How to Calculate Standard Deviation in PySpark Easily

Authored by

stats writer

February 7, 2026

ARABPSYCHOLOGY.COM

RECOMMENDED CITATION

stats writer (2026). *How to Calculate Standard Deviation in PySpark Easily*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=129689

Calculating the standard deviation (SD) is a fundamental task in statistical analysis, providing a crucial measure of data dispersion or variability within a dataset. When dealing with big data environments, tools like Apache Spark become essential for handling large volumes of data efficiently. PySpark, the Python API for Spark, provides robust functionalities, specifically within the **pyspark.sql.functions** module, to calculate various aggregate statistics, including the SD, directly on a distributed DataFrame. Utilizing these native functions ensures that calculations are optimized for performance across the cluster.

While earlier versions or alternative approaches might utilize statistical summary functions available via the **stat** module--which offers comprehensive metrics like count, mean, and quartiles--the most common and straightforward method for computing the SD relies on the dedicated functions provided in **pyspark.sql.functions**, namely **stddev** and **stddev_pop**. These functions are designed to streamline the process of obtaining descriptive statistics, making it simple to assess the spread of numerical columns within your big data pipeline, regardless of the volume or complexity of the underlying data.

This guide provides a detailed, step-by-step approach to calculating the standard deviation using these powerful PySpark methods. We will explore two primary scenarios: calculating the SD for a single column and calculating the SD concurrently across multiple columns. Furthermore, we will clarify the critical distinction between sample standard deviation (the default behavior of **stddev**) and the population standard deviation (calculated using **stddev_pop**), ensuring analysts can select the statistically appropriate measure for their specific analytical requirements.

ARABPSYCHOLOGY.COM

Calculating Statistical Dispersion: Standard Deviation in PySpark

Understanding PySpark's Statistical Functions

The following methods are available for calculating the standard deviation of a column or multiple columns in a PySpark DataFrame. Understanding these core functions, specifically **stddev** and **stddev_pop**, is essential for accurate statistical reporting in a distributed computing environment.

We will demonstrate two common implementation patterns: the highly specific calculation for a single column, which typically uses the **agg()** method for retrieval, and the bulk calculation across multiple columns, which leverages the **select()** method for streamlined output. Both methods rely on importing the requisite functions from **pyspark.sql.functions**.

Method 1: Calculating Standard Deviation for One Specific Column

The most direct route for obtaining the sample standard deviation for a single column is through

DataFrame aggregation. By importing the functions module as the alias **F**, we can succinctly apply the **F.stddev()** function to the target column. This approach is preferred when the objective is to extract a single numerical result for integration into further sequential Python code or simple logging.

from pyspark.sql import functions as F

#calculate standard deviation of values in 'game1' column
df.agg(F.stddev('game1')).collect()

This command returns the standard deviation as a floating-point number, bypassing the need to display an intermediate one-row DataFrame, making it ideal for programmatic use within larger scripting pipelines.

Method 2: Calculating Standard Deviation for Multiple Columns

To simultaneously assess the dispersion across several numerical fields, the **select()** transformation provides a clean and efficient mechanism. By passing the **stddev()** function applied individually to each desired column ('game1', 'game2', 'game3', etc.) within the **select()** call, a new DataFrame is generated.

from pyspark.sql.functions import stddev

#calculate standard deviation for game1, game2 and game3 columns
df.select(stddev(df.game1), stddev(df.game2), stddev(df.game3)).show()

This method is highly scalable and ensures that the calculations for all specified columns are performed efficiently across the Spark cluster, yielding a summary table that facilitates immediate comparison of variability.

Sample vs. Population Standard Deviation Selection

It is critical to remember that the standard **stddev** function computes the sample standard deviation (using N-1 correction). If your dataset represents the entire population under study, you must explicitly use the function designed for that statistical context.

The specialized function **stddev_pop** must be used if you intend to calculate the true population standard deviation (using N in the denominator). Using the wrong function can lead to inaccurate conclusions regarding the true spread of the population data.

