Notebook

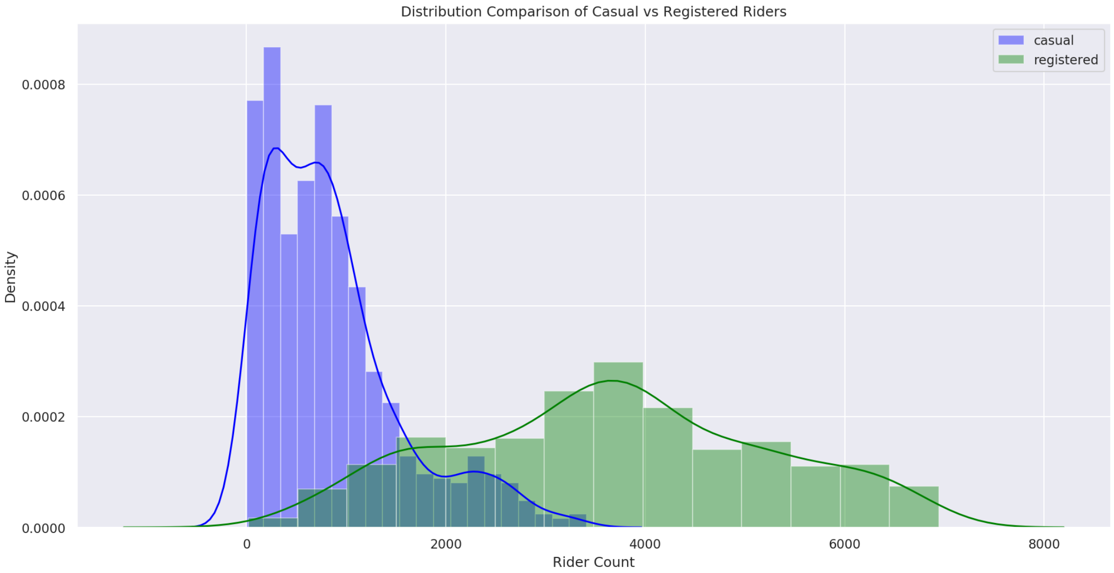
July 9, 2019

0.0.1 Question 2

Question 2a Use the sns.distplot function to create a plot that overlays the distribution of the daily counts of casual and registered users. The temporal granularity of the records should be daily counts, which you should have after completing question 1c.

