Variabile Aleatoare Continue

Gherghina Roxana - Gradinaru Stefan - Rotaru Catalin - Zenoveiov Andrei

Grupa 231

Proiectul 1

Structura echipei

- Gherghina Roxana-Ioana -> cerintele 2 si 3, documentatia
- Gradinaru Stefan -> cerintele 1 si 8
- Rotaru Catalin -> cerintele 6 si pachetul in sine
- Zenoveiov Andrei -> cerintele 4 si 5

Cerinta 1

Pentru a calcula constata de normalizare trebuie intai sa varificam daca functia are valori negative,apoi calculam integrala , daca calculul esueaza inseamna ca integrarea este divergenta, iar daca integrala este egala cu 0 inseamna ca nu exista constata de normalizare altfel afisam constanta de normalizare.

Vectorize() este folosit pentru corectitudinea si optimizare.

```
const_normalizare <- function(f) {
  if(any(sapply(seq(-100, 100, length.out = 1000), f) < 0)){
#functia seq(-100,100,lenghth.out=1000) generaza o secventa de numere
#functia sapply aplica functia f pe elementele vectorul generat de seq()
#functia any() verifica daca exista valori negative; cauta elementele din vectori si returneaza TRUE or FALSE
    stop("Functie are valori negative negativa")}# daca se gaseste un element din vector negativ se afiseaza
  else {
    tryCatch(integ <- integrate(Vectorize(f), lower = -Inf, upper = Inf)$value,
#tryCatch returneaza o erroare si cuntinua sa ruleze putand pune o conditii si controla ce se intampla pe baza conditiilor.

#functia integrate() intregreza functia primita
          error= function(err)  {
             stop("Integrala e divergenta")})  # daca nu se poate realiza integrare f afiseaza
```

```
  if(integ == 0)
    {stop("Functia data nu are contanta de nor")}
#daca integrala are valoarea 0 nu exista constanta de normalizare
  const_norm <- 1/integ
  return(const_norm)}}
```

Cerinta 2

Pentru ca o functie sa poata fi densitate de probabilitate, trebuie sa indeplineasca urmatoarele proprietati (din curs):

1. f(x) >= 0 pentru toate valorile din domeniul de valori
2. $\int_{-\infty}^{\infty} f(x)dx$ = 1, dar avand in vedere ca f(x) = 0 pentru toate valorile din afara domeniului, aceasta proprietate este echivalenta cu integrala de la (margine inferioara) la (margine superioara) = 1

Rezolvarea functioneaza pentru cazul in care avem un interval a carui lungime este finita.

```
verif_dp <- function(func, i_inf, i_sup) {
 ok <- 0
#prorietatea 1
 if(any(sapply(seq(i_inf, i_sup, by = 0.05), func)) < 0) {
  ok <- 1
  return(ok)}
#proprietatea 2
 if((integrate(Vectorize(func), i_inf, i_sup)$value) != 1) {
  ok <- 2
  return(ok)}
 return(ok)}
```

Afisam mesajul corespunzator:

```
afisarea_verif_dp <- function(func, intrv) {
 okk = verif_dp(func, intrv)
 if(okk == 0) {
```

```
  print(TRUE)}
 if(okk == 1) {
  print("Functia nu este densitate de probabilitate, deoarece incalca prorietatea 1")}
 if(okk == 2) {
  print("Functia nu este densitate de probabilitate, deoarece incalca proprietatea 2")}}
```

Am testat codul pe exemplele din curs:

1. Functia f(x) = 3 este densitate de probabilitate pe intervalul [0, 1/3], dar nu si pe [0.1, 0.2].
2. Functia f(x) = $3x^2$ este densitate de probabilitate pe intervalul [0, 1].

Cerinta 3

Facem un obiect care are un constructor care primeste functia densitate de probabilitate si domeniul de valori. Inainte de a construi obiectul, verificam daca functia data ca parametru pentru constructor este densitate de probabilitate, folosind functia de la subpunctul 2.

```
setClass("va_cont", slots = list(densitate = "function", inf = "numeric", sup = "numeric"))
va_cont <- function(densitate, inf, sup) {
 if(verif_dp(densitate, inf, sup)==0) {
  object <- new("va_cont", densitate = densitate, inf = inf, sup = sup)
  return(object)
 } else {
  afisarea_verif_dp(densitate, inf, sup)}}
```

Facem o functie care calculeaza probabilitatea pentru un interval dat:

```
probabilitate <- function(object, i, j) {
 if(i >= object@inf && j <= object@sup) {
  integrate(Vectorize(object@densitate), i, j)$value}}
```

