More advanced tidyverse functions

Modern Research Methods

slides 2-26 adapted from https://datasciencebox.org/

22 September 2021

Joining data frames

- We have multiple data frames
- We want to bring them together

Data: Women in science

Information on 10 women in science who changed the world

name
Ada Lovelace
Marie Curie
Janaki Ammal
Chien-Shiung Wu
Katherine Johnson
Rosalind Franklin
Vera Rubin
Gladys West
Flossie Wong-Staal
Jennifer Doudna

Source:

Inputs

professions

```
professions %>%
  slice(1:3)
```

```
## # A tibble: 3 × 2
##   name          profession
##   <chr>         <chr>
## 1 Ada Lovelace Mathematician
## 2 Marie Curie  Physicist and Chemist
## 3 Janaki Ammal Botanist
```

dates

```
dates %>%
  slice(1:3)
```

```
## # A tibble: 3 × 3
##   name               birth_year death_year
##   <chr>                   <dbl>      <dbl>
## 1 Janaki Ammal             1897       1984
## 2 Chien-Shiung Wu          1912       1997
## 3 Katherine Johnson        1918       2020
```

works

```
works %>%
  slice(1:3)
```

```
## # A tibble: 3 × 2
##   name         known_for
##   <chr>        <chr>
## 1 Ada Lovelace first computer algorithm
## 2 Marie Curie  theory of radioactivity,  discovery of elements polonium and rad…
## 3 Janaki Ammal hybrid species, biodiversity protection
```

Desired output

```
## Joining, by = "name"
## Joining, by = "name"
```

```
## # A tibble: 10 × 5
##    name               profession      birth_year death_year known_for
##    <chr>              <chr>                <dbl>      <dbl> <chr>
##  1 Ada Lovelace       Mathematician           NA         NA first computer algo…
##  2 Marie Curie        Physicist and …         NA         NA theory of radioacti…
##  3 Janaki Ammal       Botanist              1897       1984 hybrid species, bio…
##  4 Chien-Shiung Wu    Physicist             1912       1997 confim and refine t…
##  5 Katherine Johnson  Mathematician         1918       2020 calculations of orb…
##  6 Rosalind Franklin  Chemist               1920       1958 <NA>
##  7 Vera Rubin         Astronomer            1928       2016 existence of dark m…
##  8 Gladys West        Mathematician         1930         NA mathematical modeli…
##  9 Flossie Wong-Staal Virologist and…       1947         NA first scientist to …
## 10 Jennifer Doudna    Biochemist            1964         NA one of the primary …
```

Inputs, reminder

```
names(professions)
```

```
## [1] "name"       "profession"
```

```
names(dates)
```

```
## [1] "name"       "birth_year" "death_year"
```

```
names(works)
```

```
## [1] "name"      "known_for"
```

```
nrow(professions)
```

```
## [1] 10
```

```
nrow(dates)
```

```
## [1] 8
```

```
nrow(works)
```

```
## [1] 9
```

```
something_join(x, y)
```

- `left_join()`: all rows from x
- `right_join()`: all rows from y
- `full_join()`: all rows from both x and y
- `semi_join()`: all rows from x where there are matching values in y, keeping just columns from x
- `inner_join()`: all rows from x where there are matching values in y, return all combination of multiple matches in the case of multiple matches
- `anti_join()`: return all rows from x where there are not matching values in y, never duplicate rows of x
- ...

Setup

For the next few slides...

