

Stick Breaking Problem

02/05/2022

Studente: Emanuele Izzo

Matricola: 0307385

Indice

1	Definizione del problema	1
1.1	Nomenclatura	1
2	Prima implementazione	2
2.1	Implementazione in Python	2
2.2	Implementazione in R	4
2.3	Commento	6
3	Seconda implementazione	7
3.1	Implementazione in R	7
3.2	Commento	8

1 Definizione del problema

Definizione 1.1. Poniamo di avere davanti a noi un bastoncino di lunghezza arbitraria l , e poniamo di volerlo dividere in due punti diversi. Ogni punto di rottura è scelto uniformemente a casa e in modo indipendente. Quanto sarà lungo in media il pezzo più lungo? E quanto sarà lungo in media il pezzo più corto?

Questa che viene presentata è la definizione standard di un problema famoso di probabilità e statistica, il quale può essere generato rispetto al numero di tagli:

Definizione 1.2. Poniamo di avere davanti a noi un bastoncino di lunghezza arbitraria l , e poniamo di volerlo dividere in n punti diversi. Ogni punto di rottura è scelto uniformemente a casa e in modo indipendente. Quanto sarà lungo in media il pezzo più lungo? E quanto sarà lungo in media il pezzo più corto?

In questo articolo verrà presentata una sua soluzione, partendo dall'implementazione della simulazione del problema in Python ed R, per poi effettuare delle congetture sui dati ottenuti, effettuando inoltre una verifica matematica sui risultati constatati.

1.1 Nomenclatura

Durante questo articolo verrà utilizzata la seguente nomenclatura:

- l rappresenta la lunghezza del nostro segmento o bastoncino.
- X_1, X_2, \dots, X_n rappresentano le variabili aleatorie indipendenti e identicamente definite tramite distribuzione uniforme $U(0, l)$ che rappresentano i punti di taglio (nel caso in cui $n = 2$ si utilizzeranno le variabili aleatorie X, Y).
- $E_{\max}^{(n)}$ rappresenta la variabile aleatoria della lunghezza del segmento più lungo.
- $E_{\min}^{(n)}$ rappresenta la variabile aleatoria della lunghezza del segmento più corto.

2 Prima implementazione

La seguente implementazione simula il taglio su un segmento di lunghezza $l = 1$. Il programma esegue m ripetizioni, e per ogni singola ripetizione considera n tagli indipendenti. I risultati sono memorizzati in una tabella $n \times m$, dove ogni riga rappresenta il numero di tagli considerati, mentre ogni colonna rappresenta il numero di ripetizioni dell'esperimento effettuate. Questa tabella verrà poi utilizzata per realizzare due grafici indipendenti: uno per la lunghezza media del segmento più lungo (in base al numero di tagli effettuati) e uno per la lunghezza media del segmento più corto (sempre in base al numero di tagli effettuati). Oltre a ciò viene memorizzata una tabella con i valori medi ottenuti ad una certa ripetizione (le milestone considerate sono memorizzate in una lista).

2.1 Implementazione in Python

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import random
4
5 lista_tagli = [2, 3, 4, 5, 6, 7, 8]
6
7 lista_ripetizioni = [10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000, 25000, 50000]
8 lista_ripetizioni_s = [str(i) for i in lista_ripetizioni]
9
10 mappa_risultati_max = np.zeros((len(lista_tagli), len(lista_ripetizioni)))
11 mappa_risultati_min = np.zeros((len(lista_tagli), len(lista_ripetizioni)))
12
13 for i, num_tagli in enumerate(lista_tagli):
14     for j, ripetizioni in enumerate(lista_ripetizioni):
15         tagli = [0]*num_tagli
16
17         for ripetizione in range(ripetizioni):
18             for k in range(num_tagli):
19                 tagli[k] = random.random()
20
21             for k in range(num_tagli - 1):
22                 for h in range(0, num_tagli - k - 1):
23                     if tagli[h] > tagli[h + 1]:
24                         tagli[h], tagli[h + 1] = tagli[h + 1], tagli[h]
25
26             segmenti = [0]*(num_tagli+1)
27
28             segmenti[0] = tagli[0]
29
30             for k in range(1, num_tagli):
31                 segmenti[k] = tagli[k] - tagli[k - 1]
32
33             segmenti[num_tagli] = 1 - tagli[num_tagli - 1]
```

