hw3

March 11, 2020

1 Homework 3

-

1.0.1 Release Date: Wednesday, February 26

-

1.0.2 Due Date: Monday March 9, 12:00 PM

1.0.3 Introduction

In this homework, we will use data provided by Capital Bikeshare. The data contains information about usuage of 2500 rental bicycles throughout 300 stations in Washington DC. Through a charge or membership, renters can use the bicycles for a span of time usually 30 minutes to 60 minutes. Since the adoption of rental bicycles in 2010, renters have access to a sustainable means of commuting that reduces congestion and crowding in other modes of transportation.

We want to understand the usage of bicycles. We will walk through the various steps of exploratory data analysis with you providing insights along the way to give you a sense for each discovery and its implications. We will try to determine the factors that impact rentals such as temperature and precipitation.

As we explore the data, you will gain practice with: * Grouping records to aggregate numbers
* Generating histograms, line-charts, scatter-plots * Assessing distributions of numbers to check for outliers or gaps

We will guide you through the problems step by step. However, we encourage you to discuss with us in Office Hours and on Piazza so that we can work together through these steps.

Submission Instructions Submission of homework requires two steps. See **Homework 0** for more information.

Step 1 You are required to **submit your notebook on JupyterHub**. Please navigate to the `Assignments` tab to
- fetch - modify - validate - submit

your notebook. Consult the instructional video for more information about JupyterHub.

Step 2 You are required to **submit a copy of your notebook to Gradescope**. Follow these steps

Formatting Instructions

1. Download as HTML (`File->Download As->HTML(.html)`).
2. Open the HTML in the browser. Print to .pdf
3. Upload to Gradescope. Consult the instructional video for more information about Gradescope.
4. Indicate the location of your responses on Gradescope. You must tag your answer's page numbers to the appropriate question on Gradescope. See instructional video for more information.

Note that

- You should break long lines of code into multiple lines. Otherwise your code will extend out of view from the cell. Consider using \ followed by a new line.
- For each textual response, please include relevant code that informed your response.
- **For each plotting question, please include the code used to generate the plot. If your plot does not appear in the HTML / pdf output, then use `Image('name_of_file', embed = True)` to embed it.**
- You should not display large output cells such as all rows of a table.

Important: Gradescope points will be awarded if and only if all the formatting instructions are followed.

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** below.

Name: *list name here*

NetId: *list netid here*

Collaborators: *list names here*

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	Points
Question 4a	2
Question 4b	2
Question 5a	1
Question 5b	4
Question 5c	1
Total	30

Rubric To start the assignment, run the cell below to set up some imports. In many of these assignments (and your future adventures as a data scientist) you will use `pandas`, `numpy` and `matplotlib.pyplot`. Import each of these libraries `as` their commonly used abbreviations (e.g., `pd`, `np` and `plt`).

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set some parameters in the packages
%matplotlib inline

sns.set(font_scale=1.5)

plt.rcParams['figure.figsize'] = (12,8)
plt.rcParams['figure.dpi'] = 150

pd.options.display.max_rows = 20
pd.options.display.max_columns = 15

# Some packages to help with configuration
import os, sys
from IPython.display import display, Latex, Markdown, Image
```

[2]:
```
# TEST

assert 'pandas' in sys.modules and "pd" in locals()
assert 'numpy' in sys.modules and "np" in locals()
assert 'matplotlib' in sys.modules and "plt" in locals()
assert 'seaborn' in sys.modules and "sns" in locals()
```

1.0.4 0: Loading Bike Sharing Data

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

1.0.5 Set the location of the data

[3]:
```
home_path = os.environ["HOME"]
data_path = f'{home_path}/shared/HW3/data/bikeshare.txt'
img_path = f'{home_path}/shared/HW3/images'
```

[4]:
```
# TEST

assert "home_path" in locals()
assert 'data_path' in locals()
assert 'img_path' in locals()
```

1.0.6 Question 0

We want to examine the file contents. Remember from Week 5 Lecture 2 that we can use commands to access a file without reading it into memory. Try running the following commands to

- check the size of the file
- determine the file format
- count the number of lines in the dataset
- view the first 10 lines of the data

Note that you do not need to provide your answers.

Question 0a (File Size) How many kilobytes is the file?

[5]:
```
!du -sh $data_path
```

```
1.1M    /home/jovyan/shared/HW3/data/bikeshare.txt
```

Question 0b (File Type) What is the file type?

[6]:
```
!file $data_path
```

```
/home/jovyan/shared/HW3/data/bikeshare.txt: ASCII text
```

Question 0c (Number of Lines) How many lines in the file?

[7]:
```
!wc -l $data_path
```

```
17380 /home/jovyan/shared/HW3/data/bikeshare.txt
```

Question 0d (Content) Print the first five rows of the file.

[8]:
```
!head $data_path
```

```
instant,dteday,season,yr,mnth,hr,holiday,weekday,workingday,weathersit,temp,atem
p,hum,windspeed,casual,registered,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
5,2011-01-01,1,0,1,4,0,6,0,1,0.24,0.2879,0.75,0,0,1,1
6,2011-01-01,1,0,1,5,0,6,0,2,0.24,0.2576,0.75,0.0896,0,1,1
7,2011-01-01,1,0,1,6,0,6,0,1,0.22,0.2727,0.8,0,2,0,2
8,2011-01-01,1,0,1,7,0,6,0,1,0.2,0.2576,0.86,0,1,2,3
9,2011-01-01,1,0,1,8,0,6,0,1,0.24,0.2879,0.75,0,1,7,8
```

1.0.7 Loading the data

The following code loads the data into a Pandas DataFrame.

[9]:
```
# Run this cell to load the data.  No further action is needed
bike = pd.read_csv(data_path)
bike.head()
```

[9]:

	instant	dteday	season	yr	mnth	hr	holiday	…	temp	atemp	\
0	1	2011-01-01	1	0	1	0	0	…	0.24	0.2879	
1	2	2011-01-01	1	0	1	1	0	…	0.22	0.2727	
2	3	2011-01-01	1	0	1	2	0	…	0.22	0.2727	
3	4	2011-01-01	1	0	1	3	0	…	0.24	0.2879	
4	5	2011-01-01	1	0	1	4	0	…	0.24	0.2879	

