Recap

Creare/nominare oggetti

Creare/nominare oggetti

Gli oggetti si possono creare e tramite il comando **<-** oppure **=**

```
x1 = 3 # nome = oggetto
x1
```

```
[1] 3
```

```
x2 <- 3 # nome <- oggetto
x2
```

```
[1] 3
```

```
x1 == x2 # i due oggetti sono identici?
```

```
[1] TRUE
```

I nomi degli oggetti devono iniziare con una lettera!! NON "nome" = ...ma nome = ...

Operatori Logici

In R è possibile congiungere più relazioni per valutare una desiderata proposizione.

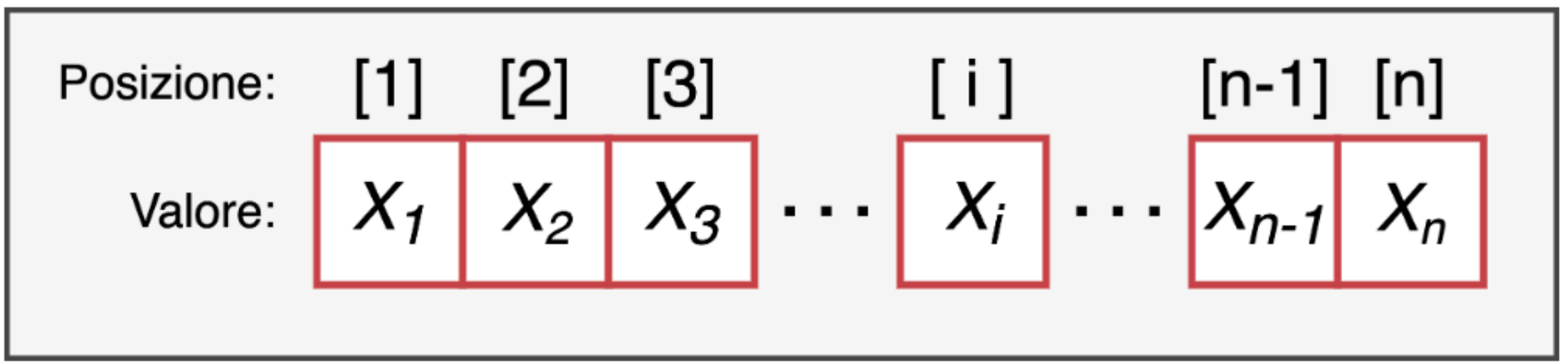
```
1  x = 30 #Assegnamo a x il valore 30.
```

Funzione	Nome	Esempio	Risulato
&	Congiunzione	`x>25 & x<60`	TRUE
|	Disgiunzione Inclusiva	`x>25 | x>60`	TRUE
!	Negazione	`!(x<18)`	TRUE

Vettori

Vettori

- **un valore**: il valore dell'elemento che può essere di qualsiasi tipo ad esempio un numero o una serie di caratteri
- **un indice di posizione**: un numero intero positivo che identifica la sua posizione all'interno del vettore.



Creare un vettore

I vettori si possono creare attraverso il comando **c( )**, indicando tra le parentesi i valori degli elementi nella sucessione desiderata e separati da una virgola.

```r
num_vect = c(1,2,3,4)
# è possibile anche utilizzare la funzione seq()
num_vect_seq = seq(from = 1,to = 4, by = 1)

num_vect;num_vect_seq
```

```
[1] 1 2 3 4
```

```
[1] 1 2 3 4
```

```r
char_vect = c("R","R","R","ok")
# è possibile anche utilizzare la funzione rep()
char_vect_rep = c(rep("R", 3), "ok")

char_vect;char_vect_rep
```

```
[1] "R"  "R"  "R"  "ok"
```

```
[1] "R"  "R"  "R"  "ok"
```

Tiplogia di vettore

Un vettore deve esssere formato da **elementi tutti dello stesso tipo**!

```
1 wrong = c(1,2,3,"non so", 4)
2 class(wrong)
```

```
[1] "character"
```

```
1 wrong
```

```
[1] "1"      "2"      "3"      "non so" "4"
```

Altrimenti si “rischia” che tutto venga trasformato a carattere.

```
1 correct = c(1,2,3,NA, 4)
2 class(correct)
```

```
[1] "numeric"
```

```
1 correct
```

```
[1]  1  2  3 NA  4
```

is.* & as.*

Possiamo testare o convertire (quando possibile) la tipologia del vettore attraverso queste funzioni **`is.`** & **`as.`**

```
is.character(char_vect)
```

```
[1] TRUE
```

```
as.numeric(char_vect) #!!
```

```
[1] NA NA NA NA
```

```
is.numeric(num_vect)
```

```
[1] TRUE
```

```
as.character(num_vect) #!!
```

```
[1] "1" "2" "3" "4"
```

```
logi_vect = c(TRUE,FALSE,TRUE)
is.logical(logi_vect)
```

```
[1] TRUE
```

```
as.numeric(logi_vect)
```

```
[1] 1 0 1
```

