

# Introduzione a R Markdown

Create amazing documents with R

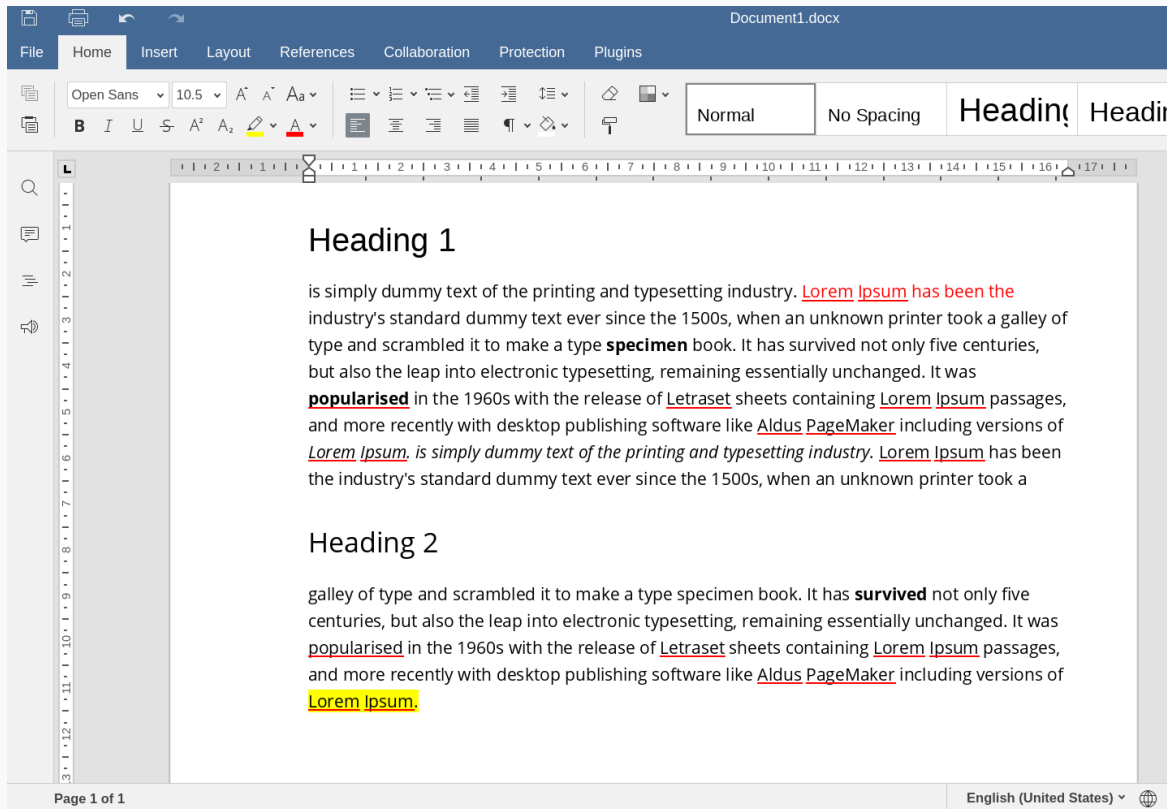
---

Filippo Gambarota  
Università di Padova

# Making documents...

Se dovete scrivere qualcosa (un report, una tesi, un documento generico) cosa vi viene in mente di utilizzare?

Probabilmente **Microsoft Word** oppure **Google Docs**. Sono ottimi software, molto *intuitivi*, *facili da imparare e estramamente popolari*.



# The pain joy of making documents

Sicuramente però vi sarà capitato di dover fare operazioni complesse come:

- gestire documenti con molte pagine
- inserire (o modificare 😱) figure e tabelle
- gestire bibliografia
- numerare paragrafi, tabelle e immagini

Me: Moves one picture  
on word slightly

Microsoft Word:



The entire Microsoft word  
document when you slightly  
move an image by 1 mm



# Why?

I programmi come Microsoft Word e affini sono definiti WYSIWYG (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et) perchè quello che vediamo mentre scriviamo è esattamente il risultato finale:

- se vogliamo mettere **grassetto** usiamo `ctrl + b` oppure clicchiamo un pulsante e vediamo subito il risultato
- se vogliamo inserire un'immagine trasciniamo il file e la spostiamo millimetro per millimetro manualmente

Questo ha il vantaggio di essere molto intuitivo e semplice, ma ci sono diversi svantaggi:

- non abbiamo (quasi) mai una **visione d'insieme del documento** (sposto l'immagine e non so cosa succede)
- pensiamo **contemporaneamente** al testo e alla formattazione
- versioni di Word (o equivalenti) diverse possono creare **problemi di compatibilità**
- quando il documento diventa pesante (> 30 pagine) ci possono essere **problemi di performance e formattazione**

# Ok.. qualche alternativa?

L'approccio alternativo è caratterizzato da **separare la formattazione e impaginazione (tedioso, complicato e superfluo) dal contenuto effettivo** in termini di testo.

Questo è possibile utilizzando **linguaggi di markup** ovvero un modo di scrivere del testo che viene **intepretato e compilato** e permette di produrre un certo tipo di risultato. Ad esempio:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

```
\documentclass[12pt]{article}
\usepackage{lingmacros}
\usepackage{tree-dvips}
\begin{document}
```

```
\section*{Notes for My Paper}
```

Don't forget to include examples of topi  
They look like this:

```
\subsection*{How to handle topicalizatio
```

I'll just assume a tree structure like (

# HTML

Questo è un esempio per scrivere un testo in html:

```
<!DOCTYPE html>
<html>
<body>
<h1>Heading 1</h1>
```

Questa è la prova di un documento in `<strong> HTML </strong>`. Come vedete assieme al testo ci sono degli elementi ( `<i> TAG </i>`) che vengono interpretati e non inseriti come testo.

```
<h2>Heading 2</h2>
```

Effettivamente è più complesso di word, ma come vedete non devo manualmente preoccuparmi della formattazione finale. Ci sono diversi vantaggi e svantaggi quindi:

```
<ul>
  <li>penso solo a scrivere</li>
  <li>poche soprore sulla formattazione</li>
  <li>però i tag non sono intuitivi e facili da scrivere</li>
</ul>
```

```
</body>
</html>
```

## Heading 1

Questa è la prova di un documento in **HTML**. Come vedete assieme al testo ci sono degli elementi ( *TAG* ) che vengono interpretati e non inseriti come testo.

