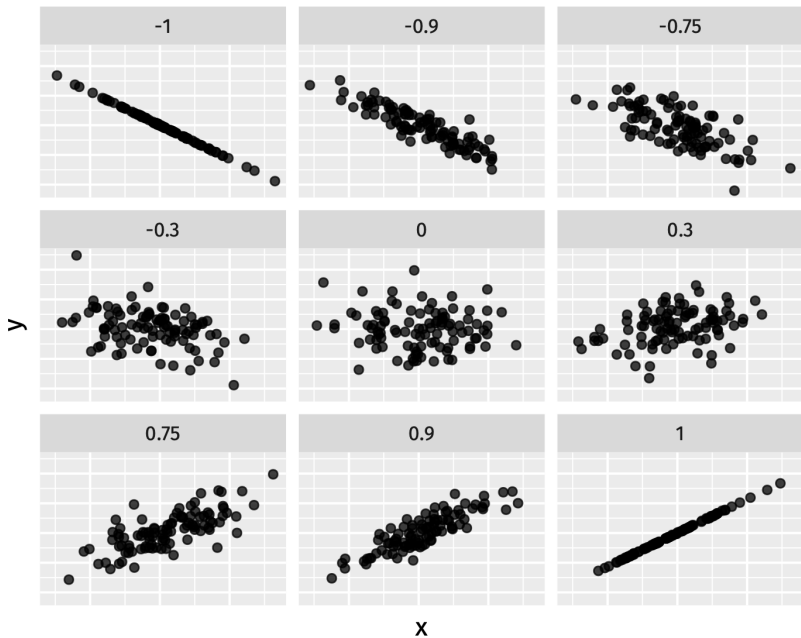
$$(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{pos. num.}) \times (\text{neg. num.}) = -$$

Small values of X tend to occur with large values of Y:

$$(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{neg. num.}) \times (\text{pos. num.}) = -$$

If these dominate \approx negative correlation

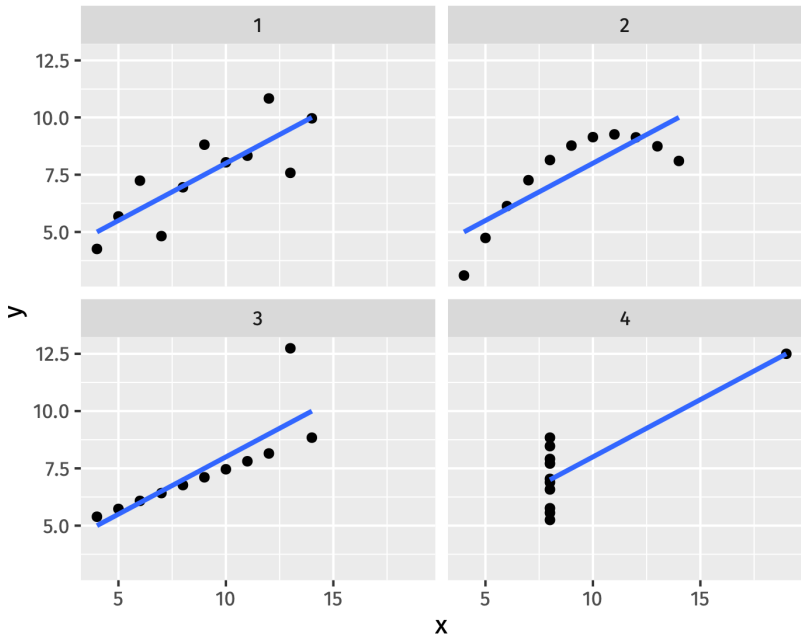
Correlation examples



Properties of correlation coefficient

- Correlation measures **linear** association
- Order doesn't matter: $\text{cor}(x,y) = \text{cor}(y,x)$
- Not affected by changes of scale:
 - $\text{cor}(x,y) = \text{cor}(ax+b, cy+d)$
 - Celsius vs. Fahrenheit: dollars vs. pesos; cm vs. in.

All 4 relationships have 0.816 correlation



Use the `cor()` function:

```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020)
```

```
## [1] NA
```

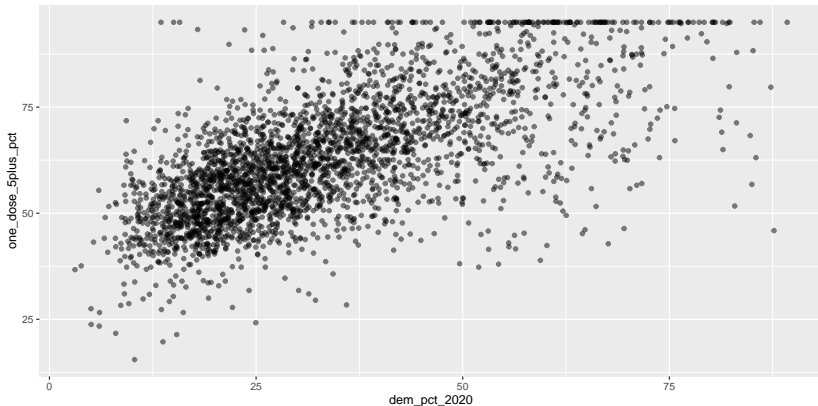
Missing values: set the `use = "pairwise"` -> available case analysis

```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020, use="pairwise")
```

```
## [1] 0.6664387
```

