The \texttt{colortbl} package*

David Carlisle

2022/06/20

Abstract

This package implements a flexible mechanism for giving coloured ‘panels’ behind specified columns in a table. This package requires the \texttt{array} and \texttt{color} packages.

1 Introduction

This package is for colouring tables (i.e., giving coloured panels behind column entries). In that it has many similarities with Timothy Van Zandt’s \texttt{colortab} package. The internal implementation is quite different though, also \texttt{colortab} works with the table constructs of other formats besides \LaTeX. This package requires \LaTeX (and its \texttt{color} and \texttt{array} packages).

First, a standard \texttt{tabular}, for comparison.

\begin{tabular}{|l|c|}
\hline
one&two\\
three&four\\
\hline
\end{tabular}

2 The \texttt{\textbackslash colormcolor} command

The examples below demonstrate various possibilities of the \texttt{\textbackslash columncolor} command introduced by this package. The vertical rules specified by \texttt{|} are kept in all the examples, to make the column positioning clearer, although possibly you would not want coloured panels and vertical rules in practice.

The package supplies a \texttt{\textbackslash columncolor} command, that should (only) be used in the argument of a \texttt{>} column specifier, to add a coloured panel behind the specified column. It can be used in the main ‘preamble’ argument of \texttt{array} or \texttt{tabular}, and also in \texttt{\textbackslash multicolumn} specifiers.

The basic format is:

\texttt{\textbackslash columncolor[\{color model\}][\{colour\}][\{left overhang\}][\{right overhang\}]}  

The first argument (or first two if the optional argument is used) are standard \texttt{color} package arguments, as used by \texttt{\textbackslash color}.

The last two arguments control how far the panel overlaps past the widest entry in the column. If the \texttt{right overhang} argument is omitted then it defaults to \texttt{left overhang}. If they are both omitted they default to \texttt{\textbackslash tabcolsep} (in \texttt{tabular}) or \texttt{\textbackslash arraycolsep} (in \texttt{array}).

If the overhangs are both set to \texttt{\textup{Opt}} then the effect is:

\(*\text{This file has version number v1.0f, last revised 2022/06/20.}\)
The default overhang of \tabcolsep produces:

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
</table>

You might want something between these two extremes. A value of .5\tabcolsep produces the following effect:

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
</table>

This package should work with most other packages that are compatible with the array package syntax. In particular it works with longtable and dcolumn as the following example shows.

Before starting give a little space: \setlength\minrowclearance{2pt}

<table>
<thead>
<tr>
<th>A long table example</th>
</tr>
</thead>
<tbody>
<tr>
<td>First two columns</td>
</tr>
<tr>
<td>P-column</td>
</tr>
<tr>
<td>p-type</td>
</tr>
<tr>
<td>and another one</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>Some long text</td>
</tr>
<tr>
<td>in the first column</td>
</tr>
<tr>
<td>aaa</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>aaa</td>
</tr>
<tr>
<td>Note that the</td>
</tr>
<tr>
<td>coloured rules in</td>
</tr>
<tr>
<td>all columns</td>
</tr>
<tr>
<td>stretch to</td>
</tr>
<tr>
<td>accommodate large</td>
</tr>
<tr>
<td>entries in one</td>
</tr>
<tr>
<td>column.</td>
</tr>
<tr>
<td>aaa</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Continued...
A long table example (continued)

<table>
<thead>
<tr>
<th>First two columns</th>
<th>Third column</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-type</td>
<td>D-type (dcolumn)</td>
</tr>
<tr>
<td>aaa</td>
<td>12.4</td>
</tr>
<tr>
<td>bbb</td>
<td>45.3</td>
</tr>
</tbody>
</table>

Depending on your driver you may get unsightly gaps or lines where the “screens” used to produce different shapes interact badly. You may want to cause adjacent panels of the same colour by specifying a larger overhang or by adding some negative space (in a \noalign between rows.

This example shows rather poor taste but is quite colourful! Inspect the source file, colortbl.dtx, to see the full code for the example, but it uses the following column types.

