Homework 3: Bike Sharing

Exploratory Data Analysis (EDA) and Visualization

Due Date: Friday October 18, 11:59 PM

Course Policies

Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** at the top of your solution.

Introduction

This assignment includes both specific tasks to perform and open-ended questions to investigate. The open-ended questions ask you to think critically about how the plots you have created provide insight into the data.

After completing this assignment, you should be comfortable with:

- reading plaintext delimited data into `pandas`
- wrangling data for analysis
- using EDA to learn about your data
- making informative plots

Grading

For free response, readers will evaluate how well you answered the question and/or fulfilled the requirements of the question.

For plots, your plots should be *similar* to the given examples. We will tolerate small variations such as color differences or slight variations in scale. However it is in your best interest to make the plots as similar as possible as similarity is subject to the readers.

Note that for ALL plotting questions from here on out, we will expect appropriate titles, axis labels, legends, etc. The following question serves as a good guideline on what is "enough": If I directly downloaded the plot and viewed it, would I be able to tell what was being visualized without knowing the question?

Submission

For this assignment and future assignments (homework and projects) you will need to upload a single document to Gradescope. To do this, you can

1. download as HTML (`File->Export Notebook As->HTML (.html)`)
2. Print HTML to PDF (.pdf)
3. Upload to Gradescope -- tagging your responses.

You are responsible for submitting and tagging your answers in gradescope. For each free response and plotting question, please include:

1. Relevant code used to generate the plot or inform your insights
2. The written free response or plot

We are doing this to make it easier on our graders and for you, in the case you need to submit a regrade request. Gradescope (as of now) is still better for manual grading.

Score breakdown

Question	Points
Question 1a	2
Question 1b	1
Question 1c	2
Question 2a	2
Question 2b	2
Question 2c	2
Question 2d	2
Question 3a	4
Question 3b	3
Question 4a	2
Question 4b	2

Question	Points
Question 5a	1
Question 5b	4

In [2]:
```
# Run this cell to set up your notebook.  Make sure ds-ua-112_utils.py is in this assignment's folder
import seaborn as sns
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import zipfile
from IPython.display import Image
from pathlib import Path
import ds_ua_112_utils

# Default plot configurations
%matplotlib inline
plt.rcParams['figure.figsize'] = (16,8)
plt.rcParams['figure.dpi'] = 150
sns.set()

from IPython.display import display, Latex, Markdown
```

Loading Bike Sharing Data

The data we are exploring is data on bike sharing in Washington D.C.

The variables in this data frame are defined as:

Variable	Description
instant	record index
dteday	date
season	1. spring 2. summer 3. fall 4. winter
yr	year (0: 2011, 1:2012)
mnth	month (1 to 12)
hr	hour (0 to 23)
holiday	whether day is holiday or not
weekday	day of the week
workingday	if day is neither weekend nor holiday
weathersit	1. clear or partly cloudy 2. mist and clouds 3. light snow or rain 4. heavy rain or snow
temp	normalized temperature in Celsius (divided by 41)
atemp	normalized "feels-like" temperature in Celsius (divided by 50)
hum	normalized percent humidity (divided by 100)
windspeed	normalized wind speed (divided by 67)
casual	count of casual users
registered	count of registered users
cnt	count of total rental bikes including casual and registered

Download the Data

In [3]:
```
# Run this cell to download the data.  No further action is needed

data_url = 'https://cims.nyu.edu/~policast/bikeshare.zip'
file_name = 'data.zip'
data_dir = '.'

dest_path = ds_ua_112_utils.fetch_and_cache(data_url=data_url, data_dir=data_dir, file=file_name)
print('Saved at {}'.format(dest_path))

zipped_data = zipfile.ZipFile(dest_path, 'r')

data_dir = Path('data')
zipped_data.extractall(data_dir)


print("Extracted Files:")
for f in data_dir.glob("*"):
    print("\t",f)
```

```
Using version already downloaded: Mon Oct  7 20:53:27 2019
MD5 hash of file: fe4d1444ec7354d027a220f46dde42b1
Saved at data.zip
Extracted Files:
	 data/bikeshare.txt
	 data/data
```

Examining the file contents

Can you identify the file format? (No answer required.)

In [4]:
```
# Run this cell to look at the top of the file.  No further action is needed
for line in ds_ua_112_utils.head(data_dir/'bikeshare.txt'):
    print(line,end="")
```

```
instant,dteday,season,yr,mnth,hr,holiday,weekday,workingday,weathersit,temp,atemp,hum,windspeed,casual,regist
ered,cnt
1,2011-01-01,1,0,1,0,0,6,0,1,0.24,0.2879,0.81,0,3,13,16
2,2011-01-01,1,0,1,1,0,6,0,1,0.22,0.2727,0.8,0,8,32,40
3,2011-01-01,1,0,1,2,0,6,0,1,0.22,0.2727,0.8,0,5,27,32
4,2011-01-01,1,0,1,3,0,6,0,1,0.24,0.2879,0.75,0,3,10,13
```

Size

Is the file big? How many records do we expect to find? (No answers required.)

In [5]:
```
# Run this cell to view some metadata.  No further action is needed
print("Size:", (data_dir/"bikeshare.txt").stat().st_size, "bytes")
print("Line Count:", ds_ua_112_utils.line_count(data_dir/"bikeshare.txt"), "lines")
```

```
Size: 1156736 bytes
Line Count: 17380 lines
```

Loading the data

The following code loads the data into a Pandas DataFrame.

In [6]:
```
# Run this cell to load the data.  No further action is needed
bike = pd.read_csv(data_dir/'bikeshare.txt')
bike.head()
```

Out[6]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registere
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	1
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	3
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	2
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	1
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	

Below, we show the shape of the file. You should see that the size of the dataframe matches the number of lines in the file, minus the header row.

In [7]:
```
bike.shape
```

Out[7]:
```
(17379, 17)
```

1: Data Preparation

A few of the variables that are numeric/integer actually encode categorical data. These include `holiday`, `weekday`, `workingday`, and `weathersit`. In the following problem, we will convert these four variables to strings specifying the categories. In particular, use 3-letter labels (`Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, and `Sat`) for `weekday`. You may simply use `yes`/`no` for `holiday` and `workingday`.

In this exercise we will *mutate* the data frame, **overwriting the corresponding variables in the data frame.** However, our notebook will effectively document this in-place data transformation for future readers. Make sure to leave the underlying datafile `bikeshare.txt` unmodified.

Question 1

Question 1a (Decoding `weekday`, `workingday`, and `weathersit`)

Decode the `holiday`, `weekday`, `workingday`, and `weathersit` fields:

1. `holiday`: Convert to `yes` and `no`. Hint: There are fewer holidays...
2. `weekday`: It turns out that Monday is the day with the most holidays. Mutate the `'weekday'` column to use the 3-letter label (`'Sun'`, `'Mon'`, `'Tue'`, `'Wed'`, `'Thu'`, `'Fri'`, and `'Sat'` ...) instead of its current numerical values. Assume `0` corresponds to `Sun`, `1` to `Mon` and so on.
3. `workingday`: Convert to `yes` and `no`.
4. `weathersit`: You should replace each value with one of `Clear`, `Mist`, `Light`, or `Heavy`.

Note if you want to revert the changes run the cell that reloads the csv.

Hint: One approach is to use the replace method of the pandas DataFrame class. We haven't discussed how to do this so you'll need to look at the documentation. The most concise way is with the approach described in the documentation as "nested-dictonaries", though there are many possible solutions.