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 [106]: ax =plt.subplots()
          plot_casual = sns.distplot(daily_counts['casual'], label = 'casual', color='blue')
          plot_registered = sns.distplot(daily_counts['registered'], label='registered', color='green')
          plt.xlabel('Rider Count')
          plt.ylabel('Density')
          plt.title('Distribution Comparison of Casual vs Registered Riders')
          plt.legend();
```



0.0.2 Question 2b

In the cell below, descibe 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.

For the registered density curve, we can see that the mode rider count is around 4000 which is also about where the median lies. There don't seem to be any outliers for this density curve as well since the density curve is mostly symmetric around the median. However, when look at the casual density curve, we can see that it is skeweded further to the left, which relates to the mode that tells us that a little over 0 rider count is the mode. Thus we can see that casual riders typically do not ride while registered riders contribute to the majority distrbution of riders.

0.0.3 Question 2c

In addition to the type of rider (casual vs. registered) and the overall count of each, what other kinds of demographic data would be useful (e.g. identity, neighborhood, monetary expenses, etc.)?

What is an example of a privacy or consent issue that could occur when accessing the demographic data you brought up in the previous question?

Some other demographic data that would be useful would be age, sex, and income.

By accessing someone's information on their identity, housing, and income, we could be breaching the privacy of someone's life as well as their could be risk of identity theft if their information is linked to that individuals name, etc.

0.0.4 Question 2d

What is an example of a privacy or consent issue that could occur when accessing the demographic data you brought up in the previous question?

By accessing someone's information on their identity, housing, and income, we could be breaching the privacy of someone's life as well as their could be risk of identity theft if their information is linked to that individuals name, etc.

0.0.5 Question 2e

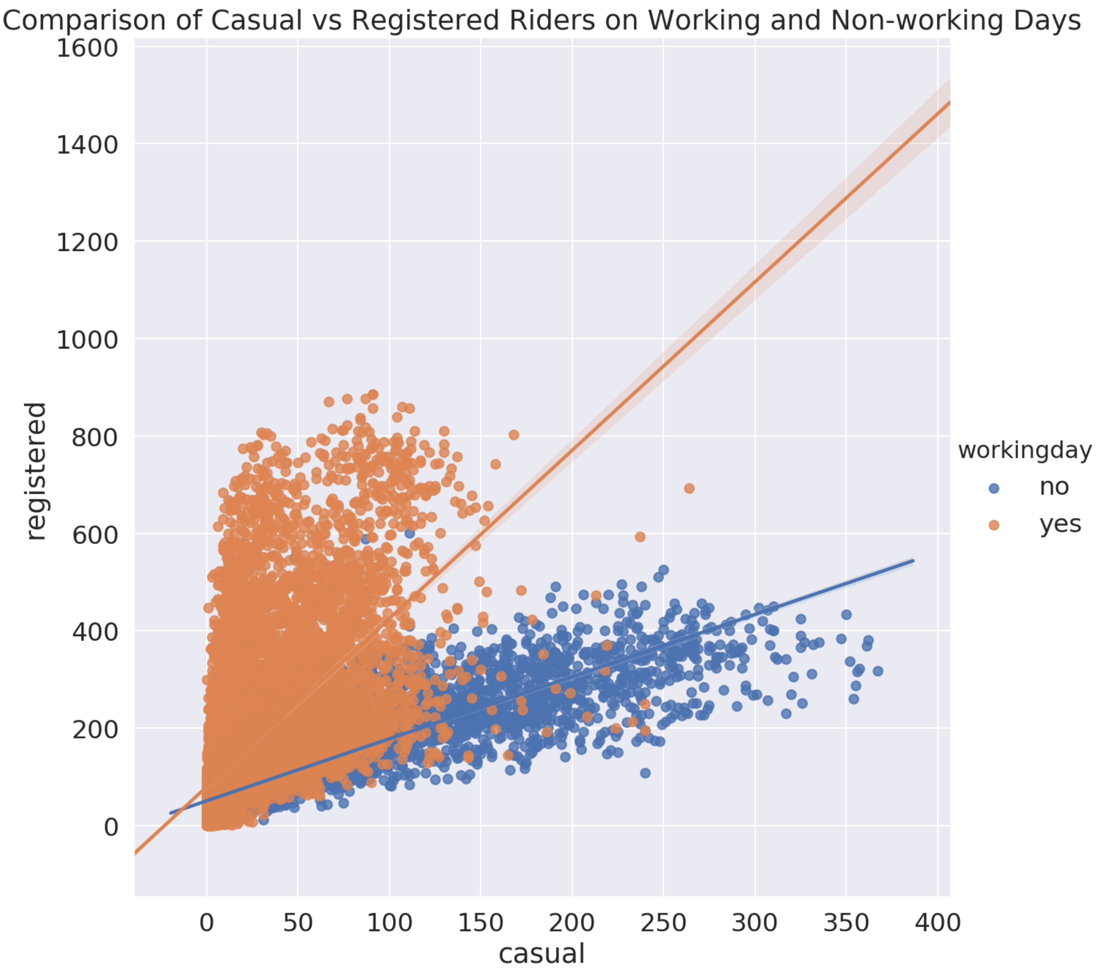
The density plots do not show us how the counts for registered and casual riders vary together. Use sns.lmplot to make a scatter plot to investigate the relationship between casual and registered counts. This time, let's use the bike DataFrame to plot hourly counts instead of daily counts.

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 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. Make sure to include a title.

Hints: * Checkout this helpful tutorial on lmplot.

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