## Heading 2


Effettivamente è più complesso di word, ma come vedete non devo manualmente preoccuparmi della formattazione finale. Ci sono diversi vantaggi e svantaggi quindi:

- penso solo a scrivere
- poche soprore sulla formattazione
- però i tag non sono intuitivi e facili da scrivere

# Latex

Questo è lo stesso esempio ma in Latex:

```
2 \usepackage[utf8]{inputenc}
3
4 \title{Latex}
5 \author{Filippo Gambarota}
6 \date{February 2022}
7
8 \begin{document}
9
10 \maketitle
11
12 \section{Introduction}
13
14 Anche con latex si possono \textbf{scrivere} documenti complessi
15 mescolando \textit{codice e testo}. Il principio è lo stesso
16 dell'HTML e come vedete il risultato è di altissima qualità.
17
18 \subsection{Sezione 2}
19
20 Facciamo anche qui pregi e difetti:
21
22 \begin{itemize}
23   \item più leggibile di HTML ma comunque meno intuitivo di word
24   \item i documenti sono di altissima qualità
25   \item formule matematiche, immagini e bibliografia sono gestite
26   in modo ottimo (al contrario di word)
27 \end{itemize}
28
29 \end{document}
```



Latex

Filippo Gambarota

February 2022

## 1 Introduction

Anche con latex si possono **scrivere** documenti complessi mescolando *codice e testo*. Il principio è lo stesso dell'HTML e come vedete il risultato è di altissima qualità.

### 1.1 Sezione 2

Facciamo anche qui pregi e difetti:

- più leggibile di HTML ma comunque meno intuitivo di word
- i documenti sono di altissima qualità
- formule matematiche, immagini e bibliografia sono gestite in modo ottimo (al contrario di word)

# Ma cosa centra tutto questo con R?

Per documenti semplici effettivamente non è necessario imparare un linguaggio come `HTML` o `Latex`. Pensate però ad una tesi di laurea, ad un report di analisi o un documento scientifico in generale dove:

- **inserire statistiche** con un certo stile di formattazione
- **modificare** diversi numeri se ci sono cambiamenti nelle analisi
- **aggiornare** figure e tabelle se vengono modificate

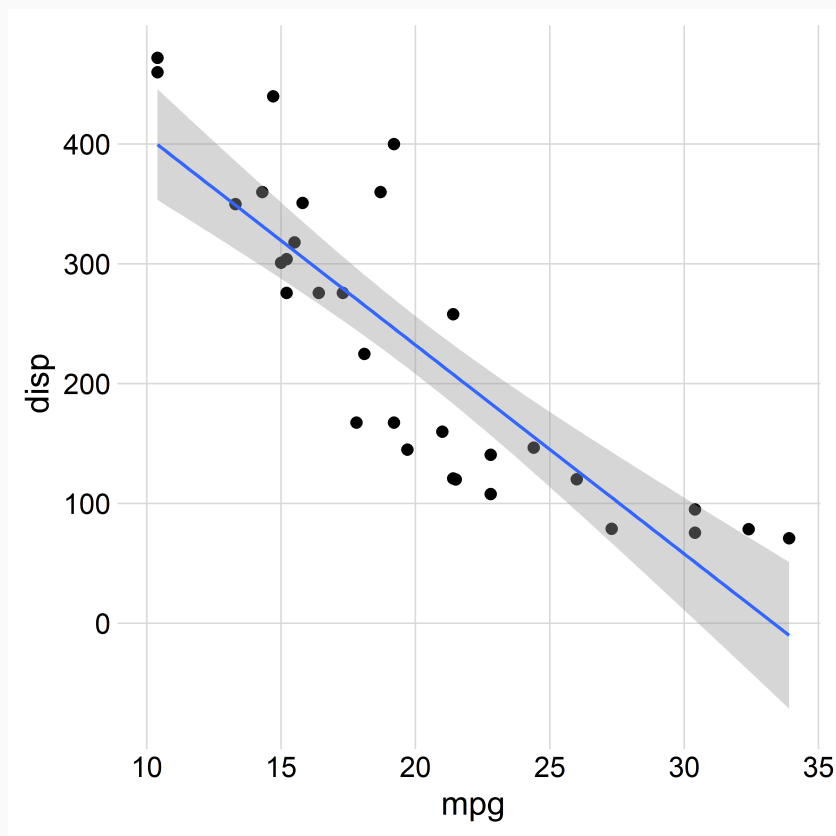
Sarebbe fantastico poter unire codice (i.e., la creazione di figure/tabelle) con il testo in un unico documento!

## Questo in programmazione si chiama **Literate Programming**



# Un esempio?

Immaginate di scrivere un report per un'analisi che avete fatto in R e volete inserire questo grafico:



# In word...

In word dovrete:

- creare il documento e scrivere tutta la parte di testo
- inserire l'immagine da un file esterno
- riposizionare e ridimensionare l'immagine, scrivere la caption

Cosa succede se l'immagine cambia?

Dovete manualmente eliminare l'immagine precedente e inserire la nuova immagine. E cosa succede se 10 grafici che avete inserito sono da cambiare? (😱)



# Un esempio?

Nel **literate programming** invece l'idea è che un certo elemento (ad esempio un grafico) viene creato con un pezzo di codice che verrà interpretato.

