The arraycols package[*]

Antoine Missier
antoine.missier@ac-toulouse.fr

May 4, 2024

1 Introduction

Although the remarkable tabularray package by Jianrui Lyu [1], developed in LaTeX3, offers many new possibilities and great flexibility in composing tables, many users are still familiar with Frank Mittelbach and David Carlisle's array package [2]. In addition to array, this modest arraycols package introduces new predefined column types for tables and also includes a command for wide horizontal rule drawing. Below is a summary of the column types and macro defined by arraycols, which will be detailed in the following section.

Column definitions	
`L`	Left adjusted column (applicable in LR mode for `array` environments or math mode for `tabular` environment)
`C`	Centered-adjusted column (similar to L but centered)
`R`	Right-adjusted column (similar to L but right-adjusted)
`t{`⟨*width*⟩`}`	Text column of fixed ⟨*width*⟩ (LR mode), similar to `p`, but with horizontal and vertical centering
`x`	Centered column in math mode with adjusted height to avoid touching the horizontal rules
`y`	Left-aligned column in math mode with adjusted height
`z{`⟨*width*⟩`}`	Centered column in math mode, similar to `x`, with adjusted height, but with fixed ⟨*width*⟩
`T`	Centered text column with adjusted width for `tabularx` environments (calculated like `X` column)
`Z`	Centered column for `tabularx`, similar to `T`, but in math mode with adjusted height, like `x` and `z`
`I`	Thick vertical rule (1 pt)
`V{`⟨*thickness*⟩`}`	Vertical rule with variable ⟨*thickness*⟩

Horizontal rules	
`\whline`	Wide horizontal rule (1 pt)

---

[*] This document corresponds to arraycols v1.5, dated 2024/05/04. Thanks to François Bastouil for initial assistance with the English translation.

If a column type has been previously defined by another package, using arraycols will overwrite it and display a warning message.

In addition to loading the array package, arraycols also requires cellspace [3], which is necessary for the `x`, `y`, `z` and `Z` column types. Moreover it relies on tabularx [4] for `T` and `Z` column types and loads makecell [5] for creation of multilined tabular cells. Note that the tablestyles package [6] also defines the column types L, C, R and Z, but differently. However, tablestyles is incompatible with makecell and therefore also with arraycols as well.

2 Usage

`L`
`C`
`R`
Referring to an example from the array package documentation, the `L`, `C` and `R` columns types, enable the reversal of the mathematical mode. This allows to achieve centered, left-aligned or right-aligned LR-mode in an `array` environment or an equivalent math-mode in a `tabular` environment. For instance, using the declaration `\begin{tabular}{|l|C|r|}` sets the second column in centered mathematical mode. Similarly, using the declaration `\begin{array}{|L|c|c|}` sets the first column in text mode, left-aligned[1].

`t{⟨width⟩}` The newly introduced column type definition `t{`⟨*width*⟩`}` horizontally and vertically centers paragraphs within the column, with a specified ⟨*width*⟩. In contrast, the traditional `p{`⟨*width*⟩`}` (in standard LaTeX) and `m{`⟨*width*⟩`}` (from the array package) column types, justifies paragraphs, while text in `t{`⟨*width*⟩`}` is centered.

`x`
`y`
In order to guarantee adequate row heights, especially for displaymath mode formulas, the package includes the column types `x` (centered) and `y` (left aligned). These column types activate the mathematical mode and allow automatic adjustment of row heights to prevent any overlap with horizontal rules in cases where the content is too tall, thanks to a functionality of the cellspace package by Josselin Noirel [3]. While cellspace is initially designed for `tabular` environments, the new `x` and `y` column types are applicable in both `tabular` and `array` environments. Examine the following examples created using `\begin{array}{|c|}` and `\begin{array}{|x|}`.

bad
$\displaystyle\lim_{\substack{x\to1\\x>1}} \ln\left(\frac{x^2}{x-1}\right)$
$\displaystyle\frac{a}{b}$
$\displaystyle\int_1^X \frac{1}{t}\,\mathrm{d}t$

good
$\displaystyle\lim_{\substack{x\to1\\x>1}} \ln\left(\frac{x^2}{x-1}\right)$
$\displaystyle\frac{a}{b}$
$\displaystyle\int_1^X \frac{1}{t}\,\mathrm{d}t$

---

[1] The declarations `L, C, R` do not work in a `tabularx` environment. Additionally, the tabulary package by David Carlisle [7] already defines the `L, C, R` (and `J`) column types for specific alignments in tables of the same type as `tabularx`. However, there is no incompatibility with arraycols because these column definitions apply exclusively within `tabulary` environments.