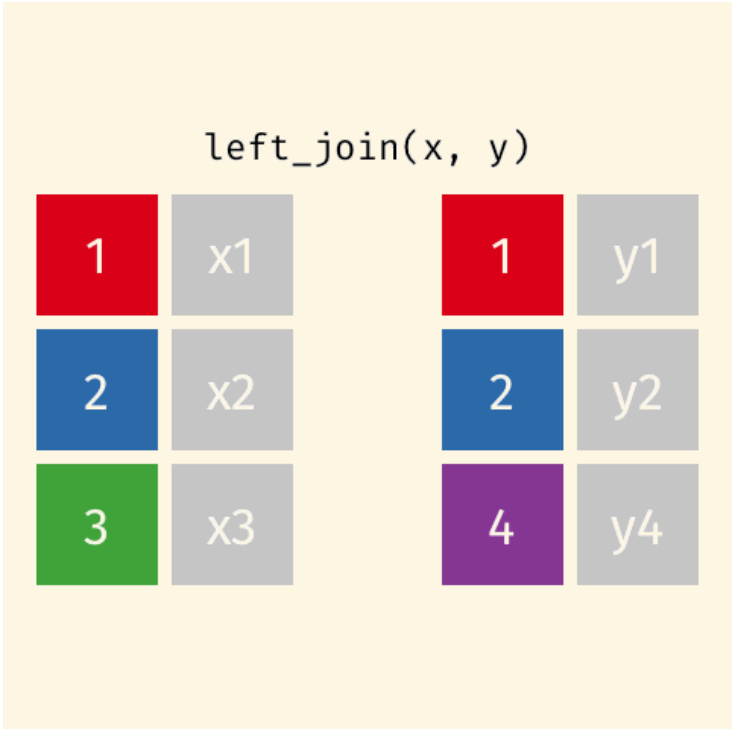
```
x
```

```
## # A tibble: 3 × 2
##      id value_x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
## 3     3 x3
```

```
y
```

```
## # A tibble: 3 × 2
##      id value_y
##   <dbl> <chr>
## 1     1 y1
## 2     2 y2
## 3     4 y4
```

left_join()




```
left_join(x, y)
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 × 3
##      id value_x value_y
##   <dbl> <chr>   <chr>
## 1     1 x1      y1
## 2     2 x2      y2
## 3     3 x3      <NA>
```

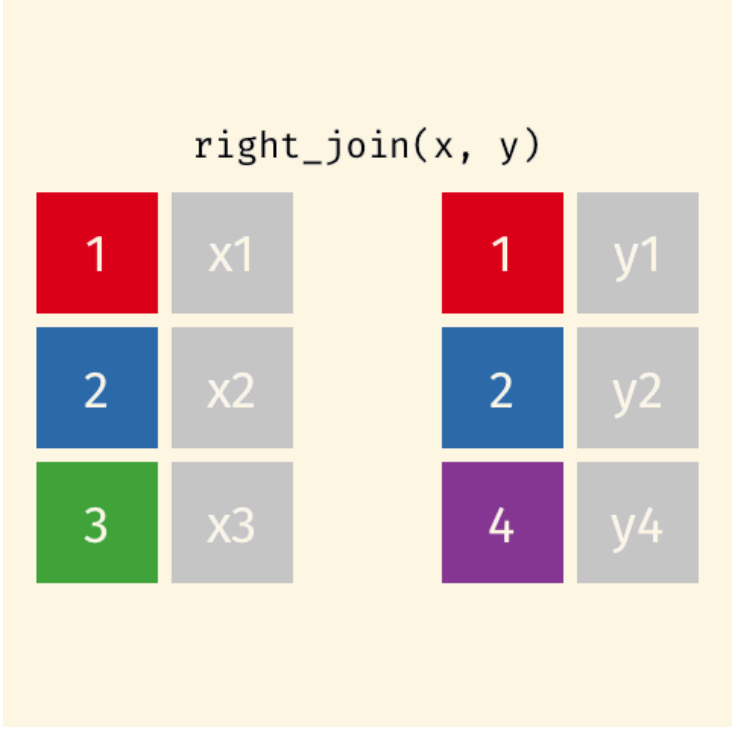
left_join()

```
professions %>%
  left_join(dates)
```

```
## Joining, by = "name"
```

```
## # A tibble: 10 × 4
##    name               profession                        birth_year death_year
##    <chr>              <chr>                                  <dbl>      <dbl>
##  1 Ada Lovelace       Mathematician                             NA         NA
##  2 Marie Curie        Physicist and Chemist                     NA         NA
##  3 Janaki Ammal       Botanist                                1897       1984
##  4 Chien-Shiung Wu    Physicist                               1912       1997
##  5 Katherine Johnson  Mathematician                           1918       2020
##  6 Rosalind Franklin  Chemist                                 1920       1958
##  7 Vera Rubin         Astronomer                              1928       2016
##  8 Gladys West        Mathematician                           1930         NA
##  9 Flossie Wong-Staal Virologist and Molecular Biologist      1947         NA
## 10 Jennifer Doudna    Biochemist                              1964         NA
```

right_join()




```
right_join(x, y)
```

```
## Joining, by = "id"
```

```
## # A tibble: 3 × 3
##      id value_x value_y
##   <dbl> <chr>   <chr>
## 1     1 x1      y1
## 2     2 x2      y2
## 3     4 <NA>    y4
```

