In [ ]:
```
# Initialize Otter
import otter
grader = otter.Notebook("lab03.ipynb")
```

In [1]:
```
# Run this cell to load all required Python libraries
exec(open("./utils.py").read())
%matplotlib inline
```

1 Lab 3 – Data Visualization

1.1 Data 6, Summer 2022

So far, we have discussed methods to interpret the data, but what if we want to present our data in a visual format? In this lab, you'll learn several important table methods for producing data visualizations. Visualizations are some of the most powerful tools in data science; they're helpful for showing data to people who don't necessarily have a background in data science, and allow data scientists like yourselves to help others understand the data in a more intuitive way.

In Lecture 8, we talked about methods we could use to visualize one variable, namely the `barh` and `hist` methods. We added the `scatter` and `plot` methods in Lecture 9. These methods allow us to visualize two or more variables at once, which can open up more patterns in the data and can further improve your ability to visualize data for people who do not necessarily understand data science.

As data scientists it is not only our job to be able to use the visualization methods we know, but it is also our job to know *when* to use which methods. As we build our toolkit of visualization techniques going forward, it's important to understand the advantages and disadvantages of each visualization type.

We will be working with the same `brfss` dataset as we did in the previous lab, so we will load that in to begin looking at the new methods.

In [2]:
```
brfss = Table.read_table("data/brfss.csv")
brfss.show(5)
```

```
<IPython.core.display.HTML object>
```

2 The barh method

The `barh` (horizontal bar chart) method is used to visualize **categorical** variable values. Categorical variables are non-numbers, like names and qualities (Color, State Names, etc.). As we saw in lecture, categorical variables come in 2 different types: *ordinal* and *nominal*. Refer to the Lecture 8 Slides to see the difference between the two types.

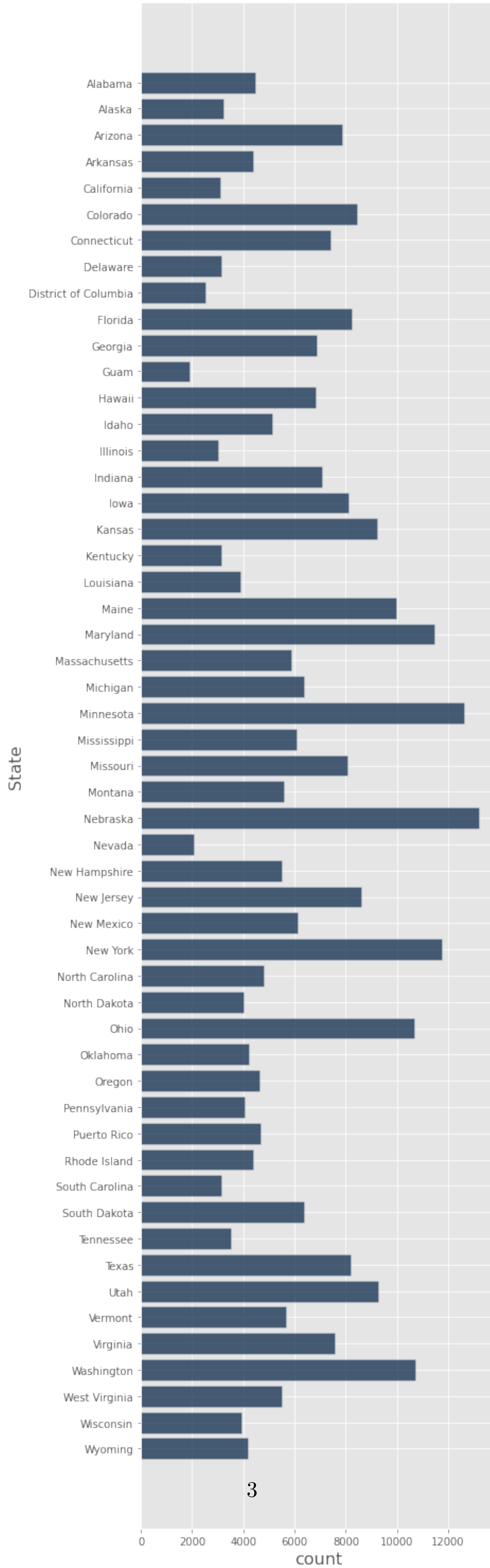
The `barh` method takes in 1 mandatory argument, which is the **name of the column** you want on the left (vertical) axis of your `barh` plot. There are also optional arguments that have to do with plotting – you'll see examples of those in this lab and in the homework. The remaining optional arguments in the `datascience` documentation linked above can also be used, feel free to try out some of the others on your own!

To use the `barh` method properly, we first need to select the columns we want to see in the graph. We should not call `barh` directly on a large `Table` because without specifying a column, we get a bar graph for every single instance of every single variable, which you can imagine results in a lot of bar graphs.

In [3]:
```
# Just run this cell to load a table with State Counts
states = Table.read_table("data/state_counts.csv")
states
```

Out[3]:
```
State                | count
Alabama              | 4477
Alaska               | 3220
Arizona              | 7846
Arkansas             | 4374
California           | 3083
Colorado             | 8454
Connecticut          | 7409
Delaware             | 3136
District of Columbia | 2533
Florida              | 8237
… (43 rows omitted)
```

In [4]:
```
# Since the `states` table only has two columns, we can plot it with barh
states.barh("State")
```



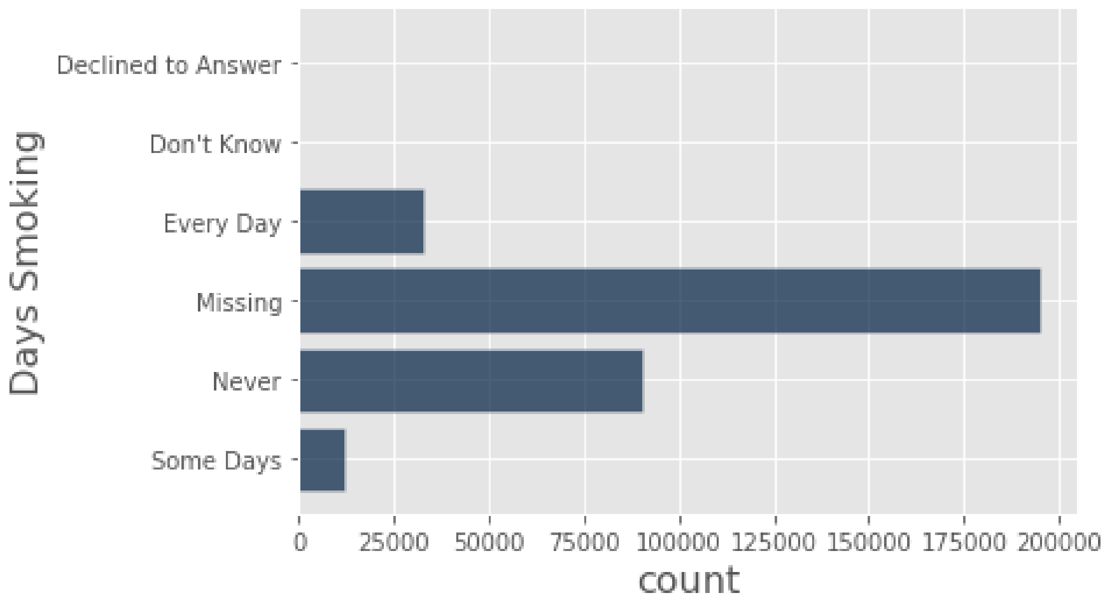
State
count
0
2000
4000
6000
8000
10000
12000
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming

Notice that each value in the “State” column is plotted with a bar with length corresponding to its count.

Question 1: Plot a horizontal bar chart that shows the counts of each category from the `"Days Smoking"` column of the `brfss` table.

Hint: Use the `smoking_counts` table.

In [5]:
```
smoking_counts = Table.read_table("data/smoking_counts.csv")
smoking_counts.barh("Days Smoking") # SOLUTION
```



2.0.1 Multiple Columns

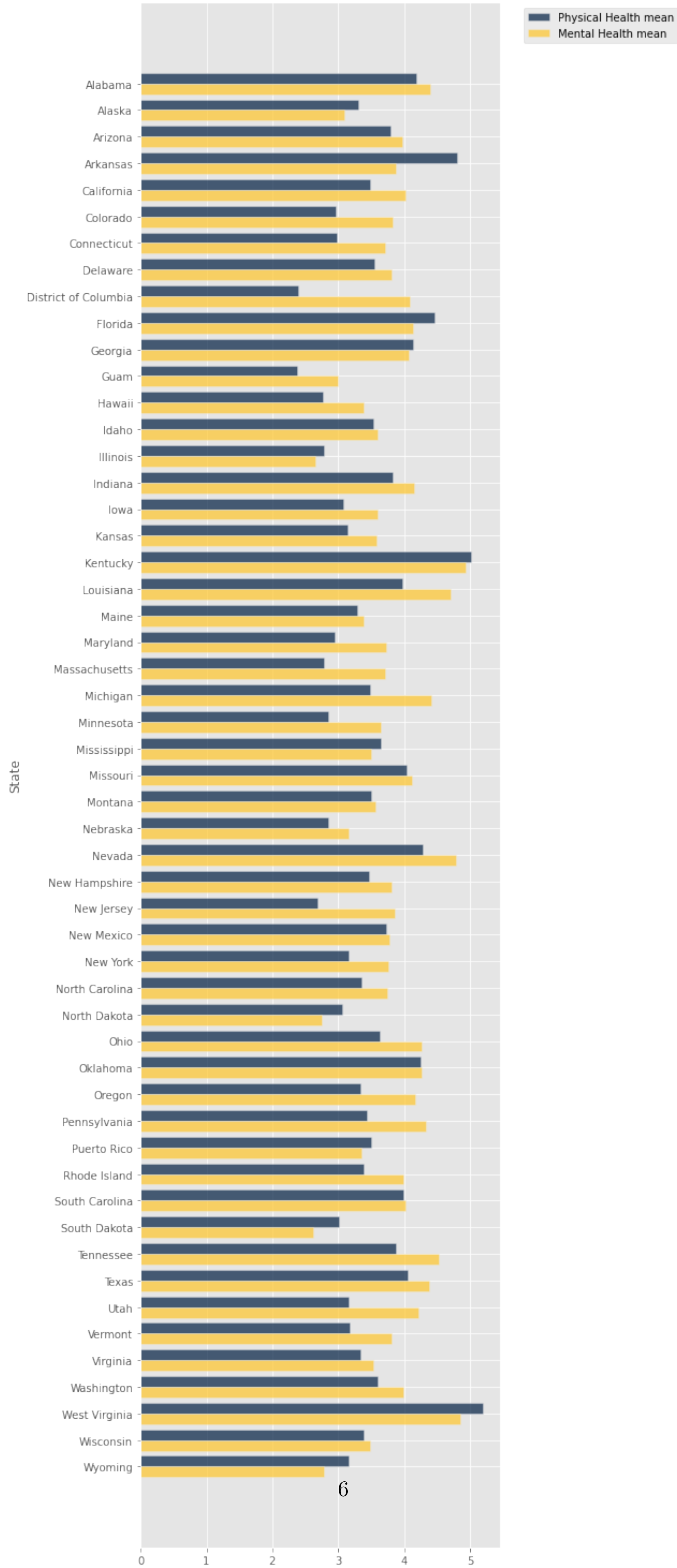
We can also use `barh` to see multiple statistics at once. Let’s use the `barh` method to see the average number of both *poor mental health* and *poor physical health* days. We’ll be using the following columns: 1. `"Physical Health"`: Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good? 2. `"Mental Health"`: Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good?

