Knitr, LaTeX, and Reproducibility

Henrik Renlund

April 9, 2014

UCR
UPPSALA CLINICAL
RESEARCH CENTER

Contents

List of Figures

List of Tables

Command	Function
`\,`	(space)
`\quad`	(more space)
`\emph{emphasize}`	*emphasize*
`\textbf{boldface}`	**boldface**
`\pageref{`*label*`}`	displays pagenumber of *label*
`\footnote{}`	a footnote, duh
`\input{`*path*`}`	input the contents of another file

Table 1: Some useful commands in LaTeX.

1 A really short introduction to LaTeX.

1.1 Structure of a document

A LaTeX source file is usually saved as a `.tex` file. Like HTML, but unlike Word, it is *not* WYSIWYG[1]. An example of a minimal document:

```
\documentclass{article}
\begin{document}
This is my minimal document.
\end{document}
```

The part before `\begin{document}` is refered to as the preamble and may include other specifications for the document, in particular one can include additional functionality through packages with `\usepackage{`*package name*`}`.

Within the document environment (i.e. between `\begin` and `\end{document}`, most ordinary text can be written 'as is', but some characters have special meaning and must be written with a '\', e.g. $, &, and %.

Other parts of the document, e.g. figure and tables, are placed in the corresponding *environment*, i.e. the specification of the object will be between a `\begin{`*environment*`}` and `\end{`*environment*`}`.

MiKTeX[2] is a good LaTeX implementation for windows. With that installed you can compile documents from the command line but it is easier to use a dedicated editor, e.g. TexMaker[3].

1.2 Math

LaTeX is a type setting system especially suited for mathematics. Math can be written within running text if put between two '$'-signs. E.g. the code `$\int_0^1 \frac{x}{1-x}\;dx$` yields: $\int_0^1 \frac{x}{1-x}\,dx$. If put within an equation-environment, mathematics becomes more beautiful still. E.g.

[1] What You See Is What You Get.
[2] `http://www.miktex.org/`
[3] `http://www.xm1math.net/texmaker/`

Figure 1: Donald Knuth - the inventor of TEX.

```
\begin{equation}
\sigma^* = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar x)^2} \label{apa}
\end{equation}
```

yields

$$\sigma^* = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar{x})^2} \tag{1}$$

with the possibility of refering to equation 1 by putting `\ref{apa}` in your source code. (The hyperlinks provided in the electronic version of this pdf are not default[4] in LATEX, but are created by Knitr.)

Some other useful commands are given in Table 1.

1.3 Tables

Data can be tabulated with `tabular` and is usually put in an table-environment (which can have a caption and reference label). The tabular command needs to include a specification for each column (`c,l,r` for center, left and right, respectively, or `p` for a fixed width). Then `&` and `\\` is used to indicating new cell and line, respectively. E.g. the code:

```
\begin{table}
\begin{center}
\begin{tabular}{|l|r p{3cm} c|c} \hline
```

[4] They can be created with the `hyperref` LATEX package.

$X =$	7.80	Hey!	6	-7.00
Smurf	9.1	This is a particularly long piece of text	1000	2

Table 2: This is my Table!

```
$X=$ & 7.80 & Hey! & 6 & -7.00 \\ \cline{2-3}
Smurf & 9.1 & This is a particularly long piece of text & 1000 & 2 \\
\end{tabular}
\caption{This is my Table!}\label{myTable}
\end{center}
\end{table}
```

gives the output shown in Table 2.

1.4 Figures

Graphs can be included with `\includegraphics[`*options*`]{`*path*`}`. Most often, you want to put an image/graph within a figure environment. The following code

```
\begin{figure}
\begin{center}
\includegraphics[scale=1]{image_donald_knuth.jpg}
\caption[Donald Knuth.]{Donald Knuth - the inventor of \TeX.}
\label{knuth}
\end{center}
\end{figure}
```

was used to produce Figure 1. The command `\caption[`*short*`][`*long*`]` places the *short* caption in the list of figures and the *long* connected to the figure.