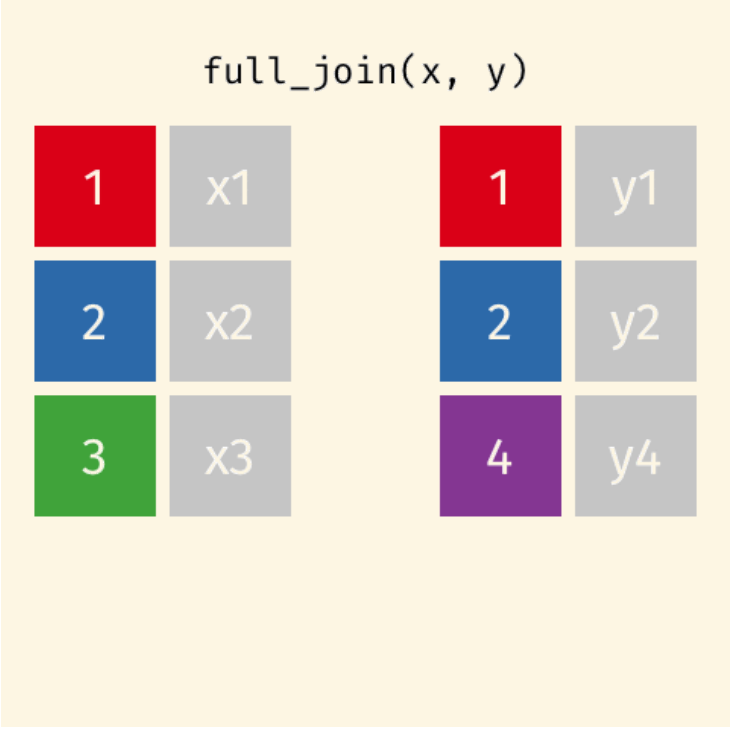
right_join()

```
professions %>%
  right_join(dates)
```

```
## Joining, by = "name"
```

```
## # A tibble: 8 × 4
##   name               profession                       birth_year death_year
##   <chr>              <chr>                                 <dbl>      <dbl>
## 1 Janaki Ammal       Botanist                               1897       1984
## 2 Chien-Shiung Wu    Physicist                              1912       1997
## 3 Katherine Johnson  Mathematician                          1918       2020
## 4 Rosalind Franklin  Chemist                                1920       1958
## 5 Vera Rubin         Astronomer                             1928       2016
## 6 Gladys West        Mathematician                          1930         NA
## 7 Flossie Wong-Staal Virologist and Molecular Biologist     1947         NA
## 8 Jennifer Doudna    Biochemist                             1964         NA
```

full_join()




```
full_join(x, y)
```

```
## Joining, by = "id"
```

```
## # A tibble: 4 × 3
##      id value_x value_y
##   <dbl> <chr>   <chr>
## 1     1 x1      y1
## 2     2 x2      y2
## 3     3 x3      <NA>
## 4     4 <NA>    y4
```

full_join()

```
dates %>%
  full_join(works)
```

```
## Joining, by = "name"
```

```
## # A tibble: 10 × 4
##    name               birth_year death_year known_for
##    <chr>                   <dbl>      <dbl> <chr>
##  1 Janaki Ammal             1897       1984 hybrid species, biodiversity protec…
##  2 Chien-Shiung Wu          1912       1997 confim and refine theory of radioac…
##  3 Katherine Johnson        1918       2020 calculations of orbital mechanics c…
##  4 Rosalind Franklin        1920       1958 <NA>
##  5 Vera Rubin               1928       2016 existence of dark matter
##  6 Gladys West              1930         NA mathematical modeling of the shape …
##  7 Flossie Wong-Staal       1947         NA first scientist to clone HIV and cr…
##  8 Jennifer Doudna          1964         NA one of the primary developers of CR…
##  9 Ada Lovelace               NA         NA first computer algorithm
## 10 Marie Curie                NA         NA theory of radioactivity,  discovery…
```