Run the following cell to show an example of how to create an *overlaid bar chart* with two statistics.

In [6]:
```
state_averages = Table.read_table("data/state_averages.csv")
state_averages
```

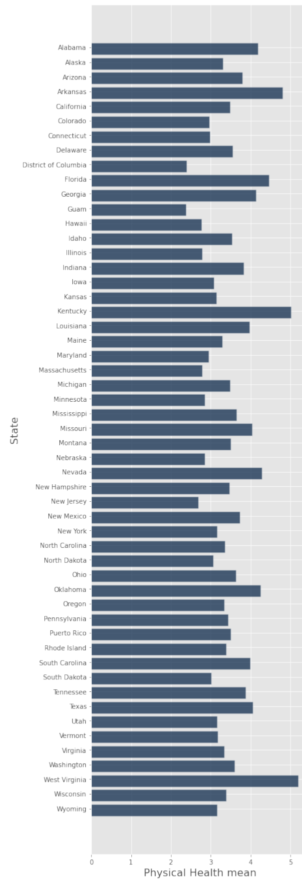
Out[6]:
```
State                | Physical Health mean | Mental Health mean
Alabama              | 4.18316              | 4.40138
Alaska               | 3.3059               | 3.09752
Arizona              | 3.80155              | 3.97515
Arkansas             | 4.80155              | 3.8802
California           | 3.49335              | 4.02141
Colorado             | 2.96747              | 3.83191
Connecticut          | 2.97962              | 3.71778
Delaware             | 3.55038              | 3.81601
District of Columbia | 2.39874              | 4.08843
Florida              | 4.46534              | 4.14338
… (43 rows omitted)
```

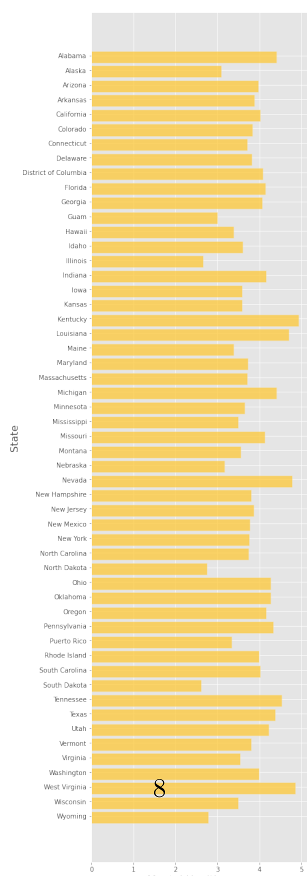
In [7]:
```
# We must group first to get our desired columns, then we can call barh
state_averages.barh("State", overlay=True)
```


Physical Health mean
Mental Health mean
State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
0
1
2
3
4
5

If we want different visualizations for each variable, we can set the optional `overlay` argument to `False`. The default value of `overlay` is `True`, so if you don't give it a value, you will get a plot with all the included variables at once.

```
In [8]: state_averages.barh("State", overlay=False)
```


State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
0
1
2
3
4
5
Physical Health mean


State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Guam
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Puerto Rico
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
0
1
2
3
4
5
Mental Health mean

That way we can choose if we want to have one plot with all our information or a new plot for each piece of information!

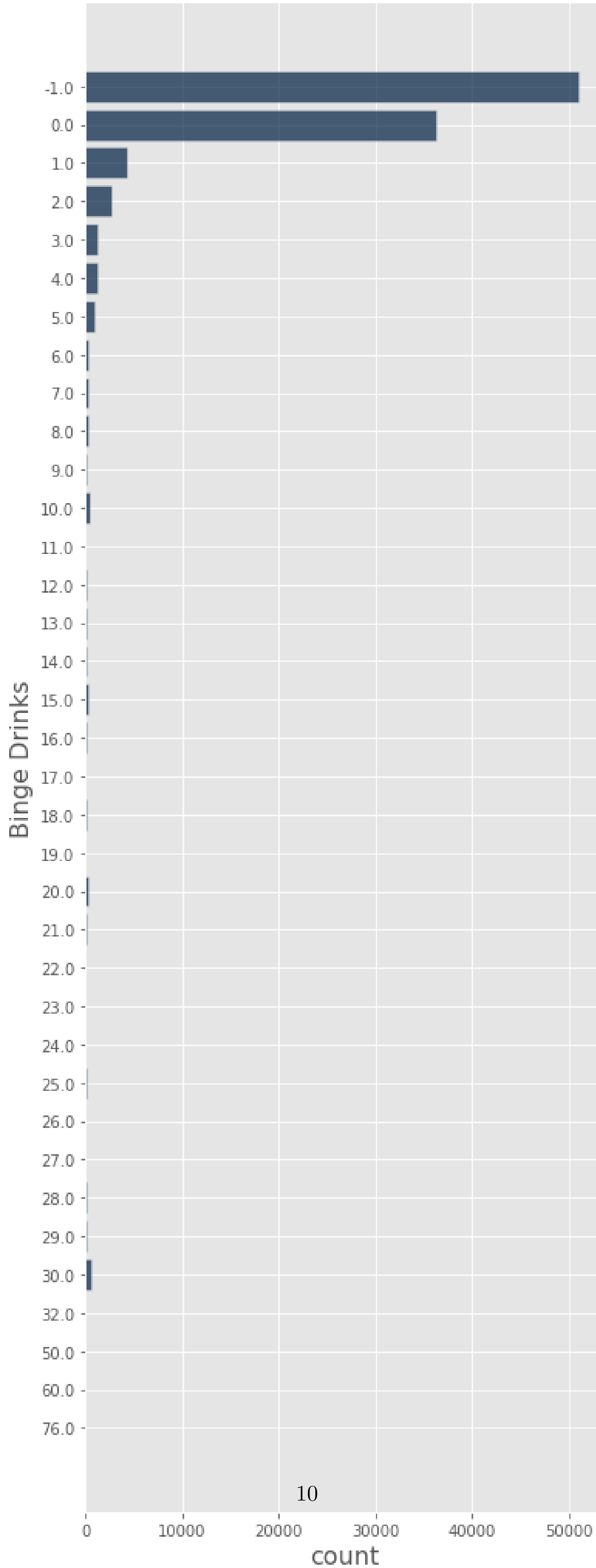
In this case, do we prefer an overlaid plot or two separate plots? Can you think of a case where we might want to have two separate plots instead of one overlaid plot? (Hint: think about the units for both variables — are they the same or different?)

Discuss with the people around you and check in with your GSI to confirm.

2.0.2 Where `barh` fails

The `barh` method works well on categorical variables, but what if we have a **numerical** variable that we want to see the distribution in one particular state? Let's see what happens if we try to use `barh` on a numerical variable (`"Binge Drinking"`) instead of a categorical variable:

```
In [9]: # Just run this cell -- don't worry about this `group` method
        brfss.group("Binge Drinks").barh("Binge Drinks")
```


Binge Drinks
-1.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
21.0
22.0
23.0
24.0
25.0
26.0
27.0
28.0
29.0
30.0
32.0
50.0
60.0
76.0
0
10000
20000
30000
40000
50000
count

As you can see, this bar plot is not particularly helpful. There are many categories that seem to not have any corresponding bar. Yet, that isn't the case! Seeing the breakdown of `"Binge Drinks"` does not provide us with any useful information, and it is also difficult to read or understand. Instead, for numerical variables, we have another visualization method that helps us visualize a numerical variable's distribution…

3 The hist method

The `hist` method allows us to see the distribution of a numerical variable. Categorical variables should be visualized using `barh`, and numerical variables should be visualized using `hist`.

The `hist` method takes in 1 mandatory argument and has several optional arguments (as is the case with `barh`, there are many other optional arguments, but here are just a few of them). For this lab, we'll set **`density` to be `False`**.

