Package 'patchDVI'

March 14, 2023

Type Package

Title Package to Patch '.dvi' or '.synctex' Files

Version 1.11.0

Author Duncan Murdoch

Maintainer Duncan Murdoch <murdoch.duncan@gmail.com>

Description Functions to patch specials in '.dvi' files,
or entries in '.synctex' files. Works with concordance=TRUE
in Sweave, knitr or R Markdown to link sources to previews.

License GPL-2

Depends R (>= 3.3.0)

Imports utils, tools, RmdConcord

Suggests knitr (>= 1.42.0), rmarkdown

SystemRequirements The 'Japanese.Rnw' vignette requires uplatex and
dvipdfmx.

VignetteBuilder knitr, rmarkdown, patchDVI

Encoding UTF-8

URL https://github.com/dmurdoch/patchDVI

BugReports https://github.com/dmurdoch/patchDVI/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-03-14 16:40:02 UTC

R topics documented:

`defSconcordance` *Support for non-Sweave concordances.*

Description

Sweave automatically inserts concordances into the output file, so they will be embedded in the '`.dvi`' or '`.pdf`' output using the LaTeX macro `\Sconcordance` from the '`Sweave.sty`' style file. Other vignette processing engines (e.g. knitr) do not use '`Sweave.sty`', so `defSconcordance` defines the macro explicitly.

The `useknitr` function is specific to knitr documents. If `writeMacro` is `TRUE`, it calls `defSconcordance` to write the macro definition to the output file. If `writeMacro` is not specified, it guesses the value by looking for `\begin{document}` somewhere in the first 100 lines of the file. It always writes the `\input` line for the concordances into the file.

Usage

```
defSconcordance()
useknitr(writeMacro)
```

Arguments

`writeMacro`	logical; whether to call defSconcordance

Details

This function is not necessary when using Sweave, it is only for other vignette engines.

With knitr, the concordances would be written to a file with a name typically be of the form '`basename-concordance.tex`', where the input file was '`basename.Rnw`'. You would insert it with code like

```
<<results="asis">>=
patchDVI::useknitr()
@
```

If the document does use the '`Sweave.sty`' style, or if it is being included in another larger document, do not use `defSconcorance`: this can be forced by calling `patchDVI::useknitr(writeMacro = FALSE)`.

Value

These functions are called for the side effect of writing out the LaTeX code.

Author(s)

Duncan Murdoch

Examples

```
defSconcordance()
```

`DVIspecials` *Get and set DVI specials*

Description

These functions get and set the \specials values in .dvi files.

Usage

```
DVIspecials(f)
setDVIspecials(f, newspecials, newname=f)
```

Arguments

`f`	character filename of .dvi file to patch
`newspecials`	character vector of new specials to put into the file
`newname`	the name of the file to write

Details

If an entry of the `newspecials` vector is `NA`, that special is removed.

Value

`DVIspecials` returns all the specials from the file in a character vector.

`setDVIspecials` is called for the side effect of patching the .dvi file; it returns `NULL` invisibly.

Author(s)

Duncan Murdoch

See Also

`patchDVI`

Examples

```
## Not run:
x <- DVIspecials("test.dvi")
x[1] <- "new value"
setDVIspecials("test.dvi", x)

## End(Not run)
```

`knitInRStudio` *Trigger processing of chapter files from RStudio.*

Description

If this function is executed within RStudio, it will knit all child from the main file of a large '`.Rnw`'-style knitr document.

Usage

```
knitInRStudio(SweaveFiles, force = FALSE, verbose = FALSE)
```

Arguments

`SweaveFiles`	The files to process. If missing, the value is set from a `.SweaveFiles` variable in the global environment.
`force`	If `TRUE`, all files are knitted; otherwise, only those that are newer than the corresponding '`.tex`' file.
`verbose`	Add some debugging information to the output.

Details

This does nothing if not run from within RStudio. If run there, it starts a second R process to do the knitting. It's assumed that this call will exist in a code chunk in another knitr document.

Set `childOutput = TRUE` in the options to the code chunk containing this if you wish to see the output of the child knitr runs in the same log as the main one.

Value

The concatenated output from the child knitting, and possibly some verbose output.

Author(s)

Duncan Murdoch

See Also

`knitAll`, which does a similar thing outside of RStudio.

knitPDF *Knit a File and Patch*

Description

This is a simple script to run knit, then LaTeX, then patchDVI.

