How to create professional LaTeX Tables

Annalena Kofler, July 28, 2025

Abstract: This document showcases how to properly format tables in LaTeX, including float alignment, cell coloring, loading values from a csv file and automatic, value-dependent coloring.[1]

1 Basic rules about tables

Before we go into the details of table formatting, I want to state some general rules regarding tables:

- **Only use tables if you have no other way of visualizing the numbers.** Tables require a significantly larger amount of human staring-time to understand its content than any thought-through dot plots, line graphs, bar chart, or even pie charts. Since humans are very good at understanding visual relationships (e.g., spatial alignment, sizes, shapes, colors, and textures), tables could be seen as the boring grandma among the ingredients included in a paper. Unfortunately, tables have become the preferred way of communicating state-of-the-art performance and SOTA comparisons in machine learning. While I believe that most information in tables can be displayed in a figure with some effort, I understand that there will be cases where a table is the only solution.
- **Intentionally decide on your table content. Less is more.** Ideally, the column and row labels are descriptive without needing excessive explanation, the compared quantities are restricted to the most relevant ones, and the included numbers are directly comparable and can easily be interpreted. If you include multiple quantities where it is unclear if lower or higher is better, add small arrows (↑ or ↓) in the column header. Try to keep the size of the table to a minimum and if you don't have another choice, use visual clues (see next point).
- **Vertical lines should be avoided at all cost!**
- **The table caption has to be located above the table (contrary to figures!).** You can simply move `\caption{This is the table caption.}` above

```
\begin{tabular}{...}
    ...
\end{tabular}
```

- **Make an effort to include visual clues.** Humans tend to grasp colors, textures, shapes, and sizes before actually reading numbers. After reading the number, the reader has to first interpret it (Is it large or small? Is high or low better? ...) and put it in relation to the other values in the table before understanding its meaning. This means that tables are hard to understand if you don't know what you are looking for. As a result, it is the duty of the writer to include visual guidance within the table which helps the reader understand and interpret the table's contents. Visual cues can be anything from marking the best numbers in bold to highlighting the cell with different colors.

2 Aligning values at the comma

Tables can be hard to read, especially when different rows contain values specified to various decimal points. Ideally, these values should align after the comma. For this purpose, the `\usepackage{siunitx}`

[1] The code for this tutorial is available in this repository.

provides the `S`-type column. You can see an example showcasing this behavior in Table 1 which can be generated with this code:

LaTeX Code

```
\usepackage{booktabs} % For \toprule etc.
\usepackage{siunitx}  % For scientific alignment
\begin{table}
    \centering
    \caption{The caption should be above the table.}
    \label{table:alignment}
    \begin{tabular}{cS}
        \toprule
        not aligned & {aligned} \\ \midrule
        1.23 & 1.23 \\
        42.1 & 42.1\\
        678.910 & 678.910\\
        \bottomrule
    \end{tabular}
\end{table}
```

Table 1: The caption should be above the table.

not aligned	aligned
1.23	1.23
42.1	42.1
678.910	678.910

3 Highlighting table values vs. coloring table cells

If you compare different methods within a table, there is a notion of *X is better than Y.*. In such cases, you can guide the reader's eye by highlighting the better value. Oftentimes, the better value is put in the `\textbf{}` or `\mathbf{}` environment, however this can mess up the decimal alignment, as clearly visible in Table 2.

Table 2: Bold text vs. cell coloring.

	Bold text		Cell coloring	
	$\mathbb{E}_p\left[\log(p/q_\theta)\right] \downarrow$	$\epsilon(\%) \uparrow$	$\mathbb{E}_p\left[\log(p/q_\theta)\right] \downarrow$	$\epsilon(\%) \uparrow$
Baseline	—	7.52(21)	—	7.52(21)
Method 1	9.1581(11)	87.02(8)	9.1581(11)	87.02(8)
Method 2	**9.0978(5)**	**99.67(1)**	9.0978(5)	99.67(1)

Instead, we can highlight the background of the cell with a specific color, using the `\usepackage{xcolor}` and defining a custom color via RBG values as `\definecolor{blue}{RGB}{63, 144, 218}`. A specific cell can be colored with the command `\cellcolor{blue!25}` where the color is followed by `!25` to indicate "mix 25% of blue with 75% of white". If you want to mix two colors where one is not white, you can use this syntax: `<color1> ! <percentage> ! <color2>`.

