Table of Contents

Use of R Markdown to generate commented notebooks with R code and figures

Markdown is simple language syntax for text formatting (so as it looks good and can be exported into pdf/html/Word or other formats). R Markdown combines R scripts/figures/outputs with Markdown language and offers a way to create annotated notebooks directly in RStudio.

How it works: in RStudio, open new R Markdown file (*File* > *New File* > *R Markdown...*). This will open new file (with *.Rmd extension) with template to fill and few advises how to prepare R Markdown file. After you finish the text, click on *Knit* button (either *Knit PDF*, *Knit HTML* or *Knit Word*) to create requested file format. Note that for successful knitting of pdf files, you need to install complete TeX installation.

Few online sources to learn basics of Markdown:

- R Markdown in RStudio - Dynamic documents for R
- Markdown basics
- R Markdown cheatsheet

Below is Markdown version (`random-walk-vectorized.Rmd`) of the commented script Vectorized version of random walk script. You can also download the knitted pdf file and Word file version.

````
---
title: "Vectorized version of random walk script"
author: "David Zelený"
date: "December 20, 2015"
output: html_document
---

The original function to generate random walk was based on the `for` loop
of steps - in each step, random value (either -1 or 1) was generated, and
added to the previous values (check [Creating a loop and function using
random walk
example](http://www.davidzeleny.net/wiki/doku.php/recol:exercise:random_wal
k)). The script of `random.walk` function looks like this:
```{r}
  random.walk <- function (no.steps)
  {
    steps.rw <- vector (mode = 'numeric', length = no.steps)
    for (k in seq (2, no.steps))
    {
      steps.rw [k] <- steps.rw [k-1] + sample (c(-1,1),1)
    }
    steps.rw
  }
```
In this case, we may _vectorize_ it - replace the loop by function(s) which
can take as arguments the whole vectors. The solution is to first generate
````

````
a vector of the length `no.steps`, which contains randomly selected values
-1 or 1, and then use this vector in function `cumsum`, which returns the
vector of cumulative sums - the vector of the same length as its argument,
where each element represents the cumulative sum of elements in the
argument.
Vectorized option is:
```{r}
  random.walk.vec <- function (no.steps)
  {
  steps.rw <- sample (c(-1,1), size = no.steps-1, replace = T)
  steps.rw <- c(0, steps.rw)
  steps.rw <- cumsum (steps.rw)
  return (steps.rw)
  }
```
We may visually check that both functions produce exactly the same output.
We will use `plot` function to draw the results of each random walk
procedure applied on 100,000 steps (takes a while). Note the use of
`set.seed` function - we need to set up the same random seed before running
each function, to make sure that the generator of random values will start
always from the same position:
```{r}
  par (mfrow = c(1,2))
  set.seed (123)
  plot (random.walk (100000), type = 'l')
  set.seed (123)
  plot (random.walk (100000), type = 'l')
```
And a comparison of speed using `microbenchmark` function from `library
(microbenchmark)` (note that here I decreased the number of steps from
100,000 to 10,000 to save time, since each function is repeated 100 times
to obtain average times):
```{r}
  library (microbenchmark)
  microbenchmark (random.walk (10000), random.walk.vec (10000))
```
Note that the vectorized version (the second row) is on average 226 times
faster than the function using `for` loop! To calculate this number, I took
the column `mean` in the table from `microbenchmark`: in average one run of
`random.walk (10000)` takes 49341 microseconds = 0.049 seconds, while
`random.walk.vec (10000)` takes only 218 microseconds = 0.0002 seconds,
49340/218 = 226. This value is changeable - depends on the speed of your
computer, and also on the number of steps the function has to do.
And because we like figures, we may draw the comparison also as a boxplot
using `boxplot.microbenchmark` function from the same library:
```{r}
  boxplot (microbenchmark (random.walk (10000), random.walk.vec (10000)))
```
````

Another option is to use R Markdown to create simple presentations. Below is a short R Markdown file

(draw-stairs.rmd) creating slides showing how to Draw stairs (to heaven). This R Markdown script can create either pdf presentation or can be published using HTML, e.g. again on RPubs.

````
---
title: "Draw stairs (to heaven)"
author: "David Zelený"
date: "December 20, 2015"
output: slidy_presentation
---

## Aim: Draw this figure
```{r, echo = FALSE}
plot (c(1:10), type = 'S', axes = F, ann = F)
```

##
### But:
> - only function `plot` is allowed
> - the script must have only **one line**, no more
> - this line must draw the complete figure

##

First, prepare the data:

```{r}
plot (1:10)
```

##
Switch off all the gingles around (`axes` and `ann`otations):
```{r}
plot (1:10, axes = FALSE, ann = FALSE)
```

##
And finally the trick - use argument `type = "S"` to connect dots with
stairs:
```{r}
plot (1:10, type = "S", axes = FALSE, ann = FALSE)
```

##
### Thank you for your attention!
````