Stick Breaking Problem

02/05/2022

Studente: Emanuele Izzo *Matricola: 0307385*

Indice

1 Definizione del problema **1**
1.1 Nomenclatura 2

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

4 Terza implementazione **9**
4.1 Implementazione in R 9
4.2 Commento 11

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 unformemente 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 unformemente 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 effettaure 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.
- $X_1, X_2, ..., X_n$ 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 varaibili aleatorie X, Y).
- $E^{(n)}_{\max}$ rappresenta la variabile aleatoria della lunghezza del segmento più lungo.
- $E^{(n)}_{\min}$ 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 $E^{(n)}_{\max}$ (in base al numero di tagli effettuati) e uno per $E^{(n)}_{\min}$ (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

```python
import numpy as np
import matplotlib.pyplot as plt
import random

lista_tagli = [2, 3, 4, 5, 6, 7, 8]

lista_ripetizioni = [10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000, 25000, 50000]
lista_ripetizioni_s = [str(i) for i in lista_ripetizioni]

mappa_risultati_max = np.zeros((len(lista_tagli), len(lista_ripetizioni)))
mappa_risultati_min = np.zeros((len(lista_tagli), len(lista_ripetizioni)))

for i, num_tagli in enumerate(lista_tagli):
    for j, ripetizioni in enumerate(lista_ripetizioni):
        tagli = [0]*num_tagli

        for ripetizione in range(ripetizioni):
            for k in range(num_tagli):
                tagli[k] = random.random()

            for k in range(num_tagli 1):
                for h in range(0, num_tagli k 1):
                    if tagli[h] > tagli[h + 1] :
                        tagli[h], tagli[h + 1] = tagli[h + 1], tagli[h]

            segmenti = [0]*(num_tagli+1)

            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 piú 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 piú 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

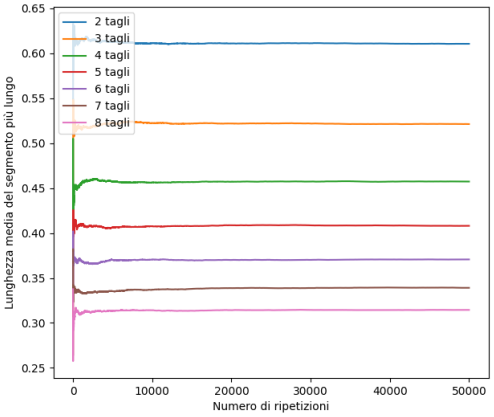

Figura 1: Grafico della lunghezza media del segmento più lungo.

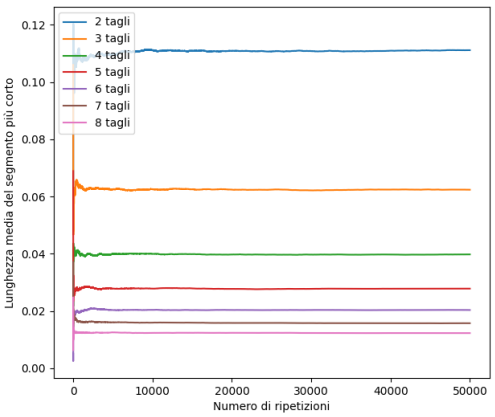

Figura 2: Grafico della lunghezza media del segmento più corto.

2.2 Implementazione in R

```r
lista_tagli <- c(2, 3, 4, 5, 6, 7, 8)
lista_ripetizioni <- c(10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000, 25000, 50000)
ripetizioni_max = lista_ripetizioni[length(lista_ripetizioni)]

mappa_risultati_max <- matrix(0, nrow = length(lista_tagli), ncol = length(lista_ripetizioni))
mappa_risultati_min <- matrix(0, nrow = length(lista_tagli), ncol = length(lista_ripetizioni))

lista_valori_medi_max <- matrix(0, nrow = length(lista_tagli), ncol = ripetizioni_max)
lista_valori_medi_min <- matrix(0, nrow = length(lista_tagli), ncol = ripetizioni_max)

for (i in seq_len(ripetizioni_max)) {
    for (j in seq_len(length(lista_tagli))) {
      num_tagli <- lista_tagli[j]

        tagli <- rep(c(0), each = num_tagli)
        tagli <- runif(num_tagli, min = 0, max = 1)
        tagli <- sort(tagli)

        segmenti <- c(0, tagli, 1)
        segmenti <- diff(segmenti)

        segmento_max <- max(segmenti, na.rm = TRUE)
        segmento_min <- min(segmenti, na.rm = TRUE)

        if (i >= 2) {
            lista_valori_medi_max[j, i] <- ((lista_valori_medi_max[j, i-1]*(i-1))+segmento_max)/(
    i)
```

