Notes on Ch 5: Data Tidying

N. Lim

2025-06-30

Prerequisites

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Tidy data

Rules for making a dataset tidy: 1. Each variable is a column; each column is a variable. 2. Each observation is a row; each row is an ovservation. 3. Each value is a cell; each cell is a single value.

Advantages of having a tidy data: - having a consistent data structure makes it easier to learn the tools that work with it because of uniformity. - allows R's vectorized functions to shine. Packages in the tidyverse like dplyr, ggplot2, etc. are designed to work with tidy data.

Examples:

```
# Compute rate per 10,000
table1 |>
  mutate(rate = cases / population * 10000)
```

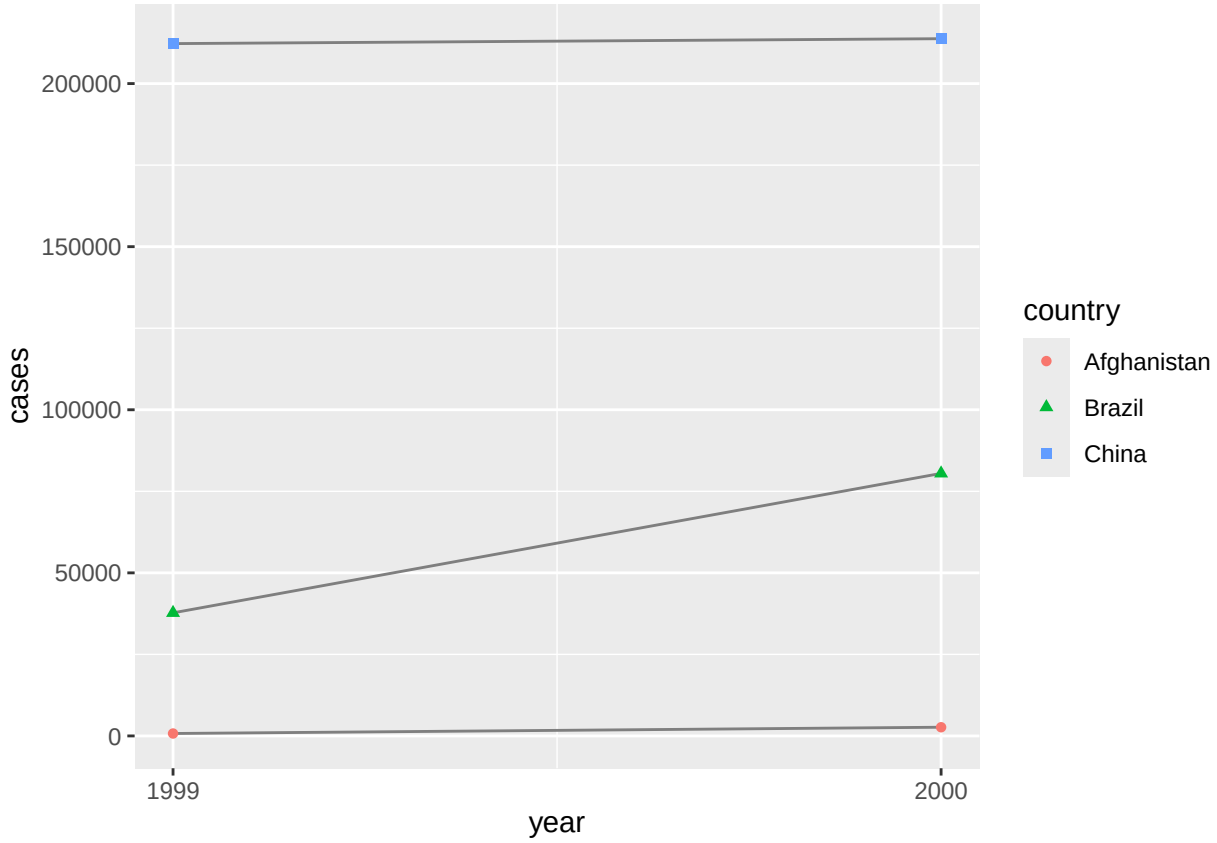
```
## # A tibble: 6 x 5
##   country      year  cases population  rate
##   <chr>       <dbl>  <dbl>      <dbl> <dbl>
## 1 Afghanistan  1999    745   19987071 0.373
## 2 Afghanistan  2000   2666   20595360 1.29
## 3 Brazil       1999  37737  172006362 2.19
## 4 Brazil       2000  80488  174504898 4.61
## 5 China        1999 212258 1272915272 1.67
## 6 China        2000 213766 1280428583 1.67
```

```
# Compute total cases per year
table1 |>
  group_by(year) |>
  summarize(total_cases = sum(cases))
```

