In [1]:
```
# Initialize Otter
import otter
grader = otter.Notebook("hw2-seda.ipynb")
```

PSTAT 100 Homework 2

In [2]:
```
import numpy as np
import pandas as pd
import altair as alt
```

Background

Gender achievement gaps in education have been well-documented over the years -- studies consistently find boys outperforming girls on math tests and girls outperforming boys on reading and language tests. A particularly controversial article was published in Science in 1980 arguing that this pattern was due to an 'innate' difference in ability (focusing, of course, on mathematics rather than on reading and language). Such views persisted in part because studying systematic patterns in achievement nationwide was a challenge due to differential testing standards across school districts and the general lack of availability of large-scale data.

It is only recently that data-driven research has begun to reveal socioeconomic drivers of achievement gaps. The Standford Educational Data Archive (SEDA), a publicly available database on academic achievement and educational opportunity in U.S. schools, has supported this effort. The database is part of a broader initiave aiming to improve educational opportunity by enabling researchers and policymakers to identify systemic drivers of disparity.

> SEDA includes a range of detailed data on educational conditions, contexts, and outcomes in school districts and counties across the United States. It includes measures of academic achievement and achievement gaps for school districts and counties, as well as district-level measures of racial and socioeconomic composition, racial and socioeconomic segregation patterns, and other features of the schooling system.

The database standardizes average test scores for schools 10,000 U.S. school districts relative to national standards to allow comparability between school districts and across grade levels and years. The test score data come from the U.S. Department of Education. In addition, multiple data sources (American Community Survey and Common Core of Data) are integrated to provide district-level socioeconomic and demographic information.

A study of the SEDA data published in 2018 identified the following persistent patterns across grade levels 3 - 8 and school ears from 2008 through 2015:

- a consistent reading and language achievement gap favoring girls;
- *no* national math achievement gap on average; and
- local math achievement gaps that depend on the socioeconomic conditions of school districts. You can read about the main findings of the study in this brief NY Times article.

Below, we'll work with selected portions of the database. The full datasets can be downloaded here.

Assignment objectives

In this assignment, you'll explore achievement gaps in California school districts in 2018, reproducing the findings described in the article above on a more local scale and with the most recent SEDA data. This will afford you an opportunity to practice the first several stages of the data science lifecycle: collect, acquaint, tidy, and explore.

Collect/acquiant

- review data documentation
- identify population, sampling frame, sample
- assess scope of inference

Tidy

- data import
- slicing and filtering
- merging multiple data frames
- pivoting tables
- renaming and reordering variables

Explore

- scatterplots
- basic plotting aesthetics
- faceted plots
- visualizing trends
- aggregation and tabulation

Communicate

- narrative summary of exploratory analysis

Collaboration

You are encouraged to collaborate with other students on the labs, but are expected to write up your own work for submission. Copying and pasting others' solutions is considered plaigarism and may result in penalties, depending on severity and extent.

If you choose to work with others, please list their names here.

Your name: Amy Nguyen

Collaborators:

---

0. Getting acquainted with the SEDA data

The cell below imports the district-level SEDA data from California in 2018. The test score data is stored in a separate file (`ca-main.csv`) from the socioeconomic and demographic covariate data (`ca-cov.csv`).

In [3]:
```
# import seda data
ca_main = pd.read_csv('data/ca-main.csv')
ca_cov = pd.read_csv('data/ca-cov.csv')
```

Test score data

The first few rows of the test data are shown below. The columns are:

Column name	Meaning
`sedalea`	District ID
`grade`	Grade level
`stateabb`	State abbreviation
`sedaleaname`	District name
`subject`	Test subject

Column name	Meaning
`cs_mn_...`	Estimated mean test score
`cs_mnse_...`	Standard error for estimated mean test score
`totgyb_...`	Number of individual tests used to estimate the mean score

In [4]:
```
ca_main.head(3)
```

Out[4]:

	sedalea	grade	stateabb	sedaleaname	subject	cs_mn_all	cs_mnse_all	totgyb_all	cs_mn_asn	cs_mnse_asn	...	totgyb_whg	cs_mn_wht	cs_mnse_wht	totgyb_wht	cs_mn_wmg	cs_mnse_wmg	totgyb_wmg	cs_
0	600001	4	CA	ACTON-AGUA DULCE UNIFIED ...	mth	-0.367007	0.108543	86.0	NaN	NaN	...	79.0	-0.208654	0.165783	35.0	-0.089003	0.518066	38.0	
1	600001	4	CA	ACTON-AGUA DULCE UNIFIED ...	rla	0.005685	0.117471	85.0	NaN	NaN	...	78.0	0.259587	0.189614	35.0	0.526942	0.602989	38.0	
2	600001	6	CA	ACTON-AGUA DULCE UNIFIED ...	rla	-0.000040	0.092172	114.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

3 rows × 59 columns

The test score means for each district are named `cs_mn_...` with an abbreviation indicating subgroup (such as mean score for all `cs_mean_all`, for boys `cs_mean_mal`, for white students `cs_mn_wht`, and so on). Notice that these are generally small-ish: decimal numbers between -0.5 and 0.5.

These means are *estimated* from a number of individual student tests and *standardized* relative to national averages. They represent the number of standard deviations by which a district mean differs from the national average. So, for instance, the value `cs_mn_all = 0.1` indicates that the district average is estimated to be 0.1 standard deviations greater than the national average on the corresponding test and at the corresponding grade level.

Q0 (a). Interpreting test score values

Interpret the average math test score for all 4th grade students in Acton-Agua Dulce Unified School District (the first row of the dataset shown above).

