Software Engineering Processes

Project Report - August 2022

Standard Deviation (σ)

Kary Sureshbhai Sutariya

Student Id: 40193909

supervised by

Dr. Pankaj Kamthan

1 Problem 1

1.1 Description

Standard deviation is the square root of variance and variance provides variability from the mean. Standard deviation is denoted by a Greek symbol σ. The equation for Standard deviation is given below.[1]

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

where:
x_i: Each value in the data set
$\overline{x}$: Mean of all values in the data set
N: Number of values in the data set

Figure 1 shows normal curve standard deviation. In this case, values are distributed equally over the range. This type of distribution is ideal for many real-time projects. For instance, data scientists always try to make their data set equally distributed and to check it, standard deviation has been utilized over the time again and again.

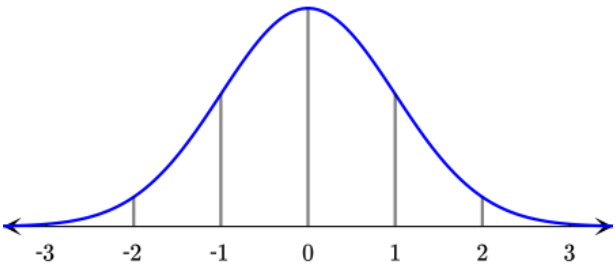

Figure 1: Graph of Standard deviation [2]

1.2 Domain and Co-domain

For Standard deviation, all real numbers are in its domain and co-domain. Real number ($\mathbb{R}$) includes rational numbers ($\mathbb{Q}$), which include the integers ($\mathbb{Z}$), which in turn include the natural numbers ($\mathbb{N}$) but not imaginary numbers(i).[3] In the mathematical term, It can be written like this.

$$\sigma : \mathbb{R} \to \mathbb{R}$$

1.3 Characteristic

- If the Standard deviation is high, numbers are scattered far from each other.
- Whereas if numbers are nearby to each other, the Standard deviation will be low.
- It is sensitive to extreme values. A small change in any extreme value can lead to drastic alteration.[4]

1.4 Use model

In this section, context of use model has been explained. The following Use model depicts where the standard deviation function can be used by users. Standard deviation is being used in various fields namely AI, Finance, Resource management, Environmental studies etc.