```
    hum  windspeed  casual  registered  cnt
0  0.81        0.0       3          13   16
1  0.80        0.0       8          32   40
2  0.80        0.0       5          27   32
3  0.75        0.0       3          10   13
4  0.75        0.0       0           1    1

[5 rows x 17 columns]
```

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.

[10]:
```
bike.shape
```

[10]:
```
(17379, 17)
```

1.0.8 1: Data Preparation

Question 1 Four of the columns in the table use numbers as categorical data. We will convert these entries to strings indicating the categories. - `holiday` * `Sun` * `Mon` * `Tue` * `Wed` * `Thu` * `Fri` * `Sat` - `weekday` * `yes` * `no` - `workingday` * `yes` * `no` - `weathersit` * `Clear` * `Mist` * `Light` * `Heavy`

Question 1a (Decoding `weekday`, `workingday`, and `weathersit`) Convert the `holiday`, `weekday`, `workingday`, and `weathersit` fields:

1. `holiday`: Convert to `yes` and `no`.
2. `weekday`: Convert to `'Sun'`, `'Mon'`, `'Tue'`, `'Wed'`, `'Thu'`, `'Fri'`, and `'Sat'`. Note `0` corresponds to `Sun`, `1` to `Mon` and so on.
3. `workingday`: Convert to `yes` and `no`.
4. `weathersit`: Convert to `Clear`, `Mist`, `Light`, or `Heavy`.

Note if you want to revert the changes run the cell above that loads the file.

Try using the `replace` method. You need to pass a dictionary where the key is the number and the value is the corresponding string. For example

```
bike["holiday"].replace({0:'no', 1:'yes'}, inplace = True)
```

[11]:
```
# convert numbers to strings in `holiday`, `weekday`, `workingday`, and␣
↪`weathersit`

bike["holiday"].replace({0:'no', 1:'yes'}, inplace = True)
bike["weekday"].replace({0:'Sun', 1:'Mon',2:'Tue',3:'Wed',4:'Thu',5:'Fri',6:
↪'Sat' }, inplace = True)
```

```
bike["workingday"].replace({0:'no', 1:'yes'}, inplace = True)
bike["weathersit"].replace({0:'Clear', 1:'Mist',2:'Light',3:'Heavy'}, inplace =␣
↪True)
#raise NotImplementedError()
```

[12]:
```
# TEST
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')
```

Question 1b (Holidays) How many entries in the data correspond to holidays? Set the variable `num_holidays` to this value.

[13]:
```
t = bike[bike["holiday"]=='yes']
num_holidays= len(t.index)
print(num_holidays)
# YOUR CODE HERE
#raise NotImplementedError()
```

```
500
```

[14]:
```
# TEST
assert 400 <= num_holidays <= 550
```

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 at the daily level. 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`)

Use `groupby` and `agg`. For a reminder about the `agg` method, you can check the documentation. Remember that we can have different aggregations in each column by passing a dictionary.

For `casual` column and `registered` column you need to take a `sum`. For the `workingday` column, you can take any of the values since we are grouping by the day. Try using the function `select_first_row` from Week 4.

[15]:
```
def select_first_row(some_series):
    return some_series[0]

daily_counts = bike.groupby("dteday").agg({"casual":np.sum,"registered":np.
↪sum,"workingday":select_first_row})
```

