

0.0.1 Question 0

Question 0A What is the granularity of the data (i.e. what does each row represent)?

each row represents the record for an individual bike rental !

Question 0B For this assignment, we'll be using this data to study bike usage in Washington D.C. Based on the granularity and the variables present in the data, what might some limitations of using this data be? What are two additional data categories/variables that you can collect to address some of these limitations?

considering that many months of the year in DC are in winter is not a good observation of how many people are overall using their bikes. We would need to collect the month of the year to be able to understand this limitation. Another one is seeing if that person lives in the city or the suburb. It would be easier to bike within the city but probably harder if you lived in the suburb.

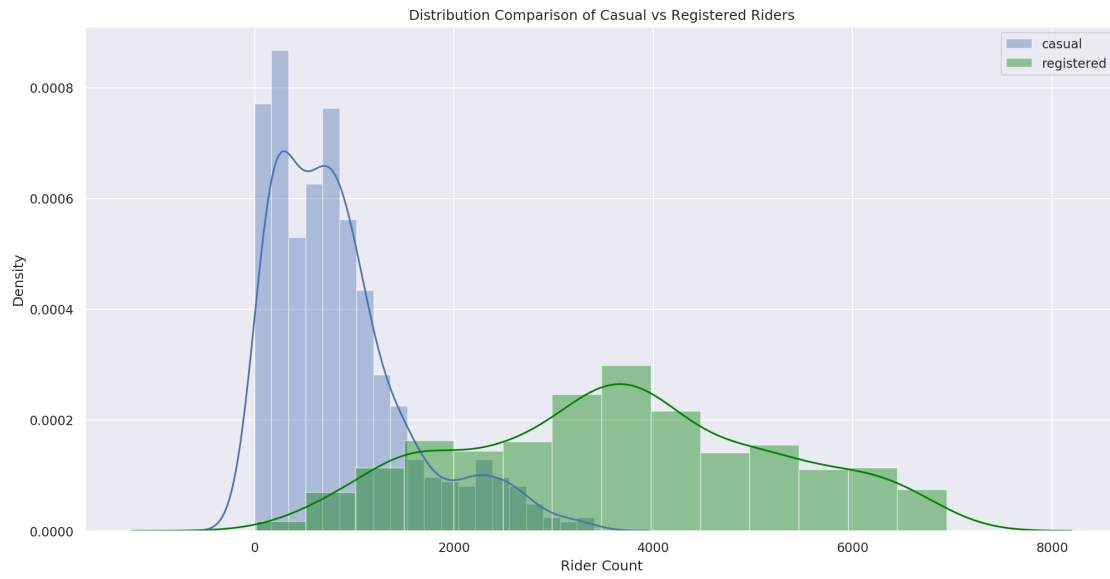
0.0.2 Question 2

Question 2a Use the `sns.distplot` function to create a plot that overlays the distribution of the daily counts of bike users, using blue to represent `casual` riders, and green to represent `registered` riders. The temporal granularity of the records should be daily counts, which you should have after completing question 1c. **You can ignore all warnings that say `distplot` is a deprecated function.**

Include a legend, xlabel, ylabel, and title. Read the [seaborn plotting tutorial](#) if you're not sure how to add these. After creating the plot, look at it and make sure you understand what the plot is actually telling us, e.g on a given day, the most likely number of registered riders we expect is ~4000, but it could be anywhere from nearly 0 to 7000.

```
In [16]: sns.distplot(daily_counts['casual'], hist=True, label='casual')
         sns.distplot(daily_counts['registered'], hist=True, label='registered', color='green')
         plt.legend()
         plt.xlabel('Rider Count')
         plt.ylabel('Density')
         plt.title('Distribution Comparison of Casual vs Registered Riders');
```

```
/opt/conda/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated alias for `displot`. Please use `displot` instead.
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for multi-dimensional arrays is deprecated.
  ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-dimensional arrays is deprecated.
  x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-dimensional arrays is deprecated.
  y = y[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated alias for `displot`. Please use `displot` instead.
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for multi-dimensional arrays is deprecated.
  ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-dimensional arrays is deprecated.
  x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-dimensional arrays is deprecated.
  y = y[:, np.newaxis]
```



0.0.3 Question 2b

In the cell below, describe the differences you notice between the density curves for casual and registered riders. Consider concepts such as modes, symmetry, skewness, tails, gaps and outliers. Include a comment on the spread of the distributions.

