UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

PROIECT

PROBABILITĂȚI ȘI STATISTICĂ

PROFESOR COORDONATOR:

COJOCEA MANUELA-SIMONA

ECHIPA DE PROIECT:

MAXIM TIBERIU (LIDER ECHIPĂ)

IACOV ANDREEA-LUCIA

IONESCU ALEXANDRU-THEODOR

VÎNAGA MĂDĂLINA

BUCUREȘTI 2021

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

MIIV - PACHET PENTRU LUCRUL CU VARIABILE ALEATOARE CONTINUE

PROFESOR COORDONATOR:

COJOCEA MANUELA-SIMONA

ECHIPA DE PROIECT:

MAXIM TIBERIU (LIDER ECHIPĂ)

IACOV ANDREEA-LUCIA

IONESCU ALEXANDRU-THEODOR

VÎNAGA MĂDĂLINA

BUCUREȘTI 2021

CUPRINS

1. Introducere

Variabilele aleatoare descriu un concept referitor la studiul matematic al fenomenelor aleatoare.

Definiție *O funcție* $X: E \to \mathrm{R}$ *se numește variabilă aleatoare dacă, pentru orice* $x \in R$*, are loc:*

$$\{X \leq x\} = \{e \in E \mid X(e) \leq x\} \in K$$

Definiție *O variabilă aleatoare X este una de tip continuu dacă funcția de repartiție* $F: R \to [0,1]$ *este continuă și există o funcție* $f: R \to \mathrm{R}$ *cu o mulțime cel mult numărabilă de puncte de discontinuitate de speța I astfel încât:*

$$F(x) = \int_{-\infty}^{\infty} f(t)dt\,, \forall\, x \in R$$

2. Descrierea Problemei și a Soluției

Cerința 1

Enunț:

Fiind dată o funcție f, introdusă de utilizator, să se determine constata de normalizare k. În cazul în care o asemenea constantă nu există, se va afișa un mesaj corespunzător către utilizator.

Rezolvare:

Pentru calcularea constantei de normalizare k, în raport cu o funcție introdusă de către utilizator, se aplică formula:

$$k = \frac{1}{\int_{-\infty}^{\infty} f(x)dx}$$

Funcția *constanta_normalizare* primește ca parametru funcția f și returnează valoarea constantei de normalizare, dacă aceasta poate fi determinată. În caz contrar, funcția afișează un mesaj de informare: *„Nu se poate determina constanta de normalizare!”*.

Implementarea codului R:

```r
constanta_normalizare <- function(f) {
  # folosim blocul tryCatch pentru a verifica dacă integrala este convergentă
  tryCatch(

    # aplicăm formula pentru constanta de normalizare
    return(1 / integrate(f = Vectorize(f), lower = -Inf, upper = Inf) $ value),

    error = function(err) {
      # dacă funcția nu are constantă de normalizare, se afișează următorul mesaj
      message(paste("Nu se poate determina constanta de normalizare!\n"))
      message(paste(err, "\n"))
      return (NA)
    }
  )
}
```

Teste ale codului implementat:

```
> constanta_normalizare(function(x) {exp(-x^2)})
[1] 0.5641896
> constanta_normalizare(function(x) {exp(x)})
[1] "Nu se poate determina constanta de normalizare!"
```

Cerința 2

Enunț:

Să se verifice dacă o funcție introdusă de utilizator este densitate de probabilitate.

Rezolvare:

Pentru ca o funcție să fie densitate de probabilitate, aceasta trebuie să îndeplinească două condiții:

$$1.\, f(x) \geq 0, \forall\, x\ din\ suport$$

$$2.\, \int_{-\infty}^{\infty} f(x)dx = 1\ sau\ P(-\infty\ < X\ <\ \infty) = 1$$

Funcția *verifica_densitate* primește ca parametru funcția f și returnează TRUE sau FALSE, și un mesaj informativ în funcție de rezultatul returnat. Funcția simulează un interval care poate fi comparat cu intervalul (-Inf, Inf) și generează valorile funcției primite ca parametru pe intervalul simulat.

