In [ ]:

```
# Initialize Otter
import otter
grader = otter.Notebook("hw1-brfss.ipynb")
```

In [ ]:

```
import numpy as np
import pandas as pd
import altair as alt
# disable row limit for plotting
alt.data_transformers.disable_max_rows()
# uncomment to ensure graphics display with pdf export
# alt.renderers.enable('mimetype')
```

Background

The Behavioral Risk Factor Surveillance System (BRFSS) is a long-term effort administered by the CDC to collect data on behaviors affecting physical and mental health, past and present health conditions, and access to healthcare among U.S. residents. The BRFSS comprises telephone surveys of U.S. residents conducted annually since 1984; in the last decade, over half a million interviews have been conducted each year. This is the largest such data collection effort in the world, and many countries have developed similar programs. The objective of the program is to support monitoring and analysis of factors influencing public health in the United States.

Each year, a standard survey questionnaire is developed that includes a core component comprising questions about: demographic and household information; health-related perceptions, conditions, and behaviors; substance use; and diet. Trained interviewers in each state call randomly selected telephone (landline and cell) numbers and administer the questionnaire; the phone numbers are chosen so as to obtain a representative sample of all households with telephone numbers. Take a moment to read about the 2019 survey here.

In this assignment you'll import and subsample the BRFSS 2019 data and perform a simple descriptive analysis exploring associations between adverse childhood experiences, health perceptions, tobacco use, and depressive disorders. This is an opportunity to practice:

- review of data documentation
- data assessment and critical thinking about data collection
- dataframe transformations in pandas
- communicating and interpreting grouped summaries

Data import and assessment

The cell below imports select columns from the 2019 dataset as a pandas DataFrame. The file is big, so this may take a few moments. Run the cell and then have a quick look at the first few rows and columns.

In [ ]:

```
# store variable names of interest
selected_vars = ['_SEX', '_AGEG5YR',
                 'GENHLTH', 'ACEPRISN',
                 'ACEDRUGS', 'ACEDRINK',
                 'ACEDEPRS', 'ADDEPEV3',
                 '_SMOKER3', '_LLCPWT']

# import full 2019 BRFSS dataset
brfss = pd.read_csv('data/brfss2019.zip', compression = 'zip', usecols = selected_v

# invert sampling weights
brfss['_LLCPWT'] = 1/brfss._LLCPWT

# print first few rows
brfss.head()
```

Question 1: Data dimensions

Check the dimensions of the dataset. Store the dimensions as `nrows` and `ncolumns`.

In [ ]:

```
nrows, ncolumns = ...

print(nrows, ncolumns)
```

In [ ]:

```
grader.check("q1")
```

Question 2: Row and column information

Now that you've imported the data, you should verify that the dimensions conform to the format you expect based on data documentation and ensure you understand what each row and each column represents.

Check the number of records (interviews conducted) reported and variables measured for 2019 by reviewing the surveillance summaries by year, and then answer the following questions in a few sentences:

- Does the number of rows match the number of reported records?
- How many columns were imported, and how many columns are reported in the full dataset?
- What does each row in the `brfss` dataframe represent?
- What does each column in the `brfss` dataframe represent

Type your answer here, replacing this text.

Question 3: Sampling design and data collection

Skim the overview documentation for the 2019 BRFSS data. Focus specifically the 'Background' and 'Data Collection' sections, read selectively for relevant details, and answer the following questions in a few sentences:

i. Who conducts the interviews and how long does a typical interview last?

ii. Who does an interviewer speak to in each household?

iii. What criteria must a person meet to be interviewed?

iv. Who *can't* appear in the survey? Give two examples.

v. What is the study population (*i.e.*, all individuals who could possibly be sampled)?

vi. Does the data contain any identifying information?

Type your answer here, replacing this text.

Question 4: Variable descriptions

You'll work with the small subset variables imported above: sex, age, general health self-assessment, smoking status, depressive disorder, and adverse childhood experiences (ACEs). The names of these variables as they appear in the raw dataset are defined in the cell in which you imported the data as `selected_vars`. It is often useful, and therefore good practice, to include a brief description of each variable at the outset of any reported analyses, both for your own clarity and for that of any potential readers. Open the 2019 BRFSS codebook in your browser and use text searching to locate each of the variable names of interest. Read the codebook entries and fill in the second column in the table below with a one-sentence description of each variable identified in `selected_vars`. Rephrase the descriptions in your own words -- do not copy the codebook descriptions verbatim.

