



# 02 - Chunk di codice

Corsi ARCA

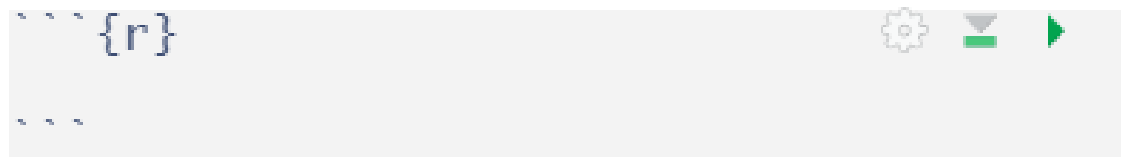
Ottavia M. Epifania

# Cosa sono

Sono quello che rende `RMarkdown` così “speciale”.

Permettono di integrare i documenti con i risultati delle analisi e con i codici usati per ottenerle (tutti i miei materiali supplementari alle riviste sono prodotti con `RMarkdown`)

Per crearli, basta usare la combinazione di tasti `Ctrl + Alt + i` (`Cmd + Alt + i` se si ha il Mac):

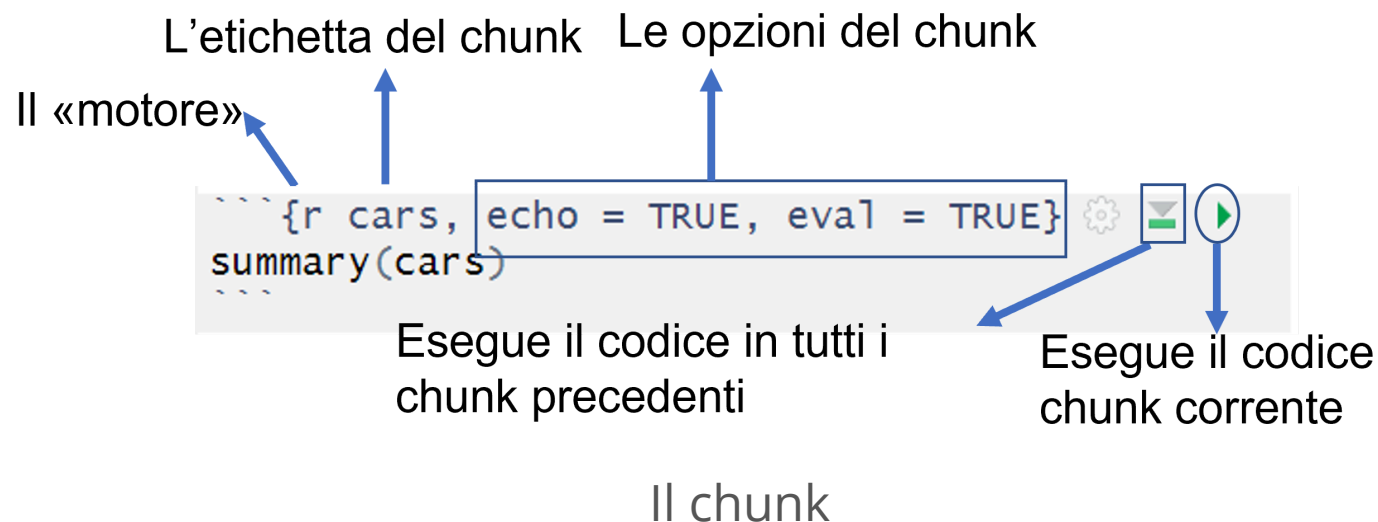


(si può anche usare l'icona



)