Sometimes it is handy to be able to put multiple images in a single figure with unique captions. This can be achieved with the `subfig` package[5] and an example thereof is Figure 2. See source code for details.

Note that Figures are 'floats' and will generally be placed where LaTeX sees fit. One can 'allow' the float to be placed in different positions, if you really want the figure in the same positions as within the code you can write `\begin{figure}[h]`, where `h` denotes *here* - but it is *not* a guarantee.

1.5 Lists

In this section, Section 1.5 on page 3, we talk about lists. There

- are

[5] Which is loaded in the preamble on the main document

(a) Yihui.

(b) Yohio.

Figure 2: Two different people. On the left, the creator of *Knitr*. On the right, the creator of *To the End*.

- ways

to create both unordered

1. and
2. ordered

lists. See source code for how this section makes use of labels, references, page references and the list commands.

1.6 How to 'attach' a file

With the `attachfile` package you can include another file within the pdf[6]. E.g. the code

```
\attachfile{documentation_attachfile.pdf}
```

yields: 

In other cases you may want to have the link be a part of text, so

```
\textattachfile{this result.}{documentation_attachfile.pdf}
```

yields: this result.

N.B. This links may not show up if the document is printed! (Thus it may be preferable to only attach with the symbol.)

[6]Supported starting with Adobe Acrobat 4.0.

(The default color in the attach package is somewhat unsatisfactory, in the source code I've actually included an additional option to specify the color according to a red-green-blue scale.)

1.7 References and Bibtex

One can enter references for each document as needed or refer to a `bib` file. The file `example_references.bib` contains one reference (and a template, not shown here);

```
@book{Knitr,
 author               = {Xie, Y.},
 journal              = {},
 publisher            = {CRC Press},
 title                = {Dynamic Documents with R and Knitr},
 year                 = {2013},
 }
```

The first line specifies the type of object ('book') and the label ('Knitr'). If we want to use this citation we write `\cite{Knitr}` and get: [1]. This requires that we include the lines

```
\bibliography{example_references}
\bibliographystyle{plain}
```

- usually towards the end of the document - but now, here:

References

[1] Y. Xie. *Dynamic Documents with R and Knitr*. CRC Press, 2013.

2 Knitr

2.1 The structure of a document

The structure of knitr file (i.e. a file that can be `knitr`:ed) depends on the document language, but the idea is the same whether you use html, markdown language, LaTeX, or some other language supported by knitr; write the document 'as usual' and include R code in 'chunks'. How to specify where a chunk begins and ends depends on the language the chunk is embedded in. In LaTeX a chunk is placed between a `<<`*chunk options*`>>=` and `@`. (Such a file should have the `rnw` extension.)

When a document is knitted, the chunks are identified and executed and - depending on the chunk options - the code and/or the output may be included in the document.

Why knitr?

- You have to make an effort to code in such a way as to make it difficult to find the code for a specific part of the document.
- The number of files you have to keep track of decreases. As this document will show; both tables and plots can be created within the document.
  $\Longrightarrow$ It is easier to reproduce your document.

Also:

- Knitr has sophisticated caching abilities, so you do not have to re-run all R code for recompilation of the document.

2.1.1 How to specify chunk options

Chunk options are specfied like arguments to an R function (but line breaks are not allowed). The first argument (unless specified by name to be something else) is the label, which can be given as `"name"`, `name` or `label="name"`. All other arguments should be named, as must the label if not specified first. As with R function arguments, they can be any R expression.

Chunk options can also be specified for the entire document. See chunk in Section 2.2.

2.2 Compilation of the source code

RStudio has a built in compilation but does not use caching as default[7]. Compilation commands can be executed separetly with the `knit2pdf` function (from the prompt or another file), or from a chunk within the document (but this should not evaluate, i.e. have `eval=FALSE` as chunk option, as part of the compilation process unless you want to get stuck in a loop).