Comparing correlations

```
covid_votes |>  
  ggplot(aes(x = dem_pct_2020, y = one_dose_5plus_pct)) +  
  geom_point(alpha = 0.5)
```

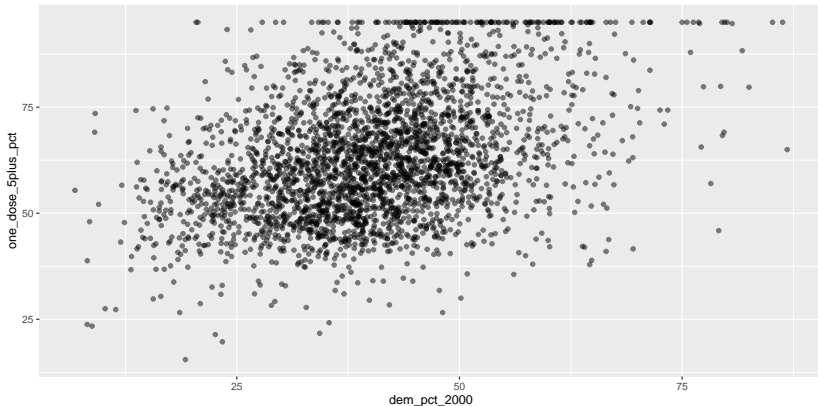


```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020, use="pairwise")
```

```
## [1] 0.6664387
```

Comparing correlations

```
covid_votes |>  
  ggplot(aes(x = dem_pct_2000, y = one_dose_5plus_pct)) +  
  geom_point(alpha = 0.5)
```

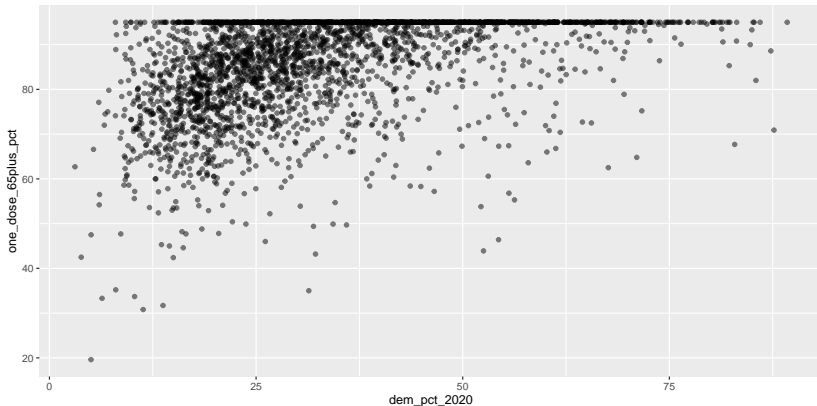


```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2000, use="pairwise")
```

```
## [1] 0.3941203
```


Comparing correlations

```
covid_votes |>  
  ggplot(aes(x = dem_pct_2020, y = one_dose_65plus_pct)) +  
  geom_point(alpha = 0.5)
```



```
cor(covid_votes$one_dose_65plus_pct, covid_votes$dem_pct_2020, use="pairwise")
```

```
## [1] 0.447203
```

3. Writing our own functions.

copy-pasting codes is tedious and prone to failure:

```
covid_votes |>
  mutate(
    one_dose_5p_z=
      (one_dose_5plus_pct - mean(one_dose_5plus_pct, na.rm = TRUE)) /
      sd(one_dose_5plus_pct, na.rm=TRUE),
    one_dose_65p_z=
      (one_dose_65plus_pct - mean(one_dose_65plus_pct, na.rm = TRUE)) /
      sd(one_dose_65plus_pct, na.rm=TRUE),
    booster_z =
      (booster_5plus_pct - mean(booster_5plus_pct, na.rm = TRUE)) /
      sd(booster_5plus_pct, na.rm=TRUE),
    dem_pct_2000_z =
      (dem_pct_2000 - mean(dem_pct_2000, na.rm = TRUE)) /
      sd(dem_pct_2000, na.rm = TRUE),
    dem_pct_2020_z =
      (dem_pct_2020 - mean(dem_pct_2020, na.rm = TRUE)) /
      sd(dem_pct_2020, na.rm = TRUE)
  )
```

Notice that all of the mutations follow the same template:

```
(x- mean(x, na.rm = TRUE)) / sd(x, na.rm= TRUE)
```

Only one thing varies: the column (or the vector) of data (variable), represented with `x`

We create functions like so:

```
name <- function(arguments) {  
  body  
}
```

Three components:

- 1 **Name:** the name of the function that we'll use to call it.
Maybe `z_score`?
- 2 **Arguments:** things that we want to vary across calls of our function. We will use `x`
- 3 **Body:** the code that the function performs

Convert our template to a function:

```
z_score <- function(x){  
  (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)  
}
```

Let's check this if it works:

```
z_score(c(1, 2, 3, 4, 5))
```

```
## [1] -1.2649111 -0.6324555  0.0000000  0.6324555  1.2649111
```

