Understanding Data Structure and Column Contents

using Python

Dr. Austin Brown

Kennesaw State University

Understanding Column Contents

- **Objective**: Learn to differentiate between numeric and character data, and identify missing data.
- **Importance**: Essential for correct data processing and analysis.
- **Key Points**
  - **info()**: Displays metadata about the dataframe, including data types and missing values.
  - **isnull().sum()**: Identifies the number of missing values in each column.
  - **Example**: Display metadata about a dataset and its missing values.

Data Structure

- In any given dataset, the way the data are arranged is paramount for understanding what the data are and what information they contain.
- Generally, we want the way data are recorded within a dataset to be "tidy".
- Tidy data is data that is well-organized and easy to work with. It has a specific structure:
  - Each variable is a column.
  - Each observation is a row.
  - Each cell has a single value.

Data Structure

- For instance, it the `cars` dataset below, the data are arranged in a tidy way:

	mpg	cyl	disp	hp	drat
Mazda RX4	21.0	6	160	110	3.90
Mazda RX4 Wag	21.0	6	160	110	3.90
Datsun 710	22.8	4	108	93	3.85
Hornet 4 Drive	21.4	6	258	110	3.08
Hornet Sportabout	18.7	8	360	175	3.15
Valiant	18.1	6	225	105	2.76

Data Structure

- Why are they considered tidy? Consider our above criteria:
  - Each variable is a column: `mpg`, `cyl`, `disp`, `hp`, `drat`.
  - Each observation is a row: Each row represents a different car.
  - Each cell has a single value: Each cell contains a single value for the variable it represents.

Data Types: Numeric vs. Character Data

- Now that we know how data should be structured, let's talk about the types of data we might encounter.
- In Python, there are two main types of variables:
  - **Numeric or Quantitative**: These are variables which are naturally measured using numbers. Variables like age, height, weight, etc. are examples of numeric variables.
  - **Character or Qualitative**: These are variables which are naturally measured using non-quantitative qualities. Variables like hair color, favorite food, etc. are examples of character variables.

Data Types: Numeric vs. Character Data

- Note, when we read data into Python, based on the values contained in each column, Python will automatically assign a data type to each column.
- Thus, it is important for us to double-check to ensure that the data types as we understand them are also the way Python has interpreted them.

Data Types: Numeric vs. Character Data

- For example, suppose we want to use the Cars.csv dataset. We have already learned how to read that data in. But how do we check to see what data types Python has assigned to each column?
- To do this, we can use the `info()` method from the Pandas library.
- The `info()` method provides metadata about the dataframe, including data types and missing values.

Data Types: Numeric vs. Character Data

```python
## Import pandas library ##
import pandas as pd
## Read in Cars.csv ##
cars = pd.read_csv("Cars.csv")
## Display metadata about the dataframe ##
print(cars.info())
```

Data Types: Numeric vs. Character Data

- As we can see from the output, we get a list of all the columns in the dataset, along with the data type of each column.
- Note, variables with `object` data type are generally character variables.
- Variables with `int64` or `float64` data types are generally numeric variables.
- As we can see, we have 12 variables in the dataset: 11 numeric and 1 character.

Data Types: Numeric vs. Character Data

- Now, let's read in the NYC Airplanes 2013 Excel dataset and perform the same operation.

```
## Read in Airplanes Data ##
airplanes = pd.read_excel('NYC Airplanes 2013.xlsx')
## Explore Data Structure ##
print(airplanes.info())
```

Data Types: Numeric vs. Character Data

- As we can see from the output, we have five character columns and four numeric columns.
- We also have 3322 observations (or rows) and 9 columns (or variables)

Producing Column Summaries: Missing Data

- In addition to understanding the data types of each column, it is also important to understand the contents of each column.
- One very common issue that arises when working with data is missing data.
- Missing data can be problematic for many reasons, including:
  - It can lead to biased results.
  - It can lead to incorrect conclusions.
  - It can lead to incorrect inferences.
- Thus, it is important to identify and address missing data in our datasets before proceeding with any analysis.

Producing Column Summaries: Missing Data

- How do we do this? One way is to use the `isnull().sum()` method in Python.
- The `isnull().sum()` method provides the number of missing values in each column.
- Let's see how this works with the `cars` dataset.

Producing Column Summaries: Missing Data

```
## Identify missing values ##
print(cars.isnull().sum())
print(airplanes.isnull().sum())
```

Producing Column Summaries: Missing Data

- As we can see, the `cars` dataset contains no missing values whereas the `airplanes` dataset has two variables which contain missing values: `year` and `speed`.
- `year` contains 70 missing values and `speed` contains 3299.
- It is important to identify and address missing values before proceeding with any analysis.