

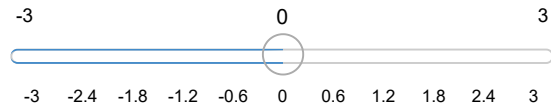
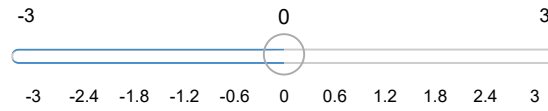
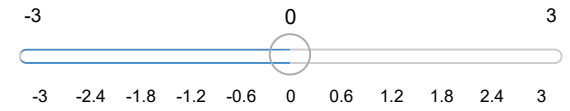
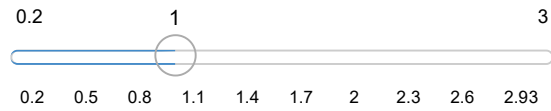
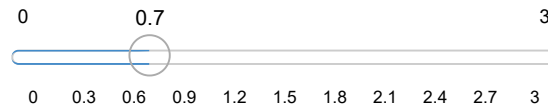
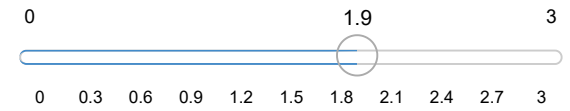
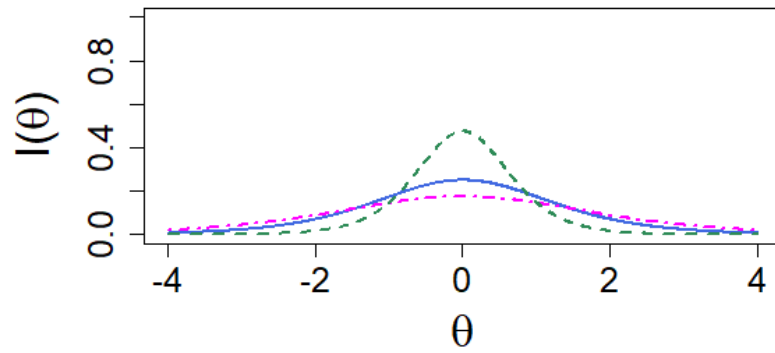
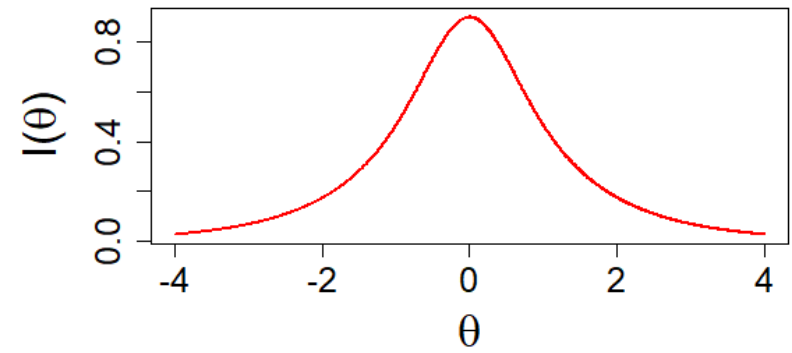


04 - Presentazioni

Ottavia M. Epifania

Perché?

- La riproducibilità
- La comodità (insomma, dipende)
- Ma soprattutto...

b1**b2****b3****a1****a2****a3****IIF****TIF**

Diverse opzioni

“Base”:

- **ioslides**: sono quelle che vi farò vedere io, sono le slide in `html` più semplici da ottenere e che permettono un risultato accettabile con relativamente poco sforzo
- **slidify**: sempre in `html`, ma sono leggermente più complesse e il risultato non è carino come `ioslides`
- **beamer_presentation**: per ottenere `pdf`. Tuttavia, se dovete fare presentazioni un pochino più complesse (e.g., volete mettere le colonne, fare animazioni complesse ecc.) conviene usare direttamente la presentazione di LaTeX

Advanced:

- [xaringam \(https://slides.yihui.org/xaringan/#1\)](https://slides.yihui.org/xaringan/#1)

N.B.: se non modificate troppo il template originale, si può saltellare allegramente tra `ioslides` e `beamer_presentation` (quindi tra `html` e `pdf`)

Nuovo documento

File → New File:

Si può scegliere direttamente il tipo di file

Il default è `ioslides`

For the love of God, NON scegliete il PowerPoint

YAML

```
---  
title: "Slide di prova"  
author: "Ottavia M. Epifania"  
date: "6/5/2022"  
output: ioslides_presentation  
---
```