The cellspace package is loaded with the `math` option[2] to efficiently manage row heights, including in matrices. Another option of cellspace, `column=Q` (with `S` being the default in cellspace)[3], was necessary to prevent any compatibility issues with the siunitx package (also loaded by pstricks-add). The `Q` declaration serves as a "modifier" that, when placed before a column type declaration, permits the adjustment of cell height, for instance "`Qc`" for a vertical adjustment within a centered column type.

Notice that another package, booktabs [8], also offers excellent row height adjustment. However, regrettably, it doesn't handle the height of vertical separators "`|`". In order to achieve a similar vertical adjustment as booktabs, we set the cellspace parameters as follows:

`\setlength{\cellspacetoplimit}{3pt}`,
`\setlength{\cellspacebottomlimit}{2pt}`.

A common issue with LaTeX tables is that there isn't enough space around horizontal rules. As seen previously, cellspace partially addresses this issue, but if you want to add some more space around the horizontal rules, it's not straightforward. First, note that Donald Arsenau's tabls package [9] produces a nice and automatic solution in this regard, but is not compatible with array nor with numprint.

Several other methods can be employed: you can increase the space on top or bottom of a particular cell by using

`\gape[`⟨`t` *or* `b`⟩`]{`⟨*text*⟩`}` or `\Gape[`⟨*height*⟩`][`⟨*depth*⟩`]{`⟨*text*⟩`}`

from the makecell package [5]. You have also the `\bigstrut` command from the bigstrut package [10], but it's less efficient and convenient. An efficient method is provided by the mdwtab package of Mark Wooding [11] with its macros `\vgap{`⟨*length*⟩`}` or `\hlx{`⟨*hlx-cmd*⟩`}`, where in `{`⟨*hlx-cmd*⟩`}` you can place `h`, representing `\hline`, and `s[`⟨*length*⟩`]`, meaning `\vgap` (among others). This package provides also many other interesting features. Finally, manual adjustments of particular rows can be achieved using the `\vstrut[`⟨*depth*⟩`]{`⟨*height*⟩`}` command from the spacingtricks package [12], These packages are not loaded by arraycols, except makecell. Have a look at their documentation.

`z{⟨width⟩}` The `z{`⟨*width*⟩`}` column type activates the mathematical mode and allows to define the column width, similar to `t{`⟨*width*⟩`}`. It also adjusts the row height, akin to the `x` column type. The content consist of a single line. When it becomes too wide, it may protrude to the right.

`T`
`Z` The tabularx package by David Carlisle [4] introduces the `X` column definition, which calculates its width in relation to the required width for the entire table. It aligns text to the left similar to `p{`⟨*width*⟩`}`. Using `\begin{tabularx}{8cm}{|c|X|X|}` adjusts the width of the `X` columns to achieve a total width of 8 cm. To complement

---

[2] The `math` option loads the amsmath package. As mentionned in the cellspace package documentation: "the amsmath package can be loaded beforehand with other packages (such as `empheq` or `mathtools`), were an incompatibility to arise from one's loading it later".

[3] The letter Q is a substitute for the default column modifier S of the cellspace package.

this, we offer the `T` declaration, which performs a similar function but centers the content horizontally. Additionally the `Z` declaration activates mathematical mode and adjusts line heights, comparable to `x` or `z`). The following example is obtained with

`\begin{tabularx}{\linewidth}{|T|y|x|Z|T|}`.

A good job	$\lim_{\substack{x\to 1\\ x>1}} \ln\left(\frac{x^2}{x-1}\right)$	$\frac{a}{b}$	$\frac{a}{b}+\int_1^X \frac{1}{t}\,\mathrm{d}t$	a multi-line piece of text

Observe that cells 3 and 4 are not vertically centered to preserve the precise alignment of fraction bars within mathematical formulas across cells. For achieving accurate vertical positioning within the last cell, we have used the powerful `\makecell[`⟨*pos*⟩`]{`⟨*content*⟩`}` command from the makecell package by Olga Lapko [5]: `\makecell{a multi-line \\ piece of text}`.

`I`
`V{⟨thickness⟩}`

The column definition `I` is mentioned in The LaTeX Companion [13] and allows for drawing a thicker *vertical* line (1 pt thick) compared to the one achieved with the standard declaration "|". For selecting the line thickness, we additionally provide the column definition `V{`⟨*thickness*⟩`}`[4].

`\whline`

Similarly, the `\whline` command, suggested in The LaTeX Companion, enables the drawing of a thicker *horizontal* line (1 pt thick) compared to the line obtained with `\hline`. Moreover, the makecell package provides the command `\Xhline{`⟨*thickness*⟩`}` enabling the choice of horizontal rule thickness.

