# Reporting using Dynamic Documents

L. Torgo

ltorgo@knoyda.com
KNOYDA, Know Your Data!

Jul, 2019



## 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

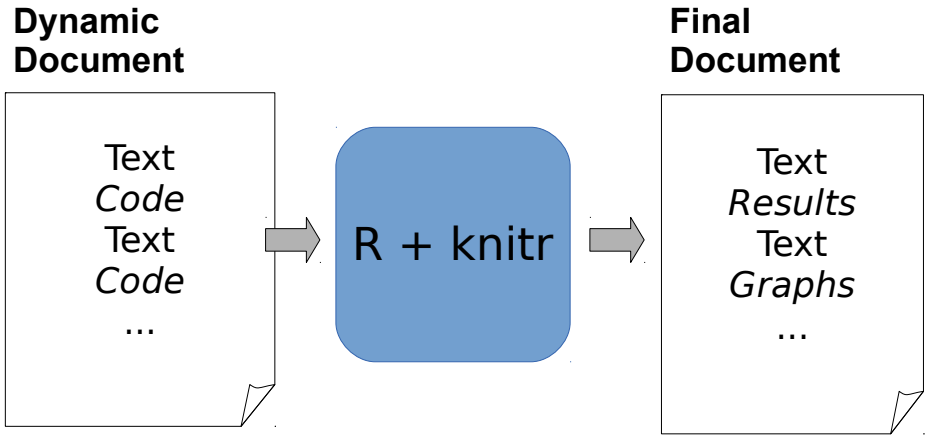
## What are Dynamic Documents

- 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







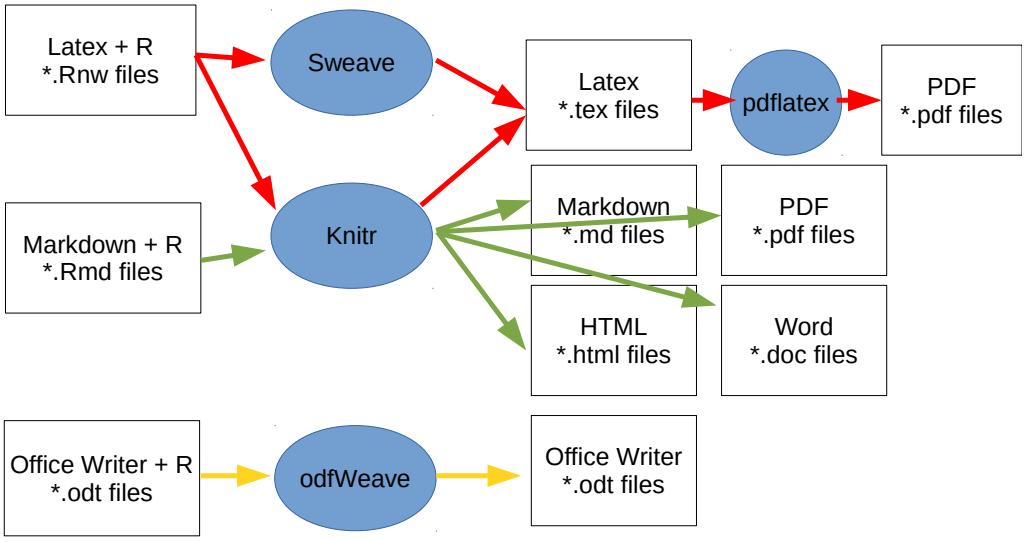

# Dynamic Documents

- 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








# Knitr

- Knitr is a platform that allows:
  - Using different dynamic document input formats
    - Typically either LaTeX+ R or Markdown + R
  - Produce output documents on different formats from the same input document
    - E.g. HTML, PDF, Word

On this short introduction we will focus on the Markdown + R combination for the input documents

# The rmarkdown package

- **rmarkdown** is an R package that allows integrating R with Markdown, using **knitr** and **pandoc** to generate the final document
- **Pandoc** is a kind of swiss-army knife of document format conversion
- **R markdown** is a literate programming approach that allows embedding R code into markdown documents
- **Markdown** is a very simple markup language that was designed to easily produce internet content





# A (Very) Brief Introduction to Markdown

- Very simple language for creating Internet content
  - Much simpler than the native language - HTML
  - rmarkdown uses one of the versions of this language, known as Pandoc's Markdown (full details at `https://pandoc.org/MANUAL.html`)
- The format is a simple text file and you do not need any special software tool to create or edit Markdown files

# Character formatting in Markdown

- Formatting is carried out with the help of small annotation tags
- Examples:
  - `**Note**` is translated into **Note** (i.e. boldface)
  - `*Note*` is translated into *Note* (i.e. italics)
  - `*x*^2^` is translated into $x^2$(i.e. superscript)
  - `*x*~2~` is translated into $x_2$(i.e. subscript)





