Noteable™

Measuring Data

Measuring data in Python

Resource

Secondary

14-18 years

EDINA THE UNIVERSITY of EDINBURGH

Noteable™

Contents

Noteable Activities for Schools: Measuring Datasets

These resources are a guide for teachers to demonstrate to the whole class or direct individual students as appropriate. The activities below can be directly distributed to pupils.

For instructions on how to install and use Noteable resources, please look at our guides for teachers in GLOW: GLOW guidance for teachers to start using Noteable.

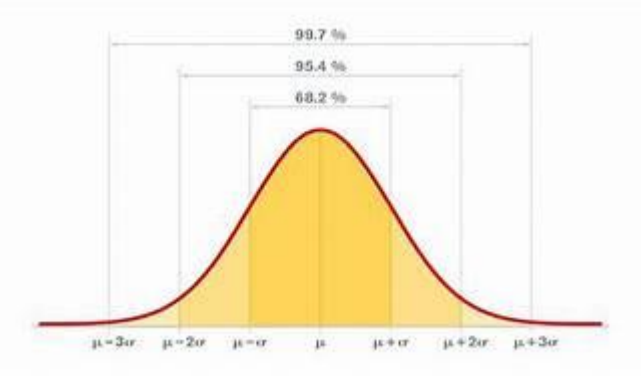
Content and Curriculum links

Level	Context	Indicators
14-18	Creating measurements to analyse data	Average Standard deviation analysis

Knowledge	Using bullet point lists to give instructions
Curriculum links (England) Computing KS3 Computing KS4	• use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions • develop and apply their analytic, problem-solving, design, and computational thinking skills
Scottish Curriculum for Excellence	**Experiences and Outcomes:** • I understand constructs and data structures in a textual programming language **TCH 4-14a** **Benchmark:** ○ Understands basic control constructs such as sequence, selection repetition, variables and numerical calculations in a textual language. ○ Identifies and explains syntax errors in a program written in a textual language. ○ Demonstrates an understanding of representations of data structures in a textual language.

All: Cross-curricular opportunities	Mathematics, English, Science, Social Studies, Art, Technology, Physical Education and Music

Measuring Data in Python



When working with data in Python, it is often useful to calculate the average and standard deviation of the data. The average, also known as the mean, is the sum of all the data points divided by the number of data points. The standard deviation is a measure of how spread out the data is from the mean. It is calculated by taking the square root of the variance, which is the average of the squared differences between each data point and the mean.

Python provides several libraries for calculating the average and standard deviation of data, including the statistics library, NumPy, and Pandas. The statistics library provides functions for calculating the mean and standard deviation of a sample of data. NumPy provides functions for calculating the mean and standard deviation of an entire population of data. Pandas provides functions for calculating the mean and standard deviation of data in a DataFrame.

Here is an example Python script that calculates the average and standard deviation of a list of numbers using the statistics library:

Python Copy code

```
import statistics


data = [1, 2, 3, 4, 5]

mean = statistics.mean(data)

stdev = statistics.stdev(data)


print("Mean:", mean)
```

```
print("Standard Deviation:", stdev)
```

This script will output the following:

Mean: 3

Standard Deviation: 1.5811388300841898

The mean of the data is 3, and the standard deviation is approximately 1.58.

If you have a large dataset, it may be more efficient to use NumPy or Pandas to calculate the average and standard deviation. Here is an example Python script that calculates the average and standard deviation of a NumPy array:

Python Copy code

```
import numpy as np


data = np.array([1, 2, 3, 4, 5])

mean = np.mean(data)

stdev = np.std(data)


print("Mean:", mean)

print("Standard Deviation:", stdev)
```

This script will output the same result as the previous script:

Mean: 3.0

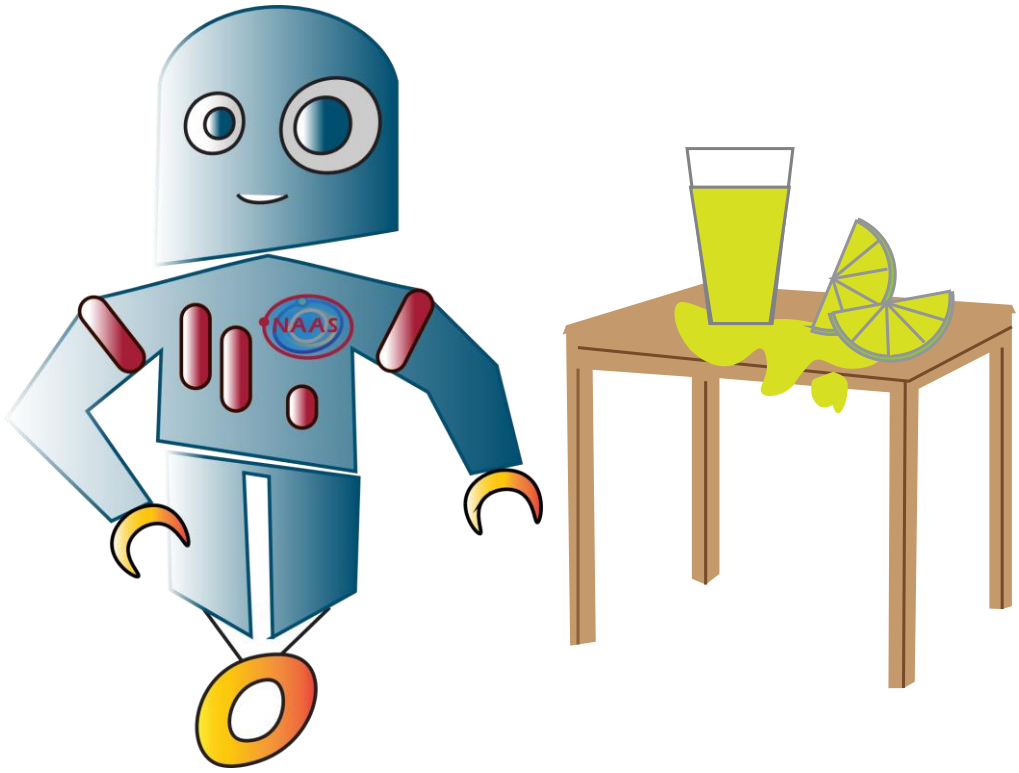
Standard Deviation: 1.4142135623730951

Note that the standard deviation calculated by NumPy is slightly different from the one calculated by the statistics library. This is because NumPy uses a slightly different formula to calculate the standard deviation.

Activity 1

NAAS, your robot friend, decided to sell his lemonade online. He created a webpage and wants to check how well his lemonade is doing by asking for feedback ratings of 1 to 5 starts for his lemonade. As each rating is received, a server-side script is used to recalculate the average rating for the lemonade and upload the average rating to the website. The webserver stores the total number of ratings received along with the overall average rating.

Using Python, write an algorithm for this script.



Activity 1 – Answer (for teachers)

NAAS, your robot friend, decided to sell his lemonade online. He created a webpage and wants to check how well his lemonade is doing by asking for feedback ratings of 1 to 5 starts for his lemonade. As each rating is received, a server-side script is used to recalculate the average rating for the lemonade and upload the average rating to the website. The webserver stores the total number of ratings received along with the overall average rating.

Using Python, write an algorithm for this script.

