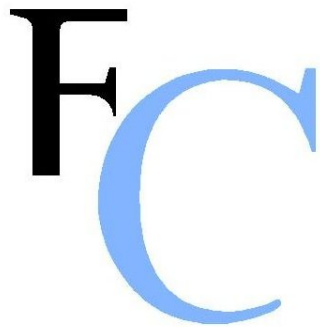
Dynamic Documents in R

L. Torgo

ltorgo@fc.up.pt

Departamento de Ciência de Computadores / Faculdade de Ciências
Universidade do Porto

Jun, 2014



Reporting

The Standard Process of Data Analysis

- Import the data into our favorite data analysis tool
- Carry out a set of data analysis steps
- Report the work by building some document (report and / or presentation)
  - Series of *copy+paste* steps from parts of the results of the analysis into some word processing and/or presentation software tool, adding some supporting text
- Frequently all process needs to be repeated / iterated!

Some of the Dangers of this Standard Approach

- Too many manual steps $\mapsto$ great potential for human error
- Too much human effort (time) in the process with many repetitive and boring tasks, like for instance the communication between different software tools
- All process is hardly recordable for future re-use, due to the extensive use of graphical user interfaces
- Small changes on the initial data require full repetition of all process!
- The Analysis and the Reporting are separated and thus great care is required to have both in "sync" avoiding reporting errors
- Very hard to share the work with other teams
- Very hard to re-use the work on similar tasks

List inspired by *Dynamic Documents with R and knitr* by Yihui Xie

Dynamic Documents

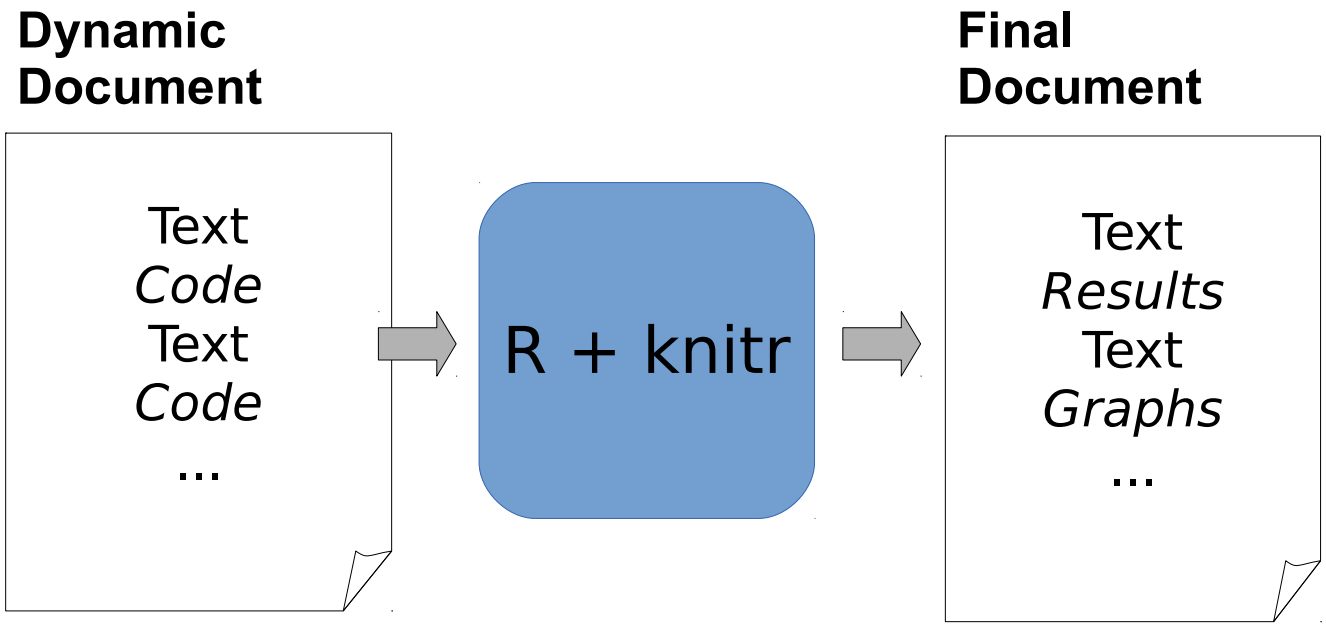
What are Dynamic Documents

- Documents that mix data analysis steps with descriptive text
- Documents that are executable by a computer program to produce the final document
- This final document is produced by a computer program from the initial document created by the user containing data analysis steps and descriptive text

Dynamic Documents solve most of the problems we have described before!

