Advanced Data Wrangling and Analysis

Going Deeper with R

Importing Data

Downloading Data Directly

The `download.file()` function lets us download files directly from websites.

```
download.file(url, destfile = "data-raw/name-of-file.xlsx")
```

Importing Excel Files

Importing Excel Files

```
data_frame <- read_excel(path = "directory_name/file_name.xlsx",
                         sheet = "name of sheet")
```

Importing Excel Files

```
german_speakers <- read_excel(path = "data-raw/german-and-french-speakers.xlsx",
                              sheet = "German speakers") %>%
  clean_names()
```

state	number_of_german_speakers_2017
<chr>	<chr>
Alabama	426
Alaska	331
Arizona	636
Arkansas	-
California	440

1-5 of 51 rows | 1-2 of 4 columns

Previous **1** 2 3 4 5 6 … 11 Next

Other Packages for Importing Data

Got SAS, Stata, or SPSS data? Use the haven package.

Other Packages for Importing Data

The rio package makes it as simple as possible to import and export data.

You give it the location of the data file in the `import()` function and it auto-detects the file type.

My Turn

Throughout, I'll be working with data on third grade math proficiency in Oregon schools.

I'll do the following:

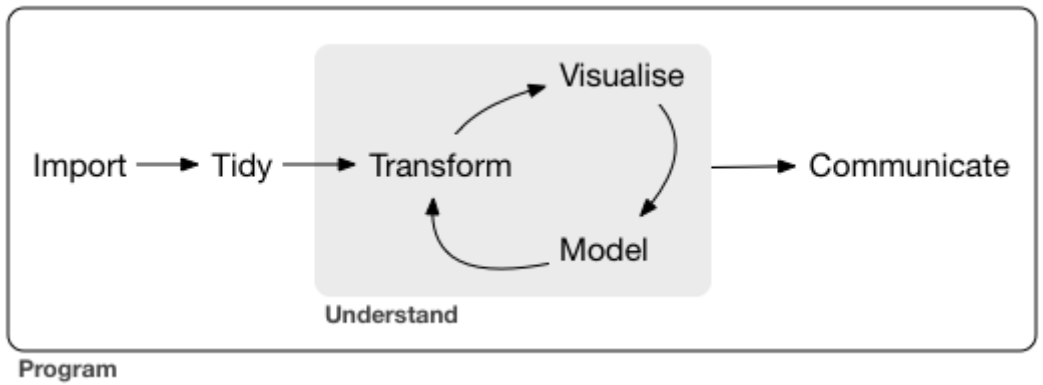
1. Create a new project
2. Create a new R script file where I'll do all of my data cleaning work
3. Download math proficiency scores from 2017-2018 and 2018-2019 and put them in a data-raw folder
4. Import the two spreadsheets into two data frames (`math_scores_17_18` and `math_scores_18_19`)

Your Turn

You'll be working with data on Oregon school enrollment by race/ethnicity.

1. Create a new project. Make sure you put it somehwere you'll be able to find it again later!
2. Download the two files (links below) using the download.file() function into a data-raw folder (which you'll need to create)
3. Create a new R script file where you'll do all of your data cleaning work
4. Import the two spreadsheets into two data frames (`enrollment_17_18` and `enrollment_18_19`)

Tidy Data


Visualise
Import
Tidy
Transform
Communicate
Model
Understand
Program

Untidy Data

state	number_of_german_speakers_2017
<chr>	<dbl>
Alabama	426
Alaska	331
Arizona	636
Arkansas	NA
California	440

1-5 of 51 rows | 1-2 of 4 columns

Previous 1 2 3 4 5 6 ... 11 Next

Untidy Data

```
german_speakers_numeric %>%
  mutate(total = number_of_german_speakers_2017 + number_of_german_speakers_2018 + number_of_
  select(state, total)
```

state	total
<chr>	<dbl>
Alabama	1532
Alaska	663
Arizona	1630
Arkansas	*NA*
California	1612

1-5 of 51 rows

Previous **1** 2 3 4 5 6 … 11 Next

Tidy Data

state	year	number
<chr>	<chr>	<dbl>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows

Previous **1** 2 3 4 5 6 … 31 Next

Tidy Data

```
german_speakers_tidy %>%
  group_by(state) %>%
  summarize(total = sum(number, na.rm = TRUE))
```

	state	total
	<chr>	<dbl>
1	Alabama	1532
2	Alaska	663
3	Arizona	1630
4	Arkansas	1192
5	California	1612

1-5 of 51 rows

Previous **1** 2 3 4 5 6 … 11 Next

The Three Rules of Tidy Data

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

We'll focus on #1 and #3. To read about examples of untidy data, check out the Tidy Data vignette.

