

Documentatie proiect Probabilitati si Statistica

Membrii echipei

Cazacu Cristian – Gabriel
Tanislav Alexia
Velcea Mihnea-Andrei

Problema I

- I. Se consideră o activitate care presupune parcurgerea secvențială a n etape. Timpul necesar finalizării etapei i de către o persoană A este o variabilă aleatoare $T_i \sim \text{Exp}(\lambda_i)$. După finalizarea etapei i , A va trece în etapa $i+1$ cu probabilitatea α_i sau va opri lucrul cu probabilitatea $1 - \alpha_i$. Fie T timpul total petrecut de persoana A în realizarea activității respective.
- 1) Construiți un algoritm în R care simulează 10^6 valori pentru v.a. T și în baza acestora aproximați $E(T)$. Reprezentați grafic într-o manieră adecvată valorile obținute pentru T . Ce puteți spune despre repartiția lui T ?
 - 2) Calculați valoarea exactă a lui $E(T)$ și comparați cu valoarea obținută prin simulare.
 - 3) În baza simulărilor de la 1) aproximați probabilitatea ca persoana A să finalizeze activitatea.
 - 4) În baza simulărilor de la 1) aproximați probabilitatea ca persoana A să finalizeze activitatea într-un timp mai mic sau egal cu σ .
 - 5) În baza simulărilor de la 1) determinați timpul minim și respectiv timpul maxim în care persoana A finalizează activitatea și reprezentați grafic timpii de finalizare a activității din fiecare simulare. Ce puteți spune despre repartiția acestor timp de finalizare a activității?
 - 6) În baza simulărilor de la 1) aproximați probabilitatea ca persoana A să se oprească din lucru înainte de etapa k , unde $1 < k \leq n$. Reprezentați grafic probabilitățile obținute într-o manieră corespunzătoare. Ce puteți spune despre repartiția probabilităților obținute?

Cerința 1:

Explicarea cerinței: simulăm 1000000 valori pentru T , apoi calculăm media lor pentru a aproxima $E(T)$. Reprezentăm grafic valorile lui T .

Pentru a simula cele 1000000 valori ale lui T am ales un nr de etape $n = 10$ și valori diferite pentru α (probabilitatea de a trece la următoarea etapă) și λ (vor reprezenta parametrii pentru fiecare etapa a activității). De asemenea, am setat un seed pentru a genera același valori random de fiecare dată când rulăm programul.

Funcția `simulate_T()` va calcula timpul total T pentru activitate. Astfel, pentru o valoare a lui T , va trece prin toate cele n etape (dacă activitatea este finalizată), iar pentru fiecare etapă se va calcula timpul de finalizare a acelei etape cu ajutorul funcției `rexp()`. Funcția `rexp()` va genera un număr random, dar care să fie cât mai apropiat de $\frac{1}{\lambda_i}$ (lambda corespunde numărului etapei, acesta fiind rata evenimentelor pentru distribuția exponențială). Mai departe se verifică dacă numărul generat random între 0 și 1 de `runif()` este mai mare decât probabilitate α_i de a trece la următoarea etapă. În acest fel verificăm dacă activitatea se oprește la etapa i sau nu, la final returnând timpul total pentru îndeplinirea activității.

Pentru fiecare 1000000 valori a fost aplicată funcția *simulate_T()* și creat un vector cu acești timpi totali. Ca să aproximăm $E(T)$ am calculat media aritmetică a tuturor timpilor din T_values .

$$E(T) \approx \frac{1}{n} \sum_{i=1}^n T_i$$

n = numărul de simulări

T_i = valoarea lui T obținută la simularea i

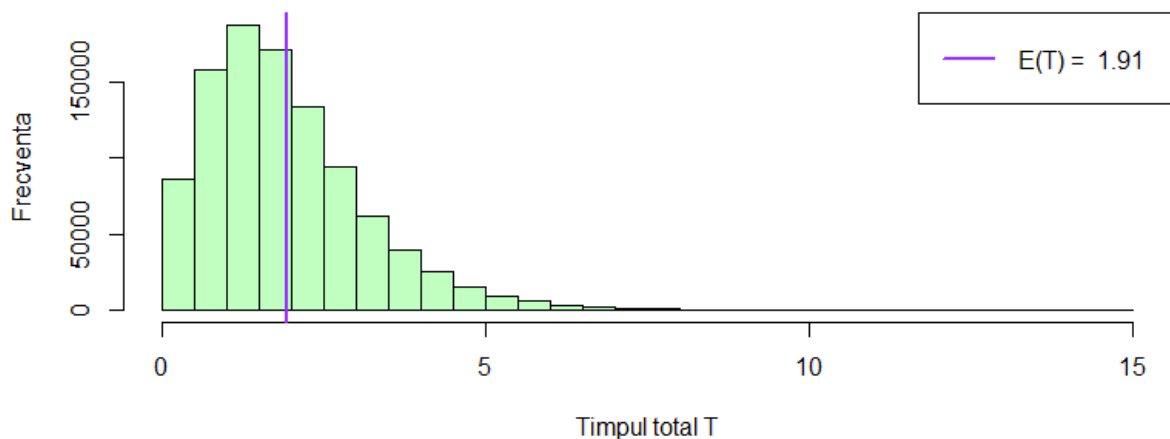
În continuare am reprezentat grafic valorile obținute printr-o histogramă cu ajutorul funcțiilor *hist()*, *abline()*, *legend()*.

```

1 set.seed(123) # pt aceleasi val random cand dai run
2 n <- 10 # nr de etape
3 lambda <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) # parametrii lambda pentru fiecare etapa
4 alpha <- c(0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05) # prob de trecere la etapa urm
5 # alfa de i este prob ca persoana A sa treaca de la etapa i la etapa i+1
6 # 1-alfa de i este prob ca persoana A sa se opreasca dupa etapa i
7 nrsim <- 1000000 # nr de simulari
8
9
10 ##### cerinta 1 #####
11
12 simulate_T <- function(n, lambda, alpha) {
13   total_time <- 0
14   for (i in 1:n) {
15     time <- rexp(1, rate = lambda[i]) # timp pentru etapa i
16     total_time <- total_time + time
17     if (runif(1) > alpha[i]) { # alege random un nr intre 0 si 1
18       break # stop dupa etapa i daca nr ales este mai mare decat prob alfa de i
19     }
20   }
21   return(total_time)
22 }
23
24 T_values <- replicate(nrsim, simulate_T(n, lambda, alpha))
25 # aplica functia/operatia simulate_T de 10^6 ori pt T
26 mean_T <- mean(T_values) # media ar a timpilor totali din simulari
27
28 # reprezentare grafica
29 hist(T_values, breaks = 50, main = "Distributia lui T", xlab = "Timpul total T", ylab = "Frecventa", col = "darkseagreen")
30 # daca freq=false arata probabilitatile
31 # breaks = nr bins/cosuri/drept alea verzi
32 abline(v = mean_T, col = "purple", lwd = 2)
33 # linia mov pt media lui T
34 legend("topright", legend = paste("E(T) = ", round(mean_T, 2)), col = "purple", lwd = 2)
35 # legenda pt linia mediei lui T
36 print(paste("Valoarea aproximata a lui E(T) =", mean_T))
37
38

```

Distributia lui T



```

> # Regula pe care o folosim
> print(paste("Valoarea aproximata a lui E(T) =", mean_T))
[1] "Valoarea aproximata a lui E(T) = 1.91413497263285"
>

```

Ce putem spune despre repartiția lui T ?

- T este o sumă de variabile aleatoare exponențiale cu probabilități de trecere între etape. Fiecare etapă i are un timp $T_i \sim \text{Exp}(\lambda_i)$, iar suma acestor timpi este condiționată de probabilitățile α_i .
- Repartiția lui T este asimetrică, deoarece timpul total poate fi foarte mare dacă persoana A parcurge multe etape, dar nu poate fi negativ.
- Are o coadă lungă la dreapta, deoarece există o probabilitate mică (dar nenulă) ca persoana A să parcurgă toate etapele.
- Media $E(T)$ este finită și poate fi calculată exact (așa cum am făcut la punctul 2) și ar trebui să fie apropiată de valoarea teoretică (valoarea exactă).
- Dacă valorile α_i sunt mici (probabilități mici de trecere la etapa următoare), persoana A se va opri rapid, iar T va avea valori mici.
- Repartiția lui T este condiționată de numărul de etape parcurse. De exemplu:
 - Dacă persoana A se oprește după prima etapă, $T \sim \text{Exp}(\lambda)$.
 - Dacă parcurge toate etapele, T este suma a n variabile exponențiale cu parametrii $\lambda_1, \lambda_2, \dots, \lambda_n$

Cerința 2:

Explicarea cerinței: calculăm exact valoarea lui $E(T)$ și o comparăm cu cea obținută prin simulare.

Ca să aflăm valoarea exactă a lui $E(T)$ vom folosi formula:

$$E(T) = \sum_{i=1}^n \left(\prod_{j=1}^{i-1} \alpha_j \right) \cdot \frac{1}{\lambda_i}$$

$$\prod_{j=1}^{i-1} \alpha_j = \text{produsul probabilităților de trecere de la etapa } i \text{ la } i+1$$

$$\frac{1}{\lambda_i} = \text{valoarea așteptată a timpului pentru etapa } i$$

Această formulă este dedusă din mai multe (alte formule) și anume:

- Valoarea așteptată a sumei de variabile aleatoare: $E(T) = E(T_1) + E(T_2) + \dots + E(T_k)$
- Probabilitatea de a ajunge la etapa i: $\prod_{j=1}^{i-1} \alpha_j$
- Contribuția fiecăruia la etapa i: $E(T_i) = \prod_{j=1}^{i-1} \alpha_j \cdot \frac{1}{\lambda_i}$
- Suma contribuțiilor: formula finală

Observăm ca cele doua rezultate obținute (valoarea aproximată și cea exactă) sunt foarte apropiate, diferența dintre ele fiind foarte mică.

```
39
40 ##### cerinta 2 #####
41
42 exact_ET <- 0
43 for (i in 1:n) {
44   prod_alpha <- if (i == 1) 1 else prod(alpha[1:(i-1)]) # produsul alpha1 * alpha2 * ... * alpha(i-1)
45   exact_ET <- exact_ET + prod_alpha * (1 / lambda[i])
46 }
47
48 print(paste("valoarea exacta a lui E(T) =", exact_ET))
49 print(paste("Dif între valoarea exacta și cea simulata =", abs(exact_ET - mean_T)))
50
51 |
52
53 ##### cerinta 3 #####

>
> print(paste("valoarea exacta a lui E(T) =", exact_ET))
[1] "valoarea exacta a lui E(T) = 1.913027488"
> print(paste("Dif între valoarea exacta și cea simulata =", abs(exact_ET - mean_T)))
[1] "Dif între valoarea exacta și cea simulata = 0.00110748463285004"
> |
```

Cerința 3:

Explicarea cerinței: aproximăm probabilitatea ca activitatea să fie finalizată (adică să ajungem la etapa n).

Am creat o funcție aproape identică cu cea de la cerința 1), dar în loc să returneze timpul total, aceasta va returna dacă activitatea a fost finalizată pentru toate cele 1000000 simulări ale lui T . Astfel, am creat un vector logic *comp* care înregistrează toate valorile de TRUE/FALSE (finalizat/nefinalizat). Probabilitatea ca se calculează folosind funcția *mean()* (media aritmetică).

```

72
73 > ##### cerinta 3 #####
74
75 simulate_T_final <- function(n, lambda, alpha) {
76   total_time <- 0
77   completed <- TRUE # presupun ca finalizeaza activitatea
78   for (i in 1:n) {
79     time <- rexp(1, rate = lambda[i]) # timp pentru etapa i
80     total_time <- total_time + time
81     if (runif(1) > alpha[i]) {
82       completed <- FALSE # nu a finalizat activitatea
83       break
84     }
85   }
86   return(completed) # returneaza TRUE daca a finalizat, FALSE altfel
87 }
88
89 comp <- replicate(nrsim, simulate_T_final(n, lambda, alpha))
90 prob_final <- mean(comp) # probabilitatea de finalizare
91 print(paste("Probabilitatea de finalizare a activitatii =", prob_final))
92

```

```

> comp <- replicate(nrsim, simulate_T_final(n, lambda, alpha))
> prob_final <- mean(comp) # probabilitatea de finalizare
> print(paste("Probabilitatea de finalizare a activitatii =", prob_final))
[1] "Probabilitatea de finalizare a activitatii = 2.1e-05"
> |

```

Cerința 4:

Explicarea cerinței: calculăm probabilitatea ca T să fie $\leq \sigma$ (unde σ este un prag dat).

Alegem o valoare pentru σ (în cazul nostru 5 pentru a cuprinde cât mai multe cazuri). Vectorul *T_values_sigma* va conține doar valorile timpilor a căror activitate a fost finalizată. Ca să calculăm probabilitatea cerută vom compara fiecare timp din *T_values_sigma* cu valoarea lui σ , returnând TRUE sau FALSE pentru fiecare comparație/valoare, iar cu ajutorul funcției *mean()* aceste valori de TRUE sunt numărate și mai apoi împărțite la numărul total de valori din vectorul *T_values_sigma*.

Probabilitatea/rezultatul nostru a fost 1 deoarece am ales o valoare pentru σ destul de mare pentru a include toate cazurile.

```

74
75 > ##### cerinta 4 #####
76
77 sigma <- 5 # valoarea lui sigma
78 # este ales mai mare sa acopere toate cazurile, deci prob o sa fie 1
79 T_values_sigma <- T_values[comp]
80 # filtreaza T_values folosind vectorul logic comp
81 # ia elementele de pe aceleasi pozitii iar daca in al doilea vector e true
82 # atunci pastreaza timpul lui T
83 prob_sigma <- mean(T_values_sigma <= sigma)
84 # compara fiecare T cu sigma si face med ar din val de true false <= sigma
85 print(paste("Probabilitatea ca T <= sigma:", prob_sigma))
86 |
87
88

```

```

> print(paste("Probabilitatea ca T <= sigma
[1] "Probabilitatea ca T <= sigma: 1"
> |

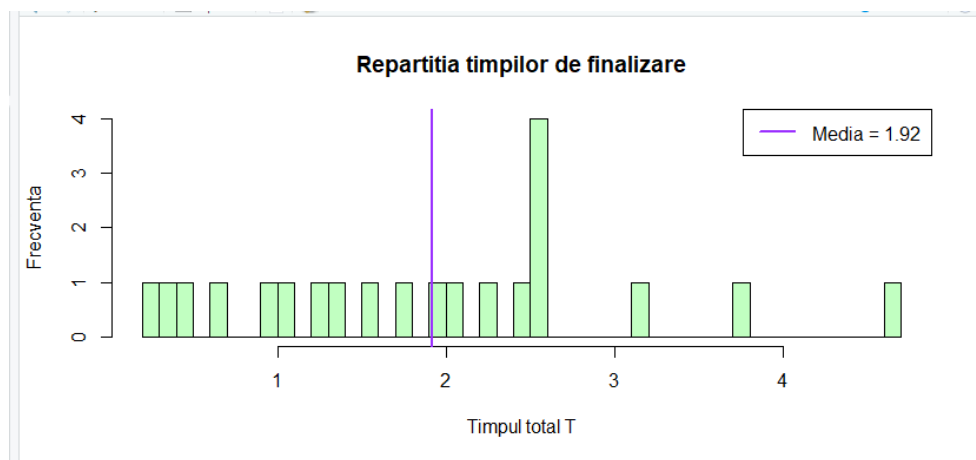
```

Cerința 5:

Explicarea cerinței: determinăm timpul minim și maxim în care se finalizează activitatea și reprezentăm grafic toți timpii obținuți.

Minimul/maximul este luat din vectorul creat la cerința anterioară și pus în variabila min_T/max_T . Aceste valori sunt reprezentate cu ajutorul unei histograme.

```
88
89 ##### cerinta 5 #####
90
91 min_T <- min(T_values_sigma)
92 max_T <- max(T_values_sigma)
93
94 print(paste("Timpul minim de finalizare=", min_T))
95 print(paste("Timpul maxim de finalizare=", max_T))
96
97 # reprezentare grafica a timpilor de finalizare
98 hist(T_values_sigma, breaks = 50, main = "Repartitia timpilor de finalizare", xlab = "Timpul total T", ylab = "Frecventa", col = "darkseagreen1")
99 abline(v = mean(T_values_sigma), col = "purple", lwd = 2) # linie pentru medie
100 legend("topright", legend = paste("Media =", round(mean(T_values_sigma), 2)), col = "purple", lwd = 2)
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



Ce putem spune despre repartiție?

- Distribuția este asimetrică, cu o coadă lungă la dreapta. Acest lucru înseamnă că există cazuri în care timpul de finalizare este mult mai mare decât media, dar acestea sunt mai puțin probabile.
- Vârful distribuției este în jurul valorii medii (1.92), ceea ce indică faptul că majoritatea timpilor de finalizare sunt concentrați în jurul acestei valori.
- Timpul minim de finalizare este aproape de 0, deoarece persoana A se poate opri foarte devreme (după prima etapă).

- Timpul maxim de finalizare este mai mare, reflectând cazurile în care persoana A parcurge toate etapele.

Cerința 6:

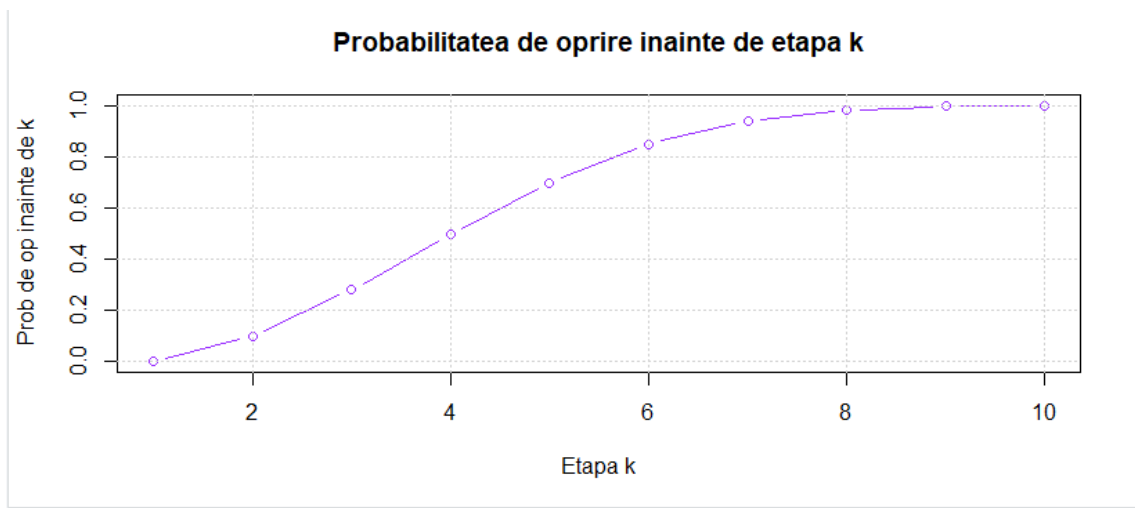
Explicarea cerinței: aproximăm probabilitatea ca activitatea să se oprească înainte de etapa k și reprezentăm aceste probabilități într-un grafic.

Am creat o funcție similară ca cea de la cerința 1), dar în loc să returneze timpul total, ea va returna etapa la care s-a oprit activitatea pentru fiecare T . Aceste etape sunt reținute într-un vector `stop_stages`. Ca să calculăm probabilitatea, am decis să iau pentru fiecare k . Astfel, vectorul `prob_stop_before_k` conține toate probabilitățile ca activitatea să se oprească înainte de k pentru fiecare k de la 1 la n . Reprezentarea grafică este dată de o linie, în loc de o histogramă.