Usage

```
knitMiktex(Rnw, main,
    weave = knitr::knit,
    ...)

knitPDFMiktex(Rnw, main,
    weave = knitr::knit,
    ...)

knitDVI(Rnw, main,
    weave = knitr::knit,
    ...)

knitPDF(Rnw, main,
    weave = knitr::knit,
    ...)

knitDVIPDFM(Rnw, main,
    weave = knitr::knit,
    ...)
```

Arguments

Rnw	The .Rnw input file
main	The .tex file on which to run latex, defaulting to the output file from Sweave
weave	The function to carry out the weaving.
...	See Details below.

Details

The ... arguments are passed to the corresponding Sweave* function.

If the Rnw argument is a filename ending in '.tex', then knitting is skipped, but the output file is still patched (because one of the other files in the project might have come from knitr).

Value

These functions are run for the side effects.

Examples

```
## Not run:
knitPDF("chapter.Rnw", "book.tex")

## End(Not run)
```

---

needsPackages *Check for and optionally install packages used by document.*

Description

This function checks for needed packages

Usage

```
needsPackages(pkgs, install = TRUE,
                    update = FALSE,
                    load = FALSE,
                    attach = FALSE)
```

Arguments

pkgs	A list of names of packages to check.
install	Whether to install missing packages.
update	Whether to update all packages that can be updated.
load	Whether to load namespaces for packages.
attach	Whether to run library calls.

Details

Each of the boolean arguments can be a vector of the same length as pkgs, applying separately to each entry.

Value

Called for the side effects.

Examples

```
# This shouldn't do anything, as patchDVI must already be
# installed!
needsPackages("patchDVI")
```

patchDVI *patch Sweave source references into .dvi file*

Description

This package allows source references in .dvi files to refer to the actual source file, rather than the intermediate .tex file. Dvi previewers such as yap and xdvi can make use of these source references to jump directly to the part of a document corresponding to a particular source line, and the previewer can send messages to editors to jump to the source corresponding to a location in the preview.

Usage

```
patchDVI(f, newname = f, patchLog = TRUE)
patchSynctex(f, newname=f, uncompress="pdftk %s output %s uncompress",
     fromDVI = NULL, patchLog = TRUE)
```

Arguments

f	A character value giving the name of the .dvi or .synctex file to patch
newname	Output will go to this file; by default to overwrite the original
patchLog	Whether to attempt to patch the '.log' file
uncompress	External command to uncompress a compressed PDF file
fromDVI	Name of DVI file in which to find concordance

Details

This function works with Sweave with option concordance=TRUE and some related functions.

Using patchDVI requires a change to the usual Sweave workflow, as follows:

1. Write the source file in .Rnw. Include the

```
\SweaveOpts{concordance=TRUE}
```

option to enable the patching.

2. Use Sweave to convert the file to .tex format.

3. Use latex to process the .tex file into a .dvi file or .pdf file.

4. Run patchDVI on the .dvi file or the .synctex file to patch in the Rnw links.

5. Use the previewer on the patched .dvi file or the original .pdf file.

All the information needed for the patching is stored in the .dvi file, so these 5 steps can be run from the command line, there's no need to keep an R session running (though they'll all work from within one session as well). At some distant date in the future maybe the previewers will learn how to read the concordance info, and step 4 will no longer be necessary.

The patchLog option makes use of the new patchLog function.

Value

A string containing a message about what was done.

Note

In a .dvi file, the concordance is stored in a TeX "special" in the following format (as of version 0.5): `concordance:<oldname>:<newname>:<firstline> <rle pairs>*`. Here `<oldname>` is the name of the .tex file to replace, `<newname>` is the name of the .Rnw file to substitute for it, and `<firstline>` is the line number from the .Rnw file of the first line in the .tex file.

The `<rle pairs>` are a sequence of pairs (length, value), separated by spaces (no parentheses), with the length indicating the number of repetitions of the value. Using `inverse.rle` these are converted to a sequence of differences in the line numbers of the concordance.

Since colons are used to delimit the elements, they must not be used in `<oldname>` or `<newname>`: names like '`C:/source.Rnw`' will not work.

For example, a concordance might contain

```
\Sconcordance{\concordance:optimization.tex:optimization.Rnw:1 2 1 2 4 39 1 1 4 ...}
```

saying that source references to optimization.tex should be replaced with optimization.Rnw. Line 1 of the .tex file corresponds to line 1 of the .Rnw file. The diffs for successive lines are `1 1 4 4 1 <repeated 39 times> 4 ...`, so the corresponding lines in the .Rnw file (including the first) are `1 2 3 7 11 12 13 ...`.

