proofread — Commands for inserting annotations

Wybo Dekker*

v1.00 2022/03/17

Abstract

The `proofread` package defines a few LaTeX commands that are useful when you proofread a latex document. These allow you to easily highlight text and add comments in the margin. Vim escape sequences are provided for inserting or removing these LaTeX commands in the source. The package is based on code for a text highlighting command that was published by Antal Spector-Zabusky in StackExchange.

Options are provided for displaying the document with extra line spacing, and for display of it in either uncorrected or corrected state, both without margin notes.

Contents

1 Usage

Note: this package is based on the `soul` package, so if you plan to highlight non-ASCII characters, you must compile your source with either `xetex`- or `luatex`-based compilers.

The commands described below display a highlighted phrase in your compiled document and place a comment in the margin, prefixed with a counter, which is indicated with `n` in the following. This counter is useful in the communication with the author of the document.

Previous versions of this package used the standard LaTeX `\marginpar` to create notes in the margin. However, although these will not overwrite each other, they can not be used in tables, minipages, footnotes and more. Therefore, this version creates margin notes using the `marginnote` package, in order to make them work in tables, minipages, footnotes and more. This implies that margin notes are placed at the vertical position where they are called, instead of being automatically stacked. So if several notes are generated on the same
`skp` line, they overwrite each other. You can prevent this by prefixing the second note on the line with `\skp` or `\skp[1]`, the third note with `\skp[2]` and so on. Multiline notes may need larger numbers.

The following supposes that you have installed vim-facilities as described in section 2 ("Installing the Vim commands").

1: delete `\del` `esc d` `\del{phrase}` displays phrase and places `n: delete` in the margin. In the `vim` editor, you can generate this code by selecting the phrase and typing `<escape>d`. After typing this escape sequence, you will be in normal mode, behind the closing brace.

2: `\yel` `esc y` `\yel[comment]{phrase}` displays phrase and places `n: comment` in the margin. In the `vim` editor, you can generate this code by selecting the phrase and typing `<escape>y`. After

*Email: wybo@dekkerdocumenten.nl

typing this escape sequence, you will be in insert mode between the square bracket pair, ready to insert the *comment*.

3: add

`\add` `\add{phrase}` displays phrase and places `n: add` in the margin. In the `vim` editor, you can generate this code *after* the current cursor position by typing `<escape>a`, or *before* the cursor position with `<escape>i`. After typing these escape sequences, you will be in insert mode between the braces pair, ready to type what should be added.

`esc a`

`esc i`

4: was: phrase

`\rep` `\rep{phrase}{replacement}` displays replacement and places `n: was: phrase` in the margin. In the `vim` editor, you can generate this code by selecting the phrase and typing `<escape>r`. After typing this escape sequence, you will be in insert mode between the second pair of braces, ready to insert the new content.

`esc r`

`\com` `\com{comment}` is used by `\del`, `\yel`, `\add`, and `\rep` to place `n: comment` in the margin. You can use it to place comment in the margin without text highlighting. In the `vim` editor, you can insert the command by typing `<escape>c`. After typing this escape sequence, you will be in insert mode between the braces pair, ready to type your comment.

`esc c`

`\hilite` `\hilite[options]{phrase}` is the command on which the above commands are based. It was published by Antal Spector-Zabusky in StackExchange. It highlights the phrase with the default colour, yellow, using the default fill opacity, 0.25; but using the options, you can change this. For example, this phrase was highlighted with blue, with a 3pt width line in yellow around it, with the command:

```
\hilite[fill=blue,draw=yellow,opacity=.5,line width=3pt}
```

Both the fill color and the draw color get 50% opacity, but you can set each individually with the `fill opacity` and `draw opacity` options. See the documentation of the tikz package for more options.

2 Installing the Vim commands

The `proofread` distribution comes with a Vimball archive named `proofread.vmb`. Edit that file in vim and run the command: `:so %`. This will install the necessary files in your `~/.vim` directory, plus the Ruby executable `proofread` in your `~/bin` directory. The latter should be in your `PATH` of course, and you'll have to set its executable flag (`chmod +x ~/bin/proofread`). Finally, in order to use these facilities, you'll have to add this line at the end of your Latex source: `% vim: syntax=proofread`.

The majority of the escape commands that are installed by this procedure have been described above. However, there are three others: The vim sequence `<escape>u` will *undo* the nearest of the three letter commands (`\del, \add, \yel, \rep, \com`) described above, which starts *before* the current cursor position (which may be even on the first character after the starting \). So if it sees `\del{something}` it will replace that with `something`, and if it sees `\add{something}` it will remove that.

