Interactive Visualization with Bokeh - Interactive Plots - 2

One should look for what is and not what he thinks should be. (Albert Einstein)

Module completion checklist

Objective	Complete
Transform and prepare data for maps	
Create simple plots using Bokeh	

Directory settings

- In order to maximize the efficiency of your workflow, you should encode your directory structure into `variables`
- We will use the `pathlib` library
- Let the `main_dir` be the variable corresponding to your course materials folder and
- `data_dir` be the variable corresponding to your `data` folder

```
# Set 'main_dir' to location of the project folder
from pathlib import Path
home_dir = Path(".").resolve()
main_dir = home_dir.parent.parent
print(main_dir)
```

```
data_dir = str(main_dir) + "/data"
print(data_dir)
```

Costa Rican poverty: case study

- We will be diving into a case study from the **Inter-American Development Bank (IDB)**
- The **IDB** conducted a competition amongst data scientists on ***Kaggle.com***
- Many countries face this same problem of inaccurately assessing social need
- The following case study on Costa Rican poverty levels is a good example of how we can use data science within social sciences

Costa Rican poverty: backstory

Costa Rican poverty level prediction

- As stated by the ‘IDB’:
  - Social programs have a hard time making sure the right people are given enough aid
  - It’s especially tricky when a program focuses on the poorest segment of the population
  - The world’s poorest typically can’t provide the necessary income and expense records to prove that they qualify

Costa Rican poverty: backstory (cont'd)

- **Proxy Means Test (PMT)**
  - In Latin America, one popular method uses an algorithm to verify income qualification, it's called the **Proxy Means Test (or PMT)**
  - With the PMT, agencies use a model that considers a family's observable household attributes like the material of their walls and ceiling, or the assets found in the home, to classify them and predict their level of need
  - While this is an improvement, accuracy remains a problem as the region's population grows and poverty declines

Costa Rican poverty: proposed solution

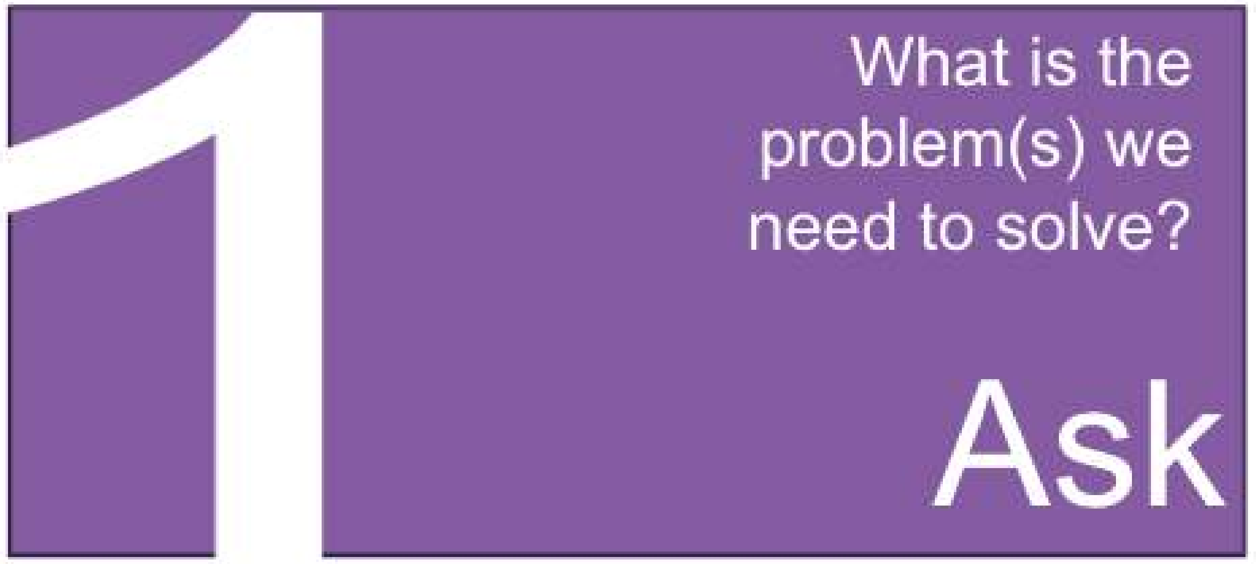
- **Proposed solution**
  - To improve on PMT, the IDB built a competition for Kaggle participants to use methods beyond traditional econometrics

