The \texttt{colortbl} package\textsuperscript{*}

David Carlisle\textsuperscript{†}

2024/07/06

\textbf{Abstract}

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

\section{Introduction}

This package is for coloring tables (i.e., giving colored 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{verbatim}
\begin{tabular}{|l|c|}
\hline
one & two \\
\hline
three & four \\
\hline
\end{tabular}
\end{verbatim}

\section{The \texttt{\textbackslash columncolor} 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 colored panels \textit{and} vertical rules in practice.

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

The basic format is:

\begin{verbatim}
\texttt{\textbackslash columncolor}[(color model)]\{(color)\} \{[left overhang]]\{[right overhang]]
\end{verbatim}

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 \textit{right overhang} argument is omitted then it defaults to

\textsuperscript{*}This file has version number v1.0i, last revised 2024/07/06.

\textsuperscript{†}Report issues to https://github.com/davidcarlisle/dpctex/issues
*left overhang.* If they are both omitted they default to \tabcolsep (in tabular) or \arraycolsep (in array).

If the overhangs are both set to 0pt then the effect is:

```
|>{\columncolor[gray]{.8}[0pt]}l|
|>{\color{white}%
 \columncolor[gray]{.2}[0pt]}l|
```

The default overhang of \tabcolsep produces:

```
|>{\columncolor[gray]{.8}l|
|>{\color{white}%
 \columncolor[gray]{.2}l|
```

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

```
|>{\columncolor[gray]{.8}[.5\tabcolsep]}l|
|>{\color{white}%
 \columncolor[gray]{.2}[.5\tabcolsep]}l|
```

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}

```
\begin{longtable}{|l|l|}
\hline
\textbf{First two columns} & \textbf{Third column} \\
\hline
\textbf{P-column} & \textbf{p-type} & \textbf{D-type (dcolumn)} & 12.34 \\hline
\textbf{Total} & \textbf{(wrong)} & 100.6 \\hline
\textbf{Some long} & \textbf{bbb} & 1.2 \\hline
\textbf{text in the} & \textbf{and some long text in the second} & 1.345 \\hline
\textbf{first column} & \textbf{column} & \\hline
\textbf{aaa} & \textbf{and another one} & \\hline
\textbf{Total} & \textbf{(wrong)} & 100.6 \\hline
\textbf{aaa} & \textbf{bbb} & 1.345 \\hline
\textbf{Note that} & \textbf{bbb} & 1.345 \\hline
\textbf{the colored} & & \\hline
\textbf{rules in all} & \textbf{rules in all} & \\hline
\textbf{columns} & \textbf{columns} & \\hline
\textbf{stretch to} & \textbf{stretch to} & \\hline
\textbf{accomodate} & \textbf{accomodate} & \\hline
\textbf{large entries} & \textbf{large entries} & \\hline
\textbf{in one} & \textbf{in one} & \\hline
\textbf{column.} & \textbf{column.} & \\hline
\end{longtable}
```

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

\begin{verbatim}
\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}[.5\tabcolsep][\tabcolsep]}l}
\newcolumntype{I}{%  
  >{\columncolor[gray]{0.8}[.5\tabcolsep][\tabcolsep]}\raggedright  
  3}
\end{verbatim}
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{\textbackslash leader} mechanism which is used by this package to insert the ‘stretchy’ colored panels is rather like \texttt{glue}, the \texttt{\textbackslash 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

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

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

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

4 \texttt{The \textbackslash rowcolor command}

As demonstrated above, one may change the color of specified rows of a table by the use of \texttt{\textbackslash multicolumn} commands in each entry of the row. However if your table is to be marked principally by \texttt{\textbackslash rowcolor}, you may find this rather inconvenient. For this reason a new mechanism, \texttt{\textbackslash rowcolor}, has been introduced\textsuperscript{1}. \texttt{\textbackslash rowcolor} takes the same argument forms as \texttt{\textbackslash ctextbackslash olumncolor}. It must be used at the \texttt{start} of a row. If the optional overhang arguments are not used the overhangs will default to the overhangs specified in any \texttt{\textbackslash ctextbackslash olumncolor} commands for that column, or \texttt{\textbackslash tabcolsep (arraycolsep in array)}.