\newcolumntype{A}{%\color{white}\columncolor{red}[.5\tabcolsep] \raggedright}p{2cm}
\newcolumntype{B}{%\columncolor{blue}[.5\tabcolsep]\color{yellow}\raggedright}p{3cm}
\newcolumntype{C}{%\columncolor{yellow}[.5\tabcolsep]D{.}{\cdot}{3.3}}
\newcolumntype{E}{%\large\bfseries\columncolor{cyan}[.5\tabcolsep]}c
\newcolumntype{F}{%\color{white}\columncolor{magenta}[.5\tabcolsep]}c
\newcolumntype{G}{%\columncolor[gray]{0.8}[.5\tabcolsep]\tabcolsep}l
\newcolumntype{H}{%\columncolor[gray]{0.8}[\tabcolsep]l}
\newcolumntype{I}{%\columncolor[gray]{0.8}[\tabcolsep][.5\tabcolsep]}D{.}{\cdot}{3.3}
3 Using the ‘overhang’ arguments for \texttt{tabular*}

The above is all very well for \texttt{tabular}, but what about \texttt{tabular*}? Here the problem is rather harder. Although \TeX’s \texttt{leader} mechanism which is used by this package to insert the ‘stretchy’ coloured panels is rather like \texttt{glue}, the \texttt{tabskip} glue that is inserted between columns of \texttt{tabular*} (and \texttt{longtable} for that matter) has to be ‘real glue’ and not ‘leaders’.

Within limits the overhang options may be used here. Consider the first table example above. If we use \texttt{tabular*} set to 3 cm with a preamble setting of

\begin{verbatim}
\begin{tabular*}{3cm}{% \\
  @{\extracolsep{\fill}} \\
  >{\columncolor[gray]{.8}[0pt][20mm]}l \\
  >{\columncolor[gray]{.8}[5mm][0pt]}l \\
  @{}}
\end{tabular*}
\end{verbatim}

Changing the specified width to 4 cm works, but don’t push your luck to 5 cm...

\begin{verbatim}
\begin{tabular}{|l|c|}
\rowcolor[gray]{.9} one & two \\
\rowcolor[gray]{.5} three & four
\end{tabular}
\end{verbatim}

4 The \texttt{\rowcolor} command

As demonstrated above, one may change the colour of specified rows of a table by the use of \texttt{\multicolumn} commands in each entry of the row. However if your table is to be marked principally by rows, you may find this rather inconvenient. For this reason a new mechanism, \texttt{\rowcolor}, has been introduced\(^1\).

\texttt{\rowcolor} takes the same argument forms as \texttt{\columncolor}. It must be used at the \textit{start} of a row. If the optional overhang arguments are not used the overhangs will default to the overhangs specified in any \texttt{\columncolor} commands for that column, or \texttt{\tabcolsep} (\texttt{\arraycolsep} in \texttt{array}).

If a table entry is in the scope of a \texttt{\columncolor} specified in the table preamble, and also a \texttt{\rowcolor} at the start of the current row, the colour specified by \texttt{\rowcolor} will take effect. A \texttt{\multicolumn} command may contain >\texttt{\rowcolor...} which will override the default colours for both the current row and column.

\begin{verbatim}
\begin{tabular}{|l|c|}
\rowcolor[gray]{.9} one & two \\
one&two\ \\
\rowcolor[gray]{.5} three & four
\end{tabular}
\end{verbatim}

5 The \texttt{\rowcolors} command

The \texttt{\rowcolors} command and its documentation originate in the \texttt{xcolor} package by Dr. Uwe Kern.

\begin{verbatim}
\rowcolors\(\langle\text{commands}\rangle\)\{\langle\text{row}\rangle\}\{\langle\text{odd-row color}\rangle\}\{\langle\text{even-row color}\rangle\}
\rowcolors*\(\langle\text{commands}\rangle\)\{\langle\text{row}\rangle\}\{\langle\text{odd-row color}\rangle\}\{\langle\text{even-row color}\rangle\}
\end{verbatim}

