The `rccol` package*

Eckhart Guthöhrlein†

Printed November 12, 2004

Abstract

This package provides a new 'right-centered' tabular column type `R`. Numbers are centered with respect to other entries, but flushed right to each other so that corresponding digits are vertically aligned. Furthermore, numbers can be rounded to the desired precision.

1 Introduction

It is often desirable to produce tables like the following.

Value 1	Value 2	Value 3
2345,765	1,6	21,00
45,130	0,2	1,65
543,990	2,0	9,90

You can see what is ment by 'right-centered' columns: The center of the largest number is centered with the title, while the numbers are aligned with their corresponding decimal places. This is, at least to the author's opinion, the most aesthetic way to typeset numbers in a table.

This package provides a general and easy-to-use solution for this problem. In addition, the numbers given as alignment entries can be rounded to a given precision and formatted in several ways.

The next section deals with the various possibilities to typeset numbers in tables with and without this package. After that, the additional facilities of `rccol` and their usage are described.

2 Numbers in tables

The typesetting of numbers in tables needs special care. Most importantly, the vertical alignment of corresponding decimal places in a column should be ensured. Furthermore, it is desirable to achieve a proper placement (centering normally looks best) with respect to the column title and other entries which are not numbers.

First, let us discuss the possibilities available with and without this package, a small example for each one is given below.

*This file has version number v1.2b dated 2004/11/12.

†Send comments or bug-reports to the author via e-mail `<e_w_g@web.de>`.

- Centered columns. This does not allow a vertical alignment of corresponding digits, except if all numbers have the exactly the same number of digits before and after the decimal sign.
- Flushed right columns. If the numbers have the same number of digits after the decimal sign, the alignment of corresponding digits will be correct. However, in any case the placement with respect to the title will not look good.
- Flushed left columns. Similar restrictions as for flushed right ones.
- Use two columns with the decimal sign inserted between them, e. g. `r@{,}l`, and typeset the title with `\multicolumn{2}{c}{Title}`. This provides correct alignment for corresponding digits, but no proper placement with respect to the title if it is wider than the numbers.
- Use the `dcolumn` package [1]. With version 1.03a, the syntax has been extended so that both correct alignment of digits and centering to other entries is possible.
- Use the `rccol` package. Both correct alignment and centering to other entries are possible. Furthermore, other features like rounding of numbers are provided.

Let us have a look at an example of these possibilities. The following table shows the first four ones.

Column type `c`	Column type `r`	Column type `l`	Spaltentyp `r@{,}l`
100,5	100,5	100,5	100,5
7,93	7,93	7,93	7,93
17,123	17,123	17,123	17,123
3141,592	3141,592	3141,592	3141,592
13	13	13	13
0,00017	0,00017	0,00017	0,00017

Obviously, none of the results is satisfying. The following table shows the result with the `dcolumn` and with the `rccol` package.

Column type `D{,}{,}{4.5}`	Column type `R{4}{5}`
100,5	100,5
7,93	7,93
17,123	17,123
3141,592	3141,592
13	13
0,00017	0,00017

None of the special features of `rccol` have been used in this example. As you can see, the final results of the two packages are identical in this case. So, as long as you do not need the features described in the next section, you can work equally well with either of the two packages.

3 User interface

The new column type `R` is used as normally in the preamble of a LaTeX table. It takes two mandatory arguments: the number of digits before and the number of digits after the decimal sign. The example table above has been produced by

```
\begin{tabular}{R{4}{3}R{1}{1}R{2}{2}}
\hline
\multicolumn{1}{c}{Value 1} & \multicolumn{1}{c}{Value 2} &
\multicolumn{1}{c}{Value 3}\tabularnewline
\hline
2345,765 & 1,6 & 21   \tabularnewline
  45,130 & 0,2 &  1,65\tabularnewline
 543,99  & 2   &  9,9 \tabularnewline
\hline
\end{tabular}
```

`\rcRoundingtrue` `\rcRoundingfalse`

By default, numbers are rounded to the given number of digits after the decimal sign, and trailing zeros are appended if they are missing in the input file. This behaviour can be toggled with the switch `\rcRoundingtrue`/`\rcRoundingfalse`, obeying the normal grouping rules, and with the package options `rounding` or `norounding` (see below.) With `\rcRoundingfalse`, the above input results into the following result. Numbers are not rounded or altered in any way; trailing zeros are not appended.