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

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

5 \texttt{The \textbackslash rowcolors command}

The \texttt{\textbackslash rowcolors} command and its documentation originate in the \texttt{xcolor} package by Dr. Uwe Kern.
One of these commands has to be executed before a table starts. \texttt{⟨row⟩} tells the number of the first row which should be colored according to the \texttt{⟨odd-row color⟩} and \texttt{⟨even-row color⟩} scheme. Each of the color arguments may also be left empty (= no color). In the starred version, \texttt{⟨commands⟩} are ignored in rows with inactive \texttt{rowcolors} status (see below), whereas in the non-starred version, \texttt{⟨commands⟩} are applied to every row of the table. Such optional commands may be \texttt{\textbackslash hline} or \texttt{\textbackslash noalign\{⟨stuff⟩\}.

The \texttt{rowcolors} status is activated (i.e., use coloring scheme) by default and/or \texttt{\textbackslash showrowcolors}, it is inactivated (i.e., ignore coloring scheme) by the command \texttt{\textbackslash hiderowcolors}. The counter \texttt{rownum} (or \LaTeX{} counter \texttt{rownum}) may be used within such a table to access the current row number.

At the present time, the \texttt{rownum} counter is only incremented in tables using \texttt{rowcolors}.

\begin{tabular}{ll}
\rowcolor{blue} & test & row 1 \\
\rowcolor{red} & test & row 2 \\
\rowcolor{green} & test & row 3 \\
\rowcolor{yellow} & test & row 4 \\
\rowcolor{black} & test & row 5 \\
\rowcolor{blue} & test & row 6 \\
\rowcolor{red} & test & row 7 \\
\rowcolor{green} & test & row 8 \\
\rowcolor{yellow} & test & row 9 \\
\hiderowcolors & test & row 10 \\
\showrowcolors & test & row 11 \\
\multicolumn{1}{l}{\texttt{\{\textbackslash multicol\{1\}\}\{\textbackslash multicolumn\{1\}\}\{\textbackslash multicolumn\{1\}\}}} & test & row 12 \\
\multicolumn{1}{l}{\texttt{\{\textbackslash multicolumn\{1\}\}}} & test & row 13 \\
\end{tabular}

6 The \texttt{\cellcolor} command

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

\footnote{At some cost to the internal complexity of this package}
7 Coloring rules.

So you want colored rules as well?

One could do vertical rules without any special commands, just use something like \vline where you’d normally use |. The space between || will normally be left white. If you want to color 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 colored 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 color.

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

8 \arrayrulecolor

\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 colors.

9 \doublerulesepcolor

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

\setlength\arrayrulewidth{2pt}\arrayrulecolor{blue}
\setlength\doublerulesepcolor{2pt}\doublerulesepcolor{yellow}
\begin{tabular}{||l||c||}
\hline\hline
one&two\ 
three&four
\hline
\end{tabular}

10 More fun with \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 \texttt{\arrayrulecolor} and \texttt{\doublerulesepcolor} 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:

```
\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} \arrayrulecolor{violet}\doublerulesepcolor[rgb]{.9,.9,1} 
\rainbowline{} 
1&2&3&4&5&6&7 
\end{tabular} 
```

11 Less fun with \texttt{\cline}

Lines produced by \texttt{\cline} are colored if you use \texttt{\arrayrulecolor} but you may not notice as they are covered up by any color panels in the following row. This is a ‘feature’ of \texttt{\cline}. If using this package you would probably better using the - rule type in a \texttt{\hhline} argument, rather than \texttt{\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 color 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 tabls packages provides a similar \tablinesep. I was going to give this the same name for compatibility with 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}
\texttt{\langle\ast\package\rangle}
Nasty hacky way used by all the graphics packages to include debugging code.
\edef\@tempa{%
\noexpand\AtEndOfPackage{%
\catcode'\noexpand\^^A\the\catcode'\^^A\relax}}
\@tempa
\catcode'\^^A=\catcode'\%
\DeclareOption{debugshow}{\catcode'\^^A=9 }
All the other options are handled by the color package.
\DeclareOption*{\PassOptionsToPackage\CurrentOption{color}}
\ProcessOptions
I need these so load them now. Actually Mark Wooding’s mdwtab package could probably work instead of array, but currently I assume array package internals so...
\RequirePackage{array,color}
\@classz First define stub for new array package code.
\if\do@row@strut\@undefined\let\do@row@strut\relax\fi
\@classz is the main function in the array package handling of primitive column types: It inserts the code for each of the column specifiers, ‘clrpmb’. The other classes deal with the other preamble tokens such as ‘@’ or ‘>’.
\def\@classsz{\@classx
\@tempcnta \count@
\prepnextr@tok
At this point the color specification for the background panel will be in the code for the ‘>’ specification of this column. This is saved in \toks\@emptokena but array will insert it too late (well it would work for c, but not for p) so fish the color 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 ! but without the spacing that does). The \newcolumntype command of array only adds ‘second class’
\end{verbatim}
column types. The re-implementations of \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.