`esc u`

`esc h`

`esc n`

The counterpart is the `<escape>h` sequence, which will *honor* the first command found before the cursor. If it sees `\del{something}` it will remove it, and if it sees `\add{something}` it will replace it with `something`. This sequence will move your cursor to the next proofread command.

The following table illustrates what happens with these two escape sequences:

	after `<escape>u`	after `<escape>h`
`\del{something}`	something	
`\add{something}`		something
`\yel{something}`	something	something
`\yel[remark]{something}`	something	something
`\com{something}`		
`\rep{old}{new}`	old	new

As you see, both sequences simply undo `\yel` and `\com` commands, because they are comments, not corrections.

3 Package options

`onehalfspacing`

`doublespacing`

`uncorrected`

`corrected`

`frame`

Options are provided, displaying the document in various spacing and correction modes. The `onehalfspacing` option displays the document with 1.5 times the normal line spacing, The `doublespacing` option displays the document with doubled line spacing. The `uncorrected` displays the document in its uncorrected state, without notes in the margin. The `corrected` displays the document in its corrected state, without notes in the margin. Finally, the `frame`

option is provided for visually handicapped users; it causes .5pt black borders to be drawn around the colored backgrounds around marked texts.

4 Implementation

⟨*package⟩

Option handling: For the `onehalfspacing` and `doublespacing` options we need the setspace package, but the memoir class has its own version for this. So if `\DoubleSpacing` is defined, we redefined the other commands needed. The `PR@spaced` remembers if margin notes need a shift up.

```
\RequirePackage{marginnote}
\ifx\undefined\DoubleSpacing
  \RequirePackage{setspace}
\else
  \let\setstretch\setSpacing
  \let\onehalfspacing\OnehalfSpacing
  \let\doublespacing\DoubleSpacing
\fi
\newif\ifPR@spaced\PR@spacedfalse
\newdimen\PR@unit\PR@unit.6\baselineskip
\DeclareOption{onehalfspacing}{\onehalfspacing\PR@spacedtrue\PR@unit.525\baselineskip}
\DeclareOption{doublespacing}{\doublespacing\PR@spacedtrue\PR@unit.35\baselineskip}
```

The default is to show corrections; the `corrected` and `uncorrected` options will show the document without those in either corrected of uncorrected state.

```
\newif\ifPR@corrected\PR@correctedfalse
\newif\ifPR@uncorrected\PR@uncorrectedfalse
\DeclareOption{corrected}{\PR@correctedtrue}
\DeclareOption{uncorrected}{\PR@uncorrectedtrue}
```

The frame option draws a black 0.5pt frame around colored areas for the visually disabled.

```
\newif\ifPR@frame\PR@framefalse
\newdimen\PR@lw\PR@lw=0pt
\DeclareOption{frame}{\PR@lw=.5pt\PR@frametrue}
\ProcessOptions
```

The following code for a text highlighting command (here renamed to `\hilite` was published by Antal Spector-Zabusky in StackExchange.

```
\RequirePackage{soul}
\RequirePackage{tikz}
\usetikzlibrary{calc}
\usetikzlibrary{decorations.pathmorphing}

\newcommand{\PR@defhiliter}[3][]{%
  \tikzset{every hiliter/.style={fill=#2,fill opacity=#3, #1}}%
}

\PR@defhiliter{yellow}{.25}

\newcommand{\PR@hilite@Dohilite}{
  \fill [ decoration = {random steps, amplitude=1pt, segment length=15pt}
        , outer sep = -15pt, inner sep = 0pt, decorate
        , every hiliter, this hiliter ]
        ($(begin hilite)+(0,8pt)$) rectangle ($(end hilite)+(0,-3pt)$) ;
}

\newcommand{\PR@hilite@Beginhilite}{
  \coordinate (begin hilite) at (0,0) ;
}

\newcommand{\PR@hilite@Endhilite}{
  \coordinate (end hilite) at (0,0) ;
}

\newdimen\PR@hilite@previous
\newdimen\PR@hilite@current
```