The average math test score for all 4th grade students in Acton-Agua Dulce Unified School District is 0.367 standard deviations lower than the national average for 4th graders.

Covariate data

The first few rows of the covariate data are shown below. The column information is as follows:

Column name	Meaning
`sedalea`	District ID
`grade`	Grade level
`sedaleanm`	District name
`urban`	Indicator: is the district in an urban locale?
`suburb`	Indicator: is the district in a suburban locale?
`town`	Indicator: is the district in a town locale?
`rural`	Indicator: is the district in a rural locale?
`locale`	Description of district locale
Remaining variables	Demographic and socioeconomic measures

In [5]:
```
ca_cov.head(3)
```

Out[5]:

	sedalea	grade	sedaleanm	urban	suburb	town	rural	locale	perind	perasn	...	snapall	snapblk	snaphsp	snapwht	single_momall	single_momblk	single_momhsp	single_momwht	seswhtblk	seswh
0	600001	4.0	ACTON-AGUA DULCE UNIFIED ...	0.0	0.0	0.0	1.0	Rural, Distant	0.003893	0.045901	...	0.035165	0.20293	0.0819	0.032362	0.084385	0.349636	0.198482	0.061653	1.839339	0.69
1	600001	5.0	ACTON-AGUA DULCE UNIFIED ...	0.0	0.0	0.0	1.0	Rural, Distant	0.003788	0.046652	...	0.035165	0.20293	0.0819	0.032362	0.084385	0.349636	0.198482	0.061653	1.839339	0.69
2	600001	6.0	ACTON-AGUA DULCE UNIFIED ...	0.0	0.0	0.0	1.0	Rural, Distant	0.003218	0.043657	...	0.035165	0.20293	0.0819	0.032362	0.084385	0.349636	0.198482	0.061653	1.839339	0.69

3 rows × 60 columns

You will only be working with a handful of the demographic and socioeconomic measures, so you can put off getting acquainted with those until selecting a subset of variables.

Q0 (b). Data semantics

In the non-public data, observational units are students -- test scores are measured for each student. However, in the SEDA data you've imported, scores are *aggregated* to the district level by grade. Let's regard estimated test score means for each grade as distinct variables, so that an observation consists in a set of estimated means for different grade levels and groups. In this view, what are the observational units in the test score dataset? Are they the same or different for the covariate dataset?

The observationl units for both the test score dataset and covariate dataset are districts in California.

Q0 (c). Sample sizes

How many observational units are in each dataset? Count the number of units in the test dataset and the number of units in the covariate dataset separately. Store the numbers as `ca_cov_units` and `ca_main_units`, respectively.

(*Hint*: use `.nunique()`.)

In [6]:
```
ca_cov_units = ca_cov.sedalea.nunique()
ca_main_units = ca_main.sedalea.nunique()
```

In [7]:
```
grader.check("q0_c")
```

Out[7]:

`q0_c` passed!

Q0 (d). Sample characteristics

Answer the questions below about the sampling design. You do not need to dig through any data documentation in order to resolve these questions.

(i) What is the relevant population for the datasets you've imported?

The relevant population for the imported datasets are all school districts in California.

(ii) About what proportion (to within 0.1) of the population is captured in the sample?

(*Hint*: have a look at this website.)

There were 855 total school districts captured in the sample while there are 1029 total school districts in California for a proportion of 0.83 of the sample captured.

(iii) Considering that the sampling frame is not identified clearly, what kind of dataset do you suspect this is (*e.g.*, administrative, data from a 'typical sample', census, etc.)?

I suspect that the dataset is administrative where this information was readily available from the U.S. Department of Education and socioeconomic/demographic information available from surveys.

Q0 (e). Scope of inference

In light of your description of the sample characteristics, what is the scope of inference for this dataset?

The scope of inference are California districts and counties for grade levels 4-8.

---

1. Tidy

Your goal will be to examine the relationship between gender achievement gaps and socioeconomic measures for school districts in California in 2018. In order to do this, the following manipulations of the imported data are needed:

- selecting columns of interest;
- filtering out non-urban districts;
- merging the covariate data with the test data; and
- putting the result in tidy format.

Since you've already had some guided practice doing this in previous assignments, you'll be left to fill in a little bit more of the details on your own in this assignment.

You'll work with the following variables from each dataset:

- **Test score data**
  - District ID
  - District name
  - Grade
  - Test subject
  - Estimated male-female gap
- **Covariate data**
  - District ID
  - Locale
  - Grade
  - Socioeconomic status (all demographic groups)
  - Log median income (all demographic groups)
  - Poverty rate (all demographic groups)
  - Unemployment rate (all demographic groups)
  - SNAP benefit receipt rate (all demographic groups)

Q1 (a). Variable names of interest

Download the codebooks by opening the 'data' directory from your Jupyter Lab file navigator (pstat100-s21-content > hw > hw2 > data), right-click the codebook .xlsx files, and select 'Download'. Identify the variables listed above, and store the column names in lists `main_vars` and `cov_vars`.

In [8]:
```
# store variable names of interest
main_vars = ['sedalea', 'sedaleaname', 'grade', 'subject', 'cs_mn_mfg']
cov_vars = ['sedalea', 'locale', 'grade', 'sesall',
            'lninc50all', 'povertyall', 'unempall', 'snapall']
```

In [9]:
```
grader.check("q1_a")
```

Out[9]:

q1_a passed!

Q1 (b). Slice columns

Use your result from Q1 (a) to slice the columns of interest from the covariate and test score data. Store the results as `main_sub` and `cov_sub`.

