Example of **knitr** with included MATLAB and Stata output

Han Oostdijk

August 12, 2015

Contents

1 introduction **2**

2 the structure of the main input document **2**

3 setup chunk **3**

4 inline_1 chunk **3**

5 the other chunks **3**

6 initialization **4**

7 an R plot **4**

8 dynamic section **4**

9 contents of MATLAB session **6**

10 contents of Stata session **7**

11 software versions used **10**

1 introduction

This document shows how **knitr** can be used in combination with LaTeX to produce dynamic documents. **knitr** started and is mostly used (at least I think so) in the R environment. However also for some other environments engines have been created to handle these. See `http://yihui.name/knitr/demo/engines/` for a list. This document shows how also results from the MATLAB and Stata environments can be used in **knitr**. The R code to do this is new but builds on and is similar to the code for the other engines. Because each engine is fully isolated from the others it is possible to combine results from the supported environments.
This document shows how to do this. It is part of a set that contains

- **example.pdf**: this document (the final output)
- **example.rnw**: the main input for the document
- **example1.r** : the R code of which the 'chunks' will be included in the document
- **knitr_engine_functions.r** : the engine code for MATLAB and Stata (besides the previously existing code for some other engines)

This document will also contain some general examples of the **knitr** functionality but for details we have to refer to the website of its author Yihui Xie: `http://yihui.name/knitr/`
The idea here is that the main input document contains the layout and the code of a report. The **knitr** application takes this document and ensures that the code is executed and inserted in the layout. If the code e.g. reads a file and the file is updated then by just 'knitting' the document again an updated report is created (without any changes to the document). Keywords associated with this way of working are **Reproducible research** and **Literate Programming**.
Where **knitr** in combination with LaTeX gives in my opinion the best control over the final output one nowadays also often sees the combination with **R Markdown**: `http://rmarkdown.rstudio.com/` . The RStudio environment supports both combinations very well: 'knitting' is done with a click on a button.

2 the structure of the main input document

The main document (in this case **example.rnw**) contains text fragments and code chunks. The text fragments contain LaTeX commands and the fixed phrases of the text. These will be copied to a tex file by **knitr**. A code chunk is analyzed by **knitr** and transfered to the appropriate engine after which the final results (dynamic phrases) are copied to the tex file. A code chunk is a fragment that starts with $<< \ldots >>=$ on a new line (where the ... part is variable) and ends with a @ on a newline.
In our case the main input document starts with the 'standard' LaTeX commands followed by the first code chunk called 'setup'. It is immediately followed by another code chunk (called 'inline_1') and after that the first text follows. After that we alternate the text fragments and the code chunks. We will give some detail about the first code chunks in the next sections.

3 setup chunk

In the first chunk we set up the environment:

- we set the default values for the chunk options by using the opts_chunk$set function.
- we set a hook that creates a named section 'xxx' if we include ss='xxx' in the chunk definition.
- we set some general options.
- we extend the system path so that the Python and Stata libraries can be found when they are needed. NB Python is not used in this document.
- we assign the newly developped code as the MATLAB and Stata engine.

This set up could be used for more than one document.

4 inline_1 chunk

In contrast to the setup chunk the inline_1 chunk contains information that is specific for this document. It contains the read_chunk call. It indicates a file that contains R fragments that later in this document can be used as code chunks. The names of the chunks can be chosen by the user: I have chosen for **inline**_xx for chunks defined in this document and for **rc**_yyy for chunks defined in the read_chunk file. In this chunk also some variables are defined, but because the chunk has as option echo=FALSE this is not visible at the place where the inline_1 chunk is executed. However we can show the variables here:
approx_e : 2.71
approx_pi : 3.14

5 the other chunks

By reading the main input file **example.rnw** it will become clear how the various code chunks work. If a chunk name is found in the file where the read_chunk call pointed to, it will be used and otherwise the inline contents is used. Some short comments on the other code chunks:

- rc_init: uses option ss="initialization" and therefore generates the section "initialization". Because chunk option 'echo' is FALSE the code is not shown in contrast to the result (because chunk option 'results' has the default value 'TRUE')
- setup_2: is used to avoid a very long chunk definition in the next chunk (the definition should always be contained in one line). Because chunk option 'include' is FALSE nothing is shown in the document

- inline_2 executes some R-code that produces a figure in a section that is called "an R plot" because of the "ss" chunk option. Use a reference to {fig:chunkname} to refer to the whole figure and a reference to {fig:chunknamex} to subfigure x where x is a sequence number.
- inline_3 has a chunk option 'code'. This means that the expression "myfun(today)" is executed in the R environment. Because a section (with name "dynamic section") is already specified in LaTeX the "ss" chunk option is not used here.