[7] It seems that RStudio opens a new R session to knit the document. You could put `require(knitr)` and `opts_chunk$set(cache=TRUE)` in `Rprofile.site` if you want caching as default.

```
# (This is a chunk of R code.)
# local chunk options: eval = FALSE
require(knitr)
# CHUNK OPTIONS: ----------------------
opts_chunk$set(
#    eval=TRUE,  # evaluation of chunks?
   cache=TRUE#, # caching of chunks?
#    echo=TRUE,  # echo code in document?
#    include=TRUE, # include output of code?
#    message=TRUE, # include messages, ...
#    error=TRUE,   # errors, ...
#    warning=TRUE  # warnings?
)
# PACKAGE OPTIONS: --------------------
opts_knit$set(
#    aliases=c( # rename often used chunk options
#       h='fig.heigth',
#       w='fig.width'
#    ),
   eval.after=c( # list of chunk options to be
      'fig.cap'  # evaluated after the chunk
   )#,
#    header = "\\newcommand{\\rlang}{\\textbf{R}}\n"  # things
#    # that go into the preamble
#    width=75 # sets R session option 'width'
#    # that _may_ effect text output
)

# THEME OPTIONS: ------------------------
# knit_theme[['set']]("name_of_theme") # set a theme
# knit_theme[['get']]() # display available themes

# COMPILATION: --------------------------
# knit('knit_r_latex.rnw') # will only create the '.tex' file.
knit2pdf(
   input='knit_r_latex.rnw',
   clean=TRUE
) # knitr::knit2pdf = knitr::knit + tools::texi2pdf

# OTHER: --------------------------------
# purl('knit_r_latex.rnw', documentation=0L)
# This can be used to extract R code
```

2.3 Chunk appearance

2.3.1 Code

```
# local chunk options: tidy = FALSE
guff <- function(x){ x <- sin(x); print(x) }
# This is a comment
x = 1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1
```

The chunks above and below are generated by the same code, but different chunk options. Above has `tidy=FALSE`, whereas Below has

- `tidy=TRUE` (default), and
- `promt=TRUE`

as well as `tidy.opts` set with[8]

- `width.cutoff = 35`,
- `replace.assign = TRUE`, and
- `keep.comment = FALSE`.

```
> guff <- function(x) {
+     x <- sin(x)
+     print(x)
+ }
> x <- 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
+     1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
+     1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
+     1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 +
+     1
```

2.3.2 Selecting parts of code

I haven't found any use for this, but there are ways to select code for evaluation, display, etc., within a chunk. Several chunk options can be given as integer vectors instead of 'TRUE/FALSE'. These act as indexes. So, e.g. the chunk with content

```
1+1
2+2
```

and with chunk option `eval=c(-1)` will exclude the first *expression* from evaluation and thus yield the following output:

[8] The list specified by `tidy.opts` is passed to `tidy.source` in the `formatR` package.

```
## 1 + 1
2 + 2

## [1] 4
```

Similarily, there are ways to handle multiple plots created in a single chunk.

2.3.3 Results

```
paste(paste(1:14, collapse = " + "), "=", sum(1:14))

## [1] "1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 = 105"

1:45

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
```

Sometimes the output looks ugly. Knitr has no options to "directly" set the width of output. One can set a package option `width` that sets the R session option 'width', which may effect the text output.

However, I find that I do not include much chunk output, so this is a minor problem. For a more advanced way to control the output, see Section 2.10.2.

2.4 Reproducible code

To make your code reproducible it is a good idea - if possible - to

- *not* use absolute paths for file names. That way, the entire directory can simply be copied and run on another system. Also,
- do *not* included data derived hard coded values. If your data frame `DATA` needs to be looped over every row, write `for(i in 1:nrow(DATA)) ...` instead of `for(i in 1:521) ...` even though `nrow(DATA)` equals 521 (since this might change). This advice includes numbers to be put into the document text - see Section 2.5 on inline values.
- include 'meta information' - see Section 3.

(The above list is most certainly not exhaustive.)