In [8]:
```
# Modify holiday weekday, workingday, and weathersit here
days = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
day_indices = range(7)
indices_to_days_dict = dict(zip(day_indices, days))

validate_indices = range(2)
validate_list = ["no", "yes"]
validate = dict(zip(validate_indices, validate_list))

weather_indices = range(1,5)
weather_list = ["Clear", "Mist", "Light", "Heavy"]
weather = dict(zip(weather_indices, weather_list))

bike["weekday"] = bike["weekday"].map(indices_to_days_dict)
bike['holiday'] = bike['holiday'].map(validate)
bike['workingday'] = bike['workingday'].map(validate)
bike['weathersit'] = bike['weathersit'].map(weather)


# bike["weekday"].replace(bike["weekday"].map(indices_to_days_dict))
# bike["holiday"].replace(bike['holiday'].map(validate))
# bike["workingday"].replace(bike['workingday'].map(validate))
# bike["weathersit"].replace(bike['weathersit'].map(weather))


#  Hint: one strategy involves df.replace(...)

### BEGIN SOLUTION
#TODO
### END SOLUTION
```

In [9]:
```
assert isinstance(bike, pd.DataFrame)
assert bike['holiday'].dtype == np.dtype('O')
assert list(bike['holiday'].iloc[370:375]) == ['no', 'no', 'yes', 'yes', 'yes']
assert bike['weekday'].dtype == np.dtype('O')
assert bike['workingday'].dtype == np.dtype('O')
assert bike['weathersit'].dtype == np.dtype('O')

# Hidden tests
assert bike.shape == (17379, 17) or bike.shape == (17379, 18)
assert list(bike['weekday'].iloc[::2000]) == ['Sat', 'Tue', 'Mon', 'Mon', 'Mon', 'Sun', 'Sun', 'Sat', 'Sun']
assert list(bike['workingday'].iloc[::2000]) == ['no', 'yes', 'yes', 'yes', 'yes', 'no', 'no', 'no', 'no']
assert list(bike['weathersit'].iloc[::2000]) == ['Clear', 'Clear', 'Clear', 'Clear', 'Clear', 'Clear', 'Clea
r', 'Clear', 'Clear']
print(True)
```

```
True
```

Question 1b (Holidays)

How many entries in the data correspond to holidays? Set the variable `num_holidays` to this value.

In [34]:
```
num_holidays = bike['holiday'].value_counts()['yes']
print(num_holidays, 'entries in the data correspond to holidays')
# len([entry for entry in bike['holiday'].values() if bike['holiday'] == 'yes'])
#.filter(lambda x: x == 'yes')



### BEGIN SOLUTION
#TODO
### END SOLUTION
```

```
500 entries in the data correspond to holidays
```

In [11]:
```
assert 400 <= num_holidays <= 550

### BEGIN HIDDEN TESTS
assert num_holidays == 500
### END HIDDEN TESTS

print(True)
```

```
True
```

Question 1c (Computing Daily Total Counts)

The granularity of this data is at the hourly level. However, for some of the analysis we will also want to compute daily statistics. In particular, in the next few questions we will be analyzing the daily number of registered and unregistered users.

Construct a data frame with the following columns:

- `casual` : total number of casual riders for each day
- `registered` : total number of registered riders for each day
- `workingday` : whether that day is a working day or not (`yes` or `no`)

Hint: `groupby` and `agg` . For the `agg` method, please check the documentation (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.core.groupby.DataFrameGroupBy.agg.html) for examples on applying different aggregations per column. If you use the capability to do different aggregations by column, you can do this task with a single call to `groupby` and `agg` . For the `workingday` column we can take any of the values since we are grouping by the day, thus the value will be the same within each group. Take a look at the `'first'` or `'last'` aggregation functions.

In [12]:
```
daily_counts = bike.groupby('dteday').agg({'casual': sum, 'registered': sum, 'workingday': 'last'})

### BEGIN SOLUTION
#TODO
### END SOLUTION
```

In [13]:
```
assert np.round(daily_counts['casual'].mean()) == 848.0
assert np.round(daily_counts['casual'].var()) == 471450.0

### BEGIN HIDDEN TESTS
assert np.round(daily_counts['registered'].mean()) == 3656.0
assert np.round(daily_counts['registered'].var()) == 2434400.0
assert sorted(list(daily_counts['workingday'].value_counts())) == [231, 500]
print(True)
### END HIDDEN TESTS
```

```
True
```

2: Exploring the Distribution of Riders

Let's begin by comparing the distribution of the daily counts of casual and registered riders.

Question 2

Question 2a

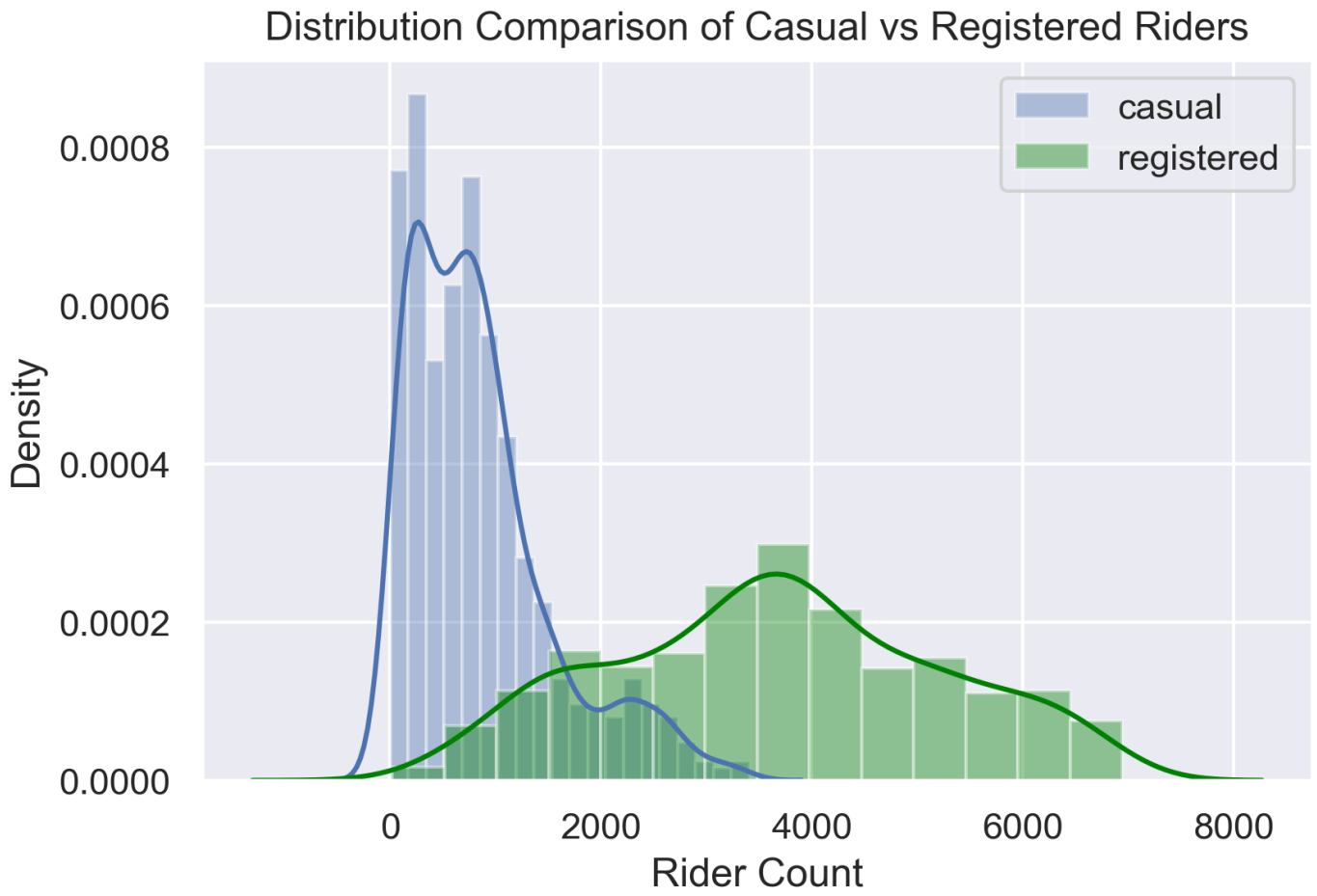
Use the `sns.distplot` (https://seaborn.pydata.org/generated/seaborn.distplot.html) function to create a plot that overlays the distribution of the daily counts of `casual` and `registered` users. The temporal granularity of the records should be daily counts, which you should have after completing question 1c.