Argument	**Description**	**Type**	**Mandatory?**
`column`	Column name whose values you want on the x-axis of your plot	Column name (string)	Yes
`density`	If `True`, then the resulting plot will be displayed not on the count of a value, but on the density of that value in the Table	boolean	No
`group`	Similar to the Table method `group`, groups rows by this label before plotting	Column name (string)	No
`overlay`	When `False`, make a new plot for each eligible statistic in the Table	boolean	No
`bins`	A NumPy array of bin boundaries you want your histogram to gather data into	array	No
`unit`	A name for the units of the plotted column	Column name (string)	No

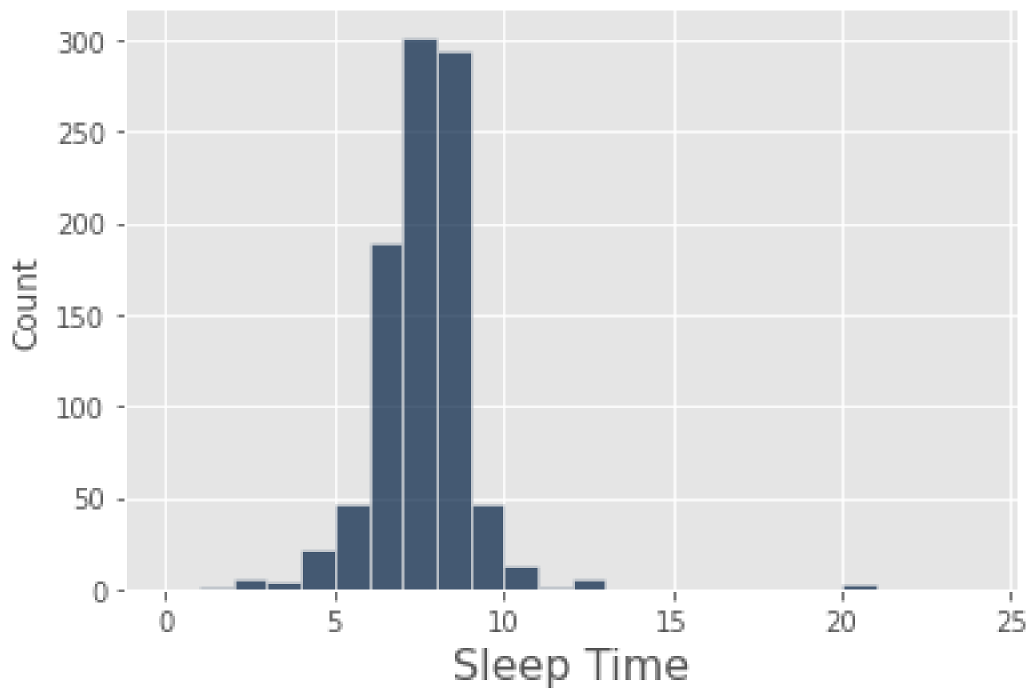
Again, in all cases, `density` should be set to `False`

Keep in mind the same plotting optional arguments mentioned in the `barh` introduction.

Let's take a look at the distribution of exercise sessions in different states to see how the `hist` method helps visualize numerical variables, first starting with our favorite state, California. We'll use the `sleep_no_negatives` table to exclude missing values (-1's).

```
In [10]: sleep_no_negatives = brfss.where("Sleep Time", are.not_equal_to(-1))
```

```
In [11]: # This plot shows the distribution of sleep time for Californians
         my_bins = np.arange(0, 25, 1)
         california = sleep_no_negatives.where("State", "California")
         california.hist("Sleep Time", density = False, bins=my_bins)
```

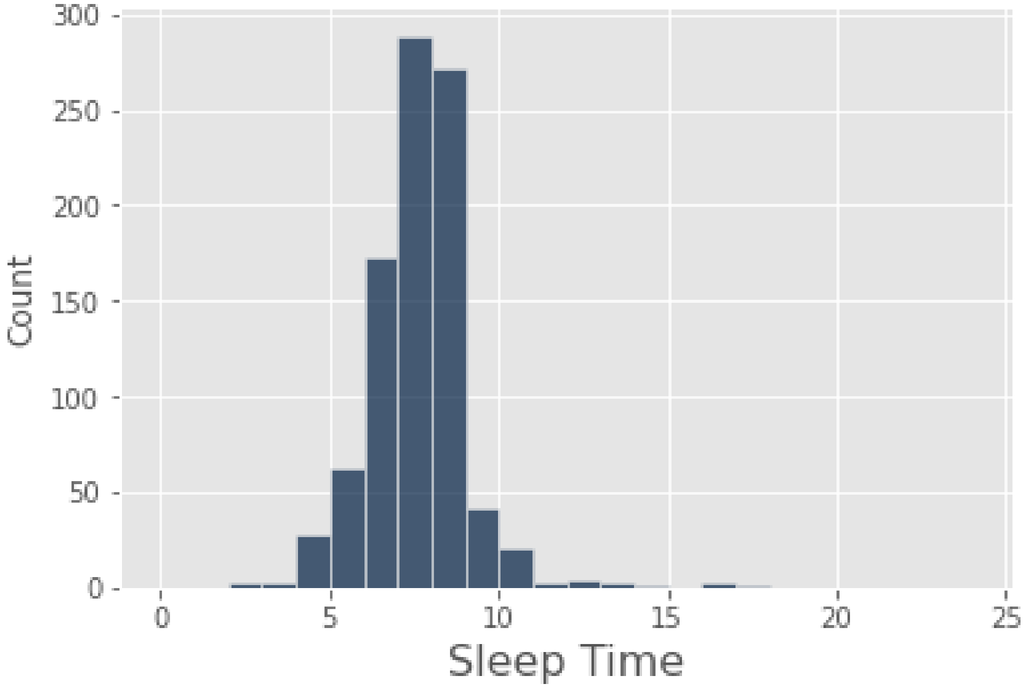


This shows us that people living in California usually tend to sleep between 7 to 8 hours a night, but there are many people who sleep more hours (10+) or few hours (less than 5). Let's see how that compares to sleep time in another state, Illinois:

Question 2: Fill in the following **code cell** to produce a histogram representing the ***distribution of sleep time*** for respondents from the state of Illinois.

Note: Set the optional `bins` argument of the `hist` method to `my_bins`. We've provided this variable for you.

```
In [12]: # This plot shows the distribution of sleep time for Illinois residents
         my_bins = np.arange(0, 25, 1)
         il = sleep_no_negatives.where("State", "Illinois") # SOLUTION
         il.hist("Sleep Time", density = False, bins=my_bins) # SOLUTION
```



3.0.1 California vs. Illinois

We can use `hist` on a `Table` with just rows for these two states and use the optional `group` argument.

Note: You'll see how `are.contained_in` works with the `where` method next week. For now, think of it as finding any rows corresponding to *either* `"California"` or `"Illinois"`.

```
In [13]: # Just run this cell to load the `il_ca` table
         il_ca = sleep_no_negatives.where("State", are.contained_in(["California", "Illinois"]))
         il_ca.show(5)
```

```
<IPython.core.display.HTML object>
```

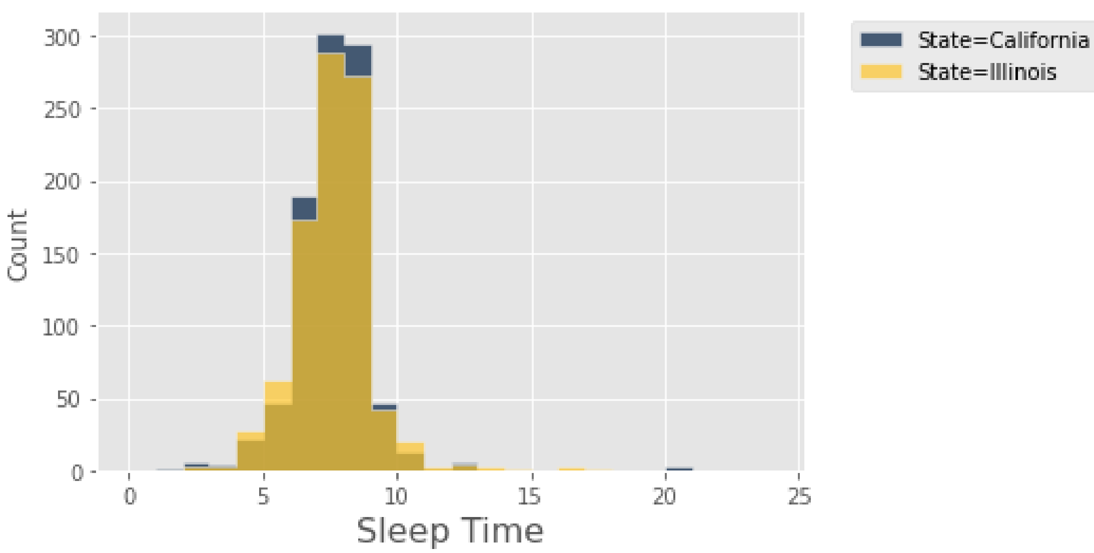
Question 3: Now that we've created our `il_ca` table, fill in the following **code cell** to produce a histogram representing the ***distribution of sleep time*** for *both* California and Illinois. You'll first need to `select` the necessary columns from `il_ca` then fill in the appropriate call to the `hist` method.

Hint: Take a look at the optional `group` argument from the description above.

Note: Set the optional `bins` argument of the `hist` method to `my_bins`. We've provided this variable for you.

```
In [14]: # This plot shows the distribution of sleep time for people from California AND New York
         my_bins = np.arange(0, 25, 1)
         ...
         # BEGIN SOLUTION NO PROMPT
         il_ca.select("State", "Sleep Time").hist("Sleep Time",
                                                  group = "State",
                                                  density = False,
                                                  bins=my_bins
                                                  )

         # END SOLUTION
```



It appears that sleep time in California is a very similar, on average, to the sleep time in California. The plot above shows the New York `Sleep Time` to be almost exactly on top of the Illinois `Sleep Time`. Let's see if we can use a table query to figure out the same information:

```
In [15]: print(f"California average:\t{np.mean(california.column('Sleep Time'))}")
         print(f"Illinois average:\t{np.mean(il.column('Sleep Time'))}")
```

```
California average:        7.095596133190118
Illinois average:        7.113812154696133
```

As we can see, the plot we made appeared to suggest that average amount of sleep should be very similar between California and Illinois, and the table operations reflected that! This is a benefit of visualization, that information can be learned about the dataset with just visual observation. It is always beneficial to back your claims about data with concrete facts about the dataset, but visualizations can help abstract away some of the confusion of looking at raw data so that non-data-scientists can better understand what is going on.

Question 4 (*Discussion*): Now, think about what would happen if you chose two states with **very different counts**, why would it be more difficult to compare them with histograms?

Once you've discussed with someone around you or a GSI, proceed with the code cells below to confirm your answers. We'll look to compare **Texas** and **Delaware**.

Type your answer here, replacing this text.

