The keyvaltable package∗

Richard Grewe
r-g+tex@posteo.net
March 17, 2019

Abstract
The keyvaltable package’s main goal is to facilitate typesetting tables...

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>...easily and yet still looking rather nicely through horizontal rules and alternating row background colors by default;</td>
</tr>
<tr>
<td>(b)</td>
<td>...in a way that by table rows that are specified as lists of key-value pairs, where the keys are column names and the corresponding values are the content of the cell in this row in the respective column;</td>
</tr>
<tr>
<td>(c)</td>
<td>...with re-usable layout for tables of the same type through named table types, of which each has a list of columns as well as further properties such as the background colors of rows; each column, in turn, has a name as well as further properties such as the heading of the column and the alignment of the column’s content.</td>
</tr>
</tbody>
</table>

Contents

1 Usage 2 1.9 Rules Between Rows . 12
1.1 Table Type Definition . 2 2 Use with Other Packages 12
1.2 Typesetting Tables . . . 3 2.1 Named References . . . 12
1.3 Tables of Collected Rows 6 2.2 Computational Cells . . 12
1.4 Setting Global Defaults 6 2.3 Cell Formatting . . . . . 13
1.5 Row Numbering and Labeling . . . . . . . . . . . . 7 |
1.6 Column Spanning . . . . 8 3 Related Packages 13
1.7 Alternative Table Environments . . . . . . . . . . . . 10 |
1.8 Special Row Formatting 10 4 Future Work 13

5 Implementation 14

∗This document corresponds to keyvaltable v1.0, dated 2019/03/17. The package is available online at http://www.ctan.org/pkg/keyvaltable and https://github.com/Ri-Ga/keyvaltable.
1 Usage

We start with a basic usage example. An explanation of the involved macros follows afterwards.

```
\NewKeyValTable{Recipe}{
  amount: align=r;
  ingredient: align=l;
  step: align=X[l];
}
\begin{KeyValTable}{Recipe}
  \Row{amount=150g, ingredient=ice cream, step=put into bowl}
  \Row{amount= 50g, ingredient=cherries, step=heat up and add to bowl}
\end{KeyValTable}
```

The example code first defines a new table type, Recipe, along with the columns that belong to this type. There are three columns (amount, ingredient, and step), whose specifications are separated with semicolons. After the separating :, for each column, the macro configures the column alignment using the align key. The alignments r (right) and l (left) are the standard tabular alignments; the X[l] alignment is provided by the tabu package (see the documentation there), which is used by default for creating the tables.

After the definition of the table type, the example creates a table of the newly defined type. For this, the example uses the KeyValTable environment and the \Row macro, once for each row. The parameter Recipe of the KeyValTable identifies the type of the table. Most notably, each row can now be produced by a single macro in which the content of the individual cells can be specified by pairs such as amount=150g, which puts “150g” into the amount column of the respective row.

The example above already shows that producing a rather nice-looking table – including alternating row colors as well as horizontal rules – without further ado. How the keyvaltable package can be used in the general case and how its visual appearance can be customized is subject of the remainder of this section.

1.1 Table Type Definition

```
\NewKeyValTable[(options)]{(tname)}{(cols)}{(headers)}
```

Table types are defined via the \NewKeyValTable macro, where

- \texttt{(tname)} is the name of the table type,
- \texttt{(cols)} is a semicolon-separated list of individual column specifications, and
- \texttt{(options)}, if provided, specify table type options that override the default table options; they must then be a comma-separated list of \texttt{(property)=(value)} pairs; the list of table options can be found at the introduction of the KeyValTable environment on page 4.
- \texttt{(headers)}, if provided, specifies custom table header rows. This argument is further described in Section 1.6. If this argument is omitted, a single header
row is produced (unless showhead=false is provided as an option) and the
individual headers in this row are determined by \( (colspec) \).

Each column specification is of the form

\[
(\text{colname}): (\text{property})=(\text{value}), (\text{property})=(\text{value}), ...
\]

In such a specification, \( (\text{colname}) \) represents the name of the column. The
\( (\text{property})=(\text{value}) \) pairs configure certain properties of the column. The \( (\text{property}) \)
can be one of the following:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description and Possible Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>This property specifies the alignment of content in the column. The ( (\text{value}) ) can be set to any column alignment understood by the \texttt{tabu} environment of the \texttt{tabu} package. This particularly includes \l, \c, \r, \p, and \X.</td>
<td>\texttt{l}</td>
</tr>
<tr>
<td>default</td>
<td>This property specifies the default value of a cell in this column, i.e., in case that a \texttt{\Row} does not provide content for the cell. By default (i.e., if unset for a column), this is an empty string.</td>
<td>\texttt{(empty)}</td>
</tr>
<tr>
<td>format</td>
<td>This property specifies a formatting macro for content of the cell. The macro can take one argument and is provided with the content of the cell as its argument. By default, the formatting macro takes the content as is but puts a \texttt{\strut} before and after the content (to yield a better vertical spacing).</td>
<td>\texttt{\kvtStrutted}</td>
</tr>
<tr>
<td>head</td>
<td>This property specifies the content of the column’s header row. The default value for this property is the name of the column.</td>
<td>\texttt{(colname)}</td>
</tr>
<tr>
<td>hidden</td>
<td>This property specifies whether a table column shall be displayed or not. The ( (\text{value}) ) for this property can be \texttt{true} (to hide the cell; the default) or \texttt{false} (to display the cell).</td>
<td>\texttt{false}</td>
</tr>
</tbody>
</table>

1.2 Typesetting Tables

The first possibility for typesetting a table using the \texttt{keyvaltable} package, is via
the \texttt{KeyValTable} environment, which the example at the beginning of this section shows. The second possibility is described in Section 1.3.

\begin{KeyValTable}\[\langle \text{options} \rangle\]{\langle \text{tname} \rangle}\end{KeyValTable}

The \texttt{KeyValTable} environment creates a table of type \( (\text{tname}) \). The type \( (\text{tname}) \)
must have been created using \texttt{\NewKeyValTable} before. The environment itself already produces a table with the columns specified for the table type, produces a header row and some horizontal lines, and sets up background colors of rows.

The \( (\text{options}) \) override default configurations, if provided, and must then be a
comma-separated list of \langle property\rangle=\langle value\rangle pairs. The following \langle property\rangle names are available:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description and Possible Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>This property specifies the table’s shape. For \langle value\rangle, the package currently supports multipage and onepage as well as tabular, tabularx, and longtable. In case of multipage, the table may span multiple pages and on each page, the column header is repeated. In case of onepage, the table does not split into multiple pages. The remaining three values use the respective environment for producing the table (see Section 1.7 for the effect).</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>This property specifies the width of the table, if the selected shape supports it (see Section 1.7).</td>
<td></td>
</tr>
<tr>
<td>showhead</td>
<td>This property specifies whether the head row shall be shown. The \langle value\rangle must be a Boolean (i.e., true or false), where true specifies that the head row is shown and false specifies that the head row is not shown.</td>
<td></td>
</tr>
<tr>
<td>showrules</td>
<td>This property specifies whether top and bottom rules as well as a rule below the head row are drawn (true) or not (false).</td>
<td></td>
</tr>
<tr>
<td>headalign</td>
<td>This property specifies the alignment for header cells. If left empty, each header cell receives the same alignment as the respective column.</td>
<td></td>
</tr>
<tr>
<td>headfmt</td>
<td>This property specifies a format to be applied to all header cells. By default, the property is empty, meaning that header cells are formatted. Otherwise, the code provided as value to this key is prepended to the text of the header cells.</td>
<td></td>
</tr>
<tr>
<td>headbg</td>
<td>This property specifies the background color of the head row. The \langle value\rangle must be a single color specification that is understood by the xcolor package. The \langle value\rangle is passed directly to the \texttt{\rowcolor} \texttt{macro}.</td>
<td></td>
</tr>
</tbody>
</table>


\begin{table}
\centering
\begin{KeyValTable}{TabOptions}
\Row{opt=shape, val=onepage}
\Row{opt=showhead, val=false}
\Row{opt=rowbg, val=blue!10..blue!15}
\end{KeyValTable}
\caption{table options demo}
\end{table}

Table 4: table options demo

\begin{table}
\centering
\begin{KeyValTable}{TabOptions2}
\Row{opt=showrules, val=false}
\Row{opt=headbg, val=blue!25}
\Row{opt=headalign, val=c}
\Row{opt=headfmt, val=\bfseries}
\end{KeyValTable}
\caption{Examples for table options}
\end{table}

Figure 1: Examples for table options

<table>
<thead>
<tr>
<th>Key</th>
<th>Description and Possible Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>rowbg</td>
<td>This property specifies the background colors of content rows. The format of the \langle value\rangle for this property must be \langle oddcolor\rangle..\langle evencolor\rangle. The first row after the header is colored with \langle oddcolor\rangle, the second row with \langle evencolor\rangle, and so forth. Both colors must be understood by the xcolor package.</td>
<td>white..black!10</td>
</tr>
</tbody>
</table>

Figure 1 demonstrates the \langle options\rangle in examples.