Include a legend, xlabel, ylabel, and title. You may want to look at the seaborn plotting tutorial (https://seaborn.pydata.org/tutorial/distributions.html) 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 anywher from almost none to 7000.

IMAGE

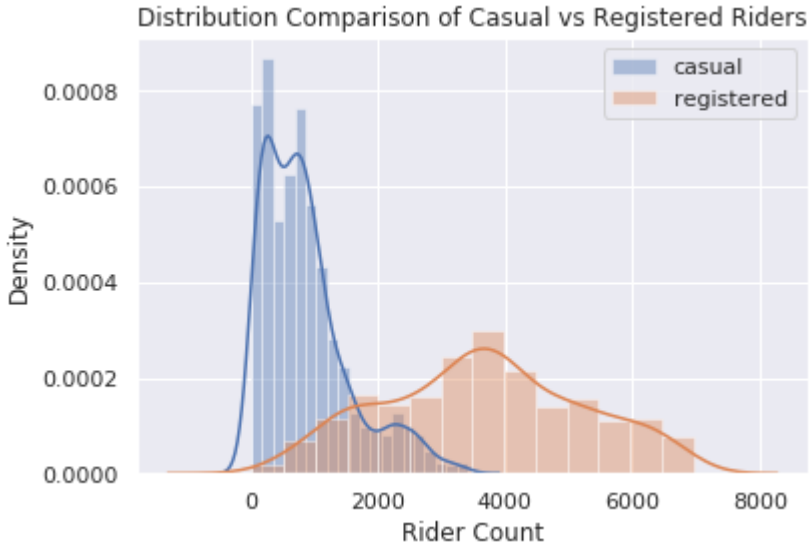
In [14]:
```
Image(filename='images/casual_v_registered.png', embed=True, height=10, width=700)
```

Out[14]:



In [15]:
```
#Template:

# vals_of_plt = bike['dteday'].loc[['casual', 'registered']]
# casual_riders = vals_of_plt.loc[vals_of_plt.index == 'causal']
# registered_riders = vals_of_plt.loc[vals_of_plt.index == 'registered']
# sns.distplot(casual_riders , label = 'casual', hist = True)
# sns.distplot(registered_riders, label = 'registered', hist = True)

casual_riders = daily_counts['casual']
registered_riders = daily_counts['registered']
sns.distplot(casual_riders, label = 'casual', hist = True)
sns.distplot(registered_riders, label = 'registered', hist = True)
plt.ylabel('Density')
plt.xlabel('Rider Count')
plt.legend()
plt.title("Distribution Comparison of Casual vs Registered Riders")



### BEGIN SOLUTION
#TODO
### END SOLUTION
```

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



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.

In [16]:
```
q2b = ''' There are differences I notice between the density curves for casual and registered riders.
In terms of modes, symmetry, skewness, tails, gaps and outliers: For the casual riders, the mode would be 200
riders (density of
.00010). There doesn't seem to be a symmetry, as there is a right skew. The tail is where the right skew occu
rs, and it mainly
shows around the 3500-4000 rider count section of the data. There doesn't seem to be any gaps, however there
 is a dip in the
data at around the 500-700 rider count. In the data set there seems to be an outlier, as there is a sudden de
crease at around
500 riders. However, there can be near symmetry in terms of the terms of the two high data values at 200 ride
rs and at nearly
1000 riders. It is also interesting that the density for the 0-100 and the 800-900 rider count are near the s
ame.

For the registered riders, the mode would be 3800-4000 riders (density of ~.0003). There is a symmetry, meani
ng the data
is unifromely distributed, so there is no skew. The tail is where the right portion of the data resides, at a
round the
6000-7500 rider count section of the data. There doesn't seem to be any gaps, and there are no outliers, as t
here are no data
points (or histogram categories) that constitute outlier status.

In terms of the spread of the distributions: The casual rider data seems to be spread mainly towards the left
area (0-1750
rider count), meaning that it is right skewed, and again, not uniformly distributed, or symmetric, concentrat
ed mainly on the
left side of the graph. The data decreases after approximately 1750 riders, where there is a tail. In the cas
e of density, the
data shows that there are higher densities of lower amounts of riders.

For the registered riders: As stated before, there is a symmetry, meaning the data is unifromely distributed,
and no skew.
The tail is where the right portion of the data resides, at around the 6000-7500 rider count section of the d
ata.
There doesn't seem to be any gaps, and there are no outlier, as there are no data points (or histogram catego
ries)
that constitute outlier status. However, what can be said about the spread of the data is that most of the da
ta resides around
the 3500-4250 rider count range. From 0-3500 and 4250-7500, there seems to be about a density of 0.0000-0.000
18. So this would
mean that the data shows a bell-curve shape, where most of the spread of data is centered around the middle.
'''
```

```
### BEGIN SOLUTION
#TODO
### END SOLUTION
```

Question 2c

The density plots do not show us how the daily counts for registered and casual riders vary together. Use `sns.lmplot` (https://seaborn.pydata.org/generated/seaborn.lmplot.html) to make a scatter plot to investigate the relationship between casual and registered 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 with over plotting. 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.

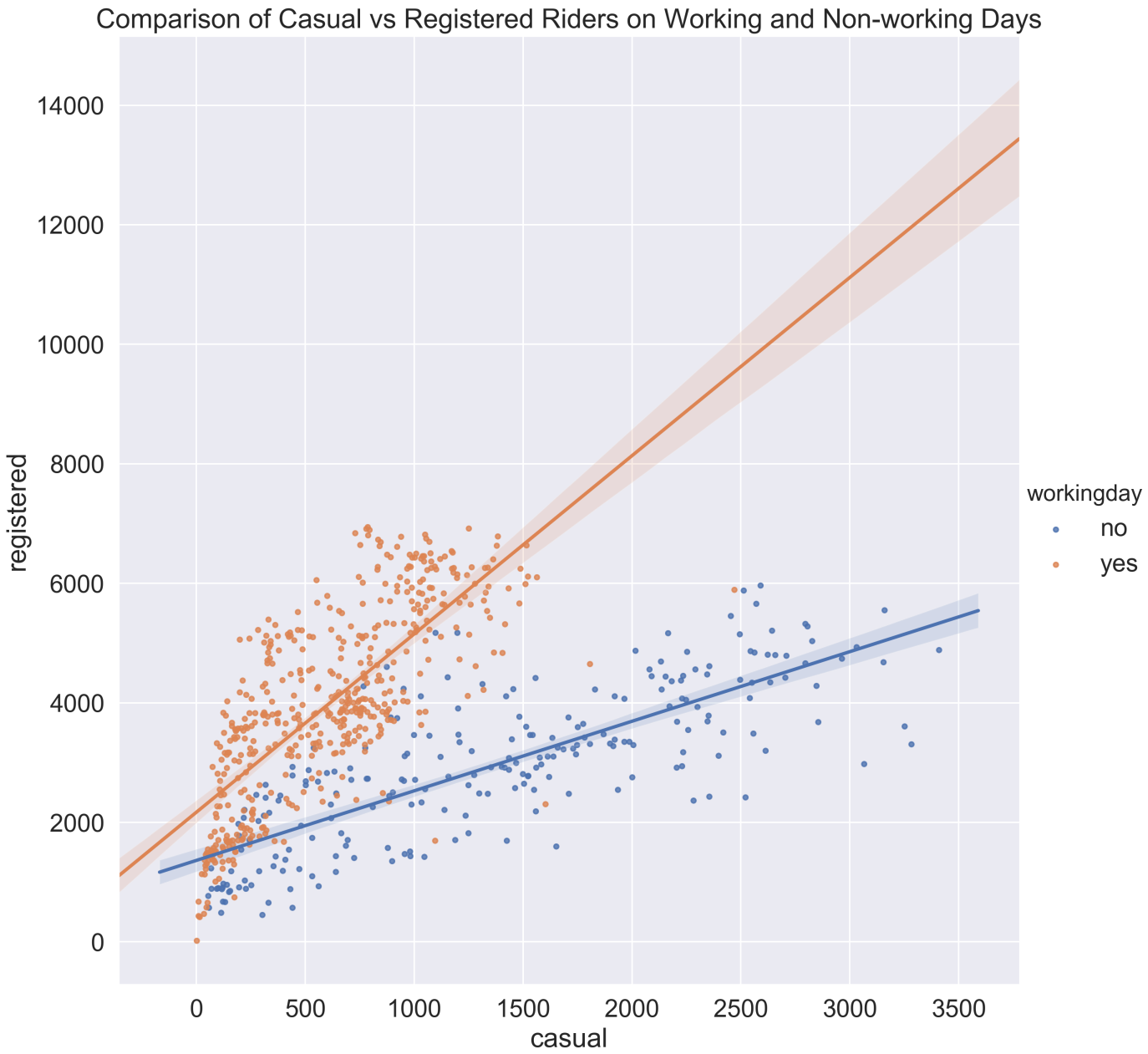
Hints:

- Checkout this helpful tutorial on `lmplot` (https://seaborn.pydata.org/tutorial/regression.html).
- You will need to set `x`, `y`, and `hue` and the `scatter_kws`.

IMAGE

In [17]:
```
Image(filename='images/casual_registered_working_nonworking.png', embed=True, height=10, width=700)
```

Out[17]:



In [18]:
```
# Make the font size a bit bigger

sns.lmplot(x='casual',y='registered',data=daily_counts, hue='workingday',scatter_kws={'s':5},height=10,fit_reg=True)
plt.ylabel('Registered')
plt.xlabel('casual')
plt.legend()
plt.title("Comparison of Casual vs Registered Riders on Working Days vs Non-Working Days")
sns.set(font_scale=1.5)


### BEGIN SOLUTION
#TODO
### END SOLUTION
```


Comparison of Casual vs Registered Riders on Working Days vs Non-Working Days
Registered
casual
0
2000
4000
6000
8000
10000
12000
14000
0
500
1000
1500
2000
2500
3000
3500
no
yes
workingday
no
yes

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?

Why might we be concerned with overplotting in examining this relationship? By "overplotting", we're referring to the term used in chapter 6.5 of the textbook (http://www.textbook.ds100.org/ch/06/viz_principles_2.html).

In [19]:
```
q2d = '''
This scatterplot seems to reveal the relationship (if any) between casual and registered riders and whether o
r not the day
is on the weekend becuase both the pairing of casual riders data points and the registered riders data points
indicates a positiv
linear correlation trend, such that with an increase in casual riders, there is also an increase in registere
d riders.
Based on the workingday, it seems that the data is more spread out, making for more casual riders, but less r
egistered riders.
This is probably due to the holidays yielding more hobby-based cyclists or biking for leisure or sport. This
 means less
registered bicylists for the bike sharing/rental. However, if it is a workingday, then there must be a lot of
registered bikes
because people have to go to work, or travel places.

The reason why might we be concerned with overplotting in examining this relationship is because not only wil
l it be difficult to
read the graph in terms of distribution, but there are many marks that make it difficult to tell where the da
ta lie,
so some of the points overlap, making it impossible to see how many points lie at the designated values. We m
ight want to try
smoothing, which is replacing sets of points with appropriate markers, but not to show every single point in
 the dataset
in order to reveal broader trends. In the situation of bike rentals, we would want to distinguish the data po
ints to see where
the data lies, and reveal the relationship (if any) between casual and registered riders and whether or not t
he day
is on the weekend.


'''

### BEGIN SOLUTION
#TODO
### END SOLUTION
```

A basic kde plot of all the data is quite easy to generate. However, this plot includes both weekend and weekday data, which isn't what we want (see example figure above).

3: Exploring Ride Sharing and Time

Question 3

Question 3a

Plot number of riders for each day in the month of June in 2011.

Make sure to add descriptive x-axis and y-axis labels and create a legend to distinguish the line for casual riders and the line for registered riders. The end result should look like the figure below. The shaded region is a bootstrap confidence interval similar to what you learned about in Data 8.

Make sure to include xlabel, ylabel, a legend, and a title.

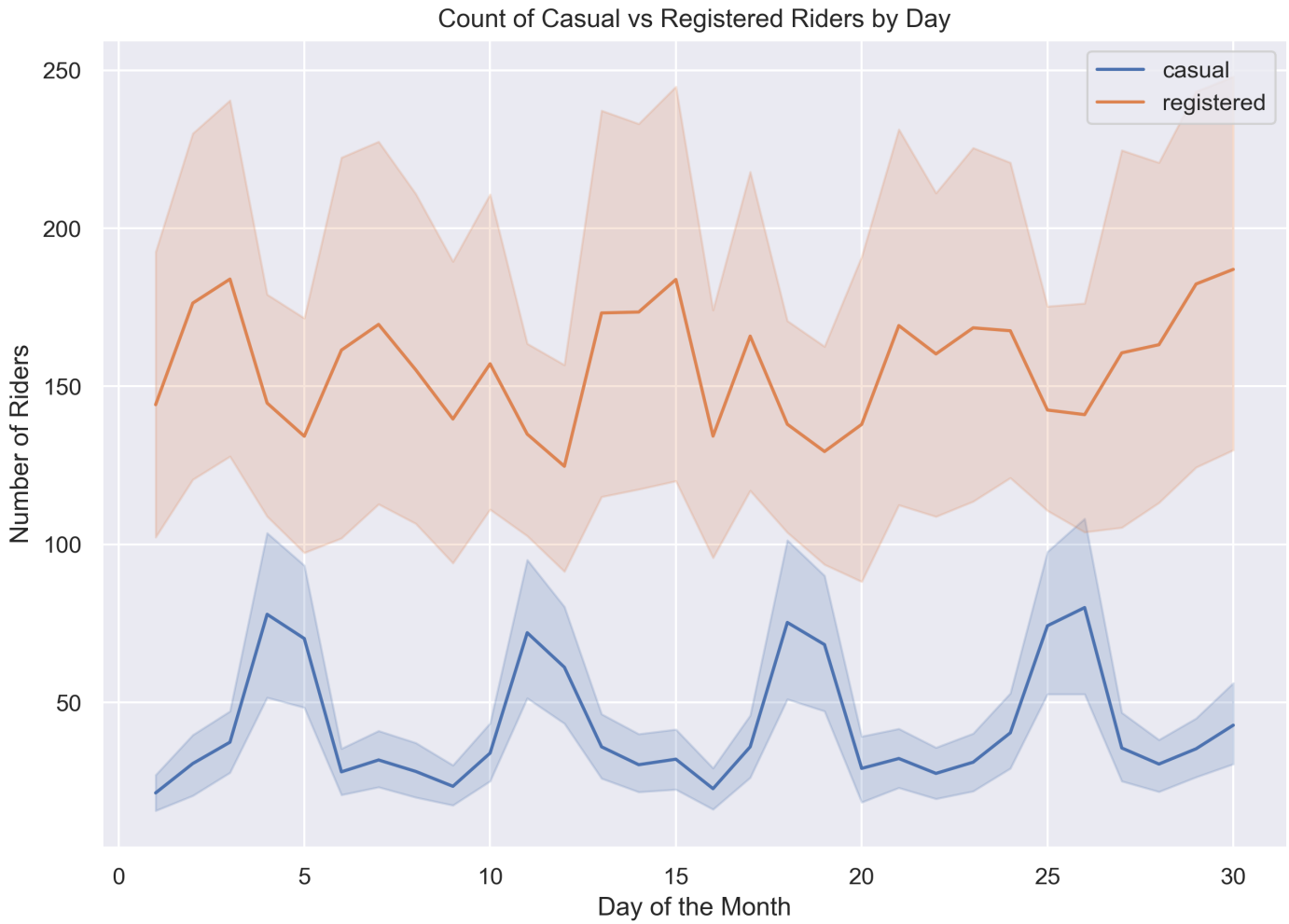
Hints:

- Add a new Series to the `bike` datafame correpsonding to the day. You can do something similar to what you did in `hw1` when you created the `postal_code_5` Seres.
- Make sure your day series is of type int. One way is to use the `map` method of the Series class, i.e. `s.map(int)` .
- Use `sns.lineplot` .