In [10]:
```
# slice columns to select variables of interest
main_sub = ca_main.loc[:, main_vars]
cov_sub = ca_cov.loc[:, cov_vars]
```

In [11]:
```
grader.check("q1_b")
```

Out[11]:

q1_b passed!

In the next step you'll merge the covariate data with the test score data. In order to do this, you can use the `pd.merge(A, B, how = ..., on = SHARED_COLS)` function, which will match the rows of `A` and `B` based on the shared columns `SHARED_COLS`. If `how = 'left'`, then only rows in `A` will be retained in the output (so `B` will be merged *to* `A`); conversely, if `how = 'right'`, then only rows in `B` will be retained in the output (so `A` will be merged *to* `B`).

A simple example of the use of `pd.merge` is illustrated below:

In [12]:
```
# toy data frames
A = pd.DataFrame(
    {'shared_col': ['a', 'b', 'c'],
    'x1': [1, 2, 3],
    'x2': [4, 5, 6]}
)

B = pd.DataFrame(
    {'shared_col': ['a', 'b'],
    'y1': [7, 8]}
)
```

In [13]:
```
A
```

Out[13]:

	shared_col	x1	x2
0	a	1	4
1	b	2	5
2	c	3	6

In [14]:
```
B
```

Out[14]:

	shared_col	y1
0	a	7
1	b	8

Below, if `A` and `B` are merged retaining the rows in `A`, notice that a missing value is input because `B` has no row where the shared column (on which the merging is done) has value `c`. In other words, the third row of `A` has no match in `B`.

In [15]:
```
# left join
```

```
pd.merge(A, B, how = 'left', on = 'shared_col')
```

Out[15]:

	shared_col	x1	x2	y1
0	a	1	4	7.0
1	b	2	5	8.0
2	c	3	6	NaN

If the direction of merging is reversed, and the row structure of `B` is dominant, then the third row of `A` is dropped altogether because it has no match in `B`.

In [16]:
```
# right join
pd.merge(A, B, how = 'right', on = 'shared_col')
```

Out[16]:

	shared_col	x1	x2	y1
0	a	1	4	7
1	b	2	5	8

Q1 (c). Merge

Follow the example above and merge the covariate and test score data on both the ***district ID*** and ***grade level***, retaining only the columns from the test score data (meaning, treat the test score data as primary and merge the covariate data *to* the test score data). Store the result as `rawdata` and print the first four rows.

Hint: When merging on multiple columns, you can utilize a list to hold both column names.

In [17]:
```
# merge covariates with gap data
rawdata = pd.merge(main_sub, cov_sub, how = 'left', on = ['sedalea', 'grade'])

# print first four rows
rawdata.head(4)
```

Out[17]:

	sedalea	sedaleaname	grade	subject	cs_mn_mfg	locale	sesall	lninc50all	povertyall	unempall	snapall
0	600001	ACTON-AGUA DULCE UNIFIED ...	4	mth	NaN	Rural, Distant	1.237209	11.392048	0.091894	0.048886	0.035165
1	600001	ACTON-AGUA DULCE UNIFIED ...	4	rla	NaN	Rural, Distant	1.237209	11.392048	0.091894	0.048886	0.035165
2	600001	ACTON-AGUA DULCE UNIFIED ...	6	rla	NaN	Rural, Distant	1.237209	11.392048	0.091894	0.048886	0.035165
3	600001	ACTON-AGUA DULCE UNIFIED ...	8	mth	-0.562855	Rural, Distant	1.237209	11.392048	0.091894	0.048886	0.035165

In [18]:
```
grader.check("q1_c")
```

Out[18]:

q1_c passed!

Q1 (d). Rename and reorder columns

Now rename and rearrange the columns of `rawdata` so that they appear in the following order and with the following names:

- District ID, District, Locale, log(Median income), Poverty rate, Unemployment rate, SNAP rate, Socioeconomic index, Grade, Gender gap, Subject

Store the result as `rawdata_mod1` and print the first four rows.

(*Hint*: first define a dictionary to map the old names to the new ones; then create a list of the new names specified in the desired order; then use `.rename()` and `.loc[]`. You can follow the renaming steps in HW1 as an example if needed.)

In [19]:
```
# define dictionary mapping for renaming columns
new_names = {'sedalea': 'District ID', 'sedaleaname': 'District', 'locale': 'Locale',
             'lninc50all': 'log(Median income)', 'povertyall': 'Poverty rate',
             'unempall': 'Unemployment rate', 'snapall': 'SNAP rate',
             'sesall': 'Socioeconomic index', 'grade': 'Grade', 'cs_mn_mfg': 'Gender gap',
             'subject': 'Subject'}

# specify order of columns
name_order = ['District ID', 'District', 'Locale', 'log(Median income)', 'Poverty rate',
            'Unemployment rate', 'SNAP rate', 'Socioeconomic index', 'Grade', 'Subject',
            'Gender gap']

# rename and reorder
rawdata_mod1 = rawdata.rename(columns = new_names).loc[:, name_order]

# print first four rows
rawdata_mod1.head(4)
```

Out[19]:

	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Grade	Subject	Gender gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	4	mth	NaN
1	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	4	rla	NaN
2	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	6	rla	NaN
3	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	8	mth	-0.562855

In [20]:
```
grader.check("q1_d")
```

Out[20]:

q1_d passed!

Q1 (e). Pivot

Notice that the Gender gap column contains the values of two variables: the gap in estimated mean test scores for math tests, and the gap in estimated mean test scores for reading and language tests. To put the data in tidier format, use `.pivot` to pivot the table so that the gender gap column is spread into two columns corresponding to the entries of `Subject`. Name the resulting columns `Math gap` and `Reading gap`, and store the result as `rawdata_mod2` and print the first four rows.

