In [1]:
```python
# Initialize Otter
import otter
grader = otter.Notebook("hw1-brfss.ipynb")
```

PSTAT 100 Homework 1

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

Background

The Behavioral Risk Factor Surveillance System (https://www.cdc.gov/brfss/index.html) (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.

The raw BRFSS survey data is submitted to the CDC for processing and compilation into a single dataset. In the process, a number of 'derived variables' are calculated based on questionnaire responses and appended to the dataset -- a simple example is weight in kg, derived from respondents' weights reported in pounds.

Take a moment to read about the 2019 survey here (https://www.cdc.gov/brfss/annual_data/annual_2019.html) (follow the link!) and familiarize yourself with the information that is publicly available. This includes data and documentation pertaining to sampling methodology, questionnaires, variable coding, derived variables, and response rates.

Assignment objectives

In this assignment you'll import and subsample the BRFSS 2019 data and perform a simple descriptive analysis exploring association between adverse childhood experiences, health perceptions, tobacco use, and depressive disorders. This is an opportunity to practice critical thinking about data collection, computational skills, and communicating results clearly and correctly.

Critical thinking about data collection. You'll examine the BRFSS data documentation and practice:

- identifying sampled units, variables measured, and the sampling frame;
- assessing study design, data integrity, and scope of inference.

Computation. You'll put into practice your data manipulation skills to tidy up the dataset:

- data import;
- slicing (selecting rows and columns);
- index manipulation (renaming and rearranging);
- value substitution for coded categorical variables (*e.g.*, 1 - 5 -> excellent - poor);
- grouping and aggregation; and
- visualization.

Communication. Finally, you'll practice:

- clear and concise description of data summaries and plots;
- proper interpretation of estimates;
- avoiding causual language when discussing observational data.

0. Data import and assessment

In this part you'll import the 2019 BRFSS data from a compressed CSV file and perform some basic qualitative assessments:

- examine the imported data;
- check your understanding of the data format;
- and investigate the data collection procedure.

Think of these tasks as comprising the '**collect**' and '**acquaint**' steps of our PSTAT 100 lifecycle: gathering data and getting to know it.

Performing these basic checks and gaining background on how the data were generated are essential steps in any analysis. You should make them standard practice in your work. Investigating the data collection procedure is especially important -- it is professionally irresponsible to analyze data without knowing where they came from, and collection protocols (known as a *sampling design*) can have strong implications for whether findings can be replicated and whether they might be biased.

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

In [3]:
```
# 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_vars)


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

Out[3]:

	GENHLTH	ADDEPEV3	ACEDEPRS	ACEDRINK	ACEDRUGS	ACEPRISN	_LLCPWT	_SE
0	3.0	2.0	2.0	2.0	2.0	2.0	135.304080	2.(
1	4.0	2.0	2.0	1.0	2.0	2.0	1454.882220	2.(
2	3.0	2.0	2.0	2.0	2.0	2.0	215.576852	2.(
3	4.0	2.0	NaN	NaN	NaN	NaN	261.282838	2.(
4	2.0	2.0	2.0	2.0	2.0	2.0	535.270103	2.(

Q0 (a). Dimensions

Check the dimensions of the dataset. Set `rows` and `columns` to the respective dimensions.

In [4]:
```
n = brfss.shape
rows = n[0]
columns = n[1]
print(rows, columns)
```