Tidying data

- we have data organized in an unideal way for our analysis
- we want to reorganize the data to carry on with our analysis

Data: Sales

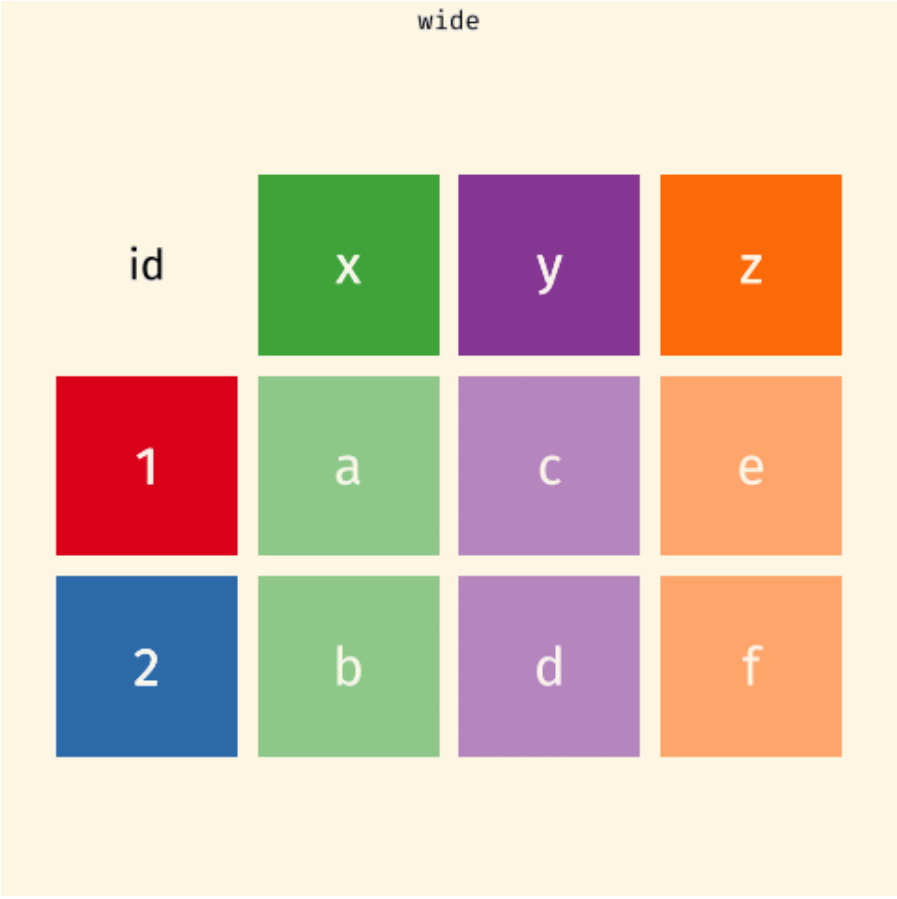
We have...

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr>
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>
```

We want...

```
## # A tibble: 6 × 3
##   customer_id item_no item
##         <dbl> <chr>   <chr>
## 1           1 item_1  bread
## 2           1 item_2  milk
## 3           1 item_3  banana
## 4           2 item_1  milk
## 5           2 item_2  toilet paper
## 6           2 item_3  <NA>
```

Pivoting data



Wider vs. longer

wider

more columns

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr>
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>
```

longer

more rows

```
## # A tibble: 6 × 3
##   customer_id item_no item
##         <dbl> <chr>   <chr>
## 1           1 item_1  bread
## 2           1 item_2  milk
## 3           1 item_3  banana
## 4           2 item_1  milk
## 5           2 item_2  toilet paper
## 6           2 item_3  <NA>
```

pivot_longer()

- data (as usual)

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

pivot_longer()

- `data` (as usual)
- `cols`: columns to pivot into longer format

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

pivot_longer()