```

34
35         segmento_min = min(segmenti)
36         segmento_max = max(segmenti)
37
38         mappa_risultati_max[i, j] += segmento_max
39         mappa_risultati_min[i, j] += segmento_min
40
41         mappa_risultati_max[i, j] /= ripetizioni
42         mappa_risultati_min[i, j] /= ripetizioni
43
44     print("Risultati per il valore medio della lunghezza massima:\n", mappa_risultati_max, "\n")
45     print("Risultati per il valore medio della lunghezza minima:\n", mappa_risultati_min, "\n")
46
47     plt.figure(figsize=(14, 6))
48
49     for i, tagli in enumerate(lista_tagli):
50         label_max = str(tagli) + " tagli"
51
52         plt.plot(lista_ripetizioni_s, mappa_risultati_max[i, :], label=label_max)
53
54     plt.xlabel("Numero di ripetizioni")
55     plt.ylabel("Lunghezza media del segmento più lungo")
56     plt.legend()
57     plt.show()
58
59     plt.figure(figsize=(14, 6))
60
61     for i, tagli in enumerate(lista_tagli):
62         label_min = str(tagli) + " tagli"
63
64         plt.plot(lista_ripetizioni_s, mappa_risultati_min[i, :], label=label_min)
65
66     plt.xlabel("Numero di ripetizioni")
67     plt.ylabel("Lunghezza media del segmento più corto")
68     plt.legend()
69     plt.show()

```

Listing 1: Implementazione della simulazione in Python

2.1.1 Esempio di esecuzione

Risultati per il valore medio della lunghezza massima:

```

[[0.61573858 0.5950687 0.59860258 0.6200694 0.61193868 0.61192919
  0.61522176 0.61485467 0.61290605 0.61129908 0.61115341 0.61053407]
 [0.54828886 0.53184066 0.53466354 0.51851209 0.5189706 0.51683647
  0.51727732 0.51869925 0.52235449 0.52231105 0.5219644 0.52120408]
 [0.43100681 0.40425148 0.42877682 0.43322646 0.45133156 0.44870239
  0.45415962 0.45999839 0.45693224 0.45619055 0.4572806 0.45726356]
 [0.36071707 0.41688664 0.4020735 0.40529555 0.40980215 0.40435052
  0.40991882 0.40749707 0.40643443 0.40755342 0.40880377 0.40802961]
 [0.37689282 0.37054659 0.37066966 0.367113 0.3707762 0.36712588
  0.36927241 0.36568083 0.37013758 0.37047337 0.37012188 0.37064343]
 [0.34725136 0.33527679 0.33549971 0.33566126 0.33455282 0.33602739
  0.33392554 0.33419257 0.33525115 0.3369335 0.33877287 0.3390487 ]
 [0.2956648 0.29371171 0.30327594 0.30163824 0.31198695 0.31346066
  0.31031932 0.31225899 0.3133879 0.31409433 0.3142561 0.31447226]]

```

Risultati per il valore medio della lunghezza minima:

```

[[0.08660171 0.1080004 0.10487104 0.10012027 0.10798673 0.10755559
  0.10788317 0.10915736 0.10964776 0.11114001 0.11070885 0.1111428 ]
 [0.04794372 0.05211744 0.05184117 0.06039198 0.06138567 0.06542447
  0.06312202 0.06280029 0.0625053 0.06257118 0.06241903 0.06236395]
 [0.02378905 0.0397924 0.04057336 0.04002184 0.0401992 0.04073934
  0.0400408 0.03935173 0.03994407 0.03996087 0.03969869 0.03977111]
 [0.03542412 0.02656841 0.02549011 0.02698397 0.02721629 0.02796704
  0.02766366 0.02821453 0.02780603 0.02783024 0.02765131 0.02781476]
 [0.01119443 0.01384033 0.01760324 0.01794204 0.01925811 0.02004111
  0.01994896 0.02077358 0.02032237 0.02038253 0.02032082 0.02032983]

```