```
# YOUR CODE HERE
#raise NotImplementedError()
```

[16]:
```
# TESTS
assert np.round(daily_counts['casual'].mean()) == 848.0
assert np.round(daily_counts['casual'].var()) == 471450.0
```

1.1 2: Exploring the Distribution of Riders

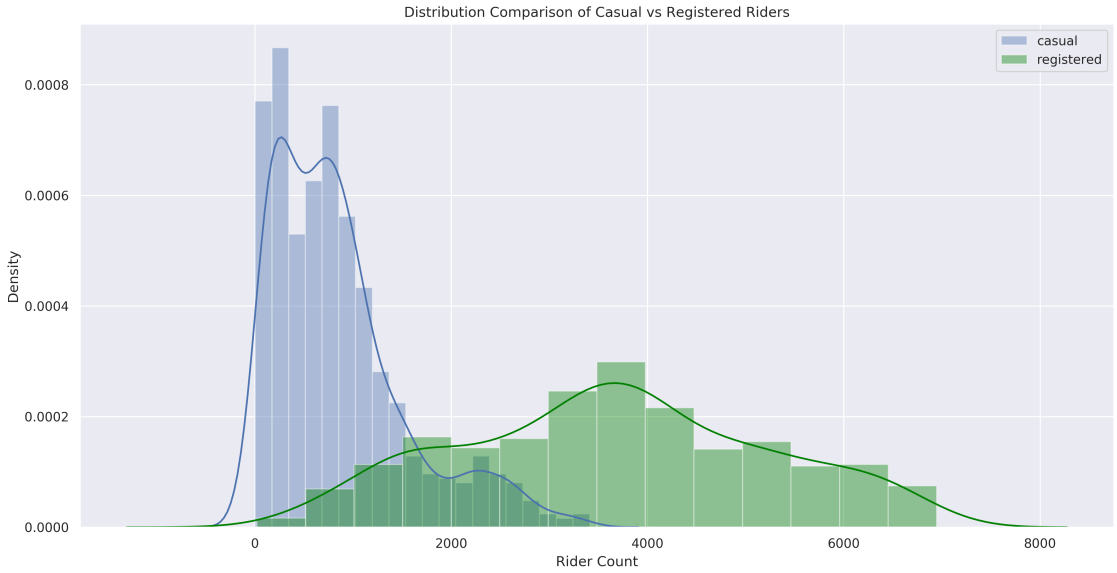
Let's begin by comparing the distribution of the daily counts of casual and registered riders.

1.1.1 Question 2

Question 2a Use `sns.distplot` function to create a chart containing two normalized histograms. Remember that normalized means we divide by the size of the data to get frequencies instead of counts. Here the frequencies for `casual` and `registered` should be in the same figure.

[17]:
```
Image(filename=f'{img_path}/casual_v_registered.png', embed=True, height=10,␣
↪width=700)
```

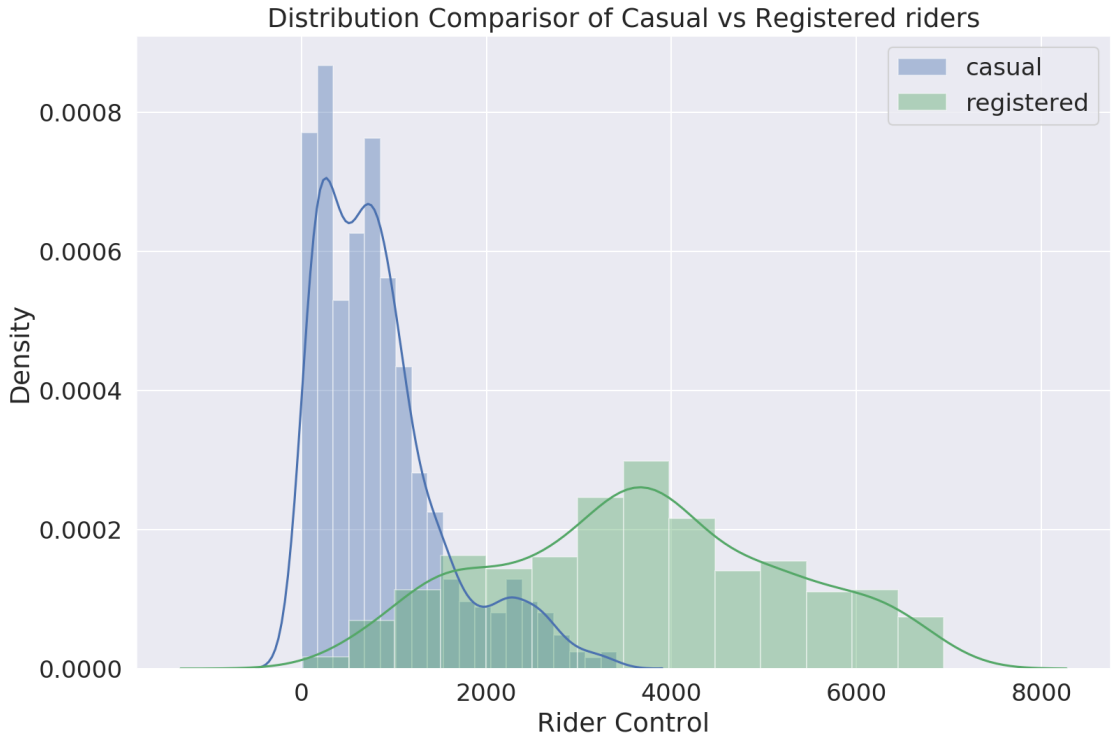
[17]:



Following Question 1c, the granularity of the records should be at the daily level. Please include a legend, xlabel, ylabel, and title. You can check the documentation for more information.

[37]:
```
# generate the histograms

ax = sns.distplot(daily_counts['casual'],label= "casual",color = 'b')
ax = sns.distplot(daily_counts['registered'],label= "registered",color = 'g')
ax.set_ylabel("Density")
ax.set_xlabel("Rider Control")
ax.set_title("Distribution Comparisor of Casual vs Registered riders")
plt.legend()
#raise NotImplementedError()
```

[37]:
```
<matplotlib.legend.Legend at 0x7ff19e5afbd0>
```



1.1.2 Question 2b

After creating the plot in Question 2a, we want to study it. What is the chart telling us? For example, on a given day, the most likely number of registered riders is about 4000, but it could be anywher from about 0 to 7000.

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. Save your answer in the variable q2b as a string.

[38]:
```
q2b = """
distribution:we can see that the density of casual riders are highter than the␣
 ↪registered riders. ALso, we can see that the high-density part of the casual␣
 ↪riders is at about 1000 while the one for registered riders is about 4000
skew: we can see that both chart are skew to both sides, but the registered␣
 ↪part are more strict while the casual one there is a gap at the middle
spread:we can see that the registered has a range about 8000, which is much␣
 ↪wider than the casual one which is only about 3500
center:for the registered, the center of the chart is the highest place while␣
 ↪for the casual one there is a gap.
"""

# YOUR CODE HERE
#raise NotImplementedError()
```

[39]:
```
# TESTS
for keyword in ['distribution', 'skew', 'spread', 'center']:
    assert keyword in q2b
```

1.1.3 Question 2c

The histograms do not show us the changes in registered and casual riders together across the different days. Use sns.scatterplot to make a scatter plot to investigate the relationship between casual and registered counts.

- Color the points in the scatterplot according to whether or not the day is working day. Set hue="workingday"
- There are many points in the scatter-plot. You can make them small to help with over-plotting. Set s = 20
- You will need to set x and y for casual and registered

[40]:
```
Image(filename=f'{img_path}/casual_registered_working_nonworking.png',␣
 ↪embed=True, height=10, width=700)
```

[40]:

[26]:
```
# generate the scatter-plot

...