  - The given dataset contains Costa Rican household characteristics with a target of four categories:
    - extreme poverty
    - moderate poverty
    - vulnerable households
    - non-vulnerable households

Costa Rican poverty: proposed solution (cont'd)

- The goal is to develop an algorithm to predict these poverty levels, that can be used on other countries facing the same problem

- We will:
  - Clean the dataset
  - Wrangle the data
  - Perform visualizations to find meaningful patterns



Load the dataset

- Let's load the entire dataset
- For reshaping and visualizations, we will be taking a specific subset
- We are now going to use the function `read_csv` to read in our `costa_rican_poverty` dataset

```
costa_rica_poverty = pd.read_csv(data_dir + '/costa_rica_poverty.csv')
print(costa_rica_poverty.head())
```

```
  household_id         ind_id  rooms  ...  age  Target  monthly_rent
0    21eb7fcc1  ID_279628684      3  ...   43       4      190000.0
1    0e5d7a658  ID_f29eb3ddd      4  ...   67       4      135000.0
2    2c7317ea8  ID_68de51c94      8  ...   92       4           NaN
3    2b58d945f  ID_d671db89c      5  ...   17       4      180000.0
4    2b58d945f  ID_d56d6f5f5      5  ...   37       4      180000.0

[5 rows x 84 columns]
```

- The entire dataset consists of `9,557` observations and `84` variables

Subsetting data

- We will explore a subset of this dataset, which includes the following variables:
  - `ppl_total`
  - `dependency_rate`
  - `num_adults`
  - `monthly rent`
  - `rooms`
  - `age`
  - `Target`
- We are choosing these variables because they illustrate the concepts best
- However, you should be able to work with (and visualize) all of your data

Subsetting data (cont'd)

- Let's subset our data so that we have the variables we need
- We are keeping `ppl_total`, `dependency_rate`, `num_adults`, `rooms`, `age`, `monthly_rent`, and `Target`
- Let's name this subset `costa_viz`

```
costa_viz = costa_rica_poverty[['ppl_total', 'dependency_rate',
                                'num_adults', 'rooms', 'age', 'monthly_rent',
                                'Target']]
print(costa_viz.head())
```

```
   ppl_total  dependency_rate  num_adults  rooms  age  monthly_rent  Target
0          1               37           1      3   43      190000.0       4
1          1               36           1      4   67      135000.0       4
2          1               36           1      8   92           NaN       4
3          4               38           2      5   17      180000.0       4
4          4               38           2      5   37      180000.0       4
```

Data prep: clean NAs

- Depending on subject matter, missing values might be significant
- Let's define the choices on how we can handle `NA`s in our data:
  - drop columns that contain any `NA`s
  - drop columns with a certain % of `NA`s
  - impute missing values
  - convert column with missing values to categorical
- Let's look at the count of `NA`s by column first:

```
print(costa_viz.isnull().sum())
```

```
ppl_total              0
dependency_rate        0
num_adults             0
rooms                  0
age                    0
monthly_rent        6860
Target                 0
dtype: int64
```

Data cleaning: NAs

- `monthly_rent` has many `NA` values!
- We could just drop this column, as the number is over 50%
- However, in this instance, we'll keep it, and **impute missing values** using the mean of the column
- There isn't a mathematical method for a precise percentage of NAs that we are OK with
- That's why your subject matter expertise is so important!

```
# Set the dataframe equal to the imputed dataset.
costa_viz = costa_viz.fillna(costa_viz.mean())
# Check how many values are null in monthly_rent.
print(costa_viz.isnull().sum())
```

```
ppl_total          0
dependency_rate    0
num_adults         0
rooms              0
age                0
monthly_rent       0
Target             0
dtype: int64
```