The symmetry of the graphs are different. The green registered riders show a symmetrical distribution. The blue casual riders show a positive skewness and show right tailed distribution.

0.0.4 Question 2c

The density plots do not show us how the counts for registered and casual riders vary together. Use `sns.lmplot` to make a scatter plot to investigate the relationship between casual and registered counts. This time, let's use the `bike` DataFrame to plot hourly counts instead of daily counts.

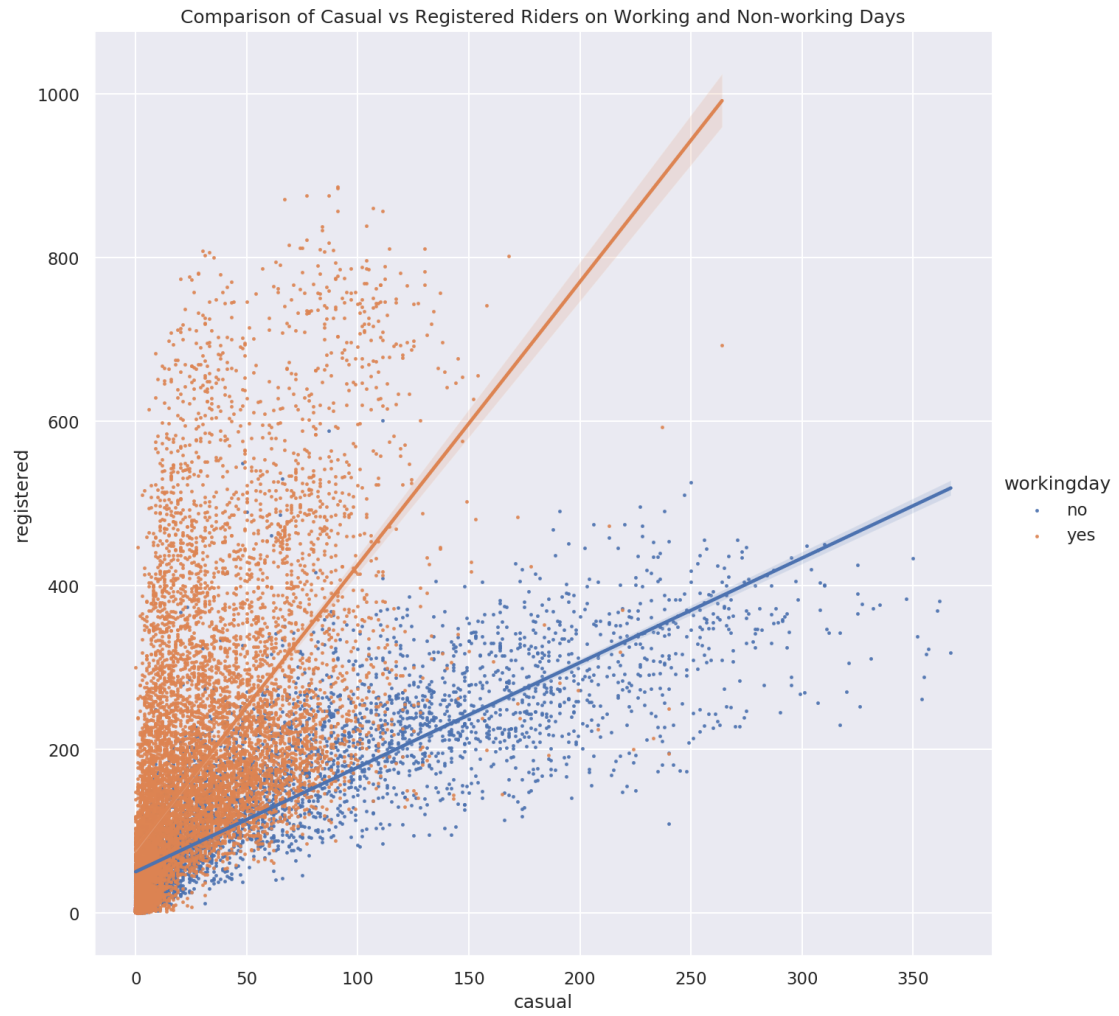
The `lmplot` function will also try to draw a linear regression line (just as you saw in Data 8). Color the points in the scatterplot according to whether or not the day is a working day (your colors do not have to match ours exactly, but they should be different based on whether the day is a working day).