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

```
##    year total_cases
##   <dbl>       <dbl>
## 1  1999      250740
## 2  2000      296920
```

```
# Visualize changes over time
ggplot(table1, aes(x = year, y = cases)) +
  geom_line(aes(group = country), color = "grey50") +
  geom_point(aes(color = country, shape = country)) +
  scale_x_continuous(breaks = c(1999, 2000))
```



Lengthening data

Reasons why most real data is untidy: 1. Data is often organized to facilitate some other goal other than analysis (for example, to make data entry easier). 2. Most people aren't familiar with the principles of tidy data.

Data in column names

```
billboard
```

```
## # A tibble: 317 x 79
##    artist     track date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
##    <chr>      <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 2 Pac      Baby~ 2000-02-26      87    82    72    77    87    94    99    NA
##  2 2Ge+her    The ~ 2000-09-02      91    87    92    NA    NA    NA    NA    NA
##  3 3 Doors D~ Kryp~ 2000-04-08      81    70    68    67    66    57    54    53
```

```
##  4 3 Doors D~ Loser 2000-10-21      76    76    72    69    67    65    55    59
##  5 504 Boyz   Wobb~ 2000-04-15      57    34    25    17    17    31    36    49
##  6 98^0       Give~ 2000-08-19      51    39    34    26    26    19     2     2
##  7 A*Teens    Danc~ 2000-07-08      97    97    96    95   100    NA    NA    NA
##  8 Aaliyah    I Do~ 2000-01-29      84    62    51    41    38    35    35    38
##  9 Aaliyah    Try ~ 2000-03-18      59    53    38    28    21    18    16    14
## 10 Adams, Yo~ Open~ 2000-08-26      76    76    74    69    68    67    61    58
## # i 307 more rows
## # i 68 more variables: wk9 <dbl>, wk10 <dbl>, wk11 <dbl>, wk12 <dbl>,
## #   wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>,
## #   wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>,
## #   wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>,
## #   wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,
## #   wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, wk42 <dbl>, ...
```

Observations on `billboard` data: - Each observation is a song. - First 3 columns (`artist`, `track`, `date.entered`) are variables that describe the song. - The proceeding columns describe the rank of the song each week for 76 weeks.

Tidying the `billboard` data using `pivot_longer()`:

```
billboard |>
  pivot_longer(
    cols = starts_with("Wk"),
    names_to = "Week",
    values_to = "rank"
  )
```

```
## # A tibble: 24,092 x 5
##    artist track                   date.entered Week   rank
##    <chr>  <chr>                   <date>       <chr> <dbl>
##  1 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk1      87
##  2 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk2      82
##  3 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk3      72
##  4 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk4      77
##  5 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk5      87
##  6 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk6      94
##  7 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk7      99
##  8 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk8      NA
##  9 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk9      NA
## 10 2 Pac  Baby Don't Cry (Keep... 2000-02-26   wk10     NA
## # i 24,082 more rows
```

This tidying approach generated `NA`s. We can get rid of them during tidying by using the `values_drop_na = TRUE` argument:

```
billboard |>
  pivot_longer(
    cols = starts_with("Wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
  )
```

```
## # A tibble: 5,307 x 5
##    artist  track                   date.entered week   rank
##    <chr>   <chr>                   <date>       <chr> <dbl>
```

```
##  1 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk1      87
##  2 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk2      82
##  3 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk3      72
##  4 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk4      77
##  5 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk5      87
##  6 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk6      94
##  7 2 Pac   Baby Don't Cry (Keep... 2000-02-26   wk7      99
##  8 2Ge+her The Hardest Part Of ... 2000-09-02   wk1      91
##  9 2Ge+her The Hardest Part Of ... 2000-09-02   wk2      87
## 10 2Ge+her The Hardest Part Of ... 2000-09-02   wk3      92
## # i 5,297 more rows
```

Note: the number or rows decreased.

Now, what happens if a song is in the top 100 for more than 76 weeks?