Converting the target variable

- Let's convert poverty to a variable with two levels, which will help to balance it out
- The four original levels would also increase the complexity of the visualizations and the code
- For this reason, we will convert levels 1, 2 and 3 to `vulnerable` and 4 to `non-vulnerable`
- The levels translate to 1, 2 and 3 as being `vulnerable` households
- Level 4 is `non-vulnerable`

```
import numpy as np
costa_viz['Target'] = np.where(costa_viz['Target'] <= 3, 'vulnerable', 'non_vulnerable')
```

```
print(costa_viz['Target'].head())
```

```
0     non_vulnerable
1     non_vulnerable
2     non_vulnerable
3     non_vulnerable
4     non_vulnerable
Name: Target, dtype: object
```

Data prep: target

- The next step of our data cleanup is to ensure the target variable is binary and has a label
- Let's look at the `dtype` of `Target`

```
print(costa_viz.Target.dtypes)
```

```
object
```

- We want to convert this to `bool` so that it is a binary class

```
costa_viz["Target"] = np.where(costa_viz["Target"] == "non_vulnerable", True, False)

# Check class again.
print(costa_viz.Target.dtypes)
```

```
bool
```

Module completion checklist

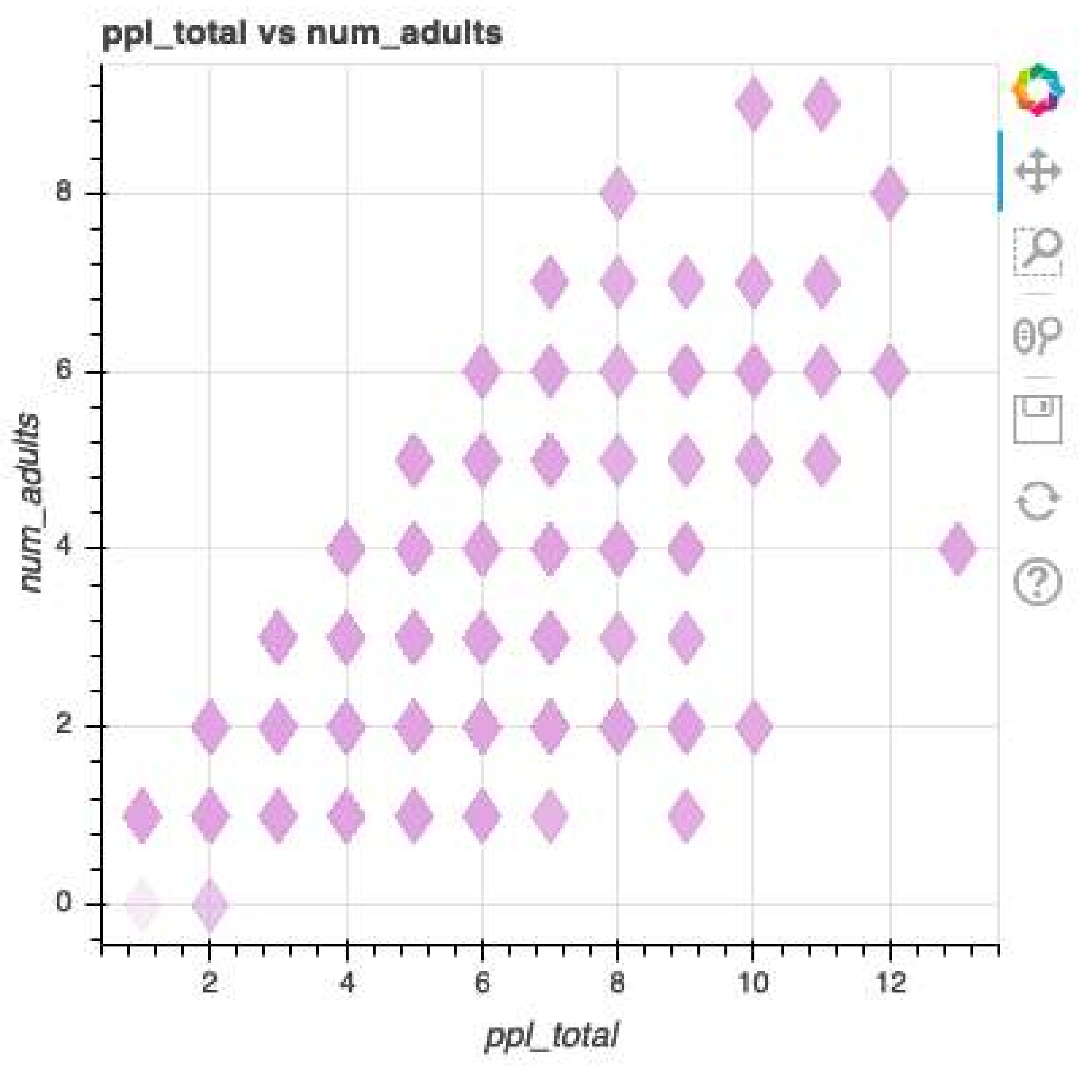
Objective	Complete
Transform and prepare data for maps	✔
Create simple plots using Bokeh	