```

103 ##### cerinta 6 #####
104
105 simulate_stop_stage <- function(n, lambda, alpha) {
106   for (i in 1:n) {
107     time <- rexp(1, rate = lambda[i])
108     if (runif(1) > alpha[i]) { # dacă se oprește după etapa i
109       return(i) # returnează etapa la care s-a oprit
110     }
111   }
112   return(n) # dacă a trecut prin toate etapele
113 }
114 # funcția care îți arată la ce etapa s-a oprit A
115
116 stop_stages <- replicate(nrsim, simulate_stop_stage(n, lambda, alpha))
117 # etapele la care s-a oprit A vector
118
119 prob_stop_before_k <- sapply(1:n, function(k) {
120   mean(stop_stages < k) # prob ca A să se oprească înainte de k
121 })
122 # se aplica pt toate k-urile de la 1 la n
123
124 # reprezentare grafică
125 plot(1:n, prob_stop_before_k, type = "b", col = "purple1",
126      xlab = "Etapa k", ylab = "Prob de op înainte de k",
127      main = "Probabilitatea de oprire înainte de etapa k")
128 grid()
129 |
130
131

```



Ce puteți spune despre repartiția probabilităților obținute?

- Aceste probabilități sunt crescătoare în raport cu k , deoarece persoana A are mai multe șanse de a se opri pe măsură ce parcurge mai multe etape.
- Probabilitățile formează o curbă crescătoare.
- La $k = 1$, probabilitatea este 0, deoarece persoana A nu poate să se oprească înainte de prima etapă.
- La $k = n$, probabilitatea este maximă, deoarece persoana A poate să se oprească în orice etapă până la n .
- Graficul arată o curbă monoton crescătoare, care reflectă faptul că probabilitatea de oprire crește odată cu creșterea lui k .
- Panta curbei depinde de probabilitățile α_i . Dacă valorile alfa sunt mici, panta este mai abruptă.

Concluzie:

În concluzie, am analizat un proces format din mai multe etape, unde fiecare etapă are un timp de execuție aleator și o probabilitate de a continua spre următoarea. Proiectul arată cum putem folosi simulările pentru a înțelege mai bine procesele aleatorii și comportamentul lor în diverse situații.

Dificultățile în realizarea cerințelor:

Începutul a fost foarte greu neștiind de unde să încep, iar înțelegerea cerinței a durat mai mult timp decât mă așteptam. Găsirea soluțiilor cerințelor nu a fost una tocmai ușoară, fiind nevoie de documentație specială care nu a fost tocmai ușor de înțeles. În ciuda acestor dificultăți, am reușit finalizarea proiectului.

Problema II

II. Construiti o aplicatie Shiny in care:

A. Sa reprezentati grafic functiile de repartitie pentru urmatoarele v.a:

1) $X, 3-2X, X^2, \sum_{i=1}^n X_i, \sum_{i=1}^n X_i^2$, unde $X, X_1, X_2 \dots X_n \text{ i.i.d. } \sim N(0,1), n \in \mathbb{N} \text{ fixat}$

2) $X, 3-2X, X^2, \sum_{i=1}^n X_i, \sum_{i=1}^n X_i^2$, unde $X, X_1, X_2 \dots X_n \text{ i.i.d. } \sim N(\mu, \sigma^2), \mu \in \mathbb{R}, \sigma > 0$
 $n \in \mathbb{N} \text{ fixat}$

3) $X, 2+5X, X^2, \sum_{i=1}^n X_i$, unde $X, X_1, X_2 \dots X_n \text{ i.i.d. } \sim \text{Exp}(\lambda), \lambda > 0, n \in \mathbb{N} \text{ fixat}$

4) $X, 3X-2, X^2, \sum_{i=1}^n X_i$, unde $X, X_1, X_2 \dots X_n \text{ i.i.d. } \sim \text{Pois}(\lambda), \lambda > 0, n \in \mathbb{N} \text{ fixat}$

5) $X, 5X-4, X^3, \sum_{i=1}^n X_i$, unde $X, X_1, X_2 \dots X_n \text{ i.i.d. } \sim \text{Binom}(r, p), r \in \mathbb{N}, p \in (0,1),$
 $n \in \mathbb{N} \text{ fixat}$

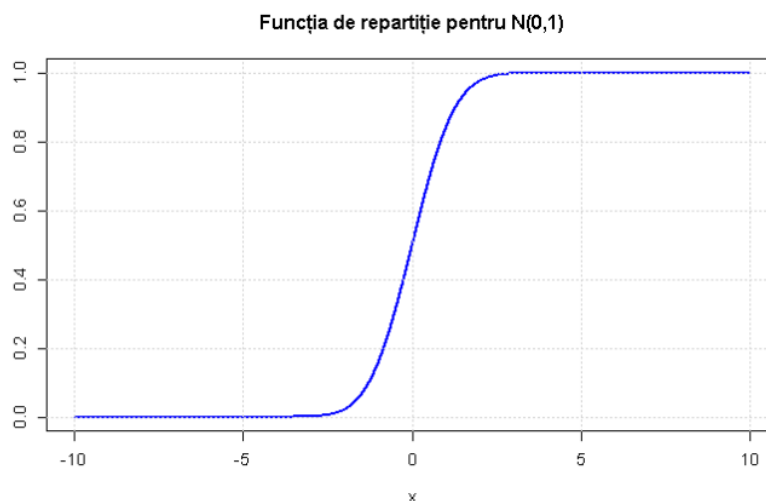
Funcția de repartitie a unei v.a este $F(x) = P(X \leq x)$.

1) X fiind $N(0,1)$

Pasii de rezolvare:

i) Pentru densitatile definite pe toata multimea numerelor reale vom discretiza un interval relevant pentru graficul functiei de repartie.

Ex:



Intrucat $F(x)$ pentru $N(0,1)$ incepe sa creasca din aproximativ -3 si se opreste in 3, este suficient sa surprind un interval apropiat de acesta pentru grafic, nu intregul domeniu.

Funcția R corespunzatoare: `x <- seq(-10, 10, 0.001)`

Intervalul trebuie discretizat deoarece $[-10,10]$ contine o infinitate de numere, iar un computer poate lucra numai cu o multime finita. Am ales **0.001** gradul de granularitate pentru a obtine un grafic cat mai precis.

Pentru a calcula functia de repartitie folosim: **pnorm(x, mean = 0, sd = 1) din R**, iar plotarea graficului o facem folosind functia **plot** ce primeste mai multe argumente pentru a personaliza graficul.

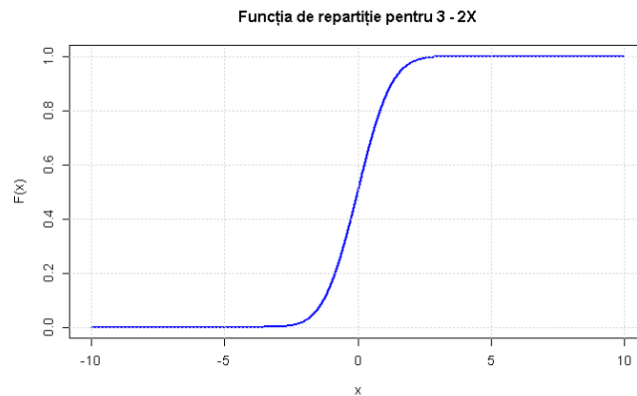
ii) $3 - 2X$, unde X este $N(0,1)$

$$Y = 3 - 2X, F(y) = P(Y \leq y) = P(3 - 2X \leq y) = P(X \geq (3 - y)/2) = 1 - F_X((3 - y)/2)$$

Pentru calcularea lui $Y = 3 - 2X$ ne vom folosi de functia de repartitie a lui F_X .

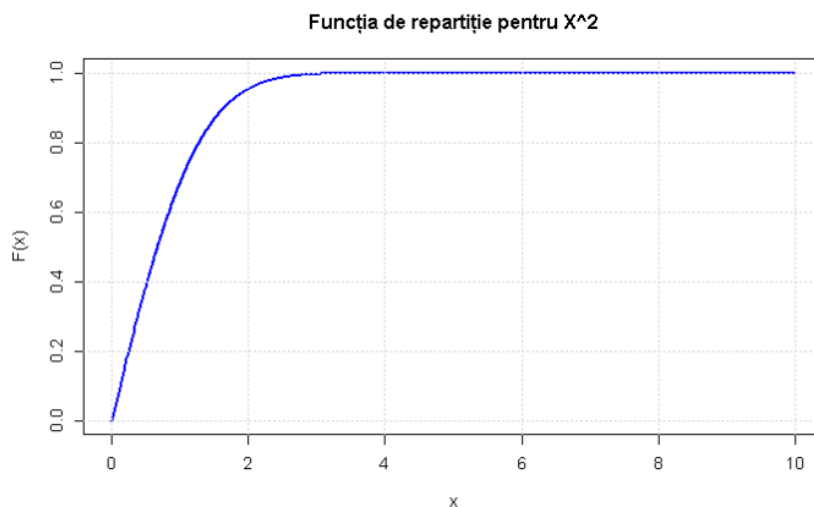
Daca domeniul pe care am reprezentat graficul lui X este $[-10,10]$, pentru a putea compara cu graficul lui Y cu X vom recalcula capetele intervalului.

$$3 - x / 2 = -10 \Leftrightarrow x = 23, 3 - x / 2 = 10 \Leftrightarrow x = -17 \Rightarrow \text{Noul domeniu este } [-17,23]$$



iii) X^2 , unde X este $N(0,1)$

$$Y = X^2, F(y) = P(Y \leq y) = P(X^2 \leq y) = P(-\sqrt{y} \leq X \leq \sqrt{y}) = F_X(\sqrt{y}) - F_X(-\sqrt{y})$$



iv) Pentru $X_1, X_2, X_3, \dots, X_n$ i.i.d de repartitie $N(0,1)$

X_i de repartitie $N(0,1)$ unde $N(\mu = 0, \sigma^2 = 1) \Rightarrow E[X_i] = 0, \text{Var}[X_i] = 1$

$$S_n = X_1 + X_2 + \dots + X_n$$

$$E[S_n] = E[X_1] + E[X_2] + \dots + E[X_n] \text{ (din proprietati)} = n * 0 = E[S_n]$$

$$\text{Var}[S_n] = \text{Var}[X_1] + \text{Var}[X_2] + \dots + \text{Var}[X_n] \text{ (sunt independente)} = n * 1 = \text{Var}[S_n]$$

Deci S_n este $N(0,n)$ si functia de repartitie o calculam cu **pnorm(x, mean = 0, sd = sqrt(n))**

v) Pentru $X_1, X_2, X_3, \dots, X_n$ i.i.d de repartitie $N(0,1)$

$$Q_n = X_1^2 + X_2^2 + \dots + X_n^2$$

$$\text{Var}[X_i] = E[X_i^2] - E[X_i]^2 \Rightarrow E[X_i^2] = \text{Var}[X_i] + E[X_i]^2 = 1 + 0^2 = 1 = E[X_i^2]$$

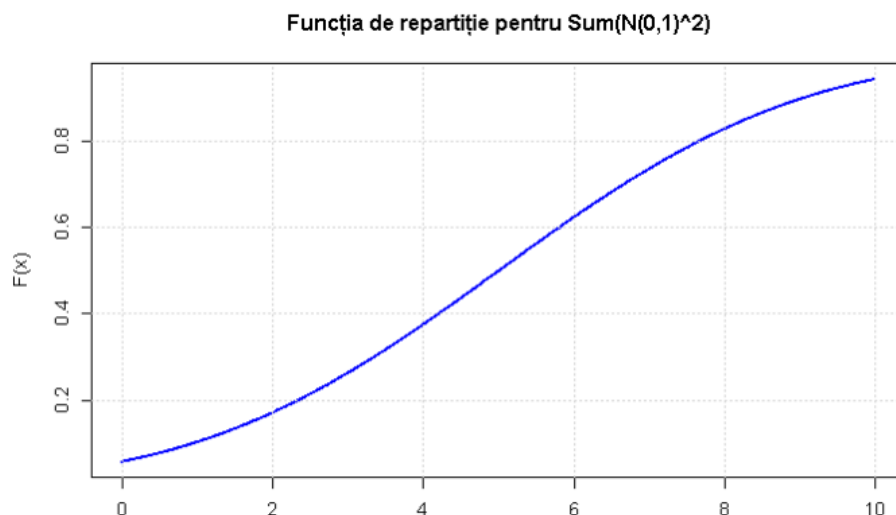
$$\text{Var}[X_i^2] = E[X_i^4] - E[X_i^2]^2 = 3 - 1 = 2 = \text{Var}[X_i^2]$$

Obs: Pentru distributia normala standard momentele pare sunt cunoscute $E[X^4] = 3$

$$E[Q_n] = E[X_1^2] + E[X_2^2] + \dots + E[X_n^2] = 1 * n = n = E[Q_n]$$

$$\text{Var}[Q_n] = \text{Var}[X_1^2] + \text{Var}[X_2^2] + \dots + \text{Var}[X_n^2] = 2 * n = \text{Var}[Q_n]$$

Deci Q_n este de distributie $N(2, 2n)$ si folosim **pnorm(x, n, sqrt(2*n))**



Grafic generat pentru $n = 5$

Obs: Tehnicile prezentate anterior se vor regasi si la subpunctele urmatoare asa ca voi adauga doar acele puncte care indica un nivel mai mare de dificultate.

2) X, X_1, X_2, \dots, X_n i.i.d de $N(\mu, \sigma^2)$

$$S_n = X_1 + X_2 + \dots + X_n$$

$$E[S_n] = E[X_1] + E[X_2] + \dots + E[X_n] = n * \mu$$

$$\text{Var}[S_n] = \text{Var}[X_1] + \text{Var}[X_2] + \dots + \text{Var}[X_n] = n * \sigma^2$$

Deci S_n are repartiția $N(n * \mu, n * \sigma^2)$

$$Q_n = X_1^2 + X_2^2 + \dots + X_n^2$$

$$E[X_i^2] = \text{Var}[X_i] + E[X_i]^2 = \sigma^2 + \mu^2$$

$$E[Q_n] = E[X_1^2] + E[X_2^2] + \dots + E[X_n^2] = n * \mu^2 + n * \sigma^2$$

$$[\text{Folosim formula } E[X^4] = 3\sigma^4 + 6\mu^2\sigma^2 + \mu^4]$$

$$\text{Var}[X_i^2] = E[X_i^4] - E[X_i^2]^2 =$$

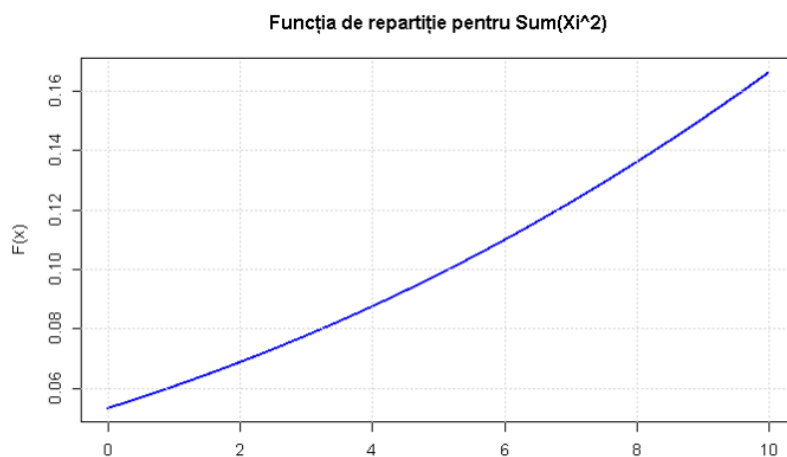
Obs: In continuare voi prescurta σ cu s

$$= 3s^4 + 6\mu^2s^2 + \mu^4 - s^4 - 2s^2\mu^2 - \mu^4 =$$

$$= 2s^4 + 4\mu^2s^2 = \text{Var}[X_i^2]$$

$$\text{Deci } \text{Var}[Q_n] = n(2s^4 + 4\mu^2s^2)$$

Concluzie: Q_n e de repartiție $N(n(\mu^2 + s^2), n(2s^4 + 4\mu^2s^2))$



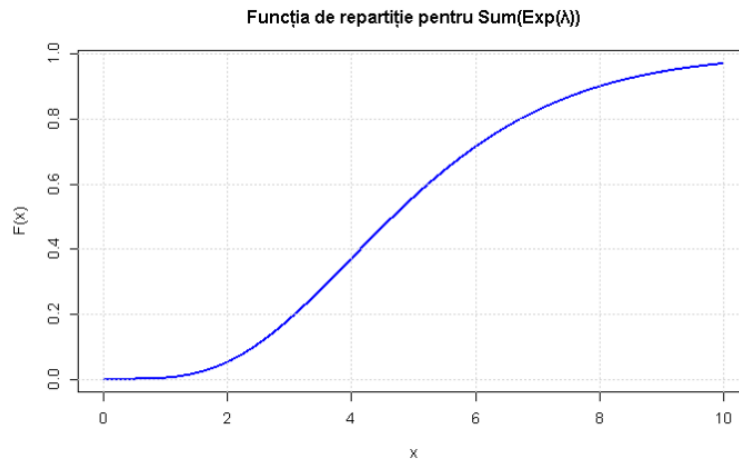
Grafic generat pentru $n = 5, \mu = 1, sd = 2$

3) Pentru subpunctul cu densitatea funcție **exponentiala** vom folosi pentru calcularea funcției de repartiție **funcția din R: pexp(x, rate = lambda)**.

In categoria aspectelor teoretice care depasesc nivelul_cursului_ si au ajutat la rezolvarea problemei voi incadra folosirea Transformatei LaPlace:

$$F(s) = \mathcal{L}\{f(t)\} = \int_{0^-}^{\infty} e^{-st} f(t) dt.$$

Valoarea transformatei cand $f(t)$ este pdf exponentiala este: $z = \lambda / (s + \lambda)$, iar pentru ca X_1, X_2, \dots, X_n sunt independente transformata lui $S_n = z^n$ este egala cu cea a functiei $\text{Gamma}(n, \lambda)$, motiv pentru care S_n are distributia $\text{Gamma}(n, \lambda)$ si vom folosi `pgamma(x, shape = n, rate = lambda)` pentru a calcula functia de repartitie.



Grafic generator pentru $n = 5$, $\lambda = 1$

- 4) Pentru a gasi repartitia lui S_n , atunci cand $X_i \sim \text{Poisson}(\lambda)$ ne vom folosi de alt concept din afara cursului: **functia generatoare de probabilitati (MGF).**

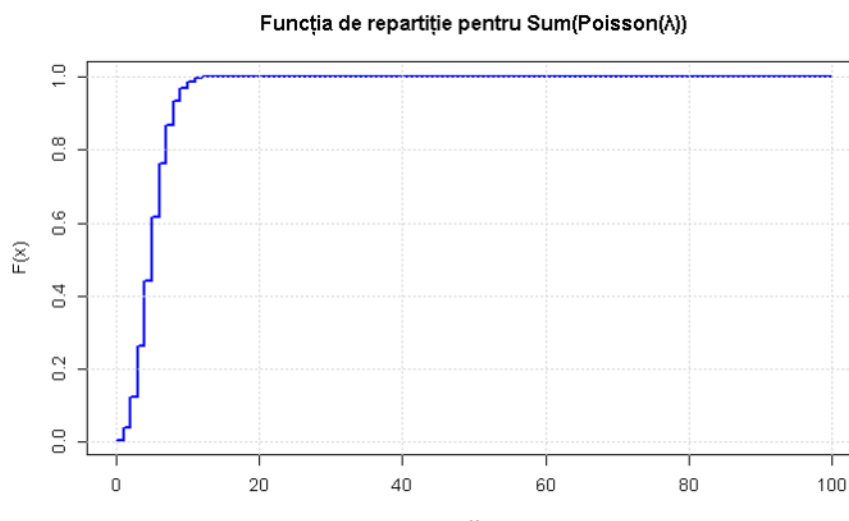
Pentru $X \sim \text{Poisson}(\lambda)$, $M_X(t) = e^{\lambda(e^t - 1)}$.

Pentru n variabile independente $X_i \sim \text{Poisson}(\lambda)$ vom avea:

$$M_{S_n}(t) = M_{X_1}(t) * M_{X_2}(t) * \dots * M_{X_n}(t) = e^{n\lambda(e^t - 1)},$$

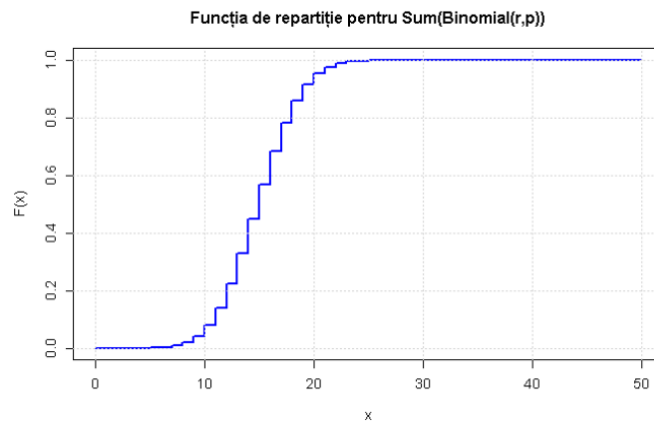
adica aceasta este functia generatoare de probabilitati a unei variabile $X \sim \text{Poisson}(n\lambda)$

In concluzie $S_n \sim \text{Poisson}(n\lambda)$.



Grafic generat pentru $n = 5$, $\lambda = 1$

- 5) Pentru repartitia lui S_n , atunci cand $X_i \sim \text{Binomial}(r, p)$, vom alege un rationament mai intuitiv. Daca o variabila $X \sim \text{Binomial}(r, p)$ reprezinta numarul de succese din r incercari, atunci daca avem n astfel de variabile, suma lor va reprezenta numarul de succese din $n \cdot r$ incercari, astfel $S_n \sim \text{Binomial}(n \cdot r, p)$.



Grafic generat pentru $n = 5, r = 10, p = 0.3$

In continuare vom discuta rezolvarile pentru partea a doua a problemei.

- B. In aplicatie constuiti cate o functie in R care afiseaza functia pentru parametri particularizabili de catre utilizator si calculeaza, media si varianta pentru v.a X definita:

a) $f(x) = cx^4, x \in (0, 2), c \in \mathbb{R}$

Pentru aceasta problema nu putem sa alegem noi constanta c pentru ca f este desitate de probabilitate cu proprietatile:

1. $f(x) \geq 0$, oricare x din $(0, 2) \Rightarrow c \geq 0$
2. $\int_{-\infty}^{\infty} f(x) dx = 1 \Rightarrow \int_{(0, 2)} cx^4 = 1 \Rightarrow c = 5/32$

In cadrul aplicatie am lasat utilizatorul sa introduca ce valori considera pentru c , dar aplicatia va computa media si varianta, doar pentru densitatea valida.

Problema propune scrierea unei functii care sa afiseze functia si sa calculeze media, respectiv varianta. Pentru functiile de la aceasta problema am decis sa compartimentez functia in 3 functii: **una pentru afisare, una pentru calcularea mediei si variantei si una pentru afisarea rezultatelor**. Aceasta abordare ajuta la o citire si urmarire a programului mai usoara.

i) Afisarea functiei:

Scriem functia ce trebuie plotata folosind keyword-ul **function(x)** .