Comment: an alternative solution is to manipulate the indices and use `.unstack()`. Either method will produce a dataframe with hierarchical column indexing; this will need to be collapsed in order to rename the columns as instructed. You may find `MultiIndex.droplevel()` to be of use.

In [21]:
```
# pivot to unstack gender gap (fixing tidy issue: multiple variables in one column)
rawdata_mod2 = rawdata_mod1.reset_index().pivot(
    index = ['District ID', 'District', 'Locale',
             'log(Median income)', 'Poverty rate',
             'Unemployment rate', 'SNAP rate',
             'Socioeconomic index', 'Grade'],
    columns = ['Subject'],
    values = 'Gender gap'
).reset_index().rename(
    columns = {'mth': 'Math gap', 'rla': 'Reading gap'}
)

# print first four rows
rawdata_mod2.head()
```

Out[21]:

Subject	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Grade	Math gap	Reading gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	4	NaN	NaN
1	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	6	NaN	NaN
2	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	8	-0.562855	-0.785321
3	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	1.912972	4	-0.025131	-0.521408
4	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	1.912972	5	0.143163	-0.210279

In [22]:

```
grader.check("q1_e")
```

Out[22]:

q1_e passed!

Q1 (f). Indexing

If necessary, remove the name of the column index ('Subject') that was induced by the pivot step using `.rename_axis()`, and store the result as `data`; otherwise, simply store a copy of the previous dataframe as `data`. Print the first four rows.

In [23]:

```
# drop the name of column index induced by pivoting
data = rawdata_mod2.rename_axis(columns=None)

# print first four rows
data.head(4)
```

Out[23]:

	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Grade	Math gap	Reading gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	4	NaN	NaN
1	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	6	NaN	NaN
2	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	8	-0.562855	-0.785321
3	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	1.912972	4	-0.025131	-0.521408

Your final dataset should match the dataframe below. You can use this to check your answer and revise any portions above that lead to different results.

In [24]:

```
# intended result
data_reference = pd.read_csv('data/tidy-seda.csv')
data_reference.head(4)
```

Out[24]:

	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Grade	Math gap	Reading gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	4.0	NaN	NaN
1	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	6.0	NaN	NaN
2	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	8.0	-0.562855	-0.785321
3	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	1.912972	4.0	-0.025131	-0.521408

Q1 (g). Sanity check

Ensure that your tidying did not inadvertently drop any observations: count the number of units in `data`. Does this match the number of units represented in the original test score data `ca_main`? Store these as `data_units` and `ca_main_units`, respectively.

(*Hint*: use `.nunique()`.)

In [25]:

```
# number of districts in tidied data compared with raw
data_units = data['District ID'].nunique()
ca_main_units = ca_main.sedalea.nunique()
```

In [26]:

```
grader.check("q1_g")
```

Out[26]:

q1_g passed!

Q1 (h). Missing values

Gap estimates were not calculated for certain grades in certain districts due to small sample sizes (not enough individual tests recorded).

(i) What proportion of rows are missing for each of the reading and math gap variables?

Store these as `math_missing` and `reading_missing`, respectively.

Hint: Can utilize the fact that both columns have the ending of "gap" to subset the dataframe.

In [27]:

```
# proportion of missing values
math_missing = data['Math gap'].isna().mean()
reading_missing = data['Reading gap'].isna().mean()
```

In [28]:

```
grader.check("q1_h_i")
```

Out[28]:

q1_h_i passed!

(ii) What proportion of *districts* have missing gap estimates for one or both test subjects for at least one grade level?

Save the value as `district_missing`.

In [29]:

```
# proportion of districts with missing values
total_districts = data['District'].nunique()

district_missing = data[data['Math gap'].isna() | data['Reading gap'].isna()].loc[
    :, ['District', 'Math gap', 'Reading gap']
].groupby('District').mean().reset_index().shape[0]/total_districts

district_missing
```

Out[29]:

```
0.5110336817653891
```

In [30]:

```
data[data['Math gap'].isna() | data['Reading gap'].isna()]
```

Out[30]:

	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Grade	Math gap	Reading gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	4	NaN	NaN
1	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	6	NaN	NaN
6	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	1.912972	7	NaN	-0.20361
8	600011	FORT SAGE UNIFIED ...	Rural, Distant	10.704570	0.159981	0.066333	0.102054	-0.478127	5	NaN	NaN
9	600011	FORT SAGE UNIFIED ...	Rural, Distant	10.704570	0.159981	0.066333	0.102054	-0.478127	7	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
3834	691136	AROMAS/SAN JUAN UNIFIED ...	Rural, Distant	11.228895	0.113883	0.056227	0.081426	0.825116	7	NaN	NaN
3836	691137	BIG OAK FLAT-GROVELAND UNIFIED ...	Rural, Distant	10.928622	0.113149	0.057457	0.118839	0.477981	5	NaN	NaN
3837	691137	BIG OAK FLAT-GROVELAND UNIFIED ...	Rural, Distant	10.928622	0.113149	0.057457	0.118839	0.477981	6	NaN	NaN
3838	691137	BIG OAK FLAT-GROVELAND UNIFIED ...	Rural, Distant	10.928622	0.113149	0.057457	0.118839	0.477981	7	NaN	NaN
3839	691137	BIG OAK FLAT-GROVELAND UNIFIED ...	Rural, Distant	10.928622	0.113149	0.057457	0.118839	0.477981	8	NaN	NaN

1162 rows × 11 columns

In [31]:
```
grader.check("q1_h_ii")
```

Out[31]: **q1_h_ii** passed!