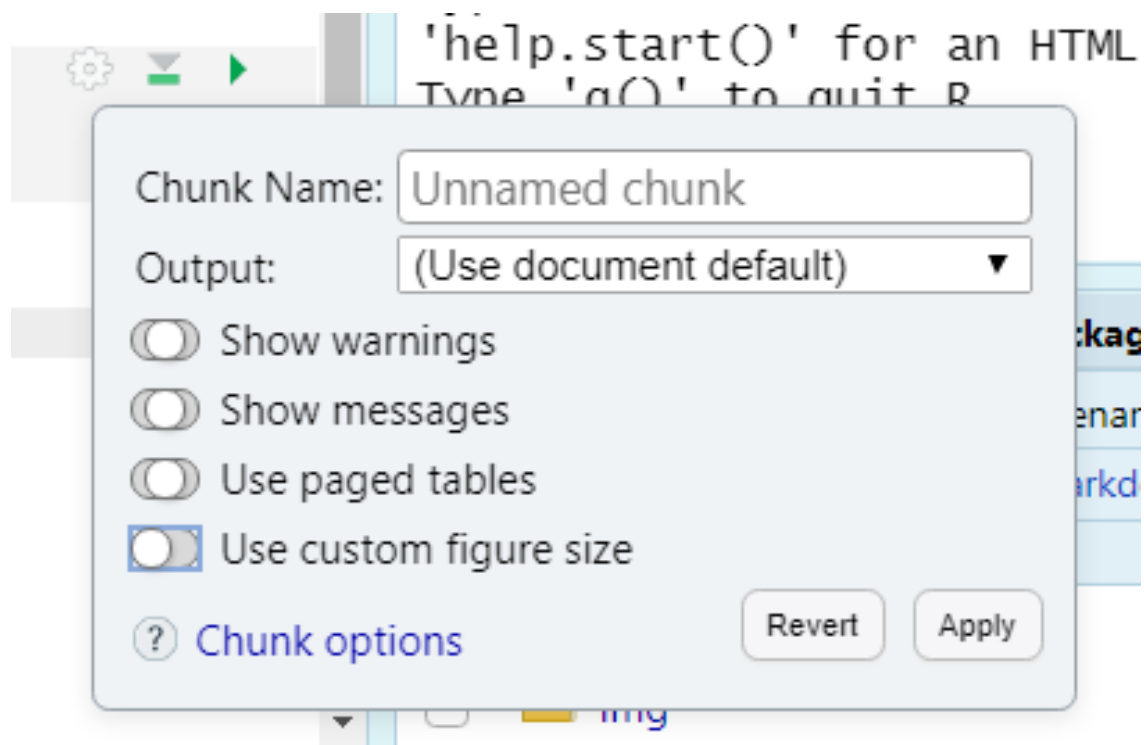
# Il file di default



Tutte le opzioni che si vogliono dare al chunk (e.g., come mostrare i risultati, come mostrare le figure, se mostrare o meno le figure ecc.) vanno separate dalla virgola e scritte a mano

# Oppure

Oppure si selezionano le opzioni base desiderate dal menu che appare cliccando sulla rotellina del chunk stesso:



A [questa pagina](#) è disponibile una lista esaustiva degli argomenti dei chunk di codice

Stessa cosa ma più schematica disponibile [qui](#)

# Setup chunk

Qui vanno specificate tutte le opzioni di default che vogliamo applicare al nostro documento, i pacchetti che pensiamo di usare per le nostre analisi, i dati, gli environment (per fare prima) eccetera.

Un esempio di setup chunk è:

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE,
                      fig.align = "center",
                      out.width = "50%")

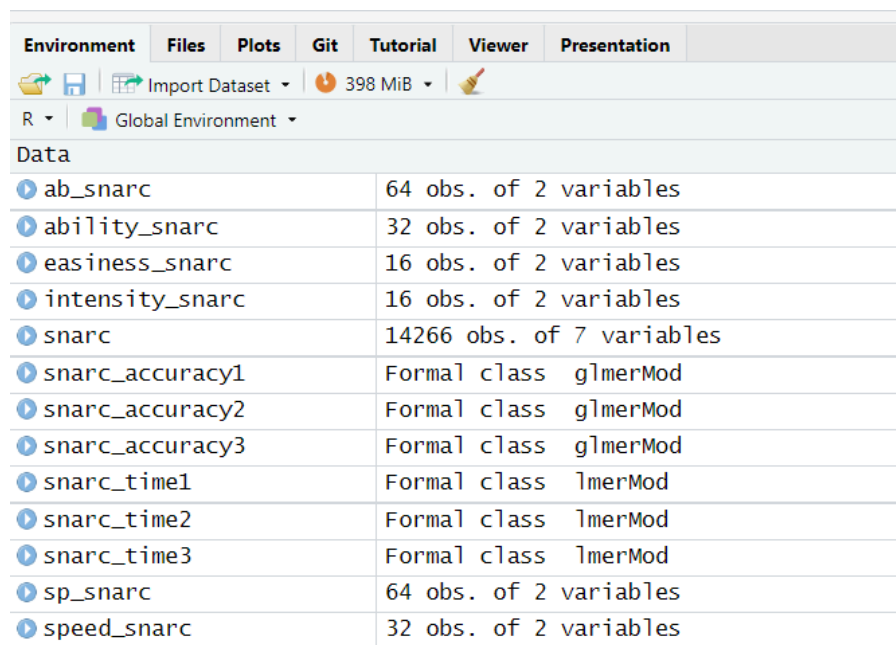
library(emoji)
library(knitr)
```
```

A meno che non venga specificato altrimenti, ogni chunk del documento seguirà le opzioni di default specificate nel setup chunk