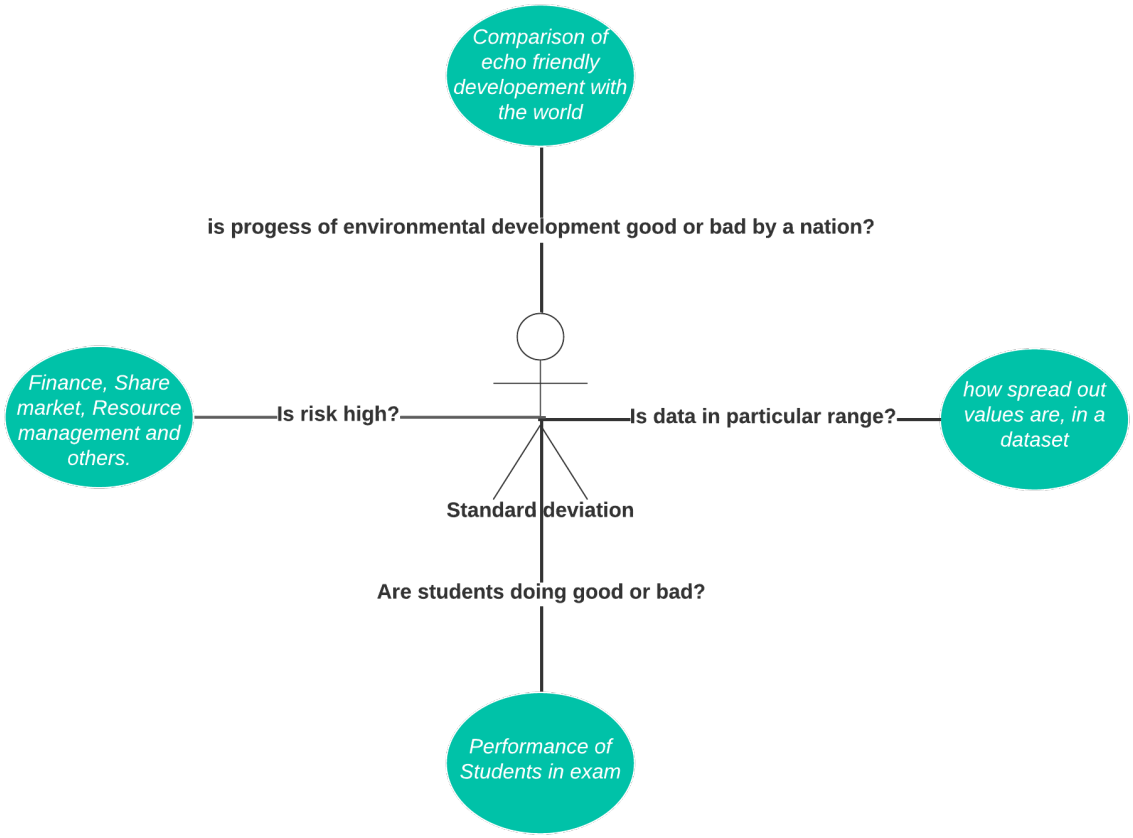

Figure 2: Use model

2 Problem 2

2.1 First Requirement

- **ID:** FR 1
- **Type:** Functional Requirement
- **Version:** 1.0
- **Priority:** Highest
- **Description:** Since standard deviation tells how close the numbers are, all the inputs must be numbers and not a character or set of characters.

2.2 Second Requirement

- **ID:** FR 2
- **Type:** Functional Requirement
- **Version:** 1.0
- **Priority:** Low
- **Description:** To calculate the mean, all real numbers must be considered.

2.3 Third Requirement

- **ID:** FR 3
- **Type:** Functional Requirement
- **Version:** 1.0
- **Priority:** High
- **Description:** Standard deviation must not have to be negative as it is the square root of variance.

2.4 Fourth Requirement

- **ID:** FR 4
- **Type:** Functional Requirement
- **Version:** 1.0
- **Priority:** Medium
- **Description:** If variance exists, Standard deviance exists and vice versa.

2.5 Fifth Requirement

- **ID:** FR 5
- **Type:** Non-Functional Requirement (System)
- **Version:** 1.0
- **Priority:** Medium
- **Description:** Higher availability of memory will result in swift execution when total input numbers are high.

2.6 Sixth Requirement

- **ID:** FR 6
- **Type:** Non-Functional Requirement (Portability)
- **Version:** 1.0
- **Priority:** High
- **Description:** Java virtual machine (JVM) is a virtual machine that enables a computer to run the program.

3 Problem 3

3.1 Pseudo Code for Iterative approach

Algorithm 1 Standard Deviation with iterative

```
function MAIN
    N ← n
    x[N] ←= 1, 5, ..., n^th number
    x̄ = CAL_MEAN(x[N], N)
    var = VARIANCE(x[N], N, x̄)
    σ = SQRT(var)
end function
function CAL_MEAN(x[N], N)
    avg ← 0
    for i ← 1 to N do
        avg ← avg + x[i]
    end for
    return avg/N
end function
function SQUARE(number)
    return number * number
end function
function SQRT(number)
    if number! = 0 then
        T ← 0
        sqrtRoot ← T/2
        do
            T ← sqrtRoot
            sqrtRoot ← (T + (number/T))/2
        while (T − sqrtRoot)! = 0
        return sqrtRoot
    else
        return 0
    end if
end function
function VARIANCE(x[N], N, avg)
    temp ← 0
    for i ← 1 to N do
        temp ← temp + SQUARE(x[i] − avg)
    end for
    return temp / N
end function
```

There are a total of 4 subordinate functions and one main function. In the above algorithm, CAL_MEAN$(x[N], N)$ and VARIANCE$(x[N], N, \overline{x})$ functions have been defined using an iterative approach.

3.2 Pseudo Code for Recursive approach

Algorithm 2 Standard Deviation with recursion

```
function MAIN
    N ← n
    k ← 0
    x[N] ←= 1, 5, ..., n^th number
    x̄ = CAL_MEAN(x[N], k, N)
    var = VARIANCE(x[N], N, k, x̄)
    σ = SQRT(var)
end function
function CAL_MEAN(x[N], k, N)
    if k == N then
        avg ← avg/N
        return avg
    else
        avg ← avg + x[k]
        k++
        return CAL_MEAN(x[N], k, N)
    end if
end function
function SQUARE(number)
    return number * number
end function
function SQRT(number)
    if number! = 0 then
        T ← 0
        sqrtRoot ← T/2
        do
            T ← sqrtRoot
            sqrtRoot ← (T + (number/T))/2
        while (T − sqrtRoot)! = 0
        return sqrtRoot
    else
        return 0
    end if
end function
function VARIANCE(x[N], N, k, avg, var = 0)
    if k == N then
        var ← var/N
        return var
    else
        var ← SQUARE(x[k] − avg)
        k++
        return VARIANCE(x[N], N, k, avg, var)
    end if
end function
```

For Recursive algorithm, CAL_MEAN$(x[N], N)$ and VARIANCE$(x[N], N, \overline{x})$ functions have been made recursive approach whereas SQUARE$(number)$ and

SQRT(*number*), both the functions are similar in both the algorithms.