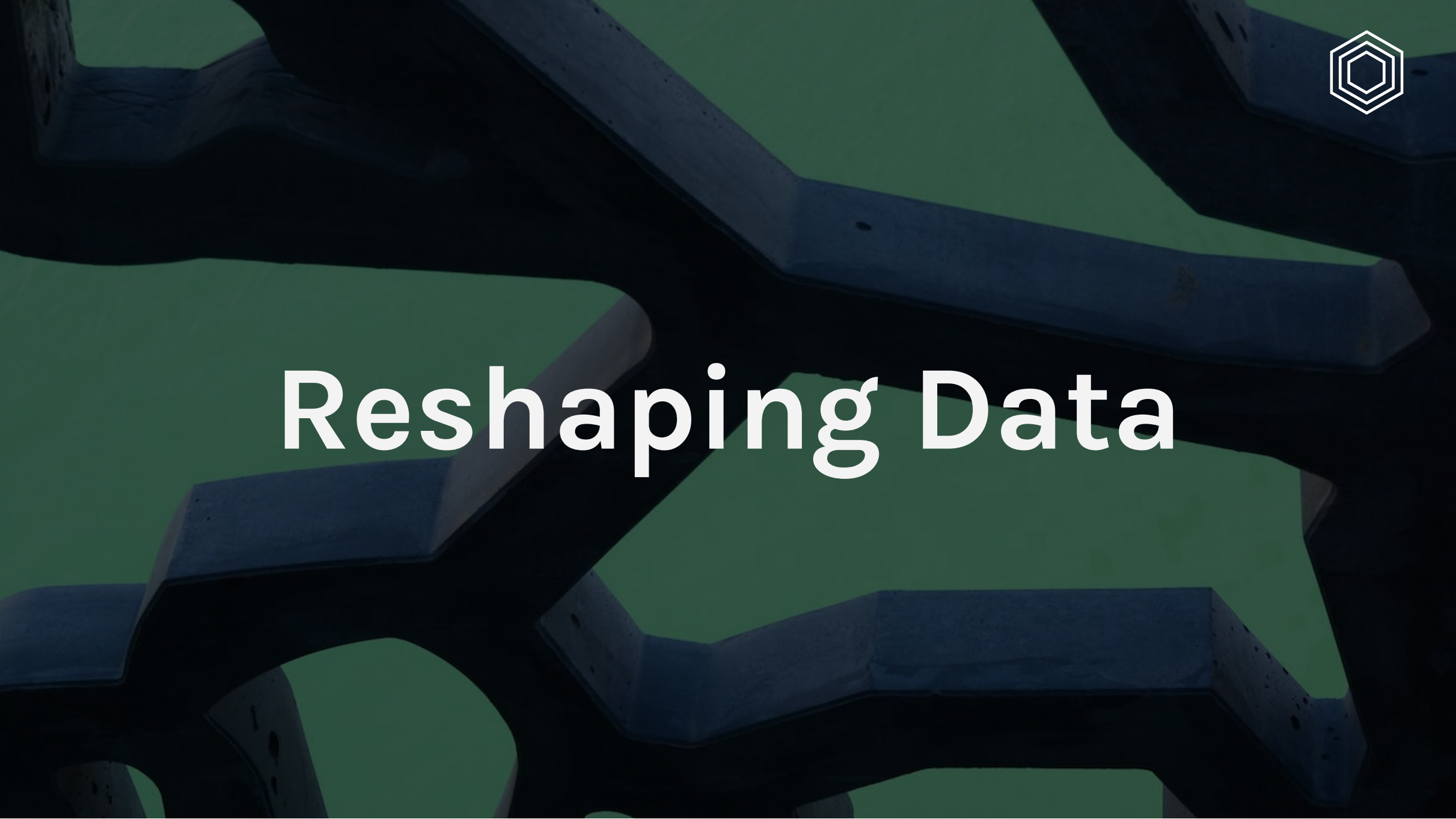
My Turn

Let's take a look at my data and see which principles of tidy data it violates

Your Turn

1. Read the Tidy Data vignette

2. Take a look at your data and see which principles of tidy data it violates



Reshaping Data

Reshaping Data: `pivot_longer()`

The `pivot_longer()` function helps us in situations where **column headers are values, not variable names**.

```
data_frame %>%
  pivot_longer(cols = columns_to_use)
```

```
data_frame %>%
  pivot_longer(cols = columns_to_use,
               names_to = "name_of_identifer_variable")
```

```
data_frame %>%
  pivot_longer(cols = columns_to_use,
               names_to = "name_of_identifer_variable",
               values_to = "name_of_value_variable")
```

Reshaping Data: `pivot_longer()`

state	number_of_german_speakers_2017
<chr>	<chr>
Alabama	426
Alaska	331
Arizona	636
Arkansas	-
California	440

1-5 of 51 rows | 1-2 of 4 columns

Previous **1** 2 3 4 5 6 … 11 Next

Reshaping Data: `pivot_longer()`

```
german_speakers %>%
  pivot_longer(cols = -state)
```

state <chr>	name <chr>	value <chr>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows

Previous **1** 2 3 4 5 6 … 31 Next

Reshaping Data: `pivot_longer()`

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number")
```

state <chr>	year <chr>	number <chr>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows Previous **1** 2 3 4 5 6 … 31 Next

Reshaping Data: `pivot_wider()`

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  pivot_wider(id_cols = year,
              names_from = "state",
              values_from = "number")
```

year <chr>	Alabama <chr>	Alaska <chr>	Arizona <chr>	Arkansas <chr>
number_of_german_speakers_2017	426	331	636	-
number_of_german_speakers_2018	395	201	858	635
number_of_german_speakers_2019	711	131	136	557

3 rows | 1-5 of 52 columns

My Turn

I'll do the following to create a new data frame called `third_grade_math_proficiency_18_19`:

1. Use `filter()` to only keep rows where the `student_group` variable is "Total Population (All Students)"
2. Use `filter()` to only keep third grade students
3. Use `select()` to only keep variables related to the **number** (not percentage) of students who are proficient in math
4. Use `pivot_longer()` to make my data frame tidy

Your Turn

Do the following to create a new data frame called `enrollment_by_race_ethnicity_18_19`:

1. Start with the `enrollment_18_19` data frame
2. select() the `district_id` variable as well as those about number of students by race/ethnicity and get rid of all others (hint: use the `contains()` helper function within `select()`)
3. Use `pivot_longer()` to convert all of the race/ethnicity variables into one variable
4. Within `pivot_longer()`, use the names_to argument to call that variable `race_ethnicity`
5. Within `pivot_longer()`, use the values_to argument to call that variable

Dealing with Missing Data

na_if()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number")
```

state	year	number
<chr>	<chr>	<chr>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows

Previous **1** 2 3 4 5 6 … 31 Next

na_if()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-"))
```

na_if()

state <chr>	year <chr>	number <chr>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows

