Dynamic documents with R

Emmanuel Kemel (HEC Paris, CNRS)

Notions seen in these slides

- xtable and stargazer
- tikz
- knitr

Tables

As we already saw, tables can be exported

- as is, using **sink()** that prints the shell output in a .txt or .doc file
- in a .csv file

There are several options that are specific to LaTeXor HTML.

Outline

xtable

The library xtable contains a generic function **xtable()** that applies to

- data.frames
- matrixes
- anovas
- lm, glm...
- times series

Example

```
mat=matrix(rnorm(25),5,5)
xtable(mat)
```

```
% latex table generated in R 3.6.3 by xtable 1.8-4 package
% Fri Oct 16 12:19:29 2020
\begin{table}[ht]
\centering
\begin{tabular}{rrrrrr}
  \hline
 & 1 & 2 & 3 & 4 & 5 \\
  \hline
1 & 0.17 & 0.21 & 0.27 & -0.15 & -1.08 \\
  2 & -1.75 & -1.78 & -1.24 & 0.14 & -1.30 \\
  3 & -0.68 & -0.96 & 2.22 & 1.39 & 0.20 \\
  4 & 1.36 & -0.44 & 0.11 & 0.72 & 0.38 \\
  5 & -0.24 & 2.40 & 0.12 & 0.16 & 0.42 \\
   \hline
\end{tabular}
\end{table}
```

Example (continued)

	1	2	3	4	5
1	-0.72	-0.79	0.39	0.89	0.42
2	-0.42	0.57	-0.31	-2.44	-1.95
3	0.51	0.95	0.56	3.03	-0.69
4	0.27	0.87	0.19	-0.60	1.61
5	0.17	0.83	0.06	-1.87	-0.15

Formatting

Several options may be useful

- include.rownames
- caption
- positions
- scalebox=0.7,
- table.placement="H" (with LaTeXpackage float)

Example

```
colnames(mat)=c("$\\alpha$","$\\beta$","$\\delta$","$\\gamma$","$\\chi$")
print(xtable(mat, caption="Printing a matrix in \\LaTeX"),
include.rownames=FALSE,scalebox=0.8,
caption.placement = "top",
sanitize.colnames.function = identity,
table.placement="H")
```

Table: Printing a matrix in LaTeX

α	β	δ	γ	χ
-0.72	-0.79	0.39	0.89	0.42
-0.42	0.57	-0.31	-2.44	-1.95
0.51	0.95	0.56	3.03	-0.69
0.27	0.87	0.19	-0.60	1.61
0.17	0.83	0.06	-1.87	-0.15

More formatting

xtable contains options

- align, digits, and display that allow to refine formatting
- these options can be set to "automatically optimized values" with option auto=TRUE

```
data(mtcars)
dat <- mtcars[1:3, 1:6]
x <- xtable(dat)
print(x)
```

	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.00	6.00	160.00	110.00	3.90	2.62
Mazda RX4 Wag	21.00	6.00	160.00	110.00	3.90	2.88
Datsun 710	22.80	4.00	108.00	93.00	3.85	2.32

Example with auto

```
x <- xtable(dat, auto=TRUE)
print(x)
```

	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.0	6	160	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875
Datsun 710	22.8	4	108	93	3.85	2.320

More formatting

- changing the exponent sign: math.style.exponents = TRUE in print()
- sanitizing
  For each component of a table, it is possible to specify a function that applies LaTeXfunctions
  Example

```
large <- function(x) { paste0(' {\\ Large {\\ bfseries ', x, ' }} ') }
italic <- function(x) { paste0(' {\\ emph { ', x, ' }} ') }
print(xtable(dat),
sanitize.rownames.function = italic,
sanitize.colnames.function = large)
```

More formatting

- latex.environment="center"
- align receives the options in tabulate
  ex: align(table) <- rep("r", 6)
  c("l", "|c", "|R { 3cm } ", "|L { 3cm } ", "| p { 3cm } |")
- rotating the whole table floating.environment = "sidewaystable"
- rotating names: rotate.rownames = TRUE, rotate.colnames = TRUE
- positioning horizontal lines: hline.after = c(1)
- width=" 0.9\\ textwidth"

xtable for models

xtable can also be applied to model objects

```
data(mtcars)
mod=lm(mpg~cyl+wt, mtcars)
print(xtable(mod))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	39.6863	1.7150	23.14	0.0000
cyl	-1.5078	0.4147	-3.64	0.0011
wt	-3.1910	0.7569	-4.22	0.0002

Outline

Principle

The package stargazer aims at begin easy to use

- regarding Latex commands
- regarding R commands

Example:
stargazer(data) produces the equivalent of
print(xtable(summary(data)))

Example