Dynamic Documents

R+knitr - an implementation of the idea



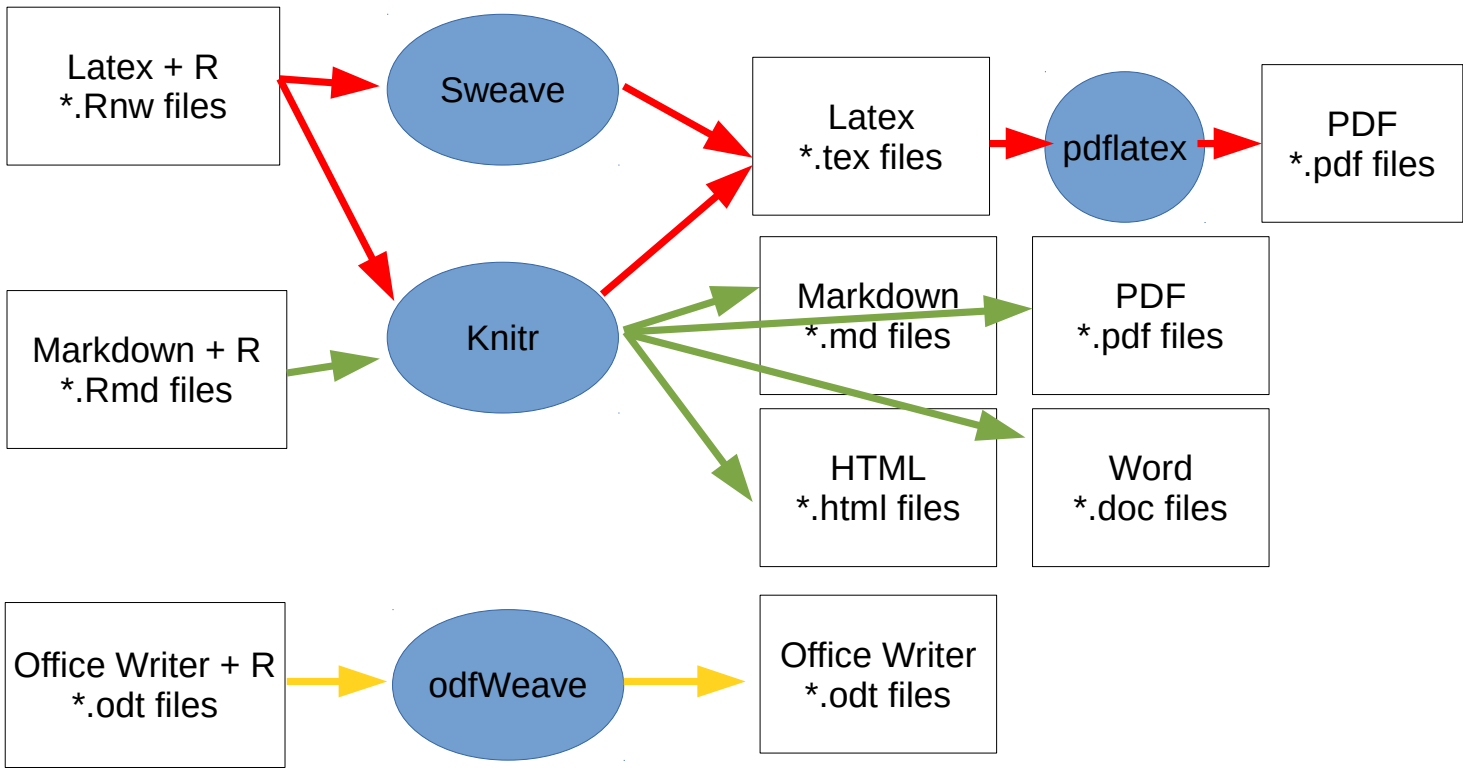
Dynamic Documents

- The idea is related with the concept of *literate programming*
- It is implemented in 3 main steps:
  1. Inspect the dynamic report and separate the data analysis code from the descriptive text
  2. Execute the code and store the results of each code chunk
  3. Produce the final document replacing the data analysis code in the original document by the results
- NOTE: all process is carried out without human intervention!

Knuth, Donald E. (1984). "Literate Programming". The Computer Journal (British Computer Society) 27 (2): 97–111.