- `data` (as usual)
- `cols`: columns to pivot into longer format
- `names_to`: name of the column where column names of pivoted variables go (character string)

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

pivot_longer()

- `data` (as usual)
- `cols`: columns to pivot into longer format
- `names_to`: name of the column where column names of pivoted variables go (character string)
- `values_to`: name of the column where data in pivoted variables go (character string)

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  values_to = "value"
  )
```

Customers → purchases

```
purchases <- customers %>%
  pivot_longer(
    cols = item_1:item_3,  # variables item_1 to item_3
    names_to = "item_no",  # column names -> new column called item_no
    values_to = "item"     # values in columns -> new column called item
    )
```

```
customers
```

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr>
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>
```

```
purchases
```

```
## # A tibble: 6 × 3
##   customer_id item_no item
##         <dbl> <chr>   <chr>
## 1           1 item_1  bread
## 2           1 item_2  milk
## 3           1 item_3  banana
## 4           2 item_1  milk
## 5           2 item_2  toilet paper
## 6           2 item_3  <NA>
```

Why pivot?

Most likely, because the next step of your analysis needs it

```
prices
```

```
## # A tibble: 5 × 2
##   item         price
##   <chr>        <dbl>
## 1 avocado       0.5
## 2 banana        0.15
## 3 bread         1
## 4 milk          0.8
## 5 toilet paper  3
```

```
purchases %>%
  left_join(prices)
```

```
## Joining, by = "item"
```

```
## # A tibble: 6 × 4
##   customer_id item_no item         price
##         <dbl> <chr>   <chr>        <dbl>
## 1           1 item_1  bread         1
## 2           1 item_2  milk          0.8
## 3           1 item_3  banana        0.15
## 4           2 item_1  milk          0.8
## 5           2 item_2  toilet paper  3
## 6           2 item_3  <NA>         NA
```

Purchases → customers

- `data` (as usual)
- `names_from`: which column in the long format contains the what should be column names in the wide format
- `values_from`: which column in the long format contains the what should be values in the new columns in the wide format

```
purchases %>%
  pivot_wider(
    names_from = item_no,
    values_from = item
  )
```

```
## # A tibble: 2 × 4
##   customer_id item_1 item_2       item_3
##         <dbl> <chr>  <chr>        <chr>
## 1           1 bread  milk         banana
## 2           2 milk   toilet paper <NA>
```

Recall the complexity dataset

- We discussed making a scatter plot to compare the complexity ratings
- But we couldn't do that when the data were tidy
- Sketch the dataframe for the data we would need to make this plot

```
complexity_data
```

```
## # A tibble: 1,440 × 4
##    sample subjectid objectid rating
##     <dbl>     <dbl> <chr>     <dbl>
##  1      1         1 54        0.243
##  2      1         5 54        0.433
##  3      1         6 54        0.265
##  4      1        10 54        0.182
##  5      1        18 54        0.136
##  6      1        36 54        0.211
##  7      1        40 54        0.358
##  8      1        42 54        0.735
##  9      1        53 54        0.428
## 10      1        58 54        0.406
## # … with 1,430 more rows
```

Data frame with mean object rating for each object id and sample

```
complexity_long <- complexity_data %>%
  group_by(objectid, sample) %>%
  summarize(mean_rating = mean(rating))

complexity_long
```

```
## # A tibble: 124 × 3
## # Groups:   objectid [62]
##    objectid sample mean_rating
##    <chr>     <dbl>       <dbl>
##  1 1             1       0.433
##  2 1             2       0.478
##  3 10            1       0.357
##  4 10            2       0.395
##  5 11            1       0.532
##  6 11            2       0.564
##  7 12            1       0.472
##  8 12            2       0.373
##  9 13            1       0.324
## 10 13            2       0.264
## # … with 114 more rows
```

Wide dataframe

```
complexity_wide <- pivot_wider(complexity_long,
            names_from = sample,
            values_from = mean_rating)

complexity_wide
```

```
## # A tibble: 62 × 3
## # Groups:   objectid [62]
##    objectid   `1`   `2`
##    <chr>    <dbl> <dbl>
##  1 1        0.433 0.478
##  2 10       0.357 0.395
##  3 11       0.532 0.564
##  4 12       0.472 0.373
##  5 13       0.324 0.264
##  6 14       0.944 0.841
##  7 15       0.248 0.170
##  8 16       0.454 0.481
##  9 17       0.387 0.358
## 10 18       0.654 0.505
## # … with 52 more rows
```

