IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4512525/
Vol. 2025 (20), Article ID 4512525

Statistics in SAP ABAP*

Marek GAŁĄZKA[1] and Hanna WDOWICKA[2]

[1]Adam Mickiewicz University, Poznań, Poland

[2]University of Economics and Business, Poznań, Poland

Correspondence should be addressed to: Marek GAŁĄZKA, galazka@amu.edu.pl

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

Statistical methods are used in research encompass planning, designing, data collection, analysis, interpretation, and reporting of findings. Through statistical analysis, raw numbers are transformed into meaningful insights, bringing data to life. The accuracy of results and conclusions depends on the appropriate application of statistical tests.

This article presents several own implementations of advanced statistical methods that are not available in the ABAP standard.

Keywords: Statistical, Statistical Technique, SAP S/4HANA, ABAP development

Introduction

Statistics is a scientific discipline focused on the collection, organization, and analysis of data, as well as drawing inferences from samples to entire populations (Amedee R., *et al.*, 2010). This process involves designing studies effectively, selecting appropriate samples, and choosing suitable statistical tests. A solid understanding of statistics is essential for example for designing epidemiological studies and clinical trials correctly. The use of improper statistical methods can lead to inaccurate conclusions, potentially resulting in incorrect practices (Sprent, P., 2003).

Modern statistics has evolved into a highly computational science (Deborah, N., and Duncan, T., 2015, Cobb, G., 2015). Many fundamental concepts in contemporary statistics are best understood through computational techniques for data analysis.

The implementation of statistical methods in various programming languages plays a crucial role in data analysis, machine learning, and scientific research. With the increasing demand for efficient and scalable data processing, programming languages offer diverse tools and libraries to facilitate statistical computations. Languages such as R, Python, MATLAB (Shaw, S., *et al.*, 2022, Brito, N., *et al.*, 2022, Sengur, D., 2023, Ren, Z., 2020) provide built-in functions and frameworks that enable researchers and developers to apply statistical techniques with ease.

In this article it was noted that in the ABAP language (Bandari, K., 2021) there is no library implementing well-known and advanced statistical methods (Longnecker, M., 2021). Therefore, the authors of this article created their own implementation of several well-known statistical methods.

Cite this Article as: Marek GAŁĄZKA and Hanna WDOWICKA, Vol. 2025 (20) "Statistics in SAP ABAP" Communications of International Proceedings, Vol. 2025 (20), Article ID 4512525,
https://doi.org/10.5171/2025.4512525

The paper is organized as follows: means are described in Section 1. In Section 2 are implemented moments. In section 3 are presented distributions. We conclude the paper in Section 4.

Means

This section describes the implementation of 3 means: Harmonic Mean, Geometric Mean, Quadratic Mean.

Harmonic Mean

For a set of *n* positive numbers $x_1,x_2,\ldots,x_n$, the harmonic mean (HM) is given by:

$$HM = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

The harmonic mean is particularly useful in situations involving rates, ratios, and inverse relationships.

To calculate some statistics we can use the standard SFLIGHT table in the SAP system.

SELECT * FROM sflight INTO TABLE @DATA(lt_sflight).

DATA(lr_stats) = NEW zcl_statistical_methods(lt_sflight).

Use →col( ) method to select a column on which make calculations.

DATA(lr_prices) = lr_stats->col('Price').

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD harmonic_mean.

* Get values from the corresponding column in the table (in our case, the SFLIGHT table and the PRICE
*column)
    DATA(lt_values)  = get_values( iv_col ).
* Get the number of elements (number of lines in the table)
    DATA(lv_lines)   = CONV f( count( ) ).
* Check if any values were taken
    CHECK lt_values IS NOT INITIAL.
* Check if any value is equal to 0. Then the harmonic mean cannot be calculated
    IF line_exists( lt_values[ value = '0.0' ] ).
      r = '0'.
      RETURN.
    ENDIF.
* Apply the formula for the harmonic mean
    r = lv_lines / REDUCE #( INIT sr = CONV f( 0 ) FOR val IN lt_values NEXT sr = sr + ( '1.0' / val-value ) ).

