Exemplar_Explore probability distributions

January 7, 2024

1 Exemplar: Explore probability distributions

1.1 Introduction

The ability to determine which type of probability distribution best fits data, calculate z-score, and detect outliers are essential skills in data work. These capabilities enable data professionals to understand how their data is distributed and identify data points that need further examination.

In this activity, you are a member of an analytics team for the United States Environmental Protection Agency (EPA). The data includes information about more than 200 sites, identified by state, county, city, and local site names. One of your main goals is to determine which regions need support to make air quality improvements. Given that carbon monoxide is a major air pollutant, you will investigate data from the Air Quality Index (AQI) with respect to carbon monoxide.

1.2 Step 1: Imports

Import relevant libraries, packages, and modules. For this lab, you will need `numpy`, `pandas`, `matplotlib.pyplot`, `statsmodels.api`, and `scipy`.

```
[1]: # Import relevant libraries, packages, and modules.

     ### YOUR CODE HERE ###

     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import statsmodels.api as sm
     from scipy import stats
```

A subset of data was taken from the air quality data collected by the EPA, then transformed to suit the purposes of this lab. This subset is a .csv file named `modified_c4_epa_air_quality.csv`. Load into a DataFrame.

```
[2]: # Load data into a DataFrame and save in a variable.

     ### YOUR CODE HERE ###

     data = pd.read_csv("modified_c4_epa_air_quality.csv")
```

Hint 1

Refer to what you learned about loading data in Python.

Hint 2

There is a function in the `pandas` library that allows you to load data from a .csv file into a DataFrame.

Hint 3

Use the `read_csv()` function and pass in the name of the csv file as a string.

1.3 Step 2: Data exploration

Display the first 10 rows of the data to get a sense of how the data is structured.

```
[3]: # Display first 10 rows of the data.

     ### YOUR CODE HERE ###


     data.head(10)
```

```
[3]:    date_local    state_name    county_name      city_name  \
     0  2018-01-01       Arizona       Maricopa         Buckeye
     1  2018-01-01          Ohio        Belmont       Shadyside
     2  2018-01-01       Wyoming          Teton   Not in a city
     3  2018-01-01  Pennsylvania   Philadelphia    Philadelphia
     4  2018-01-01          Iowa           Polk      Des Moines
     5  2018-01-01        Hawaii       Honolulu   Not in a city
     6  2018-01-01        Hawaii       Honolulu   Not in a city
     7  2018-01-01  Pennsylvania           Erie            Erie
     8  2018-01-01        Hawaii       Honolulu        Honolulu
     9  2018-01-01      Colorado        Larimer    Fort Collins

                                          local_site_name    parameter_name  \
     0                                             BUCKEYE   Carbon monoxide
     1                                           Shadyside   Carbon monoxide
     2  Yellowstone National Park - Old Faithful Snow …   Carbon monoxide
     3                             North East Waste (NEW)   Carbon monoxide
     4                                           CARPENTER   Carbon monoxide
     5                                             Kapolei   Carbon monoxide
     6                                             Kapolei   Carbon monoxide
     7                                                 NaN   Carbon monoxide
     8                                            Honolulu   Carbon monoxide
     9                     Fort Collins - CSU - S. Mason   Carbon monoxide

           units_of_measure    aqi_log
```

```
0  Parts per million  2.079442
1  Parts per million  1.791759
2  Parts per million  1.098612
3  Parts per million  1.386294
4  Parts per million  1.386294
5  Parts per million  2.708050
6  Parts per million  1.098612
7  Parts per million  1.098612
8  Parts per million  1.791759
9  Parts per million  1.945910
```

Hint 1

Refer to what you learned about exploring datasets in Python.

Hint 2

There is a function in the **pandas** library that allows you to display a specific number of rows from the top of a DataFrame.

Hint 3

Use the **head()** function and pass in how many rows from the top of the DataFrame you want to display.

The **aqi_log** column represents AQI readings that were transformed logarithmically to suit the objectives of this lab.