```
In [77]: # Make the font size a bit bigger
         sns.set(font_scale=1.5)
         sns.lmplot(x='casual', y='registered', height= 9, data=bike, fit_reg=True, hue='workingday')
         plt.title('Comparison of Casual vs Registered Riders on Working and Non-working Days');
```



0.0.6 Question 2f

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?

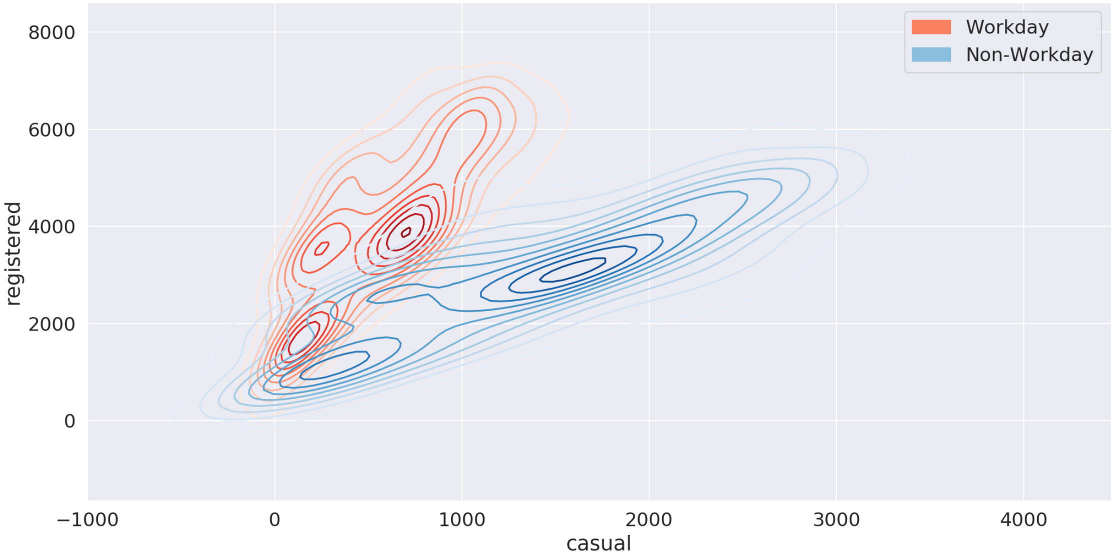
From the scatterplot, we can see that on a non-workingday there are more casual riders whereas on workingdays we see that the majority of them are registered riders. By overplotting, it makes it difficult to distinguish the relationship between registered and casual riders thus, it's harder to draw any ideas on whether a relationship exists or not.

Generating the plot with weekend and weekday separated can be complicated so we will provide a walkthrough below, feel free to use whatever method you wish however 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 tutorial to see an example of how to set colors for each dataset and how to create a legend. The legend part uses some weird matplotlib syntax that we haven't learned! You'll probably find creating the legend annoying, but it's 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).

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 [79]: import matplotlib.patches as mpatches  # see the tutorial for how we use mpatches to generate

         # 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 weekdays
         # Hint: use loc and is_workingday to splice out the relevant rows and column (casual/registere
         casual_weekday = daily_counts.loc[:,'casual'][is_workingday]
         registered_weekday = daily_counts.loc[:,'registered'][is_workingday]

         # Use sns.kdeplot on the two variables above to plot the bivariate KDE for weekday rides
         sns.kdeplot(casual_weekday, registered_weekday, cmap='Reds', shade=False)

         # Repeat the same steps above but for rows corresponding to non-workingdays
         not_workingday=daily_counts['workingday'] =='no'
         casual_weekend = daily_counts.loc[:,'casual'][not_workingday]
         registered_weekend = daily_counts.loc[:,'registered'][not_workingday]

         # Use sns.kdeplot on the two variables above to plot the bivariate KDE for weekday rides
         sns.kdeplot(casual_weekend, registered_weekend, cmap='Blues',shade=False)

         r = sns.color_palette("Reds")[2]
         b = sns.color_palette("Blues")[2]
         red_patch = mpatches.Patch(color=r, label='Workday')
         blue_patch = mpatches.Patch(color=b, label='Non-Workday')
         plt.legend(handles=[red_patch,blue_patch]);
```


Workday
Non-Workday
8000
6000
4000
2000
0
registered
−1000
0
1000
2000
3000
4000
casual

