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.

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. Nowadays 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 ([https://en.wikipedia.org/wiki/Markdown](https://en.wikipedia.org/wiki/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).”
**R Markdown** (http://rmarkdown.rstudio.com/) is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of markdown (https://daringfireball.net/projects/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 is 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:

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

After processing they become:
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:

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

**Bold** is accomplished by ** or __ 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.

*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.

To get both we can combine:
This word is ***both*** and this one is **both** too.

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

Summary Table

<table>
<thead>
<tr>
<th></th>
<th>Italic</th>
<th>Bold</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using *</td>
<td><em>italic</em> italic <strong>bold</strong> bold <em><strong>both</strong></em> both</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Using _</td>
<td><em>italic</em> italic <em>bold</em> bold <em><strong>both</strong></em> both</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combining * and _</td>
<td><em><strong>both</strong></em> both</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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 *:

**RStudio - 02: Project, markdown, R markdown**
* 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:

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>300</td>
</tr>
</tbody>
</table>

Table: Simple table with 3 columns
A simple table

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>300</td>
</tr>
</tbody>
</table>

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.

Justified table

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10 100</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20 200</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30 300</td>
<td></td>
</tr>
</tbody>
</table>

### 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 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. rendering the final document dynamically rendered.

The code chunk can be named within the brackets to describe the content, for example:

```
{r plot_test}
```

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

```
```{r plot_test}
  # 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 commands 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:

```
I think that this is an interesting calculation: `2 + 2 =` **`r 2 + 2`**
```
I think that this is an interesting calculation: \(2 + 2 = 4\) when \(R\) evaluates the value within the specially marked code.

### 3.10 HTML links:

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

For example: [http://wisc.edu](http://wisc.edu)

To create a hypertext link: place the text in square brackets `[]` followed by the link in parentheses `()` \textbf{without} any space in between. For example:

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

Will be processed as: Biochemistry Department (http://biochem.wisc.edu)

### 4 Markdown and dynamic analysis resources:

#### 4.1 R markdown


Knitr in a nutshell ([http://kbroman.org/knitr_knutshell/](http://kbroman.org/knitr_knutshell/)) a minimal tutorial by UW Prof Karl Broman ([http://kbroman.org/pages/about.html](http://kbroman.org/pages/about.html))


Knitr reference card ([https://cran.r-project.org/web/packages/knitr/vignettes/knitr-refcard.pdf](https://cran.r-project.org/web/packages/knitr/vignettes/knitr-refcard.pdf)) (PDF)

knitr example ([http://galahad.well.ox.ac.uk/repro/](http://galahad.well.ox.ac.uk/repro/)) with built-in plot animations.

#### 4.2 Markdown

Daring Fireball ([http://daringfireball.net/projects/markdown/](http://daringfireball.net/projects/markdown/)) by John Gruber

Online Markdown editors: (just markdown, NOT R Markdown!)
https://stackedit.io/ (https://stackedit.io/)
http://dillinger.io/ (http://dillinger.io/)