The *widetable* package

Claudio Beccari[*]

Version number v.2.1; last revised on 2020-01-13.

Contents

Abstract

This package allows to typeset tables of specified width, provided they fit in one page. Instead of introducing an infinite stretching glue, which has an unsymmetrical effect in standard LaTeX, here the `\tabcolsep` dimension is computed so as to have the table come out with the proper width.

1 Legalese

This file is part of the `widetable` package.

The Current Maintainer of this work is Claudio Beccari

The list of all files belonging to the distribution is given in the file ‘README.txt’.

The list of derived (unpacked) files belonging to the distribution and covered by the LPPL is contained in the README.txt file.

---

[*] `claudio dot beccari at gmail dot com`

2 Introduction

It is well known that when the standard environment `tabular*` is opened with a specified width, it is necessary to introduce in the delimiter declaration `@{...}` (possibly) of the first item of the column descriptors argument something like

```
\extracolsep{\fill}
```

in addition to other possible printable delimiters, such as vertical lines, and other fixed spacing commands. The effect is that the extra stretchable glue operates only on the left of each cell *after* (i.e. to the *right* of) the cell that received the declaration; the first cell will never get larger in spite of the presence of this glue.

Another package, ***tabularX***, normally distributed by the LaTeX 3 Team with every version of the TeX system distribution, allows to create expandable cells, provided they contain only text and, possibly, in-line math. These expandable cells are identified with the column descriptor `X`; this identifier defines a paragraph-like cell, the width of which gets determined after some runs of the typesetter on the same source tabular material, so as to find out the correct width of the textual columns.

Also ***tabu*** can build tables of specified width; but it has so many functionalities that it appears to be oversized for a simple task such as the one performed by ***widetable*** and ***tabularx***.

The approach here is a little bit different: the cell contents need not be textual and no cell width is determined in one or more runs of the typesetting program; instead the inter column glue is determined so as to fill every cell on both sides with the proper space. The macros contained in this package are insensitive to the particular kind of cell descriptors and to the presence of multiple `\multicolumn` commands. It proved to work properly also if the `array` package extensions are used. Nevertheless if multiple `\multicolumn` commands in different rows "interlace" the columns they work on, poor results would be obtained: in this case the table is typeset the same as with the environment `tabular`.

On the other hand, as well as for `tabularx`, it needs to typeset the table three times; the first two times with standard values for the inter column glue `\tabcolsep`, in order to find the exact parameters of the linear dependence of the table width from the value of that glue; then executes some computations so as to determine the final correct value of `\tabcolsep`, and on the third run it eventually typesets the table with the specified width.

The typesetting time increase needed for these three tabular runs is generally negligible, nevertheless if a specific document contained many dozens of such tables, the small compilation time increase might become perceivable.

It might be noticed that, in order to perform the necessary computations, a fractional division algorithm must be used; since 2009 any TeX installation uses the ε-TeX extensions; therefore fractional division is not any more an issue as it was in previous versions of this package.

3 Normal use of *widetable*

With this release of the bundle, the environment to be used is named `widetabular`, although the previous name, `widetable`, remains available for backwards compatibility.

This package may issue an error message when the environment includes other unhidden ones; this is explained in the Implementation section. In plain words, if a `widetabular` environment is nested into another `widetabular` one, the inner environment must be "hidden" within a group (i.e. a couple of paired braces); this might not be necessary with other tabular-like environments.

Here it is assumed that the user first uses the standard tabular environment and typesets it to its natural width; should it appear too small, and should it be typeset at a larger width, for example by filling the total `\linewidth` available at that specific point, then and only then the user switches to `widetabular`. Should the initial table be moderately larger than `\linewidth`, then it might be shrunk to `\linewidth` by means of `widetabular`, provided there are enough columns, and therefore column delimiters, to be reduced in size. Of course it is impossible to typeset any tabular with any negative value of `\tabcolsep`; or better, the software does not care, but the result might get very messy.

In other words `widetabular` should be used as a second resort, so as to correct some typesetting features of the standard environment not considered aesthetically acceptable.