Now, cleaning up our code

```
covid_votes |>
  mutate(
    one_dose_5p_z=z_score(one_dose_5plus_pct),
    one_dose_65p_z=z_score(one_dose_65plus_pct),
    booster_z =z_score(booster_5plus_pct),
    dem_pct_2000_z =z_score(dem_pct_2000),
    dem_pct_2020_z =z_score(dem_pct_2020)
  )
```

```
## # A tibble: 3,114 x 13
##   fips county state one_d-1 one_d-2 boost-3 dem_p-4 dem_p-5 one_d-6 one_d-7
##   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 26039 Crawford~ MI 55.7 77.3 31.2 43.8 34.0 -0.508 -0.829
## 2 40015 Caddo Co~ OK 83.3 95 30.3 46.4 27.1 1.40 0.843
## 3 17007 Boone Co~ IL 71.1 94.5 35.1 41.8 42.2 0.556 0.795
## 4 12055 Highland~ FL 68.9 93.7 24.7 40.3 32.5 0.404 0.720
## 5 34029 Ocean Co~ NJ 71 95 32.1 47.2 35.0 0.549 0.843
## 6 01067 Henry Co~ AL 58.5 85.5 18.2 40.1 28.0 -0.314 -0.0545
## 7 27037 Dakota C~ MN 81 95 49.5 46.9 55.7 1.24 0.843
## 8 27115 Pine Cou~ MN 56.5 85 31.7 47.0 33.9 -0.452 -0.102
## 9 51750 Radford ~ VA 41.5 73.8 1.79 46.4 53.1 -1.49 -1.16
## 10 22009 Avoyelle~ LA 59.7 80.1 21.9 45.7 28.8 -0.231 -0.564
## # ... with 3,104 more rows, 3 more variables: booster_z <dbl>,
## # dem_pct_2000_z <dbl>, dem_pct_2020_z <dbl>, and abbreviated variable names
## # 1: one_dose_5plus_pct, 2: one_dose_65plus_pct, 3: booster_5plus_pct,
## # 4: dem_pct_2000, 5: dem_pct_2020, 6: one_dose_5p_z, 7: one_dose_65p_z
```

If we want to replace our variables with z-scores, we can use the `across()` function to perform many mutations at once:

```
covid_votes |>
  mutate(across(one_dose_5plus_pct:dem_pct_2020, z_score))
```

```
## # A tibble: 3,114 x 8
##   fips county      state one_dose_5plus-1 one_d-2 boost-3 dem_p-4 dem_p-5
##   <chr> <chr>      <chr>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 26039 Crawford County MI        -0.508 -0.829    0.531  0.340  0.0471
## 2 40015 Caddo County OK          1.40  0.843    0.439  0.556 -0.387
## 3 17007 Boone County IL          0.556  0.795    0.927  0.163  0.563
## 4 12055 Highlands County FL        0.404  0.720   -0.135  0.0402 -0.0487
## 5 34029 Ocean County NJ          0.549  0.843    0.623  0.624  0.109
## 6 01067 Henry County AL        -0.314 -0.0545   -0.799  0.0255 -0.328
## 7 27037 Dakota County MN          1.24  0.843    2.40  0.598  1.41
## 8 27115 Pine County MN        -0.452 -0.102    0.577  0.612  0.0393
## 9 51750 Radford city VA         -1.49 -1.16    -2.47  0.556  1.25
## 10 22009 Avoyelles Parish LA       -0.231 -0.564   -0.424  0.501 -0.280
## # ... with 3,104 more rows, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct, 3: booster_5plus_pct,
## #   4: dem_pct_2000, 5: dem_pct_2020
```


We could also target all the numeric variables:

```
covid_votes |>
  mutate(across(where(is.numeric), z_score))
```

```
## # A tibble: 3,114 x 8
##   fips county      state one_dose_5plus-1 one_d-2 boost-3 dem_p-4 dem_p-5
##   <chr> <chr>      <chr>      <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 26039 Crawford County MI          -0.508 -0.829  0.531  0.340  0.0471
## 2 40015 Caddo County OK            1.40  0.843  0.439  0.556 -0.387
## 3 17007 Boone County IL             0.556  0.795  0.927  0.163  0.563
## 4 12055 Highlands County FL            0.404  0.720 -0.135  0.0402 -0.0487
## 5 34029 Ocean County NJ             0.549  0.843  0.623  0.624  0.109
## 6 01067 Henry County AL            -0.314 -0.0545 -0.799  0.0255 -0.328
## 7 27037 Dakota County MN             1.24  0.843  2.40  0.598  1.41
## 8 27115 Pine County MN            -0.452 -0.102  0.577  0.612  0.0393
## 9 51750 Radford city VA            -1.49 -1.16  -2.47  0.556  1.25
## 10 22009 Avoyelles Parish LA          -0.231 -0.564 -0.424  0.501 -0.280
## # ... with 3,104 more rows, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct, 3: booster_5plus_pct,
## #   4: dem_pct_2000, 5: dem_pct_2020
```