  ENDMETHOD.
```

The method call looks like this:

DATA(lv_harmonic_mean) = lr_prices->harmonic_mean( ).

Geometric Mean

The geometric mean is a type of average that is useful for sets of positive numbers, especially when dealing with proportions, percentages, or exponential growth. It is calculated by multiplying all the numbers in the set and then taking the nth root, where n is the total number of values.

$$\text{Geometric Mean} = \sqrt[n]{x_1 \times x_2 \times \cdots \times x_n}$$

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD geometric_mean.

* Get values from the corresponding column in the table (in our case, the SFLIGHT table and the PRICE
*column)
    DATA(lt_values)  = get_values( iv_col ).
* Get the number of elements (number of lines in the table)
    DATA(lv_lines)   = CONV f( count( ) ).
* Check if any values were taken
    CHECK lt_values IS NOT INITIAL.
* Apply the formula for the geometric mean
    r = REDUCE #( INIT gm = CONV f( 1 ) FOR val IN lt_values NEXT gm = gm * ( val-value ** ( '1.0' /
lv_lines ) ) ).

  ENDMETHOD.
```

The method call looks like this:

```
DATA(lv_geometric_mean) = lr_prices-> geometric_mean( ).
```

Quadratic Mean

The quadratic mean, also known as the root mean square (RMS), is a measure of the average magnitude of a set of numbers, giving more weight to larger values. It is particularly useful when dealing with quantities that can be both positive and negative, such as electrical signals, velocities, or deviations in statistics.

$$\text{Quadratic Mean (RMS)} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}}$$

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD quadratic_mean.

* Get values from the corresponding column in the table (in our case, the SFLIGHT table and the PRICE
*column)
    DATA(lt_values)  = get_values( iv_col ).
* Get the number of elements (number of lines in the table)
    DATA(lv_total)   = CONV f( count( ) ).
* Check if any values were taken
    CHECK lt_values IS NOT INITIAL.
* Apply the formula for the quadratic mean
    r = sqrt( REDUCE #( INIT sr = CONV f( 0 ) FOR val IN lt_values NEXT sr = sr + ( val-value ) ** 2 ) /
lv_total ).

  ENDMETHOD.
```

The method call looks like this:

```
DATA(lv_ quadratic _mean) = lr_prices-> quadratic_mean( ).
```

Moments

This section describes the implementation of 2 moments: Skewness, Kurtosis.

Skewness

Skewness is a statistical measure that describes the asymmetry of a probability distribution relative to its mean. It quantifies whether the distribution of a random variable leans more to one side of the mean. To estimate skewness, we typically use the sample skewness, which is calculated using the adjusted Fisher-Pearson standardized moment coefficient, a method also employed by Excel.

$$\text{skewness} = \frac{n}{(n-1)(n-2)} \frac{\sum_{i=1}^{n} (x_i - \bar{x})^3}{\left[\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2\right]^{3/2}}$$

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD skewness.

* Get values from the corresponding column in the table (in our case, the SFLIGHT table and the PRICE
*column)
    DATA(lt_values)      = _get_num_values( iv_col ).
* Calculate standard deviation
    DATA(lv_std_dev)     = CONV f( standard_deviation( iv_col ) ).
* Calculate mean value
    DATA(lv_mean_value)  = CONV f( mean( iv_col ) ).
* Get the number of elements (number of lines in the table)
    DATA(lv_total)       = CONV f( count( ) ).
* Apply the formula for the skewness
    r = lv_total * REDUCE #( INIT v = CONV f( 0 ) FOR val IN lt_values NEXT v = v + ( ( val-value -
lv_mean_value ) / lv_std_dev ) ** 3 ) / ( ( lv_total - '1.0' ) * ( lv_total - '2.0' ) ).

  ENDMETHOD.
```

To calculate skewness we need to calculate the standard deviation and mean value. In turn, to calculate the standard deviation we need the variance, because the standard deviation is the square root of the variance.

```
METHOD variance.

* Get values from the corresponding column in the table (in our case, the SFLIGHT table and the PRICE
*column)
    DATA(lt_values)      = get_values( iv_col ).
* Calculate mean value
    DATA(lv_mean_value)  = CONV f( mean( iv_col ) ).
* Get the number of elements (number of lines in the table)
    DATA(lv_total)       = CONV f( count( ) ).
* Apply the formula for the variance
    r = REDUCE #( INIT v = CONV f( 0 ) FOR val IN lt_values NEXT v = v + ( val-value - lv_mean_value ) **
2 ) / ( lv_total - '1.0' ).

  ENDMETHOD.

  METHOD standard_deviation.
```

```
* Apply the formula for the standard deviation
    r = sqrt( CONV f( variance( iv_col ) ) ).

  ENDMETHOD.
  METHOD mean.

* Apply the formula for the mean
    r = CONV f( sum( iv_col ) ) / CONV f( count( ) ).

  ENDMETHOD.
```