IMAGE

In [20]:
```
Image(filename='images/june_riders.png', embed=True, height=10, width=700)
```

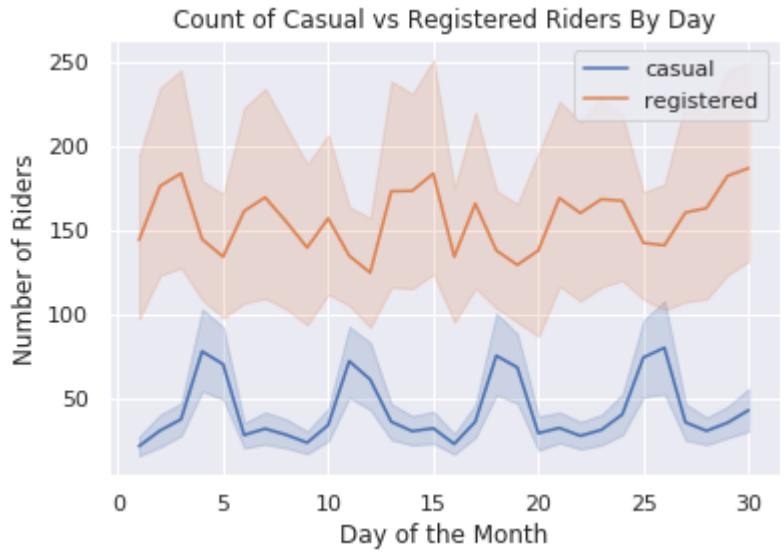
Out[20]:



In [21]:
```
import seaborn as sns; sns.set();
```

In [22]:
```
bike['Day'] = bike['dteday'].loc[(bike['yr']==0) & (bike['mnth']==6)].str[8:10].map(int)


sns.lineplot(x='Day',y ='casual', data=bike, label='casual')
sns.lineplot(x='Day',y ='registered', data=bike, label='registered')
plt.ylabel('Number of Riders')
plt.xlabel('Day of the Month')
plt.title("Count of Casual vs Registered Riders By Day")
plt.legend()


### BEGIN SOLUTION
#TODO
### END SOLUTION
```

Out[22]: <matplotlib.legend.Legend at 0x2ace43ad5b00>



Question 3b

This plot has several interesting features. How do the number of casual and registered riders compare for different days of the month? What is an interesting trend and pattern you notice between the lines? Why do you think the confidence interval for the registered riders is, on average, wider than the confidence interval for casual riders?

In [23]:
```
q4b = '''This plot has several interesting features. The number of casual and registered riders compare for
different days of the month. For different days of the month, the number of casual riders seems to be at its
 highest around the
time of the Thurs-Sun time period, while the number of registered riders seems to be higher on the weekdays
 (Mon-Thurs).

An interesting trend and pattern I notice between the lines is that The line for the casual rider number seem
s to show a
consistent pattern of increase during Thurs-Sun then decrease during Mon-Thurs throughout the week, while the
re is more trend
of fluctuation in the line representing the registered riders, where there is constant increase and decrease
 over a shorter
period of time.

The confidence interval for the registered riders is, on average, wider than the confidence interval for casu
al riders because
there is more of a fluctuation in the number of registered rider bike rentals than the number of casual rider
bike rentals since
it is easier to predict the amount of bikes rented for the week in terms of casual rider bike rentals, and be
cause the population
of registered riders is much higher. Due to an inconsistent fluctuation in the amount of bike rentals for the
registered riders,
there is a higher confidence interval, meaning the registered rental amount value ranges are wider for each w
eek. The rental
values span a wider range than the amounts for casual riders. However, it might sometimes be harder to determ
ine how many casual riders are going to use the
bike if it's a weekend or a holiday.

It's possible that for a registered rider to be someone registers as a member and downloads an app on therr p
hone, logging in
every time they use the bike, hence that is why there is a higher population of them. Whereas for casual ride
rs, maybe there’s
an option like login in as guest, no need to register. Although, I expect the trend to decrease because Plus,
people could
commute on a bike one week, then just use a car afterwards. Also, if the bikes are inefficient for commuting
 to work, people
may cancel the service. Hence for the casual riders, it's less of a problem, since they'll not care about the
size, quality,
or shape of the bike.


'''
```

```
### BEGIN SOLUTION
#TODO
### END SOLUTION
```

4: Understanding Daily Patterns

Question 4

Question 4a

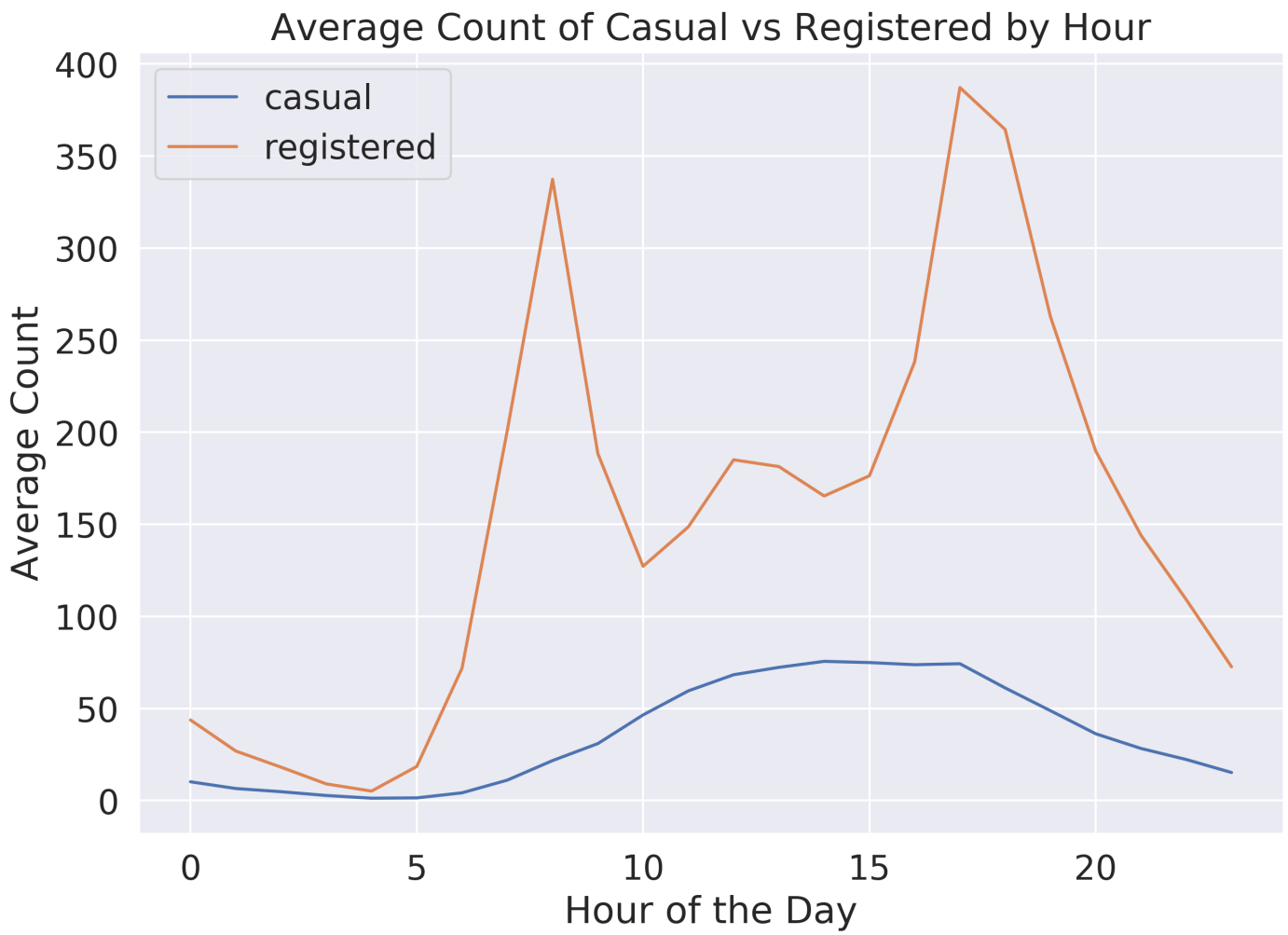
Let's examine the behavior of riders by plotting the average number of riders for each hour of the day over the **entire dataset** (not just June 2011), stratified by rider type.