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

\begin{verbatim}
15 \expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
Save the entry into a box (using a double group for color safety as usual).

16 \@addtopreamble{%
17 \setbox\z@\hbox\bgroup\bgroup
18 \CT@everycr{}
19 \ifcase \@chnum
20 \hskip\stretch{.5}\kern\z@
21 \dollarbegin
22 \insert@column
23 \dollarend\do@row@strut\hskip\stretch{.5}\or
24 \dollarbegin \insert@column \dollarend \do@row@strut \hfill
25 \or
26 \hfill\kern\z@ \dollarbegin \insert@column \dollarend \do@row@strut
27 \or
28 \ifx\ar@align@mcell\@undefined
29 \$\vcenter
30 \@startpbox{\@nextchar}\insert@column \@endpbox $
31 \else
32 \setbox\ar@mcellbox\vbox
33 \@startpbox{\@nextchar}\insert@column \@endpbox
34 \ar@align@mcell
35 \do@row@strut
36 \fi
37 \or
38 \vtop \@startpbox{\@nextchar}\insert@column \@endpbox\do@row@strut
39 \or
40 \vbox \@startpbox{\@nextchar}\insert@column \@endpbox\do@row@strut
41 \fi
Close the box register assignment.
42 \egroup\egroup
\end{verbatim}

The main new stuff.

\begin{verbatim}
\begingroup
Initialise color command and overhands.
\CT@setup
Run any code resulting from \columncolor commands.
\CT@columncolor
Run code from \rowcolor (so this takes precedence over \columncolor).
\CT@rowcolor
\end{verbatim}
Run code from \cellcolor (so this takes precedence over both \columncolor and \rowcolor).

This is \relax unless one of the three previous commands has requested a color, in which case it will be \CT@@do@color which will insert \leaders of appropriate color.

Nothing to do with color 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 color panels).

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 \fil glue (which will not be ignored) but it should total \fil. If this box contributes \fil glue, then the leaders will not reach the full width of the entry. In the case of \multicolumn entries it is actually possible for this box to contribute \shrink glue, in which case the colored panel for that entry will be too wide. Tough luck.

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

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

Add a leader of the specified color, with natural width the width of the entry plus the specified overhangs and \fil stretch. Surround by negative kerns so total natural width is not affected by overhang.

For quick debugging with xdvi (which can’t do colors). Limit the size of the rule, so I can see the text as well.
Now glue to exactly compensate for the leaders.
\hskip-\wd\z@ \@plus -1fill 

