Introduction to RMarkdown->PDF examples

Han Oostdijk (www.hanoostdijk.nl)

January 2016

Contents

Purpose of this document

This document describes some examples of the use of **R Markdown** [RMarkdown] to produce PDF files. The examples differ in the measure that additional LaTeX functionality has been used.

NB. handling of bibliographies in the manner done here did not work anymore in package rmarkdown version 0.9 . This was corrected in 0.9.1 so please update the package to this version or later when you need this functionality.

Software used

This document and the ones described are created by editing the input files (rmd-, tex- and bib-files) in the **RStudio** [RStudio] environment and pressing the **Knit** button.
This button-click executes the following workflow:

- the R package **knitr** [knitr] is used to analyze the rmd-source and to execute the knitr chuncks. It also uses the information in the yaml header. The result is an md-file.
- the md-file is converted by **Pandoc** [Pandoc] to a tex-file
- the tex-file is converted by LaTeX to a pdf-file

The following software is necessary:

- a distribution of the typesetting system LaTeX. Working on Windows I use MiKTeX
- a distribution of R with the **rmarkdown** and **knitr** packages
- the **Pandoc** software. The **RStudio** environment contains a copy of this software
- **RStudio** is not necessary but makes this workflow very easy: recommended!
- and the R packages you want to use: in these examples **ggplot2** and **ggthemes**

The examples

iris_data_set_vm1.rmd

In this basic example no additional LaTeX is used. The main advantage is that is trivial to convert the rmd-file to a html- or a docx-format. The main disadvantage is that internal references to figures and tables are not available.

iris_data_set_vm2.rmd

Here we use the basic LaTeX commands **\label**, **\ref** and **\pageref** to get the internal references that were missing in the first example. And also we show the use of a *child chunk*: to handle the references we made a change to the *setup chunk* and saved it in a separate rmd-file that we will include from now on.

iris_data_set_vm3.rmd

In this example we use additional LaTeX packages to

- set the default font to be sans-serif (via include of **header.tex**)
- load the package *subfig* so that in a chunk two figures can be placed side by side (via include of **header.tex**)
- (re) define some text macros (via include of **header.tex**)
- redefine some of the colors that are used for highlighting the R-code and its background (via include of **header.tex**). With one of these settings the background color was made darker so that it would be just visible when the document is printed.
- load and set some attributes of the package *fancyhdr* that enables the use of headers and footers (via include of **extra1.tex**)
- set additional attributes for *fancyhdr* (via the chunk **setheader** in iris_data_set_bib1.rmd)

We have structured these LaTeX commands in three separate groups:

- in the external permanent file **header.tex** the commands that go in the LaTeX preamble part and don't change for each document
- in the internal file **extra1.tex** the commands that go in the LaTeX preamble part and are specific for this document
- in the internal file **extra2.tex** the commands that go in front of the LaTeX body part and are specific for this document. In our examples no contents: a candidate for inclusion would be the \chead command but because the header is not constant (see next paragraph) we need to use an engine_R chunk and not a engine_cat chunk.

Apart from these LaTeX changes we also show what can be done in the yaml-header:

- define your own parameters by using the *params* keyword:
  - we use the *doc_version* parameter to include a version number in the header of each page

  - we use the *altplot* parameter to change the program flow by changing some text and including or omitting a figure. We do this by setting some text variables dependent on the parameter and using the parameter to decide about executing and echoing chunks.

- specify e.g. the page orientation by using the *geometry* keyword
- specify that a table of contents is to be included
- specify the *knit* command that will be executed. Here we use it to explicitly specify the name of the pdf-file and to ensure that the intermediate md-file is not removed after processing. By specifying *keep_tex: yes* the intermediate tex-file will also be kept. This can be useful for debugging when the output is not as expected.

We also show here how to use an internal bibliography. This is a list of references that we include at the end of the document. You can also use (and reuse) a bibliography that is stored in an external file. Handling of that is shown in the next example.

iris_data_set_vm4.rmd

This example is the nearly the same as the previous example. The difference is that the parameter *altplot* is now set to FALSE and that the bibiography is now in an external file. The latter is convenient when you often reference the same items. The bibliography has to be specified in the yaml-header and is fully handled by Pandoc and not by the LaTeX processor. Therefore the Pandoc way of referring has to be used and not the \cite method.

iris_data_set_vm5.rmd

In the last examples we used a parameter to distinguish the two cases: only one plot or with an additional plot. We used the parameter as a boolean flag for the *eval* and *echo* parameters in the *r1b* chunk. However each parameter in a chunk can be an R expression. We use this in the current example: we include child documents where the file name is an R expression dependent on the parameter. The example shows a table where depending on a parameter *sortorder* the observations with the highest or lowest values of the variable *Sepal.Length* are displayed.

iris_data_set_vm6.rmd

In the previous examples we used only a limited amount of LaTeX in our code. In this example we show how to display two or more tables side by side. This is convenient (saves space and avoids turning over pages) when dealing with many small tables. The LaTeX code needed for this was found on **stackoverflow** [Marcin Kosiński] and packaged by me in an R-function. In this example we place the two possible tables of the previous example side by side.

Session Info for created documents

```
## R version 3.2.0 (2015-04-16)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 8 x64 (build 9200)
##
## locale:
## [1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
```

```
##
## other attached packages:
## [1] knitr_1.12.3
##
## loaded via a namespace (and not attached):
##  [1] magrittr_1.5    formatR_1.2.1   tools_3.2.0     htmltools_0.2.6 yaml_2.1.13
##  [6] stringi_1.0-1   rmarkdown_0.9.2 stringr_1.0.0   digest_0.6.8    evaluate_0.8
```

References

[knitr] knitr: A General-Purpose Package for Dynamic Report Generation in R,Yihui Xie,http://yihui.name/knitr/

[RStudio] RStudio Version 0.99.489 © 2009-2015 RStudio, Inc.

[Pandoc] Pandoc a universal document converter, http://pandoc.org/

[RMarkdown] R Markdown — Dynamic Documents for R http://rmarkdown.rstudio.com/

[Marcin Kosiński] Answer on a stackoverflow question http://stackoverflow.com/questions/23926671/side-by-side-xtables-in-rmarkdown