While there is no data available after the 76th week, it can be computed from the dataset. First, we must parse the week number to make it numerical:

Parsing the week column (using ‘readr::parse_number()“):

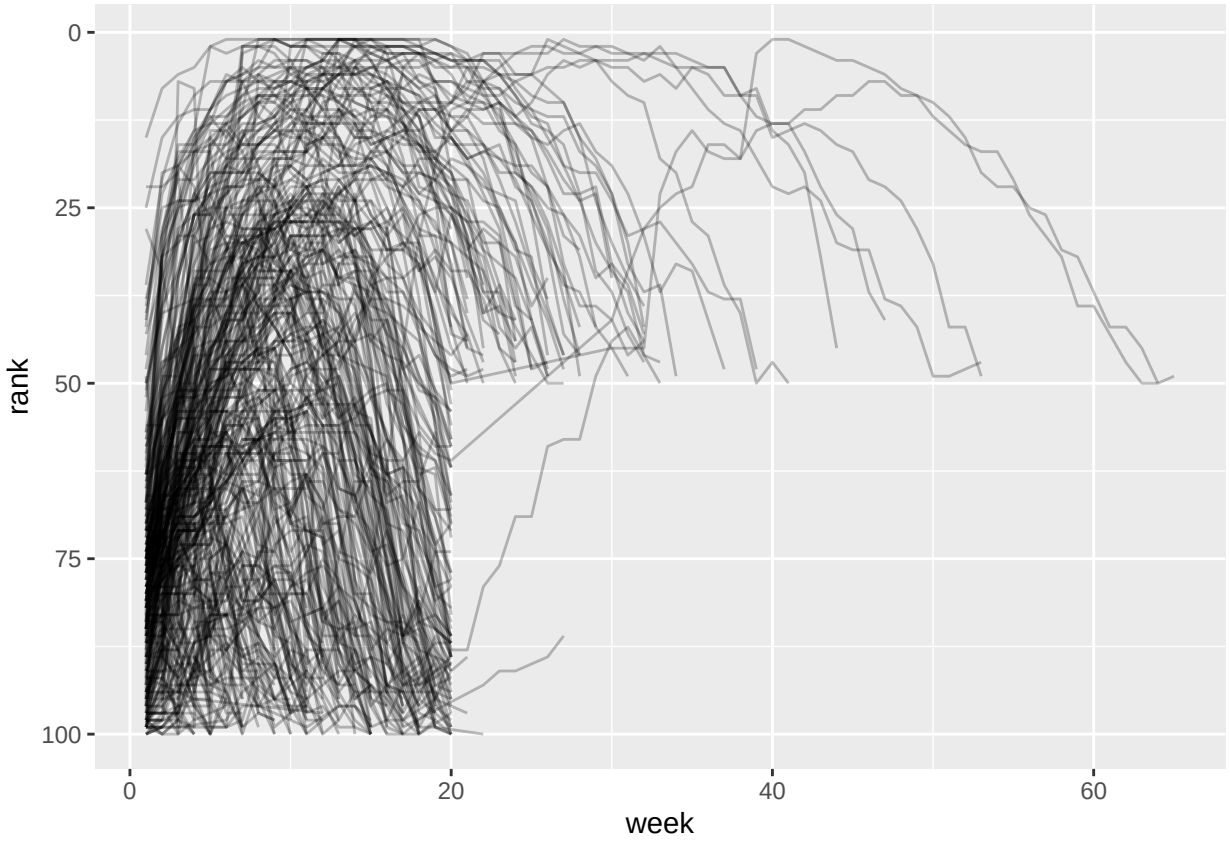
```
billboard_longer <- billboard |>
  pivot_longer(
    cols = starts_with("Wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
  ) |>
  mutate(
    week = parse_number(week)
  )

billboard_longer
```

```
## # A tibble: 5,307 x 5
##    artist  track                   date.entered  week  rank
##    <chr>   <chr>                   <date>       <dbl> <dbl>
##  1 2 Pac   Baby Don't Cry (Keep... 2000-02-26       1    87
##  2 2 Pac   Baby Don't Cry (Keep... 2000-02-26       2    82
##  3 2 Pac   Baby Don't Cry (Keep... 2000-02-26       3    72
##  4 2 Pac   Baby Don't Cry (Keep... 2000-02-26       4    77
##  5 2 Pac   Baby Don't Cry (Keep... 2000-02-26       5    87
##  6 2 Pac   Baby Don't Cry (Keep... 2000-02-26       6    94
##  7 2 Pac   Baby Don't Cry (Keep... 2000-02-26       7    99
##  8 2Ge+her The Hardest Part Of ... 2000-09-02       1    91
##  9 2Ge+her The Hardest Part Of ... 2000-09-02       2    87
## 10 2Ge+her The Hardest Part Of ... 2000-09-02       3    92
## # i 5,297 more rows
```

Visualizing how song ranking varies over time:

```
billboard_longer |>
  ggplot(aes(x = week, y = rank, group = track)) +
  geom_line(alpha = 0.25) +
  scale_y_reverse()
```



We can see that no song can stay in the top 100 for more than 20 weeks.

