Writing Sweaved Guides: Using the CRMDA Template

Paul E. Johnson, CRMDA, pauljohn@ku.edu
Meghan Sullivan, CRMDA,
meg.sullivan@ku.edu

Guide No: 00

Keywords: Sweave, LaTeX, guide, listings
See crmda.ku.edu/guides for updates.

Mar. 12, 2018

Abstract

This shows how we use R (R Core Team, 2017) to make guide documents using the CRMDA style. This is a LaTeX "noweb" guide document generated with the Sweave engine.

Contents

1 Introduction

This shows how we use R (R Core Team, 2017) to make guide documents using the CRMDA style.

Create a skeleton document by opening R and running

```
library(stationery)
initWriteup("rnw2pdf-guide-sweave")
```

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

We suggest you

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

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

1. Compile this 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.

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-guide-sweave"

5 Compile as usual with LyX, 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", engine = "Sweave")
   ```

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

   ```
   $ ./rnw2pdf.sh --engine='"Sweave"' 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` → `Sweave`. That will create a file named "skeleton.Rnw". This is the equivalent of the command line statement

   ```
   $ lyx -e sweave 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 Sweave), and then the weaved 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 weaving 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", engine = "Sweave")
```

   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 "tangled" (if using knitr, it is referred to as a purled file).

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

```
$ ./rnw2pdf.sh --engine='"Sweave"' 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", engine="Sweave" wd = "project")
```

   the shell command would be

```
$ ./rnw2pdf --engine='"Sweave"' --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 Sweave 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.

If you are editing these files in LyX, it is sufficient to simply compile as usual. That will handle the chore of converting a sequence of document types to arrive at PDF.

6 Code Chunk Check

Illustrative R code can be included in the document. The author has a good deal of control over how, and at which, the input and output are displayed. Correctly formatted LaTeX code can be written by R functions and it can appear in the document. The vignette "code_chunks" has full details. This is a brief highlight.

6.1 The listings package

The document preamble includes settings for the LaTeX package listings, which is used to display code input and output. Inline references to code can be marked for highlighting (by LaTeX macro "\code") that will mimic the color styling of the code displays.

One advantage of using our Sweave-based LaTeX documents is the listings class can handle very long lines (allows linewrap) and also lets us have fine grained control over the display of code input and output. In guide documents, we have ine numbers turned on. is used. Among its benefits, we get "line wrap" on long lines.

The listings class used here allows within-document style changes. We expect that report documents will not be customized by most authors, but guide documents are less formal. In order to make output fit within the indicated space, it may be necessary to fiddle with the font size, for example. Here are the highlights:

1. The font size and colors of R input chunks are controlled by LaTeX settings "Rsize", "Rbackground" and "Rcolor". Output displays depend on "Routsize", "Routbackground", and "Routcolor".

2. The font can be adjusted by declarations like this

```
\def\Rsize{\huge\ttfamily}
\def\Routsize{\huge}
```

These can be placed at the very beginning of the document to control all following chunks, but they can be placed immediately before any chunk to adjust just that one chunk.

3. Colors can be specified in many ways

```
\def\Rbackground{\color[gray]{0.90}}
\def\Routbackground{\color[gray]{0.40}}
\def\Rcolor{\color[gray]{0.60}
\def\Routcolor{\color[rgb]{0.9, 0.1, 0.1}]}
\def\Rcommentcolor{\color{green}}
```

To demonstrate this customization, compare these two chunks. The first uses the defaults:

```
x <- rnorm(100)
mean(x)
```

```
[1] 0.2451972
```

While the second offers a shockingly beautiful offering (emphasis on shockingly).

```
x <- rnorm(100)
mean(x)
```

```
[1] 0.04523311
```

Note we use a LaTeX group here–the squiggly braces–to confine the beautifying impact of the change to the immediately following output.

```
{
\def\Rbackground{\color[rgb]{0.4, 0.7, 0.6}}
\def\Routbackground{\color[rgb]{0.6, 0.5, 0.8}}
\def\Routcolor{\color{blue}}
\def\Routsize{\huge}
\input{tmpout/t-rnorm1.tex}
}
```

Otherwise, at least in this document type, the change applies to all following chunks.

6.2 Raw R input and output

Consider a regression.