# Sections and subsections

- An hash (#) character before a text line indicates a first level section heading
- If instead you use two or three hashes we get second and third level sections

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:

```
# 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





# 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)
```

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

# R Markdown

## Embedding R code into documents

````
# 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 interpreted 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)

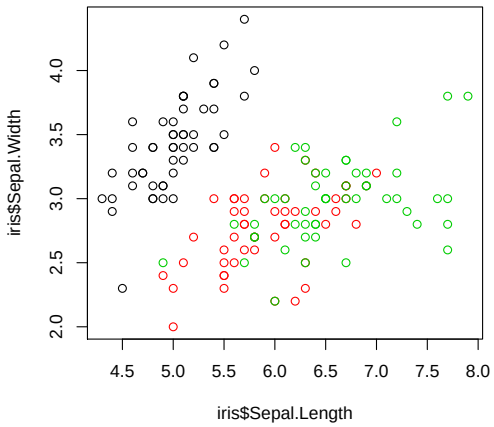
- Note that each chunk may include several options separated by commas
- Many more options exist! See an exhaustive list with explanations at `https://yihui.name/knitr/options`

# Code generating figures

- When your code produces a plot the respective figure is produced after the code chunk, e.g.

```
```{r}
plot(iris$Sepal.Length,iris$Sepal.Width,col=iris$Species)
```
```

```
plot(iris$Sepal.Length,iris$Sepal.Width,col=iris$Species)
```








# Code generating tables

- One of the most pratical solutions for tables is to use the function `kable`, like in the following example

```
```{r echo=FALSE}
knitr::kable(head(iris), caption = 'First rows of iris')
```
```

Table: First rows of iris

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |

# Generating the Final Document

- Function `render()` from package **rmarkdown** receives as input a markdown file and produces the final document using R to execute the code chunks:

```
library(rmarkdown)
render("initialDoc.Rmd")
```

- The type of output document that is generated can be controlled through meta-data information that is included in the begining of the document,

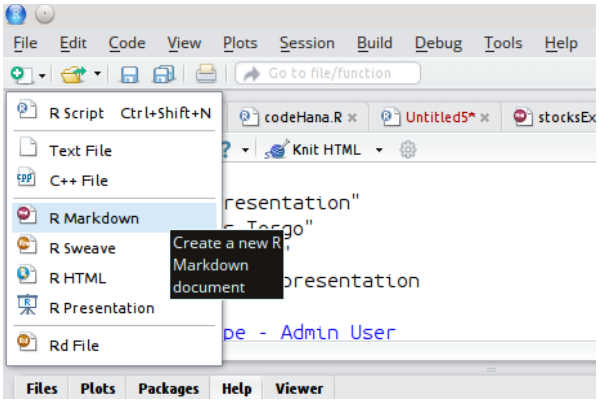
```
---
title: "My First Dynamic Report"
output:
  pdf_document:
    toc: true
    highlight: zenburn
---
```



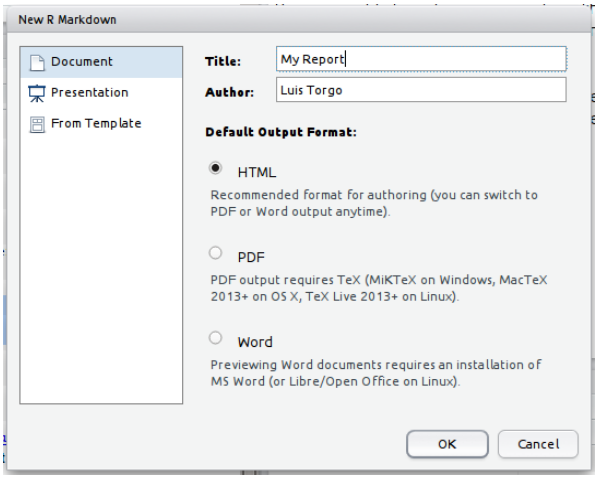


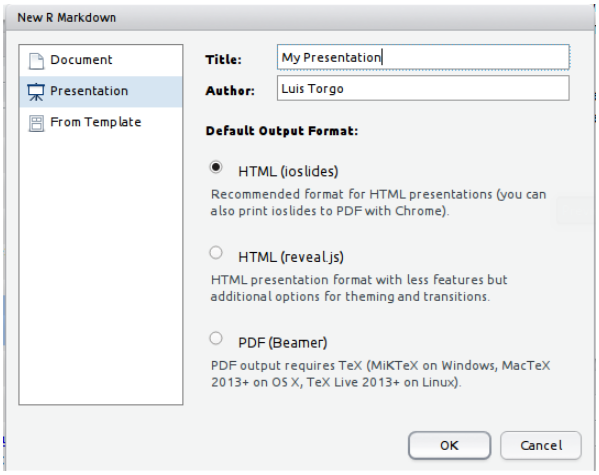
# Using the RStudio interface

RStudio facilitates the creation of dynamic reports and presentations using R Markdown



# Selecting the Type of Document Using RStudio interface

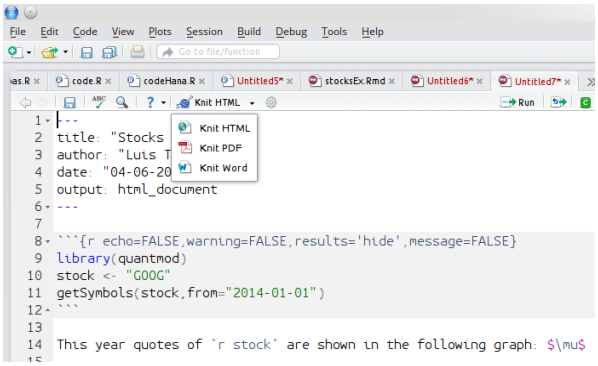


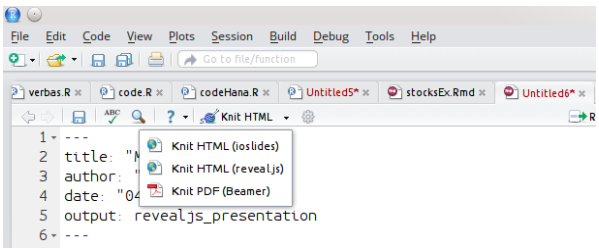








# Generating the Output Document Using RStudio interface



# Main Types of Output Documents

- Documents
  - HTML Documents
  - R Notebooks
  - PDF Documents
  - Word Documents
  - OpenDocument text documents
  - Rich Text documents
  - Markdown documents
  - R package vignettes
- Presentations
  - ioslides
  - Slidy
  - Beamer
  - PowerPoint





# Other Types of Outputs

- Dashboards
- Tufte Handouts
- xaringan presentations
- reveal.js presentations
- Websites
- HTML documentation for R packages
- Books
- Journal articles
- Interactive tutorials

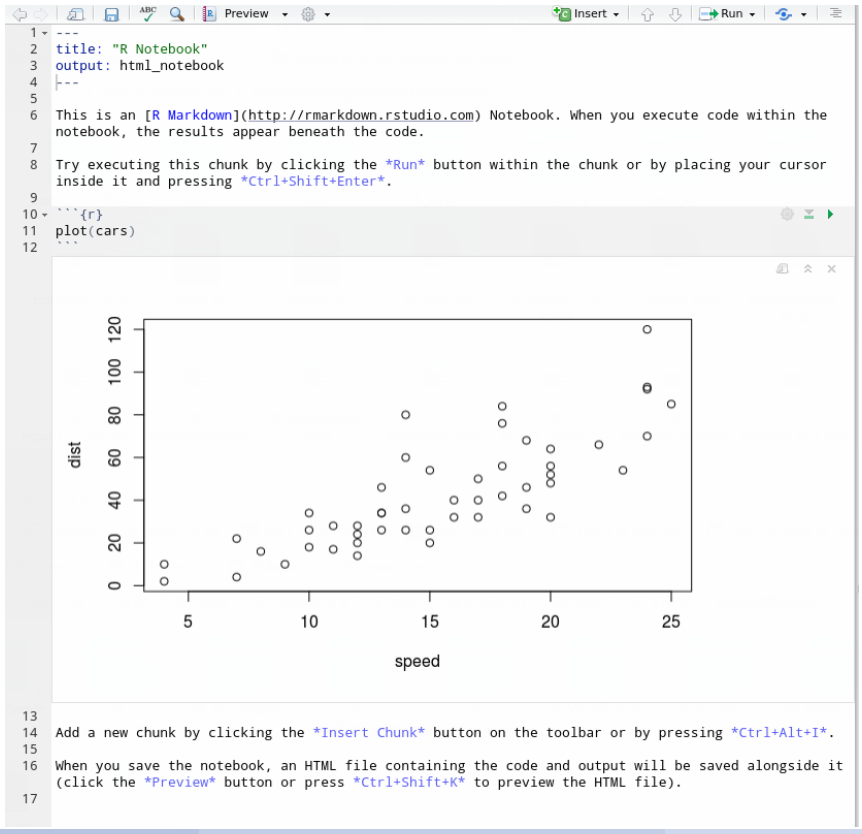
# R Notebooks

- R Notebooks are one type of R Markdown documents
- They are a type of R markdown document with `output: html_notebook` in the YALM heading
- Their main characteristics are that code chunks that can be executed independently and interactively, with output visible immediately below the chunk.
- R Notebooks are a nice way of interaction with R and at the same time documenting your activity. Recommended for class work!
- As any R markdown document, R Notebooks are also easily shareable with your collaborators





# An example of an R Notebook



# Creating R Notebooks

- Using RStudio menus: *File -> New File-> R Notebook*
- Using `output: html_notebook` in your YAML document header:

```
---
title: "Day 1 Classes Log"
output: html_notebook
---
```





# Saving and Sharing R Notebooks

- When you save the R Notebook two files are created:
  - The standard `.Rmd` (R markdown) file
  - An html (with extension `.nb.html`) that can be opened on any browser to visualize the notebook
- You may share your notebook through both files
- Actually, as the `.nb.html` file contains the `.Rmd` file, you may share only this file and when your colleagues open it in RStudio it will extract automatically the `Rmd` file and open it in the editor!

# RMarkdown with Parameters

- Sometimes we need to produce similar reports for different subsets of data
  - e.g. the same report for sales of a company on different countries
- The programatic nature of dynamic reports are ideal for these setups
- The ability of specifying parameters further increases flexibility
  - We can specify parameters of a dynamic report in the YALM header of the document through the `params` field





# An Illustrative Example

## Showing the quotes of a stock for the last X days

- Package **quantmod** provides many functions to deal financial markets data
  - function `getSymbols` can be used to download quotes data from Yahoo finance
  - function `candleChart` can be used to plot this data as a candle chart
- It is easy to write a simple report that downloads the quotes of a company and shows the respective candle chart
- What if we want to do that for different companies? Do we need to write different reports?! No!

# A Possible Solution with Parameters

```
---
title: "Stock Report"
author: "Luis Torgo"
output: html_document
params:
  ticker: MSFT
  nDays: 30
  showCode: FALSE
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = params$showCode)
```

## The Quotes of `r params$ticker` for the last `r params$nDays` days

```{r include=FALSE}
library(quantmod)
dat <- getSymbols(params$ticker,
                  from=Sys.Date()-params$nDays,
                  to=Sys.Date(),
                  auto.assign = FALSE)
```

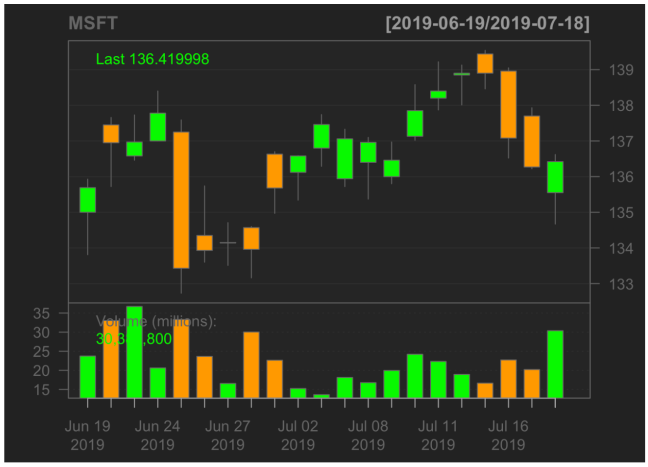

```{r quotes}
candleChart(dat,name=params$ticker)
```
```

Files | Plots | Packages | Help | Viewer — Publish

## Stock Report

Luis Torgo

### The Quotes of MSFT for the last 30 days








# Three ways of using the flexibility

- Change the Rmarkdown field and run the document in RStudio (not very interesting...)
- Call the function `rmarkdown::render()` specifying other values for the parameters you want to change the values in the Rmarkdown file

  `rmarkdown::render("report.Rmd',params=list(ticker="AAPL"))`
- Use an interactive interface at RStudio to set these values by producing the document using the option (in the menu) `Knit with Parameters...`

# Going a step further...

## Automating the report generation for a set of stocks

app3.R | app4.R | rmdParams.Rmd | R data sets | genAllReports.R

Source on Save | Run | Source

```
1  stocks <- c("MSFT","AAPL","TSLA","CRM",'BABA')
2
3  for(s in stocks) {
4    rmarkdown::render("rmdParams.rmd",
5                      params = list(ticker = s),
6                      output_file = paste0(s,".html"))
7  }
8
```





# More Information on R Markdown

Much more information is available at:

http://rmarkdown.rstudio.com/

Check also these cheat sheets:

https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf

Full information at the book

https://bookdown.org/yihui/rmarkdown/