(iii) Do you expect that this missingness is related to any particular district attribute(s)?

I suspect that the missingness for gap estimates is related to the district's locale since most of the missing values come from rural districts.

---

2. Explore

For the purpose of visualizing the relationship between estimated gender gaps and socioeconomic variables, you'll find it more helpful to store a non-tidy version of the data. The cell below rearranges the dataset so that one column contains an estimated gap, one column contains the value of a socioeconomic variable, and the remaining columns record the gap type and variable identity.

Ensure that your results from part 1 match the reference dataset before running this cell.

In [32]:
```
# format data for plotting
plot_df = data.melt(
    id_vars = name_order[0:9],
    value_vars = ['Math gap', 'Reading gap'],
    var_name = 'Gap type',
    value_name = 'Gap'
).melt(
    id_vars = ['District ID', 'District', 'Locale', 'Gap type', 'Gap', 'Grade'],
    value_vars = name_order[3:8],
    var_name = 'Socioeconomic variable',
    value_name = 'Measure'
)

# preview
plot_df.head()
```

Out[32]:

	District ID	District	Locale	Gap type	Gap	Grade	Socioeconomic variable	Measure
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	Math gap	NaN	4	log(Median income)	11.392048
1	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	Math gap	NaN	6	log(Median income)	11.392048
2	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	Math gap	-0.562855	8	log(Median income)	11.392048
3	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	Math gap	-0.025131	4	log(Median income)	11.607236
4	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	Math gap	0.143163	5	log(Median income)	11.607236

Altair, by default, limits the number of rows for input dataframes. We will need to disable this behavior in order to generate plots of this dataset.

In [33]:
```
# disable row limit for plotting
alt.data_transformers.disable_max_rows()
```

Out[33]: DataTransformerRegistry.enable('default')

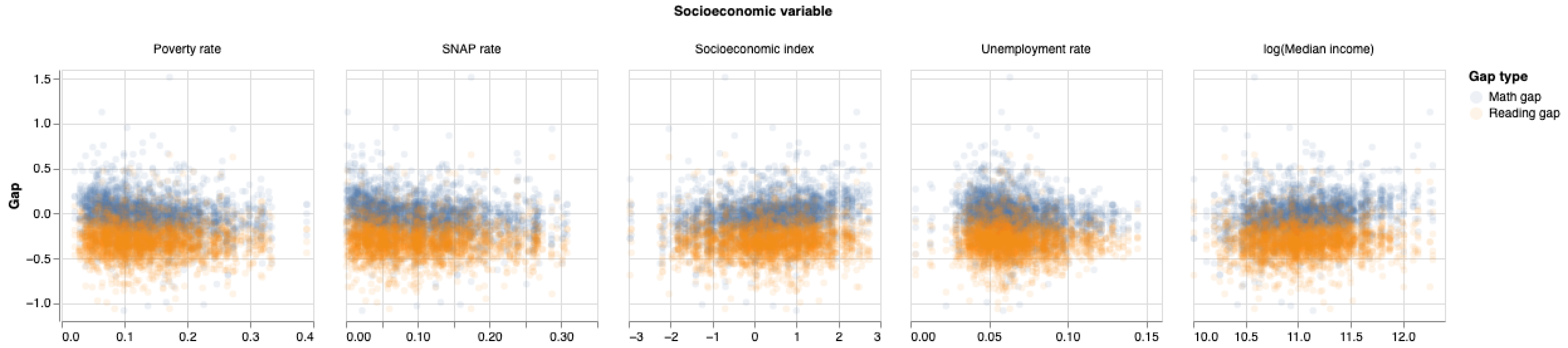
Relationship between gender gaps and socioeconomic factors

The cell below generates a panel of scatterplots showing the relationship between estimated gender gap and socioeconomic factors for all grade levels by test subject. The plot suggests that the reading gap favors girls consistently across the socioeconomic spectrum -- in a typical district girls seem to outperform boys by 0.25 standard deviations of the national average. By contrast, the math gap appears to depend on socioeconomic factors -- boys only seem to outperform girls under *better* socioeconomic conditions.

In [34]:
```
# plot gap against socioeconomic variables by subject for all grades
fig1 = alt.Chart(plot_df).mark_circle(opacity = 0.1).encode(
    y = 'Gap',
    x = alt.X('Measure', scale = alt.Scale(zero = False), title = ''),
    color = 'Gap type'
).properties(
    width = 200,
    height = 200
).facet(
    column = alt.Column('Socioeconomic variable')
).resolve_scale(x = 'independent')

fig1
```

Out[34]:



Q2 (a). Relationships by grade level

Does the pattern shown in the plot above persist within each grade level?

(i)

Modify the plot above to show these relationships by grade level: generate a panel of scatterplots of gap against socioeconomic measures by subject, where each column of the panel corresponds to one socioeconomic variable and each row corresponds to one grade level; the result should by a 5x5 panel. Resize the width and height of each facet so that the panel is of reasonable size.

(*Hint*: you may find it useful to have a look at the altair documentation on compound charts, and lab 2, for examples to follow.)

In [35]:
```
# plotting codes here
fig2a = alt.Chart(plot_df).mark_circle(opacity = 0.1).encode(
    y = 'Gap',
    x = alt.X('Measure', scale = alt.Scale(zero = False), title = ''),
    color = 'Gap type'
).properties(
    width = 200,
    height = 200
).facet(
    column = alt.Column('Socioeconomic variable'),
    row = 'Grade'
).resolve_scale(x = 'independent')

# display
fig2a
```

Out[35]:

(ii) is the pattern consistent across grade level?