We could also target only the first dose variables:

```
covid_votes |>
  mutate(across(starts_with("one_dose"), z_score))
```

```
## # A tibble: 3,114 x 8
##   fips county      state one_dose_5plus-1 one_d-2 boost-3 dem_p-4 dem_p-5
##   <chr> <chr>      <chr>      <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 26039 Crawford County MI          -0.508 -0.829 31.2 43.8 34.0
## 2 40015 Caddo County OK           1.40 0.843 30.3 46.4 27.1
## 3 17007 Boone County IL            0.556 0.795 35.1 41.8 42.2
## 4 12055 Highlands County FL            0.404 0.720 24.7 40.3 32.5
## 5 34029 Ocean County NJ             0.549 0.843 32.1 47.2 35.0
## 6 01067 Henry County AL           -0.314 -0.0545 18.2 40.1 28.0
## 7 27037 Dakota County MN             1.24 0.843 49.5 46.9 55.7
## 8 27115 Pine County MN           -0.452 -0.102 31.7 47.0 33.9
## 9 51750 Radford city VA            -1.49 -1.16 1.79 46.4 53.1
## 10 22009 Avoyelles Parish LA          -0.231 -0.564 21.9 45.7 28.8
## # ... with 3,104 more rows, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct, 3: booster_5plus_pct,
## #   4: dem_pct_2000, 5: dem_pct_2020
```

`starts_with` is a tidyway (`tidyselect`)

It looks for an exact match from the beginning of characters

There are other functions too such as:

`ends_with()`: ends with an exact suffix

`contains()`: contains a literal string

`matches()`: matches a regular expression

`num_range()`: matches a numerical range like x01, x02, x03, etc.

Adding arguments to our function

What if we want to be able to control `na.rm` in the calls to `mean()` and `sd()` in our `z_score` function? Add an argument!

```
z_score2 <- function(x, na.rm = FALSE){  
  (x - mean(x, na.rm = na.rm)) / sd(x, na.rm = na.rm)  
}
```

```
head(z_score2(covid_votes$one_dose_5plus_pct))
```

```
## [1] NA NA NA NA NA NA
```

```
head(z_score2(covid_votes$one_dose_5plus_pct, na.rm=TRUE))
```

```
## [1] -0.5076545  1.3982427  0.5557809  0.4038615  0.5488754 -0.3143026
```

4. Causality review



Potential outcomes:

- $Y_i(1)$ is the value that the outcome would take if gave unit i **treatment** and changed nothing else about them
- $Y_i(0)$ is the value that the outcome would take if gave unit i **no treatment** and changed nothing else about them
- Not the **possible values** of the outcome

Treatment: $T_i = 1$ if vaccinated, $T_i = 0$ if not

Outcome: $Y_i = 1$ if acquired COVID after 12 weeks, $Y_i = 0$ if not

- What are the potential outcomes $Y_i(0)$ and $Y_i(1)$?
- Why not compare early volunteers for the vaccine to the overall population?

5. Pivoting data longer

Mortality data

```
library(TPDDdata)
mortality
```

```
## # A tibble: 217 x 52
```

```
##   country      count~1 indic~2 `1972` `1973` `1974` `1975` `1976` `1977` `1
##   <chr>         <chr>   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <
## 1 Aruba        ABW     Mortal~   NA    NA    NA    NA    NA    NA    NA
## 2 Afghanistan AFG     Mortal~ 291   285.  280.  274.  268   262.  2
## 3 Angola       AGO     Mortal~   NA    NA    NA    NA    NA    NA    NA
## 4 Albania      ALB     Mortal~   NA    NA    NA    NA    NA    NA    NA
## 5 Andorra      AND     Mortal~   NA    NA    NA    NA    NA    NA    NA
## 6 United Arab~ ARE     Mortal~ 80.1   72.6  65.7  59.4  53.6  48.3
## 7 Argentina    ARG     Mortal~ 69.7   68.2  66.1  63.3  59.8  55.7
## 8 Armenia      ARM     Mortal~   NA    NA    NA    NA    87.1  83.6
## 9 American Sa~ ASM     Mortal~   NA    NA    NA    NA    NA    NA
## 10 Antigua and~ ATG     Mortal~ 26.9   25.1  23.5  22.1  20.8  19.5
## # ... with 207 more rows, 42 more variables: `1979` <dbl>, `1980` <dbl>,
## #   `1981` <dbl>, `1982` <dbl>, `1983` <dbl>, `1984` <dbl>, `1985` <dbl>,
## #   `1986` <dbl>, `1987` <dbl>, `1988` <dbl>, `1989` <dbl>, `1990` <dbl>,
## #   `1991` <dbl>, `1992` <dbl>, `1993` <dbl>, `1994` <dbl>, `1995` <dbl>,
## #   `1996` <dbl>, `1997` <dbl>, `1998` <dbl>, `1999` <dbl>, `2000` <dbl>,
## #   `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>, `2005` <dbl>,
## #   `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>, `2010` <dbl>, ..
```

Mortality data in a "wide" format (years in columns)

We can convert this to country-year rows with `pivot_longer()`

```
mydata |>
  pivot_longer(
    cols = <variables to pivot>,
    names_to = <new variable to put column names>,
    values_to = <new variable to put column values>
  )
```