# Warning

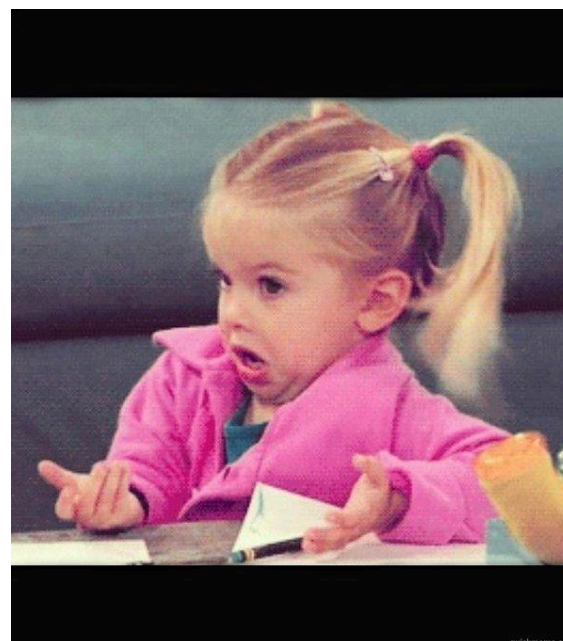
L'environment di R e l'environment di RMarkdown sono due cose diverse

Il fatto che voi vediate degli oggetti nel vostro environment come nell'esempio qui sotto non implica che siano importati automaticamente dentro RMarkdown



| Environment            | Files                     | Plots | Git | Tutorial | Viewer | Presentation |
|------------------------|---------------------------|-------|-----|----------|--------|--------------|
| Import Dataset 398 MiB |                           |       |     |          |        |              |
| R Global Environment   |                           |       |     |          |        |              |
| Data                   |                           |       |     |          |        |              |
| ab_snarc               | 64 obs. of 2 variables    |       |     |          |        |              |
| ability_snarc          | 32 obs. of 2 variables    |       |     |          |        |              |
| easiness_snarc         | 16 obs. of 2 variables    |       |     |          |        |              |
| intensity_snarc        | 16 obs. of 2 variables    |       |     |          |        |              |
| snarc                  | 14266 obs. of 7 variables |       |     |          |        |              |
| snarc_accuracy1        | Formal class glmrMod      |       |     |          |        |              |
| snarc_accuracy2        | Formal class glmrMod      |       |     |          |        |              |
| snarc_accuracy3        | Formal class glmrMod      |       |     |          |        |              |
| snarc_time1            | Formal class lmerMod      |       |     |          |        |              |
| snarc_time2            | Formal class lmerMod      |       |     |          |        |              |
| snarc_time3            | Formal class lmerMod      |       |     |          |        |              |
| sp_snarc               | 64 obs. of 2 variables    |       |     |          |        |              |
| speed_snarc            | 32 obs. of 2 variables    |       |     |          |        |              |

R environment



RMarkdown

# Warning I

Dovete ogni volta scrivere il codice per generare l'environment in modo che corra *senza errori*

Un'altra opzione è di importare l'environment salvato con il comando `load()`

# Your turn!

- Importate il vostro dataset nel setup chunk e assegnatelo a un oggetto:

```
> data(nome_dataset)
> dati = nome_dataset
```

- Create un nuovo chunk dove esplorate il dataset (`head(dati)`)

## ADVANCED

- Fate sparire il codice di R dall'output
- Salvate l'environment di R e richiamatelo nel setup chunk (hint: `load()`)

