R Package mat2tex: Easily inject R matrices in LaTeX math equations

Mark Heckmann
University of Bremen, Germany
Version 0.1
https://github.com/markheckmann/mat2tex

Abstract

This document describes how to use the **mat2tex** package. The mat2tex package was written to facilitate the combination of matrices and LaTeX code. **mat2tex** is a mini-language with several operators and functions to allow to combine LaTeX math equation code and R objects, especially matrices, very easily.

Keywords: matrix, knitr, R.

1. Installation

To use `mat2tex` within LaTeXdocuments install the package from *github*. Windows users have to make sure that *Rtools* is installed in order to install from gihub.

```
library(devtools)
install_github("mat2tex", "markheckmann")
```

Then load the **mat2tex** package.

```
library(mat2tex)
```

2. Quick start

It is recommended to add `\usepackage{amsmath}` in the document preamble. Otherwise not all math environments `mat2tex` includes can be used. Now we can start. Let's create the matrix A.

```
set.seed(1)
A <- matrix(runif(4), 2)
```

To display the matrix wrap the folling code in `knitr` chunk with the arguments `echo=FALSE` and `results='asis'`. The following code concatenates the `texcode` chunks using the `%_%`

operator. The first chunk adds the math environment `$$` as a string. The second converts the matrix A into LaTeXcode. The last one closes the `$$` environment again.

```
"$$" %_% xm(A) %_% "$$"
```

$$\begin{pmatrix} 0.27 & 0.57 \\ 0.37 & 0.91 \end{pmatrix}$$

You may use the `xx` function to get the same results.

```
xx(A)
```

$$\begin{pmatrix} 0.27 & 0.57 \\ 0.37 & 0.91 \end{pmatrix}$$

Here also, the shorthand `$$` environment is used which is the default in `mat2tex` as it is the standard for RMarkdown files. Working with `.Rnw` files it is conventient to change the default settings to the `\equation` environment to get numerated equations instead by typing

```
mat2tex_options(mathenvir=3)
```

Now we get numbered equations as the default.

```
xx(A)
```

$$\begin{pmatrix} 0.27 & 0.57 \\ 0.37 & 0.91 \end{pmatrix} \tag{1}$$

To reference Equation (2) you may also add a label using the `label` argument. So you can reference it using `\eqref{mylabel}` or `\ref{mylabel}` in your `.Rnw` document.

```
xx(A, label="mylabel")
```

$$\begin{pmatrix} 0.27 & 0.57 \\ 0.37 & 0.91 \end{pmatrix} \tag{2}$$

Let's create one final example. We want to display the values of the singular value decomposition of A. This time we want to use square brackets and display three digits. To achieve this set the default matrix type to `bmatrix` (the default is `pmatrix`) and to `digits=3`.

```
mat2tex_options(mtype="bmatrix", digits=3)
```

We will use the function `xmt` which is the same as `xm` except that it additionally adds a transpose sign to the matrix.

```
d <- svd(A)
xx("A = UDV^T =", d$u, diag(d$d), xmt(d$v))
```

$$A = UDV^T = \begin{bmatrix} -0.541 & -0.841 \\ -0.841 & 0.541 \end{bmatrix} \begin{bmatrix} 1.167 & 0.000 \\ 0.000 & 0.024 \end{bmatrix} \begin{bmatrix} -0.391 & -0.920 \\ -0.920 & 0.391 \end{bmatrix}^T \tag{3}$$

To find more examples and get more information have a look at the other package vignettes or visit `https://github.com/markheckmann/mat2tex`.

3. Math environments

If you included `\usepackage{amsmath}` in the preamble you can make use of several math environments defined in the package. The environments are explained here `https://www.sharelatex.com/learn/Aligning_equations_with_amsmath`. To indicate the math environment you can either use the corresponding number or its name (see `?xx`).