```
f <- function(x) input$numerators/input$denominators * x^4
```

Obs: Am ales sa impart c in numarator si impartitor pentru accesibilitate, pentru a fi mai usor introdus in interfata aplicatiei.

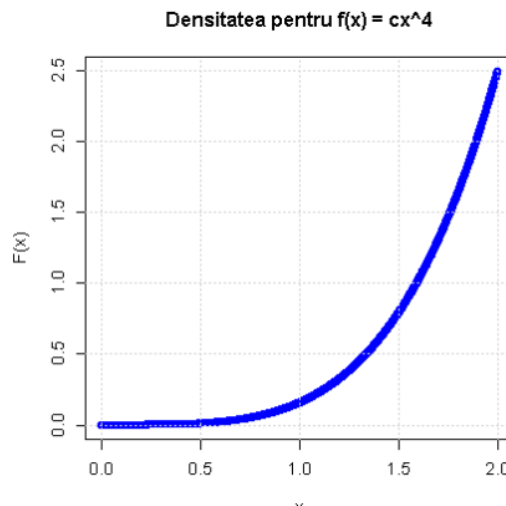
Tinand cont ca functia este definita pe intervalul (0,2), asa ca vom folosi din nou pasul de distretizare, utilizat si la problema cu functiile de repartitie.

```
interval <- seq(0.001, 1.999, 0.001)
```

Inainte de plotare vom aplica functia pe valorile intervalului folosind functia **sapply**.

```
f_x <- sapply(interval, f)
```

In cele din urma folosim functia **plot(interval, f_x ...)** pentru a afisa graficul.



Graficul functiei f

cand $c = 5/32$

ii) Calcularea mediei si variantei

Media $E[X] = \int_0^2 x \cdot f(x) dx$

Varianța $Var[X] = E[X^2] - E[X]^2$

$E[X^2] = \int_0^2 x^2 \cdot f(x) dx$

Functia pe care o vom folosi pentru a calcula integralele in R este **integrate**.

Ex (din cod):

```
#Medie E(x)
```

```
average <- integrate(function(x) x * f_x_A(x), 0, 2)$value unde:
```

- f_{x_A} este densitatea variabilei care este înmulțită cu x pentru calcularea mediei
- $0, 2$ este intervalul de integrare

Media patratică se calculează analog, iar varianta se calculează prin scădere între cele două, respectând formula.

iii) Verificăm dacă densitatea este validă.

Calculăm valoarea integralei pe tot domeniul pentru densitate:

#Integrala de validare

`integral_f <- integrate(f_x_A(), 0, 2)$value`, unde f_{x_A} este densitatea de prob.

Pentru o densitate corectă ea va fi aproximativ 1.

Înainte de afișare testăm validitatea densității:

`if (abs(stats$integral_f - 1) > 1e-6)`, făcând abstracție de stats, verificăm dacă integrala este aproximativ 1 luând în considerare eroare de rotunjire (poate fi 0.999999).

În continuare plotăm rezultate folosind funcția `cat`.

```
Media E[X] = 1.666667
Varianța Var(X) = 0.07936508
```

Media și varianta pt $5/32 \cdot x^4$

```
C nu face ca f(x) să fie o densitate validă!
Integrală = 1.2
```

Mesaj de eroare pentru densitatea invalidă

Pentru subpunctele următoare conceptele prezentate anterior vor fi din nou folosite, cu mici modificări, motiv pentru care voi detalia doar problemele care adaugă un nivel de noutate sau sunt mai dificile.

$$b) f(x) = ax + bx^2, \quad 0 < x < 1, \quad a, b \in \mathbb{R}$$

În cadrul acestui punct, în funcție de a și b , există mai multe densități valide.

Ne vom folosi din nou de proprietățile densității de probabilitate:

- $ax + bx^2 \geq 0$, oricare x din $(0,1)$

Avem cazurile:

$a > 0, b > 0$, at pt orice a, b $f(x) > 0$

Cazuri favorabile: $a = 0, b \geq 0$; $b = 0, a \geq 0$

Cazuri nefavorabile: $a = 0, b < 0$; $b = 0, a < 0$; $a < 0, b < 0$

Raman cazurile $a > 0, b < 0$ si $a < 0, b > 0$

I. $a > 0, b < 0, x$ din $(0,1)$

$$ax + bx^2 \geq 0 \Rightarrow x(a + bx) \geq 0 \Rightarrow a + bx \geq 0 \Leftrightarrow x \leq -a/b \Leftrightarrow 1 \leq -a/b \Leftrightarrow a \geq -b$$

Vrem ca $-a/b \geq x$ pe tot intervalul $(0,1)$, deci e suficient sa fie ≥ 1

II. $a < 0, b > 0, x$ din $(0,1)$

$$ax + bx^2 \geq 0 \Rightarrow x(a + bx) \geq 0 \Rightarrow a + bx \geq 0 \Leftrightarrow x \geq -a/b \Leftrightarrow 0 \geq -a/b \Leftrightarrow -a < 0 \text{ imposibil } a < 0$$

Vrem ca $x \geq -a/b$ pe tot domeniul $(0,1)$, deci e suficient sa comparam cu minimul

- Integrala $(0,1) (ax + bx^2) dx = 1$, din calcule $\Rightarrow ax^2/2 + bx^3/3|_{(0,1)} = 1 \Leftrightarrow a/2 + b/3 = 1 \Leftrightarrow a = (6-2b)/3$

Folosind noile conditii de la $f(x) \geq 0$ le vom adauga la calculul integralei pe domeniu.

```
if (abs(stats$integral_f - 1) > 1e-6 & check) {  
  cat("A, B nu fac ca f(x) să fie o densitate validă!\n")  
  cat("Integrală =", stats$integral_f, "\n")  
  check <- FALSE  
}  
  
if(check & a < 0){  
  cat("A, B nu fac ca f(x) să fie o densitate validă!\n")  
  check <- FALSE  
}  
  