There are many points in the scatter plot, so make them small to help reduce overplotting. Also make sure to set `fit_reg=True` to generate the linear regression line. You can set the `height` parameter if you want to adjust the size of the `lmplot`.

Hints: * Checkout this helpful [tutorial on lmplot](#).

- You will need to set `x`, `y`, and `hue` and the `scatter_kws`.

```
In [33]: # Make the font size a bit bigger
sns.set(font_scale=1)
sns.lmplot(x='casual', y='registered', data = bike, hue = 'workingday', fit_reg=True, height=9
plt.title('Comparison of Casual vs Registered Riders on Working and Non-working Days');
```



0.0.5 Question 2d

What does this scatterplot seem to reveal about the relationship (if any) between casual and registered riders and whether or not the day is on the weekend? What effect does [overplotting](#) have on your ability to describe this relationship?

The number of casual riders and registered riders show a positive correlation. Using overplotting clusters the dots to become unreadable. We have no way of knowing how many blue dots are present because the orange dots overlap.

Generating the plot with weekend and weekday separated can be complicated so we will provide a walk-through below, feel free to use whatever method you wish if you do not want to follow the walkthrough.

Hints: * You can use `loc` with a boolean array and column names at the same time * You will need to call `kdeplot` twice. * Check out this [guide](#) to see an example of how to create a legend. In particular, look at how the example in the guide makes use of the `label` argument in the call to `plt.plot()` and what the `plt.legend()` call does. This is a good exercise to learn how to use examples to get the look you want. * You will want to set the `cmap` parameter of `kdeplot` to "Reds" and "Blues" (or whatever two contrasting colors you'd like). You are required for this question to use two sets of contrasting colors for your plots.

After you get your plot working, experiment by setting `shade=True` in `kdeplot` to see the difference between the shaded and unshaded version. Please submit your work with `shade=False`.

```
In [41]: # Set 'is_workingday' to a boolean array that is true for all working_days
is_workingday = daily_counts['workingday']== 'yes'

# Bivariate KDEs require two data inputs.
# In this case, we will need the daily counts for casual and registered riders on workdays
casual_workday = daily_counts.loc[is_workingday, 'casual']
registered_workday = daily_counts.loc[is_workingday, 'registered']

# Use sns.kdeplot on the two variables above to plot the bivariate KDE for weekday rides
sns.kdeplot(casual_workday, registered_workday, label='Workday', cmap='Reds')

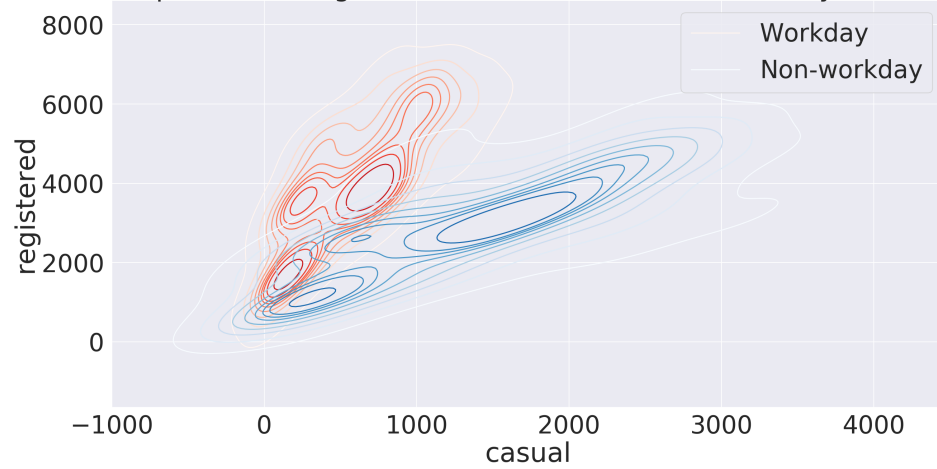
# Repeat the same steps above but for rows corresponding to non-workingdays
casual_non_workday = daily_counts.loc[~is_workingday, 'casual']
registered_non_workday = daily_counts.loc[~is_workingday, 'registered']

# Use sns.kdeplot on the two variables above to plot the bivariate KDE for non-workingday rides
sns.kdeplot(casual_non_workday, registered_non_workday, label='Non-workday', cmap='Blues')

plt.legend()
plt.title('KDE Plot Comparison of Registered vs Casual Riders in Workday and Non-Workday')
sns.set(rc={'figure.figsize':(20,10)})
sns.set(font_scale=3)
```

```
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
warnings.warn(
```