Previous 1 2 3 4 5 6 … 31 Next

replace_na()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0))
```

replace_na()

state <chr>	year <chr>	number <chr>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows Previous **1** 2 3 4 5 6 … 31 Next

My Turn

I'll convert all of the missing values in the `number_proficient` variable to NA using `na_if()`

I don't have any values where using `replace_na()` makes sense, but you'll use it shortly!

Your Turn

1. Convert all of the missing values in the `number_of_students` variable to NA using `na_if()`

2. Convert all of the NA values you just made to 0 using `replace_na()`.

Changing Variable Types

Changing Variable Types

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0))
```

state <chr>	year <chr>	number <chr>
Alabama	number_of_german_speakers_2017	426
Alabama	number_of_german_speakers_2018	395
Alabama	number_of_german_speakers_2019	711
Alaska	number_of_german_speakers_2017	331
Alaska	number_of_german_speakers_2018	201

1-5 of 153 rows

Previous 1 2 3 4 5 6 … 31 Next

Changing Variable Types

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  summarize(total = sum(number))
```

```
## Error: `summarise()` argument `total` errored.
## ℹ `total` is `sum(number)`.
## x invalid 'type' (character) of argument
```

Changing Variable Types

To change variable types, you use the `mutate()` function combined with `as.numeric()`, `as.character()`, etc.

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  summarize(total = sum(number))
```

total
<dbl>
80314

1 row

My Turn

1. Convert the `number_proficient` variable to numeric by using `as.numeric()`
2. Count the number of students at each proficiency level

Your Turn

1. Convert the `number_of_students` variable to numeric by using `as.numeric()`
2. Make sure you can use your `number_of_students` variable to count the total number of students in Oregon

Advanced Variable Creation

recode()

```
data_frame %>%
  mutate(variable = recode(variable, "old_value" = "new_value"))
```

recode()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = recode(year,
                       "number_of_german_speakers_2017" = "2017",
                       "number_of_german_speakers_2018" = "2018",
                       "number_of_german_speakers_2019" = "2019"))
```

recode()

state <chr>	year <chr>	number <dbl>
Alabama	2017	426
Alabama	2018	395
Alabama	2019	711
Alaska	2017	331
Alaska	2018	201

1-5 of 153 rows

Previous 1 2 3 4 5 6 ... 31 Next

if_else()

```
data_frame %>%
  mutate(variable = if_else(variable == "some_value",
                            "if_true_value",
                            "else_value"))
```

if_else()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = if_else(year == "number_of_german_speakers_2017", "2017", year)) %>%
  mutate(year = if_else(year == "number_of_german_speakers_2018", "2018", year)) %>%
  mutate(year = if_else(year == "number_of_german_speakers_2019", "2019", year))
```

if_else()

state	year	number
<chr>	<chr>	<dbl>
Alabama	2017	426
Alabama	2018	395
Alabama	2019	711
Alaska	2017	331
Alaska	2018	201

1-5 of 153 rows

Previous 1 2 3 4 5 6 ... 31 Next

str_remove()

```
data_frame %>%
  mutate(variable = str_remove(variable, "text to remove"))
```

str_remove()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = str_remove(year, "number_of_german_speakers_"))
```

str_remove()

state	year	number
<chr>	<chr>	<dbl>
Alabama	2017	426
Alabama	2018	395
Alabama	2019	711
Alaska	2017	331
Alaska	2018	201

1-5 of 153 rows

Previous **1** 2 3 4 5 6 … 31 Next

Image courtesy of [Allison Horst]

parse_number()

```
data_frame %>%
  mutate(variable = parse_number(variable))
```

parse_number()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = parse_number(year))
```

parse_number()

state <chr>	year <dbl>	number <dbl>
Alabama	2017	426
Alabama	2018	395
Alabama	2019	711
Alaska	2017	331
Alaska	2018	201

1-5 of 153 rows Previous **1** 2 3 4 5 6 … 31 Next

case_when()

```
data_frame %>%
  mutate(variable = case_when(
    variable == "some_value" ~ "new_value",
    variable == "some_other_value" ~ "new_value_2",
    variable == "some_third_value" ~ "new_value_3",
    TRUE ~ "value_for_all_observations_that_dont_match_any_above_criteria"
  ))
```

case_when()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = case_when(
    year == "number_of_german_speakers_2017" ~ "2017",
    year == "number_of_german_speakers_2018" ~ "2018",
    year == "number_of_german_speakers_2019" ~ "2019"
  ))
```

case_when()

state	year	number
<chr>	<chr>	<dbl>
Alabama	2017	426
Alabama	2018	395
Alabama	2019	711
Alaska	2017	331
Alaska	2018	201

1-5 of 153 rows

Previous 1 2 3 4 5 6 … 31 Next

More complicated `case_when()`

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = parse_number(year)) %>%
  mutate(number_categorical = case_when(
    number < 500 ~ "Less than 500",
    between(number, 500, 1000) ~ "Between 500 and 1000",
    number > 1000 ~ "Greater than 1000"
  ))
```

More complicated `case_when()`

state	year	number	number_categorical
<chr>	<dbl>	<dbl>	<chr>
Alabama	2017	426	Less than 500
Alabama	2018	395	Less than 500
Alabama	2019	711	Between 500 and 1000
Alaska	2017	331	Less than 500
Alaska	2018	201	Less than 500

1-5 of 153 rows

Previous **1** 2 3 4 5 6 … 31 Next

My Turn

I'll convert all instances of the `proficiency_level` variable to more meaningful observations using:

1. `recode()`
2. `if_else()`
3. `str_remove()`
4. `parse_number()`
5. `case_when()`

I'll then use `case_when()` to convert the proficiency level into a dichotomous (i.e. Proficient/Not Proficient).

