

Notebook

July 9, 2019

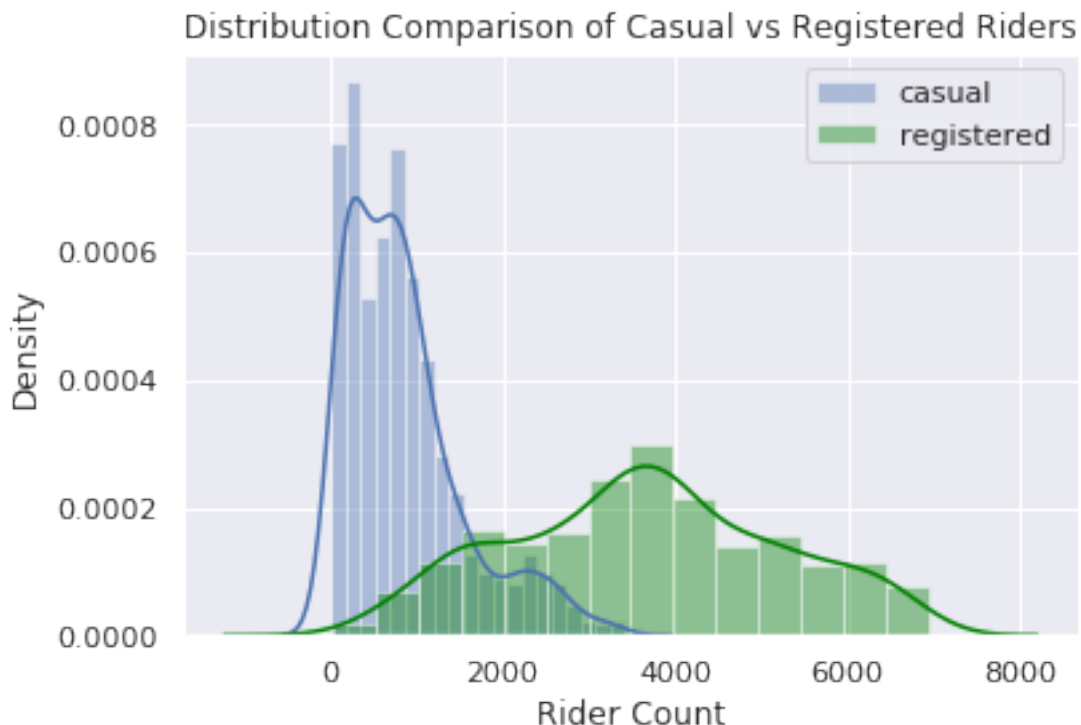
0.0.1 Question 2

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 [15]: plt.figure(figsize= (6,4))
sns.distplot(daily_counts['casual'],label = 'casual')
sns.distplot(daily_counts['registered'],label = 'registered',color = 'green')
plt.legend()
plt.xlabel('Rider Count')
plt.ylabel('Density')
plt.title('Distribution Comparison of Casual vs Registered Riders')
```