În cazul în care numărul valorilor obținute mai mici decât 0 este nenul, atunci funcția dată nu îndeplinește prima condiție, deci nu este densitate de probabilitate.

Dacă prima condiție este îndeplinită se verifică valoarea integralei funcției să fie egală cu 1. În cazul în care nu este egală cu 1, funcția nu este densitate de probabilitate.

Dacă și a doua condiție a fost îndeplinită, se returnează valoarea TRUE și un mesaj favorabil (*„Funcția este densitate de probabilitate.”*). În cazurile când funcția nu îndeplinește una dintre cele două condiții, se va returna valoarea FALSE și mesajul *„Funcția NU este densitate de probabilitate.”*.

Implementarea codului R:

```
verifica_densitate <- function(f) {
  tryCatch(
    {
      # Interval simulat care poate fi comparat cu (-Inf,Inf)
      interval <- seq(-99999, 99999, 0.01)

      # Prin intermediul comenzii sapply(), se generează valorile funcției f pe interval
      valori <- sapply(interval, f)

      # Verificăm prima condiție: f(x) >= 0
      if (sum(valori < 0) > 0) {
        error
      }

      # Calculăm integrală din funcția introdusă pe -Inf, Inf
      integrala <- integrate(f = Vectorize(f), lower = -Inf, upper = Inf) $ value

      # Verificăm dacă valoarea integralei este egală cu 1 (a doua condiție), ținând cont
       de posibilele erori
      if (integrala < 0.99 && integrala > 1.001) {
        error
      }

      # Dacă funcția îndeplinește ambele condiții, rezultă că este densitate de
       probabilitate.
      print("Funcția este densitate de probabilitate.")
      return (TRUE)
    },

    # Afișare mesaj, pentru cazul contrar.
      error = function(err) {
        print("Funcția NU este densitate de probabilitate.")
        return (FALSE)
    }
  )
}
```

Teste ale codului implementat:

```
> verifica_densitate(function(x) {if((x>=0)&&(x<=pi/2)){cos(x)}else{0}})
[1] "Funcţia este densitate de probabilitate."
[1] TRUE
> verifica_densitate(function(x) {if((x>=0)&&(x<=pi)){cos(x)}else{0}})
[1] "Funcţia NU este densitate de probabilitate."
[1] FALSE
```

Cerința 3

Enunț:

Să se creeze un obiect de tip variabilă aleatoare continuă pornind de la o densitate de probabilitate introdusă de utilizator. Funcția trebuie să aibă opțiunea pentru variabile aleatoare unidimensionale, respectiv bidimensionale.

Rezolvare:

Am definit clasa VAC (Variabile Aleatoare Continue), care primește ca atribute *densitate (function)*, reprezentând densitatea de probabilitate a variabilei aleatoare, *bidimensionala (logical)*, reprezentând o variabilă de control prin care utilizatorul specifică dacă această variabilă este unidimensională (FALSE) sau bidimensională (TRUE), și *suport (list)*, reprezentând intervalul pe care este definită funcția.

Am creat un constructor, care primește ca valori implicite *bidimensională = FALSE* și *list = list(c(-Inf, Inf))*.

Dificultăți în realizarea cerinței:

Clasa trebuia să conțină câteva funcții (calcularea mediei, dispersiei, afișarea atributelor), însă, din cauza unor erori, legate de lucrul cu obiectele în R, acest lucru nu a fost posibil.

Implementarea codului R:

```r
# Definirea clasei
setClass("VAC",
         slots = list(densitate = "function", bidimensionala = "logical", suport = "list"))

# Constructor clasa
VAC <- function(densitate, bidimensionala = FALSE, suport = list(c(-Inf, Inf))) {
  res <- new("VAC", densitate = densitate, bidimensionala = bidimensionala, suport = suport)
  return (res)
}
```

Cerința 4

Enunț:

Să se reprezinte grafic densitatea și funcția de repartiție pentru diferite valori ale parametrilor repartiției. În cazul în care funcția de repartiție nu este dată într-o formă explicită (ex. repartiția normală), se acceptă reprezentarea grafică a unei aproximări a acesteia.