if(check & b < 0){  
  if(a == 0 | a <= 0){  
    cat("A, B nu fac ca f(x) să fie o densitate validă!\n")  
    check <- FALSE  
  }  
  else{  
    if(a < -b){  
      {  
        cat("A, B nu fac ca f(x) să fie o densitate validă!\n")  
        check <- FALSE  
      }  
    }  
  }  
}
```

Cod care verifica $f(x) \geq 0$, pt a, b alesi de utilizator

$$c) f(x) = \frac{4}{x(x+1)(x+2)}, x \in \mathbb{N}^*$$

Pentru punctul c), avem prima v.a discreta. Vom calcula media si varianta astfel:

$$E[X] = x * p(x), \text{ unde } f \text{ este pdf pentru } X$$

$$E[X^2] = x^2 * p(x)$$

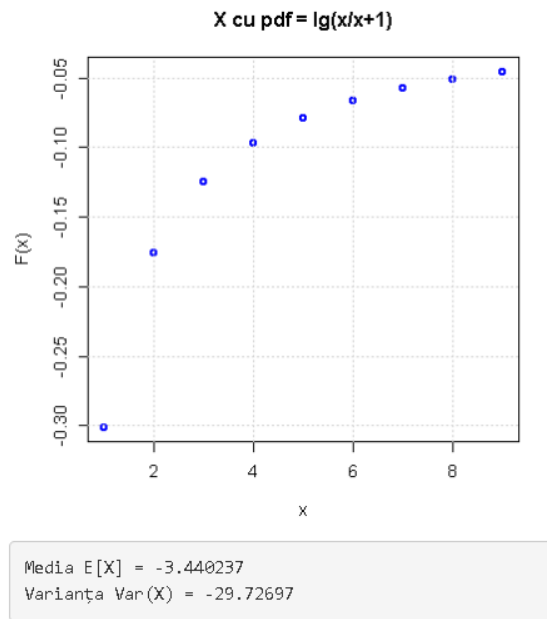
$$\text{Var}[X] = E[X^2] - E[X]^2$$

Pentru ca nu putem lucra pe domenii infinite, voi alege doar un numar foarte mare pe care voi reprezenta functia.

Pe acest domeniu, aplicam pdf si formulele prezentate anterior si afisam rezultatele fara a mai verifica conditii extra.

$$d) f(x) = \lg\left(\frac{x}{x+1}\right), x \in \{1, 2, \dots, 9\}$$

Punctul d) se rezolva identic cu c), pentru ca avem o v.a discreta si primim pdf.



Grafic pdf pentru punctul d)

Obs: La aceste subpuncte nu am gasit alternative pentru a oferi utilizatorului optiunea de a selecta variabile.

$$e) f(x) = \frac{\theta^2}{1+\theta} (1+x)e^{-\theta x}, x > 0, \theta > 0$$

Punctul e) se rezolva la fel cu a), b) verificand proprietatile densitatii de probabilitate. Pentru e) vom demonstra ca pentru oricare teta > 0(notat „o” pentru o mai usoara scriere) ne ofera o densitate de probabilitate valida.

f densitate de probabilitate =>

- i) $f(x) \geq 0$, pt oricare x, ceea ce este adevarat pt ca primul termen este mereu pozitiv, paranteza deasemenea si exponentiala la fel.
- ii) $\int_0^{\infty} f(x)dx = 1$

$$\int_0^{\infty} \frac{\theta^2}{1+\theta} (1+x) e^{-\theta x} dx = 1 \Leftrightarrow \frac{\theta^2}{1+\theta} \int_0^{\infty} (1+x) e^{-\theta x} dx = 1 \Leftrightarrow$$

$$\Leftrightarrow \frac{\theta^2}{1+\theta} \left(\int_0^{\infty} e^{-\theta x} dx + \int_0^{\infty} x \cdot e^{-\theta x} dx \right) = 1 \quad (*)$$

$$\int_0^{\infty} e^{-\theta x} dx = \left[-\frac{1}{\theta} e^{-\theta x} \right]_0^{\infty} = -\frac{1}{\theta} \cdot e^{-\theta x} \Big|_0^{\infty} = -\frac{1}{\theta} (0 - 1) = \frac{1}{\theta}$$

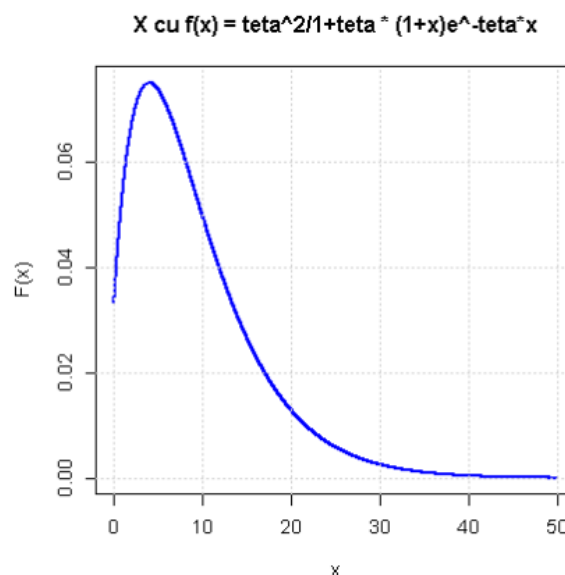
$$\int_0^{\infty} x \cdot e^{-\theta x} dx = \int_0^{\infty} x \cdot \left(-\frac{1}{\theta} \cdot e^{-\theta x} \right)' dx = -\frac{1}{\theta} \cdot x \cdot e^{-\theta x} \Big|_0^{\infty} + \frac{1}{\theta} \int_0^{\infty} e^{-\theta x} dx =$$

$$= 0 - 0 + \frac{1}{\theta} \cdot \frac{1}{\theta} = \frac{1}{\theta^2} \quad \text{? Putem folosi L'Hôpital}$$

$$\text{În (*) : } \frac{\theta^2}{1+\theta} \left(\int_0^{\infty} e^{-\theta x} dx + \int_0^{\infty} x \cdot e^{-\theta x} dx \right) = 1 \Leftrightarrow \frac{\theta^2}{1+\theta} \left(\frac{1}{\theta} + \frac{1}{\theta^2} \right) = 1$$

$$\Leftrightarrow \frac{\theta^2}{1+\theta} \cdot \frac{1+\theta}{\theta^2} = 1 \quad \textcircled{A} \text{ pt } \forall \theta > 0, \forall x > 0$$

Din demonstrație concluzionăm că pentru oricare θ ales de utilizator densitatea rămâne validă. Am verificat asta prin refolosirea funcției care calculează integrala pe domeniu și o compară cu 1. Calcularea mediei și varianței se face la fel în punctele anterioare prin integrare pe domeniu și folosirea formulelor.



Media $E[X] = 9.166667$
 Varianța $Var(X) = 49.30556$

Graficul densității e cu $\theta = 0.2$

$$f) f(x) = \begin{cases} \frac{1}{3}e^x, & x < 0 \\ \frac{1}{3}, & 0 \leq x < 1 \\ \frac{1}{3}e^{-(x-1)}, & x \geq 1 \end{cases}$$

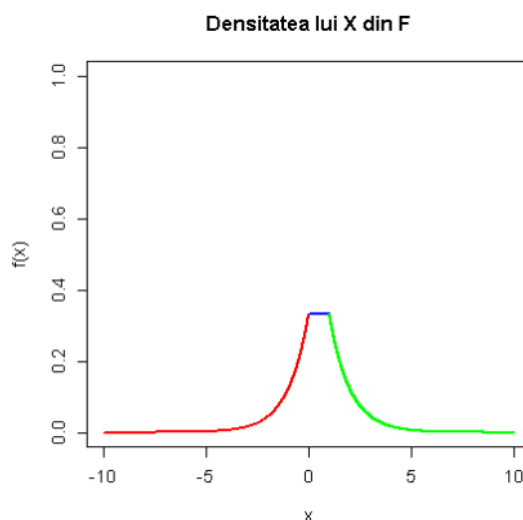
Pentru densitatea f), avem de fapt 3 densitati pe 3 intervale diferite. Pentru a calcula media vom calcula media pe fiecare dintre intervale folosind tehnicile prezentate anterior. Dupa ce le obtinem facem suma ponderata cu probabilitatile pe fiecare interval.

$$E[X] = P1 * E[X](-\text{Inf},0) + P2 * E[X][0,1) + P3 * E[X][1,\text{Inf}],$$

unde P_i este probabilitatea ca x sa fie in intervalul i .

Vom calcula la fel $E[X^2]$ si Varianta folosind formula $E[X^2] - E[X]^2$.

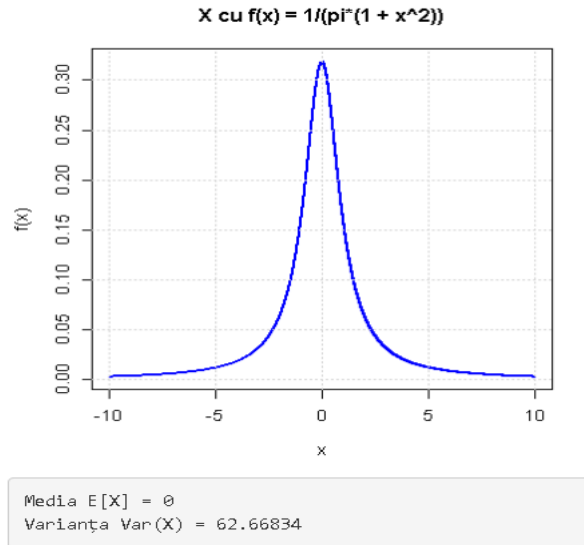
Pentru plotarea graficului pe ramuri am generat initial un grafic care se muleaza pe 0x pe tot domeniul si am adaugat ca si linii cele 3 densitati folosind functia **lines**.



Media $E[X] = 0.166667$
 Varianta $\text{Var}(X) = 0.787037$

$$g) f(x) = \frac{1}{\pi(1+x^2)}$$

Pentru distributia Cauchy media si varianta nu sunt definite in sensul clasic, ele fiind nelimitate. Densitatea este simetrica fata de 0, ceea ce face ca media teoretica sa fie 0, iar integrala pentru calculul variantei este infinita(sau divergenta) motiv pentru care am considerat luarea unui esantion semnificativ din distributie care sa captureze majoritatea valorilor importante. Am decis asupra intervalului $[-30,30]$. Ca si atunci cand este nevoie sa discretizam un interval infinit, R nu poate sa calculeze o integrala pe o infinitate de valori.



Graficul densitatii Cauchy

Cum functioneaza Shiny?

Singura biblioteca extra folosita in cadrul proiectului este „**shiny**” importata folosind **library(shiny)**.

In crearea unei aplicatii shiny vom jongla constant intre cele 2 partii ale ei: componenta UI(user interface), adica ceea ce vede si foloseste utilizatorul pentru a interactiona si partea de Server care se ocupa de calculele din spatele rezultatelor afisate.

Cele mai folosite elemente de UI folosite in cadrul proiectului au fost **titlePanel**, **sidebarLayout**, **sidebarPanel**, **mainPanel**, **plotOutput**, **sliderInput**, **numericInput**, **verbatimTextOutput**.

Titlul si componentele de layout si panourile genereaza structura aplicatiei, respectiv modul in care este organizata.

SliderInput, numericInput permit utilizatorului sa modifice variabilele aplicatie si sunt capturate in componenta de Server folosind sintaxa „input\$numericInput”.

PlotOutput si verbatimTextOutput permit afisarea graficelor si a mesajelor text.

Conventie: Pentru organizarea in componenta UI a componentelor pentru fiecare task am folosit comentarii cu rolul de a indruma utilizator.

Ex:

```
#UI Side Normal(0,1)
#####
titlePanel("Funcția de repartiție a unei variabile normale standard"),
sidebarLayout(
  sidebarPanel(),
  mainPanel(
    plotOutput("normalStdPlot"),
    plotOutput("transformedStdPlot"),
    plotOutput("squaredStdPlot")
  )
),
sidebarLayout(
  sidebarPanel(
    sliderInput("stdN", "Alege numărul de variabile", min = 1, max = 10, value = 1, step = 1)
  ),
  mainPanel(
    plotOutput("sumStd"),
    plotOutput("sumSquaredStd")
  )
),
#####
```

Blocul de cod a fost delimitat și deasupra vedem utilitatea acestuia, aici fiind prezente componentele UI pentru Ex1, plotarea graficelor pentru normala standard.

PlotOutput au parteneri în componenta Server care le indică ce să afișeze folosind numele din paranteze.

Componenta Server

Ex:

```
#Server Side Normal(0,1)
#####

output$normalStdPlot <- renderPlot({
  x <- seq(-10, 10, 0.001)
  y <- pnorm(x, mean = 0, sd = 1)
  plot(x, y, type = "l", col = "blue", lwd = 2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru N(0,1)"))
  grid()
})

# ?Pentru Y = 3-2X, P(y)=P(3-2X<=y)=P(X>=(3-y)/2) = 1 - P(X<=(3-y)/2)
output$transformedStdPlot <- renderPlot({
  x <- seq(-17, 23, 0.001)
  transformedX <- (3-x)/2
  y <- 1- pnorm(transformedX, mean = 0, sd = 1)
  plot(-transformedX,y, type = "l", col = "blue", lwd = 2,
        xlab = "x", ylab = "F(x)",
        main = paste("Funcția de repartiție pentru 3 - 2X"))
  grid()
})
```

Putem observa că convenția se păstrează și pentru partenerii de pe partea de server.

Output\$numeGrafic decide ce se va plota in componenta ui. Am grupat graficele in functie de repartia comuna. Toate graficele pentru normala standard sunt delimitate de marcatori folosind comentarii.

Dificultati in realizarea problemelor:

- Dificultati la rezolvarea seriilor de variabile aleatoare(Poisson, Exponentiala)
- Aspecte teoretice legate de distributia Cauchy

Concluzii:

- Aplicatiile Shiny pot fi folosite pentru a oferi o intelegere intuitiva a lucrului cu variabile aleatoare prin posibilitatea de a modifica parametrii ai acestora dupa bunul plac.
- Desi v.a au o multitudine de repartitii, prelucrarea acestora este foarte similara.

Problema III

3) Pentru v.a. X definită prin densitatea/funcția de masă de mai jos estimați parametrul θ prin metoda verosimilității maxime și prin metoda momentelor (pe foaie!), apoi construiți o funcție în R care preia eșantionul dat și întoarce estimațiile realizate în baza celor 2 estimatori. Comparați estimația dată de metoda verosimilității maxime cu valoarea lui θ dată de o metodă numerică.

DESCRIEREA PROBLEMEI

Problema constă în estimarea unui parametru necunoscut, notat cu θ , pentru o variabilă aleatoare X a cărei densitate (pentru variabile continue) sau funcție de masă (pentru variabile discrete) este cunoscută. Scopul este de a estima θ folosind două metode:

1. Metoda Verosimilității Maxime (MLE - Maximum Likelihood Estimation):

- Această metodă găsește valoarea lui θ care maximizează funcția de verosimilitate, adică probabilitatea de a observa eșantionul dat, condiționată de θ .

2. Metoda Momentelor (MM - Method of Moments):

- Această metodă estimează θ prin echivalarea momentelor teoretice ale distribuției cu momentele de eșantion (de exemplu, media eșantionului cu media teoretică).

După estimarea lui θ folosind cele două metode, se cere implementarea unei funcții în R care să calculeze și să returneze estimările. În plus, se compară estimația obținută prin MLE cu o estimație numerică a lui θ (obținută prin optimizare).

Un **scop palpabil** pentru această problemă ar fi obținerea unor estimări precise ale parametrului θ pentru distribuția variabilei aleatoare X , folosind metode statistice bine-cunoscute (MLE și MM), și validarea acestor estimări prin compararea lor cu o metodă numerică

ASPECTE TEORETICE

a) Funcția de verosimilitate (Likelihood Function)

- Funcția de verosimilitate $L(\theta)$ măsoară probabilitatea de a observa eșantionul dat, condiționată de parametrul θ .
- Pentru un eșantion x_1, x_2, \dots, x_n , funcția de verosimilitate este:

$$L(\theta) = \prod_{i=1}^n f(x_i; \theta),$$

unde $f(x_i; \theta)$ este densitatea (pentru variabile continue) sau funcția de masă (pentru variabile discrete).

b) Metoda Verosimilității Maxime (MLE)

- Scopul este de a găsi valoarea lui θ care maximizează $L(\theta)$.
- Se folosește adesea log-verosimilitatea $\ell(\theta) = \ln L(\theta)$ pentru simplificarea calculelor.
- Estimatorul MLE este soluția ecuației:

$$\frac{d\ell(\theta)}{d\theta} = 0.$$

c) Metoda Momentelor (MM)

- Scopul este de a echivala momentele teoretice ale distribuției cu momentele de eșantion.
- De exemplu, pentru o distribuție cu un singur parametru θ :

$E[X]$ =media teoretică=media de eșantion.

- Se rezolvă ecuația pentru θ .

d) Optimizare numerică

- Atunci când estimarea analitică este dificilă, se folosește optimizarea numerică (de exemplu, funcția optim în R) pentru a maximiza log-verosimilitatea.

e) Distribuția Poisson

Definiție

- Distribuția Poisson este o distribuție discretă care modelează numărul de evenimente rare care au loc într-un interval de timp sau spațiu.
- Parametru: λ (rata de apariție a evenimentelor).

Funcția de masă

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, 2, \dots$$

Proprietăți

- Media: $E[X] = \lambda$.
- Varianța: $Var(X) = \lambda$.

Estimarea parametrului λ

- MLE: $\hat{\lambda}_{MLE} = \frac{1}{n} \sum_{i=1}^n x_i$.
- MM: $\hat{\lambda}_{MM} = \frac{1}{n} \sum_{i=1}^n x_i$

d) Distribuția Gamma

Definiție

- Distribuția Gamma este o distribuție continuă folosită pentru a modela timpi de așteptare sau sume de variabile aleatoare exponențiale.
- Parametri:
 - α (parametrul de formă, shape).
 - β (parametrul de scară, scale) sau $\theta = \frac{1}{\beta}$ (parametrul de rată, rate).

Funcția de densitate

$$f(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \quad x > 0,$$

Proprietăți

- Media: $E[X] = \frac{\alpha}{\beta}$
- Varianța: $Var(X) = \frac{\alpha}{\beta^2}$.

Estimarea parametrului λ

- MLE: $\hat{\theta}_{MLE} = \frac{\alpha}{\bar{x}}$.
- MM: $\hat{\theta}_{MM} = \frac{\alpha}{\bar{x}}$.

e) Distribuția Geometrică

Definiție

- Distribuția Geometrică este o distribuție discretă care modelează numărul de încercări până la primul succes într-o succesiune de încercări Bernoulli.
- Parametru: p (probabilitatea de succes).

Funcția de masă

$$P(X = k) = (1 - p)^{k-1} p, \quad k = 1, 2, 3, \dots$$

Proprietăți

- Media: $E[X] = \frac{1}{p}$
- Varianța: $Var(X) = \frac{1-p}{p^2}$

Estimarea parametrului p

- MLE: $\hat{p}_{MLE} = \frac{1}{\bar{x}}$.
- MM: $\hat{p}_{MM} = \frac{1}{\bar{x}}$.

a) $f_{\theta}(x) = e^{-2\theta} \cdot \frac{2\theta^x}{x!}, x \in \mathbb{N}, \theta \in \mathbb{R}$

→ Metoda Verosimilitatii Maxime (MLE)

Pentru f_{θ} dat avem functia de verosimilitate:

$$L(\theta) = \prod_{i=1}^n e^{-2\theta} \cdot \frac{2\theta^{x_i}}{x_i!}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

$$\ln L(\theta) = \sum_{i=1}^n (-2\theta + x_i \ln(2\theta) - \ln(x_i!))$$

Iar, derivand:

$$\frac{d}{d\theta} \ln L(\theta) = -2n + \frac{1}{\theta} \sum_{i=1}^n x_i$$

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^n x_i}{2n}$$

→ Metoda Momentelor

Pentru f_{θ} dat media distributiei Poisson este:

$$E[X] = 2\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \frac{\bar{X}}{2}$$

Unde \bar{X} reprezinta media esantionului

Avem urmatoare implementare in R:

```
1. # Esantionul
2. sample_a <- c(8, 12, 6, 14, 9, 12, 15, 7, 15, 7, 10, 10, 14, 9, 12, 15, 11, 6, 8, 6, 8, 8, 9,
12, 13, 10, 11, 11, 13, 15, 10, 8, 7, 8, 13, 9, 9, 13, 12, 9, 10, 6, 10, 8, 10, 11, 12, 11, 9,
10, 7, 8, 8, 16, 7, 15, 10, 10, 8, 14, 13, 4, 11, 13, 6, 9, 13, 10, 10, 12, 11, 5, 6, 4, 9, 6, 9,
7, 13, 9, 11, 5, 5, 9, 15, 10, 11, 10, 14, 7, 11, 9, 14, 10, 5, 10, 8, 12, 13, 11)
3.
4. # Estimator MLE
5. theta_mle_a <- sum(sample_a) / (2 * length(sample_a))
6.
7. # Estimator MM
8. theta_mm_a <- mean(sample_a) / 2
9.
10. # Definirea functiei de log-verosimilitate
11. log_likelihood_a <- function(theta) {
12.   -2 * theta * length(sample_a) + sum(sample_a) * log(2 * theta) - sum(lfactorial(sample_a))
13. }
14.
15. # Maximizarea log-verosimilitatii
16. result_a <- optim(par = 1, fn = log_likelihood_a, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))
17.
18. # Estimator numeric
19. theta_numeric_a <- result_a$par
20.
21. # Afisare rezultate
22. cat("Estimator numeric pentru theta:", theta_numeric_a, "\n")
23. cat("Estimator MLE pentru theta:", theta_mle_a, "\n")
24. cat("Estimator MM pentru theta:", theta_mm_a, "\n")
25.
```

In urma careia, obtinem:

```
1. > cat("Estimator numeric pentru theta:", theta_numeric_a, "\n")
2. Estimator numeric pentru theta: 4.97
3. > cat("Estimator MLE pentru theta:", theta_mle_a, "\n")
4. Estimator MLE pentru theta: 4.97
5. > cat("Estimator MM pentru theta:", theta_mm_a, "\n")
6. Estimator MM pentru theta: 4.97
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $4.97 \cdot 10^{-11}$)

```
theta_numeric_a | 4.97000000079766
theta_mle_a     | 4.97
theta_mm_a      | 4.97
```


b) $f_{\theta}(x) = C_n^x \cdot \theta^x (1 - \theta)^{(1-x)}$, $x \in \{1, 2, 3, \dots, n\}$, $\theta \in (0, 1)$, n fixat