Your Turn

1. Remove the "x_2018_2019" portion of the `race_ethnicity` variable using `str_remove()`
2. Convert all instances of the `race_ethnicity` variable to more meaningful observations (e.g. turn "american_indian_alaska_native" into "American Indian/Alaskan Native") using any of the following:
    - `recode()`
    - `if_else()`
    - `case_when()`

Advanced Summarizing

group_by() + mutate()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = parse_number(year)) %>%
  group_by(year) %>%
  mutate(pct = number / sum(number)) %>%
  arrange(year, state)
```

group_by() + mutate()

state <chr>	year <dbl>	number <dbl>	pct <dbl>
Alabama	2017	426	0.016217451
Alaska	2017	331	0.012600883
Arizona	2017	636	0.024211969
Arkansas	2017	0	0.000000000
California	2017	440	0.016750419

1-5 of 153 rows

Previous **1** 2 3 4 5 6 … 31 Next

ungroup()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = parse_number(year)) %>%
  group_by(year) %>%
  mutate(pct = number / sum(number)) %>%
  slice_max(pct, 1) %>%
  arrange(year, state)
```

ungroup()

state	year	number	pct
<chr>	<dbl>	<dbl>	<dbl>
Ohio	2017	960	0.03654637
Delaware	2018	998	0.03721381
West Virginia	2019	974	0.03577200

3 rows

ungroup()

```
german_speakers %>%
  pivot_longer(cols = -state,
               names_to = "year",
               values_to = "number") %>%
  mutate(number = na_if(number, "-")) %>%
  mutate(number = replace_na(number, 0)) %>%
  mutate(number = as.numeric(number)) %>%
  mutate(year = parse_number(year)) %>%
  group_by(year) %>%
  mutate(pct = number / sum(number)) %>%
  ungroup() %>%
  slice_max(pct, 1)
```

ungroup()

state	year	number	pct
<chr>	<dbl>	<dbl>	<dbl>
Delaware	2018	998	0.03721381

1 row

My Turn

I'll calculate the percent of students at each school who are proficient in math. To do this, I'll need to use both:

- `group_by()` and `summarize()`
- `group_by()` and `mutate()`

And I can't forget to `ungroup()`!

Your Turn

Create a new variable called `pct` that shows each race/ethnicity as a percentage of all students in each district

You'll need to use `group_by()` and `mutate()`

Don't forget to `ungroup()` at the end!

Binding Data Frames

bind_rows()

`german_speakers_2018`

state	number
<chr>	<dbl>
Alabama	395
Alaska	201
Arizona	858
Arkansas	635
California	318

1-5 of 5... Previous **1** 2 3 4 5 … 11 Next

`german_speakers_2019`

state	number
<chr>	<dbl>
Alabama	711
Alaska	131
Arizona	136
Arkansas	557
California	854

1-5 of 5... Previous **1** 2 3 4 5 … 11 Next

bind_rows()

```
german_speakers_2018_2019 <- bind_rows(german_speakers_2018,
                                       german_speakers_2019)
```

```
german_speakers_2018_2019
```

	state <chr>	number <dbl>
1	Alabama	395
2	Alaska	201
3	Arizona	858
4	Arkansas	635
5	California	318

1-5 of 102 rows

Previous **1** 2 3 4 5 6 … 21 Next

bind_rows()

```
german_speakers_2018 <- german_speakers_2018 %>%
  mutate(year = 2018)

german_speakers_2019 <- german_speakers_2019 %>%
  mutate(year = 2019)

german_speakers_2018_2019 <- bind_rows(german_speakers_2018,
                                       german_speakers_2019)
```

bind_rows()

```
german_speakers_2018_2019
```

	state <chr>	number <dbl>	year <dbl>
1	Alabama	395	2018
2	Alaska	201	2018
3	Arizona	858	2018
4	Arkansas	635	2018
5	California	318	2018

1-5 of 102 rows Previous **1** 2 3 4 5 6 … 21 Next

bind_cols()

```
german_speakers_2018 <- german_speakers_2018 %>%
  mutate(year = 2018)

german_speakers_2019 <- german_speakers_2019 %>%
  mutate(year = 2019)

german_speakers_2018_2019 <- bind_cols(german_speakers_2018,
                                       german_speakers_2019)
```

bind_cols()

```
german_speakers_2018_2019
```

	state...1	number...2	year...3	state...4
	<chr>	<dbl>	<dbl>	<chr>
1	Alabama	395	2018	Alabama
2	Alaska	201	2018	Alaska
3	Arizona	858	2018	Arizona
4	Arkansas	635	2018	Arkansas
5	California	318	2018	California