Rezolvare:

Am realizat câte o funcție pentru afișarea graficului densității și câte o funcție pentru reprezentarea grafică a funcției de repartiție. Pentru culoarea graficului, am folosit librăria *RColorBrewer.* Apelul funcției *meniu_afisare_grafice* deschide, în consolă, o zonă interactivă prin care utilizatorul poate selecta ce dorește să afișeze.

Dificultăți în realizarea cerinței:

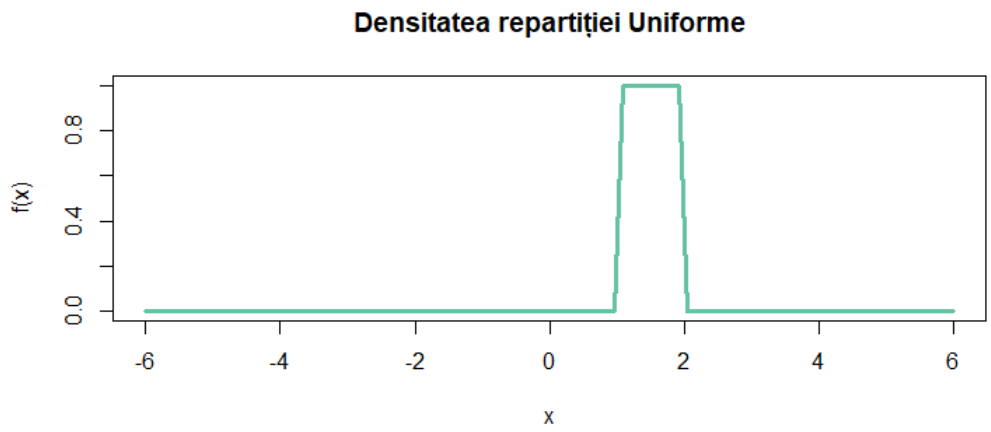
Am încercat să construim și funcția pentru o repartiție oarecare, însă nu am găsit o modalitate prin care să folosim repartiția dată de utilizator. Din această cauză, în script apare doar funcția *grafic_densitate_oarecare*, comentată.

Implementarea codului R:

Atașăm doar unul dintre cazurile posibile:

```
library("RColorBrewer")

# 1. Repartiție uniformă
grafic_densitate_uniforma <- function(min, max){
  curve(expr = dunif(x = x, min = min, max = max),
        from = -3 * max,
        to = 3 * max,
        main = "Densitatea repartiției Uniforme",
        col = brewer.pal(n = 3, name = "Set2"),
        lwd = 3,
        ylab = "f(x)"
  )
}

grafic_repartitie_uniforma <- function(min, max){
  curve(expr = punif(q = x, min = min, max = max),
        from = -3 * max,
        to = 3 * max,
        main = "Funcția de repartiție Uniformă",
        col = brewer.pal(n = 3, name = "Set2"),
        lwd = 3,
        ylab = "F(x)"
  )
}

meniu_afisare_grafice <- function() {
  cat("Introduceți un număr din lista de mai jos:\n")
  cat("1. Afișare Densitate Repartiție\n")
  cat("2. Afișare Funcție de Repartiție\n")

  enter <- as.numeric(readline(prompt = "Numărul ales este: "))
  switch(EXPR = enter,
         meniu_afisare_densitati(),
         meniu_afisare_functie_repartitie()
  )
}
```

Teste ale codului implementat:



Cerința 5

Enunț:

Să se calculeze media, dispersia și momentele inițiale și centrate până la ordinul 4, dacă există. Atunci când unul dintre momente nu există, se va afișa un mesaj corespunzător către utilizator.

Rezolvare:

Am construit funcții, pentru fiecare elemente cerute în enunțul problemei. La final, am apelat toate cele patru funcții, iar rezultatele le-am introdus într-o listă, pentru a fi ușor accesibile utilizatorului.

