# Instructions for guides using knitted code chunks

**Johnson Paul E.**, Directior, CRMDA
<pauljohn@ku.edu>
**Second Author**, CRMDA <second@ku.edu>

CENTER FOR RESEARCH METHODS & DATA ANALYSIS
College of Liberal Arts & Sciences

**Guide No**: 48

**Keywords**: stationery, guide, knitr
See https://crmda.ku.edu/guides for updates.

February 24, 2018

**Abstract**

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

## Contents

**1 Introduction** 1

**2 LyX: Cautions** 2

**3 What to edit** 2

**4 Check our documentation** 2

**5 Compile the PDF document** 3

**6 Code Chunk Check** 5
6.1 knitr issues/features . . . . . 5
6.2 Code Chunks . . . . . 5

**7 References** 6

**8 Session Info** 7

# 1 Introduction

Create a skeleton document by opening R and running

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

That will create a folder "writeup/rnw2pdf-guide-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.

# 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.

Authors are free to edit the theme files, to replace logo images and, of course, to insert their addresses.

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

# 5 Compile the PDF document

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 like 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.lyx", 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.lyx
```

   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.lyx
```

   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.

# 6 Code Chunk Check

## 6.1 knitr issues/features

The original R (R Core Team, 2017) approach to combining code, output within the document is called Sweave. The knitr approach is an alternative package. knitr is helpful especially because it can be used with R markdown to produce HTML web pages.

Styling of knitr code chunks is different than Sweave. It appears we lose line-wrap entirely. I can't figure how to make Sweavel "listings" environments take over the knitr presentation.

We have not yet learned the ins-and-outs of correcting the knitr code and R output chunks to match our desired style. That is one reason to prefer the sweave-based templates we offer.

## 6.2 Code Chunks

If you choose to use knitr rather than Sweave as the chunk-processor–and by using this template that is what you did decide–you should study the knitr chunk options. These are discussed in some detail in the vignette on code chunks.

Here is an example of a data frame being created and a glm is estimated:

```
set.seed(234234)
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
-2.2212  -0.7115  -0.1668   0.4829   2.4448

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.93581    0.03821  50.658   <2e-16 ***
x            0.03175    0.04596   0.691     0.49
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 111.39  on 99  degrees of freedom
Residual deviance: 110.91  on 98  degrees of freedom
AIC: 485.89

Number of Fisher Scoring iterations: 4
```

Some code that might be used to create a regression table using outreg from the rockchalk package is the following. As you see, the output "splats" into the document, it is not in a floating figure or table.

```
library(rockchalk)
or <- outreg(list("My Poisson"= m1), varLabels = c("x" = "A Normal Predictor"), tight = FALSE)
cat(or, file="tmpout/t-pois15.tex")
cat(or)
```

| | My Poisson | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 1.936*** | (0.038) |
| A Normal Predictor | 0.032 | (0.046) |
| N | 100 | |
| Deviance | 110.909 | |
| $-2LLR(Model\chi^2)$ | 0.477 | |

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

It is possible, however, to input the same chunk into a table, as we have done in Table 1. That is the typically-preferred format for presentation of tables in guides.

Table 1: A Poisson Regression

| | My Poisson | |
|---|---|---|
| | Estimate | (S.E.) |
| (Intercept) | 1.936*** | (0.038) |
| A Normal Predictor | 0.032 | (0.046) |
| N | 100 | |
| Deviance | 110.909 | |
| $-2LLR(Model\chi^2)$ | 0.477 | |

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

# 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

Please leave this section for guide documents. It is important for replication.

```
Error:  <text>:4:16:  unexpected symbol
3:    print("Warnings:")
4:    warnings()}utreg
^
```