To better understand the quantity of data you are working with, display the number of rows and the number of columns.

```
[4]:  # Display number of rows, number of columns.

      ### YOUR CODE HERE ###

      data.shape
```

[4]: (260, 8)

Hint 1

Refer to what you learned about exploring datasets in Python.

Hint 2

Every DataFrame in **pandas** has a property that gives you access to the number of rows and number of columns in that DataFrame.

Hint 3

Call the **shape** property of the DataFrame, which will display the number of rows and the number of columns as a tuple.

Now, you want to find out whether `aqi_log` fits a specific type of probability distribution. Create a histogram to visualize the distribution of `aqi_log`. Then, based on its shape, visually determine if it resembles a particular distribution.

```
[5]:  # Create a histogram to visualize distribution of aqi_log.

      ### YOUR CODE HERE ###

      data["aqi_log"].hist();
```
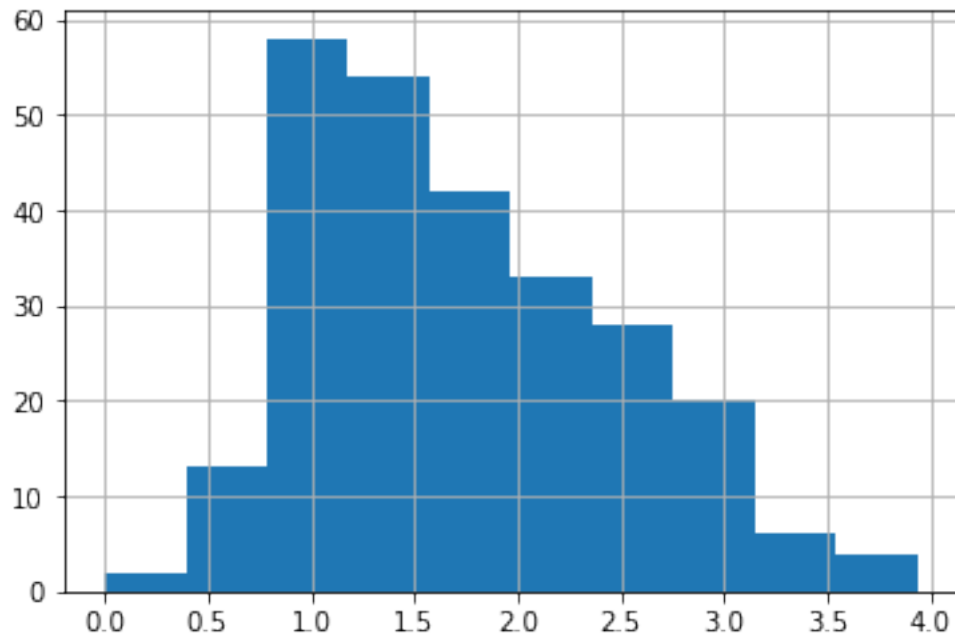

Hint 1

Refer to the video about creating a histogram to visualize the distribution of a particular variable in the data.

Hint 2

There is a function in the `matplotlib` library that can be called to create a histogram.

Hint 3

The `hist()` function can be called directly on the `aqi_log` column from the data.

A semicolon can be used at the end as a quick way to make sure only the plot gets displayed (other text does not get displayed).

Question: What do you observe about the shape of the distribution from the histogram?

There is a slight right skew, but it still appears to be a bell shape. This shape suggests that the distribution of this data should be approximately normal.

1.4   Step 3: Statistical tests

Use the empirical rule to observe the data, then test and verify that it is normally distributed.