```
library(stargazer)
data(cars)
stargazer(cars)
```

Table:

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
speed	50	15.400	5.288	4	12	19	25
dist	50	42.980	25.769	2	26	56	120

Display Options

- summary=FALSE for printing raw data
- rownames=FALSE
- flip = TRUE for transposing

Example (continued)

```
data(cars)
stargazer(cars[1:8,], summary=F)
```

Table:

	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26

Displaying several regression models

The package is particularly adapted for printing several models

- different models can be displayed side by side
- a large diversity of models can be handled.

```
data(mtcars)
mod1=lm(mpg~cyl,mtcars)
mod2=lm(mpg~cyl+wt,mtcars)
mod3=lm(mpg~cyl*wt,mtcars)
```

Illustration

Table: Regressions

	Dependent variable:		
	mpg		
	(1)	(2)	(3)
cyl	−2.876***	−1.508***	−3.803***
	(0.322)	(0.415)	(1.005)
wt		−3.191***	−8.656***
		(0.757)	(2.320)
cyl:wt			0.808**
			(0.327)
Constant	37.885***	39.686***	54.307***
	(2.074)	(1.715)	(6.128)
Observations	32	32	32
R^2	0.726	0.830	0.861
Adjusted R^2	0.717	0.819	0.846
Residual Std. Error	3.206 (df = 30)	2.568 (df = 29)	2.368 (df = 28)
F Statistic	79.561*** (df = 1; 30)	70.908*** (df = 2; 29)	57.618*** (df = 3; 28)
Note:			*p<0.1; **p<0.05; ***p<0.01

More options

The generic function stargazer() allows for a lot of options that specify editing and calculations

- For descriptive statistics
  the statistics that are needed can be specified

```
stargazer(mtcars, nobs = FALSE, mean.sd = TRUE, median = TRUE,
stargazer(mtcars,summary.stat = c("n", "p75", "sd"))
```

- For regressions
  - option **se** allows to specify the function used to compute standard errors
  - option **ci**
- For more information and examples see
  the stargazer manual
  http://jakeruss.com/cheatsheets/stargazer.html

Example of regression with robust se and ci