```
Out[15]: Text(0.5, 1.0, 'Distribution Comparison of Casual vs Registered Riders')
```



0.0.2 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 mode of casual riders is around 200. The distribution of casual riders is not symmetric but skews toward right, that is, has a right tails. There is a gap around 400 or 500 casual riders. And the distribution has some outliers around 3000 or 4000.

The mode of registered riders is around 4000. The distribution of registered riders is symmetric and has no skewness. There is no gap and has some outliers bwtween 1 to 500.

The distribution of registered riders is like normal distribution with the mean of rider count is around 4000. While the number of casual riders is relatively small than the number of registered riders. It is not like normal distribution because there is no negative count value. There is a gap in the distribution of casual riders maybe because of some riding events for riding amateurs.

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?

Identity, status, income are useful factors.

Privacy or consent issue maybe occur, for example, some people don't want to tell their status and income to others.

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?

Privacy or consent issue maybe occur, for example, some people don't want to tell their status and income to others.

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.

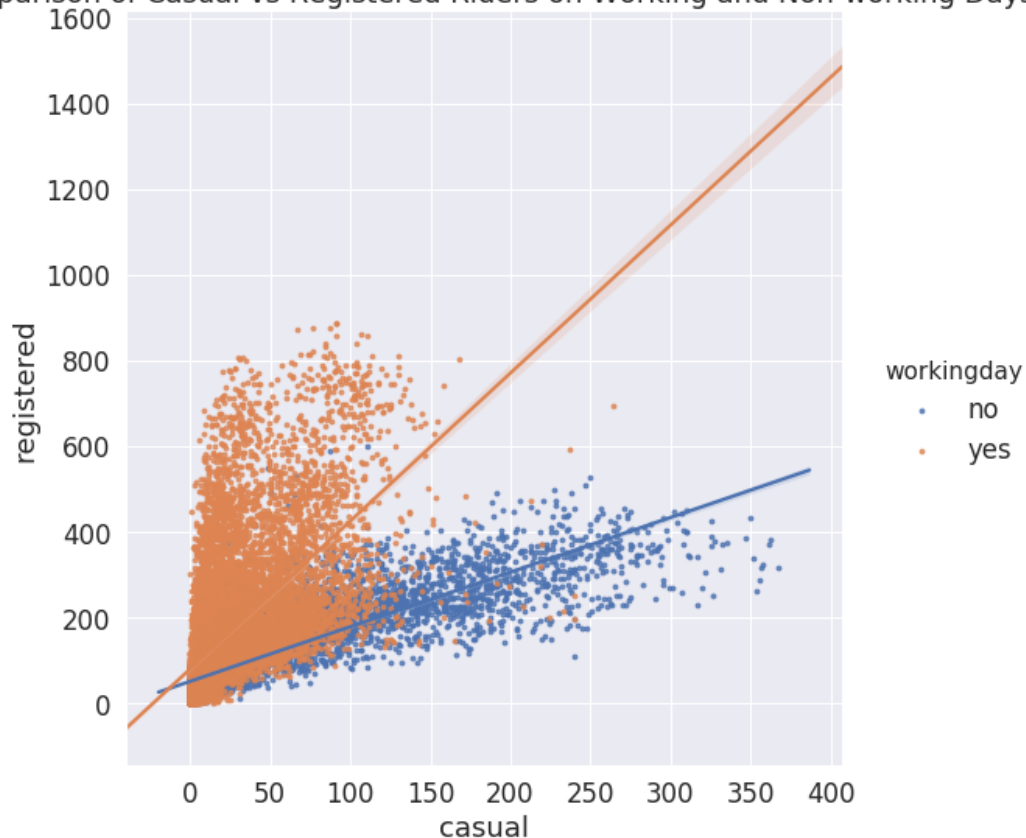
Hints: * Checkout this helpful [tutorial on lmplot](#).

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

```
In [16]: # Make the font size a bit bigger
sns.set(font_scale=1.5)
sns.lmplot(x="casual" , y="registered", data=bike,hue = 'workingday',fit_reg=True, height = 8,
plt.title('Comparison of Casual vs Registered Riders on Working and Non-working Days')
```