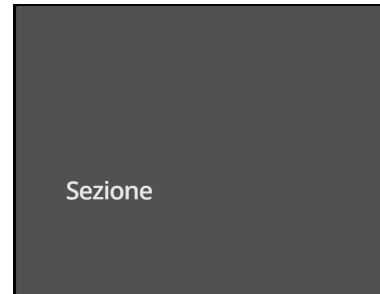
Questa è la base da cui si possono aggiungere pezzi per personalizzare le slide

Ad esempio, se si vogliono le slide “larghe” (come quelle che vi sto facendo vedere io) e un logo nella slide di titolo:

```
output:  
  ioslides_presentation:  
    logo: percorso-del-logo/logo.png  
    widescreen: yes
```

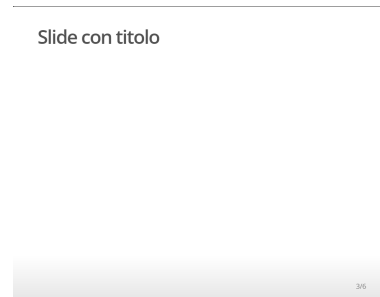
Sezioni e titoli

Nuova sezione



Sezione

Slide con titolo



Slide con titolo

3/6

Spostarsi dal default

Essendo un documento `html`, utilizza il linguaggio `html` e `css` per cambiare l'aspetto delle slide:

output:

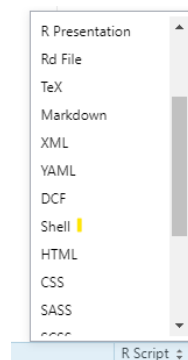
`ioslides_presentation:`

`css: percorso-al-css/style.css`

`logo: percorso-del-logo/logo.png`

`widescreen: yes`

Come creare il nostro `css`? Semplicemente aprendo un nuovo file `R` (`shift + control + n`) e andando a selezionare il tipo di file che vogliamo in basso a destra:



aggiungere link al `css` di prova

CSS

```
/* Cambia colore dei titoli */
```

```
h2 {  
  color: red;  
}
```

```
h3 {  
  color: blue;  
}
```

```
/* Imposta la dimensione del carattere della presentazione e l'interlinea */
```

```
.remark-slide-content {  
  font-size: 20px;  
  line-height: 1.6;  
}
```

```
/* Slide della sezione */
```

Formattazione

Grassetto, corsivo, corsetto, colori

Corsivo

Corsivo

****Grassetto****

Grassetto

******Corsetto******

Corsetto

`Parola colorata (HTML)`

Parola colorata (HTML)

Cambiare la dimensione del testo

`Miniscuolo, ma davvero minuscol`

Miniscuolo, ma davvero minuscolo

`Miniscuolo, ma non troppo minus`

Miniscuolo, ma non troppo minuscolo

`Piccolo ma non troppo`

Piccolo ma non troppo

`Dai che ci siamo!`

Dai che ci siamo!

`Dimensione leggibile`

Dimensione leggibile

Immagini

Markdown

```
![Caption](percorso-alla-figura)
```



Come pensavo di preparare il corso

RMarkdown

```
```{r, fig.cap = "Caption", fig.align="center"  
knitr::include_graphics(path = "percorso-alla-figura")
}```
```



Come è andata realmente

# Immagini

Se volete mettere le immagini in un punto specifico della slide, tipo:



Usate questo codice all'interno del testo:

```

```

# Contenuti “incrementali”

Per rendere i contenuti incrementali basta aggiungere `{.build}` accanto al titolo della slide:

## Titolo della slide `{.build}`



Ogni contenuto su una riga diversa verrà mostrato a ogni click. Per fare in modo che un contenuto venga mostrato tutto insieme (secondo le vostre esigenze):

```
<div>
```

Voglio che questa riga

E questa riga

Vengano mostrate insieme

```
</div>
```

Questa riga invece la voglio vedere dopo



# Colonne

Per creare delle slide con le colonne, si può scrivere semplicemente:

```
Titolo della slide {.columns-2}
```

In questo modo però non avete il controllo di quando la prima colonna diventa la seconda.

Aggiungete questo codice all'inizio della presentazione, prima del setup chunk:

```
<style>
.forceBreak { -webkit-column-break-after: always; break-after: column; }
</style>
```

e questo nel punto in cui volete passare dalla prima colonna alla seconda

```
<p class="forceBreak"></p>
```

# Ricapitolando

`## Titolo della slide {.columns}`

Testo nella prima colonna

`<p class="forceBreak"></p>`

Testo nella seconda colonna

# Colonne più flessibili

Usando `{.columns}` non si ha molta flessibilità → il testo viene automaticamente messo in due colonne

Per avere qualcosa di più flessibile, bisogna lavorare di più:

Testo fuori dalle colonne

```
<div style="float: left; width: 50%; text-align: left;">
```

Testo colonna sinistra con allineamento a sinistra

