

# Data Wrangling

Brian Kim

2024-11-20

## Basic

### Import required packages

```
import pandas as pd
import matplotlib.pyplot as plt
```

- **pandas**: Library primarily used for data manipulation and analysis
  - **matplotlib**: An object-oriented plotting library.
- 

### Load dataset

```
dataset = pd.read_csv('smoker.csv')
```

- `pd.read_csv('smoker.csv')`: `pd` is a common abbreviation for the pandas library in Python
  - `read_csv()` is a pandas function specifically designed to read data from comma-separated value (CSV) files.
  - `'smoker.csv'` specifies the path to the CSV file.
- 

### Inspect dimension

```
dataset.shape
```

(10000, 3)

- Return a tuple representing the dimensionality of the DataFrame.
- 

## Table information(Rows, columns, etc)

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   smoker      10000 non-null  int64
 1   treatment   10000 non-null  int64
 2   outcome     10000 non-null  int64
dtypes: int64(3)
memory usage: 234.5 KB
```

- Print a concise summary of a DataFrame.
- 

This DataFrame might represent data from a study with:

- **smoker**: A binary indicator (e.g., 0 for non-smokers, 1 for smokers).
  - **treatment**: A binary variable indicating whether a treatment was applied (e.g., 0 for no treatment, 1 for treatment).
  - **outcome**: A numerical outcome related to the study.
- 

## Check first few rows

```
dataset.head()
```

	smoker	treatment	outcome
0	0	0	0
1	1	1	1
2	2	1	5
3	3	1	67
4	4	0	8

**Check first few rows**

```
dataset.head(7)
```

	smoker	treatment	outcome
0	0	0	0
1	1	1	1
2	2	1	5
3	3	1	67
4	4	0	8
5	5	0	0
6	0	5	0

**Check last few rows**

```
dataset.tail()
```

	smoker	treatment	outcome
9995	1	1	0
9996	0	0	1

---

	smoker	treatment	outcome
9997	0	0	0
9998	1	1	0
9999	1	1	0

---



---

### Calculate mean of column

```
dataset['treatment'].mean()
```

0.3032

---

### Count rows of same value

```
dataset['treatment'].value_counts()
```

```
treatment
0    6991
1    3004
5         1
6         1
7         1
8         1
2         1
Name: count, dtype: int64
```

- Return a Series containing counts of unique values.
- 

### Count rows of same value

```
dataset['outcome'].value_counts()
```

```
outcome
0      7324
1      2665
5         2
8         2
6         2
67        1
9         1
7         1
4         1
3         1
Name: count, dtype: int64
```

---

### Sum of column

```
dataset.sum()
```

```
smoker      3055
treatment   3032
outcome     2793
dtype: int64
```

- Return the sum of the values over the requested axis.
- 

### Sum of row

```
dataset.sum(axis = 1)
```

```

0      0
1      3
2      8
3     71
4     12
..
9995    2
9996    1
9997    0
9998    2
9999    2
Length: 10000, dtype: int64

```

- `axis = 0` (the default if you omit `axis`) would sum the values in each column.
- `axis = 1` sums the values in each row, resulting in a Pandas Series with the sum for each row.

---

## Statistics