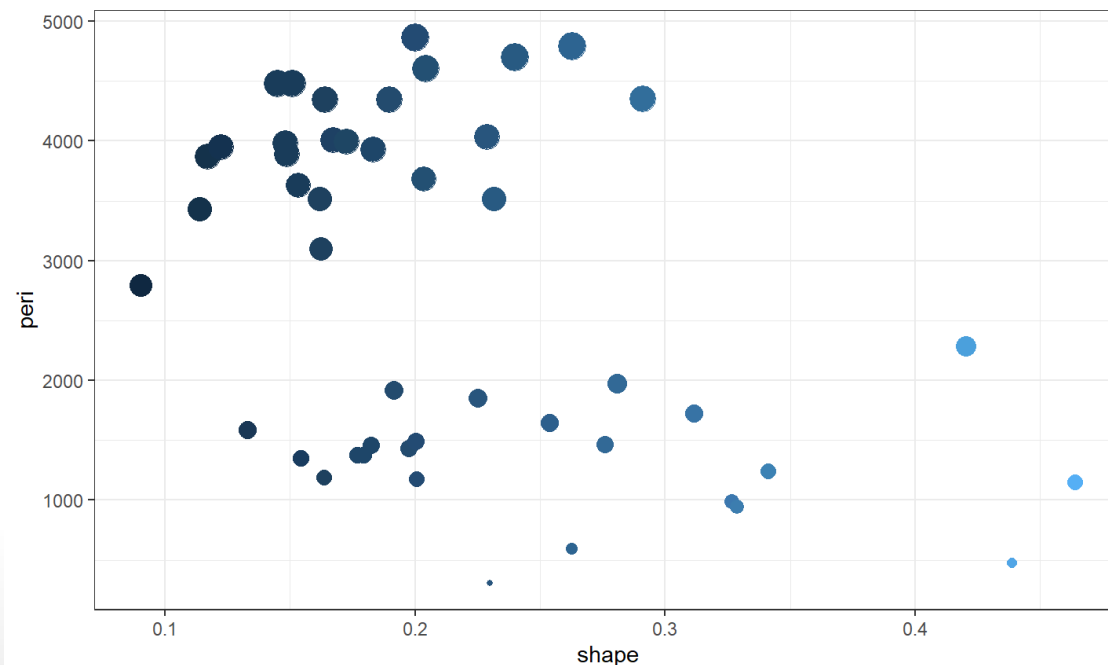


Codice e risultati

# Codice eseguito e risultati mostrati

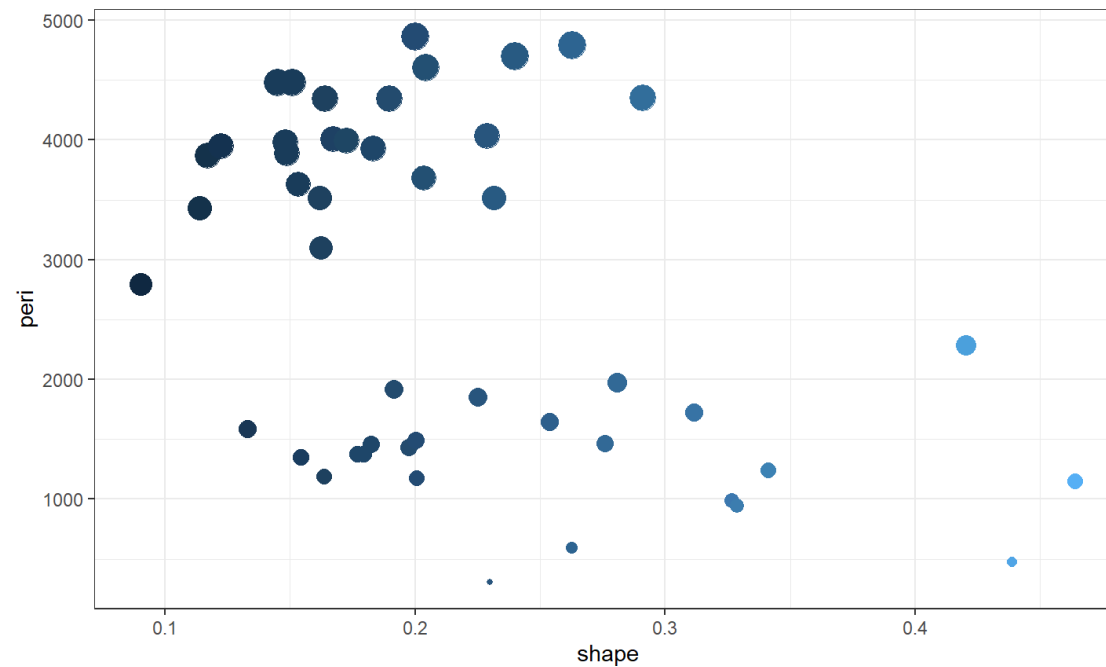
```
`r, echo=TRUE, eval=TRUE}  
ggplot(rock,  
      aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +  
      theme_bw() + theme(legend.position = "none")  
  
`r`
```

```
> ggplot(rock, aes(y = peri, x = shape, color = shape, size = peri)) + geom_point() +  
+       theme_bw() + theme(legend.position = "none")
```



# Risultati senza codice

```
```{r, echo=FALSE, eval=TRUE}  
ggplot(rock,  
       aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +  
       theme_bw() + theme(legend.position = "none")  
  
````
```



# Codice senza risultati (non eseguito)

```
```{r, echo=TRUE, eval=FALSE}  
ggplot(rock,  
       aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +  
       theme_bw() + theme(legend.position = "none")  
```
```

```
> ggplot(rock, aes(y = peri, x = shape, color = shape, size = peri)) + geom_point() +  
+       theme_bw() + theme(legend.position = "none")
```