→ Metoda Verosimilitatii Maxime (MLE)

Pentru f_{θ} dat avem functia de verosimilitate:

$$L(\theta) = \prod_{i=1}^n \ln \binom{14}{x_i} \theta^{x_i} (1 - \theta)^{14-x_i}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

$$\ln L(\theta) = \sum_{i=1}^n \left(\ln \binom{14}{x_i} + x_i \ln \theta + (14 - x_i) \ln(1 - \theta) \right)$$

Iar, derivand:

$$\frac{d}{d\theta} \ln L(\theta) = \frac{\sum_{i=1}^n x_i}{\theta} - \frac{14n - \sum_{i=1}^n x_i}{1 - \theta}$$

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^n x_i}{14n}$$

→ Metoda Momentelor

Pentru f_{θ} dat media distributiei Binomiale este:

$$E[X] = 14\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \frac{\bar{X}}{14}$$

Unde \bar{X} reprezinta media esantionului

Avem urmatoare implementare in R:

```
1. # Esantionul
2. sample_b <- c(3, 2, 1, 4, 2, 3, 4, 1, 3, 2, 2, 4, 2, 1, 7, 5, 4, 5, 5, 2, 3, 4, 3, 1, 2, 4,
1, 1, 2, 3, 1, 3, 1, 4, 1, 3, 1, 6, 1, 3, 3, 4, 3, 1, 3, 2, 2, 3, 2, 4, 1, 1, 2, 6, 3, 1, 3, 6,
1, 2, 3, 6, 3, 2, 2, 2, 4, 2, 1, 3, 3, 4, 2, 3, 4, 1, 4, 4, 6, 3, 3, 5, 2, 2, 2, 3, 1, 3, 1, 3,
3, 5, 3, 4, 3, 2, 4, 2, 3, 3)
3.
4. # Estimator MLE
5. theta_mle_b <- sum(sample_b) / (14 * length(sample_b))
6.
7. # Estimator MM
8. theta_mm_b <- mean(sample_b) / 14
9.
10. # Definirea functiei de log-verosimilitate
11. log_likelihood_b <- function(theta) {
12.   sum(dbinom(sample_b, size = 14, prob = theta, log = TRUE))
13. }
14.
15. # Maximizarea log-verosimilitatii
16. result_b <- optim(par = 0.5, fn = log_likelihood_b, method = "Brent", lower = 0, upper = 1,
control = list(fnscale = -1))
17.
18. # Estimator numeric
19. theta_numeric_b <- result_b$par
20.
21. # Afisare rezultate
22. cat("Estimator numeric pentru theta:", theta_numeric_b, "\n")
23. cat("Estimator MLE pentru theta:", theta_mle_b, "\n")
24. cat("Estimator MM pentru theta:", theta_mm_b, "\n")
```

In urma careia, obtinem:

```
1. > cat("Estimator numeric pentru theta:", theta_numeric_b, "\n")
2. Estimator numeric pentru theta: 0.2014286
3. > cat("Estimator MLE pentru theta:", theta_mle_b, "\n")
4. Estimator MLE pentru theta: 0.2014286
5. > cat("Estimator MM pentru theta:", theta_mm_b, "\n")
6. Estimator MM pentru theta: 0.2014286
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $0.2014286 \cdot 10^{-13}$)

```
theta_numeric_b    | 0.201428571414209
theta_mm_b         | 0.201428571428571
theta_mle_b        | 0.201428571428571
```

c) $f_{\theta}(x) = e^{-\frac{x}{\theta}} \cdot \frac{x^{\alpha-1}}{\Gamma(\alpha) \cdot \theta^{\alpha}}, x \in (0, \infty), \theta \in (0, \infty), \alpha \in (0, \infty) \text{ fixat}$

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_{θ} dat avem functia de verosimilitate:

$$L(\theta) = \prod_{i=1}^n \frac{e^{-\frac{x_i}{\theta}} x_i^{\alpha-1}}{\Gamma(\alpha) \theta^{\alpha}}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

$$\ln L(\theta) = \sum_{i=1}^n \left(-\frac{x_i}{\theta} + (\alpha - 1) \ln x_i - \ln \Gamma(\alpha) - \alpha \ln \theta \right)$$

Iar, derivand:

$$\frac{d}{d\theta} \ln L(\theta) = \frac{\sum_{i=1}^n x_i}{\theta^2} - \frac{n\alpha}{\theta}$$

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^n x_i}{n\alpha}$$

→Metoda Momentelor

Pentru f_{θ} dat media distributiei Gamma este:

$$E[X] = \alpha\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \frac{\bar{X}}{\alpha}$$

Unde \bar{X} reprezinta media esantionului

Avem urmatoare implementare in R:

```
1. # Esantionul
2. sample_c <- c(6.269128, 25.204245, 13.994878, 13.391437, 11.458827, 10.565065, 11.706398,
10.625808, 7.485952, 16.353358, 9.277565, 8.566438, 14.788638, 6.830955, 9.542004, 20.272463,
36.562137, 12.244005, 16.084879, 11.454008, 15.592298, 6.332908, 13.106441, 6.198981, 15.726780,
7.883712, 35.124934, 11.856011, 13.766200, 16.534869, 16.803648, 11.196542, 19.785629, 26.300717,
21.270154, 7.192149, 5.882948, 15.812796, 10.963237, 24.963600, 13.802383, 15.281262, 10.310398,
20.940469, 23.992540, 15.869985, 12.041726, 12.521264, 10.869006, 15.386514, 14.636832,
18.104562, 17.029779, 4.506616, 20.941222, 12.050877, 9.757833, 20.070802, 12.472900, 6.474476,
15.059776, 13.157344, 9.124414, 13.768482, 24.354934, 12.363936, 11.110749, 9.092514, 17.856801,
14.757801, 13.898665, 9.119410, 11.430184, 11.958829, 13.516191, 10.701083, 14.713596, 10.121266,
16.945351, 13.524070, 14.742403, 19.165805, 10.338392, 12.327837, 19.619227, 7.328246, 14.894399,
19.631003, 7.622796, 12.343832, 13.138183, 10.061520, 17.674638, 9.675168, 12.115561, 15.182861,
13.292479, 17.888244, 16.695139, 2.952334)
4. # Parametrul alpha
5. alpha_c <- 7
6.
7. # Estimator MLE
8. theta_mle_c <- sum(sample_c) / (alpha_c * length(sample_c))
9.
10. # Estimator MM
11. theta_mm_c <- mean(sample_c) / alpha_c
12.
13. # Definirea functiei de log-verosimilitate
14. log_likelihood_c <- function(theta) {
15.   sum(dgamma(sample_c, shape = alpha_c, scale = theta, log = TRUE))
16. }
17.
18. # Maximizarea log-verosimilității
19. result_c <- optim(par = 1, fn = log_likelihood_c, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))
20.
21. # Estimator numeric
22. theta_numeric_c <- result_c$par
23.
24. # Afisare rezultate
25. cat("Estimator MLE pentru theta:", theta_mle_c, "\n")
26. cat("Estimator MM pentru theta:", theta_mm_c, "\n")
27. cat("Estimator numeric pentru theta:", theta_numeric_c, "\n")
```

În urma careia, obținem:

```
1. > cat("Estimator MLE pentru theta:", theta_mle_c, "\n")
2. Estimator MLE pentru theta: 1.993285
3. > cat("Estimator MM pentru theta:", theta_mm_c, "\n")
4. Estimator MM pentru theta: 1.993285
5. > cat("Estimator numeric pentru theta:", theta_numeric_c, "\n")
6. Estimator numeric pentru theta: 1.993285
```

Deci, diferențele nu sunt de așa natură astfel încât să altereze rezultatul semnificativ (dar în urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratețe $n \cdot 10^{-10}$

theta_numeric_c	1.99328521693552
theta_mm_c	1.99328521571429
theta_mle_c	1.99328521571429

$$\mathbf{d)} \ f_{\theta}(x) = \frac{\theta^x}{(1+\theta)^{(1+x)}}, x \in \mathbb{N}, \theta \in (0, \infty)$$

→Metoda Verosimilitatii Maxime (MLE)

Pentru f_{θ} dat avem functia de verosimilitate:

$$L(\theta) = \prod_{i=1}^n \frac{\theta^{x_i}}{(1+\theta)^{1+x_i}}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

$$\ln L(\theta) = \sum_{i=1}^n (x_i \ln \theta - (1+x_i) \ln(1+\theta))$$

Iar, derivand:

$$\frac{d}{d\theta} \ln L(\theta) = \frac{\sum_{i=1}^n x_i}{\theta} - \frac{n + \sum_{i=1}^n x_i}{1+\theta}$$

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^n x_i}{n}$$

→Metoda Momentelor

Pentru f_{θ} dat media distributiei Geometrice este:

$$E[X] = \theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \bar{X}$$

Unde \bar{X} reprezinta media esantionului

Avem urmatoare implementare in R:

```
1. # Esantionul
2. sample_d <- c(6, 3, 24, 24, 4, 56, 10, 13, 2, 28, 24, 2, 22, 11, 2, 8, 118, 2, 14, 19, 7, 9,
8, 189, 2, 9, 21, 6, 6, 2, 3, 2, 3, 18, 3, 2, 21, 1, 5, 9, 11, 13, 19, 76, 1, 5, 9, 4, 57, 1, 2,
16, 5, 2, 20, 8, 1, 40, 6, 4, 19, 6, 3, 2, 4, 9, 1, 5, 10, 12, 6, 525, 19, 6, 17, 2, 5, 159, 5,
62, 6, 3, 45, 21, 23, 3, 17, 2, 1, 1, 474, 15, 3, 3, 7, 7, 13, 4, 38, 4)
3.
4. # Estimator MLE
5. theta_mle_d <- mean(sample_d)
6.
7. # Estimator MM
8. theta_mm_d <- mean(sample_d)
9.
10. # Definirea functiei de log-verosimilitate
11. log_likelihood_d <- function(theta) {
12.   sum(dgeom(sample_d, prob = 1 / (1 + theta), log = TRUE))
13. }
14.
15. # Maximizarea log-verosimilității
16. result_d <- optim(par = 1, fn = log_likelihood_d, method = "Brent", lower = 0, upper = 1000,
control = list(fnscale = -1))
17.
18. # Estimator numeric
19. theta_numeric_d <- result_d$par
20.
21. # Afisare rezultate
22. cat("Estimator MLE pentru theta:", theta_mle_d, "\n")
23. cat("Estimator MM pentru theta:", theta_mm_d, "\n")
24. cat("Estimator numeric pentru theta:", theta_numeric_d, "\n")In urma careia, obtinem:
25. 1. > cat("Estimator MLE pentru theta:", theta_mle_d, "\n")

1. Estimator MLE pentru theta: 25.85
2. > cat("Estimator MM pentru theta:", theta_mm_d, "\n")
3. Estimator MM pentru theta: 25.85
4. > cat("Estimator numeric pentru theta:", theta_numeric_d, "\n")
5. Estimator numeric pentru theta: 25.85
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $n \cdot 10^{-8}$

```
theta_numeric_d | 25.8500003901255
theta_mm_d      | 25.85
theta_mle_d     | 25.85
```

e) $f_{\theta}(x) = \frac{\alpha}{\theta} x^{\alpha-1} \cdot e^{-\frac{x}{\theta}}, x \in (0, \infty), \theta \in (0, \infty), \alpha \in (0, \infty)$ fixat

→ Metoda Verosimilitatii Maxime (MLE)

Pentru f_{θ} dat avem functia de verosimilitate:

$$L(\theta) = \prod_{i=1}^n \frac{\alpha}{\theta} x_i^{\alpha-1} e^{-\frac{x_i}{\theta}}$$

Pe aceasta, o folosim pentru a afla log-verosimilitatea:

$$\ln L(\theta) = \sum_{i=1}^n \left(\ln \alpha - \ln \theta + (\alpha - 1) \ln x_i - \frac{x_i}{\theta} \right)$$

Iar, derivand:

$$\frac{d}{d\theta} \ln L(\theta) = \frac{\sum_{i=1}^n x_i}{\theta^2} - \frac{n}{\theta}$$

Si egaland cu 0 obtinem estimatorul MLE:

$$\hat{\theta}_{MLE} = \frac{\sum_{i=1}^n x_i}{n\alpha}$$

→ Metoda Momentelor

Pentru f_{θ} dat media distributiei Gamma este:

$$E[X] = \alpha\theta$$

Si deci avem estimatorul MM:

$$\hat{\theta}_{MM} = \frac{\bar{X}}{\alpha}$$

Unde \bar{X} reprezinta media esantionului

Avem urmatoare implementare in R:

```
1. # Esantionul
2. sample_e <- c(3.5930579, 2.1027540, 1.7820777, 9.6550388, 6.8803846, 0.7388358, 2.9194654,
3.1178660, 1.2323236, 2.9776820, 1.1172078, 2.4184586, 3.3258971, 1.9498871, 2.6088612,
3.9535062, 3.0389107, 4.4226628, 3.9366318, 2.4551569, 5.2814487, 5.6778622, 4.7683935,
1.1581498, 3.1270783, 4.1473311, 7.4830426, 1.1342893, 1.7773392, 7.7510826, 1.3919927,
2.3613291, 2.6234826, 1.6562602, 1.4992235, 2.3455062, 3.8458809, 5.8333841, 3.3834034,
1.5202546, 3.1248186, 5.3029567, 3.6225571, 4.8309931, 3.1579595, 3.2640258, 3.9538891,
4.0796841, 4.0991772, 3.2779944, 2.5002127, 3.0654695, 1.6996010, 3.2175175, 1.9033087,
4.4052061, 2.3158379, 2.4778345, 5.4382190, 4.9141207, 6.0978745, 1.1428936, 3.5639106,
7.4541937, 7.7778289, 3.2859563, 0.7432908, 1.4442696, 3.6619932, 2.8361371, 4.3180773,
1.6763585, 4.4464154, 2.5049617, 0.4448735, 5.0518839, 3.4151834, 1.6823650, 5.4517583,
2.8212788, 2.1566837, 2.9893287, 1.6925123, 6.5197938, 4.2165408, 1.6728425, 2.7650830,
2.6742755, 2.9622047, 0.7809781, 1.3913415, 5.3430751, 2.4859925, 3.7329465, 6.3129236,
0.6635228, 3.7640343, 2.1850174, 4.3773328, 5.0931544)
3.
4. # Parametrul alpha
5. alpha_e <- 3
6.
7. # Estimator MLE
8. theta_mle_e <- sum(sample_e) / (alpha_e * length(sample_e))
9.
10. # Estimator MM
11. theta_mm_e <- mean(sample_e) / alpha_e
12.
13. # Definirea functiei de log-verosimilitate
14. log_likelihood_e <- function(theta) {
15.   sum(dgamma(sample_e, shape = alpha_e, scale = theta, log = TRUE))
16. }
17.
18. # Maximizarea log-verosimilității
19. result_e <- optim(par = 1, fn = log_likelihood_e, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))
20.
21. # Estimator numeric
22. theta_numeric_e <- result_e$par
23.
24. # Afișare rezultate
25. cat("Estimator MLE pentru theta:", theta_mle_e, "\n")
26. cat("Estimator MM pentru theta:", theta_mm_e, "\n")
27. cat("Estimator numeric pentru theta:", theta_numeric_e, "\n")
28.
```

In urma careia, obtinem:

```
1. 1. > cat("Estimator MLE pentru theta:", theta_mle_e, "\n")
2. 2. Estimator MLE pentru theta: 1.124153
3. 3. > cat("Estimator MM pentru theta:", theta_mm_e, "\n")
4. 4. Estimator MM pentru theta: 1.124153
5. 5. > cat("Estimator numeric pentru theta:", theta_numeric_e, "\n")
6. 6. Estimator numeric pentru theta: 1.124153
```

Deci, diferentele nu sunt de asa natura astfel incat sa altereze rezultatul semnificativ (dar in urma unei analize mai complexe asupra datelor stocate pot fi observate de la nivelul de acuratete $n \cdot 10^{-10}$)

```
theta_numeric_e | 1.12415290480678
theta_mm_e      | 1.12415290633333
theta_mle_e     | 1.12415290633333
```


Concluzii

Putem observa in urma tuturor calculelor, ca folosind oricare din metode, obtinem rezultate extrem de similare, deci pentru orice aplicatie reala, o putem aplica pe oricare.

Sursele care au ajutat la rezolvarea problemelor:

P1.

- Cursurile domnului profesor Niculescu Cristian
- https://alexamarioarei.quarto.pub/curs-ps-fmi/Introducere_R/Chapter_4/Elemente_grafica_R.html
- <https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Index-vectors>
- https://www.math.uaic.ro/~maticiuc/didactic/Probability_Theory_Course_7_8_9_10.pdf
- https://en.wikipedia.org/wiki/Exponential_distribution
- https://en.wikipedia.org/wiki/Expected_value
- https://en.wikipedia.org/wiki/Markov_chain
- <https://www.geeksforgeeks.org/exponential-distribution-in-r-programming-dexp-pexp-qexp-and-rexp-functions/>

P2.

- Codul primit in cadrul laboratorului
- https://alexamarioarei.quarto.pub/curs-ps-fmi/Introducere_R/
- Materialele de curs ale domnului profesor Niculescu Cristian
- <https://mathworld.wolfram.com/CauchyDistribution.html>
- <https://shiny.posit.co> (documentatia shiny)
- Documentatia prezenta in R Studio
- Materialele de seminar pentru calculul cu v.a continue si discrete.

P3.

- <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/GammaDist.html>
- <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/optim.html>
- https://www.probabilitycourse.com/chapter8/8_2_3_max_likelihood_estimation.php
- Materialele de curs ale domnului profesor Niculescu Cristian
- Materialele de seminar pentru calculul cu v.a continue si discrete.