```
dataset.describe()
```

	smoker	treatment	outcome
count	10000.000000	10000.000000	10000.000000
mean	0.305500	0.303200	0.279300
std	0.470948	0.475702	0.823383
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	1.000000	1.000000	1.000000
max	5.000000	8.000000	67.000000

- Generate descriptive statistics.
-

## 1. Summary Statistics

- **count:** The number of non-missing values in each column. All columns have 10,000 entries, indicating no missing data.
  - **mean:** The average value for each column.
  - **std (Standard Deviation):** The spread or variability of the values in each column.
  - **min:** The minimum value in each column.
  - **25% (First Quartile):** The value below which 25% of the data falls.
  - **50% (Median):** The middle value in the dataset when sorted.
  - **75% (Third Quartile):** The value below which 75% of the data falls.
  - **max:** The maximum value in each column.
- 

## 2. Insights for Each Column

### `smoker`

- **Description:** Represents smoking status or level.
  - **Range:** 0 to 5.
  - **Mean:** 0.3055, indicating most individuals fall in the lower range.
  - **Quartiles:** The 25th, 50th, and 75th percentiles are all 0 or 1, showing that most observations are concentrated in the lower values, likely indicating that most participants are non-smokers or light smokers.
  - **Max:** A value of 5 suggests possible categories beyond smoker/non-smoker.
- 

### `treatment`

- **Description:** Represents whether a treatment was applied or its intensity.
  - **Range:** 0 to 8.
  - **Mean:** 0.3032, showing that treatments were applied infrequently on average.
  - **Quartiles:** The 25th, 50th, and 75th percentiles are all either 0 or 1, suggesting most participants received no or minimal treatment.
  - **Max:** A value of 8 suggests a scale or multiple levels of treatment.
-

**outcome**

- **Description:** Represents the result or response to smoking/treatment.
  - **Range:** 0 to 67.
  - **Mean:** 0.2793, indicating that outcomes are generally low on average.
  - **Quartiles:**
    - The 25th, 50th, and 75th percentiles are all 0 or 1, showing that most outcomes are low, with a few high values driving the upper range.
  - **Max:** A value of 67 indicates a significant outlier or rare event with an extreme outcome.
- 

### 3. Key Observations

#### 1. **smoker and treatment Are Skewed:**

- Both have a mean below 0.31, with most observations clustered around 0 or 1.
- Their maximum values (5 for **smoker** and 8 for **treatment**) suggest a few high values or categories.

#### 2. **outcome Has High Variability:**

- A large standard deviation (0.823) compared to the mean (0.2793) indicates high variability.
  - A max of 67, compared to an interquartile range of 0–1, suggests significant outliers.
- 

### 4. Recommendations for Further Analysis

#### 1. **Explore Distribution:**

- Visualize the data with histograms or boxplots to confirm the skewness and outliers.
- Investigate the meaning of high values in **smoker**, **treatment**, and **outcome**.

#### 2. **Analyze Relationships:**

- Check correlations between **smoker**, **treatment**, and **outcome**.
  - Stratify data by **smoker** levels to see if **treatment** impacts **outcome** differently for different smoking statuses.
-



### 3. Outlier Investigation:

- Examine rows with `smoker = 5`, `treatment = 8`, or `outcome > 1` to understand if these represent valid cases or data entry issues.

### 4. Statistical Tests:

- Test the impact of `smoker` and `treatment` on `outcome` using regression or ANOVA.

---

## Statistics

```
dataset.describe().round(2)
```

	smoker	treatment	outcome
count	10000.00	10000.00	10000.00
mean	0.31	0.30	0.28
std	0.47	0.48	0.82
min	0.00	0.00	0.00
25%	0.00	0.00	0.00
50%	0.00	0.00	0.00
75%	1.00	1.00	1.00
max	5.00	8.00	67.00

- Generate descriptive statistics.

## Use AI in coding?

Question: How to round down to two decimal places?

Answer from AI:

Several methods exist to round down a Pandas `describe()` output to two decimal places. Here are a few, assuming your `dataset` is a Pandas DataFrame:

### Method 1: Using `.applymap()`

This method applies a rounding function to every element in the `describe()` output. This is generally the most straightforward approach.

```
import pandas as pd
import numpy as np

# ... your code to load and prepare dataset ...

rounded_description = dataset.describe().applymap(lambda x: np.floor(x * 100) / 100)
print(rounded_description)
```

`np.floor` rounds down to the nearest integer. We multiply by 100, round, then divide by 100 to achieve two decimal places.

### Method 2: Using `style.format()` (for better display)

This method is better for displaying the results nicely, especially in Jupyter Notebooks or similar environments. It doesn't change the underlying data, only how it's presented.

```
import pandas as pd

# ... your code to load and prepare dataset ...

styled_description = dataset.describe().style.format("{:.2f}")
print(styled_description) #or display(styled_description) in Jupyter
```

`"{:.2f}"` is a format specifier that rounds to two decimal places.

### Method 3: Rounding individual columns (more control)

If you only want to round specific columns, you can target them individually:

```
import pandas as pd

# ... your code to load and prepare dataset ...

for col in dataset.describe().columns:
    dataset.describe()[col] = np.floor(dataset.describe()[col] * 100) / 100

print(dataset.describe())
```

This offers more granular control but involves more typing if you have many columns.

### Choosing the right method:

- **Method 1:** Best if you need the rounded numbers for further calculations.

- **Method 2:** Best for visually presenting the rounded data, especially in interactive environments.
- **Method 3:** Best if you only need to round specific columns or want to avoid rounding certain statistics (e.g., count).

Remember to replace `# ... your code to load and prepare dataset ...` with your actual code to load and prepare your `dataset` DataFrame. Choose the method that best suits your needs and context. Method 2 is generally recommended for simple display purposes due to its clarity and ease of use.