The same format is used in a stream object in a .pdf file. (Version 2.12.2 and earlier of R used a different format which was actually invalid according to the PDF specification.) This is not always successfully read by the patching code, because some versions of `pdflatex` compress parts of the file, and `patchDVI` currently can't read compressed parts of files. We suggest installing the `pdftk` open source PDF toolkit (available from https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit/) to decompress the file; `patchSynctex` will make use of it if found. The `uncompress` argument can be modified to call a different uncompress utility. The Latex commands `\pdfcompresslevel=0\pdfobjcompresslevel=0` at the start of your document instruct `pdflatex` not to compress the file, but some Latex packages (e.g. beamer) will override these settings.

If your workflow involves using `latex` and `dvipdfm`, then you will need a recent enough version of `latex` that supports the `-synctex=1` option, and then you can specify the name of the '`.dvi`' file in the `fromDVI` argument.

This format is subject to change in future R releases.

Author(s)

Duncan Murdoch

References

The yap previewer is part of MikTeX, from www.miktex.org.

See Also

DVIspecials for the function to read .dvi files, SweaveMiktex for a simple script to run Sweave, MikTeX and patchDVI.

Examples

```
## Not run:
patchDVI("mypaper.dvi")

## End(Not run)
```

---

patchLog *Patch a LaTeX log file.*

Description

This function parses a standard LaTeX log file and applies patches to it based on an Sweave concordance, so that error messages and warnings refer to the '.Rnw' file rather than the '.tex' file.

Usage

```
patchLog(f, newname = f, concords = NULL, max_print_line = 79, Cstyle = FALSE)
```

Arguments

f	The filename of the log file.
newname	The filename to write the patched file to.
concords	Concordances read from the '.dvi' or '.pdf'. If NULL, an attempt is made to read these from the source.
max_print_line	The line length at which the latex compiler will wrap error messages.
Cstyle	If TRUE, only look for C-style error messages "filename:num: msg". If FALSE, look for classic messages "! msg". If NA, look for both, pick the more popular style.

Details

The parsing is based on the log parser written by Jonathan Kew, Stefan Löffler, Antonio Macrì, and Henrik Skov Midtiby for TeXWorks. That parser assumes error lines are reported in the standard format; C style errors are not currently supported. patchLog can detect C style errors, but standard format is assumed by default, and is preferred.

Value

This function is called mainly for the side effect of writing the new log file. It returns the concords object invisibly.

Author(s)

Duncan Murdoch

See Also

patchDVI, patchSynctex

---

pdf_documentC *R Markdown driver to add concordance*

Description

This drivers replace the like-named **rmarkdown** driver with one that outputs Commonmark rather than Pandoc Markdown. Commonmark is a dialect of Markdown for which the Pandoc driver for Commonmark supports output of source position information. By using this function as your output driver, you can get that in your own documents.

Usage

```
pdf_documentC(latex_engine = "pdflatex",
           sourcepos = TRUE,
           defineSconcordance = TRUE,
           run_latex = TRUE, ...)
```

Arguments

latex_engine	Command to convert '.tex' file to '.pdf'.
sourcepos	Whether to include source position information.
defineSconcordance	If TRUE, insert a definition of the \Sconcordance macro just before \begin{document}.
run_latex	Whether to run the latex_engine to produce a PDF.
...	Other arguments to pass to the base driver.

Details

This driver modifies the standard **rmarkdown** driver pdf_document, but uses Commonmark and adds concordances.

Value

An R Markdown output format object which will add concordance information.

Author(s)

Duncan Murdoch

pdf_with_patches *Convert an R Markdown pdf driver to one that handles concordances.*

Description

This function produces a new driver which matches the old one in most respects, but adds an argument `sourcepos` (and possibly others). If that argument is `TRUE` (the default) then concordances are handled by the new driver.

This is used to produce `pdf_documentC` in this package, and should also work on other drivers that produce PDF output using Pandoc.

Usage

```
pdf_with_patches(driver)
```

Arguments

`driver`	An R Markdown driver that produces HTML, LaTeX or PDF using Pandoc.

Details

This function makes use of the `RmdConcord::pdf_with_concordance` function. That function adds concordances to the output of the original driver; this function patches the PDF file so that source references in it refer to the original source file.

