

# 05-xaringan

Ottavia M. Epifania



# Cos'è xaringan

- Un altro modo per fare le slide usando sempre R
- Utilizza RMarkdown ma tramite un motore che non è pandoc
- Il risultato: slide molto belle con meno fatica
- Offre infinite possibilità ma vedremo solo alcune feature

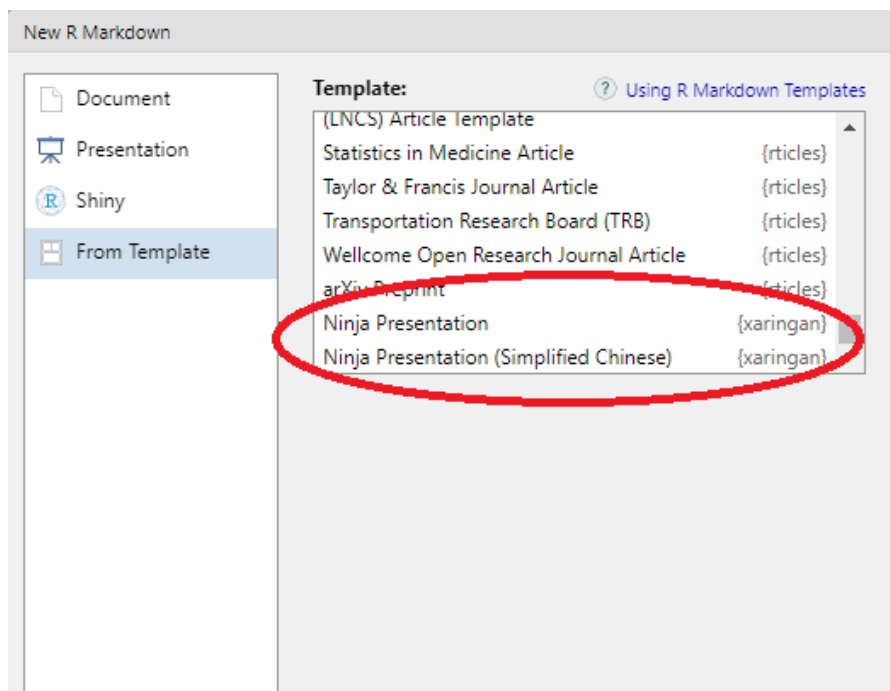
# Installazione

Nella console di RStudio basta scrivere:

```
> install.packages("xaringan")
```

premere invio e attendere il termine dell'installazione-

Per aprire un nuovo file:



# Il file di default

- Il file di default che viene creato all'apertura contiene un sacco di informazioni utili su xaringan
- Fornisce un template per nuove presentazioni
- Capendo come e dove modificare quel template vi trovate con una presentazione più che dignitosa!

**Presentazioni in xaringan**

# Yaml

```
---  
xaringan::moon_reader:  
  lib_dir: libs  
  css: [default, default-fonts]  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    highlightLanguage: r  
    highlightSpans: true  
    countIncrementalSlides: false  
---
```

# Yaml: dettagli

```
---
xaringan::moon_reader:
  lib_dir: libs
  css: [default, default-fonts]
  nature:
    highlightStyle: github
    highlightLines: true
    highlightLanguage: r
    highlightSpans: true
    countIncrementalSlides: false
---
```

Si possono definire i css da usare per la presentazione. xaringan fornisce una serie di css tra cui poter scegliere:

```
> names(xaringan::list_css())
```

```
## [1] "chocolate-fonts" "chocolate"      "default-fonts"  "de
## [5] "duke-blue"       "fc-fonts"       "fc"            "gl
```



# Yaml: dettagli

```
---  
xaringan::moon_reader:  
  lib_dir: libs  
  css: [default, default-fonts]  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    highlightLanguage: r  
    highlightSpans: true  
    countIncrementalSlides: false  
---
```