6 initialization

```
## [1] "workdir: D:/data/R/knitr_matlab_stata"
```

7 an R plot

```
set.seed(1121)
x=rnorm(20)
par(mar=c(4,4,.1,.1),cex.lab=.95,cex.axis=.9,mgp=c(2,.7,0),tcl=-.3,las=1)
## two plots side by side (option fig.show='hold')
boxplot(x)
hist(x,main='')

today = Sys.Date()
```

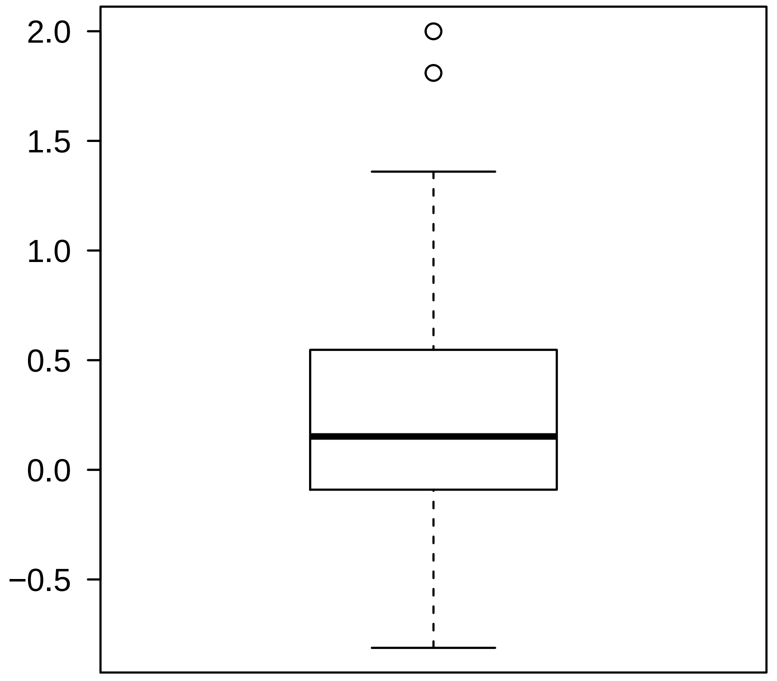
The whole figure is Figure 1 while the boxplot can be found in Figure 1a and the histogram in Figure 1b.
The variable 'today' will be used as argument in section 8.

8 dynamic section

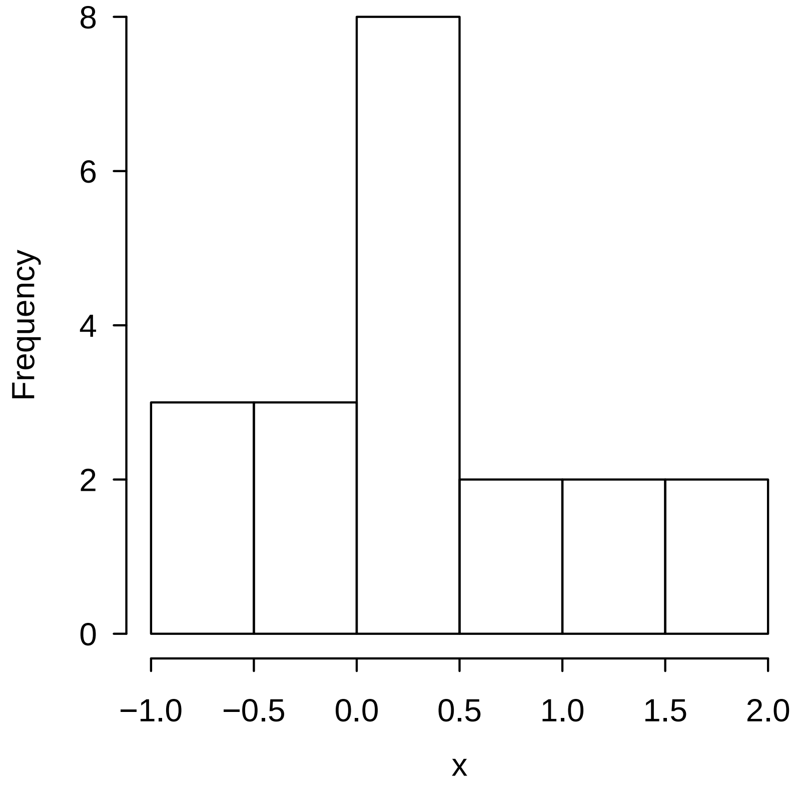
The remainder of this section is determined by the value of the variable 'today' that was set earlier.

```
myfun(today)

## [1] "today it is Wednesday"
```