KDE Plot Comparison of Registered vs Casual Riders in Workday and Non-Workday



Question 3b What additional details can you identify from this contour plot that were difficult to determine from the scatter plot?

the sharing of casual riders are between 0-800 and 1100-2000 during the workday. The sharing of registered riders are between 800-900 and 2500-2400 during the workday

the sharing of casual riders are between 0-300, 200-500, 500-900 during non-workdays. The sharing of registered riders are between 1000-2200, 3000-4000, 3000-5000 during non-workdays.

0.1 4: Joint Plot

As an alternative approach to visualizing the data, construct the following set of three plots where the main plot shows the contours of the kernel density estimate of daily counts for registered and casual riders plotted together, and the two "margin" plots (at the top and right of the figure) provide the univariate kernel density estimate of each of these variables. Note that this plot makes it harder see the linear relationships between casual and registered for the two different conditions (weekday vs. weekend).

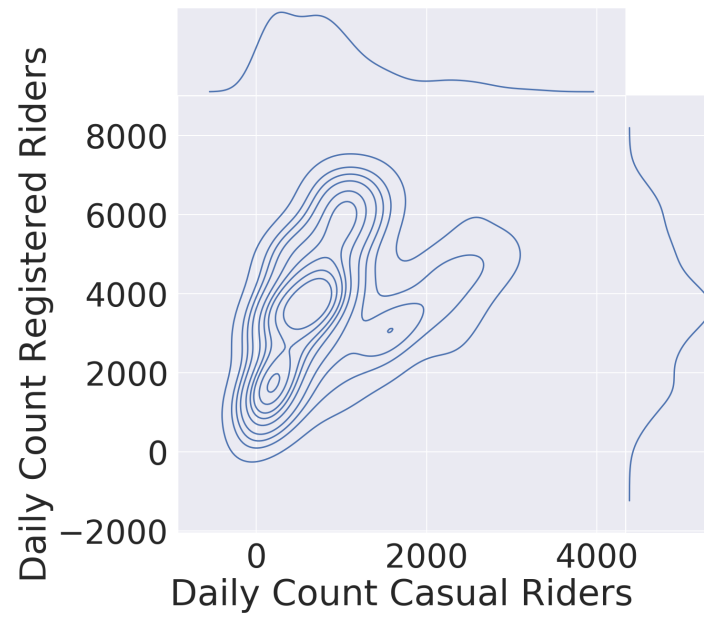
Hints: * The [seaborn plotting tutorial](#) has examples that may be helpful. * Take a look at `sns.jointplot` and its `kind` parameter. * `set_axis_labels` can be used to rename axes on the contour plot. * `plt.suptitle` from lab 1 can be handy for setting the title where you want. * `plt.subplots_adjust(top=0.9)` can help if your title overlaps with your plot

```
In [22]: sns.jointplot('casual', 'registered', data=daily_counts, kind='kde', height=10, space=0, color=
```

```
plt.suptitle('KDE Countours of Casual vs Registered Rider Count')
plt.subplots_adjust(top=0.9)
```

```
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
warnings.warn(
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for mu
ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-d
x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-d
y = y[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support for mu
ndim = x[:, None].ndim
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for multi-d
x = x[:, np.newaxis]
/opt/conda/lib/python3.8/site-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for multi-d
y = y[:, np.newaxis]
```

