Grupa 264 | Proiect 1

Proiect: Probabilități și statistică

Megherlich Andreea

Schmidt Robert Eduard

Ursoiu Ioana Cristina

Cuprins

[Cerința 1 2](#_Toc138180778)

[Procedură 2](#_Toc138180779)

[Funcția de masă de probabilitate (FMP) 2](#_Toc138180780)

[Funcția de densitate de probabilitate (FDP) 2](#_Toc138180781)

[Cum verificăm? 2](#_Toc138180782)

[Implementare 3](#_Toc138180783)

[Cerința 2 3](#_Toc138180784)

[Procedură 3](#_Toc138180785)

[Implementare 4](#_Toc138180786)

[Cerința 3 4](#_Toc138180787)

[Procedură 4](#_Toc138180788)

[Implementare 5](#_Toc138180789)

[Cerința 4 5](#_Toc138180790)

[Procedură 5](#_Toc138180791)

[Implementare 6](#_Toc138180792)

[Cerința 5 7](#_Toc138180793)

[Procedură 7](#_Toc138180794)

[Implementare 7](#_Toc138180795)

[Cerința 8 9](#_Toc138180796)

[Procedură 10](#_Toc138180797)

[Implementare 10](#_Toc138180798)

[Cerința 9 11](#_Toc138180799)

[9.1 11](#_Toc138180800)

[9.2 11](#_Toc138180801)

[9.3 12](#_Toc138180802)

[9.4 13](#_Toc138180803)

[Cerința 10 13](#_Toc138180804)

[Bibliografie 15](#_Toc138180805)

# Cerința 1

## Procedură

Pentru a verifica dacă o funcție introdusă de utilizator este o funcție de masă/densitate de probabilitate, trebuie să înțelegem diferența dintre cele două concepte.

### Funcția de masă de probabilitate (FMP)

Această funcție este asociată cu variabile aleatoare **discrete**. Funcția de masă de probabilitate atribuie probabilități fiecărui posibil rezultat distinct al variabilei aleatoare. Deci, pentru fiecare valoare posibilă a variabilei, funcția de masă de probabilitate indică probabilitatea asociată cu acea valoare. Suma tuturor probabilităților atribuite valorilor posibile trebuie să fie egală cu 1.

Un exemplu bine cunoscut de funcție de masă de probabilitate este aruncarea unui zar. Rezultatul aruncării poate avea valori aparținând unei mulțimi finite (1, 2, 3, 4, 5, 6), deci reprezintă o variabilă aleatoare discretă. Suma probabilităților va da mereu 1.

### Funcția de densitate de probabilitate (FDP)

Această funcție este asociată cu variabile aleatoare **continue**. Funcția de densitate de probabilitate reprezintă distribuția de probabilitate a variabilei aleatoare continue. În contrast cu funcția de masă de probabilitate, care atribuie probabilități discrete, funcția de densitate de probabilitate indică probabilitatea că variabila aleatoare ia o valoare într-un anumit interval. Aria sub curba funcției de densitate de probabilitate într-un anumit interval reprezintă probabilitatea ca variabila să fie în acel interval. În cazul funcțiilor de densitate de probabilitate, nu există necesitatea ca suma probabilităților să fie 1.

Un exemplu poate fi înălțimea unei populații, ce poate avea teoretic o infinitate de valori, astfel reprezentând o variabilă aleatoare continuă.

### Cum verificăm?

Pentru a verifica dacă o funcție este o funcție de masă de probabilitate sau o funcție de densitate de probabilitate, puteți utiliza următorii pași:

* Verificați dacă funcția are **valori non-negative** pentru toate posibilele valori ale variabilei. O funcție de masă/densitate de probabilitate nu poate avea valori negative.
* Verificați dacă **suma tuturor valorilor** funcției este egală cu 1 (pentru **FMP**) sau **integrala funcției** este egală cu 1 (pentru **FDP**). Acest lucru verifică condiția de normalizare a probabilităților.

Dacă funcția îndeplinește ambele condiții de mai sus, atunci este o funcție de masă/densitate de probabilitate corespunzătoare.