(a) the box plot



(b) the histogram

Figure 1: plot created by R-code

9 contents of MATLAB session

Output of a MATLAB session can be included. By saving plots in the eps format these plots can be included in the document as shown in the example. It proved to be necessary to add some waiting time to give the MATLAB session the opportunity to close the logfile (where the output is copied from) and to write the plotfile. We set this waiting time with the chunck-option `waittime`. We specified in the `setup` section a default value of 20 seconds. The chunck option for this section is specified (but remember the actual specification has no linebreaks) as

```
<<mymatlab, engine="matlab", engine.path="C:/Program Files/MATLAB/R2015a/bin/matlab.exe",
engine.opts="-nosplash -nodesktop -minimize", echo=TRUE, eval=TRUE, tidy=TRUE,
cache=FALSE, background="#ffffff">>=
```

MATLAB code that will be executed (and results preceded by ##):

```
fprintf('start of MATLAB session (version %s)\n',version)
rng(1954)
x=rand(1,6);
y=x+10
x=rand(1,6);
y=x+10
rng(1954)
x=rand(1,6);
y=x+10
figure('Visible','off')
plot(1:3,[2 1 2],'b-',1:3,[2 3 2],'ro')
axis([0,4,0,3.5])
legend({'line','dot'},'Location','northeast')
xlabel('between zero and four')
ylabel('between 0.0 and 3.5')
print('matlabgraphfile','-depsc')
fprintf('end of MATLAB session\n')
```

```
## start of MATLAB session (version 8.5.0.197613 (R2015a))
##
## y =
##
##    10.9928   10.6461   10.9186   10.7415   10.7292   10.5120
##
##
## y =
##
##    10.3875   10.6286   10.4533   10.7339   10.0963   10.2445
##
##
## y =
##
##    10.9928   10.6461   10.9186   10.7415   10.7292   10.5120
##
## end of MATLAB session
```

The plot created by the MATLAB session is Figure 2. It is inserted in the document with:

```
\begin{figure}[tbp]
\centering
\includegraphics[width=.45\textwidth]{matlabgraphfile}
\caption{plot created by MATLAB code}\label{fig:matlabgraph}
\end{figure}
```

10 contents of Stata session

In the same way as for MATLAB the results of a Stata session can be included. Here we did not encounter time problems so no waiting time was necessary. As in the MATLAB case figures can be included by using the eps format. See Figure 3 for an example.
Stata code that will be executed (and results preceded by ##):

```
about
use "D:/data/R/knitr_matlab_stata/stata_testfile.dta",clear
replace sex = "X"  if _n > 5
```