```
            lista_valori_medi_min[j, i] < ((lista_valori_medi_min[j, i-1]*(i-1))+segmento_min)/(
    i)
      } else {
            lista_valori_medi_max[j, i] = segmento_max/i
            lista_valori_medi_min[j, i] = segmento_min/i
        }
    }

    if (i %in% lista_ripetizioni) {
        for (j in seq_len(length(lista_tagli))) {
            k < which(lista_ripetizioni == i)[[1]]

            mappa_risultati_max[j, k] < lista_valori_medi_max[j, i]
            mappa_risultati_min[j, k] < lista_valori_medi_min[j, i]
        }
    }
}

par(mfrow = c(1, 2))
legenda < c("2 tagli", "3 tagli", "4 tagli", "5 tagli", "6 tagli", "7 tagli", "8 tagli")

matplot(seq_len(ripetizioni_max), y = t(lista_valori_medi_max), type = c("l"), pch = 1, col = 1:7
    , xlab = "Numero di ripetizioni", ylab = "Lunghezza media del segmento piú lungo")
legend("topright", legend = legenda , col = 1:7, pch = 1)

matplot(seq_len(ripetizioni_max), y = t(lista_valori_medi_min), type = c("l"), pch = 1, col = 1:7
    , xlab = "Numero di ripetizioni", ylab = "Lunghezza media del segmento piú corto")
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

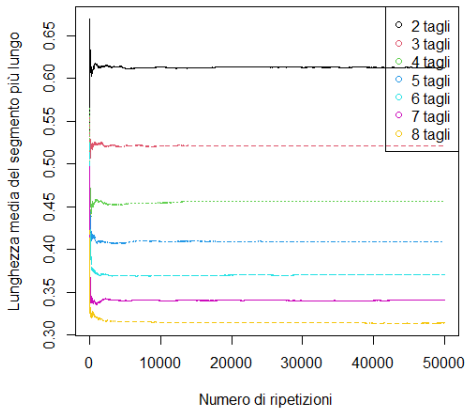

Figura 3: Grafico della lunghezza media del segmento più lungo.

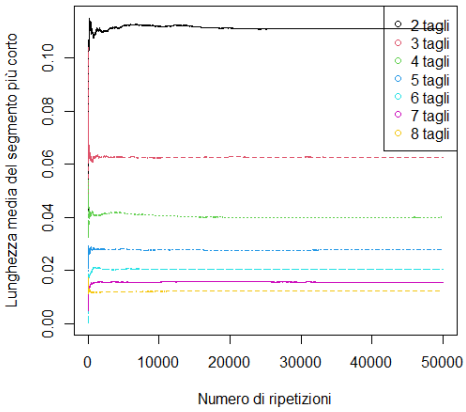

Figura 4: Grafico della lunghezza media del segmento più corto.

2.3 Commento

Prima di partire con l'analisi dei dati ottenuti, si vuole fare unj attimo un confronto tra le due implementazioni: se osserviamo i codici scritti, possiamo notare che R, rispetto a Python, permettere di trattare in modo più semplice sia le matrici che le rappresentazioni grafiche dei dati. Per tali motivazioni, le future implementazioni di simulazioni e codici saranno solo in R. Passando all'analisi dei dati, come possiamo notare dai grafici generati e dalle tabelle mostrate sopra, i risulati ottenuti presentano un alto grado di somiglianza, permettendo di effettaure 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 effettauta 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	$E^{(8)}_{\max} \in [\frac{1}{9}, \frac{1}{8}]$
3	0.06300079	0.06267810	0.06240760	0.06271635	0.06259380	$E^{(8)}_{\max} \in [\frac{1}{17}, \frac{1}{15}]$
4	0.04162267	0.04166360	0.04072910	0.03991147	0.04010663	$E^{(8)}_{\max} \in [\frac{1}{26}, \frac{1}{24}]$
5	0.02785043	0.02810729	0.02781651	0.02758278	0.02776478	$E^{(8)}_{\max} \in [\frac{1}{37}, \frac{1}{35}]$
6	0.02040449	0.02055592	0.02043586	0.02046256	0.02044115	$E^{(8)}_{\max} \in [\frac{1}{50}, \frac{1}{48}]$
7	0.01549501	0.01564508	0.01566362	0.01569236	0.01560599	$E^{(8)}_{\max} \in [\frac{1}{64}, \frac{1}{63}]$
8	0.01179953	0.01203343	0.01214537	0.01227710	0.01235044	$E^{(8)}_{\max} \in [\frac{1}{85}, \frac{1}{80}]$