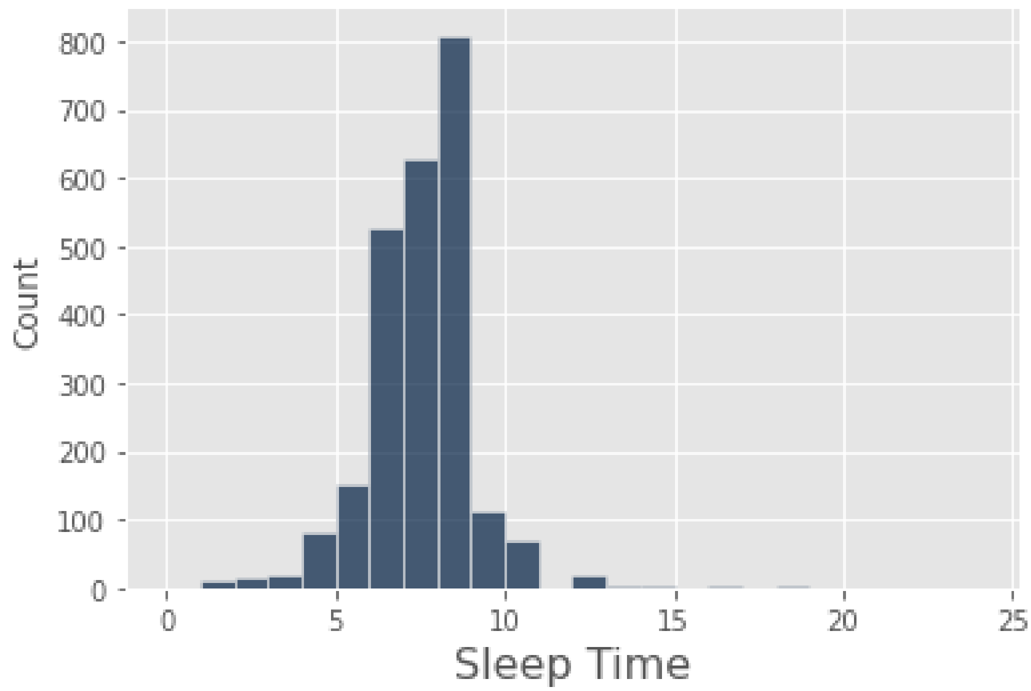
SOLUTION: Since we set `density=False` when creating our histograms, the state with more data points will be much taller than the histogram for the state with very few data points. Since the two states are not on the same scale, it is difficult to compare them.

```
In [16]: # Just run this cell
         texas = sleep_no_negatives.where("State", "Texas")
         delaware = sleep_no_negatives.where("State", "Delaware")
         print(f"Texans in `cleaned_exercise_mental_health` dataset: {texas.num_rows}")
         print(f"Delawareans in `cleaned_exercise_mental_health` dataset: {delaware.num_rows}")
```

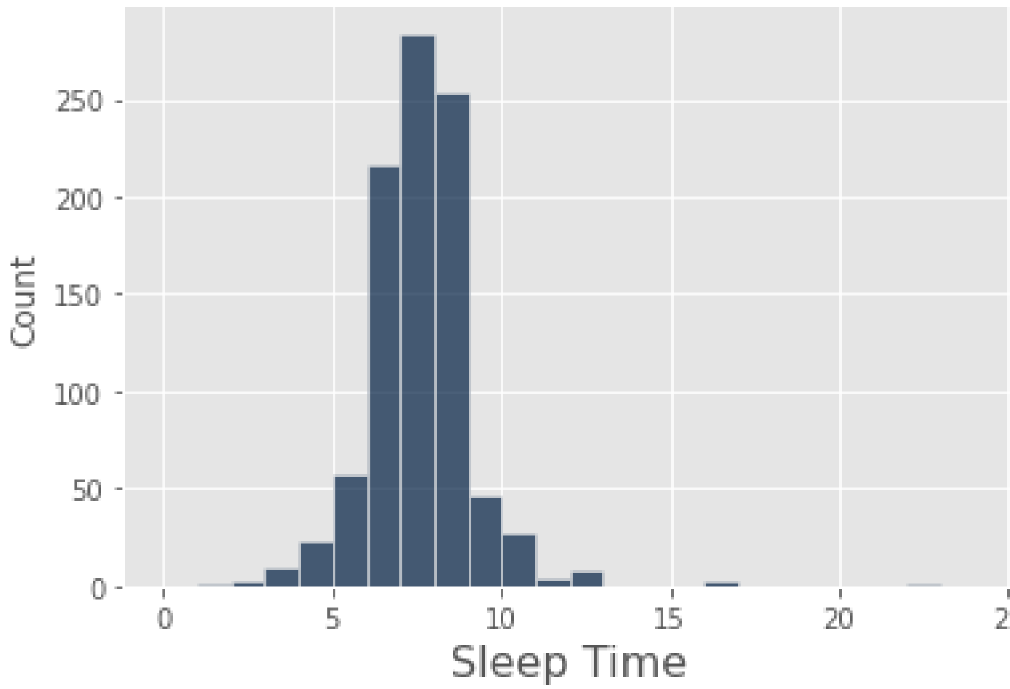
```
Texans in `cleaned_exercise_mental_health` dataset: 2458
Delawareans in `cleaned_exercise_mental_health` dataset: 938
```

Each individual plot looks fine:

```
In [17]: # This plot shows the distribution of sleep times for Texas respondents
         my_bins = np.arange(0, 25, 1)
         texas.hist("Sleep Time", density = False, bins=my_bins)
```




```
In [18]: # This plot shows the distribution of sleep times for Delaware respondents
         my_bins = np.arange(0, 25, 1)
         delaware.hist("Sleep Time", density = False, bins=my_bins)
```



Take a look at the y-axis on both of these plots. What do you think will happen when we try to plot them on the same graph?

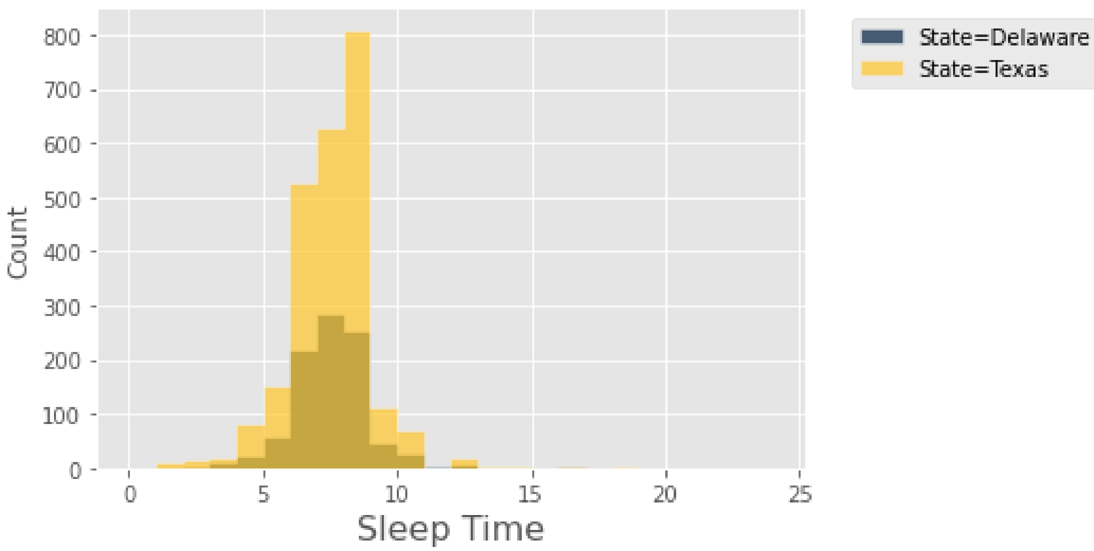
```
In [19]: # Just run this cell
         texas_delaware = sleep_no_negatives.where("State", are.contained_in(["Texas", "Delaware"]))
         texas_delaware.show(5)
```

```
<IPython.core.display.HTML object>
```

Question 5: Using the code in **Question 3** as reference, produce a histogram showing the distribution of sleep time for respondents from *Delaware* and *Texas*. What do you notice about this plot?

```
In [20]: # This plot shows the distribution of the number of times people from Delaware and Texas exerc
         my_bins = np.arange(0, 25, 1)
         ...
         # BEGIN SOLUTION NO PROMPT
         texas_delaware.select("State", "Sleep Time").hist("Sleep Time",
                                                           group = "State",
                                                           density = False,
                                                           bins=my_bins
```

```
)
# END SOLUTION
```



As you can see, there is so much more Texas data than Delaware data that we can hardly make comparisons between the two. Trying to figure out information from this plot is very difficult, so we would either have to use another type of visualization or change the perspective of this plot to be able to learn from it.

4 The scatter method

As we mentioned, visualizing two variables can show us patterns in the data that can help us learn new information. The `scatter` method allows us to see the relationship between two numerical variables in our data using a **scatter plot**. The first provided column name goes along the x-axis and the second goes along the y-axis.

Let's take a look at the relationship between **Physical Health** and **Alcohol Consumption**. For reference, here are the following questions from the original BRFSS Survey that correspond to our `"Physical Health"` and `"Binge Drinks"` columns.

> **Physical Health:** Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good?

> **Binge Drinks**: Considering all types of alcoholic beverages, how many times during the past 30 days did you have 5 or more drinks for men or 4 or more drinks for women on an occasion?

4.0.1 Housekeeping

Question 6: As was the case with our previous visualizations lab, we know that the missing numerical values are encoded as `-1`s. Create a new table called `scatter_cleaned` which contains every row from the original `brfss` table that *does not* contain a `-1` in either the `"Physical Health"` column or the `"Binge Drinks"` column.