Value

A new driver function.

Examples

```
pdf_with_patches(rmarkdown::latex_fragment)
```

readDVI *Demo function to read a .dvi file*

Description

This is an R function that reads a .dvi file (it was the prototype for `DVIspecials`). Modify it to see other parts of a .dvi.

Usage

```
readDVI(f, show=c("bop", "special", "fntdef", "preamble"))
```

Arguments

f	filename
show	which type of records to print

Value

readDVI prints as it reads, it doesn't return anything useful.

---

renderPDF *Render R Markdown and Patch*

Description

These are simple scripts to render, then run LaTeX, then patch.

Usage

```
renderMiktex(Rmd, main,
    weave = rmarkdown::render,
    ...)

renderPDFMiktex(Rmd, main,
    weave = rmarkdown::render,
    ...)

renderDVI(Rmd, main,
    weave = rmarkdown::render,
    ...)

renderPDF(Rmd, main,
    weave = rmarkdown::render,
    ...)

renderDVIPDFM(Rmd, main,
    weave = rmarkdown::render,
    ...)
```

Arguments

Rmd	The .Rmd input file
main	The .tex file on which to run latex, defaulting to the output file from Sweave
weave	The function to carry out the weaving.
...	See Details below.

Details

The ... arguments are passed to the corresponding Sweave* function.

If the Rmd argument is a filename ending in '.tex', then knitting is skipped, but the output file is still patched (because one of the other files in the project might have come from R Markdown).

Value

These functions are run for the side effects.

Examples

```
## Not run:
renderPDF("chapter.Rnw", "book.tex")

## End(Not run)
```

SweaveAll *Apply Sweave to a collection of files.*

Description

This function allows a master Sweave input file to specify other files in the project. All modified files will be run through Sweave. Following the run, several post-processing steps may be performed.

Usage

```
SweaveAll(SweaveFiles, make = 1, PostSweaveHook = NULL,
          force = TRUE, verbose = FALSE,
          weave = utils::Sweave, ...)
knitAll(RnwFiles, make = 1, PostKnitHook = NULL,
          force = TRUE, verbose = FALSE,
          weave = knitr::knit, ...)
renderAll(RmdFiles = NULL, make = 1, PostKnitHook = NULL,
          force = TRUE, verbose = FALSE,
          weave = rmarkdown::render, ...)
```

Arguments

SweaveFiles, RnwFiles, RmdFiles	character vector naming the files to run through weave.
make	integer indicating which files to run weave on. See below.
PostSweaveHook, PostKnitHook	default hook to be called after each weave call. See below.
force	If TRUE, all files in SweaveFiles will be run, otherwise only those which are newer than the corresponding '.tex' file.
verbose	Mainly for debugging: print progress messages.

weave	The function to be used for converting from source to '.tex'. It will be called as weave(filename, ...) for each file in SweaveFiles and other files identified as needing weaving.
...	Extra parameters to pass to weave.

Details

This function runs weave repeatedly. It starts by checking for a global variable .SweaveFiles, and augments the SweaveFiles argument with any unprocessed files in that list. It then runs weave on each file listed in SweaveFiles.

Before each run, it removes the global variables .PostSweaveHook, .SweaveFiles, and .TexRoot. The individual Sweave files may set these variables to control post- processing as described below.

Following each run, SweaveAll checks whether the code chunks created a variable .PostSweaveHook in the global environment. If present, it should be a function (or the name of a function; match.fun is used to do the lookup) taking a single argument. If not present, the PostSweaveHook (or PostKnitHook) argument will be used as a default. SweaveAll will pass the name of the '.tex' output file to this function, which can do any required postprocessing. For example, we have used it to remove local pathnames from output strings.

SweaveAll then checks again for .SweaveFiles, which if present should be a character vector of filenames. These are collected, and after the Sweave runs specified by SweaveFiles, the make parameter controls which other files are run. The default value of 1 works as follows: any new files added via .SweaveFiles that were not already processed are checked. If they are newer than the corresponding '.tex' file, or the '.tex' file does not exist, they are processed. They may also return new filenames in the global .SweaveFiles variable and those will be processed as well.

Setting make = 2 forces all mentioned files to be processed. With make = 0, only the original vector of files passed to SweaveAll will be processed.

SweaveAll also checks for a global variable named .TexRoot. If present, it will be placed in the first position in the result vector. (If it was already in the vector, later copies will be removed.)