Pentru a calcula toate cele patru elemente, am folosit următoarele formule:

$$Media: \int_{-\infty}^{\infty} x * f(x)dx$$

$$Dispersia: \int_{-\infty}^{\infty} (x - medie) * f(x)dx$$

$$Momentul\ initial\ de\ ordin\ i: \int_{-\infty}^{\infty} x^i * f(x)dx$$

$$Momentul\ centrat\ de\ ordin\ i: \int_{-\infty}^{\infty} (x - medie)^i * f(x)dx$$

Implementarea codului R:

```
# Funcție care calculează media
medie <- function(f) {
  tryCatch(
    {
      prod <- function(x) {
        x * f(x)
      }
      return (integrate(f = Vectorize(prod), lower = -Inf, upper = Inf) $ value)
    },
    error = function(err) {
      message(paste("Nu se poate calcula media!\n"))
      message(paste(err, "\n"))
      return (NA)
    }
  )
}

# Funcție care calculează dispersia
dispersie <- function(f) {
  tryCatch(
    {
      prod <- function(x) {
        (x - medie(f)) ^ 2 * f(x)
      }
      return (integrate(f = Vectorize(prod), lower = -Inf, upper = Inf) $ value)
    },
    error = function(err) {
      message(paste("Nu se poate calcula dispersia!\n"))
      message(paste(err, "\n"))
      return (NA)
    }
  )
}

# Funcție care calculează momentele inițiale
momente_initiale <- function(f) {
  mom_ini <- list()
  for (i in 1:4) {
    tryCatch(
      {
        prod <- function(x) {
          x ^ i * f(x)
        }
        int <- integrate(f = Vectorize(prod), lower = -Inf, upper = Inf) $ value
        mom_ini <- append(mom_ini, int)
      },
      error = function(err) {
        message(paste("Nu se poate calcula momentul initial de ordin ", i, "!\n"))
      }
    )
  }
  return (mom_ini)
}
```

```r
# Funcție care calculează momentele centrate
momente_centrate <- function(f) {
  mom_cen <- list()
  for (i in 1:4) {
    tryCatch(
      {
        prod <- function(x) {
          (x - medie(f)) ^ i * f(x)
        }
        int <- integrate(f = Vectorize(prod), lower = -Inf, upper = Inf) $ value
        mom_cen <- append(mom_cen, int)
      },
      error = function(err) {
        message(paste("Nu se poate calcula momentul centrat de ordin ", i, "!\n"))
      }
    )
  }
  return (mom_cen)
}

# Funcție care adaugă rezultatele apelurilor funcțiilor de mai sus într-o singură listă
m_d_mi_mc <- function(f) {
  result_1 <- medie(f)
  result_2 <- dispersie(f)
  result_3 <- momente_initiale(f)
  result_4 <- momente_centrate(f)
  result <- list(
    "medie" = result_1,
    "dispersie" = result_2,
    "momente_initiale" = result_3,
    "momente_centrate" = result_4
  )
}
```

Teste ale codului implementat:

```
> exemplu <- m_d_mi_mc(g)
> exemplu
$medie
[1] 1.570796

$dispersie
[1] 0.4674021

$momente_initiale
$momente_initiale[[1]]
[1] 1.570796

$momente_initiale[[2]]
[1] 2.934802

$momente_initiale[[3]]
[1] 6.078362

$momente_initiale[[4]]
[1] 13.48692
```

```
$momente_centrate
$momente_centrate[[1]]
[1] 2.337732e-06

$momente_centrate[[2]]
[1] 0.4674021

$momente_centrate[[3]]
[1] 3.280958e-07

$momente_centrate[[4]]
[1] 0.4792553
```

Cerința 6

Enunț:

Să se calculeze media și dispersia unei variabile aleatoare g(X), unde X are o repartiție continuă cunoscută, iar g este o funcție continuă precizată de utilizator.

