

# DOCUMENTATION FOR MATRIX2LATEX

ØYSTEIN BJØRNDAL

Takes a python matrix or nested list and converts to a LaTeX table or matrix. Author: ob@cakebox.net

This software is published under the GNU GPL, by the free software foundation. For further reading see: <http://www.gnu.org/licenses/licenses.html#GPL>

The following packages and definitions are recommended in the latex preamble

```
% scientific notation, 1\{e}{9} will print as 1x10^9
\providecommand{\e}[1]{\ensuremath{\times 10^{\#1}}}
\usepackage{amsmath} % needed for pmatrix
\usepackage{booktabs} % Fancy tables
...
\begin{document}
...
```

## 1. ARGUMENTS

1.1. **matrix.** A numpy matrix or a nested list TODO:

- Any python structure that looks like a rectangular matrix.
- Remove dependency on numpy (might be more portable to other systems)

1.2. **Filename.** File to place output, extension .tex is added automatically. File can be included in a LaTeX document by `\input{filename}`. If filename is None or not a string, output will be returned in a string

1.3. **\*environments.** Use `matrix2latex(m, None, "align*", "pmatrix", ...)` for matrix. This will give

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Use `matrix2latex(m, "test", "table", "center", "tabular" ...)` for table. Table is default so given no arguments: table, center and tabular will be used. The above command is then equivalent to `matrix2latex(m, "test", ...)`

1.4. **\*\*keywords.**

1.4.1. *transpose.* Flips the table around in case you messed up. Equivalent to `matrix2latex(m.H, ...)` if m is a numpy matrix.

1.4.2. *format*. Printf syntax format, e.g. `("%.2f$"`. Default is `("%g$"`. This format is then used for all the elements in the table.

1.4.3. *formatColumn*. A list of printf-syntax formats, e.g. `["%.2f$", "%g$"]`. Must be of same length as the number of columns. Format `i` is then used for column `i`.

1.4.4. *alignment*. Used as an option when `tabular` is given as environment. `\begin{tabular}{alignment}` A latex alignment like `c`, `l` or `r`. Can be given either as one per column e.g. `"ccc"`. Or if only a single character is given e.g. `"c"`, it will produce the correct amount depending on the number of columns. Default is `"r"`.

1.4.5. *rowLabels*. A row at the top used to label the columns. Must be a list of strings.

1.4.6. *columnLabels*. A column used to label the rows. Must be a list of strings

1.4.7. *caption*. Use to define a caption for your table. Inserts `\caption` after `\end{tabular}`.

1.4.8. *label*. Used to insert `\label{...}` after `\end{tabular}` Default is filename without extension.

Both `caption` and `label` will do nothing if `tabular` environment is not used.

```
from matrix2latex import matrix2latex
from numpy import matrix
m = matrix("1 2 4;3 4 6") # numpy matrix or
m = [[1, 2, 4], [3, 4, 6]] # python nested list
matrix2latex(m, "test", "table", "center", "tabular", format="%.2f$", alignment="lcr")
# or since table, center and tabular is default:
t = matrix2latex(m, format="%.2f$", alignment="lcr")
# produces:
```

1.00	3.00
2.00	4.00
4.00	6.00

## 2. USAGE EXAMPLES

### 2.1. Minimal.

```
from matrix2latex import matrix2latex
m = [[1, 2, 3], [1, 4, 9]] # python nested list
t = matrix2latex(m)
print t
```

---

1	1
2	4
3	9

---

2.2. **Labels.** Using the same minimal example from above we can add row labels

```
rl = ['$x$', '$x^2$']
t = matrix2latex(m, rowLabels=rl)
```

---

$x$	$x^2$
1	1
2	4
3	9

---

Or swapping it around

```
cl = ['$x$', '$x^2$']
t = matrix2latex(m, columnLabels=cl, transpose=True)
```

---

$x$	1	2	3
$x^2$	1	4	9

---

2.3. **Caption.** We can easily add a caption

```
t = matrix2latex(m, rowLabels=rl,
                  caption='Nice table!')
```

TABLE 1. Nice table!

---

$x$	$x^2$
1	1
2	4
3	9

---

2.4. **Label.** We can use `label='niceTable'` but if we save it to file the default label is the filename, so:

```
matrix2latex(m, 'niceTable', rowLabels=rl,
             caption='Nice table!')
```

can be referenced by `\ref{niceTable}`. Table 2 was included in latex by `\input{niceTable}`.

TABLE 2. Nice table!

$x$	$x^2$
1	1
2	4
3	9

TODO: add simple real world example.