\(^1\)At some cost to the internal complexity of this package
One of these commands has to be executed before a table starts. \texttt{\textlangle row\textrangle} tells the number of the first row which should be colored according to the \texttt{\langle odd-row color\rangle} and \texttt{\langle even-row color\rangle} scheme. Each of the color arguments may also be left empty (= no color). In the starred version, \texttt{\langle commands\rangle} are ignored in rows with inactive \texttt{rowcolors status} (see below), whereas in the non-starred version, \texttt{\langle commands\rangle} are applied to every row of the table. Such optional commands may be \texttt{\textbackslash hline} or \texttt{\textbackslash noalign}({\textlangle stuff\textrangle}).

\texttt{\textbackslash showrowcolors} \texttt{\textbackslash hiderowcolors} \texttt{\rownum} \texttt{\rowcolors\textbackslash hline\{3\}\{green\}\{yellow\} \arrayrulecolor\{red\} \begin\{tabular\}\{ll\}
\texttt{test} & \texttt{row \therownum} \\
\texttt{test} & \texttt{row \therownum} \\
\texttt{test} & \texttt{row \therownum} \\
\texttt{\arrayrulecolor\{black\}} \\
\texttt{test} & \texttt{row \therownum} \\
\texttt{\rowcolor\{blue\}} \\
\texttt{test} & \texttt{row \therownum} \\
\texttt{\hiderowcolors} \\
\texttt{test} & \texttt{row \therownum} \\
\texttt{\showrowcolors} \\
\texttt{\multicolumn\{1\}\{>{\textbackslash colormode\{red\}}l\}{test} & \texttt{row \therownum} \\
\end\{tabular\}

6 The \texttt{\textbackslash cellcolor} command

A background colour can be applied to a single cell of a table by beginning it with \texttt{\multicolumn\{1\}\{>{\textbackslash rowcolor…} (or \texttt{\textbackslash colormode} if no row-colour is in effect) but this has some deficiencies: 1) It prevents data within the cell from triggering the colouration; 2) The alignment specification must be copied from the top of the tabular, which is prone to errors, especially for \texttt{p{\textbackslash column}} columns; 3) \texttt{\multicolumn\{1\}} is just silly. Therefore, there is the \texttt{\textbackslash cellcolor} command, which works like \texttt{\textbackslash colormode} and \texttt{\textbackslash rowcolor}, but over-rides both of them; \texttt{\cellcolor} can be placed anywhere in the tabular cell to which it applies.

7 Colouring rules.

So you want coloured rules as well?
One could do vertical rules without any special commands, just use something like !{\color{green}\vline} where you’d normally use |. The space between || will normally be left white. If you want to colour that as well, either increase the overhang of the previous column (to \tabcolsep + \arrayrulewidth + \doublerulesep) Or remove the inter rule glue, and replace by a coloured rule of the required thickness. So

%!{\color{green}\vline}
@!{\color{yellow}\vrule width \doublerulesep}
%!{\color{green}\vline}

Should give the same spacing as || but more colour.

However colouring \hline and \cline is a bit more tricky, so extra commands are provided (which then apply to vertical rules as well).

8 \textbackslash arrayrulecolor

\textbackslash arrayrulecolor takes the same arguments as \color, and is a global declaration which affects all following horizontal and vertical rules in tables. It may be given outside any table, or at the start of a row, or in a > specification in a table preamble. You should note however that if given mid-table it only affects rules that are specified after this point, any vertical rules specified in the preamble will keep their original colours.