Value 1	Value 2	Value 3
2345,765	1,6	21
45,130	0,2	1,65
543,99	2	9,9

The `R` column specifier also takes optional arguments (thanks go to Rolf Niepraschk for suggesting this and encouraging me). The optional parameters determine the characters used for the decimal sign in the input (`.tex`) and the output file (`.dvi` or perhaps `.pdf`), so the full syntax for the `R` column is

`\begin{tabular}{...R[`⟨*character*⟩`][`⟨*character*⟩`]{`⟨*number*⟩`}{`⟨*number*⟩`}...}`

For example, `R[,][.]{1}{1}` will produce a column centered for numbers with one digit before and one after the decimal sign, which is a comma in the input file, but a point in the output. If only one optional parameter is given, this character will be used for input and output, so `R[!]{1}{1}` is equivalent to `R[!][!]{1}{1}`.

Another optional parameter is a `-` character after the `R` as in `R-{4}{0}`. In this case, the width of a preceding negative sign (to be exact, the width of `$-$`) will be taken into account when centering.

`\rcDecimalSign` `\rcDecimalSignInput` `\rcDecimalSignOutput`

The decimal sign can be changed with the commands `\rcDecimalSign`, affecting both input and output. It can be changed separately for input and output with `\rcDecimalSignInput` and `\rcDecimalSignOutput`. These commands obey the normal grouping rules. Furthermore, there are some package options for setting the defaults (see below). To give an example, let's typeset the above table once again, but with `\rcDecimalSignOutput{\ensuremath{\cdot}}`.

Value 1	Value 2	Value 3
2345·765	1·6	21·00
45·130	0·2	1·65
543·990	2·0	9·90

4 Package Options

The `rccol` package can be loaded as any LaTeX package via

`\usepackage[`⟨*option list*⟩`]{rccol}`

with the following options.

rounding Enable rounding, equivalent to `\rcRoundingtrue` at the beginning of the document. This is the default, so explicitly using this option is not necessary.

norounding Disable rounding by default, equivalent to `\rcRoundingfalse` at the beginning of the document.

comma Use a comma as the decimal sign, equivalent to `\rcDecimalSign{,}` at the beginning of the document. This is the default, so it is normally not necessary to explicitly give this option.

german See comma.

english Use a point as the decimal sign, equivalent to `\rcDecimalSign{.}` at the beginning of the document.

USenglish See english.

point See english.

5 Additional hints and remarks

- Any tokens in the alignment entries which are not digits are put in front of the alignment entry. This can be used to switch to bold face or to add something like '>' before the number. If the tokens have any width, it is your job to correct this, e. g. by putting them into a `\makebox[0pt][r]`. You must enclose commands with arguments inside an extra pair of curly braces to keep them together, otherwise error messages will occur. Example:

```
\begin{tabular}{R{4}{3}R{1}{1}R{2}{2}}
\hline
\multicolumn{1}{c}{Value 1} & \multicolumn{1}{c}{Value 2} &
\multicolumn{1}{c}{Value 3}\tabularnewline
\hline
2345,765 & {\mathversion{bold}} 1,6   & 21    \tabularnewline
  45,130 & 0,2                       &  1,65 \tabularnewline
 543,99  & {\makebox[0mm][r]{$>{}$}}2 &  9,9  \tabularnewline
\hline
\end{tabular}
```

Result:

Value 1	Value 2	Value 3
2345,765	1,6	21,00
45,130	0,2	1,65
543,990 >	2,0	9,90

- Negative values are allowed as arguments for the `R` column. This makes it possible to round to multiples of ten, for example, by giving '`-2`' as the second mandatory argument. In general, rounding is performed to a precision of 10^{-n}, if n is the second mandatory argument of `R`.
- The first mandatory argument of the `R` column specifier does not need to be the real number of digits before the decimal point. You can shift the output to the left or to the right by changing this value if you are not satisfied by the centered result. Negative values are legal.
- As with any macro, single token arguments can be given without braces. This means that '`R{4}{3}`' is equivalent to '`R43`'. Maybe you will prefer the shorter alternative.

6 Requirements

The `rccol` package needs the `array` [2] and the `fltpoint` [3] packages.

7 Final Remarks

There are some open problems and desirable features concerning this package, e. g. formatted output of numbers. However, I have ceased using LaTeX, so I don't think that I will make further extensions to `rccol`. You are free to do it yourself – the license is LPPL. Furthermore, you might be interested in the following packages, available from CTAN if not already included in your distribution:

- `dcolumn` of course,
- `numprint` for formatting of numbers,
- `fp` for calculations inside TeX.