```
[0.0219998  0.01865011 0.01636204 0.01806386 0.01738275 0.01631534
0.01620892 0.01628887 0.01602965 0.01593225 0.01583929 0.01571782]
[0.01213457 0.0133284  0.01172017 0.01271646 0.01230642 0.01247752
0.01239619 0.01233221 0.01250442 0.01230482 0.01234876 0.01226798]]
```

Listing 2: Esempio di esecuzione del programma

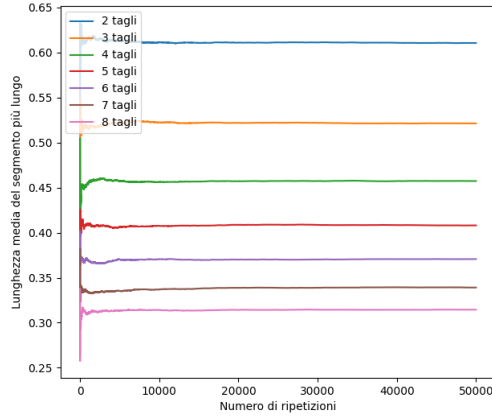


Figura 1: Grafico della lunghezza media del segmento più lungo.

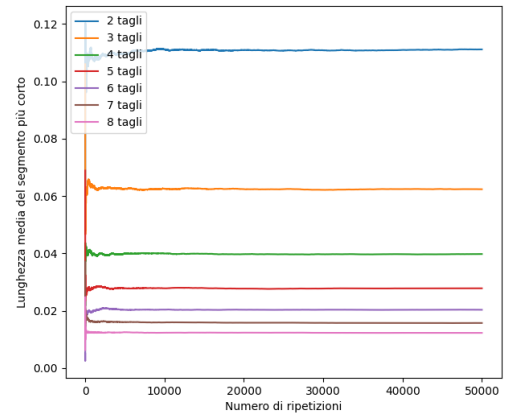


Figura 2: Grafico della lunghezza media del segmento più corto.

2.2 Implementazione in R

```
1 lista_tagli <- c(2, 3, 4, 5, 6, 7, 8)
2 lista_ripetizioni <- c(10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000, 25000, 50000)
3 ripetizioni_max <- lista_ripetizioni[length(lista_ripetizioni)]
4
5 mappa_risultati_max <- matrix(0, nrow = length(lista_tagli), ncol = length(lista_ripetizioni))
6 mappa_risultati_min <- matrix(0, nrow = length(lista_tagli), ncol = length(lista_ripetizioni))
7
8 lista_valori_medi_max <- matrix(0, nrow = length(lista_tagli), ncol = ripetizioni_max)
9 lista_valori_medi_min <- matrix(0, nrow = length(lista_tagli), ncol = ripetizioni_max)
10
11 for (i in seq_len(ripetizioni_max)) {
12   for (j in seq_len(length(lista_tagli))) {
13     num_tagli <- lista_tagli[j]
14
15     tagli <- rep(c(0), each = num_tagli)
16     tagli <- runif(num_tagli, min = 0, max = 1)
17     tagli <- sort(tagli)
18
19     segmenti <- c(0, tagli, 1)
20     segmenti <- diff(segmenti)
21
22     segmento_max <- max(segmenti, na.rm = TRUE)
23     segmento_min <- min(segmenti, na.rm = TRUE)
24
25     if (i >= 2) {
26       lista_valori_medi_max[j, i] <- ((lista_valori_medi_max[j, i-1]*(i-1)) + segmento_max) / (
27         i)
28       lista_valori_medi_min[j, i] <- ((lista_valori_medi_min[j, i-1]*(i-1)) + segmento_min) / (
29         i)
30     } else {
31       lista_valori_medi_max[j, i] = segmento_max / i
32       lista_valori_medi_min[j, i] = segmento_min / i
33     }
34   }
35 }
```