Le diverse modalità con cui si possono mostrare i chunk di codice:

```
> arta, ascetic, dark, default, far, github, googlecode, idea, ir-bl  
> magula, monokai, rainbow, solarized-dark, solarized-light, sunburst  
> tomorrow, tomorrow-night-blue, tomorrow-night-bright, tomorrow-nigh  
> tomorrow-night-eighties, vs, zenburn
```

# Yaml: dettagli

```
---  
xaringan::moon_reader:  
  lib_dir: libs  
  css: [default, default-fonts]  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    highlightLanguage: r  
    highlightSpans: true  
    countIncrementalSlides: false  
---
```

Permette di evidenziare le righe di codice segnate:

- \* all'inizio delle riga di codice
- Codice "rinchiuso" tra {{ }}
- Codice seguito da #<<

# Yaml: dettagli

```
---  
xaringan::moon_reader:  
  lib_dir: libs  
  css: [default, default-fonts]  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    highlightLanguage: r  
    highlightSpans: true  
    countIncrementalSlides: false  
---
```

Ogni linguaggio ha un suo modo specifico di essere riportato e si può scegliere

# Yaml: dettagli

```
---  
xaringan::moon_reader:  
  lib_dir: libs  
  css: [default, default-fonts]  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    highlightLanguage: r  
    highlightSpans: true  
    countIncrementalSlides: false  
---
```

Se viene messo `highlightSpans: true` permette di evidenziare solo delle parti specifiche di codice invece che la riga intera.

Basta mettere il codice che si vuole evidenziare dentro i backtick ``codice`` da evidenziare :

```
iris `%>%`
```

# Yaml: dettagli

```
---  
xaringan::moon_reader:  
  lib_dir: libs  
  css: [default, default-fonts]  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    highlightLanguage: r  
    highlightSpans: true  
    countIncrementalSlides: false  
---
```

Settando `countIncrementalSlides: true` le slide incrementalі vengono conteggiate nel totale

# Formattazioni generali

# Creare una nuova slide

---

*# Titolo*

*## Sottotitolo*

Testo

---

I tre tick iniziali --- sono fondamentali per creare la nuova slide, non basta più mettere solo il cancelletto

# Le pause

---

*# Titolo*

*## Sottotitolo (non è obbligatorio)*

Testo che appare subito

--

Testo che appare dopo la pausa

--

Testo che appare al terzo click del mouse

---



# Due colonne, stessa grandezza

**Testo nella parte sinistra**

*Testo nella parte destra*

**Parola colorata (HTML)**

Testo mini

# Due colonne, stessa ampiezza: Codice

```
> .pull-left[
+ **Testo nella parte sinistra**
+
+ <br>
+
+ <br>
+
+ <span style="color:red">Parola colorata (HTML)</span>]
+
+
+ .pull-right[
+ .center[*Testo nella parte destra*]
+
+ <br>
+
+ <br>
+
+ <font size="2">Testo mini</font>
+ ]
```

# Colonne di ampiezza differente

- Un
- Elenco
- Puntato

```
> iris %>%  
+   summary()
```

```
##      Sepal.Length      Sepal.Width      Petal.Length  
##      Min.       :4.300      Min.       :2.000      Min.       :1.000  
##      1st Qu.:5.100      1st Qu.:2.800      1st Qu.:1.600  
##      Median :5.800      Median :3.000      Median :4.350  
##      Mean   :5.843      Mean   :3.057      Mean   :3.758  
##      3rd Qu.:6.400      3rd Qu.:3.300      3rd Qu.:5.100  
##      Max.    :7.900      Max.    :4.400      Max.    :6.900  
##           Species  
##      setosa      :50  
##      versicolor:50  
##      virginica  :50  
##  
##  
##
```

# Due colonne, ampiezza differente: Codice

```
> .left-column[  
+ - Un  
+  
+ - Elenco  
+  
+ - Puntato  
+ ]  
+  
+ .right-column[  
+  
+ ]
```

# Immagini

![Caption](percorso-alla-figura)

```
`{r, fig.cap = "Caption", fig.knitr::include_graphics(path = "  
`
```