Pivoting the mortality data

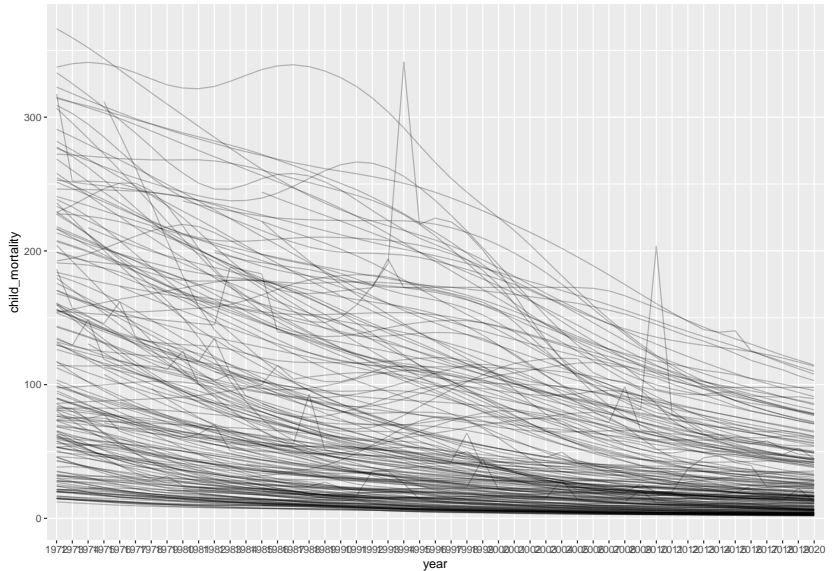
```
mortality |>
  select(-indicator) |>
  pivot_longer(
    cols=`1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  )
```

```
## # A tibble: 10,633 x 4
##   country country_code year  child_mortality
##   <chr>    <chr>      <chr>          <dbl>
## 1 Aruba    ABW        1972             NA
## 2 Aruba    ABW        1973             NA
## 3 Aruba    ABW        1974             NA
## 4 Aruba    ABW        1975             NA
## 5 Aruba    ABW        1976             NA
## 6 Aruba    ABW        1977             NA
## 7 Aruba    ABW        1978             NA
## 8 Aruba    ABW        1979             NA
## 9 Aruba    ABW        1980             NA
## 10 Aruba   ABW        1981             NA
## # ... with 10,623 more rows
```

Let's do line plots!

```
mortality |>
  select(-indicator) |>
  pivot_longer(
    cols=`1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  ) |>
  ggplot(mapping = aes(x = year, y = child_mortality, group = year)) |>
  geom_line(alpha = 0.25)
```

Hmm, what's going on?



Making sure year is numeric

By default, pivoted column names are characters, but we can transform them:

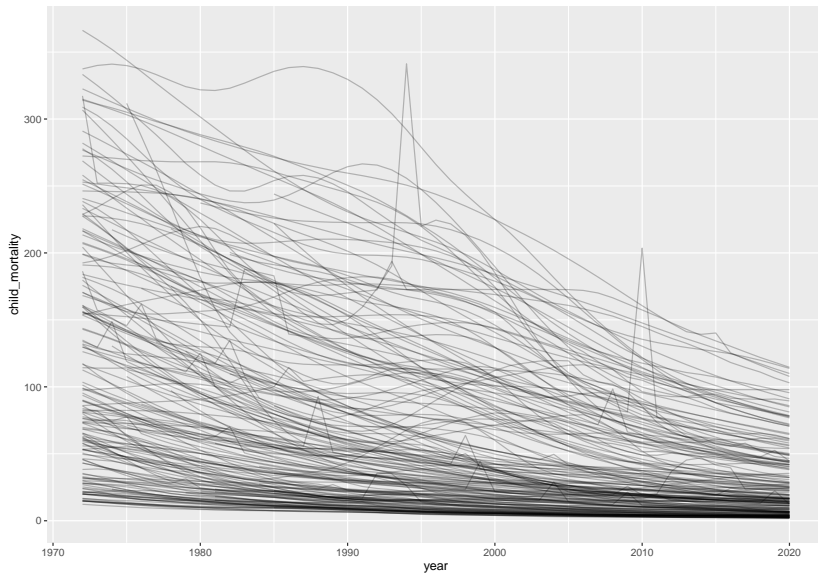
```
mortality_long <- mortality |>
  select(-indicator) |>
  pivot_longer(
    cols=`1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  ) |>
  mutate(year = as.integer(year))
mortality_long
```

```
## # A tibble: 10,633 x 4
##   country country_code year child_mortality
##   <chr>    <chr>      <int>      <dbl>
## 1 Aruba    ABW         1972         NA
## 2 Aruba    ABW         1973         NA
## 3 Aruba    ABW         1974         NA
## 4 Aruba    ABW         1975         NA
## 5 Aruba    ABW         1976         NA
## 6 Aruba    ABW         1977         NA
## 7 Aruba    ABW         1978         NA
## 8 Aruba    ABW         1979         NA
## 9 Aruba    ABW         1980         NA
```

Let's (re)do line plots!

```
mortality_long |>  
  ggplot(mapping = aes(x = year, y = child_mortality, group= country)) +  
  geom_line(alpha=0.25)
```