Metoda pentru afisarea obiectului va_cont:

```
setMethod("show", "va_cont", function(object) {
 print("Densitatea de probabilitate: ")
 print(object@densitate)
```

```
print(paste("Marginea inferioara ", object@inf))
print(paste("Marginea superioara ", object@sup)) })
```

Am creat, pentru a testa clasa, obiectul :

v <- va_cont(densitate = proba, 0, 1/3)

Si am calculat probabilitatea:

w <- probabilitate(v, 0.1, 0.2)

Cerinta 4

Functie ce afiseaza functia de repartitie si densitatea pentru o multime de valori aleatorii distribuite normal (sau Gaussian) -> p4n

Apel functie: nume_variabila <- p4n()

Functia genereaza valori distribuite gaussian pentru care calculeaza functia de repartitie normala si densitatea.

Valorile functiei de repartitie sunt afisate cu o histograma rosie, iar densitatea cu o linie portocalie.

-generarea valorilor-

dNormala <- rnorm(sample(1:100, 1), mean = 0, sd = 1)

rnorm() <- obtinem valori aleatorii distribuite uniform

sample() <- obtinem un intreg cuprins intre 1 si 100 (dimensiunea)

pnorm() <- calculam functia de repartitie pentru distributia obtinuta

seq() <- creeaza o secventa de numere

dnorm() <- calculeaza densitatea

xsz, ysz <- variabile pentru afisarea densitatii

lines(xsz, ysz) <- functia ce ploteaza linia cu valoarile densitatii

Functie ce afiseaza functia de repartitie si densitatea pentru o multime de valori aleatorii -> p4aprox

Apel functie -> nume_variabila <- p4aprox()

Functia genereaza valori aleatorii pentru care se aproximeaza functia de repartitie si densitatea normala.

Nota: datorita factorului aleator (atat valorile cat si numarul de valori), normala poate aproxima mai bine sau mai putin bine datele primite.

-generarea valorilor-

c(sample(1:100, sample(1:100,1)) <- generam un vector de dimensiune intre 1 si 100 de intregi aleatori intre 1 si 100.

Pentru mai multe detalii vezi p4n().

Cerinta 5

Functie ce calculeaza media unei variabile aleatoare continue -> media

Apel functie: media(f) unde 'f' este o functie ce reprezinta o variabila aleatoare continua.

Exemplu apel:

```
media(function(x){
  rep <- 2*x^3
  rep[x > 1] = 0
  rep[x < 0] = 0
  return(rep)})
```

Pentru a calcula media am folosit definitia din curs (integrate(x*f(x), -Inf, +Inf)).

Functie ce calculeaza dispersia unei variabile aleatoare continue -> dispersia

Apel functie: dispersia(f) unde 'f' este o functie ce reprezinta o variabila aleatoare continua.

Exemplu apel:

```
dispersia(function(x){
  rep <- 2*x^3
  rep[x > 1] = 0
  rep[x < 0] = 0
  return(rep) })
```

Pentru a calcula dispersia, am calculat momentele centrate de ordinul 2 (momentC(f, 2)).

Pentru mai multe detalii vezi momentC.

Functie ce calculeaza momentele centrate pana la un ordin dat -> momentC

Apel functie: momentC(f, ord) unde 'f' este o functie ce reprezinta o variabila aleatoare continua si 'ord' este ordinul momentelor calculate.

Exemplu apel:

```
momentC(function(x){
  rep <- 2*x^3
  rep[x > 1] = 0
  rep[x < 0] = 0
  return(rep) }, 4)
```

Pentru a calcula momentele centrate pana la un ordin dat am folosit definitia din curs. (integrate(arg, -Inf, +Inf) - unde arg este produsul diferentei lui x si media(f), la puterea ordinului 'ord', cu f(x),pentru oricare x)

Deasemenea, in cazul in care momentele centrate nu pot fi calculate pana la ordinul dat, o eroare este returnata.

Functie ce calculeaza momentele initiale pana la un ordin dat -> momenteInit

Apel functie: momenteInit(f, ord), unde 'f' este o functie ce reprezinta o variabila aleatoare continua si 'ord' este ordinul momentelor calculate.

Exemplu apel:

```
momenteInit(function(x){
  rep <- 2*x^3
  rep[x > 1] = 0
  rep[x < 0] = 0
  return(rep) }, 4)
```

Pentru a calcula momentele initiale pana la un ordin dat am folosit definitia din curs (integrate(arg, -Inf, +Inf) - unde arg este produsul lui x la puterea ordinului 'ord' cu f(x), pentru oricare x).