Indicizzazione

Possiamo selezionare, eliminare, estrarre elementi semplicemente usando l'indice di posizione tramite le parentesi quadre vettore[pos]

```
# Creo un vettore formato da 20 numeri casuali pescati da 1 a 100
my_vect = round(runif(n = 20,min = 1,max = 100))
my_vect
```

```
 [1] 40 63 34 98 62 41 76 94 37 76 66 18 82 92 82 73 69 48 15 74
```

```
my_vect[1] # estraggo il primo elemento
```

```
[1] 40
```

```
my_vect[1:5] # estraggo i primi 5 elementi
```

```
[1] 40 63 34 98 62
```

```
my_vect[c(1,4,2,9)] # estraggo elementi a scelta
```

```
[1] 40 98 63 37
```

```
my_vect[length(my_vect)] #ultimo elemento (perchè?)
```

```
[1] 74
```

Inidicizzazione negativa

Allo stesso modo possiamo decidere di estrarre tutti gli elementi del vettore eccetto alcuni

```
my_vect[-c(1)] #tutti tranne il primo elemento
```

```
[1] 63 34 98 62 41 76 94 37 76 66 18 82 92 82 73 69 48 15 74
```

```
my_vect[-c(1:10)] #tutti tranne i primi 10
```

```
[1] 66 18 82 92 82 73 69 48 15 74
```

Indicizzazione Logica

Indicizzare con la posizione è l'aspetto più semplice e intuitivo. E' possibile anche selezionare tramite valori **TRUE** e **FALSE**: possiamo estrarre elementi dal vettore basandoci su specifiche condizioni logiche

```
1 numeri = 1:7
2 numeri
```

```
[1] 1 2 3 4 5 6 7
```

```
1 numeri[numeri<2]
```

```
[1] 1
```

Fattori

Fattori

I fattori si possono creare sia convertendo un vettore character attraverso il comando **`as.factor()`** che creando esplicitamente un fattore attraverso il comando **`factor()`**

```
char_vect = rep(c("hello","ciao", "hola"), each = 2)
my_fact = as.factor(char_vect)

my_fact2 = factor(rep(c("hello","ciao", "hola"), each = 2))

my_fact == my_fact2
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE
```

Caratteristiche dei fattori

In pratica assegnano un'etichetta ad un valore numerico intero:

```
1 typeof(my_fact)
```

```
[1] "integer"
```

```
1 as.integer(my_fact)
```

```
[1] 2 2 1 1 3 3
```

```
1 my_fact
```

```
[1] hello hello ciao  ciao  hola  hola
Levels: ciao hello hola
```

Livelli del fattore

I fattori permettono di avere dei livelli **`levels()`** come metadati, a prescindere da quali siano effettivamente presenti nel vettore

```
1 levels(my_fact)
```

```
[1] "ciao"  "hello" "hola"
```

```
1 my_fact2 = my_fact[my_fact!="ciao"]
2 my_fact2
```

```
[1] hello hello hola  hola
Levels: ciao hello hola
```

E' possibile però eslcudere i livelli non più utili attraverso il comando **`droplevels()`**

```
1 droplevels(my_fact2)
```

```
[1] hello hello hola  hola
Levels: hello hola
```

E’ possibile anche rinominare i livelli del fattore

```
# il fattore che ho creato
my_fact
```

```
[1] hello hello ciao  ciao  hola  hola
Levels: ciao hello hola
```

```
# creo un altro fattore "my_fact_lev" identico a "my_fact"
my_fact_lev = my_fact
my_fact_lev == my_fact
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE
```

```
# rinomino i livelli del fattore
levels(my_fact_lev) = c("italiano","inglese","spagnolo")

my_fact_lev
```

```
[1] inglese  inglese  italiano italiano spagnolo spagnolo
Levels: italiano inglese spagnolo
```

```
my_fact
```

```
[1] hello hello ciao  ciao  hola  hola
Levels: ciao hello hola
```