\CT@extract
Now the code to extract the \columncolor commands.
\def\CT@extract#1\columncolor#2#3\@nil{%  
\if\noexpand#2%  
! is a fake token inserted at the end.  
\let\CT@column@color\@empty  
\else  
If there was an optional argument  
\if\[\noexpand#2%  
\CT@extractb{#1}#3\@nil  
\else  
No optional argument  
\def\CT@column@color{\CT@color{#2}}%  
\CT@extractd{#1}#3\@nil  
\fi  
\fi}  
\CT@extractb
Define \CT@column@color to add the right color, and save the overhang lengths.  
Finally reconstitute the saved ‘>’ tokens, without the color specification.  
First grab the color spec, with optional arg.
\def\CT@extractb#1#2[\#3]{%  
\def\CT@column@color{\CT@color[#2]{#3}}%  
\CT@extractd{#1}}%  
\CT@extractd
Now look for left-overhang (default to \col@sep).
\def\CT@extractd#1{\@testopt{\CT@extracte{#1}}\col@sep}  
\CT@extracte
Same for right-overhang (default to left-overhang).
\def\CT@extracte#1[#2][\#3]{\@testopt{\CT@extractf{#1}[\#2][\#3]}{#2}}  
\CT@extractf
Add the overhang info to \CT@do@color, for executing later.
{\catcode`\!\active  
\gdef\CT@extractf#1[#2][\#3][\#4]\columncolor#5\@nil{%  
\@tempdimb#2\relax  
\@tempdimc#3\relax  
\edef!{\string!}%  
\edef\CT@column@color{\CT@column@color\@tempdimb\the\@tempdimb\@tempdimc\the\@tempdimc\relax}%  
\toks\@tempcnta{#1#4}}}%  
\CT@everycr
Steal \everypar to initialise row colors
\let\CT@everycr\everycr  
\newtoks\everycr  
\CT@everycr{\noalign{\global\let\CT@row@color\relax}\the\everycr}
\CT@start
\def\CT@start{%
\let\CT@arc@save\CT@arc@
\let\CT@drsc@save\CT@drsc@
\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{%\CT@start\@tabarray}
\endarray
\expandafter\def\expandafter\endarray\expandafter{%\endarray\CT@end}
\multicolumn
\multicolumn
Patch \multicolumn to restore color settings. Done this way to work wth different versions depending on the age of the array package.
\def\@tempa#1\@arstrut#2\relax{
\long\def\multicolumn##1##2##3{\#1%row@color
\let\CT@cell@color\relax
\let\CT@column@color\relax
\let\CT@do@color\relax
\@arstrut#2}}
\expandafter\@tempa\multicolumn{#1}{#2}{#3}\relax
\let\@temp\relax
\@classvi
Colored 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
\@classvii
\fi}
\multicolumn
\multicolumn
\multicolumn
\@classvii
Colored rules and rule separations.
\def\@classvii{\ifcase\@lastchclass
\@acol \or
\ifx\CT@drsc@\relax
\@addtopreamble{\hskip\doublerulesep}\
\else
\@addtopreamble{{\CT@drsc@\vrule\@width\doublerulesep}}\
\fi
\or
\@acol \or
\@classvi
\fi}
doesn’t really work, as it comes behind the colored panels, but at least make it the right color (the bits you can see, anyway).
\advance\@multispan\@ne
\ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
\advance\@multicnt\@ne
\advance\@multispan\@ne
{\CT@arc\leaders\hrule\@height\arrayrulewidth\hfill}%
\cr
\noalign{\vskip-\arrayrulewidth}}
\minrowclearance The row height fudge length.
\newlength\minrowclearance
\minrowclearance=0pt
\@mkpream While expanding the preamble \array passes tokens through an \edef. It doesn’t
\@mkpreamarray 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@rowa
\def\CT@rowa[#1]#2{%
\gdef\CT@row@color{\CT@color[#1]{#2}}%
\CT@rowc}
\CT@rowb
\def\CT@rowb#1{%
\gdef\CT@row@color{\CT@color{#1}}%
\CT@rowc}
\CT@rowc
\def\CT@rowc{%
\@ifnextchar[\CT@rowd{\ifnum'{=0}\fi}}
\CT@rowd
\def\CT@rowd[#1]{\@testopt{\CT@rowe[#1]}{#1}}
\CT@rove
204 \def\CT@rove[#1][#2]{% 205  \@tempdimb#1% 206  \@tempdimc#2% 207  \edef\CT@row@color{% 208    \expandafter\noexpand\CT@row@color 209    \@tempdimb\the\@tempdimb 210    \@tempdimc\the\@tempdimc 211    \relax}% 212  \ifnum0='\fi}
\@ifxempty
213 \def\@ifxempty#1{\@@ifxempty#1\@@ifxempty\XC@@}
214 \def\@@ifxempty#1#2\XC@@ 215 {\ifx#1\@@ifxempty 216 \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
\rowcolors \rowcolors* 217 \global\rownum=\z@ 218 \@rowcolorstrue 219 \if@rowcmd 220 \def\@ROWcolors 221 {\if@rowcolors 222 \noalign{\ifnum\rownum<#2}\@norowcolor 223 \else 224 \ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi} 225 \fi} 226 \else 227 \def\@ROWcolors 228 {\if@rowcolors 229 \ifnum\rownum<#2}\noalign{\@norowcolor} 230 \else 231 \#1\if@rowcolors 232 \noalign{\relax\ifnum\rownum<#2}\@norowcolor\else 233 \ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi\fi 234 \fi\fi} 235 \else 236 \def\@ROWcolors 237 {\if@rowcolors 238 \ifnum\rownum<#2}\noalign{\@norowcolor}\else 239 \#1\noalign{\ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi}\fi\fi