```

31     }
32   }
33
34   if (i %in% lista_ripetizioni) {
35     for (j in seq_len(length(lista_tagli))) {
36       k < which(lista_ripetizioni == i)[1]
37
38       mappa_risultati_max[j, k] < lista_valori_medi_max[j, i]
39       mappa_risultati_min[j, k] < lista_valori_medi_min[j, i]
40     }
41   }
42 }
43
44 par(mfrow = c(1, 2))
45 legenda < c("2 tagli", "3 tagli", "4 tagli", "5 tagli", "6 tagli", "7 tagli", "8 tagli")
46
47 matplot(seq_len(ripetizioni_max), y = t(lista_valori_medi_max), type = c("l"), pch = 1, col = 1:7
48         , xlab = "Numero di ripetizioni", ylab = "Lunghezza media del segmento piú lungo")
49 legend("topright", legend = legenda , col = 1:7, pch = 1)
50
51 matplot(seq_len(ripetizioni_max), y = t(lista_valori_medi_min), type = c("l"), pch = 1, col = 1:7
52         , xlab = "Numero di ripetizioni", ylab = "Lunghezza media del segmento piú corto")
53 legend("topright", legend = legenda , col = 1:7, pch = 1)

```

Listing 3: Implementazione della simulazione in R

2.2.1 Esempio di esecuzione

Risultati per il valore medio della lunghezza massima:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0.6260548	0.6180070	0.6187026	0.6174884	0.6074084	0.6134051
[2,]	0.4953807	0.5272483	0.5209433	0.5142318	0.5198002	0.5218453
[3,]	0.4487301	0.4589225	0.4575917	0.4437390	0.4517266	0.4527655
[4,]	0.4476307	0.4332816	0.4190160	0.4201961	0.4161928	0.4158346
[5,]	0.4098746	0.4201613	0.4004402	0.3958838	0.3776479	0.3769756
[6,]	0.3914930	0.3603590	0.3537549	0.3397202	0.3432720	0.3380176
[7,]	0.3551978	0.3277009	0.3256202	0.3268313	0.3221935	0.3262765

	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	0.6162088	0.6148884	0.6115781	0.6131054	0.6127974	0.6125408
[2,]	0.5225836	0.5205731	0.5208298	0.5208504	0.5206880	0.5210683
[3,]	0.4573809	0.4521926	0.4533760	0.4544021	0.4564114	0.4561257
[4,]	0.4092556	0.4084903	0.4075503	0.4094178	0.4093914	0.4087888
[5,]	0.3725873	0.3702594	0.3694764	0.3695492	0.3701447	0.3706831
[6,]	0.3359866	0.3417617	0.3404122	0.3402320	0.3405575	0.3404541
[7,]	0.3209351	0.3174291	0.3157064	0.3148847	0.3143497	0.3140642

Risultati per il valore medio della lunghezza minima:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0.09003096	0.10191572	0.09995308	0.10694489	0.11317019	0.10975878
