An easier interface to insert figures, tables and other objects in LaTeX

Erik Zöllner

October 28, 2024
easyfloats v1.1.0

Abstract

In standard LaTeX inserting objects like figures or tables requires too much knowledge for beginners and too much typing effort and hardcoding for people like me. This package aims to make insertion of figures and tables easier for both beginners and experts. Despite the term *floats* in it's name it also allows to disable floating of such objects. It also defines a new placement `I` similar to `H` which respects the indentation in an `itemize`.

https://gitlab.com/erzo/latex-easyfloats

Contents

1 Examples

Let's start with a few examples. Environments, commands and keys defined by this package are links (both in the code and in the text). Clicking on them will get you to their explanation in section 3.

Appendix A gives a motivation why this package is useful. There is a list of related packages in appendices B and C. Package names link to the rather short description in that list.

The examples have been tested with pdflatex 2024.10.2.

1.1 Table

Use the `tableobject` environment for inserting tables. Pass caption and label as keyword arguments. You can't mess up the order of caption and label and you get a warning if you forget to specify them. You don't need two environments (one for the float, one for the table—`tableobject` can do both). booktabs (and array) are loaded automatically (if not disabled, see section 3.5).

The last argument, `cl`, is not an argument of the `tableobject` environment but of the standard LaTeX `tabular` environment which is started by the `tableobject` environment because it is specified with `env`. It specifies the columns of the table: The `l` stands for one left aligned column and the `c` for one centered column. An `r` would stand for a right aligned column and a `p{⟨width⟩}` for a column which breaks the cell contents to a specified width. The siunitx package defines `S` for a column which aligns numbers and in the `tabularx` environment defined by the tabularx package `X` is available for a `p` column which automatically takes as much space as there is available. You can use the `\newcolumntype` command (defined by the array package) to define your own column types, e.g. `\newcolumntype{M}{>{$}c<{$}}` to define a column type `M` for centered math content. See the array package documentation [1, chapter 1] for more column specification options.

`&` is the column separator and `\\` specifies the end of a row.

The table is formatted with `\toprule`, `\midrule` and `\bottomrule` as explained in the booktabs documentation [2, chapter 2].

```
\documentclass{article}
\usepackage{easyfloats}

\begin{document}
\begin{tableobject}{caption=Some catcodes, label=tbl:catcodes,
↪  env=tabular}{cl}
    \toprule
        Catcode & Meaning          \\
    \midrule
        0       & Escape Character \\
        1       & Begin Group      \\
        2       & End Group        \\
        \vdots  & \quad \vdots     \\
    \bottomrule
```

```
\end{tableobject}
\end{document}
```

You can reduce typing effort even further by using the `table head` key, see section 1.5.

If you are not familiar with how to reference a label, see section 1.12.

1.2 Graphic

Use the `\includegraphicobject` command to insert a graphic. It is a wrapper around graphicx' `\includegraphics` command taking the same arguments. No need for a surrounding `figure` environment. I have extended the allowed optional keyword argument to also accept caption, label and more, see section 3.3.1. `details` are appended to the caption below the figure but not in the list of figures. Select with the `graphicx` or `graphbox` package options whether you want to use the commonly used graphicx package or it's extension graphbox.

```
\documentclass{article}
\usepackage{easyfloats}
\usepackage{hyperref}
\objectset[figure]{graphic width=.8\linewidth,
↪  graphic auto label prefix=fig:}

\begin{document}
\includegraphicobject[%
    caption = CTAN lion drawing by Duane Bibby,
    details = Thanks to \href
    ↪  {https://ctan.org/lion/files}{www.ctan.org}.,
]{graphics/ctan_lion}

\listoffigures
\end{document}
```

The graphic automatically gets the label `fig:graphics/ctan_lion`, based on the file name and the specified prefix. You can override the automatically generated label by passing the key `label` to the optional argument. If you omit `caption` the file name `graphics/ctan_lion` is used. See `auto label`, `auto caption`, `auto label strip path`, `auto caption strip path` and `auto label prefix`.

If you are not familiar with how to reference a label, see section 1.12.

1.3 Subfigures

The following example shows a figure consisting of two graphics displayed side by side, each having their own caption and label. The graphics have a distance of one quad, i.e. 1 em, and fill the entire width of the line. (em is a unit of distance relative to the font. If you are unfamiliar with the units supported by TeX you can read up on them in *The TeXbook* [3, pages 57 and 60] or *TeX by Topic* [4, chapters 8.2.1 and 4.3.1].

I am loading the calc package in order to use an expression as `subobject linewidth`. (Alternatively you could use the ε-TeX primitive `\dimexpr`.) As I want to put two graphics next to each other, each of them should be half of the available line width minus half of the distance between them wide. Additionally I need to subtract a tiny bit more in order to compensate for rounding errors in the division which would cause an uninteded line break with the initial setting `align = \centering`. (The rounding errors would not need to be compensated if you set the value of `align` to an empty string).

Pay attention to *not* insert an empty line between the subobjects, otherwise they will be placed below each other instead of side by side. If you want them to be placed below each other you can use the `ver` option.

By default subobjects do not show up in the list of figures. You can change that with `\captionsetup[sub]{list=true}`. `\captionsetup` is explained in the `caption` package documentation [5].

```
\documentclass{article}
\usepackage{easyfloats}
\usepackage{calc}

\graphicspath{{graphics/}}
\objectset{%
    subobject sep = \quad,
    subobject linewidth = .5\linewidth - .5em - .01pt,
    subpage outer pos = auto-inverted,
}
\objectset[figure]{%
    graphic width = \linewidth,
    graphic keepaspectratio,
    graphic auto label prefix = fig:,
}

\begin{document}
\begin{figureobject}{caption={Two subfigures},
↪  label=fig:subfigures}
    \includegraphicsubobject[caption={CTAN Lion}]{ctan_lion}
    \includegraphicsubobject[caption={\TeX\ engines}]{tex_engines}
\end{figureobject}
\end{document}
```

Each of the subfigures gets it's own label, generated based on the given prefix and the file name: `fig:ctan_lion` and `fig:tex_engines`. You can override the automatically generated labels with the `label` key if needed. How to reference the labels is shown in section 1.12.

If you want to use the full width for graphics in subobjects but less than the full width for graphics in main objects you can achieve that with

```
\objectset[figure]{graphic width = .8\linewidth}
\AtBeginSubobject{\AtBeginGraphicObject{%
    \objectset{graphic width = \linewidth}%
}}
```

1.4 Other subobjects

Use the `subobject` environment to combine two (or more) subobjects to one big object. The `contains subobjects` option causes the `env` option to be applied to the subobjects instead of the containing object. I am decreasing the `subobject linewidth` because the tables fill only a small part of the width so the distance between them would be too big if each was centered on `.5\linewidth`. (If you don't want to set the `subobject linewidth` to an explicit value take a look at the `subcaptionbox` option and the example provided there. Setting `subobject sep` to the desired distance and aligning the subojects with `subpage align=\raggedleft` and `subpage align=\raggedright` does not work well because `\raggedleft` and `\raggedright` do not affect the caption. The alignment of the caption could be changed with `\captionsetup{justification=raggedleft,singlelinecheck=false}` but then the caption would not be centered relative to the tabular.)

Pay attention to *not* insert an empty line between the subobjects, otherwise they will be placed below each other instead of side by side. If you want them to be placed below each other you can use the `ver` option.

By default subobjects do not show up in the list of tables. You can change that with `\captionsetup[sub]{list=true}`. `\captionsetup` is explained in the caption package documentation [5].

```
\documentclass{article}
\usepackage{easyfloats}

\objectset[table]{env=tabular}
\captionsetup[sub]{list=true}

\begin{document}
\begin{tableobject}{contains subobjects,
        caption = Two test tables,
        label = tab:abc-123,
        subobject linewidth = .25\linewidth,
    }
    \begin{subobject}{caption=Abc \& 123}{rl}
        \toprule
        abc & 123 \\
        de  & 45  \\
        f   & 6   \\
        \bottomrule
    \end{subobject}
    \begin{subobject}{caption=123 \& abc}{lr}
        \toprule
        123 & abc \\
        45  & de  \\
        6   & f   \\
        \bottomrule
    \end{subobject}
\end{tableobject}
\end{document}
```

1.5 Longtable

If you are undecided whether to use floating `tabular`s or `longtable`s which can break across pages you can use the following approach. Changing between them is as easy as changing `env=longtable` to `env=tabular` once. The table head and foot are set by the key `table head` and are by default formatted with the booktabs package. (If you don't like this you can change the definition of `table head` with `table head style`.) The column specification cannot be given as a separate argument (like in the example above) but must be set with the `arg` key because otherwise the column specification would be after the table head.

```
\documentclass{article}
\usepackage[longtable]{easyfloats}
\usepackage{siunitx}

\newcommand\pminfty{\multicolumn1r{$\pm\infty$}}

\objectset[table]{env=longtable}

\begin{document}
\begin{tableobject}{%
    caption = Trigonometric functions,
    label = tbl:trifun,
    arg = {
        S[table-format=2.0, table-space-text-post=\si{\degree}]
        ↪  <{\si{\degree}} !\quad
        *2{S[table-format=+1.2]}
        S[table-format=+4.2]
    },
    table head = \multicolumn1{c!\quad}{$x$} & $\sin x$
    ↪  & $\cos x$ & $\tan x$,
}

      0   &   0.00 &  1.00 &   0.00 \\
      5   &   0.09 &  1.00 &   0.09 \\
     10   &   0.17 &  0.98 &   0.18 \\
     15   &   0.26 &  0.97 &   0.27 \\
     20   &   0.34 &  0.94 &   0.36 \\
     25   &   0.42 &  0.91 &   0.47 \\
     30   &   0.50 &  0.87 &   0.58 \\
     35   &   0.57 &  0.82 &   0.70 \\
     40   &   0.64 &  0.77 &   0.84 \\
     45   &   0.71 &  0.71 &   1.00 \\
     50   &   0.77 &  0.64 &   1.19 \\
     55   &   0.82 &  0.57 &   1.43 \\
     60   &   0.87 &  0.50 &   1.73 \\
     65   &   0.91 &  0.42 &   2.14 \\
     70   &   0.94 &  0.34 &   2.75 \\
     75   &   0.97 &  0.26 &   3.73 \\
     80   &   0.98 &  0.17 &   5.67 \\
     85  &   1.00 &  0.09 &  11.43 \\
     90  &   1.00 &  0.00 & \pminfty \\
\end{tableobject}
\begin{tableobject}{%
    caption = Squared trigonometric functions,
    label = tbl:trifun2,
    arg = {
        S[table-format=2.0, table-space-text-post=\si{\degree}]
        ↪  <{\si{\degree}} !\quad
        *2{S[table-format=+1.2]}
        S[table-format=+4.2]
    },
    table head = \multicolumn1{c!\quad}{$x$} & {$\sin^2 x$}
    ↪  & {$\cos^2 x$} & {$\tan^2 x$},
}

      0  &   0.00 &  1.00 &   0.00 \\
      5  &   0.01 &  0.99 &   0.01 \\
     10  &   0.03 &  0.97 &   0.03 \\
     15  &   0.07 &  0.93 &   0.07 \\
     20  &   0.12 &  0.88 &   0.13 \\
     25  &   0.18 &  0.82 &   0.22 \\
     30  &   0.25 &  0.75 &   0.33 \\
     35  &   0.33 &  0.67 &   0.49 \\
     40  &   0.41 &  0.59 &   0.70 \\
     45  &   0.50 &  0.50 &   1.00 \\
     50  &   0.59 &  0.41 &   1.42 \\
     55  &   0.67 &  0.33 &   2.04 \\
     60  &   0.75 &  0.25 &   3.00 \\
     65  &   0.82 &  0.18 &   4.60 \\
     70  &   0.88 &  0.12 &   7.55 \\
     75  &   0.93 &  0.07 &  13.93 \\
     80  &   0.97 &  0.03 &  32.16 \\
     85  &   0.99 &  0.01 & 130.65 \\
     90  &   1.00 &  0.00 & \pminfty \\
\end{tableobject}
\begin{tableobject}{%
    caption = Cubed trigonometric functions,
    label = tbl:trifun3,
    arg = {
        S[table-format=2.0, table-space-text-post=\si{\degree}]
        ↪  <{\si{\degree}} !\quad
        *2{S[table-format=+1.2]}
        S[table-format=+4.2]
    },
    table head = \multicolumn1{c!\quad}{$x$} & {$\sin^3 x$}
    ↪  & {$\cos^3 x$} & {$\tan^3 x$},
}
```

```
      0  &   0.00 &  1.00 &   0.00 \\
      5  &   0.00 &  0.99 &   0.00 \\
     10  &   0.01 &  0.96 &   0.01 \\
     15  &   0.02 &  0.90 &   0.02 \\
     20  &   0.04 &  0.83 &   0.05 \\
     25  &   0.08 &  0.74 &   0.10 \\
     30  &   0.12 &  0.65 &   0.19 \\
     35  &   0.19 &  0.55 &   0.34 \\
     40  &   0.27 &  0.45 &   0.59 \\
     45  &   0.35 &  0.35 &   1.00 \\
     50  &   0.45 &  0.27 &   1.69 \\
     55  &   0.55 &  0.19 &   2.91 \\
     60  &   0.65 &  0.13 &   5.20 \\
     65  &   0.74 &  0.08 &   9.86 \\
     70  &   0.83 &  0.04 &  20.74 \\
     75  &   0.90 &  0.02 &  51.98 \\
     80  &   0.96 &  0.01 & 182.41 \\
     85  &   0.99 &  0.00 & 1493.29 \\
     90  &   1.00 &  0.00 & \pminfty \\
\end{tableobject}
\end{document}
```