The method call looks like this:

```
DATA(lv_ skewness) = lr_prices-> skewness ( ).
```

Kurtosis

Kurtosis is a statistical measure that describes the shape of a probability distribution's tails in relation to a normal distribution. It tells us how much of the variance in the data is due to extreme values (outliers).

$$\text{kurtosis} = \frac{1}{(n-1)} \frac{\sum_{i=1}^{n} (x_i - \bar{x})^4}{\left[\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2\right]^2}$$

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD kurtosis.

* Get values from the corresponding column in the table (in our case, the SFLIGHT table and the PRICE
*column)
    DATA(lt_values)      = get_values( iv_col ).
* Calculate standard deviation
    DATA(lv_std_dev)     = CONV f( standard_deviation( iv_col ) ).
* Calculate mean value
    DATA(lv_mean_value)  = CONV f( mean( iv_col ) ).
* Get the number of elements (number of lines in the table)
    DATA(lv_total)       = CONV f( count( ) ).
* Apply the formula for the kurtosis
    r = REDUCE #( INIT v = CONV f( 0 ) FOR val IN lt_values NEXT v = v + ( ( val-value - lv_mean_value ) /
lv_std_dev ) ** 4 ) / ( lv_total - '1.0' ).

  ENDMETHOD.
```

The method call looks like this:

```
DATA(lv_ kurtosis) = lr_prices-> kurtosis ( ).
```

Distributions

This section describes the implementation of 5 distributions: Continuous Uniform Distribution, Continuous Normal Distribution, Bernoulli Distribution, Binomial Distribution, Geometric Distribution.

Continuous Uniform Distribution

A continuous uniform distribution is a probability distribution where all outcomes within a given range are equally likely. It is defined by a constant probability density over an interval [a, b], meaning every value in that range has the same chance of occurring.

For a continuous uniform distribution between a and b, the probability density function (PDF) is:

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$$

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD uniform.

* Create a pseudorandom number generator for Floating Point Numbers
    DATA(lr_randomizer) = cl_abap_random_float=>create( cl_abap_random=>seed( ) ).
* Generate the appropriate number of pseudo-random numbers
    DO iv_size TIMES.
* Next Pseudo Random Integer in [0,1]
      DATA(lv_rand_float) = lr_randomizer->get_next( ).
* Add a pseudo random number to the r table (float table)
      APPEND ( iv_low + ( iv_high – iv_low ) * lv_rand_float ) TO r.

    ENDDO.

  ENDMETHOD.
```

If we want for example generate a sample of 10000 values from a uniform distribution in the interval [1, 50], the method call looks like this:

```
DATA(lt_uniform_values) = zcl_statistical_methods=>uniform( iv_low = 1 iv_high = 50 iv_size = 10000 ).
```

Continuous Normal Distribution

The normal distribution, also called the Gaussian distribution, is one of the most important probability distributions in statistics. It describes a continuous probability distribution where values are symmetrically distributed around the mean, forming the characteristic bell-shaped curve.

The normal distribution is defined by the probability density function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

To generate the normal distribution, the Box–Muller Transform Sampling Method is used.

The Box–Muller transform is a method for generating normally distributed random numbers from uniformly distributed random numbers. It is widely used in simulations, Monte Carlo methods, and statistical modeling.

Computers can easily generate uniform random numbers (between 0 and 1) but not normally distributed numbers. The Box–Muller transform converts two independent uniform random numbers into two independent standard normal (mean = 0, variance = 1) random numbers.