```
dat <- data.frame(x = rnorm(100), y = rpois(100, lambda = 7))
m1 <- glm(y ~ x, data = dat, family = "poisson")
summary(m1)
```

```
Call:
glm(formula = y ~ x, family = "poisson", data = dat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0907  -0.7677  -0.0396   0.6178   2.2594

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)  1.99148    0.03702  53.788   <2e-16 ***
x           -0.03120    0.03977  -0.784    0.433
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 114.94  on 99  degrees of freedom
Residual deviance: 114.33  on 98  degrees of freedom
AIC: 495.04

Number of Fisher Scoring iterations: 4
```

6.3 Controlling display of chunk input and output

The chunk option "include" can be used to regulate whether the input and output appear immediately in the document. When combined with "echo" and "results", we can have a great deal of control.

In this chunk, we create a regression table object, but we hide everything:

```
library(rockchalk)
or <- outreg(list("My Poisson"= m1), varLabels = c("x" = "A Normal
   Predictor"), tight = FALSE)
```

To display the object `or` to the reader, we have two options.

The standard Sweave approach is to include another chunk, and then cause the LaTeX markup for the object `or` to be woven directly into the document (depends on "results=tex").

	My Poisson	
	Estimate	(S.E.)
(Intercept)	1.991***	(0.037)
A Normal Predictor	-0.031	(0.040)
N	100	
Deviance	114.327	
$-2LLR(Model\chi^2)$	0.615	

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

That chunk might be placed in a LaTeX floating table object, of course, as we show next.

A second option, which is more convenient, is to simply use LaTeX to input the saved file. When the code chunk "pois11" is executed, it creates a file named "tmpout/t-pois11.tex". In Table 1 we demonstrate how that can be included in a numbered floating table.

The reader has not yet had a chance to see the code chunk that calculated the regression. The code chunk that ran the regression was marked "include=F, results=hide". To show that code, there are (at least) 2 ways.

1. The chunk is named "pois10". Because we have the R Sweave argument split=T in the above, the code file is written separately and we can retrieve it with an input statement like so:

   ```
   \input{tmpout/t-pois10}
   ```

Because this note is included as an R vignette, and the R packaging policy prohibits the use of split=T in vignettes, this approach cannot be used in this document.

2. An standard approach using Sweave itself, without split, is to create another chunk and then display it inside double "<<>>" brackets. Here we turn off evaluation (set eval=F) to prevent R from re-running the code chunk:

```
library(rockchalk)
or <- outreg(list("My Poisson"= m1), varLabels = c("x" = "A
   Normal Predictor"), tight = FALSE)
```

Those two code displays should be identical in the document.

6.4 Line wrap

This chunk shows what happens if the R input long. The line wrapping power of the listings class prevents code input from running into the margin.

```
dataFrame <- data.frame(x1 = rnorm(100, m = 13, s = 23), x2 =
   rnorm(100, m = 13, s = 23), x3 = rnorm(100, m = 13, s = 23))
```

7 References

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

8 Session Info

```
R version 3.4.3 (2017-11-30)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 17.10
```

Table 1: A Poisson Regression

	My Poisson	
	Estimate	(S.E.)
(Intercept)	1.991***	(0.037)
A Normal Predictor	-0.031	(0.040)
N	100	
Deviance	114.327	
$-2LLR(Model\chi^2)$	0.615	

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

```
Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  base

other attached packages:
[1] rockchalk_1.8.110 stationery_0.76

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.15       compiler_3.4.3     nloptr_1.0.4       plyr_1.8.4
     methods_3.4.3
 [6] tools_3.4.3        digest_0.6.15      lme4_1.1-15        evaluate_0.10.1
     nlme_3.1-131
[11] lattice_0.20-35    mgcv_1.8-23        openxlsx_4.0.17    Matrix_1.2-12
    parallel_3.4.3
[16] SparseM_1.77       pbivnorm_0.6.0     stringr_1.2.0      knitr_1.19
    MatrixModels_0.4-1
[21] stats4_3.4.3       rprojroot_1.3-2    nnet_7.3-12        grid_3.4.3
    foreign_0.8-69
[26] rmarkdown_1.8      lavaan_0.5-23.1097 minqa_1.2.4        car_2.1-6
    magrittr_1.5
[31] backports_1.1.2    htmltools_0.3.6    MASS_7.3-48        kutils_1.34
    splines_3.4.3
[36] pbkrtest_0.4-7     mnormt_1.5-5       xtable_1.8-2       quantreg_5.35
    quadprog_1.5-5
[41] stringi_1.1.6
```

```
## Don't delete this. It puts the interactive session options
## back the way they were. If this is compiled within a session
## it is vital to do this.
options(opts.orig)
```