\Row{\langle options\rangle}\{\langle content\rangle\}

A table row is produced by the \Row macro. The \langle content\rangle must be a comma-separated list of \langle cname\rangle=\langle text\rangle pairs. The \langle cname\rangle identifies a column that was registered for the table type \langle tname\rangle. The \langle text\rangle specifies the content of the cell in the respective column. Each column for which no \langle text\rangle is provided in \langle content\rangle, will result in a cell that is filled with the column’s default value.

The \langle options\rangle argument customizes row properties and is further explained in Section 1.8.
### 1.3 Tables of Collected Rows

As an alternative to producing a table within a single environment, the `keyvaltable` package offers a way to scatter individual rows throughout a document and display the full table later. This method can be useful when table rows are strongly connected to portions of text outside of the table. The method then allows specifying the rows together with the connected text rather than separately in the table environment. Table types for this method are defined via `\NewKeyValTable` as previously described.

\AddKeyValRow{\langle tname \rangle}{\langle content \rangle}

A table row is produced by the `\AddKeyValRow` macro. The \langle tname \rangle identifies the table type and the \langle content \rangle provides the content of the cells in the row. The format of the \langle content \rangle is the same as for the `\Row` macro described in Section 1.2.

\ShowKeyValTable[\langle options \rangle]{\langle tname \rangle}

A table of all the rows defined via `\AddKeyValRow` can be displayed by the `\ShowKeyValTable` macro. The parameters have the same meaning as for the `KeyValTable` environment. This macro resets the list of rows for the specified table type.

\begin{KeyValTableContent}{\langle tname \rangle}
\end{KeyValTableContent}

For simplifying the addition of rows, the `KeyValTableContent` environment can be used. In this environment, the `\Row` macro can be used just like in the `KeyValTable` environment. The only difference is that the `KeyValTableContent` environment does not cause the table to be displayed. For displaying the content collected in `KeyValTableContent` environments, the `\ShowKeyValTable` macro can be used.

The following example demonstrates the use, based on the previously defined `Recipe` table type.

\begin{verbatim}
\AddKeyValRow{Recipe}{amount=3, ingredient=balls of snow, step=staple all 3 balls}
\begin{KeyValTableContent}{Recipe}
\Row{amount=1, ingredient=carrot, step=stick into top ball}
\Row{amount=2, ingredient=coffee beans, step=put diagonally above carrot}
\end{KeyValTableContent}
\ShowKeyValTable{Recipe}
\end{verbatim}

<table>
<thead>
<tr>
<th>amount</th>
<th>ingredient</th>
<th>step</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>balls of snow</td>
<td>staple all 3 balls</td>
</tr>
<tr>
<td>1</td>
<td>carrot</td>
<td>stick into top ball</td>
</tr>
<tr>
<td>2</td>
<td>coffee beans</td>
<td>put diagonally above carrot</td>
</tr>
</tbody>
</table>

### 1.4 Setting Global Defaults

\kvtSet{\langle options \rangle}

The `keyvaltable` package allows changing the default values globally for the parameters of tables and columns. This can be done by using the `\kvtSet` macro.

\begin{verbatim}
\kvtSet{headbg=red, default=?, align=r}
\NewKeyValTable{Defaults}{x; y}
\begin{KeyValTable}{Defaults}
\Row{x=1}
\Row{y=4}
\end{KeyValTable}
\end{verbatim}

| \begin{tabular}{c c c}
  \hline
  \textbf{amount} & \textbf{ingredient} & \textbf{step} \\
  \hline
  1 & ? & 4 \\
  \hline
\end{tabular} | \begin{tabular}{c c}
  \hline
  \textbf{x} & \textbf{y} \\
  \hline
  1 & ? \\
  \hline
\end{tabular} |
Notice the use of the `\NewKeyValTable` in the example. Column properties, including the separating : can be omitted completely, making the definition of a table type very simple.

### 1.5 Row Numbering and Labeling

The mechanism of default column values enables a simple means for automatic row numbering. For this, one can use one of three row counters provided by the `keyvaltable` package: `kvtRow`, `kvtTypeRow`, and `kvtTotalRow`. The counters are explained after the following example, which demonstrates the use for the case of the `kvtRow` counter.

```latex
\NewKeyValTable{Numbered1}{
  line: align=r, head=\#,
  default=\textbf{thekvtRow};
  text: align=l, head=\textbf{Text}}
\begin{KeyValTable}{Numbered1}
\Row{text=First row}
\Row{text=Second row}
\end{KeyValTable}
```

**kvtRow** The `kvtRow` counter counts the row in the current table. The row number excludes the header row of the table. If the table spans multiple pages, the row number also excludes the repeated headings on subsequent pages.

**kvtTypeRow** The `kvtTypeRow` counter counts the rows in the current table and includes the number of rows of all previous tables of the same type.

**kvtTotalRow** The `kvtTotalRow` counter counts the rows in the current table and includes the number of rows of all previous tables produced using the `keyvaltable` package.

Row numbering can easily be combined with row labeling. The following example shows how the `format` column property can be used for this purpose.

```latex
\NewKeyValTable{Labeled}{
  label: align=r, head=\textbf{\#},
  format=\kvtLabel{\kvtRow};
  text: align=l, head=\textbf{Text}}
\begin{KeyValTable}{Labeled}
\Row{text=First row, label=first}
\Row{text=After row \ref{first}}
\end{KeyValTable}
```

The `\kvtLabel` macro shows the current value of the `\counter` – in particular `kvtRow`, `kvtTypeRow`, and `kvtTotalRow` – and sets the `\label` to the value of `\counter`. When using the macro with the `format` property, only the first argument `\counter` must be provided, as the above example shows. The second argument `\label` is provided by the respective cell content.

The `\kvtLabel` macro should work well with packages that change the referencing, like `cleveref` or `varioref`. When using a package that adds an optional argument to the `\label` command (like `cleveref` does), the `\labelopts` can be used to pass an optional argument to `\label`. This feature is demonstrated in Section 2.1.
1.6 Column Spanning

Combining multiple consecutive cells in a row to a single cell (aka column spanning) can serve several purposes. The **keyvaltable** package supports the following purposes:

1. grouping of columns through cells in the table’s header that span all cells in the group and assign a joint title to the group;
2. individual combinations of cells in the table data.

The remainder of this section describes how each of the purposes can be addressed in KeyValTable environments.

**Column groups in table headers** Column groups in table headers can be specified by the *(headers)* parameter of \[NewKeyValTable\]. The following two examples illustrate how this parameter can be used for specifying column groups. The first example produces a single header row in which two columns are grouped with a single header, one column has a normal header, and in which one column is not provided with a header.

```
\NewKeyValTable{ColGroup}{
id: align=r, default=\htekvtRow.;
amount: align=r; ingredient: align=l;
step: align=X[l];
}
\begin{KeyValTable}{ColGroup}
\Row{amount=150g, ingredient=ice cream, step=put into bowl}
\Row{amount= 50g, ingredient=cherries, step=heat up and add to bowl}
\end{KeyValTable}
```

The second example shows how multiple header rows can be specified and, particularly, how the normal column headers can be displayed through the use of “::”.

```
\NewKeyValTable{ColGroup2}{
date: align=r, head=\textbf{date};
min/Berlin: align=r, head=min;
max/Berlin: align=r, head=max;
min/Paris: align=r, head=min;
max/Paris: align=r, head=max;
}
\begin{KeyValTable}{ColGroup2}
\Row{date=01.01.1970, min/Berlin=0\degree C, max/Berlin=...}
\Row{date=01.01.1970, min/Paris=...}
\Row{date=01.01.1970, min/Paris=max/Paris=min/Paris+max/Paris:
  head=\textbf{temperature}\}
\Row{date=01.01.1970, min/Paris+max/Paris=max/Paris:
  head=\textbf{Paris}}
\Row{date=01.01.1970, min/Berlin+max/Berlin+min/Paris=...}
\Row{date=01.01.1970, min/Berlin+max/Berlin=min/Paris=max/Paris:
  head=\textbf{temperature}}
\Row{date=01.01.1970, min/Paris+max/Paris=...}
\end{KeyValTable}
```

The syntax for *(headers)* is as follows:
• \textit{headers} is a list, separated by “\\”, where each element in the list specifies the columns of a single header \textit{row}.