Anexa cod

Cod Problema 1

```
1. set.seed(123) # pt aceleasi val random cand dai run
2. n <- 10 # nr de etape
3. lambda <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) # parametrii lambda pentru fiecare etapa
4. alpha <- c(0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05) # prob de trecere la etapa
urm
5. # alfa de i este prob ca persoana A sa treaca de la etapa i la etapa i+1
6. # 1-alfa de i este prob ca persoana A sa se opreasca dupa etapa i
7. nrSim <- 1000000 # nr de simulari
8.
9.
10. ##### cerinta 1 #####
11.
12. simulate_T <- function(n, lambda, alpha) {
13.   total_time <- 0
14.   for (i in 1:n) {
15.     time <- rexp(1, rate = lambda[i]) # timp pentru etapa i
16.     total_time <- total_time + time
17.     if (runif(1) > alpha[i]) { # alege random un nr intre 0 si 1
18.       break # stop dupa etapa i daca nr ales este mai mare decat prob alfa de i
19.     }
20.   }
21.   return(total_time)
22. }
23.
24. T_values <- replicate(nrSim, simulate_T(n, lambda, alpha))
25. # aplica functia/operatia simulate_T de 10^6 ori pt T
26. mean_T <- mean(T_values) # media ar a timpilor totali din simulari
27.
28. # reprezentare grafica
29. hist(T_values, breaks = 50, main = "Distributia lui T", xlab = "Timpul total
T", ylab="Frecventa", col = "darkseagreen1")
30. # daca freq=false arata probabilitatile
31. # breaks = nr bins/cosuri/drept alea verzi
32. abline(v = mean_T, col = "purple1", lwd = 2)
33. # linia mov pt media lui T
34. legend("topright", legend = paste("E(T) = ", round(mean_T, 2)), col = "purple1", lwd = 2)
35. # legenda pt linia mediei lui T
36. print(paste("Valoarea aproximata a lui E(T) =", mean_T))
37.
38.
39.
40. ##### cerinta 2 #####
41.
42. exact_ET <- 0
43. for (i in 1:n) {
44.   prod_alpha <- if (i == 1) 1 else prod(alpha[1:(i-1)]) # produsul alpha1 * alpha2 * ... *
alpha(i-1)
45.   exact_ET <- exact_ET + prod_alpha * (1 / lambda[i])
46. }
47.
48. print(paste("Valoarea exacta a lui E(T) =", exact_ET))
49. print(paste("Dif intre valoarea exacta si cea simulata =", abs(exact_ET - mean_T)))
50.
51.
52.
53. ##### cerinta 3 #####
54.
55. simulate_T_final <- function(n, lambda, alpha) {
56.   total_time <- 0
57.   completed <- TRUE # presupun ca finalizeaza activitatea
```

```

58.   for (i in 1:n) {
59.     time <- rexp(1, rate = lambda[i]) # timp pentru etapa i
60.     total_time <- total_time + time
61.     if (runif(1) > alpha[i]) {
62.       completed <- FALSE # nu a finalizat activitatea
63.       break
64.     }
65.   }
66.   return(completed) # returneaza TRUE daca a finalizat, FALSE altfel
67. }
68.
69. comp <- replicate(nrSim, simulate_T_final(n, lambda, alpha))
70. prob_final <- mean(comp) # probabilitatea de finalizare
71. print(paste("Probabilitatea de finalizare a activitatii =", prob_final))
72.
73.
74.
75. ##### cerinta 4 #####
76.
77. sigma <- 5 # valoarea lui sigma
78. # este ales mai mare sa acopere toate cazurile, deci prob o sa fie 1
79. T_values_sigma <- T_values[comp]
80. # filtreaza T_values folosind vectorul logic comp
81. # ia elementele de pe aceleasi pozitii iar daca in al doilea vector e true
82. # atunci pastreaza timpul lui T
83. prob_sigma <- mean(T_values_sigma <= sigma)
84. # compara fiecare T cu sigma is face med ar din val de true false <= sigma
85. print(paste("Probabilitatea ca T <= sigma:", prob_sigma))
86.
87.
88.
89. ##### cerinta 5 #####
90.
91. min_T <- min(T_values_sigma)
92. max_T <- max(T_values_sigma)
93.
94. print(paste("Timpul minim de finalizare=", min_T))
95. print(paste("Timpul maxim de finalizare=", max_T))
96.
97. # reprezentare grafica a timpilor de finalizare
98. hist(T_values_sigma, breaks = 50, main = "Repartitia timpilor de finalizare", xlab = "Timpul
total T", ylab = "Frecventa", col = "darkseagreen1")
99. abline(v = mean(T_values_sigma), col = "purple1", lwd = 2) # linie pentru medie
100. legend("topright", legend = paste("Media =", round(mean(T_values_sigma), 2)), col =
"purple1", lwd = 2)
101.
102.
103. ##### cerinta 6 #####
104.
105. simulate_stop_stage <- function(n, lambda, alpha) {
106.   for (i in 1:n) {
107.     time <- rexp(1, rate = lambda[i])
108.     if (runif(1) > alpha[i]) { # daca se opreste dupa etapa i
109.       return(i) # returneaza etapa la care s-a oprit
110.     }
111.   }
112.   return(n) # daca a trecut prin toate etapele
113. }
114. # functia care iti arata la ce etapa s-a oprit A
115.
116. stop_stages <- replicate(nrSim, simulate_stop_stage(n, lambda, alpha))
117. # etapele la care s-a oprit A vector
118.
119. prob_stop_before_k <- sapply(1:n, function(k) {
120.   mean(stop_stages < k) # prob ca A sa se opreasca inainte de k
121. })
122. # se aplica pt toate k-urile de la 1 la n
123.
124. # reprezentare grafica
125. plot(1:n, prob_stop_before_k, type = "b", col = "purple1",

```

```

126.     xlab = "Etapa k", ylab = "Prob de op inainte de k",
127.     main = "Probabilitatea de oprire inainte de etapa k")
128. grid()

```

Cod Problema 2

```

1. library(shiny)
2.
3. ui <- fluidPage(
4.
5.   #UI Side Normal(0,1)
6.   #####
7.   titlePanel("Funcția de repartitie a unei variabile normale standard"),
8.   sidebarLayout(
9.     sidebarPanel(),
10.    mainPanel(
11.      plotOutput("normalStdPlot"),
12.      plotOutput("transformedStdPlot"),
13.      plotOutput("squaredStdPlot")
14.    )
15.  ),
16.  sidebarLayout(
17.    sidebarPanel(
18.      sliderInput("stdN", "Alege numarul de variabile", min = 1, max = 10, value = 1, step
=1)
19.    ),
20.    mainPanel(
21.      plotOutput("sumStd"),
22.      plotOutput("sumSquaredStd")
23.    )
24.  ),
25.
26.  #####
27.
28.  #UI Side Normal(mu, sigma^2)
29.  #####
30.  titlePanel("Funcția de repartiție a unei variabile normale"),
31.  sidebarLayout(
32.    sidebarPanel(
33.      sliderInput("mu", "Alege media (\u03BC):", min = -10, max = 10, value = 0, step =
0.1),
34.      sliderInput("sigma", "Alege deviația standard (\u03C3):", min = 0.1, max = 10, value =
1, step = 0.1)
35.    ),
36.    mainPanel(
37.      plotOutput("cdfPlot"),
38.      plotOutput("transformedPlot"),
39.      plotOutput("squaredPlot")
40.    )
41.  ),
42.  sidebarLayout(
43.    sidebarPanel(
44.      sliderInput("n", "Alege n:", min = 1, max = 10, value = 1, step = 1),
45.      sliderInput("muSum", "Alege media (\u03BC):", min = -10, max = 10, value = 0, step =
0.1),
46.      sliderInput("sigmaSum", "Alege deviația standard (\u03C3):", min = 0.1, max = 10,
value = 1, step = 0.1)
47.    ),
48.    mainPanel(
49.      plotOutput("sumCdf"),
50.      plotOutput("sumSquaredCdf")
51.    )
52.  ),
53.  #####
54.

```

```

55. #UI Side Exponential(lambda)
56. #####
57. titlePanel("Funcția de repartiție a unei variabile exponențiale"),
58. sidebarLayout(
59.   sidebarPanel(
60.     sliderInput("lambda", "Alege lambda (\u03BB)", min = 0.001, max = 10, value = 1, step
= 0.1),
61.   ),
62.   mainPanel(
63.     plotOutput("cdfExpDefault"),
64.     plotOutput("transformedExp"),
65.     plotOutput("squaredExp")
66.   )
67. ),
68. sidebarLayout(
69.   sidebarPanel(
70.     sliderInput("nExp", "Alege n", min = 1, max = 10, value = 1, step = 1),
71.     sliderInput("sumLambda", "Alege lambda (\u03BB)", min = 0.001, max = 10, value = 1,
step = 0.1)
72.   ),
73.   mainPanel(
74.     plotOutput("sumExp")
75.   )
76. ),
77. #####
78.
79. #UI Side Poisson(lambda)
80. #####
81. titlePanel("Funcția de repartiție a unei variabile poisson"),
82. sidebarLayout(
83.   sidebarPanel(
84.     sliderInput("lambdaPoisson", "Alege lambda (\u03BB)", min = 0.001, max = 30, value =
1, step = 0.1),
85.   ),
86.   mainPanel(
87.     plotOutput("cdfPoisson"),
88.     plotOutput("transformedPoisson"),
89.     plotOutput("squaredPoisson")
90.   )
91. ),
92. sidebarLayout(
93.   sidebarPanel(
94.     sliderInput("poissonN", "Alege n", min = 1, max = 10, value = 1, step = 1),
95.     sliderInput("sumPoisLambda", "Alege lambda (\u03BB)", min = 0.001, max = 30, value =
1, step = 0.1),
96.   ),
97.   mainPanel(
98.     plotOutput("sumPoisson")
99.   )
100. ),
101. #####
102.
103. #UI Side Binomial(r,p)
104. #####
105. titlePanel("Funcția de repartiție a unei variabile binomială"),
106. mainPanel(
107.   sliderInput("r", "Alege numărul de experimente", min = 0, max = 100, value= 10, step =
1),
108.   sliderInput("p", "Alege probabilitatea de succes", min = 0.01, max = 1, value = 0.3,
step = 0.01),
109.   plotOutput("cdfBinomial"),
110.   plotOutput("transformedBinomial"),
111.   plotOutput("squaredBinomial")
112. ),
113. mainPanel(
114.   sliderInput("sumBinN", "Alege numărul de variabile", min = 1, max = 10, value = 1, step
= 1),
115.   sliderInput("sumR", "Alege numărul de experimente", min = 0, max = 100, value= 10, step
= 1),

```

```

116.     sliderInput("sumP", "Alege probabilitatea de succes", min = 0.01, max = 1, value = 0.3,
step = 0.01),
117.     plotOutput("sumBinomial")
118.   ),
119.   #####
120.
121.   #Ex. 2
122.   #Să construieți câte o funcție în R care afișează funcția pentru parametri
particularizabili de
123.   #câtre utilizator și calculează și media și varianța pentru v.a. X
124.
125.   #####
126.
127.   #a)  $f(x) = cx^4$ ,  $x$  din  $(0,2)$ ,  $c$  din R
128.   mainPanel(
129.     titlePanel("Densitatea, media si varianta pentru  $f(x) = cx^4$ "),
130.     sidebarLayout(
131.       sidebarPanel(
132.         numericInput("numerator", "Introdu un numarator pentru c:", value = 5, min = 1, max
= 100, step = 1),
133.         numericInput("denominator", "Introdu un numitor pentru c:", value = 32, min = 1, max
= 100, step = 1)
134.       ),
135.       mainPanel(
136.         plotOutput("densityA"),
137.         verbatimTextOutput("resultsA")
138.       )
139.     ),
140.   ),
141.
142.   #####
143.
144.   #b)  $f(x) = ax + bx^2$ ,  $0 < x < 1$ ,  $a, b$  din R
145.   mainPanel(
146.     titlePanel("Densitatea, media si varianta pentru  $f(x) = ax + bx^2$ ,  $a = (6 - 2b) / 3$ "),
147.     sidebarLayout(
148.       sidebarPanel(
149.         numericInput("ex2A", "Introdu un A valid raportat la b: ", value = 0, min = -20, max
= 20, step = 0.1),
150.         numericInput("ex2B", "Introdu un B valid raportat la a: ", value = 3, min = -20, max
= 20, step = 0.1)
151.       ),
152.       mainPanel(
153.         plotOutput("densityB"),
154.         verbatimTextOutput("resultsB")
155.       )
156.     )
157.   ),
158. ),
159.
160.   #####
161.
162.   #c)  $f(X) = 4/x(x+1)(x+2)$ ,  $x$  din  $N \setminus \{0\}$ 
163.   mainPanel(
164.     titlePanel("Pdf, media si varianta pentru X cu pdf =  $4/x(x+1)(x+2)$ "),
165.     sidebarLayout(
166.       sidebarPanel(
167.         ),
168.       mainPanel(
169.         plotOutput("densityC"),
170.         verbatimTextOutput("resultsC"),
171.       )
172.     )
173.   ),
174.
175.   #####
176.   #d)  $f(x) = \lg(x/x+1)$ ,  $x$  din  $\{1,9\}$ 
177.   mainPanel(
178.     titlePanel("Pdf, media si varianta pentru x cu pdf =  $\lg(x/x+1)$ "),
179.     sidebarLayout(

```

```

180.     sidebarPanel(),
181.     mainPanel(
182.         plotOutput("densityD"),
183.         verbatimTextOutput("resultsD")
184.     )
185. ),
186. ),
187. #####
188. #e)  $f(x) = \text{teta}^2 / (1 + \text{teta}) * (1 + x)e^{-\text{teta} * x}$ ,  $x > 0$ ,  $\text{teta} > 0$ 
189. mainPanel(
190.     titlePanel("Densitatea, media si varianta pentru X din E"),
191.     sidebarLayout(
192.         sidebarPanel(
193.             sliderInput("teta", "Introdu variabila teta > 0: ", value = 1, min = 0.01, max = 5,
194. step = 0.01)
195.         ),
196.         mainPanel(
197.             plotOutput("densityE"),
198.             verbatimTextOutput("resultsE"),
199.         )
200.     )
201. ),
202. #####
203. #f)  $f(x) = 1/3e^{-x}$ ,  $x < 0$ ;  $f(x) = 1/3$ ,  $0 \leq x < 1$ ;  $f(x) = 1/3 * x^{1-x}$ ,  $x \geq 1$ 
204. mainPanel(
205.     titlePanel("Densitatea, media si varianta pentru X din F"),
206.     sidebarLayout(
207.         sidebarPanel(),
208.         mainPanel(
209.             plotOutput("densityF"),
210.             verbatimTextOutput("resultsF")
211.         )
212.     )
213. ),
214. ),
215. #####
216. #g)  $f(x) = 1/(\pi * (1 + x^2))$ 
217. mainPanel(
218.     titlePanel("Densitatea, media si varianta pentru X din G"),
219.     sidebarLayout(
220.         sidebarPanel(),
221.         mainPanel(
222.             plotOutput("densityG"),
223.             verbatimTextOutput("resultsG")
224.         )
225.     )
226. ),
227. ),
228. )
229. )
230. )
231.
232.
233. server <- function(input, output) {
234.
235.     #Server Side Normal(0,1)
236.     #####
237.
238.     output$normalStdPlot <- renderPlot({
239.         x <- seq(-10, 10, 0.001)
240.         y <- pnorm(x, mean = 0, sd = 1)
241.         plot(x, y, type = "l", col = "blue", lwd = 2,
242.             xlab = "x", ylab = "F(x)",
243.             main = paste("Funcția de repartiție pentru N(0,1)"))
244.         grid()
245.     })
246.
247.     # ?Pentru  $Y = 3 - 2X$ ,  $P(y) = P(3 - 2X \leq y) = P(X \geq (3 - y)/2) = 1 - P(X \leq (3 - y)/2)$ 
248.     output$transformedStdPlot <- renderPlot({

```

```

249.     x <- seq(-17, 23, 0.001)
250.     transformedX <- (3-x)/2
251.     y <- 1 - pnorm(transformedX, mean = 0, sd = 1)
252.     plot(-transformedX, y, type = "l", col = "blue", lwd = 2,
253.          xlab = "x", ylab = "F(x)",
254.          main = paste("Funcția de repartiție pentru 3 - 2X"))
255.     grid()
256. })
257.
258. # Pentru Y = X^2, P(X^2<=y)=P(-sqrt(y)<=X<=sqrt(y))=Fx(sqrt(y)) - Fx(-sqrt(y))
259. output$squaredStdPlot <- renderPlot({
260.     x <- seq(0, 100, 0.01)
261.     sqrtX <- sqrt(x)
262.     y <- pnorm(sqrtX, mean = 0, sd = 1) - pnorm(-sqrtX, mean = 0, sd = 1)
263.     plot(sqrtX, y, type = "l", col = "blue", lwd = 2,
264.          xlab = "x", ylab = "F(x)",
265.          main = paste("Funcția de repartiție pentru X^2"))
266.     grid()
267. })
268.
269. #Pentru Y = Sn
270. output$sumStd <- renderPlot({
271.     n <- input$stdN
272.     x <- seq(-10, 10, 0.001)
273.     y <- pnorm(x, mean = 0, sd = sqrt(n))
274.     plot(x, y, type = "l", col = "blue", lwd = 2,
275.          xlab = "x", ylab = "F(x)",
276.          main = paste("Funcția de repartiție pentru Sum(N(0,1))"))
277.     grid()
278. })
279.
280. #Pentru Y = Qn
281. output$sumSquaredStd <- renderPlot({
282.     n <- input$stdN
283.     x <- seq(0, 10, 0.01)
284.     y <- pnorm(x, mean = n, sd = sqrt(2 * n))
285.     plot(x, y, type = "l", col = "blue", lwd = 2,
286.          xlab = "x", ylab = "F(x)",
287.          main = paste("Funcția de repartiție pentru Sum(N(0,1)^2)"))
288.     grid()
289. })
290.
291. #####
292.
293. #Server Side Normal(mu, sigma^2)
294. #####
295.
296. #Pentru X = N(mu, sigma^2)
297. output$cdfPlot <- renderPlot({
298.     x <- seq(-10, 10, 0.001)
299.     y <- pnorm(x, mean = input$mu, sd = input$sigma)
300.
301.     plot(x, y, type = "l", col = "blue", lwd = 2,
302.          xlab = "x", ylab = "F(x)",
303.          main = paste("Funcția de repartiție pentru N(", input$mu, ",", input$sigma^2, ")"))
304.     grid()
305. })
306.
307. #Pentru Y = 3 - 2X
308. output$transformedPlot <- renderPlot({
309.     x <- seq(-17, 23, 0.001)
310.     transformed_x <- (3-x)/2
311.     y <- 1 - pnorm(transformed_x, mean = input$mu, sd = input$sigma)
312.     plot(-transformed_x, y, type = "l", col = "blue", lwd = 2,
313.          xlab = "x", ylab = "F(x)",
314.          main = paste("Funcția de repartiție pentru 3-2X"))
315.     grid()
316. })
317.
318. #Pentru Y = X^2