2.5 Inline output

With `\ Sexpr{`*expression/variable*`}` one can print evaluated expressions or variables from the work space (global environment).

```
# local chunk options: cache = FALSE
x_precise <- 1.23456789
x_small <- 1.23456e-06
x_text <- "Colorless green ideas sleep furiously."
x_latex <- "If $f(x)=x$, then \\[ \\int_0^1f(x)\\,dx=\\frac12. \\]"
# Notice that in R you need to escape the '\' with another '\'.
```

Putting `x_precise` in you code gives: 1.2346
`x_small` will produce: 1.2346×10^{-6}
`x_text` will produce: Colorless green ideas sleep furiously.
`x_latex` will produce: If $f(x) = x$, then

$$\int_0^1 f(x)\,dx = \frac{1}{2}.$$

`floor(x_precise)+sin(pi/2)` will produce: 2

One potential problem with inline values is that *if* the values are set in a cached chunk, they may not be in the workspace at compile time, thus raising an error. Two possible solutions,

- put `cache=FALSE` in all chunks defining inline values, or,
- save inline values to a separate folder, the contents of which are loaded in a separate chunk (where `eval=FALSE`).

2.6 Plots

Graphics can be inserted into the document 'directly'.

- `dev` specifies the graphical device; for rnw-files (LaTeX) the default is `pdf`, for rmd (markdown) and rhtml (HTML) it is `png`.
- `fig.height, fig.width` - specifies height and width *in inches for all devices* (default $7 = 504$ pixels).
- `fig.align` can be 'left', 'right' and 'center'.
- plots are stored in a subfolder 'figure'.
- `fig.cap` gives the possibility to record a caption (in rnw)

If `fig.cap` is set, then Knitr will encapsulate the plot in the LaTeX figure environment, so additional options become available

- `fig.env` the environment (default is `figure`).
- `fig.pos` position arrangement for float (default: `''`).
- `fig.scap` if `NULL` (default) all words before '.', ';' or ':' will be used as the short caption. If non-`NULL` it provides the short caption.

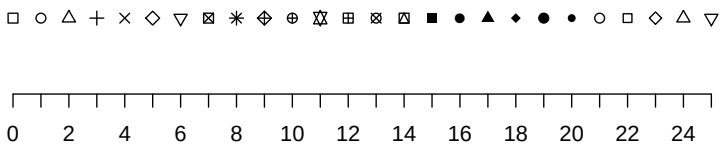


Figure 3: Possible point characters in R. This sentence will not show in the list of figures.

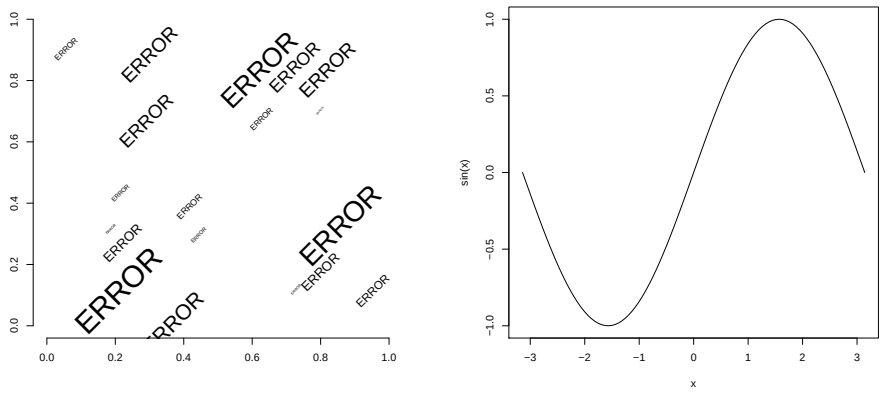
- `fig.lp` a possible prefix to the default label = chunk label. Default: `'fig:'`.