Hint: If you're having trouble with the code, feel free to reference the `barh` section of this lab.

```
In [21]: scatter_cleaned = brfss.where("Physical Health", are.not_equal_to(-1)).where("Binge Drinks", a
         scatter_cleaned
```

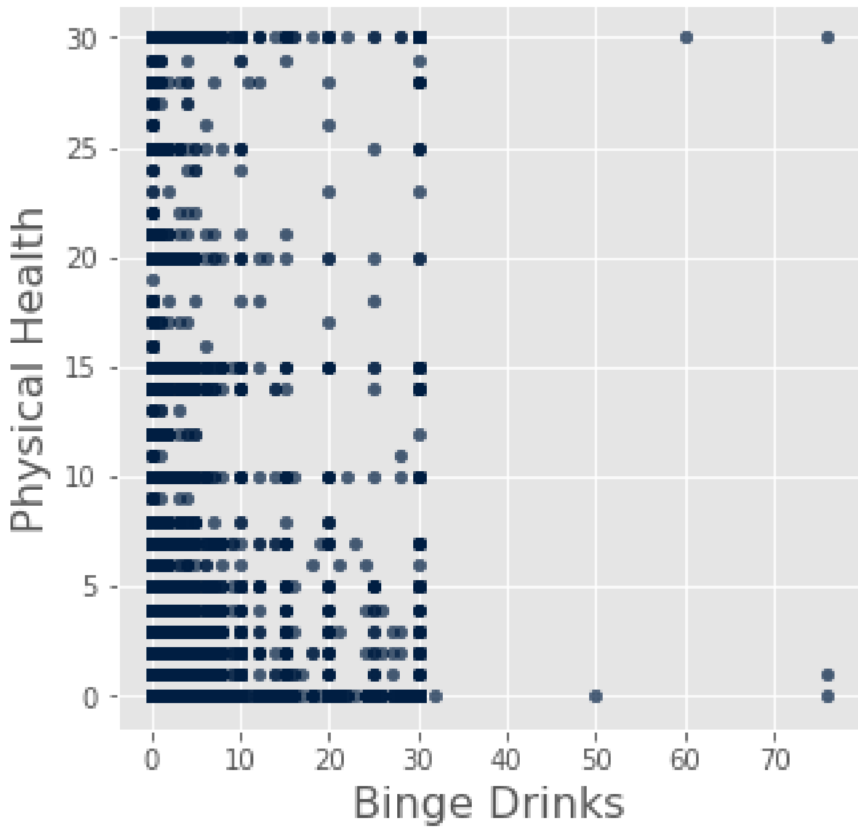
```
Out[21]: State         | Day  | Month | Year | Cell Phone | College Housing (Cell) | College Housing (La
         Nebraska      | 2    | 2     | 2020 | Yes        | Missing                | Missing
         Utah          | 2    | 4     | 2020 | Yes        | Missing                | Missing
         Florida       | 0    | 12    | 2020 | Yes        | Missing                | Missing
         Massachusetts | 0    | 8     | 2020 | Missing    | Missing                | Missing
         Utah          | 3    | 12    | 2020 | Yes        | Missing                | Missing
         New Jersey    | 2    | 10    | 2020 | Yes        | Missing                | Missing
         New Hampshire | 0    | 3     | 2020 | Missing    | Missing                | Missing
         Missouri      | 3    | 10    | 2020 | Missing    | Missing                | Missing
         Oklahoma      | 0    | 7     | 2020 | Yes        | Missing                | Missing
         Connecticut   | 0    | 12    | 2020 | Yes        | Missing                | Missing
         … (48393 rows omitted)
```

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

4.0.2 Producing Scatter Plots

Now, we can call `scatter` on the `scatter_cleaned` table. Run the following cell to do so.

```
In [24]: scatter_cleaned.scatter("Binge Drinks", "Physical Health")
```



Just like that, you've produced your first scatter plot! It looks a little messy, however. Oftentimes scatter plots can suffer from what's known as **overplotting**: when many data points fall on top of each other, creating a blob of data. When data is *overplot*, it's often difficult to see the individual data points on the scatter plot.

To fix this, we attempt to focus in on a smaller subset of the data. In this case, we'll look at points in which `"Binge Drinks"` falls between 0 and 30 days and the `"Physical Health"` column falls between 0 and 30 days.

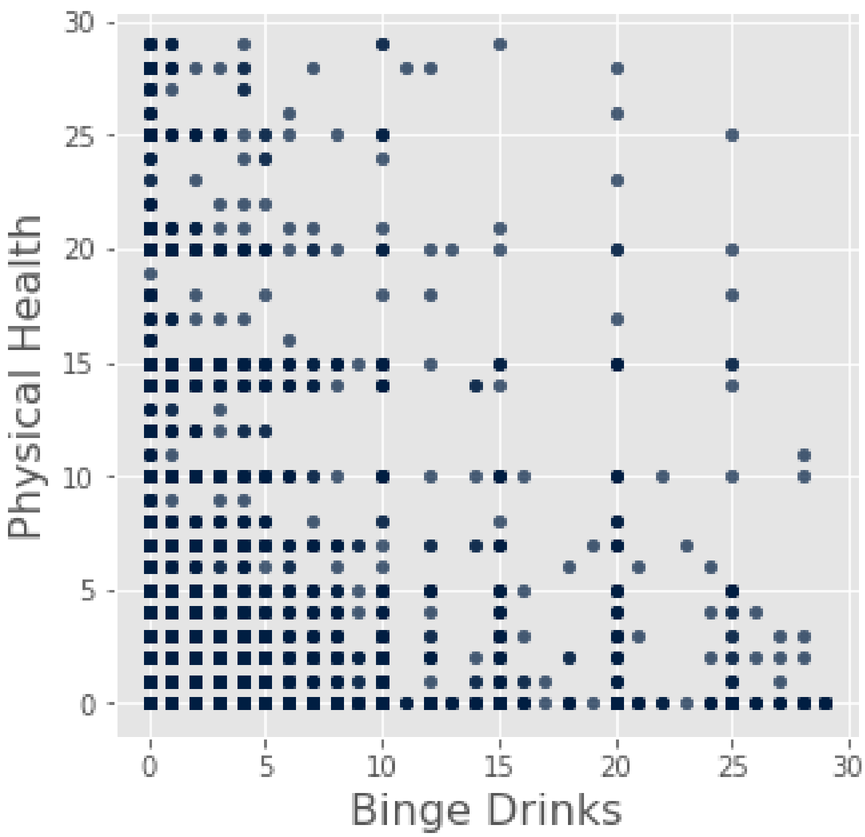
In [25]:
```
# Create a smaller subset of data
scatter_reduced = scatter_cleaned.where("Binge Drinks",
                                        are.below(30)).where("Physical Health", are.below(30))
scatter_reduced
```

Out[25]:
```
State          | Day   | Month | Year | Cell Phone | College Housing (Cell) | College Housing (La
Nebraska       | 2     | 2     | 2020 | Yes        | Missing                | Missing
Utah           | 2     | 4     | 2020 | Yes        | Missing                | Missing
Florida        | 0     | 12    | 2020 | Yes        | Missing                | Missing
```

```
Massachusetts | 0     | 8     | 2020 | Missing    | Missing               | Missing
Utah          | 3     | 12    | 2020 | Yes        | Missing               | Missing
New Jersey    | 2     | 10    | 2020 | Yes        | Missing               | Missing
New Hampshire | 0     | 3     | 2020 | Missing    | Missing               | Missing
Missouri      | 3     | 10    | 2020 | Missing    | Missing               | Missing
Oklahoma      | 0     | 7     | 2020 | Yes        | Missing               | Missing
Connecticut   | 0     | 12    | 2020 | Yes        | Missing               | Missing
… (45990 rows omitted)
```

Question 7: Using the `scatter_reduced` table, produce a scatterplot that plots “`Binge Drinks"` on the x-axis and `"Exercise Sessions (Past Month)"` on the y-axis. The code should be very similar to the previous scatter plot.

```
In [26]: scatter_reduced.scatter("Binge Drinks", "Physical Health") # SOLUTION
```



That looks a little better! There is still a cluster of data points in the bottom left corner, but a clear relationship can be seen between the two variables.

Question 8 (*Discussion*): What relationship between binge drinking and number of days with poor physical health does the above scatter plot reveal? Discuss with someone around you and check in with your GSI once you've agreed on an answer.

Type your answer here, replacing this text.

SOLUTION: As the number of days spent binge drinking increases, the more the number of *good physical health days* decreases; there is an inverse relationship.

4.0.3 The `group` and `labels` Optional Arguments

The `scatter` method also allows you to specify specific groups or labels for each data point using the `group` or `labels` keyword arguments.

Say we wanted to investigate the relationship between an individual's **number of children** and their **mental health**. The corresponding questions from the original BRFSS survey were as follows: > **Mental Health**: Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good?

> **Children**: How many children less than 18 years of age live in your household?

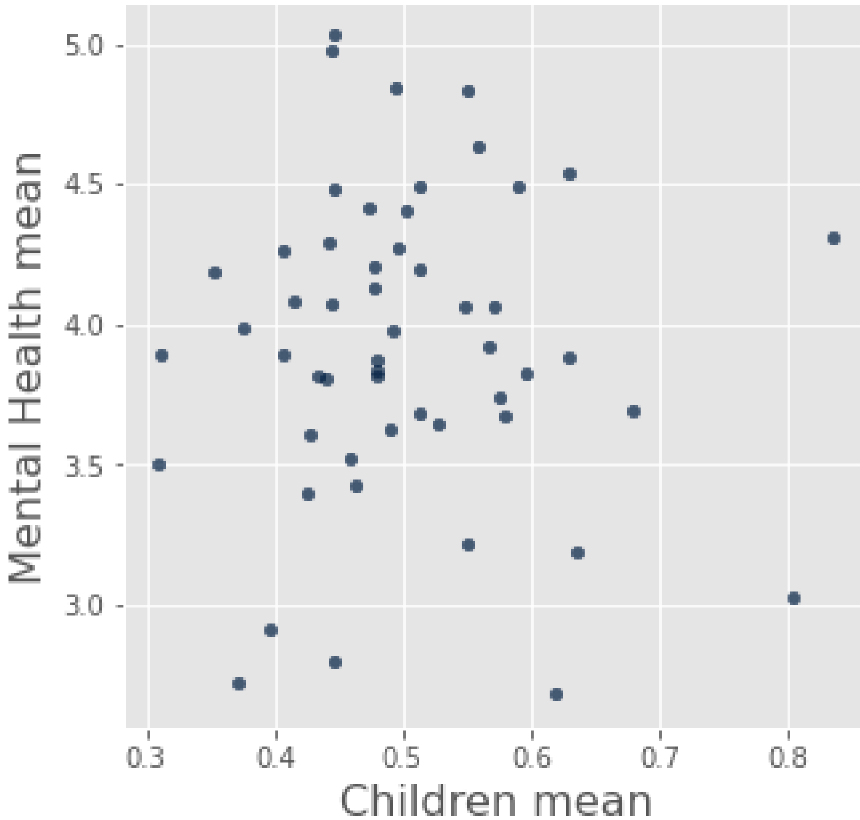
Question 9: In order to take advantage of the optional arguments, let's first load an additional table from the `"states_scatter.csv"` file. We'll provide the code for this.