Questo è il codice per produrre il grafico:

```
mtcars %>%  
  ggplot(aes(x = mpg, y = disp)) +  
  geom_point(size = 3) +  
  geom_smooth(method = "lm") +  
  cowplot::theme_minimal_grid(font_size = 20)
```

# Un esempio?

... testo

```
mtcars %>%  
ggplot(aes(x = mpg, y = disp)) +  
geom_point(size = 3) +  
geom_smooth(method = "lm") +  
cowplot::theme_minimal_grid(font_size = 20)
```

... ancora testo

... testo

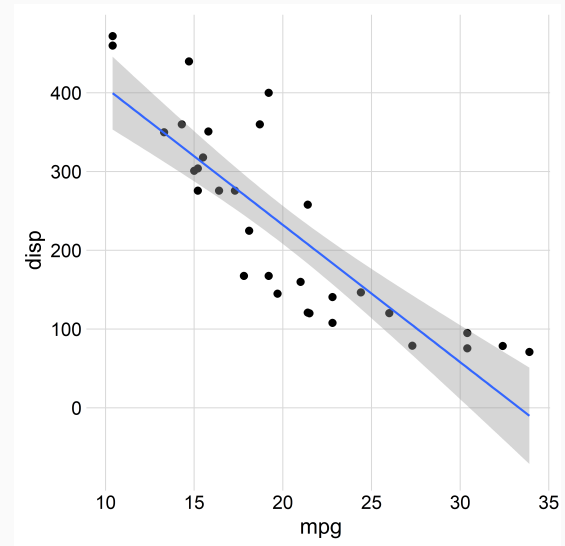


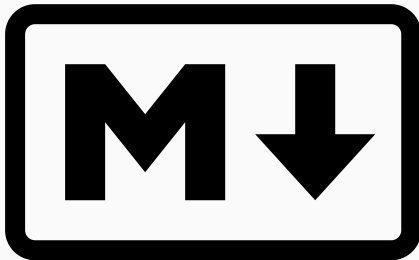
Figure 1. Awesome figure!

... ancora testo

# Literate programming (LP) in R?

Come abbiamo visto, per utilizzare LP abbiamo bisogno di un linguaggio di **markup** (HTML, Latex, etc.) e ovviamente di un linguaggio di programmazione.

Tra tutti i linguaggi di markup, uno in particolare è emerso recentemente per semplicità, facilità di lettura e si apprende in circa 30 minuti: Il linguaggio **Markdown**.



Vediamo un esempio!

<https://dillinger.io/>

(BTW queste stesse slide sono scritte in Markdown! 😊)

# R Markdown

R Markdown è la fusione dei linguaggi Markdown e R per poter creare documenti, slide, siti web, curriculum, tesi, articoli scientifici `combinando codice e testo`.

```
---  
title: "R Markdown Example"  
author: "Filippo Gambarota"  
date: "2/13/2022"  
output: html_document  
---
```

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

## ## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
```${r cars}  
summary(cars)  
```
```

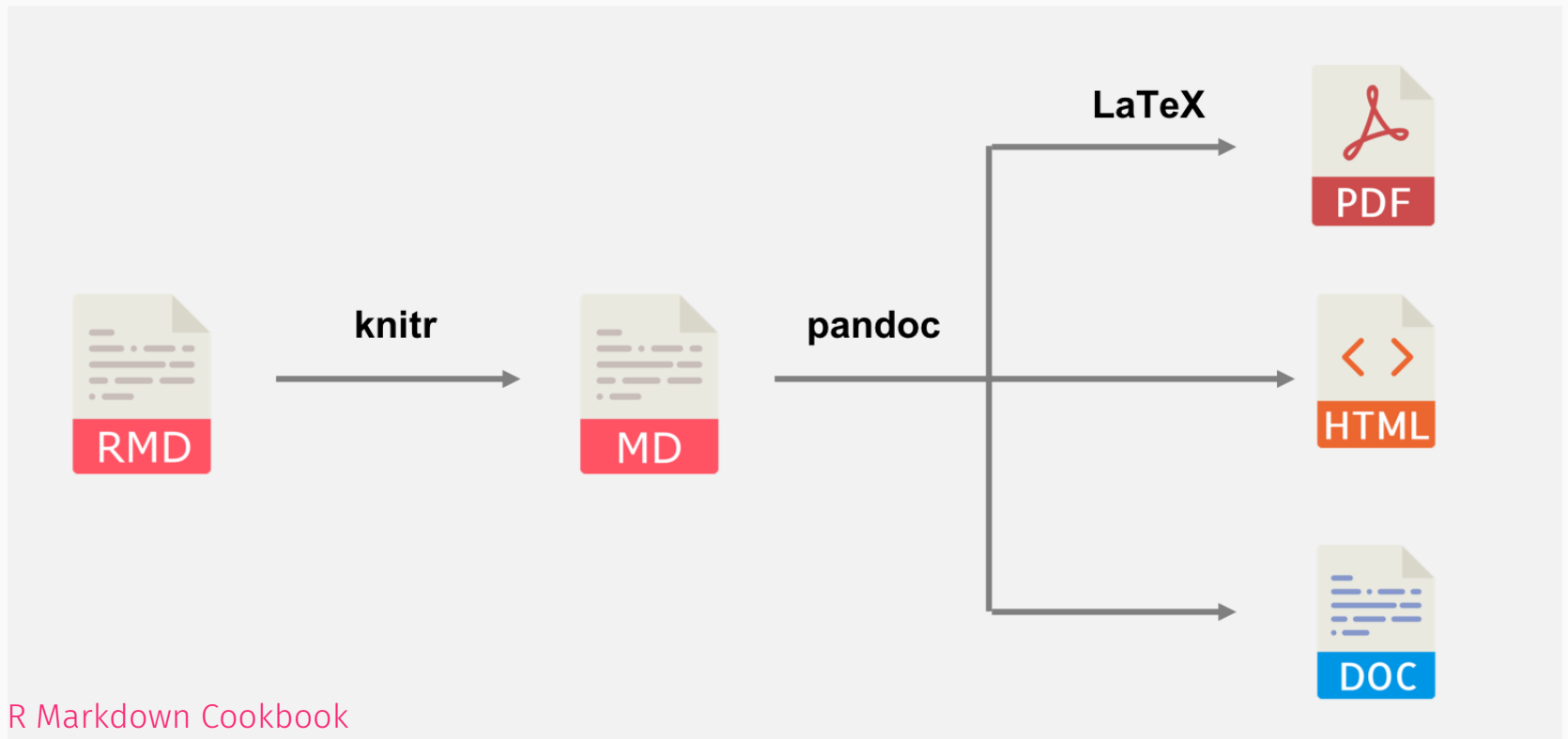
## ## Including Plots

You can also embed plots, for example:

```
```${r pressure, echo=FALSE}  
plot(pressure)  
```
```

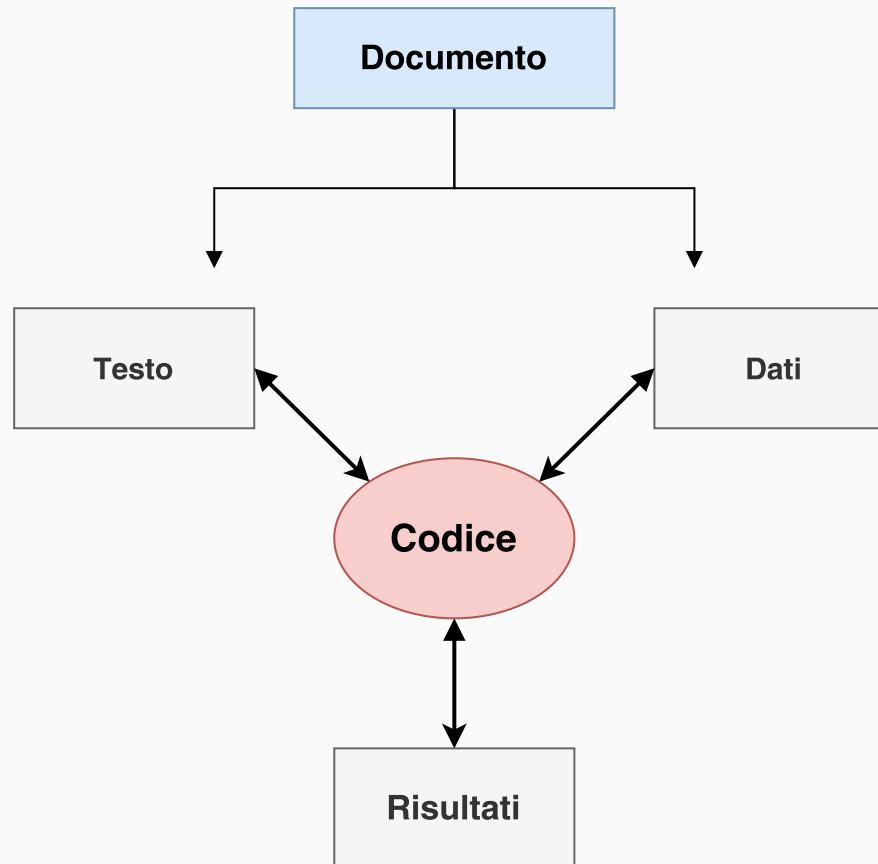
# Perchè R Markdown?

Scrivendo in `HTML` possiamo principalmente scrivere documenti `HTML` (visualizzabili con un browser web). Scrivendo in Latex possiamo creare principalmente documenti `PDF`. Il linguaggio Markdown può essere usato per produrre qualsiasi tipo di documento<sup>1</sup>:



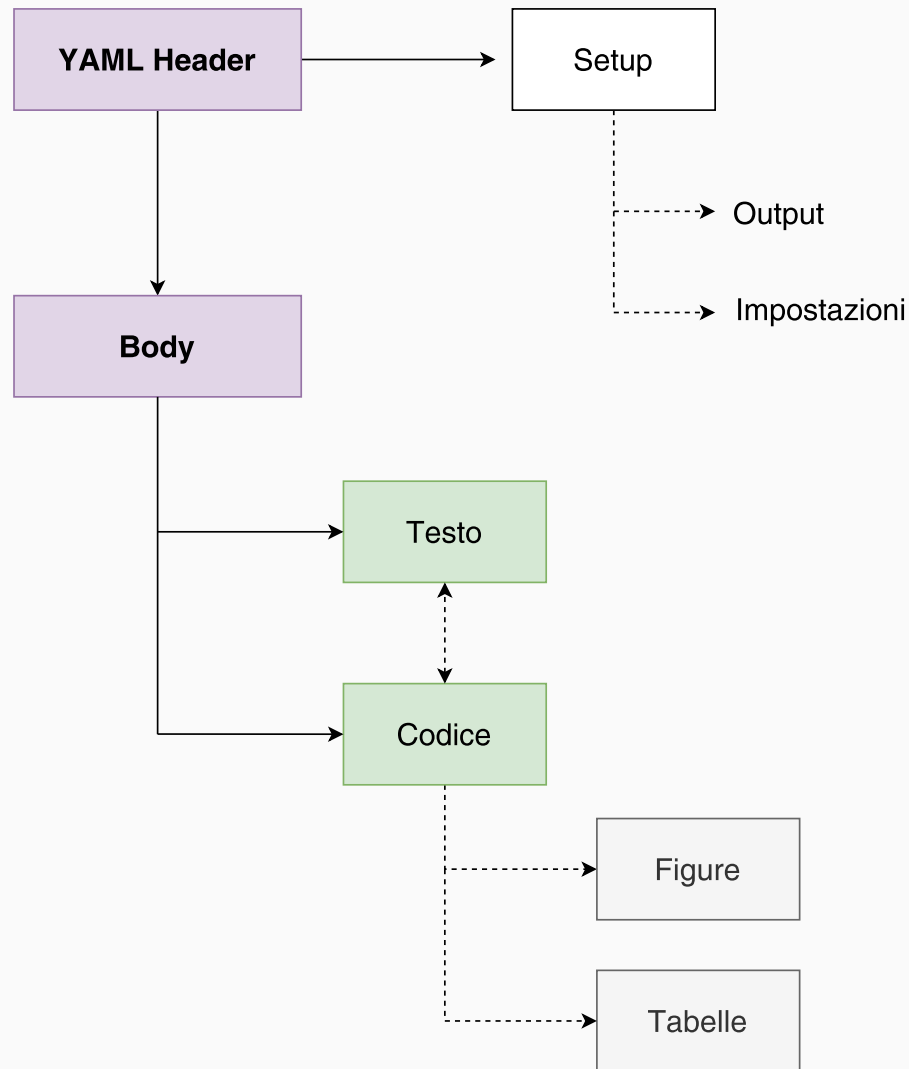
# Un documento dinamico

Un documento R Markdown è **dinamico** perchè un cambiamento nel testo, nei dati o nel codice automaticamente aggiorna l'output essendo che **figure, tabelle e numeri sono generati durante la compilazione**





# The Big Picture



# YAML Header

# YAML Header

All'inizio di ogni documento RMarkdown è presente una sezione (YAML header) dove indicare le impostazioni generali del documento:

```
---  
title: "Document Title"  
author: "Autore"  
date: "2/18/2022"  
output: html_document  
---
```

Ci sono tantissime opzioni che dipendono:

- dal tipo di documento (presentazione, report, tesi, etc.)
- dal tipo di output (html, pdf, word, etc.)

# YAML Header

L'unica cosa importante è scrivere questa sezione in modo corretto. Lo `YAML` è un linguaggio di markup interpretato come codice e richiede delle regole:

- `argomento: valore`
- indenzazione corretta (spazi, a capo)
- argomenti corretti (esempio `output: html-document` è sbagliato, vi darà errore)

In generale lo `YAML` equivale alle impostazioni di un normale documento word che controlla i vari metadati, struttura, etc. La maggior parte delle volte non dovete scrivere questa sezione manualmente o comunque ci sono diversi template dove al massimo aggiustare alcune parti.

Body

# Body - Testo

Il corpo del documento è la parte principale che viene intesa come **testo** letteralmente. In questo caso dobbiamo scrivere (utilizzando un linguaggio di markup) quello che vogliamo sia visualizzato nel risultato finale:

```
---  
title: "Document Title"  
author: "Autore"  
date: "2/18/2022"  
output: html_document  
---
```

```
# Heading 1
```

Questo è il mio primo documento in **\*\*RMarkdown\*\***.

Effettivamente all'inizio è un *\*pochino strano\** non visualizzare subito il risultato finale ma i vantaggi sono tantissimi! Aspettate di vedere quando useremo il ``codice``.

```
## Heading 2
```

Vi renderete conto, dopo averlo utilizzato un pochino, che è facile da leggere e scrivere anche se non è esattamente immediato come un documento word.

# Body - Codice

Ci sono due tipi di codice che potete inserire in un documento RMarkdown:

- **code chunks**
- `inline code`

La differenza è che i code chunks sono operazioni più complesse che sono chiaramente distinte dal testo mentre gli inline code sono parte integrante del testo.

```
```{r, eval=TRUE}
print(1:5)
```
```

```
## [1] 1 2 3 4 5
```

Questo invece è un inline-code. Se voglio scrivere il numero `10` posso fare anche in questo modo: scrivo il numero `10`. Non si vede direttamente dalle slide ma io non ho scritto *verbatim* il numero 10 ma è il risultato di ``r 5+5``

# Codice - Chunks

Vediamo più nel dettaglio come è composto un chunk di codice:

The diagram illustrates the components of an R code chunk. It features three labels at the top with arrows pointing to specific parts of a code block below:

- Il "motore" del chunk** (The "engine" of the chunk): A red arrow points to the opening curly brace of the chunk options.
- Nome del chunk** (Name of the chunk): A green arrow points to the text `r pressure` within the chunk options.
- Altre opzioni del chunk** (Other options of the chunk): A blue arrow points to the `echo=FALSE` option within the chunk options.

```
... {r pressure, echo=FALSE}  
# ..codice  
plot(pressure)  
# ..codice  
...
```



# Codice - Inline

Il formato dell'inline invece è più semplice, si utilizzano i backticks ``r codice``.

L'espressione viene quindi interpretata come codice ed il risultato viene inserito all'interno del testo. Immaginiamo di avere un oggetto R salvato nell'ambiente principale:

```
## [1] -0.7068766 -1.7352091  0.6627869  0.9603524  0.2116784 -0.3300235  0.1607794 -0.710
## [10]  1.3689565
```

Se durante il testo vogliamo fare riferimento all'oggetto `x` possiamo riportare caratteristiche di `x` senza esplicitamente scrivere i numeri:

- La media di `x` è ``r mean(x)`` -> -0.0914981
- La deviazione standard di `x` è ``r sd(x)`` -> 0.9416894
- L'errore standard della media di `x` è ``r sd(x)/sqrt(length(x))`` -> 0.2977883

# Espressioni Matematiche

# Espressioni Matematiche

Qualche volta dobbiamo scrivere espressioni matematiche  $\frac{x}{2}$ , lettere greche  $\mu_x$  o espressioni complesse  $y_i = \beta_0 + \beta_1 X_1 + \epsilon_i$ . In word queste sono estremamente complesse da scrivere mentre in RMarkdown è sufficiente imparare la sintassi di Latex:

- `$\frac{x}{2}$`  ->  $\frac{x}{2}$
- `$y_i = \beta_0 + \beta_1 X_1 + \epsilon_i$`  ->  $y_i = \beta_0 + \beta_1 X_1 + \epsilon_i$

Qualche tutorial su come scrivere formule e simboli in RMarkdown:

- <https://rpruim.github.io/s341/S19/from-class/MathinRmd.html>
- <https://rmd4sci.njtierney.com/math>

# Immagini

# Codice - Immagini

Per inserire immagini possiamo utilizzare la **sintassi markdown** oppure **codice R**:

```

```



```
```{r, fig.cap = "The caption of my image"}  
knitr::include_graphics("img/ex-img2.png")  
```
```



The caption of my image

# Tabelle

# Tabelle

Le tabelle sono un pochino più complicate da creare completamente a mano (anche se ci sono **soluzioni** più automatiche). La sintassi Markdown è la seguente:

```
Colonna1	Colonna2	Colonna3
1	4	7
2	5	8
3	6	9
```

Crea questa tabella:

| <b>Colonna1</b> | <b>Colonna2</b> | <b>Colonna3</b> |
|-----------------|-----------------|-----------------|
| 1               | 4               | 7               |
| 2               | 5               | 8               |
| 3               | 6               | 9               |

# Tabelle

Solitamente però le tabelle sono: **statistiche descrittive** o **risultati di modelli statistici**. In questo caso crearle è estremamente semplice ed efficiente con alcuni pacchetti R basandosi su `modelli` salvati come oggetto o da `dataframe`:

```
## # A tibble: 3 x 6
##   Species      mean    sd      se    min    max
##   <fct>      <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 setosa      5.01  0.352  0.0498  4.3    5.8
## 2 versicolor 5.94  0.516  0.0730  4.9    7
## 3 virginica  6.59  0.636  0.0899  4.9    7.9
```



# Tabelle

```
library(kableExtra)
```

```
iris_summ %>%
```

```
  kable() %>%
```

```
  kable_styling(bootstrap_options = c("condensed", "striped"),  
                full_width = FALSE)
```

| <b>Species</b> | <b>mean</b> | <b>sd</b> | <b>se</b> | <b>min</b> | <b>max</b> |
|----------------|-------------|-----------|-----------|------------|------------|
| setosa         | 5.006       | 0.3524897 | 0.0498496 | 4.3        | 5.8        |
| versicolor     | 5.936       | 0.5161711 | 0.0729976 | 4.9        | 7.0        |
| virginica      | 6.588       | 0.6358796 | 0.0899270 | 4.9        | 7.9        |

# Tabelle

Ci sono alcuni pacchetti (e.g., `sjplot`) che partendo da un modello fittato in R creano una tabella pronta per essere messa in un paper o nella tesi:

```
library(sjPlot)  
sjPlot::tab_model(fit)
```

| <b>mpg</b>                               |                  |               |                  |
|--|------------------|---------------|------------------|
| <i>Predictors</i>                        | <i>Estimates</i> | <i>CI</i>     | <i>p</i>         |
| (Intercept)                              | 19.20            | -7.62 – 46.02 | 0.154            |
| disp                                     | -0.04            | -0.05 – -0.02 | <b>&lt;0.001</b> |
| qsec                                     | 0.38             | -0.58 – 1.33  | 0.427            |
| gear                                     | 0.71             | -1.80 – 3.21  | 0.567            |
| Observations                             | 32               |               |                  |
| R <sup>2</sup> / R <sup>2</sup> adjusted | 0.725 / 0.695    |               |                  |

# Gestire Bibliografia?

# Gestire Bibliografia

Anche la gestione della bibliografia è estremamente semplice. L'idea è di avere un file `.bib` con tutti i riferimenti che vogliamo citare. I file bib possono essere generati dai vari software per gestire la bibliografia (Mendeley, Zotero, etc.) oppure direttamente dai database (Google Scholar):

The screenshot shows a Google Scholar search for 'vogel machizawa'. The search results are displayed in a list format. The first result is titled 'Neural activity predicts individual differences in visual working memory capacity' by Edward K. Vogel and Maro G. Machizawa. A red circle with the number '1' is placed over the 'Cite' button for this article. A 'Cite' dialog box is open, showing citation formats for MLA, APA, Chicago, Harvard, and Vancouver. A red circle with the number '2' is placed over the 'BibTeX' button in the dialog box.

Google Scholar

vogel machizawa

Articles

About 2,650 results (0.06 sec)

Any time

Since 2022

Since 2021

Since 2018

Custom range...

Sort by relevance

Sort by date

Any type

Review articles

include patents

include citations

Create alert

[HTML] Neural activity predicts individual differences in visual working memory capacity

EK Vogel, MG Machizawa - Nature, 428(6984), 748-751.

Contrary to our rich phenomenological knowledge, the primate visual system can maintain representations of visual working memory capacity.

☆ Save **1** Cite Cited by

[HTML] Neural measures of visual working memory

EK Vogel, AW McCollough, M G Machizawa - Nature, 428(6984), 748-751.

The capacity of visual short-term memory is limited. To understand this limitation, it is important to know how many objects can be processed simultaneously 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

☆ Save Cite Cited by

Electrophysiological measures of visual working memory

AW McCollough, MG Machizawa - Nature, 428(6984), 748-751.

Visual working memory (WM) is limited. To understand this limitation, it is important to know how many objects can be processed simultaneously 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

X Cite

MLA Vogel, Edward K., and Maro G. Machizawa. "Neural activity predicts individual differences in visual working memory capacity." *Nature* 428.6984 (2004): 748-751.

APA Vogel, E. K., & Machizawa, M. G. (2004). Neural activity predicts individual differences in visual working memory capacity. *Nature*, 428(6984), 748-751.

Chicago Vogel, Edward K., and Maro G. Machizawa. "Neural activity predicts individual differences in visual working memory capacity." *Nature* 428, no. 6984 (2004): 748-751.

Harvard Vogel, E.K. and Machizawa, M.G., 2004. Neural activity predicts individual differences in visual working memory capacity. *Nature*, 428(6984), pp.748-751.

Vancouver Vogel EK, Machizawa MG. Neural activity predicts individual differences in visual working memory capacity. *Nature*. 2004 Apr;428(6984):748-51.

**2**

BibTeX EndNote RefMan RefWorks

# Gestire Bibliografia

Poi trovate questo, che estrae le informazioni rilevanti in un formato (.bib) leggibile da R