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 $E^{(n)}_{\max}$ ed $E^{(n)}_{\min}$ differenziando il numero di casistiche (ossia fissando il numero di tagli per ogni singola simulazione), vogliamo calcolare come questi valori medi variano in base al numero di tagli (e quindi variando il 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

```
tagli_max <- 2000
ripetizioni_max <- 500

lista_valori_medi <- matrix(0, nrow = 2, ncol = tagli_max - 1)
lista_valori_medi_log <- matrix(0, nrow = 2, ncol = tagli_max - 1)

for (i in seq_len(ripetizioni_max)) {
    for (j in 2:tagli_max) {
        tagli <- rep(c(0), each = j)
        tagli <- runif(j, min = 0, max = 1)
        tagli <- sort(tagli)

        segmenti <- c(0, tagli, 1)
        segmenti <- diff(segmenti)

        segmento_max <- max(segmenti, na.rm = TRUE)
        segmento_min <- min(segmenti, na.rm = TRUE)

        lista_valori_medi[1, j - 1] <- lista_valori_medi[1, j - 1] + segmento_max
        lista_valori_medi[2, j - 1] <- lista_valori_medi[2, j - 1] + segmento_min
        lista_valori_medi_log[1, j - 1] <- lista_valori_medi_log[1, j - 1] + segmento_max
        lista_valori_medi_log[2, j - 1] <- lista_valori_medi_log[2, j - 1] + segmento_min
    }
}
```

```r

lista_valori_medi <- lista_valori_medi / ripetizioni_max
lista_valori_medi_log <- lista_valori_medi_log / ripetizioni_max
lista_valori_medi_log <- log(lista_valori_medi_log)

par(mfrow = c(1, 2))
x <- 2:tagli_max
x_log <- log(x)

legenda <- c("Pezzo piú lungo", "Pezzo piú corto")
matplot(x, y = t(lista_valori_medi), type = c("l"), pch = 1, col = 1:7, xlab = "Numero di tagli",
    ylab = "Lunghezza media")
legend("topright", legend = legenda , col = 1:7, pch = 1)

legenda <- c("Pezzo piú lungo", "Pezzo piú corto")
matplot(x_log, y = t(lista_valori_medi_log), type = c("l"), pch = 1, col = 1:7, xlab = "Numero di
    tagli", ylab = "Lunghezza media (logaritmica)")
legend("bottomleft", legend = legenda , col = 1:7, pch = 1)
```

Listing 5: Implementazione della simulazione in R

3.1.1 Esempio di esecuzione

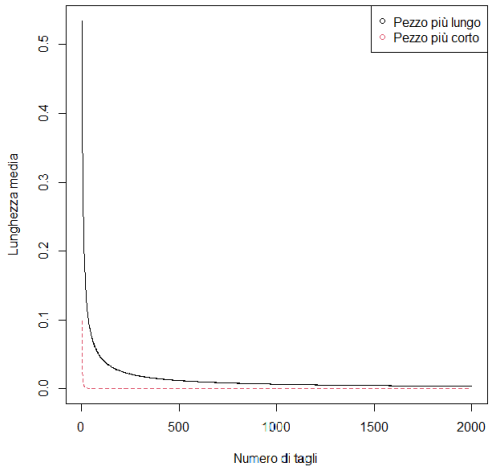

Figura 5: Grafico della lunghezza media del segmento più lungo e del più corto.