\CT@rove
\@ifxempty \langle\textit{arg}\rangle \{\langle\textit{empty}\rangle \{\langle\textit{non-empty}\rangle \}
Tests without expanding, whether the argument \langle\textit{arg}\rangle is empty and executes the following code accordingly; \langle\textit{arg}\rangle must not start with the token \XC@@. Can also be used within \edef. 213 \def\@ifxempty#1{\@@ifxempty#1\@@ifxempty\XC@@}
214 \def\@@ifxempty#1#2\XC@@ {\ifx#1\@@ifxempty 215 \expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
\rowcolors \rowcolors* 217 \global\rownum=\z@ 218 \@rowcolorstrue 219 \if@rowcmd 220 \def\@ROWcolors 221 {\if@rowcolors 222 \noalign{\ifnum\rownum<#2}\@norowcolor 223 \else 224 \ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi} 225 \fi} 226 \else 227 \def\@ROWcolors 228 {\if@rowcolors 229 \ifnum\rownum<#2}\noalign{\@norowcolor} 230 \else 231 \#1\if@rowcolors 232 \noalign{\relax\ifnum\rownum<#2}\@norowcolor\else 233 \ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi\fi\fi 234 \fi\fi} 235 \else 236 \def\@ROWcolors 237 {\if@rowcolors 238 \ifnum\rownum<#2}\noalign{\@norowcolor}\else 239 \#1\noalign{\ifodd\rownum\@oddrowcolor\else\@evenrowcolor\fi}\fi\fi

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

In the starred version, \langle\textit{commands}\rangle are ignored in rows with inactive rowcolors status (see below), whereas in the non-starred version, \langle\textit{commands}\rangle are applied to every row of the table.
\showrovcolors Switch coloring mode on/off.
\hiderovcolors
\if@rovcmd
\newif\if@rovcolors
\newif\if@rowcmd
\rownum Reserve a counter register. Also alias as a \LaTeX{} counter (but not via \newcounter as should not be in the reset list.)
\c@rownum
\if@rowcmd
\let\c@rownum\rownum
\providecommand\therownum{\arabic{rownum}}
\cellcolor \cellcolor applies the specified color to just its own tabular cell. It is defined robust, but without using \DeclareRobustCommand or \newcommand because those forms are not used elsewhere, and would not work in very old \LaTeX.
\edef\cellcolor{\noexpand\protect\expandafter\noexpand\csname cellcolor \endcsname}
\@namedef{cellcolor}{\@ifnextchar[{{\CT@cellc\@firstofone}{\CT@cellc\@gobble[}]{}}}
\def\CT@cellc#1[#2]#3{\expandafter\gdef\expandafter\CT@cell@color\expandafter{\CT@color#1[#2]{#3}\global\let\CT@cell@color\relax}}
\global\let\CT@cell@color\relax
\dcolumn support. the D column sometimes internally converts a c column to an 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.
\AtBeginDocument{%
\def\@tempa{\hfil\egroup\box\z@\box\tw@}%
\ifx\@tempa\DC@endright
\def\DC@endright{\edef\CT@cellc{\hfill}}%
\global\let\CT@cellc\relax
\DC@endright
dcolumn, only want to fudge it in the D{.}{.}{3} case, not the new D{.}{.}{3.3} possibility. \hfill has already been inserted, so need to remove 1fill’s worth of stretch.
Old dcolumn code.

