

The *widetable* package

Claudio Beccari*

Version number v.2.0; last revised on 2019-06-25.

Contents	6 Using the ε-TeX facilities	4
1 Legalese	1 7 Usage	5
2 Introduction	2 8 Warnings	6
3 Normal use of <i>widetable</i>	3 9 Acknowledgements	7
4 The method	3 10 Implementation	7
5 The <i>xparse</i> package	4 11 Conclusion	9

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 L^AT_EX, 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.

This work may be distributed and/or modified under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in <http://www.latex-project.org/lppl.txt> and version 1.3 or later is part of all distributions of LaTeX version 2003/12/01 or later.

This work has the LPPL maintenance status “maintained”.

The Current Maintainer of this work is Claudio Beccari

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

The list of derived (unpacked) files belonging to the distribution and covered by the LPPL is defined in the manifest.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 `@{...}` of (possibly) as 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 L^AT_EX 3 Team with every version of the T_EX 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 are obtained, in the sense that 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 extrapolate 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 table runs in general is negligible, nevertheless if a specific document contained many dozens of such tables, the compilation time 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 T_EX installation uses the ε -T_EX extensions; therefore fractional division is not any more an issue as it was in previous versions of this package.

3 Normal use of *widetable*

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

Here it is assumed that the table is first typeset 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 `tabular` environment is changed to `widetable`. Should the initial table be moderately larger than `\linewidth`, then it might be shrunk to `\linewidth` with `widetable`, provided there are enough columns, and therefore column delimiters, to be reduced in size. Of course it’s impossible to typeset any table with any negative value of `\tabcolsep`; or better, it is possible, but the result in general is very messy.

In other words `widetable` should be used as a second resort, so as to correct some typesetting features not considered aesthetically acceptable.

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

```
\begin{widetable}{\langle width \rangle}[\langle alignment \rangle]{\langle column descriptors \rangle}
\langle row of cells \rangle \\
\langle row of cells \rangle \\
...
\langle row of cells \rangle \\
\langle row of cells \rangle \\
\end{widetable}
```

4 The method

The principle on which this little package is based is the following; suppose a certain table is typeset with an inter column glue $t_0 = 0$ pt and that its width turns out to be l_0 ; suppose the same tabular material is typeset again with an inter column glue $t_1 > 0$ 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 , 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 the enclosing boxes.

5 The *xparse* package

The previous version of this package 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 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 and easier to read and maintain. The number of macros for this package is reduced by a factor of about four, compared to the previous version of this package.

6 Using the $\varepsilon\text{-TeX}$ 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 $\varepsilon\text{-TeX}$ extended functionalities are now part of all the interpreters of the \LaTeX language, *pdf \TeX* , *luat \TeX* , and *xet \TeX* ; this renders this package usable for all main typesetting programs, *pdf \LaTeX* , *lualat \TeX* , and *xelat \TeX* .

In facts such $\varepsilon\text{-TeX}$ 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

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

The intermediate results are actually done with integer arithmetics (internally lengths are coded as an integer number of scaled points) but are done in double words so as to avoid underflows and overflows almost always. Some unusual situations might exist where underflows or overflows may 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 $\varepsilon\text{-TeX}$ 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 `widetable` 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 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 table 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 table has been taken from

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
Pippo	Pluto	4	Walt Disney
Paperino	Topolino		

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

the `X` column, and the table does not look right.

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

In table 4 the column specifications are the same as those used in the code of 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 must span the whole width, the solution given in table 4 is the only one acceptable among the four shown in these examples. It's up to the user to choose among these four solutions in terms of the actual table 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 `widetable` 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.

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 never use such “acrobatic” table 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

We identify the package and the necessary L^AT_EX kernel

```
1 \ProvidesPackage{widetable}[2019-06-24 v.1.5 A simpler way to produce
2   tables of specified width]
```

We require the `xparse` package in order to define the environment `widetable` with its extended commands. This package version should be younger than 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 T_EX system installation.

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

The special environment opening macro requires the following syntax:

```
\begin{widetable}{\langle width \rangle}[\langle alignment \rangle]{\langle column descriptors \rangle}
\langle table body \rangle
\end{widetable}
```

At this point it will be the `widetable` environment responsibility to call `\WT@scale` with the proper arguments

We further define the start of the tabular typesetting that will be useful in a while. Actually the table parameters are usable also in the closing part of the environment (thanks to `xparse`), so that when the `\langle width \rangle` and the `\langle column descriptors \rangle` are given to the opening environment statement, these parameters can be used again and again.

Actually the `widetable` environment can contain other environments, even another `widetable` environment, 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 `widetable` environment, while it is not strictly required for differently named 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

```
4 \DeclareDocumentEnvironment{widetable}{m O{c} m +b}
5 {% OPENING WIDETABLE COMMANDS
6 }%
```

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 L^AT_EX-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 usable as #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 6pt (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 6pt 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 `widetable`.

```
7 {% CLOSING WIDETABLE
8   \dimen0=#1 % required width
9   \tabcolsep=\z@
10  \setbox\z@=\hbox{\tabular{#3}#4\endtabular}%
11  \tabcolsep=6pt\relax
12  \setbox\tw@=\hbox{\tabular{#3}#4\endtabular}%
13  \ifdim\dimen0>\wd\z@
14    \tabcolsep=\dimexpr \tabcolsep*(\dimen0-\wd\z@)/(\wd\tw@-\wd\z@)\relax
15  \else
16    \ifdim\dimen0<\wd\z@
17      \PackageWarning{widetable}{%
18        The minimum width of the tabular material\MessageBreak
```



```

19         amounts to \the\wd\z@, and is larger\MessageBreak
20         than the required width of \the\dimen0\MessageBreak
21         \MessageBreak
22         The table is typeset with the default\MessageBreak
23         column spacing}%
24     \fi
25 \fi
26 \tabular[#2]{#3}#4\endtabular
27 \ignorespacesafterend
28 }

```

Notice the test and the warning: `widetable` modifies the table only if its width, typeset with `\tabcolsep` equal to zero, is smaller than the requested width; otherwise it typesets the table with the default inter column glue. The warning message is produced only if the requested width is smaller the minimum width.

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 all tables have the same general look. But here we used the phrase “natural width” as the table width when the inter column glue is set to zero

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.

```

29
30 \endinput

```