Dynamic Documents

Different implementations of the concept



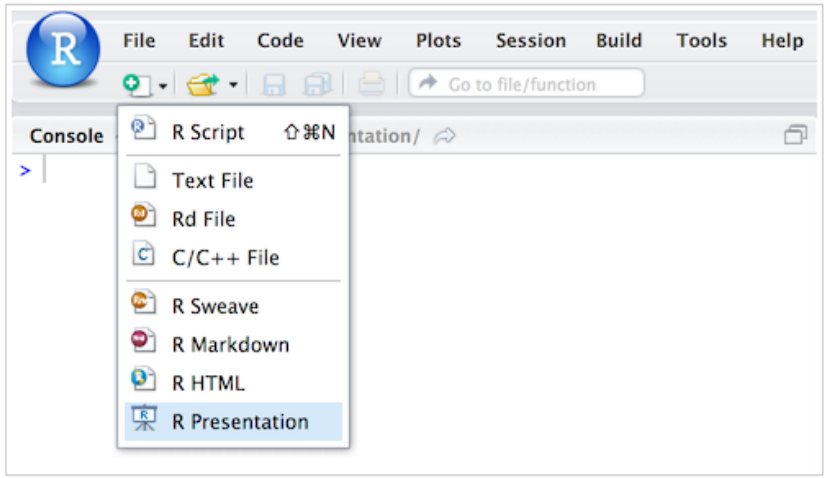
R+knitr

- Knitr is a platform that allows:
  - Using different document formats
    - LaTeX, HTML and Markdown
  - Using different programming languages to implement the data analysis steps
    - R, Python, awk, C++ and shell scripts

RStudio

Recent versions of RStudio facilitate the creation of dynamic reports and presentations using R and Markdown

- **R Markdown** - reports
- **R Presentation** - presentations



A Small Illustrative Example

A (Very) Brief Introduction to Markdown

- Very simple language for creating Internet content
  - Much simpler than the native language - HTML
- The format is a simple text file and you do not need any special software tool to create Markdown files
- In the end Markdown will be automatically translated into HTML such that the end result can be visualized on any modern Web browser

Character formatting in Markdown

- Formatting is carried out with the help of small annotation tags
- Examples:

  `**Note**` or `__Note__`

  translated into **Note** (i.e. boldface)

Sections and subsections

Illustrations on the use of Markdow in Dynamic Reports

Introduction

R comes with a series of data sets. We may use the function **data** to load them, as show in the following example:

- A text line where the following line contains three or more equal signs (=’s) is promoted to a section
- If instead you use dashes (−’s) you get a sub-section

```
Illustrations on the use of Markdow in Dynamic Reports
======================================================

Introduction
---------------

R comes with a series of data sets. We may use the function
**data** to load them, as show in the following example:
```

Lists of Items

```
OM prototype - Admin User
=========================

- Stage 1
  - Define topics of interest
  - Define web sources
      * create crawlers (**Potentially Challe
      * collect data
- Stage 2
  - Obtain tagged data (**Potentially Challe
  - Obtain, evaluate and select models
```

OM prototype - Admin User

- Stage 1
  - Define topics of interest
  - Define web sources
    - create crawlers (Potentially Challenging)
    - collect data
- Stage 2
  - Obtain tagged data (Potentially Challenging)
  - Obtain, evaluate and select models

Two columns, embed images and links

```
Monitoring and Forecasting Water Quality Parameters
===================================================

- Collaboration with
[Águas do Douro e Paiva,SA](http://addp.pt/pt/home.php)
- FCT project
[MORWAQ](http://liaad.inescporto.pt/modys/projects/morwaq)
- Some of the main results:
  - Software prototype for monitoring and forecasting
    water quality parameters
  - Several publications
     - KDD'2011, ECAI'2010
***
![ADdP Network](addpNet.png "The ADdP Network")
```

Monitoring and Forecasting Water Quality Parameters

- Collaboration with Águas do Douro e Paiva,SA
- FCT project MORWAQ
- Some of the main results:
  - Software prototype for monitoring and forecasting water quality parameters
  - Several publications
    - KDD'2011, ECAI'2010

Embed R code into documents - R Markdown

Knitr *code chunks*

````
The Distribution of Sepal Length
================================

An histogram of the variable
can be obtained with:
```{r eval=FALSE}
hist(iris$Sepal.Length)
```

***
```{r echo=FALSE}
data(iris)
hist(iris$Sepal.Length)
```
````

