

LaTeX Equations with symbolicMatrix, Eqn and matrix2latex

Phil Chalmers, Michael Friendly

07 August, 2024

L^AT_EX is the de facto standard for communication and publication in scientific documents and it is very easy to typeset mathematical expressions like Pythagoras' theorem, $a^2 + b^2 = c^2$ (using `a^2 + b^2 = c^2`) once you learn the notation. Writing equations with arrays, matrices and vectors is somewhat more challenging. Many people rely on interactive L^AT_EX editors like Overleaf, MathType, or online versions that provide a menu-driven interface with fill-in templates for matrices.

There are already some tools available in R for producing L^AT_EX output for tables (`xtable::xtable()`), R objects (`Hmisc::latex()`) and statistical models (`equatiomatic::extract_eq()`).

The `matlib` package extends these, providing a collection of functions that simplify using L^AT_EX notation for matrices, vectors and equations in documentation and in writing:

- `symbolicMatrix()`: Constructs the L^AT_EX code for a symbolic matrix, whose elements are a symbol, with row and column subscripts.
- `matrix2latex()`: Constructs the L^AT_EX code for a symbolic matrix, whose elements are a symbol, with row and column subscripts
- `showEqn()`: Shows what matrices **A**, **b** look like as the system of linear equations, **Ax = b**, but written out as a set of equations.
- `Eqn()`: A wrapper to produce LaTeX expressions that can be copied/pasted into documents or used directly in `.Rmd` or `.qmd` documents to compile to equations.

When used directly in R, they produce their output to the console (using `cat()`). In a `.Rmd` or `.qmd` document, use the chunk options: `results='asis'`, `echo=FALSE` so that `knitr` just outputs the text of the equations to the document. The rendering of the equations is handled by `pandoc`.

Using symbolicMatrix() and Eqn()

`symbolicMatrix()` constructs the L^AT_EX code for a symbolic matrix, whose elements are a symbol, with row and column subscripts. For example, by default (with no arguments) it produces the expression for a matrix whose elements are x_{ij} , for $i = 1, 2, \dots, n; j = 1, 2, \dots, m$ in a L^AT_EX `\begin{pmatrix} \dots \end{pmatrix}` environment that looks like this:

```
\begin{pmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{21} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & & \vdots \\
  x_{n1} & x_{n2} & \cdots & x_{nm}
\end{pmatrix}
```

The code above appears in the console. To render this as a matrix in a document, this must be wrapped in a display math environment, which is provided by `Eqn()` and used in a code chunk with the `results = 'asis'` option, giving:

```
symbolicMatrix() |> Eqn(number = FALSE)
```

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

For chunk output in a document, you will get a L^AT_EX error, “missing \$ inserted” if you forget to use `Eqn()` or otherwise fail to make a math environment.

Some other examples:

- A 3×3 identity matrix with square brackets, specified as an equation with a left-hand side \mathbf{I}_3 . The first argument to `symbolicMatrix()` can be any numeric matrix.

```
symbolicMatrix(diag(3), lhs = "\\mathbf{I}_3",
               matrix="bmatrix") |>
Eqn(number = FALSE)
```

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- A row or column vector; the first argument to `symbolicMatrix()` must be a **matrix**, so wrap an R vector in `matrix()`, supplying `nrow=1` (or `ncol = 1`):

```
symbolicMatrix(matrix(LETTERS[1:4], nrow=1)) |> Eqn(number = FALSE)
```

$$(A \quad B \quad C \quad D)$$

```
symbolicMatrix(matrix(letters[1:3], ncol=1)) |> Eqn(number = FALSE)
```

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

The SVD

As a more complicated example, here we write out the L^AT_EX equation for the singular value decomposition (SVD) of a general $n \times p$ matrix \mathbf{X} using `Eqn()` and `symbolicMatrix()`. In Rmd markup, `Eqn()` can be given an equation label. Two calls to `Eqn()` produce separate equations in the output.

Both of these equations are numbered (by default). (`Eqn()` uses the L^AT_EX `equation` environment, `\begin{equation} ... \end{equation}`, or `equation*` if the equation is not numbered (`number = FALSE`)). The two calls to `Eqn()` are rendered as separate equations, center aligned.

```
Eqn("\\mathbf{X} = \\mathbf{U} \\mathbf{\\Lambda} \\mathbf{V}^{\\top}", label='eqn:svd')
Eqn(symbolicMatrix("u", "n", "k", lhs = ''),
     symbolicMatrix("\\lambda", "k", "k", diag=TRUE),
     symbolicMatrix("v", "k", "p", transpose = TRUE))
```

This produces the two numbered equations:

$$\mathbf{X} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^{\top} \tag{1}$$

$$= \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ v_{21} & v_{22} & \cdots & v_{2p} \\ \vdots & \vdots & & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kp} \end{pmatrix}^{\top} \quad (2)$$

The matrix names in Equation (1) are printed in a **boldface** math font, typically used for matrices and vectors. Note that when using L^AT_EX code in R expressions each backslash (\) must be doubled (\\) in R because \ is the escape character.