Variable name	Description
`GENHLTH`	
`_SEX`	
`_AGEG5YR`	
`ACEPRISN`	
`ACEDRUGS`	
`ACEDRINK`	
`ACEDEPRS`	
`ADDEPEV3`	

Subsampling

To simplify life a little, we'll draw a large random sample of the rows and work with that in place of the full dataset. This is known as **subsampling**.

The cell below draws a random subsample of 10k records. Because the subsample is randomly drawn, we should not expect it to vary in any systematic way from the overall dataset, and distinct subsamples should have similar properties -- therefore, results downstream should be similar to an analysis of the full dataset, and should also be possible to replicate using distinct subsamples.

In [ ]:

```
# for reproducibility
np.random.seed(32221)

# randomly sample 10k records
samp = brfss.sample(n = 10000,
                    replace = False,
                    weights = '_LLCPWT')
```

Asides:

- Notice that the random number generator seed is set before carrying out this task -- this ensures that every time the cell is run, the same subsample is drawn. As a result, the computations in this notebook are *reproducible*: when I run the notebook on my computer, I get the same results as you get when you run the notebook on your computer.
- Notice also that *sampling weights* provided with the dataset are used to draw a weighted sample. Some respondents are more likely to be selected than others from the general population of U.S. adults with phone numbers, so the BRFSS calculates derived weights that are inversely proportional to estimates of the probability that the respondent is included in the survey. This is a somewhat sophisticated calculation, however if you're interested, you can read about how these weights are calculated and why in the overview documentation you used to answer the questions above. We use the sampling weights in drawing the subsample so that we get a representative sample of U.S. adults with phone numbers.
- Notice the missing values. How many entries are missing in each column? The cell below computes the proportion of missing values for each of the selected variables. We'll return to this issue later on.

In [ ]:

```
# proportions of missingness
samp.isna().mean()
```

Tidying

In the following series of questions you'll tidy up the subsample by performing these steps:

- selecting columns of interest;
- replacing coded values of question responses with responses;
- defining new variables based on existing ones;
- renaming columns.

The goal of this is to produce a clean version of the dataset that is well-organized, intuitive to navigate, and ready for analysis.

The variable entries are coded numerically to represent certain responses. These should be replaced by more informative entries. We can use the codebook to determine which number means what, and replace the values accordingly.

The cell below replaces the numeric values for `_AGEG5YR` by their meanings, illustrating how to use `.replace()` with a dictionary to convert the numeric coding to interpretable values. The basic strategy is:

1. Store the variable coding for `VAR` as a dictionary `var_codes`.
2. Use `.replace({'VAR': var_codes})` to modify values.

If you need additional examples, check the pandas documentation for `.replace()`.

In [ ]:

```
# dictionary representing variable coding
age_codes = {
    1: '18-24', 2: '25-29', 3: '30-34',
    4: '35-39', 5: '40-44', 6: '45-49',
    7: '50-54', 8: '55-59', 9: '60-64',
    10: '65-69', 11: '70-74', 12: '75-79',
    13: '80+', 14: 'Unsure/refused/missing'
}

# recode age categories
samp_mod1 = samp.replace({'_AGEG5YR': age_codes})

# check result
samp_mod1.head()
```

Question 5: Recoding variables

Following the example immediately above and referring to the 2019 BRFSS codebook, replace the numeric codings with response categories for each of the following variables:

- `_SEX`
- `GENHLTH`

- `_SMOKER3`

Notice that above, the first modification (slicing) was stored as `samp_mod1`, and was a function of `samp`. You'll follow this pattern, creating `samp_mod2`, `samp_mod3`, and so on so that each step (modification) of your data manipulations is stored separately, for easy troubleshooting.

i. Recode `_SEX`: define a new dataframe `samp_mod2` that is the same as `samp_mod1` but with the `_SEX` variable recoded as `M` and `F`.

ii. Recode `GENHLTH`: define a new dataframe `samp_mod3` that is the same as `samp_mod2` but with the `GENHLTH` variable recoded as `Excellent`, `Very good`, `Good`, `Fair`, `Poor`, `Unsure`, and `Refused`.

iii. Recode `_SMOKER3`: define a new dataframe `samp_mod4` that is the same as `samp_mod3` but with `_SMOKER3` recoded as `Daily`, `Some days`, `Former`, `Never`, and `Unsure/refused/missing`.

iv. Print the first few rows of `samp_mod4`.