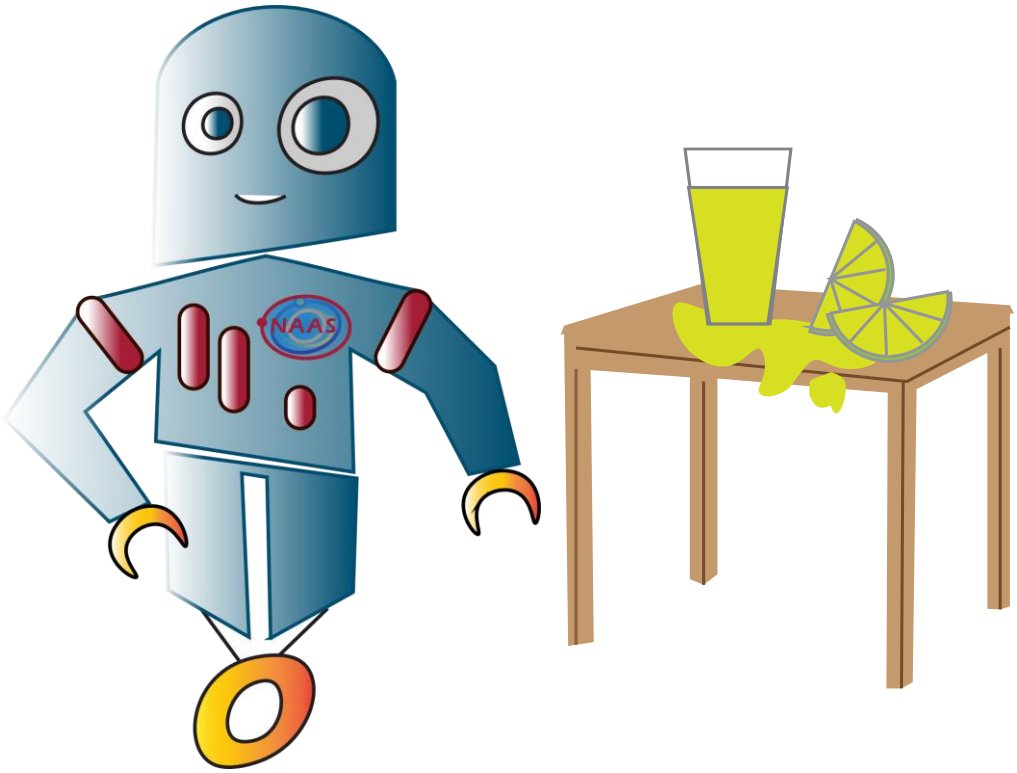
Python Copy code

```python
# Import necessary libraries
import pandas as pd


# Load the data from the CSV file
df = pd.read_csv('Lemonade_Ratings.csv')


# Assume the first column is 'Rating'
df.columns = ['Rating']


# Calculate the average rating
average_rating = df['Rating'].mean()


# Print the average rating
print(f"The average rating for all juices is {average_rating:.2f}")
```

Activity 2 – A step forward

NAAS lemonade website is going really well and he wants to expand his juice business. He expended his selection and now his webpage sells lemonade and other four flavours of juice. Now the choice of juices are: lemon, orange, pineapple, strawberry and apple.

He wants to check how well he is doing and which juice is the most popular by asking for feedback ratings of 1 to 5 starts for his juices. As each rating is received, a server-side script is used to recalculate the average rating for the juices and upload the average rating to the website. The webserver stores the total number of rating received for each item, along with the overall average rating for each juice. Using Python, write an algorithm for this script.



Activity 2 – Answer (for teachers)

To implement the algorithm for tracking and updating the average ratings for NAAS's juices, you can use a similar approach as before. Here's a basic example:

Python Copy code

```python
import pandas as pd


# Load the CSV file into a DataFrame

df = pd.read_csv('Juice_Ratings.csv')


# Calculate the weighted average of star ratings for each juice

df['Average Rating'] = (df['1 star'] * 1 + df['2 stars'] * 2 + df['3 stars'] * 3 + df['4 stars'] * 4 + df['5 stars'] * 5) / (df[['1 star', '2 stars', '3 stars', '4 stars', '5 stars']].sum(axis=1))


# Round the average ratings to one decimal place

df['Average Rating'] = df['Average Rating'].round(1)


# Print the average ratings for each juice

print(df[['Juice', 'Average Rating']])


# Calculate and print the overall average rating across all juices

overall_avg_rating = (df[['1 star', '2 stars', '3 stars', '4 stars', '5 stars']].values * [1, 2, 3, 4, 5]).sum() / df[['1 star', '2 stars', '3 stars', '4 stars','5 stars']].values.sum()

overall_avg_rating = round(overall_avg_rating, 1)

print(f'Overall Average Rating: {overall_avg_rating}')
```

Activity 3(a)

Are you up for a challenge?

Consider you work for a touristic agency that offers trips to several European destinations. The website manager asks for feedback from 30 customers who took part in each of the trips in the past year. It asked three questions to the customers with a simple answer of ‘yes’ or ‘no’: “Did the trip meet your expectations?”, Would you go on the same trip again?”, “Would you recommend the trip to a friend?”. As each answer is received, a server-side script is used to recalculate the average number of ‘yes’ for the trip and upload the average number to the website. The webserver stores the total number of positive and negative answers received for each destination and also gives the total standard deviation of ‘yes’. Using Python, write an algorithm for this script.

Activity 3(b)

Initial test of a function showed the existence of a logic error. This was corrected. 1) Use an example to explain how a logic error could occur by changing one line of code. 2) Describe other types of error that may occur when coding.