Note that the first equation can be referenced because it was labeled: “As seen in Equation (1) ...”. (In Quarto, equation labels must be of the form #eq-label and equation references are of the form @eq-label)

Systems of equations

As another example, the chunk below shows a system of equations $\mathbf{Ax} = \mathbf{b}$ written out using symbolic matrices. Note the use of `cat()` to insert arbitrary L^AT_EX into the stream

```
Eqn(symbolicMatrix("a", nrow = "m", ncol="n", matrix="bmatrix"),
     symbolicMatrix("x", nrow = "n", ncol=1),
     cat("\\quad=\\quad"),
     symbolicMatrix("b", nrow = "m", ncol=1))
```

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \quad (3)$$

aligned environment

You can also align separate equations relative to some symbol like an = sign to show separate steps of re-expression, using the option `Eqn(..., align=TRUE)`.

Show the singular value decomposition again, but now as two separate equations aligned after the = sign. Note the locations of the & operator for alignment, specified as the left-hand side (`lhs`) of the second equation.

```
Eqn("\\mathbf{X} &= \\mathbf{U} \\mathbf{\\Lambda} \\mathbf{V}^{\\top}",
     Eqn_newline(),
     symbolicMatrix("u", "n", "k", lhs = '&'),
     symbolicMatrix("\\lambda", "k", "k", diag=TRUE),
     symbolicMatrix("v", "k", "p", transpose = TRUE),
     align=TRUE)
```

$$\mathbf{X} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^{\top} \quad (4)$$

$$= \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ v_{21} & v_{22} & \cdots & v_{2p} \\ \vdots & \vdots & & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kp} \end{pmatrix}^{\top} \quad (5)$$

Note that in this example, there are three calls to `symbolicMatrix()`, wrapped inside `Eqn()`. `Eqn_newline()` emits a newline (\\) between equations.

matrix2latex

The `matrix2latex()` function can also generate symbolic equations from numeric or character matrices. For numeric matrices, it can round the values or show results as fractions.

```
A <- matrix(1:12, nrow=3, ncol=4, byrow = TRUE) / 6
matrix2latex(A, fractions = TRUE, brackets = "b") |> Eqn(number = FALSE)
```

$$\begin{bmatrix} 1/6 & 1/3 & 1/2 & 2/3 \\ 5/6 & 1 & 7/6 & 4/3 \\ 3/2 & 5/3 & 11/6 & 2 \end{bmatrix}$$

Say we want to show the matrix $[A|b]$ involved in the system of equations $Ax = b$. Create these as a character matrix and vector:

```
A <- matrix(paste0('a_', 1:9), 3, 3, byrow = TRUE) |> print()
```

```
##      [,1] [,2] [,3]
## [1,] "a_1" "a_2" "a_3"
## [2,] "a_4" "a_5" "a_6"
## [3,] "a_7" "a_8" "a_9"
```

```
b <- paste0("\\beta_", 1:3) |> print()
```

```
## [1] "\\beta_1" "\\beta_2" "\\beta_3"
```

Then use `matrix2latex()` on `cbind(A,b)` and pipe the result of `matrix2latex()` to `Eqn()`:

```
matrix2latex(cbind(A,b)) |> Eqn(number=FALSE)
```

$$\begin{bmatrix} a_1 & a_2 & a_3 & \beta_1 \\ a_4 & a_5 & a_6 & \beta_2 \\ a_7 & a_8 & a_9 & \beta_3 \end{bmatrix}$$

All the R tricks for creating and modifying matrices can be used in this way.

showEqn

`showEqn()` is designed to show a system of linear equations, $Ax = b$, but written out as a set of equations individually. With the option `latex = TRUE` it writes these out in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ form.

Here, we create a character matrix containing the elements of a 3×3 matrix A , whose elements are of the form a_{ij} and two character vectors, b_i and x_i .

```
A <- matrix(paste0("a_{", outer(1:3, 1:3, FUN = paste0), "}"),
            nrow=3) |> print()
```

```
##      [,1]      [,2]      [,3]
## [1,] "a_{11}" "a_{12}" "a_{13}"
## [2,] "a_{21}" "a_{22}" "a_{23}"
## [3,] "a_{31}" "a_{32}" "a_{33}"
```

```
b <- paste0("b_", 1:3)
```

```
x <- paste0("x", 1:3)
```

`showEqn(..., latex = TRUE)` produces the three equations in a single `\begin{array} ... \begin{array}` environment.

```
showEqn(A, b, vars = x, latex=TRUE)
```

If this line was run in an R console, it would produce:

```
\begin{array}{llllllll}
a_{11} \cdot x_1 & + & a_{12} \cdot x_2 & + & a_{13} \cdot x_3 & = & b_1 & \\
a_{21} \cdot x_1 & + & a_{22} \cdot x_2 & + & a_{23} \cdot x_3 & = & b_2 & \\
a_{31} \cdot x_1 & + & a_{32} \cdot x_2 & + & a_{33} \cdot x_3 & = & b_3 & \\
\end{array}
```

Evaluating the above code in an unnumbered L^AT_EX math environment via `Eqn()` gives the desired result:

```
showEqn(A, b, vars = x, latex=TRUE) |> Eqn(number=FALSE)
```

$$\begin{array}{rclclclcl} a_{11} \cdot x_1 & + & a_{12} \cdot x_2 & + & a_{13} \cdot x_3 & = & b_1 \\ a_{21} \cdot x_1 & + & a_{22} \cdot x_2 & + & a_{23} \cdot x_3 & = & b_2 \\ a_{31} \cdot x_1 & + & a_{32} \cdot x_2 & + & a_{33} \cdot x_3 & = & b_3 \end{array}$$