As an example, we can refer to 3 by `\ref{fig:pch}`.

```
# local chunk options: label='pch', fig.height=3,
# fig.width=7, fig.cap='Possible point characters in R. This
# sentence will not show in the list of figures.',
# fig.align='center', fig.scap=NULL,
# tidy.opts=list(width.cutoff=60)
plot(x = 0:25, y = rep(0, 26), pch = 0:25, xaxp = c(0, 25, 25),
    yaxt = "n", xlab = "", ylab = "", bty = "n")
```

With the `subfig` package attached in the preamble, we can also create figures with subplots. For this task it is convienient to to specify `out.width` instead of `fig.width`. The former can be set, as text, to LaTeX variables. The following code produces Figure 4, which consists of subfigure 4a and 4b. (Subplots inherit the *label* of the main figure with the number of 'order of appearance concatenated to it.)

```
# local chunk options: label="subplots",
# out.width='0.49\\linewidth',
# fig.cap="Subfigures created with Knitr",
# fig.align='center',
# tidy.opts=list(width.cutoff=35 ),
# fig.subcap=c("One plot", "Another plot")
plot(0,0,type='n', xlab="", ylab="", bty='n',xlim=0:1, ylim=0:1)
n <- 20; text(runif(n),runif(n),"ERROR", srt=45, cex=3*runif(n))
curve(sin(x),-pi,+pi)
```



(a) One plot (b) Another plot

Figure 4: Subfigures created with Knitr

2.6.1 Attach the graph

With the LaTeX package attach we can also refer to the graph that knitr creates and include it. By default, knitr stores plots in a subdirectory 'figure' with the same name as the chunk label.

```
# local chunk options:
#     label="plot_attach"
#     fig.cap="A graph that is also attached.
#              \\attachfile{figure/plot_attach.pdf}"
n <- 5005; x = rnorm(n); y = rnorm(n) + sqrt(abs(x))
plot(x,y, xlab="", ylab="", xaxt='n', yaxt='n', bty='n')
```

2.6.2 An alternative

An alternative way to include a plot is to specify all but the `\includegraphics...`.

`\begin{figure}`

(A chunk with `echo=FALSE` that specifies a graph, e.g. `plot(1:2,2:1)`)

`\caption{A caption}\ref{a_ref}`
`\end{figure}`

2.7 Tables

There is no built in option to tabulate data and/or create table environments. We can of course display tables within the output of a chunk

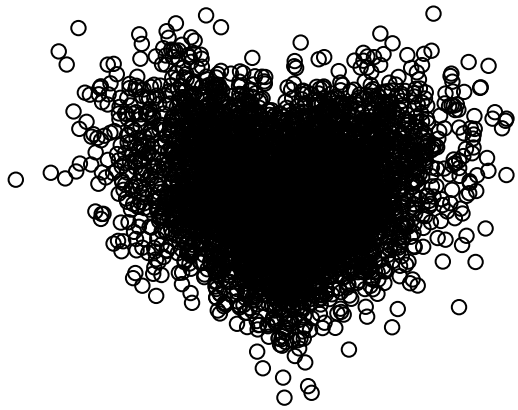


Figure 5: A graph that is also attached.

```
m <- matrix(rpois(20, 10), ncol = 4, nrow = 5)
rownames(m) <- c("Age", "Height", "Weight", "Socioeconomic status",
    "Foo")
colnames(m) <- rep(c("Male", "Female"), 2)
m
```

```
##                       Male Female Male Female
## Age                      6      9   12     10
## Height                   8      9   11      9
## Weight                  11      9   12      9
## Socioeconomic status    11     18   18     11
## Foo                      5      7    9     12
```

but that isn't very pretty.