There we go




```
library(TPDData)
spotify
```

```
## # A tibble: 490 x 54
##   Track N-1 Artist week1 week2 week3 week4 week5 week6 week7 week8 week9 week10
##   <chr>      <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 The Box   Roddy~    1     1     1     1     1     1     1     1     1     1
## 2 ROXANNE   Arizo~    2     4     5     4     4     4     6     7     9    21
## 3 Yummy     Justi~    3     6    17    17    17    24    15    32    NA    NA
## 4 Circles   Post ~    4     7     9    10     7    10    11    10    17    26
## 5 BOP       DaBaby    5     5     7     5    11    12    18    18    32    47
## 6 Falling   Trevo~    6     8    10     7     6     8    10    11    18    28
## 7 Dance Mo~ Tones~    7    13    13    12    12    13    17    13    21    33
## 8 Bandit (~ Juice~    8    11    14    14    15    20    27    26    42    NA
## 9 Futsal S~ Lil U~    9     9    19    21    24    32    40    49    NA    NA
## 10 everythi~ Billi~   10    17    28     9     8    11    14    17    29    NA
## # ... with 480 more rows, 42 more variables: week11 <dbl>, week12 <dbl>,
## #   week13 <dbl>, week14 <dbl>, week15 <dbl>, week16 <dbl>, week17 <dbl>,
## #   week18 <dbl>, week19 <dbl>, week20 <dbl>, week21 <dbl>, week22 <dbl>,
## #   week23 <dbl>, week24 <dbl>, week25 <dbl>, week26 <dbl>, week27 <dbl>,
## #   week28 <dbl>, week29 <dbl>, week30 <dbl>, week31 <dbl>, week32 <dbl>,
## #   week33 <dbl>, week34 <dbl>, week35 <dbl>, week36 <dbl>, week37 <dbl>,
## #   week38 <dbl>, week39 <dbl>, week40 <dbl>, week41 <dbl>, week42 <dbl>, ...
```

Last approach isn't ideal because of the week prefix:

```
spotify |>
  pivot_longer(
    cols = c(`Track Name`, -Artist),
    names_to = "week_of_year",
    values_to = "rank"
  )
```

```
## # A tibble: 25,480 x 4
##   `Track Name` Artist      week_of_year  rank
##   <chr>         <chr>         <chr>      <dbl>
## 1 The Box      Roddy Ricch week1         1
## 2 The Box      Roddy Ricch week2         1
## 3 The Box      Roddy Ricch week3         1
## 4 The Box      Roddy Ricch week4         1
## 5 The Box      Roddy Ricch week5         1
## 6 The Box      Roddy Ricch week6         1
## 7 The Box      Roddy Ricch week7         1
## 8 The Box      Roddy Ricch week8         1
## 9 The Box      Roddy Ricch week9         1
## 10 The Box     Roddy Ricch week10        1
## # ... with 25,470 more rows
```

Removing a column name prefix

```
spotify |>
  pivot_longer(
    cols = c(`Track Name`, -Artist),
    names_to = "week_of_year",
    values_to = "rank",
    names_prefix = "week"
  ) |>
  mutate(
    week_of_year = as.integer(week_of_year)
  )
```

Removing a column name prefix

```
## # A tibble: 25,480 x 4
##   `Track Name` Artist      week_of_year rank
##   <chr>         <chr>         <int> <dbl>
## 1 The Box      Roddy Ricch          1     1
## 2 The Box      Roddy Ricch          2     1
## 3 The Box      Roddy Ricch          3     1
## 4 The Box      Roddy Ricch          4     1
## 5 The Box      Roddy Ricch          5     1
## 6 The Box      Roddy Ricch          6     1
## 7 The Box      Roddy Ricch          7     1
## 8 The Box      Roddy Ricch          8     1
## 9 The Box      Roddy Ricch          9     1
## 10 The Box     Roddy Ricch         10     1
## # ... with 25,470 more rows
```

6. Joining datasets

```
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 6
##   country      continent  year lifeExp      pop gdpPercap
##   <fct>        <fct>    <int>  <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952   28.8  8425333    779.
## 2 Afghanistan Asia      1957   30.3  9240934    821.
## 3 Afghanistan Asia      1962   32.0 10267083    853.
## 4 Afghanistan Asia      1967   34.0 11537966    836.
## 5 Afghanistan Asia      1972   36.1 13079460    740.
## 6 Afghanistan Asia      1977   38.4 14880372    786.
## 7 Afghanistan Asia      1982   39.9 12881816    978.
## 8 Afghanistan Asia      1987   40.8 13867957    852.
## 9 Afghanistan Asia      1992   41.7 16317921    649.
## 10 Afghanistan Asia      1997   41.8 22227415    635.
## # ... with 1,694 more rows
```

What if we want to add the `child_mortality` variable to the `gapminder` data?

Just add the columns?

```
gapminder |>
  select(country, year) |>
  head()
```

```
## # A tibble: 6 x 2
##   country      year
##   <fct>      <int>
## 1 Afghanistan 1952
## 2 Afghanistan 1957
## 3 Afghanistan 1962
## 4 Afghanistan 1967
## 5 Afghanistan 1972
## 6 Afghanistan 1977
```

```
mortality_long |>
  select(country, year) |>
  head()
```

```
## # A tibble: 6 x 2
##   country year
##   <chr>  <int>
## 1 Aruba   1972
## 2 Aruba   1973
## 3 Aruba   1974
## 4 Aruba   1975
## 5 Aruba   1976
## 6 Aruba   1977
```

Rows are not aligned properly!

A **primary key** is a variable or set of variables that uniquely identifies rows in the data

- {country, year} in the gapminder data

A **foreign key** is the corresponding variable(s) in another table

- {country, year} in the mortality_long data

If we align the two tables based on these variables, we can add variables from one to the other

Checking that the keys are unique

Things get weird if these keys are not unique. Let's check.