# Codice eseguito, nessun risultato, nessun codice

```
```{r, include=FALSE}  
library(ggplot2)  
```
```

Non appare niente, non sono prodotti risultati visibili

L'opzione `include=FALSE` solitamente viene usata nel setup chunk e in tutti quei chunk di cui non interessa vedere un risultato ma che contengono un codice importante

# Messaggi di errore, warning, messaggi

- Molti pacchetti di R mostrano dei messaggi quando vengono caricati o quando viene usata qualche funzione. Per toglierli: `message=FALSE`
- Per togliere i warning (ad esempio quando usiamo `lme4` e fittiamo modelli molto complessi): `warning=FALSE`
- Per togliere i messaggi di errore (e.g., viene chiamato un oggetto che non esiste): `error=FALSE`

```
```{r, message=FALSE, error=FALSE, warning=FALSE}  
ggplot(mtcars, aes(x=cyl,y=mpg, group=cyl)) + geom_boxplot()  
```
```

⚠ i messaggi di errore/warning sono spariti, gli errori che li causano esistono ancora!!!

# `eval=FALSE` **VS.** `results='hide'`

`eval=FALSE`

- Il codice **non** viene eseguito
- Se ci sono degli errori, non ve ne accorgete
- Particolarmente utile quando si vuole spiegare un codice molto complesso (ma dovete essere sicuri che funzioni)

`results='hide'`

- Il codice **viene** eseguito
- Se ci sono degli errori il codice non va
- Particolarmente utile quando si vuole spiegare un codice più semplice

# Risultati un po' più carini

Di default, i risultati vengono mostrati così:

```
##      area      peri      shape perm
## 1 4990 2791.90 0.0903296 6.3
## 2 7002 3892.60 0.1486220 6.3
## 3 7558 3930.66 0.1833120 6.3
```

I cancelletti si possono togliere con l'argomento `comment`, settato uguale a quello che più vi piace:

```
```{r, comment=" "}  
rock  
```
```

```
      area      peri      shape  perm  
1  4990 2791.900 0.0903296    6.3  
2  7002 3892.600 0.1486220    6.3  
3  7558 3930.660 0.1833120    6.3  
4  7352 3869.320 0.1170630    6.3  
5  7943 3948.540 0.1224170   17.1  
6  7979 4010.150 0.1670450   17.1  
7  9333 4345.750 0.1896510   17.1  
8  8209 4344.750 0.1641270   17.1  
9  8393 3682.040 0.2036540   11.0
```



# Risultati un po' più carini PT. II

Il codice si può troncare in modo che non venga quell'orrore che avete visto:

Setup chunk:

```
> 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)
+ })
```

Nel chunk che volete tagliare:

```
```{r, comment=" ", out.lines=4}  
rock  
```
```

|      | area | peri     | shape     | perm |
|------|------|----------|-----------|------|
| 1    | 4990 | 2791.900 | 0.0903296 | 6.3  |
| 2    | 7002 | 3892.600 | 0.1486220 | 6.3  |
| 3    | 7558 | 3930.660 | 0.1833120 | 6.3  |
| .... |      |          |           |      |

Ricordate che una delle righe che volete mostrare è “mangiata” dai nomi delle colonne

# Your turn!

- Create un nuovo chunk per il summary dei dati `summary(dati)` **senza** codice e con “NA” al posto degli hashtag nell’output
- Nuovo chunk dove eseguite il codice del vostro dataset che trovate [qui](#) (non fate la regressione e il grafico) **ma non mostrate né il codice né i risultati**
- Nuovo chunk dove eseguite il codice del grafico (`plot(data$y ~ data$x)`) **senza** codice
- Nuovo chunk dove mostrate il codice del grafico (`plot(data$y ~ data$x)`) **senza** risultati
- Tagliate l’output del vostro dataset in modo che vengano mostrate le prime 10 osservazioni

Le tabelle

# La sintassi classica (Don't panic)

```
Colonna 1	Colonna 2	Colonna 3
1	2	3
4	5	6
```

Che mi dà:

| Colonna 1 | Colonna 2 | Colonna 3 |
|-----------|-----------|-----------|
| 1         | 2         | 3         |
| 4         | 5         | 6         |

Ci sono dei compilatori automatici che ci risparmiano la fatica, ma non è comunque il massimo

Stessa sintassi in `html` e `PDF`

# Integrare con R!

Ci sono diversi pacchetti che ci permettono di creare tabelle direttamente dai dataset senza fare fatica:

- [xtable](#)
- [stargazer](#) ❤️
- [sjPlot](#) (per le tabelle di correlazione non c'è niente di meglio)
- [kable e kableExtra](#)

In pratica, noi facciamo le nostre analisi e il pacchetto scelto le sistema in tabelle pronte all'uso!