## Implementare

În primul rând verificăm dacă funcția are valori non-negative.

has\_nonnegative\_values <- all(user\_function(1:10) >= 0)

Pentru verificare calculăm suma (pentru FMP) sau integrala (pentru FDP).

sum <- sum(user\_function(1:10))

integral <- integrate(user\_function, lower = -Inf, upper = Inf)$value

Verificăm dacă suma sau integrala este aproape de 1, utilizând o toleranță de 0.0001.

is\_mass\_function <- abs(sum - 1) < 0.0001

is\_density\_function <- abs(integral - 1) < 0.0001

Returnăm rezultatul calculului.

if (has\_nonnegative\_values && is\_mass\_function) {

return(1)

} else if (has\_nonnegative\_values && is\_density\_function) {

return(2)

} else {

return(0)

}

Afișăm rezultatele printr-o funcție.

print\_density\_or\_mass <- function(user\_function) {

result <- density\_or\_mass(user\_function)

if (result == 1) {

print("Funcția introdusă de utilizator este o funcție de masă de probabilitate.")

} else if (result == 2) {

print("Funcția introdusă de utilizator este o funcție de densitate de probabilitate.")

} else {

print("Funcția introdusă de utilizator nu este o funcție de masă de probabilitate și nici o funcție de densitate de probabilitate.")

}

}

# Cerința 2

## Procedură

O constantă de normalizare pentru o funcție este un număr care se înmulțește cu funcția pentru a obține o distribuție de probabilitate. O distribuție de probabilitate este o funcție care descrie probabilitatea ca o variabilă aleatoare să ia anumite valori. Pentru a determina o constantă de normalizare, trebuie să folosim proprietatea că suma sau integrala unei distribuții de probabilitate este egală cu 1. Astfel, dacă f este funcția dată și c este constanta de normalizare, avem:

## Implementare

Calculăm constanta de normalizare k.

normalization\_constant <- integrate(user\_function, lower = -Inf, upper = Inf)$value

Verificăm dacă constanta de normalizare există și determinăm dacă funcția este o funcție de masă sau de densitate de probabilitate. Afișăm rezultatele direct.

tolerance <- 0.0001

if (is.finite(normalization\_constant)) {

if ((1 / normalization\_constant - 1) < tolerance) {

print("Funcția introdusă de utilizator este o functie de densitate de probabilitate.")

} else {

print("Funcția introdusă de utilizator este o funcție de masă de probabilitate.")

}

print(paste("Constanta de normalizare k =", 1 / normalization\_constant))

} else {

print("Nu există o constantă de normalizare pentru funcția introdusă de utilizator.")

}

# Cerința 3

## Procedură

Pentru a rezolva această problemă, trebuie:

1. Să identificăm repartiția de probabilitate a variabilei aleatoare și să stabilim parametrii ei.
2. Să calculăm densitatea de probabilitate sau funcția de masă pentru fiecare valoare posibilă a variabilei aleatoare, folosind formula corespunzătoare repartiției.
3. Să reprezentăm grafic densitatea de probabilitate sau funcția de masă pe un sistem de coordonate, având pe axa orizontală valorile variabilei aleatoare și pe axa verticală probabilitățile asociate.
4. Să calculăm funcția de repartiție pentru fiecare valoare posibilă a variabilei aleatoare, folosind formula corespunzătoare repartiției sau integrând densitatea de probabilitate.
5. Să reprezentăm grafic funcția de repartiție pe un sistem de coordonate, având pe axa orizontală valorile variabilei aleatoare și pe axa verticală probabilitățile cumulate asociate.

## Implementare

Definim intervalul de valori pentru x.

x <- seq(-5, 5, length.out = 100)

Calculăm valorile densității de probabilitate/funcției de masă pentru x.

density <- user\_function(x)

Calculăm valorile funcției de repartiție pentru x.

cumulative <- cumsum(density)

Folosim plot pentru a crea graficul pentru densitatea de probabilitate/funcția de masă și pentru funcția de repartiție.

jpeg("../../density\_plot.jpg", width = 350, height = 350)

plot(x, density,

type = "l", xlab = "x", ylab = "Probabilitate",

main = "Densitatea de probabilitate/Functia de masa"

)

dev.off()

jpeg("../../repartition\_plot.jpg", width = 350, height = 350)

plot(x, cumulative,

type = "l", xlab = "x", ylab = "Probabilitate",

main = "Functia de repartitie"

)

dev.off()

# Cerința 4

## Procedură

Media este media aritmetică a unui set de date și se obține adunând toate valorile și împărțind suma la numărul de valori. Dispersia măsoară gradul de variație sau imprăștiere a valorilor în jurul mediei. Există mai mulți indicatori de dispersie, cum ar fi varianța, abaterea standard și amplitudinea interquartilică.