9 \textbackslash doublerulesepscolor

Having coloured your rules, you’ll probably want something other than white to go in the gaps made by || or \hline\hline. \textbackslash doublerulesepscolor works just the same way as \textbackslash arrayrulecolor. The main thing to note that if this command is used, then \textbackslash longtable will not ‘discard’ the space between \hline\hline at a page break. \TeX{} has a built-in ability to discard space, but the coloured ‘space’ which is used once \textbackslash doublerulesep is in effect is really a third rule of a different colour to the two outer rules, and rules are rather harder to discard.\)

\begin{tabular}{||l||c||}
\hline\hline
one & two \\
three & four \\
\hline\hline
\end{tabular}

10 More fun with \textbackslash hhline

The above commands work with \hhline from the hhline package, however if hhline is loaded in addition to this package, a new possibility is added. You may use >{}\ldots{} to add declarations that apply to the following - or = column rule. In particular you may give \textbackslash arrayrulecolor and \textbackslash doublerulesepscolor declarations in this argument.
Most manuals of style warn against over use of rules in tables. I hate to think what they would make of the following rainbow example:

![Rainbow Table]

\newcommand\rainbowline[1]{{% 
\arrayrulecolor{red} \doublerulesepcolor[rgb]{.3,.3,1} \% #1:=% 
\arrayrulecolor{orange} \doublerulesepcolor[rgb]{.4,.4,1} \% =% 
\arrayrulecolor{yellow} \doublerulesepcolor[rgb]{.5,.5,1} \% =% 
\arrayrulecolor{green} \doublerulesepcolor[rgb]{.6,.6,1} \% =% 
\arrayrulecolor{blue} \doublerulesepcolor[rgb]{.7,.7,1} \% =% 
\arrayrulecolor{indigo} \doublerulesepcolor[rgb]{.8,.8,1} \% =% 
\arrayrulecolor{violet} \doublerulesepcolor[rgb]{.9,.9,1} \% =:#1|% 
}} 
\arrayrulecolor{red} \doublerulesepcolor[rgb]{.3,.3,1} \begin{tabular}{||*7{>{\columncolor[gray]{.9}}c}||} 
\rainbowline{t} 
Richard & of & York & gave & battle & in & vain \multicolumn{1}{>{\columncolor[gray]{.9}}c||}{vain} 
\rainbowline{} 
1 & 2 & 3 & 4 & 5 & 6 & 7 \multicolumn{1}{>{\columncolor[gray]{.9}}c||}{} 
\rainbowline{b} 
\end{tabular} 

11 Less fun with \cline

Lines produced by \cline are coloured if you use \arrayrulecolor but you may not notice as they are covered up by any colour panels in the following row. This is a ‘feature’ of \cline. If using this package you would probably better using the - rule type in a \hhline argument, rather than \cline.
12 The \minrowclearance command

As this package has to box and measure every entry to figure out how wide to make the rules, I thought I may as well add the following feature. ‘Large’ entries in tables may touch a preceding \hline or the top of a colour panel defined by this style. It is best to increase \extrarowsep or \arraystretch sufficiently to ensure this doesn’t happen, as that will keep the line spacing in the table regular. Sometimes however, you just want to \LaTeX to insert a bit of extra space above a large entry. You can set the length \minrowclearance to a small value. (The height of a capital letter plus this value should not be greater than the normal height of table rows, else a very uneven table spacing will result.)

Donald Arseneau’s \texttt{tabls} packages provides a similar \texttt{tablinesep}. I was going to give this the same name for compatibility with \texttt{tabls}, but that is implemented quite differently and probably has different behaviour. So I’ll keep a new name for now.

13 The Code

\begin{verbatim}
1 \langle*package\rangle
2 \edef\@tempa{%
3 \noexpand\AtEndOfPackage{%
4 \catcode'\noexpand\^^A\the\catcode'\^^A\relax}}
5 \@tempa
6 \catcode'\^^A=\catcode'%
7 \DeclareOption{debugshow}{\catcode'\^^A=9 }

All the other options are handled by the color package.
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{color}}
9 \ProcessOptions

I need these so load them now. Actually Mark Wooding’s \texttt{mdwtab} package could probably work instead of \texttt{array}, but currently I assume \texttt{array} package internals so.
10 \RequirePackage{array,color}
11 \@classz
\end{verbatim}

