Instructions for rnw2pdf reports with knitted chunks

Paul E. Johnson, Director, CRMDA
Meghan Sullivan, CRMDA

May. 4, 2018

This shows how we use R [?] to make report documents using the CRMDA style. This is a LaTeX "noweb" report generated with the knitr engine.

Contents

1 Introduction

Create a skeleton document by opening R and running

```
library(stationery)
initWriteup("rnw2pdf-report-knit")
```

That will create a folder "writeup/rnw2pdf-report-knit" (unless you request otherwise by setting the dir argument).

We suggest you

1. Compile the skeleton document *as is* to test your setup
2. Make revisions incrementally, and re-compile often to make sure nothing has been broken.

Don't make changes that you don't understand in the code chunks above or the last chunks below.

1425 Jayhawk Blvd.
Watson Library, Suite 470
Lawrence, KS 66045-7594

Web: https://crmda.ku.edu
Email: crmda@ku.edu
Phone: 785-864-3353

2 LyX: Cautions

The document preamble has manual settings for margins (geometry) as well as hyperlinks (PDF hyperref). Don't use the LyX pull down menu to revise them. Edit preamble or config files for that. Repeat **CAUTION**: Don't change the page margins or settings for hyperlinks with pull down menus.

3 What to edit

Title and author information

The first block in the document has the title and author information.

Footer information

The footer in this document uses data that is provided in a file named "addressFooter.tex". After the document is compiled for the first time, that document should be available in the theme folder.

About the theme folder

The theme folder should be empty when the initProject() function is run.

There is an R code chunk above called "texcopy". It will copy configuration files from the package into the theme folder. After running this for the first time, those files will not be automatically replaced by the scripts.

That means authors are free to edit them to fit their needs.

If the author makes an error in editing a theme file, it is safe to delete the erroneous file and run the compile script again. That will copy a fresh version of the theme file into the directory.

4 Check our documentation

There are several vignettes distributed with this package. Please review them.

1. "crmda": the package framework overview
2. "code_chunks": discusses display of code in LaTeX documents
3. "instructions-rnw2pdf-report-knit"

5 Compile as usual, or with rnw2pdf

The skeleton file is provided in 2 formats, LyX and Rnw.

In either case, please understand that compiling is a two step process.

1. knitting: Run R to do the calculations in the R code chunks and write out a LaTeX file
2. compiling: Run a LaTeX program, such as pdflatex or xelatex to convert the LaTeX file to pdf. It is usually necessary to run the compiler two or more times, along with a separate bibliography program. If it is available, we suggest an aggregator named texi2pdf, which will handle this effort.

Edit the LyX file.

There are 4 methods, we hope one will suit your workflow.

1. Use the LyX editor. The file can be compiled to PDF in LyX, just like any other LyX file. LyX handles conversion from LyX to Rnw to tex to PDF. This has the same effect as using LyX from the command line. The following will create the PDF file using pdflatex as the final comiler:

   ```
   $ lyx -e pdf2 skeleton.lyx
   ```

   Because lyx uses a separate working directory for the compilation work, the project directory stays clean. None of the intermediate LaTeX files (*.log, *.log, *.bbl) will appear.

2. Open an R session and make sure the working directory is the same as the project writeup.

   ```
   rnw2pdf("skeleton.lyx")
   ```

3. The shell script `rnw2pdf.sh` is provided in the same folder. It can be run in the shell as

   ```
   $ ./rnw2pdf.sh skeleton.lyx
   ```

   In the discussion in the next sub section, we outline usage of additional arguments with rnw2pdf for the compilation of Rnw files. All of those arguments are equally applicable in this context.

4. In case you want to track the steps of compiling one by one, open the file in LyX. Use the pull down menu `File` → `Export` → `Rnw (knitr)`. That will create a file named "skeleton.Rnw". This is the equivalent of the command line statement

   ```
   $ lyx -e knitr skeleton.lyx
   ```

   After that Rnw file is created, proceed as described in the next subsection.

   This two-step process is valuable for debugging. It makes it easier to spot trouble by focusing on the separate transitions.

Edit the Rnw file

The Rnw file we provide is produced by LyX, it is an intermediate step in the document production sequence. A two step compilation procedure is necessary. First, one must convert the "Rnw" file to "pdf" (with knit), and then the knitted tex file is compiled into pdf by pdflatex (or one of the other LaTeX compilers).

The work flow here will vary, depending on your experience and the editor you choose to use. Here are some possibilities:

1. You may have a "noweb" aware editor. Emacs, Rstudio, and others have menus that can initiate the work of knitting and rendering the document.
2. Open an R session and make sure the working directory is the same as the project writeup.

```
rnw2pdf("skeleton.Rnw")
```

   Additional arguments can be used, mainly to control the verbosity of the output and the creation of subsidiary files. Our function, by default, will create and R file summary of the command chunks. This file is referred to as a "purled" or "tangled" file.