Use Costa Rican data for plots

- We're ready to create plots with `costa_viz`

```
p = figure(title = "ppl_total vs num_adults",
           x_axis_label = 'ppl_total',
           y_axis_label = 'num_adults',
           plot_width = 400, plot_height = 400)

p.diamond(costa_viz['ppl_total'],
          costa_viz['num_adults'],
          size = 20,
          color = "plum",
          alpha = 0.2)
show(p)
```



vbar() and hbar()

- To see the count of the categorical levels, we will use the `Target` variable

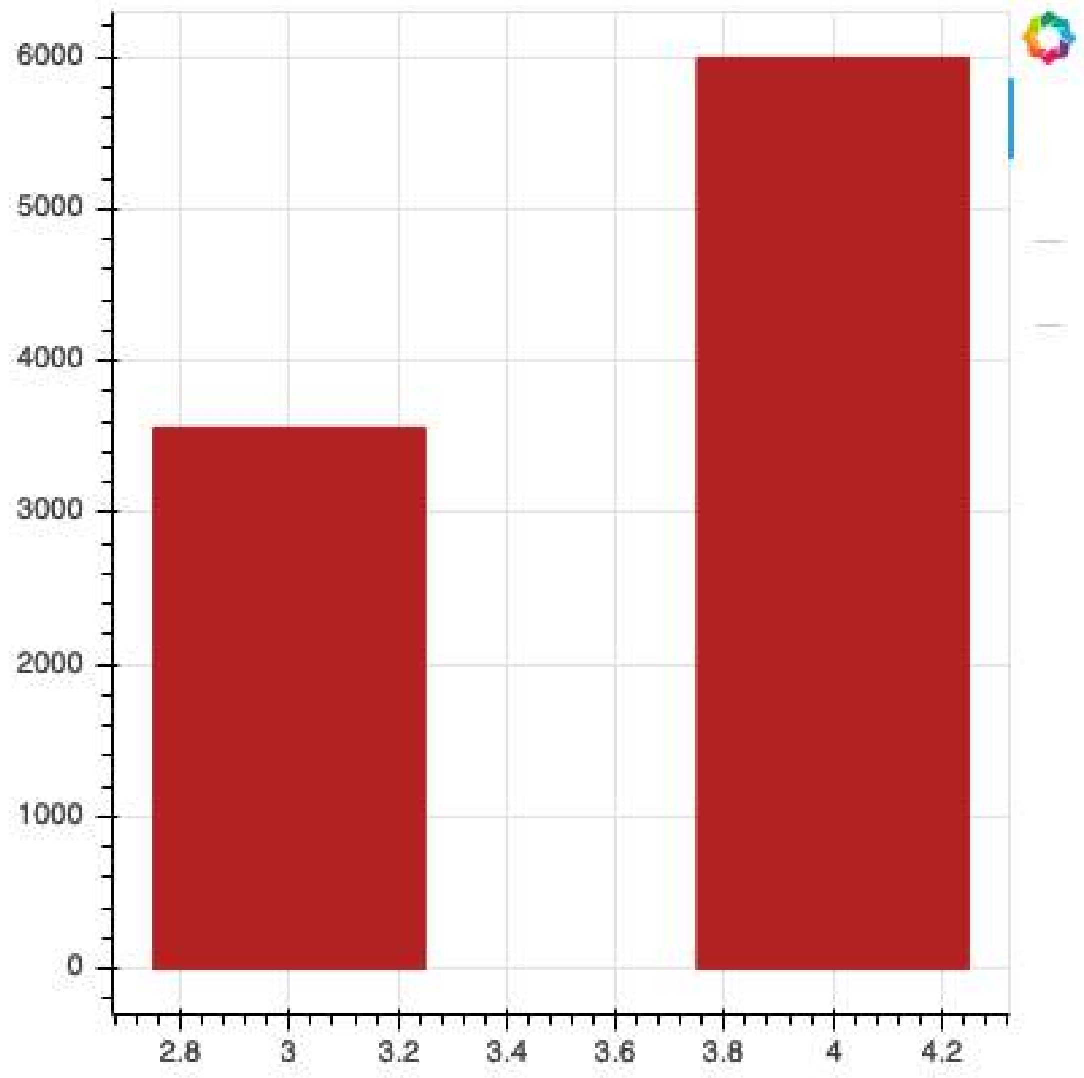
```
costa_viz.Target.value_counts()
```

```
True      5996
False     3561
Name: Target, dtype: int64
```

```
p = figure(plot_width=400, plot_height=400)

p.vbar(x = [4, 3, 2, 1],
       width = 0.5,
       bottom = 0,
       top =
costa_viz.Target.value_counts(),
       color = "firebrick")

show(p)
```