```
data(mtcars)
mod3=lm(mpg~cyl*wt,mtcars)
library(sandwich)
cov=vcovHC(mod3, type = "HC")
robust.se <- sqrt(diag(cov))
stargazer(mod3,mod3, se=list(NULL, robust.se),ci=TRUE, ci.level=0.95)
```

Example of regression with robust se and ci

Table: Standard versus robust CI

	Dependent variable:	
	mpg	
	(1)	(2)
cyl	−3.803***	−3.803***
	(−5.773, −1.833)	(−5.458, −2.149)
wt	−8.656***	−8.656***
	(−13.203, −4.108)	(−12.294, −5.017)
cyl:wt	0.808**	0.808***
	(0.167, 1.450)	(0.296, 1.321)
Constant	54.307***	54.307***
	(42.297, 66.317)	(43.782, 64.832)
Observations	32	32
R^2	0.861	0.861
Adjusted R^2	0.846	0.846
Residual Std. Error (df = 28)	2.368	2.368
F Statistic (df = 3; 28)	57.618***	57.618***
Note:		*p<0.1; **p<0.05; ***p<0.01

Figures

- We saw how to export R plots into separated files (.pdf, .gif.)
- The is a package tikzDevice, that converts R plots into (non efficient) tikz script.

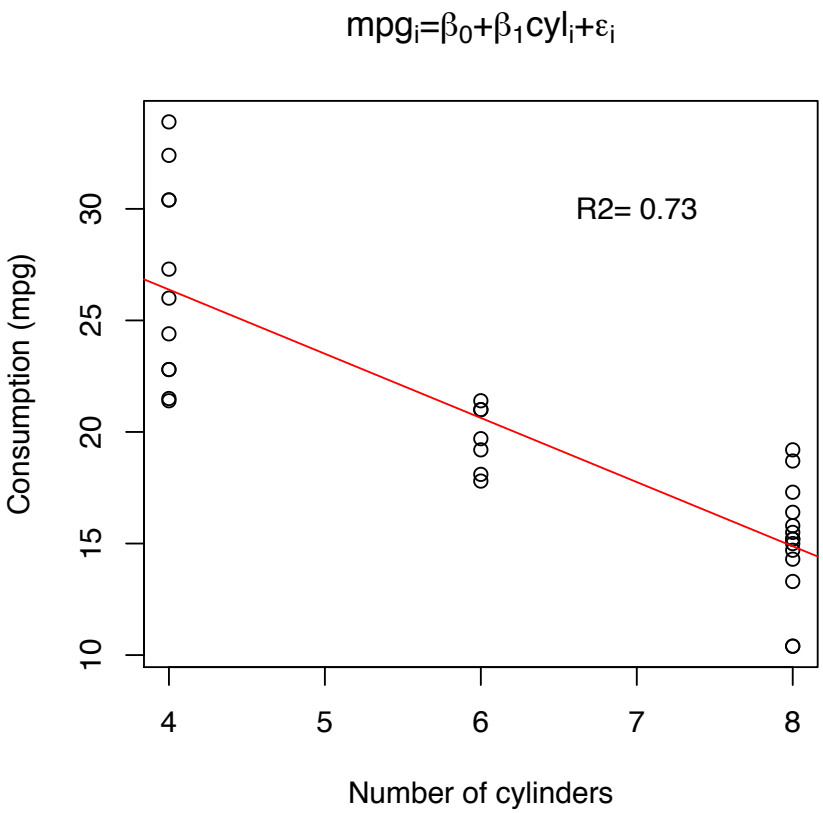
Example: standard plot code

Here is a code of a standard R plot

```
data(mtcars)
mod1=lm(mpg~cyl,mtcars)
plot(mpg~cyl,mtcars,
main=expression(paste(mpg[i],"=",beta[0],"+",beta[1],"cyl"[i],"+",epsilon[i])),
xlab="Number of cylinders", ylab="Consumption (mpg)")
text(7,30,paste("R2=",round(summary(mod1)$r.squared,2)))
abline(mod1, col="red")
```

Example: standard plot result

Here is the result



Example: tikz plot

Here is the code to turn the R plot into a tikz figure

- In R

```
library(tikzDevice)
tikz(file="my_picture.tikz", width=3, height=3)
plot(mpg~cyl,mtcars,main="$mpg_i=\\beta_0+\\beta_1 cyl_i+\\epsilon_i$",xlab="Number of cylinders", ylab="Consum
text(6.5,30,"$R^2=$")
text(7.5,30,round(summary(mod1)$r.squared,2))
abline(mod1, col="red")
dev.off()
```

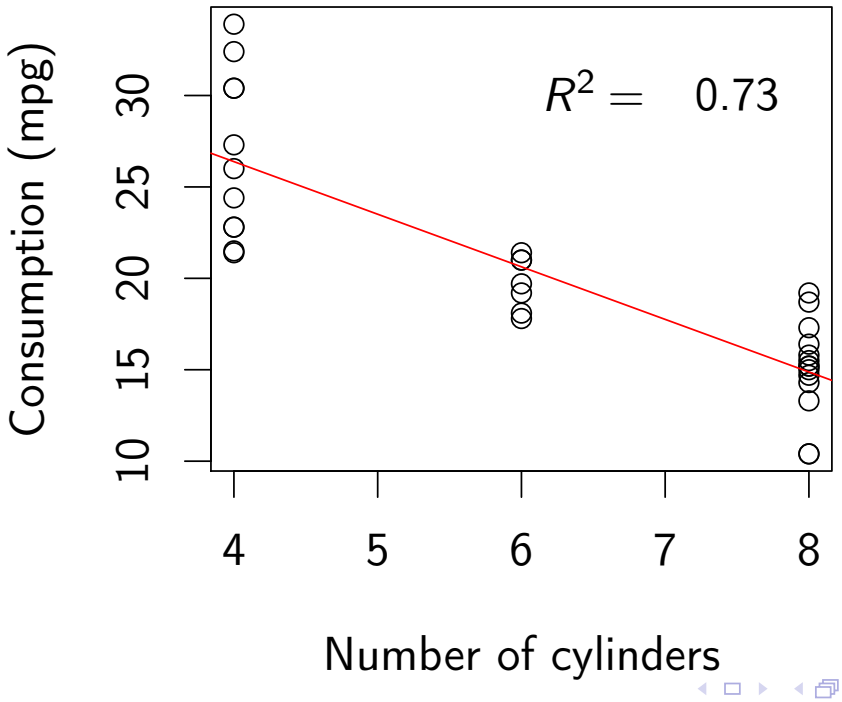
- in LaTeX