Then, using the `states` table, produce a scatter plot that plots the average children against the average number of poor mental health days.

```
In [27]: states = Table.read_table("data/states_scatter.csv")
         states
```

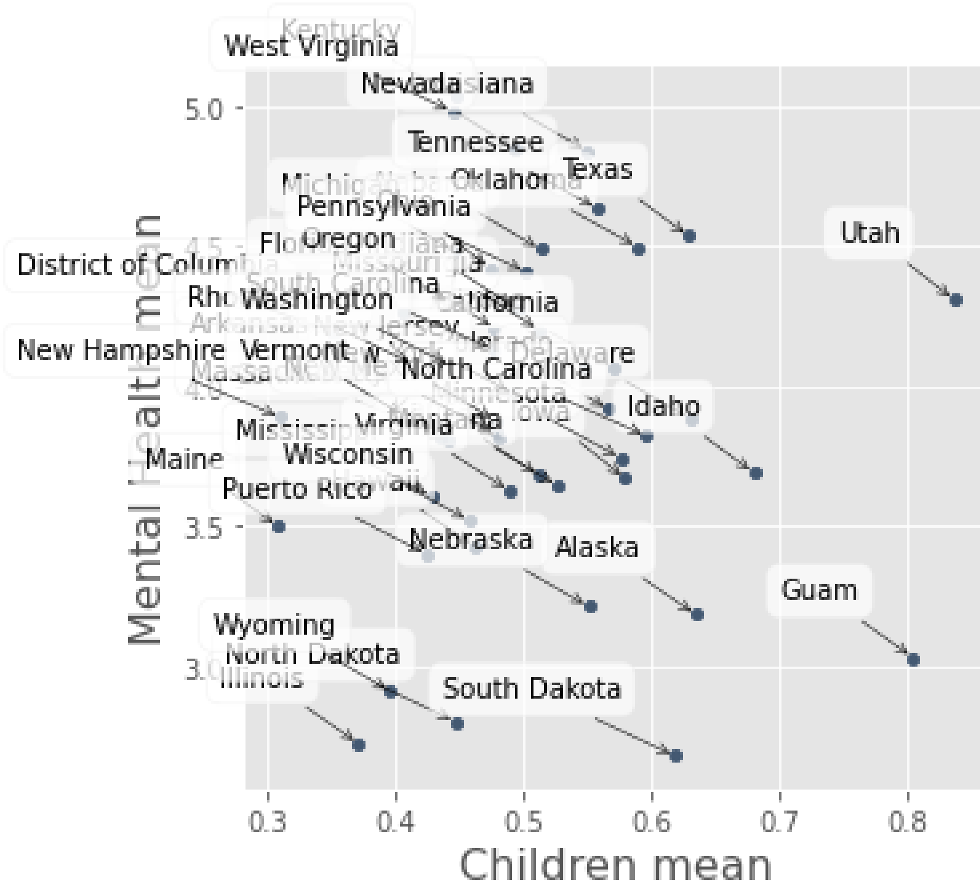
```
Out[27]: State                | Children mean | Mental Health mean
         Alabama              | 0.513538      | 4.48851
         Alaska               | 0.634738      | 3.18423
         Arizona              | 0.547451      | 4.06928
         Arkansas             | 0.376173      | 3.9864
         California           | 0.570633      | 4.0649
         Colorado             | 0.56661       | 3.91767
         Connecticut          | 0.440965      | 3.80377
         Delaware             | 0.630137      | 3.88617
         District of Columbia | 0.353083      | 4.19186
         Florida              | 0.407194      | 4.26833
         … (43 rows omitted)
```

```
In [28]: states.scatter("Children mean", "Mental Health mean") # SOLUTION
```

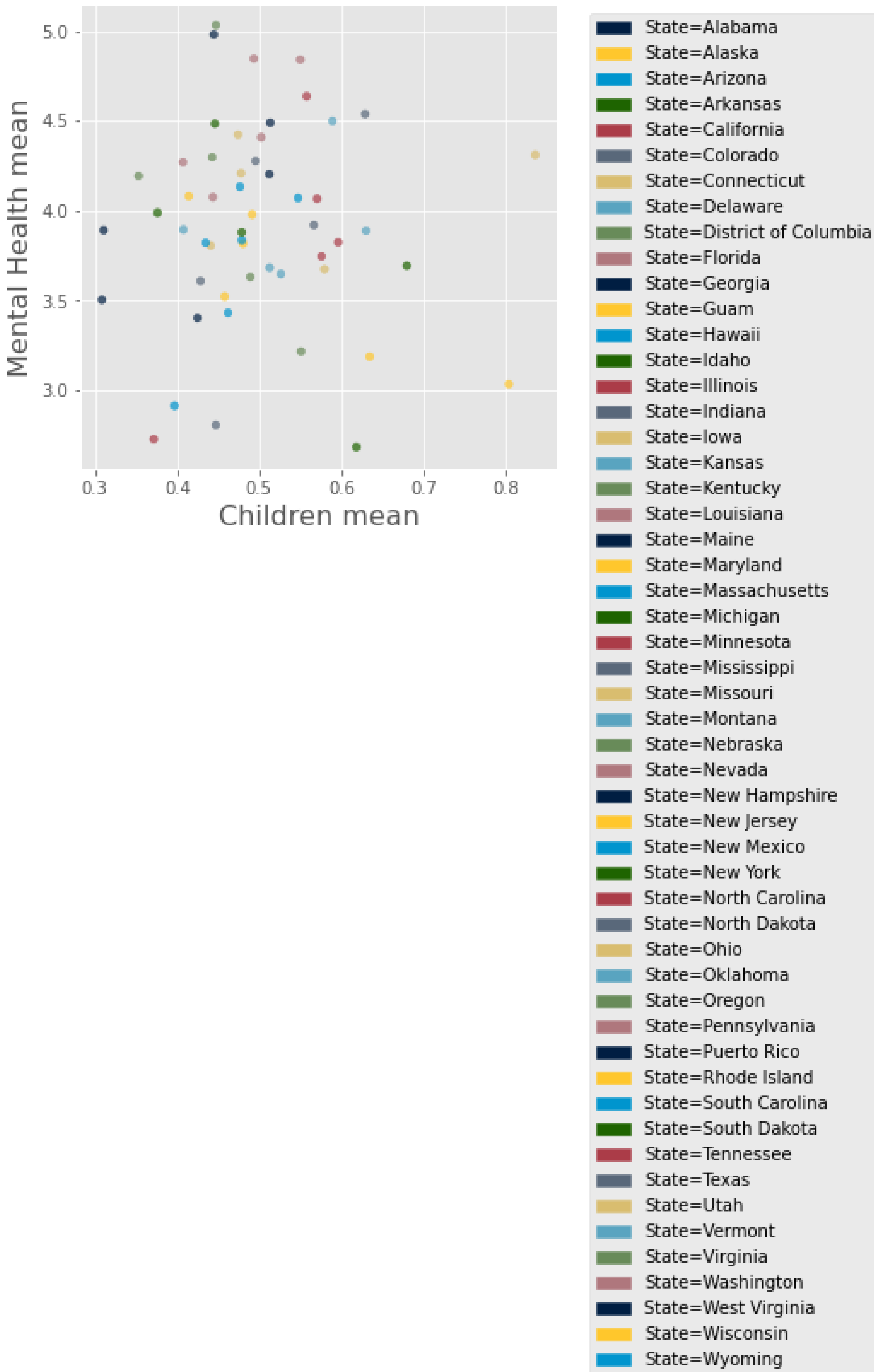


This plot looks good, but it is difficult to see which points correspond to which states. To give each data point it's city name, we can use the `group` or `label` arguments:

```
In [29]: states.scatter("Children mean", "Mental Health mean", labels="State")
```




```
In [30]: states.scatter("Children mean", "Mental Health mean", group="State")
```


Mental Health mean
Children mean
5.0
4.5
4.0
3.5
3.0
0.3
0.4
0.5
0.6
0.7
0.8
State=Alabama
State=Alaska
State=Arizona
State=Arkansas
State=California
State=Colorado
State=Connecticut
State=Delaware
State=District of Columbia
State=Florida
State=Georgia
State=Guam
State=Hawaii
State=Idaho
State=Illinois
State=Indiana
State=Iowa
State=Kansas
State=Kentucky
State=Louisiana
State=Maine
State=Maryland
State=Massachusetts
State=Michigan
State=Minnesota
State=Mississippi
State=Missouri
State=Montana
State=Nebraska
State=Nevada
State=New Hampshire
State=New Jersey
State=New Mexico
State=New York
State=North Carolina
State=North Dakota
State=Ohio
State=Oklahoma
State=Oregon
State=Pennsylvania
State=Puerto Rico
State=Rhode Island
State=South Carolina
State=South Dakota
State=Tennessee
State=Texas
State=Utah
State=Vermont
State=Virginia
State=Washington
State=West Virginia
State=Wisconsin
State=Wyoming

As you can see, one of these plots is easier to read than the other, so we were better off using the `group` argument in this case. Moreover, since there are so many states, some of the colors get reused, making it difficult to inerpret the *second* scatter plot. However, in practice, it may be useful to use `labels`, not `group`, so think about when it may be useful to use each argument.

Scatter plots are useful when visualizing two numerical variables together. If you want to plot two numerical variables but one of those variables corresponds to time, we can use a line plot to visualize the non-time variable as time passes.

5 The plot method

Similar to `scatter`, we give plot the names of two numerical columns and it creates a **line plot** for us. If we want to draw multiple line plots on the same set of axes, we give it a table with multiple numerical columns, and tell it which one contains the values for the x-axis.