How does pivoting work?

```
df <- tribble(
  ~id,  ~bp1, ~bp2,
   "A",  100,  120,
   "B",  140,  115,
   "C",  120,  125
)
```

where `id` stands for patient id, `bp1` and `bp2` are two blood pressure measurements for each patient.

To tidy the data, we need three variables: `id` (already exists), `measurement` (bp1 or bp2), and `value` (the readings).

Tidying the data using `pivot_longer()`:

```
df |>
  pivot_longer(
    cols = bp1:bp2,
    names_to = "measurement",
    values_to = "value"
  )
```

```
## # A tibble: 6 x 3
##   id    measurement value
##   <chr> <chr>       <dbl>
```

```
## 1 A     bp1           100
## 2 A     bp2           120
## 3 B     bp1           140
## 4 B     bp2           115
## 5 C     bp1           120
## 6 C     bp2           125
```

When there are any variables in column names

Observe the following data:

```
who2
```

```
## # A tibble: 7,240 x 58
##    country      year sp_m_014 sp_m_1524 sp_m_2534 sp_m_3544 sp_m_4554 sp_m_5564
##    <chr>       <dbl>    <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1 Afghanistan  1980       NA        NA        NA        NA        NA        NA
##  2 Afghanistan  1981       NA        NA        NA        NA        NA        NA
##  3 Afghanistan  1982       NA        NA        NA        NA        NA        NA
##  4 Afghanistan  1983       NA        NA        NA        NA        NA        NA
##  5 Afghanistan  1984       NA        NA        NA        NA        NA        NA
##  6 Afghanistan  1985       NA        NA        NA        NA        NA        NA
##  7 Afghanistan  1986       NA        NA        NA        NA        NA        NA
##  8 Afghanistan  1987       NA        NA        NA        NA        NA        NA
##  9 Afghanistan  1988       NA        NA        NA        NA        NA        NA
## 10 Afghanistan  1989       NA        NA        NA        NA        NA        NA
## # i 7,230 more rows
## # i 50 more variables: sp_m_65 <dbl>, sp_f_014 <dbl>, sp_f_1524 <dbl>,
## #   sp_f_2534 <dbl>, sp_f_3544 <dbl>, sp_f_4554 <dbl>, sp_f_5564 <dbl>,
## #   sp_f_65 <dbl>, sn_m_014 <dbl>, sn_m_1524 <dbl>, sn_m_2534 <dbl>,
## #   sn_m_3544 <dbl>, sn_m_4554 <dbl>, sn_m_5564 <dbl>, sn_m_65 <dbl>,
## #   sn_f_014 <dbl>, sn_f_1524 <dbl>, sn_f_2534 <dbl>, sn_f_3544 <dbl>,
## #   sn_f_4554 <dbl>, sn_f_5564 <dbl>, sn_f_65 <dbl>, ep_m_014 <dbl>, ...
```

Info and observations re `who2` data: - dataset was created by WHO and contains info about tuberculosis diagnoses. - first 2 columns: country and year - succeeding columns consist of sp_ or rel_ or ep_, then m_ or f_, plus some numbers. - they stand for measurement methods (sp/rel/ep), gender (m or f), and the age range (0-14, 15-24, etc.)

Now that we got to know the data, we can do the pivot operation.

```
who2 |>
  pivot_longer(
    cols = !(country:year),
    names_to = c("diagnosis", "gender", "age"),
    names_sep = "_",
    values_to = "count"
  )
```

```
## # A tibble: 405,440 x 6
##    country      year diagnosis gender age   count
##    <chr>       <dbl> <chr>     <chr>  <chr> <dbl>
##  1 Afghanistan  1980 sp        m      014      NA
##  2 Afghanistan  1980 sp        m      1524     NA
##  3 Afghanistan  1980 sp        m      2534     NA
##  4 Afghanistan  1980 sp        m      3544     NA
##  5 Afghanistan  1980 sp        m      4554     NA
```

```
##  6 Afghanistan  1980 sp        m      5564     NA
##  7 Afghanistan  1980 sp        m      65       NA
##  8 Afghanistan  1980 sp        f      014      NA
##  9 Afghanistan  1980 sp        f      1524     NA
## 10 Afghanistan  1980 sp        f      2534     NA
## # i 405,430 more rows
```

When data and variagble names are in column headers

What to do when the column names include a mix of values and variable names?