Value

The names of all files produced. The first element of Sweavefiles will produce the first element of the result if .TexRoot was not specified; other elements will depend on which files were processed.

Author(s)

Duncan Murdoch

Examples

```
## Not run:
# Run Sweave on a main file, and obtain other files from there.
SweaveAll("main.Rnw")

# Run knitr on the same file.
SweaveAll("main.Rnw", weave = knitr::knit)

## End(Not run)
```

SweaveMiktex *Run Sweave and Patch*

Description

This is a simple script to run Sweave, then convert the LaTeX file, then patch it.

Usage

```
SweaveMiktex(Rnw, main,
    cmd = "texify",
   options = "--tex-option=-src-specials --tex-option=-interaction=nonstopmode",
    includedir = "--tex-option=--include-directory=",
    stylepath = FALSE,
    source.code = NULL,
    make = 1,
    preview = 'yap "%s"',
    patchLog=TRUE,
    sleep = 0, ...)

SweavePDFMiktex(Rnw, main,
    cmd = "texify --pdf",
    options = "--tex-option=-synctex=-1 --tex-option=-interaction=nonstopmode",
    includedir = "--tex-option=-include-directory=",
    stylepath = FALSE,
    source.code = NULL,
    make = 1,
    preview = NULL,
    patchLog = TRUE,
    sleep = 0, ...)

SweaveDVI(Rnw, main,
    texinputs = NULL,
    source.code = NULL,
    make = 1,
    links = NULL,
    preview = NULL,
    patchLog = TRUE, ...)

SweavePDF(Rnw, main,
    texinputs = NULL,
    source.code = NULL,
    make = 1,
    links = NULL,
    preview = NULL,
    patchLog = TRUE, ...)
```

```
SweaveDVIPDFM(Rnw, main,
    latex = "latex", latexOpts = "-synctex=1 -interaction=nonstopmode",
    dvipdfm = "dvipdfm", dvipdfmOpts = "",
    texinputs = NULL,
    source.code = NULL,
    make = 1,
    preview = NULL,
    patchLog = TRUE, ...)
```

Arguments

`Rnw`	The .Rnw input file
`main`	The .tex file on which to run latex, defaulting to the output file from Sweave
`cmd`	The LaTeX command to run
`options`	Options to pass to LaTeX
`includedir`	The prefix for an option to give the location of Sweave.sty
`stylepath`	`stylepath` argument to pass to `Sweave`
`source.code`	If not `NULL`, a filename to source before running `Sweave`
`make`	Value to pass to `SweaveAll` to control which Sweave files are processed
`preview`	Command to use to preview result, or `NULL` for no preview
`patchLog`	Whether to attempt to patch the console log and '`.log`' file
`sleep`	Number of seconds to sleep if an error is detected in the run
`texinputs`	Extra paths to pass to `texi2dvi`
`links`	If not `NULL`, an option to turn on production of source links via Synctex or source specials
`latex,latexOpts`	The function to call to convert the '`.tex`' to '`.dvi`', and options to put on the command line.
`dvipdfm, dvipdfmOpts`	The function to call to convert the '`.dvi`' to '`.pdf`', and options to put on the command line.
`...`	Additional options to pass to `SweaveAll`

Details

If the `Rnw` argument is a filename ending in '`.tex`', then weaving is skipped, but the '`.dvi`' file is still patched (because one of the other files in the project might have come from Sweave).

The `source.code` argument may be used to temporarily install code before running the document through the weaver (e.g. `Sweave`). It could install an experimental version of `Sweave`, or set up some variables to be used by other code chunks, etc. (Note that things are sourced locally, whereas `Sweave` runs things in the global environment, so the file will need to make explicit assignments there, and it should avoid using names that clash with the argument names to `SweaveMiktex` or `SweavePDFMiktex`.)

The functions invoke Latex differently. SweaveMiktex is designed to call Miktex's texify command to produce a '.dvi' file. SweavePDFMiktex does the same, but produces a '.pdf'. SweaveDVI and SweavePDF are similar, but use a modified version of R's texi2dvi function to invoke Latex. SweaveDVIPDFM does the conversion to '.pdf' in two stages.

The patchLog option makes use of the new patchLog function.

Value

These functions are run for the side effects.

Examples

```
## Not run:
SweaveMiktex("chapter.Rnw", "book.tex")

## End(Not run)
```

Index