Rezolvare:

Pentru rezolvarea acestei probleme, am folosit următoarele două formule:

$$Media = \int_{-\infty}^{\infty} g(x) * f(x)dx$$

$$Dispersia = \int_{-\infty}^{\infty} (g(x) - media)^2 * f(x)dx$$

Implementarea codului R:

```
medie_dispersie <- function(g, f) {
  #media variabilei aleatoare g(X)
  prod_func <- function(x) {
    return (g(x) * f(x))
  }
  medie <- integrate(f = Vectorize(prod_func), lower = -Inf, upper = Inf) $ value

  #dispersia variabilei aleatoare g(X)
  prod_func_patrat <- function(x) {
    return ((g(x) - medie) ^ 2 * f(x))
  }
  dispersie <- integrate(f = Vectorize(prod_func_patrat), lower = -Inf, upper = Inf) $
value

  result <- list("medie" = medie, "dispersie" = dispersie)
  return(result)
}
```

Teste ale codului implementat:

```
> f <- function(x) dnorm(x, 0.5, 1)
> g <- function(x) dlnorm(x, 0.7, 1)
> medie_dispersie(f, g)
$medie
[1] 0.1682298

$dispersie
[1] 0.02573259
```

Cerința 7

Enunț:

Să se creeze o funcție P care permite calculul diferitelor tipuri de probabilități asociate unei variabile aleatoare continue, similar funcției P din pachetul ***discreteRV***.

Rezolvare:

Am realizat funcția P(), care deschide, în consolă un meniu cu opțiuni. Utilizatorul este rugat să aleagă una dintre cele două opțiuni puse la dispoziție (<, >).

Dificultăți în realizarea cerinței:

Am dorit implementarea a mai multor cazuri, însă soluția gândită din noi consta în crearea unei clase care să prelucreze obiecte de acest tip, obiectele supraîncărcând „operatorii” <, >, | etc. Dat fiind faptul că, nici la cerința 3 nu am reușit să modelăm o clasă, am recurs la soluția prezentată mai jos.

Implementarea codului R:

```
P <- function(f) {
  cat("1. <\n")
  cat("2. >\n")
  enter <- as.numeric(readline(prompt = "Optiunea aleasa este: "))
  limita <- as.numeric(readline(prompt = "Capatul integralei este: "))

  switch(EXPR = enter,
         {
           res <- integrate(f = Vectorize(f), lower = -Inf, upper = limita) $ value
           return (res)
         },
         {
           res <- integrate(f = Vectorize(f), lower = limita, upper = Inf) $ value
           return (res)
         })
}
```

Teste ale codului implementat:

```
> g <- function(x) {
+   if((x >= 0) && (x <= pi)) {sin(x) / 2} else {0}
+ }
> P(g)
1. <
2. >
Optiunea aleasa este: 1
Capatul integralei este: 1.58
[1] 0.5046003
```

Cerința 8

Enunț:

Afișați o „fișă de sinteză” care să conțină informații de bază despre respectiva repartiție (cu precizarea sursei informației). Relevant aici ar fi să precizați pentru ce e folosită în mod uzual acea repartiție, semnificația parametrilor, media, dispersia etc.

Rezolvare:

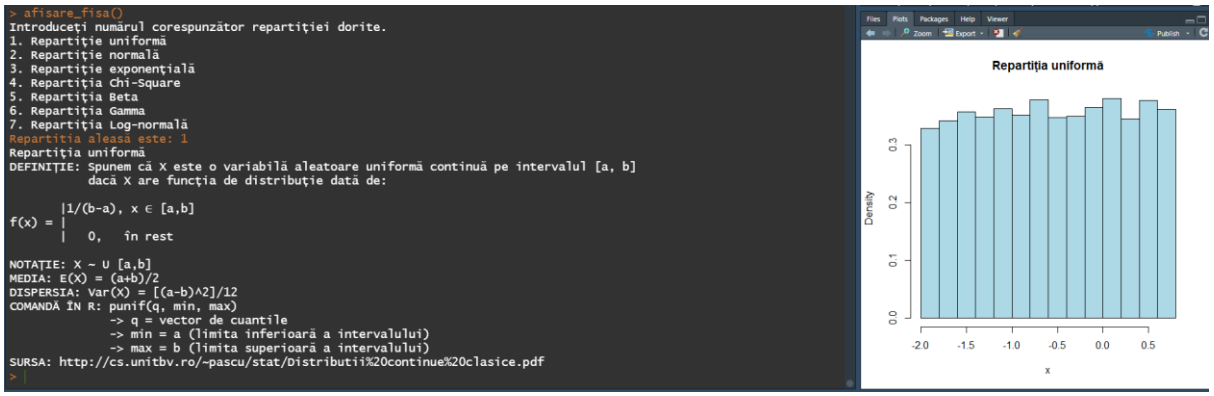
Pentru rezolvarea acestei cerințe, am realizat o funcție care prezintă un meniu cu fișele de sinteză aferente repartițiilor implementate în cod. Când utilizatorul selectează o repartiție, se va apela funcția care afișează fișa de sinteză pentru respectiva repartiție. Funcțiile au fost implementate în directorul *Repartitii*, iar afișarea acesteia se face în consolă, dar și în zona din dreapta, unde se va afișa o histogramă a repartiției alese.

Implementarea codului R:

```
afisare_fisa <- function() {
  cat("Introduceți numărul corespunzător repartiției dorite.\n")
  cat("1. Repartiție uniformă\n")
  cat("2. Repartiție normală\n")
  cat("3. Repartiție exponențială\n")
  cat("4. Repartiția Chi-Square\n")
  cat("5. Repartiția Beta\n")
  cat("6. Repartiția Gamma\n")
  cat("7. Repartiția Log-normală\n")
  enter <- as.numeric(readline(prompt = "Repartitia aleasă este: "))

  switch(  EXPR = enter,
           {
             uniforma()
           },
           {
            normala()
           },
           {
             exponential()
           },
           {
             chisquare()
           },
           {
             beta()
           },
           {
             gamma()
           },
           {
             log_normal()
           }
  )
}
```

```r
# Repartiția uniformă
uniforma <- function(){
  cat("Repartiția uniformă\nDEFINIȚIE: Spunem că X este o variabilă aleatoare u
niformă continuă pe intervalul [a, b]
           dacă X are funcția de distribuție dată de: \n
       |1/(b-a), x ∈ [a,b]
f(x) = |
       |   0,   în rest \n\n")

  cat("NOTAȚIE: X ~ U [a,b] \n")
  cat("MEDIA: E(X) = (a+b)/2 \n")
  cat("DISPERSIA: Var(X) = [(a-b)^2]/12 \n")
  cat("COMANDĂ ÎN R: punif(q, min, max)
              -> q = vector de cuantile
              -> min = a (limita inferioară a intervalului)
              -> max = b (limita superioară a intervalului) \n")
  hist(runif(10000, min = -2, max = 0.8), freq = FALSE, xlab = 'x', main = "Rep
artiția uniformă", col = "lightblue")
  cat("SURSA: http://cs.unitbv.ro/~pascu/stat/Distributii%20continue%20clasice.
pdf")
}
```

Teste ale codului implementat:



Cerința 10

Enunț:

Să se calculeze covarianța și coeficientul de corelație pentru două variabile aleatoare continue. Trebuie să folosiți densitatea comună a celor două variabile aleatoare.

Rezolvare:

Se verifică dacă suportul lui X, respectiv Y, conține -Inf sau Inf. În acest caz, nu se pot efectua calculele.

