The Matrex Functions Guide

Table of Contents

What are function templates

Matrex is equivalent to a spreadsheet application. The equivalent to spreadsheet formulas (for example *log*) are the Matrex **function templates**.

When you apply a template to matrices and/or parameters in a project, it becomes a **function**.

So, if x is a matrix and log a template, log(x) is a function.

Function templates and spreadsheet formulas

In Matrex version 1.0 there are **105** function templates. Some of them are very simple (like exp, plus...), others are more complex (query, sumby...).

The idea is that you should always find a template or a **combination** of templates that is equivalent to a spreadsheet formula.

If you do not, it is generally because:

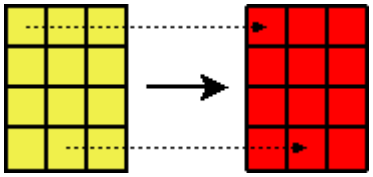
- The template is used for **special purposes** (financial, chemical...), and therefore is not included in the main library. You can possibly find it in an optional library.

- Sometimes the fact that Matrex templates **are applied to blocks**, not to cells, makes it impossible to have a template that is equivalent to spreadsheet formulas that depend by the cells structure of spreadsheets.
  In this case it is generally needed to **reformulate** the problem to solve in terms of matrices.
- Some spreadsheet formulas are **simply not needed** in Matrex. For example formulas to handle set of cells that change size dynamically.

If still you don't find a template that is equivalent to a spreadsheet formula, you can require it or, if you prefer, build it yourself.

Templates Types

Some of the templates can be called **scalar**, because they just apply a mathematical function to each item of the input matrix and write the result in the same position in the result matrix. Example of these templates are sys.standard.exp and sys.standard.plus. Many of the templates in the standard package are of this kind.



Other templates can be called **vector** templates, because they act on the columns of the input matrices (for example sys.db.query, sys.inport.csv, sys.util.sort).

The **matrix** templates are those working on the whole input matrix, like all the templates in sys.matrix but also many of the ones in sys.generation and sys.util.

...as

What it means when the name of a template ends with **"as"**? It means that the size of the output matrix will be the same as the input matrix.

For example the output matrix of **sys.generation.fillas**, which contains everywhere the same number, has the same size as the input matrix.

sys.generation.fill instead does not need input matrices, because the size of the output matrix is determined by the sizeX and sizeY parameters.

..as templates are very useful. A typical **example** is when you need to sum to to all the numbers contained in matrex “a” the same number (suppose 1). In the expression parser you would enter the expression:

plus(a, fillas(a, “1”))

Templates catalog

This guide does not want to cover all the single templates. Its purpose is to explain how to **find the template that solves your problem** and how to use the templates together.

Since the templates are organized in packages, I will discuss them package by package.

sys.date

This package contains all templates that have to do directly with dates.

To build dates from numerical values (year, month...) use **compose**.

Alternatively you can convert texts to dates using the **sys.util.fromstring** template.

A third possibility is to use **frommilliseconds**, which converts the number of milliseconds from 1.1.1970 to dates.

The template compose counterpart is **extract** that extracts parts from dates (year, month...) .

The template sys.util.fromstring counterpart is **sys.util.tostring**, which converts dates (also) to strings. Tostring does not allow you to specify the format of the result strings. To do that use instead **format**.

The template frommilliseconds counterpart is **tomilliseconds**, which converts dates to milliseconds.

Once you have a matrix of dates, you can sum (or subtract) numeric values (years, months...) to them, using **add**.

fill and **fillas** are used to fill a matrix with a specific date (today, now, yesterday and tomorrow).

interval and **intervalas** are used to build a vector containing a date interval. They can be useful to build charts.

sys.db

This package contains the templates that have to do with databases.

By now the only template in this package is **query**. This templates converts the columns resulting from an SQL query in vectors.

sys.generation

This package contains the templates to generate matrices (no input matrix, only output matrices).

To generate a matrix containing everywhere the same number use **fill** or **fillas**.

To generate a vector containing a numeric interval use **interval** or **intervalas**. Useful to build charts.

To generate a matrix containing random numbers use **random** or **randomas**. Useful for tests when you need a matrix of a certain size, but with any content.

To generate a matrix as the replication of another matrix use **tile** and **tileas**. It is equivalent to fill, just the element filling up the result matrix is not a number, but a matrix.

sys.import

This package contains the templates to import matrices from external souces (files...).