1.6 Local definitions in tables

If you want to define a command locally for one table you cannot put it's definition in the first cell because each cell is a separate group (meaning that the definition will be forgotten at the end of the cell). Instead I provide the `exec` key whose value is executed inside of the object but before `env`. If you want to tinker around with catcodes keep in mind that arguments are always read entirely before expansion and execution. The ε-TEX primitive `\scantokens` can be useful to define active characters. If you are unfamiliar with how TEX processes a file you can read up on it in *TEX by Topic* [4, section 1].

```
\documentclass{article}
\usepackage{easyfloats}
\usepackage[table]{xcolor}

% avoid Warning: Font shape `OMS/cmtt/m/n' undefined
\usepackage[T1]{fontenc}
% change font to latin modern
% because the default font shows tilde as super script.
% lmodern can also help against unclean/pixelated fonts
% and trouble with copying ligatures from pdf.
% lmodern is relatively close to the default font.
\usepackage{lmodern}

\colorlet{rowbg}{gray!50}

\newcommand\charsym[1]{\texttt{#1}}
```

```
\newcommand\charname[1]{$\langle$#1$\rangle$}

\begin{document}
\begin{tableobject}{%
    caption = Category Codes,
    details = Highlighted catcodes have no tokens.,
    label = tbl:catcodes,
    env = tabular,
    arg = cll,
    table head = Catcode & Meaning & Characters,
    exec = {%
        \catcode`* = \active
        \scantokens{\def*{\rowcolor{rowbg}}}%
        \catcode`= = \the\catcode`&%
        \catcode`, = \the\catcode`&%
    },
}
    *  0 = Escape character,   \charsym\textbackslash       \\
       1 = Begin grouping,     \charsym\{                   \\
       2 = End grouping,       \charsym\}                   \\
       3 = Math shift,         \charsym\$                   \\
       4 = Alignment tab,      \charsym\&                   \\
    *  5 = End of line,        \charname{return}            \\
       6 = Parameter,          \charsym\#                   \\
       7 = Superscript,        \charsym\^                   \\
       8 = Subscript,          \charsym\_                   \\
    *  9 = Ignored character,  \charname{null}              \\
      10 = Space,              \charname{space} and
                               \charname{tab}               \\
      11 = Letter,             \charsym{a}--\charsym{z} and
                               \charsym{A}--\charsym{Z}     \\
      12 = Other,              other characters             \\
      % "In plain TeX this is only the tie character ~"
      % TeX by Topic, page 30
      13 = Active character,   \charsym{\string~}           \\
    * 14 = Comment character,  \charsym\%                   \\
    * 15 = Invalid character,  \charname{delete}            \\
\end{tableobject}
\end{document}
```

1.7 New object style / `tikzobject`

You can easily define new object environments. For more information see section 3.3.5.

```
\documentclass{article}
\usepackage{easyfloats}
\usepackage{tikz}
```

```
\NewObjectStyle{tikz}{type=figure, env=tikzpicture}
% I am not using `arg=[3D]` so that I can still pass an optional
↪ argument to tikz3dobject
\NewObjectStyle{tikz3d}{type=figure, env=tikzpicture,
↪ exec=\tikzset{3D}}

\tikzset{
    3D/.style = {
        x = {(-3.85mm, -3.85mm)},
        y = {(1cm, 0cm)},
        z = {(0cm, 1cm)},
    },
}
\objectset{warn no label=false}

\begin{document}
\begin{tikzobject}{caption=2D coordinate system}
    \newcommand\n{5}
    \newcommand\w{.075}
    \draw[->] (0,0) -- ++(\n,0);
    \draw[->] (0,0) -- ++(0,\n);
    \foreach \i in {1,...,\the\numexpr\n-1} {
        \draw (\i,0) +(0,\w) -- +(0,-\w);
        \draw (0,\i) +(\w,0) -- +(-\w,0);
    }
\end{tikzobject}
\begin{tikz3dobject}{caption=3D coordinate system}
    \newcommand\n{5}
    \newcommand\w{.075}
    \draw[->] (0,0,0) -- ++(\n,0,0);
    \draw[->] (0,0,0) -- ++(0,\n,0);
    \draw[->] (0,0,0) -- ++(0,0,\n);
    \foreach \i in {1,...,\the\numexpr\n-1} {
        \draw (\i,0,0) +(0,\w,0) -- +(0,-\w,0);
        \draw (0,\i,0) +(\w,0,0) -- +(-\w,0,0);
        \draw (0,0,\i) +(0,\w,0) -- +(0,-\w,0);
    }
\end{tikz3dobject}
\end{document}
```

If you are interested in three dimensional drawing see also the Ti*k*Z library perspective [6, section 63 Three Point Perspective Drawing Library].

1.8 New float type `diagram`

I am using `\AtEndPreamble` defined by the etoolbox package to defer loading of cleveref after hyperref. For more information about cleveref and varioref see section 1.12.

I am defining `\listofdiagrams` analogous to `\listoffigures` and `\listoftables` with the help of the `\listof` command defined by the float package (which is loaded automatically by easyfloats).

For more information see section 3.3.5.

```
\documentclass{article}

% ------- floats -------
\usepackage{easyfloats}
\usepackage[section]{placeins}
\usepackage{etoolbox}
\usepackage{varioref}
\AtEndPreamble{\usepackage{cleveref}}

% new float type "diagram"
\newfloat{diagram}{htbp}{lod}
\floatname{diagram}{Diagramm}
\NewObjectStyle{diagram}{%
    type = diagram,
    float style = plain,
    graphic auto label prefix = diagram:,
    graphic width = \linewidth,
}
\newcommand\includediagramobject[1][]{\includegraphicobject
↪  [diagram][#1]}
\newcommand\listofdiagrams{\listof{diagram}{List of Diagrams}}
\AtEndPreamble{\crefname{diagram}{Diagram}{Diagrams}}

% ------- TikZ -------
\usepackage{tikz}
\newcommand\defcalc[2]{\pgfmathparse{#2}\let#1=\pgfmathresult}
\ExplSyntaxOn
\let\trimspaces=\tl_trim_spaces:N
\ExplSyntaxOff

% ------- numbers and units -------
\usepackage{siunitx}

% ------- links -------
\usepackage{hyperref}

% ========== document ==========
\begin{document}
\section{\TeX}
\Vref{diagram:graphics/tex_engines} shows different \TeX\ engines.
\includediagramobject{graphics/tex_engines}

\section{Air}
\Vref{fig:air} shows the composition of air.
\begin{diagramobject}{caption=Composition of Air, label=fig:air,
↪  env=tikzpicture}
```

```
    \newcommand\composition{%
        78.1  / Nitrogen       / blue,
        20.87 / Oxygen         / green,
         1    / Argon          / orange,
         0.03 / Carbon dioxide / red
    }%
    \tikzset{draw and fill/.style={draw=#1,fill=#1!50}}
    \def\radius{3}
    \foreach \percentage/\element/\col [remember=\nextangle as
    ↪  \angle (initially 90)] in \composition {
        \trimspaces\col
        \defcalc\nextangle{\angle - \percentage*3.6}
        \draw[draw and fill=\col] (0,0) -- (\angle:\radius) arc
        ↪  (\angle:\nextangle:\radius) -- cycle;
    }
    \node at (\radius, 0) [overlay, xshift=2em, inner sep=0pt,
    ↪  anchor=west, text width=.5\linewidth] {
        \leavevmode\\  % empty line at the top to negate the empty
        ↪  line at the bottom
        \foreach \percentage/\element/\col in \composition {
            \trimspaces\col
            \trimspaces\element
            \tikz\node[draw and fill=\col, minimum width=1em,
            ↪  minimum height=1em]{};
            \element:~\qty{\percentage}{\percent} \\
        }
    };
\end{diagramobject}

\listofdiagrams
\end{document}
```

1.9 Split object

If a float gets too big you can split it up with the `\splitobject` command so that the object can be broken across a page break. I am using `\DeclareCaptionLabelFormat` and `\captionsetup` to add the prefix “Continued” to the caption of the continuation as shown in the package caption documentation [5, section 3.3 Continued floats].

```
\documentclass{article}
\usepackage{easyfloats}

\graphicspath{{graphics/}}
\objectset{%
    graphic width = \linewidth,
    graphic auto label = false,
}

\DeclareCaptionLabelFormat{continued}{Continued #1~#2}
```

```
\captionsetup[ContinuedFloat]{labelformat=continued}

\begin{document}
\begin{figureobject}{%
    caption = {A figure too big to fit on one page},
    label = fig:4-lions,
    subobject ver,
    subobject linewidth = .7\linewidth,
}
    \includegraphicsubobject[caption=Lion~1]{ctan_lion}
    \includegraphicsubobject[caption=Lion~2]{ctan_lion}

    \splitobject
    \includegraphicsubobject[caption=Lion~3]{ctan_lion}
    \includegraphicsubobject[caption=Lion~4]{ctan_lion}
\end{figureobject}
\end{document}
```

1.10 Custom options

This package uses the pgfkeys package to define the options which can be passed to the environments and commands defined by this package. Therefore you can use the usual pgfkeys handlers to add custom options. The following example defines a new key `center subcaptions` which centers the captions of all subobjects. For more information see section 3.1 and the *TikZ & PGF Manual* [6, Utilities/Key Management].

```
\documentclass{article}
\usepackage{easyfloats}
\usepackage{calc}

\objectset{%
    % define new option `center subcaptions`
    center subcaptions/.code =
        {\subcaptionsetup{justification=centering}},
    center subcaptions/.value forbidden,
    %
    % set predefined options
    warn no label = false,
    subobject sep = \quad,
    subobject linewidth = .5\linewidth - 1em,
    subpage outer pos = auto-inverted,
}

\begin{document}
\begin{figureobject}{center subcaptions, caption = {
        The caption of the main object is \emph{not} influenced
        by the custom \texttt{center subcaptions} key.
}}
```

```
    \begin{subobject}{caption={Left subobject}}
        \rule{\linewidth}{.75\linewidth}
    \end{subobject}
    \begin{subobject}{caption = {%
            Right subobject \\
            is influenced by \texttt{center subcaptions}
    }}
        \rule{\linewidth}{.75\linewidth}
    \end{subobject}
\end{figureobject}
\end{document}
```

1.11 Nonfloating objects

If your professor absolutely won't allow floating objects you can easily disable them globally (for all objects based on the `object` environment defined by this package which is internally used by `tableobject` and `\includegraphicobject`).

```
\objectset{placement=H}
```

1.12 How to reference objects

When you insert floating objects you probably want to reference them in the text. Although this package does not change anything about referencing I have added an example for the sake of completeness.

The standard LaTeX commands are `\ref{⟨label⟩}` (to insert the number of a figure/table/whatever) and `\pageref{⟨label⟩}` (to insert the page number). The hyperref package automatically turns them into links. The varioref package defines new commands `\vref{⟨label⟩}` and `\Vref{⟨label⟩}` which automatically add a reference to the page if the label is on a different page. The cleveref package redefines `\vref{⟨label⟩}` and `\Vref{⟨label⟩}` to automatically insert the type of object in front of the reference so that you don't need to write out "figure~".

`\Vref{⟨label⟩}` is to be used at the beginning of the sentence and `\vref{⟨label⟩}` is to be used inside of a sentence. The distinction is important for capitalization and whether to use an abbreviation. You can disable abbreviations by loading the cleveref package with the `noabbrev` option.