```
</div>
```

```
<div style="float: right; width: 50%; text-align: right;">
```

Testo colonna destra con allineamento a destra

```
</div>
```

# Your turn

- Create una slide con due colonne
- Nella colonna di sinistra: Testo
- Nella colonna di destra: Testo + immagine legata al vostro dataset
- Rendere i contenuti incrementali

## ADVANCED

- Allineamento del testo nella colonna a sinistra a destra
- Testo in blu nella colonna di destra
- Far apparire testo e immagine della colonna di destra insieme

**Codice e risultati**

# Chunk

Funziona esattamente come prima: `ctrl + alt + i` apre un nuovo chunk di codice:

```
```{r}
```

```
```
```

ed eredita tutte le opzioni impostate nel setup chunk a meno che non vengano specificate altre opzioni nel chunk specifico

# Piccoli trucchi

A volte si vuole mostrare molto codice o risultati molto lunghi che non stanno nella slide. Per risolvere questo “problema”:

- Ridurre il font del codice
- “Troncare” l’output di R in modo che rientri comodamente nella slide
- Scrollare il codice

# Ridurre il font del codice

All'inizio della presentazione definite una vostra classe in cui specificate la dimensione del font:

```
<style>
.myClass {font-size: 14px;}
</style>
```

(se avete già iniziato `<style>` perché avete definito la divisione delle colonne, don't worry e scrivete `.myClass` su una nuova riga)

Nel chunk di codice in cui volete ridurre il font:

```
```{r class.source="myClass"}  
  
```
```



# Fare scrollare il codice

Aggiungere questo chunk all'inizio della presentazione (anche prima del setup chunk)

```
```css
pre {
  max-height: 700px;
  overflow-y: auto;
}

pre[class] {
  max-height: 500px;
}

.scroll-100 {
  max-height: 500px;
  overflow-y: auto;
  background-color: inherit;
}
```
```

Nel chunk di cui si vuole fare scrollare il codice aggiungere l'argomento:  
`class.output="scroll-100"` ed è fatta!

# Troncare l'output del codice

Si deve usare l'argomento `out.lines`, ma richiede un po' di lavoro di preparazione.

Nel `setup chunk` (ripreso da <https://bookdown.org/yihui/rmarkdown-cookbook/hook-truncate.html> (<https://bookdown.org/yihui/rmarkdown-cookbook/hook-truncate.html>)):

```
hook_output <- knitr::knit_hooks$get("output")
```

```
knitr::knit_hooks$set(output = function(x, options) {
 if (!is.null(n <- options$out.lines)) {
 x <- xfun::split_lines(x)
 if (length(x) > n) {
 # truncate the output
 x <- c(head(x, n), "....\n")
 }
 x <- paste(x, collapse = "\n")
 }
 hook_output(x, options)
})
```

# Troncare l'output

Nel chunk specifico che si vuole troncare:

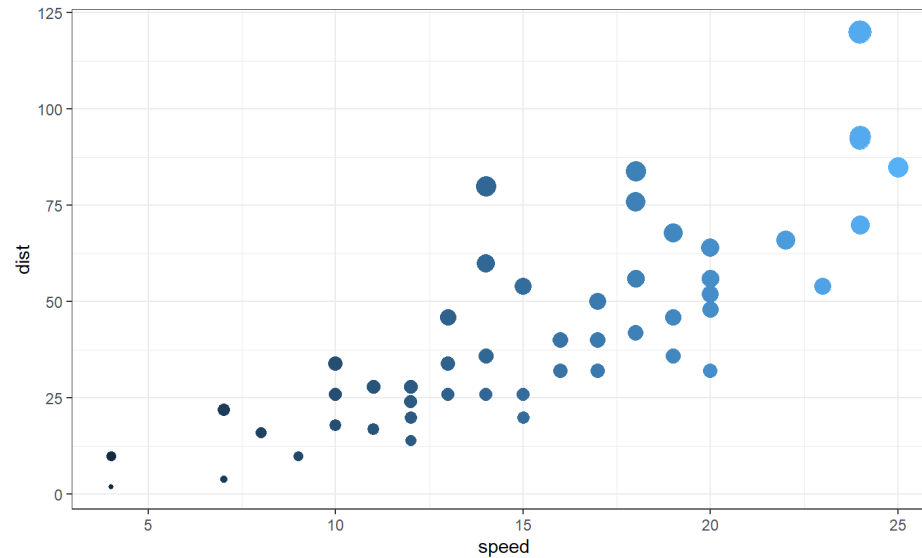
```
```{r out.lines=4}```  
cars  
```
```

