lutabulartools

some useful tabular tools (LuaLaTeX-based)

Kale Ewasiuk (kalekje@gmail.com)

2022–02–27

lutabulartools is a package that contains a few useful macros to help with tables. Most functions require LuaLaTeX. This package redefines the `tabular` and `tabular*` environments. The following packages are loaded: `booktabs, multirow, makecell, xparse, array, xcolor, colortbl, luacode, penlight,`

1 Options

The author likes tables with left and right-most columns flush to the end. Since the `tabular` env is redefined in this, the author took the liberty to automatically pad the `tabular` cell spec with `@{}` on each end. If you don't want this, you can pass `notrim` to the package. Or, you can manually fix it in a `tabular` with `!{}` like so:

```
1 \begin{tabular}{!{}c!{}}\toprule
2 Xyz\\\bottomrule    \end{tabular}
```

Xyz

2 \MC – Magic Cell

`\MC` (magic cell) combines the facilities of `\multirow` and `\multicolumn` from the `multirow` package, and `\makcell` from the titular package. With the help of LuaLaTeX, it takes an easy-to-use cell specification and employs said commands as required. `\MC` will not work properly if your table is only 1 column wide (you probably don't need `MC` in that case anyway...). Here is the usage:

`\MC * [cell spec] <cell format> (override multicolumn col) {contents}`

`*` This will wrap the entire command in {}. This is necessary for `siunitx` single-column width columns. However, the `\MC` command attempts to detect this automatically.

`[cell spec]` Any letters placed in this argument are used for cell alignment. You can use one of three: "`t`", "`m`", "`b`" for top, middle, bottom (vertical alignment), or "`l`", "`c`", "`r`" for

horizontal alignment. By default, `\MC` will try to autodetect the horizontal alignment based on the current column. If it can't, it will be left-aligned. The default vertical alginment is top. More on this in section **??**.

This argument can also contain two integers, separated by a comma (if two are used). "`C,R`", "`C`", or "`,R`" are a valid inputs, where `R`=rows (int), and `C`=columns, (int). If you want a 1 column wide, multirow cell, you can pass "`,R`". These numbers can be negative (positive numbers occupy columns to the right and rows below, and negative numbers occupy columns to the left and rows above). If no spec is passed, (argument empty), `\MC` acts like a `makecell`. Additionally, you can pass "`+`" in place of C (number of columns wide), and it will make the cell width fill until the end of the current row.

Examples:
"`\MC [2,2]`" means two columns wide, two rows tall.
"`\MC [2,1]`" or "`\MC [2]`" means two columns wide, one row tall.
"`\MC [1,2]`" or "`\MC [,2]`" means one column wide, two rows tall.
"`\MC [+,2]`", if placed in the first column, occupies the entire row and is two rows tall.
"`\MC [+,2]`", if placed in the second column, occupies the second column to the end of the table and is two rows tall.
In any of these examples, you can place the alignment letters anywhere. So, "`\MC [l1,2b]`" and "`\MC [1,2 lb]`" are both left-bottom aligned (spaces are ignored).

`(override multicolumn)` You may want to adjust the column specification of a multicolumn cell, for example, using `(@{}c@{})` to remove padding between the cell.

`<cell format>` You can place formatting like `\bfseries` here.

2.1 Defaults

The `tabular[*]` environment is re-defined to use Lua pattern matching to parse the column specification of the table. This is done to determine how many columns there are, and what the current column type is, even if specifications like `r@{.}l*{3}{r}` are used. If you have defined a column that expands many, you should register it with `\setMCrepl {?}{??}` where `?` is your column and `??` is what it expands to. You can also specify default horizontal and vertical alignments (ie if alignment not passed to `\MC`) for an arbitrary column by `\setMChordef {?}{l|r|c}` and
`\setMChordef {?}{t|m|b}`, where `?` is the column. To add a column that should be surrounded by brackets for `siunitx` purposes, do so with `\addMCsicol {?}`. S is included by default.

2.2 Examples

2.2.1 A good use for headers

```
\begin{tabular}{ l l l }\toprule
\MC[+m]<\itshape>{A Decent
                  Example}\\\midrule
    & \MC[2m]{Heading} \\\cmidrule{2-}
\MC[b,-2]{Multi\\Line}  & A & B   \\\midrule
\end{tabular}
```

A Decent Example		
Multi	Heading	
Line	A	B

2.2.2 A small example

```
\begin{tabular}{ l l l }\midrule
 A & \MC[mc2,2]{Lttrs}    \\
 B &                       \\
 1 & A & B   \\
\end{tabular}
```

A	Lttrs	
B		
1	A	B

2.2.3 A small bad example

Notice that the top-aligned p-column doesn't play particularly well with the middle aligned `\MC`

```
\begin{tabular}{ p{1cm} l }\toprule
   hello\newline world
        & \MC[mr]{11\\2} \\
\end{tabular}
```

hello world	11 2

2.2.4 If you insist on vertical lines

```
\begin{tabular}{|c|c|c|} \hline
1 & 2 & 3\\\hline
4 & \MC[2,2cm](@{}c@{}|)%
    <\ttfamily>{5}\\\cline{1-1}
  & \MC[2](r|){} \\\hline%hacky fix
6 & 7 & 8\\\hline
\end{tabular}
```