Momentul inițial și centrat este o măsură a formei distribuției unei variabile aleatoare. Momentul inițial de ordinul k este media aritmetică a puterilor k ale valorilor variabilei. Momentul centrat de ordinul k este media aritmetică a puterilor k ale diferențelor dintre valorile variabilei și media sa. Momentul inițial și centrat se folosește pentru a descrie caracteristicile unei distribuții, cum ar fi simetria, curtosisul și dispersia.

## Implementare

Definim intervalul de valori pentru x.

x <- seq(-10, 10, length.out = 1000)

Calculăm media.

mean\_value <- sum(x^1 \* user\_function(x))

Calculăm dispersia.

variance <- integrate(function(x) (x - mean\_value)^2 \* user\_function(x), lower = -Inf, upper = Inf)$value

Calculați momentele inițiale până la ordinul 4

moment\_1 <- integrate(function(x) (x - mean\_value)^1 \* user\_function(x), lower = -Inf, upper = Inf)$value

moment\_2 <- integrate(function(x) (x - mean\_value)^2 \* user\_function(x), lower = -Inf, upper = Inf)$value

moment\_3 <- integrate(function(x) (x - mean\_value)^3 \* user\_function(x), lower = -Inf, upper = Inf)$value

moment\_4 <- integrate(function(x) (x - mean\_value)^4 \* user\_function(x), lower = -Inf, upper = Inf)$value

Calculați momentele centrate până la ordinul 4

central\_moment\_1 <- moment\_1

central\_moment\_2 <- moment\_2

central\_moment\_3 <- moment\_3

central\_moment\_4 <- moment\_4 - 3 \* variance^2

Afișați rezultatele

if (!is.finite(moment\_1)) {

print("Momentul inițial de ordinul 1 nu există.")

} else {

print(paste("Momentul inițial de ordinul 1:", moment\_1))

}

if (!is.finite(moment\_2)) {

print("Momentul inițial de ordinul 2 nu există.")

} else {

print(paste("Momentul inițial de ordinul 2:", moment\_2))

}

if (!is.finite(moment\_3)) {

print("Momentul inițial de ordinul 3 nu există.")

} else {

print(paste("Momentul inițial de ordinul 3:", moment\_3))

}

if (!is.finite(moment\_4)) {

print("Momentul inițial de ordinul 4 nu există.")

} else {

print(paste("Momentul inițial de ordinul 4:", moment\_4))

}

if (!is.finite(central\_moment\_1)) {

print("Momentul centrat de ordinul 1 nu există.")

} else {

print(paste("Momentul centrat de ordinul 1:", central\_moment\_1))

}

if (!is.finite(central\_moment\_2)) {

print("Momentul centrat de ordinul 2 nu există.")

} else {

print(paste("Momentul centrat de ordinul 2:", central\_moment\_2))

}

if (!is.finite(central\_moment\_3)) {

print("Momentul centrat de ordinul 3 nu există.")

} else {

print(paste("Momentul centrat de ordinul 3:", central\_moment\_3))

}

if (!is.finite(central\_moment\_4)) {

print("Momentul centrat de ordinul 4 nu există.")

} else {

print(paste("Momentul centrat de ordinul 4:", central\_moment\_4))

}

# Cerința 5

## Procedură

Media unei variabile aleatoare g(X) se calculează folosind formula:

Dispersia unei variabile aleatoare g(X) se calculează folosind formula:

## Implementare

Definim funcția g(x) specificată de utilizator (în exemplu folosim x2).

g <- function(x) {

return(x^2)

}

Definim funcția de densitate de probabilitate a lui X.

densitate\_x <- function(x) {

return(dnorm(x, mean = 0, sd = 1))

}

În funcția principală prelucrăm datele.