Your plot should look like the following:

IMAGE

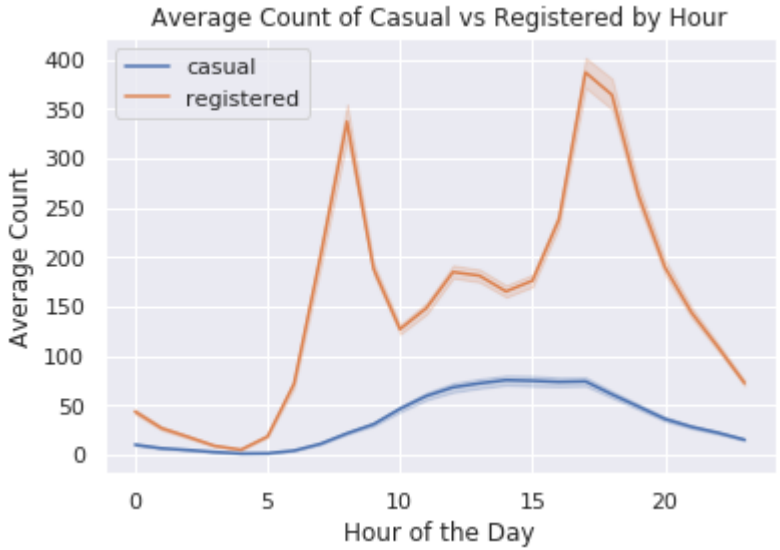
In [24]:
```
Image(filename='images/diurnal_bikes.png', embed=True, height=10, width=700)
```

Out[24]:



In [25]:
```
sns.lineplot(x='hr',y ='casual', data=bike, label='casual')
sns.lineplot(x='hr',y ='registered', data=bike, label='registered')
plt.ylabel('Average Count')
plt.xlabel('Hour of the Day')
plt.title("Average Count of Casual vs Registered by Hour")
plt.legend()
### BEGIN SOLUTION
#TODO
### END SOLUTION
```

Out[25]: <matplotlib.legend.Legend at 0x2ace4343c198>



Question 4b

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

In [26]:
```
q5b = ''' What I observed from the plot is that the trend of average count of bike rentals for casual riders
 seems to show a
slight incline, peaking at the 10:00 am to 5:00 pm time period. This means that the number of bike rentals du
ring each hour of
the day is low in volume, the highest amount being around 75-80 riders during that peak time of 10:00 am to
 5:00 pm.

In terms of the trend of average count of bike rentals for registered riders, it seems that during the day, t
here are many more
bike rentals in total for each hour than for the casual riders. The trend shows a steady incline then declin
e, peaking at the
8:00 am and 5:00 pm time periods. in between both of the times, there is a steady decline, as well as a decli
ne after 5:00 pm.

I hypothesize that the meaning of the peaks in the registered riders' distribution is that there are very few
bike rentals for
casual riders, mainly because people rent the bike for leisure, sport, or exercise. It's most likely that not
many people will
be renting the bikes, no matter if it's a weekday or a weekend. It's very possible that since the amount of r
entals for casual
riders is very low during the weekdays, that brought the averages down for the casual riders.

For the registered riders, it makes sense that ~330 riders at 8:00 am and ~380 riders at 5:00 pm rent bikes a
t those times. This
can be explained as people needing to go to work, and they are in a rush to arrive to work. This peak in the
 registered rider's
distribution shows that most registered riders rent bikes in the early morning or near evening, due to workin
g schedules.


'''

### BEGIN SOLUTION
#TODO
### END SOLUTION
```

5: Exploring Ride Sharing and Weather

Now let's examine how the weather is affecting rider's behavior. First let's look at how the proportion of casual riders changes as weather changes.

Question 5

Question 5a

Create a new column `prop_casual` in the `bike` dataframe representing the proportion of casual riders out of all riders.

In [27]:
```
bike['prop_casual'] = bike['casual'] / bike['cnt']

### BEGIN SOLUTION
#TODO
### END SOLUTION
```

In [28]:
```
assert int(bike["prop_casual"].sum()) == 2991

### BEGIN HIDDEN TESTS
assert np.round(bike["prop_casual"].mean(), 2) == 0.17

print(True)
### END HIDDEN TESTS
```

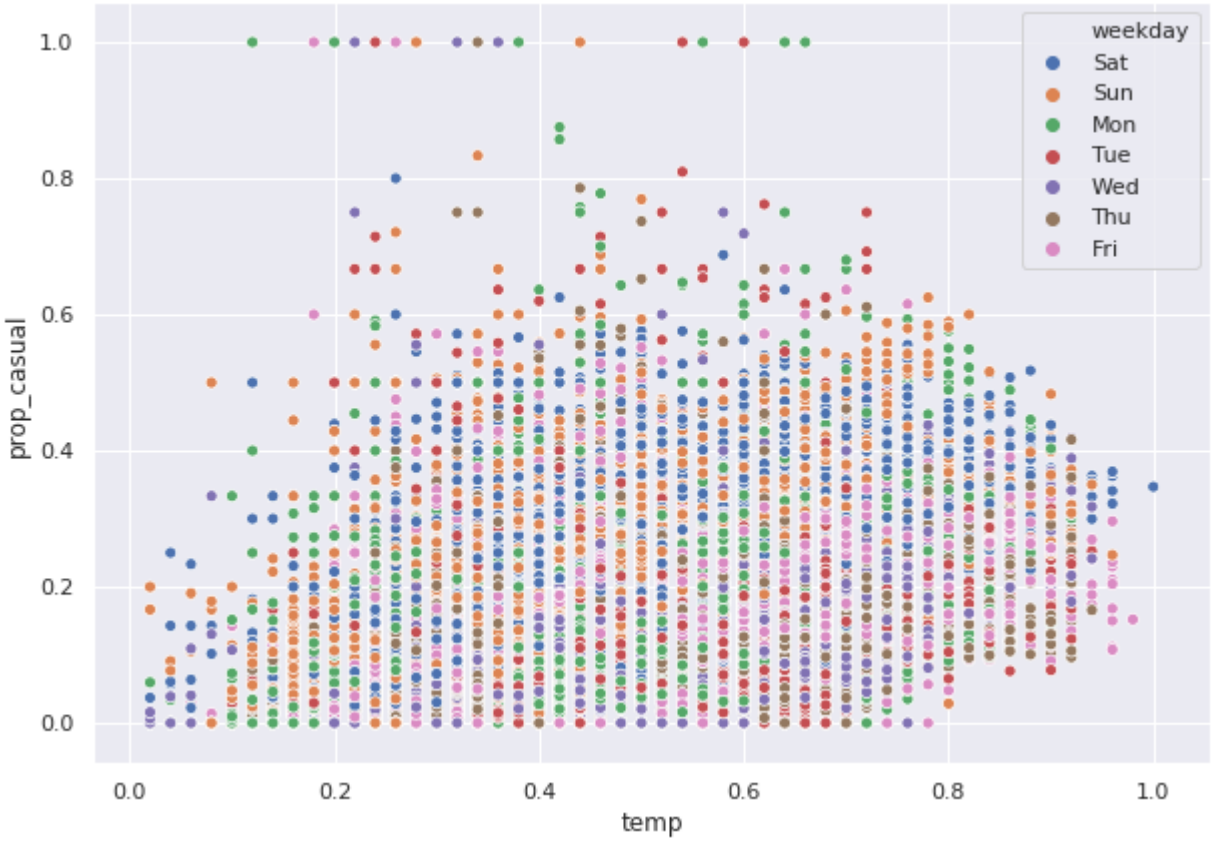
```
True
```

In [ ]:
```

```

Question 5b

In order to examine the relationship between proportion of casual riders and temperature, we can create a scatterplot using `sns.scatterplot`. We can even use color/hue to encode the information about day of week. Run the cell below, and you'll see we end up with a big mess that is impossible to interpret.