8 Implementation

8.1 DriverFile

The `docstrip` utility can be used to generate a driver file producing the documentation.

```
1 ⟨*driver⟩
2 \documentclass{ltxdoc}
3 \usepackage{dcolumn}
4 \usepackage{rccol}
5 \AtBeginDocument{\DeleteShortVerb{\|}} % the default is not needed
```

```
%\OnlyDescription    % typeset only user documentation
\AlsoImplementation  % typeset implementation details
\EnableCrossrefs    % say \DisableCrossrefs if index is ready
%\DisableCrossrefs
\CodelineIndex      % index using codeline numbers
\RecordChanges      % keep change log
%\OldMakeindex       % use if your Makeindes is pre-v2.9
\begin{document}
   \DocInput{rccol.dtx}
\end{document}
⟨/driver⟩
```

8.2 LaTeX package info and options

```
⟨*rcmain⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{rccol}[2004/11/12 v1.2b right-centered columns]
\DeclareOption{rounding}{\AtBeginDocument{\rcRoundingtrue}}
\DeclareOption{norounding}{\AtBeginDocument{\rcRoundingfalse}}
\DeclareOption{USenglish}{\AtBeginDocument{\rcDecimalSign.}}
\DeclareOption{english}{\AtBeginDocument{\rcDecimalSign.}}
\DeclareOption{point}{\AtBeginDocument{\rcDecimalSign.}}
\DeclareOption{german}{\AtBeginDocument{\rcDecimalSign,}}
\DeclareOption{ngerman}{\AtBeginDocument{\rcDecimalSign,}}
\DeclareOption{austrian}{\AtBeginDocument{\rcDecimalSign,}}
\DeclareOption{naustrian}{\AtBeginDocument{\rcDecimalSign,}}
\DeclareOption{comma}{\AtBeginDocument{\rcDecimalSign,}}
\ProcessOptions*\relax
```

8.3 Require packages

To work properly, `rccol` needs the `array` and the `fltpoint` packages. An up-to-date version of `fltpoint` is necessary.

```
\RequirePackage{array}
\RequirePackage{fltpoint}[2004/11/12]
```

8.4 Registers & Co

`\rc@preskip` `\rc@postskip` Allocate dimen registers `\rc@preskip` and `\rc@postskip`.

```
\newdimen\rc@preskip
\newdimen\rc@postskip
```

`\rc@digits` `\rc@preothertoks` `\rc@postothertoks` Allocate token registers `\rc@digits`, `\rc@preothertoks` and `\rc@postothertoks`.

```
\newtoks\rc@digits
\newtoks\rc@preothertoks
\newtoks\rc@postothertoks
```

`\ifrcRounding` Initialize `\ifrcRounding` as `true`.

```
\newif\ifrcRounding
\rcRoundingtrue
```

8.5 Definition of the R column

\NC@rewrite@R
\rc@rewrite@
\rc@rewrite@@
\rc@rewrite@@@

The \newcolumntype mechanism of array.sty is used to set up the new column type. In order to have optional parameters, the rewrite macro is redefined to take one or two optional parameters, read by \rc@rewrite@ and \rc@rewrite@@. The optional parameters are used to change the decimal sign inside the column if this is desired. An appropriate \fpDecimalSign is put into the token registers \rc@digits (input) and \rc@preothertoks (output), which are somewhat misused for that purpose. The macro \rc@rewrite@@@ reads the two mandatory arguments (the number of digits before and after the decimal point) and appends everything to the preamble so far. At the end \NC@find is called to continue the rewriting process (see [2] for further details).

```
\newcolumntype{R}{}
\def\NC@rewrite@R{%
   \rc@digits{}%
   \rc@preothertoks{}%
   \@ifnextchar-{\rc@rewrite}{\rc@rewrite\relax}}
\def\rc@rewrite#1{%
   \ifx#1-%
      \def\rc@withsign{{\setbox0\hbox{$-$}\hskip\wd0}}%
   \else
      \def\rc@withsign{}%
   \fi
   \@ifnextchar[{\rc@rewrite@}{\rc@rewrite@@@}}
\def\rc@rewrite@[#1]{%
   \rc@digits{\rcDecimalSignInput{#1}}%
   \@ifnextchar[{\rc@rewrite@@}%
      {\rc@preothertoks{\rcDecimalSignOutput{#1}}\rc@rewrite@@@}}
\def\rc@rewrite@@[#1]{%
   \rc@preothertoks{\rcDecimalSignOutput{#1}}\rc@rewrite@@@}
\def\rc@rewrite@@@#1#2{%
   \edef\rc@rewrite@scratch{\the\@temptokena>{\the\rc@digits
      \rc@withsign\noexpand\rc@begin}c<{\noexpand\rc@end
      \the\rc@preothertoks\noexpand\rc@write{#1}{#2}}}
   \@temptokena\expandafter{\rc@rewrite@scratch}%
   \NC@find}
```