`Chunk+Insert Chunk` in RStudio

The Distribution of Sepal Length

An histogram of the variable can be obtained with:

```
hist(iris$Sepal.Length)
```

Code Chunks in R Markdown

- Code chunks are parts of a document that should be executed by R
- In R Markdown they are indicated as follows:

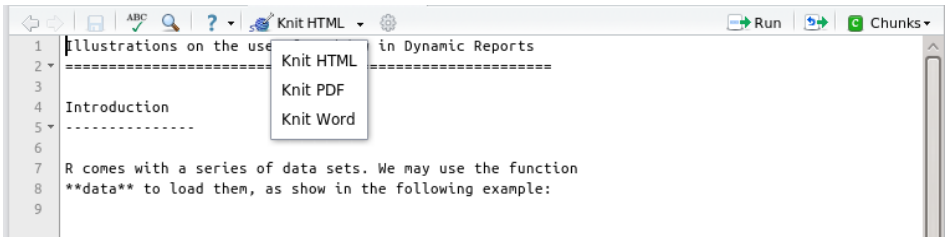
```
```{r}

```
```

- Everything between these two lines will be executed by R
- The result of the execution will be part of the final document
- Inline code can be used like this: `` `r 2+2` ``

Code Chunk Options

- The way code chunks are used by Knitr is controllable through a series of chunk options
- Without any option code chunks are:
  1. Executed
  2. A code block is added to the final document
  3. A results block is added to the final block
- Chunk options allow to adjust these defaults
- This can be done on each individual chunk or globally

Some common chunk options

eval (`TRUE`) - allows to control whether the chunk code is to be executed or not by R. In the following example the code is inserted into the final document but it will not be executed by R

````
```{r eval=FALSE}
hist(iris$Sepal.Length)
```
````

echo (`TRUE`) - allows to hide the R code from the final document (with the value `FALSE`), with only the results of the execution being included in that document (e.g. a figure)

- Note that each chunk may include several options separated by commas
- Many more options exist! See an exhaustive list with explanations at `http://yihui.name/knitr/options`

What if your favorite reporting tool is Microsoft Word?

- Difficult task as Microsoft uses closed formats
- Several alternatives exist
- Most are based on the `Pandoc` tool available at `http://johnmacfarlane.net/pandoc/`
- Pandoc is a generic tool for converting between several document formats

Installing Pandoc

- In Linux it is directly available in package repositories
- In Mac OS X it is also easy to install following the instructions at Pandoc web site
- In Windows it is slightly more complex, but there is an R package that facilitates the task (`installr`). This package includes a specific function for installing Pandoc under Windows
- Note that the following illustration assumes you have already installed package `installr` !

```
library(installr)
install.pandoc()
```

From R+Markdown to Microsoft Word

Method 1

- The report generation process consists of:
  - Write the report in R+Markdown, i.e. on a .Rmd file (for instance using RStudio)
  - Convert the file from Rmd into HTML using Knitr

```
library(knitr)
knit2html("myReport.Rmd")
```

  - Note that the previous step could also be accomplished using the specific button available in RStudio
  - Finally, use Pandoc to convert HTML into DOCX

```
fileName <- "myReport"
system(paste0("pandoc -o ",fileName,".docx ",myReport,".md"))
```

Some additional “tricks”

- The conversion process (from .md to .docx) has “some problems”...
- Some of them can be solved by adding the following code in the beginning of the R Markdown file of the report:

````
 ```{r set_knitr_chunk_options, echo=FALSE}
opts_chunk$set(echo=FALSE,message=FALSE,results = "asis")
```


```{r load_pander_methods}
require(pander)
replace.print.methods <- function(PKG_name = "pander") {
   PKG_methods <- as.character(methods(PKG_name))
   print_methods <- gsub(PKG_name, "print", PKG_methods)
   for(i in seq_along(PKG_methods)) {
      f <- eval(parse(text=paste(PKG_name,":::", PKG_methods[i], sep = "")
      assign(print_methods[i], f, ".GlobalEnv")
   }
}
replace.print.methods()
```
````

From R+Markdown to Microsoft Word

Method 2 - R Markdown (version 2)

- The most recent versions of RStudio greatly facilitate this task
- The R+Markdown report creation interface now includes several options as format for the final output report



- The usual HTML format
- PDF format
- Word format

- This means we can do everything (from R+Markdown to docx) with a single button click!

Small Illustrative Example