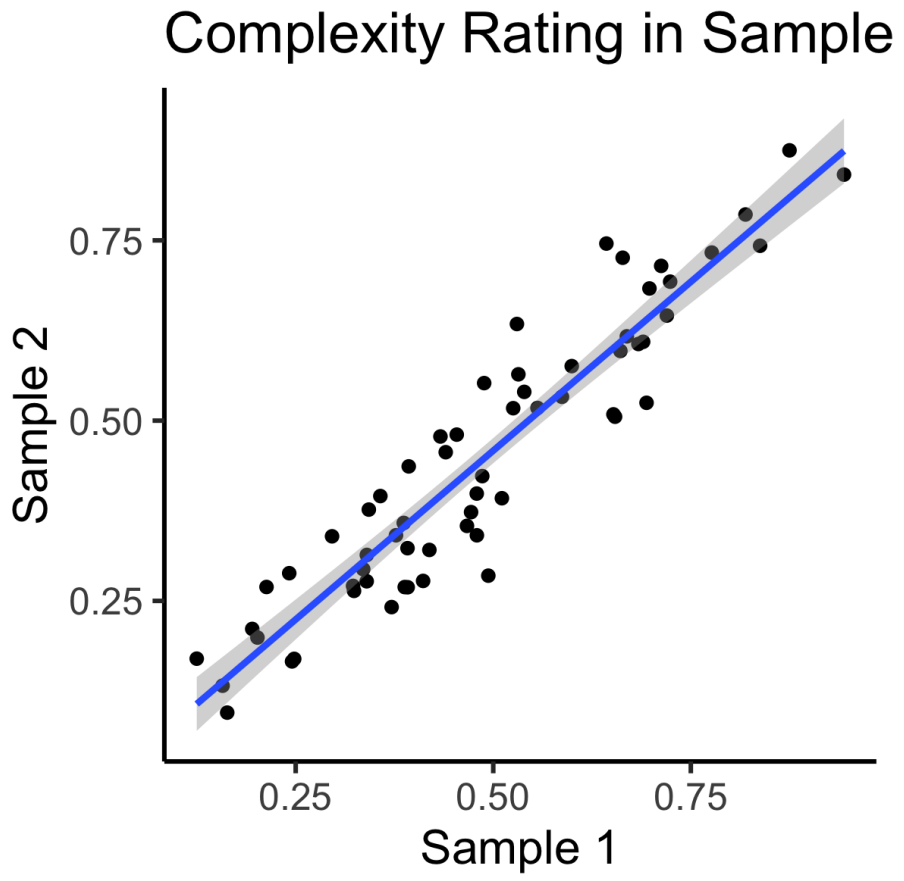
rename()

Syntax: `new_name = old_name`

```
complexity_wide_renamed <-complexity_wide %>%
  rename(sample_1 = `1`,
         sample_2 = `2`)
```

Plotting the wide data

```
ggplot(complexity_wide_renamed,
       aes(x = sample_1,
           y = sample_2)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_classic(base_size = 15) +
  xlab("Sample 1") +
  ylab("Sample 2") +
  ggtitle("Complexity Rating in Sample 1 versus  2")
```



A new tidyverse function: count()

count() is a useful shortcut for group_by() %>% summarize(num = n()).