In [29]:
```
plt.figure(figsize=(10, 7))
sns.scatterplot(data=bike, x="temp", y="prop_casual", hue="weekday");
```

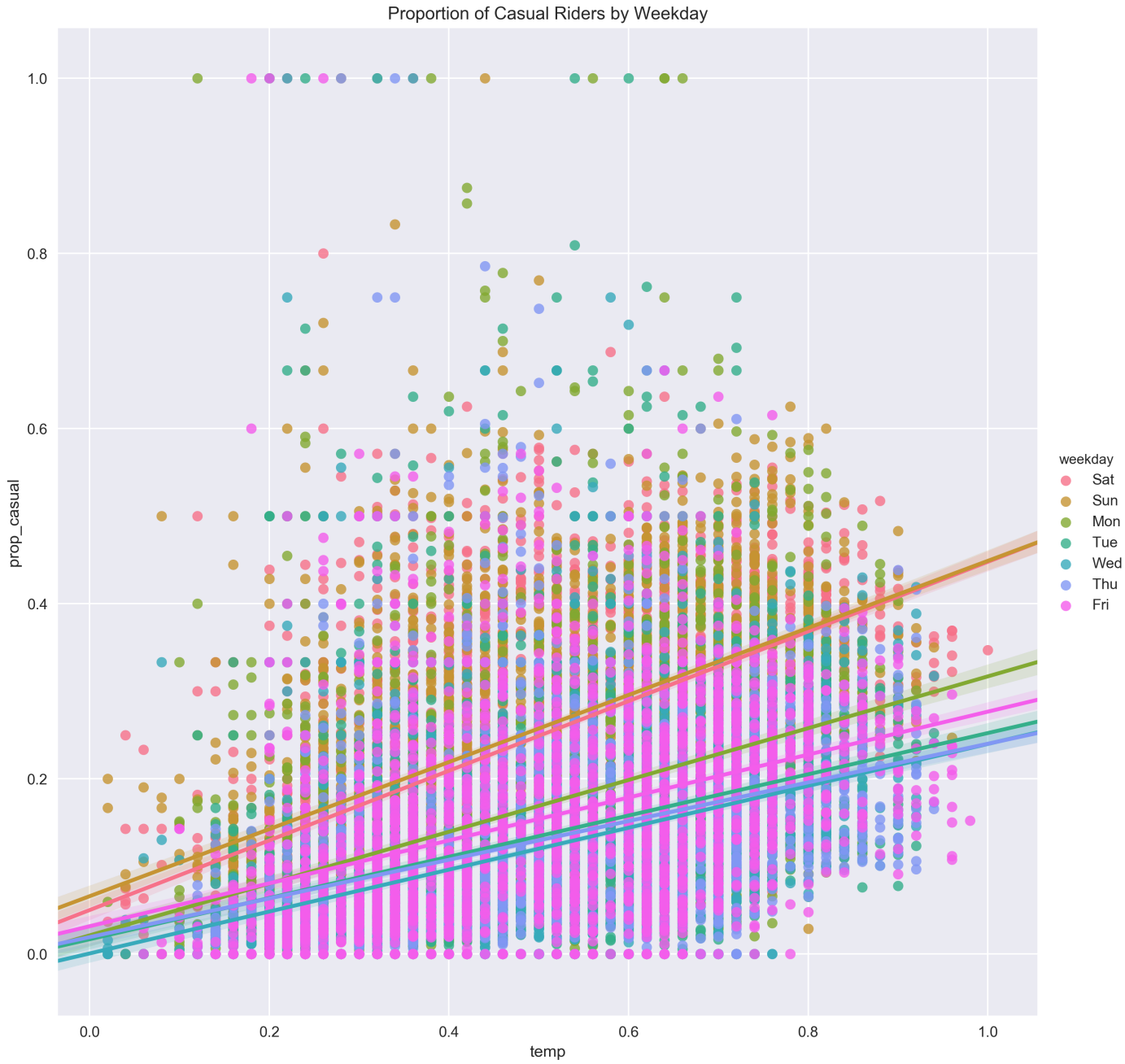


We could attempt linear regression using `sns.lmplot` as shown below, which hint at some relationships between temperature and proportional casual, but the plot is still fairly unconvincing.

IMAGE

In [30]:
```
Image(filename='images/curveplot_temp_prop_casual_2.png', embed=True, height=10, width=700)
```

Out[30]:

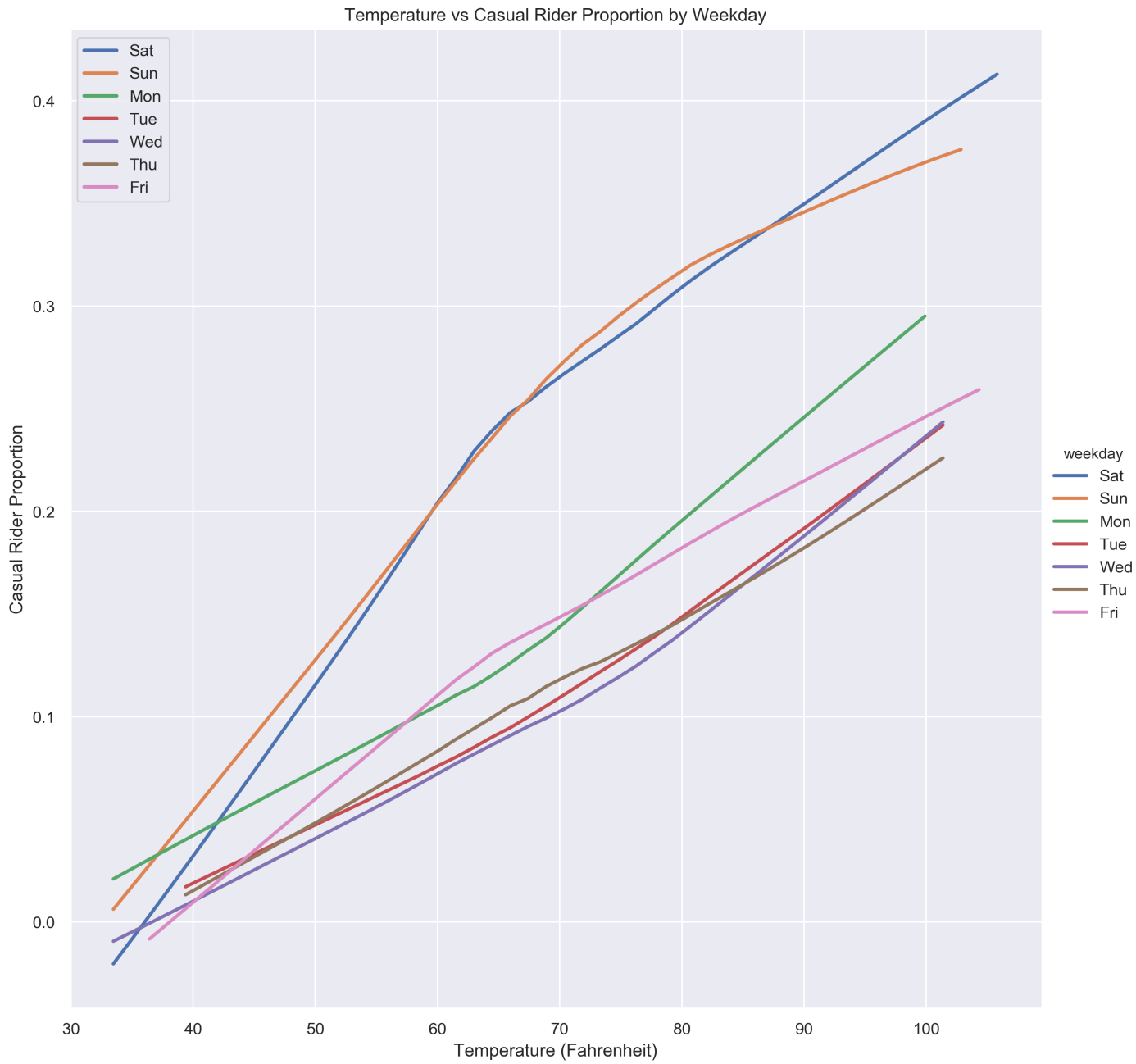


In our case with the bike ridership data, we want 7 curves, one for each day of the week. The x-axis will be the temperature and the y-axis will be some version of the proportion of casual riders. We want to remove the underlying scatter-plot.

IMAGE

In [31]:
```
Image(filename="images/curveplot_temp_prop_casual.png", embed=True, height=10, width=700)
```

Out[31]:



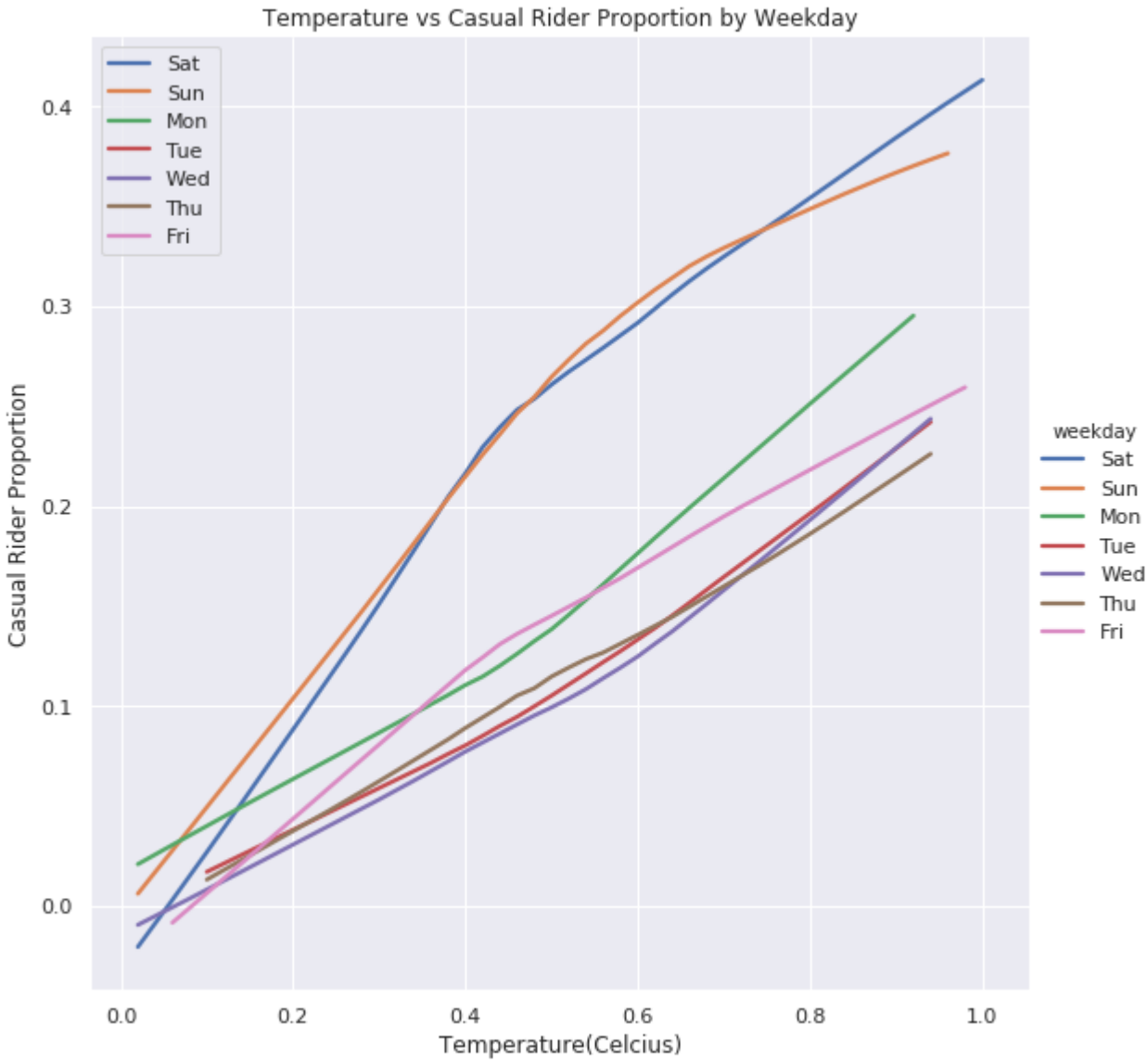
Hints:

- Start by just plotting only one day of the week to make sure you can do that first.
- Try taking `lowess = True` in `sns.lmplot` for a better fit to the trends over the 7 curves. This will allow the curves to wiggle between points.
- 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.

Note: If you prefer putting your plot in Celsius, that's fine as well!

In [32]:
```
sns.lmplot(x='temp',y='prop_casual',data=bike,hue='weekday',scatter=False,height=8,lowess=True)
plt.ylabel('Casual Rider Proportion')
plt.xlabel('Temperature(Celcius)')
plt.legend()
plt.title("Temperature vs Casual Rider Proportion by Weekday")
sns.set(font_scale=1.5)
plt.figure(figsize=(10,8))

### BEGIN SOLUTION
#TODO
### END SOLUTION
```

Out[32]: <Figure size 720x576 with 0 Axes>



<Figure size 720x576 with 0 Axes>

Question 5b

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

In [33]:
```
q5c = ''' What I see from the curve plot is that for Mon-Sun, as the Temperature in Celcius increases, the Ca
sual rider
proportion increases as well, such that there is a positive linear trend. In the case of weekend days (Sat an
d Sun), that is the
highest proportion of casual riders, mainly because if the weather is warmer, more people are likely to go ou
tside and rent a
bike for leisure, sport, or exercise. Mon and Friday seem to show a somewhat high proportion, however, Tues-T
hurs seems to have
the lowest casual rider proportion.

Prop_casual is changing as a function of temperature since as the temperature increases steadily, the proport
ion of casual riders
increases as well. This shows a linear trend, meaning that prop_casual increases as a function of temperatur
e.

I noticed something else interesting, which is the trend lines resemble a curve, some concave up, others conc
ave down.
For example, while the Temperature (Celcius) increases, the proportion increases, however the rate of increas
e on
a Saturday tends to slow down around the 0.45 area. This is most likely because if the temperature is higher,
there is more
inclination of people to rent bikes, hence the higher casual rider proportion. Also, Wednesday seems to have
 the lowest casual
proportion amounts of all, since in the middle of the week, people are less likely to rent bikes, even though
as the temperature
increases, then is a consecutive increase in the casual rider proportion


'''

### BEGIN SOLUTION
#TODO
### END SOLUTION
```