ax = sns.scatterplot(x="casual", y="registered",hue="workingday"␣
↪,data=daily_counts,s=20)
#raise NotImplementedError()
```

1.1.4 Question 2d

What does this scatterplot seem to reveal about the relationship between casual and registered riders. How does the relationship change whether it is 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.

Save your answer in the variable q2d as a string.

[48]:
```
q2d = """
We can see that both two opitions seem can be fit into a linear function where␣
 ↪the "yes" one have a higher slope than the "no" one.
There are more registered riders on work day while more casual riders on␣
 ↪weedend.
It is hard to find the correlation between the point and variables when it␣
 ↪comes to overplotting.
"""

# YOUR CODE HERE
#raise NotImplementedError()
```

[49]:
```
# TESTS
for keyword in ['linear', 'correlation', 'casual', 'registered', 'work', 'day',␣
↪'overplotting']:
    assert keyword in q2d
```

[50]:
```
bike.hr.unique()
```

[50]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23])
```

1.1.5 3: Exploring Ride Sharing and Time

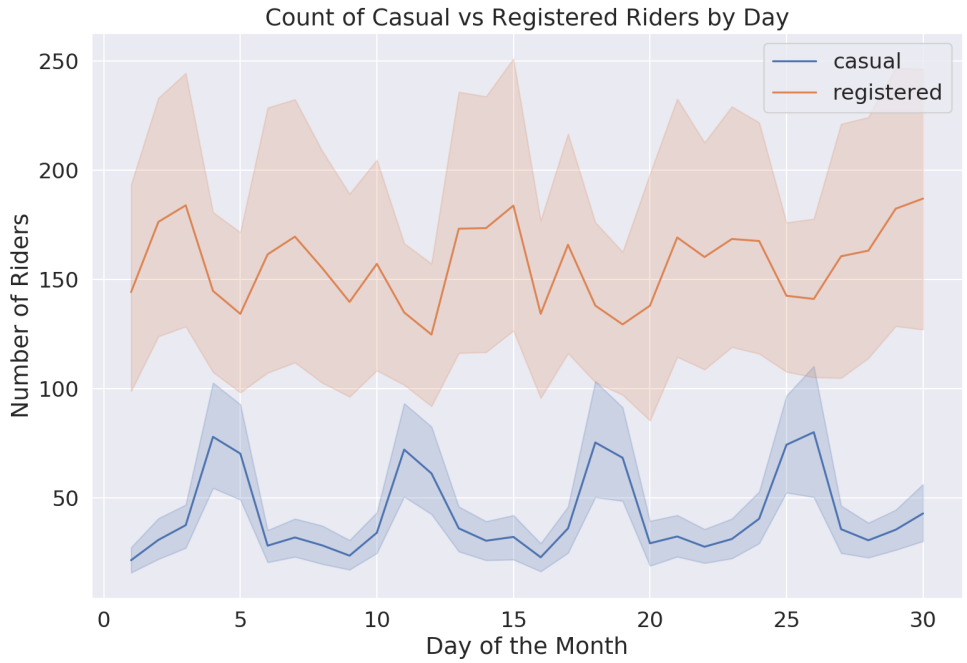
Question 3 We want to understand the patterns of casual and registered riders over time.

Question 3a Let us plot average number of riders per hour for each day in the month of June in 2011. Along the horizontal axis we will have the day of the month. Along the vertical axis we will have the average number of riders across the different hours of the day.

We will use the bike dataframe with the function sns.lineplot. Remeber that the bike dataframe has granularity at the hourly level. We will use sns.lineplot to plot the day column against the casual column and the registered column. Since each day has 24 hours, we have 24 possible numbers to associate with the day. By default, the function sns.lineplotwill take an average of the numbers.

[51]:
```
Image(filename=f'{img_path}/june_riders.png', embed=True, height=10, width=700)
```

[51]:



On some days the number of riders will vary a lot over the hours. Other days the number of riders will vary a little over the hours. Later in the semester, we will learn to bound the variations of numbers with **confidence intervals**. Intuitively, we should think that

- large confidence interval means the data is spread out
- small confidence interval means the data is not spread out

In the plot, the shading around the lines are the confidence intervals for each day.

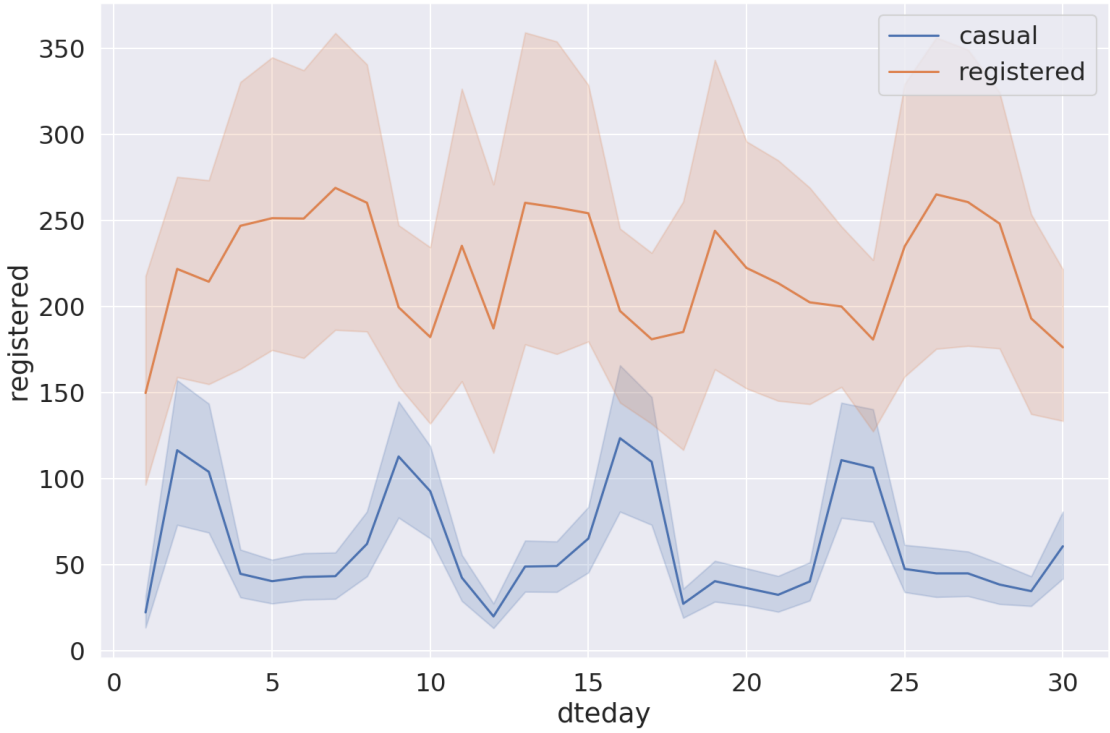
[52]:
```
def extract_day(some_string):
    return int(some_string[-2:])
```