KDE Countours of Casual vs Registered Rider Count



0.2 5: Understanding Daily Patterns

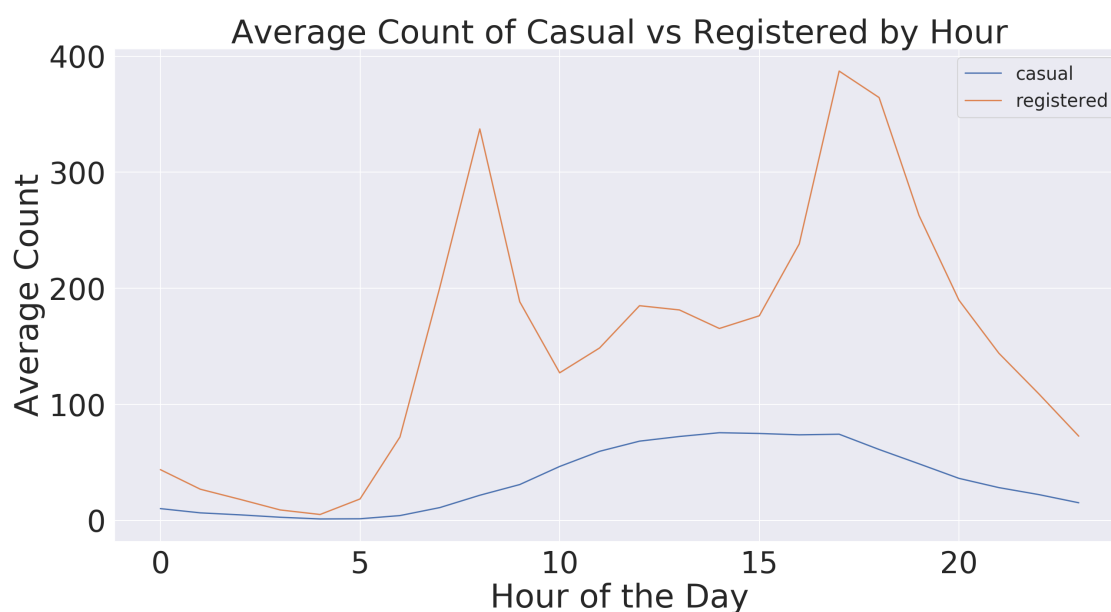
0.2.1 Question 5

Question 5a Let's examine the behavior of riders by plotting the average number of riders for each hour of the day over the **entire dataset**, stratified by rider type.

Your plot should look like the plot below. While we don't expect your plot's colors to match ours exactly, your plot should have different colored lines for different kinds of riders.

```
In [24]: registered_5a= bike['registered'].groupby(bike['hr']).agg('mean')
casual_5a = bike['casual'].groupby(bike['hr']).agg('mean')
plt.plot(casual_5a, label='casual')
plt.plot(registered_5a, label='registered')
plt.legend(prop={'size':20})
plt.title('Average Count of Casual vs Registered by Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Average Count')
```

```
Out[24]: Text(0, 0.5, 'Average Count')
```



Question 5b What can you observe from the plot? Hypothesize about the meaning of the peaks in the registered riders' distribution.

casual riders have a smaller smoother line which mean that the number of riders does not fluxuate as much. The peak is between 13-15:00 during the day. However its peak is lower than 100 riders per day.

registered riders have 3 modes, one at 8am, 13:00, then 17:00. the peak of the mode is closest to 400 riders!

the peaks represent the largest number of registered riders during a certian time of the day.

In our case with the bike ridership data, we want 7 curves, one for each day of the week. The x-axis will be the temperature and the y-axis will be a smoothed version of the proportion of casual riders.

You should use `statsmodels.nonparametric.smoothers_lowess.lowess` just like the example above. Unlike the example above, plot ONLY the lowess curve. Do not plot the actual data, which would result in overplotting. For this problem, the simplest way is to use a loop.

You do not need to match the colors on our sample plot as long as the colors in your plot make it easy to distinguish which day they represent.

Hints: * Start by just plotting only one day of the week to make sure you can do that first.

- The `lowess` function expects y coordinate first, then x coordinate.
- Look at the top of this homework notebook for a description of the temperature field to know how to convert to Fahrenheit. By default, the temperature field ranges from 0.0 to 1.0. In case you need it, $\text{Fahrenheit} = \text{Celsius} * \frac{9}{5} + 32$.

Note: If you prefer plotting temperatures in Celsius, that's fine as well!

```
In [32]: from statsmodels.nonparametric.smoothers_lowess import lowess
```

```
plt.figure(figsize=(10,8))
bike['fatemp']=bike['temp']* 41*(9/5)+32
weekday = ['Sat', 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri']
for i in weekday:
    x = bike[bike['weekday']==i]['fatemp'].array
    y = bike[bike['weekday']==i]['prop_casual'].array
    y_smooth_curve = lowess(y, x, return_sorted=False)
    sns.lineplot(x, y_smooth_curve, label=i)

plt.title('Temperature vs Casual Rider Proportion by Weekday')
plt.xlabel('Temperature (Fahrenheit)')
plt.ylabel('Casual Rider Proportion')
plt.legend(prop={'size':20})
```