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

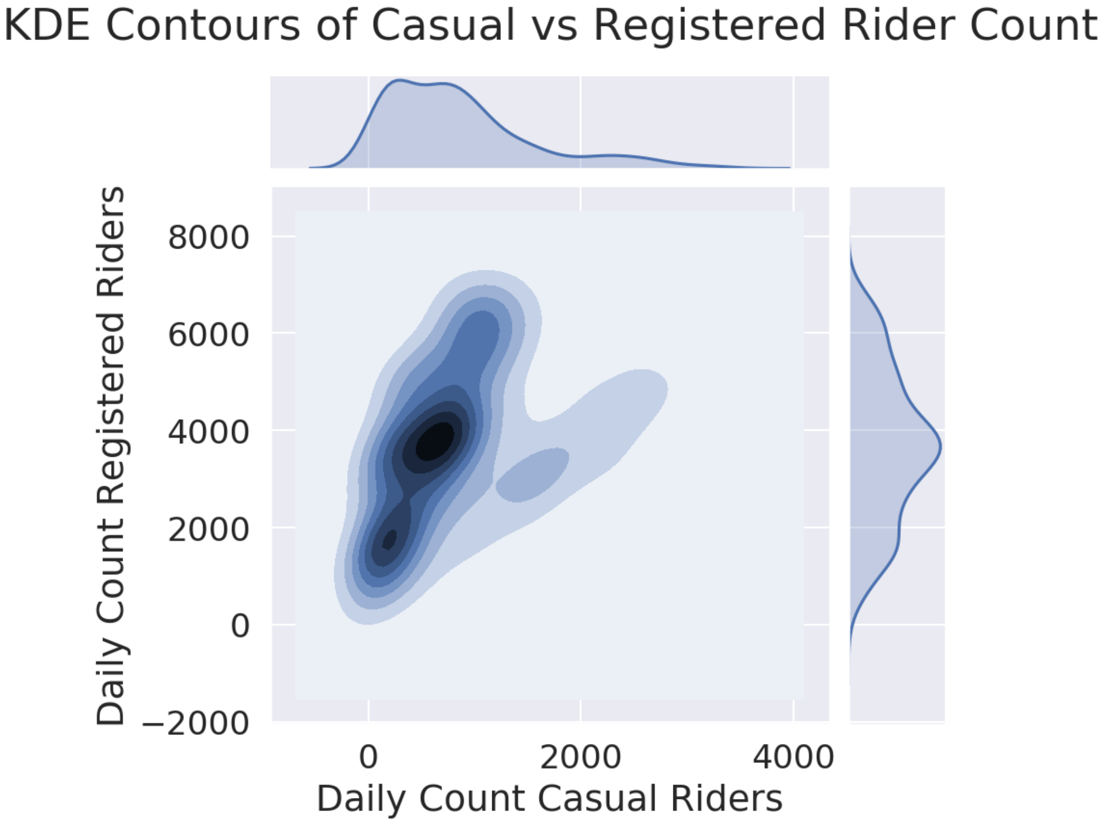
We noticed that in this contour plot, we can see that during the workday there is greatest concentration of approximately 1000 casual riders and 4000 registered riders that we were not able to notice before. We can similarly look at the non-workday and see that the greatest concentration for casual riders if about 1500 and for registered rider it's between 2000 and 4000 riders. Thus, we are able to so greater variability between the workday and nonworkday of the registered and casual riders.

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 [80]: joint_plot=sns.jointplot(x=daily_counts['casual'], y=daily_counts['registered'], kind='kde')
         joint_plot.set_axis_labels('Daily Count Casual Riders', 'Daily Count Registered Riders')
         plt.suptitle('KDE Contours of Casual vs Registered Rider Count')
         plt.subplots_adjust(top=0.9);
```



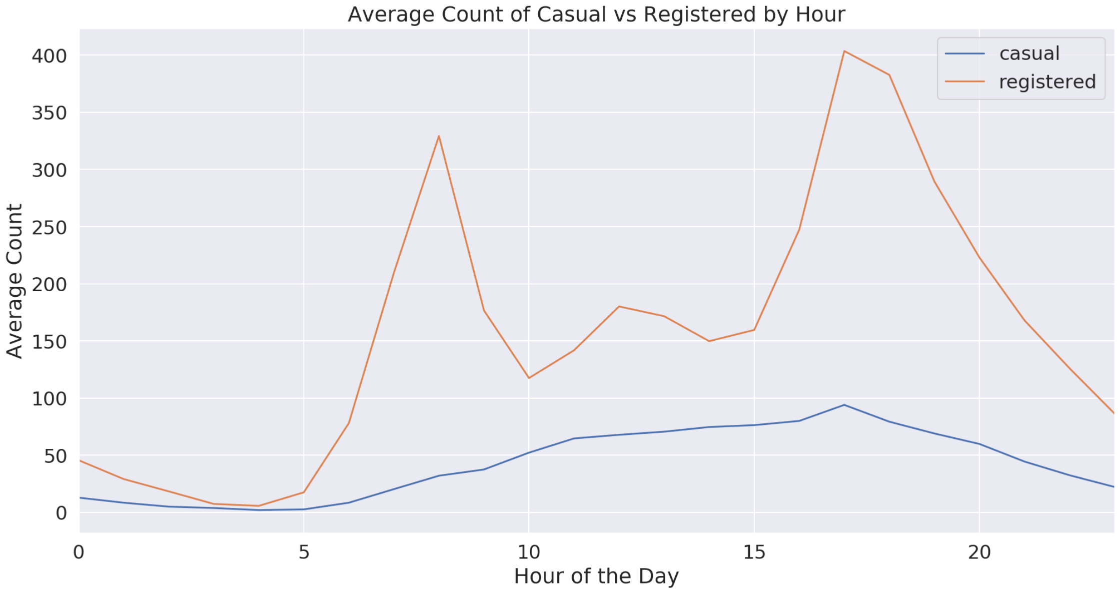
0.2 5: Understanding Daily Patterns

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 following:

```
In [81]: hours_of_the_day = bike[bike["dteday"].str.contains("2011-06-")][["hr","casual","registered","c
         hours_of_the_day.plot()
         plt.xlabel('Hour of the Day')
         plt.ylabel('Average Count')
         plt.title('Average Count of Casual vs Registered by Hour');
```



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

From this plot, we can see that for casual riders, they typically ride anytime between 10 AM to 7PM where there are not any massive jumps in the casual rider's distrubtion. However, in comparison we see that registered riders have peaks at around 8AM and 8PM and that can possibly be described by those who are registered use their bikes to commute to and from work.

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.

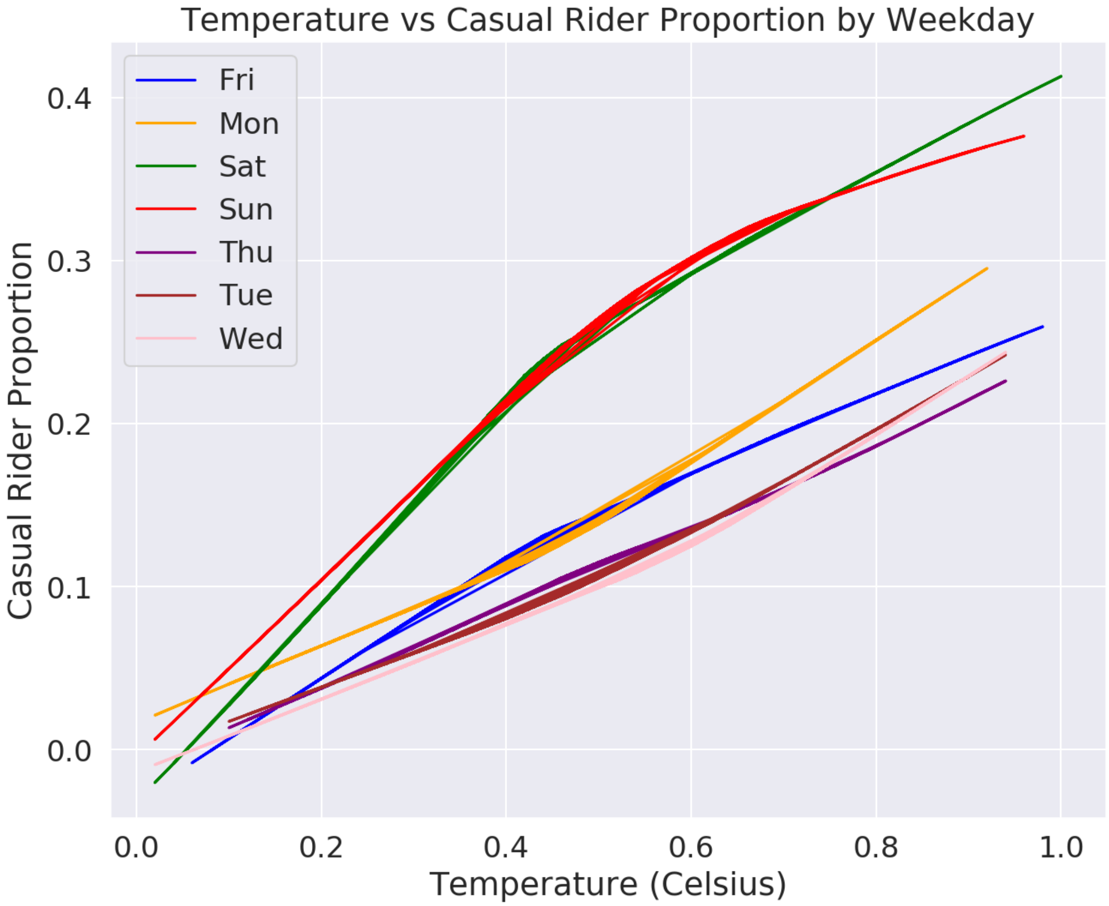
You should use `statsmodels.nonparametric.smoothers_lowess.lowess` just like the example above. Unlike the example above, plot ONLY the lowess curve. Do not plot the actual data, which would result in overplotting. For this problem, the simplest way is to use a loop.

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, Fahrenheit $=$ Celsius $* \frac{9}{5} + 32$.

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