E’ possibile sia riordinare che rinominare i livelli di un fattore attraverso la funzione `factor()`

```
my_fact
```

```
[1] hello hello ciao  ciao  hola  hola
Levels: ciao hello hola
```

```
# di default l'ordine di assegnazione è alfabetico
as.integer(my_fact)
```

```
[1] 2 2 1 1 3 3
```

```
new_fact = factor(my_fact, levels = c("hello","ciao","hola"))
new_fact
```

```
[1] hello hello ciao  ciao  hola  hola
Levels: hello ciao hola
```

```
as.integer(new_fact)
```

```
[1] 1 1 2 2 3 3
```

```
new_fact2 = factor(my_fact, levels = c("hello","ciao","hola"),
                   labels = c("inglese","italiano","spagnolo"))
new_fact2
```

```
[1] inglese  inglese  italiano italiano spagnolo spagnolo
Levels: inglese italiano spagnolo
```

Ricordate: l'argomento ***`levels`*** dentro la funzione **`factor()`** serve a **riordinare** i livelli del fattore, mentre la funzione **`levels()`** li **rinomina** non riordina!

```
# il fattore che ho creato
my_fact
```

```
[1] hello hello ciao  ciao  hola  hola
Levels: ciao hello hola
```

```
as.integer(my_fact)
```

```
[1] 2 2 1 1 3 3
```

```
levels(my_fact) = c("hello","ciao","hola")

# Cosa notate rispetto al vettore my_fact visualizzato sopra?
my_fact
```

```
[1] ciao  ciao  hello hello hola  hola
Levels: hello ciao hola
```

```
as.integer(my_fact)
```

```
[1] 2 2 1 1 3 3
```

Liste

Liste

Le liste sono strutture flessibili che possono contenere oggetti di tipo differente e di differenti dimensioni (ogni elemento può essere a sua volta una lista)

```r
my_list = list("ciao!", 1:3, c(TRUE,FALSE),
               list("ciao!", 1:3, c(TRUE,FALSE)))
str(my_list)
```

```
List of 4
 $ : chr "ciao!"
 $ : int [1:3] 1 2 3
 $ : logi [1:2] TRUE FALSE
 $ :List of 3
  ..$ : chr "ciao!"
  ..$ : int [1:3] 1 2 3
  ..$ : logi [1:2] TRUE FALSE
```

Liste

La lista pur essendo unidimensionale, si sviluppa in profondità

```
1 str(my_list)
```

```
List of 4
 $ : chr "ciao!"
 $ : int [1:3] 1 2 3
 $ : logi [1:2] TRUE FALSE
 $ :List of 3
  ..$ : chr "ciao!"
  ..$ : int [1:3] 1 2 3
  ..$ : logi [1:2] TRUE FALSE
```

Liste

Primo Livello: lista

```
1 str(my_list[2])
```

```
List of 1
 $ : int [1:3] 1 2 3
```

Secondo Livello: vettore numerico

```
1 str(my_list[[2]])
```

```
int [1:3] 1 2 3
```

Terzo Livello: numero intero

```
1 str(my_list[[2]][1])# numero intero
```

```
int 1
```

Liste

E’ possibile assegnare dei nomi agli elementi della lista…

```r
my_list_nam = list(parola = "ciao!",numeri =  1:3,
                   logico = c(TRUE,FALSE),
                 lista = list("ciao!", 1:3, c(TRUE,FALSE)))
my_list_nam
```

```
$parola
[1] "ciao!"

$numeri
[1] 1 2 3

$logico
[1]  TRUE FALSE

$lista
$lista[[1]]
[1] "ciao!"

$lista[[2]]
[1] 1 2 3
```

Liste

… ed accerdervi chiamandoli per “nome” attraverso l’operatore **$**

```
my_list_nam$parola
```

```
[1] "ciao!"
```

```
my_list_nam$lista #lista creata come elemento della lista
```

```
[[1]]
[1] "ciao!"

[[2]]
[1] 1 2 3

[[3]]
[1]  TRUE FALSE
```

Liste - Attributi

Come per i vettori anche le liste hanno una lunghezza (**`lenght()`**) ed eventualmente dei nomi (**`names()`**). Il comando **`str()`** (struttura) è molto utile per le liste perchè fornisce una visione sulla struttura:

```
1 length(my_list_nam); names(my_list_nam)
```

```
[1] 4
```

```
[1] "parola" "numeri" "logico" "lista"
```

```
1 str(my_list_nam)
```

```
List of 4
 $ parola: chr "ciao!"
 $ numeri: int [1:3] 1 2 3
 $ logico: logi [1:2] TRUE FALSE
 $ lista :List of 3
  ..$ : chr "ciao!"
  ..$ : int [1:3] 1 2 3
  ..$ : logi [1:2] TRUE FALSE
```