• Each \textit{row}, in turn, is also a list. The elements of this list are separated by “;” (as in the columns specification of \texttt{NewKeyValTable}) and each element specifies a header \textit{cell}.

• Each \textit{cell} is of the form

\[(\textit{col})^{+...+\textit{col}}: \textit{property} = \textit{value}, \textit{property} = \textit{value}, ...\]

where each \textit{col} is the name of a column. The specified header cell then spans each of the listed columns. The columns must be displayed consecutively, though not necessarily in the same order in which they are specified in \textit{cell}.

• The \textit{property} = \textit{value} pairs configure certain properties of the header cell. The \textit{property} can be one of the following:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description and Possible Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>This property specifies the alignment of content in the column. The \textit{value} can be set to any column alignment understood by the \texttt{tabu} environment of the \texttt{tabu} package. This particularly includes l, c, r, p, and X.</td>
<td>\texttt{c}</td>
</tr>
<tr>
<td>head</td>
<td>This property specifies the content of the column’s header row. The default value for this property is the name of the column.</td>
<td></td>
</tr>
</tbody>
</table>

\textbf{Manual column spanning with \texttt{\multicolumn}} The \texttt{\multicolumn} macro can be used for the content of a cell. The effect of this is that a number of subsequent cells are spanned over with the content of the cell. The following example demonstrates the use.

\begin{verbatim}
\NewKeyValTable{MultiCol}{
  col1: align=l;
  col2: align=l;
  col3: align=l;}
\begin{KeyValTable}{MultiCol}
\Row{col1=1, col2=\texttt{\multicolumn{1}{r}{2}}, col3=3}
\Row{col1=1, col2=\texttt{\multicolumn{2}{c}{2+3}}}
\Row{col1=\texttt{\multicolumn{2}{c}{1+2}}, col3=3}
\Row{col1=\texttt{\multicolumn{3}{c}{1+2+3}}}
\end{KeyValTable}
\end{verbatim}

\begin{verbatim}
\begin{tabular}{|c|c|c|}
\hline
\texttt{col1} & \texttt{col2} & \texttt{col3} \\
\hline
1 & 2 & 3 \\
1 & 2+3 & \\
1+2 & 3 & \\
1+2+3 & \\
\hline
\end{tabular}
\end{verbatim}

A word of warning: The \texttt{\multicolumn} macro implicitly constrains the ordering of columns. For instance, in the above example, switching columns 2 and 3 would lead to an error in the second row (because \texttt{col2} is the rightmost column and therefore cannot span two columns) and also in the third row (because \texttt{col1} spans two columns but the second, \texttt{col3} is not empty). Thus, column spanning via \texttt{\multicolumn} should be used with care.
1.7 Alternative Table Environments

Originally, the keyvaltable package uses the tabu package and tabu, resp. longtabu environments for typesetting the actual tables. Through the shape option of tables, the table environment used by keyvaltable tables can be changed. Table 14 on the next page compares the possible shapes/environments with regards to whether they support tables that span multiple pages, whether they support X-type (variable-width) columns, and whether their width can be specified (through the width option). Finally, the table also displays the package(s) that must be loaded manually when the respective shapes are used. Examples can be found in Figure 2 on the following page.

1.8 Special Row Formatting

Through the \texttt{(options)} argument of the \texttt{\Row\[\{options\}\]\{content\}} and the \texttt{\KeyValRow\[\{tname\}\[\{options\}\]\{content\}}} macros, special options of the row can be configured. As with other option arguments of the keyvaltable package, the options must be a comma-separated list of key-value pairs. The following table lists the supported option keys and their meaning.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description and Possible Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>hidden</td>
<td>This property specifies whether the row shall be hidden (true) or not (false). If only hidden is used without a value, this is equivalent to hidden=true.</td>
<td>false</td>
</tr>
<tr>
<td>bg</td>
<td>This property specifies the background color for the particular row. If left empty, the default color as determined by the rowbg option of the table applies.</td>
<td>(empty)</td>
</tr>
<tr>
<td>above</td>
<td>This property specifies extra vertical space above the row. Note that this space is currently not colored with the row’s background color but with the page’s background color. The argument, if provided, is directly passed to \texttt{\vspace}.</td>
<td>(empty)</td>
</tr>
<tr>
<td>below</td>
<td>Analogously to above, this property specifies extra vertical space below the row.</td>
<td>(empty)</td>
</tr>
<tr>
<td>around</td>
<td>This property is a short-hand for setting both, above and below, to the same value.</td>
<td>(empty)</td>
</tr>
</tbody>
</table>

The following example demonstrates the options.

```
\begin{KeyValTable}\{Recipe\}
\Row\{amount=150g, ingredient=ice cream, step=put into bowl\}
\Row\{amount= 50g, ingredient=cherries, step=heat up and add to bowl\}
\Row\{hidden\}\{amount=25g, ingredient=cream, step=decorate on top\}
\Row\{above=1ex, bg=red!10!white\}\{step=serve with a smile\}
\end{KeyValTable}
```

<table>
<thead>
<tr>
<th>amount</th>
<th>ingredient</th>
<th>step</th>
</tr>
</thead>
<tbody>
<tr>
<td>150g</td>
<td>ice cream</td>
<td>put into bowl</td>
</tr>
<tr>
<td>50g</td>
<td>cherries</td>
<td>heat up and add to bowl</td>
</tr>
<tr>
<td></td>
<td></td>
<td>serve with a smile</td>
</tr>
<tr>
<td>shape</td>
<td>environment</td>
<td>multipage</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>onepage</td>
<td>tabu</td>
<td>no</td>
</tr>
<tr>
<td>multipage</td>
<td>longtabu</td>
<td>yes</td>
</tr>
<tr>
<td>tabular</td>
<td>tabular</td>
<td>no</td>
</tr>
<tr>
<td>tabularx</td>
<td>tabularx</td>
<td>no</td>
</tr>
<tr>
<td>longtable</td>
<td>longtable</td>
<td>yes</td>
</tr>
</tbody>
</table>

Packages marked with “*” only need to be loaded if automatic loading is not disabled via the NoTabuPkg option to the keyvaltable package.

Table 14: Comparison of table shapes / environments

```latex
\NewKeyValTable[showrules=false]{ShapeNoX}{
  id: align=l, default=\the\kvtTypeRow;
  l: align=l; c: align=c; r: align=r;}
  \begin{KeyValTable}[shape=\tabular]{ShapeNoX}
  \Row{l=left, c=center, r=right}
  \Row{l=left-2, c=2-center-2, r=2-right}
  \end{KeyValTable}
\begin{KeyValTable}[shape=\longtable]{ShapeNoX}
  \Row{l=left, c=center, r=right}
  \Row{l=left-2, c=2-center-2, r=2-right}
  \end{KeyValTable}
```

```latex
\NewKeyValTable[showrules=false]{ShapeWithX}{
  id: align=l, default=\the\kvtTypeRow;
  l: align=l; X: align=X; r: align=r;}
  \begin{KeyValTable}[shape=\tabularx]{ShapeWithX}
  \Row{l=left, X=expandable, r=right}
  \Row{l=left-2, X=expandable-2, r=2-right}
  \end{KeyValTable}
\begin{KeyValTable}[shape=\onepage]{ShapeWithX}
  \Row{l=left, X=expandable, r=right}
  \Row{l=left-2, X=expandable-2, r=2-right}
  \end{KeyValTable}
\begin{KeyValTable}[shape=\multipage]{ShapeWithX}
  \Row{l=left, X=expandable, r=right}
  \Row{l=left-2, X=expandable-2, r=2-right}
  \end{KeyValTable}
```

Figure 2: Examples for the shape option
1.9 Rules Between Rows

Additional horizontal rules between rows can simply be added by placing the respective rule command between \Row commands. The following example demonstrates this possibility.

\begin{KeyValTable}{Recipe}
\Row{amount=150g, ingredient=ice cream, step=put into bowl}
\Row{amount=50g, ingredient=cherries, step=heat up and add to bowl}
\midrule
\Row{step=serve with a smile}
\end{KeyValTable}

\begin{center}
\begin{tabular}{ccc}
\hline
amount & ingredient & step \\
\hline
150g & ice cream & put into bowl \\
50g & cherries & heat up and add to bowl \\
\hline
\end{tabular}
\end{center}

2 Use with Other Packages

2.1 Named References

The \kvtLabel feature of the keyvaltable package can be used together with named references, as provided by the cleveref package. A name to a row label can be given by using the optional first argument to the \kvtLabel formatting macro and specifying the name to use using crefname. The following example uses “row” for the optional argument and “line” for the displayed name of the reference.