```
418268 10
```

```
In [5]: #print dataset
        brfss
```

```
Out[5]:
```

	GENHLTH	ADDEPEV3	ACEDEPRS	ACEDRINK	ACEDRUGS	ACEPRISN	_LLCPWT
0	3.0	2.0	2.0	2.0	2.0	2.0	135.304080
1	4.0	2.0	2.0	1.0	2.0	2.0	1454.882220
2	3.0	2.0	2.0	2.0	2.0	2.0	215.576852
3	4.0	2.0	NaN	NaN	NaN	NaN	261.282838
4	2.0	2.0	2.0	2.0	2.0	2.0	535.270103
...	...	...	...	...	...	...	...
418263	3.0	2.0	NaN	NaN	NaN	NaN	580.339120
418264	2.0	2.0	NaN	NaN	NaN	NaN	296.172873
418265	2.0	2.0	NaN	NaN	NaN	NaN	807.717193
418266	3.0	2.0	NaN	NaN	NaN	NaN	219.023284
418267	3.0	2.0	NaN	NaN	NaN	NaN	532.437767

418268 rows × 10 columns

Q0 (b). 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. This is an essential step in any analysis -- one can't conduct a useful analysis without properly understanding the entries in a dataset.

Check the number of records (interviews conducted) reported and variables measured for 2019 by reviewing the [surveillance summaries by year (https://www.cdc.gov/brfss/annual_data/all_years/states_data.htm)](https://www.cdc.gov/brfss/annual_data/all_years/states_data.htm) (follow the link!), and then answer the following questions.

- 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

1. The number of rows does in fact match the number of reported records (418268).
2. 10 columns were imported however there are 342 total variables reported in the full dataset and we only selected 10 of them.
3. Each row in the brfss dataframe represents the different reported records
4. Each column in the brfss dataframe represents a differnet variable or attribute for each reported record/observation there are a total.

Q0 (c). Sampling design and data collection

It is essential to devote effort to understanding how data were obtained before taking any further steps, no matter how basic those steps might be. Without adequate background, it is easy to miss potential complications affecting how an analyst should engage with a project. To name a few examples:

- convenience or haphazard sampling (*e.g.*, surveying my neighbors) could entail that patterns in the data are non-generalizable, as they may be difficult or impossible to replicate if the data are collected anew;
- there may be sources of bias inherent in measurements (*e.g.*, failing to properly calibrate lab equipment) that could produce misleading results;
- ethical issues (*e.g.*, data from animal experiments, conflicts of interest, sensitive user information, etc.) may require withdrawal from a project for personal reasons or demand extra caution to protect subject privacy.

These matters are never evident from data itself, and require critical qualitative assessment of data collection procedures.

Skim the overview documentation (https://www.cdc.gov/brfss/annual_data/2019/pdf/overview-2019-508.pdf) for the 2019 BRFSS data. Focus specifically the 'Background' and 'Data Collection' sections, and read between the technical information (don't worry if you don't understand everything in the documentation -- you're not expected to!), and answer the following questions.

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

An interviewer speaks to a randomly selected adult in the household.

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

Adults (18+ years) with an active phone number and residing in a private residence or college housing are eligible to be interviewed.

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

1. Adults who do not have cellular phones or access to a landline cannot appear in the survey.
2. Children cannot appear in the survey.

iv. Who conducts the interviews and how long does the core portion of a typical interview last?

State health personnel or contractors conduct interviews which include a questionaire lasting about 17 minutes and state-added questions adding an additional 5-10 minutes to the intertview time.

v. How would you describe the study population (*i.e.*, all individuals who could possibly be sampled)?

The study population consists of adults in the U.S. with phone numbers.

vi. Does the data contain any identifying information on respondents?

No, as a precaution to protect the respondents, specific variables like sub-state geographic identifiers, detailed race or ethnicity, and being older than 80 years of age in a given year are removed from the dataset.

Now that you have a good understanding of how the data are formatted and how they were collected, you can start engaging with the dataset in an informed way.

For this assignment, you'll work with just a few of the 300+ variables: 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 stored in the cell in which you imported the data.

Q0 (d) -- variable descriptions

With a narrowed set of variables to focus on, a final step in gathering background understanding is to determine clearly the meaning of each variable and how its measurements are recorded in the dataset (*e.g.*, text, numeric/continuous, numeric/categorical, logical, etc.). **This is yet another essential step in any analysis**. 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 (https://www.cdc.gov/brfss/annual_data/2019/pdf/codebook19_llcp-v2-508.HTML) 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`.

Variable name	Description
`GENHLTH`	
`_SEX`	
`_AGEG5YR`	
`ACEPRISN`	
`ACEDRUGS`	
`ACEDRINK`	
`ACEDEPRS`	
`ADDEPEV3`	
`_SMOKER3`	

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.

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.

In [6]:
```
# for reproducibility
np.random.seed(32221)

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

Aside. 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 represent 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 them in drawing the subsample so that we get a representative sample of U.S. adults with phone numbers.