The syntax for using the environment `widetabular` is the same as that of the `tabular*` one; the only difference is the name. Therefore one has to specify:

```
\begin{widetabular}{⟨width⟩}[⟨alignment⟩]{⟨column descriptors⟩}
⟨row of cells⟩\\
⟨row of cells⟩\\
...
⟨row of cells⟩\\
⟨row of cells⟩\\
\end{widetabular}
```

4 The method

The principle on which this little package is based is the following one: suppose a certain tabular is typeset with an inter column glue $t_0 = 0\,\mathrm{pt}$ and that its width turns out to be l_0; suppose the same tabular material is typeset again with an inter column glue $t_1 > 0\,\mathrm{pt}$ so that the table gets as large as $l_1 > l_0$. Then, if the table has to be as wide as l the inter column glue t should be

$$t = \frac{l - l_0}{l_1 - l_0} \cdot t_1$$

Therefore we need to run the typesetting of the same tabular material with the two values of the inter column glue set to zero and to t_1, respectively, so as to find the widths l_0 and l_1. Afterwards we have to determine the correct final value t to get the desired value l, and typeset once again the same tabular material for the last time.

Of course the first two runs must put their results into suitable boxes so as to avoid outputting them into the output file, while at the same time allowing to record the width of such enclosing boxes.

5 The *xparse* package

The previous version of this package already used the *xparse* package functionalitIes; but this latter package evolves and now it is possible to “save” the body of the table as an argument to the opening commands, therefore now it is much simpler to use the same table-body several times. Of course this body is saved in internal memory areas, but this task is implemented by the internal macros written in the L3 LaTeX language. The advantage of using the *xparse* package consists in a much shorter code that is easier to read and maintain. The number of macros for this package is reduced by a factor of about four, compared to the previous versions 1.x of this package.

6 Using the ε-TeX facilities

The L3 language and its libraries now offer the user some LaTeX interfaces to its internal macros to the point of executing also floating point operations that span a wide range; the suitable package would be *xfp*. Nevertheless it appears sort of overdone for the simple computations needed here..

At the same time the ε-TeX extended functionalities are now part of all the interpreters of the LaTeX language, pdftex, luatex, and xetex; this renders this package usable with any typesetting programs based on LaTeX: pdflatex, lualatex, and xelatex.

In facts such ε-TeX extensions provide also a scaling operation: given the length L_1 and two homogeneous quantities X_1 and X_2 (where such quantities may be either two integer numbers, or two dimensions), such scaling operation scales L_1 to L_2 by computing

$$L_2 = \frac{L_1 X_2}{X_1}$$

The intermediate results are actually done with integer arithmetics (internally lengths are coded as an integer number of scaled points) but they are done in double words so as to avoid underflows and overflows almost always. Some unusual situations might exist where underflows or overflows do occur, but they must be very unusual, and very unlikely to happen for the calculations of this package. This would happen if for any reason no inter column glue is available; we have difficulties imagining such a table and cannot make even a silly example.

The use of the ε-TeX extensions implies that this package works correctly only with modern engines and kernel formats.

Name	role	age	activity
William John	father	45	employee
Mary Elisabeth	mother	42	elementary school teacher
Joan Laura	daughter	14	junior high school student
Jack Johnathan	son	8	elementary school pupil

Table 1: A regular table typeset with `tabular` and its width is its natural one

Name	role	age	activity
William John	father	45	employee
Mary Elisabeth	mother	42	elementary school teacher
Joan Laura	daughter	14	junior high school student
Jack Johnathan	son	8	elementary school pupil

Table 2: A table typeset with `tabular*` where the total width has been set to `\textwidth`

7 Usage

As explained above, the normal usage of `widetabular` requires the same syntax as that of `tabular*` except that no explicit stretchable glue has to be inserted in the column separators as it is necessary to do with `tabular*`. Examine the table shown in table 1[1] that is typeset at its natural width.

The same table can be built with `tabular*` as in table 2.

As it can be seen, large inter column spaces are inserted right at the left of the contents of every cell except the first one, and the table appears too much spread out.

The tabular can be built also with the environment `tabularx`, defined by the `tabularX` package; see the result in table 3.

As it is noticeable the whole space to enlarge the tabular has been used by the `X` column, and the table does not look right.

Now we show the difference by using the `widetabular` environment in table 4.