Mai întâi calculăm media variabilei aleatoare g(X).

mean\_gx <- integrate(function(x) g(x) \* densitate\_x(x), lower = -Inf, upper = Inf)$value

Apoi calculăm dispersia.

var\_gx <- integrate(function(x) (g(x) - mean\_gx)^2 \* densitate\_x(x), lower = -Inf, upper = Inf)$value

Afișăm rezultatele.

print(paste("Media variabilei aleatoare g(X):", mean\_gx))

print(paste("Dispersia variabilei aleatoare g(X):", var\_gx))

# Cerința 6

## Procedură

Prin funcția pe care o vom impementa vom calcula probabilitățile marginale și condiționate atât pentru variabilele aleatoare discrete cât și pentru variabilele aleatoare continue.

## Implementare

Declarăm o funcție pentru calculul probabilității marginale sau condiționate.

P <- function(X, condition = NULL) {

if (is.null(condition)) {

# Calculul probabilității marginale

return(sum(X) / length(X))

} else {

# Calculul probabilității condiționate

return(sum(X[condition]) / sum(condition))

}

}

Calculul probabilității marginale pentru o variabilă aleatoare discretă:

X <- rbinom(1000, size = 10, prob = 0.5)

prob\_marginal <- P(X)

print(prob\_marginal)

Calculul probabilității marginale pentru o variabilă aleatoare continuă:

X <- rnorm(1000, mean = 0, sd = 1)

prob\_marginal <- P(X)

print(prob\_marginal)

Calculul probabilității condiționate pentru o variabilă aleatoare discretă:

X <- rpois(1000, lambda = 3)

prob\_conditioned <- P(X, condition = X >= 2)

print(prob\_conditioned)

Calculul probabilității condiționate pentru o variabilă aleatoare continuă:

X <- rexp(1000, rate = 0.5)

prob\_conditioned <- P(X, condition = X > 2)

print(prob\_conditioned)

# Cerința 7

## Procedură

Pentru a calcula covarianța și coeficientul de corelație, trebuie să cunoaștem funcția de masă sau densitatea comună a celor două variabile aleatoare. Aceasta ne dă probabilitatea ca X și Y să ia anumite valori simultan. De exemplu, dacă X și Y sunt variabile discrete, atunci funcția lor de masă comună este:

Dacă X și Y sunt variabile continue, atunci funcția lor de densitate comună este:

unde F(x,y) este funcția lor de distribuție comună.

## Implementare

Mai întâi definim setul de valori pentru variabilele X și Y.

set\_valori\_X <- c(1, 2, 2, 2)

set\_valori\_Y <- c(1, 2, 1, 2)

Efectuăm calculul covarianței utilizând funcția de masă comună.

covarianta <- sum((set\_valori\_X - mean(set\_valori\_X)) \* (set\_valori\_Y - mean(set\_valori\_Y)) \* f\_pmf(set\_valori\_X, set\_valori\_Y))

Calculăm deviația standard a variabilei X și Y.

dev\_std\_X <- sqrt(sum((set\_valori\_X - mean(set\_valori\_X))^2 \* f\_pmf(set\_valori\_X, set\_valori\_Y)))

dev\_std\_Y <- sqrt(sum((set\_valori\_Y - mean(set\_valori\_Y))^2 \* f\_pmf(set\_valori\_X, set\_valori\_Y)))

Calculăm coeficientul de corelație.

coef\_corel <- covarianta / (dev\_std\_X \* dev\_std\_Y)

Returnăm rezultatele.

return(list(covarianta = covarianta, coeficient\_corelatie = coef\_corel))

# Cerința 8

## Procedură

Pentru a rezolva această problemă trebuie să urmăm câțiva pași:

1. Identificăm funcția de masă/densitatea comună a celor două variabile aleatoare, notată cu f(x,y).

2. Calculăm funcțiile de masă/densitățile marginale ale fiecărei variabile aleatoare, notate cu f\_X(x) și f\_Y(y), prin integrarea sau sumarea funcției comune pe domeniul celeilalte variabile.