Follow these steps to produce the chart.

1. Create a table consisting of the riders in June 2011
    - You need to filter the rows by yr equals 0 and mnth equals 6
    - Should should copy the filtered table. Try using the copy method
2. Extract the day from the dteday column
    - We have provided you a function called extract_date. Use it with apply to obtain the day as a number.
3. Use sns.lineplot to plot casual and registered riders
    - Include xlabel, ylabel, legend, title in the figure
    - To answer the question about confidence intervals look at the shading

around the line.

[53]:
```
r =bike[(bike['yr']==1)&(bike['mnth']==6)].copy()
t = r["dteday"].apply(extract_day)
r["dteday"] = t
ax = sns.lineplot(x="dteday", y="casual", data=r,label = "casual")
ax = sns.lineplot(x="dteday", y="registered", data=r,label = "registered")

#bike_june_2011=sns.lineplot(x="Day of the Month", y="Number of Riders", data=t)
# YOUR CODE HERE
#raise NotImplementedError()
```



Question 3b This plot has several interesting features.

- How does the number of casual riders compare to that of registered riders?
- Is there a temporal pattern?

Please select your answers among the provide answers. Save them as a dictionary in the variable q3b.

[54]:
```
q3b = {
    'overall number of riders':'casual riders are less than the registered␣
↪riders' , # 'casual riders are less/more than the registered riders'
```

```
    'day-of-week pattern for casual riders':'exists' , # 'exists', 'does not␣
 ↪exist'
    'confidence interval for casual riders':'smaller than that of registered␣
 ↪riders' , #  'larger/smaller than that of registered riders'
}

# YOUR CODE HERE
#raise NotImplementedError()
```

[55]:
```
# TESTS
assert q3b['overall number of riders'] in [
    'casual riders are less than the registered riders',
    'casual riders are more than the registered riders',]
assert q3b['day-of-week pattern for casual riders'] in ['exists', 'does not␣
 ↪exist',]
assert q3b['confidence interval for casual riders'] in [
    'smaller than that of registered riders',
    'larger than that of registered riders',]
```

1.2 4: Understanding Daily Patterns

1.2.1 Question 4

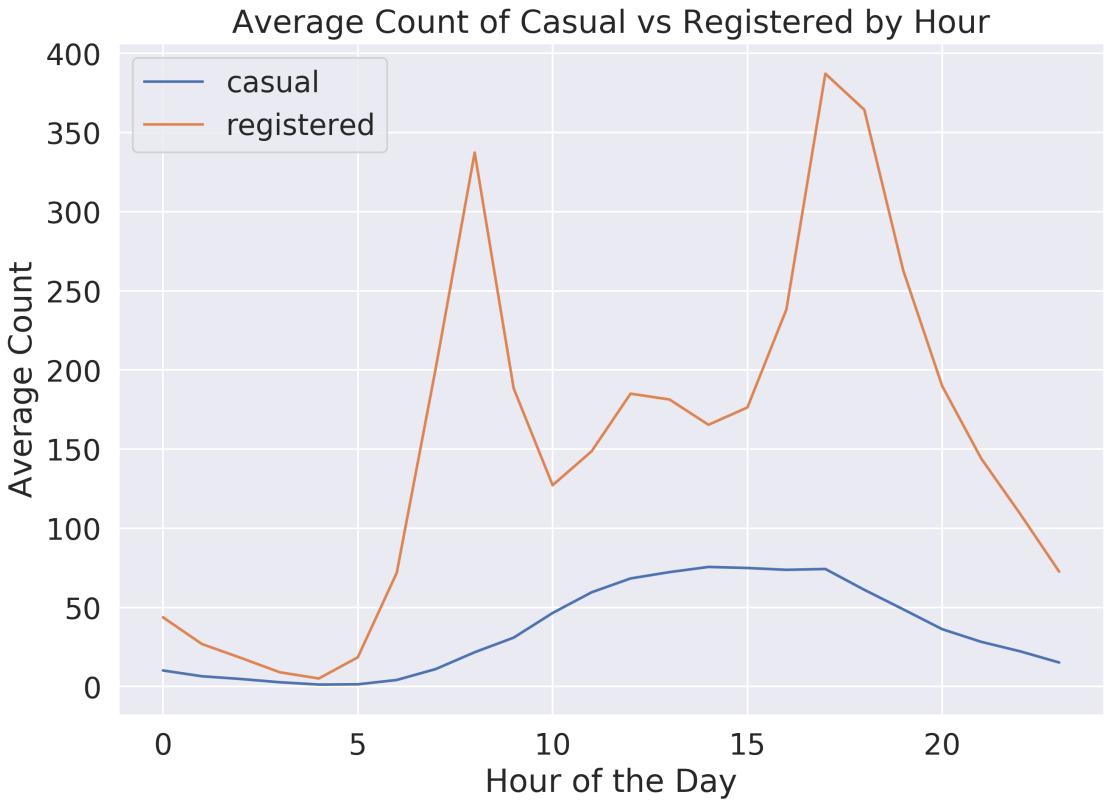
Having studied the rentals at the daily level, let us try to investigate trends throughout the day at the hourly level.