Given two independent random numbers U_1 and U_2 from a uniform distribution on (0,1] we compute:

$$Z_0 = \sqrt{-2 \ln U_1} \cos(2\pi U_2)$$

$$Z_1 = \sqrt{-2 \ln U_1} \sin(2\pi U_2)$$

where:

- Z_0 and Z_1 are independent standard normal random variables (mean = 0, variance = 1).
- U_1 and U_2 are uniform random variables between 0 and 1.

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD normal.

*Use the Box–Muller transform sampling method
    DATA(lt_numbers) = standard( iv_size ).
* Generate random numbers with normal distribution
    r = VALUE #( FOR x IN lt_numbers ( x * sqrt( iv_variance ) + iv_mean ) ).

  ENDMETHOD.

  METHOD standard.
**********************************************************************
*   Box–Muller transform sampling method
**********************************************************************
* Set the number of random numbers
    DATA(i) = SWITCH i( size MOD 2 WHEN 0 THEN size / 2 ELSE ( size + 1 ) / 2 ).
* Generate numbers according to the Box–Muller Transform Sampling Method
    DO i TIMES.
* Generate two uniform random numbers (between 0 and 1)
      DATA(lt_u_1) = uniform( ).
      DATA(lt_u_2) = uniform( ).
* Generate two independent random numbers from a uniform distribution on (0,1]
     APPEND ( sqrt( ( -2 ) * log( CONV f( lt_u_1[ 1 ] ) ) ) * cos( 2 * acos( -1 ) * CONV f( lt_u_2[ 1 ] ) ) ) TO r.

     IF sy-index EQ i AND i MOD 2 EQ 0.
       APPEND ( sqrt( ( -2 ) * log( CONV f( lt_u_1[ 1 ] ) ) ) * sin( 2 * acos( -1 ) * CONV f( lt_u_2[ 1 ] ) ) ) TO r.
     ENDIF.

    ENDDO.

  ENDMETHOD.
```

If we want for example generate a sample of 1000 values from a normal distribution with mean = -3 and variance 13, the method call looks like this:

```
DATA(lt_normal_values) = zcl_statistical_methods =>normal( iv_mean = '-3' iv_variance = 13 iv_size = 1000
).
```

Bernoulli Distribution

The Bernoulli distribution is a discrete probability distribution that describes a random experiment with only two possible outcomes:

1. Success (usually denoted as 1) with probability *p*.

2. Failure (denoted as 0) with probability 1-*p*.

It is the simplest type of discrete probability distribution and forms the basis of the binomial distribution.

The probability mass function (PMF) of a Bernoulli-distributed random variable X is:

$$P(X = x) = \begin{cases} p, & \text{if } x = 1 \\ 1 - p, & \text{if } x = 0 \end{cases}$$

where:

- p is the probability of success ($0 \leq p \leq 1$).
- 1-p is the probability of failure.

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD bernoulli.

* Generate uniform distribution of size iv_size
    DATA(lt_unif) = uniform( iv_size = iv_size ).
* Generate Discrete Bernoulli Distribution
    r = VALUE #( FOR u IN lt_unif ( COND #( WHEN u LE iv_p THEN 1 ELSE 0 ) ) ).

  ENDMETHOD.
```

If we want for example generate a sample of 100 values from a bernoulli distribution with probability parameter = 0.8, the method call looks like this:

```
DATA(lt_bernoulli_values) = zcl_statistical_methods => bernoulli( iv_p = '0.8' iv_size = 1000 ).
```

Binomial Distribution

The binomial distribution is a discrete probability distribution that describes the number of successes in a fixed number of independent trials, where each trial has only two possible outcomes: success (1) or failure (0).

It is an extension of the Bernoulli distribution, which models a single trial.

The probability of getting exactly k successes in n trials is:

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

where:

- n = number of trials
- p = probability of success in each trial
- 1 - p = probability of failure
- X = number of successes in n trials (can be 0, 1, 2, ..., n)

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
  METHOD binomial.
```

```
* Generate Discrete Binomial Distribution
   r = VALUE #( FOR i = 0 UNTIL i = iv_size ( REDUCE f( LET b = bernoulli( iv_size = iv_size iv_p = iv_p )
IN INIT s = 0 FOR x IN b NEXT s = s + x ) ) ).