# Tabelle senza fatica

Semplicemente l'output di R:

```
> (summary_cars = rock %>%  
+   group_by(perm) %>%  
+   summarise(mean = mean(peri), sd = sd(peri), min = min(peri), max = max(peri)))
```

```
## # A tibble: 12 × 5  
##       perm  mean    sd   min   max  
##   <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1     6.3 3621.  553. 2792. 3931.  
## 2    17.1 4162.  213. 3949. 4346.  
## 3    58.6 4685.  174. 4479. 4864.  
.....
```

😁 rapido, efficace, colpo d'occhio

Nella tesi non sta molto bene

# stargazer

- 😊 Fa in automatico i summary dei dataset
  - 😊 Sistema in tabella i risultati dei modelli
  - 😊 Stessa sintassi per html e PDF
  - 😊 Perfetto per model comparison
  - 😞 Non va bene per riportare le tabelle di summary preparate da noi.
- è ottimizzato per LaTeX (ossia per produrre i PDF).

## FONDAMENTALE:

```
```{r, results='asis'}
```

```
```
```



# stargazer summary

```
```{r, results='asis'}  
library(stargazer)  
stargazer(rock, type="latex", summary = TRUE,  
          title= "Tabella di summary", digits = 2, header=FALSE)  
```
```

```
##  
## \begin{table}[!htbp] \centering  
##   \caption{Tabella di summary}  
##   \label{}  
## \begin{tabular}{@{\extracolsep{5pt}}lccccc}  
## \\\[-1.8ex]\hline  
## \hline \\\[-1.8ex]  
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St.  
## \hline \\\[-1.8ex]  
## area & 48 & 7,187.73 & 2,683.85 & 1,016 & 12,212 \\  
## peri & 48 & 2,682.21 & 1,431.66 & 308.64 & 4,864.22 \\  
## shape & 48 & 0.22 & 0.08 & 0.09 & 0.46 \\  
## perm & 48 & 415.45 & 437.82 & 6.30 & 1,300.00 \\  
## \hline \\\[-1.8ex]
```

# stargazer model

```
> m1 = lm(peri ~ perm, data = rock)
> stargazer(m1, type = "html", summary = TRUE, title = "Risultati del modello", digits = 2,
+   header = FALSE)
```

| Risultati del modello   |                                     |
|-------------------------|-------------------------------------|
|                         | <i>Dependent variable:</i>          |
|                         | peri                                |
| perm                    | -2.42***<br>(0.32)                  |
| Constant                | 3,685.77***<br>(195.06)             |
| Observations            | 48                                  |
| R <sup>2</sup>          | 0.55                                |
| Adjusted R <sup>2</sup> | 0.54                                |
| Residual Std. Error     | 975.40 (df = 46)                    |
| F Statistic             | 55.25*** (df = 1; 46)               |
| Note:                   | $p < 0.1$ ; $p < 0.05$ ; $p < 0.01$ |

# stargazer model comparison

```
> m0 = lm(peri ~ 1, data = rock)
> m1 = lm(peri ~ perm, data = rock)
> stargazer(m0, m1, type = "html", title = "Model comparison", digits = 2, intercept.top = TRUE,
+   intercept.bottom = FALSE, header = FALSE)
```

| Model comparison        |                                     |                         |
|-------------------------|-------------------------------------|-------------------------|
|                         | <i>Dependent variable:</i>          |                         |
|                         | peri                                |                         |
|                         | (1)                                 | (2)                     |
| Constant                | 2,682.21***<br>(206.64)             | 3,685.77***<br>(195.06) |
| perm                    |                                     | -2.42***<br>(0.32)      |
| Observations            | 48                                  | 48                      |
| R <sup>2</sup>          | 0.00                                | 0.55                    |
| Adjusted R <sup>2</sup> | 0.00                                | 0.54                    |
| Residual Std. Error     | 1,431.66 (df = 47)                  | 975.40 (df = 46)        |
| F Statistic             | 55.25*** (df = 1; 46)               |                         |
| Note:                   | $p < 0.1$ ; $p < 0.05$ ; $p < 0.01$ |                         |

# Your turn!

- Tabella di summary del vostro dataset con 3 decimali
- Tabella di summary del vostro modello di regressione
- Caption per ognuna delle tabelle

## ADVANCED

- Tabella di model comparison con modello nullo  $m_0$  ( $m_0 = \text{lm}(y \sim 1, \text{data} = \text{dati})$ )

# Codice (e risultati) nel testo

Avete visto che nelle mie slide appaiono degli elementi scritti con la formattazione del codice di R

Ottenerli è molto semplice: ``R`` mi restituisce R

Ma molto di più: ``r mean(rock$peri)`` mi restituisce 2682.2119375 (Provateci!)