Cerinta 6

Functie ce afiseaza media unei variabile aleatoare g(X), unde X este repartitia uniforma si g o functie precizata de utilizator -> media_uniform

```
densitate <- function(x){ 1/(b-a) }  #functia de densitate pentru repartitia uniforma
Func_final = Multiply(Func, densitate)    #se inmultesc cele doua functii
medie <- integrate(Vectorize(Func_final), a, b)    #se calculeaza integrala conform definitiei E(Y) = E(h(X))
```

```
medie = as.numeric(medie[1])   #pentru a putea fi afisata ca o fractie
return(fractions(medie))    #se afiseaza media
```

Apel: media_uniform(function(x) {functie_utilizator}, parametrul1, parametrul2); parametrii repartitiei sunt dati la apelare.

Functie ce afiseaza densitatea conform formulei Var(X) = E(X^2) - E(X)^2 -> dispersia_uniform

```
Func1 = Multiply(Func, Func)
exp1 <- media_uniform(Func1, a, b)  #calculam mai intai media pentru X^2
exp2 <- media_uniform(Func, a, b)   #calculam apoi media pentru E(X)^2
dispersie = as.numeric(exp1[1]) - as.numeric(exp2[1])^2  #disperia calculata dupa
                                                          formula de mai sus
return(fractions(dispersie))     #afisarea dispersiei ca fractie
```

Apel: dispersia_uniform(function(x) {functie_utilizator}, parametrul1, parametrul2)

Functie ce inmulteste doua functii -> Multiply

Exemple de apeluri:

```
> media_uniform(function(x) {x}, 0, 1)
[1] 1/2
```

Cerinta 8

Functia rep primeste un string care reprezinta un tip de repartitie,daca denumirea repartitie nu exista se afiseaza Stringul introdus nu este valid"

Pentru stocarea informatiei am ales sa folosesc data.frame() care structureaza vectori asemanatori unei "matrici". Data frame suntliste cu accelasi numar de randusi si cu nume unice de randuri.

Data freameurile sunt utile doare pot fi folosite si ulterior.

#Amarioarei-Lab1.pdf

#Curs 5/prof. Cristian Niculescu

#http://dep2.mathem.pub.ro/pdf/didactice/Probabilitati%20si%20statistica.pdf

```
rep <- function(nume_repartitie){
  if(nume_repartitie=="Repartitie_binominala_extern")
    {binominala
  }else if(nume_repartitie=="Repartitie_poisson_extern")
  {poisson}
  else if(nume_repartitie=="Repartitie_continua_uniforma_extern")
  {continua_uniforma}
  else if(nume_repartitie=="Repartitie_gamma_extern")
  {gamma_}
  else if(nume_repartitie=="Repartitie_beta_extern")
  {beta_}
  else if(nume_repartitie=="Repartitie_X2_extern")
  {helmert}

  else if(nume_repartitie=="Repartitie_uniforma")
  {curs_uniforma}
  else if(nume_repartitie=="Repartitie_exponentiala")
  {curs_exponentiala}
  else if(nume_repartitie=="Repartitie_normala")
  {curs_normala}
  else stop("Stringul introdus nu este valid")

}
rep(binominala)
binominala <- data.frame(
                  " " =
c("Notiune:","Definitie:","Definitie","Definitie","Media:","Dispersia:","Sursa:"),
                  " "=c("Repartitia Binominala",
                        "Variabila  aleatoare  discretă X urmează legea binomială(X are o repartiţie
binomială) cu parametrii",
```

"n şi p (n∈ℕ, 0<p<1) dacă ia valorile 0,1,2,....,n cu probabilităţile",

"P(X=k) =Combinari de n luate cate k de p^k*q^n-k, k=0,1,2,....,n, unde q=1-p", " E(X)=np", "Var(X)=npq.",
"http://dep2.mathem.pub.ro/pdf/didactice/Probabilitati%20si%20statistica.pdf"))

Bibliografie

1. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/Normal
2. https://www.datacamp.com/community/tutorials/make-histogram-basic-r
3. https://rdatatable.gitlab.io/data.table/library/xts/html/period.apply.html
4. https://www.datacamp.com/community/tutorials/r-tutorial-apply-family
5. https://www.datacamp.com/community/tutorials/r-objects-and-classes
6. https://stat.ethz.ch/R-manual/R-devel/library/methods/html/setClass.html
7. https://study.com/academy/lesson/any-and-all-functions-in-r-programming.html
8. http://www.endmemo.com/r/seq.php
9. https://r-coder.com/sapply-function-r/
10. https://en.wikipedia.org/wiki/Normalizing_constant
11. http://dep2.mathem.pub.ro/pdf/didactice/Probabilitati%20si%20statistica.pdf