From the plots, the pattern appears to be consistent across grade levels, and continues to suggest that the reading gap favors girls regardless of grade level.

Q2 (b). Do gaps shift across grade levels?

(i)

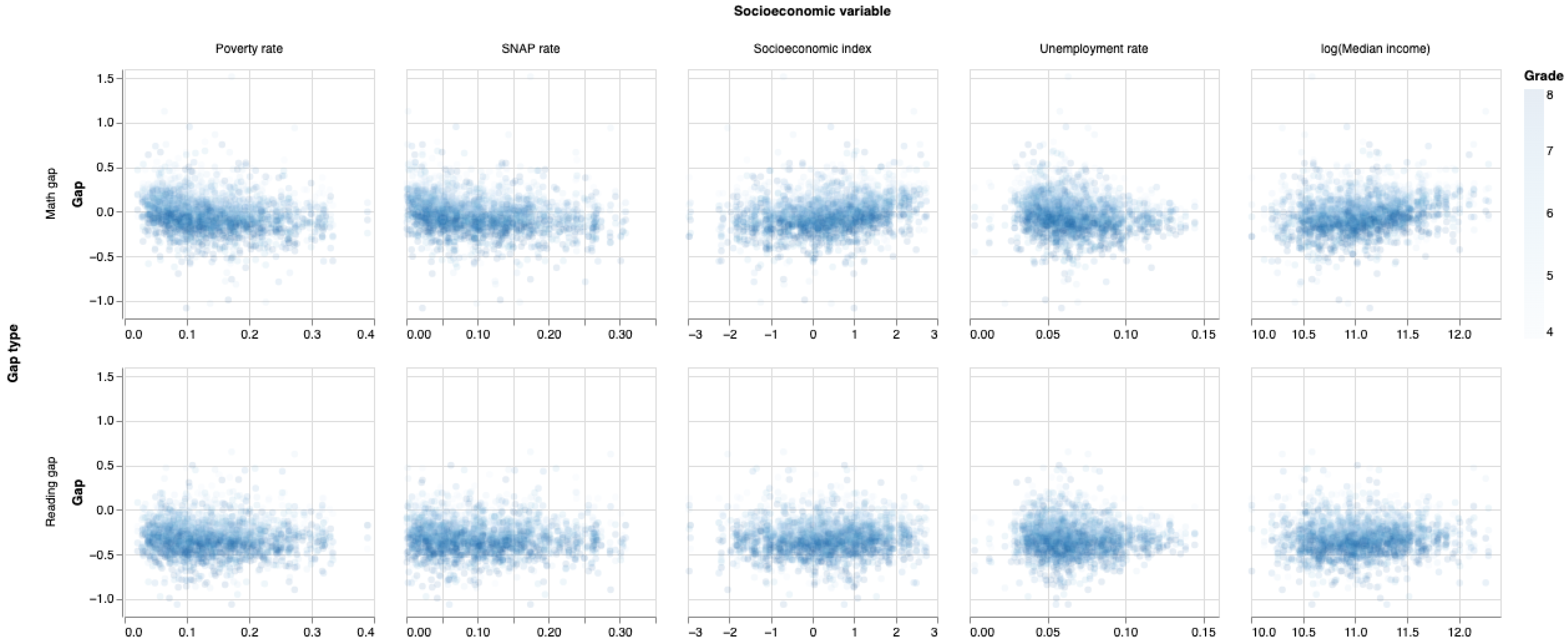
Construct a 2x5 panel of scatterplots showing estimated achievement gap against each of the 5 socioeconomic variables, with one row per test subject. Display grade level using a color gradient.

(*Hint:* plot gap against measure, facet by gap type (rows) and socioeconomic variable (columns), and color by grade.)

In [36]:
```
# plotting codes here
fig2b = alt.Chart(plot_df).mark_circle(opacity = 0.1).encode(
    y = 'Gap',
    x = alt.X('Measure', scale = alt.Scale(zero = False), title = ''),
    color = alt.Color('Grade')
).properties(
    width = 200,
    height = 200
).facet(
    column = alt.Column('Socioeconomic variable'),
    row = 'Gap type'
).resolve_scale(x = 'independent')


# display
fig2b
```

Out[36]:



(ii) Do the gaps seem to shift with grade level?

The color gradient allows us to see that gaps do tend to shift with grade level. Younger grade levels tend to have a larger gap. For 8th graders, the gap appears to center about 0 for math and for the reading gap, about 0.25 standard deviations (in favor of girls). Additionally, there are more outliers for grade level 4 with math gaps favoring boys.

Aggregating by grade

While the magnitude of the achievement gaps seems to depend very slightly on grade level (figure 2b), the *relationship* between achievement gap and socioeconomic factors does not differ from grade to grade (figure 2a). In what follows, you'll look at the average relationship between estimated achievement gap and median income after aggregating across grade. The cell below computes the mean of each variable across grade levels for each district.

In [37]:

```
# aggregate across grades
data_agg = data.groupby(['District ID', 'District', 'Locale']).mean().reset_index().drop(columns = 'Grade')
data_agg.head()
```

Out[37]:

	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Math gap	Reading gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	1.237209	-0.562855	-0.785321
1	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	1.912972	0.061163	-0.242572
2	600011	FORT SAGE UNIFIED ...	Rural, Distant	10.704570	0.159981	0.066333	0.102054	-0.478127	-0.015417	-0.191400
3	600012	TWIN RIDGES ELEMENTARY ...	Rural, Distant	10.589787	0.179102	0.059158	0.074903	-0.096379	NaN	NaN
4	600013	ROCKLIN UNIFIED ...	Suburb, Large	11.399662	0.060338	0.045533	0.035016	1.398133	0.054454	-0.312638

Similar to working with the disaggregated data, it will be helpful for plotting to melt the two gap variables into a single column.