There are a few functions for turning R tables, dataframes, and matrices into LaTeX code, we recommend `latex` from the `Hmisc` package. This function features the highly useful `cgroup` for 'grouping' columns together (in equal sizes, or as determined by `n.cgroup`) and a similar argument for rows. Note; one has

Hey!	Group A		Group B	
	Male	Female	Male	Female
Continuous				
Age	6	9	12	10
Height	8	9	11	9
Weight	11	9	12	9
Categorical				
Socioeconomic status	11	18	18	11
Foo	5	7	9	12

Table 3: My very presentable data. 

to provide a caption for the table to get a label. Normally, we use `latex` to write the table to a file which we `\input` into our document, but there is a better way.

```
# local chunk option: results='asis'
# library(Hmisc)
dir.create("table", showWarnings=FALSE) # create subdirectory
write.table(m, file="table/presentable.txt") # + file to attach
dummy <- latex(
   object=m,
   file="",
   label="tab:ex",
   cgroup=c("Group A", "Group B"),
   #n.cgroup=c(2,2),  # not needed
   rgroup=c("Continuous", "Categorical"),
   n.rgroup=c(3,2),
   title="Hey!",
   caption="My very presentable data.
   \\attachfile{table/presentable.txt}",
   caption.lot="Presentable data", # for the LOT
   caption.loc="bottom" # 'top' by default
   #ctable=TRUE
   )
```

With chunk option `results='asis'` the results of the chunk is not 'interpreted' in any way, just inserted into the LaTeX file. Since the result of the chunk *is* LaTeX code, it does its thing - which is to produce Table 3.

2.7.1 Attach the table

Similarily to how we attached the plot in Section 2.6.1 we can attach a table in a suitable format, so that the reader can easily access the data therein. We

need only decide what format and where to put the data. I find it convenient to put these in a subdirectory, say, `table`. N.B. the attached file can have (almost) *any* extension.

2.8 Cache

If `cache=TRUE` the contents of a chunk, along with its output, is stored and reloaded 'lazily' (i.e. if needed). (If you Knitr the rnw file, you can see unevaluated variables as 'promises' in the work space.)

```
x <- 0
Sys.sleep(5)
x <- 1
```

In general, a cached chunk will only be re-evaluated if there is any change to the code within.

2.8.1 Cache dependencies

```
apa <- 10
```

Due to caching, if a value defined in one cache is changed, a reference to it in another chunk will not *unless* the latter is made depend on the former.

```
# local chunk options: tidy=FALSE, dependson="cache_A"
apa + 1

## [1] 11
```

There exists a chunk option `autodep` which is supposed to identify dependencies automatically. It is said to be 'experimental'.

2.8.2 Update caching

By creating arbitrary variables in the chunk options, we can make a chunk update its cache depending on factors other than change of code. The following chunk will update its cache whenever the R version is changed or when a change to `example_cache.txt` is made.

```
# local chunk options: cache=TRUE,
# redo1=file.info('example_cache.txt')[['mtime']],
# redo2=R.version.string
scan("example_cache.txt")

## [1] -87
```

2.9 More chunkiness

2.9.1 Chunk recycling

Chunks can be referenced to outside or inside other chunks. If we were to make repeated use of the code within the following chunk, with label `layers`,

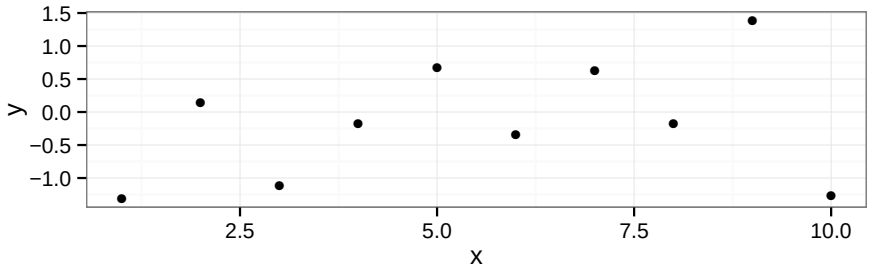
```
geom_point() + theme_bw()

## NULL
```

we can simply make a reference to it. The following code

```
ggplot(data.frame(x=1:10, y=rnorm(10)), aes(x,y)) +
   <<layers>>
```

will 'appear to be':

```
ggplot(data.frame(x=1:10, y=rnorm(10)), aes(x,y)) +
geom_point() + theme_bw()
```



and produce the corresponding output. Similarily you can reuse an entire chunk anywhere in your document by creating an empty chunk with the same label. (A non-empty chunk with the same label will throw an error.)