3. Calculăm funcțiile de masă/densitățile condiționate ale unei variabile aleatoare în funcție de cealaltă, notate cu f\_{X|Y}(x|y) și f\_{Y|X}(y|x), prin împărțirea funcției comune la funcțiile marginale corespunzătoare.

## Implementare

Definim funcția de masă comună (pentru variabile discrete). Începem prin definirea tabelei de probabilități conjuncte.

probabilities <- matrix(c(0.1, 0.2, 0.15, 0.25, 0.05, 0.1), nrow = 2, byrow = TRUE)

Alegem valori unice pentru variabilele X și Y.

x\_values <- unique(x)

y\_values <- unique(y)

Creăm un data frame pentru asocierea valorilor variabilelor X și Y cu probabilitățile corespunzătoare.

data <- expand.grid(X = x\_values, Y = y\_values)

data$Prob <- as.vector(probabilities)

data$Prob[match(data$X, x) & match(data$Y, y)]

Definirea valorilor aleatoare pentru variabilele X și Y

x\_values <- c(1, 2, 3) # Valorile posibile ale variabilei X

y\_values <- c(1, 2, 3) # Valorile posibile ale variabilei Y

Funcțiile de masă marginale pentru X și Y:

marginal\_X <- function(x) {

joint\_probs <- joint(x\_values, y\_values)

marginal\_probs <- rowSums(joint\_probs)

marginal\_probs[match(x, x\_values)]

}

marginal\_Y <- function(y) {

joint\_probs <- joint(x\_values, y\_values)

marginal\_probs <- colSums(joint\_probs)

marginal\_probs[match(y, y\_values)]

}

Funcțiile de masă condiționale pentru X și Y:

conditional\_X\_given\_Y <- function(x, y) {

joint\_probs <- joint(x\_values, y\_values)

marginal\_probs\_Y <- marginal\_Y[y]

conditional\_probs <- joint\_probs[, y] / marginal\_probs\_Y

conditional\_probs[match(x, x\_values)]

}

conditional\_Y\_given\_X <- function(y, x) {

joint\_probs <- joint(x\_values, y\_values)

marginal\_probs\_X <- marginal\_X[x]

conditional\_probs <- joint\_probs[x, ] / marginal\_probs\_X

conditional\_probs[match(y, y\_values)]

}

# Cerința 9

## 9.1

Calculăm mediana, media și deviația standard.

mediana <- median(set\_valori)

media <- mean(set\_valori)

deviatia\_standard <- sd(set\_valori)

Afișăm valorile calculate.

cat("Mediana:", mediana, "\n")

cat("Media:", media, "\n")

cat("Deviația standard:", deviatia\_standard, "\n")

Afișăm histograma și adăugăm linii verticale pentru mediană, medie și deviație standard.

hist(set\_valori, main = "Histograma", xlab = "Valori", ylab = "Frecventa", col = "lightblue", breaks = "FD")

abline(v = mediana, col = "red", lwd = 2)

abline(v = media, col = "blue", lwd = 2)

abline(v = c(media - deviatia\_standard, media + deviatia\_standard), col = "green", lwd = 2, lty = 2)

Efectuăm operațiile pe fiecare set.

afisare\_histograma\_si\_statistici(set\_a)

afisare\_histograma\_si\_statistici(set\_b)

afisare\_histograma\_si\_statistici(set\_c)

afisare\_histograma\_si\_statistici(set\_d)

afisare\_histograma\_si\_statistici(set\_e)

## 9.2

1. Ajustarea distribuției normale

fit\_norm <- MASS::fitdistr(set[set > 0], "normal")

AIC\_norm <- AIC(fit\_norm)

BIC\_norm <- BIC(fit\_norm)

2. Ajustarea distribuției Poisson

fit\_pois <- MASS::fitdistr(set[set > 0], "Poisson")

AIC\_pois <- AIC(fit\_pois)

BIC\_pois <- BIC(fit\_pois)

3. Ajustarea distribuției exponențiale

fit\_exp <- MASS::fitdistr(set[set > 0], "exponential")

AIC\_exp <- AIC(fit\_exp)

BIC\_exp <- BIC(fit\_exp)

4. Ajustarea distribuției gamma

fit\_gamma <- MASS::fitdistr(set[set > 0], "gamma")