The introductory table has been typeset with a column declaration `I` serving as a separator between the two text columns. Horizontal rules at the beginning and end of the table are accomplished using `\whline`, while a `\Xhline{0.8pt}` rule is employed after the legend rows. The formatting of header rows is achieved using the `\thead` command from the makecell package. Lastly, following a recommendation of the array package [2], an additional 1 pt has been added to the standard height of each row within this table. This adjustment is implemented with the command `\setlength{\extrarowheight}{1pt}`[5].

3 Implementation

```
\RequirePackage{array}
\RequirePackage[math,column=Q]{cellspace}
\RequirePackage{tabularx} % must be loaded after cellspace
\RequirePackage{makecell}

\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
```

---

[4] The definition of `V` would have been simplified by utilizing an optional argument for `I`, but unfortunately, this approach doesn't function.

[5] As stated in the array package documentation: "This is important for tables with horizontal lines because those lines normally touch the capital letters".

```
\newcolumntype{R}{>{$}r<{$}}
\newcolumntype{t}[1]{>{\centering\arraybackslash}m{#1}}
```

The `cellspace` package provides the `S` modifier (we used `Q` instead), which, when placed before a column declaration, allows for the adjustment of cell content height to prevent any overlap with horizontal rules. The spacing between the content and the horizontal rules is governed by the parameters `\cellspacetoplimit` and `\cellspacebottomlimit`.

```
\newcolumntype{x}{>{$}Qc<{$}}
\newcolumntype{y}{>{$}Ql<{$}}
\setlength{\cellspacetoplimit}{3pt}
\setlength{\cellspacebottomlimit}{2pt}
\newcolumntype{z}[1]{>{$}Q{>{\centering\arraybackslash}p{#1}}<{$}}
```

For the `z` column type, we employed the `p` declaration instead of `m` (which should automatically center content). This choice ensures proper alignment of mathematical expressions within cells of the same row. The same result can be achieved with the following definition: `\newcolumntype{z}[1]{>{$}Q{W{c}{#1}}<{$}}` with `W{c}` defined in the `array` package.

```
\newcolumntype{T}{>{\centering\arraybackslash}X}
\newcolumntype{Z}{>{$}QT<{$}}
```

Like `X`, the `T` columns are not vertically centered. Although it's possible to achieve this by using the command `\renewcommand{\tabularxcolumn}[1]{m{#1}}` (with `m` instead of default value `p`), unfortunately, this approach has a global effect on all column declarations based on `X`, including `T` and `Z`. As a result, it could disrupt the alignment of mathematical expressions within cells of the same row.

```
\newcolumntype{I}{!{\vrule width 1pt}}
\newcolumntype{V}[1]{!{\vrule width #1}}
\newlength\savedwidth
\newcommand{\whline}{%
    \noalign{\global\savedwidth\arrayrulewidth
        \global\arrayrulewidth 1pt}
    \hline
    \noalign{\global\arrayrulewidth\savedwidth}
}
```

References

[1] *Tabularray – Typeset Tabulars and Arrays with LaTeX3*, Jianrui Lyu, CTAN, 2024A 2024/02/16.

[2] *A new implementation of LATEX's tabular and array environment*, Frank Mittelbach, David Carlisle, CTAN, v2.4k revised 2018/12/30.

[3] *The cellspace package*, Josselin Noirel, CTAN, v1.8.1 2019/03/11.

[4] *The tabularx package*, David Carlisle, CTAN, v2.11.b 2016/02/03.

[5] *The makecell package*, Olga Lapko, CTAN, v0.1e 2009/08/03.

[6] *The tablestyles package*, Matthias Pospiech, CTAN, v0.1 2014/06/27.

[7] *The tabulary package*, David Carlisle, CTAN, v1.10 2014/06/11.

[8] *Publication quality tables in LaTeX*, package booktabs by Simon Fear, CTAN, v1.618033 2016/04/29.

[9] *The tabls package*, Donald Arseneau, CTAN, v3.5 2010/02/26.

[10] *The multirow, bigstrut and bigdelim packages*, Piet van Oostrum, Øystein Bache, Jerry Leichter, CTAN, v2.4 2019/01/01.

[11] *The mdwtab package*, Mark Wooding, CTAN, v1.9 1998/04/28.

[12] *The spacingtricks package*, Antoine Missier, CTAN, v1.8 2023/12/06.

[13] *The LaTeX Companion*. Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, 2nd edition, Pearson Education, 2004.