As you have learned, the empirical rule states that, for every normal distribution: - 68% of the data fall within 1 standard deviation of the mean - 95% of the data fall within 2 standard deviations of the mean - 99.7% of the data fall within 3 standard deviations of the mean

First, define two variables to store the mean and standard deviation, respectively, for `aqi_log`. Creating these variables will help you easily access these measures as you continue with the calculations involved in applying the empirical rule.

```
[6]:  # Define variable for aqi_log mean.

      ### YOUR CODE HERE ###

      mean_aqi_log = data["aqi_log"].mean()


      # Print out the mean.

      ### YOUR CODE HERE ###

      print(mean_aqi_log)
```

```
1.7669210929985577
```

```
[7]:  # Define variable for aqi_log standard deviation.

      ### YOUR CODE HERE ###

      std_aqi_log = data["aqi_log"].std()


      # Print out the standard deviation.

      ### YOUR CODE HERE ###

      print(std_aqi_log)
```

```
0.7147155520223721
```

Hint 1

Refer to the lesson about calculating the mean and standard deviation for a particular variable in the data.

Hint 2

There are functions in the `numpy` library that can be called to calculate mean and standard deviation, respectively.

Hint 3

The `mean()` function can be called directly on the `aqi_log` column from the data to compute the mean.

The `std()` function can be called directly on the `aqi_log` column from the data to compute the standard deviation.

Now, check the first part of the empirical rule: whether 68% of the `aqi_log` data falls within 1 standard deviation of the mean.

To compute the actual percentage of the data that satisfies this criteria, define the lower limit (for example, 1 standard deviation below the mean) and the upper limit (for example, 1 standard deviation above the mean). This will enable you to create a range and confirm whether each value falls within it.

```
[8]:   # Define variable for lower limit, 1 standard deviation below the mean.

       ### YOUR CODE HERE ###

       lower_limit = mean_aqi_log - 1 * std_aqi_log


       # Define variable for upper limit, 1 standard deviation above the mean.

       ### YOUR CODE HERE ###

       upper_limit = mean_aqi_log + 1 * std_aqi_log


       # Display lower_limit, upper_limit.

       ### YOUR CODE HERE ###

       print(lower_limit, upper_limit)
```

1.0522055409761855 2.48163664502093

Hint 1

Refer to the video about using the empirical rule.

Hint 2

The lower limit here is $mean - 1 * std$.

The upper limit here is $mean + 1 * std$.

The `print` function can be called to display.

Hint 3

Use the variables that you defined for mean and standard deviation of `aqi_log`, ensuring the spelling is correct.

Call the **print** function and pass in the values one after the other, with a comma between them.

```
[9]: # Display the actual percentage of data that falls within 1 standard deviation␣
     ↪of the mean.

     ### YOUR CODE HERE ###

     ((data["aqi_log"] >= lower_limit) & (data["aqi_log"] <= upper_limit)).mean() *␣
     ↪100
```

```
[9]: 76.15384615384615
```

Hint 1

Refer to the video about using the empirical rule.

Hint 2

The **>=** operator can be used to confirm whether one value is greater than or equal to another value.

The **<=** operator can be used to check whether one value is less than or equal to another value.

The **&** operator can be used to check if one condition and another condition is met.

Hint 3

The **mean()** function can be used to compute the proportion of the data that satisfies the specified conditions.

Multiplying that proportion by 100 can get you the percentage.

Now, consider the second part of the empirical rule: whether 95% of the **aqi_log** data falls within 2 standard deviations of the mean.

To compute the actual percentage of the data that satisfies this criteria, define the lower limit (for example, 2 standard deviations below the mean) and the upper limit (for example, 2 standard deviations above the mean). This will enable you to create a range and confirm whether each value falls within it.

```
[10]: # Define variable for lower limit, 2 standard deviations below the mean.

      ### YOUR CODE HERE ###


      lower_limit = mean_aqi_log - 2 * std_aqi_log


      # Define variable for upper limit, 2 standard deviations below the mean.

      ### YOUR CODE HERE ###

      upper_limit = mean_aqi_log + 2 * std_aqi_log
```

```python
# Display lower_limit, upper_limit.

### YOUR CODE HERE ###

print(lower_limit, upper_limit)
```

0.33748998895381344 3.1963521970433018

Hint 1

Refer to the video about using the empirical rule.

Hint 2

The lower limit here is $mean - 2 * std$.

The upper limit here is $mean + 2 * std$.