\usepackage{cleveref}
\crefname{row}{line}{lines}
\NewKeyValTable[headfmt=\bfseries]{NamedRef}
{label: align=r, head=Line, format=\kvtLabel[\row]{kvtRow};
text: align=l, head=Text}
\begin{KeyValTable}{NamedRef}
\Row{text=First row, label=one}
\Row{text=After \cref{one}}
\end{KeyValTable}

2.2 Computational Cells

The mechanism of cell formatting macros enables a simple means for automatically computing formulas contained in a column. This can be done, for instance using the xint package and defining a custom format macro (here \Math) that takes over the computation.

\usepackage{xintexpr}
\newcommand\Math[1]{\xinttheexpr trunc(#1, 1)\relax}
\NewKeyValTable{Calculating}
{type; value: align=r, format=\Math}
\begin{KeyValTable}{Calculating}
\Row{type=simple, value=10+5.5}
\Row{type=advanced, value=0.2*(9+2^8)}
\end{KeyValTable}

\begin{center}
\begin{tabular}{ll}
\hline
type & value \\
\hline
simple & 15.5 \\
advanced & 53.0 \\
\hline
\end{tabular}
\end{center}
2.3 Cell Formatting

The `keyvaltable` package can be used together with the `makecell` package in at least two ways:

1. formatting header cells using the head property of columns;
2. formatting content cells using the format property of columns.

The following example gives an impression.

```latex
\usepackage{makecell}
\renewcommand\theadfont{\bfseries}
\renewcommand\theadalign{lt}
\NewKeyValTable{Header}{
  first: head=\thead{short};
  second: head=\thead{two\ lines};}
\begin{KeyValTable}{Header}
  \Row{first=just a, second=test}
\end{KeyValTable}
```

3 Related Packages

I’m not aware of any \LaTeX\ packages that pursue similar goals or provide similar functionality. The following \LaTeX\ packages provide loosely related functionalities to the `keyvaltable` package.

- **ctable**: This package focuses on typesetting tables with captions and notes. With this package, the specification of table content is quite close to normal `tabular` environments, except that the package’s table creation is done via a macro, \ctable.

- **easytable**: This package provides an environment `TAB` which simplifies the creation of tables with particular horizontal and vertical cell alignments, rules around cells, and cell width distributions. In that sense, the package aims at simpler table creation, like `keyvaltable`. However, the package does not pursue separation of content from presentation or re-use of table layouts.

- **tabularkv**: Despite the similarity in the name, this package pursues a different purpose. Namely, this package provides means for specifying table options such as width and height through an optional key-value argument to the `tabularkv` environment. This package does not use a key-value like specification for the content of tables.

4 Future Work

- support for further table environments, such as `xltabular`: The existing code structure should make this not too complicated. Particularly for `xltabular`, a spurious “missing } inserted” error occurs.

- improved row coloring that makes sure that the alternation re-starts on continued pages of a table that spans several pages
5 Implementation

We use \texttt{etoolbox} for some convenience macros that make the code more easily maintainable and use \texttt{xkeyval} for options in key–value form. The \texttt{trimspaces} package is used once for trimming spaces before a string comparison.

\begin{verbatim}
\RequirePackage{etoolbox}
\RequirePackage{xkeyval}
\RequirePackage{trimspaces}
\end{verbatim}

We use \texttt{booktabs} for nice horizontal lines and \texttt{xcolor} for row coloring.

\begin{verbatim}
\PassOptionsToPackage{table}{xcolor}
\RequirePackage{xcolor}
\RequirePackage{booktabs}
\end{verbatim}

5.1 Setting Defaults

\texttt{\kvtSet{⟨options⟩}} set the default options, which apply to all tables typeset with the package.

\begin{verbatim}
\newcommand\kvtSet[1]{\bgroup
  \def\kvt@@presetqueue{\egroup}
  \setkeys[kvt]{defaults}{#1}{}
  \kvt@@presetqueue}
\end{verbatim}

\texttt{\kvt@lazypreset} collects a request for pre-setting \texttt{⟨head keys⟩} in family key \texttt{⟨family⟩}. Using this macro, one can avoid causing problems with using \texttt{xkeyval}’s \texttt{\presetkeys} inside the \texttt{⟨function⟩} defined for a key (e.g., via \texttt{\define@key}). The collected requests can be performed by expanding the \texttt{\kvt@lazypreset} macro.

\begin{verbatim}
\newcommand\kvt@lazypreset[2]{% 
  \appto\kvt@@presetqueue{\presetkeys[kvt]{#1}{#2}{}}}
\end{verbatim}

\texttt{\kvt@addtableprop} adds a new table option, named \texttt{⟨name⟩} and with default value \texttt{⟨default⟩}.

\begin{verbatim}
\newcommand\kvt@addtableprop[2]{% 
  \define@key[kvt]{defaults}{#1}{% 
    \kvt@lazypreset{Table}{#1=##1}}%
    \presetkeys[kvt]{defaults}{#1=#2}{}%
    \define@cmdkey[kvt]{Table}{#1}{}%
    \presetkeys[kvt]{Table}{#1=#2}{}%}
\end{verbatim}

\texttt{\kvt@addchoicetableprop} adds a new table option, named \texttt{⟨name⟩} and with default value \texttt{⟨default⟩} and possible values from the comma-separated list provided by \texttt{⟨choice⟩}.

\begin{verbatim}
\newcommand\kvt@addchoicetableprop[3]{% 
  \define@choicekey[kvt]{defaults}{#1}{#3}{% 
    \kvt@lazypreset{Table}{#1=##1}}%
    \presetkeys[kvt]{defaults}{#1=#2}{}%
    \define@choicekey[kvt]{Table}{#1}{#3}{% 
      \csdef{cmdkvt@Table@#1}{##1}}%
    \presetkeys[kvt]{Table}{#1=#2}{}%}
\end{verbatim}
The `\kvt@addbooltableprop{⟨name⟩}{⟨default⟩}` macro adds a new table option, named ⟨name⟩ and with default value ⟨default⟩ and possible values being booleans (true, false).

```latex
26 \newcommand{\kvt@addbooltableprop}[2][]{%
27 \define@boolkey{kvt}{defaults}{#1}{%
28 \kvt@lazypreset{Table}{#1=##1}}%
29 \presetkeys{kvt}{defaults}{#1=#2}{}%
30 \define@boolkey{kvt}{Table}{#1}%
31 \presetkeys{kvt}{Table}{#1=#2}{}%}
```

The `\kvt@addcolumnprop{⟨name⟩}{⟨default⟩}` macro adds a new column option, named ⟨name⟩ and with default value ⟨default⟩.

```latex
33 \newcommand{\kvt@addcolumnprop}[2][]{%
34 \define@key{kvt}{defaults}{#1}{%
35 \kvt@lazypreset{Column}{#1=##1}}%
36 \presetkeys{kvt}{defaults}{#1=#2}{}%
37 \define@key{kvt}{Column}{#1}{%
38 \csdef{kvt@col@#1@c}{##1}}%
39 \presetkeys{kvt}{Column}{#1=#2}{}%}
```

The `\kvt@addchoicecolumnprop{⟨name⟩}{⟨initial⟩}{⟨default⟩}{⟨choice⟩}` macro adds a new column option, named ⟨name⟩, with initial value ⟨initial⟩, with default argument value ⟨default⟩, and possible values from the comma-separated list provided by ⟨choice⟩.

```latex
41 \newcommand{\kvt@addchoicecolumnprop}[4][]{%
42 \define@choicekey{kvt}{defaults}{#1}{#4}[#3]{%
43 \kvt@lazypreset{Column}{#1=##1}}%
44 \presetkeys{kvt}{defaults}{#1=#2}{}%
45 \define@choicekey{kvt}{Column}{#1}{#4}[#3]{
46 \csdef{kvt@col@#1@c}{##1}}%
47 \presetkeys{kvt}{Column}{#1=#2}{}%}
```

The following are the known column properties and their defaults as well as the known table properties and their defaults.

```latex
49 \kvt@addtableprop{rowbg}{white..black!10}
50 \kvt@addtableprop{headbg}{black!14}
51 \kvt@addbooltableprop{showhead}{true}
```
When adding further shape options below, ensure to also add a corresponding `\kvt@DefineStdTabEnv` counterpart further below in the code.