3.3 Mind map

The below mind map helps to understand how all the subordinate functions will be executed by algorithms. Mind map for both is same. The reason for that is same structure has been used. The inner approach and the number of arguments are different for subordinate functions. Execution of algorithms will start from the main function. Next, CAL_MEAN() will be called to calculate mean and this mean will be passed to VARIANCE(). Moreover, VARIANCE() will use SQUARE() function to calculate the square of numbers. In the last, SQRT() will be called to get square root of the variance and it will give the standard deviation.

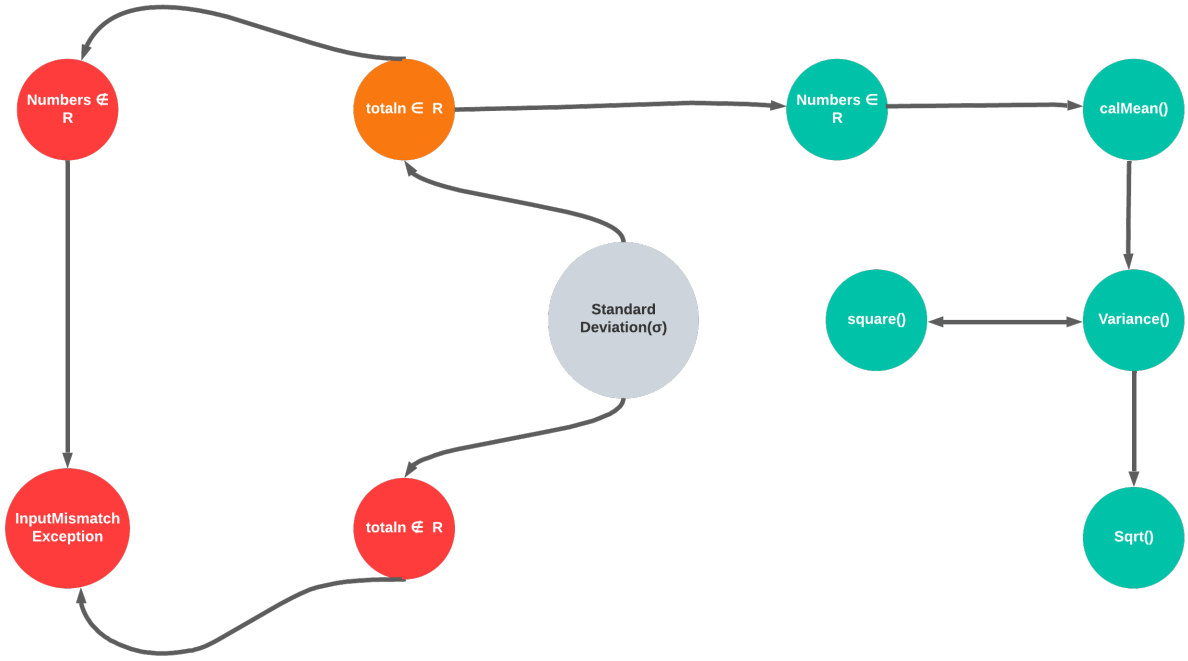

Figure 3: mindmap

3.4 Iterative Algorithm

Reason to opt Iterative approach: Iterative approach is less dependent on memory and more on computation power of the system. To get results with less space availability, this approach has been implemented in this project.

Pros:

- It doesn't generate overhead.
- Never will face the problem of Stack overflow.
- Easy to explain to even children want to learn to code.

Cons:

- For large input, calculation time will be higher.
- Doesn't utilization memory up to the mark which will lead to slow execution.[5]

3.5 Recursive Algorithm

Reason to opt Recursive approach: Recursion is totally dependent on memory and will can give faster execution. Also, to analyze how use of more memory can play a vital role in the execution of code.

Pros:

- Fully utilize memory up to the mark for rapid execution.
- It is suitable for the small size of the input.

Cons:

- For large input, it is slow and maybe, it can produce stack over error.
- Complex to understand.
- generates overhead during execution. Doesn't utilization memory up to the mark which will lead to slow execution. [6]

4 Problem 4

The standard deviation function has been implemented in pure Java applications. Moreover, it is compatible with Eclipse and IntelliJ. However, High-end user can run it from the command line. More details are available in 'Readme.md' file.