Note that cleveref must be loaded last, even after hyperref which should usually be loaded last.

```
\documentclass{article}
\usepackage{easyfloats}
\usepackage[hidelinks, pdfusetitle]{hyperref}
\usepackage{varioref}
\usepackage[nameinlink]{cleveref}

\graphicspath{{graphics/}}
\objectset{%
    placement = htbp,
    graphic auto label prefix = fig:,
    graphic width = .8\linewidth,
}

\begin{document}
    \section{Introduction}
    I will show a figure in \vref{sec:a-figure}.

    \clearpage

    \section{A~Figure}
    \label{sec:a-figure}
    \includegraphicobject{ctan_lion}

    \Vref{fig:ctan_lion} shows \TeX's mascot holding \emph{The
    ↪  \TeX book}.
    Notice that \vref{fig:ctan_lion} looks different in the middle
    ↪  of a sentence.
\end{document}
```

Warning: Using varioref can lead to not converging behavior. For example: You use `\vref{sec:foo}` in section 1 to reference `\label{sec:foo}` at the beginning of section 2. In the first run it prints "??" as reference because it does not know the referenced object yet. The reference is on page 1 and section 2 begins at the end of page 2. In the second run it therefore prints "section 2 on the following page". Section 2 moves to the beginning of page 3 because the reference text is now longer. In the third run it therefore prints "section 2 on page 3". Section 2 moves back to page 2 because the reference text is now shorter. In the fourth run it therefore prints "section 2 on the following page" again. And section 2 moves to the beginning of page 3 again. Every time you compile the document the beginning of section 2 will be somewhere else and the reference will never be correct. This may cause other references (e.g. `\pageref{LastPage}` (defined by the lastpage package) to jump as well. The same can happen when referencing an earlier label.

You don't need to worry about this until the very end because this behavior may get resolved while inserting, deleting or rewriting contents or formatting the document. But if this still happens when you have finished everything else you can solve this by inserting a `\clearpage` or replacing the `\vref` with a `\fullref`. `\fullref{⟨label⟩}` is defined by the varioref package and is by default equivalent to `\cref{⟨label⟩} on \cpageref{⟨label⟩}` (if cleveref is loaded—if not it's default is equivalent to `\ref{⟨label⟩} on page~\pageref{⟨label⟩}`). Using `\clearpage` may be preferable in order to keep the referencing manner consistent.

See also the varioref documentation, section 7 *A few warnings*.

2 Names

You have probably heard the term *floating object* or *float* for short. That is mainly what this package is about. However, I intended to avoid the term *floating* in the name of this package because this package also allows to globally disable the floating of those objects. Therefore I decided to name this package *objects*.

This name, however, has been rejected by TEX Live as being too generic. And they are right, especially for people with an object oriented programming background that name might be misleading. TEX Live has informed me that floating objects are still called floats even if they are technically not floating. Therefore I have decided to rename this package to *easyfloats*.

I have *not* changed the user interface because the package has already been online for more than half a year on my gitlab repository and I don't know how many people are using the package already. Therefore all commands and environments defined by this package still carry the old name *object* in them.

3 Documentation

This section contains the documentation on how to use this package.

Section 3.1 gives general information on options which environments and commands defined by this package may take. The options themselves are explained in sections 3.2 and 3.3 where the environments and commands defined by this package are explained.

Section 3.4 describes what is happening when loading this package. Section 3.5 describes the options which can be passed to `\usepackage` when loading this package.

Section 3.6 explains a few features which may help you to get a better understanding about what is going on. This might be useful if you run into unexpected errors or this package behaves different than you expected.

3.1 Options

The environments and commands defined by this package take options (implemented with the pgfkeys package). Options are a comma separated list of ⟨*key*⟩s or ⟨*key*⟩=⟨*value*⟩ pairs.

Which keys are allowed for which environment/command and which values are allowed for which key is specified in sections 3.2 and 3.3 where the environments and commands are documented. This section gives general information about these options.

This section does *not* apply to the package options which are explained in section 3.5.

3.1.1 Initial vs default values

I am using the words *initial value* and *default value* like they are used in the *TikZ & PGF Manual* [6].

The *initial* value of an option is the value which is used if the key is *not* given.

The *default* value of an option is the value which is used if the key is given without a value. Most keys don't have a default value, i.e. if you use the key you must explicitly give it a value.

3.1.2 Options scope

Setting an option always applies until the end of the current group. For the argument of an environment this is the corresponding `\end` command. For the argument of the `\includegraphicobject` command this is the end of this command. For the argument of `\objectset` this may be the end of the document.

If you are not familiar with the concept of groups in TEX *TEX by Topic* [4, chapter 10] is one possible place to read up on it.

3.1.3 Special characters in options

If a value contains a comma or an equals sign it must be wrapped in curly braces.

Spaces before and after a comma (separating an option) and before or after an equals sign (separating key and value) are ignored. However, a space after the opening brace is *not* ignored. So if you put the first key on the next line make sure to comment out the linebreak. If a leading or trailing space in a value is desired wrap the value in curly braces.

`\par` (aka an empty line) is forbidden in keys but allowed in values.

3.1.4 Key patterns

Sometimes I am talking about entire groups of keys instead of individual keys. I specify those groups with a pattern which matches the keys that I am referring to. In these patterns parenthesis stand for something optional and angular brackets for wildcards.

For example the pattern (⟨*env*⟩) `arg`(`s`) matches the keys `tabular* arg` and `args` (and many more) but not `env arg` because `env` is not an existing environment.

If a key has a version which ends on a `+` to append a value instead of replacing it the space in front of the `+` is optional.

3.1.5 Key name vs key path

pgfkeys organizes all keys “in a large tree that is reminiscent of the Unix file tree.” [6, page 954] The keys of this package are located in the three paths `/object`, `/subobject` and `/graphicobject`.

In error messages thrown by the pgfkeys package the full path of a key is shown.

When setting keys, however, you need not and should not specify the full path. The commands and environments of this package set the path automatically. Using full paths does not directly cause an error or a warning but trying to set options for a style or style group with `\objectset` causes undefined behavior.

Therefore, error messages thrown directly by this package omit the path and show the name of the key only.

3.1.6 Key types

In pgfkeys there are different types of keys. Which type a key belongs to is relevant for debugging if you want to check the value of a key, see section 3.6.

(sto) *storing key*: Keys of this type are like a variable. They store the given value. This value can be showed using the `.show value` handler (see section 3.6).

(exe) *executed key*: Keys of this type are like a function. They execute some predefined code and possibly take a value as argument.

(bool) *boolean key*: is a special case of an executed key which sets a plain TEX if command. This if command and it's meaning can be showed with the `.show boolean` handler (which is *not* contained in pgfkeys, I have defined it in this package).

The allowed values for a key of this type are `true` and `false`. The default value (i.e. the value which is assumed if the key is given without a value) is `true`.

(fwd) *forwarding key*: is a special case of an executed key which calls another key.

(hdl) *handler*: Keys defined in the path `/handlers`. They can be applied to other keys by appending them to the path. For users of this package they can be helpful for debugging. For example `\objectset{env/.show value}` shows the value of the key `env`.

The pgfkeys package also defines handlers which expand the value. I haven't come up with an example where this might be useful in the context of this package but e.g. `tabular arg/.expand once=\colspec,` works as expected.

(unk) *unknown key handler*: is a special key which is called if a given key does not exist and it's name is not a handler. I am using this to implement key patterns.

3.1.7 Styles

This package defines two styles, one for figures and one for tables.

You can think of these styles as an extension of the float package's float styles.

These styles are somewhat inspired by the pgfkeys styles but are different. They are neither set nor applied in the same way.

A style is a list of options which is not set immediately but locally for each object belonging to that style.

The options of a style can be set by passing the name of the style as an optional argument to the `\objectset` command, e.g. `\objectset[figure]{⟨options⟩}` or `\objectset[table]{⟨options⟩}`.

A style is applied by using the corresponding environment (e.g. `figureobject` or `tableobject`) or `\graphicobjectstyle{⟨style⟩}` for `\includegraphicobject`.

New styles can be defined with `\NewObjectStyle` as explained in section 3.3.5.

3.1.8 Style groups

This package defines one group of styles called `all` which contains all defined styles.

When setting options one can use a group name instead of a style name. In that case the options are set for all styles in the group.

3.1.9 Options processing order

1. Options set with `\objectset{⟨options⟩}` have the lowest priority.
2. Options set for a specific style with `\objectset[⟨styles⟩]{⟨options⟩}` take precedence because they are set later (at the object, not the `\objectset` command).
3. Options passed directly to the object have the highest priority.

For example:

```
\objectset[figure]{placement=p}
\objectset{placement=H}
\objectset[table]{placement=htbp}
```

Given the above preamble both figure- and tableobjects are floating. Tableobjects are allowed to be placed where they are specified in the source code. Figureobjects are put on a separate float page. The second line (which would disable floating) has no effect (unless you define a custom style) because it is overridden not only by the third but also the first line.

3.2 Environments

This package defines the following environments. Each of them takes exactly one mandatory argument, options as a comma separated key=value list.

3.2.1 `object` environment

Env `object` The `object` environment is used internally by `figureobject` and `tableobject`. Don't use this directly. You can define more environments like `figureobject` or `tableobject` with `\NewObjectStyle` if needed.

This environment redefines the `\caption` and `\label` commands to set the `caption`/ `label` option so that you can use them as usual except you cannot create several labels. If you really need several labels for the same object put the additional `\label` command(s) inside of the caption argument, there `\label` has it's original meaning. The location or the order of `\caption` and `\label` inside of the object environment is not relevant. Nevertheless I recommend to always put the `\label` after the `\caption` as it is usually required in order to get the references right (if you choose to use these commands instead of the options). Where the caption is typeset (above or below the object) is determined by the float style.

This environment takes the following options:

- `type = ⟨type⟩` (sto)

  The floating environment to use, e.g. `figure` or `table`.

- `float style = plain|plaintop|ruled|boxed|⟨empty⟩` (sto)
  Initial value: empty.

  How the object is supposed to look like, most importantly whether the caption is supposed to be above or below the object. See the float package for more information.

If the value is empty the float type is *not* restyled before the/each object. However, this package restyles `table` to `plaintop` and `figure` to `plain` when it is loaded. The reasoning is explained in [7].

- `caption = ⟨text⟩` (sto)

  The caption to place above or below the float.

  The appearance of the caption can be configured using `\captionsetup` defined by the `caption` package. The `caption` package is loaded automatically by this package. If you want to change the horizontal alignment of the caption take a look at the options `justification` and `singlelinecheck`.

- `list caption = ⟨text⟩` (sto)

  The caption to place in the list of ⟨*type*⟩s. If this is not given, the value of `caption` is used instead.

- `details = ⟨text⟩` (sto)

  This is appended to the caption which is placed above or below the object but not to the list of ⟨*type*⟩s.

  ```
  caption=CTAN lion drawing by Duane Bibby,
  details=Thanks to \url{www.ctan.org}.
  ```

  is equivalent to

  ```
  list caption=CTAN lion drawing by Duane Bibby,
  caption=CTAN lion drawing by Duane Bibby.
  ↪   Thanks to \url{www.ctan.org}.
  ```

- `details sep = ⟨text⟩` (sto)
  Initial value: a full stop followed by a space.

  The separator to be placed between caption and details if details are given.

- `label = ⟨label⟩` (sto)

  Defines a label to reference this object.

- `add label = ⟨label⟩` (sto)

  Defines an additional label which can be used synonymously to label. If this key is given several times, only the last one will have an effect.

- `placement = [htbp]+!?|H|I|⟨empty⟩` (sto)
  Initial value: empty.

  The optional argument passed to the floating environment. Allowed values:

  - any combination of the letters `htbp` (where no letter is occurring more than once), optionally combined with an exclamation mark. This means that the object will be a floating object. The order of the letters makes no difference. They have the following meanings:

    - `h`: LaTeX is allowed to place the object `h`ere, where it is defined.

    - `t`: LaTeX is allowed to place the object at the `t`op of a page.

    - `b`: LaTeX is allowed to place the object at the `b`ottom of a page.
    - `p`: LaTeX is allowed to place the object on a separate `p`age only for floats.
    - `!`: "LaTeX ignores the restrictions on both the number of floats that can appear and the relative amounts of float and non-float text on the page." [8, page 27]
  - `H`: LaTeX places the object exactly here, no matter how unfitting that may be. In contrast to a single `h` or `h!` where the object is still a floating object which may float somewhere else if it does not fit here, `H` means here and nowhere else. `H` is defined by the float package which is loaded by this package automatically.
  - `I`: LaTeX places the object exactly here, considering the current line width and indentation. `H` uses the full `\columnwidth`, just like a real float which floats here. `I` respects the indentation of an `itemize`, shrinking the max used width to `\linewidth`. This is a new placement defined by this package, using the code of `H` with two small changes. `I` can only be used with the environments and commands defined by this package, it cannot be passed to a normal floating environment directly.
  - empty: do *not* pass the optional argument. In this case the placement of the float can be changed using the `\floatplacement` command of the float package. I have defined this key instead of advertising `\floatplacement` because `\floatplacement` does not allow the value `H`.

- `align = ⟨code⟩` $^{\text{(sto)}}$
  Initial value: `\centering`.

  TeX code which is inserted at the beginning of the ⟨*type*⟩ environment.

- `exec = ⟨code⟩` $^{\text{(sto)}}$ / `exec += ⟨code⟩` $^{\text{(exe)}}$
  Initial value: empty.

  TeX code which is inserted at the beginning of the ⟨*type*⟩ environment before align. Can be used to define a command for this object, see section 1.6.

- `graphic ⟨option⟩ = ⟨value⟩` $^{\text{(unk)}}$

  Is applied to `\includegraphicobject` and `\includegraphicsubobject`. Is ignored for other objects.