```
speed dist
1 4 2
2 4 10
3 7 4
....
```

FYI: Funziona anche quando compilate i file in PDF

# Grafici

```
```{r out.width="50%", fig.align='center'}```  
library(ggplot2)  
ggplot(cars, aes(x=speed, y=dist, size =dist, color =speed)) + geom_point() +  
  theme_bw() + theme(legend.position = "none")  
```
```



Shiny

# Cos'è e a cosa serve



<https://shiny.rstudio.com/#:~:text=Shiny%20is%20an%20R%20package,%2C%20htmlv>  
è un pacchetto che permette di sviluppare delle app

Le app hanno bisogno di un server per poter essere condivise con il mondo, ma potete sempre costruirle e tenerle in locale (sono più utili di quello che pensate)



Ad esempio <https://fisppa.psy.unipd.it/DscoreApp/> è nata come app sul mio computer per fare le cose in fretta...

# Installazione e logica

Come ogni pacchetto di R:

```
install.packages("shiny")
```

Ogni shiny app ha due componenti principali:

ui (User Interface):

è il “contenitore” della app

gestisce l’aspetto della app

è quello che vede lo user

server (chi fa il lavoro)

è il “contenuto” della app

svolge tutti i calcoli che vanno a  
popolare lo ui

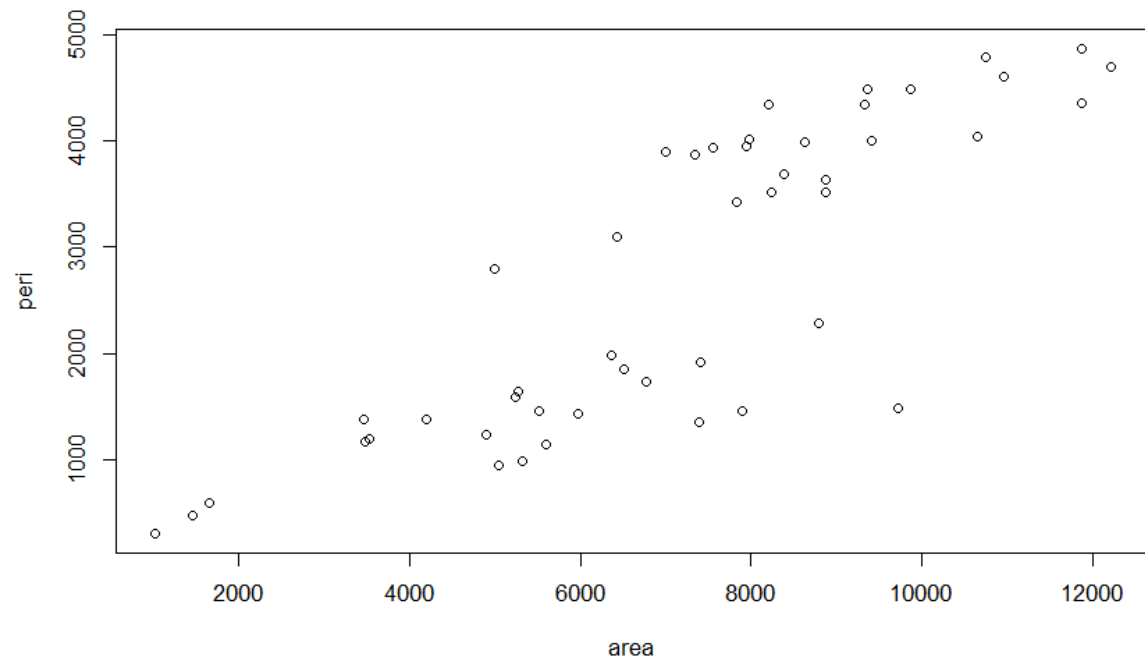
Lo user non sa neanche che esiste

# Come inserirla in RMarkdown

Aggiungete allo YAML: runtime: shiny e vederete che knit viene sostituito da Run Presentation

**Choose a dataset:**

rock ▼





# La App

```
shiny::shinyApp(
 ui = fluidPage(
 sidebarLayout(
 sidebarPanel(
 selectInput(inputId = "dataset", # nome dell'input per il server
 label = "Choose a dataset:", # nome dell'input per lo user
 choices = c("rock", "pressure")) # opzioni
),

 mainPanel(
 plotOutput(#qui voglio un grafico
 "graph"
)
)
)
),

 server = function(input, output){
```

**Tutto molto bello ma...**

# E il pdf?

Due opzioni:

1. Aprire la presentazione html in un browser (for the love of God, usate Chrome) e stampate il file in PDF
2. Usate pagedown che fa da solo l'operazione di cui sopra:

```
install.packages("pagedown") pagedown::chrome_print("percorso-al-file-
html/presentazione.html")
```