\texttt{@classz} is the main function in the \texttt{array} package handling of primitive column types: It inserts the code for each of the column specifiers, ‘\texttt{c}l\texttt{r}p\texttt{m}b’. The other classes deal with the other preamble tokens such as ‘\texttt{G}’ or ‘\texttt{>}’.

\begin{verbatim}
12 \def\@classsz{\@classx
13 \@tempcnta \count@
14 \prepnext@tok
\end{verbatim}

At this point the colour specification for the background panel will be in the code for the ‘\texttt{>}’ specification of this column. This is saved in \texttt{\toks@temptokena} but \texttt{array} will insert it too late (well it would work for \texttt{c}, but not for \texttt{p}) so fish the colour stuff out of that token register by hand, and then insert it around the entry.

Of course this is a terrible hack. What is really needed is a new column type that inserts stuff in the right place (rather like \texttt{!} but without the spacing that that does). The \texttt{\newcolumntype} command of \texttt{array} only adds ‘second class’
column types. The re-implementations of \texttt{newcolumntype} in my blkarray or Mark Wooding’s mdwtab allow new ‘first class’ column types to be declared, but stick with array for now. This means we have to lift the stuff out of the register before the register gets emptied in the wrong place.

\begin{verbatim}
\texttt{\expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil}
\end{verbatim}

Save the entry into a box (using a double group for colour safety as usual).

\begin{verbatim}
\@addtopreamble{%
\setbox\z@\hbox{\bgroup\group
\CT@everycr{}
\ifcase \@chnum
\hskip\stretch{.5}\kern\z@
\dollarbegin\insert@column\dollarend\do@row@strut\hskip\stretch{.5}\or
\hfill\kern\z@\dollarbegin\insert@column\dollarend\or
\vtop\@startpbox{\@nextchar}\insert@column\@endpbox\do@row@strut\or
\vbox\@startpbox{\@nextchar}\insert@column\@endpbox\do@row@strut\or
\fi
\or
\vtop\@startpbox{\@nextchar}\insert@column\@endpbox\do@row@strut\or
\vbox\@startpbox{\@nextchar}\insert@column\@endpbox\do@row@strut\fi
\egroup\egroup
\end{verbatim}

The main new stuff.

\begin{verbatim}
\begingroup
\CT@setup
Run any code resulting from \texttt{\columncolor} commands.
\CT@column@color
Run code from \texttt{\rowcolor} (so this takes precedence over \texttt{\columncolor}).
\CT@row@color
Run code from \texttt{\cellcolor} (so this takes precedence over both \texttt{\columncolor} and \texttt{\rowcolor}).
\CT@cell@color
\end{verbatim}
This is \relax unless one of the three previous commands has requested a colour, in which case it will be \CT@do@color which will insert \leaders of appropriate colour.

\CT@do@color
\endgroup

Nothing to do with colour this bit, since we are boxing and measuring the entry anyway may as well check the height, so that large entries don’t bump into horizontal rules (or the top of the colour panels).

\tempdima\ht\z@
\advance\tempdima\minrowclearance
\vrule\@height\tempdima\@width\z@

It would be safer to leave this boxed, but unboxing allows some flexibility. However the total glue stretch should either be finite or fil (which will be ignored). There may be fill glue (which will not be ignored) but it should \textit{total \texttt{0fill}}. If this box contributes fill glue, then the leaders will not reach the full width of the entry. In the case of \texttt{\multicolumn} entries it is actually possible for this box to contribute \texttt{shrink} glue, in which case the coloured panel for that entry will be too wide. Tough luck.

\unhbox\z@}%
\prepnext@tok}

\CT@setup Initialise the overhang lengths and the colour command.

\def\CT@setup{%
\@tempdimb\col@sep
\@tempdimc\col@sep
\def\CT@color{%
\global\let\CT@do@color\CT@@do@color
\color}}

\CT@@do@color The main point of the package: Add the colour panels.

Add a leader of the specified colour, with natural width the width of the entry plus the specified overhangs and \texttt{1fill} stretch. Surround by negative kerns so total natural width is not affected by overhang.