1. Data tidying -- slicing, recoding, renaming

Here you'll **tidy** up the subsample, performing the following steps:

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

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

Q1 (a). Column selection

Use `selected_vars` and slice `samp` to obtain the variables of interest and exclude the others using the `.loc` method. (See lab 1.)

```
In [7]: samp
```

```
Out[7]:
```

	GENHLTH	ADDEPEV3	ACEDEPRS	ACEDRINK	ACEDRUGS	ACEPRISN	_LLCPWT
214514	1.0	1.0	NaN	NaN	NaN	NaN	1930.429631
300840	2.0	2.0	NaN	NaN	NaN	NaN	1561.253858
128960	3.0	2.0	NaN	NaN	NaN	NaN	176.379050
399136	3.0	2.0	1.0	1.0	2.0	2.0	944.718264
132641	2.0	2.0	NaN	NaN	NaN	NaN	311.757606
...	...	...	...	...	...	...	...
309377	2.0	2.0	2.0	2.0	2.0	2.0	3891.341955
339196	3.0	1.0	1.0	1.0	1.0	1.0	1048.084290
204031	2.0	2.0	NaN	NaN	NaN	NaN	296.672012
74131	3.0	2.0	2.0	2.0	1.0	2.0	8187.262900
270459	4.0	2.0	NaN	NaN	NaN	NaN	3441.358422

10000 rows × 10 columns

In [8]:
```
# slice columns of interest
samp_mod1 = samp.loc[:, selected_vars]

# check result
samp_mod1.head()
```

Out[8]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	2.0	1.0	1.0	NaN	NaN	NaN	NaN	
300840	1.0	3.0	2.0	NaN	NaN	NaN	NaN	
128960	2.0	13.0	3.0	NaN	NaN	NaN	NaN	
399136	2.0	1.0	3.0	2.0	2.0	1.0	1.0	
132641	2.0	1.0	2.0	NaN	NaN	NaN	NaN	

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

Out[9]: **q1_a** passed!

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.

In [10]:
```
# proportions of missingness -- uncomment after resolving q1a
samp_mod1.isna().mean()
```

Out[10]:
```
_SEX        0.0000
_AGEG5YR    0.0000
GENHLTH     0.0000
ACEPRISN    0.7541
ACEDRUGS    0.7538
ACEDRINK    0.7537
ACEDEPRS    0.7531
ADDEPEV3    0.0001
_SMOKER3    0.0000
_LLCPWT     0.0000
dtype: float64
```

Recoding categorical variables

Now notice that 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 (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html) for `.replace()` .

In [11]:
```
# 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_mod2 = samp_mod1.replace({'_AGEG5YR': age_codes})

# check result
samp_mod2.head()
```

Out[11]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	2.0	18-24	1.0	NaN	NaN	NaN	NaN	
300840	1.0	30-34	2.0	NaN	NaN	NaN	NaN	
128960	2.0	80+	3.0	NaN	NaN	NaN	NaN	
399136	2.0	18-24	3.0	2.0	2.0	1.0	1.0	
132641	2.0	18-24	2.0	NaN	NaN	NaN	NaN	

Q1 (b). Recoding variables

Following the example immediately above and referring to the 2019 BRFSS codebook (https://www.cdc.gov/brfss/annual_data/2019/pdf/codebook19_llcp-v2-508.HTML), 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`; the second modification (recoding age) was stored as `samp_mod2` and was a function of `samp_mod1`. You'll follow this pattern so that each step of your data manipulations is stored separately, for easy troubleshooting.

i. Recode `_SEX`

Define a new dataframe `samp_mod3` that is the same as `samp_mod2` but with the `_SEX` variable recoded as `M` and `F`. Print the first few rows of the result using `.head()`.