Se in un chunk precedente avessimo assegnato `x = mean(mtcars$mpg)`, avremmo potuto semplicemente scrivere ``r x``

# Codice, equazioni e codice nelle equazioni

$$\bar{X} = \frac{\sum_{i=1}^n x_i}{n}$$

Per mpg:

$$\bar{X} = \frac{\sum_{i=1}^n x_i}{n} = \frac{642.9}{32} = 20.09$$

```
$\frac{\text{round}(\text{sum}(\text{mtcars}$mpg$), 2)}{\text{nrow}(\text{mtcars})} = \text{round}(\text{round}(\text{sum}(\text{mtcars}$mpg$), 2) / \text{nrow}(\text{mtcars}), 2)$
```

# Your turn!

- Provate a scrivere la formula per standardizzare la variabile  $y$  sup:

$$z = \frac{x_i - \bar{X}}{sd}$$

- Standardizzate il primo valore della  $y$  del vostro dataset e riportate il risultato nell'equazione

**Saltellando tra HTML e PDF**



# Output specifici per formati specifici

Esiste la possibilità di far apparire degli oggetti diversi a seconda dell'output richiesto (HTML o PDF)

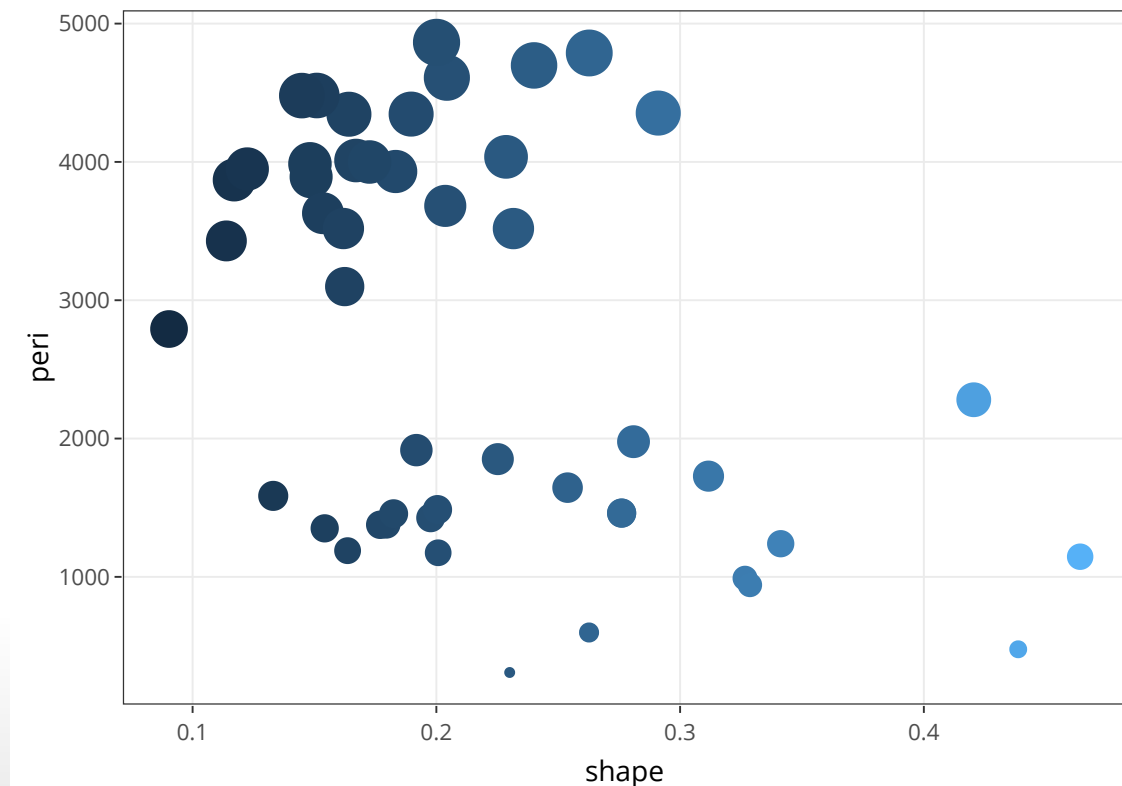
- `knitr::is_latex_output()`: Testa che il file venga compilato per produrre un PDF
- `knitr::is_html_output()`: Testa che il file venga compilato per produrre un HTML

Queste funzioni si possono usare sia negli argomenti dei chunk di codice sia nel codice riportato all'interno del testo