1-5 of 51 rows | 1-5 of 7 columns

Previous **1** 2 3 4 5 6 ... 11 Next

My Turn

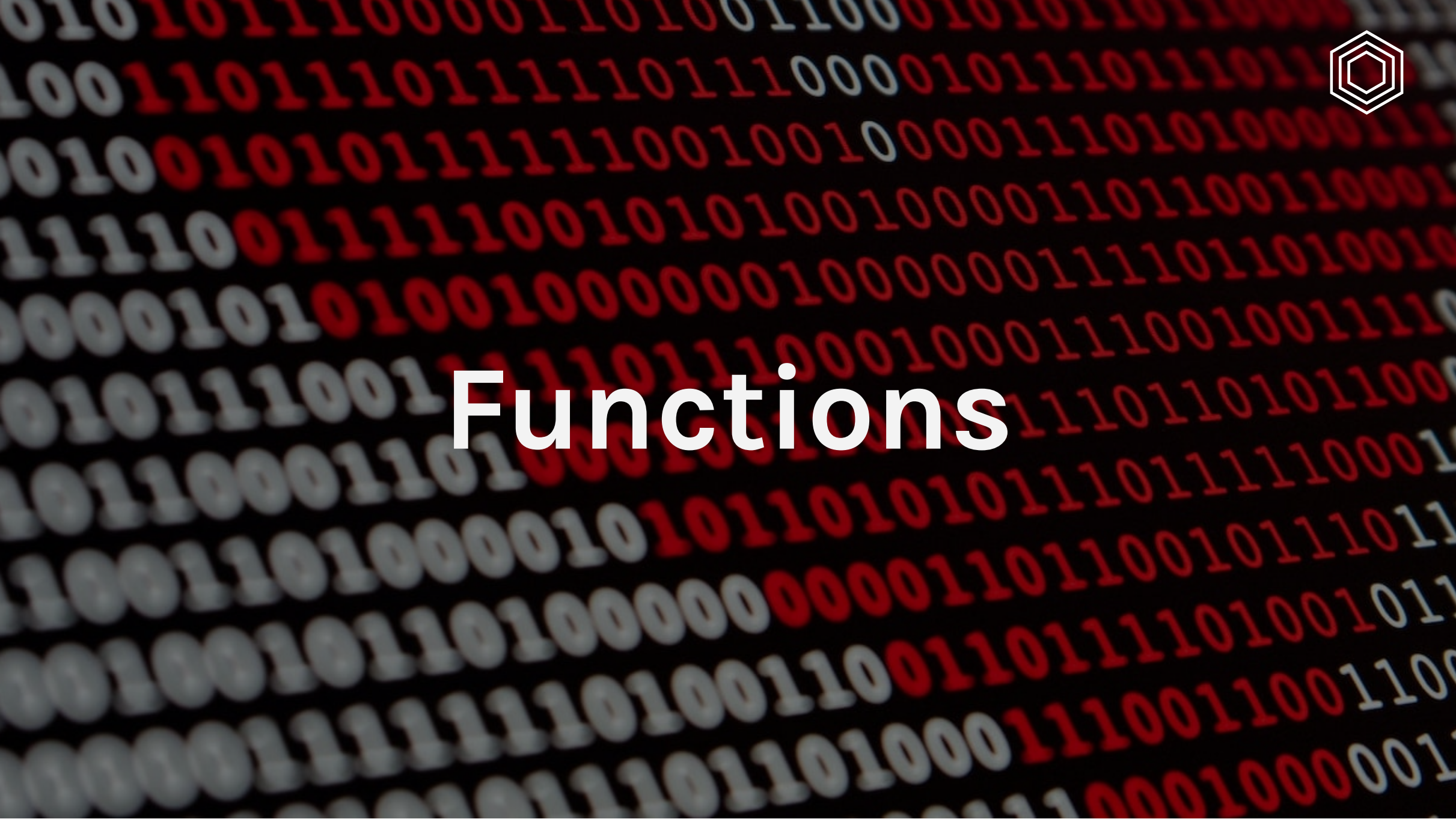
I'll import 2017-2018 math proficiency data and then use bind_rows() to make a `third_grade_math_proficiency` data frame.

Your Turn

1. Import 2017-2018 enrollment data into a data frame called `enrollment_by_race_ethnicity_17_18` and clean it using the code you used for the 2018-2019 data

2. Use bind_rows() to make a `enrollment_by_race_ethnicity` data frame.

Hint: You'll need to change some of your code from importing the 2018-2019 data to make the `race_ethnicity` variable get recoded correctly!



Functions

When to Use Functions?

You should consider writing a function whenever you've copied and pasted a block of code more than twice (i.e. you now have three copies of the same code).

What is a Function?

Source: [R for the Rest of Us Blog]

Why Use Functions?

1. You can give a function an evocative name that makes your code easier to understand.

2. As requirements change, you only need to update code in one place, instead of many.

3. You eliminate the chance of making incidental mistakes when you copy and paste (i.e. updating a variable name in one place, but not in another).

Source: R for Data Science, Chapter 19

My Turn

I'll create a function to clean each year of math proficiency data, then use `bind_rows()` to bind them together

Your Turn

Create a function to clean each year of enrollment data, then use `bind_rows()` to bind them together

Arguments you'll need to use:

- Data year
- Text to remove in the `str_remove()` line

Data Merging

Data Merging

All of the animations and explanations used here come from the tidyexplain project by Garrick Aden-Buie.

Data Merging

```
german_speakers_2019
```

state <chr>	number <dbl>
Alabama	711
Alaska	131
Arizona	136
Arkansas	557
California	854

1-5 of 51 rows

Previous **1** 2 3 4 5 6 … 11 Next

Data Merging

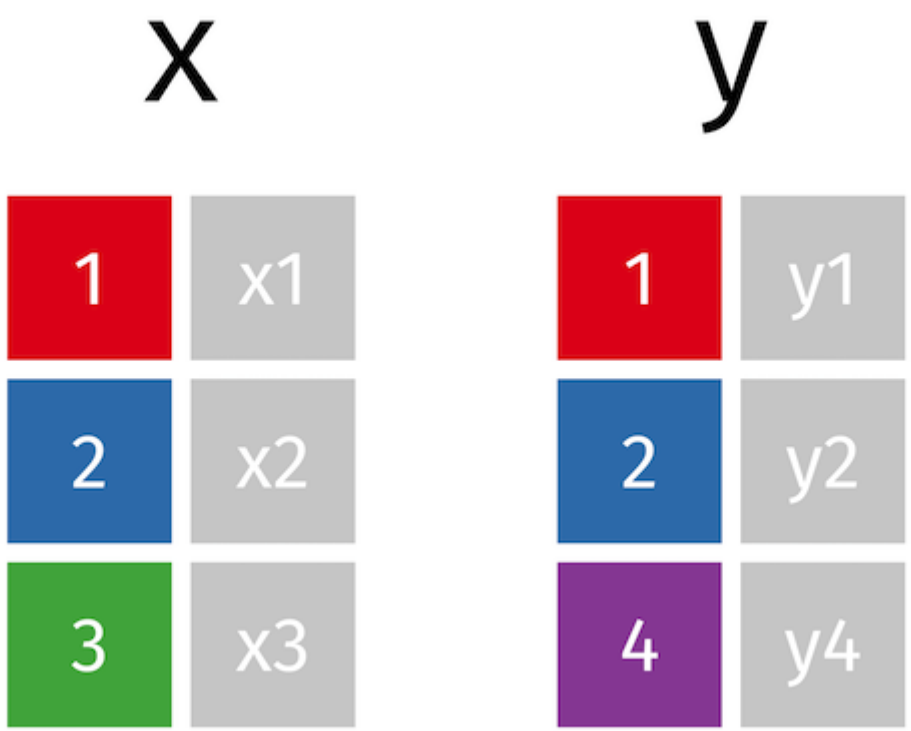
```
french_speakers_2019 <- read_excel(path = "data-raw/german-and-french-speakers.xlsx",
                                   sheet = "French speakers 2019")
```

state	french_speakers
<chr>	<dbl>
Alabama	1678
Alaska	1441
Arizona	1002
Arkansas	1558
California	1935