8.6 Macros for reading and processing the numbers

\rc@begin

The macro \rc@begin is inserted in front of every R alignment entry. It initializes the token registers \rc@digits and \rc@othertoks and calls \rc@getnexttok to read the following tokens.

```
\def\rc@begin{%
   \rc@digits={}%
   \rc@preothertoks={}%
   \rc@postothertoks={}%
   \let\rc@othertoks\rc@preothertoks
   \rc@getnexttok
}
```

\rc@getnexttok
\rc@savedigit
\rc@savetok

The macro \rc@getnexttok is used to read the alignment entry token after token. The macros \rc@savedigit and \rc@savetok are used to store digits and other tokens in the respective token register (\rc@digits or \rc@othertoks).

`\rc@savedigit` switches `\rc@othertoks` form `\rc@preothertoks` to `\rc@postothertoks`. After processing the token, `\rc@next` is called which has been `\let` equal to the appropriate next action.

```
\def\rc@savedigit#1{\let\rc@othertoks\rc@postothertoks
   \edef\rc@scratch{\the\rc@digits#1}%
   \rc@digits=\expandafter{\rc@scratch}}
\def\rc@savetok#1{\rc@othertoks=\expandafter{\the\rc@othertoks#1}}
\def\rc@getnexttok#1{%
```

First, check if `\tabularnewline` or `\rc@end` follows.

```
   \ifx#1\tabularnewline
      \let\rc@next\tabularnewline
   \else
      \ifx#1\rc@end
         \let\rc@next\rc@end
      \else
```

If not, further tokens have to be read. If the current token is allowed by the syntax of fp numbers, save it in `\rc@digits`, otherwise and if it is neither `\ignorespaces` nor `\unskip`, save it in `\rc@othertoks`.

```
         \let\rc@next\rc@getnexttok
         \ifx#10\rc@savedigit#1\else\ifx#11\rc@savedigit1\else\ifx#12%
            \rc@savedigit2\else\ifx#13\rc@savedigit3\else\ifx#14%
            \rc@savedigit4\else\ifx#15\rc@savedigit5\else\ifx#16%
            \rc@savedigit6\else\ifx#17\rc@savedigit7\else\ifx#18%
            \rc@savedigit8\else\ifx#19\rc@savedigit9\else%
            \if\noexpand#1\fp@decimalsign\rc@savedigit\fp@decimalsign%
            \else\ifx#1-\rc@savedigit-\else\ifx#1+\rc@savedigit+%
            \else\ifx#1e\rc@savedigit e\else\ifx#1E\rc@savedigit E%
         \else
            \ifx#1\ignorespaces\else
               \ifx#1\unskip\else
                  \rc@savetok{#1}%
               \fi
            \fi
         \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
      \fi
   \fi
```

Call `\rc@next`.

```
   \rc@next
}
```

`\rc@end` This macro marks the end of the alignment entry and reads the contents of `\rc@digits` into a `fltpoint` register.

```
\def\rc@end{\fpDecimalSign{\rc@decimalsigninput}%
   \fp@regread@raw{rc@temp}{\the\rc@digits}}%
```

`\rc@output` The macro `\rc@output` is called by `\rc@write`. It takes one argument: the text to be typeset. It checks if the material is to be visible, and in this case if negative space has to be inserted, or if only positive space with the same width as the text has to be output. Any text to be finally output is appended to `\rc@number`.

```
\def\rc@output#1{%
```

Set a box with the text in it to get its width.

```
    \setbox0=\hbox{#1}%
```

If the current position is greater than the first digit of the number, append nothing and increase `\rc@preskip`.

```
    \ifnum\fp@loopcount>\ar@getul{rc@temp}\relax
       \def\rc@toappend{}%
       \advance\rc@preskip by \wd0
```

Similarly, if the last digit of the number has a position greater than the actual one, do the following: if rouinding is enabled, append `#1`, if not, append nothing and increase `\rc@postskip`.

```
    \else
       \ifnum\fp@loopcount<\ar@getll{rc@temp}\relax
          \ifrcRounding
             \def\rc@toappend{#1}%
          \else
             \def\rc@toappend{}%
             \advance\rc@postskip by \wd0
          \fi
```

Otherwise, append `#1`. If the position is greater than the desired number of digits before the decimal sign, decrease `\rc@preskip`. Accordingly, decrease `\rc@postskip` if necessary.

```
       \else
          \def\rc@toappend{#1}%
          \ifnum\fp@loopcount>\rc@outmax\relax
             \advance\rc@preskip by -\wd0
          \else
             \ifnum\fp@loopcount<\rc@outmin\relax
                \advance\rc@postskip by -\wd0
             \fi
          \fi
       \fi
    \fi
```