# Matematica

Esattemente come in Rmarkdown:

$$y = \alpha + \beta X + \varepsilon$$

$$y = \alpha + \beta X + \varepsilon$$

$$\frac{p}{1-p}$$

$$\frac{p}{1-p}$$

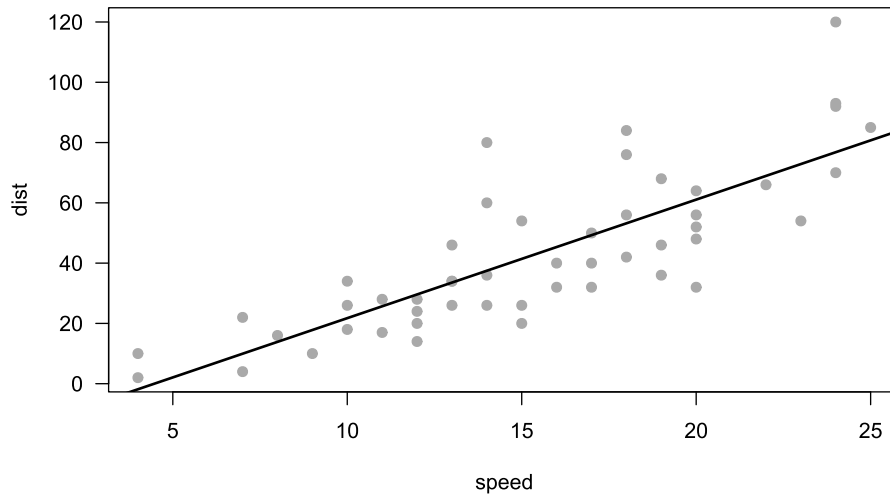
# Codice R

```
> fit = lm(dist ~ 1 + speed, data = cars)
> coef(summary(fit))
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-17.579095	6.7584402	-2.601058	1.231882e-02
speed	3.932409	0.4155128	9.463990	1.489836e-12

# Plot

```
> par(mar = c(4, 4, 1, 0.1))  
> plot(cars, pch = 19, col = "darkgray", las = 1)  
> abline(fit, lwd = 2)
```





# Tabelle

Le tabelle devono essere in formato HTML (non vale la sintassi specifica di RMarkdown).

Va usato direttamente il pacchetto **kable** per ottenere delle tabelle velocemente:

```
> knitr::kable(head(iris), format = "html")
```

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

# Tabelle... pro

Va installato il pacchetto DT (`install.packages("DT")`).

```
```{r}
library(DT)
DT::datatable(
  head(iris, 10),
  fillContainer = FALSE, options = list(pageLength = 8)
)
```
```

Show  entries

Search:

|   | Sepal.Length  | Sepal.Width  | Petal.Length  | Petal.Width  | Species |
|---|--|---|--|---|---------|
| 1 | 5.1  | 3.5   | 1.4  | 0.2   | setosa  |
| 2 | 4.9  | 3   | 1.4  | 0.2   | setosa  |
| 3 | 4.7  | 3.2   | 1.3  | 0.2   | setosa  |
| 4 | 4.6  | 3.1   | 1.5  | 0.2   | setosa  |

# Codice scrollabile

Può essere utile avere il codice che non si interrompe alla fine della pagina ma che scorre.

Per farlo, va aggiunto un po' di codice:

```
` ``{css, echo=F, eval = T}

.inverse {
  background-color: #272822;
  color: #d6d6d6;
  text-shadow: 0 0 20px #333;
}
.scrollable {
  height: 500px;
  overflow-y: auto;
}
.scrollable-auto {
  height: 80%;
  overflow-y: auto;
}
.remark-slide-number {
```

# Un esempio

[illegible]

# Your turn!

Provate a creare una presentazione con xaringan che abbia:

- Una slide con una tabella (possibilmente versione pro!)
- Una slide con due colonne di uguale ampiezza (a sx il codice per fittare il modello a dx il summary dei risultati)
- Una slide con due colonne di diversa ampiezza (a sx il codice per fittare il modello a dx il grafico dei risultati)
- Una slide con codice scrollabile