Note: The **stddev** function uses the sample standard deviation formula to calculate the standard

ARABPSYCHOLOGY.COM

deviation. If you would instead like to use the population standard deviation formula, then use the **stddev_pop** function instead.

Setting Up the Sample PySpark DataFrame

The following examples utilize a small, illustrative DataFrame to clearly demonstrate the syntax and resulting output for each calculation method. This setup phase initializes the Spark environment and constructs the data object on which all subsequent statistical queries will run.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

#define data
data = ,
,
,
,
]

#define column names
columns =

#create dataframe using data and column names
df = spark.createDataFrame(data, columns)

#view dataframe
df.show()
```

```
+-------+-----+-----+-----+
| team|game1|game2|game3|
+-------+-----+-----+-----+
| Mavs| 25| 11| 10|
| Nets| 22| 8| 14|
| Hawks| 14| 22| 10|
| Kings| 30| 22| 35|
| Bulls| 15| 14| 12|
|Blazers| 10| 14| 18|
+-------+-----+-----+-----+
```

This resulting DataFrame, **df**, contains six records and four columns, providing the necessary

ARABPSYCHOLOGY.COM

numerical data points ('game1', 'game2', 'game3') for our standard deviation calculations.

Detailed Example 1: Calculate Standard Deviation for One Specific Column

To apply Method 1 in practice, we calculate the dispersion for the 'game1' column, isolating its numerical values and aggregating them using **F.stddev**. This single operation yields the sample standard deviation, providing a measure of how tightly clustered the 'game1' scores are around their average.

```
from pyspark.sql import functions as F

#calculate standard deviation of column named 'game1'
df.agg(F.stddev('game1')).collect()
```

```
7.5806771905065755
```

The standard deviation of values recorded in the **game1** column is determined to be approximately **7.5807**. This quantitative value reflects the variation in scores across the teams for this specific game metric.

Detailed Example 2: Calculate Standard Deviation for Multiple Columns

Leveraging Method 2, we execute a simultaneous calculation of the sample standard deviation for 'game1', 'game2', and 'game3'. The use of **df.select()** produces an immediate, comparative output, revealing the relative variability across all three game metrics in a single, aggregated row.

```
from pyspark.sql.functions import stddev

#calculate standard deviation for game1, game2 and game3 columns
df.select(stddev(df.game1), stddev(df.game2), stddev(df.game3)).show()
```

```
+-----------------+-----------------+-----------------+
|stddev_samp(game1)|stddev_samp(game2)|stddev_samp(game3)|
+-----------------+-----------------+-----------------+
|7.5806771905065755| 5.741660619251774| 9.544631999192006|
+-----------------+-----------------+-----------------+
```

From the resulting output DataFrame, we can clearly observe and contrast the calculated standard deviations for all three game columns, noting that 'game3' exhibits the largest dispersion. This comparative insight is valuable for understanding where data points show the greatest fluctuation.

ARABPSYCHOLOGY.COM

The standard deviation of values in the **game1** column is **7.5807**.

The standard deviation of values in the **game2** column is **5.7417**.

The standard deviation of values in the **game3** column is **9.5446**.

Handling Null Values in Calculations

In practical data analysis, numerical columns often contain missing or null values. It is important to know how the chosen statistical function interacts with this missing data.

The **stddev** function in PySpark is designed to be fault-tolerant regarding missing data. By default, it automatically ignores any null entries encountered in the column during the calculation of the aggregate statistic. This ensures that the result is based purely on the valid, non-null observations present.

Note: If there are null values in the column, the **stddev** function will ignore these values by default.

Further PySpark Capabilities

Mastering the calculation of standard deviation is a fundamental step in distributed data analysis. PySpark offers an extensive suite of functions for statistical analysis, allowing users to perform complex transformations and metrics calculation efficiently on massive datasets.

The following tutorials explain how to perform other common tasks in PySpark:

ARABPSYCHOLOGY.COM