\def\CT@do@color{%
\global\let\CT@do@color\relax
\tempdima\wd\z@
\advance\tempdima\tempdimb
\advance\tempdima\tempdimc
\kern-\tempdimb
\leaders\vrule

For quick debugging with xdvi (which can’t do colours). Limit the size of the rule, so I can see the text as well.

```
^A \@height\p@\@depth\p@
\hskip\tempdima\@plus 1fill
\kern-\tempdimc
```

Now glue to exactly compensate for the leaders.

\hskip-\wd\z@ \@plus -1fill }
\CT@extract Now the code to extract the \columncolor commands.
\CT@extractb Define \CT@columncolor to add the right colour, and save the overhang lengths.
\CT@extractd Now look for left-overhang (default to \col@sep).
\CT@extracte Same for right-overhang (default to left-overhang).
\CT@extractf Add the overhang info to \CT@do@color, for executing later.
\CT@everycr Steal \everypar to initialise row colours
\CT@start
\let\CT@row@color@save\CT@row@color
\let\CT@cell@color@save\CT@cell@color
\global\let\CT@cell@color\relax}
\CT@end
\def\CT@end{\global\let\CT@arc@\CT@arc@save\global\let\CT@drsc@\CT@drsc@save\global\let\CT@row@color\CT@row@color@save\global\let\CT@cell@color\CT@cell@color@save}
\shortstack\shortstack
\gdef\@ishortstack#1{\CT@start\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\CT@end\egroup}
\@tabarray
array and tabular (delayed for delarray)
\AtBeginDocument{%\expandafter\def\expandafter\@tabarray\expandafter{\expandafter\CT@start\@tabarray}}
\endarray
\def\endarray{%\crcr \egroup \egroup\@arrayright\gdef\@preamble{}\CT@end}
\multicolumn \multicolumn
\long\def\multicolumn#1#2#3{\multispan{#1}\begingroup\def\@addamp{\if@firstamp\@firstampfalse\else\@preamerr 5\fi}%\@mkpream{#2}\@addtopreamble\@empty\endgroup\def\@sharp{#3}\let\CT@cell@color\relax\let\CT@column@color\relax\let\CT@do@color\relax\@arstrut \@preamble\null\ignorespaces}
\@classvi Coloured rules and rule separations.
\def\@classvi{\ifcase \@lastchclass\@acol \or\ifx\CT@drsc@\relax\@addtopreamble{\hskip\doublerulesep}%\else\@addtopreamble{{\CT@drsc@\vrule\@width\doublerulesep}}%\fi\or\@acol \or\@classvii\fi}
\doublerulesepcolor
140 \def\doublerulesepcolor#1\{(\CT@drs[#1])

\CT@drs
141 \def\CT@drs#1#2{%
142 \ifdim\baselineskip=\z@\noalign\fi
143 {\gdef\CT@drsc@{\color#1{#2}}}

\CT@drsc@
144 \let\CT@drsc@\relax

\arrayrulecolor
145 \def\arrayrulecolor#1\{(\CT@arc[#1])

\CT@arc
146 \def\CT@arc#1#2{%
147 \ifdim\baselineskip=\z@\noalign\fi
148 {\gdef\CT@arc@{\color#1{#2}}}

\CT@arc@
149 \let\CT@arc@\relax

hline

\@arrayrule
150 \def\@arrayrule{{\addtopreamble\{\CT@arc@\vline\}}}

\hline
151 \def\hline{%
152 \noalign{\ifnum0='}\fi
153 \let\hskip\vskip
154 \let\vrule\hrule
155 \let\@width\@height
156 {\CT@arc@\vline}\%
157 \futurelet
158 \reserved@a\@xhline}

\@xhline
159 \def\@xhline{\if\reserved@a\hline
160 {\if\CT@drsc@\relax
161 \vskip
162 \else
163 {\CT@drsc@\hrule}\@height
164 \fi
165 \{\doublerulesep\%
166 \fi
167 \ifnum0='\{\fi\}}