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

\def\DC@endright{% 
  \hfill\box\z@\box\tw@\hskip\stretch{-.5}}%
\fi
\fi

\AtBeginDocument{% 
  \ifx\hhline\@undefined\else
    \def\HH@box#1#2{\vbox{{% 
      \ifx\CT@drsc@\relax\else
        \global\dimen\thr@@\tw@\arrayrulewidth
        \global\advance\dimen\thr@@\doublerulesep
        {\CT@drsc@
          \hrule \@height\dimen\thr@@ 
          \vskip-\dimen\thr@@}%
      \fi
      \CT@arc@
      \hrule \@height\arrayrulewidth \@width #1 
      \vskip\doublerulesep 
      \hrule \@height\arrayrulewidth \@width #2}}}
    \def\HH@loop{\ifx\@tempb\def\next##1{\the\toks@\cr}\else\let\next\HH@let
      \ifx\@tempb|\if@tempswa
        \ifx\CT@drsc@\relax
          \HH@add{\hskip\doublerulesep}
        \else
          \HH@add{{\CT@drsc@\vrule\@width\doublerulesep}}%
        \fi
      \fi\@tempswatrue
      \HH@add{{\CT@arc@\vline}}\else
        \ifx\@tempb:\if@tempswa
          \HH@add{\hskip\doublerulesep}
        \fi\@tempswatrue
        \ifx\@tempb~\@tempswafalse
          \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
        \fi
Stop the backspacing for t and b, it messes up the underlying color.

\ifx\CT@drsc@\relax
\HH@add{\hfil}\else
\HH@add{%
\CT@drsc@\leaders\hrule@\height@\HH@height\hfil}%)%\fi
\else
\fi
\else
\ifx\@tempb-\@tempswafalse
\gdef\HH@height{\arrayrulewidth}%
\if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
\HH@add{%\CT@arc@\leaders\hrule@\height@\arrayrulewidth\hfil}%\else
\ifx\@tempb=\@tempswafalse
\gdef\HH@height{\dimen\thr@@}%
\if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
\HH@add
{\rlap{\copy\@ne}\leaders\copy\@ne\hfil\llap{\copy\@ne}}\else
\ifx\@tempb t\HH@add{%
\def\HH@height{\dimen\thr@@}%
\HH@box\doublerulesep\z@}\@tempswafalse\else
\ifx\@tempb b\HH@add{%
\def\HH@height{\dimen\thr@@}%
\HH@box\z@\doublerulesep}\@tempswafalse\else
\ifx\@tempb>\def\next##1##2{%
\HH@add{\baselineskip\p@elax
#2}%
\global\setbox\@ne\HH@box\doublerulesep\doublerulesep}%)%\else
\ifx\@tempb=\@sptoken\let\next\HH@spacelet\else
\PackageWarning{hhline}{}{\meaning\@tempb\space ignored in \noexpand\hhline argument\MessageBreak}%
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
\next}
\lowercase{\def\HH@spacelet}{\futurelet\@tempb \HH@loop}
\fi}
\ExplSyntaxOn

longtable support.

\AtBeginDocument{\def\LT@hline{%
\ifx\LT@next\hline
\global\let\LT@next\@gobble\else
\CT@drsc@\leaders\hrule@\height@\HH@height\hfil}%)%\global\let\LT@hline\@empty\fi}
\cs_if_exist:NF\tag_mc_begin:n{
\cs_new:Npn\tag_mc_begin:n#1{}}
\cs_new:Nnpn\tag_mc_end:{}{}
\AtEndDocument{\def\LT@hline{%
\ifx\LT@next\hline
\global\let\LT@next\@gobble\else
\CT@drsc@\leaders\hrule@\height@\HH@height\hfil}%)%\global\let\LT@hline\@empty\fi}
\noalign{\penalty-\@medpenalty\vskip\doublerulesep}\
\else\
gdef\CT@LT@sep{%\
\multispan{LT@cols}{%\
\tag_mc_begin:n{artifact}\n\CT@dsc\leaders\hrule@height\doublerulesep\hfill\
\tag_mc_end: \int_gdecr:N \g__tbl_row_int\n}\cr}\
\fi\
\else\
global\let\LT@next\empty\
gdef\CT@LT@sep{%\
\noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}\
\fi\
\ifnum0='{\fi}%\
\multispan{LT@cols\n{\tag_mc_begin:n{artifact}\n\CT@arc\leaders\hrule@height\arrayrulewidth\hfill\n\tag_mc_end: \int_gdecr:N \g__tbl_row_int\n}\cr}\
\CT@LT@sep\
\multispan{LT@cols\n{\tag_mc_begin:n{artifact}\n\CT@arc\leaders\hrule@height\arrayrulewidth\hfill\n\tag_mc_end: \int_gdecr:N \g__tbl_row_int\n}\cr}\
\noalign{\penalty@M}%\
\LT@next\
\ExplSyntaxOff

\package