In [12]:
```
# define dictionary
sex_codes = { 1.0:'M', 2.0:'F' }

# recode
samp_mod3 = samp_mod2.replace({'_SEX': sex_codes})

# check
samp_mod3.head()
```

Out[12]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	F	18-24	1.0	NaN	NaN	NaN	NaN	
300840	M	30-34	2.0	NaN	NaN	NaN	NaN	
128960	F	80+	3.0	NaN	NaN	NaN	NaN	
399136	F	18-24	3.0	2.0	2.0	1.0	1.0	
132641	F	18-24	2.0	NaN	NaN	NaN	NaN	

In [13]:
```
grader.check("q1_b_i")
```

Out[13]: **q1_b_i** passed!

ii. Recode `GENHLTH`

Define a new dataframe `samp_mod4` that is the same as `samp_mod3` but with the `GENHLTH` variable recoded as `Excellent` , `Very good` , `Good` , `Fair` , `Poor` , `Unsure` , and `Refused` . Print the first few rows of the result using `.head()` .

In [14]:
```
# define dictionary
health_codes = {
    1.0:'Excellent',
    2.0:'Very good',
    3.0:'Good',
    4.0:'Fair',
    5.0:'Poor',
    7.0:'Unsure',
    9.0:'Refused',
}

# recode
samp_mod4 = samp_mod3.replace({'GENHLTH':health_codes})

# check using head()
samp_mod4.head()
```

Out[14]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	F	18-24	Excellent	NaN	NaN	NaN	NaN	
300840	M	30-34	Very good	NaN	NaN	NaN	NaN	
128960	F	80+	Good	NaN	NaN	NaN	NaN	
399136	F	18-24	Good	2.0	2.0	1.0	1.0	
132641	F	18-24	Very good	NaN	NaN	NaN	NaN	

In [15]:
```
grader.check("q1_b_ii")
```

Out[15]: **q1_b_ii** passed!

iii. Recode `_SMOKER3`

Define a new dataframe `samp_mod5` that is the same as `samp_mod4` but with `_SMOKER3` recoded as `Daily` , `Some days` , `Former` , `Never` , and `Unsure/refused/missing` . Print the first few rows of the result using `.head()` .

In [16]:
```
# dictionary
smoke_codes = {1.0:'Daily',2.0:'Some days',
               3.0:'Former',4.0:'Never', 9.0:'Unsure/refused/missing'}

# recode
samp_mod5 = samp_mod4.replace({'_SMOKER3':smoke_codes})

# check
samp_mod5.head()
```

Out[16]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	F	18-24	Excellent	NaN	NaN	NaN	NaN	
300840	M	30-34	Very good	NaN	NaN	NaN	NaN	
128960	F	80+	Good	NaN	NaN	NaN	NaN	
399136	F	18-24	Good	2.0	2.0	1.0	1.0	
132641	F	18-24	Very good	NaN	NaN	NaN	NaN	

In [17]:
```
grader.check("q1_b_iii")
```

Out[17]: **`q1_b_iii`** passed!