The code to generate this table is included in the following for completeness:

LaTeX Code

```
    \usepackage{amssymb, amsmath}
    \usepackage{tabularx}
    \usepackage{booktabs} % For \toprule etc.
    \usepackage{siunitx} % For scientific alignment
    \usepackage{xcolor} % For cell color command
    \definecolor{blue}{RGB}{63, 144, 218} % Define background color

    \begin{table}[ht]
    \caption{Bold text vs. cell coloring.}
    \label{table:bold_vs_cell_color}
    \centering
    \begin{tabularx}{\linewidth}{
        X
        SS
        c
        !{\color{white}\ }S % Text before S forces some white space between columns.
        !{\color{white}\ }S
    }
        \toprule
        &
            \multicolumn{2}{c}{Bold text} &
            &
            \multicolumn{2}{c}{Cell coloring} \\

        \cmidrule{2-3}
        \cmidrule{5-6}
        & \multicolumn{1}{c}{$\mathbb{E}_p \left[ \log(p/q_\theta)\right] \,
    \downarrow$} &
            \multicolumn{1}{c}{$\epsilon (\unit{\percent}) \, \uparrow$} &
            &
            \multicolumn{1}{c}{$\mathbb{E}_p \left[ \log(p/q_\theta)\right] \,
    \downarrow$} &
            \multicolumn{1}{c}{$\epsilon (\unit{\percent}) \, \uparrow$} \\
        \midrule
        Baseline &
            \textemdash &
            7.52(0.21) &
            &
            \textemdash &
            7.52(0.21) \\ \addlinespace[.25em]
        Method 1 &
            9.1581(11) &
            87.02(8) &
            &
            9.1581(11) &
            87.02(8) \\ \addlinespace[.25em]
        Method 2 &
            \textbf{9.0978(5)} & % Try out both
            $\mathbf{99.67(1)}$ & % Try out both
            &
            \cellcolor{blue!25} 9.0978(5) &
            \cellcolor{blue!25} 99.67(1) \\
        \bottomrule
    \end{tabularx}
\end{table}
```

4 Importing csv values in LaTeX

Oftentimes, values are manually typed or copied over from another file into the LaTeXdocument. This is error-prone since it is very easy to miss a number which can completely change the scientific result. Instead, you can let your analysis script write the results into a `.csv` file, load it in LaTeXand print these values automatically in the text or into your table. This makes updating numbers so much easier and takes the human copying error out of the equation. It is especially helpful, if you need to include a lot of numbers.

4.1 Loading individual values

If you want to write individual values in your text, you can use the `datatool` package[2] You can define a macro in the preamble:

LaTeX Code

```
% In preamble
\usepackage{datatool}
\newcommand{\printval}[1]{% define shortcut for convenience
    \DTLfetch%
        {mydata}% database name {key}% key column name {#1}% key value
        {value}% value column name
    \xspace% handle spacing
}
```

and print it in the main document via:

LaTeX Code

```
% In main document
\DTLloaddb{mydata}{mydata.csv} % load CSV file into database ‘mydata‘
The value of x is \printval{x} and y = \printval{y}!
```

4.2 Loading a full table

To load a full table from a csv file[3], we use the `csvsimple` package and read in values within the `tabularx` environment via `\csvreader`. `csvreader` follows the syntax

LaTeX Code

```
\csvreader[<options>]{<filename>}{<column definitions>}{<code to execute per row>}
```

where `<code to execute per row>` is most often specified as

LaTeX Code

```
{\csvcoli & \csvcolii & \csvcoliii & \csvcoliv & ...}
```

We import the file `tables/table_plain.csv` with the following code:

[2]Code from this StackExchange answer.
[3]This part is based on this StackExchange discussion and its examples.