1	2	3
4	5	
6	7	8

2.2.5 A perhaps useful example

```
\begin{tabularx}{\linewidth}{S[table-format=2.1,table-alignment=left]X}
% err & ... \\% ERROR, not wrap
  \MC{Error,\%} & Comment \\% MC helps
 3.1 & \MC[,2]{multi-line\\comment}\\
 10.1& \\
 \MC[2c]{... ...} \\
\end{tabularx}
```

Error,%	Comment
3.1	multi-line
10.1	comment
	... ...

2.2.6 A messy example

```
\begin{tabular}{| c | c | c | c | c |c |}\toprule
 \MC[2,2cm]<\ttfamily>{2,2cm}   & \MC[2r]<\ttfamily>{2r} & 5 & \MC[,2b]<\ttfamily>{,2b}\\
   &   & 3 & 4 & 5 & \\\midrule
 1 & 2 & \MC[2l](@{}l)<\ttfamily>{2l (\@\{\}l)} & 5 & 6666\\\cmidrule{3-4}
 1 & \MC[+r]<\ttfamily>{+r}  \\
  \\
 1 & 2 & 3 & 4 & 5 & \MC[,-2]<\ttfamily>{,\\-2}\\
\end{tabular}
```

2,2cm		2r		5	,2b
		3	4	5	
1	2	2l (@{}l)		5	6666
1					+r
					,
1	2	3	4	5	-2

3 Some additional rules

This package also redefines the `booktabs` midrules.

`\gmidrule` is a full gray midrule.

By taking advantage of knowing how many columns there are (if you chose to redefine `tabular`), you can specify individual column numbers (for a one column wide rule), or reference with respect to the last column (blank, `+1`, `+0`, or `+` means last column, `+2` means second last column, for example) or omit the last number.

`\cmidrule` is a single partial rule, with the above features

`\gcmidrule` is a single partial gray rule, with the above features

You can add multiple "`cmidrule`"'s with the `(g)cmidrules` command. Separate with a comma. You can apply global trimming of the rules with the "`()`" optional argument, and then override it for a specific rule by placing "`r`" or "`l`" with the span specification.

`\gcmidrules` Can produce multiple, light gray partial rules

`\cmidrules` Can produce multiple black partial rules.

Here's an example:

```
\begin{tabular}{c c c c c c}
 1 & 2 & 3 & 4 & 5 & 6\\
    \cmidrule{+1}  % rule on last column
 1 & 2 & 3 & 4 & 5 & 6\\
    \cmidrules{1,3-+3,+} % rule on first col, third to third last col, and last col
 1 & 2 & 3 & 4 & 5 & 6\\
    \cmidrules{1,3-+3rl,+} % same as above, but trim middle
 1 & 2 & 3 & 4 & 5 & 6\\
    \cmidrules(l){1,r3-+3,+1}% trim left for all, but only trim right for middle rule
 1 & 2 & 3 & 4 & 5 & 6\\
    \gcmidrule{+1}  % rule on last column
 1 & 2 & 3 & 4 & 5 & 6\\
    \gcmidrules{1,3-+3,+} % rule on first col, third to third last col, and last col
 1 & 2 & 3 & 4 & 5 & 6\\
    \gcmidrules{1,3-+3rl,+} % same as above, but trim middle
 1 & 2 & 3 & 4 & 5 & 6\\
    \gcmidrules(l){1,r3-+3,+1}% trim left for all, but only trim right for middle rule
\end{tabular}
```

1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

3.1 Midrule every Xth row

\midruleX With this command, you can place a rule every X rows. You can change the step size and what kind of midrule you prefer.

```
\def\midruleXstep{5}
\def\midruleXrule{\gmidrule}
```

To use, insert `@{\midrulex }` at the end of each row using the tabular column spec. Before you want counting to begin, you should apply `\resetmidruleX` to avoid header rows being counted. Use `\noalign {\resetmiduleX }` if you need place a rule on the same line or in a cell.

```
\def\midruleXstep{3}
\def\midruleXrule{\cmidrules{1,3-4r}}
\begin{tabular}{rlc@{\midruleX}}
                % ^^^ inject midrule
\toprule
Num  & . & . & .  \\
\midrule\resetmidruleX  % reset
1    & & &  \\
2    & & &  \\
3    & & &  \\
4    & & &  \\
5    & & &  \\
6    & & &  \\
7    & & &  \\
8    & & &  \\
9    & & &  \\
10   & & &  \\
11   & & &  \\
\resetmidruleX % no bottom rule
12   & & &  \\
\bottomrule
\end{tabular}
```

Num	.	.	.
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			