4.1 Error Handling

Human is to err. As said that, there is a chance for wrong input from a user. To handle that, the Input Mismatch Exception has been used in the code for total numbers and numbers.

4.2 Error Messaging

Error messages are paramount. Many times developers use Systemoutprintln in the production phase. This is a wrong practice performed by the coder but here, logger function of java.util library has been used to pass error messages. As well as the program will log appropriate messages in such conditions.

4.3 Debugger

A debugger is a special tool often used programmers to find out bugs.[7] All high level language code can be easily debugged. Find out bugs can be a tedious task but it also depended on the skills of the programmer. In this project, the debugger has been used as shown in Figure 4.

Pros:

- For any programmer, it is not possible to able to think about all corner cases and debugger can help in these situations.
- Able to modify values at run time.
- User can put any number of breakpoints to evaluate certain sections of program.
- To see the flow of control.

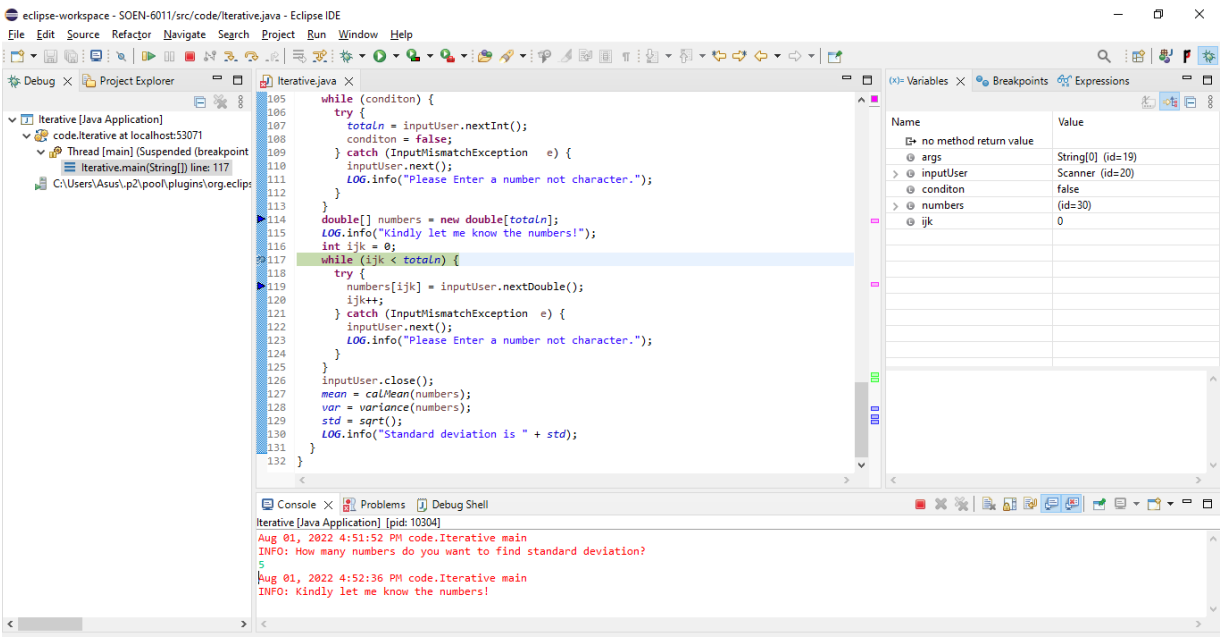

Figure 4: Screenshot of debugger

Cons:

- With different IDES, debugging process is different.
- Doesn't provide a 'Rollback' feature.

4.4 Pragmatic Quality Checking Tool

Quality checking helps to increase the quality of code so in future if other programmer tries to upgrade or understand code, he/she doesn't face any problems. There are many Pragmatic Quality Checking Tools. The one which one used in this project is the below. This one is available on Eclipse Market place.

Name of Tool: eclipse-cs[8]

Standards: Sun Code Conventions and Google Java Style.

