HW Week 1

September 10, 2024

1. Pick one of the datasets from the ChatBot session(s) of the TUT demo (or from your own ChatBot session if you wish) and use the code produced through the ChatBot interactions to import the data and confirm that the dataset has missing values.

[16]:
```python
import pandas as pd
url = "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
↪data/2020/2020-05-05/villagers.csv"
df = pd.read_csv(url)
df.isna().sum()
```

[16]:
```
row_n           0
id              1
name            0
gender          0
species         0
birthday        0
personality     0
song           11
phrase          0
full_id         0
url             0
dtype: int64
```

2. Start a new ChatBot session with an initial prompt introducing the dataset you're using and request help to determine how many columns and rows of data a pandas DataFrame has, and then

a. use code provided in your ChatBot session to print out the number of rows and columns of the dataset; and,
b. write your own general definitions of the meaning of "observations" and "variables" based on asking the ChatBot to explain these terms in the context of your dataset

[2]:
```python
import pandas as pd

# Load the dataset from the provided URL
url = "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
↪data/2020/2020-05-05/villagers.csv"
villagers_df = pd.read_csv(url)
```

```
# Get the shape of the dataset (rows, columns)
villagers_df.shape
```

```
[2]: (391, 11)
```

Observation Definition: Refers to a signle data entry in a dataset about which information is gathered. Correspond to rows in a dataset. Ex. In my dataset, each observation represents one villager.

Variable Definition: The traits or qualities that are being measured or recorded for every observation. Correspond to columns in a dataset. Ex. In my dataset, each variable corresponds to a different trait or quality of the villagers.

3. Ask the ChatBot how you can provide simple summaries of the columns in the dataset and use the suggested code to provide these summaries for your dataset.

```
[3]: # Summarize numeric columns
print(villagers_df.describe())
```

```
            row_n
count  391.000000
mean   239.902813
std    140.702672
min      2.000000
25%    117.500000
50%    240.000000
75%    363.500000
max    483.000000
```

```
[11]: # Count unique values in a categorical column (e.g., 'species')
print(villagers_df['species'].value_counts())
```

```
species
cat         23
rabbit      20
frog        18
squirrel    18
duck        17
dog         16
cub         16
pig         15
bear        15
mouse       15
horse       15
bird        13
penguin     13
sheep       13
elephant    11
wolf        11
```

```
ostrich      10
deer         10
eagle         9
gorilla       9
chicken       9
koala         9
goat          8
hamster       8
kangaroo      8
monkey        8
anteater      7
hippo         7
tiger         7
alligator     7
lion          7
bull          6
rhino         6
cow           4
octopus       3
Name: count, dtype: int64
```

[7]:
```python
# Find unique values in a column
print(villagers_df['personality'].unique())
```

```
['cranky' 'peppy' 'uchi' 'lazy' 'normal' 'snooty' 'jock' 'smug']
```

4. If the dataset you're using has (a) non-numeric variables and (b) missing values in numeric variables, explain (perhaps using help from a ChatBot if needed) the discrepancies between size of the dataset given by df.shape and what is reported by df.describe() with respect to (a) the number of columns it analyzes and (b) the values it reports in the "count" column.

[12]:
```python
import pandas as pd

# Load the dataset
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
 ↪csv"
titanic_df = pd.read_csv(url)

#Get the shape of the DataFrame
titanic_df.shape
```

[12]:
```
(891, 15)
```

[13]:
```python
# Get summary statistics for numeric columns
titanic_df.describe()
```