The `plot` method allows us to see how non-time variables change over time. Let's use `plot` to look at the exercise patterns over the course of the year. First, we will look at a single line plot using `plot`:

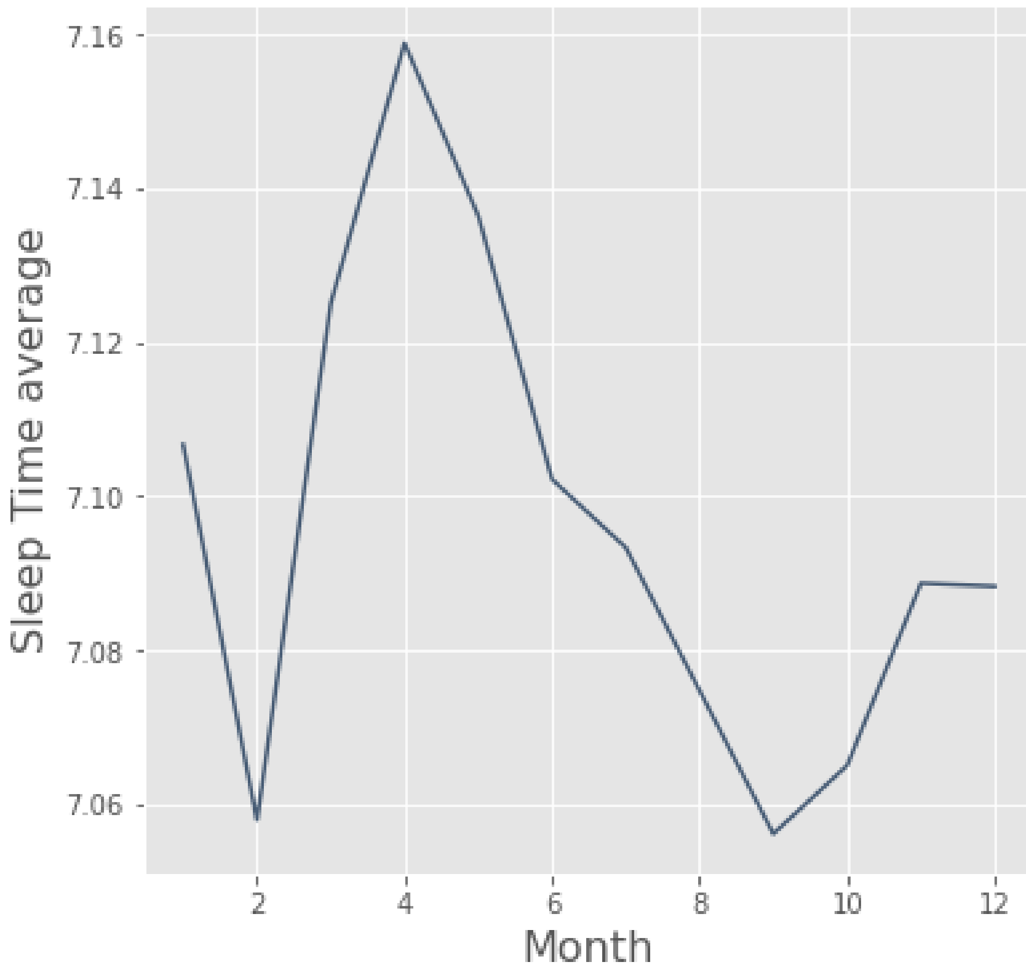
In [31]:
```
# Just run this cell to load a new table
months = Table.read_table("data/months.csv")
months
```

Out[31]:
```
Month | Sleep Time average
1     | 7.10667
2     | 7.05796
3     | 7.12505
4     | 7.15879
5     | 7.13634
6     | 7.10216
7     | 7.09329
8     | 7.07472
9     | 7.05614
10    | 7.06504
… (2 rows omitted)
```

Question 10: Using the `months` table and the `plot` method, produce a *line plot* that plots the average sleep time over time

Hint: You'll want to plot the month on the x-is and average exercise sessions on the y-axis.

```
In [32]: months.plot("Month", "Sleep Time average") # SOLUTION
```

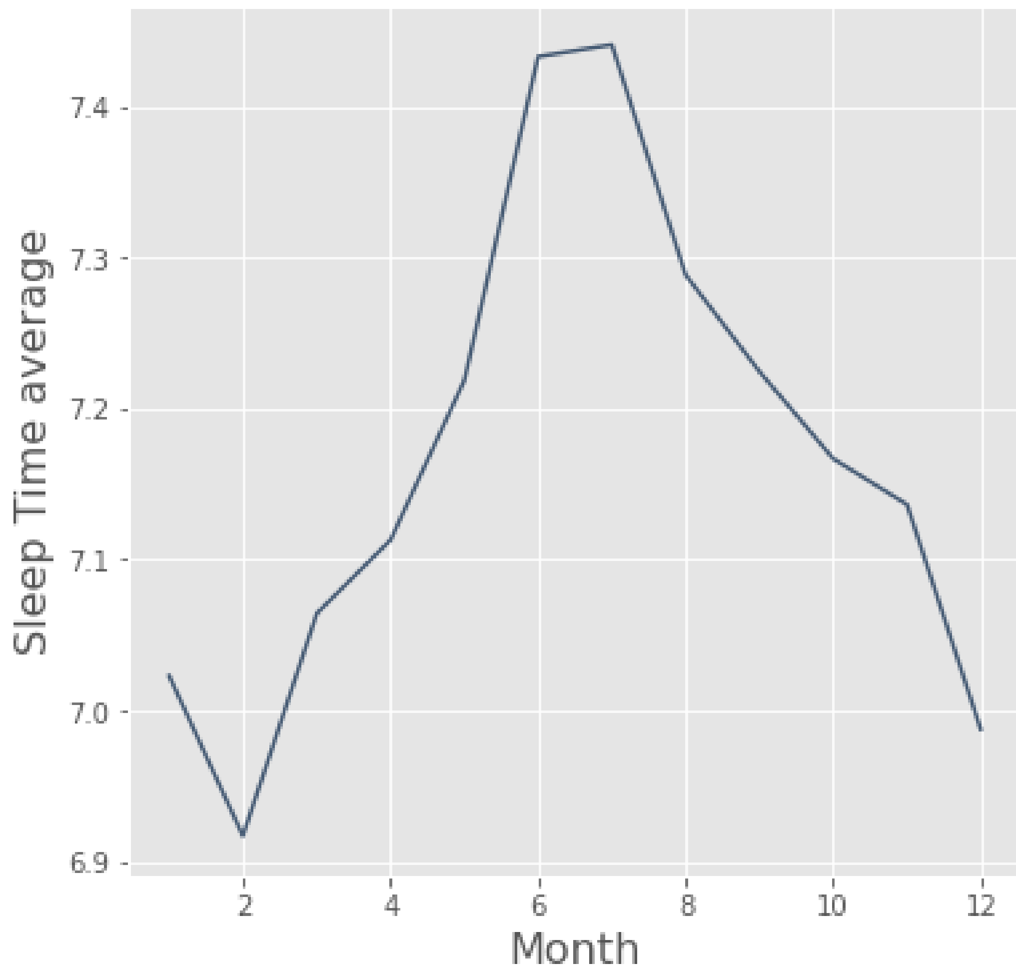


Identifying Temporal Patterns

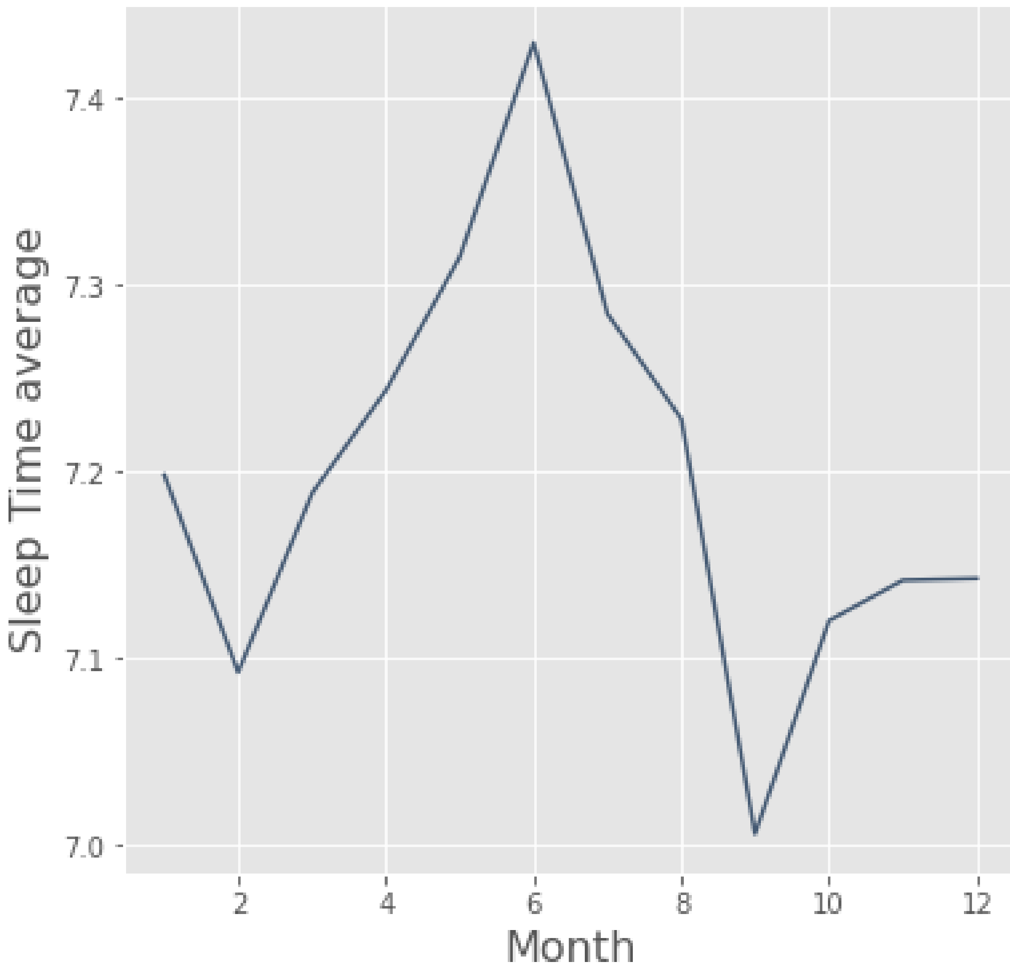
Line plots are incredibly effective tools for identifying temporal patterns (i.e. changes over time). Let's utilize our newfound knowledge of the `plot` method to uncover underlying temporal patterns within our BRFSS data. Run the following cells and answer the question that follows.

```
In [33]: # Run this cell -- you should understand how this code works
         vermont = sleep_no_negatives.where("State", "Vermont")
         florida = sleep_no_negatives.where("State", "Florida")
```

In [34]:
```
# Run this cell to produce a line plot for Vermont
vt_grouped = vermont.group("Month", np.average)
vt_grouped.plot("Month", "Sleep Time average")
```



In [35]:
```
# Run this cell to produce a line plot for Florida
fl_grouped = florida.group("Month", np.average)
fl_grouped.plot("Month", "Sleep Time average")
```



5.0.1 Multiple Variables

If we want to see multiple variables on one plot, we can include them in the table we call `plot` on.