Question 4a We want to plot the average number of riders for each hour of the day. Moreover, we want to stratify by casual and registered. Note that we are **not** limiting to June 2011 like in Question 3a.

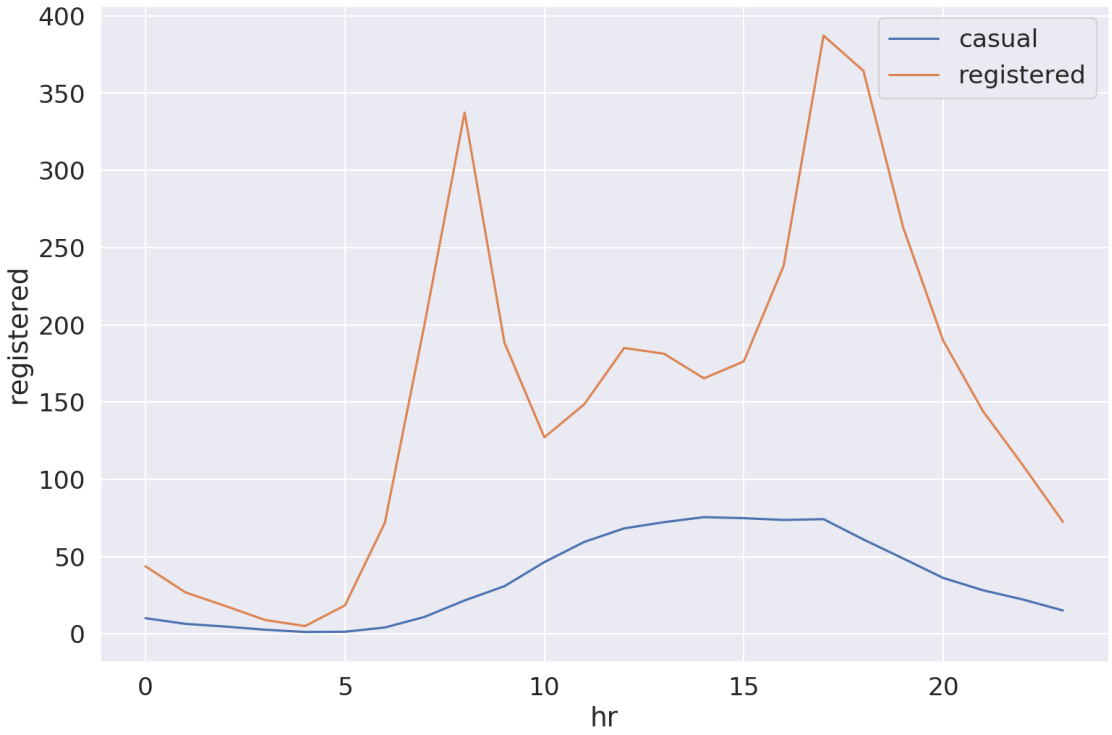
Please use groupby with the hr column to generate the data for sns.lineplot.

[56]:
```
Image(filename=f'{img_path}/diurnal_bikes.png', embed=True, height=10,␣
 ↪width=700)
```

[56]:



[57]:
```
# group by hour to take the average.
r = bike.groupby(["hr"]).mean()
#r["Hours"] = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]
ax = sns.lineplot( x = bike["hr"],y="casual", data=r,label = "casual")
ax = sns.lineplot( x = bike["hr"],y="registered", data=r,label = "registered")
# YOUR CODE HERE
#raise NotImplementedError()
```



Question 4b What can you observe from the plot? Save your answers in the dictionary variable below. Answers should be boolean (either True or False)

[58]:
```
q4b = {
    'hourly pattern exist for casual riders': True, # True/False
    'hourly pattern exist for registered riders': True, # True/False
    'distribution looks unimodal for casual riders': True, # True/False
    'distribution looks unimodal for registered riders': False, # True/False
    'distribution for the casual riders includes strong spikes': False, # True/
 ↪False
    'distribution for the registered riders includes strong spikes': True, #␣
 ↪True/False
}

# YOUR CODE HERE
#raise NotImplementedError()
```

[59]:
```
# TESTS
assert len(q4b) == 6
for k, v in q4b.items():
    assert isinstance(v, bool)
```

1.2.2 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.