Q1 (c). Value replacement

Now all the variables *except* the adverse childhood experience and depressive disorder question responses are non-numeric. Notice 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_mod6` that is the same as `samp_mod5` 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 [18]:
```
# define dictionary
answer_key = {1.0:'Yes', 2.0:'No', 7.0:'Unsure', 9.0: 'Refused'}

# recode
samp_mod6 = samp_mod5.replace({'ACEPRISN': answer_key,
                               'ACEDRUGS': answer_key,
                               'ACEDRINK': answer_key,
                               'ACEDEPRS': answer_key,
                               'ADDEPEV3': answer_key})

# check using head()
samp_mod6.head()
```

Out[18]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	F	18-24	Excellent	NaN	NaN	NaN	NaN	
300840	M	30-34	Very good	NaN	NaN	NaN	NaN	
128960	F	80+	Good	NaN	NaN	NaN	NaN	
399136	F	18-24	Good	No	No	Yes	Yes	
132641	F	18-24	Very good	NaN	NaN	NaN	NaN	

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

In [19]:
```
samp_mod6.dtypes
```

Out[19]:
```
_SEX          object
_AGEG5YR      object
GENHLTH       object
ACEPRISN      object
ACEDRUGS      object
ACEDRINK      object
ACEDEPRS      object
ADDEPEV3      object
_SMOKER3      object
_LLCPWT      float64
dtype: object
```

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

In [20]:
```
# coerce to categorical
samp_mod7 = samp_mod6.astype('category')

# check new data types
samp_mod7.dtypes
```

Out[20]:
```
_SEX        category
_AGEG5YR    category
GENHLTH     category
ACEPRISN    category
ACEDRUGS    category
ACEDRINK    category
ACEDEPRS    category
ADDEPEV3    category
_SMOKER3    category
_LLCPWT     category
dtype: object
```

Q1 (d). Define ACE indicator variable

Your objective will be to look for associations between adverse childhood experiences and the other variables by calculating the proportion of respondents who reported experiencing ACEs. This task 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.

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_mod8` , and print the first few rows using `.head()` .

```
In [21]: # copy samp_mod7
         samp_mod8 = samp_mod7.copy()

         # ace column positions
         ace_positions = samp_mod8.columns.str.startswith('ACE')

         # ace data
         ace_data = samp_mod8.loc[:,ace_positions]

         # ace yes indicators
         ace_yes = ace_data == 'Yes'

         # number of yesses
         ace_numyes = ace_yes.sum(axis=1)

         # assign new variable
         samp_mod8['adverse_conditions'] = ace_numyes > 0

         # check result using .head()
         samp_mod8.head()
```

Out[21]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	F	18-24	Excellent	NaN	NaN	NaN	NaN	
300840	M	30-34	Very good	NaN	NaN	NaN	NaN	
128960	F	80+	Good	NaN	NaN	NaN	NaN	
399136	F	18-24	Good	No	No	Yes	Yes	
132641	F	18-24	Very good	NaN	NaN	NaN	NaN	

Q1 (e). 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 [22]:
```python
# copy modification 8
samp_mod9 = samp_mod8.copy()

# define missing indicator using loc
samp_mod9.loc[:,'adverse_missing'] = ace_data.isnull().values.any(axis
=1)

# check using head()
samp_mod9.head()
```

Out[22]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
214514	F	18-24	Excellent	NaN	NaN	NaN	NaN	
300840	M	30-34	Very good	NaN	NaN	NaN	NaN	
128960	F	80+	Good	NaN	NaN	NaN	NaN	
399136	F	18-24	Good	No	No	Yes	Yes	
132641	F	18-24	Very good	NaN	NaN	NaN	NaN	

Q1 (f). 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 answering the ACE questions*. Use your indicator variable `adverse_missing` to filter out respondents who were not asked the ACE questions.

In [23]:
```python
# solution
samp_mod10 = samp_mod9[samp_mod9['adverse_missing'] == False]
```

Q1 (g). 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, and store the result as `samp_mod9` . 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 [24]:
```
# copy samp_mod10
samp_mod11 = samp_mod10.copy()

# define new variable using loc
samp_mod11.loc[:,'depression'] = (samp_mod11.loc[:,'ACEDEPRS'] == 'Yes
')

# check using .head()
samp_mod11.head()
```

Out[24]:

	_SEX	_AGEG5YR	GENHLTH	ACEPRISN	ACEDRUGS	ACEDRINK	ACEDEPRS	ADDE
399136	F	18-24	Good	No	No	Yes	Yes	
379021	F	45-49	Good	No	No	No	No	
187304	F	30-34	Excellent	No	Yes	Yes	No	
311584	F	40-44	Excellent	No	No	Yes	No	
403020	F	70-74	Excellent	No	No	Yes	No	

Q1 (h). 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 state, general health, sex, age, and smoking status. Check the pandas documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rename.html) 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.

