The adjustbox Package

Martin Scharrer

martin@scharrer-online.de

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

Version v0.2 – 2011/01/27

Abstract

This package provides macros missing in `graphics/x` 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 `graphicx` (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 (La)TeX material, which is subsequently placed inside a `\hbox`. However no macros are provided to trim or clip (La)TeX 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 section 2.1. Possible advanced values for the macro arguments are mentioned in section 2.2. The existing verbatim support is explained in section 2.3. Finally section 2.4 compares the existing macros with the corresponding options of `\adjustbox`. 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`. See the example section for examples of this macros.

2.1 Box Modification Macros

Trim Box Content

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

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 plainTEX syntax for environments (`\trimbox ... \endtrimbox`) can not be used because it will trigger `\trimbox` in macro mode.

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⟩}
  ⟨text⟩
\end{clipbox*}
```

The environment versions of \clipbox and \clipbox*.

Adjust Box Content

```
\adjustbox{⟨includegraphics options⟩}{⟨text⟩}
```

The \adjustbox macro is the general form of all box modifying macros mentioned in the introduction. It can be thought as an \includegraphics for (LA)TEX 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⟩}
  ⟨text⟩
\end{adjustbox}
```

The environment version of \adjustbox.

2.2 Argument Values

Parsing

The argument values are parsed by versatile \pgfmathparse of the already used pgf package. See the pgfmanual for detailed information. This allows very complex arithmetic expressions as any of the trim/clip coordinates or other numeric options. 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 '{ }'.

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.

Default unit

If no unit is provided for of the bounding box coordinates (llx, lly, urx, ury) then PostScript points (*big points*, bp, 72 bp = 1 inch) 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 (LA)TEX material and users will mostly use pt values this internal conversion to bp got disabled for them to avoid unnecessary rounding errors.

Examples for Argument Values

`\trimbox{{.5\width} 10 {log10(10)/sin(45) + 1} 10pt}{`⟨*content*⟩`}` 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 an 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` ⟨*content*⟩`\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 LATEX 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 then the originals this is only of benefit if combinations are to be replaced or verbatim text must be supported.

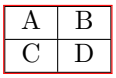
Table 1: Alternatives for existing Macros

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

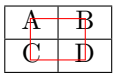
3 Examples

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
```

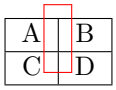



```
\trimbox{10 5 10 5}{\example}
```



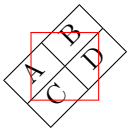

```
\clipbox{10 5 10 5}{\example}
```

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

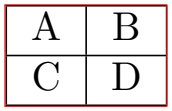



```
\clipbox*{15 5 25 30}{\example}
```

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




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




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


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

Environment example:

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

4 Implementation

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

clipbox

\clipbox

```
\newcommand\clipbox{%
    \begingroup
    \def\adjustbox@name{clipbox}%
    \@ifstar
        {\adjustbox@{clip,viewport=}}%
        {\adjustbox@{clip,trim=}}%
}

\def\endclipbox{%
    \endadjustbox
}
```

clipbox*

```
\newenvironment{clipbox*}
    {\begin{clipbox}*}
    {\end{clipbox}}
```

trimbox

\trimbox

```
\newcommand\trimbox{%
    \begingroup
    \def\adjustbox@name{trimbox}%
    \@ifstar
        {\adjustbox@{viewport=}}%
        {\adjustbox@{trim=}}%
}
```

```
\def\endtrimbox{%
    \endadjustbox
}
```

trimbox*

```
\newenvironment{trimbox*}
    {\begin{trimbox}*}
    {\end{trimbox}}
```

adjustbox

\adjustbox

```
\newcommand\adjustbox{%
    \begingroup
    \tracinggroups=1%
    \def\adjustbox@name{adjustbox}%
    \adjustbox@{}%
}

\def\endadjustbox{%
    \unskip
    \egroup
    \color@endgroup
    \egroup
    \adjustbox@@
}
```

\adjustbox@

```
\def\adjustbox@#1#2{%
    \def\adjustbox@setkeys{\setkeys{Gin}{#1#2}}%
    \ifx\@currenvir\adjustbox@name
        \expandafter\def\expandafter\@currenvir\↙
           expandafter{\@currenvir\empty}%
        \def\next{%
            \setbox\@tempboxa\hbox\bgroup
                \color@setgroup\bgroup
                \ignorespaces
        }%
```

```
    \else
        \def\next{%
            \setbox\@tempboxa\hbox\bgroup%0
                \color@setgroup\bgroup%
                \aftergroup\color@endgroup
                \aftergroup\egroup%
                \aftergroup\adjustbox@@
                \@ifnextchar\bgroup
                    {\let\@let@token=}%
                    {\adjust@box}%
        }%
    \fi
    \next
}
```

\adjust@box

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

\adjustbox@@

```
\def\adjustbox@@{%
```

Set the box dimension macros.

```
  \def\width{\wd\@tempboxa}%
  \def\height{\ht\@tempboxa}%
  \def\depth{\dp\@tempboxa}%
  \@tempdimc=\ht\@tempboxa
  \advance\@tempdimc by \dp\@tempboxa\relax
  \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.

```
    \def\pgfmathresultunitscale{1bp}%
    \let\pgfmathpostparse\pgfmathscaleresult
    \let\Gin@defaultbp\pgfmathsetmacro
    \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.

```
    \@tempswatrue
    \toks@{{\adjustbox@@@}}%
    \adjustbox@setkeys
    \Gin@esetsize
    \the\toks@
    \endgroup
}
```

\adjustbox@@@

```
\def\adjustbox@@@{%
    \def\Gin@llx{0}%
    \Gin@defaultbp\Gin@lly{+-\dp\@tempboxa}%
    \Gin@defaultbp\Gin@urx{+\wd\@tempboxa}%
    \Gin@defaultbp\Gin@ury{+\ht\@tempboxa}%
    \Gin@viewport@code
    \begin{pgfpicture}%
        \pgfpathmoveto{\pgfqpoint{\Gin@llx pt}{\↙
           Gin@lly pt}}%
        \pgfpathlineto{\pgfqpoint{\Gin@urx pt}{\↙
           Gin@lly pt}}%
        \pgfpathlineto{\pgfqpoint{\Gin@urx pt}{\↙
           Gin@ury pt}}%
        \pgfpathlineto{\pgfqpoint{\Gin@llx pt}{\↙
           Gin@ury pt}}%
        \pgfpathclose
        \expandafter\pgfusepath\ifGin@clip{clip}\else↙
           {use as bounding box}\fi\relax
        \pgfset{inner sep=\z@,outer sep=\z@}%
        \pgfnode{rectangle}{base west}{\box\@tempboxa↙
           }{}{}%
    \end{pgfpicture}%
}
```