lutabulartools

some useful tabular tools (LuaLaTeX-based)

Kale Ewasiuk (kalekje@gmail.com)

2021–11–07

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

1 \MC – Magic Cell

`\MC` (magic cell) combines the facilities of `\multirow` and `\multicolumn` from the `multirow` package, and `\makecell` from the titular package. With the help of LuaLaTeX, it takes an easy-to-use cell specification and employs said commands as required. 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.

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. 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]`" or means two columns wide, one row tall.

"\MC [1,2]" or "\MC [,2]" means one column wide, two rows tall.
In any of these examples, you can place the alignment letters anywhere.

(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.

Here's an 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	
		3	4	5	,2b
1	2	2l ({}l)		5	6666
1					+r
					,
1	2	3	4	5	-2

1.1 Notes

This package redefines the tabular and tabular* environments. It uses Lua pattern matching to parse the column specification of the table to know how many columns there are, and what the current column type is. If you have defined a column that creates many, it will not work. This will be worked out in later package revisions.

2 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}\toprule
 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
\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

2.1 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}
```

Usage: Insert midrulex at the end of each row using the column spec. Before you want counting to begin, you should apply resetmidruleX (also to avoid header rows).

```
\def\midruleXstep{4}
\def\midruleXrule{\cmidrules{1,3-4}}
\begin{tabular}{rclc@{\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			