\cline \cline doesn’t really work, as it comes behind the coloured panels, but at least
make it the right colour (the bits you can see, anyway).
168 \def\@cline#1-#2\@nil{%
169 \omit
170 \@multicnt#1\%
\minrowclearance The row height fudge length.
\newlength\minrowclearance
\minrowclearance=0pt
\@mkpream While expanding the preamble array passes tokens through an \edef. It doesn’t use \protect as it thinks it has full control at that point. As the redefinition above adds \color, I need to add that to the list of commands made safe.
\let\@mkpreamarray\@mkpream
\def\@mkpream{\
\let\CT@setup\relax\let\CT@color\relax\let\CT@do@color\relax\let\color\relax\let\CT@column@color\relax\let\CT@row@color\relax\let\CT@cell@color\relax\@mkpreamarray}\
\CT@do@color For similar reasons, need to make this non-expandable
\let\CT@do@color\relax
\rowcolor
\def\rowcolor{%\noalign{\ifnum0='}fi\global\let\CT@do@color\CT@@do@color\@ifnextchar\[
\CT@rowa\CT@rowb}\
\CT@rowb#1{\
\CT@rowc}
\CT@rowc\def\CT@rowc{\@ifnextchar\[
\CT@rowd\CT@roww}\
\CT@roww#1{%\gdef\CT@rowcolor{\CT@color[#1]{#2}}\CT@rowc}
\CT@rowc
\def\CT@rowcolor#1{%\gdef\CT@rowcolor{\CT@color[#1]{#2}}\CT@rowc}
\CT@rowc
\def\CT@rowcolor{%\gdef\CT@rowcolor{\CT@color#1}}\CT@rowc}
\CT@rowc
\def\CT@rowcolor{%\gdef\CT@rowcolor{%\ifnum'=0'\fi}}\CT@rowd\CT@roww}\
\CT@roww#1{%\@testopt{\CT@rowe[#1]{#1}}
Tests whether the argument \langle arg\rangle is empty and executes the following code accordingly; \langle arg\rangle must not start with the token \XC@@. Can also be used within \edef.

\def\@ifxempty#1\@@ifxempty\XC@@ \@@ifxempty#1#2\XC@@\{\ifx#1\@@ifxempty\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi\}

\rowcolors \(\langle\text{commands}\rangle\}\{\langle\text{row}\rangle\}\{\langle\text{odd-row color}\rangle\}\{\langle\text{even-row color}\rangle\}

Defines alternating colors for the next tabular environment. Starting with row \langle row\rangle, odd and even rows get their respective colors. The color arguments may also be left empty (= no color). Optional commands may be \hline or \noalign{\langle stuff\rangle}.

In the starred version, \(\langle\text{commands}\rangle\) are ignored in rows with inactive rowcolors status (see below), whereas in the non-starred version, \(\langle\text{commands}\rangle\) are applied to every row of the table.

\def\rowcolors\{\@ifstar\@rowcmdfalse\rowc@lors\{\@rowcmdtrue\rowc@lors\}\}

\def\rowc@lors\[#1\]#2#3#4\%\{\global\rownum=\z@\global\@rowcolorstrue\@ifxempty{#3}\{\def\@oddrowcolor{\@norowcolor}\}{\def\@oddrowcolor{\gdef\CT@row@color{\CT@color{#3}}}\}@ifxempty{#4}\{\def\@evenrowcolor{\@norowcolor}\}{\def\@evenrowcolor{\gdef\CT@row@color{\CT@color{#4}}}\}\if@rowcmd\def\@rowcolors\{\@ifstar\{\@rowcmdfalse\rowc@lors\}\{\@rowcmdtrue\rowc@lors\}\}\fi\else\def\@rowcolors\{\@ifstar\{\@rowcmdfalse\rowc@lors\}\{\@rowcmdtrue\rowc@lors\}\}\fi\}

