

Gov 50: 11. Tidying and Joining Data

Matthew Blackwell

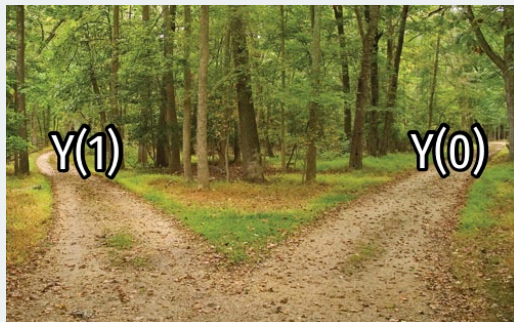
Harvard University

Roadmap

1. Causality review
2. Pivoting data longer
3. Joining data sets

1/ Causality review

Potential outcomes



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

COVID-19 vaccine trials



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

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

2/ Pivoting data longer

Mortality data

```
library(tidyverse)
library(gov50data)
mortality
```

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

```
##   country      country_code indicator `1972` `1973` `1974`
##   <chr>        <chr>         <chr>   <dbl> <dbl> <dbl>
## 1 Aruba        ABW           Mortalit~ NA     NA     NA
## 2 Afghanistan AFG           Mortalit~ 291    285.   280.
## 3 Angola       AGO           Mortalit~ NA     NA     NA
## 4 Albania      ALB           Mortalit~ NA     NA     NA
## 5 Andorra      AND           Mortalit~ NA     NA     NA
## 6 United Arab ~ ARE           Mortalit~ 80.1   72.6   65.7
## 7 Argentina    ARG           Mortalit~ 69.7   68.2   66.1
## 8 Armenia      ARM           Mortalit~ NA     NA     NA
## 9 American Sam~ ASM           Mortalit~ NA     NA     NA
## 10 Antigua and ~ ATG           Mortalit~ 26.9   25.1   23.5
```

```
## # i 207 more rows
```

```
## # i 46 more variables: `1975` <dbl>, `1976` <dbl>,
## #   `1977` <dbl>, `1978` <dbl>, `1979` <dbl>, `1980` <dbl>,
## #   `1981` <dbl>, `1982` <dbl>, `1983` <dbl>, `1984` <dbl>,
## #   `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>,
```

Pivoting longer

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"
  )
```

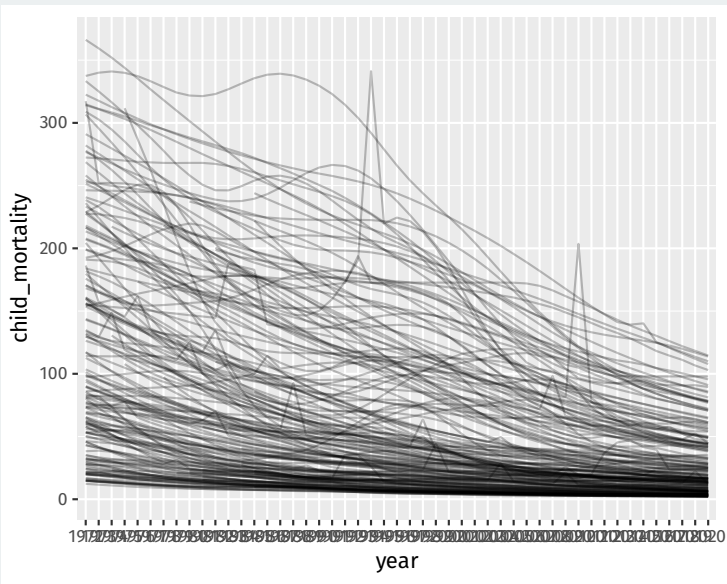
```
## # A tibble: 10,633 x 4
```

```
##   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
## # i 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 = country)) +
  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
```

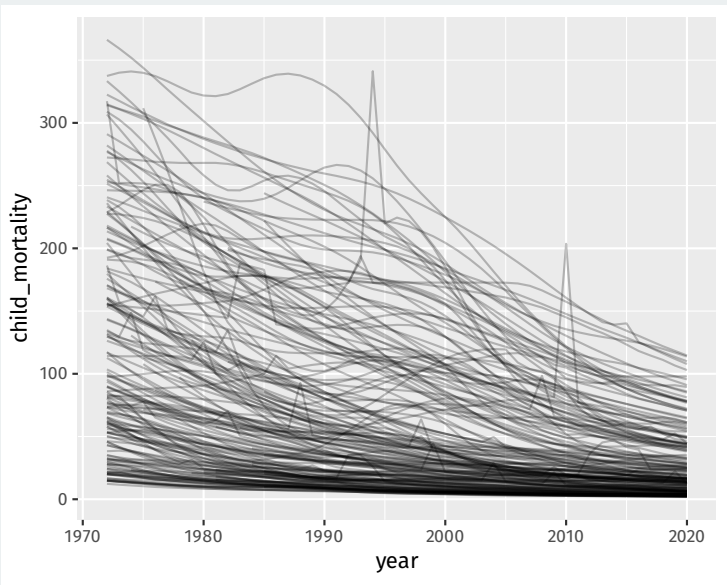
```
## # A tibble: 10,633 x 4
```

```
##   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
```

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



Spotify data