1.2.3 Question 5

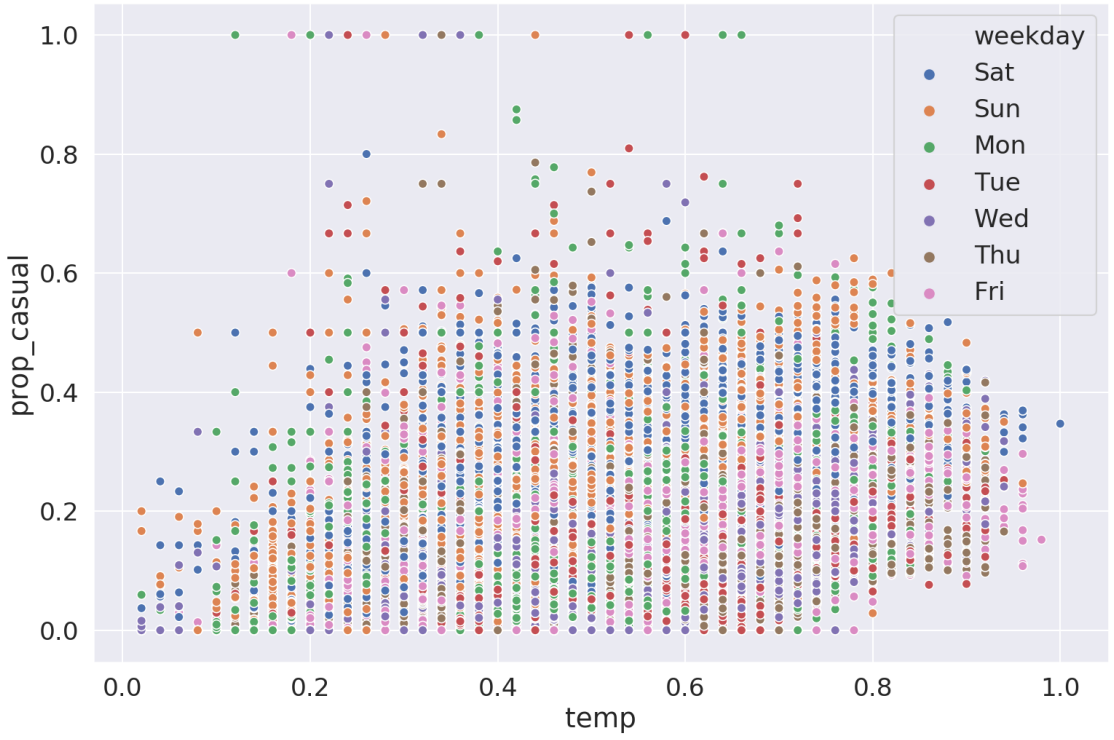
Question 5a Create a new column prop_casual in the bike dataframe representing the proportion of casual riders out of all riders.

[60]:
```
# add `prop_casual` to the table

# YOUR CODE HERE
bike

bike["prop_casual"] =bike["casual"]/bike["cnt"]
#bike["prop_casual"].sum()
#raise NotImplementedError()
```

[61]:
```
# TESTS
assert int(bike["prop_casual"].sum()) == 2991
```

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 messy chart.

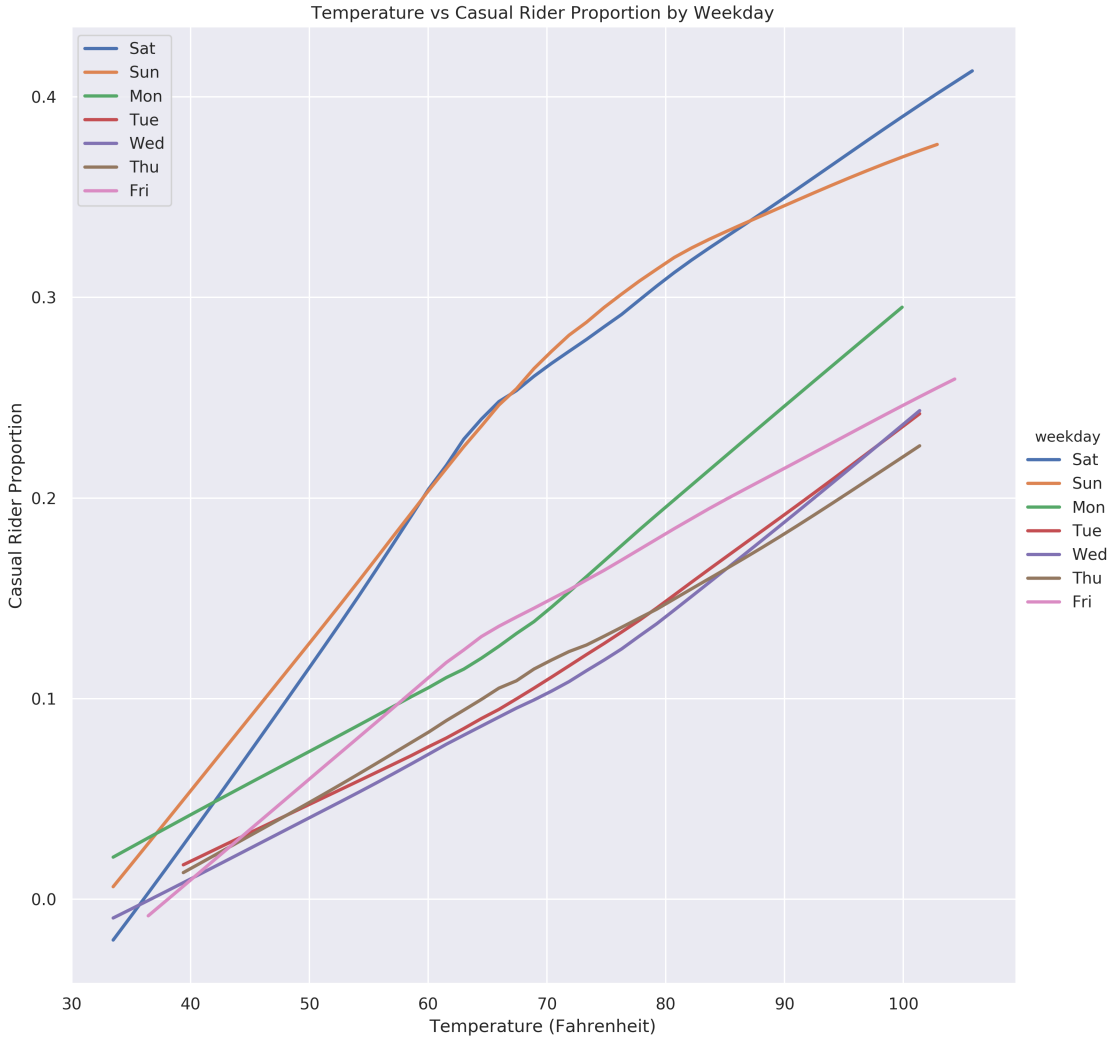
[62]:
```
sns.scatterplot(data=bike, x="temp", y="prop_casual", hue="weekday");
```



Instead 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.