In [ ]:

```
# define dictionary for sex
sex_codes = ...

# recode sex
samp_mod2 = ...

# define dictionary for health
health_codes = ...

# recode health
samp_mod3 = ...

# define dictionary for smoking
smoke_codes = ...

# recode smoking
samp_mod4 = ...

# print a few rows
...
```

In [ ]:

```
grader.check("q5")
```

Question 6: Value replacement

Now all the variables *except* the adverse childhood experience and depressive disorder question responses are represented interpretably. In the codebook that the answer key is identical for these remaining variables.

The numeric codings can be replaced all at once by applying `.replace()` to the dataframe with an argument of the form

- `df.replace({'var1': varcodes1, 'var2': varcodes1, ..., 'varp': varcodesp})`

Define a new dataframe `samp_mod5` that is the same as `samp_mod4` but with the remaining variables recoded according to the answer key `Yes` , `No` , `Unsure` , `Refused` . Print the first few rows of the result using `.head()` .

```
In [ ]: # define dictionary
        answer_codes = ...

        # recode
        samp_mod5 = ...

        # check using head()
        ...
```

```
In [ ]: grader.check("q6")
```

Finally, all the variables in the dataset are categorical. Notice that the current data types do not reflect this.

```
In [ ]: samp_mod5.dtypes
```

Let's coerce the variables to `category` data types using `.astype()` .

```
In [ ]: # coerce to categorical
        samp_mod6 = samp_mod5.astype('category')

        # check new data types
        samp_mod6.dtypes
```

Question 7: Define ACE indicator variable

Downstream analysis of ACEs will be facilitated by having an indicator variable that is a `1` if the respondent answered 'Yes' to any ACE question, and a `0` otherwise -- that way, you can easily count the number of respondents reporting ACEs by summing up the indicator or compute the proportion by taking an average.

To this end, define a new logical variable:

- `adverse_conditions` : did the respondent answer yes to any of the adverse childhood condition questions?

You can accomplish this task in several steps:

1. Obtain a logical array indicating the positions of the ACE variables (hint: use `.columns` to obtain the column index and operate on the result with `.str.startswith(...)`). Store this as `ace_positions`.
2. Use the logical array `ace_positions` to select the ACE columns via `.loc[]`. Store this as `ace_data`.
3. Obtain a dataframe that indicates whether each entry is a 'Yes' (hint: use the boolean operator `==`, which is a vectorized operation). Store this as `ace_yes`.
4. Compute the row sums using `.sum()`. Store this as `ace_numyes`.
5. Define the new variable as `ace_numyes > 0`.

Store the result as `samp_mod7`, and print the first few rows using `.head()`.

In [ ]:

```
# copy samp_mod6
samp_mod7 = samp_mod6.copy()

# ace column positions
ace_positions = ...

# ace data
ace_data = ...

# ace yes indicators
ace_yes = ...

# number of yesses
ace_numyes = ...

# assign new variable
samp_mod7['adverse_conditions'] = ...

# check result using .head()
...
```

In [ ]:

```
grader.check("q7")
```

Question 8: Define missingness indicator variable

As you saw earlier, there are some missing values for the ACE questions. These arise whenever a respondent is not asked these questions. In fact, answers are missing for nearly 80% of the respondents in our subsample. We should keep track of this information. Define a missing indicator:

- `adverse_missing`: is a response missing for at least one of the ACE questions?

In [ ]:

```
# copy modification 7
samp_mod8 = samp_mod7.copy()

# define missing indicator using loc
...
```

```
# check using head()
```

In [ ]:

```
grader.check("q8")
```

Question 9: Filter respondents who did not answer ACE questions

Since values are missing for the ACE question if a respondent was not asked, we can remove these observations and do any analysis *conditional on respondents having been asked the ACE questions*. Use your indicator variable `adverse_missing` to filter out respondents who were not asked the ACE questions.