[13]:
```
         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
```

```
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

a. Number of columns being analyzed.

df.shape reports all columns (numeric, categorical, and others) while df.describe() only reports numeric columns by default. In the Titanic dataset, columns like ‘survived’, ‘age’, ‘fare’, etc., are numeric. Columns like ‘sex’, ‘embarked’, and ‘class’ are categorical and won’t be included unless specified. Therefore, df.shape reports the total number of columns (15), while df.describe() only reports numeric columns by default (6), leading to fewer columns being analyzed.

b. Values reported in the “count” column.

The “count” column in df.describe() shows the number of non-missing values for each numeric column. If a column has missing values, the “count” will be less than the total number of rows reported by df.shape. For example, in the Titanic dataset, the ‘age’ column has missing values so its “count” will be less than 891 (714 = the total number of rows without missing values).

ChatBot Session: https://chatgpt.com/share/ba55d53d-a978-41f6-857e-831fffb0d9f5

Summary of Interactions:

1. Dataset Information: Dataset: Villagers dataset. Objective: Determine the number of columns and rows in the dataset using Python. Solution: Used the pandas library to read the dataset and determine its shape.

2. Definitions: Observations: Refers to individual data points or entries in the dataset (rows). Variables: Refers to different characteristics or features measured (columns).

3. Providing Simple Summaries of Columns: Methods to summarize columns include: .describe(): For numerical columns, provides summary statistics. .info(): Gives basic information about the DataFrame, including non-null counts and data types. .value_counts(): For categorical columns, shows counts of unique values. .head(): Displays the first few rows of the DataFrame. .isnull().sum(): Checks for missing values.

4. Discrepancies between df.shape and df.describe(): Number of Columns Analyzed: df.shape shows total columns, while df.describe() shows only numerical columns by default. To include all columns, use df.describe(include=‘all’). Values in the “count” Column: df.describe() shows non-null counts for numerical columns, which may be less than the total rows if there are missing values.

5. Use your ChatBot session to help understand the difference between an “attribute”, and a “method” and then provide your own paraphrasing summarization of that difference.

Attributes are accessed directly (without parenthese) and represnt a value or property associated with an object. For example, df.shape is an attribute of a DataFrame object. It returns a tuple representing the DataFrame dimensions (Rows, Columns). Since it’s a value, parentheses are not needed to access it.

Methods are functions associated with an object that was called with parentheses (may require arguments or need to be executed to generate a result). Methods perform operations on the object or return some data based on its state. For example, df.describe() is a method of a DataFrame object. It provides statistics of the DataFrame's columns and needs parentheses to call it and generate results.

6. The df.describe() method provides the 'count', 'mean', 'std', 'min', '25%', '50%', '75%', and 'max' summary statistics for each variable it analyzes. Give the definitions (perhaps using help from the ChatBot if needed) of each of these summary statistics.

Count: The number of values (non-missing) in the column.

Mean: The average value of the column. It's calculated as the sum of all the values divided by the count of values.

Standard Deviation: It indicates how much the values deviate from the mean in the column. A higher standard deviation indicates more spread out values.

Min: The smallest value in the column.

25%: The 25th percentile (first quartile). This is the value below which 25% of the data points fall.

50%: The 50th percentile (median). Half of the data points fall below this value, and half are above it.

75%: The 75th percentile (third quartile). This is the value below which 75% of the data points fall (top 25% of data).

Max: The largest value in the column.

ChatBot Session: https://chatgpt.com/share/71d8fbb9-41f0-4bb1-b8a7-d3a06c5e0c93

Here's a summary of our conversation:

1. Difference Between Attributes and Methods: Attributes: Values or properties associated with an object, accessed directly without parentheses (e.g., df.shape). Methods: Functions associated with an object, accessed with parentheses, which perform actions or computations (e.g., df.describe()).
2. Summary Statistics Provided by df.describe(): Count: Number of non-null values in the column. Mean: Average value of the column. Std (Standard Deviation): Measure of variation or dispersion in the column. Min: Smallest value in the column. 25%: 25th percentile (first quartile), value below which 25% of the data points fall. 50%: 50th percentile (median), middle value when data points are sorted. 75%: 75th percentile (third quartile), value below which 75% of the data points fall. Max: Largest value in the column.

7. Missing data can be considered "across rows" or "down columns". Consider how df.dropna() or del df['col'] should be applied to most efficiently use the available non-missing data in your dataset and briefly answer the following questions in your own words.

1.Provide an example of a "use case" in which using df.dropna() might be peferred over using del df['col']

2.Provide an example of "the opposite use case" in which using del df['col'] might be preferred over using df.dropna()

3.Discuss why applying del df[‘col’] before df.dropna() when both are used together could be important

4.Remove all missing data from one of the datasets you’re considering using some combination of del df[‘col’] and/or df.dropna() and give a justification for your approach, including a “before and after” report of the results of your approach for your dataset.

1. While cleaning a dataset which contains information about your store products, you notice that some of the prices are missing. You want to clean your dataset by removing rows where the column labeled “Price” has missing values because these rows may skew the data.In this case, df.dropna() should be used to remove rows with missing values in the specific column, “Price,” while preserving the rest of your dataset.
2. On the other hand, you can use del df[‘col’] if you want to completely remove an entire column (such as “Price”), regardless of whether it has missing values or not.
3. Dropping df[‘col’] first will stop dropna() from taking that column into consideration if it is not necessary for your analysis and you still intend to remove it. Applying dropna() first could have an unintended effect on rows (deleting rows that you want to keep) which would be ineffective if you want to eliminate the column completely.
4. I used df.dropna(axis=0) because I didn’t want to delete entire column categories but instead, delete the rows with missing values. The number of rows changed (891 to 182) and the number of columns remained the same (15). I demonstrated that it produced 0 missing values in all the columns using .isna().sum().

