RStudio - 02: Project, markdown, R markdown

Jean-Yves Sgro

May 2, 2017

Contents

1 Goal: set-up an RStudio project to analyse enzymatic results

1.1 Create a new project

An **RStudio** project is a set-up that uses a specific directory to contain the data that we want to use for analysis.

To set-up a new project follow do the following in **RStudio**:

1. Click the `File` menu button, then `New Project`.
2. Click `New Directory`.
3. Click `Empty Project`.
4. Type in the name of the directory to store your project, *e.g.* `Project_1`.
5. For now don't check "Create a git repository" (should be unselected by default)
6. Click the `Create Project` button.

The project will be saved in the default directory: `/Users/YourName` on the Mac unless you navigate to *e.g.* the `Desktop` to more easily find the it later (strongly suggested.)

1.2 Creating reports for "reproducible research"

Goal: to analyse data in reproducible way.

Analyzing data in a reproducible manner is easier to achieve if there is a proper documentation of the process. In "old days" this was achieved by adding comments within computer programs. Nowdays it is possible to "weave" the **computer code** (the `R` commands in our case) with a **narrative** of the process (the story we need to report) in a manner where they are both updated automatically if the analysis method should change or if the data is altered or updated with new data.

This allows the "dynamic" creation of output report documents in *e.g.* PDF, HTML or MSWord format. The output document can be overwritten or replaced when data or methods are updated.

2 What are markdown ar R Markdown?

> *"Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML)."* John Gruber inventor of markdown.
>
> *R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of markdown (an easy to write plain text format) with embedded R code chunks that are run so their output can be included in the final document. R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).*

We first need to learn just a little bit of `R` **Markdown** for this purpose. Then the `knitr` package will compile the file we create with a "story" and `R` code into one final, dynamically created document.

> *Note*: this document was created with `R` **Markdown**!

3 Basic markdown syntax

Markdown is very simple and helps create structured text easily. As detailed above it is originally a method for creating HTML (mark-up language) and therefore there are similarities. For those that are more into Microsoft Word, the same type of structure easily applies as well with the "Styles" for titles and paragraphs.

There are a few variations on how to accomplish some effects, but we'll limit the examples to just the basic necessity.

3.1 Headings:

The following code is one way to create `Headings` levels:

```
# Heading 1
## Heading 2
### Heading 3
#### Heading 4
```

After processing they become title headings in the final document:

Heading 1

Heading 2

Heading 3

Heading 4

3.2 Normal text:

Normal text is simply typed and paragraphs are created with an empty line in between.

For example the following text (placed here in a mode that does not process)

```
This is some text that
will become one line when processed.

A blank line will make a new
paragraph to continue the story.
```

will be processed as follows (which resembles the writing of this document as it was made in this manner!)

This is some text that will become one line when processed.

A blank line will make a new paragraph to continue the story.

3.3 Bold and italic

3.3.1 Italics

Italics are accomplished by * or _ before *and* after the word:

```
This word is *italic* and this one is _italic_ too.
```

This word is *italic* and this one is *italic* too.

3.3.2 Bold

Bold is accomplished by ** (2 stars *) or __ (2 underscores _) before ***and*** after the word:

```
This word is **bold** and this one is __bold__ too.
```

This word is **bold** and this one is **bold** too.

3.3.3 Both italics and bold

To get both we can combine for a total of 3 markers before ***and*** after the word or phrase:

```
This word is  ***both*** and this one is _**both**_ too.
```

This word is ***both*** and this one is ***both*** too.

Table 1: Summary Table

	Italic	**Bold**	***Both***
Using *	`*italic*` *italic*	`**bold**` **bold**	`***both***` ***both***
Using _	`_italic_` *italic*	`__bold__` **bold**	`___both___` ***both***
Combining * and _			`_**both**_` ***both***

3.4 Quotes

Quotes are easily created with the > sign, for example:

```
> This is quoted text, simply by using a single `>` at the beginning of the line.
> Syntax highlight may or may not work: **bold** _italic_ ***both***
>
> > *This is a quote within a quote marked as italics.*
```

Becomes:

> This is quoted text, simply by using a single `>` at the beginning of the line. Syntax highlight may or may not work: **bold** *italic* ***both***
>
> > *This is a quote within a quote marked as italics.*

3.5 Lists

Bullet list start with `*`:

```
* first item
* second item
* third item
```

becomes:

- first item
- second item
- third item

Ordered lists start with numbers:

```
1. first item
2. second item
3. third item
```

But in fact the manual numbering does not matter, this also works!

```
1. first item
1. second item
1. third item
```

Both become:

1. first item
2. second item
3. third item

3.6 Tables

Basic Markdown tables are created by lining up the columns and making headers, like so:

```
 Column 1    Column 2    Column 3
 --------    --------    --------
   1           10          100
   2           20          200
   3           30          300

 Table: Simple table with 3 columns
```

Table 2: A simple table

Column 1	Column 2	Column 3
1	10	100
2	20	200
3	30	300

Another method allows justification to the right or left with the `:` position within the underlining header (second line.) Here column 1 is justified center, column 2 to the right and column 3 to the left as indicated by the position and number of `:` in the underline row.

```
| Column 1   | Column 2  | Column 3  |
| :--------: | --------: | :-------- |
|  1         |   10      |   100     |
|  2         |   20      |   200     |
|  3         |   30      |   300     |
```

Table 3: Justified table

Column 1	Column 2	Column 3
1	10	100
2	20	200
3	30	300

Note: In a previous document we used function `kable()` from package `knitr` to format existing `R` matrices or data frames.