Pentru a calcula densitățile marginale și mediile se folosesc formulele:

$$f_x(x) = \int_{-\infty}^{\infty} f(x,y)\,dy$$

$$E[X] = \int_{-\infty}^{\infty} x * f(x)dx$$

Covarianța s-a calculat folosind formula:

$$Cov(X,Y) = E[(X - E[X]) * (Y - E[Y])]$$

Variațele și coeficientul de corelație se calculează folosind:

$$Var(X) = E[(X - E[X])^2]$$

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X) * Var(Y)}}$$

Implementarea codului în R:

```
suport_verif <- function(sX, sY) {
  if(sX[1] == -Inf || sX[2] == Inf){
    return (FALSE)
  }

  if(sY[1] == -Inf || sY[2] == Inf){
    return (FALSE)
  }

  return (TRUE)
}


covarianta_corelatie <- function(f, sX, sY){
  if(suport_verif(sX, sY) == FALSE){
    message(print("Un capăt al suportului este -Inf sau Inf!\n"))
    return(NA)
  }

  # caclulează cele două densități marginale
  densitate_marginala_X <- Vectorize(function(x) {
    integrate(f = function(y){ f(x,y) },
              lower = sY[1],
              upper = sY[2]) $ value
  })

  densitate_marginala_Y <- Vectorize(function(y) {
    integrate(f = function(x){ f(x,y) },
              lower = sX[1],
              upper = sX[2]) $ value
  })
```

```
  # calculează media lui X, repesctiv Y
  medie_X <- integrate(f = function(x) { return (x * densitate_marginala_X(x)) },
                       lower = sX[1],
                       upper = sX[2]) $ value

  medie_Y <- integrate(f = function(y) { return (y * densitate_marginala_Y(y)) },
                       lower = sY[1],
                       upper = sY[2]) $ value

  cov <- function(x, y) { return ((x - medie_X) * (y - medie_Y) * f(x, y))}

  covarianta <- integrate(f = Vectorize(function (y) {
                            integrate(f = function (x) { cov(x, y) },
                                      lower = sX[1],
                                      upper = sX[2]) $ value
                            }),
                          lower = sY[1],
                          upper = sY[2]) $ value

  # calculăm variațele celor două variabile aleatoare
  varianta_X <- integrate(f = function(x) { return ((x - medie_X) ^ 2 * densitate_m
arginala_X(x)) },
                          lower = sX[1],
                          upper = sX[2]) $ value

  varianta_Y <- integrate(f = function(y) { return ((y - medie_Y) ^ 2 * densitate_m
arginala_Y(y)) },
                          lower = sY[1],
                          upper = sY[2]) $ value

  coeficient_corelatie <- covarianta / sqrt(varianta_X * varianta_Y)

  res <- list("covarianta" = covarianta, "coeficient_corelatie" = coeficient_corela
tie)
  return (res)
}
```

Teste ale codului implementat:

```
> covarianta_corelatie(f, sX, sY)
$covarianta
[1] 0.01851852

$coeficient_corelatie
[1] 0.2325581
```

Cerința 11

Enunț:

Pornind de la densitatea comună a două variabile aleatoare continue, să se construiască densitățile marginale continue și densitățile condiționate.

Rezolvare:

Am folosit formulele de calcul pentru densitățile marginale menționate la cerința 11. De asemenea, am calculat cele două densități condiționate, folosind cele două densități marginale.

$$f_1(x|y) = \frac{f(x,y)}{f_y(y)}$$

$$f_2(y|x) = \frac{f(x,y)}{f_x(x)}$$

Implementarea codului în R:

```
densitati_marginale_conditionate <- function(f, sX, sY) {
  densitate_marginala_X <- Vectorize(function(x) {
    integrate(f = function(y){ f(x, y) },
              lower = sY[1],
              upper = sY[2]) $ value
  })

  densitate_marginala_Y <- Vectorize(function(y) {
    integrate(f = function(x){ f(x, y) },
              lower = sX[1],
              upper = sX[2]) $ value
  })

  # densitate_conditionata_X <- Vectorize(function(y) {
  #   g <- function(x) {
  #     f(x, y)
  #   }
  #   return (g / densitate_marginala_Y(y))
  # })
  #
  # densitate_conditionata_Y <- Vectorize(function(x) {
  #   g <- function(y) {
  #     f(x, y)
  #   }
  #   return (g / densitate_marginala_Y(x))
  # })
}
```

Cerința 12

Enunț:

Să se construiască suma și diferența a două variabile aleatoare continue independente, folosind formula de convoluție.

