

# Package: patchSynctex (via r-universe)

September 1, 2024

**Type** Package

**Title** Communication Between Editor and Viewer for Literate Programs

**Version** 0.1-4

**Date** 2016-12-12

**Depends** tools, stringr

**Enhances** knitr, utils

**Description** This utility eases the debugging of literate documents ('noweb' files) by patching the synchronization information (the '.synctex(.gz)' file) produced by 'pdflatex' with concordance information produced by 'Sweave' or 'knitr' and 'Sweave' or 'knitr' ; this allows for bilateral communication between a text editor (visualizing the 'noweb' source) and a viewer (visualizing the resultant 'PDF'), thus bypassing the intermediate 'TeX' file.

**License** GPL(>=2)

**URL** <https://github.com/EmmanuelCharpentier/patchSynctex>

**Repository** <https://emmanuelcharpentier.r-universe.dev>

**RemoteUrl** <https://github.com/emmanuelcharpentier/patchsynctex>

**RemoteRef** HEAD

**RemoteSha** 6d8e7af1dece544922282e2396aa0f2c89cc3422

## Contents

patchSynctex-package . . . . .	2
patchSynctex . . . . .	4

<b>Index</b>	<b>7</b>
--------------	----------

---

patchSynctex-package *Allows for communication between .Rnw editor and .pdf viewer.*

---

## Description

Utility patching the `.synctec(.gz)` file resulting from then LaTeXing of a `.tex` file with the concordance information from the `.concordance.tex` file resulting from knitting (possibly Sweaving) of the `.Rnw` source, allowing(for)back)ward searching from the editor to the viewer and back.

## Details

Synchronising TeX source and DVI/PDF output considerably eases the editing (and, sometimes, debugging) of such documents, by allowing interaction between source and end-result. Modern DVI/PDF viewers and editors support this synchronization, either directly, by using information embedded in the `.dvi` or `.pdf` file, or by using an auxilliary file generated by a LaTeX run with option `\synctex=1`, teh `.synctex(.gz)` file.

However, when one uses a tool such as Sweave or `knitr` to generate a dynamic document (compounding analysis, computations and report), the resulting `.tex` file is no longer the original source, and the references to the `.tex` file are of little help in the revision of such a document.

Duncan Murdoch's **patchDVI** package aims at solving this problem by using a *concordance* file that can be generated by passing a `concordance=TRUE` option so Sweave or `knit*`. This `-concordance.tex` file contains an RLE encoding of the correspondence between the `.Rnw` source and the `.tex` intermediary file.

**patchDVI** uses this concordance information to patch the DVI pointers to source with the relevant pointers to the `.Rnw` file.

This package also has a `patchSynctex` function, attempting to find the relevant information in the `.pdf` file. However, some (most ?) PDFs happens to be impenetrable to **patchDVI**'s `patchSynctex` function.

The present package aims at patching the `.synctex(.gz)` file with pointers to the original `.Rnw` source(s), using exclusively the `.synctex(.gz)` file as a source of information. Therefore, the tools (editors and viewers) must support Synctex in such a way that, when a `.synctex(.gz)` file is present, this information is preferred to the information present in the DVI or PDF document.

This turns out, according to some serious googling and limited testing, to be the case with :

- under Linux: emacs, gedit (editors), evince, okular (viewers), RStudio (IDE);
- under MS-Windows: emacs (editor), Sumatra (viewer), RStudio (IDE).

Furthermore, the `.synctex(.gz)` file file remains necessary for any synchronization, whereas a patched `.dvi` file can in principle be used without the `.synctex(.gz)` file.

This package is mostly aimed at **programs** (integrated development environments (IDEs), such as RStudio, or programmable editors, such as emacs and gedit) able to get R to execute code: it gives them an easy-to-use interface to a simple function does the necessary patches for a given document (characterized by its name sans extension).

Its unique function can still be used from an interactive R session for debugging purposes.

