Instructions From CodeCademy

David Mukajanga

9/28/2021

# On Introduction To Data Cleaning

### 1.

Now that we have the necessary datasets, we can use programming to clean and organize the data so it’s useful for our purposes.

Let’s first look at the user\_data.csv file. We can use Pandas to transform this CSV file into a printable table, known as a ***DataFrame***.

To see part of the user\_data DataFrame, copy and paste the following line:

print(user\_data.head(15))

How does the DataFrame look different from the original CSV file? What would happen if you change the number inside the parentheses?

* The Code:

import codecademylib3\_seaborn  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Import the CSV files and create the DataFrames:  
user\_data = pd.read\_csv("user\_data.csv")  
pop\_data = pd.read\_csv("pop\_data.csv")  
  
# Paste print code here:  
  
  
# Paste merge code here:  
  
  
# Paste location code here:

### 2.

Remember that our original user\_data.csv file only told us the names of the towns and cities our users live in. But we need to determine if someone lives in a rural or urban location.

The first step in getting that information is to take the population data from the pop\_data.csv file and add it to our user\_data DataFrame. In Pandas, this process is known as *merging*.

To merge, copy and paste the following line of code:

new\_df = pd.merge(user\_data, pop\_data)

Then paste

print(new\_df.head(15))

to inspect the new table!

### 3.

Now that we’ve added population data to our user\_data DataFrame, we can use the new DataFrame (saved in the variable new\_df) to determine whether someone lives in an “urban” or “rural” location and then add that classification to a new column, named location.

We can determine the location category by comparing the size of the population to a definition of size for an urban population. If an urban population is classified as having a population of 100,000 or more people, then any city with a population above that will be labeled “urban” and anyone below that will be labeled “rural”.

Copy and paste the following code to create and populate a new location column, based on how a user’s location is classified.

new\_df.loc[new\_df.population\_proper < 100000, "location"] = "rural"  
new\_df.loc[new\_df.population\_proper >= 100000, "location"] = "urban"

Don’t forget to inspect the new DataFrame!

print(new\_df.head(15))

# On Exploring the Data..

### 1.

Recall that we’re interested in understanding if a user’s location is correlated to their age. So far, we’ve mostly looked at location information. Let’s explore our data to learn more about the average age of our users.

Previous data scientists at the company have determined that the mean age of your users is around 29. Let’s use a histogram to visualize the distribution of all of the ages in our dataset.

Copy and paste the following code to plot a histogram:

age = new\_df["age"]  
sns.displot(age)  
   
plt.show()

Looking at the visualization, can we say that our dataset is evenly distributed, meaning that each age is represented equally? Or do we see higher numbers of certain ages? What does this histogram tell us about our user population?

### 2.

Now that we have a sense of the age distribution of our user base, let’s see if there is a relationship between a user’s age and their location.

We can calculate the mean age of users based on their location, using the following code:

location\_mean\_age = new\_df.groupby("location").age.mean()   
   
print(location\_mean\_age)

Take a look at the output in the terminal window. Is the average age of a user younger in an urban or a rural setting?

### 3.

Copy and paste the following code to visualize the means using a barplot:

plt.close()  
sns.barplot(  
 data=new\_df,  
 x= "location",  
 y= "age"  
)  
plt.show()

Scroll down to see the second plot.

### 4.

Other visualizations provide other information about the relationship between age and location. For example, a violin plot shows distributions of age like a histogram, but creates separate distributions for each category.

plt.close()  
sns.violinplot(x="location", y="age", data=new\_df)  
   
plt.show()

Scroll down to see the third plot. It looks like age is evenly distributed in urban areas between 20 and 40 years old, whereas age is concentrated around 30 in rural areas.

The code..

import codecademylib3\_seaborn  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Import CSVs:  
user\_data = pd.read\_csv("user\_data.csv")  
pop\_data = pd.read\_csv("pop\_data.csv")  
  
# Merged tables with location data:  
new\_df = pd.merge(user\_data, pop\_data)  
new\_df.loc[new\_df.population\_proper < 100000, "location"] = "rural"  
new\_df.loc[new\_df.population\_proper >= 100000, "location"] = "urban"  
  
# Paste histogram code:  
age = new\_df["age"]  
sns.displot(age)  
   
plt.show()  
  
# Paste mean age location code:  
location\_mean\_age = new\_df.groupby("location").age.mean()   
   
print(location\_mean\_age)  
  
# Paste barplot code:  
plt.close()  
sns.barplot(  
 data=new\_df,  
 x= "location",  
 y= "age"  
)  
plt.show()  
  
# Paste violinplot code:  
plt.close()  
sns.violinplot(x="location", y="age", data=new\_df)  
   
plt.show()

# On Modeling the Data

### 1.

Now that we’ve gotten a sense of our data, it’s time to build a model.