Liste - Indicizzazione

La differenza tra le parentesi quadre riguarda il fatto se vogliamo fare un subset della lista ottenendo un'altra lista oppure se vogliamo accedere direttamente all'elemento interno:

```
1 my_list_nam[1] #ottenendo un'altra lista
```

```
$parola
[1] "ciao!"
```

```
1 my_list_nam[[1]] # accedo all'elemento interno
```

```
[1] "ciao!"
```

Liste - Indicizzazione

Se vogliamo selezionare più elementi (quindi fare un vero e proprio subset della lista) dobbiamo sempre usare le parentesi quadre singole:

```
#lista composta dai primi due elementi della my_list_nam
my_list_nam[1:2]
```

```
$parola
[1] "ciao!"

$numeri
[1] 1 2 3
```

Liste - Indicizzazione Nested

Le liste hanno una struttura unidimensionale ma che si può sviluppare in profondità. Per selezionare elementi nested si possono concatenare più parentesi:

```
1 my_list_nam[[4]]
```

```
[[1]]
[1] "ciao!"

[[2]]
[1] 1 2 3

[[3]]
[1]  TRUE FALSE
```

```
1 my_list_nam[[4]][[1]]
```

```
[1] "ciao!"
```

```
1 my_list_nam[[4]][[2]]
```

```
[1] 1 2 3
```

Matrici

Matrici

Le matrici sono una struttura dati **bidimensionale** (caratterizzate da 2 dimensioni `dim()`) dove il numero di righe rappresenta la dimensione 1 e il numero di colonne la dimensione 2.

```
1  my_mat = matrix(data = 1:10, nrow = 2, ncol = 5, byrow = FALSE)
2  my_mat
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

```
1  matrix(data = 1:10, nrow = 2, ncol = 5, byrow = TRUE)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

Matrici - Caratteristiche

- Possono contenere **una sola tipologia** di dati
- Essendo **bidimensionali**, abbiamo bisogno di due indici di posizione (righe e colonne) per identificare un elemento
- Possono essere viste come un **insieme** di singoli **vettori**

Matrici - Caratteristiche

Il numero di righe e colonne non deve essere lo stesso necessariamente (matrice quadrata) ma il numero di righe deve essere compatibile con il vettore data:

```
1 matrix(data = 1:10, ncol = 3, nrow = 3)
```

```
Warning in matrix(data = 1:10, ncol = 3, nrow = 3): data length [10] is not a
sub-multiple or multiple of the number of rows [3]
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrici

Cosa fa R di default?

```
1 matrix(data = 1:10, ncol = 3, nrow = 3)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
1 matrix(data = 1:2, ncol = 3, nrow = 3)
```

```
     [,1] [,2] [,3]
[1,]    1    2    1
[2,]    2    1    2
[3,]    1    2    1
```

`warnings`: la funzione ci informa di qualcosa di potenzialmente problematico, ma (circa!!) tutto liscio

Matrici - Indicizzazione

Per identificare uno o più elementi nella matrice abbiamo bisogno di indici/e di riga e/o colonna separati da virgola, sempre con le parentesi quadre: matrice[riga, colonna]

```
my_mat
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

```
my_mat[1,1]
```

```
[1] 1
```

Matrici - Indicizzazione

E’ possibile anche selezionare un’intera riga o colonna

```
1 my_mat[1,]
```

```
[1] 1 3 5 7 9
```

```
1 my_mat[,1]
```

```
[1] 1 2
```

Matrici - Indicizzazione

Come per i vettori, anche in questo caso possiamo applicare l'indicizzazione logica:

```
1 my_mat>2
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,] FALSE TRUE TRUE TRUE TRUE
[2,] FALSE TRUE TRUE TRUE TRUE
```

```
1 # Tutti gli elementi maggiori di due
2 my_mat[my_mat>2]
```

```
[1]  3  4  5  6  7  8  9 10
```

Vettori e Matrici

I vettori si creano attraverso la funzione **`c()`** e possono essere concatenati tra loro sempre attraverso la stessa funzione:

```
my_vect1 = c(1:4); my_vect2 = c(5:10)
my_vect12 = c(my_vect1,my_vect2)
my_vect12
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

Matrici

Similarmente, le matrici possono essere unite tra loro attraverso i comandi **`cbind()`** e **`rowbind()`** :

```
1 my_mat = matrix(1:4,2,2)
2
3 cbind(my_mat, my_mat)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    1    3
[2,]    2    4    2    4
```