```
[12]: import pandas as pd
      url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
       ↪csv"
      titanic_df = pd.read_csv(url)
      titanic_df
```

```
[12]:      survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
      0           0       3    male  22.0      1      0   7.2500        S   Third
      1           1       1  female  38.0      1      0  71.2833        C   First
      2           1       3  female  26.0      0      0   7.9250        S   Third
      3           1       1  female  35.0      1      0  53.1000        S   First
      4           0       3    male  35.0      0      0   8.0500        S   Third
      ..        …     …     …   …    …    …      …     …
      886         0       2    male  27.0      0      0  13.0000        S  Second
      887         1       1  female  19.0      0      0  30.0000        S   First
      888         0       3  female   NaN      1      2  23.4500        S   Third
      889         1       1    male  26.0      0      0  30.0000        C   First
      890         0       3    male  32.0      0      0   7.7500        Q   Third

             who  adult_male deck  embark_town alive  alone
      0      man        True  NaN  Southampton    no  False
      1    woman       False    C    Cherbourg   yes  False
      2    woman       False  NaN  Southampton   yes   True
      3    woman       False    C  Southampton   yes  False
```

```
4      man        True  NaN  Southampton    no   True
..     …         …  …         …    …    …
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

```
[13]: titanic_df.isna().sum()
```

```
[13]: survived         0
      pclass           0
      sex              0
      age            177
      sibsp            0
      parch            0
      fare             0
      embarked         2
      class            0
      who              0
      adult_male       0
      deck           688
      embark_town      2
      alive            0
      alone            0
      dtype: int64
```

```
[16]: titanic_df.dropna(axis=0)
```

```
[16]:      survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
      1           1       1  female  38.0      1      0  71.2833        C  First
      3           1       1  female  35.0      1      0  53.1000        S  First
      6           0       1    male  54.0      0      0  51.8625        S  First
      10          1       3  female   4.0      1      1  16.7000        S  Third
      11          1       1  female  58.0      0      0  26.5500        S  First
      ..        …     …     …   …    …    …      …      …   …
      871         1       1  female  47.0      1      1  52.5542        S  First
      872         0       1    male  33.0      0      0   5.0000        S  First
      879         1       1  female  56.0      0      1  83.1583        C  First
      887         1       1  female  19.0      0      0  30.0000        S  First
      889         1       1    male  26.0      0      0  30.0000        C  First

             who  adult_male deck  embark_town alive  alone
      1    woman       False    C    Cherbourg   yes  False
      3    woman       False    C  Southampton   yes  False
```

```
6      man         True    E  Southampton    no   True
10   child        False    G  Southampton   yes  False
11   woman        False    C  Southampton   yes   True
..     …            …   …          …    …     …
871  woman        False    D  Southampton   yes  False
872    man         True    B  Southampton    no   True
879  woman        False    C    Cherbourg   yes  False
887  woman        False    B  Southampton   yes   True
889    man         True    C    Cherbourg   yes   True

[182 rows x 15 columns]
```

```
[17]: titanic_df.dropna().isna().sum()
```

```
[17]: survived       0
      pclass         0
      sex            0
      age            0
      sibsp          0
      parch          0
      fare           0
      embarked       0
      class          0
      who            0
      adult_male     0
      deck           0
      embark_town    0
      alive          0
      alone          0
      dtype: int64
```

ChatBot Session: https://chatgpt.com/share/78f602ca-15a0-4b44-930a-8f2d22ce8941

Here's a summary of our interaction:

Use of df.dropna() vs del df['col']: We discussed scenarios where df.dropna() might be preferred over del df['col'], specifically when you want to remove rows with missing values in certain columns while preserving other parts of the dataset. We also covered why applying del df['col'] before df.dropna() can be important, as it can improve efficiency, prevent unnecessary row removal, and maintain data clarity.

Deleting a Column in Python: I provided several methods to delete a column in a pandas DataFrame: Using del df['col'] to remove a column directly. Using df.drop('col', axis=1) to remove a column and return a new DataFrame. Using df.pop('col') to remove the column and return it.