[2,]	0.07740875	0.06395687	0.06500265	0.06416767	0.06316936	0.06169191
[3,]	0.04501717	0.03790637	0.03650874	0.03769275	0.04148461	0.04177674
[4,]	0.02165994	0.02790972	0.02790481	0.02746902	0.02697296	0.02788135
[5,]	0.01883297	0.01463707	0.01709341	0.01730480	0.01901692	0.01955036
[6,]	0.01723716	0.01222684	0.01151447	0.01266379	0.01389765	0.01490531
[7,]	0.01539266	0.01370301	0.01159470	0.01203663	0.01224663	0.01176085

	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	0.10903763	0.10987499	0.11213826	0.11191859	0.11091813	0.11096490
[2,]	0.06258968	0.06300079	0.06267810	0.06240760	0.06271635	0.06259380
[3,]	0.04058035	0.04162267	0.04166360	0.04072910	0.03991147	0.04010663
[4,]	0.02791358	0.02785043	0.02810729	0.02781651	0.02758278	0.02776478
[5,]	0.02111396	0.02040449	0.02055592	0.02043586	0.02046256	0.02044115
[6,]	0.01544700	0.01549501	0.01564508	0.01566362	0.01569236	0.01560599
[7,]	0.01158724	0.01179953	0.01203343	0.01214537	0.01227710	0.01235044

Listing 4: Esempio di esecuzione del programma

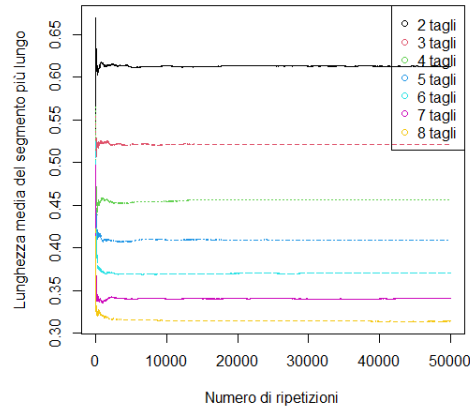


Figura 3: Grafico della lunghezza media del segmento più lungo.

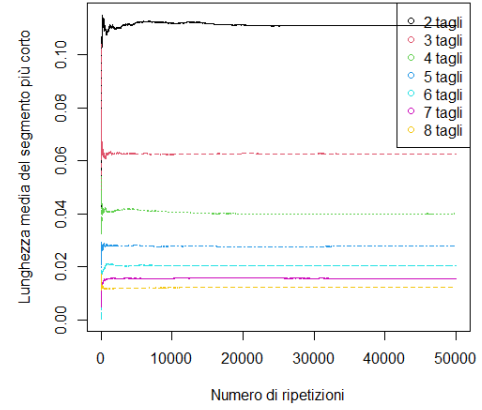


Figura 4: Grafico della lunghezza media del segmento più corto.

2.3 Commento

Come possiamo notare dai grafici generati e dalle tabelle mostrate sopra, i risultati ottenuti presentano un alto grado di somiglianza, permettendo di effettuare una stima di quelli che potrebbero essere i veri valori delle medie prese in esame (per comodità verranno considerati solo le approssimazioni dopo 2500, 5000, 10000, 25000 e 50000 ripetizioni dalla simulazione effettuata in R):

N. di tagli	2500 ripetizioni	5000 ripetizioni	10000 ripetizioni	25000 ripetizioni	50000 ripetizioni	Approssimazione
2	0.6148884	0.6115781	0.6131054	0.6127974	0.6125408	$E_{\max}^{(2)} \in [\frac{11}{18}, \frac{12}{18}]$
3	0.5205731	0.5208298	0.5208504	0.5206880	0.5210683	$E_{\max}^{(3)} \in [\frac{9}{18}, \frac{19}{36}]$
4	0.4521926	0.4533760	0.4544021	0.4564114	0.4561257	$E_{\max}^{(4)} \in [\frac{16}{36}, \frac{33}{72}]$
5	0.4084903	0.4075503	0.4094178	0.4093914	0.4087888	$E_{\max}^{(5)} \in [\frac{29}{72}, \frac{59}{144}]$