3.7 Code blocks and inline code

3.7.1 Inline code

Quoting simple code in `monospaced font` is accomplished within the text with a the single backtick ‘ around the word. For example:

```
This `code` will be seen as a `special word` within the text when interpreted.
```

Will become: This `code` will be seen as a `special word` within the text when interpreted.

3.7.2 Code block

A code block is a series of lines that are all code. For example here is a code block in `R`:

````
```
# This code block starts with 3 backticks and will end the same way.
x <- rnorm(10)
y <- seq(1:10)
plot(x, y)
```
````

When processed this will become:

```
# This code block starts with 3 backticks and will end the same way.
x <- rnorm(10)
```

```
y <- seq(1:10)
plot(x, y)
```

This code is *static* and ***not*** *dynamic*: it is only *formatted* to look like code.

3.8 Dynamic Code block

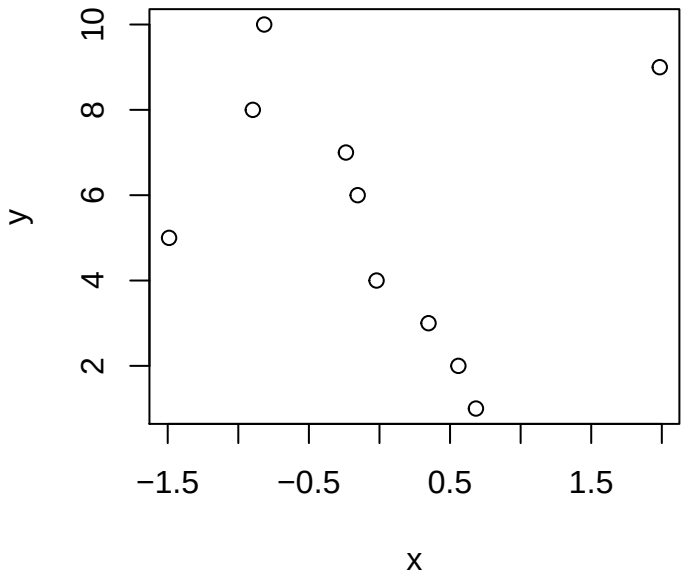
In `R` Markdown we mark `R` code with `{r}` so that when we finalize the document this code **will be run and the data read and updated**, etc. making the final document *dynamically* rendered and updated.

The code `chunck` can be named wihtin the brackets to describe the content, for example: `{r plot_test}`

Therefore, to have this code ***executed*** it needs to be written as:

Then within the document the code will be automatically syntax-highlighted and the code executed: in this case it creates a plot, itself embedded wihtin the final document.

```r
# This code block starts with 3 backticks and will end the same way.
x <- rnorm(10)
y <- seq(1:10)
plot(x, y)
```



> Note: there are special command that can be added to the `{r}` description to prevent the code from running, prevent the output from being printed or to prevent the code from being displayed if you don't want to reveal it. We will detail these later if and when the necessity comes along.

3.9 Dynamic Inline code

Code can also be included within the text and **computed on the fly** by `R`. This is useful when writing reports with specific values calculated or taken out of model evaluations.

This is marked by the letter `r` within the code to signify that `R` should compute it:

Simple example:

Will be processed as:

I think that this is an intersting calculation: `2 + 2 =` **4** when `R` evalulates the value within the specially marked code.

To really experience the fact that this is calculated **on the fly** we can add a random number to `2` (note that the result will be different each time as this is random! The number can be positive or negative.)

Let's add a random number to `2`: `2 + rnorm(1) =` **2.4336608**.

Note: a special `R` command `set.seed()` can be used so that results don't change each time even for "random" number generation.

Here is a fancier example: we'll set the seed to a user-defined value so that results are the same every time; then we assign the randomly generated number to object `x`; then we print `x` and add it to `2` all the while explaingin what we do in the text!

(*Note*: In the above code there is a space between ‘ and `r` to prevent calculations to occur... The embedded code within that gives the result below was properly formatted to obtain the correct output.)

When processed this will become:

Let's create a random number fixed with every iteration and assgin it to the variable `x`.

We can print the value of `x` which is **0.0187462**.

We can add `x` it to `2`: x + 2 = **2.0187462**

3.10 HTML links:

Any link that starts with `http://` or `https://` will automaticall become a clickable link.

For example: http://wisc.edu

To create a hypertext link: place the text in square brackets `[]` followed by the link in parentheses `()` **without** any space in betweem. For example:

```
[Biochemistry Department](http://biochem.wisc.edu)
```

Will be processed as a live link: Biochemistry Department

4 Markdown and dynamic analysis resources:

4.1 R markdown

RStudio Cheat Sheet (PDF)

Knitr in a knutshell a minimal tutorial by UW Prof Karl Broman

Minimal examples by Yihui Xie the author of `knitr`

Knitr reference card (PDF)

knitr example with built-in plot animations.

4.2 Markdown

Daring Fireball by John Gruber

Online Markdown editors: (just markdown, NOT R Markdown!)

https://stackedit.io/

http://dillinger.io/

4.2.1 Markdown editors:

- The Best Markdown Editor for Windows (Archived April 7, 2016: http://bit.ly/2p7ZYKg)
- Best Markdown Editors for OS X (Archived Februray 12, 2017: http://bit.ly/2pqFgWr)
- 10 Best Markdown Editors for Linux (Archive January 10, 2017: http://bit.ly/2pDCvSv)