\if@rowcmd
\def\@rowcolors\{\#1\if@rowcolors\noalign{\fi}\else\\ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi\}\else
\\if@rowcolors
\{\#1\if@rowcolors\noalign{\fi}\else\\ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi\}\else
\fi\}

\global\rownum=\z@
\global\@rowcolorstrue
\@ifxempty\{\global\@oddrowcolor\}%
\@ifxempty\{\global\@evenrowcolor\}%
\if\@rowcmd
\\ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi%
\texttt{\CT@everycr{\@rowcolors\the\everycr}}
\texttt{\CT@everycr{\ignorespaces}}
\texttt{\def\@rowcolors{\noalign{\global\advance\rownum\@ne}@rowcolors}}
\texttt{\let\@rowcolors\@empty}
\texttt{\showrowcolors}
\texttt{\hiderowcolors}
\texttt{\if@rowcolors}
\texttt{\if@rowcmd}
\texttt{\newif\if@rowcolors}
\texttt{\newif\if@rowcmd}
\texttt{\rownum}
\texttt{\c@rownum}
\texttt{\let\c@rownum\rownum}
\texttt{\providecommand\therownum{\arabic{rownum}}}
\texttt{\cellcolor{\cellcolor}}
\texttt{\cellcolor} applies the specified colour to just its own tabular cell. It is defined robust, but without using \texttt{\DeclareRobustCommand} or \texttt{\newcommand{\ldots}} because those forms are not used elsewhere, and would not work in very old \LaTeX.\texttt{\edef\cellcolor{\noexpand\protect\expandafter\noexpand\csname cellcolor \endcsname}}
\texttt{\@namedef{cellcolor }{\CT@cellc[#1]{\#2}{\#3}}}\texttt{\def\CT@cellc[#1]{\#2}{\#3}{\global\let\CT@cell@color\relax}}
\texttt{\global\let\CT@cell@color\relax}
\texttt{\DC@endright} dcolumn support. the \texttt{D} column sometimes internally converts a \texttt{c} column to an \texttt{r} one by squashing the supplied glue. This is bad news for this package, so redefine it to add negative glue to one side and positive to the other to keep the total added zero.
\texttt{\AtBeginDocument{\let\@tempa{$\hfil\box@z@\box@w@}}}
\texttt{\def\DC@endright{\expandafter\protect\expandafter\expandafter\csname cellcolor \endcsname}}
\texttt{\@namedef{cellcolor }{\CT@cellc[#1][#2][#3]}\texttt{\edef\cellcolor{\noexpand\protect\expandafter\noexpand\csname cellcolor \endcsname}}
\texttt{\@ifnextchar[\CT@cellc@firstofone]{\CT@cellc@gobble[]}\texttt{\def\CT@cellc@firstofone[\CT@cellc@gobble[]]{\CT@cellc@gobble[]}\texttt{\def\CT@cellc@firstofone[\CT@cellc@gobble[]]{\CT@cellc@gobble[]}}
\texttt{\global\let\CT@cell@color\relax}}\texttt{\global\let\CT@cell@color\relax}
Old dcolumn code.

\def\DC@endright{\hfil\egroup}\hfill\box\z@\box\tw@%}
\fi
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}\hfill\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}\hfill\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.

\def\DC@endright{\hfil\egroup}%
\hskip\stretch{-.5}\box\z@\box\tw@%}
\else
\def\@tempa{$\hfil\egroup\hfill\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\Old dcolumn code.
Stop the backspacing for t and b, it messes up the underlying colour.
\else
global\let\LT@next\empty
gdef\CT@LT@sep{
noalign{\penalty-@lowpenalty\vskip-\arrayrulewidth}\%
\fi
\ifnum0=’{\fi}\%
multispan\LT@cols
\{\CT@arc\leaders\hrule\@height\arrayrulewidth\hfill\}cr
\CT@LT@sep
multispan\LT@cols
\{\CT@arc\leaders\hrule\@height\arrayrulewidth\hfill\}cr
noalign{\penalty@M}%
\LT@next\}
\fi}

\package