1-5 of 9 rows

Previous **1** 2 Next

Joins



Joins

```
type_of_join(x, y,
             by = "id_variable"))
```

What if we don't have a variable with the same name in both data frames?

```
type_of_join(x, y,
             by = c("id_variable_x" = "id_variable_y"))
```

What if we need to join on multiple variables with different names in both data frames?

```
type_of_join(x, y,
             by = c("id_variable_a" = "id_variable_b",
                    "id_variable_c" = "id_variable_d"))
```

Which join should I use?

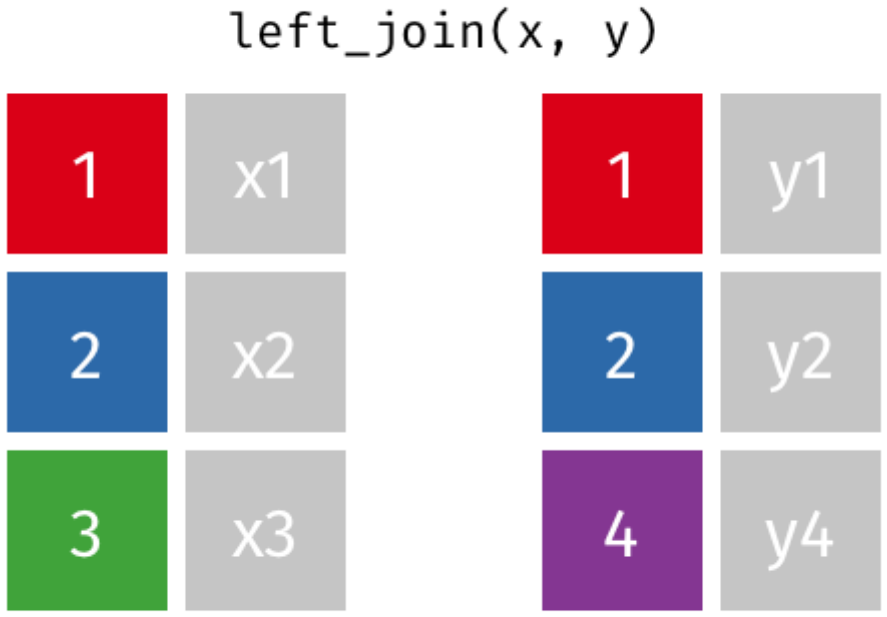
Mutating joins

> A mutating join allows you to combine variables from two tables. It first matches observations by their keys, then copies across variables from one table to the other. - R for Data Science

`left_join()`

All rows from x, and all columns from x and y. Rows in x with no match in y will have NA values in the new columns.

left_join()



left_join()

```
left_join(german_speakers_2019,
          french_speakers_2019,
          by = "state")
```

	state <chr>	number <dbl>	french_speakers <dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

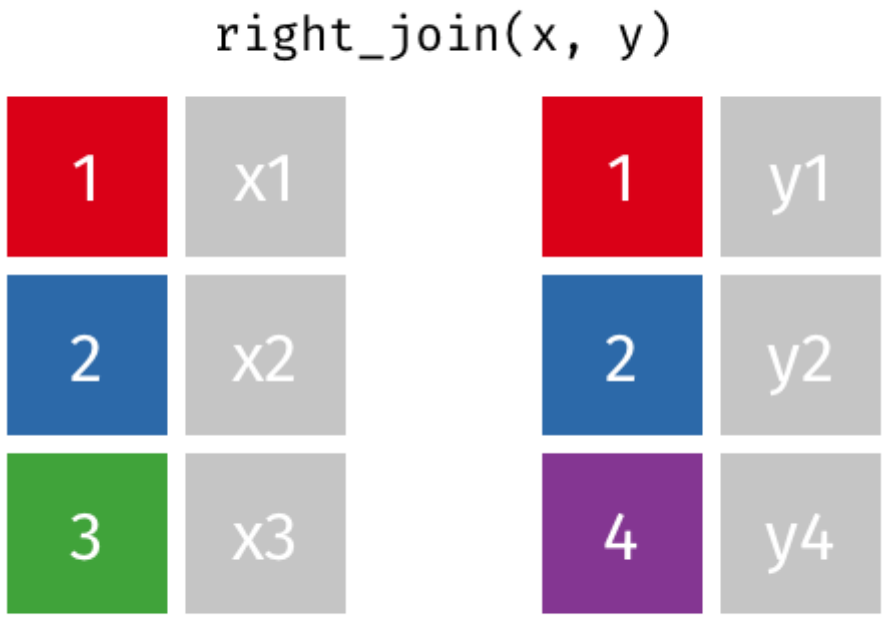
1-5 of 51 rows

Previous **1** 2 3 4 5 6 … 11 Next

`right_join()`

All rows from y, and all columns from x and y. Rows in y with no match in x will have NA values in the new columns.

right_join()



right_join()

```
right_join(german_speakers_2019,
           french_speakers_2019,
           by = "state")
```

	state	number	french_speakers
	<chr>	<dbl>	<dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