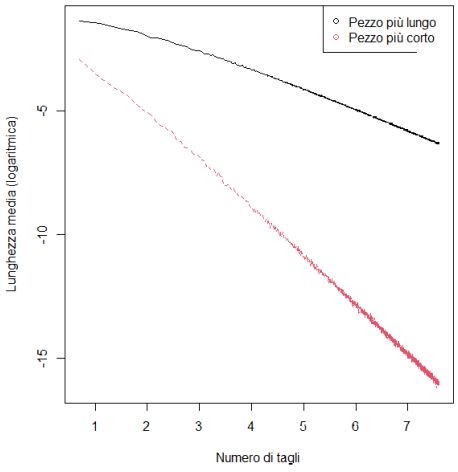

Figura 6: Grafico logaritmico della lunghezza media del segmento più lungo e del più corto.

3.2 Commento

Prima di tutto si vuole fare un commento sulla scelta dei valori di tagli massimi e ripetizioni massime: pihcé 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. Una cosa interessante che possiamo dedurre dai grafici è l'andamento dei valori medi: poniamo che la $E^{(n)}_{\max}$ e $E^{(n)}_{\min}$ seguono un andamento del tipo:

$$E^{(n)}_{\max} \cong ax^b \qquad \text{con } a, b \in \mathbb{R}$$
$$E^{(n)}_{\min} \cong cx^d \qquad \text{con } c, d \in \mathbb{R}$$

Se andiamo a rappresentare questi valori in un grafo logaritmico, l'andamento che dovremmo ottenere dovrebbe essere:

$$\begin{aligned}
\ln(E_{\max}^{(n)}) &\cong \ln(ax^b) = \\
&= \ln a + \ln x^b = \\
&= \ln a + b \ln x && \text{con } a, b \in \mathbb{R} \\
\ln(E_{\min}^{(n)}) &\cong \ln(cx^d) = \\
&= \ln c + \ln x^d = \\
&= \ln c + d \ln x && \text{con } c, d \in \mathbb{R}
\end{aligned}$$

E quindi tali valori dovrebbero seguire un andamento lineare. Se osserviamo il grafo logaritmico che abbiamo generato, possiamo osservare che questo andamento è visibile: in particolare, se effettuiamo una regressione lineare sui dati, non solo viene confermata la nostra ipotesi, ma siamo anche in grado di calcolare i valori dei coefficienti. Quello che dobbiamo fare è semplicemente effettaure una regressione lineare suii valori logaritmici delle medie.

4 Terza implementazione

Questa terza simulazione lavora in modo simile alla seconda, lavorando però solo sui valori logaritmici. In particolare, ciò che va a fare è una regressione lineare sui valori con relativa rappresentazione grafica. Tale implementazione verrà svolta solo in R.

4.1 Implementazione in R

```
tagli_max < 2000
ripetizioni_max < 500

lista_valori_medi_log < matrix(0, nrow = 2, ncol = tagli_max   1)

for (i in seq_len(ripetizioni_max)) {
    for (j in 2:tagli_max) {
        tagli < rep(c(0), each = j)
        tagli < runif(j, min = 0, max = 1)
        tagli < sort(tagli)

        segmenti < c(0, tagli, 1)
        segmenti < diff(segmenti)

        segmento_max < max(segmenti, na.rm = TRUE)
        segmento_min < min(segmenti, na.rm = TRUE)

        lista_valori_medi_log[1, j 1] < lista_valori_medi_log[1, j 1] + segmento_max
        lista_valori_medi_log[2, j 1] < lista_valori_medi_log[2, j 1] + segmento_min
    }
}

lista_valori_medi_log < lista_valori_medi_log / ripetizioni_max
lista_valori_medi_log < log(lista_valori_medi_log)

par(mfrow = c(1, 2))
x < 2:tagli_max
```

```
x_log < log(x)

modello_1 < lm(lista_valori_medi_log[1, 1:tagli_max 1] ~ x_log)
modello_2 < lm(lista_valori_medi_log[2, 1:tagli_max 1] ~ x_log)

legenda < c("Pezzo piú lungo", "Pezzo piú corto")
matplot(x_log, y = t(lista_valori_medi_log), type = c("l"), pch = 1, col = 1:7, xlab = "Numero di
    tagli", ylab = "Lunghezza media (logaritmica)")
abline(modello_1, col="blue")
abline(modello_2, col="blue")
legend("bottomleft", legend = legenda , col = 1:7, pch = 1)
```