Activity 3(a) – Answer (for teachers)

Python Copy code

```python
import pandas as pd

import numpy as np


# Load data from CSV file

df = pd.read_csv('Touristic agency.csv')


# Calculate the average number of 'yes' for each destination

df['Average Yes'] = df[['Meet expectations', 'Would go again', 'Would
recommend']].mean(axis=1).round(1)


# Calculate total standard deviation of ‘yes’

total_std_dev = np.std(df[['Meet expectations', 'Would go again', 'Would
recommend']].values).round(1)


print("Average Yes per Destination:")

print(df[['Destination', 'Average Yes']])


print("\nTotal Standard Deviation of ‘Yes’:", total_std_dev)
```

The script uses the Pandas library to read the CSV file and calculate the average number of ‘yes’ for each destination. It then calculates the total standard deviation of ‘yes’ across all destinations. The results show that Italy, Greece, and Ireland received the highest average number of ‘yes’, while Germany, Malta, and Belgium received the lowest. The total standard deviation of ‘yes’ is 5.2, indicating that there is some variability in the responses across all destinations.

Activity 3(b) – Answer (for teachers)

A logic error occurs when the code runs without any syntax errors, but the output is not what was expected. For example, consider the following code that is supposed to calculate the average of two numbers:

```python
a = 5
b = 10
average = a + b / 2
print("The average is:", average)
```

The output of this code will be The average is: 10.0, which is not the expected result. This is because the order of operations is incorrect: the division should be done before the addition. To fix this, we can add parentheses around the addition operation:

```python
a = 5
b = 10
average = (a + b) / 2
print("The average is:", average)
```

The output of this corrected code will be The average is: 7.5, which is the expected result.

2. Other types of errors that may occur when coding include:

a) **Syntax Errors:**

- Occur when the code violates the language's syntax rules.
- Usually easy to identify as they result in immediate error messages.
- Examples include missing colons, unmatched parentheses, or misspelled keywords.

b) **Runtime Errors:**

- Occur during the execution of the program.
- Examples include division by zero, accessing an index that is out of range, or trying to use a variable that has not been defined.

c) **Semantic Errors:**

- Occur when the code runs without any syntax or runtime errors, but the logic is incorrect.
- These errors are often subtle and can lead to unexpected behavior.
- Examples include using the wrong formula in a calculation or misunderstanding the requirements.

d) **Concurrency Errors:**

- Occur in multithreaded or multiprocessing programs when multiple threads or processes access shared data concurrently.
- Examples include race conditions, deadlocks, and data inconsistency due to lack of synchronization.

e) **Input/Output Errors:**

- Occur when there are issues with reading from or writing to files, databases, or external systems.
- Examples include file not found errors, permission issues, or incorrect data format.

It's important to thoroughly test code, handle exceptions appropriately, and use debugging tools to identify and fix errors in different stages of development.

Cross-curricular opportunities

1. **Mathematics: Data Analysis and Graphing**

   Use the average ratings calculated in the juice or lemonade examples to create bar graphs or pie charts. This integrates mathematics and statistics with the data collected from the ratings.

2. **English Language Arts: Writing Reviews or Descriptions**

   Ask students to write reviews for their favourite juice or lemonade, incorporating descriptive language and persuasive writing techniques. This activity enhances language arts skills while expressing opinions on the products.

3. **Science: Nutrition and Health**

   Explore the nutritional content of the different juices. Compare sugar levels, vitamin content, or other nutritional aspects. Discuss the impact of different juices on health. This activity combines science with the real-world context of juice products.

4. **Social Studies: Market Research and Geography**

   Research and analyse the popularity of juice flavours in different regions or countries. Discuss how cultural preferences may influence the popularity of certain juices. This activity ties in social studies with market research and geography.

5. **Art: Design a Juice Advertisement**

   Have students create visually appealing advertisements for the juices or lemonade. This activity combines art and marketing skills, allowing students to express creativity while promoting the products.

6. **Technology: Website Development**

   Extend the lemonade or juice selling scenario by having students create a simple website for NAAS's online store. This involves elements of technology and web development.

7. **Physical Education: Healthy Living Campaign**

   Connect the juice or lemonade theme to a healthy living campaign. Discuss the importance of hydration and encourage physical activity. This interdisciplinary approach integrates physical education with health and wellness.

8. **Music: Jingle or Commercial Song Composition**

   Challenge students to create a jingle or song for promoting the juice or lemonade. This activity combines musical creativity with marketing skills.

These cross-curricular activities provide students with a holistic learning experience, allowing them to apply skills from various subject areas to real-world scenarios. Teachers can adapt and modify these activities based on the grade level and specific learning objectives.

Copyrights