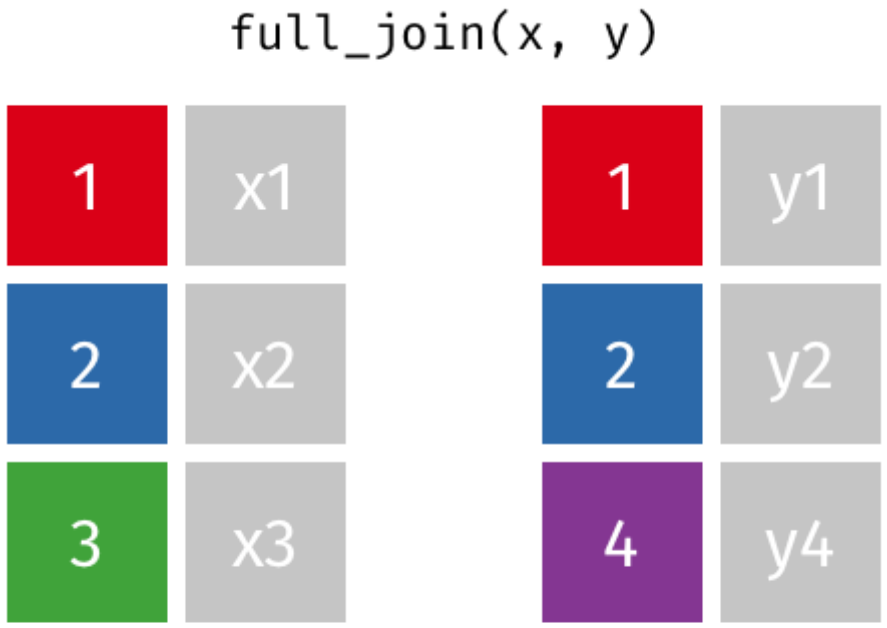
1-5 of 9 rows

Previous 1 2 Next

`full_join()`

All rows and all columns from both x and y. Where there are not matching values, returns NA for the one missing.

full_join()



full_join()

```
full_join(german_speakers_2019,
          french_speakers_2019,
          by = "state")
```

	state <chr>	number <dbl>	french_speakers <dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

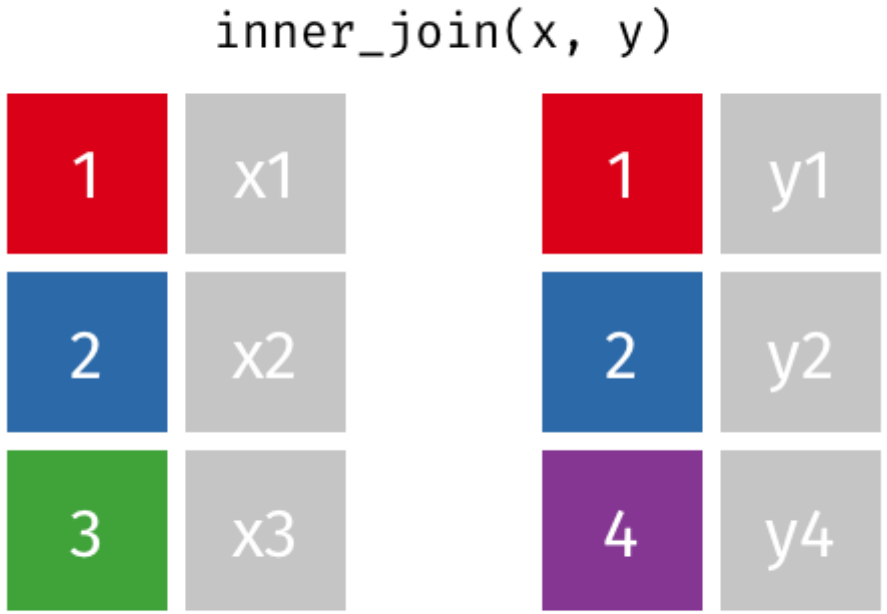
1-5 of 51 rows

Previous **1** 2 3 4 5 6 … 11 Next

`inner_join()`

All rows and all columns from both x and y. Where there are not matching values, returns NA for the one missing.

inner_join()



inner_join()

```
inner_join(german_speakers_2019,
           french_speakers_2019,
           by = "state")
```

	state	number	french_speakers
	<chr>	<dbl>	<dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

1-5 of 9 rows

Previous **1** 2 Next

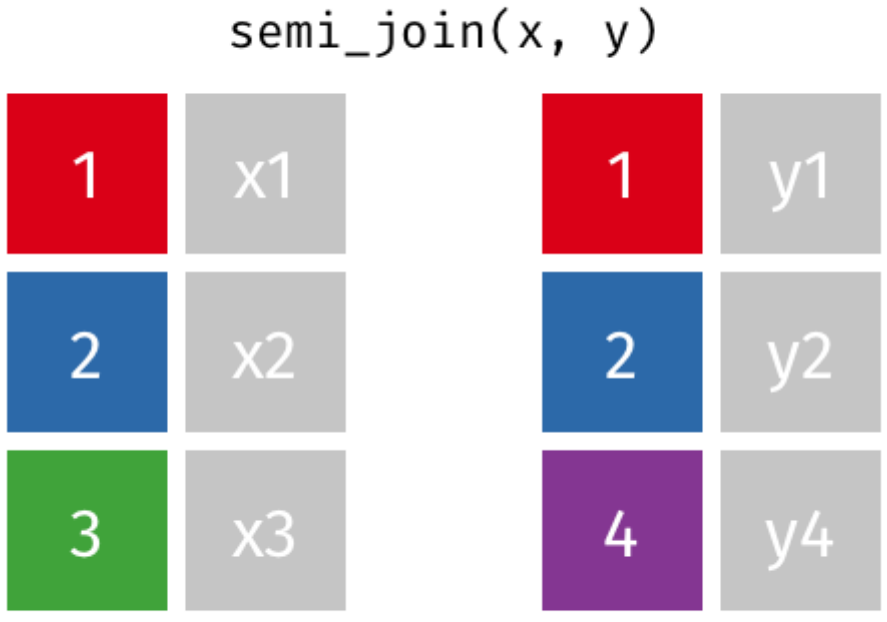
Filtering joins

> Filtering joins match observations in the same way as mutating joins, but affect the observations, not the variables - R for Data Science