In table 4 the column specifications are the same as those used in the code for table 1, but only the spaces separating the columns have been modified, not the column types and widths. Of course one may object that the table spaces are too wide and table 1 looks better. But if, for example, in a certain document all tables are required to span the whole measure, the solution shown in table 4

---

[1] Notice that here the name “table” is used to refer to the *`table`* floating environment and its caption, while “tabular” is reserved to the tabular itself and its contents.

Name	role	age	activity
William John	father	45	employee
Mary Elisabeth	mother	42	elementary school teacher
Joan Laura	daughter	14	junior high school student
Jack Johnathan	son	8	elementary school pupil

Table 3: A table typeset with `tabularx` where the total width has been set to `\textwidth`

Name	role	age	activity
William John	father	45	employee
Mary Elisabeth	mother	42	elementary school teacher
Joan Laura	daughter	14	junior high school student
Jack Johnathan	son	8	elementary school pupil
Goofy Pluto Donald Duck Mickey Mouse		4	Walt Disney

Table 4: A table typeset with `widetabular` where the total width has been set to `\textwidth`

is the only one acceptable among these four examples. It's up to the user to chose among these four solutions in terms of the actual tabular contents and the stylistic constraints the document must fulfil. If the examples were typeset with the (horizontal and vertical rules that emphasise each cell (instead of using only the `booktabs` horizontal rules) it would be more evident how the various environments shape the cells and where they insert the extra spacing so as to reach the desired width.

8 Warnings

Normally `widetabular` works well as described in the example shown in table 4. Nevertheless there are some issues that may alter its smooth working.

One such issue takes place when the specified table width is shorter than the natural width. In this case the table is typeset as in table 1 at its natural width, but a warning is issued that explains why: the warning looks like this

```
Package widetable Warning: The minimum width of the tabular material
(widetable)                amounts to 225.19809pt, and is larger
(widetable)                than the required width of 177.5pt
(widetable)
(widetable)                The table is typeset with the default
(widetable)                column spacing on input line 415.
```

As usual the warning is contained into the `.aux` file and in the console, if the shell editor displays it..

When some adjacent cells are grouped with the `\multicolumn` command; the table might come out of the correct specified width even if the spanned cells (in different rows) do not belong to the same columns, but the table looks very ugly; we cannot say that `widetable` is responsible of this ugliness, or if the table is ill formed because of using such overlapping spanned cells; the best suggestion is to avoid using such “acrobatic” tabular compositions.

9 Acknowledgements

I must deeply thank Enrico Gregorio for the revision of this package macros and for his wise suggestions about the correct programming style. If some glitch still remains in the programming style, that is just my fault.

10 Implementation

This package has been already identified by the commands extracted by the `doctrip` package, during the `.dtx` file compilation.

We require the `xparse` package in order to define the `widetable` environment with its extended commands. This package version should be younger the the specified date contained in the optional argument. If it is not, a warning is issued; but expect errors. It is a warning that urges the user to upgrade his/her TeX system installation.

```
\RequirePackage{xparse}[2019-05-01]
```

We require the `xparse` package in order to define the environment `widetable` with its extended commands. Tis package version should be younger the the specified date contained in the optional argument. If it is not, a warning is issued; but expect errors. It is a warning that should urge the user to upgrade his/her TeX system installation.

The special environment opening macro requires the following syntax:

```
\begin{widetable}{⟨width⟩}[⟨alignment⟩]{⟨column descriptors⟩}
⟨table body⟩
\end{widetable}
```

We further define the `tabular` environment typesetting. Actually, with the new `xparse` faciltiies, the opening command parameters can be used also in the closing part of the environment, so that when the ⟨*width*⟩ and the ⟨*column descriptors*⟩ are given to the opening environment statement, they can be used again and again also by the closing commands.

Actually the `widetabular` environment can contain other environments, even another `widetabular` ones, but the external one should not be upset by the internal ones. In order to achieve this result, it is necessary that embedded environments are hidden within a group delimited by a pair of matching braces; this is compulsory for an embedded `widetabular` environment, while it is not strictly required for other environments.

The environment opening and closing actions are defined by means of low level commands.