Handling KeyError: We explored why a KeyError might occur, particularly when trying to access or modify a DataFrame with a non-existent column. I explained common causes like column name typos or trying to access an index that's out of range, and I provided code examples to handle and debug this issue.

This summary captures our key discussions about working with pandas DataFrames in Python, focusing on column deletion, handling missing data, and debugging errors.

8. Give brief explanations in your own words for any requested answers to the questions below.
9. Use your ChatBot session to understand what df.groupby("col1")["col2"].describe() does and then demonstrate and explain this using a different example from the "titanic" data set other than what the ChatBot automatically provide for you.

```
[2]: import pandas as pd
     url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv"
     titanic_df = pd.read_csv(url)
     titanic_df.groupby('sex')["age"].describe()
```

```
[2]:         count       mean        std   min   25%   50%   75%   max
     sex
     female  261.0  27.915709  14.110146  0.75  18.0  27.0  37.0  63.0
     male    453.0  30.726645  14.678201  0.42  21.0  29.0  39.0  80.0
```

First df.groupby("col1")["col2"].describe() groups the DataFrame (Ex: titanic_df) by the unique values in a column (Ex: Column "Sex" with unique values "male" and "female"). After grouping by the desired column, a second column (Ex: "Age") is selected for further operations. Finally, the describe() function generates a summary of statistics for each group in "col2."

2. Assuming you've not yet removed missing values in the manner of question "7" above, df.describe() would have different values in the count value for different data columns depending on the missingness present in the original data. Why do these capture something fundamentally different from the values in the count that result from doing something like df.groupby("col1")["col2"].describe()?

df.describe() looks at each column independently, so it captures the missingness within the column itself. For example, titanic_df['age'].describe() would present all non-missing values in the entered column (714). On the other hand, df.groupby("col1")["col2"].describe() reflects the interaction between the two given columns. It demonstrates how the values of "col2" are distributed within the unique groups in "col1," and only considers the non-missing values for col2 in the grouped statistics. For example, titanic_df.groupby('sex')["age"].describe() would present the count for non-missing values in "age," grouped by the unique values in column "sex" (see question 8, part 1).

```
[8]: titanic_df['age'].describe()
```

```
[8]: count    714.000000
     mean      29.699118
     std       14.526497
     min        0.420000
     25%       20.125000
     50%       28.000000
     75%       38.000000
```

```
max      80.000000
Name: age, dtype: float64
```

3. Intentionally introduce the following errors into your code and report your opinion as to whether it's easier to (a) work in a ChatBot session to fix the errors, or (b) use google to search for and fix errors: first share the errors you get in the ChatBot session and see if you can work with ChatBot to troubleshoot and fix the coding errors, and then see if you think a google search for the error provides the necessary toubleshooting help more quickly than ChatGPT

For errors A-F, ChatGPT was easier to work with to fix the errors. It provided possible reasons for errors in a concise manner with examples to accompany the explanation. For example, for error A, ChatGPT explained that "you might not have imported pandas or you're using it without the standard alias," which is true. It further suggested the code "import pandas as pd" to fix the error. Though similar answers can be found on google, they require more time and effort to find and the language used in the explanation is often too complex to be understood by a beginner.

The response from both ChatGPT and google weren't helpful for error G because the NameError led both to assume that the variable wasn't defined. However, defining the variable is not the solution, changning it to type string by adding quotations is.

ChatBot Session: https://chatgpt.com/share/3ccf749d-1e78-4966-a463-830b8080a948

Here's a summary of our interaction:

- GroupBy Operation: You asked about the functionality of df.groupby("col1")["col2"].describe(). I explained that it groups the DataFrame by col1 and then provides descriptive statistics for col2 within each group.
- NameError Troubleshooting: You encountered several NameError issues: NameError: name 'sex' is not defined: I suggested checking for typos, ensuring sex is defined or exists as a column in your DataFrame, and verifying its scope. NameError: name 'age' is not defined: Similar advice was given for the age variable or column, including checking its definition, spelling, and scope.
- HTTP Error 404: You mentioned an HTTP 404 error, and I provided general troubleshooting steps for dealing with missing resources or incorrect URLs.
- AttributeError: You had an AttributeError related to 'SeriesGroupBy' object has no attribute 'describje'. I corrected the typo to 'describe'.

9. Have you reviewed the course wiki-textbook and interacted with a ChatBot (or, if that wasn't sufficient, real people in the course piazza discussion board or TA office hours) to help you understand all the material in the tutorial and lecture that you didn't quite follow when you first saw it?

Yes.