```
\begin{figure}
\include{"my_picture.tikz"}
\end{figure}
```

Example: the result

Here is a tikz figure

$$mpg_i = \beta_0 + \beta_1 cyl_i + \varepsilon_i$$



Comments

- The ticks file can then be included into a latex document
- The tikz file takes less memory
- The figure is creates when the latex document is compiled, an advantage is that
  - it is possible to include latex formula on the plot text
  - the font of the plot text will be the same a the one of the rest of the latex document

Outline

Motivation

- Principle
  Having a single file that contains the text and the code for the statistical analysis

- Advantages
  - Automating table and figure transfer
  - Reproductibility (for you and for others)

Literate Programming

The idea of literate programming is to mix

- the program code to do computing
- the narratives that explain what is being done by the program code

Overall, literate programming involves three steps

1. parse the source document and separate code from narratives
2. execute source code and return results
3. mix results from the source code with the original narratives

Inline input vs chunk

- Suppose you can to include a result in a line of text,
  The value of π is $\{\{round(pi, 2\}\}$
- Longer/more complicated pieces of code can be included in chunks
  They are lines of code that can be
  - displayed or not
  - executed or not

  Which produce in output to be displayed.

Weave and tangle

Because you document contains both source code and narratives, the two natural operations that can be considered are

- executing the source code: weaving
- extracting the source code as a script: tangling

Dynamic documents with R

- The previous package/language that was used to creating dynamic documents with R was Sweave
  The language had several limitations, regarding the creation and editing of graphics, and was completed by a series of packages
- Now, the workhorse of dynamic documents with R is knitr that fixes a series of limitations of previous alternatives
  - has a lot of easily manipulated options
  - can also be programmed when further refinements are needed.

Outline

Knitr

In this presentation, we present a package for literate programming called knitr
The packages uses R code, but can be used to embed chunks from other languages

- python, Stata (after manipulation), C++

Several editors have specific functionalities for the package

- RStudio
- Lyx

Principle

- Your (unique) working file will be a my_file.Rnw document
  It can be edited with any text editor, but RStudio is particularly well adapted
- Apply knitr("my_file.Rnw") to
  - execute the chunks
  - create a my_file.tex file that contains the narratives and the results of the source code
- Compile the my_file.tex with your favorite Latex compiler

In practice

- With RStudio, you go directly from the .Rnw file to the pdf file the .tex file is produced silently
- With Lyx, you go directly from the .Lyx file to the pdf file
- Created figures are placed in a Figure folder

Outline

Installation

1. Install the knitr package:
   install.packages('knitr', dependencies = TRUE)
2. Make sure that you have a Latex distribution installed
3. Configuration of RStudio
   1. make sure that RStudio access the Latex distribution
   2. precise that knitr should be used (instead of Sweave) in compilation options
4. Configuration of Lyx