Rezolvare:

Am folosit formulele:

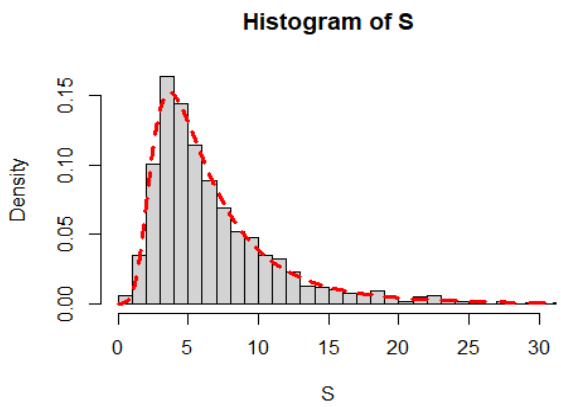
$$Suma: f_z = \int_{-\infty}^{\infty} f(x) * \big(g(z-x)\big)dx$$

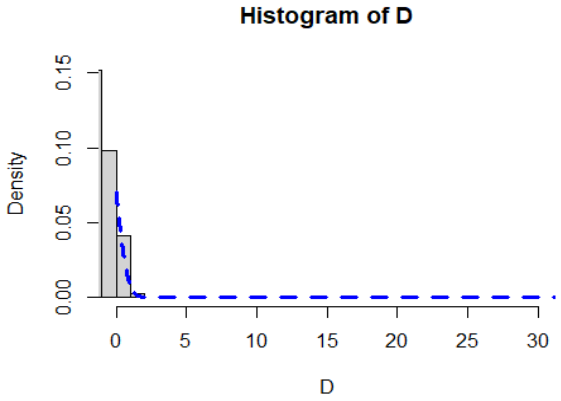
$$Diferenta: f_z = \int_{-\infty}^{\infty} f(x) * \big(g(x-z)\big)dx$$

Implementarea codului în R:

```r
# suma a doua variabile aleatoare, folosind formula de convolutie
sumaVA_convolutie <- function(f, g) {
  function(z) {
    integrate(
      f = function(x) {
        f(x) * g(z - x)
      },
      lower = -Inf,
      upper = Inf) $ value
  }
}

# diferenta a doua variabile aleatoare, folosind formula de convolutie
difVA_convolutie <- function(f, g) {
  function(z) {
    integrate(
      f = function(x) {
        f(x) * g(x - z)
      },
      lower = -Inf,
      upper = Inf) $ value
  }
}
```

Teste ale codului implementat:





3. Concluzii

Pachetul construit de echipa noastră se bazează pe un fundament matematic și are ca scop facilitarea lucrului cu variabilele aleatoare. Prin crearea acestui proiect, am realizat un model prin care elementele teoretice, care pot părea abstracte pentru mulți utilizatori, pot fi folosite în aplicații concrete din viața reală. Astfel, construirea acestui pachet a transformat teoria probabilităților în instrumente informatice.

Concluzionând, conform descrierilor prezentate anterior în documentație, au fost create soluții optime pentru diferitele probleme apărute în lucrul cu variabilele aleatoare.

4. Bibliografie

https://www.afahc.ro/ro/facultate/cursuri/luculescu/4.%20Functii%20de%20repartitie.pdf

http://cs.unitbv.ro/~pascu/stat/Distributii%20continue%20clasice.pdf

http://math.etc.tuiasi.ro/rstrugariu/cursuri/SPD2015/c7.pdf

http://math.etc.tuiasi.ro:81/rosu/didactic/MS%20II_Curs_Variabile%20aleatoare%20continue_I.pdf

https://rdocumentation.org/packages/stats/versions/3.6.2

https://en.wikipedia.org/wiki/Cauchy_distribution