Listing 6: Implementazione della simulazione in R

4.1.1 Esempio di esecuzione

```
Modello 1 (lunghezza media del segmento piú lungo):
lm(formula = lista_valori_medi_log[1, 1:tagli_max    1] ~ x_log)

Coefficients:
(Intercept)        x_log
    0.02554      0.82893

Modello 2 (lunghezza media del segmento piú corto):
Call:
lm(formula = lista_valori_medi_log[2, 1:tagli_max    1] ~ x_log)

Coefficients:
(Intercept)        x_log
     0.9049       1.9857
```

Listing 7: Esempio di esecuzione del programma

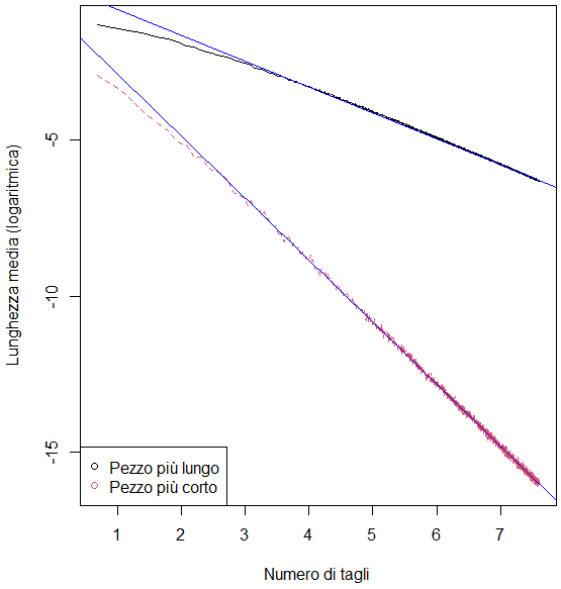

Figura 7: Grafico logaritmico della lunghezza media del segmento più lungo e del più corto, Viene mostrata in blu la linea di regressione dei valori.

4.2 Commento

Come discusso prima, i valori seguono un andamento lineare, nello specifico:

$$\ln(E_{\max}^{(n)}) \cong 0.02554 - 0.82893 \ln x =$$
$$\ln(E_{\min}^{(n)}) \cong -0.9049 - 1.9857 \ln x =$$

Poiché i valori $E_{\max}^{(n)}$ ed $E_{\min}^{(n)}$ sono espressi tramite logaritmo, è necessario rimuoverlo applicando ad entrambi i termini e^x:

$$\begin{aligned}
E_{\max}^{(n)} &\cong e^{0.02554 - 0.82893 \ln x} = \\
&= e^{-0.82893 \ln x} = \\
&= 1.02586 e^{\ln x^{-0.82893}} = \\
&= 1.02586 x^{-0.82893} \\
E_{\min}^{(n)} &\cong e^{-0.9049 - 1.9857 x} = \\
&= e^{-0.9049} e^{-1.9857 \ln x} = \\
&= 0.4045 e^{\ln x^{-1.9857}} = \\
&= 0.4045 x^{-1.9857}
\end{aligned}$$

Dove x indica il numero di tagli effettuali sul bastoncino. Va detto che è possibile approssimare ancora di più effettuando degli arrotondamenti sui coefficienti ottenuti:

$$E_{\max}^{(n)} \cong 1.02586 x^{-0.82893 \cong \frac{21}{20 x^{\frac{4}{5}}}}$$
$$E_{\min}^{(n)} \cong= 0.4045 x^{-1.9857} \cong \frac{1}{2x^2}$$

Bisogna notare ceh questa approssimazione diventa precisa/affidabile per x grande: per esempio, osservando il grafico sopra, preso $x \geq e^3 \cong 20$, l'approssimazione calcolata si avvicina moltissimo al valore esatto per entrambe le medie. Si può anche dire che le funzioni trovate sono dei lower bound alle due medie, ossia.

$$E_{\max}^{(n)} \in O(\frac{21}{20 x^{\frac{4}{5}}})$$
$$E_{\min}^{(n)} \in O(\frac{1}{2x^2})$$