```
1 rbind(my_mat, my_mat)
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    1    3
[4,]    2    4
```

Dataframe

Dataframe

Il dataframe è la struttura più “complessa”, utile e potente di R. Da un punto di vista intuitivo è un foglio excel mentre da un punto di vista di R è una tipologia di lista con alcune caratteristiche/restrizioni

- ogni elemento della lista è un **vettore** con un **nome associato** (aka una colonna)
- ogni lista/colonna deve avere lo stesso numero di elementi
- di conseguenza ogni riga ha lo stesso numero di elementi (**struttura rettangolare**)

Crezione dataframe

Attraverso la funzione data.frame() è possibile creare un dataframe…

```r
# Creo un dataframe con 3 colonne
my_df = data.frame( col1 = 1:4, col2 = letters[1:4],
                    col3 = rnorm(4))
my_df
```

```
  col1 col2       col3
1    1    a  1.2726752
2    2    b -0.5706625
3    3    c -2.4899153
4    4    d -2.0592201
```

Attributi

Il **`dataframe`** ha sia gli attributi della **lista** ovvero i ***names*** ma anche gli attributi della **matrice** ovvero le ***dimensioni*** (righe e colonne)

```
1 typeof(my_df)
```

```
[1] "list"
```

```
1 attributes(my_df)
```

```
$names
[1] "col1" "col2" "col3"

$class
[1] "data.frame"

$row.names
[1] 1 2 3 4
```

```
1 dim(my_df)
```

```
[1] 4 3
```

Indicizzazione

Si posso utlizzare sia le parentesi quadre **[ ]** che il simbolo del dollaro **$**

```
my_df[1] # estraggo un data.frame 5x1
```

```
  col1
1    1
2    2
3    3
4    4
```

```
my_df[[1]]# estraggo la prima colonna del data.frame
```

```
[1] 1 2 3 4
```

```
my_df$col1# estraggo la prima colonna del data.frame
```

```
[1] 1 2 3 4
```

```
my_df[1,1]#  estraggo il primo elemento della prima colonna del data.frame
```

```
[1] 1
```

Indicizzazione - Operatori relazionali

Una delle operazioni più comuni che dovrete affrontare sarà sicuramente quella di estrarre/valutare un sottoinsieme di valori presenti nel vostro dataset:

```r
# includo solo le righe in cui alla colonna1 i valori sono maggiori di 2
my_df[my_df$col1 > 2, ]
```

```
  col1 col2      col3
3    3    c -2.489915
4    4    d -2.059220
```

```r
my_df[my_df[1] > 2, ]
```

```
  col1 col2      col3
3    3    c -2.489915
4    4    d -2.059220
```

Esempi

```
  col1 col2       col3
1    1    a  1.2726752
2    2    b -0.5706625
3    3    c -2.4899153
4    4    d -2.0592201
```

Indicizzazione

Ci sono anche dei modi alternativi e più compatti di indicizzare. Ad esempio usando la funzione **`subset()`**:

```
1 subset(iris, subset = Species == "setosa" & Petal.Length > 1.7)
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
25          4.8         3.4          1.9         0.2  setosa
45          5.1         3.8          1.9         0.4  setosa
```

equivalente a:

```
1 iris[iris$Species == "setosa" & iris$Petal.Length > 1.7,]
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
25          4.8         3.4          1.9         0.2  setosa
45          5.1         3.8          1.9         0.4  setosa
```

E’ possibile anche selezionare colonne piuttosto che righe attraverso l’argomento ***select***:

```
1 head(subset(iris, select = c(Sepal.Length, Species)), n = 3)
```

```
  Sepal.Length Species
1          5.1  setosa
2          4.9  setosa
3          4.7  setosa
```

Possiamo anche combinare le due cose:

```r
head(subset(iris, Species == "setosa" & Sepal.Length > 4,
            c(Sepal.Length, Species)), n = 3)
```

```
  Sepal.Length Species
1          5.1  setosa
2          4.9  setosa
3          4.7  setosa
```

equivalente a:

```r
head(iris[iris$Species == "setosa" & iris$Sepal.Length > 4,
          c("Sepal.Length","Species")],n = 3)
```

```
  Sepal.Length Species
1          5.1  setosa
2          4.9  setosa
3          4.7  setosa
```

$$$$$$

La maggiorparte delle volte vi troverete ad accedere alle variabili tramite l'operatore **$**. Questo comando può essere utilizzato anche per creare una nuova variabile…

```
1 iris$somma = iris$Sepal.Length + iris$Sepal.Width
2 str(iris)
```

```
'data.frame':   150 obs. of  6 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1
1 1 ...
 $ somma       : num  8.6 7.9 7.9 7.7 8.6 9.3 8 8.4 7.3 8 ...
```

ESAME

https://etherpad.wikimedia.org/p/arca-corsoR