Here are some key-value table options:

**\kvtTableOpt**

The `\kvtTableOpt{⟨optname⟩}` macro, inside a `KeyValTable` environment, expands to the value of the table option `⟨optname⟩`.

```latex
\newcommand{\kvtTableOpt}[1]{\csname cmdkvt@Table@#1\endcsname}
```

**\kvtStrutted**

The `\kvtStrutted{⟨arg⟩}` macro prefixes and suffixes the argument `⟨arg⟩` with a \strut. When used for formatting cell content, this makes sure that there is some vertical space between the content of a cell and the top and bottom of the row.

```latex
\newcommand{\kvtStrutted}[1]{\strut #1\ifhmode\expandafter\strut\fi}
```

### 5.2 Declaring Key-Value Tables

**\NewKeyValTable**

The `\NewKeyValTable{⟨options⟩}{⟨tname⟩}{⟨colspecs⟩}{⟨headers⟩}` declares a new key-value table type, identified by the given `⟨tname⟩`. The columns of the table type are specified by `⟨colspecs⟩`. The optional `⟨options⟩`, if given, override the default table options for tables of type `⟨tname⟩`.

```latex
\newcommand{\NewKeyValTable}[4]{% 
  \@ifnextchar[{}{}% 
  \kvt@NewKeyValTable{#1}{#2}{#3}{#4}}
```

The `\kvt@NewKeyValTable{⟨options⟩}{⟨tname⟩}{⟨colspecs⟩}{⟨headers⟩}` macro is an auxiliary macro used for parsing the fourth, optional argument of `\NewKeyValTable`.

```latex
\def{\kvt@NewKeyValTable}{⟨options⟩}{⟨tname⟩}{⟨colspecs⟩}{⟨headers⟩}}
```

First initialize the “variables”.

```latex
\csdef{\kvt@options@}{\#1}% \csdef{\kvt@headings@}{\#2}% \csdef{\kvt@NewKeyValTable@}{\#3}% \csdef{\kvt@NewKeyValTable@}{\#4}%
```

The following adds a zero-width column to the left of every table. This column serves the purpose of “holding” the code that `keyvaltable` uses for formatting a row (e.g., parsing `\Row` arguments). This code is partly not expandable. The reason for not putting this code into the first actual column of tables is that this code would prevent `\multicolumn` to be used in the first column. Fixme: Ideally, the whole extra column should be removed through sufficient use of `\noalign` in headers and rows, such that even the presence of `\multicolumn` does not produce errors.
Now parse \(\langle\text{colspecs}\rangle\), a semicolon-separated list of individual column specifications, and add the columns to the table. Each \(\text{\textbackslash do}\{\langle\text{clospec}\rangle\}\) takes the specification for a single column.

\[
\text{\textbackslash def\texttt{do}#1{%,}
\text{\texttt{kvt\textbackslash parsecolspec\{#2\}#1::\texttt{\textbackslash undefined}\%}
\text{\texttt{kvt\texttt{\textbackslash dossvlist}\{#3\}\%}}}
\text{The following terminates the argument list of \texttt{\textbackslash defaultheader}.}
\]

Finally, parse \(\langle\text{headers}\rangle\), also a semicolon-separated list of individual column groups, where \"\\\" marks a new row of column groups. If \(\langle\text{headers}\rangle\) is omitted (or empty), then simply take the result of \texttt{\textbackslash parsecolspec} as the header row.

\[
\text{\textbackslash ifstrempty\{#4\}}
\text{\texttt{\textbackslash kvt\textbackslash parsecolspec\{#1\}\{#2\}\{#3\}\{\texttt{\textbackslash undefined}\%}}}
\text{The following stores the column's properties. The column is only added if the hidden option is not set to true.}
\]

\[
\text{\textbackslash ifcsstring\{#1\}#1\{\texttt{true}\}{}{%,}
\text{\texttt{\textbackslash cseappto\{#1\}\{#2\}\{true\}\{#3\}\{\texttt{\textbackslash undefined}\%}}}
\text{Append the column heading to \texttt{\textbackslash defaultheader}. Hence, the appended tokens are enclosed in curly braces. If no head is specified for the column, \langle\text{cname}\rangle is used for the column head. Otherwise, the head value is used.}
\]

\[
\text{\texttt{\textbackslash ifcsvoid\{#1\}#1\{#2\}\{\texttt{false}\%}}
\text{The following creates the column key that can be used by the row macros to set the content of the column's content in that row.}
\]

\[
\text{\texttt{\textbackslash define@cmdkey[KeyValTable]{#1}{#2}{}{}}}
The \kvt@defaultheader{(head1)}...{(headn)}\@nil macro, takes n header cell titles, (head1) to (headn) and formats them based on the headfmt and headalign options. More precisely, when fully expanded, \kvt@defaultheader yields “\rowcolor{& \hfill \texttt{fmtthead1}} \& \ldots \& \texttt{fmttheadn} “. In the above, \rowcolor{ \langle headbg \rangle}.\newcommand\kvt@defaultheader{% \noexpand\rowcolor{\cmdkvt@Table@headbg} % \kvt@defaultheader@i}
\newcommand\kvt@defaultheader@i[1]{% \kvt@ifnil{#1}{\noexpand\tabularnewline}{% \unexpanded{&}% \ifdefvoid\cmdkvt@Table@headalign{\expandonce\cmdkvt@Table@headfmt\unexpanded{#1}}{\noexpand\multicolumn{1}{\expandonce\cmdkvt@Table@headalign}{\expandonce\cmdkvt@Table@headfmt\unexpanded{#1}}}}% \kvt@defaultheader@i}}\kvt@ifnil macro expands to \texttt{iftrue} if \langle val \rangle is \@nil, and expands to \texttt{iffalse} otherwise. Fixme: The \relax in the following is not fully ideal as it is not swallowed by the \ifx and therefore remains in the macro’s expansion.\newcommand\kvt@ifnil[1]{% \ifx\@nil#1\relax \expandafter\@firstoftwo\else \expandafter\@secondoftwo\fi}
\kvt@HackIntercolSpace macro captures the negative space that cancels out the spacing otherwise caused by the extra column that the package adds.\newcommand\kvt@HackIntercolSpace{% @{\hspace{-.5\arrayrulewidth}}}\kvt@alltables is an etoolbox list containing the names of all tables declared by \NewKeyValTable.\newcommand\kvt@alltables{}5.3 Custom Column Headers
\kvt@parseheadrows{(tname)}{(headers)} macro parses the \langle headers \rangle argument of \NewKeyValTable.
\newcommand\kvt@parseheadrows[2]{% \csdef{kvt@@colgroups@#1}{}% \csdef{kvt@headrowcount@#1}{0}% \bgroup\def\kvt@@parseheadrows{}% Now loop over \langle headers \rangle to split \langle headers \rangle by \\. Append each item, which specifies a single header row, to \kvt@parseheadrows for subsequent parsing by \kvt@parseheadrow. If an item equals the special sequence “::”, then the original header for the columns is added as header row.\def\do##1{%
Increment the header row counter for each \-separated item of \texttt{\textlangle headers\textrangle}.

Finally, escape the inner group and overwrite the headings with the result of the parsing.

The \texttt{\kvt\parheadrow\textlangle tname\textrangle\textlangle colspec\textrangle} macro parses a single header row and appends the resulting table code to \texttt{\kvt\colgroups\textlangle tname\textrangle}.

First parse \texttt{\textlangle colspec\textrangle}, populating the \texttt{\kvt\colgrpof\textlangle colname\textrangle} macros that associate each column with the column group to which the column belongs.