```
spotify
```

```
## # A tibble: 490 x 54
##   `Track Name`   Artist week1 week2 week3 week4 week5 week6
##   <chr>          <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 The Box       Roddy~    1     1     1     1     1     1
## 2 ROXANNE       Arizo~    2     4     5     4     4     4
## 3 Yummy         Justi~    3     6    17    17    17    24
## 4 Circles       Post ~    4     7     9    10     7    10
## 5 BOP           DaBaby    5     5     7     5    11    12
## 6 Falling       Trevo~    6     8    10     7     6     8
## 7 Dance Monkey  Tones~    7    13    13    12    12    13
## 8 Bandit (with ~ Juice~    8    11    14    14    15    20
## 9 Futsal Shuffl~ Lil U~    9     9    19    21    24    32
## 10 everything i ~ Billi~   10    17    28     9     8    11
## # i 480 more rows
## # i 46 more variables: week7 <dbl>, week8 <dbl>,
## #   week9 <dbl>, week10 <dbl>, 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>, ...
```

Pivoting not ideal

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
```


Removing a column name prefix

When the data in the column name has a fixed prefix, we can use the `names_prefix` to remove it when moving the data to rows

```
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
## # i 25,470 more rows
```

3/ Joining data sets

Gapminder data

```
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.
## # i 1,694 more rows
```

Joining data sets

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

Just add the columns? Rows are not aligned properly!

```
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
```

Key variables

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:

```
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	8.43e6	779.	NA
## 2	Afghanistan	1957	30.3	9.24e6	821.	NA
## 3	Afghanistan	1962	32.0	1.03e7	853.	NA
## 4	Afghanistan	1967	34.0	1.15e7	836.	NA
## 5	Afghanistan	1972	36.1	1.31e7	740.	291
## 6	Afghanistan	1977	38.4	1.49e7	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
```

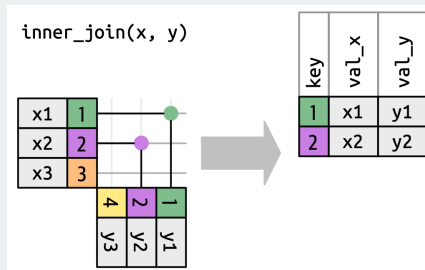
	country	year	lifeExp	pop	gdpPercap	child_mortality
	<chr>	<int>	<dbl>	<int>	<dbl>	<dbl>
## 1	Afghanistan	1972	36.1	1.31e7	740.	291
## 2	Afghanistan	1977	38.4	1.49e7	786.	262.
## 3	Afghanistan	1982	39.9	1.29e7	978.	231.
## 4	Afghanistan	1987	40.8	1.39e7	852.	198.
## 5	Afghanistan	1992	41.7	1.63e7	649.	166.
## 6	Afghanistan	1997	41.8	2.22e7	635.	142.

How inner joins work

Two data sets:

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

Find matching keys:

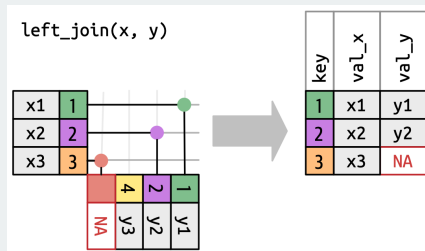


How left joins work

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
## # i 336,766 more rows
```

Planes data

```
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
## # i 3,312 more rows
```

year here is manufacture year.

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
##   <int> <dtm>          <chr>  <chr> <chr>    <chr>   <chr> <chr>
## 1  2013 2013-01-01 05:00:00 EWR    IAH    N14228  UA     <NA>  <NA>
## 2  2013 2013-01-01 05:00:00 LGA    IAH    N24211  UA     <NA>  <NA>
## 3  2013 2013-01-01 05:00:00 JFK    MIA    N619AA  AA     <NA>  <NA>
## 4  2013 2013-01-01 05:00:00 JFK    BQN    N804JB  B6     <NA>  <NA>
## 5  2013 2013-01-01 06:00:00 LGA    ATL    N668DN  DL     <NA>  <NA>
## 6  2013 2013-01-01 05:00:00 EWR    ORD    N39463  UA     <NA>  <NA>
## 7  2013 2013-01-01 06:00:00 EWR    FLL    N516JB  B6     <NA>  <NA>
## 8  2013 2013-01-01 06:00:00 LGA    IAD    N829AS  EV     <NA>  <NA>
## 9  2013 2013-01-01 06:00:00 JFK    MCO    N593JB  B6     <NA>  <NA>
## 10 2013 2013-01-01 06:00:00 LGA    ORD    N3ALAA  AA     <NA>  <NA>
## # i 336,766 more rows
## # i 1 more variable: seats <int>
```

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
##   <int> <dtm>      <chr>  <chr> <chr>    <chr>    <int>
## 1  2013 2013-01-01 05:00:00 EWR   IAH    N14228  UA      1999
## 2  2013 2013-01-01 05:00:00 LGA   IAH    N24211  UA      1998
## 3  2013 2013-01-01 05:00:00 JFK   MIA    N619AA  AA      1990
## 4  2013 2013-01-01 05:00:00 JFK   BQN    N804JB  B6      2012
## 5  2013 2013-01-01 06:00:00 LGA   ATL    N668DN  DL      1991
## 6  2013 2013-01-01 05:00:00 EWR   ORD    N39463  UA      2012
## 7  2013 2013-01-01 06:00:00 EWR   FLL    N516JB  B6      2000
## 8  2013 2013-01-01 06:00:00 LGA   IAD    N829AS  EV      1998
## 9  2013 2013-01-01 06:00:00 JFK   MCO    N593JB  B6      2004
## 10 2013 2013-01-01 06:00:00 LGA   ORD    N3ALAA  AA      NA
```

```
## # i 336,766 more rows
```

```
## # i 3 more variables: type <chr>, engine <chr>, 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
##   <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
```

```
## # i 336,766 more rows
```

```
## # i 4 more variables: manufacture_year <int>, type <chr>,
```

```
## #   engine <chr>, seats <int>
```