In [25]:
```
# slice and rename
data = samp_mod11.drop(columns=['ACEPRISN', 'ACEDRUGS', 'ACEDRINK','AC
EDEPRS',
                               'ADDEPEV3','_LLCPWT','adverse_missing'
]
                      ).rename(columns={'_SEX':'sex', '_AGEG5YR':'age'
,'GENHLTH':'general_health',
                                 '_SMOKER3':'smoking'})

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

Out[25]:

	sex	age	general_health	smoking	adverse_conditions	depression
399136	F	18-24	Good	Never	True	True
379021	F	45-49	Good	Never	False	False
187304	F	30-34	Excellent	Daily	True	False
311584	F	40-44	Excellent	Some days	True	False
403020	F	70-74	Excellent	Never	True	False

---

2. 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** and **analyze** whether there is an apparent association between experiencing adverse childhood conditions and self-reported health, smoking status, and depressive disorders.

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.

Q2 (a). 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 [26]: # proportion of respondents reporting at least one adverse condition
         #drop depression column
         mean_ace = data.drop(columns = 'depression').mean()

         # print
         mean_ace
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: Futu
reWarning: Dropping of nuisance columns in DataFrame reductions (wit
h 'numeric_only=None') is deprecated; in a future version this will
raise TypeError.  Select only valid columns before calling the reduc
tion.
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
Out[26]: adverse_conditions    0.366816
         dtype: float64
```

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 [27]: # proportions grouped by general health
         data.drop(
             columns = 'depression'
         ).groupby(
             'general_health'
         ).mean()
```

Out[27]:

	adverse_conditions
general_health	
Excellent	0.322404
Fair	0.428571
Good	0.364084
Poor	0.430657
Refused	NaN
Unsure	0.166667
Very good	0.355828

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 [28]:
```
# same as above, rearranging index
ace_health = data.drop(
    columns = 'depression'
).groupby(
    'general_health'
).mean().loc[list(health_codes.values()), :]

# print
ace_health
```

Out[28]:

	adverse_conditions
general_health	
Excellent	0.322404
Very good	0.355828
Good	0.364084
Fair	0.428571
Poor	0.430657
Unsure	0.166667
Refused	NaN

Q2 (b). 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).

In [29]:
```
# proportions grouped by smoking status
ace_smoking = data.drop(
    columns = 'depression'
).groupby(
    'smoking'
).mean().loc[list(smoke_codes.values()), :]

# print
ace_smoking
```

Out[29]:

smoking	adverse_conditions
Daily	0.576512
Some days	0.453782
Former	0.409020
Never	0.299427
Unsure/refused/missing	0.250000

Q2 (c). 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.

In [30]:
```
# proportions grouped by having experienced depression
ace_depr = data.groupby('depression').mean()

# print
ace_depr
```

Out[30]:

depression	adverse_conditions
False	0.22997
True	1.00000

Q2 (d). 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.

In [31]:
```
# group by general health and sex
ace_health_sex = data.drop(
    columns = 'depression'
).groupby(['general_health','sex']).mean()