`semi_join()`

All rows from x where there are matching values in y, keeping just columns from x.

semi_join()



semi_join()

```
semi_join(german_speakers_2019,
          french_speakers_2019,
          by = "state")
```

state <chr>	number <dbl>
Alabama	711
Alaska	131
Arizona	136
Arkansas	557
California	854

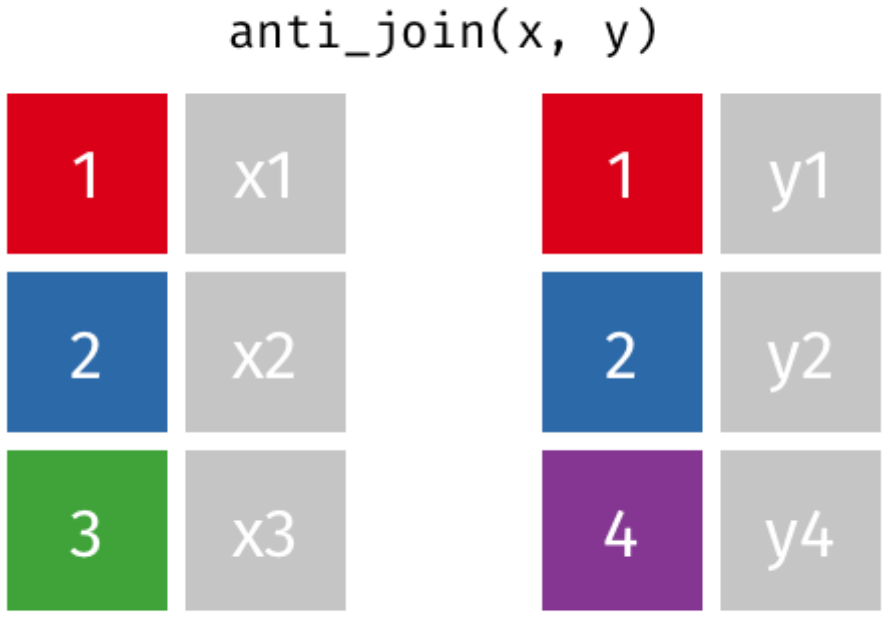
1-5 of 9 rows

Previous 1 2 Next

`anti_join()`

All rows from x where there are not matching values in y, keeping just columns from x.

anti_join()



anti_join()

```
anti_join(german_speakers_2019,
          french_speakers_2019,
          by = "state")
```

state <chr>	number <dbl>
Florida	958
Georgia	821
Hawaii	931
Idaho	0
Illinois	827

1-5 of 42 rows Previous **1** 2 3 4 5 6 … 9 Next

My Turn

1. Download the oregon-districts-and-schools.xlsx file into the data-raw folder.
2. Import a new data frame called `oregon_districts_and_schools` from oregon-districts-and-schools.xlsx
3. Merge the `oregon_districts_and_schools` data frame into the `third_grade_math_proficiency` data frame so I can see the names of the schools as well as associated districts

Your Turn

1. Download the oregon-districts.xlsx file into the data-raw folder.
2. Import a new data frame called `oregon_districts` from oregon-districts.xlsx
3. Merge the `oregon_districts` data frame into the `enrollment_by_race_ethnicity` data frame so you can see the names of the districts


Renaming Variables

Renaming Variables

```
french_and_german_speakers_2019 <- left_join(german_speakers_2019,
                                             french_speakers_2019,
                                             by = "state")
```

	state	number	french_speakers
	<chr>	<dbl>	<dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

1-5 of 51 rows

Previous **1** 2 3 4 5 6 … 11 Next

rename()

```
data_frame %>%
  rename(new_variable_name = old_variable_name)
```

rename()

```
french_and_german_speakers_2019 <- left_join(german_speakers_2019,
                                             french_speakers_2019,
                                             by = "state") %>%
  rename(german_speakers = number)
```

```
french_and_german_speakers_2019
```

	state	german_speakers	french_speakers
	<chr>	<dbl>	<dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

1-5 of 51 rows Previous **1** 2 3 4 5 6 … 11 Next

set_names()

```
french_and_german_speakers_2019 <- left_join(german_speakers_2019,
                                             french_speakers_2019,
                                             by = "state") %>%
  set_names("state", "german_speakers", "french_speakers")
```

```
french_and_german_speakers_2019
```

	state	german_speakers	french_speakers
	<chr>	<dbl>	<dbl>
1	Alabama	711	1678
2	Alaska	131	1441
3	Arizona	136	1002
4	Arkansas	557	1558
5	California	854	1935

1-5 of 51 rows Previous **1** 2 3 4 5 6 … 11 Next

My Turn

I'll use `rename()` and/or `set_names()` to give the `third_grade_math_proficiency` data frame more meaningful names.

Your Turn

Use either `rename()` or `set_names()` to give the `enrollment_by_race_ethnicity` data frame more meaningful variable names. In particular, change the `pct` variable to something more descriptive.

Quick Interlude to Reorganize Your Code

Your Turn

Reorganize your code so that you only create the `enrollment_by_race_ethnicity` data frame in one place.

Exporting Data

write_csv()

```
write_csv(french_and_german_speakers_2019,
          path = "data/french_and_german_speakers_2019.csv")
```

write_rds()

```
write_rds(french_and_german_speakers_2019,
          path = "data/french_and_german_speakers_2019.rds")
```

My Turn

1. Export my `third_grade_math_proficiency` data frame as a CSV
2. Export my `third_grade_math_proficiency` data frame as an RDS file

Your Turn

Export the `enrollment_by_race_ethnicity` data frame as an RDS file in the data directory (you'll need to make this directory)