```
Out[16]: Text(0.5, 1, 'Comparison of Casual vs Registered Riders on Working and Non-working Days')
```

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?

Whether there are working days or not, the number of registered riders is proportionally related to the number of casual riders. That is to say, when the number of registered riders increases, the number of casual riders also increases. When there are working days, the number of casual riders is small and increases slowly. While if there is no working day, the number of casual riders increase relative faster.

There is overplotting issue in this graph, because the orange points are overlapping together that it is difficult to see how many points lie in that part, even the orange points block out the blue points so that we cannot see where these blues are.

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 [18]: plt.figure(figsize=(10,6))
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/registered)
casual_weekday = daily_counts['casual'].loc[is_workingday]
registered_weekday = daily_counts['registered'].loc[is_workingday]

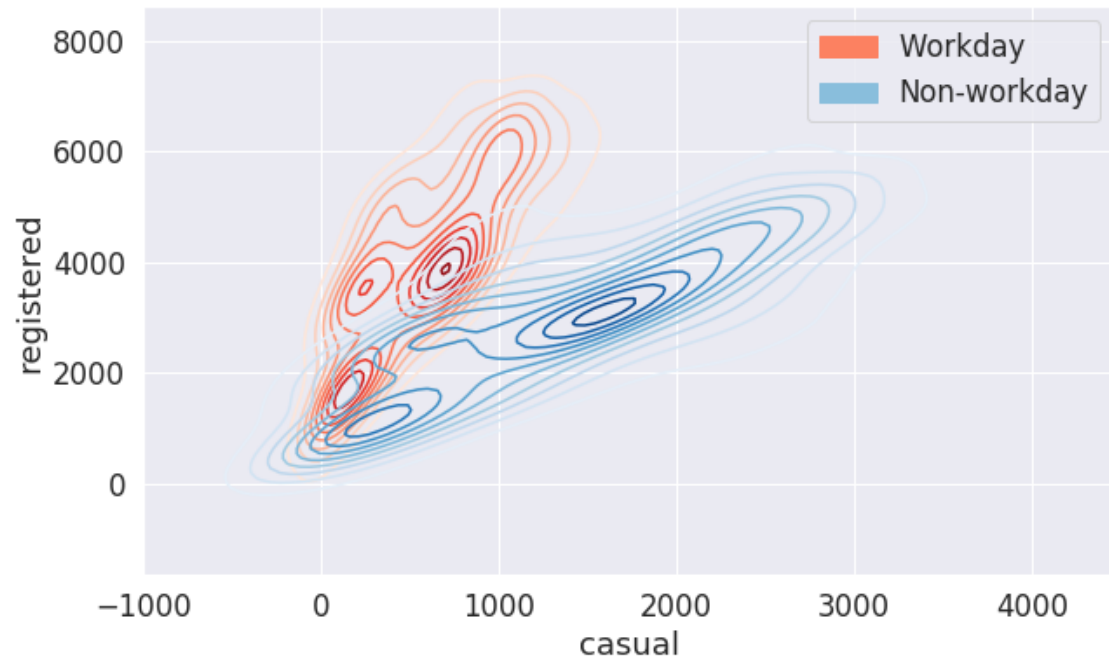
# Use sns.kdeplot on the two variables above to plot the bivariate KDE for weekday rides
sns.kdeplot(casual_weekday,registered_weekday,cmap = 'Reds')

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

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

# set legend
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])
```

```
Out[18]: <matplotlib.legend.Legend at 0x7fbe5bf3b518>
```



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

On workdays, in this graphs, much data are concentrated to (casual: 100, registered: 2000) and (casual: 800, registered: 4000), on non workdays, in this graphs, much data are concentrated to (casual: 300, registered:1000) and (casual: 1500, registered: 3000).