# pivot table for better display
ace_health_sex.reset_index(
).pivot(
    index = 'general_health',
    columns = 'sex',
    values = 'adverse_conditions'
).loc[list(health_codes.values()), :]
```

Out[31]:

sex	F	M
general_health		
Excellent	0.327485	0.317949
Very good	0.356132	0.355499
Good	0.417266	0.308081
Fair	0.488636	0.356164
Poor	0.443038	0.413793
Unsure	0.000000	0.333333
Refused	NaN	NaN

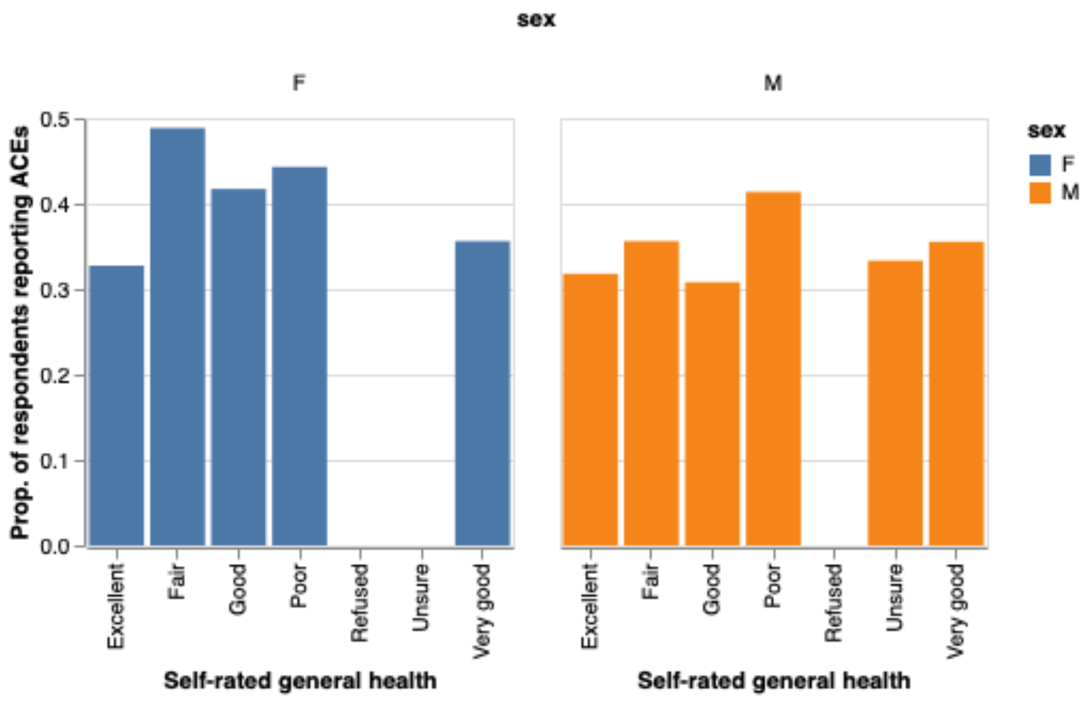
The table in the last question is a little tricky to read. This information would be better displayed in a plot. The example below generates a bar chart showing the summaries you calculated in Q2(d), with the proportion on the y axis, the health rating on the x axis, and separate bars for the two sexes.

```
In [32]: # coerce indices to columns for plotting
plot_df = ace_health_sex.reset_index()

# specify order of general health categories
genhealth_order = pd.CategoricalDtype(list(health_codes.values()), ord
ered = True)
plot_df['general_health'] = plot_df.general_health.astype(genhealth_or
der)