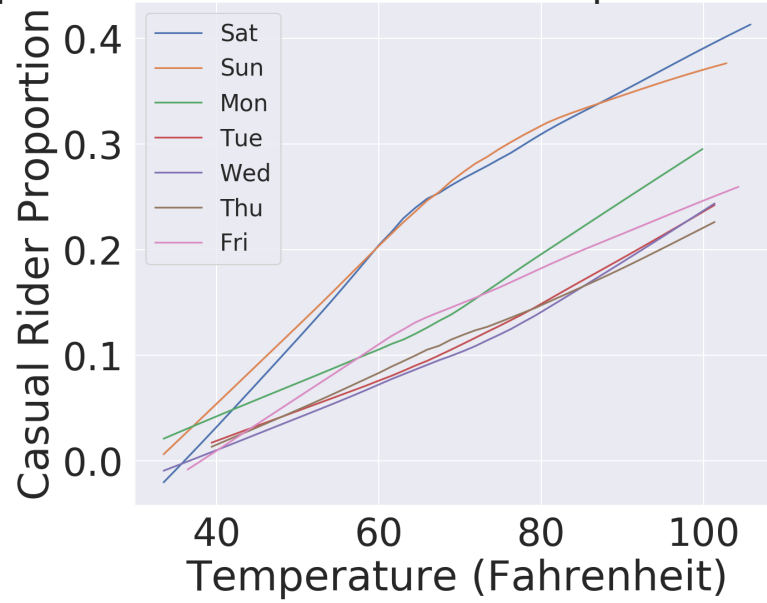
```
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: {'x': 'fatemp', 'y': 'prop_casual'} in the function call, instead of as positional arguments.
warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: {'x': 'fatemp', 'y': 'prop_casual'} in the function call, instead of as positional arguments.
warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: {'x': 'fatemp', 'y': 'prop_casual'} in the function call, instead of as positional arguments.
warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: {'x': 'fatemp', 'y': 'prop_casual'} in the function call, instead of as positional arguments.
warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: {'x': 'fatemp', 'y': 'prop_casual'} in the function call, instead of as positional arguments.
warnings.warn(
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword arguments: {'x': 'fatemp', 'y': 'prop_casual'} in the function call, instead of as positional arguments.
warnings.warn(
```

```
warnings.warn(  
/opt/conda/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var  
warnings.warn(  

```

Out[32]: <matplotlib.legend.Legend at 0x7f1720de6d00>

Temperature vs Casual Rider Proportion by Weekday



Question 6c What do you see from the curve plot? How is `prop_casual` changing as a function of temperature? Do you notice anything else interesting?

the biggest slopes are Saturday and Sunday but in general we see a positive correlation between temperature and rider prop. I notice that sat, sun, and friday show less linear lines showing more concavity.

0.2.2 Question 7

Question 7A Imagine you are working for a Bike Sharing Company that collaborates with city planners, transportation agencies, and policy makers in order to implement bike sharing in a city. These stakeholders would like to reduce congestion and lower transportation costs. They also want to ensure the bike sharing program is implemented equitably. In this sense, equity is a social value that is informing the deployment and assessment of your bike sharing technology.

Equity in transportation includes: improving the ability of people of different socio-economic classes, genders, races, and neighborhoods to access and afford the transportation services, and assessing how inclusive transportation systems are over time.

Do you think the `bike` data as it is can help you assess equity? If so, please explain. If not, how would you change the dataset? You may discuss how you would change the granularity, what other kinds of variables you'd introduce to it, or anything else that might help you answer this question.

i

Question 7B Bike sharing is growing in popularity and new cities and regions are making efforts to implement bike sharing systems that complement their other transportation offerings. The goals of these efforts are to have bike sharing serve as an alternate form of transportation in order to alleviate congestion, provide geographic connectivity, reduce carbon emissions, and promote inclusion among communities.

Bike sharing systems have spread to many cities across the country. The company you work for asks you to determine the feasibility of expanding bike sharing to additional cities of the U.S.

Based on your plots in this assignment, what would you recommend and why? Please list at least two reasons why, and mention which plot(s) you drew your analysis from.

Note: There isn't a set right or wrong answer for this question, feel free to come up with your own conclusions based on evidence from your plots!

Type your answer here, replacing this text.