  ⟨*option*⟩ can be any key which is unique to one of these two commands and any key allowed by the `\includegraphics` command (see graphicx/graphbox package). Unlike `\setkeys{Gin}{⟨options⟩}` this works with all keys (compare graphicx documentation [9, section 4.6], unfortunately it's not getting more specific than "Most of the keyval keys").

  I am checking if the key is existing immediately but I cannot check the value (only whether it is required). Therefore if you pass a wrong value the error message will not appear where you set this option but at the object where it is applied.

  If you set `graphic width` globally and want to override it locally you can use `graphic width=!`. This is a feature of the graphicx package but it is not well documented in it's documentation [9]. (Which is why I am mentioning it here.) The exclamation mark is mentioned for the `\resizebox` command.

- `env =` ⟨*env*⟩ (sto)
  Initial value: empty.

  The name of an additional inner environment in which the body is wrapped, e.g. `tabular`, `tabularx`, `tikzpicture`. If empty the body is *not* wrapped in another environment (additional to object).

  Please note that using this option can lead to difficult to find errors with confusing error messages if you forget that you used it or it has a different value than you think it has. In this case `show env args` may help you.

  Please note that due to the way how environments are implemented in LaTeX2 (this will change in LaTeX3 [10]) it is not possible to check whether a given name is an environment or a command. But if you pass something that is *not* defined you will get an error.

  If you have loaded the longtable package (either with the package option `longtable` or with a `\usepackage{longtable}`) you can set the value of this key to longtable. In that case the necessary changes are performed so that the content of this object environment is set in a `longtable` environment and does *not* float but can span across page breaks. In this case `type`, `placement` and `align` are ignored.

- ⟨*env*⟩ `arg =` ⟨*value*⟩ (unk)

  The value is wrapped in braces and passed as argument to the additional inner environment if the value of `env` is not empty and ⟨*env*⟩ equals the value of `env`. Arguments to this environment can be given as an argument to the `*object` environment as well but this key provides the possibility to pass arguments on a global level (or to override a globally passed argument). For example this can be used to give all tabularx-tables a consistent width:

```
% in preamble
\objectset[table]{tabularx arg=.8\linewidth}

% in document
\begin{tableobject}{caption=Test Table, label=tab1,
↪  env=tabularx}{XX}
    ...
\end{tableobject}
```

- ⟨*env*⟩ `args =` ⟨*value*⟩ (unk)

  Same like ⟨*env*⟩ `arg` except that the value is *not* wrapped in braces. This can be used to pass several arguments or an optional argument. Please not that this key cannot be used to pass exactly one undelimited argument consisting of more (or less) than one token because `\pgfkeys` (which I am using internally) strips several levels of braces.

- `arg =` ⟨*value*⟩ (unk)

  If `env` has a non-empty value this is an abbreviation of ⟨*env*⟩ `arg` where ⟨*env*⟩

is the value of `env`.

Please note that because this key depends on the value of another key the order in which these two keys are given is important.

The value of `env` is considered when this key is evaluated. If you use `\objectset[⟨styles⟩]{⟨options⟩}` (with it's optional argument) the processing of the keys is delayed but it makes some basic error handling already so that the line numbers are as fitting as possible. For this error handling only the options passed to this call of the command are considered. (Trying to consider previously set values correctly would make things more difficult because you might be applying these options to several styles at once where one might have `env` set and another not.) Therefore the following causes an error message:

```
\objectset[table]{env=tabularx}
\objectset[table]{arg=.8\linewidth}
```

While this would not:

```
\objectset{env=tabularx}
\objectset{arg=.8\linewidth}
```

Anyway, I recommend to always use this option directly after `env` (if you intend to use it). `env` and it's `args` belong together:

```
\objectset{env=tabularx, arg=.8\linewidth}
```

- `args = ⟨value⟩` (unk)

  If `env` has a non-empty value this is an abbreviation of `⟨env⟩ args` where `⟨env⟩` is the value of `env`. The notes on error handling of the `arg` key apply to this key as well.

- `(⟨env⟩) arg(s) += ⟨value⟩` (unk)

  A plus sign can be appended to the key (patterns) `⟨env⟩ arg`, `⟨env⟩ args`, `args` and `arg`. In that case a possibly previously passed argument is not overridden but this value is appended to it. For example the following pattern allows to easily switch between tabular and tabularx tables on a global level:

```
% in preamble
\objectset[table]{tabularx arg=.8\linewidth, env=tabularx}
\newcolumntype{Y}{>{\raggedleft\arraybackslash}X}

% in document
\begin{tableobject}{caption=Test Table, label=tab1,
↪  tabular arg=lr, tabularx arg+=XY}
    ...
\end{tableobject}
```

- `first head = ⟨code⟩` (sto)

  Is inserted at the beginning of the object (if `env` is non-empty: inside of the inner environment and after possibly specified `(⟨env⟩) arg(s)`). If this is not given, `head` is used instead.

- `last foot = ⟨code⟩` (sto)

  Is inserted at the end of the object (if `env` is non-empty: inside of the inner environment). If this is not given, `foot` is used instead.

- `head = ⟨code⟩` (sto)
  Initial value: empty.

  This value is used for `first head` if `first head` is not given. If `env=longtable` this is the head after a pagebreak inside of the table.

- `foot` (sto)
  Initial value: empty.

  This value is used for `last foot` if `last foot` is not given. If `env=longtable` this is the foot before a pagebreak inside of the table.

- `table head = ⟨code⟩` (exe)

  This is a convenience key which sets `first head`, `last foot`, `head` and `foot`. The value is the column headers without rules/lines and without the trailing `\\`.

- `table break text = ⟨text⟩` (exe)
  Initial value: `(to be continued)`.

  A text put in the `foot` by `table head`.

- `table head style = ⟨code⟩` (exe)

  Defines how `table head` fills out `first head`, `last foot`, `head` and `foot`.

  Initial value:

```
{%
    first head =
        \toprule
        #1 \\
        \midrule,
    head =
        #1 \\
        \midrule,
    foot =
        \midrule
        \ifx\object@tableBreakText\@empty
        \else
            \multicolumn{\the\LT@cols}{r@\relax}
            ↪  {\object@tableBreakText}%
        \fi,
    last foot =
        \bottomrule,
}
```

(Note the curly braces which are required because the value contains commas and equal signs, see section 3.1.3. `\the\LT@cols` is the number of columns of the longtable and `\object@tableBreakText` is the value of `table break text`. Commands containing an @ in their name are internal commands and can only be used between `\makeatletter` and `\makeatother`, see also [11].)

- `show env args = true|false` (bool)
  Default value: `true`. Initial value: `false`.

  Show the code which is assembled from the `env` and (⟨*env*⟩) `arg(s) (+)` keys before executing it. See section 3.6. Please note that arguments may be given as additional arguments and not as (⟨*env*⟩) `arg(s) (+)` like in `\begin{tableobject}{env=tabular}{cl}`. Such arguments are *not* shown by this key. This key applies to subobjects as well.

- `warn no caption = true|false` (bool)
  Default value: `true`. Initial value: `true`.

  Give a warning if `caption` is *not* given.

- `warn no label = true|false` (bool)
  Default value: `true`. Initial value: `true`.

  Give a warning if `label` is *not* given.

- `warn other env = true|false` (bool)
  Default value: `true`. Initial value: `false`.

  Give a warning when ⟨*env*⟩ `args` is given if `env` does not have the value ⟨*env*⟩ and the value of `env` is not empty. This applies to subobjects as well.

  The `\objectset` command if used with it's optional argument does not set the options immediately but stores them in different macros for different object styles. Therefore if you change this value for certain styles this change does not affect following `\objectset` commands. Without the optional style argument, however, the change takes effect immediately.

  In order to avoid duplicates this warning is printed only where the key is passed by the user and *not* where it is applied implicitly because of a previous `\objectset[`⟨*styles*⟩`]{`⟨*options*⟩`}`.

- `contains subobjects = true|false` (bool) / `sub = true|false` (fwd)
  Default value: `true`. Initial value: `false`.

  Specifies that this object contains subobjects, see section 3.2.4. Is relevant only if `env` is set. The value of `env` is applied to the subobjects instead of this object. This is not executed immediately but only after all options have been processed so that you do not need to pay special attention to pass `env` before `contains subobjects`.

  If this is *not* given (or more precisely: if this is false) and the value of `env` is *not* empty I look ahead whether the object contains a subobject. If I find a subobject I pretend you had passed this option and print a warning. I insist on you explicitly passing this option because the lookahead does not work in all situations. It ignores space and `\par` tokens but if there is any other token before the subobject, for example a `\small` to fit two tables side by side which are a little too wide (which may not be the best solution but an easy quick fix) or a `\typeout` for debugging, the lookahead does not find the subobject (possibly) resulting in unpredictable errors. For example if you set `env=tabular` it will most likely complain about an "`Illegal pream-token`" or about a "`Missing number, treated as zero`" with `env=tabular*` because the required arguments are missing.

All `(⟨env⟩) arg(s) (+)` options apply to subobjects as well.

Additionally the following options are passed through to the corresponding options of all subobjects inside of this object, they are all forwarding keys. See `subobject` environment.

- `subobject linewidth = ⟨dimen⟩` (fwd)
- `subobject sep = ⟨code⟩` (fwd)
- `subobject hor = ⟨code⟩` (fwd)
- `subobject hor sep (+)= ⟨code⟩` (fwd)
- `subobject ver = ⟨code⟩` (fwd)
- `subobject ver sep (+)= ⟨code⟩` (fwd)
- `subobject exec (+)= ⟨code⟩` (fwd)
- `subobject env = ⟨env⟩` (fwd)
- `subcaptionbox` (fwd)
- `subcaptionbox inner pos = c|l|r|s|⟨empty⟩` (fwd)
- `subpage` (fwd)
- `subpage outer pos = c|t|b|T|B|auto|Auto|auto-inverted|Auto-inverted|⟨empty⟩` (fwd)
- `subpage height = ⟨dimen⟩` (fwd)
- `subpage inner pos = c|t|b|s|⟨empty⟩` (fwd)
- `subpage align = ⟨code⟩` (fwd)
- `subobject warn no caption = true|false` (fwd)
- `subobject warn no label = true|false` (fwd)

You can also define your own options as shown in section 1.10.

3.2.2 `figureobject` environment

Env `figureobject` Is used for inserting figures. Takes the same options like the `object` environment. It differs in the following initial values:

- `type=figure`

3.2.3 `tableobject` environment

Env `tableobject` Is used for inserting tables. Takes the same options like the `object` environment. It differs in the following initial values:

- `type=table`

3.2.4 `subobject` environment

Env `subobject` To be used inside of an `*object` environment if you want to place several images/tables/whatever together. See also `\includegraphicsubobject`.

I recommend to *not* put anything between the subobjects manually so that you can control their positioning with the `hor` and `ver` options. (Spaces after a subobject are ignored but empty lines are not.)

Unlike the `object` environment, `\caption` and `\label` *cannot* be used inside of the subobject environment. Use the `caption` and `label` options instead.

There are two different backends available, both provided by the subcaption package. See the `subcaptionbox` and `subpage` keys.

The `subobject` environment has exactly one mandatory argument, a comma separated list of the following options.

The following options correspond to those of an `object`. See section 3.2.1.

- `label = ⟨label⟩` $^{\text{(sto)}}$
- `caption = ⟨text⟩` $^{\text{(sto)}}$
- `list caption = ⟨text⟩` $^{\text{(sto)}}$

  (The subcaption package disables subcaptions in the list of figures/tables/whatever by default. To enable them use `\captionsetup[sub]{list=true}`.)
- `details = ⟨text⟩` $^{\text{(sto)}}$
- `details sep = ⟨text⟩` $^{\text{(sto)}}$
- `exec = ⟨code⟩` $^{\text{(sto)}}$ / `exec += ⟨code⟩` $^{\text{(exe)}}$
- `graphic ⟨option⟩ = ⟨value⟩` $^{\text{(unk)}}$

  (This key is completely useless. It only has a meaning in the context of `\includegraphicsubobject` but there these options can be used directly without the prefix `graphic`. I am allowing it anyway in order to support the same key like in `\objectset` which is supported by `\includegraphicobject` as well.)
- `env = ⟨env⟩` $^{\text{(sto)}}$

  (See also the `contains subobjects` option of the `object` environment.)
- `(⟨env⟩) arg(s) (+) = ⟨value⟩` $^{\text{(unk)}}$

  (All values passed to the corresponding keys of the `object` environment apply to this option, too.)
- `warn no caption = true|false` $^{\text{(bool)}}$
- `warn no label = true|false` $^{\text{(bool)}}$
- `warn other env = true|false` $^{\text{(fwd)}}$
- `show env args = true|false` $^{\text{(fwd)}}$

The following options are unique for the `subobject` environment:

- `linewidth = ⟨dimen⟩` (sto)
  Initial value: `.5\linewidth`.

