

The adjustbox Package

Martin Scharrer

martin@scharrer-online.de

<http://www.ctan.org/pkg/adjustbox>

Version v0.3 – 2011/03/20

Abstract

This package provides macros missing in `graphics` to trim, clip and generally adjust boxed \LaTeX material. The macros allow for verbatim content. Equivalent environments are also provided. The trim and clip operation are implemented using the `pgf` package, which supports both DVI/PS and PDF output.

This package is new and the implementation might change in upcoming releases!

1 Introduction

The standard \LaTeX package `graphicsx` (the extended version of `graphics`) provides the macro `\includegraphics` [*options*] {*file name*} which can be used to include graphic files. Several options can be used to scale, resize, rotate, trim and/or clip the graphic. The macros `\scalebox`, `\resizebox` and `\rotatebox` are also provided to apply the corresponding operation on \LaTeX material, which is subsequently placed inside a `\hbox`. However no macros are provided to trim or clip \LaTeX material, most likely because this operations are not done by \TeX but by the output format, i.e. using PostScript (PS) or PDF operations.

This package provides the missing macros `\clipbox` and `\trimbox` as well as the general `\adjustbox` macro. The clipping and trimming operations are implemented using a `pgfpicture` environment from the `pgf` package which supports both PS and PDF output.

2 Usage

This section describes the usage of the provided macros, which are outlined in [subsection 2.1](#). Possible advanced values for the macro arguments are mentioned in [subsection 2.2](#). The existing verbatim support is explained in [subsection 2.3](#). Finally [subsection 2.4](#) compares the existing macros with the corresponding options of `\adjustbox`. See the example [section 3](#) for examples of this macros.

It is recommended to also read the *Graphics Guide* (`grfguide`, i.e. the manual of the `graphics/x` packages), to understand the existing options for `\includegraphics`.

2.1 Box Modification Macros

Trim Box Content

```
\trimbox*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨text⟩}
```

Trim Box Content

The macro `\trimbox` trims the given amount from the lower left (ll) and the upper right (ur) corner of the box. This means that the amount `⟨llx⟩` is trimmed from the left side, `⟨lly⟩` from the bottom and `⟨urx⟩` and `⟨ury⟩` from the right and top of the box, respectively. Trimming means that the official size of the box is reduced, but no material is actual removed. The material in the trimmed areas simply swaps over the official border.

If the starred version is used the four coordinates are taken as the viewport instead, i.e. the box is trimmed to the rectangle described by the coordinates.

```
\begin{trimbox}*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}  
  ⟨text⟩  
\end{trimbox}
```

```
\begin{trimbox*}{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}  
  ⟨text⟩  
\end{trimbox*}
```

The `trimbox` and `trimbox*` environments do the same as the corresponding macros. Special care is taken so that the macros and the environments can have the same name. Because of this the star can be either part of the name or an optional argument. Also the plain \TeX syntax for environments (`\trimbox ... \endtrimbox`) can not be used because it will trigger `\trimbox` as a macro.

Clip Box Content

```
\clipbox*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}{⟨text⟩}
```

The `\clipbox` macro works like the `\trimbox` and trims the given amounts from the `⟨text⟩`. However, in addition the trimmed material is also clipped, i.e. it is not shown in the final document. Note that the material will still be part of the output file but is simply not shown. It might be exported using special tools, so using `\clipbox` (or `\includegraphics[clip,trim=...]`) to censor classified information would be a bad idea. The starred version will again use the given coordinates as viewport.

```
\begin{clipbox}*{⟨llx⟩ ⟨lly⟩ ⟨urx⟩ ⟨ury⟩}  
  ⟨text⟩  
\end{clipbox}
```

```
\begin{clipbox*}{\llx \lly \urx \ury}
  \langle text \rangle
\end{clipbox*}
```

Die obigen Makros als Umgebungen. Es gelten die gleichen Regeln wie bei den trimbox Umgebungen.

Adjust Box Content

```
\adjustbox{\includegraphics options}{\langle text \rangle}
```

The `\adjustbox` macro is the general form of all box modifying macros mentioned in the introduction. It can be thought as an `\includegraphics` for \LaTeX material. It supports the same set of *options*, however they are provided as a mandatory not as an optional argument. An `\adjustbox` without options would not make sense and can be replaced by a simple `\mbox`. There is no starred version of this macro. See also [Table 1](#) for a comparison of `\adjustbox` with the other macros.

```
\begin{adjustbox}{\includegraphics options}
  \langle text \rangle
\end{adjustbox}
```

The environment version of `\adjustbox`.

2.2 Argument Values

The argument values are parsed by versatile `\pgfmathparse` of the already used `pgf` package. This allows very complex arithmetic expressions as any of the trim/clip coordinates or other numeric options. See the `pgfmanual` for detailed information. Note that the four values for `\trimbox` and `\clipbox` as well as for the `trim` and `viewport` option of `\adjustbox` are separated by spaces. If the expression of any of this values holds a space or ends with a macro (eats trailing spaces!) it must be wrapped into braces `{ }`.