LaTeX Code

```
\begin{table*}[ht]
    \caption{Table with values loaded from csv.}
    \label{table:load_from_csv}
    \centering
    \begin{tabularx}{0.85\linewidth}{lSSSSSSS}
        \\ \toprule
        Categories & \multicolumn{1}{c}{A} & \multicolumn{1}{c}{B} &
\multicolumn{1}{c}{C} & \multicolumn{1}{c}{D} & \multicolumn{1}{c}{E} &
\multicolumn{1}{c}{F} & \multicolumn{1}{c}{G} % scientific column has trouble with
string column names
        \\ \midrule
        \csvreader[late after line=\\,
        late after last line=\\ \bottomrule
        ]{tables/table_plain.csv}
          {}
          {\csvcoli & \csvcolii & \csvcoliii & \csvcoliv & \csvcolv & \csvcolvi &
\csvcolvii & \csvcolviii}
    \end{tabularx}
\end{table*}
```

and the result can be found in Table 3.

Table 3: Table with values loaded from csv.

Categories	A	B	C	D	E	F	G
a	17.68	48.71	84.09	100.0	84.09	48.71	17.68
b	48.71	84.09	100.0	84.09	48.71	17.68	3.13
c	84.09	100.0	84.09	48.71	17.68	3.13	0.12
d	100.0	84.09	48.71	17.68	3.13	0.12	0.0
e	84.09	48.71	17.68	3.13	0.12	0.0	0.12
f	48.71	17.68	3.13	0.12	0.0	0.12	3.12
g	17.68	3.13	0.12	0.0	0.12	3.12	17.68
h	3.13	0.12	0.0	0.12	3.12	17.68	48.71
i	0.12	0.0	0.12	3.12	17.68	48.71	84.09
j	0.0	0.12	3.12	17.68	48.71	84.09	100.0
k	0.12	3.12	17.68	48.71	84.09	100.0	84.09
l	3.12	17.68	48.71	84.09	100.0	84.09	48.71
m	17.68	48.71	84.09	100.0	84.09	48.71	17.68
n	48.71	84.09	100.0	84.09	48.71	17.68	3.13
o	84.09	100.0	84.09	48.71	17.68	3.13	0.12

5 Automatically coloring a csv table with a color gradient

For large tables, we might want to automatically import the csv file and color the cell dependent on its float value[4]. Since it is not easy to define continuous colormap in LaTeX, we interpolate between a multiple individual colors to create a divergent colormap. Please be aware that red-green colormaps are not suitable for color impaired or colorblind readers, and we will use a red-blue colormap from this guide.

This requires the definition of a more elaborate macro. We start with defining the colors we want to interpolate and the interpolation values:

[4]This code is partially based on this StackExchange example which is in turn based partially on this link

LaTeX Code

```
\usepackage[table]{xcolor}
\usepackage{csvsimple}
\usepackage{etoolbox}
\usepackage{pgf} % for calculating the values for gradient
\definecolor{max_perc}{RGB}{016, 101, 171} % blue1
\definecolor{five_perc}{RGB}{142, 196, 222} %blue3
\definecolor{one_perc}{RGB}{249, 249, 249} % grayish1
\definecolor{towards_one_perc}{RGB}{246, 164, 130} % peach2
\definecolor{zero_perc}{RGB}{179, 021, 041} % red2
\newcommand*{\opacity}{40}% opacity of the cell background color
% Data set related: Color transition values
\newcommand*{\minval}{0.0}
\newcommand*{\valA}{0.8}
\newcommand*{\valB}{1.0}
\newcommand*{\valC}{5.0}
\newcommand*{\maxval}{70.0}
```

and finally, we can define the makro for the gradient function:

LaTeX Code

```
% Gradient function with intermediate colors
\newcommand{\gradient}[1]{%
    \pgfmathsetmacro{\value}{#1}%
    % Clamp to range [\minval, \maxval]
    \pgfmathsetmacro{\value}{min(max(\value,\minval),\maxval)}%
    % Color values in [\minval, \valA]
    \ifdim\value pt<\valA pt
        \pgfmathsetmacro{\pctraw}{100*(\value - \minval)/(\valA - \minval)}%
        \pgfmathparse{int(round(\pctraw))}%

\edef\colorcmd{\noexpand\cellcolor{towards_one_perc!\pgfmathresult!zero_perc!\opacity}}%
    % Color values in [\valA, \valB]
    \else\ifdim\value pt<\valB pt
        \pgfmathsetmacro{\pctraw}{100*(\value - \valA)/(\valB - \valA)}%
        \pgfmathparse{int(round(\pctraw))}%

\edef\colorcmd{\noexpand\cellcolor{one_perc!\pgfmathresult!towards_one_perc!\opacity}}%
    % Color values in [\valB, \valC]
    \else\ifdim\value pt<\valC pt
        \pgfmathsetmacro{\pctraw}{100*(\value - \valB)/(\valC - \valB)}%
        \pgfmathparse{int(round(\pctraw))}%

\edef\colorcmd{\noexpand\cellcolor{five_perc!\pgfmathresult!one_perc!\opacity}}%
    % Color values in [\valC, \maxval]
    \else
        \pgfmathsetmacro{\pctraw}{100*(\value - \valC)/(\maxval - \valC)}%
        \pgfmathparse{int(round(\pctraw))}%

\edef\colorcmd{\noexpand\cellcolor{max_perc!\pgfmathresult!five_perc!\opacity}}%
    \fi\fi\fi
    % Add percentage symbol after float
    \colorcmd\rule{0pt}{2.5ex}\hspace{4pt}#1~\%\hspace{3pt}
}
```

We can now simply wrap the floating point values with `\text{\gradient{...}}` when saving them into the csv file and create the table via:

LaTeX Code

```
\begin{table*}[ht]
    \caption{Colored table with values imported from csv.}
    \label{table:colored_imported_table}
    \centering
    \begin{tabularx}{0.85\linewidth}{lc@{}c@{}c@{}c@{}c@{}c@{}c@{}} % @{} removes
white spaces between columns
        \toprule
        Event & A & B & C & D & E & F & G \\ \midrule
        \csvreader[late after line=\\,late after last line=\\
\bottomrule]{tables/table_gradient.csv}
          {}
          {\csvcoli & \csvcolii & \csvcoliii & \csvcoliv & \csvcolv & \csvcolvi &
\csvcolvii & \csvcolviii}
    \end{tabularx}
\end{table*}
```

The final table is displayed in Table 4. Unfortunately, it is not possible to load a plain csv file and color the cells that contain the maximal value per column, since `\csvreader` reads the data line by line.

Table 4: Colored table with values imported from csv.

Event	A	B	C	D	E	F	G
a	17.68 %	48.71 %	84.09 %	100.0 %	84.09 %	48.71 %	17.68 %
b	48.71 %	84.09 %	100.0 %	84.09 %	48.71 %	17.68 %	3.13 %
c	84.09 %	100.0 %	84.09 %	48.71 %	17.68 %	3.13 %	0.12 %
d	100.0 %	84.09 %	48.71 %	17.68 %	3.13 %	0.12 %	0.0 %
e	84.09 %	48.71 %	17.68 %	3.13 %	0.12 %	0.0 %	0.12 %
f	48.71 %	17.68 %	3.13 %	0.12 %	0.0 %	0.12 %	3.12 %
g	17.68 %	3.13 %	0.12 %	0.0 %	0.12 %	3.12 %	17.68 %
h	3.13 %	0.12 %	0.0 %	0.12 %	3.12 %	17.68 %	48.71 %
i	0.12 %	0.0 %	0.12 %	3.12 %	17.68 %	48.71 %	84.09 %
j	0.0 %	0.12 %	3.12 %	17.68 %	48.71 %	84.09 %	100.0 %
k	0.12 %	3.12 %	17.68 %	48.71 %	84.09 %	100.0 %	84.09 %
l	3.12 %	17.68 %	48.71 %	84.09 %	100.0 %	84.09 %	48.71 %
m	17.68 %	48.71 %	84.09 %	100.0 %	84.09 %	48.71 %	17.68 %
n	48.71 %	84.09 %	100.0 %	84.09 %	48.71 %	17.68 %	3.13 %
o	84.09 %	100.0 %	84.09 %	48.71 %	17.68 %	3.13 %	0.12 %