  The horizontal space available for the subobject. The content of the subobject is centered within this width. If two subobjects displayed side by side have a small width they may appear too far apart from each other with the initial value. Then you can decrease this value so that they come closer together. (With `subcaptionbox` this value may be empty. In that case the subobject takes as much space as it needs and `\linewidth` inside of the subobject is the same like in the parent object.)

  If you want to place more than two subobjects side by side you must decrease this value accordingly. Keep in mind that you need to consider the width of `hor sep` as well if you changed it.

  Dimensions can be given relative to other dimensions or in numbers. Aside from absolute units like `pt` or `cm` TEX also recognizes units relative to the current font size: `em` and `ex`. For more information on dimensions see *The TEXbook* [3, chapter 10] or *TEX by Topic* [4, chapter 8].

- `sep = ⟨code⟩` (sto)

  A separator which is inserted before each subobject except for the first subobject inside of the current parent object.

- `hor = ⟨code⟩` (exe)
  Default value: empty.

  Set the value of `sep` to the value of `hor sep` so that the subobjects are placed side by side. If you pass a value the value will be appended to `sep` after setting it to `hor sep`.

  Please note that options are only valid until the end of a group. Therefore if you use this inside of a subobject it does *not* apply for the following subobject. Instead use `subobject hor` on the parent object.

- `hor sep = ⟨code⟩` (sto) / `hor sep += ⟨code⟩` (exe)
  Initial value: empty.

  The separator to be used if the subobjects are suppossed to be placed side by side.

  Please note that `hor` must be used *after* setting this key, otherwise this option will not take effect.

- `ver = ⟨code⟩` (exe)
  Default value: empty.

  Set the value of `sep` to the value of `ver sep` so that the subobjects are placed below each other. If you pass a value the value will be appended to `sep` after setting it to `ver sep`.

  Please note that options are only valid until the end of a group. Therefore if you use this inside of a subobject it does *not* apply for the following subobject. Instead use `subobject ver` on the parent object.

- `ver sep` = ⟨*code*⟩ (sto) / `ver sep +=` ⟨*code*⟩ (exe)
  Initial value: `\par\bigskip`.

  The separator to be used if the subobjects are suppossed to be placed below each other.

  Please note that `ver` must be used *after* setting this key, otherwise this option will not take effect.

- `subcaptionbox` (exe)

  The subcaption package provides several possibilities to insert subobjects. This option tells the subobject environment to use the `\subcaptionbox` command instead of the `subfigure` or `subtable` environment, see option `subpage`. (This key does *not* take a value.)

  This option allows to pass an empty value to `linewidth`. It can be useful if you have subobjects with a small width so that you don't need to try different `subobject linewidth`s. The example in section 1.4 could be rewritten as following:

```
\documentclass{article}
\usepackage{easyfloats}

\objectset[table]{%
    env = tabular,
    subcaptionbox,
    subobject linewidth =,
    subobject hor = \qquad,
}
\captionsetup[sub]{list=true}

\begin{document}
\begin{tableobject}{contains subobjects,
        caption = Two test tables,
        label = tab:abc-123,
    }
    \begin{subobject}{caption=Abc \& 123, arg=rl}
        \toprule
        abc & 123 \\
        de  & 45  \\
        f   & 6   \\
        \bottomrule
    \end{subobject}
    \begin{subobject}{caption=123 \& abc, arg=lr}
        \toprule
        123 & abc \\
        45  & de  \\
        6   & f   \\
        \bottomrule
    \end{subobject}
\end{tableobject}
```

```
\end{document}
```

Note that this works only if the subobject captions are very short. If they are wider than the content the line breaks which looks ugly.

If you want to use this option with `env=tabular` (or similar) you must pass the column specification with the option `arg=lr` (instead of as a separate argument). Otherwise you will get the error message “`Package array Error: Illegal pream-token (\BODY): ‘c’ used.`”

This option is *not* compatible with `env=tabularx` and does *not* allow verbatim content inside of the subobject. You can try to work around the verbatim limitation (1) by saving the problematic content in a macro before hand using commands such as `\urldef` from the url package, (2) by using commands which replace tokens such as `\mintinline` from the minted package (does not work with catcodes which do not produce tokens – most importantly the comment character `%`), (3) by changing the problematic catcodes before the `subobject` (but you still need *some* escape, begin group and end group characters in order to start and end the subobject – but which do not need to be the default characters `\`, `{` and `}`), (4) by defining a macro before hand under what ever catcode regime you want, you can even append contents from yet another catcode regime using the `\gappto` command from the etoolbox if you want. If you want to learn more about catcodes (category codes) see *The TEXbook* [3, chapter 7] or *TEX by Topic* [4, chapter 2]. But the easier way is probably to use `subpage` instead – that’s why it’s the initial backend.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{easyfloats}
\objectset{subcaptionbox}

\NewDocumentCommand
    \verbdef {O{\def}mv} {#1#2{#3}}

\verbdef\leftcontent.\verb|\ { } $ & # ^ _ % ~|.
\verbdef\rightcontent|\NewDocumentCommand\verbdef|
\appto\rightcontent{\\\quad}
\verbdef[\appto]\rightcontent|{O{\def}mv}{#1#2{#3}}|

\begin{document}
\begin{figureobject}{caption=subcaptionbox,
↪  label=fig:subcaptionbox}
    \begin{subobject}{caption=Limitation}
        \raggedright
        You cannot write \\
        \texttt{\leftcontent}
        \par
    \end{subobject}
    \begin{subobject}{caption=Workaround}
        \raggedright
        But you can work around it with \\
        \texttt{\rightcontent}
        \par
    \end{subobject}
\end{figureobject}
\end{document}
```

`\NewDocumentCommand` is documented in [12].

- `subcaptionbox inner pos = c|l|r|s|⟨empty⟩` (sto)

  The horizontal position of the content in the box. Also allowed is any justification defined with `\DeclareCaptionJustification` (see the caption package documentation). An empty value means that this optional argument is *not* passed to the `\subcaptionbox` command. This option has no effect if `linewidth` is empty. I discourage using this option because it destroys the alignment of (sub)object and (sub)caption.

- `subpage` (exe)

  This is (after `subcaptionbox`) the second and initial backend for the `subobject` environment. It uses the `subfigure`/`subtable` environment defined by the subcaption package. (Since version 1.5 the subcaption package also provides the more generic `subcaptionblock` environment but this package still uses the old `subfigure` and `subtable` environments. `subfigure` and `subtable` are nowadays specialized versions of `subcaptionblock` which have no other difference than that they produce if a warning if they are used with a different main object type.) (This key does *not* take a value.)

  The `subfigure` and `subtable` environments are minipages and take the same arguments which can be set with `linewidth`, `subpage outer pos`, `subpage height` and `subpage inner pos`. Both the content and the caption go inside of the minipage.

  If the subobjects are placed next to each other this option requires that either the contents or the captions of the subobjects have the same height and that `subpage outer pos` is set accordingly. Otherwise the captions will not be aligned.

- `subpage outer pos = c|t|b|T|B|auto|Auto|auto-inverted|Auto-inverted|⟨empty⟩` (sto)
  Initial value: `auto`.

  The vertical position of the minipage (containing both content and caption) on the baseline.

  Additionally to the values `t`, `c` and `b` supported by the minipage environment the subcaption package v1.2 adds the allowed values `T` and `B` and this key also allows the values `auto`, `Auto`, `auto-inverted`, `Auto-inverted` and empty.

  While `t` and `b` align the top/bottom *baseline* of the content `T` and `B` align the very top/bottom of the content.

  `c` aligns the center of the content.

  `auto` means `t` if the caption is displayed at the top or `b` if the caption is displayed at the bottom so that the captions are aligned if the *captions* have the same height.

`auto-inverted` means `b` if the caption is displayed at the top or `t` if the caption is displayed at the bottom so that the captions are aligned if the *contents* have the same height (although the height of the top most row does not count in this regard so `auto-inverted` is also good for graphics with different heights).

`Auto` and `Auto-inverted` behave like their lower case versions but use `T`/`B` instead of `t`/`b`.

If content and captions have different heights `auto-inverted` gives good results for `\includegraphicsubobject`. For tables `subcaptionbox` is required to align the caption (but note the limitations regarding verbatim content). If `subcaptionbox` does not work for you you can try `subpage outer pos = c` or giving the shorter caption the same height with `\\~` and then using `subpage outer pos = auto`.

If a subobject has neither caption nor label `auto` may not work as expected. Instead `Auto` can be used which is based on `T` and `B` instead of `t` and `b`. Note that `Auto` requires version 1.2 or newer of the subcaption package.

Empty is equivalent to `c`.

Except for the values which are added by this package (e.g. `auto` and `auto-inverted`) all other values are passed through to the `subfigure`/`subtable` environment which silently ignores invalid values.

- `subpage height =` ⟨*dimen*⟩ (sto)
  Initial value: empty.

  The height of the minipage. An empty value means that this optional argument is *not* passed to the `subfigure`/`subtable` environment.

  Dimensions can be given relative to other dimensions or in numbers. Aside from absolute units like `pt` or `cm` TEX also recognizes units relative to the current font size: `em` and `ex`. For more information on dimensions see *The TEXbook* [3, chapter 10] or *TEX by Topic* [4, chapter 8].

- `subpage inner pos = c|t|b|s|`⟨*empty*⟩ (sto)
  Initial value: empty.

  The vertical position of the content on the minipage. Empty means that this optional argument is *not* passed to the `subfigure`/`subtable` environment. This option has no effect if `subpage height` is empty.

- `subpage align =` ⟨*code*⟩ (sto)
  Initial value: `\centering`.

  TEX code which is inserted at the beginning of the `subfigure`/`subtable` environment.

If you want to change the numbering of subobjects please refer to the subcaption package documentation [13, section 5 *The `\DeclareCaptionSubType` command*]. The subcaption package is loaded automatically by this package.

3.3 Commands

In this section I am describing the commands defined by this package.

3.3.1 `\includegraphicobject` command

`\includegraphicobject`

`\includegraphicobject{⟨filename⟩}`
`\includegraphicobject[⟨options⟩]{⟨filename⟩}`
`\includegraphicobject[⟨style⟩][⟨options⟩]{⟨filename⟩}`

Is used for inserting graphics from a different file. It is very similar to graphicx' `\includegraphics` command, except that the graphic is automatically set in a `figureobject` environment. You can change this by setting the object style with `\graphicobjectstyle` or an additional optional argument given *before* the usual optional argument. The mandatory argument is the same: The name of the graphics file to include *without* the file extension. The optional argument accepts—aside from all the options defined by graphicx/graphbox—also all options of the `figureobject` environment. Additionally there are the following unique options:

- `auto caption = true|false` (bool)
  Default value: `true`. Initial value: `true`.

  If no `caption` is given the file name is used as caption. All underscores in the file name are replaced by `\textunderscore`. This option is intended to be used on a global level but works in the optional argument of this command as well.

- `auto caption strip path = true|false` (bool)
  Default value: `true`. Initial value: `false`.

  If `auto caption` is true and the file name is used as caption a possibly leading path is stripped (everything before and including the last slash in ⟨*filename*⟩). This is initially false because I am assuming that in most cases where the path should not be displayed `\graphicspath{{path/}}` would be used.

- `auto label = true|false` (bool)
  Default value: `true`. Initial value: `true`.

  If no `label` is given the file name is used as label. This option is intended to be used on a global level but works in the optional argument of this command as well.

- `auto label strip path = true|false` (bool)
  Default value: `true`. Initial value: `false`.

  If `auto label` is true and the file name is used as label a possibly leading path is stripped (everything before and including the last slash in ⟨*filename*⟩). This is initially false because I am assuming that in most cases where the filename without path is unique `\graphicspath{{path/}}` would be used.

- `auto label prefix = ⟨prefix⟩` (sto)
  Initial value: ⟨*empty*⟩.

  If `auto label` is true and the file name is used as label prepend ⟨*prefix*⟩ to the label.

- `warn env = true|false` (bool)
  Default value: `true`. Initial value: `true`.

  Give a warning if `env` is not empty.

- `no env = true|false` (bool)
  Default value: `true`. Initial value: `true`.

  Reset `env` to an empty value. This happens after evaluating `warn env`.

- `before graphic = ⟨code⟩` (sto)
  Initial value: ⟨*empty*⟩.

  Arbitrary TEX code to be inserted before the `\includegraphics`.

- `after graphic = ⟨code⟩` (sto)
  Initial value: ⟨*empty*⟩.

  Arbitrary TEX code to be inserted after the `\includegraphics`.

You may not use this command inside of an `*object` environment. Otherwise you will get an "`object environment may not be nested`" error. See also `\includegraphicsubobject`.

3.3.2 `\includegraphicsubobject` command

`\includegraphicsubobject`

`\includegraphicsubobject{⟨filename⟩}`
`\includegraphicsubobject[⟨options⟩]{⟨filename⟩}`

To be used if you want to place several graphics from different files in one object.

It takes the same options like `\includegraphicobject` except that it takes options for the `subobject` environment instead of options for the `object` environment. Also it does *not* take the optional ⟨*style*⟩ argument.

You may not use this command outside of an `*object` environment. Otherwise you will get a "`subobject environment may not be used outside of an object`" error. See also `\includegraphicobject`.

3.3.3 `\splitobject` command

`\splitobject`

`\splitobject`

This can be used inside of an `object` environment which is too big to fit on one page. It splits an object so that it can be broken across pages. This command must be used directly inside of the object, it cannot be used inside of subgroups/environments inside of the object. It uses the `\ContinuedFloat` command defined by the caption package. See the example in section 1.9.

The `label` is placed on the first part of the object. If you want to specify the second part of the object you can use the `\label{⟨second-label⟩}` command after `\splitobject`.

3.3.4 Setting options globally

`\objectset`

`\objectset{⟨options⟩}`
`\objectset[⟨styles⟩]{⟨options⟩}`

Sets the passed options for all following objects until the end of the current group. All options of the `object` environment are allowed.

A comma separated list of styles or style groups can be given in an optional argument. In that case the options are not set immediately but appended to the specified style(s). The options are set locally for any following object of the specified style(s) in the same group. Although setting the options is delayed the options are checked immediately so that error messages and warnings point to the line where the option is specified in the code, not where it is technically set. (In order for that to work properly it is important that options are specified with the key name only and not with the full path, see section 3.1.5.) However, the value can usually *not* be checked immediately, only whether it is required or not. Therefore if you pass a wrong value the error message will not appear where you set this option but at the object where it is applied. An exception is the key `env` where the value is checked immediately for plausibility whether it might be the name of an environment.

If ⟨*styles*⟩ is empty or an empty group the options are not applied. No error or warning is printed.

There is a style group called `all` which all styles belong to. `\objectset{⟨options⟩}` and `\objectset[all]{⟨options⟩}` are mostly equivalent except that the former (without optional argument) is more efficient because it sets the options immediately and the latter (with the optional argument given) is able to override options set for a style.

`\graphicobjectstyle` `\graphicobjectstyle{⟨style⟩}` can be used to change the object style used by `\includegraphicobject`. For example, if you have a single table in a file called *catcodes.pdf* you can insert it as following. Alternatively, you can use the optional ⟨*style*⟩ argument.