Parsing

Space=Separator

```
\width \height \depth \totalheight
```

This \LaTeX lengths hold the original dimension of *text* and can be used as part of the arguments to `\adjustbox`, `\trimbox` and `\clipbox`. The `totalheight` is the height plus depth.

If no unit is provided for of the bounding box coordinates (`llx`, `lly`, `urx`, `ury`) then PostScript points (*big points*, `bp`, $72 \text{ textbp} = 1 \text{ textinch}$) are used, as it is the default behaviour of the `trim` and `viewport` ptions of `graphicx`'s `\includegraphics`. Note that `graphicx` converts all values, independent if a unit is provided or not, internally to `bp`, because `graphics` where traditionally stored in Encapsulated PostScript (EPS) files. The more modern PDF files also use `bp` instead of `pt`. Because the `adjustbox` package macros target \LaTeX material and users will mostly use `pt` values this internal conversion to `bp` got disabled for them to avoid unnecessary rounding errors.

Default unit

Examples for Argument Values

`\trimbox{.5\width} 10 {log10(10)/sin(45) + 1} 10pt}{text}`

will trim half the original amount from the left, 10 bp from the bottom and 2.42328 bp from the right (bp, because no unit was used in the formula; change e.g. ‘+ 1’ to ‘+ 1pt’ to get 2.42328 pt), as well as 10 pt from the top.

2.3 Verbatim Support

The macros read the `<text>` as \TeX `\hbox` and not as a macro argument in order to support verbatim content. This means that the braces around the content can also be written as `\bgroup` and `\egroup`:

`\trimbox{1 2 3 4}\bgroup <text>\egroup`

Special care is taken to allow the `<text>` to be a single macro (except `\bgroup`) without any braces:

`\clipbox{1 2 3 4}\somemacro`

This is to support the questionable habit of some \TeX users to drop the braces for single token arguments. All environments support verbatim content.

2.4 Alternatives for existing Macros

The flexible `\adjustbox` can also be used as an alternative to existing macros from the `graphics` package as shown by Table 1. Because it is longer than the originals this is only of benefit if combinations are to be replaced or verbatim text must be supported.

Table 1: Alternatives for existing Macros

Original Macro (w/o text argument)	Alternative (w/o text argument)
<code>\rotatebox{<angle>}</code>	<code>\adjustbox{angle=<angle>}</code>
<code>\scalebox{<factor>}</code>	<code>\adjustbox{scale=<factor>}</code>
<code>\scalebox{<x-factor>}[<y-factor>]</code>	<code>\adjustbox{width=<x-factor>\width,height=<y-factor>\height}</code>
<code>\reflectbox</code>	<code>\adjustbox{width=-\width,height=\height}</code>
<code>\resizebox{<width>}{<height>}</code>	<code>\adjustbox{width=<width>,height=<height>}</code>
<code>\resizebox*{<width>}{<totalheight>}</code>	<code>\adjustbox{width=<width>,totalheight=<totalheight>}</code>
<code>\trimbox{<llx> <lly> <urx> <ury>}</code>	<code>\adjustbox{trim=<llx> <lly> <urx> <ury>}</code>
<code>\trimbox*{<llx> <lly> <urx> <ury>}</code>	<code>\adjustbox{viewport=<llx> <lly> <urx> <ury>}</code>
<code>\clipbox{<llx> <lly> <urx> <ury>}</code>	<code>\adjustbox{trim=<llx> <lly> <urx> <ury>,clip}</code>
<code>\clipbox*{<llx> <lly> <urx> <ury>}</code>	<code>\adjustbox{viewport=<llx> <lly> <urx> <ury>,clip}</code>

3 Examples

The following examples show the application of the package macros on an example text. The result is placed in a tight, colored frame box to show the resulting dimensions.

`\example`

A	B
C	D

`\trimbox{10 5 10 5}{\example}`

A	B
C	D

`\clipbox{10 5 10 5}{\example}`

A	B
C	D

`\trimbox*{15 5 25 30}{\example}`

A	B
C	D

`\clipbox*{15 5 25 30}{\example}`

A
C

`\adjustbox{trim=10 5 10 5,angle=45}{\example}`

A	B
C	D

`\adjustbox{scale=1.5}{\example}`

A	B
C	D

`\adjustbox{width=180pt,height=20pt}{\example}`

A	B
C	D

`\adjustbox{width=180pt,height=20pt,keepaspectratio}{\example}`

A	B
C	D

Environment example:

`\begin{adjustbox}{angle=2}:`

`verbatim inside \begin{adjustbox}{angle=2} ... \end{adjustbox}`

4 Implementation

```
1 \RequirePackage{graphicx}[1999/02/16]
2 \RequirePackage{pgf}
```

`clipbox`

`\clipbox`

```
3 \newcommand\clipbox{%
4   \begingroup
5   \def\adjustbox@name{clipbox}%
6   \@ifstar
7     {\adjustbox@{clip,viewport=}}%
8     {\adjustbox@{clip,trim=}}%
9 }
10 \def\endclipbox{%
11   \endadjustbox
12 }
```