## Integration in editors and IDEs

*This section will be enriched by **your** contributions.*

**ESS under emacs:** The logical point of insertion of `.synctex(.gz)` patching is after each PDF compilation. It may be a bit wasteful (when two or more compilations are necessary, in order, for example, to fix references), but there is no standard way to determine if a LaTeX run is final.

This can be achieved by placing an advice on the function used to compile to PDF/DVI. The example shows how to advice PDF compilation.

*Up to emacs 24.3:* The following elisp snippet can be placed in your emacs initialization file (e. g. `~/.emacs` in Unix(-like) systems):

```
(defadvice ess-svw-PDF (after ess-run-patchKnitr (&optional
      PDFLATEX-CMD)
      activate)
  "Patches the .synctex.gz file after PDF compilation"
  (interactive)
  (let* ((fn (buffer-file-name))
        (fe (file-name-extension fn)))
    (if (string= fe "Rnw")
        (progn
          (setq RPScmd (concat "patchSynctex('"
                              (file-name-sans-extension fn)
                              "'"))
                (ess-create-temp-buffer "tampon-ess-execute")
                (ess-execute "require(patchSynctex)" 'buffer "tampon-ess-execute")
                (ess-execute RPScmd 'buffer "tampon-ess-execute")
                (kill-buffer "tampon-ess-execute")))))
    (ad-activate 'ess-svw-PDF)
```

*From emacs 24.4 onwards:* The previous elisp snippet has to be adapted to the newer advice system of elisp:

```
(defun ess-svw-patch-after-PDF (&optional PDFLATEX-CMD)
  "Patch the synctex file after PDF compilation"
  (let* ((fn (buffer-file-name))
        (fe (file-name-extension fn)))
    (if (string= fe "Rnw")
        (progn
          (setq RPScmd (concat "patchSynctex('"
                              (file-name-sans-extension fn)
                              "'"))
                (ess-create-temp-buffer "tampon-ess-execute")
                (ess-execute "require(patchSynctex)" 'buffer "tampon-ess-execute")
                (ess-execute RPScmd 'buffer "tampon-ess-execute")
                (kill-buffer "tampon-ess-execute")))))
    (advice-add 'ess-svw-PDF :after #'ess-svw-patch-after-PDF)
```

**AUCTeX under emacs:**

This is a work in progress : AUCTeX is more sophisticated than ESS about PDF production. There is no single function to advice (and knitting is not yet well integrated, by the way...). In the meantime, ESS functions remain available. *Stay tuned...*

**Eclipse + StatET:** Create an external build configuration (Sweave document processing) and place :

```
require(knitr);
opts_knit$set(concordance = TRUE);
texfile <- knit("${resource_loc:${source_file_path}}", encoding="UTF-8")
```

in the sweave tab, and :

```
syntex <- if (opts_knit$get('concordance'))"-synctex=1" else "-synctex=0";
command=paste("latexmk -pdf", syntex, "-interaction=nonstopmode", shQuote(texfile));
print(paste("Command ",command,"...\n"));
print(shell(command),intern = TRUE);
if (opts_knit$get('concordance')){
  require(patchSynctex);
  patchSynctex(texfile);
}
print(paste0(substr(texfile,1, nchar(texfile)-3), "pdf"))
```

in the LaTeX tab.

## Note

The current (1.8) version of knitr does not yet implement concordance for multifile projects (i. e. children chunks).

## Author(s)

Jan Gleixner, Emmanuel Charpentier

Maintainer: Emmanuel Charpentier [emm.charpentier@free.fr](mailto:emm.charpentier@free.fr)

## References

Duncan Murdoch's excellent [patchDVI](#) has a great vignette explaining the basics of the problem.