This code:

```
gapminder %>%
  group_by(country) %>%
  summarize(num_countries = n())
```

```
## # A tibble: 142 × 2
##    country     num_countries
##    <fct>               <int>
##  1 Afghanistan            12
##  2 Albania                12
##  3 Algeria                12
##  4 Angola                 12
##  5 Argentina              12
##  6 Australia              12
##  7 Austria                12
##  8 Bahrain                12
##  9 Bangladesh             12
## 10 Belgium                12
## # … with 132 more rows
```

Does the same as this:

```
gapminder %>%
  count(country)
```

```
## # A tibble: 142 × 2
##    country         n
##    <fct>       <int>
##  1 Afghanistan    12
##  2 Albania        12
##  3 Algeria        12
##  4 Angola         12
##  5 Argentina      12
##  6 Australia      12
##  7 Austria        12
##  8 Bahrain        12
##  9 Bangladesh     12
## 10 Belgium        12
## # … with 132 more rows
```

A new tidyverse function: `glimpse()`

Glimpse is useful for getting the "big picture" view of your data frame.

```
glimpse(gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", …
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, …
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, …
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8…
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12…
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, …
```

`summary()` does something similiar:

```
summary(gapminder)
```

```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
```

A new tidyverse function: `distinct()`

- `distinct()` returns a subset of rows in your data frame (similiar to `filter()`)
- Specifically, `distinct` returns ONE row in your data frame for each value of a variable you pass it.

Abacus dataset (Barner, et al. 2018)

- Does training kids to use an abacus help with their math skills?

Let's read in the dataset

```
abacus_data <- read_csv("data/tidy_majic_data.csv")
```

```
## Rows: 2094 Columns: 8

## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (6): subid, class_num, grade, group, time, task
## dbl (2): year, score

##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Here's what it looks like

```
abacus_data %>%
  head() %>%
  kable(format = "html")
```

subid	class_num	grade	group	year	time	task	score
S1-02-02	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-03	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-08	S1_02	First Grade	Control	2015	Pre	Place Value	0.00
S1-02-08	S1_02	First Grade	Control	2016	Post	Place Value	0.36
S1-02-15	S1_02	First Grade	Control	2016	Post	Place Value	0.64

- The `abacus_data` data frame contains 2,094 rows - one row for each subject-task-time combination.
- For example, the following code returns a data fram with ONE row for each subject id.

```
abacus_data %>%
 distinct(subid)
```

```
## # A tibble: 188 × 1
##    subid
##    <chr>
##  1 S1-02-02
##  2 S1-02-03
##  3 S1-02-08
##  4 S1-02-15
##  5 S1-02-17
##  6 S1-03-04
##  7 S1-03-05
##  8 S1-03-06
##  9 S1-03-09
## 10 S1-03-14
## # … with 178 more rows
```

- The following code returns a data frame with ONE row for each subject id and time.

```
abacus_data %>%
 distinct(subid, time)
```

```
## # A tibble: 349 × 2
##    subid    time
##    <chr>    <chr>
##  1 S1-02-02 Pre
##  2 S1-02-03 Pre
##  3 S1-02-03 Post
##  4 S1-02-08 Pre
##  5 S1-02-08 Post
##  6 S1-02-15 Post
##  7 S1-02-17 Pre
##  8 S1-02-17 Post
##  9 S1-03-04 Post
## 10 S1-03-05 Post
## # … with 339 more rows
```

- You can keep the other variables in the data frame by adding the argument `.keep_all = T` to `distinct()`.

```
abacus_data %>%
distinct(subid, time, .keep_all = T)
```

```
## # A tibble: 349 × 8
##    subid    class_num grade       group          year time  task        score
##    <chr>    <chr>     <chr>       <chr>         <dbl> <chr> <chr>       <dbl>
##  1 S1-02-02 S1_02     First Grade Control        2015 Pre   Place Value  0
##  2 S1-02-03 S1_02     First Grade Control        2015 Pre   Place Value  0
##  3 S1-02-03 S1_02     First Grade Control        2016 Post  Place Value  0.36
##  4 S1-02-08 S1_02     First Grade Control        2015 Pre   Place Value  0
##  5 S1-02-08 S1_02     First Grade Control        2016 Post  Place Value  0.36
##  6 S1-02-15 S1_02     First Grade Control        2016 Post  Place Value  0.64
##  7 S1-02-17 S1_02     First Grade Control        2015 Pre   Place Value  0.09
##  8 S1-02-17 S1_02     First Grade Control        2016 Post  Place Value NA
##  9 S1-03-04 S1_03     First Grade Mental Abacus  2016 Post  Place Value  0.55
## 10 S1-03-05 S1_03     First Grade Mental Abacus  2016 Post  Place Value  0.91
## # … with 339 more rows
```