`clipbox*`

```
13 \newenvironment{clipbox*}
14   {\begin{clipbox}*}
15   {\end{clipbox}}
```

`trimbox`

`\trimbox`

```
16 \newcommand\trimbox{%
17   \begingroup
18   \def\adjustbox@name{trimbox}%
19   \@ifstar
20     {\adjustbox@{viewport=}}%
21     {\adjustbox@{trim=}}%
22 }
23 \def\endtrimbox{%
24   \endadjustbox
25 }
```

`trimbox*`

```

26 \newenvironment{trimbox*}
27   {\begin{trimbox}*}
28   {\end{trimbox}}

```

adjustbox

\adjustbox

```

29 \newcommand\adjustbox{%
30   \begingroup
31   \def\adjustbox@name{adjustbox}%
32   \adjustbox@{}%
33 }
34 \def\endadjustbox{%
35   \unskip
36   \egroup
37   \color@endgroup
38   \egroup
39   \adjustbox@@
40 }

```

\adjustbox@

```

41 \def\adjustbox@#1#2{%
42   \def\adjustbox@setkeys{\setkeys{Gin}{#1#2}}%
43   \ifx\@currenir\adjustbox@name
44     \edef\@currenir{\@currenir\noexpand\empty}%
45     \def\next{%
46       \setbox\@tempboxa\hbox\bgroup
47         \color@setgroup\bgroup
48         \ignorespaces
49     }%
50   \else
51     \def\next{%
52       \setbox\@tempboxa\hbox\bgroup%0
53         \color@setgroup\bgroup%
54         \aftergroup\color@endgroup
55         \aftergroup\egroup%
56         \aftergroup\adjustbox@@
57         \@ifnextchar\bgroup
58         {\let\@let@token=}%
59         {\adjust@box}%
60     }%
61   \fi
62   \next
63 }

```

`\adjust@box`

```
64 \def\adjust@box#1{%  
65   #1\egroup  
66 }
```

`\adjustbox@@`

```
67 \def\adjustbox@@{%  
    Set the box dimension macros.  
68   \def\width{\wd\@tempboxa}%  
69   \def\height{\ht\@tempboxa}%  
70   \def\depth{\dp\@tempboxa}%  
71   \@tempdimc=\ht\@tempboxa  
72   \advance\@tempdimc by \dp\@tempboxa\relax  
73   \def\totalheight{\@tempdimc}%
```

Locally redefine `\Gin@defaultbp` to use `\pgfmathsetmacro` with `bp` as the default unit. This should yield the same results (apart of smaller rounding errors) if values are given without unit but avoids the internal conversion to `bp` of values with units.

```
74   \def\pgfmathresultunitscale{1bp}%  
75   \let\pgfmathpostparse\pgfmathscaleresult  
76   \let\Gin@defaultbp\pgfmathsetmacro  
77   \let\setlength\pgfmathsetlength
```

The rest of the code was adapted from the `\Gin@ii` macro from the `graphicx` package. The *temp switch a* is set to `true` to indicate to `graphicx` that the scaling should be done internal, so this package doesn't have to do it. The content including macro `\adjustbox@@@` is but into place, the saved options are activated and the final size is set. The typesetting of the content is finally done by executing the token register.

```
78   \@tempwattrue  
79   \toks@{\adjustbox@@@}%  
80   \adjustbox@setkeys  
81   \Gin@esetsize  
82   \the\toks@  
83   \endgroup  
84 }
```

`\adjustbox@@@`


```

85 \def\adjustbox@@@{%
86   \def\Gin@llx{0}%
87   \Gin@defaultbp\Gin@lly{+-\dp\@tempboxa}%
88   \Gin@defaultbp\Gin@urx{+\wd\@tempboxa}%
89   \Gin@defaultbp\Gin@ury{+\ht\@tempboxa}%
90   \Gin@viewport@code
91   \begin{pgfpicture}%
92     \pgfpathmoveto{\pgfqpoint{\Gin@llx pt}{\Gin@lly pt}}%
93     \pgfpathlineto{\pgfqpoint{\Gin@urx pt}{\Gin@lly pt}}%
94     \pgfpathlineto{\pgfqpoint{\Gin@urx pt}{\Gin@ury pt}}%
95     \pgfpathlineto{\pgfqpoint{\Gin@llx pt}{\Gin@ury pt}}%
96     \pgfpathclose
97     \expandafter\pgfusepath\ifGin@clip{clip}\else
98     {use as bounding box}\fi\relax
99     \pgfset{inner sep=\z@,outer sep=\z@}%
100    \pgfnode{rectangle}{base west}{\box\@tempboxa}
101    \pgfsetbaselinepointnow{\pgfpointanchor{
102      ADJUSTBOX@NODE}{base}}%
103  \end{pgfpicture}%
104 }

```