By now the only template in this package is **csv**. It is used to import vectors from the columns of a CSV or SDV file.

sys.logical

This package contains the templates to do logical operations among matrices.

and, or, not apply the and, or and not operations to boolean matrices.

if returns a matrix containing items of the *then* or *else* matrices depending by the items of the boolean *condition* matrix.

sys.mathstat.by

This package contains all the ...by templates. They work in the same way as all the aggregate functions in SQL. For example the equivalent of *SELECT SUM(...) ... GROUP BY ...)* is **sumby**.

The ...by templates are used to aggregate the values of the input matrices. The difference with, for example, sys.mathstat.sum, is that the final result contains **more than one**

element per column.

This is obtained using the first column of the **keys** input matrix as the keys for this sum. For each key, the rows with this key are summarized in the same row.

For example if the first column contains (*a,b,b,a,a,b,a*) the result contains 2 rows, *a* and *b*. The *a* row contains the sum of all the *a* rows, and the same for the *b* row.

The reason to use these templates is the same of using *SELECT SUM(...) ... GROUP BY ...)* when querying a database. For example when you want to know the average price payd to buy a car, for each car model.

sys.mathstat.distributions

This package contains the templates to calculate *PDF* (probability density function) and *CDF* (cumulative density function) of the various statistical distributions.

Normal, exponential, poisson, binomial, chisquared, hypergeometric distributions are included.

sys.mathstat

This package contains aggregate templates: **sum, arithmeticmean, geometricmean, min, max, variance, iterativesum** and **product**.
All of these, a part iterativesum, return a vector with one aggregate element for each column of the input matrix.
For example sum calculates, for each column of the input matrix, the sum of all the elements of the column and writes it in the output vector.

Also **variance**, which is a statistical template, work in the same way calculating the variance for each column of the input matrix.

iterativesum is different because it sums iteratively all the items of the input matrix by column, obtaining a matrix with the same size of the input matrix. So, if you have the vector [1,2,3] as input, the output will be [1,3,6]

sys.matrix

This package contains the templates calculating mathematics of the matrices. The code of these templates uses the **Apache commons-math** library (http://jakarta.apache.org/commons/).

So you have **determinant, norm, identity, transpose, inverse**.

But you have also **times**, which calculates a matrices product.

And you have **solve**, which solves a linear system with as coefficient the first matrix and as constant the second.

sys.standard.trigonometric

This package contains the templates calculating trigonometric functions. They are scalar operators, so they are applied to each element of the input matrices.

They are **cos, sin, tan, acos, asin, atan, cosh, sinh, tanh.**

sys.standard

This package contains the mathematical templates that are used more frequently. They are scalar operators, so they are applied to each element of the input matrices.

plus, minus, times, divide are equivalent to the binary operators +, -, *, /.

div, mod are the integer division and module.

equal, less, lesseq, more, moreeq are equivalent to the logical operators =, <, <=, >, >=.

minus1 negates the argument, **sign** returns its sign (1 for positive, 0 for 0, -1 for negative).

abs, exp, log, pow, sqrt behave in the same way as the well known functions with the same name.

ceil, floor, round, trunc convert decimal numbers to integers.

degrees, radians convert between degrees and radians.

binaryblock is a more complex template. It applies a binary function (plus, minus, times divide, pow) to the first input matrix and to blocks of the second input matrix.

sys.string

This package contains the templates that work specifically with text matrices. They are scalar operators, so they are applied to each element of the input matrices.

cat concatenates texts. It means that each element of the output matrix is the concatenation of the same element in each input matrix.

compare, equals compare the elements of the input text matrices.

substring extracts a substring of each element of the input text matrix.

trim removes the spaces from the start and the end of each element of the input text matrix.

lowercase and **uppercase** change the case of the texts in the input matrix.

sys.util

This package contains the templates that work with the structure of the matrices (break them in pieces, search in them...).

They are generally the most used when the project is of medium or big size.

hbreak, vbreak, hglue, vglue break matrices in pieces or put them together in bigger matrices.

at returns the elements of a matrix with the given indices, **lookup** returns the indices of some elements in a matrix. These templates are normally used together.

size returns the sizes of some matrices.

fromstring, tostring convert a text matrix to any kind of matrix and viceversa.

sort sorts all the input matrices by the first column of the first matrix.

transpose transposes any kind of matrix, unlike sys.matrix.transpose, which transposes only numeric matrices.

queue adds matrix *x* at the bottom, top, right or left of matrix *y*. Since it does it each time the *x* changes, it queues all the changes of *x* to *y*.