In [38]:

```
# format for plotting
agg_plot_df = data_agg.melt(
    id_vars = name_order[0:7],
    value_vars = ['Math gap', 'Reading gap'],
    var_name = 'Subject',
    value_name = 'Average estimated gap'
)

agg_plot_df.head()
```

Out[38]:

	District ID	District	Locale	log(Median income)	Poverty rate	Unemployment rate	SNAP rate	Subject	Average estimated gap
0	600001	ACTON-AGUA DULCE UNIFIED ...	Rural, Distant	11.392048	0.091894	0.048886	0.035165	Math gap	-0.562855
1	600006	ROSS VALLEY ELEMENTARY ...	Suburb, Large	11.607236	0.041418	0.048269	0.028006	Math gap	0.061163
2	600011	FORT SAGE UNIFIED ...	Rural, Distant	10.704570	0.159981	0.066333	0.102054	Math gap	-0.015417
3	600012	TWIN RIDGES ELEMENTARY ...	Rural, Distant	10.589787	0.179102	0.059158	0.074903	Math gap	NaN
4	600013	ROCKLIN UNIFIED ...	Suburb, Large	11.399662	0.060338	0.045533	0.035016	Math gap	0.054454

Q2 (c). District average gaps

Construct a scatterplot of the average estimated gap against log(Median income) by subject for each district and add trend lines.

In [39]:

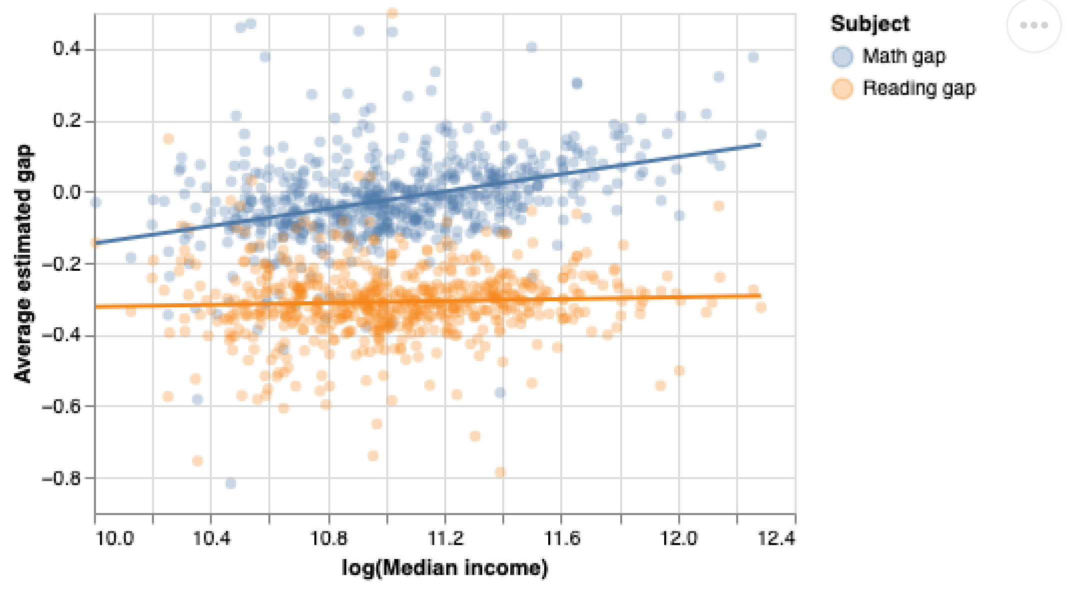
```
# scatterplot
scatter = alt.Chart(agg_plot_df).mark_circle(opacity = 0.3).encode(
    y = 'Average estimated gap',
    x = alt.X('log(Median income)', scale = alt.Scale(zero = False), title = 'log(Median income)'),
    color = alt.Color('Subject')
).properties(
    width = 350,
    height = 250
)

# trend line
trend = scatter.transform_regression(
    groupby = ['Subject'],
    on = 'log(Median income)',
    regression = 'Average estimated gap'
).mark_line(color = 'black')

# combine layers
fig2c = scatter + trend

# display
fig2c
```

Out[39]:



Now let's try to capture this pattern in *tabular* form. The cell below adds an `Income bracket` variable by cutting the median income into 8 contiguous intervals using `pd.cut()`, and tabulates the average socioeconomic measures and estimated gaps across districts by income bracket. Notice that with respect to the gaps, this displays the pattern that is shown visually in the figures above.

In [40]:
```
data_agg['Income bracket'] = pd.cut(np.e**data_agg['log(Median income)'], 8)
data_agg.groupby('Income bracket').mean().drop(columns = ['District ID', 'log(Median income)'])
```

Out[40]:

Income bracket	Poverty rate	Unemployment rate	SNAP rate	Socioeconomic index	Math gap	Reading gap
(21980.176, 46455.372]	0.194870	0.072689	0.155061	-0.651999	-0.070284	-0.309743
(46455.372, 70736.321]	0.134078	0.063788	0.095303	0.291085	-0.034061	-0.315545
(70736.321, 95017.269]	0.088713	0.052785	0.048242	1.110433	0.004239	-0.302114
(95017.269, 119298.218]	0.064131	0.046848	0.030548	1.640159	0.050006	-0.287117
(119298.218, 143579.167]	0.050315	0.044343	0.011023	2.167272	0.090138	-0.289529
(143579.167, 167860.115]	0.043896	0.042379	0.008451	2.382258	0.084683	-0.335975
(167860.115, 192141.064]	0.040552	0.040120	0.010159	2.652906	0.175793	-0.232306
(192141.064, 216422.013]	0.047097	0.054055	0.002555	2.588499	0.267301	-0.299798

Q2 (d). Proportion of districts with a math gap.