   1. select knitr as a module in the document options
   2. for windows users, make sure that the path to R is correctly specified

Quick use

- The function **stitch()** can be used to produce a .tex containing the source code and the results.
- When executed under RStudio, a .pdf is produced.
- This function thus offers a sophisticated alternative to **sink()**

Inline source code and chunks

- Results of computations can be included among the rest of the text (inline) with function **Sexpr{}**
  examples:
  the value of π is **Sexpr{round(pi,2)}** produces "the value of π is 3.14"
  the value of π is **Sexpr{round(pi,4)}** produces "the value of π is 3.1416"
- Longer scripts are placed in chunks, the syntax for a knitr chunk is
  << chunk_name, echo=T, eval=T, options>>=
  some R code
  @

Chunk names

- Chunk names
  - do not need to follow R syntax
  - are used to name figures
  - can be useful to identify pieces of code after tangling
  - can be used to re-use a previous chunk
- Chunk names you be UNIQUE otherwise the document does not compile
- If no name is given, a chunk name is automatically created with an incremental number

Chunk options: editing

- eval= TRUE or FALSE is the code evaluated (executed)
- echo=TRUE or FALSE is the (text) code reported in the final document
- message=TRUE or FALSE, should (error) messages be printed
- warning= TRUE or FALSE, should warnings be printed
- size='large','small','scriptsize', 'tiny', for the size of the output (code and results)

Refinements

- prompt=FALSE for removing the > sign in the result output
- highlight=TRUE, using code highlighting when code is printed
- tidy=TRUE or FALSE, organizing the R code
- strip.white=TRUE or FALSE whether to remove the white lines in the beginning or end of a source chunk in the output

Chunk options: results

The results option takes the following values

- 'markup'
- 'asis' the output should be treated as Latex code (needed when using xtable())
- 'hold' all the outputs are produced at the end of the chunk
- 'hide' the results are not displayed

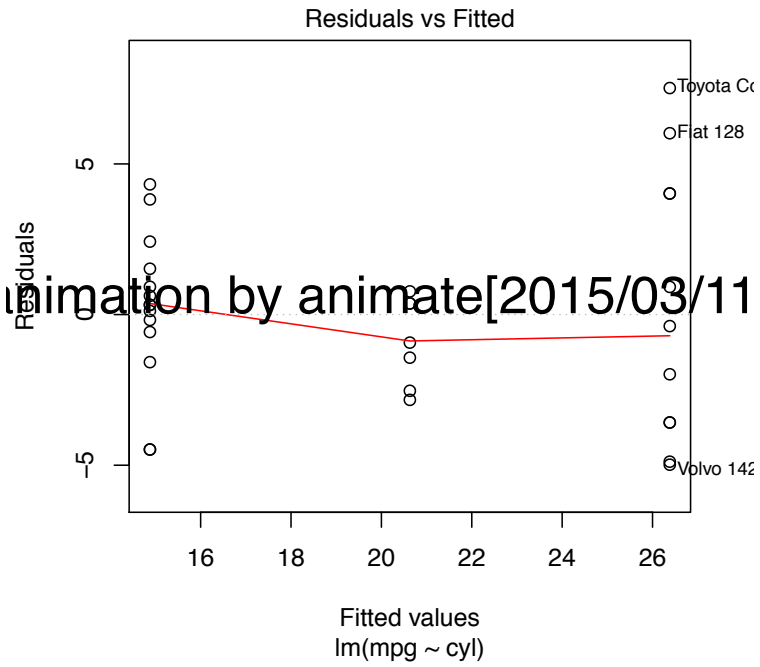
Chunk options: figures

One of the main advantage of knitr is the production of figures.

- fig.show takes values 'asis', 'hold', 'animate' for creating an animation with multiple plots, 'hide'
- dev refers to the graphic device (generally 'pdf', or 'tikz')
- fig.width and fig.height refer to the size of the produced figure
- out.width refers to the size of the output, e.g '.45\\linewidth'
- fig.align for latex position options
- fig.cap is the argument of the caption
- fig.pos latex position option

Illustration

```
data(mtcars)
mod=lm(mpg~cyl,mtcars)
plot(mod,ask=FALSE)
```



animation by animate[2015/03/11

Other features

- Options can be set globally, in a chunk at the beginning of the document
  e.g

```
opts_chunk$set(fig.path="Graphiques/beamer-", fig.align="center",size="footnotesize",fig.height=7,fig.wi
```

- a cache option allows to keep the output of chunks in memory so that they do not have to be re-executed at each compilation.

Comment

- Note that all chunks are executed in a single R session
  therefore, an object created in a chunk can be refereed to in a subsequent chunk
- However, each chunk re-initiates the working directory to the one containing the file
  you may have to re-specify the directory at the beginning of several chunks.

The end

Thank you.