```
In [87]: from statsmodels.nonparametric.smoothers_lowess import lowess

         plt.figure(figsize=(10,8))
         temp_weekday = bike[['prop_casual', 'temp', 'weekday']].groupby('weekday')
         colors=['blue', 'orange','green','red','purple','brown','pink']
         k=0

         for i,j in temp_weekday:
             ysmooth = lowess(j['prop_casual'], j['temp'], return_sorted=False)
             plt.plot(j['temp'], ysmooth, 'r-', color=colors[k], label=i)
             k+=1

         plt.xlabel('Temperature (Celsius)')
         plt.ylabel('Casual Rider Proportion')
         plt.title('Temperature vs Casual Rider Proportion by Weekday')
         plt.legend();
```


Temperature vs Casual Rider Proportion by Weekday
Fri
Mon
Sat
Sun
Thu
Tue
Wed
Casual Rider Proportion
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Temperature (Celsius)

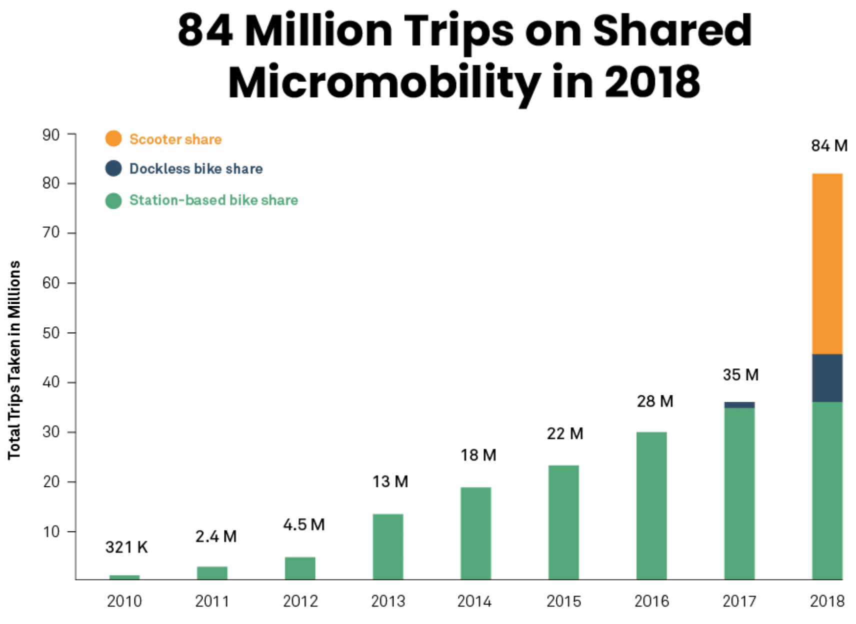
Question 6c What do you see from the curve plot? How is `prop_casual` changing as a function of temperature? Do you notice anything else interesting?

From the Cruve plot we see that the proportion of casual rider on Saturday and Sunday are higher than the rest of the weekdays. And we can see that prop_casual is generall increasing as the temperature increases. The interesting thing is that Saturday and Sunday sees an a bigger increase in the proportion of casual riders from 0.2 to 0.4 Celsius than the rest of the weekdays.

Question 6d Based on the data you have explored (distribution of orders, daily patterns, weather, additional data/information you have seen), do you think bike sharing should be realistically scaled across major cities in the the US in order to alleviate congestion, provide geographic connectivity, reduce carbon emissions, and promote inclusion among communities? Why or why not? Provide a visualisation and justify how it supports your answer

```
In [91]: #Yes, bike sharing should be realistically scaled
         #across major cities in the US in order to alleviate
         #congestion, connectivity, reduce carbon emissions,
         #and promote inclusion
         #because as we can see from ride sharing
         #and weather comparisons,
         #we see that as temperature increases then people
         #are much more likely to bike ride.
         #As we can see from our daily patterns plots,
         #it is seen that most riders tend to
         #ride in the mornings or the evenings,
         #which is typically when people are not working jobs.
         #Thus, if we can implement bike sharing during those times
         #when people are commuting to work then it will
         #reduce carbon emissions and alleviate some of
         #the congestion that is on our highways.
         #In the image I provided below, it illustrates that
         #scooter share has seen a huge influx of users in 2018,
         #which can be a sign that micromobility would
         #fill the demand for those who commute to and form work.

         from IPython.display import Image
         Image(filename='images/84-Million-Trips.png')
```

Out[91]:



Source: NACTO