Initialize variables for the subsequent loop. The \texttt{\kvt\tmpgrphd} macro collects the code for the cells of the current header row. The \texttt{\kvt\span} counter specifies how many columns the current cell shall span. Finally, \texttt{\kvt\curgrp} and \texttt{\kvt\lastgrp} hold the name of the group in which the current column and, respectively, previous column are in. Each of the two macros is undefined if there is no such column group.

Next, loop over all displayed (non-hidden) columns stored in \texttt{\kvt\colkeys\textlangle tname\textrangle}. The following \texttt{\do\textlangle colname\textrangle} collects (spanned) columns as specified in \texttt{\textlangle colspec\textrangle}, in the ordering in which the table’s columns are displayed. The spanned columns are stored in \texttt{\kvt\tmpgrphd}.

If the column group has not changed, simply increase the spanning counter.

Otherwise, i.e., if the column group has changed, then conclude the previous column (if there was one) and reset the span to 1 (to count for the column in \texttt{\kvt\curgrp}) and set \texttt{\kvt\lastgrp} to the current one.
consecutive columns, but it is not}\%
\{Compare `|\kvt@@curgrp|' to the column ordering as specified
in `\string\NewKeyValTable{#1}'\}\%
\kvt@@span\@ne \let\kvt@@lastgrp\kvt@@curgrp\%
}\dolistcsloop{kvt@colkeys@#1}\%
\kvt@concludecolumn
Finally, conclude the whole header row and append the row to the overall list of
rows, stored in \kvt@@colgroups@⟨tname⟩, while ending the current \TeX group.
\appto\kvt@@tmpgrphd{\tabularnewline}\%
\edef\do{\noexpand\csappto{kvt@@colgroups@#1}{\noexpand
\unexpanded{\expandonce{\kvt@@tmpgrphd}}}\noexpand\n\noexpand\n\noexpand\rowcolor{\cmdkvt@Table@headbg}}}\%
\expandafter\egroup\do
\kvt@@span
The counter \kvt@@span is used temporarily in macros for counting how many
columns are spanned by column groups.
\newcount\kvt@@span
\kvt@concludecolumn
The \kvt@concludecolumn macro appends a cell, potentially spanning multiple
columns, to the row under construction (which is in \kvt@@tmpgrphd).
\newcommand\kvt@concludecolumn{\%
The following conditional checks whether this is the first column group in the
header row. If this is the case, then the \kvt@@extraalign macro is set to
\kvt@HackIntercolSpace, such that the \multicolumn below does not throw
away this spacing.
\ifdefequal\kvt@@tmpgrphd\@empty\%
{\let\kvt@@extraalign\kvt@HackIntercolSpace}\%
{\let\kvt@@extraalign\@empty}\%
\appto\kvt@@tmpgrphd{&}\%
\ifdefvoid\kvt@@lastgrp{}{\%
\eappto\kvt@@tmpgrphd{\noexpand\multicolumn{\the\kvt@@span}{\noexpand\n\noexpand\n\noexpand\csexpandonce{kvt@@colgrp@align@\kvt@@lastgrp}}\%
\noexpand\csexpandonce{kvt@@colgrp@head@\kvt@@lastgrp}}}\%
Mark the column group as already used and concluded, such that another use of
the same column group can be detected and raise an error.
\cslet{kvt@@colgrpdone@\kvt@@lastgrp}{\@ne}}\%
\kvt@parsehdcolsnap
The \kvt@parsehdcolsnap{⟨tname⟩}{⟨cname⟩}{⟨config⟩}{⟨empty⟩}@undefined macro
parses a single header column (resp. column group), ⟨cname⟩. For a column group,
⟨cname⟩ can consist of multiple, “+”-separated column names.
\def\kvt@parsehdcolsnap#1#2:#3:#4@undefined{%
First link the individual columns of a column group to the group. In this, ensure
that no column is contained in more than one column group.
\def\kvt@colreg#1{%
\inlistcs#1\{kvt@colkeys@#1\}{}
in table type `#1'}{Check the `\string\NewKeyValTable{#1} for
the names of known columns and check `#1' for a typo.}}%
\ifcsmacro{kvt@@colgrpof@##1}
\ifcsmacro{kvt@@colgrpof@##1}{\kvt@error{Column `##1' used in more than one column group}
{Check the fourth, optional argument of `\string\NewKeyValTable
and eliminate multiple occurrences of column `##1'.}}}
\csdef{kvt@@colgrpof@##1}{#2}}%
\kvt@forpsvlist{\kvt@@colreg}{#2}%

Now parse the ⟨config⟩ of the column, resp. column group.
\def\kvt@@colgrp{#2}%
\setkeys[kvt]{ColGroup}{#3}{}

The following defines the options for header cells.
\define@key[kvt]{ColGroup}{head}{%\csdef{kvt@@colgrp@head@\kvt@@colgrp}{#1}}%\define@key[kvt]{ColGroup}{align}{%\csdef{kvt@@colgrp@align@\kvt@@colgrp}{#1}}%\presetkeys[kvt]{ColGroup}{align=c}{}%}

5.4 Row Numbering and Labeling

The following counters simplify row numbering in key-value tables. One can use
a table-local counter (kvtRow), a table-type local counter (kvtTypeRow), and a
global counter (kvtTotalRow).

kvtRow The kvtRow counter can be used by cells to get the current row number. This row
number (in contrast to taburow) does not count table headers. That is, kvtRow
provides the current content row number, even in tables that are spread over
multiple pages.
191 \newcounter{kvtRow}{}

kvtTypeRow The kvtTypeRow counter can be used by cells to get the current row number,
including all previous rows of tables of the same type. This counter works together
with the \kvt@rowcount⟨⟨tname⟩⟩ macro, which keeps track of the individual row
counts of the ⟨iname⟩ type.
192 \newcounter{kvtTypeRow}{}

kvtTotalRow The kvtTotalRow counter can be used by cells to get the current row number,
including all previous KeyValTable tables.
193 \newcounter{kvtTotalRow}{}
194 \setcounter{kvtTotalRow}{0}{}

\kvtLabel\{⟨labelopts⟩\}{⟨counter⟩}{⟨label⟩} macro sets a label, named ⟨label⟩, for the current value of the \LaTeX{} counter named ⟨counter⟩.
195 \newcommand{\kvtLabel}{\kvtLabel}[3][{}]

The following imitates a \refstepcounter in the sense of setting the current label,
but it does not touch the ⟨counter⟩ (in case someone added some custom hooks to
them).
196 \setcounter{kvt@LabelCtr}{\value{#2}}%
Next, define the \textit{label} (if provided) and show the value of \textit{counter}.
\begin{verbatim}
197 \addtocounter{kvt@LabelCtr}{-1}\%
198 \refstepcounter{kvt@LabelCtr}\%

The \texttt{kvt@LabelCtr} counter is an auxiliary counter for setting labels, used by \texttt{\kvtLabel}.
\end{verbatim}

5.5 Key-Value Table Content

\texttt{KeyValTable} The \texttt{KeyValTable[\{options\}][\{tname\}]} environment encloses a new table whose type is identified by the given \textit{tname}. Table options can be overridden by providing \textit{options}.
\begin{verbatim}
203 \newenvironment{KeyValTable}[2][]{\%
204 \bgroup\%
\end{verbatim}

The following saves the row counter value outside the table environment but still in the then-local scope.
\begin{verbatim}
212 \AfterEndEnvironment{KeyValTable}{\%}
213 \csdef{kvt@rowcount@\kvt@@recenttable}{\thekvtTypeRow}\%
\end{verbatim}

\texttt{\kvt@SetOptions} The \texttt{\kvt@SetOptions[\{tname\}][\{options\}]} set the specific table options in the current environment, based on the options for table type \textit{tname} and the specific \textit{options}.
\begin{verbatim}
214 \newcommand{\kvt@SetOptions}[2][]{\%
215 \bgroup\def{\kvt@do}{\egroup\noexpand\%}
216 \setkeys{\kvt}{\{Table\}%
217 \csexpandonce{\kvt@options@\#1},\unexpanded{\#2}}\%
218 \egroup}
\end{verbatim}

5.5.1 Table Environment Code

\texttt{\kvt@StartTabularlike} The \texttt{\kvt@StartTabularlike[\{env\}][\{tname\}][\{bLong\}][\{bTabu\}][\{bWidth\}]} macro begins a table environment for the given table type \textit{tname}. The \textit{env} parameter specifies the concrete environment name. The parameters \textit{bLong}, \textit{bTabu}, and
\langle b\text{Width} \rangle$ are Boolean parameters (expecting value true or value false). They specify whether the table environment supports multi-page tables ($\langle b\text{Long} \rangle$), whether the environment is a tabu environment ($\langle b\text{Tabu} \rangle$), and whether the environment supports specifying the width of the table ($\langle b\text{Width} \rangle$).