Checking primary key is unique

```
gapminder |>  
  count(country, year) |>  
  filter(n>1)
```

```
## # A tibble: 0 x 3
```

Checking foreign key is unique

```
mortality_long |>  
  count(country, year) |>  
  filter(n>1)
```

```
## # A tibble: 0 x 3
```

left_join(): add variables to primary table

left_join() keeps all rows from the first argument/piped data:

```
gapminder |>
  left_join(mortality_long) |>
  select(country, year, lifeExp, pop, gdpPercap, child_mortality) |>
  head(n = 6)
```

```
## Joining with `by = join_by(country, year)`
```

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

	country	year	lifeExp	pop	gdpPercap	child_mortality
	<chr>	<int>	<dbl>	<int>	<dbl>	<dbl>
## 1	Afghanistan	1952	28.8	8425333	779.	NA
## 2	Afghanistan	1957	30.3	9240934	821.	NA
## 3	Afghanistan	1962	32.0	10267083	853.	NA
## 4	Afghanistan	1967	34.0	11537966	836.	NA
## 5	Afghanistan	1972	36.1	13079460	740.	291
## 6	Afghanistan	1977	38.4	14880372	786.	262.

Rows in primary table not in foreign table: new values are NA

inner_join(): add and filter

`inner_join()` adds the variables from the foreign table and filters to rows in both tables:

```
gapminder |>
  inner_join(mortality_long) |>
  select(country, year, lifeExp, pop, gdpPercap, child_mortality) |>
  head(n=6)
```

```
## Joining with `by = join_by(country, year)`
```

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

##	country	year	lifeExp	pop	gdpPercap	child_mortality
##	<chr>	<int>	<dbl>	<int>	<dbl>	<dbl>
## 1	Afghanistan	1972	36.1	13079460	740.	291
## 2	Afghanistan	1977	38.4	14880372	786.	262.
## 3	Afghanistan	1982	39.9	12881816	978.	231.
## 4	Afghanistan	1987	40.8	13867957	852.	198.
## 5	Afghanistan	1992	41.7	16317921	649.	166.
## 6	Afghanistan	1997	41.8	22227415	635.	142.

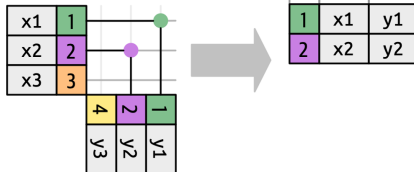
How inner joins work

Two data sets:

x		y	
1	x1	1	y1
2	x2	2	y2
3	x3	4	y3

Finding matching keys:

`inner_join(x, y)`

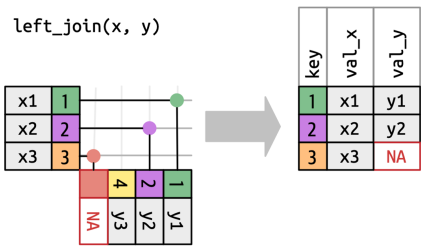


How left joins work

Two data sets:

x		y	
1	x1	1	y1
2	x2	2	y2
3	x3	4	y3

Keep all x keys:



More complicated example

```
library(nycflights13)
flights2 <- flights |>
  select(year, time_hour, origin, dest, tailnum, carrier)
flights2
```

```
## # A tibble: 336,776 x 6
##   year time_hour      origin dest  tailnum carrier
##   <int> <dtm>      <chr>  <chr> <chr>  <chr>
## 1  2013 2013-01-01 05:00:00 EWR   IAH   N14228  UA
## 2  2013 2013-01-01 05:00:00 LGA   IAH   N24211  UA
## 3  2013 2013-01-01 05:00:00 JFK   MIA   N619AA  AA
## 4  2013 2013-01-01 05:00:00 JFK   BQN   N804JB  B6
## 5  2013 2013-01-01 06:00:00 LGA   ATL   N668DN  DL
## 6  2013 2013-01-01 05:00:00 EWR   ORD   N39463  UA
## 7  2013 2013-01-01 06:00:00 EWR   FLL   N516JB  B6
## 8  2013 2013-01-01 06:00:00 LGA   IAD   N829AS  EV
## 9  2013 2013-01-01 06:00:00 JFK   MCO   N593JB  B6
## 10 2013 2013-01-01 06:00:00 LGA   ORD   N3ALAA  AA
## # ... with 336,766 more rows
```

```
planes2 <- planes |>
  select(tailnum, year, type, engine, seats)
planes2
```

```
## # A tibble: 3,322 x 5
##   tailnum  year type                engine      seats
##   <chr>    <int> <chr>                <chr>    <int>
## 1 N10156   2004 Fixed wing multi engine Turbo-fan    55
## 2 N102UW   1998 Fixed wing multi engine Turbo-fan   182
## 3 N103US   1999 Fixed wing multi engine Turbo-fan   182
## 4 N104UW   1999 Fixed wing multi engine Turbo-fan   182
## 5 N10575   2002 Fixed wing multi engine Turbo-fan    55
## 6 N105UW   1999 Fixed wing multi engine Turbo-fan   182
## 7 N107US   1999 Fixed wing multi engine Turbo-fan   182
## 8 N108UW   1999 Fixed wing multi engine Turbo-fan   182
## 9 N109UW   1999 Fixed wing multi engine Turbo-fan   182
## 10 N110UW  1999 Fixed wing multi engine Turbo-fan   182
## # ... with 3,312 more rows
```