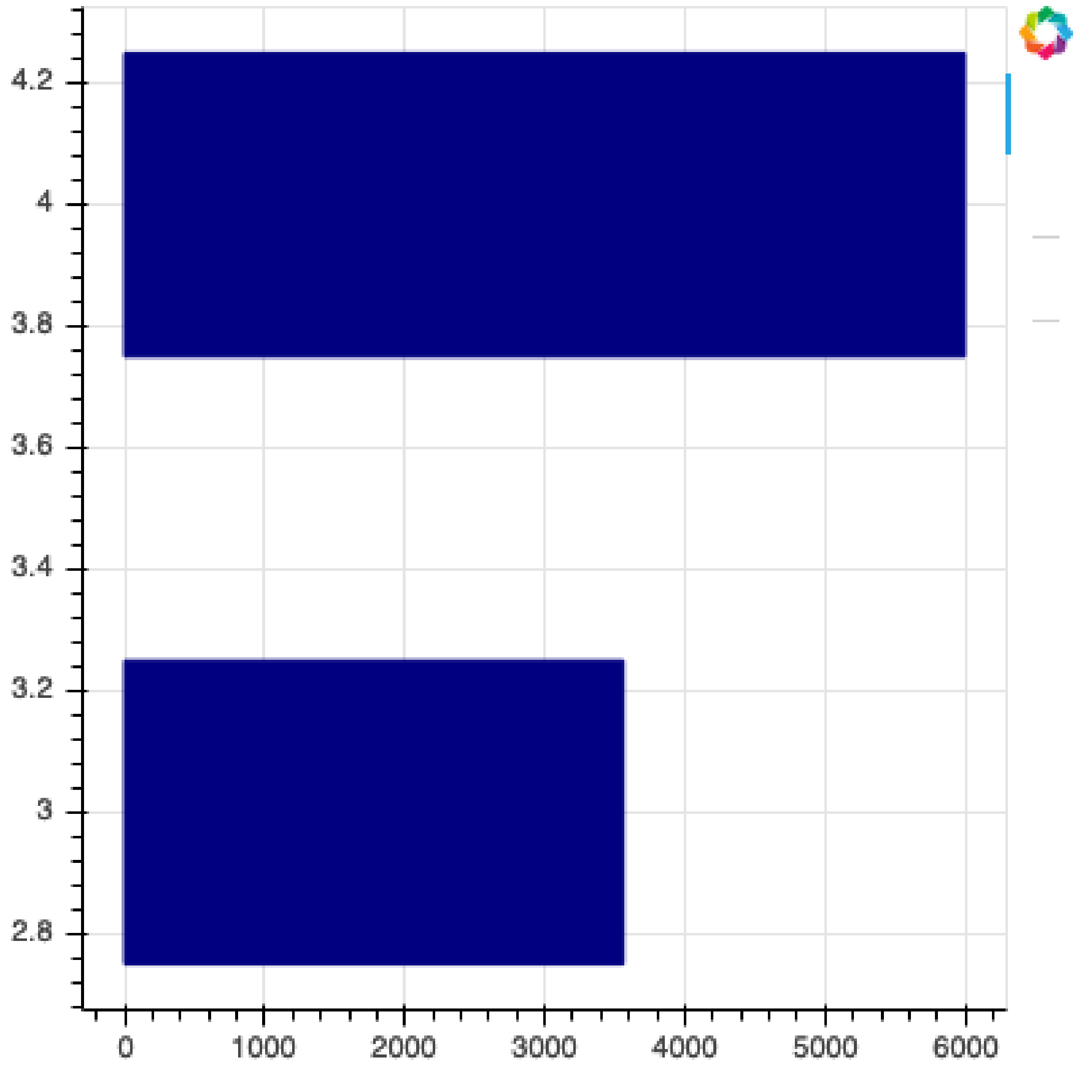
vbar() and hbar() (cont'd)

- Similarly, horizontal bar charts can be created using `.hbar()`

```
p = figure(plot_width = 400, plot_height = 400)

p.hbar(y = [4, 3, 2, 1],
       height = 0.5,
       left = 0,
       right = costa_viz.Target.value_counts(),
       color = "navy")

show(p)
```