Append to `\rc@number` what is to be appended.

```
    \edef\rc@number{\rc@number\rc@toappend}%
} % end \rc@output
```

`\rc@write` The macro `\rc@write` takes two arguments: the number of digits before and after the decimal sign.

```
\def\rc@write#1#2{%
```

Initialize `\rc@preskip` and `\rc@postskip`. Initialize `\rc@number`.

```
    \rc@preskip=0pt
    \rc@postskip=0pt
    \def\rc@number{}%
```

At the beginning, the tokens to be inserted in front of the number, saved in `\rc@preothertoks`. This gives the user the possibility to change parameters locally for one alignment entry.

```
    \the\rc@preothertoks
```

If rounding is enabled, do it.

```
\ifrcRounding
   \fp@reground{rc@temp}{-#2}%
\fi
\fpDecimalSign{\rc@decimalsignoutput}%
```

Determine start and end position for the output loop. Save the desired number of digits in `\rc@outmax` and `\rc@outmin`. The loop starts at the desired maximal position or at the upper limit of the number, depending on which is greater. The final value is determined accordingly. The correction of -1 for `#1` is necessary because the first digit of a number with n digits before the decimal sign means 'multiple of $10^{(n-1)}$'.

```
\fp@tempcount=#1\relax
\advance\fp@tempcount by -1
\edef\rc@outmax{\number\fp@tempcount}%
\edef\rc@outmin{\number-#2}%
\fp@loopcount=\rc@outmax\relax
\fp@settomax{\fp@loopcount}{\fp@loopcount}{\ar@getul{rc@temp}}%
\fp@settomin{\fp@loopcountii}{\rc@outmin}{\ar@getll{rc@temp}}%
```

The main loop follows. For output, `\rc@output` is used to append anything to `\rc@number`. The thousand separator is inserted in the appropriate places, that means if $(n \bmod 3 = 2) \vee ((n \bmod 3 = -1) \wedge (n \neq -1))$ with n standing for the value of `\fp@loopcount`. The decimal sign is inserted in front of the digit if $n = -1$.

```
\loop
   \fp@modulo{\fp@loopcount}{3}%
   \ifnum\fp@param=2
      \rc@output{\fp@thousandsep}%
   \else
      \ifnum\fp@param=-1
         \ifnum\fp@loopcount=-1\else
            \rc@output{\fp@thousandsep}%
         \fi
      \fi
   \fi
   \ifnum\fp@loopcount=-1
      \rc@output{\fp@decimalsign}%
   \fi
   \fp@getdigit{rc@temp}{\fp@loopcount}%
   \rc@output{\fp@param}%
\ifnum\fp@loopcount>\fp@loopcountii
   \advance\fp@loopcount by -1
\repeat
```

If the number is smaller than zero, a minus sign is put in front of it. The usage of a token register prevents further expansion.

```
\fp@regcomp{rc@temp}{@0}%
\if\fp@param<%
   \rc@digits={\llap{$-$}}%
   \edef\rc@number{\the\rc@digits\rc@number}%
\fi
```

`\rc@number` is preceded by the computed white space and by any tokens that have been found in the alignment entry and stored in `\rc@preothertoks` or

\rc@postothertoks.

```
\hfill
  \hskip\rc@preskip
  \rc@number
  \hskip\rc@postskip
  \the\rc@postothertoks
\hfill
}
```

8.7 Parameters and configuration

\rcDecimalSign
\rcDecimalSignInput
\rcDecimalSignOutput

The macros \rcDecimalSignInput and \rcDecimalSignOutput can be used to change the default use of a comma as the decimal sign. This can be done separately for input and output. \rcDecimalSign changes both.

```
\def\rcDecimalSign#1{\rcDecimalSignInput{#1}\rcDecimalSignOutput{#1}}
\def\rcDecimalSignInput#1{\edef\rc@decimalsigninput{#1}}
\def\rcDecimalSignOutput#1{\edef\rc@decimalsignoutput{#1}}
\rcDecimalSign{,}
```

⟨/rcmain⟩

References

[1] David Carlisle, "The dcolumn package". The dcolumn package is part of the standard LaTeX distribution.

[2] Frank Mittelbach, David Carlisle, "A new implementation of LaTeX's tabular and array environment". The array package is part of the standard LaTeX distribution.

[3] Eckhart Guthöhrlein, "The fltpoint package", v1.1b, last revised 2004/11/12. The fltpoint package is available from CTAN.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Change History