[63]:
```
Image(filename=f'{img_path}/curveplot_temp_prop_casual.png', embed=True,␣
↪height=10, width=700)
```

[63]:



You should start by plotting one of seven curves.

- You must convert the temperature to Farenheit. Note that the Section ?? indicate the values in temp have been scaled to lie in the range 0 to 1 through division by 41. You must multiply by 41 before converting between Celsius and Farenheit with the formula

$$\text{Farenheit} = \frac{9}{5}\ \text{Celsius}\ + 32$$

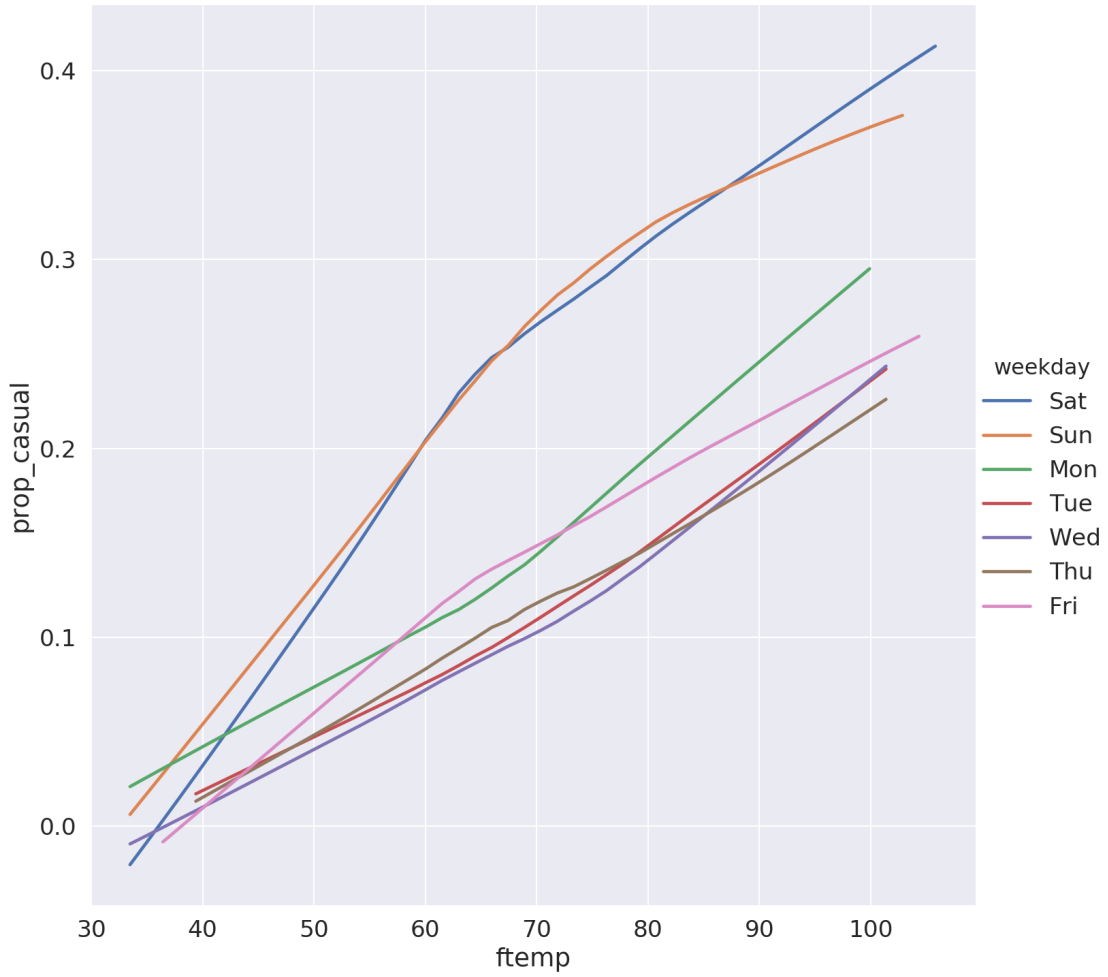
- Use sns.lmplot to generate the chart
  - Set scatter = False to remove the scatter-plot
  - Try setting lowess = True for a better fit to the trends over the 7 curves. This will allow the curves to wiggle between points.

After you can produce the chart for one curve, try combining the data to produce a chart for all seven curves.

- Set hue = "weekday" to have the lines separated by day of the week

[64]:
```
# curves for each day of the week

bike["ftemp"] = bike["temp"]*41*(9/5)+32
sns.lmplot(x = 'ftemp',y='prop_casual',data = bike, scatter =False, lowess =␣
↪True,hue = 'weekday',height=10 )
#raise NotImplementedError()
```

[64]: <seaborn.axisgrid.FacetGrid at 0x7fbed5163d10>



Question 5c What do you see from the curve plot? How is prop_casual changing as a function of temperature? Do you notice anything else interesting?

[65]:

```
q5c = {
    'as the temperature increases, proportion of casual riders': "increases", #␣
 ↪"increases/decreases/doesn't change"
    'weekends have more/less/same proportion of casual riders': 'more' #  'more/
 ↪less/same'
}

# YOUR CODE HERE
#raise NotImplementedError()
```

[66]:
```
# TESTS
assert q5c['as the temperature increases, proportion of casual riders'] in [
    'increases', 'decreases', "doesn't change"]
assert q5c['weekends have more/less/same proportion of casual riders'] in [
    'more', 'less', 'same'
]
```

1.3 Congrats !

You are finished with HW3. We have covered many components of the seaborn package. Please see the documentation for more information about these charts from the assignment:

- distplot
- lmplot
- lineplot
- scatterplot