 ENDMETHOD.
```

If we want for example generate a sample of 100 values from a binomial distribution with probability parameter = 0.4 and number of trials = 15, the method call looks like this:

```
DATA(lt_binomial_values) = zcl_statistical_methods => binomial( iv_n = 15 iv_p = '0.4' iv_size = 1000 ).
```

Geometric Distribution

The Geometric distribution is a discrete probability distribution that models the number of trials needed before the first success occurs in a series of independent Bernoulli trials, where each trial has the same probability of success *p*. In other words, it answers the question: "How many trials does it take to get the first success?"

It is a memoryless distribution, meaning the probability of success on the next trial is always the same, regardless of the number of previous trials.

The probability mass function (PMF) of a Geometric random variable X is:

$$P(X = k) = (1 - p)^{k-1}p, \quad k = 1, 2, 3, \ldots$$

where:

- p = probability of success on a single trial (i.e., $0 < p \leq 1$)
- X = the number of trials until the first success (can be 1, 2, 3, ...)
- $(1 - p)^{k-1}$ represents the probability of $k - 1$ failures before the first success.

The method in the ZCL_STATISTICAL_METHODS class is implemented as follows:

```
 METHOD geometric.

* Generate Discrete Geometric Distribution
   r = VALUE #( FOR i = 0 UNTIL i = iv_size ( ceil( log( 1 - _unique( uniform( ) ) ) / log( 1 – iv_p ) ) ) ).

 ENDMETHOD.
```

If we want for example generate a sample of 100 values from a geometric distribution with probability parameter = 0.6, the method call looks like this:

```
DATA(lt_geometric_values) = zcl_statistical_methods =>geometric( iv_p = '0.6' iv_size = 100 ).
```

Conclusion

Advanced statistical methods are essential tools for extracting meaningful insights from complex data. Whether for regression analysis, time series forecasting, machine learning, or hypothesis testing, these techniques help researchers, analysts, and businesses make data-driven decisions. By using these methods, organizations can optimize their operations, improve their products and services, and gain a competitive edge in their respective industries.

SAP ABAP (Advanced Business Application Programming) is a powerful programming language used to develop applications within the SAP ecosystem, particularly in enterprise resource planning (ERP) systems. Statistics in SAP ABAP are crucial for ensuring optimal system performance and efficient application operation. By collecting

and analyzing performance data, developers and system administrators can identify issues, optimize processes, and improve the overall user experience.

Unfortunately, the ABAP language does not offer a standard library where some of the mostcommon descriptive statistics functions have been included together with simple tools to generate distributions and produce empirical inference analyses. Therefore, this article can be very helpful for ABAP programmers.

References

- Amedee, R., Winters, A., Winters, R. (2010), 'Statistics: A brief overview', *The Ochsner Journal,* 10(3), 213–216.
- Bandari, K. (2021), 'Complete ABAP', *SAP Press Rheinwerk*.
- Brito, N., Naranjo, S., Nunez, V., Ordonez, E. (2022), 'Analysis of the use of the Python programming language for statistical calculations', *Espirales revista multidisciplinaria de invesitgación científica*, 6(2), 1-13.
- Cobb, G. (2015), "Mere Renovation is Too Little Too Late: We Need to Rethink our Undergraduate Curriculum from the Ground Up," *The American Statistician*, 69, 266–282.
- Deborah, N., Duncan, T. (2015), 'Explorations in Statistics Research: An Approach to Expose Undergraduates to Authentic Data Analysis', *The American Statistician,* 69, 292-299.
- Longnecker, M. (2021), 'An Introduction to Statistical Methods and Data Analysis', *Cengage Learning, Inc.*
- Ren, Z. (2020), 'An Example of Using MATLAB for Data Analysis—The Correlation between College Entrance Exam and Students' Performance at Universities', *PiscoMed Publishing*, 2(2), 39-43.
- Sengur, D. (2023), 'Using of MATLAB Statistics Toolbox for Data Analysis in Social Sciences with Chat GPT-3 prompts', *Turkish Journal of Science and Technology*, 18(2), 353-361.
- Shaw, S., Son, J., Stigler, J., Tucker, M. (2022), 'Teaching Statistics and Data Analysis with R', *Journal of Statistics and Data Science Education*, 31(1), 1-32.
- Sprent, P. (2003), 'Statistics in medical research', *Swiss Med Wkly*, 133, 522–529.