`tabularray` and `kableExtra`

Flexible LaTeX tables

Table of contents

This vignette requires this `kableExtra` development branch:

```
remotes::install_github("vincentarelbundock/kableExtra@tabularray")
```

This vignette demonstrates how to harness the capabilities of `tabularray` and `kableExtra` for crafting advanced LaTeX tables in R. Readers will learn to use the `kbl()` function for transforming data frames into sophisticated LaTeX tables. The integration of `kableExtra` with `tabularray` brings a host of benefits, including versatile cell alignment and the ability to create colorful, multirow, and multicolumn tables. The guide covers controlling rows, columns, cells, line manipulation, and table width management, aiming to equip users with the knowledge to create responsive, visually appealing, and well-structured LaTeX tables.

Click here to visit the `tabularray` website and read its documentation.

To create tables with `tabularray` in `kableExtra`, users only need to set the `tabular` argument of the `kbl` function:

```
library(kableExtra)
pkgload::load_all()
```

```
i Loading kableExtra
```

```
df <- data.frame(Car = row.names(mtcars), mtcars[, 1:3])[1:4,]
row.names(df) <- NULL

kbl(df, tabular = "tblr")
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

Why `tabularray`?

`kableExtra` supports the `tabularray` package for LaTeX to create tables with advanced formatting options. `tabularray` offers several benefits:

- Versatile cell alignment
- Multirow and multicolumn support

- Flexible column types
- Advanced line customization
- Colorful table options
- Improved vertical spacing
- Compatibility with LaTeX3
- Support for long tables
- Integration with popular LaTeX libraries

One of the most important benefits of using `tabularray` is that code generated by `kableExtra` is very readable, and much easier to edit manually than when using other LaTeX packages. For example, here is the code for a table with colored and bolded rows:

```r
kbl(df, tabular = "tblr") |>
  row_spec(2:3, bold = TRUE, background = "pink") |>
  cat()
```

```
\begin{tblr}[
]
{
colspec={Q[halign=l]Q[halign=r]Q[halign=r]Q[halign=r]},
rowspec={Q[]Q[]Q[bg=pink, font=\bfseries]Q[bg=pink, font=\bfseries]Q[]Q[]},
}
\hline
Car & mpg & cyl & disp\\
\hline
Mazda RX4 & 21.0 & 6 & 160\\
Mazda RX4 Wag & 21.0 & 6 & 160\\
Datsun 710 & 22.8 & 4 & 108\\
Hornet 4 Drive & 21.4 & 6 & 258\\
\hline
\end{tblr}
```

Here are some of the important things to notice:

- The tabular environment starts with `\being{tblr}` and ends with `\end{tblr}`.
- The data and style are completely independent: The tabular content is untouched, and we add `colspec` and `rowspec` headers to specify the style.
- Each row and each column gets a `Q[]` entry, which acts as a styling operator. We can insert arguments in the square brackets to change the style of all the cells in a column or row. The available arguments are described in the `tabularray` documentation.

In contrast, when using other LaTeX packages, each entry of the table must be modified, which arguably makes the code messier and harder to edit:

```r
kbl(df, format = "latex") |>
  row_spec(2:3, bold = TRUE, background = "pink") |>
  cat()
```

```
\begin{tabular}[t]{l|r|r|r}
\hline
Car & mpg & cyl & disp\\
\hline
Mazda RX4 & 21.0 & 6 & 160\\
\hline
\cellcolor{pink}{\textbf{Mazda RX4 Wag}} & \cellcolor{pink}{\textbf{21.0}} & \cellcolor{pink
\hline
\cellcolor{pink}{\textbf{Datsun 710}} & \cellcolor{pink}{\textbf{22.8}} & \cellcolor{pink}{\
\hline
Hornet 4 Drive & 21.4 & 6 & 258\\
\hline
\end{tabular}
```

Text styles

The core functions to modify text styles in `kableExtra` are `cell_spec()`, `row_spec()`, and `column_spec()`. All the arguments of these functions are supported, except for 3 arguments of the `row_spec()` function: `angle`, `font_size`, and `align`. Users can achieve a similar effect using the `cell_spec()` function (see the section below on cell-specific settings).

Here is an example of a table with bold text and strikethroughs:

```r
kbl(df, tabular = "tblr") |>
  row_spec(2:3, bold = TRUE) |>
  column_spec(1, strikeout = TRUE)
```

~~Car~~	mpg	cyl	disp
~~Mazda RX4~~	21.0	6	160
~~Mazda RX4 Wag~~	**21.0**	**6**	**160**
~~Datsun 710~~	**22.8**	**4**	**108**
~~Hornet 4 Drive~~	21.4	6	258

Colors

In `tabularray`, color names are supported through the integration of the `xcolor` and `ninecolors` packages. The `xcolor` package is a comprehensive solution in LaTeX for color customization, offering a wide range of predefined color names and the ability to define custom colors using various color models like RGB, CMYK, and HTML. This flexibility allows for precise color specification and is ideal for setting text, table elements, and other document components in LaTeX.

The basic LaTeX colors are: black, blue, brown, cyan, darkgray, gray, green, lightgray, lime, magenta, olive, orange, pink, purple, red, teal, violet, white, yellow.

The `ninecolors` package adds suffix to 9 of those colors to set 1 of 13 different hues: gray, red, brown, yellow, olive, green, teal, cyan, azure, blue, violet, magenta, purple. For all colors, 0 means black, and 10 is white. Two colors with the same numbered suffix have the same luminance (ex: `gray3` and `olive3`). These nine colors are carefully selected to ensure proper color contrast according to the Web Content Accessibility Guidelines (WCAG). This feature is particularly useful for creating documents with high readability and accessibility standards. To get proper WCAG Color Contrast, the `ninecolors` author recommends choosing two colors with different names, with at least a 5 unit difference in level.

The `col_spec()`, `row_spec()`, and `cell_spec()` functions support these colors out of the box:

```r
df |>
  kbl(tabular = "tblr") |>
  column_spec(1, color = "purple3") |>
  row_spec(2:3, background = "azure9")
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

Of course, `kableExtra` themes are also supported:

```r
df |>
  kbl(tabular = "tblr") |>
  kable_styling(latex_options = "striped")
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

Precedence: Rows, Columns, Cells

In some contexts, user-specified settings like cell colors or text styles enter in conflict. In such cases, these two rules are applied:

1. `cell_spec()` settings have the highest precedence.
2. The last function call determines if row or column settings have precedence: `column_spec()` vs `row_spec()`

In the following example, we define conflicting background colors at the cell, column, and row level. In that case, the cell wins. When there is only a conflict between rows and columns, rows win because `row_spec()` is called after `column_spec()`.

```r
df2 <- df
df2[1, 1] <- cell_spec(df2[1, 1], background = "yellow", format = "tblr")
kbl(df2, tabular = "tblr", escape = FALSE) |>
    column_spec(1:2, background = "pink") |>
    row_spec(1:2, background = "azure8")
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

Lines: Horizontal and Vertical

The `tabularray` package offers advanced table formatting options in LaTeX, especially for vertical and horizontal lines. The syntax to control these lines is slightly different than for other output format in `kableExtra`, so it deserves some explanation. Consider this example:

```r
kbl(df,
    tabular = "tblr",
    toprule = "", midrule = "", bottomrule = "",
    linesep = "hlines={dash=dotted, fg=brown6}",
    vline = "vlines={dash=dashed, fg=green4, wd=2pt}"
)
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

To achieve this result, we began by removing the default horizontal rules (`toprule`, `midrule`, `bottomrule`), otherwise, there would be double horizontal lines in the table. Then, we use the `linesep` argument to tell `tabularray` how to format horizontal lines in the table, and the `vline` argument to control vertical lines.

When `vlines` and `hlines` are in plural form, they control all lines in the table. `tabularray` also supports another syntax for specifying individual lines. For example, we can control the color, width, and type of lines as follows::

```r
kbl(df,
    tabular = "tblr",
    toprule = "", midrule = "", bottomrule = "",
    linesep = "hline{1-6}={dash=solid, fg=brown6}",
    vline = "vline{2,3}={dash=dotted, fg=green4}"
)
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

Width

To fix the width of certain columns or expand the table to full page width, `kableExtra` can use `X` or `Q` columns in `tabularray`. `X` columns are designed for scenarios where you want the table columns to automatically adjust their widths to fill the entire available space. This feature is particularly useful for creating tables that span the full width of a page or container. When you use `X` columns, the width of each column is proportionally divided based on the available space, allowing for a responsive and evenly distributed layout. On the other hand, `Q` columns function like the standard `p` columns in LaTeX, where you manually specify the width of each column. The content in these columns is wrapped to fit within the set width. This type of column is ideal when precise control over column width is needed, such as in tables with varying content lengths or specific design requirements.

When using the `kable_styling()` function in R for LaTeX output and setting `full_width = TRUE`, the table automatically employs `X` columns. This configuration makes the table expand to fill the width of its container, with each column adjusting its width to fit proportionally. This is a key feature for creating full-width tables that need to be responsive and aesthetically balanced in their layout.

This example illustrates how to customize the width of a single columns, while expanding the rest of the table to fill the entire page width:

```r
kbl(df,
    tabular = "tblr",
    align = "lccc") |>
    kable_styling(full_width = TRUE) |>
    column_spec(1, width = "8cm")
```

Car	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258

Cell-specific settings

To apply cell-specific settings, we can use the `cell_spec()` function. In `kbl()`, it is necessary to use `escape=TRUE` and `tabular="tblr"`. In `cell_spec()`, it is necessary to use `tabular="tblr"`:

```r
df2 <- df
df2[2, 2] <- cell_spec(
  df2[2, 2],
  background = "pink",
  align = "c",
```