The $\texttt{\kvt@recenttable}$ allows the \texttt{\AfterEndEnvironment} hook for KeyValTable to access the most recent table type.

In $\texttt{\kvt@do}$, the start code for the environment, including the header rows, is gathered, with expansion to fill in all the table settings and options.

The $\texttt{\kvt@stepcounters}$ macro increments all row counters by $\langle \text{delta} \rangle$. If $\langle \text{delta} \rangle$ is omitted, $\langle \text{delta} \rangle = 1$.

The $\texttt{\kvt@DefineStdTabEnv}$ macro defines the macros needed for the given $\langle \text{shape} \rangle$ value. If $\langle \text{shape} \rangle$ is omitted,
\langle env \rangle \text{ (the name of the environment to use for the shape) is used as \langle shape \rangle value. The \langle endpatch \rangle parameter expects macro code that shall be run at the beginning of the KeyValTable environment to (locally) patch macros related to the end code of \langle env \rangle for ensuring that the bottom rule, \texttt{\kvt@rule{bottom}}, is displayed. If \langle endpatch \rangle is empty, the rule is displayed via \texttt{\kvt@EndTable\langle shape \rangle}. Otherwise, \texttt{\kvt@EndTable\langle shape \rangle} equals \texttt{end\langle env \rangle}. Environments such as \texttt{tabularx} require the latter to parse for the end of the environment. The parameters \langle bLong \rangle, \langle bTabu \rangle, and \langle bWidth \rangle are the same as for \texttt{\kvt@StartTabularlike}.}

Note: In the future, the macro could automatically add \langle option \rangle to the list of possible values for the shape option.

\begin{verbatim}
252 \newcommand{\kvt@DefineStdTabEnv}{dblargin{\kvt@DefineStdTabEnv@i}}
253 \newcommand{\kvt@DefineStdTabEnv@i}[6]{%
254 \expandafter{\newcommand{\csname kvt@StartTable@#1\endcsname}[1]{% #2}[##1]{#3}{#4}{#5}}%
255 \csedef{\kvt@EndTable@#1}{%
256 \ifstrempty{#6}{\noexpand\kvt@@rule{bottom}}{}%
257 \expandafter{\noexpand\csname end#2\endcsname}}%
258 \ifstrempty{#6}{}{\csdef{\kvt@@patchenvend@#2}{#6}}}
260 \kvt@DefineStdTabEnv{tabular}{false}{false}{false}{}
261 \kvt@DefineStdTabEnv{longtable}{true}{false}{false}{}
262 \kvt@DefineStdTabEnv{tabularx}{false}{false}{true}{{%
263 \preto{\TX@endtabularx}{\toks@\expandafter{\the\toks@\kvt@@rule{bottom}}}}%
264 \kvt@DefineStdTabEnv{tabu}{false}{true}{true}{}
265 \kvt@DefineStdTabEnv{xltabular}{true}{false}{true}{{%
266 \preto{\XLT@ii@TX@endtabularx}{\toks@\expandafter{\the\toks@\kvt@@rule{bottom}}}}%
267 \kvt@DefineStdTabEnv[onepage]{true}{true}{}
268 \kvt@DefineStdTabEnv[multipage]{true}{true}{}\end{verbatim}

The following lines define the macros for the various table shapes / environments.

\begin{verbatim}
270 \newcommand{\kvt@dottedrowcolors}[2]{%
271 \kvt@dottedrowcolors@i{#1}#2@nil}
272 \def{\kvt@dottedrowcolors@i#1#2..#3@nil}{%
273 Since \texttt{\rowcolors} expects its color arguments to specify the odd and even color,
274 we swap arguments depending on the parity of \langle start-row \rangle to ensure \langle color1 \rangle is
275 applied to \langle start-row \rangle.}
276 \ifnumodd{#1}{\rowcolors{#1}{#2}{#3}}%
277 \ifnumeven{#1}{\rowcolors{#1}{#3}{#2}}%
\end{verbatim}

5.5.2 Environment-Independent Parts

The following block declares the known row options.
The `\kvt@AddKeyValRow{⟨pre⟩}{⟨post⟩}{⟨tname⟩}{⟨options⟩}{⟨content⟩}` macro composes a row for the table of type `⟨tname⟩` from the given `⟨content⟩` and `⟨options⟩`. The `⟨content⟩` is a key-value list that specifies the content of the individual cells in the row. The result is returned in macro `\kvt@@row`. The arguments `⟨pre⟩` and `⟨post⟩` are expanded at the very beginning, resp. end of the macro. They allow to control grouping (`\bgroup` and `\egroup`) as well as table placement via `\noalign`.

It’s essential that `⟨pre⟩` above comes even before `\@ifnextchar` and, therefore, cannot be moved into `\kvt@AddKeyValRow@i`. The `\@ifnextchar` is not fully expandable and therefore any `\noalign` (in `⟨pre⟩`) following `\@ifnextchar` would lead to “misplaced `\noalign`” errors.

The `\kvt@AddKeyValRow@ii{⟨post⟩}{⟨tname⟩}{⟨content⟩}` macro mainly processes `⟨content⟩` as well as `⟨options⟩` that have already been parsed by `\kvt@AddKeyValRow@i`. Initialize and first add the \noalign material to the row.

Place the `\everyrow` hook after `\noalign`.

The following loop uses `\do{⟨cname⟩}` to append the content of all columns (in the given format and using the given default value), where each column value is
in \cmdKeyValTable@\{tname\}@\{cname\}. Note that currently the default value is formatted using the given format macro – a design decision.

301 \kvt@span=0\relax
302 \def\do##1{%
First recover the cell content (either the specified value for the row or, if no value is specified for the row, the cell’s default value) without formatting.
303 \ifcsvoid{cmdKeyValTable@#2@##1}
304 {\letcs\kvt@@cell{kvt@col@default@#2@##1}}
305 {\letcs\kvt@@cell{cmdKeyValTable@#2@##1}}%
Separately also already create the formatted content.
306 \def\kvt@fmtcell{{\csexpandonce{kvt@col@format@#2@##1}{%}
307 {\expandonce\kvt@@cell}}%
Next, check whether a column-spanning cell is active (\kvt@span > 0). If this is the case, ensure that if the raw cell content in the current column is empty, then formatting does not make the cell non-empty and, thereby, cause errors with the active column-spanning cell.
308 \ifnumgreater{\kvt@span}{0}
309 \advance{\kvt@span}\m@ne
310 {\ifstrempty{\kvt@@cell}{\def\kvt@@fmtcell{}}{}%}
311 \appto{\kvt@@row}{&}%
Now check whether the cell itself spans multiple columns.
312 {\expandafter\kvt@CheckMulticolumn\kvt@@cell
313 \relax\relax\relax\relax\undefined(#2){##1}%
314 {\expandafter\appto\expandafter\kvt@row\expandafter{\kvt@fmtcell}%
315 \dolistcsloop{kvt@colkeys@#2}}%
Finally, add the concluding newline for the row as well as the vertical space after the row, if requested.
316 \appto{\kvt@row}{\tt\tabularnewline}%
317 {\ifdefvoid{cmdkvt@Row@below}{%}
318 \appto{\kvt@row}{\noexpand\noalign{\vspace{%}
319 \expandonce{cmdkvt@Row@below}}}%
At the very end of the expansion text, put \texttt{(post)}.
320 #1}

\kvt@everyrow The \kvt@everyrow\{\texttt{code}\} registers \texttt{code} to be included in the first (invisible) cell of every row.
321 \newcommand{\kvt@everyrow}[1]{\def{\kvt@everyrow[#1]}{\kvt@everyrow[#1]}}
322 \newcommand{\kvt@everyrow}{}

Initialize the hook for every row to increment the row counters.
323 \kvt@everyrow\{\kvt@stepcounters\%

\kvt@CheckMulticolumn The \kvt@CheckMulticolumn\{\texttt{arg1}\}\{\texttt{arg2}\}\{\texttt{arg3}\}\{\texttt{arg4}\}\undefined\{\texttt{tname}\}\{\texttt{colname}\} macro checks whether a cell’s initial content (captured by \texttt{arg1} to \texttt{arg4}), starts a multi-column cell. If this is the case, the macro records the arguments to \texttt{\multicolumn} for use by \kvt@AddKeyValRow. In this case,
\( \langle \text{arg1} \rangle = \langle \text{multicolumn} \rangle, \langle \text{arg2} \rangle = \langle n \rangle \) (number of columns to span), \( \langle \text{arg3} \rangle = \langle \text{format} \rangle \) (column alignment), and \( \langle \text{arg4} \rangle = \langle \text{item} \rangle \) (the content of the cell).