---

patchSynctex

*Create correspondence between .pdf and .Rnw files*

---

## Description

patchSynctex(foo) uses the concordance file foo-concordance.tex generated by knitting (possibly Sweaveing) foo.Rnw with the option concordance=TRUE to patch foo.synctex(.gz) with information pointing to foo.Rnw.

**Usage**

```
patchSynctex(nwfile, syncfile=NULL, verbose=FALSE, ...)
```

**Arguments**

nwfile	name of the file to patch (used sans extension).
syncfile	output sync file (if nwfile is related to an included file).
verbose	if TRUE, emit a message stating the number of patches. Useful for debugging integration in your tools.
...	Unused. Allows any argument forced by your tools to be passed without causing an error.

**Details**

This function reads the information given in the `nwfile-concordance.tex` file (which *must* exist) to patch the `nwfile.synctex(.gz)` file, which originally contains pointers from `nwfile.pdf` to the source in `nwfile.tex` with information pointing to the latter's source in `nwfile.Rnw`.

Editors and viewers supporting Synctex will be able to use this information to allow forward- and backward search between PDF and its original source, thus easing debugging.

The `nwfile` will be used sans extension ; this allows your favorite IDE to pass either the name of the noweb file or the name of the `.tex` file.

The `syncfile` argument allows to force the addition of the patched references to the main `syncfile` for a document that uses subfiles. This is a workaround that may or may not be well-supported by `knitr` and/or your viewer.

The function may raise errors (files not found), warnings (no patch found to be done) or messages (number of patched locations).

This function is principally intended for use by programmable IDEs able to execute R code. It is documented mostly for debugging purposes.

**Value**

Nothing useful.

**Note**

The current (1.8) version of `knitr` does not yet implement concordance for multifile projects (i. e. children chunks).

**Author(s)**

Jan Gleixner, Emmanuel Charpentier <emm.charpentier@free.fr>

**References**

Duncan Murdoch's excellent patchDVI package: <https://cran.r-project.org/package=patchDVI>.

## Examples

```

if(requireNamespace("tools", quietly=TRUE) &&
  requireNamespace("knitr", quietly=TRUE)) {
  ## Minimal demonstrative knitr example.
  RnwSrc<-"
\\synctex=1
\\documentclass{article}
<<Setup, eval=TRUE, echo=FALSE, results='hide'>>=
opts_knit$set(concordance=TRUE, self.contained=TRUE)##$
require(patchSynctex)
@
\\author{A.~U.~Thor}
\\title{A minimal \\textsf{knitr} example}
\\date{Some time}
\\begin{document}
\\maketitle

```

A first paragraph of text, which offers a target for forward search`\\,` for example, in `\\textsf{emacs}` with `\\textsf{AUCTeX}`, typing ```C-c~C-v''` should bring you to your PDF viewer in the corresponding typeset line.

```

<<TestFig, echo=FALSE>>=
curve(sin(x), from=-pi, to=pi, main='A curve generated by R',
      sub='back-searching from here should bring you close to the \\TestFig\\' chunk.')
@

```

This second paragraph of text is also a convenient target for back-searching. For example, in `\\textsf{evince}`, a ```Ctrl>-click''` should bring you to the `\\textsf{noweb}` source, bypassing the `\\LaTeX` intermediate file.

```

\\end{document}
"
  require(knitr)
  cat(RnwSrc, file="Minimal.Rnw")
  knit2pdf("Minimal.Rnw", quiet=TRUE)
  D1<-file.info("Minimal.synctex.gz")$mtime
  patchSynctex("Minimal", verbose=TRUE)
  ## should print a message telling the number of patches
  D2<-file.info("Minimal.synctex.gz")$mtime
  D1!=D2
  ## should return TRUE
  ## To see the effect, try (example on a Linux system)
  ## Not run: system("evince Minimal.pdf")
}

```

# Index

\* **~documentation**

patchSynctex, [4](#)

\* **~interface**

patchSynctex, [4](#)

patchSynctex, [4](#)

patchSynctex-package, [2](#)