```
\begingroup
\graphicobjectstyle{table}
\includegraphicobject[caption=Catcodes]{catcodes}
\endgroup
```

`\graphicspath` `\graphicspath{{path/}}`: see `graphicx` package documentation [9, section 4.5].

3.3.5 New object styles and types

This section explains how to define a new object ⟨*style*⟩ in the sense of section 3.1.7. It is *not* about how to define a new ⟨*floatstyle*⟩ which can be used as value for the `float style` key. Examples are given in section 1.7 and section 1.8.

`\NewObjectStyle` `\NewObjectStyle{⟨style⟩}{⟨options⟩}` defines a new environment called ⟨*style*⟩`object` analogous to `figureobject` and `tableobject`. ⟨*options*⟩ are set for the new object style as if you had used `\objectset[⟨style⟩]{⟨options⟩}`. You must always specify the `type`. If this package is loaded without `allowstandardfloats` the float environment which is passed to `type` is redefined to issue a warning that ⟨*style*⟩`object` should be used instead. This warning should not influence the environment's usual behavior. If the float environment was already passed as `type` to a previous call of `\NewObjectStyle` it is not redefined again but ⟨*style*⟩`object` is appended to the list of replacement suggestions.

If you define a new object style you may also want to define a new float type. The float package (which is automatically loaded by this package) defines the following command for doing so:

`\newfloat` `\newfloat{⟨type⟩}{⟨placement⟩}{⟨ext⟩}[⟨within⟩]`

- ⟨*type*⟩ is the floating environment to be defined. This value is also used as the float name which is displayed in front of the caption, therefore it should be capitalized. Alternatively the name can be changed using `\floatname{⟨type⟩}{⟨name⟩}`.
- ⟨*placement*⟩ is the value to be used if the `placement` key is not given (or has an empty value). This is initially `tbp` for the standard float types.
- ⟨*ext*⟩ is the extension of a file used to save the list of ⟨*type*⟩s. This is `lof` (list of figures) for `type=figure` and `lot` (list of tables) for `type=table`. This file extension should be unique.
- ⟨*within*⟩ is a counter whose value is prepended to the ⟨*type*⟩ counter. The ⟨*type*⟩ counter is reset every time the ⟨*within*⟩ counter is incremented.
- Make sure an appropriate default float style is active when using `\newfloat`. The default float style can be activated using `\floatstyle{⟨floatstyle⟩}`, see the float package documentation [14]. It should be `plain` for something like an image or `plaintop` for something like a table. The reasoning is explained in [7]. Alternatively you can specify the float style using the `float style` key in the ⟨*options*⟩ of `\NewObjectStyle`.

`\crefname` `\Crefname` If you are using the cleveref package (which I recommend you do) you also need to tell it how to reference the new float type with `\crefname{⟨type⟩}{⟨singular⟩}{⟨plural⟩}`. If you want an abbreviation in the middle of a sentence but the full word at the beginning of the sentence you specify the beginning of the sentence with `\Crefname` and the middle of the sentence with `\crefname`.

`\NewObjectStyle` automatically defines the corresponding environment needed for `subobject` if possible, i.e. if the caption package is new enough. If the caption package is older than August 30, 2020 and you want to use subobjects you need to define the subtype manually by putting the following line *before* loading this package [15]:

```
\AtBeginDocument{\DeclareCaptionSubType{⟨type⟩}}
```

`\trivfloat` The trivfloat package provides the `\trivfloat{⟨type⟩}` command which is an easier alternative to `\newfloat`. If you use it you should be aware that it does not define the new float type environment immediately but at `\begin{document}`. This does *not* affect `\NewObjectStyle` (you can still use it directly afterwards) but it means that the float style active at `\begin{document}` is applied and not the float style active at `\trivfloat`. Therefore I recommend to pass the `float style` option to `\NewObjectStyle`, then it does not matter which float style was active when the float type was defined because it is restyled before each use of an object where this option applies. `\trivfloat` must be used before `\AtBeginDocument{\DeclareCaptionSubType{⟨type⟩}}`.

`\DeclareFloatingEnvironment` The newfloat package provides the `\DeclareFloatingEnvironment[⟨options⟩]{⟨type⟩}` command which is a newer alternative to `\newfloat` and `\trivfloat`. With it's key=value options it is more intuitive than `\newfloat` and more flexible than `\trivfloat`. Unlike `\newfloat` and `\trivfloat` it automatically capitalizes ⟨*type*⟩ before using it as float name. It seems to ignore `\floatstyle` so you need to specify that in the options. The newfloat package is written by the same author like the subcaption package so you don't need to worry about defining subtypes manually.

3.3.6 New object style groups

Several object styles can be combined to a group. You can set options for all styles contained in a group using `\objectset[⟨group⟩]{⟨options⟩}`.

`\NewObjectStyleGroup`

`\NewObjectStyleGroup{⟨group⟩}{⟨styles*⟩}` defines a new style group consisting of the styles ⟨*styles**⟩. ⟨*styles**⟩ is a comma separated list of styles. In contrast to ⟨*styles*⟩ it may *not* contain style groups.

`\AddObjectStyleToGroup`

`\AddObjectStyleToGroup{⟨group⟩}{⟨style⟩}` adds an existing style to an existing group.

3.3.7 Hooks

This package provides several commands similar to `\AtBeginDocument` which take one argument, TeX code which is executed at a later point in time.

`\AtBeginObject`

`\AtBeginObject{⟨code⟩}` runs ⟨*code*⟩ every time at the begin of an `object` environment (including `figureobject`, `tableobject` and `\includegraphicobject`). This hook is inside of the group but before any options are processed.

`\AtBeginSubobject`

`\AtBeginSubobject{⟨code⟩}` runs ⟨*code*⟩ every time at the begin of a `subobject` environment (including `\includegraphicsubobject`). This hook is inside of the group but before any options are processed.

`\AtBeginGraphicObject`

`\AtBeginGraphicObject{⟨code⟩}` runs ⟨*code*⟩ every time in `\includegraphicobject` and `\includegraphicsubobject`. This hook is after the object/subobject hook but before any options are processed.

`\AtEndObject`

`\AtEndObject{⟨code⟩}` runs ⟨*code*⟩ every time at the end of an `object` environment (including `figureobject`, `tableobject` and `\includegraphicobject`). This hook is after a potential `env` has been closed but before the caption and label are processed. (Yes, caption and label are always processed in `\end` even if the caption is displayed above the float. The float package is responsible for putting the caption where it belongs.)

3.4 Initialization

This package uses the float package to restyle `table` to `plaintop` and `figure` to `plain` so that captions of tables appear always above the table and captions of figures always below the figure. The reasoning is explained in [7]. If you really want to place captions differently you can do that with `\restylefloat` (see float package documentation [14]) or by setting the `float style` option. However, I would advice to rethink why you would want to do that.

Unless this package is loaded with the `allowstandardfloats` option it redefines the `table` and `figure` environments to issue a warning if they are used directly. This warning should not influence their usual behavior, though. Instead of `table`/`figure` you should use `tableobject`/`figureobject` or `\includegraphicobject`, otherwise this package cannot help you.

Unless this package is loaded with the `nographic` option it loads the graphicx package in order to include graphics. This also guarantees that the paper size of the generated pdf matches LaTeX' point of view (instead of depending on the system settings). [16]

Unless this package is loaded with the `noarray` option it loads the array package which defines additional column specification features like `>{⟨prefix⟩}`, `<{⟨suffix⟩}` and `!{⟨addcolsep⟩}` and the `\newcolumntype{⟨col⟩}[⟨args⟩]{⟨spec⟩}` command. It also changes the implementation of how lines (rules) are drawn but that is irrelevant if you use the recommendations given in the booktabs package documentation [2, section 2 *The layout of formal tables*]. Loading the array package is merely for convenience. This package does not use any of it's features.

Unless this package is loaded with the `nobooktabs` option it loads the booktabs package which defines commands for formatting tables, most importantly `\toprule`, `\midrule` and `\bottomrule`. These are used by the `table head` key unless you redefine it using `table head style`.

Other packages loaded by this package are listed in appendix B.

3.5 Package options

The package options are implemented using the standard LaTeX package options handling functionality as described in [17]. Therefore they do *not* take any values but consist of keys only. Instead I usually provide two separate keys, one which enables an option and another which disables the option. The keys with a ● are active by default and the keys with a ○ are inactive by default.

● `graphicx` use the graphicx package as backend for `\includegraphicobject`.

○ `graphbox` use the graphbox package as backend for `\includegraphicobject`.

○ `nographic` do not load graphicx or graphbox. If you use this option the commands `\includegraphicobject` and `\includegraphicsubobject` are not defined.

Warning: Without driver specific packages like graphicx, geometry or hyperref the paper size of the resulting pdf depends on the system settings, independent of what you set in LaTeX. [16]

● `array` load the array package. There is no difference between using this package option or a separate `\usepackage{array}`.

○ `noarray` do *not* load the array package.

● `booktabs` load the booktabs package. There is no difference between using this package option or a separate `\usepackage{booktabs}`.

○ `nobooktabs` do *not* load the booktabs package. Please note that the `table head` key initially relies on the booktabs package. If you want to use it with this package option you need to redefine it with `table head style`. Please read sections 1 and 2 of the booktabs documentation [2] before deciding against booktabs.

○ `longtable` load the longtable package. There is no difference between using this package option or a separate `\usepackage{longtable}`.

● `nolongtable` do *not* load the longtable package.

○ `allowstandardfloats` do not redefine the `table` and `figure` environments. Without this option they are redefined to issue a warning if they are used directly. This warning should not influence their usual behavior. Instead of `table`/`figure` you should use `tableobject`/`figureobject` or the command `\includegraphicobject`, otherwise this package cannot help you.

3.6 Help

If you get an error message and don't understand where it comes from or the output is different from what you expected the following features may give you a better understanding of what this package is doing. These features are based on the TEX primitive `\show`.

`\show` — `\show` shows (among other information) the parameter text and the replacement text of a macro on the terminal and in the log file. If you use one of the following features you most likely want to know the replacement text which is the part between `->` and the last `.` on the line. In errorstopmode mode (i.e. without `-interaction=nonstopmode` which most IDEs pass by default) TEX stops after `\show` and waits until you confirm that you have read the information and it may proceed by pressing enter. For more information see *TEX by Topic* [4, section 34.1].

Handler `.show value` — The `.show value` handler can be used to show the value of a storing key (see pgfkeys documentation [6, section 87.4.9 *Handlers for Key Inspection*]). For example:

```
\includegraphicsubobject[sep/.show value]{⟨filename⟩}
```

Handler `.show boolean` — This package also defines a new handler called `.show boolean` which can be used to show the value of a boolean key. For example:

```
\objectset{warn other env/.show boolean}
```

`\ShowObjectStylesInGroup` — `\ShowObjectStylesInGroup{⟨group⟩}` shows all object styles which are contained in the given group. The styles are wrapped in curly braces so that I can iterate over them with the LATEX command `\@tfor`.

`\ShowObjectStyleOptions` — `\ShowObjectStyleOptions{⟨style⟩}` shows the options set for a specific style. Please note that this does *not* show directly set options (i.e. options set by `\objectset` without the optional argument and options in the options argument of the object).

Key `show env args` — See also the `show env args` key of the `object` and `subobject` environments.

4 Installation

This package is contained in TEX Live (latex-extra) and MiKTEX. Therefore it is probably already installed on your system if you have a full LATEX installation. In case it is not you can install it on Arch-Linux with `pacman -S texlive-latexextra` or on Debian with `apt install texlive-latex-extra`.

If your TEX distribution is old and you want to use the newest version of this package download the sty-file from `https://gitlab.com/erzo/latex-easyfloats/-/blob/master/easyfloats.sty?ref_type=heads` and put it next to your main document. (Or, if you care about readability of the source code, download the dtx-file instead and change it's extension to sty. The dtx-file contains indentations and more comments and is therefore easier to read for humans but slower to parse for TEX.)

5 Bug reports and contributions

If you find a bug please open an issue for it on `https://gitlab.com/erzo/latex-easyfloats/-/issues` including a minimal example where the bug occurs, an explanation of what you expected to happen and the version of LATEX and the packages you are using (which are included in the log file). Issues which are not reproducible will be closed.

If you have a feature request please open an issue for it on `https://gitlab.com/erzo/latex-easyfloats/-/issues` including a minimal example which you would like to work, an explanation of what it should do and a use case explaining why this would be useful.

Before opening an issue please check that there is not yet an issue for it already.

If you want to resolve an issue yourself please create a merge request. Make the changes in *easyfloats.dtx*. You can generate the sty file with `tex easyfloats.ins` but you do not need to do that manually because *test/autotest.py* does that automatically for you. Before creating a merge request please make sure that the automated tests still pass. Run the python3 script *test/autotest.py* from the project root or test directory without arguments. While running the tests it shows a progress bar in square brackets. A dot stands for a successful test, an F for a failed test and an E for an error in the test script. Merge requests where a test prints F will most likely be rejected. If you get an E please create a bug report issue.

Please use *tabs* for indentation.

A merge request should include:

- The changes to *easyfloats.dtx*
- The automatically generated *easyfloats.sty*
- Additions to the documentation
- Automated tests in the *test* directory to make sure the new feature or bug fix does not break in the future
- A link in the merge request description to the issue which it is supposed to close

6 Change log

The Gitlab symbols are links to the corresponding commit/tag on Gitlab.

6.1 v1.1.0

New features:

- `placement = I`
- `\splitobject`
- `subpage outer pos = auto-inverted|Auto-inverted`
- `auto label prefix`
- `\AtEndObject`

Bugfixes:

- close all groups which `\includegraphicobject` opens
- don't indent subobjects on 2nd/3rd/xth line

Documentation additions:

- new example: custom options, section 1.10
- new example: varioref, section 1.12
- new example: subfigures, section 1.3
- new example: how to create new float type, section 1.8
- new example: how to circumvent subcaptionbox limitations, section 3.2.4
- added more background information to the first example
- how to achieve a different graphics width for subfigures
- revised subobjects example
- added note which LaTeX version has been used for testing
- added more info to examples
- mention TikZ library perspective
- `\crefname`
- `\Crefname`
- show texdoc command to open the corresponding document in appendix D

Documentation updates:

- updated section regarding the installation
- `\floatplacement{figure}{H}` works nowadays

Documentation fixes:

- fixed wrong example, section 1.7

- added initial value of `table break text` to documentation
- table break text is an executed key, not a storing key
- fixed typos
- cleveref noabbrev: reworded to make it a neutral statement
- show correct tilde symbols
- updated comment on lmodern
- set label in first two examples instead of suppressing the warning
- added prefixes to labels in examples

6.2 v1.0.0

The first release.

7 License

This package and it's documentation are distributed under the *LaTeX Project Public License*, version 1.3 or later. See *license.txt*. The preamble of the documentation may alternatively, at your choice, be reused under the terms of the *WTFPL*, version 2 or later.

Additionally to the rights granted by the LaTeX Project Public License you have permission to freely distribute *unmodified* copies of the files *easyfloats.sty* and *doc/easyfloats.pdf* without other files of this work. The other files of this work can be found at: `https://gitlab.com/erzo/latex-easyfloats`

The examples and tests are distributed under the *WTFPL*, version 2 or later. See *test/license.txt*.

A Motivation

In this section I will explain how to insert figures and tables in standard LaTeX without this package and how this package can improve that. If you are only interested in how to use this package not why, see section 1 for examples and section 3 for an explanation of the commands, environments and options defined by this package.

A.1 Graphics

Inserting a graphic without using this package requires 6 lines of code (graphicx or graphbox must be loaded for `\includegraphics`):

```
\begin{figure}
    \centering
    \includegraphics[graphic width=.8\linewidth]{ctan_lion}
    \caption{CTAN lion drawing by Duane Bibby}
    \label{ctan_lion}
\end{figure}
```

Lines 1 and 6 open/close a floating environment. The content of this environment can float around so that it won't cause a bad page break. You don't need this if you really just want to insert a graphic exactly here (like a logo in a header) but a graphic cannot break across pages so if it is too large for the end of the current page it will move to the next page leaving the end of this page empty. This is a waste of paper and may confuse a reader by suggesting this might be the end of a chapter. A floating environment can help you by putting the figure where it fits best.

The placement determines where a float is allowed to be placed. Initially that's the top or bottom of a text page or a separate page just for floats. The placement can be specified for a single floating object by passing an optional argument to the floating environment or for all floating objects using the `\floatplacement` command defined by the float package. (The floating environments `figure` and `table` are standard LaTeX and do not require the float package.) The allowed values for the placement are described in the description of the `object` environment's `placement` key.

There are people who are concerned that a figure not sitting at the exact position might confuse a reader. However, a graphic naturally attracts the reader's attention. Therefore it does not matter where it is located on the double page. The reader will see it.

Of course the author must ensure that the figure does not float too far away. If that is the case changing the size of this or another graphic, `\usepackage[section]{placeins}`, `\FloatBarrier` (defined by the placeins package), moving this block of lines in the code, changing the placement or tweaking the parameters which govern the placing of floats [8, page 28] can help.

Line 2 centers the graphic horizontally on the line.

The `\centering` command is used instead of the `center` environment because the latter would insert additional vertical space.

```
\begin{center}
    ...
\end{center}
```

is in LaTeX2[1] (somewhat simplified[2]) equivalent to

```
\begingroup
\center
    ...
\endcenter
\endgroup
```

This means that if you accidentally try to use `\centering` as an environment instead of a command you will *not* get an error. You might expect to get an error at least for `\endcentering` not being defined but the TeX primitive

---

[1] This will change in LaTeX3 [10].

[2] `\begin` checks that it's argument is defined, `\end` checks that it's argument matches that of `\begin` and deals with `\ignorespacesafterend` and `\@endparenv`. Since 2019/10/01 `\begin` and `\end` are robust. Since 2020/10/01 they include hooks. [18, section *ltmiscen.dtx*]

`\csname` which is used to produce the `\endcentering` token instead defines it to `\relax`, a no operation.

The output, however, will not be as desired: the group is closed before the end of the paragraph and `\centering` is forgotten before it can take effect.

Line 3 inserts the graphic. This requires the graphicx or graphbox package.

If you want all graphics to have the same width you can set the `width` globally with `\setkeys{Gin}{width=⟨dimen⟩}`. However, that does not work with all options. Unfortunately the graphicx package documentation [9, section 4.6] is not getting more specific than that this works with “Most of the keyval keys”.

Line 4 inserts the caption.

Captions for a figure should be placed *below* the figure. Captions for a table should be placed *above* the table. [7]

`\caption` can be used inside of a floating environment only. If you need a caption for a non-floating object you can either use `\captionof{⟨type⟩}{⟨caption⟩}` defined by the capt-of or caption package or use a floating environment with the placement `H` defined by the float package.

Line 5 defines a label. This is not visible in the output but can be referenced using `\ref{⟨label⟩}` or `\pageref{⟨label⟩}`. You might want to consider using the cleveref package for references.

The label must be set inside of or after the caption. A label always refers to the last `\refstepcounter` inside of the current group. [18, section *ltxref.dtx*] `\refstepcounter` is used for example by `\caption` and `\section`. Therefore if you use `\label` after the caption it refers to the caption. If you use it before the caption it refers to the current section/subsection/subsubsection.

There are many things that a beginner can do wrong without even getting a warning. Three out of this six lines are always the same (lines 1, 2 and 6). I don’t want to always write them out. There is no way to easily switch floating on or off globally.

* * *

This package reduces these six lines to a single command and loads graphicx automatically (unless this package is loaded with the `nographic` option).

```
\includegraphicobject[%
    caption = CTAN lion drawing by Duane Bibby,
    graphic width = .8\linewidth,
]{ctan_lion}
```

The floating environment is applied automatically. It can be changed using the `type` key but I discourage doing so manually. Instead I recommend to use the separate optional ⟨*style*⟩ argument if necessary. If you do not want the object to float you can pass `placement=H`. This works also globally with `\objectset`.

`\centering` is applied automatically. It can be changed using the `align` key.

You can set any of the options passed to the `\includegraphics` command globally using:

```
\objectset[figure]{graphic width=.8\linewidth}
```

Caption and label can be passed as options. Which one is specified first makes no difference. I recommend to stick with caption first in case you ever need to work without this package and to not confuse other people who are not familiar with this package. If you omit one of them the file name is used. See `auto label`, `auto caption`, `auto label strip path` and `auto caption strip path`.

Whether the caption is put above or below the object is specified by the `float style`.

A.2 Tables

Inserting a table is similar to inserting a graphic except that you replace the `\includegraphics` command with an environment which creates a table, place the caption above the table not below it and use another floating environment, namely `table` instead of `figure`.

The following example (not using this package) requires the booktabs package for the horizontal rules and the caption package to have an appropriate space below the caption.

```
 1  \begin{table}
 2      \centering
 3      \caption{Some catcodes}
 4      \label{catcodes}
 5      \begin{tabular}{cl}
 6          \toprule
 7              Catcode & Meaning          \\
 8          \midrule
 9              0       & Escape Character \\
10              1       & Begin Group      \\
11              2       & End Group        \\
12              \vdots  & \quad \vdots     \\
13          \bottomrule
14      \end{tabular}
15  \end{table}
```

What I have said about floating environments, `\centering`, `\caption` and `\label` in appendix A.1 is also valid for tables. New are lines 5–14. We now have two environments nested inside of each other. The outer environment (lines 1 and 15) is the floating environment. The inner environment (lines 5–14) is the environment which creates the table. The inner environment takes a column specification telling LaTeX how many columns the table has and how they are supposed to be aligned. In this case that is `cl`: Two columns, the first centered, the second left aligned. For more information about column specifications see the array package documentation [1, section 1].

`\toprule`, `\midrule` and `\bottomrule` (defined by the booktabs package) produce horizontal lines. They differ in the width of the line and/or spacing around them.

In contrast to the standard LaTeX `\hline` command they have proper spacing around them.

`&` separates columns, `\\` separates rows. Indentation and spaces at the beginning and end of a cell are ignored.

* * *

Using this package we don't need two environments and we don't even need to type out the rule commands if we use `table head`. The packages caption, booktabs and array are loaded automatically (unless you load this package with `nobooktabs` or `noarray`).

```
\begin{tableobject}{%
    caption = Some catcodes,
    label = catcodes,
    env = tabular,
    arg = cl,
    table head = Catcode & Meaning,
}
    0       & Escape Character \\
    1       & Begin Group      \\
    2       & End Group        \\
    \vdots  & \quad \vdots     \\
\end{tableobject}
```

Also we gain the possibility to easily switch between different tabular-like environments, see section 1.5 and the example given for the `(⟨env⟩) arg(s) +` key in section 3.2.1.

A.3 Subobjects

There are several packages to combine several figures/tables into a single floating environment. *Das LaTeX 2_ε-Sündenregister* [19] recommends using subcaption over subfig and the long deprecated subfigure.

The subcaption package provides several ways to do this. The first one is using the `\subcaptionbox` command.