The opening part of the environment reduces to nothing else but the background L3 functions executed by the `\begin` command implementation and the correct parsing of the list of argument descriptors

```
\DeclareDocumentEnvironment{widetabular}{m O{c} m +b}
{% OPENING WIDETABLE COMMANDS
}%
```

These arguments have the following meanings.

Argument number one It is mandatory and represents the desired table width.

Argument number two It is optional. With a default value of `c` the table is aligned with respect to its math axis; the other possible values are `t` for top alignment, and `b` for bottom alignment; they are the same values used for the LaTeX-kernel tabular environments.

Argument number three It is mandatory; it should contain all the column descriptors and inter-column separators, possibly in the extended forms provided by the *array* package.

Argument number four It represents an *xparse* functionality by which the whole environment body is internally saved in a sort of verbatim mode and become usable again and again as argument `#4`.

The closing statement will actually do the whole job. It first sets `\tabcolsep` to zero and typesets the resulting table into box zero; it uses, with parameter `#4`, the table body collected with the argument descriptor `b` of the opening command.

Then it sets `\tabcolsep` to 6 pt (the default value) and typesets again the table into box two. The width of box zero is l_0 and that of box two is l_1; these are the lengths needed by the equation that evaluates the final typesetting inter column spacing.

The arbitrary constant of 6 pt is t_1, and the specified width l (parameter `#1`) are used to compute the new value of `\tabcolsep`. The subtractions are computed directly on the dimensions and passed to a `\dimexpr` expression so as to determine the new `\tabcolsep` value.

The table is eventually typeset without using boxes, while the contents of box zero and box two, upon exiting the environment, are restored to any value they might have contained before entering `widetabular`.

```
{% CLOSING WIDETABLE
   \dimen0=#1 % required width
   \tabcolsep=\z@
   \setbox\z@=\hbox{\tabular{#3}#4\endtabular}%
   \tabcolsep=6pt\relax
   \setbox\tw@=\hbox{\tabular{#3}#4\endtabular}%
   \ifdim\dimen0>\wd\z@
```

```
      \tabcolsep=%
          \dimexpr\tabcolsep*(\dimen0-\wd\z@)/(\wd\tw@-\wd\z@)\relax
    \else
      \ifdim\dimen0<\wd\z@
         \PackageWarning{widetable}{%
         The minimum width of the tabular material\MessageBreak
         amounts to \the\wd\z@,  and is larger\MessageBreak
         than the required width  of \the\dimen0\MessageBreak
         \MessageBreak
         The table is typeset with the default\MessageBreak
         column spacing}%
       \fi
    \fi
    \tabular[#2]{#3}#4\endtabular
  \ignorespacesafterend
}
```

For backward compatibility we let the names `\widetable` and `\endwidetable` equal respectively to `\widetabular` and `\endwidetabular`, so that the old name of the environment provided by this package is still usable; compiling old documents is till possible; nevertheless these “old” names are discouraged; in a future they might not be available any more.

```
\let\widetable\widetabular \let\endwidetable\endwidetabular
```

Notice the test and the warning: `widetabular` modifies the table width only if its minimum width (obtained with `\tabcolsep` equal to zero) is smaller than the requested width; otherwise it typesets the table with the default inter column glue, and outputs the warning message.

11 Conclusion

Tables should always have their standard inter column spaces, but… The default value of `\tabcolsep` is fixed by the document class, it is not prescribed by a supreme law: therefore what does it mean “natural width”. Probably the one determined by the class default value of `\tabcolsep`, so that all tables have the same general look. But here we used the phrase “minimum width” as that of the tabulars width when the inter column glue is set to zero; we avoided speaking of the “natural width” because the phrase is not specific.

Nevertheless sometimes a table is slightly wider than the current measure; why not shrink the table by shrinking `\tabcolsep` by the right amount in order to fit the measure? The result might be a table where only the inter column spaces are shrunk, not the whole table, fonts, drawings, and figures included, a result easily obtainable with a `\resizebox` command available through the *`graphicx.sty`* package. Nobody forbids to follow this technique, of course, but the `widetable` route might yield a better result.

The same is true when a natural width table is slightly shorter than the measure; enlarging it by retouching the `\tabcolsep` inter column space might be the right solution in order to avoid a multitude of slightly different indents or left margins.

```

\endinput
```