# 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
)
```

Out[32]:



Q2 (e). 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.

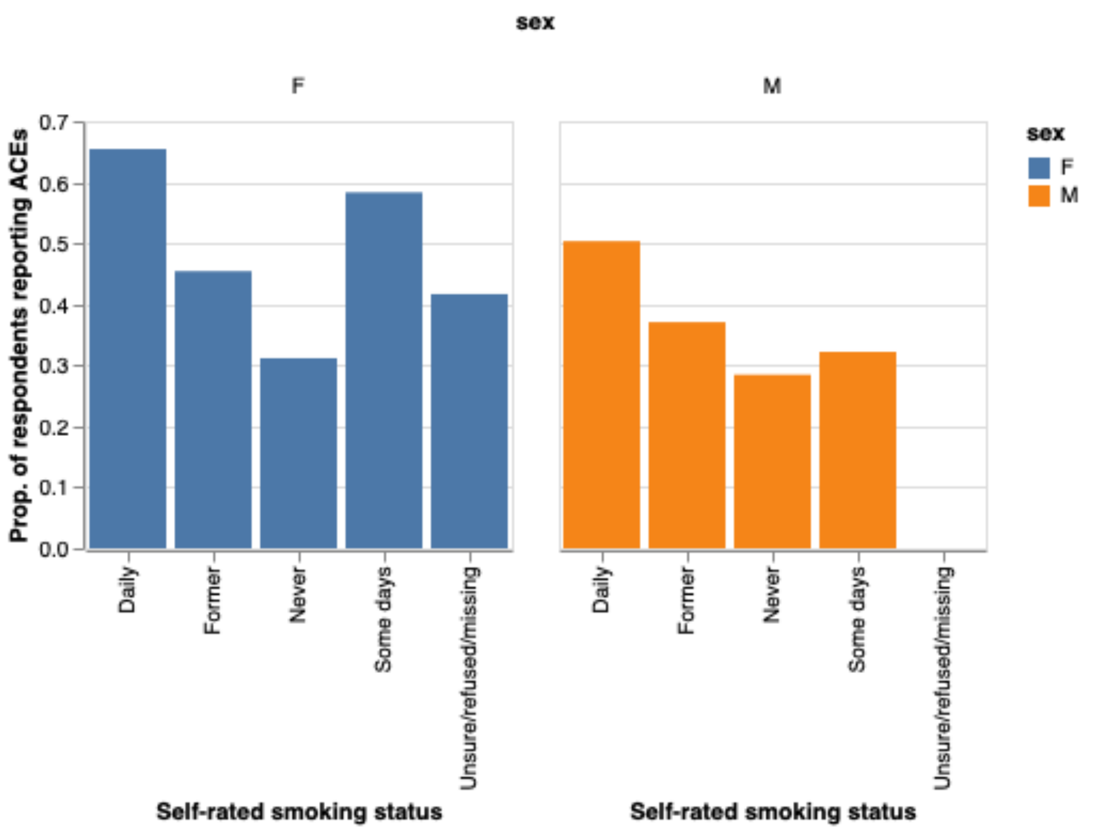
```
In [33]: # plot codes go here
ace_smoking_sex = data.drop(
    columns = 'depression'
).groupby(['smoking','sex']).mean()

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

# specify order of general health categories
smoking_order = pd.CategoricalDtype(list(smoke_codes.values()), ordere
d = True)
plot_df['smoking'] = plot_df.smoking.astype(smoking_order)

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

```
Out[33]:
```



3. Communicating results

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

Q3 (a). 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.

Answer

From our analysis, we can see there is an observed association between reporting ACEs and general health, smoking status, and depression among survey respondents who answered the ACE questions. Those who reported fair or poor health, smoked daily, and/or were diagnosed with depression showed higher portions of reported ACE's from respondents. Furthermore, women were more susceptible reporting ACE's in comparison to male respondents.

Q3 (b). Generalization

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.

Answer

I do think that these findings provide evidence of an association among the general public (not just the individuals survey) because the dataframe is large. The BRFSS collected a well representative sample of the U.S. adult population who have phone numbers. It is also very seldom for an adult not to have access to a phone giving randomization.

Q3 (c). 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.

Answer

Potential sources of bias include hesitation or non-responses from adults not being comfortable enough answering questions during the interview about their smoking habits, general health, or depression.

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.

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 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. Select *File* > *Download as* > *Notebook*
5. Submit to Gradescope

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

In [34]:
```
grader.check_all()
```

Out[34]:
```
q0_a results: All test cases passed!

q1_a results: All test cases passed!

q1_b_i results: All test cases passed!

q1_b_ii results: All test cases passed!

q1_b_iii results: All test cases passed!
```