2.9.2 External chunks

There is a special function `read_chunk` to read chunks defined in separate files.

```
read_chunk("example_external_chunk.r")
cat(readLines("example_external_chunk.r"), sep = "\n")

## ## @knitr external
## silly_function <- function(x) -abs(x)
```

If a chunk is read this way we can 'extract it' by setting up a empty chunk with the same label which yields

```
silly_function <- function(x) -abs(x)
```

Although, it is unclear what the benefit is compared to having a chunk with

```
source("example_external_chunk.r")
```

2.9.3 Child documents

The file `example_child.rnw` is a *child* document to the current rnw file. Now, by including an empty[9] chunk with option `child` set to the filename, we include it in way similar to LaTeX's `\input`.

(This is text from the child document.)

```
k <- 12^2  # (This code is from the child document)
```

The advantage of child documents is that they allow you to break down a large document into manageable parts. By specifying the parent document with `set_parent`, we can compile the child document borrowing the preamble of the parent.

2.10 Under the hood: hooks!

Knitr uses a set of functions called 'hooks' to determine the output of the R code. One is allowed to add new hooks or to redefine existing ones. Knitr distinguishes between *chunk* and *output* hooks.

2.10.1 Chunk hooks

A function `frameish` defined with `knit_hook$set` will be a chunk hook if defined with arguments `before`, `options`, and `envir`. They work as follows:

- `before`: if `frameish` is any non-`NULL` value, this hook will be called upon. Each chunk hook is called twice; before *and* after the chunk itself. The first time, `before` is `TRUE` and the second time it is `FALSE`.
- `options`:`options` is a list of all arguments given as chunk options.
- `envir`: is the working environment.

```
knit_hooks$set(frameish=function(before, options, envir){
   a <- options$frameish
   if(!is.numeric(a)) a <- 0.4
   s <- paste0("\\noindent\\rule{\\linewidth}{",a,"pt}")
```

[9] It does not necessarily have to be empty, but it seems that the contens are ignored.

```
   # s is a line
   z <- capture.output(str(
      options[c("label", "eval", "results", "frameish")]
      ))
   z <- gsub("$", "\\$", z, fixed=TRUE)
   z <- paste(z, sep = "", collapse = "\\newline\\indent\\indent")
   # z shows the 'str' of a subset of 'options'
   u <- paste("\\verb|", ls(envir), "|", sep = "", collapse = ", ")
   # u lists the chunk environment
   if(before){
      s
   } else {
      paste(s, "Some  chunk options:", z,
         "\\newline Objects available in the above chunk:", u)
   }
})
```

To default hooks (all of which are of the output kind) and the above hook `frameish` in action:

```
names(knit_hooks$get(default = TRUE))

## [1] "source"   "output"   "warning"  "message"  "error"    "plot"
## [7] "inline"   "chunk"    "document"
```

Some chunk options: List of 4
$ label : chr "chunk.hook.test"
$ eval : logi TRUE
$ results : chr "markup"
$ frameish: num 1.5
Objects available in the above chunk: `a_list_of_files`, `apa`, `dummy`, `FILE`, `files`, `guff`, `hook_output`, `k`, `m`, `n`, `silly_function`, `x`, `x_latex`, `x_precise`, `x_small`, `x_text`, `y`

2.10.2 Output hooks

The output hook will have arguments[10] `x` and `options`. They works as follows:

- `x`: this is the 'captured response' from R.
- `options`: same as before; `options` is a list of all arguments given as chunk options.

[10] Actually, hooks `inline` and `document` have only argument `x`.