```
@article{vogel2004,  
  title={Neural activity predicts individual differences in visual working memory capacity},  
  author={Vogel, Edward K and Machizawa, Maro G},  
  journal={Nature},  
  volume={428},  
  number={6984},  
  pages={748--751},  
  year={2004},  
  publisher={Nature Publishing Group}  
}
```

# Gestire Bibliografia

Aggiornando lo `YAML` con il percorso del file `.bib` ed eventualmente un file `.csl` (che definisce lo stile della bibliografia, e.g., APA 7th):

```
---  
title: "Document Title"  
author: "Autore"  
date: "2/18/2022"  
output: html_document  
bibliography: files/references.bib  
csl: files/apa7.csl  
---
```

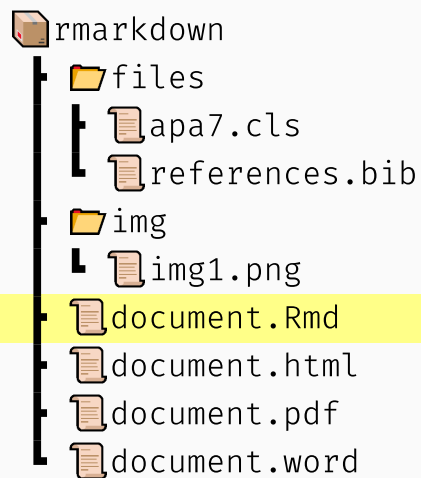
Poi potete durante la scrittura inserire una citazione inline `[@vogel2004]` e questa verrà inserita come citazione inline (Vogel & Machizawa, 2004) e come citazione alla fine in termini voce bibliografica completa:

- Vogel, E. K., & Machizawa, M. G. (2004). Neural activity predicts individual differences in visual working memory capacity. *Nature*, 428(6984), 748-751.

# Come organizzare un documento

# Come organizzare un documento

Solitamente siamo abituati ad avere un unico documento, ad esempio `documento.docx` che contiene tutto. Quando lavoriamo con R Markdown dobbiamo organizzarci in modo diverso:



Esiste il file `.Rmd` e tutti i file secondari (immagini, bibliografia, etc.). Il file `.Rmd` viene compilato per creare poi i vari output (`.html`, `.pdf`, `.word`, etc.)



Altre cose che posso fare?

# Altre cose che posso fare?

- Slides (queste slides sono fatte in RMarkdown)
  - [Xaringan](#)
- Curriculum (ci sono dei template)
  - [Vitae](#)
  - [Pagedown](#)
- Libri
  - [Bookdown](#)
- Siti web
  - [Blogdown](#)
  - [Distill](#)


# Qualche consiglio finale

# Qualche consiglio finale

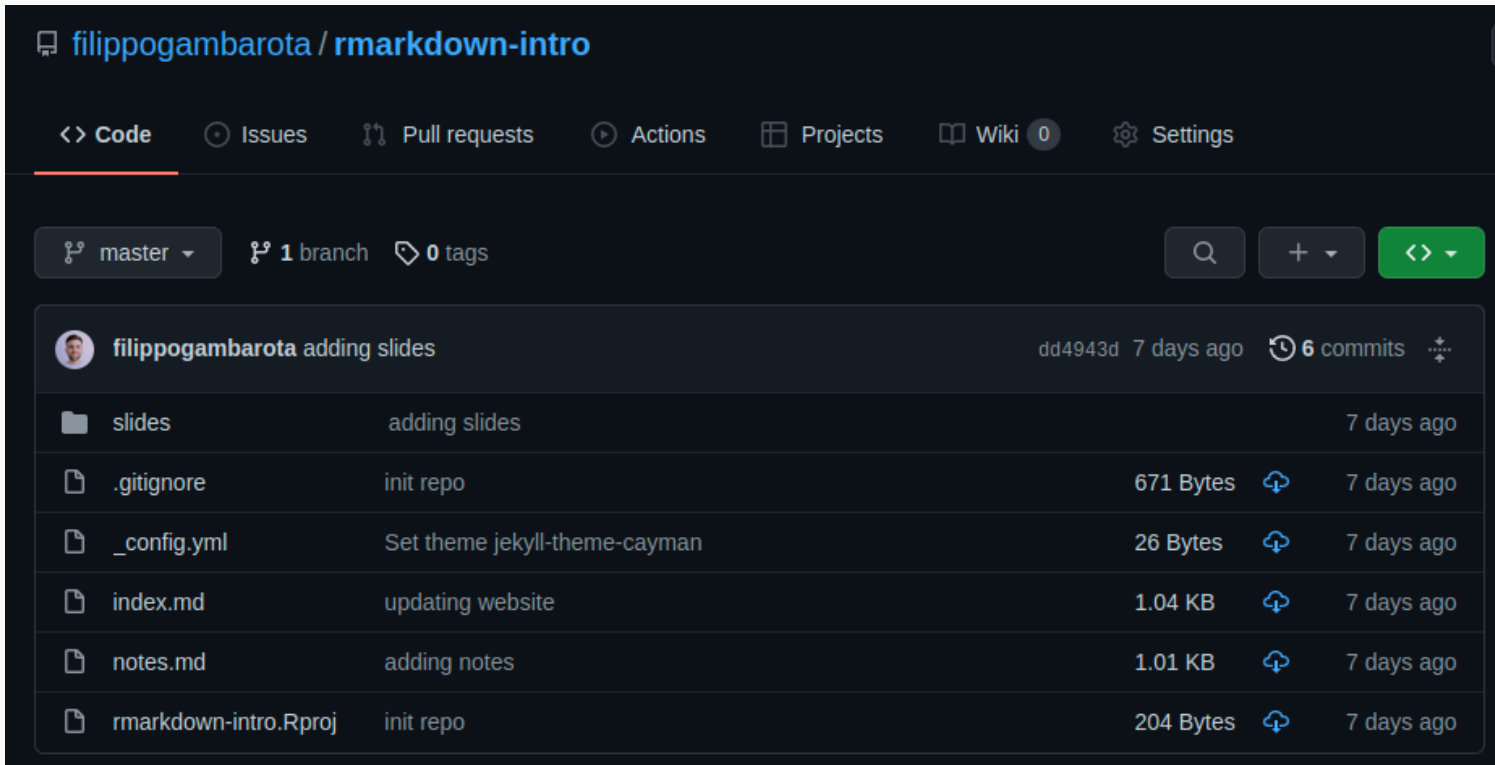
Le potenzialità di RMarkdown sono immense ma anche la curva di apprendimento è notevole, soprattutto di funzioni più avanzate. Quindi:

- Trovate un **template** per le vostre esigenze (report, tesi), scrivendo `rmarkdown template` su Google
- Se avete necessità, modificate alcune parti del template in modo da capire anche le funzionalità più avanzate
- Non preoccupatevi più di:
  - copiare e incollare numeri, statistiche e risultati
  - spostare manualmente tabelle e immagini
  - gestire a mano la bibliografia

# Resources

Tutto il materiale che riguarda l'introduzione a R Markdown si trova nella repository  [rmarkdown-intro](#). Potete accedere direttamente al [sito](#) Dove trovate:

- il codice per riprodurre queste slide
- i template di documenti, slide ed esercizi
- alcuni riferimenti come libri, tutorial e siti utili



filippogambarota / **rmarkdown-intro**

<> Code Issues Pull requests Actions Projects Wiki 0 Settings

master 1 branch 0 tags

filippogambarota adding slides dd4943d 7 days ago 6 commits

|                       |                               |           |            |
|-----------------------|-------------------------------|-----------|------------|
| slides                | adding slides                 |           | 7 days ago |
| .gitignore            | init repo                     | 671 Bytes | 7 days ago |
| _config.yml           | Set theme jekyll-theme-cayman | 26 Bytes  | 7 days ago |
| index.md              | updating website              | 1.04 KB   | 7 days ago |
| notes.md              | adding notes                  | 1.01 KB   | 7 days ago |
| rmarkdown-intro.Rproj | init repo                     | 204 Bytes | 7 days ago |

Some extras...

# Trackdown

Vi segnalo **Trackdown**, un pacchetto R creato e mantenuto da [Claudio Zandonella Callegher](#) al quale ho partecipato anche io.

Il pacchetto vi permette di lavorare su documenti R Markdown in modo collaborativo usando Google Docs. Se volete saperne di più è disponibile un [video](#) introduttivo



# Per approfondire...

Se volete ulteriormente approfondire, il Dipartimento di Psicologia dello Sviluppo e della Socializzazione sta organizzando dei corsi avanzati su argomenti utili nella ricerca (**Applied Research Courses Academy, ARCA**)

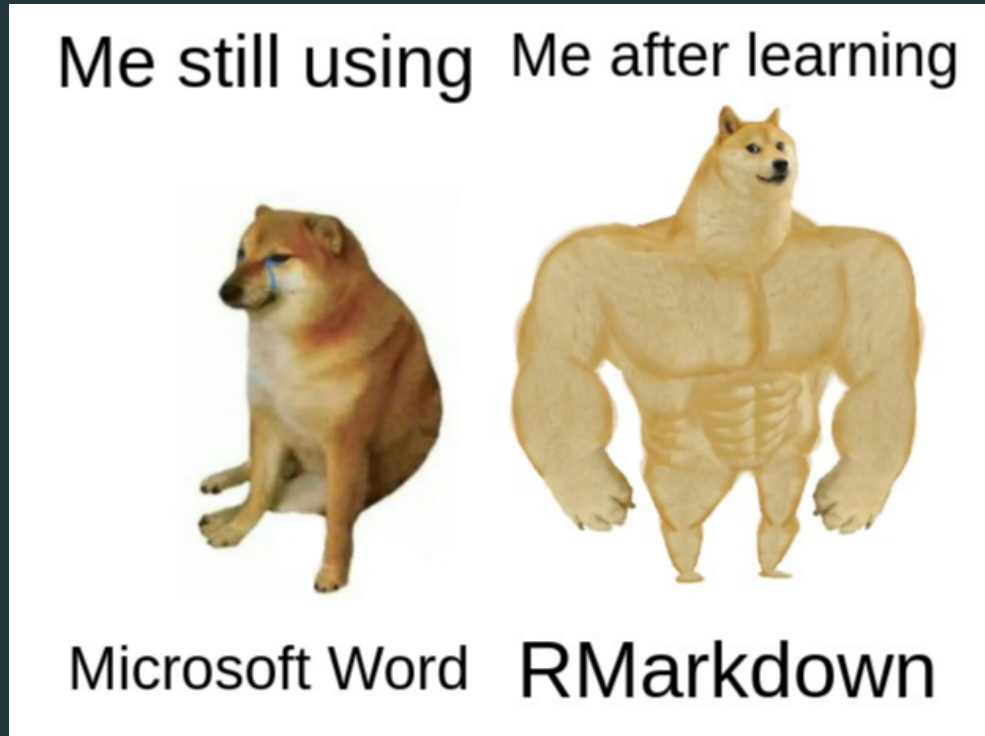
In particolare, la Dott.ssa **Ottavia Epifania** tiene un corso approfondito su R Markdown

<https://www.dpss.unipd.it/arca/RMarkdown>





Are you ready to create amazing documents? 😄



✉ [filippo.gambarota@gmail.com](mailto:filippo.gambarota@gmail.com)

🐦 [@fgambarota](https://twitter.com/fgambarota)

🔄 [filippogambarota](https://github.com/filippogambarota)