The `print` function can be called to display.

Hint 3

Use the variables that you defined for mean and standard deviation of `aqi_log`, ensuring the spelling is correct.

Call the `print` function and pass in the values one after the other, with a comma between them.

```python
[11]: # Display the actual percentage of data that falls within 2 standard deviations
      ↪of the mean.

      ### YOUR CODE HERE ###

      ((data["aqi_log"] >= lower_limit) & (data["aqi_log"] <= upper_limit)).mean() *
      ↪100
```

[11]: 95.76923076923077

Hint 1

Refer to the video section about using the empirical rule.

Hint 2

The `>=` operator can be used to confirm whether one value is greater than or equal to another value.

The `<=` operator can be used to check whether one value is less than or equal to another value.

The `&` operator can be used to check if one condition and another condition is met.

Hint 3

The `mean()` function can be used to compute the proportion of the data that satisfies the specified conditions.

Multiplying that proportion by 100 can get you the percentage.

Now, consider the third part of the empirical rule:whether 99.7% of the `aqi_log` data falls within 3 standard deviations of the mean.

To compute the actual percentage of the data that satisfies this criteria, define the lower limit (for example, 3 standard deviations below the mean) and the upper limit (for example, 3 standard deviations above the mean). This will enable you to create a range and confirm whether each value falls within it.

```
[12]: # Define variable for lower limit, 3 standard deviations below the mean.

      ### YOUR CODE HERE ###

      lower_limit = mean_aqi_log - 3 * std_aqi_log


      # Define variable for upper limit, 3 standard deviations above the mean.

      ### YOUR CODE HERE ###

      upper_limit = mean_aqi_log + 3 * std_aqi_log


      # Display lower_limit, upper_limit.

      ### YOUR CODE HERE ###

      print(lower_limit, upper_limit)
```

-0.3772255630685586 3.911067749065674

Hint 1

Refer to the video about using the empirical rule.

Hint 2

The lower limit here is $mean - 3 * std$.

The upper limit here is $mean + 3 * std$.

The `print` function can be called to display.

Hint 3

Use the variables that you defined for mean and standard deviation of `aqi_log`, ensuring the spelling is correct.

Call the `print` function and pass in the values one after the other, with a comma between them.

```
[13]: # Display the actual percentage of data that falls within 3 standard deviations
      ↪of the mean.

      ### YOUR CODE HERE ###
```

```
((data["aqi_log"] >= lower_limit) & (data["aqi_log"] <= upper_limit)).mean() *␣
 ↪100
```

[13]: 99.61538461538461

Hint 1

Refer to the video about using the empirical rule.

Hint 2

The >= operator can be used to confirm whether one value is greater than or equal to another value.

The <= operator can be used to check whether one value is less than or equal to another value.

The & operator can be used to check if one condition and another condition is met.

Hint 3

The `mean()` function can be used to compute the proportion of the data that satisfies the specified conditions.

Multiplying that proportion by 100 can get you the percentage.

1.5 Step 4: Results and evaluation

Question: What results did you attain by applying the empirical rule? - About 76.15% of the data falls within 1 standard deviation of the mean. - About 95.77% of the data falls within 2 standard deviation of the mean. - About 99.62% of the data falls within 3 standard deviations of the mean.

The 95.77% is very close to 95%, and the 99.62% is very close to 99.7%. The 76.15% is not as close to 68%, but relatively close. Overall, from applying the empirical rule, the data appears to be not exactly normal, but could be considered approximately normal.

Question: How would you use z-score to find outliers?

Since z-score indicates the relative position of values (for instance, z-score measures how many standard deviations below or above the mean a data point is), it can be used to detect outliers.

Z-score could be used to identify values that lie more than 3 standard deviations below or above the mean. These values may be considered outliers.