# Compilato in HTML ma non in PDF

```
```{r plot-interactive, echo=FALSE, eval=knitr::is_html_output() }
grafico = ggplot(rock,
  aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +
  theme_bw() + theme(legend.position = "none")
plotly::ggplotly(grafico)

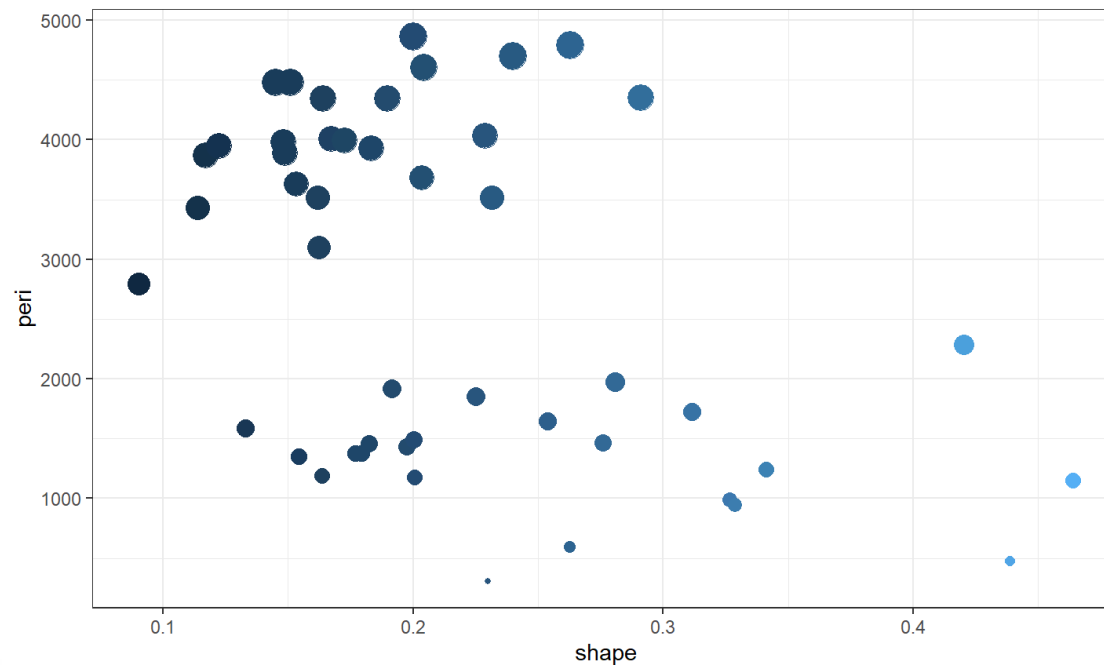
```
```



# Compilato in PDF ma non in HTML

```
```{r plot-static, echo=FALSE, eval=knitr::is_latex_output()}
grafico = ggplot(rock,
  aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +
  theme_bw() + theme(legend.position = "none")
grafico

```
```



# Possono stare nello file

Pandoc gestisce automaticamente l'output che viene mostrato a seconda del motore di compilazione che viene utilizzato

```
```{r, echo=FALSE, eval=knitr::is_html_output()}\ngrafico = ggplot(rock,\n  aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +\n  theme_bw() + theme(legend.position = "none")\nplotly::ggplotly(grafico)\n```
```

```
```{r, echo=FALSE, eval=knitr::is_latex_output()}\ngrafico = ggplot(rock,\n  aes(y=peri,x=shape, color =shape, size = peri)) + geom_point() +\n  theme_bw() + theme(legend.position = "none")\ngrafico\n```
```

# In-line code

Alcuni tag sono specifici per HTML o PDF, ad esempio per il cambio colore del testo

Se si utilizza il codice `\color{red}` Voglio una frase rossa `\normalcolor` in un file compilato in PDF si ottiene questo risultato:

Voglio una frase rossa

Se lo si utilizza in un file compilato in HTML il risultato potrebbe non essere dei migliori:

```
\color{red} Voglio una frase rossa \normalcolor
```

# In-line code condizionato

Vanno usate le funzioni `knitr::is_latex_output()` e `knitr::is_html_output()`, con qualche aggiunta:

- Va espressamente testato il motore della compilazione (funzione `ifelse()`) all'interno di un in-line code di R (``r``)
- Va definito l'output `asis`

```
`r` knitr::asis_output(ifelse(knitr::is_html_output(), '<span  
style='color:red'>parola</span>', '\color{red} Voglio una frase  
rossa \normalcolor'))`
```

# Your turn!

- Provate a inserire un grafico interattivo e uno statico, condizionando la loro comparsa al motore di compilazione
- Cambiate il colore del testo...sia per HTML sia per PDF!