324 \def\kvt@CheckMulticolumn#1#2#3#4\@undefined#5#6{
325 \ifx#1\multicolumn
326 \kvt@@span=#2\relax \advance\kvt@@span\m@ne
327 \edef\kvt@@fmtcell{\unexpanded{\multicolumn{#2}{#3}}%}
328 \csexpandonce{kvt@col@format@#5@#6}{\expandonce{#4}}}%
329 \fi}

5.6 Collecting Key-Value Table Content

The \ShowKeyValTable[\langle options \rangle]{\langle tname \rangle} macro shows a table of type \langle tname \rangle with given \langle options \rangle. The rows must have been collected using \Row in KeyValTableContent environments or using \AddKeyValRow.

330 \newcommand\ShowKeyValTable[2][{}]{%
331 \begin{KeyValTable}[\#1]{\#2}%
332 \csuse{kvt@rows@#2}%
333 \end{KeyValTable}%
334 \csdef{kvt@rows@#2}{}%}

\AddKeyValRow The \AddKeyValRow[\langle tname \rangle][\langle options \rangle][\langle content \rangle] adds a row with a given \langle content \rangle to the existing content for the next table of type \langle tname \rangle that is displayed with \ShowKeyValTable. The \langle content \rangle and \langle options \rangle parameters are the same as with \kvt@AddKeyValRow. The resulting row (\kvt@row) is globally appended to \kvt@rows[\langle tname \rangle].

335 \newcommand\AddKeyValRow[1][]{%
336 \kvt@AddKeyValRow
337 \bgroup
338 \csxappto{kvt@rows[\#1]}{\expandonce{\kvt@row}}\egroup
339 {\#1}}

KeyValTableContent The KeyValTableContent[\langle tname \rangle] environment acts as a container in which rows can be specified without automatically being displayed. In this environment, rows can be specified via the \Row[\langle content \rangle] macro, which is supposedly shorter than using \AddKeyValRow[\langle tname \rangle][\langle content \rangle].

340 \newenvironment{KeyValTableContent}[1][]{%
341 \def\Row{\AddKeyValRow[\#1]}%}

5.7 Package Options

The tabu is used by default for typesetting the tables, additionally with longtable for tables that can span multiple pages. If the default packages are never used or the tabu package shall be loaded manually, the noTabuPkg option can be used.

342 \define@boolkey[kvt]{PackageOptions}[kvt@@]{noTabuPkg}[true]{}}
Next, set default package options and process them.

\ExecuteOptionsX[kvt]<PackageOptions>{%
 noTabuPkg=false,
}%
\ProcessOptionsX[kvt]<PackageOptions>\relax

Finally, implement the outcome of the options parsing.

\ifbool{kvt@@noTabuPkg}{}{%
\RequirePackage{longtable,tabu}}

5.8 Auxiliary Code

\kvt@dossvlist
The \kvt@dossvlist\{⟨list⟩\} macro parses a semicolon-separated list and runs \do⟨item⟩ for every element of the list.
\DeclareListParser{\kvt@dossvlist}{;}

\kvt@forpsvlist
The \kvt@forpsvlist\{⟨handler⟩\{⟨list⟩\}} parses a ‘+’-separated list.
\DeclareListParser*{\kvt@forpsvlist}{+}

\kvt@dobrklist
The \kvt@dobrklist\{⟨list⟩\} parses a ‘\\’-separated list.
\DeclareListParser{\kvt@dobrklist}{\\}

\kvt@error
\kvt@warn
\newcommand\kvt@error[2]{\PackageError{keyvaltable}{#1}{#2}}
\newcommand\kvt@warn[1]{\PackageWarning{keyvaltable}{#1}}

Change History

v0.1
General: Initial version ........... 1

v0.2
\NewKeyValTable: Added table-type options ........... 16
\kvtLabel: Added macro for row labeling ............. 21
General: Added “shape” table option .................. 15

v0.3
\kvt@StartTabularlike: Added showhead option .......... 23
\kvtLabel: Robustified for use with, e.g., cleveref ........... 21
\kvtStrutted: Fix for cells with vertical material .......... 16

v0.3b
General: Package author’s name change .................. 1

v1.0
\NewKeyValTable: Added optional headers argument ........ 16
Added zero-width column for \multicolumn .............. 16
\kvtAddKeyValRow: Added [⟨options⟩] .................... 25
\kvtAddKeyValRow@ii: Added \multicolumn support .......... 26
\kvt@StartTabularlike: Added width option ............. 23
Implemented showrules option 23
General: Enabled default “true” for “hidden” ............ 15
Index

Symbols
\@dblarg .................. 252
\@empty ................. 139, 161, 163, 290
\@firstoftwo .............. 113
\@finextchar .............. 67, 284
\@ne ..................... 144, 151, 171
\@nil .................... 82, 112, 271, 272
\@secondoftwo ............ 114
\@undefined ................ 80, 87, 137, 172, 313, 324
\\ ........................ 351

A
\AddKeyValRow ............ 6, 335, 341
\addtocounter ......... 197, 249, 250, 251
\advance ................... 144, 309, 326
\AfterEndEnvironment ...... 212
\appto 12, 126, 129, 130, 154, 164, 300, 311, 314, 316
\arrayrulewidth ........... 116

B
\begin ........................ 331
\bgroup 7, 121, 136, 204, 206, 215, 227, 337

C
\cmdkvt@Row@above 281, 295, 297
\cmdkvt@Row@below 281, 317, 319
\cmdkvt@Row@bg ........... 298, 299
\cmdkvt@Table@headalign .. 106, 108
\cmdkvt@Table@headbg .. 101, 156
\cmdkvt@Table@headfmt .. 107, 109
\cmdkvt@Table@rowbg .. 232, 245
\cmdkvt@Table@shape .. 208, 210
\cmdkvt@Table@width .. 236, 237
counters:
\kvtRow .................. 7
\kvtTotalRow ............ 7
\kvtTypeRow ............. 7
\csappto .................. 82, 93, 155
\csdef 24, 31, 38, 46, 71, 72, 74, 75, 76, 77, 84, 119, 120, 182, 187, 189, 213, 259, 334
\cseappto ................ 91, 94, 127
\csedef .................. 73, 130, 256
\csexpandonce .......... 91, 94, 128, 169, 170, 217, 239, 306, 328
\cslet .................. 171
\csletcs .................. 134
\csname ................... 64, 233, 254, 258
\csuse 131, 201, 208, 210, 222, 224, 226, 230, 242, 332
\csxappto ................ 338

D
\DeclareListParser ....... 349, 350, 351
\define@boolkey ....... 27, 30, 277, 342
\define@choicekey ..... 20, 23, 42, 45
\define@cmdkey ......... 17, 97, 276, 278, 279, 280
\define@key ........... 14, 34, 37, 186, 188
\do 79, 123, 137, 142, 155, 158, 302
\dolistcsloop ............ 152, 315
\eappto .................. 166, 296, 299, 318
\egroup 8, 133, 158, 206, 211, 215, 227, 338
\else .......................... 113
\end ........................ 333
\endcsname ............. 64, 233, 254, 258
\endhead .................. 246
environments:
KeyValTable ............ 3, 203
KeyValTableContent . 6, 340
\ExecuteOptionsX ........ 343
\expandafter .......... 65, 113, 114, 125, 133, 145, 158, 206, 233, 254, 258, 263, 266, 300, 312, 314

F
\fi .................. 65, 114, 145, 329

H
\hspace ................... 116

I
\ifbool 221, 228, 229, 234, 235, 241, 244, 246, 289, 347