Note that this dramatically limits the scope of inference for subsequent analyses to only those locations where the ACE module was included in the survey.

In [ ]:

```
samp_mod9 = ...
```

In [ ]:

```
grader.check("q9")
```

Question 10: Define depression indicator variable

It will prove similarly helpful to define an indicator for reported depression:

- `depression` : did the respondent report having been diagnosed with a depressive disorder?

Follow the same strategy as above for the ACE variables, and store the result as `samp_mod10` . See if you can perform the calculation of the new variable in a single line of code. Print the first few rows using `.head()` .

In [ ]:

```
# copy samp_mod9
samp_mod10 = samp_mod9.copy()

# define new variable using loc
...

# check using .head()
...
```

In [ ]:

```
grader.check("q10")
```

Question 11: Final dataset

For the final dataset, drop the respondent answers to individual questions, the missingness indicator, and select just the derived indicator variables along with general health, sex, age, and smoking status. Check the pandas documentation for `.rename()` and follow the examples to rename the latter variables:

- general_health
- sex
- age
- smoking

See if you can perform both operations (slicing and renaming) in a single chain. Store the result as `data`.

In [ ]:

```
samp_mod10.columns
```

In [ ]:

```
# slice and rename
data = ...

# check using .head()
```

In [ ]:

```
grader.check("q11")
```

Descriptive analysis

Now that you have a clean dataset, you'll use grouping and aggregation to compute several summary statistics that will help you explore whether there is an apparent association between experiencing adverse childhood conditions and self-reported health, smoking status, and depressive disorders in areas where the ACE module was administered.

The basic strategy will be to calculate the proportions of respondents who answered yes to one of the adverse experience questions when respondents are grouped by the other variables.

Question 12: Proportion of respondents reporting ACEs

Calculate the overall proportion of respondents in the subsample that reported experiencing at least one adverse condition (given that they answered the ACE questions). Use `.mean()`; store the result as `mean_ace` and print.

In [ ]:

```
# proportion of respondents reporting at least one adverse condition
mean_ace = ...

# print
mean_ace
```

In [ ]:

```
grader.check("q12")
```

Does the proportion of respondents who reported experiencing adverse childhood conditions vary by general health?

The cell below computes the porportion separately by general health self-rating. Notice that the depression variable is dropped so that the result doesn't also report the proportion of respondents reporting having been diagnosed with a depressive disorder. Notice also that the proportion of missing values for respondents indicating each general health rating is shown.

In [ ]:

```
# proportions grouped by general health
data.drop(
    columns = 'depression'
).groupby(
    'general_health'
).mean(numeric_only = True)
```

Notice that the row index lists the general health rating out of order. This can be fixed using a `.loc[]` call and the dictionary that was defined for the variable coding.

In [ ]:

```
# same as above, rearranging index
ace_health = data.drop(
    columns = 'depression'
).groupby(
    'general_health'
).mean(
    numeric_only = True
).loc[list(health_codes.values()), :]

# print
ace_health
```

Question 13: Association between smoking status and ACEs

Does the proportion of respondents who reported experiencing adverse childhood conditions vary by smoking status?

Following the example above for computing the proportion of respondents reporting ACEs by general health rating, calculate the proportion of respondents reporting ACEs by smoking status (be sure to arrange the rows in appropriate order of smoking status) and store as `ace_smoking`.

In [ ]:

```
# proportions grouped by smoking status
ace_smoking = data.drop(
    columns = 'depression'
).groupby(
    'smoking'
).mean(
    numeric_only = True
...
```

```

# print
ace_smoking
```

In [ ]:
```
grader.check("q13")
```

Question 14: Association between depression and ACEs

Does the proportion of respondents who reported experiencing adverse childhood conditions vary by smoking status?

Calculate the proportion of respondents reporting ACEs by whether respondents had been diagnosed with a depressive disorder and store as `ace_depr`.

In [ ]:
```
# proportions grouped by having experienced depression
ace_depr = data.groupby(
    'depression'
).mean(
    numeric_only = True
...

# print
ace_depr
```

In [ ]:
```
grader.check("q14")
```

Question 15: Exploring subgroupings

Does the apparent association between general health and ACEs persist after accounting for sex?