For example, consider this dataset:

```
household
```

```
## # A tibble: 5 x 5
##   family dob_child1 dob_child2 name_child1 name_child2
##    <int> <date>     <date>     <chr>       <chr>
## 1      1 1998-11-26 2000-01-29 Susan       Jose
## 2      2 1996-06-22 NA         Mark        <NA>
## 3      3 2002-07-11 2004-04-05 Sam         Seth
## 4      4 2004-10-10 2009-08-27 Craig       Khai
## 5      5 2000-12-05 2005-02-28 Parker      Gracie
```

- dataset contains info about 5 families: the date of birth for each child, and the names of each child.
- the column names contain 2 variables (dob and name), plus some qualifier (child1 or child2).

Pivoting using the `.value` "sentinel"

```
household |>
  pivot_longer(
    cols = !family,
    names_to = c(".value", "child"),
    names_sep = "_",
    values_drop_na = TRUE
  )
```

```
## # A tibble: 9 x 4
##   family child  dob        name
##    <int> <chr>  <date>     <chr>
## 1      1 child1 1998-11-26 Susan
## 2      1 child2 2000-01-29 Jose
## 3      2 child1 1996-06-22 Mark
## 4      3 child1 2002-07-11 Sam
## 5      3 child2 2004-04-05 Seth
## 6      4 child1 2004-10-10 Craig
## 7      4 child2 2009-08-27 Khai
## 8      5 child1 2000-12-05 Parker
## 9      5 child2 2005-02-28 Gracie
```

Widening data

When one observation is spread across multiple rows, we can use `pivot_wider()` to increase the number of columns and make the dataset "wider"

Example:

```
cms_patient_experience
```

```
## # A tibble: 500 x 5
##    org_pac_id org_nm                              measure_cd measure_title prf_rate
##    <chr>      <chr>                               <chr>      <chr>            <dbl>
##  1 0446157747 USC CARE MEDICAL GROUP INC          CAHPS_GRP~ CAHPS for MI~       63
##  2 0446157747 USC CARE MEDICAL GROUP INC          CAHPS_GRP~ CAHPS for MI~       87
##  3 0446157747 USC CARE MEDICAL GROUP INC          CAHPS_GRP~ CAHPS for MI~       86
##  4 0446157747 USC CARE MEDICAL GROUP INC          CAHPS_GRP~ CAHPS for MI~       57
##  5 0446157747 USC CARE MEDICAL GROUP INC          CAHPS_GRP~ CAHPS for MI~       85
##  6 0446157747 USC CARE MEDICAL GROUP INC          CAHPS_GRP~ CAHPS for MI~       24
##  7 0446162697 ASSOCIATION OF UNIVERSITY PHYSI~ CAHPS_GRP~ CAHPS for MI~       59
##  8 0446162697 ASSOCIATION OF UNIVERSITY PHYSI~ CAHPS_GRP~ CAHPS for MI~       85
##  9 0446162697 ASSOCIATION OF UNIVERSITY PHYSI~ CAHPS_GRP~ CAHPS for MI~       83
## 10 0446162697 ASSOCIATION OF UNIVERSITY PHYSI~ CAHPS_GRP~ CAHPS for MI~       63
## # i 490 more rows
```

- core unit being studied is an organization.
- each organization is spread across 6 rows.