What happens with naive join?

```
flights2 |>
  left_join(planes2)
```

```
## Joining with `by = join_by(year, tailnum)`
```

```
## # A tibble: 336,776 x 9
```

##	year	time_hour	origin	dest	tailnum	carrier	type	engine	seats
##	<int>	<dtm>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<int>
## 1	2013	2013-01-01 05:00:00	EWB	IAH	N14228	UA	<NA>	<NA>	NA
## 2	2013	2013-01-01 05:00:00	LGA	IAH	N24211	UA	<NA>	<NA>	NA
## 3	2013	2013-01-01 05:00:00	JFK	MIA	N619AA	AA	<NA>	<NA>	NA
## 4	2013	2013-01-01 05:00:00	JFK	BQN	N804JB	B6	<NA>	<NA>	NA
## 5	2013	2013-01-01 06:00:00	LGA	ATL	N668DN	DL	<NA>	<NA>	NA
## 6	2013	2013-01-01 05:00:00	EWB	ORD	N39463	UA	<NA>	<NA>	NA
## 7	2013	2013-01-01 06:00:00	EWB	FLL	N516JB	B6	<NA>	<NA>	NA
## 8	2013	2013-01-01 06:00:00	LGA	IAD	N829AS	EV	<NA>	<NA>	NA
## 9	2013	2013-01-01 06:00:00	JFK	MCO	N593JB	B6	<NA>	<NA>	NA
## 10	2013	2013-01-01 06:00:00	LGA	ORD	N3ALAA	AA	<NA>	<NA>	NA

```
## # ... with 336,766 more rows
```


Specify the joining variables

```
flights2 |>
  left_join(planes2, by = "tailnum")
```

```
## # A tibble: 336,776 x 10
##   year.x time_hour      origin dest  tailnum carrier year.y type    engine
##   <int> <dtm>         <chr> <chr> <chr>   <chr>   <int> <chr>  <chr>
## 1  2013 2013-01-01 05:00:00 EWR   IAH   N14228  UA     1999 Fixed ~ Turbo-
## 2  2013 2013-01-01 05:00:00 LGA   IAH   N24211  UA     1998 Fixed ~ Turbo-
## 3  2013 2013-01-01 05:00:00 JFK   MIA   N619AA  AA     1990 Fixed ~ Turbo-
## 4  2013 2013-01-01 05:00:00 JFK   BQN   N804JB  B6     2012 Fixed ~ Turbo-
## 5  2013 2013-01-01 06:00:00 LGA   ATL   N668DN  DL     1991 Fixed ~ Turbo-
## 6  2013 2013-01-01 05:00:00 EWR   ORD   N39463  UA     2012 Fixed ~ Turbo-
## 7  2013 2013-01-01 06:00:00 EWR   FLL   N516JB  B6     2000 Fixed ~ Turbo-
## 8  2013 2013-01-01 06:00:00 LGA   IAD   N829AS  EV     1998 Fixed ~ Turbo-
## 9  2013 2013-01-01 06:00:00 JFK   MCO   N593JB  B6     2004 Fixed ~ Turbo-
## 10 2013 2013-01-01 06:00:00 LGA   ORD   N3ALAA  AA     NA <NA>  <NA>
## # ... with 336,766 more rows, and 1 more variable: seats <int>
```

Change variables names

```
flights2 |>
  left_join(planes2 |> rename(manufacture_year = year))
```

```
## Joining with `by = join_by(tailnum)`
```

```
## # A tibble: 336,776 x 10
```

```
##   year time_hour      origin dest tailnum carrier manufac~1 type engine
##   <int> <dtm>      <chr>  <chr> <chr>  <chr>      <int> <chr> <chr>
## 1  2013 2013-01-01 05:00:00 EWR   IAH   N14228  UA        1999 Fixe~ Turbo~
## 2  2013 2013-01-01 05:00:00 LGA   IAH   N24211  UA        1998 Fixe~ Turbo~
## 3  2013 2013-01-01 05:00:00 JFK   MIA   N619AA  AA        1990 Fixe~ Turbo~
## 4  2013 2013-01-01 05:00:00 JFK   BQN   N804JB  B6        2012 Fixe~ Turbo~
## 5  2013 2013-01-01 06:00:00 LGA   ATL   N668DN  DL        1991 Fixe~ Turbo~
## 6  2013 2013-01-01 05:00:00 EWR   ORD   N39463  UA        2012 Fixe~ Turbo~
## 7  2013 2013-01-01 06:00:00 EWR   FLL   N516JB  B6        2000 Fixe~ Turbo~
## 8  2013 2013-01-01 06:00:00 LGA   IAD   N829AS  EV        1998 Fixe~ Turbo~
## 9  2013 2013-01-01 06:00:00 JFK   MCO   N593JB  B6        2004 Fixe~ Turbo~
## 10 2013 2013-01-01 06:00:00 LGA   ORD   N3ALAA  AA        NA <NA> <NA>
## # ... with 336,766 more rows, 1 more variable: seats <int>, and abbreviated
## #   variable name 1: manufacture_year
```