We want to understand the relationship between age and location, so let’s make a plot that compares the two. In this case, we want to know if age is dependent on location.

The following code will produce a scatter plot, which plots data on a graph based on two variables. We’ll use age on the y-axis and population on the x-axis:

x = new\_df["population\_proper"]  
y = new\_df["age"]  
   
plt.scatter(x, y, alpha=0.5)

### 2.

We can sort of start to see some patterns. See the dots in vertical lines? We can tell that those are all users of different ages in the same city!

One useful model that works with a scatter plot is a linear regression because it draws a line of best fit.

Copy and paste the following code to create a linear regression model using age and population data, below the scatterplot:

sns.regplot(x="population\_proper", y="age", data=new\_df)

Now if we have a new user who is from a different city, we can use our model to predict what age they might be given the population of the city. Scroll down to see the second plot!

### The Code:

import codecademylib3\_seaborn  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Import CSVs:  
user\_data = pd.read\_csv("user\_data.csv")  
pop\_data = pd.read\_csv("pop\_data.csv")  
  
# Merged tables with location data:  
new\_df = pd.merge(user\_data, pop\_data)  
new\_df.loc[new\_df.population\_proper < 100000, "location"] = "rural"  
new\_df.loc[new\_df.population\_proper >= 100000, "location"] = "urban"  
  
# Paste code for scatter plot:  
x = new\_df["population\_proper"]  
y = new\_df["age"]  
   
plt.scatter(x, y, alpha=0.5)  
  
# Paste code for linear regression:  
sns.regplot(x="population\_proper", y="age", data=new\_df)  
  
# Show plot  
plt.show()

# On Communicating The Findings

### 1.

Here is our visualization that shows the correlation between age and population. Let’s improve this graph so we can include it in a presentation to our product team!

First, let’s change the style and colors of the figure so that the line is more visible.

plt.close()  
   
sns.set\_style("darkgrid")  
sns.set\_palette("bright")  
sns.despine()  
   
sns.regplot(x="population\_proper", y="age", data=new\_df)  
   
plt.show()

After you add the above code to script.py and run the file, scroll down in the browser component on the right-hand side of the screen to see the new visualization.

### 2.

Copy and paste the code below into the editor to change the tick marks on the axes so the population sizes are clearer:

ax = plt.subplot(1, 1, 1)  
ax.set\_xticks([100000, 1000000, 2000000, 4000000, 8000000])  
ax.set\_xticklabels(["100k", "1m", "2m","4m", "8m"])  
   
plt.show()

After you add the above code to script.py and run the file, scroll down in the browser component on the right-hand side of the screen to see the new visualization.

### 3.

Lastly, let’s re-title the axes and give the plot a title to improve the presentation of the data

ax.set\_xlabel("City Population")  
ax.set\_ylabel("User Age")  
plt.title("Age vs Population")  
   
plt.show()

After you add the above code to script.py and run the file, scroll down in the browser component on the right-hand side of the screen to see the new visualization.

### The Code:

import codecademylib3\_seaborn  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Import CSVs:  
user\_data = pd.read\_csv("user\_data.csv")  
pop\_data = pd.read\_csv("pop\_data.csv")  
  
# Merged tables with location data:  
new\_df = pd.merge(user\_data, pop\_data)  
new\_df.loc[new\_df.population\_proper < 100000, "location"] = "rural"  
new\_df.loc[new\_df.population\_proper >= 100000, "location"] = "urban"  
  
# Plot linear regression:  
sns.regplot(x="population\_proper", y="age", data=new\_df)  
plt.show()  
  
  
# Paste code to change the figure style and palette:  
plt.close()  
   
sns.set\_style("darkgrid")  
sns.set\_palette("bright")  
sns.despine()  
   
sns.regplot(x="population\_proper", y="age", data=new\_df)  
   
plt.show()  
  
  
# Paste code to change the axes:  
ax = plt.subplot(1, 1, 1)  
ax.set\_xticks([100000, 1000000, 2000000, 4000000, 8000000])  
ax.set\_xticklabels(["100k", "1m", "2m","4m", "8m"])  
   
plt.show()  
  
  
# Paste code to title the axes and the plot:   
ax.set\_xlabel("City Population")  
ax.set\_ylabel("User Age")  
plt.title("Age vs Population")  
   
plt.show()