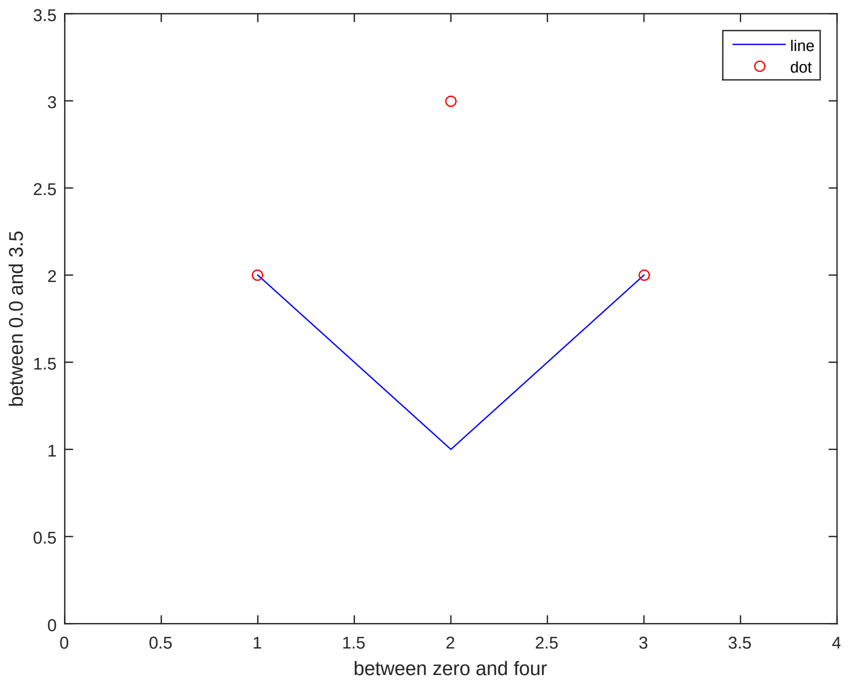


Figure 2: plot created by MATLAB code

```
label define gender 0 "F" 1 "M" 2 "X"
encode sex, generate(gender) label(gender)
list , nolabel
quiet graph box length, over(gender) title(knitr graph) name(graph1,replace)
quiet graph export "D:\data\R\knitr_matlab_stata\statagraph1.eps", as(eps) preview(off) replace
```

```
##  . about
##
## Stata/SE 13.1 for Windows (64-bit x86-64)
## Revision 03 Jun 2015
## Copyright 1985-2013 StataCorp LP
##
## Total physical memory:    16737264 KB
## Available physical memory: 14339560 KB
##
## Single-user Stata perpetual license:
##        Serial number:  401306217178
##          Licensed to:  Han Oostdijk
##                        Han Oostdijk Quantitative Consultancy
##
## . use "D:/data/R/knitr_matlab_stata/stata_testfile.dta",clear
##
## . replace sex = "X"  if _n > 5
## (2 real changes made)
##
## . label define gender 0 "F" 1 "M" 2 "X"
##
## . encode sex, generate(gender) label(gender)
##
## . list , nolabel
##
##      +-----------------------+
##      | sex   length   gender |
##      |-----------------------|
##   1. |   F       23        0 |
##   2. |   M       34        1 |
##   3. |   F       25        0 |
##   4. |   M       26        1 |
##   5. |   F       23        0 |
##      |-----------------------|
##   6. |   X       24        2 |
##   7. |   X       30        2 |
##      +-----------------------+
##
## . quiet graph box length, over(gender) title(knitr graph) name(graph1,replace)
##
## . quiet graph export "D:\data\R\knitr_matlab_stata\statagraph1.eps", as(eps) preview(off) r
## > eplace
```

11 software versions used

We used

- RStudio Version 0.99.441 - © 2009-2015 RStudio, Inc.
- pdfTeX, Version 3.1415926-2.4-1.40.13 (MiKTeX 2.9)

```
sessionInfo()
```

```
## 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
```

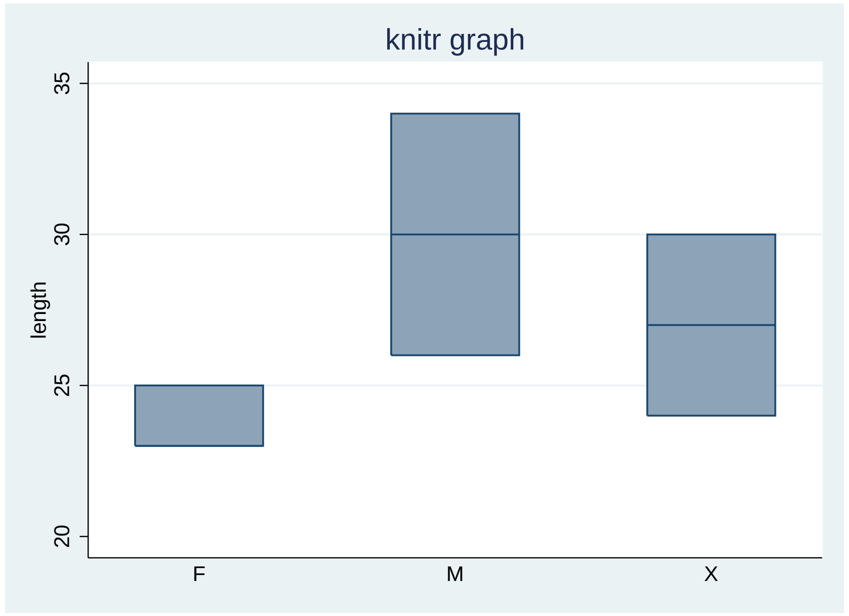


Figure 3: plot created by Stata code

```
## [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_1.10.5
##
## loaded via a namespace (and not attached):
## [1] magrittr_1.5  formatR_1.2   tools_3.2.0   stringi_0.4-1 highr_0.5
## [6] stringr_1.0.0 evaluate_0.7
```