Compute the z-score for every `aqi_log` value. Then, add a column named `z_score` in the data to store those results.

```
[14]: # Compute the z-score for every aqi_log value, and add a column named z_score␣
      ↪in the data to store those results.

      ### YOUR CODE HERE ###

      data["z_score"] = stats.zscore(data["aqi_log"], ddof=1) # ddof=degrees of␣
      ↪freedom correction (sample vs. population)
```

```
# Display the first 5 rows to ensure that the new column was added.

### YOUR CODE HERE ###

data.head()
```

[14]:   date_local   state_name   county_name   city_name \
       0   2018-01-01   Arizona   Maricopa   Buckeye
       1   2018-01-01   Ohio   Belmont   Shadyside
       2   2018-01-01   Wyoming   Teton Not in a city
       3   2018-01-01 Pennsylvania Philadelphia   Philadelphia
       4   2018-01-01   Iowa   Polk   Des Moines

                                              local_site_name   parameter_name \
       0   BUCKEYE   Carbon monoxide
       1   Shadyside   Carbon monoxide
       2   Yellowstone National Park - Old Faithful Snow … Carbon monoxide
       3   North East Waste (NEW)   Carbon monoxide
       4   CARPENTER   Carbon monoxide

           units_of_measure   aqi_log   z_score
       0   Parts per million 2.079442   0.437265
       1   Parts per million 1.791759   0.034753
       2   Parts per million 1.098612 -0.935070
       3   Parts per million 1.386294 -0.532557
       4   Parts per million 1.386294 -0.532557

Hint 1

Refer to the video about calculating z-score.

Hint 2

There is a function in the **stats** module of the **scipy** library that you can call to calculate z-score.

Hint 3

Call the **zscore()** function and pass in the **aqi** column from the data.

Identify the parts of the data where **aqi_log** is above or below 3 standard deviations of the mean.

```
[15]: # Display data where `aqi_log` is above or below 3 standard deviations of the
      ↪mean

      ### YOUR CODE HERE ###

      data[(data["z_score"] > 3) | (data["z_score"] < -3)]
```

```
[15]:       date_local state_name county_name city_name local_site_name  \
       244  2018-01-01   Arizona    Maricopa   Phoenix    WEST PHOENIX

            parameter_name  units_of_measure   aqi_log   z_score
       244  Carbon monoxide Parts per million  3.931826  3.029044
```

Hint 1

Refer to the video about outlier detection.

Hint 2

The > operator can be used to evaluate whether one value is greater than another value.

The < operator can be used to evaluate whether one value is less than another value.

The | operator can used to evaluate whether one condition or another condition is met.

Hint 3

To index the DataFrame, place a pair of parantheses around the evaluation of the two conditions and pass that into a pair of square brackets. This will allow you to get all rows in the data where the specified criteria is met.

Make sure the spelling of the column matches the name you specified when creating that column.

Question: What do you observe about potential outliers based on the calculations?

The `aqi_log` for West Phoenix is slightly above 3 standard deviations of the mean. This means that the air quality at that site is worse than the rest of the sites represented in the data.

Question: Why is outlier detection an important part of this project?

Detecting outliers is important because they can reveal two important things, depending on the context: First, they can identify measurements that were taken incorrectly. Second, they can highlight parts of the data that can be focused on to make improvements.

For example, if the `aqi_log` for West Phoenix is considered an outlier, then that site can be studied further to determine what practices or changes might improve the air quality.

1.6 Considerations

What are some key takeaways that you learned during this lab?

- Plotting the data using a histogram, then observing the shape, enables you to visually determine whether the data is normally distributed.
- The empirical rule can be used to verify whether a distribution is normal.
- The mean and standard deviation are important measures when applying the empirical rule to a distribution.
- Z-score allows you to identify potenial outliers in the data.

What summary would you provide to stakeholders? Consider the distribution of the data and which sites would benefit from additional research. - The distribution of the `aqi_log` data is approximately normal. - Using statistical methods, it was determined that the site

at West Phoenix has worse air quality than the other sites. - Consider allocating more resources toward further examining this site in order to improve its air quality.

Reference

US EPA, OAR. 2014, July 8. Air Data: Air Quality Data Collected at Outdoor Monitors Across the US.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.