Example: how to trim code output
We saw in Section 2.3.3 that output can be ugly. E.g.

```
paste(paste(1:14, collapse = " + "), " = ", sum(1:25))

## [1] "1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14  =  325"

1:25

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25
```

One can set a package option `width` that sets the R session parameter 'width' but it is not a guarantee for aestethic output. As an example on how to remedy ugly output Yihui gives an example of the following hook:

```
hook_output = knit_hooks$get("output")
knit_hooks$set(output = function(x, options) {
    # this hook is used only when the linewidth option is not NULL
    if (!is.null(n <- options$linewidth)) {
        x = knitr:::split_lines(x)
        # any lines wider than n should be wrapped
        if (any(nchar(x) > n))
            x = strwrap(x, width = n)
        x = paste(x, collapse = "\n")
    }
    hook_output(x, options)
})
```

Which can be put to the test

```
# local chunk option: linewidth=45
paste(paste(1:14, collapse = " + "), " = ", sum(1:25))

## [1] "1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 +
## 10 + 11 + 12 + 13 + 14 = 325"

1:25

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
## 16 17 18 19 20 21 22 23
## [24] 24 25
```

3 Meta information

It is always a good idea to end with this chunk:

```
sessionInfo()

## R version 3.0.2 (2013-09-25)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
##
## locale:
## [1] LC_COLLATE=Swedish_Sweden.1252  LC_CTYPE=Swedish_Sweden.1252
## [3] LC_MONETARY=Swedish_Sweden.1252 LC_NUMERIC=C
## [5] LC_TIME=Swedish_Sweden.1252
##
## attached base packages:
## [1] splines   stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] fortunes_1.5-2  codetools_0.2-8 ggplot2_0.9.3.1 Hmisc_3.12-2
## [5] Formula_1.1-1   survival_2.37-4 knitr_1.5
##
## loaded via a namespace (and not attached):
##  [1] cluster_1.14.4     colorspace_1.2-4   dichromat_2.0-0
##  [4] digest_0.6.3       evaluate_0.5.1     formatR_0.10
##  [7] grid_3.0.2         gtable_0.1.2       highr_0.3
## [10] labeling_0.2       lattice_0.20-24    MASS_7.3-29
## [13] munsell_0.4.2      plyr_1.8           proto_0.3-10
## [16] RColorBrewer_1.0-5 reshape2_1.2.2     rpart_4.1-3
## [19] scales_0.2.3       stringr_0.6.2      tools_3.0.2
```

With the following commands

```
files <- c(
   "knit_r_latex.rnw",
   "_compile.r",
   "example_cache.txt",
   "example_child.rnw",
   "example_code_LaTeX.tex",
   "example_external_chunk.r",
   "documentation_attachfile.pdf",
   "example_references.bib",
   "image_donald_knuth.jpg",
   "image_yihui.jpg",
   "image_yohio.jpg",
   "image_knit-logo.png",
   "ucrlogo.pdf"
   )
dir.create(path="reproducibility")
for(FILE in files)
```

```
   file.copy(from=FILE, to=paste0("reproducibility/", FILE))
zip(zipfile="reproducibility", files="reproducibility")
unlink(x="reproducibility", recursive=TRUE, force=TRUE)
```

you can create a zip-file with all files necessary to recreate the document. However, *Adobe does not allow you to attach a zip file to a pdf*[11].

Of course, for documentation purposes there is no difficulty in attaching each file separately.

```
# this chunk creates a LaTeX code string to include all
# files listed in 'files' from the previous chunk
a_list_of_files <-
   paste0("\\begin{itemize}\n",
      paste0(
         "\\item ",
         gsub("_", "\\_", files, fixed=TRUE),
         " \\attachfile{",
         files,"}",
         collapse="\n"
         ),
      "\n\\end{itemize}")
# use \Sexpr{a_list_of_files} to include the list
```

Source files:

- knit_r_latex.rnw 
- _compile.r 
- example_cache.txt 
- example_child.rnw 
- example_code_LaTeX.tex 
- example_external_chunk.r 
- documentation_attachfile.pdf 
- example_references.bib 
- image_donald_knuth.jpg 
- image_yihui.jpg 

[11] As far as I understand it, this can not be unlocked in a trivial manner and is due to difficulties for firewalls in scanning such attachments.

- image_yohio.jpg 
- image_knit-logo.png 
- ucrlogo.pdf