```



```

319. output$squaredPlot <- renderPlot({
320.   x <- seq(0, 100, 0.01)
321.   sqrtX <- sqrt(x)
322.   y <- pnorm(sqrtX, mean = input$mu, sd = input$sigma)
323.   plot(sqrtX, y, type = "l", col = "blue", lwd = 2,
324.         xlab = "x", ylab = "F(x)",
325.         main = paste("Funcția de repartiție pentru X^2"))
326.   grid()
327. })
328.
329. #Pentru Sn = N(n*mu, n * sigma^2)
330. output$sumCdf <- renderPlot({
331.   n <- input$n
332.   x <- seq(-10, 10, 0.001)
333.   y <- pnorm(x, mean = n * input$muSum, sd = sqrt(n) * input$sigmaSum)
334.   plot(x, y, type = "l", col = "blue", lwd = 2,
335.         xlab = "x", ylab = "F(x)",
336.         main = paste("Funcția de repartiție pentru Sum(Xi)"))
337.   grid()
338. })
339.
340. #Pentru Qn = N(n*(s^2 + mu^2), n *(2*s^4 + 4*mu^2*s^2))
341. output$sumSquaredCdf <- renderPlot({
342.   n <- input$n
343.   selMean <- input$muSum
344.   selSd <- input$sigmaSum
345.   x <- seq(0, 10, 0.001)
346.
347.   mean <- n*(selMean^2 + selSd^2)
348.   sd <- sqrt(n*(2*selSd^4 + 4*selMean^2*selSd^2))
349.
350.   y <- pnorm(x, mean = mean, sd = sd)
351.   plot(x, y, type = "l", col = "blue", lwd = 2,
352.         xlab = "x", ylab = "F(x)",
353.         main = paste("Funcția de repartiție pentru Sum(Xi^2)"))
354.   grid()
355. })
356.
357. #####
358.
359. #Server Side Exponential(lambda)
360. #####
361.
362. # Pentru X = Exp(lambda)
363. output$cdfExpDefault <- renderPlot({
364.   x <- seq(0, 6, 0.001)
365.   y <- pexp(x, rate = input$lambda)
366.   plot(x, y, type = "l", col = "blue", lwd = 2,
367.         xlab = "x", ylab = "F(x)",
368.         main = paste("Funcția de repartiție pentru Exp(\u03BB)"))
369.   grid()
370. })
371.
372. # Pentru Y = 2 + 5X, F(y)=P(Y<=y)=P(2+5X<=y)=P(X<=(y-2)/5)=Fx((y-2)/5)
373. output$transformedExp <- renderPlot({
374.   x <- seq(0.001, 6, 0.001)
375.   transformedX <- (x-2)/5
376.   y <- pexp(transformedX, rate = input$lambda)
377.   plot(transformedX, y, type = "l", col = "blue", lwd = 2,
378.         xlab = "x", ylab = "F(x)",
379.         main = paste("Funcția de repartiție pentru 2 + 5X"))
380.   grid()
381. })
382.
383. #Pentru Y = X^2
384. output$squaredExp <- renderPlot({
385.   x <- seq(0, 36, 0.01)
386.   sqrtX <- sqrt(x)
387.   y <- pexp(sqrtX, rate = input$lambda) # y e mereu pozitiv, din definitia exp
388.   plot(sqrtX, y, type = "l", col = "blue", lwd = 2,

```

```

389.         xlab = "x", ylab = "F(x)",
390.         main = paste("Funcția de repartiție pentru X^2"))
391.     grid()
392. })
393.
394. #Pentru Sn = Gamma(n, lambda)
395. output$sumExp <- renderPlot({
396.     n <- input$nExp
397.     x <- seq(0.001, 10, 0.001)
398.     y <- pgamma(x, shape = n, rate = input$sumLambda)
399.     plot(x, y, type = "l", col = "blue", lwd = 2,
400.         xlab = "x", ylab = "F(x)",
401.         main = paste("Funcția de repartiție pentru Sum(Exp(\u03BB))"))
402.     grid()
403. })
404.
405. #####
406.
407. #Server Side Poisson(lambda)
408. #####
409.
410. # Pentru X = Poisson(lambda)
411. output$cdfPoisson <- renderPlot({
412.     x <- seq(0, 20, 0.01) # am folosit aceasta discretizare pt a obtine un grafic mai estetic
413.     y <- ppois(x, input$lambdaPoisson)
414.     plot(x, y, type = "l", col = "blue", lwd = 2,
415.         xlab = "x", ylab = "F(x)",
416.         main = paste("Funcția de repartiție pentru Poisson(\u03BB)"))
417.     grid()
418. })
419.
420. #Pentru Y = 3X - 2, F(y)=P(Y<=y)=P(3X-2<=y)=P(X<=(y+2)/3)
421. output$transformedPoisson <- renderPlot({
422.     x <- seq(0, 20, 0.01)
423.     transformedX <- (x + 2)/3
424.     y <- ppois(transformedX, input$lambdaPoisson)
425.     plot(transformedX, y, type = "l", col = "blue", lwd = 2,
426.         xlab = "x", ylab = "F(x)",
427.         main = paste("Funcția de repartiție pentru 3X-2"))
428.     grid()
429. })
430.
431. #Pentru Y = X^2
432. output$squaredPoisson <- renderPlot({
433.     x <- seq(0, 20, 0.01)
434.     sqrtX <- sqrt(x)
435.     y <- ppois(sqrtX, input$lambdaPoisson)
436.     plot(sqrtX, y, type = "l", col = "blue", lwd = 2,
437.         xlab = "x", ylab = "F(x)",
438.         main = paste("Funcția de repartiție pentru X^2"))
439.     grid()
440. })
441.
442. #Pentru Y = Poisson(n*lambda)
443. output$sumPoisson <- renderPlot({
444.     n <- input$poissonN
445.     x <- seq(0.001, 20 * n, 0.001)
446.     y <- ppois(x, n * input$sumPoisLambda)
447.     plot(x, y, type = "l", col = "blue", lwd = 2,
448.         xlab = "x", ylab = "F(x)",
449.         main = paste("Funcția de repartiție pentru Sum(Poisson(\u03BB))"))
450.     grid()
451. })
452.
453. #####
454.
455. #Server Side Binomial(r,p)
456. #####
457.
458. #Pentru X ~ Binomial(r,p)

```

```

459. output$cdfBinomial <- renderPlot({
460.   r <- input$r
461.   p <- input$p
462.   x <- seq(0,r,0.01)
463.   y <- pbinom(x, size = r, prob = p)
464.   plot(x, y, col = "blue", lwd =2,type="l",
465.        xlab = "x", ylab = "F(x)",
466.        main = paste("Funcția de repartiție pentru Binomial(r,p)"))
467.   grid()
468. })
469.
470. #Pentru Y = 5X-4=>P(5x-4<=y)=P(X<=(y+4)/5)
471. output$transformedBinomial <- renderPlot({
472.   r <- input$r
473.   p <- input$p
474.   x <- seq(-4, 46, 0.01)
475.   transformedX <- (x + 4)/5
476.   y <- pbinom(transformedX, size = r, prob = p)
477.   plot(x, y, col = "blue", lwd =2, type="l",
478.        xlab = "x", ylab = "F(x)",
479.        main = paste("Funcția de repartiție pentru 5X-4"))
480.   grid()
481. })
482.
483. #Pentru Y=X^2
484. output$squaredBinomial <- renderPlot({
485.   r <- input$r
486.   p <- input$p
487.   x <- seq(0,r ^ 2, 0.01)
488.   sqrtX <- sqrt(x)
489.   y <- pbinom(sqrtX, size = r, prob = p)
490.   plot(sqrtX, y, col = "blue", lwd =2, type="l",
491.        xlab = "x", ylab = "F(x)",
492.        main = paste("Funcția de repartiție pentru X^2"))
493.   grid()
494. })
495.
496. #Pentru Sn = Binomial(n*r, p)
497. output$sumBinomial <- renderPlot({
498.   n <- input$sumBinN
499.   p <- input$sumP
500.   r <- input$sumR
501.   x <- seq(0,(r * n),0.01)
502.   y <- pbinom(x, size = r * n, prob = p)
503.   plot(x, y, col = "blue", lwd =2, type="l",
504.        xlab = "x", ylab = "F(x)",
505.        main = paste("Funcția de repartiție pentru Sum(Binomial(r,p)"))
506.   grid()
507. })
508. #####
509.
510. #Ex 2.
511.
512. #a) f(x) = cx^4, x din (0,2), c din R
513. #####
514.
515.
516. f_x_A <- reactive(function(x) (input$numerator / input$denominator) * x^4)
517.
518. calculate_results_A <- reactive({
519.   num <- input$numerator
520.   den <- input$denominator
521.   c <- num / den
522.
523.   #Integrala de validare
524.   integral_f <- integrate(f_x_A(), 0, 2)$value
525.
526.   #Medie E(x)
527.   average <- integrate(function(x) x * f_x_A()(x), 0, 2)$value
528.

```

```

529.     #Medie  $x^2 E(x^2)$ 
530.     averageSqr <- integrate(function(x) x^2 * f_x_A()(x), 0, 2)$value
531.
532.     #Varianta
533.     variance <- averageSqr - average^2
534.
535.     list(integral_f = integral_f, average = average, variance = variance)
536.
537. })
538.
539. output$densityA <- renderPlot({
540.     interval <- seq(0.001, 1.999, 0.001)
541.     f <- function(x) input$numerator/input$denominator * x^4
542.     f_x <- sapply(interval, f)
543.     plot(interval, f_x, col = "blue", lwd = 2,
544.          xlab = "x", ylab = "F(x)",
545.          main = paste("Densitatea pentru f(x) = cx^4"))
546.     grid()
547. })
548.
549. output$resultsA <- renderPrint({
550.     stats <- calculate_results_A()
551.
552.     #Verificam daca densitatea este valida
553.     if (abs(stats$integral_f - 1) > 1e-6) { #Luam in considerare eroare de rotunjire
554.         cat("C nu face ca f(x) să fie o densitate validă!\n")
555.         cat("Integrală =", stats$integral_f, "\n")
556.     }
557.     else{
558.         cat("Media E[X] =", stats$average, "\n")
559.         cat("Varianța Var(X) =", stats$variance, "\n")
560.     }
561. })
562.
563. #####
564.
565. #b)  $f(x) = ax + bx^2$ , x din (0,1), a,b din R
566. #####
567.
568. f_x_B <- reactive(function(x) input$ex2A * x + input$ex2B * x^2)
569.
570. calculate_results_B <- reactive({
571.
572.     #integrala de validare
573.
574.     integral_f <- integrate(f_x_B(), 0, 1)$value
575.
576.     #Medie E(x)
577.     average <- integrate(function(x) x * f_x_B()(x), 0, 1)$value
578.
579.     #Medie  $x^2 E(x^2)$ 
580.     averageSqr <- integrate(function(x) x^2 * f_x_B()(x), 0, 1)$value
581.
582.     #Varianta
583.     variance <- averageSqr - average^2
584.
585.     list(integral_f = integral_f, average = average, variance = variance)
586. })
587.
588. output$densityB <- renderPlot({
589.     interval <- seq(0.001, 0.999, 0.001)
590.     f <- function(x) input$ex2A * x + input$ex2B * x^2
591.     f_x <- sapply(interval, f)
592.     plot(interval, f_x, col = "blue", lwd = 2,
593.          xlab = "x", ylab = "F(x)",
594.          main = paste("Densitatea pentru f(x) = ax + bx^2"))
595.     grid()
596. })
597.
598. output$resultsB <- renderPrint({

```

```

599. stats <- calculate_results_B()
600. a <- input$ex2A
601. b <- input$ex2B
602. check <- TRUE
603.
604. if (abs(stats$integral_f - 1) > 1e-6 & check) {
605.   cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
606.   cat("Integrală =", stats$integral_f, "\n")
607.   check <- FALSE
608. }
609.
610. if(check & a < 0){
611.   cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
612.   check <- FALSE
613. }
614.
615. if(check & b < 0){
616.   if(a == 0 | a <= 0){
617.     cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
618.     check <- FALSE
619.   }
620.   else{
621.     if(a < -b)
622.     {
623.       cat("A, B nu fac ca f(x) să fie o densitate validă!\n")
624.       check <- FALSE
625.     }
626.   }
627. }
628.
629. if(check)
630. {
631.   cat("Media E[X] =", stats$average, "\n")
632.   cat("Varianța Var(X) =", stats$variance, "\n")
633. }
634. })
635. #####
636.
637. #c)  $f(x) = 4/x(x+1)(x+2)$ , x din  $N \setminus \{0\}$ 
638. #####
639.
640. f_x_C <- reactive(function(x) 4/(x*(x+1)*(x+2)))
641.
642. output$densityC <- renderPlot({
643.   dom <- 1 : 30
644.   pdf <- function(x) 4/((x+1)*(x+2))
645.   p_x <- sapply(dom, pdf)
646.   plot(dom, p_x, col = "blue", lwd = 2,
647.     xlab = "x", ylab = "F(x)",
648.     main = paste("X cu pdf = 4/(x(x+1)(x+2))"))
649.   grid()
650. })
651.
652. calculate_results_C <- reactive({
653.
654.   dom <- 1 : 30000
655.
656.   pdf_avg <- function(x) 4/((x+1)*(x+2))
657.   pdf_avg_sqr <- function(x) 4*x/((x+1)*(x+2))
658.
659.   pAvg_x <- sapply(dom, pdf_avg)
660.   pAvgSqr_x <- sapply(dom, pdf_avg_sqr)
661.
662.   #Media E[x]
663.   average <- sum(pAvg_x)
664.
665.   #Media E[X^2]
666.   averageSqr <- sum(pAvgSqr_x)
667.
668.   #Varianta

```

```

669.     variance <- averageSqr - average^2
670.
671.     list(variance = variance, average = average)
672. })
673.
674. output$resultsC <- renderPrint({
675.     stats <- calculate_results_C()
676.
677.     cat("Media E[X] =", stats$average, "\n")
678.     cat("Varianța Var(X) =", stats$variance, "\n")
679. })
680.
681. #d)  $f(x) = \lg(x/x+1)$ ,  $x$  din  $\{1, 2, \dots, 9\}$ 
682. #####
683.
684. f_x_D <- function(x) log10(x/(x+1))
685.
686. output$densityD <- renderPlot({
687.     dom <- 1:9
688.     p_x <- sapply(dom, f_x_D)
689.     plot(dom, p_x, col = "blue", lwd = 2,
690.          xlab = "x", ylab = "F(x)",
691.          main = paste("X cu pdf = lg(x/x+1)"))
692.     grid()
693. })
694.
695. calculate_results_D <- reactive({
696.
697.     dom <- 1 : 9
698.
699.     pdf_avg <- function(x) x * log10(x/(x+1))
700.     pdf_avg_sqr <- function(x) x^2 * log10(x/(x+1))
701.
702.     pAvg_x <- sapply(dom, pdf_avg)
703.     pAvgSqr_x <- sapply(dom, pdf_avg_sqr)
704.
705.     #Media E[x]
706.     average <- sum(pAvg_x)
707.
708.     #Media E[X^2]
709.     averageSqr <- sum(pAvgSqr_x)
710.
711.     #Varianta
712.     variance <- averageSqr - average^2
713.
714.     list(variance = variance, average = average)
715. })
716.
717. output$resultsD <- renderPrint({
718.     stats <- calculate_results_D()
719.
720.     cat("Media E[X] =", stats$average, "\n")
721.     cat("Varianța Var(X) =", stats$variance, "\n")
722. })
723.
724. #####
725. #e)  $f(x) = \frac{\text{teta}^2}{1+\text{teta}} * (1+x)e^{-\text{teta}*x}$ ,  $x > 0$ ,  $\text{teta} > 0$ 
726.
727. #t <- reactive(input$teta)
728. #f_x_E <- reactive(function(x) ((t^2)/(1+t))*(1+x)*(2.718)^(-t*x) )
729.
730. output$densityE <- renderPlot({
731.     t <- input$teta
732.     interval <- seq(0.001, 50, 0.001)
733.
734.     f <- function(x) ((t^2)/(1+t))*(1+x)*(exp(1))^(t*x)
735.     f_x <- sapply(interval, f)
736.
737.     plot(interval, f_x, col = "blue", lwd = 2,
738.          xlab = "x", ylab = "F(x)", type = "l",