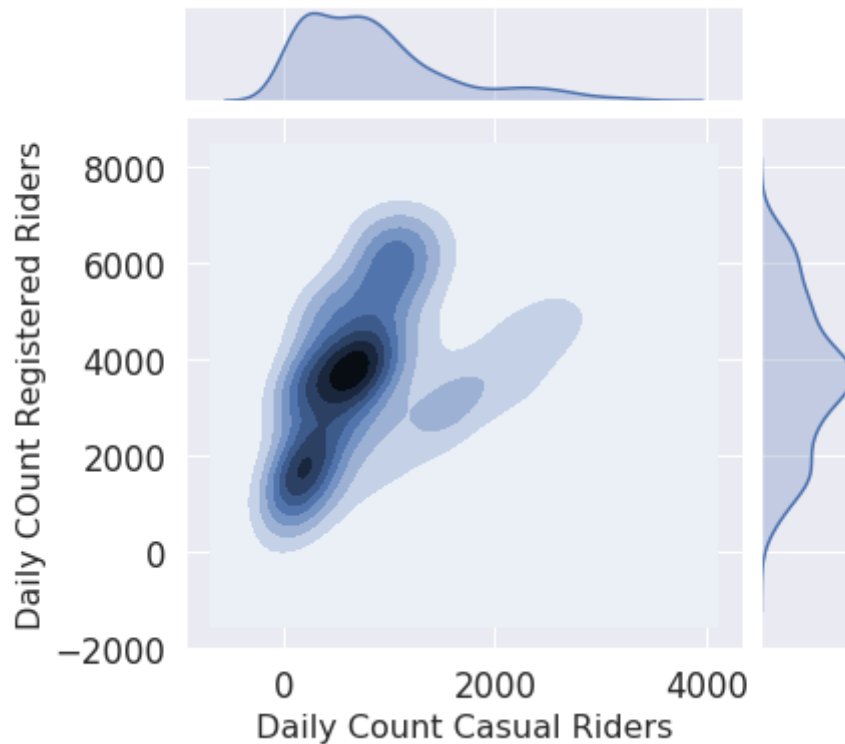
0.1 4: Joint Plot

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 [19]: h = sns.jointplot(x="casual", y="registered", data=daily_counts, kind="kde");  
         h.set_axis_labels('Daily Count Casual Riders', 'Daily Count Registered Riders', fontsize=16)  
         plt.suptitle('KDE Contours of Casual vs Registered Rider Count')  
         plt.subplots_adjust(top=0.9)
```

KDE Contours 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.

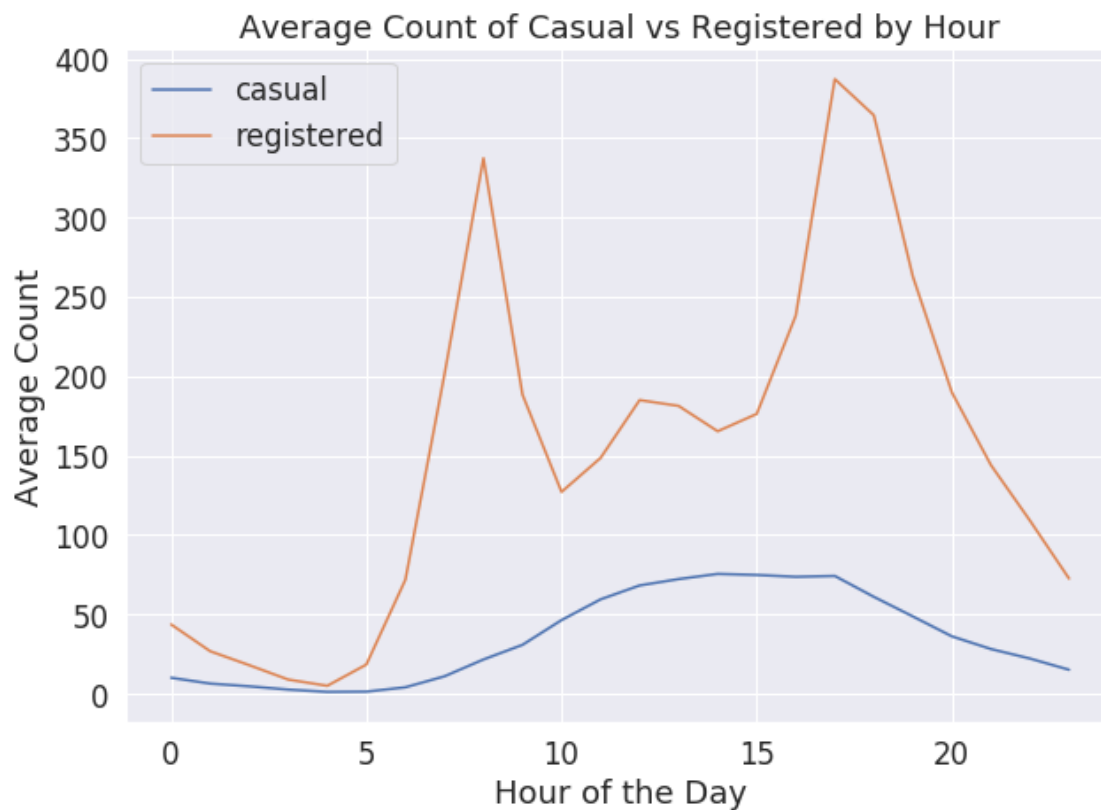
Your plot should look like the following:

```
In [20]: plt.figure(figsize = (10,7))
         newframe = bike.groupby(bike['hr']).agg({'casual':{'casual':'mean'},'registered':{'registered':
         newframe.columns = newframe.columns.droplevel(0)
         newframe.head()

         sns.lineplot(newframe.index,newframe['casual'],label = 'casual')
         sns.lineplot(newframe.index,newframe['registered'],label = 'registered')

         plt.legend()
         plt.xlabel('Hour of the Day')
         plt.ylabel('Average Count')
         plt.title('Average Count of Casual vs Registered by Hour')
```

```
Out[20]: Text(0.5, 1.0, '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.

I can observe the line of casual riders is relatively flat, when it is daytime, the average count of casual riders increase a little bit. The line of registered riders has two peaks at about 7-8 am and 5-6 pm in a day.

It seems that registered riders usually go to work and go back home by bike sharing, because peaks happens at 7-8 am and 5-6 pm which is exactly the rush hours.