\hilite

```
\DeclareRobustCommand*\hilite[1][]{%
  \ifPR@frame%
    \tikzset{this hiliter/.style={#1,draw=black,line width=\PR@lw}}%
  \else%
    \tikzset{this hiliter/.style={#1}}%
  \fi
  \SOUL@setup
  %
  \def\SOUL@preamble{%
    \begin{tikzpicture}[overlay, remember picture]
      \PR@hilite@Beginhilite
      \PR@hilite@Endhilite
    \end{tikzpicture}%
  }%
  %
  \def\SOUL@postamble{%
    \begin{tikzpicture}[overlay, remember picture]
      \PR@hilite@Endhilite
      \PR@hilite@Dohilite
    \end{tikzpicture}%
  }%
  %
  \def\SOUL@everyhyphen{%
    \discretionary{%
      \SOUL@setkern\SOUL@hyphkern
      \SOUL@sethyphenchar
      \tikz[overlay, remember picture] \PR@hilite@Endhilite ;%
    }{%
    }{%
      \SOUL@setkern\SOUL@charkern
    }%
  }%
  %
  \def\SOUL@everyexhyphen##1{%
    \SOUL@setkern\SOUL@hyphkern
    \hbox{##1}%
    \discretionary{%
      \tikz[overlay, remember picture] \PR@hilite@Endhilite ;%
    }{%
    }{%
      \SOUL@setkern\SOUL@charkern
    }%
  }%
  %
  \def\SOUL@everysyllable{%
    \begin{tikzpicture}[overlay, remember picture]
      \path let \p0 = (begin hilite), \p1 = (0,0) in \pgfextra
        \global\PR@hilite@previous=\y0
        \global\PR@hilite@current =\y1
      \endpgfextra (0,0) ;
      \ifdim\PR@hilite@current < \PR@hilite@previous
        \PR@hilite@Dohilite
        \PR@hilite@Beginhilite
      \fi
    \end{tikzpicture}%
    \the\SOUL@syllable
    \tikz[overlay, remember picture] \PR@hilite@Endhilite ;%
  }%
  \SOUL@
}
```

Reduce minimum vertical space between margin paragraphs; if the memoir class is active, use the outer margin:

```
\AtBeginDocument{\marginparpush2pt}
\ifx\undefined\marginparmargin\else\marginparmargin{outer}\fi
```

We need a save box and a counter for prefixing the margin paragraphs:

```
\newcount\PR@markerno\PR@markerno=1
```

\com Make a raggedright margin note, in footnote fontsize, prefixed with the counter plus a colon:

```
\newcommand{\com}[1]{%
  \marginnote{%
    \setstretch{1}%
    \raggedright%
    \footnotesize%
    \the\PR@markerno: #1%
  }[\PR@skip]%
  \global\PR@skip0pt
  \global\advance\PR@markerno1%
}
```

\skp

```
\newdimen\PR@skip\PR@skip0\PR@unit
\newcommand{\skp}[1][1]{%
 \PR@skip#1\PR@unit%
}
```

\del

```
\newcommand{\del}[1]{%
  \com{delete}%
  \hilite[red]{#1}%
}
```

\yel

```
\newcommand{\yel}[2][]{%
  \com{#1}%
  \hilite{#2}%
}
```

\add

```
\newcommand{\add}[1]{%
  \com{add}%
  \hilite[green]{#1}%
}
```

\rep

```
\newcommand{\rep}[2]{%
  \com{was: #1}%
  \hilite[blue]{#2}%
}
```

If one of the `corrected` or `uncorrected` options is used, redefine the above commands accordingly:

```
\ifPR@uncorrected
  \ifPR@corrected
    \PackageError{proofread}
      {corrected and uncorrected options are mutually exclusive}
      {You may not use both the corrected and uncorrected options}
  \fi
  \def\com#1{} \def\add#1{} \def\del#1{#1} \def\rep#1#2{#1}
  \renewcommand{\yel}[2][]{#2}\renewcommand{\hilite}[2][]{#2}
\else
  \ifPR@corrected
    \def\com#1{} \def\add#1{#1} \def\del#1{} \def\rep#1#2{#2}
    \renewcommand{\yel}[2][]{#2}\renewcommand{\hilite}[2][]{#2}
  \fi
\fi
\endinput
```

⟨/package⟩

5 History

v1.00 2015/09/06
First public release
v1.01 2015/12/07
running counter was sometimes not advanced
v1.02 2017/02/28
- Now works in tables, footnotes, minipages and more
- package options frame, corrected, uncorrected added
- package options doublespacing, onehalfspacing added
- skp command for manual moving down marginnotes
- vim commands are provided for removing notes, either honoring or undoing the correction
v1.03 - 2018/07/31
- added warning to use xetex or luatex for non-ASCII source
v1.04 2019/04/25
- removed overstriking in the `\del` command, as it forces the text on a single line

6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.