6	0.3702594	0.3694764	0.3695492	0.3701447	0.3706831	$E_{\max}^{(6)} \in [\frac{53}{144}, \frac{54}{144}]$
7	0.3417617	0.3404122	0.3402320	0.3405575	0.3404541	$E_{\max}^{(7)} \in [\frac{48}{144}, \frac{99}{288}]$
8	0.3174291	0.3157064	0.3148847	0.3143497	0.3140642	$E_{\max}^{(8)} \in [\frac{90}{288}, \frac{183}{576}]$

Tabella 1: Valori simulati con relativa approssimazione della lunghezza media del segmento più lungo.

N. di tagli	2500 ripetizioni	5000 ripetizioni	10000 ripetizioni	25000 ripetizioni	50000 ripetizioni	Approssimazione
2	0.10987499	0.11213826	0.11191859	0.11091813	0.11096490	$E_{\max}^{(8)} \in [\frac{1}{9}, \frac{1}{8}]$
3	0.06300079	0.06267810	0.06240760	0.06271635	0.06259380	$E_{\max}^{(8)} \in [\frac{1}{17}, \frac{1}{15}]$
4	0.04162267	0.04166360	0.04072910	0.03991147	0.04010663	$E_{\max}^{(8)} \in [\frac{1}{26}, \frac{1}{24}]$
5	0.02785043	0.02810729	0.02781651	0.02758278	0.02776478	$E_{\max}^{(8)} \in [\frac{1}{37}, \frac{1}{35}]$
6	0.02040449	0.02055592	0.02043586	0.02046256	0.02044115	$E_{\max}^{(8)} \in [\frac{1}{50}, \frac{1}{48}]$
7	0.01549501	0.01564508	0.01566362	0.01569236	0.01560599	$E_{\max}^{(8)} \in [\frac{1}{64}, \frac{1}{63}]$
8	0.01179953	0.01203343	0.01214537	0.01227710	0.01235044	$E_{\max}^{(8)} \in [\frac{1}{85}, \frac{1}{80}]$

Tabella 2: Valori simulati con relativa approssimazione della lunghezza media del segmento più corto.

A vista d'occhio si potrebbe presumere che tali valori siano rappresentabili tramite una formula, la quale però non è ricavabile al momento.

3 Seconda implementazione

Questa seconda simulazione lavora in modo simile alla prima, ma viene utilizzata per memorizzare un diverso set di dati: invece che calcolare il valore medio della lunghezza del segmento più lungo e del segmento più corto differenziando il numero di casistiche (ossia il numero di taglio per ogni singola simulazione), vogliamo calcolare come questi valori medi variano in base al numero di tagli. Tali valori verranno poi mostrati graficamente tramite due grafici, di cui uno verrà scalato logaritmicamente. Per questa simulazione verrà realizzata una sola versione in R.

3.1 Implementazione in R