Markers for categorical data

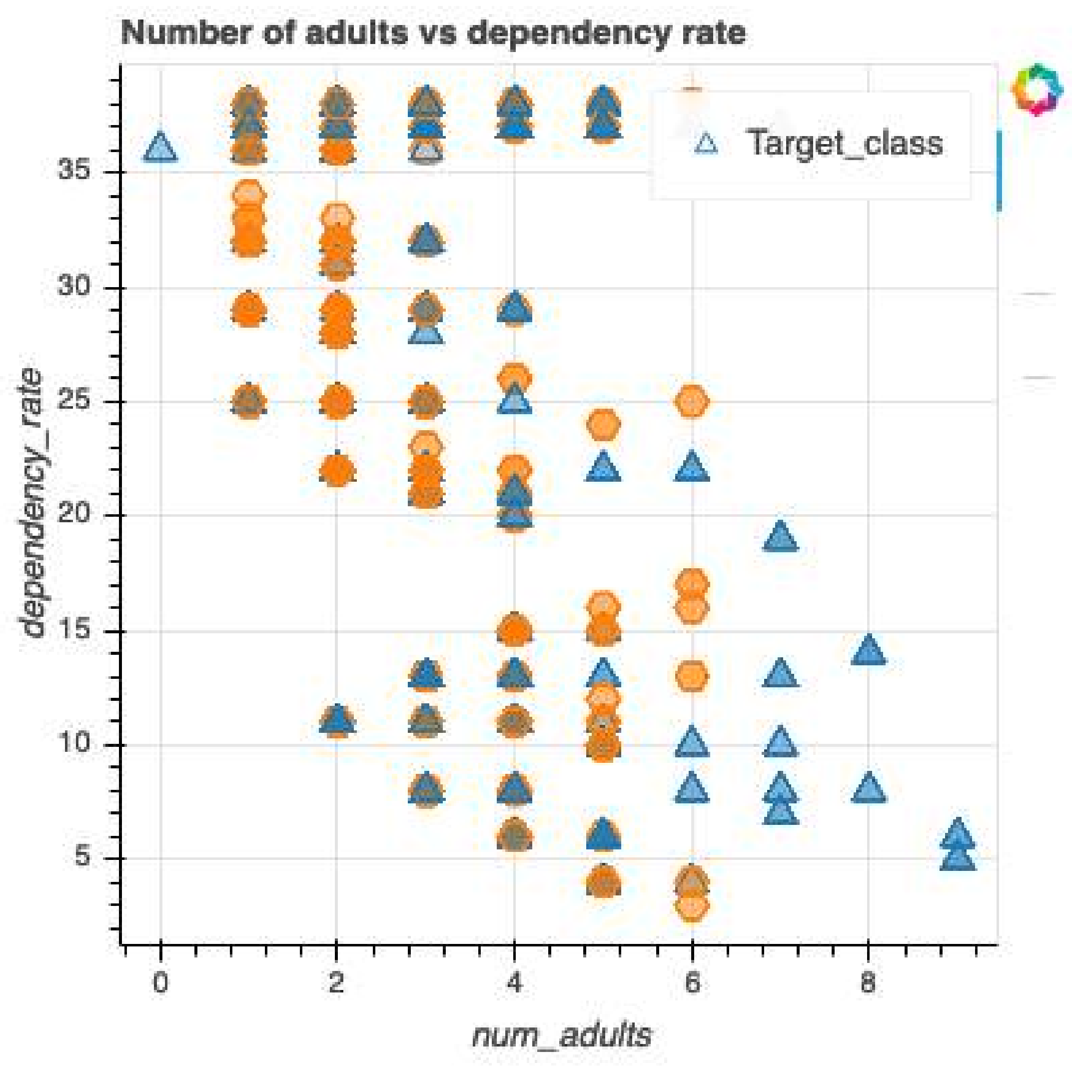
- It is also possible to map categorical data to marker types
- This example shows the use of `factor_mark()` to display different markers or different categories in the input data
- It also demonstrates the use of `factor_cmap()` to colormap those same categories

```
LEVELS = ['non_vulnerable', 'vulnerable']
MARKERS = ['triangle', 'hex']
costa_viz['Target_class'] =
np.where(costa_viz['Target']==True,
'non_vulnerable','vulnerable')

p = figure(title = "Number of adults vs
dependency rate",
           x_axis_label = 'num_adults',
           y_axis_label = 'dependency_rate')
```

Markers for categorical data

```
p.scatter("num_adults", "dependency_rate",
          source = costa_viz,
          legend_label = "Target_class",
          fill_alpha = 0.1,
          size = 12,
          marker = factor_mark('Target_class',
                               MARKERS,
                               LEVELS),
          color = factor_cmap('Target_class',
                              'Category10_7',
                              LEVELS))
show(p)
```



Knowledge check

Link: ***Click here to complete the knowledge check***

Module completion checklist

Objective	Complete
Transform and prepare data for maps	✔
Create simple plots using Bokeh	✔

This completes our module

You are now ready to try Tasks 3-10 in the Exercise for this topic