```
\begin{table}
    \centering
    \caption{Category and character codes}
    \label{codes}
    \subcaptionbox{Category codes\label{catcodes}}{%
        \begin{tabular}{cl}
            \toprule
                Catcode & Category         \\
            \midrule
                0       & Escape Character \\
                1       & Begin Group      \\
                2       & End Group        \\
                \vdots  & \quad \vdots     \\
            \bottomrule
        \end{tabular}%
    }%
    \qquad
    \subcaptionbox{Character codes\label{charcodes}}{%
        \begin{tabular}{cr<{\hspace{1.3em}}}
            \toprule
                Character      & \multicolumn1c{Charcode} \\
            \midrule
                \textbackslash & \number`\\               \\
                \{             & \number`\{               \\
                \}             & \number`\}               \\
                \vdots         & \vdots \phantom{0}       \\
            \bottomrule
        \end{tabular}%
    }%
\end{table}
```

As the subobjects are inside of an argument they cannot contain code which relies on changing catcodes e.g. `\verb`. Aside from that it just doesn't seem elegant to put an environment inside of an argument.

If you accidentally put the label in the second argument of `\subcaptionbox` instead of in the first it refers to the parent object instead of the subobject and you won't get an error or a warning for that.

Note how I have commented out line breaks in order to avoid undesired spaces.

The second way is to use the `subfigure`/`subtable` environment. Because the subobject is not inside of an argument it is possible to use `\verb`.

```
\begin{table}
    \caption{Category and character codes}
    \label{codes}
    \begin{subtable}{.5\linewidth}
        \centering
        \caption{Category codes}
        \label{catcodes}
        \begin{tabular}{cl}
            \toprule
                Catcode & Category         \\
            \midrule
                0       & Escape Character \\
                1       & Begin Group      \\
                2       & End Group        \\
                \vdots  & \quad \vdots     \\
            \bottomrule
        \end{tabular}%
    \end{subtable}%
    \begin{subtable}{.5\linewidth}
        \centering
        \caption{Character codes}
        \label{charcodes}
        \begin{tabular}{cr<{\hspace{1.3em}}}
            \toprule
                Character & \multicolumn1c{Charcode} \\
            \midrule
                \verb|\|  & \number`\\               \\
                \verb|{|  & \number`\{               \\
                \verb|}|  & \number`\}               \\
                \vdots    & \vdots \phantom{0}       \\
            \bottomrule
        \end{tabular}%
    \end{subtable}%
\end{table}
```

But why having different environments for subfigures and subtables? The floating environment specifies the type already.

These environments are based on a minipage and require you to always explicitly specify the width of this minipage. On the one hand I don't want to always type that out. On the other hand I want to be able to change the width once for all subobjects for easier consistency.

Caption and label must be placed correctly, see appendix A.1. Even if you restyle the floating environment to always put the caption at the top or bottom using the float package this does *not* apply to subobjects.

It is important to comment out line breaks because the widths of the two minipages add up to the line width, a space between them would cause an overfull hbox or a line break.

We need two `\centering`s, one for each subobject. Remember what I said about `\centering` and `center` in appendix A.1.

* * *

This package defines an environment called `subobject` which is a unified wrapper around `\subcaptionbox` and `subfigure`/`subtable`. Which of these two backends should be used can be specified with the `subcaptionbox` and `subpage` options. `subpage` is used by default so that you can usually use `\verb` in the content.

`subobject` can be used inside of any `*object` environment. If you define a new object environment with `\NewObjectStyle` it defines a corresponding subpage environment like `subfigure`/`subtable` if it does not exist already and if the caption package is new enough. If the caption package is older than August 30, 2020 you need to define the subtype manually by putting the following line *before* loading this package [15]:

```
\AtBeginDocument{\DeclareCaptionSubType{⟨type⟩}}
```

You don't need to write out the width, `.5\linewidth` is used automatically. You can change this value for all subobjects using

```
\objectset{subobject linewidth=⟨dimen⟩}
```

Caption and label are given as options like for `tableobject`. Their order does not matter. They are placed above or below the subobject based on the internal command `\caption@iftop` defined by the `caption` package.

Spaces after `\begin{subobject}` and before and after `\end{subobject}` are ignored so you don't need to comment out the line breaks.[3] Just make sure you don't have an empty line between the subobject environments. That would *not* be ignored.

`\centering` is inserted automatically. It can be changed with `subpage align`.

```
\begin{tableobject}{caption=Category and character codes,
↪  label=codes, env=tabular, sub}
    \begin{subobject}{caption=Category codes, label=catcodes}{cl}
        \toprule
            Catcode & Category         \\
        \midrule
            0       & Escape Character \\
            1       & Begin Group      \\
            2       & End Group        \\
            \vdots  & \quad \vdots     \\
        \bottomrule
    \end{subobject}
    \begin{subobject}{caption=Character codes, label=charcodes}
    ↪  {cr<{\hspace{1.3em}}}
        \toprule
            Character & \multicolumn1c{Charcode} \\
        \midrule
            \verb|\|  & \number`\\               \\
            \verb|{|  & \number`\{               \\
            \verb|}|  & \number`\}               \\
            \vdots    & \vdots \phantom{0}       \\
        \bottomrule
    \end{subobject}
\end{tableobject}
```

A separator for the subobjects could be defined globally using `sep`, see also `hor` and `ver`.

For including a graphic from an external file this package defines a wrapper command around `subobject` and `\includegraphics` in order to reduce the typing effort:

```
\begin{figureobject}{caption=Two lions, label=lions, sub}
    \includegraphicsubobject[caption=A lion]{lion-1}
    \includegraphicsubobject[caption=Another lion]{lion-2}
\end{figureobject}
```

---

[3] Actually, spaces after `\begin{subobject}` and before `\end{subobject}` are ignored only if `env` is empty. But if `env` is not empty I am expecting it to be a tabular-like environment where spaces are ignored at the beginning and end of a cell or a tikzpicture where spaces are ignored as well. Spaces after `\end{subobject}` are ignored regardless of `env`.

B Used packages

This section lists the packages which are loaded by this package in order to provide it's functionality. This list may be interesting for further customization or a deeper understanding how this package works. This list does *not* list all packages which are loaded by default, other packages which may be useful when dealing with floats are listed in appendix C.

- float for `placement=H` and `float style`. It also gives you the possibility to define new float types.
- caption In the standard document classes there is no distance at all between a table and it's caption above. The caption package fixes this. It also defines the `\phantomcaption` command which I am using in case that no caption is given. (The documentation of `\phantomcaption` is in the subcaption package.) It also gives you the possibility to customize the layout of captions but I am not changing the default layout. And it is a dependency of the subcaption package.
- subcaption for subobjects
- graphicx/graphbox for inserting graphics with `\includegraphicobject` (see package options `graphicx`, `graphbox` and `nographic`)
- pgfkeys for parsing key=value lists
- etoolbox, a collection of small helpers for programming
- environ to define environments which save their content in a macro. I am using this for the `subcaptionbox` backend of the `subobject` environment.

C Other packages

This section lists other packages which may be useful in combination with this package.

- placeins When loaded with the `section` package option it prevents floats from floating to another section. It provides the `\FloatBarrier` command which prevents floats from floating past a certain point.
- flafter ensures that floats are not placed before their inclusion in the source code. (With the placement=t it is possible that they are placed on the top of the same page.)
- booktabs for formatting tables. This is loaded by default unless you use the package option `nobooktabs`. Please read sections 1 and 2 of the booktabs documentation [2].
- xcolor When loaded with the package option `table` it provides commands for coloring tables.

  `\rowcolor{⟨color⟩}` sets a background color for a single row. See section 1.6.

  `\rowcolors{⟨firstrow⟩}{⟨oddcolor⟩}{⟨evencolor⟩}` can be used with the `exec` key and sets alternating row colors for the entire table.

- array extends the column specification syntax and defines the `\newcolumntype` command to define custom column types. It also changes the approach how rules are drawn but that is irrelevant if you apply booktabs' guidelines [2, section 2 *The layout of formal tables*]. This is loaded by default unless you use the package option `noarray`.
- siunitx for typesetting numbers and units. It provides the `S` column to align numbers at their decimal separator.
- tabularx A table where the columns adapt to the width of the table, not the other way around. Unlike `tabular*` the space goes into the columns, not between the columns.
- longtable provides tables where a pagebreak is allowed, see section 1.5. This can be loaded automatically with the package option `longtable`.
- hyperref automatically creates links in the pdf document for example from references to floating objects. By default links are set in a red box, you can disable the red box with the package option `hidelinks`. With the package option `pdfusetitle` it automatically sets the pdf title and author based on `\title` and `\author`. hyperref should usually be loaded last.
- varioref automatically inserts a page reference (e.g. "on the following page" or "on page 42") after the reference if the reference is on a different page than the label when using `\vref{⟨label⟩}`/`\Vref{⟨label⟩}`, see the example in section 1.12.
- cleveref automatically inserts the type of reference (e.g. "figure" or "section") in front of the reference when using `\cref{⟨label⟩}`/`\Cref{⟨label⟩}` (defined by cleveref) or `\vref{⟨label⟩}`/`\Vref{⟨label⟩}` (defined by varioref). With the `nameinlink` package option the type in front of the number becomes part of the link created by hyperref. By default `\cref`/`\vref` (to be used inside of a sentence) use abbreviations but `\Cref`/`\Vref` (to be used at the beginning of a sentence) do not because abbreviating at the beginning of a sentence is considered bad style [20]. Abbreviations can be disabled with the `noabbrev` option. cleveref must be loaded last, even after hyperref. See the example in section 1.12.
- biblatex If you input graphics you need to specify the source. Biblatex creates an entire bibliography for you.
- tikz is an amazingly powerful package to create your own graphics in LaTeX.
- newfloat provides a more modern command to define new floating environments than the float package.
- minted provides syntax highlighting for source code.

For more information about floats see `https://latexref.xyz/Floats.html` (it seems this is an html version of the above quoted pdf [8]).

D References

Works cited in this documentation (ordered by appearance in this document, if there are several versions on CTAN I am referring to the English pdf):

[1] Frank Mittelbach and David Carlisle. *A new implementation of LaTeX's tabular and array environment*. Oct. 1, 2020. `texdoc array`. URL: `https://ctan.org/pkg/array`.

[2] Simon Fear. *Publication quality tables in LaTeX*. Apr. 29, 2016. `texdoc booktabs`. URL: `https://ctan.org/pkg/booktabs`.

[3] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1986. ISBN: 0-201-13448-9.

[4] Victor Eijkhout. *TeX by Topic*. 2007. `texdoc texbytopic`. URL: `https://ctan.org/pkg/texbytopic`.

[5] Axel Sommerfeldt. *Customizing captions of floating environments*. Sept. 12, 2020. `texdoc caption`. URL: `https://ctan.org/pkg/caption`.

[6] Till Tantau. *TikZ & PGF Manual*. May 9, 2019. `texdoc tikz`. URL: `https://ctan.org/pkg/pgf`.

[7] *Why should a table caption be placed above the table?* URL: `https://tex.stackexchange.com/questions/3243/why-should-a-table-caption-be-placed-above-the-table` (visited on 07/17/2020).

[8] *LaTeX 2ε: An unofficial reference manual*. Oct. 2018. `texdoc latex2e`. URL: `https://ctan.org/pkg/latex2e-help-texinfo`.

[9] D. P. Carlisle. *Packages in the 'graphics' bundle*. June 1, 2017. `texdoc graphicx`. URL: `https://ctan.org/pkg/graphicx`.

[10] Joseph Wright. *LaTeX3 and document environments*. URL: `https://www.texdev.net/2011/01/09/latex3-and-document-environments/` (visited on 07/17/2020).

[11] Manuel. Sept. 16, 2014. URL: `https://tex.stackexchange.com/questions/201348/why-doesnt-makeatletter-work-inside-newcommand`.

[12] LaTeX Project Team. *LaTeX for authors*. May 23, 2023. `texdoc usrguide`. URL: `https://ctan.org/pkg/usrguide`.

[13] Axel Sommerfeldt. *The subcaption package*. Aug. 24, 2020. `texdoc subcaption`. URL: `https://ctan.org/pkg/subcaption`.

[14] Anselm Lingnau. *An Improved Environment for Floats*. Nov. 8, 2001. `texdoc float`. URL: `https://ctan.org/pkg/float`.

[15] user2574. July 19, 2012. URL: `https://tex.stackexchange.com/a/63967`.

[16] David Carlisle. Dec. 5, 2017. URL: `https://tex.stackexchange.com/questions/404673/paperwidth-too-large/404693#comment1008643_404673`.

[17] The LaTeX3 Project. *LaTeX 2ε for class and package writers*. Feb. 15, 2006. `texdoc clsguide`. URL: `https://ctan.org/pkg/clsguide`.

[18] Johannes Braams et al. *The LaTeX 2ε Sources*. Oct. 1, 2020. `texdoc source2e`. URL: `https://ctan.org/pkg/source2e`.

[19] Mark Trettin and Marc Ensenbach. *Das LaTeX 2ε-Sündenregister*. German. Feb. 3, 2016. `texdoc l2tabu`. URL: `https://ctan.org/pkg/l2tabu`. An English translation of an older version is available at `https://ctan.org/pkg/l2tabu-english`.

[20] Toby Cubitt. Aug. 10, 2016. URL: `https://tex.stackexchange.com/questions/256849/cleveref-change-behaviour-of-cref-to-use-the-abbreviated-form#comment791998_256849`.