```

1 tagli_max < 1000
2 ripetizioni_max < 1000
3
4 lista_valori_medi < matrix(0, nrow = 2, ncol = tagli_max - 1)
5 lista_valori_medi_log < matrix(0, nrow = 2, ncol = tagli_max - 1)
6
7 for (i in seq_len(ripetizioni_max)) {
8   for (j in 2:tagli_max) {
9     tagli < rep(c(0), each = j)
10    tagli < runif(j, min = 0, max = 1)
11    tagli < sort(tagli)
12
13    segmenti < c(0, tagli, 1)
14    segmenti < diff(segmenti)
15
16    segmento_max < max(segmenti, na.rm = TRUE)
17    segmento_min < min(segmenti, na.rm = TRUE)
18
19    lista_valori_medi[1, j - 1] < lista_valori_medi[1, j - 1] + segmento_max
20    lista_valori_medi[2, j - 1] < lista_valori_medi[2, j - 1] + segmento_min
21    lista_valori_medi_log[1, j - 1] < lista_valori_medi_log[1, j - 1] + segmento_max
22    lista_valori_medi_log[2, j - 1] < lista_valori_medi_log[2, j - 1] + segmento_min
23  }
24 }
```

```

25
26 lista_valori_medi <- lista_valori_medi / ripetizioni_max
27 lista_valori_medi_log <- lista_valori_medi_log / ripetizioni_max
28 lista_valori_medi_log <- log(lista_valori_medi_log)
29
30 par(mfrow = c(1, 2))
31
32 legenda <- c("Pezzo piú lungo", "Pezzo piú corto")
33 matplot(2:tagli_max, y = t(lista_valori_medi), type = c("l"), pch = 1, col = 1:7, xlab = "Numero
    di tagli", ylab = "Lunghezza media")
34 legend("topright", legend = legenda, col = 1:7, pch = 1)
35
36 legenda <- c("Pezzo piú lungo", "Pezzo piú corto")
37 matplot(2:tagli_max, y = t(lista_valori_medi_log), type = c("l"), pch = 1, col = 1:7, xlab = "
    Numero di tagli", ylab = "Lunghezza media (logaritmica)")
38 legend("topright", legend = legenda, col = 1:7, pch = 1)

```

Listing 5: Implementazione della simulazione in R

3.1.1 Esempio di esecuzione

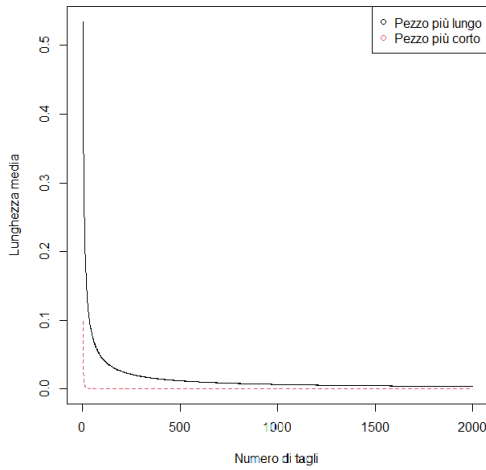


Figura 5: Grafico della lunghezza media del segmento piú lungo.

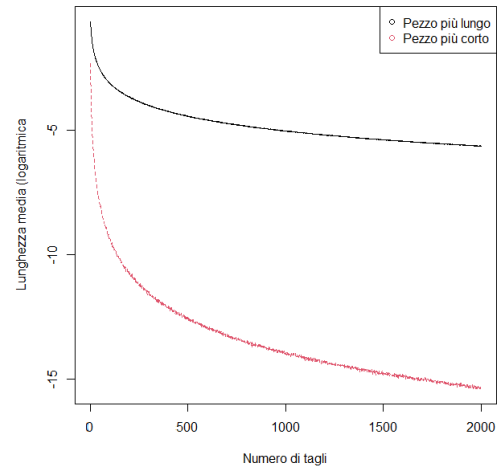


Figura 6: Grafico della lunghezza media del segmento piú corto.

3.2 Commento

Prima di tutto si vuole fare un commento sulla scelta dei valori di tagli massimi e ripetizioni massime: poiché siamo interessati all'andamento del valore delle lunghezze medie, e non ai loro valori esatti, possiamo considerare un'approssimazione meno precisa a favore di un maggior numero di tagli da poter trattare. Osservando appunto l'andamento dei valori medi, possiamo notare che segue un andamento esponenziale, con esponente pari ad una funzione decrescente a valori negativi, ossia:

$$y = ae^{bf(x)} + c, \text{ per qualche } a, b, c \in \mathbb{R}$$

Esempi di funzioni $f(x)$ decrescenti che assumono valori negativi sono:

- $f(x) \in \Theta(\coth(x))$, ossia $f(x)$ è una funzione iperbolica.
- $f(x) \in \Theta(-\sqrt[n]{x})$, ossia $f(x)$ è una generica funzione irrazionale decrescente.

- $f(x) \in \Theta(-\ln x)$, ossia $f(x)$ è una funzione logaritmica decrescente.

Notare che in tutte le funzioni sopra è presente il termine δ , usato per il decentramento della funzione rispetto all'asse x . Nel nostro caso la funzione che potrebbe avvicinarsi molto ai valori medi calcolati è:

$$y = a(x^{-b}e^d) + c, \text{ con } a, b, c, d \in \mathbb{R}$$

Questa funzione deriva dall'utilizzo di una $f(x)$ logaritmica decrescente (ossia $f(x) \in \Theta(-\ln x)$).