equation and split

Here you can see that `xx` output can be nested, i.e. math environemnts can be nested.

```
splt <- xx("AA^T & = ", A, t(A), "\\\\",
           "& =", A %*% t(A), e="split")
xx(splt, e="equation", label="eq1")
```

$$\begin{split} AA^T &= \begin{bmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{bmatrix} \begin{bmatrix} 0.266 & 0.372 \\ 0.573 & 0.908 \end{bmatrix} \\ &= \begin{bmatrix} 0.399 & 0.619 \\ 0.619 & 0.963 \end{bmatrix} \end{split} \tag{4}$$

multiline

Looks ugly here. For demonstration purposes only.

```
xx("AA^T  = ", A, t(A), "\\\\",
   " =", A %*% t(A), e=9, label="eq2")
```

$$\begin{multline} AA^T = \begin{bmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{bmatrix} \begin{bmatrix} 0.266 & 0.372 \\ 0.573 & 0.908 \end{bmatrix} \\ = \begin{bmatrix} 0.399 & 0.619 \\ 0.619 & 0.963 \end{bmatrix} \end{multline} \tag{5}$$

align

Sam as *equation and split* but each line is numbered.

```
xx("AA^T  &= ", A, t(A), "\\\\",
   "&=", A %*% t(A), e=5, label="eq3")
```

$$
\begin{aligned}
AA^T &= \begin{bmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{bmatrix} \begin{bmatrix} 0.266 & 0.372 \\ 0.573 & 0.908 \end{bmatrix} \qquad (6) \\
&= \begin{bmatrix} 0.399 & 0.619 \\ 0.619 & 0.963 \end{bmatrix} \qquad (7)
\end{aligned}
$$

gather

```
xx("AA^T  = ", A, t(A), "\\\\",
   "=", A %*% t(A), e=7, label="eq3b")
```

$$
\begin{gathered}
AA^T = \begin{bmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{bmatrix} \begin{bmatrix} 0.266 & 0.372 \\ 0.573 & 0.908 \end{bmatrix} \qquad (8) \\
= \begin{bmatrix} 0.399 & 0.619 \\ 0.619 & 0.963 \end{bmatrix} \qquad (9)
\end{gathered}
$$

4. Matrix types

You can generate different matrix types using the `mtype` argument in `xm`. Available types are defined in the `amsmath` package: `matrix`, `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` and `Vmatrix`.

```
xx(xm(A, mtype="matrix"))
```

$$
\begin{matrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{matrix} \qquad (10)
$$

```
xx(xm(A, mtype="pmatrix"))
```

$$
\begin{pmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{pmatrix} \qquad (11)
$$

```
xx(xm(A, mtype="bmatrix"))
```

$$\begin{bmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{bmatrix} \tag{12}$$

To print row and/or column names as well, bordermatrix can be used.

```
rownames(A) <- letters[1:2]
colnames(A) <- LETTERS[1:2]
xx(xm(A, mtype="bordermatrix"))
```

$$\begin{array}{cc} & \begin{array}{cc} A & B \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{pmatrix} 0.266 & 0.573 \\ 0.372 & 0.908 \end{pmatrix} \end{array} \tag{13}$$

5. More features

Beside the `%_%` operator, there are similar operators that additionally enter horizontal spaces in the formula. The operator `%_1%`, `%_2%` to `%_5%` will enter spaces of ascending width. `%_0%` will insert a slighly negative width.

```
"$$ 1 + 1 =" %_0% "2 $$"
"$$ 1 + 1 =" %_% "2 $$"     # default operator
"$$ 1 + 1 =" %_1% "2 $$"
"$$ 1 + 1 =" %_2% "2 $$"
"$$ 1 + 1 =" %_3% "2 $$"
"$$ 1 + 1 =" %_4% "2 $$"
"$$ 1 + 1 =" %_5% "2 $$"
```

$$1 + 1 =\!2$$

$$1 + 1 = 2$$

$$1 + 1 =\,2$$

$$1 + 1 =\:2$$

$$1 + 1 =\;2$$

$$1 + 1 =\quad 2$$

$$1 + 1 =\qquad 2$$

You can also apply custom spaces using the `s` function. The function takes a numeric value which defines the width as a multiple of the letter `m`. The function also accepts negative values.

```r
"$$ 1 + 1= " %_% s(2) %_% "2 $$"
```

$$1 + 1 = \quad 2$$

Or equivalently

```r
xx("1 + 1= ", s(2), "2")
```

Affiliation:

Mark Heckmann
University of Bremen, Germany
E-mail: heckmann@uni-bremen.de
R-blog: http://ryouready.wordpress.com
Website: http://www.markheckmann.de