What proportion of districts in each income bracket have an average estimated math achievement gap favoring boys? Answer this question by performing the following steps:

- Append an indicator variable `Math gap favoring boys` to `data_agg` that records whether the average estimated math gap favors boys by more than 0.1 standard deviations relative to the national average.
- Compute the proportion of districts in each income bracket for which the indicator is true: group by bracket and take the mean. Store this as `income_bracket_boys_favored`

In [41]:
```
# define indicator
data_agg['Math gap favoring boys'] = data_agg['Math gap'] > 0.1

# proportion of districts with gap favoring boys, by income bracket
income_bracket_boys_favored = data_agg.groupby(
    ['Income bracket']
).mean().loc[:, 'Math gap favoring boys'].reset_index()

income_bracket_boys_favored
```

Out[41]:

	Income bracket	Math gap favoring boys
0	(21980.176, 46455.372]	0.036585
1	(46455.372, 70736.321]	0.061224
2	(70736.321, 95017.269]	0.084337
3	(95017.269, 119298.218]	0.232143
4	(119298.218, 143579.167]	0.388889
5	(143579.167, 167860.115]	0.444444
6	(167860.115, 192141.064]	0.500000
7	(192141.064, 216422.013]	1.000000

In [42]:
```
grader.check("q2_d")
```

Out[42]:

q2_d passed!

Q2 (e). Statewide averages

To wrap up the exploration, calculate a few statewide averages to get a sense of how some of the patterns above compare with the state as a whole.

(i) Compute the statewide average estimated achievement gaps.

Store the result as `state_avg`.

In [43]:
```
# statewide average
state_avg = data_agg.loc[:, ['Math gap', 'Reading gap']].mean()

state_avg
```

Out[43]:
```
Math gap      -0.020746
Reading gap   -0.308014
dtype: float64
```

In [44]:
```
grader.check("q2_e_i")
```

Out[44]:

q2_e_i passed!

(ii) Compute the proportion of districts in the state with a math gap favoring boys.

Store this result as `math_boys_proportion`

In [45]:
```
# proportion of districts in the state with a math gap favoring boys
total_districts = data_agg['District'].nunique()

math_boys_proportion = data_agg[data_agg['Math gap favoring boys'] == True].loc[
    :, 'Math gap favoring boys'
].sum()/total_districts

math_boys_proportion
```

Out[45]:
```
0.08421052631578947
```

In [46]:
```
grader.check("q2_e_ii")
```

Out[46]:

q2_e_ii passed!

(iii) Compute the proportion of districts in the state with a math gap favoring girls.

You will need to define a new indicator within `data_agg` to perform this calculation.

In [47]:
```
# new indicator
data_agg['Math gap favoring girls'] = data_agg['Math gap'] < -0.1

# proportion of districts in the state with a math gap favoring girls
total_districts = data_agg['District'].nunique()

math_girls_proportion = data_agg[data_agg['Math gap favoring girls'] == True].loc[
    :, 'Math gap favoring girls'
].sum()/total_districts

math_girls_proportion
```

Out[47]:
```
0.1391812865497076
```

In [48]:
```
grader.check("q2_e_iii")
```

Out[48]:

q2_e_iii passed!

---

3. Communicating results

Take a moment to review and reflect on your findings, and then answer the questions below.

Q3 (a). Summary

Write a brief summary of your exploratory analysis. What have you discovered about educational achievement gaps in California school districts? Aim to answer in 3-5 sentences or less.

My exploratory analysis suggests that for California in 2018, girls typically outperformed boys in both reading and math. The California average achievement gap for math and reading, respectively, favors girls by 0.0207 and 0.308 standard deviations relative to the national average. It was also observed that socioeconommic factors played into the achievement gaps such that there is a positively linear relationship between family income and math gaps favoring boys. However, this association does not apply to reading gaps; the reading gap remained fixed favoring girls regardless of family wealth.

Q3 (b). Hypothesize!

It's a cliche in statistics that 'correlation is not causation'. In your exploratory analysis, you identified a correlation between socioeconomic factors and achievement gaps. But clearly, affluence does not directly cause a math achievement gap favoring boys. What factors do you think might explain this association?

Wealthier families are more likely to afford resources such as math tutors that would excel boys' mathematical performance. This association might also be due to the fact that at grades 4-8, boys of this age may have behavioral issues and thus parents and teachers will be monitoring their school progress. Whereas for girls, parents may assume their daughter won't need additional help if they are well-behaved in school.

Submission

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Generate PDF copy
5. Submit both notebook and PDF to Gradescope

To double-check your work, the cell below will rerun all of the autograder tests.

In [49]:
```
grader.check_all()
```

Out[49]:
```
q0_c results: All test cases passed!

q1_a results: All test cases passed!

q1_b results: All test cases passed!

q1_c results: All test cases passed!

q1_d results: All test cases passed!

q1_e results: All test cases passed!

q1_g results: All test cases passed!

q1_h_i results: All test cases passed!

q1_h_ii results: All test cases passed!

q2_d results: All test cases passed!

q2_e_i results: All test cases passed!

q2_e_ii results: All test cases passed!

q2_e_iii results: All test cases passed!
```