Question 11: For both the `vermont_averages.csv` and `florida_averages.csv` files, read the file into two new tables, `vt_health` and `fl_health`, respectively. Then, for each table, select the following columns: >1. Month 2. Physical Health average 3. Mental Health average

Finally, produce a scatter plot with *one line per variable* that is not `"Month"`. That is, `"Month"` is what should be plotted on the x-axis.

```
In [36]: vt_health = Table.read_table("data/vermont_averages.csv") # SOLUTION
```

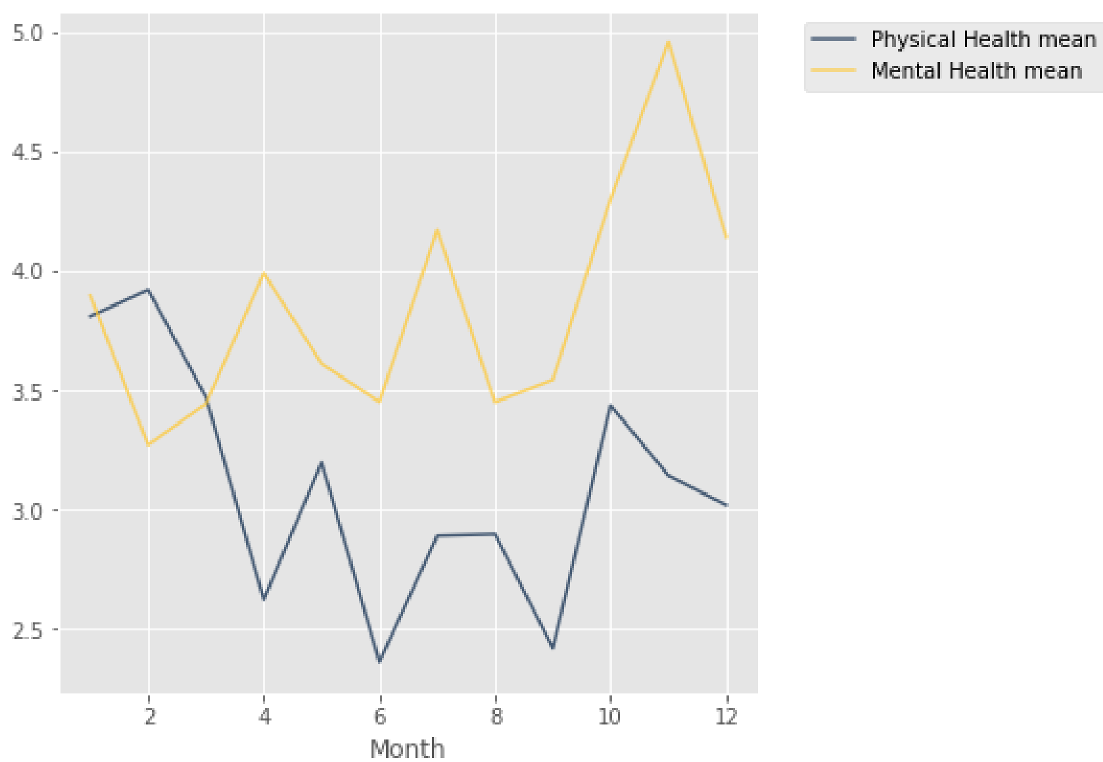
```
vt_health
```

Out[36]:

```
Month | State mean | Day mean | Year mean | Cell Phone mean | College Housing (Cell) mean | Co
1     | nan        | 1.34384  | 2020.06   | nan             | nan                         | na
2     | nan        | 1.18019  | 2020      | nan             | nan                         | na
3     | nan        | 0.949937 | 2020      | nan             | nan                         | na
4     | nan        | 1.23678  | 2020      | nan             | nan                         | na
5     | nan        | 0.926975 | 2020      | nan             | nan                         | na
6     | nan        | 0.757282 | 2020      | nan             | nan                         | na
7     | nan        | 0.663212 | 2020      | nan             | nan                         | na
8     | nan        | 1.11048  | 2020      | nan             | nan                         | na
9     | nan        | 1.09434  | 2020      | nan             | nan                         | na
10    | nan        | 1.28485  | 2020      | nan             | nan                         | na
… (2 rows omitted)
```

In [37]:

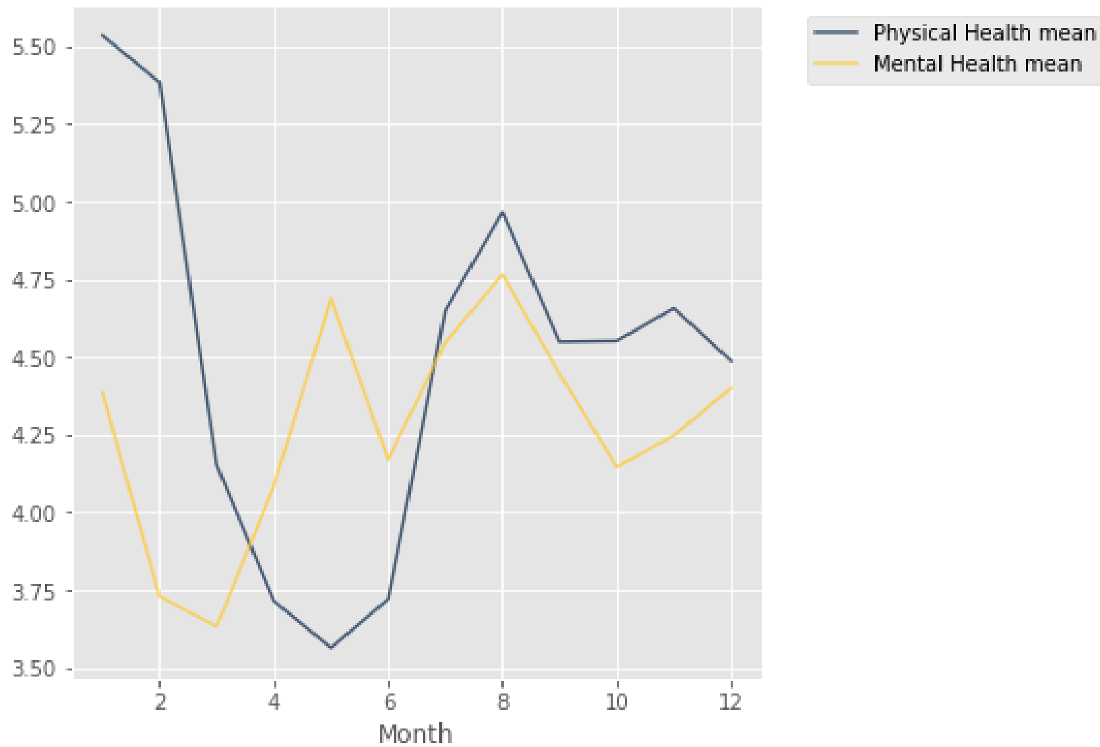
```
# Create the line plot
...
# BEGIN SOLUTION NO PROMPT
vt_health.select("Month",
                 "Physical Health mean",
                 "Mental Health mean").plot("Month")
# END SOLUTION
```



In [38]:
```
fl_health = Table.read_table("data/florida_averages.csv") # SOLUTION
fl_health
```

Out[38]:
```
Month | State mean | Day mean | Year mean | Cell Phone mean | College Housing (Cell) mean | Col
1     | nan        | 1.5877   | 2020.04   | nan             | nan                         | nan
2     | nan        | 0.554878 | 2020      | nan             | nan                         | nan
3     | nan        | 0.857633 | 2020      | nan             | nan                         | nan
4     | nan        | 0.925311 | 2020      | nan             | nan                         | nan
5     | nan        | 0.639903 | 2020      | nan             | nan                         | nan
6     | nan        | 0.563107 | 2020      | nan             | nan                         | nan
7     | nan        | 1.05165  | 2020      | nan             | nan                         | nan
8     | nan        | 1.15581  | 2020      | nan             | nan                         | nan
9     | nan        | 1.06263  | 2020      | nan             | nan                         | nan
10    | nan        | 0.438679 | 2020      | nan             | nan                         | nan
… (2 rows omitted)
```

In [39]:
```
# Create the line plot
...
# BEGIN SOLUTION NO PROMPT
fl_health.select("Month",
                 "Physical Health mean",
                 "Mental Health mean").plot("Month")
# END SOLUTION
```



Question 12 (*Discussion*): What insights can you draw for each state about how **mental and physical** health change over the course of the year?

Note: Remember that a *higher* value for both `"Mental Health"` and `"Physical Health"` corresponding to a *larger* number of days where the individual considered their mental or physical health to be *poor*.

Type your answer here, replacing this text.

SOLUTION: You can see here that average number of poor mental health days in Vermont *increases significantly* between August and November. On other hand, this seems to be the time of year in Florida where individuals, on average, have the *least* number of poor mental health days.

5.0.2 Choose a State

We've just looked at two states, but there are many more to investigate. Run the following cell to experiment with other states.

```
In [40]: def plot_state(state):
             state_tbl = brfss.where("General Health", are.not_equal_to(-1)).where("State", \
                             state).where("Physical Health", are.not_equal_to(-1))
             grouped = state_tbl.group("Month", np.average)
             reduced = grouped.select("Month",
                            "Physical Health average",
                            "Mental Health average")
             reduced.plot("Month")
             plt.title(f"{state} Line Plots")
             plt.ylim(0,15)
             plt.ylabel("Number of Days")

         state_names = ['Alabama','Alaska','Arizona','Arkansas','California', 'Colorado', 'Connecticut'
          'District of Columbia', 'Florida', 'Georgia', 'Guam', 'Hawaii', 'Idaho', 'Illinois', 'Indiana
          'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minneso
          'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire', 'New Jersey', 'New Mexico', 'Ne
          'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania', 'Puerto Rico'
          'South Carolina', 'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washi
          'West Virginia', 'Wisconsin', 'Wyoming']
```

```
In [41]: interact_manual(plot_state, state=state_names);
```

```
interactive(children=(Dropdown(description='state', options=('Alabama', 'Alaska', 'Arizona', 'Arkansas'
```

5.1 Done!

That's it! There's nowhere for you to submit this, as labs are not assignments. However, please ask any questions you have with this notebook in lab or on Ed.

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

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

5.2 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

In [ ]:
```
# Save your notebook first, then run this cell to export your submission.
grader.export(pdf=False)
```