3. The shell script `rnw2pdf.sh` is provided in the same folder. It can be run in the shell as

```
$ ./rnw2pdf.sh skeleton.Rnw
```

   The command script answers to all of the arguments followed by the R function `rnw2pdf`. The usage is nearly identical. Where the R function call would be

```
rnw2pdf("skeleton.Rnw", purl = FALSE, clean = FALSE, verbose =
   TRUE, keep_tex = TRUE)
```

   the shell command would be

```
$ ./rnw2pdf --purl=FALSE --clean=FALSE --keep_tex=TRUE --
   verbose=TRUE skeleton.Rnw
```

   The only difference in usage arises when a quoted string must be passed through. Suppose the files are in a subdirectory named "project". Inside the R code, the quoted string to specify the directory where the file resides (the working directory) would be like so:

```
rnw2pdf("skeleton.lyx", wd = "project")
```

   the shell command would be

```
$ ./rnw2pdf --wd='"project"' skeleton.Rnw
```

   Note the single quotes that are protecting the double quotes.

4. Our shell script is not the only way to use command line tools to get this done. One can run shell commands such as:

```
$ R CMD knit skeleton.Rnw
```

   That will create `skeleton.tex`, which we compile with

```
$ texi2pdf skeleton.tex
```

   The major difference between running this and the script we provide is that our script will handle LyX files and it will, by default, will create a purled copy of the R code.

Table 1: A Regression Table

	First Model	
	Estimate	(S.E.)
(Intercept)	0.022	(0.104)
Excellent Predictor	0.095	(0.091)
N	100	
RMSE	1.011	
R^2	0.011	

$*p \leq 0.05** p \leq 0.01***p \leq 0.001$

6 Code Chunk Check

Reports don't include raw code chunks

What is the difference between a guide and a report? Simply put, a report document does not reveal source code and it should not distract the reader with raw output. A report is supposed to be presentable to a non R user.

As a result, it is important to think about whether to embed the R code in the document at all. One might keep the R code in another file, and use it to generate graphs and tables that can be included into this report document.

As we have mentioned in the "code_chunks" vignette, there are some differences in the way code chunks are used in Sweave and knitr. These differences are especially important in controlling whether, and where, the output of code chunks is presented in a document. Of all of the changes brought by the knitr code chunk engine, the proliferation of chunk options will be the most noticeable. The author of the knitr engine, Yihui Xie, maintains the canonical documentation on code chunk options on his website, `https://yihui.name/knitr/options`.

When code is embedded in the report document

To prevent the code chunks from printing their input and raw output into the document, the code chunks will often employ flags like "include=FALSE" and "echo=FALSE". The code will need to write the tables and figures into files, and they can be inserted when it suits the author to do so.

There are many R packages that generate LaTeX output which is presentable, close to the final desired result. In the development process, we will use these nearly-good-enough tables and then, when a final presentation demands find-grained adjustments, we do so manually on the table files.

Some examples of nearly-good-enough tables are now presented. The regression result presented in Table 1 may not be perfect by APA standards, but it is certainly good enough for intermediate versions. This is produced by the `outreg()` function in the rockchalk package. There are, of course, many other packages designed to generate the same sort of LaTeX markup.

In the kutils package, we made a function `semTable()` that presents structural equation models. The result in Table 2 is an example.

In the perfect world, the tables that come from software would be perfectly presentable. This is the idea of the fully reproducible research document. At the current time, that is unrealistic.

Table 2: A Confirmatory Factor Analysis Table

	Model			
	Estimate	SE	z	p
	Factor Loadings			
visual				
x1	0.90	0.08	11.13	.000
x2	0.50	0.08	6.43	.000
x3	0.66	0.07	8.82	.000
textual				
x4	0.99	0.06	17.47	.000
x5	1.10	0.06	17.58	.000
x6	0.92	0.05	17.08	.000
speed				
x7	0.62	0.07	8.90	.000
x8	0.73	0.07	11.09	.000
x9	0.67	0.07	10.30	.000
	Latent Variances			
visual	1.00[+]			
textual	1.00[+]			
speed	1.00[+]			
	Fit Indices			
RMSEA	0.09			

[+]Fixed parameter

Authors need to leave a back door though which they can answer the very demanding requirements of journals and colleagues who insist on tables aligned "just so" with labels that have particular styles. The software will more closely approach that ideal, of course. But, for now, we make due.

In R, literate documents (using either knitr or Sweave) can be created with the document option "split" set as TRUE. If this is done, then an output file is generated for each and every the individual code chunk. In the CRMDA, our custom is to save these in a directory called "tmpout". The split option is not allowed in documents prepared as R vignettes, but it is still possible to get the work done. R has functions, mainly `cat`, that can be used to save output tables.

7 Session Information

Leave the code chunks below. But the visible words and section name should be removed. Session Information is usually not written into a report, but an output file is created by the following pieces.