```

```

739.     main = paste("X cu f(x) = teta^2/1+teta * (1+x)e^-teta*x"))
740.     grid()
741. })
742.
743. calculate_results_E <- reactive({
744.
745.     #integrala de validare
746.     t <- input$teta
747.     f_x_E <- function(x) ((t*t)/(1+t))*(1+x)*(exp(1))^(t*x)
748.
749.     integral_f <- integrate(f_x_E, 0, upper = Inf)$value
750.
751.     #Medie E(x)
752.     average <- integrate(function(x) x * f_x_E(x), 0, upper = Inf)$value
753.
754.     #Medie x^2 E(x^2)
755.     averageSqr <- integrate(function(x) x^2 * f_x_E(x), 0, upper = Inf)$value
756.
757.     #Varianta
758.     variance <- averageSqr - average^2
759.
760.     list(integral_f = integral_f, average = average, variance = variance)
761. })
762.
763. output$resultsE <- renderPrint({
764.     stats <- calculate_results_E()
765.
766.     if (abs(stats$integral_f - 1) > 1e-6) {
767.         cat("Teta nu face ca f(x) să fie o densitate validă!\n")
768.         cat("Integrală =", stats$integral_f, "\n")
769.     }
770.     else{
771.         cat("Media E[X] =", stats$average, "\n")
772.         cat("Varianța Var(X) =", stats$variance, "\n")
773.     }
774. })
775.
776. #####
777. #f) f(x) = 1/3e^x, x<0; f(x) = 1/3, 0<=x<1; f(x) = 1/3*x^1-x, x>= 1
778.
779. output$densityF <- renderPlot({
780.     interval1 <- seq(-10,-0.001, 0.001)
781.     interval2 <- seq(0, 0.999, 0.001)
782.     interval3 <- seq(1, 10, 0.001)
783.     fullInterval <- seq(-10,10,0.001)
784.
785.     f1 <- function(x) 1/3 * (exp(1))^x
786.     f2 <- function(x) 1/3
787.     f3 <- function(x) 1/3* (exp(1))^(1-x)
788.
789.     f_x_1 <- sapply(interval1, f1)
790.     f_x_2 <- sapply(interval2, f2)
791.     f_x_3 <- sapply(interval3, f3)
792.
793.
794.     plot(fullInterval, sapply(fullInterval, function(x) 0), type = "l", col = "white", lwd =
2, ylim = c(0, 1),
795.         xlab = "x", ylab = "f(x)", main = "Densitatea lui X din F")
796.     lines(interval1,f_x_1, col = "red", lwd = 2)
797.     lines(interval2,f_x_2, col = "blue", lwd = 2)
798.     lines(interval3,f_x_3, col = "green", lwd = 2)
799. })
800.
801. calculate_results_F <- reactive({
802.
803.     interval1 <- seq(-10,-0.001, 0.001)
804.     interval2 <- seq(0, 0.999, 0.001)
805.     interval3 <- seq(1, 10, 0.001)
806.
807.     f1 <- function(x) 1/3 * (exp(1))^x

```

```

808. f2 <- function(x) 1/3
809. f3 <- function(x) 1/3* (exp(1))^(1-x)
810.
811. #Calculam mediile pe ramuri
812. avg1 <- integrate(function(x) x * f1(x), lower = Inf, 0)$value
813. avg2 <- integrate(function(x) x * f2(x), 0, 1)$value
814. avg3 <- integrate(function(x) x * f3(x), 1, upper = Inf)$value
815.
816. #Calculam ponderile
817. P1 <- integrate(f1, lower = Inf, 0)$value
818. P2 <- integrate(Vectorize(f2), lower = 0, 1)$value
819. P3 <- integrate(f3, lower = 1, Inf)$value
820.
821. #Medie E(x)
822. average <- P1 * avg1 + P2 * avg2 + P3 * avg3
823.
824. avg1Sqr <- integrate(function(x) x^2 * f1(x), lower = Inf, 0)$value
825. avg2Sqr <- integrate(function(x) x^2 * f2(x), 0, 1)$value
826. avg3Sqr <- integrate(function(x) x^2 * f3(x), 1, upper = Inf)$value
827.
828. #Medie x^2 E(x^2)
829.
830. averageSqr <- P1*avg1Sqr + P2*avg2Sqr + P3*avg3Sqr
831.
832. #Varianta
833. variance <- averageSqr - average^2
834.
835. list(average = average, variance = variance)
836. })
837.
838. output$resultsF <- renderPrint({
839.   stats <- calculate_results_F()
840.
841.   cat("Media E[X] =", stats$average, "\n")
842.   cat("Varianta Var(X) =", stats$variance, "\n")
843. })
844. #####
845. #g) f(x) = 1/(pi*(1 + x^2))
846.
847. output$densityG <- renderPlot({
848.
849.   interval <- seq(-10, 10, 0.001)
850.   f <- function(x) 1/(pi * (1 + x^2))
851.   f_x <- sapply(interval, f)
852.   plot(interval, f_x, col = "blue", lwd = 2,
853.         xlab = "x", ylab = "f(x)", type = "l",
854.         main = paste("X cu f(x) = 1/(pi*(1 + x^2))"))
855.   grid()
856. })
857.
858. calculate_results_G <- reactive({
859.
860.   f_x_E <- function(x) 1/(pi * (1 + x^2))
861.
862.   #Medie E(x)
863.   average <- integrate(function(x) x * f_x_E(x), lower = -100, upper = 100)$value
864.
865.   #Medie x^2 E(x^2)
866.   averageSqr <- integrate(function(x) x^2 * f_x_E(x), lower = -100, upper = 100)$value
867.
868.   #Varianta
869.   variance <- averageSqr - average^2
870.
871.   list(average = average, variance = variance)
872. })
873.
874. output$resultsG <- renderPrint({
875.   stats <- calculate_results_G()
876.
877.   cat("Media E[X] =", stats$average, "\n")

```



```

878.     cat("Varianța Var(X) =", stats$variance, "\n")
879.   })
880.
881. }
882. shinyApp(ui = ui, server = server)

```

Cod Problema 3

```

1. # REZOLVARE a)
2. # REZOLVARE a)
3. # REZOLVARE a)
4. # REZOLVARE a)
5. # REZOLVARE a)
6. # REZOLVARE a)
7. # REZOLVARE a)
8. # REZOLVARE a)
9. # REZOLVARE a)
10. # REZOLVARE a)
11. # REZOLVARE a)
12. # REZOLVARE a)
13.
14. # Esantionul
15. sample_a <- c(8, 12, 6, 14, 9, 12, 15, 7, 15, 7, 10, 10, 14, 9, 12, 15, 11, 6, 8, 6, 8, 8,
9, 12, 13, 10, 11, 11, 13, 15, 10, 8, 7, 8, 13, 9, 9, 13, 12, 9, 10, 6, 10, 8, 10, 11, 12, 11, 9,
10, 7, 8, 8, 16, 7, 15, 10, 10, 8, 14, 13, 4, 11, 13, 6, 9, 13, 10, 10, 12, 11, 5, 6, 4, 9, 6, 9,
7, 13, 9, 11, 5, 5, 9, 15, 10, 11, 10, 14, 7, 11, 9, 14, 10, 5, 10, 8, 12, 13, 11)
16.
17. # Estimator MLE
18. theta_mle_a <- sum(sample_a) / (2 * length(sample_a))
19.
20. # Estimator MM
21. theta_mm_a <- mean(sample_a) / 2
22.
23. # Definirea functiei de log-verosimilitate
24. log_likelihood_a <- function(theta) {
25.   -2 * theta * length(sample_a) + sum(sample_a) * log(2 * theta) - sum(lfactorial(sample_a))
26. }
27.
28. # Maximizarea log-verosimilitatii
29. result_a <- optim(par = 1, fn = log_likelihood_a, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))
30.
31. # Estimator numeric
32. theta_numeric_a <- result_a$par
33.
34. # Afisare rezultate
35. cat("Estimator numeric pentru theta:", theta_numeric_a, "\n")
36. cat("Estimator MLE pentru theta:", theta_mle_a, "\n")
37. cat("Estimator MM pentru theta:", theta_mm_a, "\n")
38.
39.
40. # REZOLVARE b)
41. # REZOLVARE b)
42. # REZOLVARE b)
43. # REZOLVARE b)
44. # REZOLVARE b)
45. # REZOLVARE b)
46. # REZOLVARE b)
47. # REZOLVARE b)
48. # REZOLVARE b)
49. # REZOLVARE b)
50. # REZOLVARE b)
51. # REZOLVARE b)
52.
53. # Esantionul
54. sample_b <- c(3, 2, 1, 4, 2, 3, 4, 1, 3, 2, 2, 4, 2, 1, 7, 5, 4, 5, 5, 2, 3, 4, 3, 1, 2, 4,
1, 1, 2, 3, 1, 3, 1, 4, 1, 3, 1, 6, 1, 3, 3, 4, 3, 1, 3, 2, 2, 3, 2, 4, 1, 1, 2, 6, 3, 1, 3, 6,

```

```

1, 2, 3, 6, 3, 2, 2, 2, 4, 2, 1, 3, 3, 4, 2, 3, 4, 1, 4, 4, 6, 3, 3, 5, 2, 2, 2, 3, 1, 3, 1, 3,
3, 5, 3, 4, 3, 2, 4, 2, 3, 3)
55.
56. # Estimator MLE
57. theta_mle_b <- sum(sample_b) / (14 * length(sample_b))
58.
59. # Estimator MM
60. theta_mm_b <- mean(sample_b) / 14
61.
62. # Definirea functiei de log-verosimilitate
63. log_likelihood_b <- function(theta) {
64.   sum(dbinom(sample_b, size = 14, prob = theta, log = TRUE))
65. }
66.
67. # Maximizarea log-verosimilitatii
68. result_b <- optim(par = 0.5, fn = log_likelihood_b, method = "Brent", lower = 0, upper = 1,
control = list(fnscale = -1))
69.
70. # Estimator numeric
71. theta_numeric_b <- result_b$par
72.
73. # Afisare rezultate
74. cat("Estimator numeric pentru theta:", theta_numeric_b, "\n")
75. cat("Estimator MLE pentru theta:", theta_mle_b, "\n")
76. cat("Estimator MM pentru theta:", theta_mm_b, "\n")
77.
78. # REZOLVARE c)
79. # REZOLVARE c)
80. # REZOLVARE c)
81. # REZOLVARE c)
82. # REZOLVARE c)
83. # REZOLVARE c)
84. # REZOLVARE c)
85. # REZOLVARE c)
86. # REZOLVARE c)
87. # REZOLVARE c)
88. # REZOLVARE c)
89. # REZOLVARE c)
90.
91. # Esantionul
92. sample_c <- c(6.269128, 25.204245, 13.994878, 13.391437, 11.458827, 10.565065, 11.706398,
10.625808, 7.485952, 16.353358, 9.277565, 8.566438, 14.788638, 6.830955, 9.542004, 20.272463,
36.562137, 12.244005, 16.084879, 11.454008, 15.592298, 6.332908, 13.106441, 6.198981, 15.726780,
7.883712, 35.124934, 11.856011, 13.766200, 16.534869, 16.803648, 11.196542, 19.785629, 26.300717,
21.270154, 7.192149, 5.882948, 15.812796, 10.963237, 24.963600, 13.802383, 15.281262, 10.310398,
20.940469, 23.992540, 15.869985, 12.041726, 12.521264, 10.869006, 15.386514, 14.636832,
18.104562, 17.029779, 4.506616, 20.941222, 12.050877, 9.757833, 20.070802, 12.472900, 6.474476,
15.059776, 13.157344, 9.124414, 13.768482, 24.354934, 12.363936, 11.110749, 9.092514, 17.856801,
14.757801, 13.898665, 9.119410, 11.430184, 11.958829, 13.516191, 10.701083, 14.713596, 10.121266,
16.945351, 13.524070, 14.742403, 19.165805, 10.338392, 12.327837, 19.619227, 7.328246, 14.894399,
19.631003, 7.622796, 12.343832, 13.138183, 10.061520, 17.674638, 9.675168, 12.115561, 15.182861,
13.292479, 17.888244, 16.695139, 2.952334)
93.
94. # Parametrul alpha
95. alpha_c <- 7
96.
97. # Estimator MLE
98. theta_mle_c <- sum(sample_c) / (alpha_c * length(sample_c))
99.
100. # Estimator MM
101. theta_mm_c <- mean(sample_c) / alpha_c
102.
103. # Definirea functiei de log-verosimilitate
104. log_likelihood_c <- function(theta) {
105.   sum(dgamma(sample_c, shape = alpha_c, scale = theta, log = TRUE))
106. }
107.
108. # Maximizarea log-verosimilitatii
109. result_c <- optim(par = 1, fn = log_likelihood_c, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))

```

```

110.
111. # Estimator numeric
112. theta_numeric_c <- result_c$par
113.
114. # Afisare rezultate
115. cat("Estimator MLE pentru theta:", theta_mle_c, "\n")
116. cat("Estimator MM pentru theta:", theta_mm_c, "\n")
117. cat("Estimator numeric pentru theta:", theta_numeric_c, "\n")
118.
119. # REZOLVARE d)
120. # REZOLVARE d)
121. # REZOLVARE d)
122. # REZOLVARE d)
123. # REZOLVARE d)
124. # REZOLVARE d)
125. # REZOLVARE d)
126. # REZOLVARE d)
127. # REZOLVARE d)
128. # REZOLVARE d)
129. # REZOLVARE d)
130. # REZOLVARE d)
131.
132. # Esantionul
133. sample_d <- c(6, 3, 24, 24, 4, 56, 10, 13, 2, 28, 24, 2, 22, 11, 2, 8, 118, 2, 14, 19, 7, 9,
8, 189, 2, 9, 21, 6, 6, 2, 3, 2, 3, 18, 3, 2, 21, 1, 5, 9, 11, 13, 19, 76, 1, 5, 9, 4, 57, 1, 2,
16, 5, 2, 20, 8, 1, 40, 6, 4, 19, 6, 3, 2, 4, 9, 1, 5, 10, 12, 6, 525, 19, 6, 17, 2, 5, 159, 5,
62, 6, 3, 45, 21, 23, 3, 17, 2, 1, 1, 474, 15, 3, 3, 7, 7, 13, 4, 38, 4)
134.
135. # Estimator MLE
136. theta_mle_d <- mean(sample_d)
137.
138. # Estimator MM
139. theta_mm_d <- mean(sample_d)
140.
141. # Definirea functiei de log-verosimilitate
142. log_likelihood_d <- function(theta) {
143.   sum(dgeom(sample_d, prob = 1 / (1 + theta), log = TRUE))
144. }
145.
146. # Maximizarea log-verosimilității
147. result_d <- optim(par = 1, fn = log_likelihood_d, method = "Brent", lower = 0, upper = 1000,
control = list(fnscale = -1))
148.
149. # Estimator numeric
150. theta_numeric_d <- result_d$par
151.
152. # Afisare rezultate
153. cat("Estimator MLE pentru theta:", theta_mle_d, "\n")
154. cat("Estimator MM pentru theta:", theta_mm_d, "\n")
155. cat("Estimator numeric pentru theta:", theta_numeric_d, "\n")
156.
157.
158. # REZOLVARE e)
159. # REZOLVARE e)
160. # REZOLVARE e)
161. # REZOLVARE e)
162. # REZOLVARE e)
163. # REZOLVARE e)
164. # REZOLVARE e)
165. # REZOLVARE e)
166. # REZOLVARE e)
167. # REZOLVARE e)
168. # REZOLVARE e)
169. # REZOLVARE e)
170.
171. # Esantionul
172. sample_e <- c(3.5930579, 2.1027540, 1.7820777, 9.6550388, 6.8803846, 0.7388358, 2.9194654,
3.1178660, 1.2323236, 2.9776820, 1.1172078, 2.4184586, 3.3258971, 1.9498871, 2.6088612,
3.9535062, 3.0389107, 4.4226628, 3.9366318, 2.4551569, 5.2814487, 5.6778622, 4.7683935,
1.1581498, 3.1270783, 4.1473311, 7.4830426, 1.1342893, 1.7773392, 7.7510826, 1.3919927,

```

```

2.3613291, 2.6234826, 1.6562602, 1.4992235, 2.3455062, 3.8458809, 5.8333841, 3.3834034,
1.5202546, 3.1248186, 5.3029567, 3.6225571, 4.8309931, 3.1579595, 3.2640258, 3.9538891,
4.0796841, 4.0991772, 3.2779944, 2.5002127, 3.0654695, 1.6996010, 3.2175175, 1.9033087,
4.4052061, 2.3158379, 2.4778345, 5.4382190, 4.9141207, 6.0978745, 1.1428936, 3.5639106,
7.4541937, 7.7778289, 3.2859563, 0.7432908, 1.4442696, 3.6619932, 2.8361371, 4.3180773,
1.6763585, 4.4464154, 2.5049617, 0.4448735, 5.0518839, 3.4151834, 1.6823650, 5.4517583,
2.8212788, 2.1566837, 2.9893287, 1.6925123, 6.5197938, 4.2165408, 1.6728425, 2.7650830,
2.6742755, 2.9622047, 0.7809781, 1.3913415, 5.3430751, 2.4859925, 3.7329465, 6.3129236,
0.6635228, 3.7640343, 2.1850174, 4.3773328, 5.0931544)
173.
174. # Parametrul alpha
175. alpha_e <- 3
176.
177. # Estimator MLE
178. theta_mle_e <- sum(sample_e) / (alpha_e * length(sample_e))
179.
180. # Estimator MM
181. theta_mm_e <- mean(sample_e) / alpha_e
182.
183. # Definirea functiei de log-verosimilitate
184. log_likelihood_e <- function(theta) {
185.   sum(dgamma(sample_e, shape = alpha_e, scale = theta, log = TRUE))
186. }
187.
188. # Maximizarea log-verosimilității
189. result_e <- optim(par = 1, fn = log_likelihood_e, method = "Brent", lower = 0, upper = 100,
control = list(fnscale = -1))
190.
191. # Estimator numeric
192. theta_numeric_e <- result_e$par
193.
194. # Afișare rezultate
195. cat("Estimator MLE pentru theta:", theta_mle_e, "\n")
196. cat("Estimator MM pentru theta:", theta_mm_e, "\n")
197. cat("Estimator numeric pentru theta:", theta_numeric_e, "\n")
198.

```