Version: 10.3.2

Published By: Lars ödderitzsch

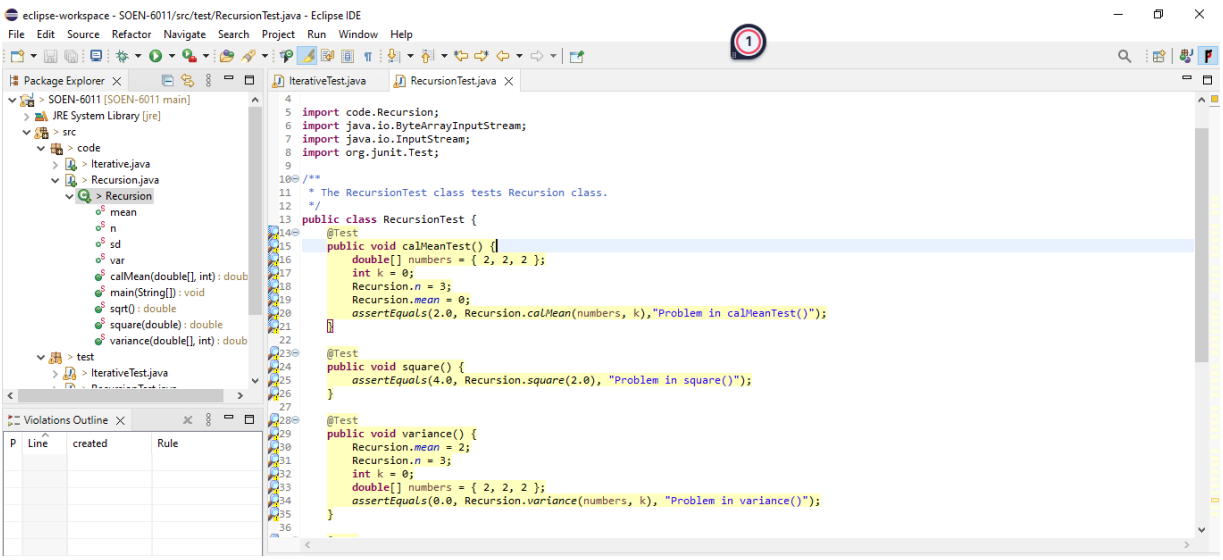

Figure 5: Screen shot of check style

Pros:

- Can easily integrate with different IDES.
- User can add custom rules.
- Bring unification to a team project.

Cons:

- Frequent recompile is required.
- Because of cache, sometimes even if the problem has been solved but it shows there is one.
- Simple rules can lead to large changes.
- Doesn't improve the quality of code like PMD tool.

5 Problem 5

All unit test cases for Standard Deviation (σ) ensure that functions work as expected when given various kinds of inputs. For this tedious task, Junit framework has utilized. It helps to figure out the actual behaviour of code in a kind of simulated situation.

5.1 Standards for unit tests

In the project, to check the standard of unit tests Check-Style plug-in has been used. Initially, these unit tests have problems like multiple assert statements in one unit test which had been separated into different unit cases like sqrtTest1() and sqrtTest2(). Adding to that, all the test cases are traceable to requirements.

5.2 Unit case traceability with requirements

Each and every test case is intertwined with one of the requirements. There is one to one relationship between test cases and requirements. More details can be captured from the following tables.

Unit Test 1	
Test Case ID	TR 1
Requirement ID	FR 1
Name of test case	mainTest()
Input(s)	c 2 2 g 2
Expected Output	0
Actual Output	0
Test Result	Success

Table 1: Unit test for requirement 1

Unit Test 2	
Test Case ID	TR 2
Requirement ID	FR 2
Name of test case	calMeanTest()
Input(s)	2 2 2
Expected Output	0
Actual Output	0
Test Result	Success

Table 2: Unit test for requirement 2

Unit Test 3.1	
Test Case ID	TR 3.1
Requirement ID	FR 3
Name of test case	sqrtTest1()
Input(s)	0.0
Expected Output	0
Actual Output	0
Test Result	Success

Table 3: Unit test for requirement 3

Unit Test 3.2	
Test Case ID	TR 3.2
Requirement ID	FR 3
Name of test case	sqrtTest2()
Input(s)	4.0
Expected Output	2.0
Actual Output	2.0
Test Result	Success

Table 4: Unit test for requirement 3

Unit Test 4	
Test Case ID	TR 4
Requirement ID	FR 4
Name of test case	variance()
Input(s)	2 2 2
Expected Output	0.0
Actual Output	0.0
Test Result	Success

Table 5: Unit test for requirement 4

Moving ahead, the requirement 5 and so forth are non-functional requirements so there is no way to write test cases for these non-functional requirements.

6 Bibliography

1. https://en.wikipedia.org/wiki/Standard_deviation
2. https://mathleaks.com/mediawiki/images/5/5a/Mljsx Concept standard distribution 1 old.svg
3. https://en.wikipedia.org/wiki/Domain_of_a_function
4. https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch12/5214891-eng.htm
5. https://www.dcsl.com/pros-cons-iterative-software-development/
6. https://medium.com/@williambdale/recursion-the-pros-and-cons
7. https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html
8. https://checkstyle.org/eclipse-cs/#!/