AIC\_gamma <- AIC(fit\_gamma)

BIC\_gamma <- BIC(fit\_gamma)

Comparăm criteriile AIC și BIC.

criteria <- data.frame(

Distributie = c("Normala", "Poisson", "Exponentiala", "Gamma"),

AIC = c(AIC\_norm, AIC\_pois, AIC\_exp, AIC\_gamma),

BIC = c(BIC\_norm, BIC\_pois, BIC\_exp, BIC\_gamma)

)

criteria

Alegem distribuția cu cele mai mici criterii AIC și BIC.

distributie\_AIC <- criteria$Distributie[which.min(criteria$AIC)]

distributie\_BIC <- criteria$Distributie[which.min(criteria$BIC)]

print(paste("Distributia potrivita conform criteriului AIC este:", distributie\_AIC))

print(paste("Distributia potrivita conform criteriului BIC este:", distributie\_BIC))

## 9.3

1. Estimarea prin metoda verosimilității maxime

mle\_a <- MASS::fitdistr(set\_a, "normal")$estimate

mle\_b <- MASS::fitdistr(set\_b, "normal")$estimate

mle\_c <- MASS::fitdistr(set\_c, "normal")$estimate

mle\_d <- MASS::fitdistr(set\_d, "normal")$estimate

mle\_e <- MASS::fitdistr(set\_e, "normal")$estimate

2. Estimarea prin metoda momentelor

moment\_a <- c(mean(set\_a), sd(set\_a))

moment\_b <- c(mean(set\_b), sd(set\_b))

moment\_c <- c(mean(set\_c), sd(set\_c))

moment\_d <- c(mean(set\_d), sd(set\_d))

moment\_e <- c(mean(set\_e), sd(set\_e))

Comparăm rezultatele.

comparatie <- data.frame(

Set = c("a", "b", "c", "d", "e"),

Parametri\_MLE = c(mle\_a, mle\_b, mle\_c, mle\_d, mle\_e),

Parametri\_Moment = c(moment\_a, moment\_b, moment\_c, moment\_d, moment\_e)

)

comparatie

## 9.4

…

test\_a <- shapiro.test(set\_a)

test\_b <- shapiro.test(set\_b)

test\_c <- shapiro.test(set\_c)

test\_d <- shapiro.test(set\_d)

test\_e <- shapiro.test(set\_e)

rezultate <- data.frame(

Set = c("a", "b", "c", "d", "e"),

P\_Value = c(test\_a$p.value, test\_b$p.value, test\_c$p.value, test\_d$p.value, test\_e$p.value),

Normala = NA

)

rezultate$Normala[rezultate$P\_Value > 0.05] <- "Da"

rezultate$Normala[rezultate$P\_Value <= 0.05] <- "Nu"

# Cerința 10

Pentru ultima cerință am construit un meniu interactiv.

Citirea setului de valori:

set\_valori <- readline(prompt = "Introduceti setul de valori (separate prin virgula): ")

set\_valori <- as.numeric(strsplit(set\_valori, ",")[[1]])

Meniul interactiv:

while (TRUE) {

cat("\nAlegeți opțiunea dorită:")

cat("\n1. Histograma, mediana, media și deviația standard")

cat("\n2. Identificarea repartiției")

cat("\n3. Estimarea parametrilor în baza celor 5 eșantioane")

cat("\n4. Verificarea verosimilității extragerii dintr-o repartiție normală")

cat("\n0. Ieșire\n")

optiune <- readline(prompt = "Opțiune: ")

if (optiune == "0") {

break

}

if (optiune == "1") {

ex\_1(set\_valori)

} else if (optiune == "2") {

ex\_2(set\_valori)

} else if (optiune == "3") {

ex\_3(set\_valori)

} else if (optiune == "4") {

ex\_4(set\_valori)

} else {

cat("Opțiune invalidă! Vă rugăm să alegeți o opțiune validă.\n")

}

}

# Bibliografie

* [https://en.wikipedia.org/wiki/Probability\_distribution#Absolutely\_continuous\_probability\_distribution](https://en.wikipedia.org/wiki/Probability_distribution%23Absolutely_continuous_probability_distribution)