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

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

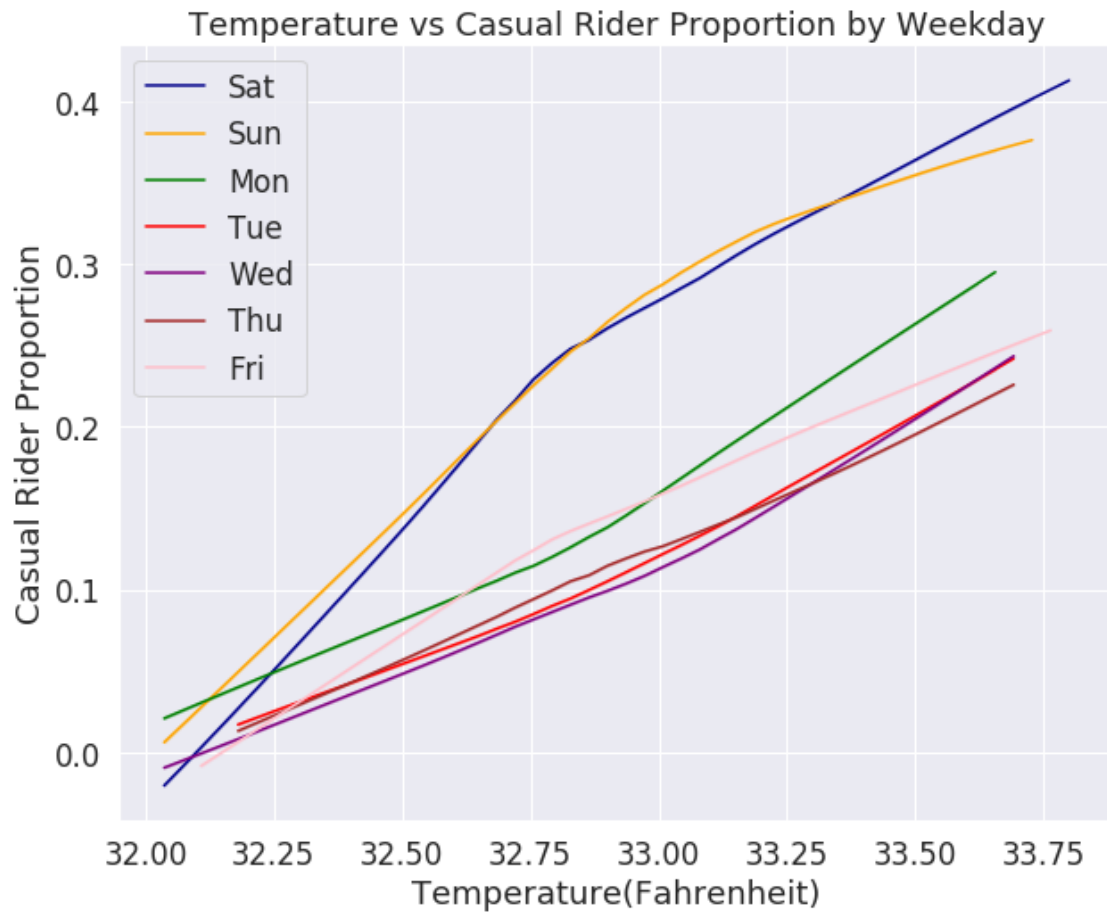
```
In [28]: from statsmodels.nonparametric.smoothers_lowess import lowess

plt.figure(figsize=(10,8))
bike = bike.assign(F_temp = lambda x: x.temp * 9/5 + 32)
weekdays = ['Sat', 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri']
colors = ['darkblue', 'orange', 'green', 'red', 'purple', 'brown', 'pink']

for i, j in zip(weekdays, colors):
    bikeselect = bike[bike['weekday'] == i]
    xo = bikeselect['F_temp']
    yo = bikeselect['prop_casual']
    ysmooth = lowess(yo, xo, return_sorted=False)
    sns.lineplot(xo, ysmooth, label=i, color=j)

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

```
Out[28]: Text(0.5, 1.0, '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?

There are more casual riders on Saturdays and Sundays than in others. Casual rider proportion is getting larger when the temperature is getting higher.

An interesting thing is that when the temperature is low, there seems little difference of casual rider proportion between each 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

I do think bike sharing should be scaled across major cities in the US.

Based on the data explored, we know that the number of registered riders is larger than the number of casual riders and there are much more registered riders in workdays. Also we can know that most of riders use bike sharing when going to work in the morning and going back home in the evening.

As the fact that there may be many people have to go to work from other places, in order to alleviate congestion, reduce carbon emissions, promote inclusion among communities, save time and money, it is necessary to scale the bike sharing across major cities in the US.