Repeat the calculation of the proportion of respondents reporting ACEs by general health rating, but also group by sex. Store the result as `ace_health_sex`.

In [ ]:
```
# group by general health and sex
ace_health_sex = data.drop(
    columns = 'depression'
).groupby(
    ['general_health', 'sex']
...
```

In [ ]:
```
grader.check("q15")
```

The cell below rearranges the table a little for better readability.

In [ ]:
```
# pivot table for better display
ace_health_sex.reset_index().pivot(columns = 'sex', index = 'general_health', value
```

Even after rearrangement, the table in the last question is a little tricky to read (few people like visually scanning tables). This information would be better displayed in a plot. The example below generates a bar chart showing the summaries you calculated in Q2(d), with the proportion on the y axis, the health rating on the x axis, and separate bars for the two sexes.

In [ ]:

```
# coerce indices to columns for plotting
plot_df = ace_health_sex.reset_index()

# specify order of general health categories
genhealth_order = list(health_codes.values())
plot_df.general_health.cat.set_categories(genhealth_order, inplace=True)
plot_df.sort_values(["general_health"], inplace=True)

# plot
alt.Chart(plot_df).mark_bar().encode(
    x = alt.X('general_health',
              sort = ['general_health'],
              title = 'Self-rated general health'),
    y = alt.Y('adverse_conditions',
              title = 'Prop. of respondents reporting ACEs'),
    color = 'sex',
    column = 'sex'
).properties(
    width = 200,
    height = 200
)
```

Question 16: Visualization

Use the example above to plot the proportion of respondents reporting ACEs against smoking status for men and women.

Hint: you only need to modify the example by substituting smoking status for general health.

In [ ]:

```

```

In [ ]:

```
# dataframe of proportions grouped by smoking status
ace_smoking_sex = ...

# coerce indices to columns for plotting
...

# specify order of general health categories
...

# plot
...
```

Communicating results

Here you'll be asked to reflect briefly on your findings.

Question 17: Summary

Is there an observed association between reporting ACEs and general health, smoking status, and depression among survey respondents who answered the ACE questions?

Write a two to three sentence answer to the above question summarizing your findings. State an answer to the question in your first sentence, and then in your second/third sentences describe exactly what you observed in the foregoing descriptive analysis of the BRFSS data. Be precise, but also concise. There is no need to describe any of the data manipulations, survey design, or the like.

Type your answer here, replacing this text.

Question 18: Scope of inference

Recall from the overview documentation all the care that the BRFSS dedicates to collecting a representative sample of the U.S. adult population with phone numbers. Do you think that your findings provide evidence of an association among the general public (not just the individuals survey)? Why or why not? Answer in two sentences.

Type your answer here, replacing this text.

Question 19: Bias

What is a potential source of bias in the survey results, and how might this affect the proportions you've calculated?

Answer in one or two sentences.

Type your answer here, replacing this text.

Comment

Notice that the language 'association' is non-causual: we don't say that ACEs cause (or don't cause) poorer health outcomes. This is intentional, because the BRFSS data are what are known as 'observational' data, *i.e.* not originating from a controlled experiment. There could be unobserved factors that explain the association.

To take a simple example, dog owners live longer, but the reason is simply that dog owners walk more -- so it's the exercise, not the dogs, that cause an increase in longevity. An observational study that doesn't measure exercise would show a positive association between dog ownership and lifespan, but it's a non-causal relationship.

(As an interesting/amusing aside, there is a well known study that established an association between birdkeeping and lung cancer; obviously this is non-causal, yet the study authors recommended that individuals at high risk for cancer avoid 'avian exposure', as they were unsure of the mechanism.)

So there could easily be unobserved factors that account for the observed association in the BRFSS data. We guard against over-interpreting the results by using causally-neutral language.

Submission

1. Save the notebook.
2. Restart the kernel and run all cells. (**CAUTION**: if your notebook is not saved, you will lose your work.)
3. Carefully look through your notebook and verify that all computations execute correctly and all graphics are displayed clearly. You should see **no errors**; if there are any errors, make sure to correct them before you submit the notebook.
4. Download the notebook as an `.ipynb` file. This is your backup copy.
5. Export the notebook as PDF and upload to Gradescope.

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

In [ ]:

```
grader.check_all()
```