Viewing the complete set of values for `measure_cd`, and `measure_title`

```
cms_patient_experience |>
  distinct(measure_cd, measure_title)
```

```
## # A tibble: 6 x 2
##   measure_cd   measure_title
##   <chr>        <chr>
## 1 CAHPS_GRP_1  CAHPS for MIPS SSM: Getting Timely Care, Appointments, and Infor~
## 2 CAHPS_GRP_2  CAHPS for MIPS SSM: How Well Providers Communicate
## 3 CAHPS_GRP_3  CAHPS for MIPS SSM: Patient's Rating of Provider
## 4 CAHPS_GRP_5  CAHPS for MIPS SSM: Health Promotion and Education
## 5 CAHPS_GRP_8  CAHPS for MIPS SSM: Courteous and Helpful Office Staff
## 6 CAHPS_GRP_12 CAHPS for MIPS SSM: Stewardship of Patient Resources
```

Pivoting the dataset using `pivot_wider()`

```
cms_patient_experience |>
  pivot_wider(
    names_from = measure_cd,
    values_from = prf_rate
  )
```

```
## # A tibble: 500 x 9
##    org_pac_id org_nm          measure_title CAHPS_GRP_1 CAHPS_GRP_2 CAHPS_GRP_3
##    <chr>      <chr>           <chr>               <dbl>       <dbl>       <dbl>
##  1 0446157747 USC CARE MEDICA~ CAHPS for MI~          63          NA          NA
##  2 0446157747 USC CARE MEDICA~ CAHPS for MI~          NA          87          NA
##  3 0446157747 USC CARE MEDICA~ CAHPS for MI~          NA          NA          86
##  4 0446157747 USC CARE MEDICA~ CAHPS for MI~          NA          NA          NA
##  5 0446157747 USC CARE MEDICA~ CAHPS for MI~          NA          NA          NA
##  6 0446157747 USC CARE MEDICA~ CAHPS for MI~          NA          NA          NA
##  7 0446162697 ASSOCIATION OF ~ CAHPS for MI~          59          NA          NA
##  8 0446162697 ASSOCIATION OF ~ CAHPS for MI~          NA          85          NA
##  9 0446162697 ASSOCIATION OF ~ CAHPS for MI~          NA          NA          83
## 10 0446162697 ASSOCIATION OF ~ CAHPS for MI~          NA          NA          NA
## # i 490 more rows
```

```
## # i 3 more variables: CAHPS_GRP_5 <dbl>, CAHPS_GRP_8 <dbl>, CAHPS_GRP_12 <dbl>
```

After pivoting wider, still there are multiple rows for each organization. This is because we did not supply to `pivot_wider()` information about which column/s can be considered as unique identifier (they have values that uniquely identify each row).

```
cms_patient_experience |>
  pivot_wider(
    id_cols = starts_with("org"),
    names_from = measure_cd,
    values_from = prf_rate
  )
```

```
## # A tibble: 95 x 8
##    org_pac_id org_nm CAHPS_GRP_1 CAHPS_GRP_2 CAHPS_GRP_3 CAHPS_GRP_5 CAHPS_GRP_8
##    <chr>      <chr>        <dbl>       <dbl>       <dbl>       <dbl>       <dbl>
##  1 0446157747 USC C~          63          87          86          57          85
##  2 0446162697 ASSOC~          59          85          83          63          88
##  3 0547164295 BEAVE~          49          NA          75          44          73
##  4 0749333730 CAPE ~          67          84          85          65          82
##  5 0840104360 ALLIA~          66          87          87          64          87
##  6 0840109864 REX H~          73          87          84          67          91
##  7 0840513552 SCL H~          58          83          76          58          78
##  8 0941545784 GRITM~          46          86          81          54          NA
##  9 1052612785 COMMU~          65          84          80          58          87
## 10 1254237779 OUR L~          61          NA          NA          65          NA
## # i 85 more rows
## # i 1 more variable: CAHPS_GRP_12 <dbl>
```

How does pivot_wider work??

It turns this data:

```
df <- tribble(
  ~id, ~measurement, ~value,
  "A",        "bp1",    100,
  "B",        "bp1",    140,
  "B",        "bp2",    115,
  "A",        "bp2",    120,
  "A",        "bp3",    105
)
```

into:

```
df |> pivot_wider(
  names_from = measurement,
  values_from = value
)
```

```
## # A tibble: 2 x 4
##   id      bp1   bp2   bp3
##   <chr> <dbl> <dbl> <dbl>
## 1 A       100   120   105
## 2 B       140   115    NA
```

- pivot_wider() effectively performs select-distinct-mutate operations:

```r
df |>
  select(-measurement, -value) |>
  distinct() |>
  mutate(
    x = NA, y = NA, z = NA
  )
```

```
## # A tibble: 2 x 4
##   id    x     y     z
##   <chr> <lgl> <lgl> <lgl>
## 1 A     NA    NA    NA
## 2 B     NA    NA    NA
```

Then it fills all the missing values using the source data.