PROIECT

PROBABILITĂȚI ȘI STATISTICĂ

PARTEA I

Studenți:

Gheorghe Andrei-Bogdan

Marian Rareș-Ștefan

Tudorache Andrei-Theodor

Boț George

**Problema I**

set.seed(6)

# Generam 100 de intrări

n <- 100

# Generam datele pentru fiecare coloana

pret <- round(runif(n, 50, 300), 2)

gen\_muzical <- sample(c("Rock", "Pop", "Jazz", "Classical", "Hip-Hop", "Electronic"), n, replace = TRUE)

data <- sample(seq(as.Date('2023-01-01'), as.Date('2023-12-31'), by="day"), n, replace = TRUE)

oras <- sample(c("Bucuresti", "Cluj-Napoca", "Timisoara", "Iasi", "Brasov", "Constanta"), n, replace = TRUE)

artisti <- replicate(n, list(sample(c("Eminem", "Florin Salam", "Rihanna", "Drake", "Tyga", "Mihai Treistariu"), sample(1:3, 1), replace = TRUE)), simplify = FALSE)

soldout <- sample(c("da", "nu"), n, replace = TRUE)

# Construim dataframe-ul

c\_data <- data.frame(

  Pret = pret,

  Gen\_Muzical = gen\_muzical,

  Data = data,

  Oras = oras,

  Artisti = I(artisti),

  SoldOut = soldout,

  stringsAsFactors = FALSE

)

c\_data$SoldOut <- sample(c("da", "nu"), nrow(c\_data), replace = TRUE, prob = c(0.01, 0.99))

f1 <- function(buget, intervale) {

  # Convertim datele de start si de sfarsit ale intervalelor in format Date

  intervale <- lapply(intervale, function(x) {

    list(start = as.Date(x$start), end = as.Date(x$end))

  })

  # Filtram evenimentele care se incadreaza in intervalele de timp specificate

  evenimente\_filtrate <- c\_data[FALSE, ]

  for (interval in intervale) {

    evenimente\_filtrate <- rbind(evenimente\_filtrate, subset(c\_data, Data >= interval$start & Data <= interval$end))

  }

  # Functie recursiva pentru a gasi combinatiile de evenimente care se incadreaza in buget

  combina\_evenimente <- function(evenimente, buget) {

    if (nrow(evenimente) == 0) {

      return(list())

    }

    rezultate <- list()

    for (i in 1:nrow(evenimente)) {

      eveniment <- evenimente[i, ]

      pret <- eveniment$Pret

      if (pret <= buget) {

        ramase <- evenimente[-i, ]

        sub\_comb <- combina\_evenimente(ramase, buget - pret)

        if (length(sub\_comb) == 0) {

          rezultate <- append(rezultate, list(list(eveniment)))

        } else {

          for (comb in sub\_comb) {

            rezultate <- append(rezultate, list(c(list(eveniment), comb)))

          }

        }

      }

    }

    return(rezultate)

  }

  # Obtinem toate combinatiile de evenimente care se incadreaza in buget

  combinatii <- combina\_evenimente(evenimente\_filtrate, buget)

  # Convertim rezultatele in format usor de citit

  combinatii\_formate <- lapply(combinatii, function(comb) {

    do.call(rbind, comb)

  })

  return(combinatii\_formate)

}

f2 <- function(buget, intervale) {

  intervale <- lapply(intervale, function(x) {

    list(start = as.Date(x$start), end = as.Date(x$end))

  })

  # Filtram evenimentele care se incadreaza in intervalele de timp specificate și care nu sunt soldout

  evenimente\_filtrate <- c\_data[FALSE, ]

  for (interval in intervale) {

    evenimente\_filtrate <- rbind(evenimente\_filtrate, subset(c\_data, Data >= interval$start & Data <= interval$end & SoldOut == "nu"))

  }

  combina\_evenimente <- function(evenimente, buget) {

    if (nrow(evenimente) == 0) {

      return(list())

    }

    rezultate <- list()

    for (i in 1:nrow(evenimente)) {

      eveniment <- evenimente[i, ]

      pret <- eveniment$Pret

      if (pret <= buget) {

        ramase <- evenimente[-i, ]

        sub\_comb <- combina\_evenimente(ramase, buget - pret)

        if (length(sub\_comb) == 0) {

          rezultate <- append(rezultate, list(list(eveniment)))

        } else {

          for (comb in sub\_comb) {

            rezultate <- append(rezultate, list(c(list(eveniment), comb)))

          }

        }

      }

    }

    return(rezultate)

  }

  combinatii <- combina\_evenimente(evenimente\_filtrate, buget)

  combinatii\_formate <- lapply(combinatii, function(comb) {

    do.call(rbind, comb)

  })

  return(combinatii\_formate)

}

intervale <- list(

  list(start = '2023-06-01', end = '2023-06-30'),

  list(start = '2023-09-01', end = '2023-09-30')

)

buget <- 200

combinatii\_evenimente <- f1(buget, intervale)

print(combinatii\_evenimente)

combinatii\_soldout <- f2(buget, intervale)

print(combinatii\_soldout)

**Dataframe-ul generat (câteva rânduri)**

Pret Gen\_Muzical Data Oras Artisti SoldOut

1 201.57 Pop 2023-04-12 Constanta c("Flori.... nu

2 284.41 Jazz 2023-07-24 Timisoara Rihanna nu

3 116.09 Electronic 2023-10-17 Cluj-Napoca Eminem nu

4 145.02 Jazz 2023-09-26 Cluj-Napoca Drake nu

5 251.87 Hip-Hop 2023-01-15 Timisoara c("Rihan.... nu

6 294.52 Pop 2023-04-17 Iasi Rihanna nu

7 289.48 Hip-Hop 2023-03-14 Cluj-Napoca c("Flori.... nu

8 240.68 Classical 2023-06-04 Constanta Rihanna nu

9 177.41 Jazz 2023-06-26 Bucuresti c("Rihan.... nu

10 66.12 Hip-Hop 2023-05-03 Cluj-Napoca c("Flori.... nu

11 210.89 Pop 2023-03-17 Bucuresti c("Emine.... nu

12 278.98 Jazz 2023-09-22 Cluj-Napoca Rihanna nu

13 73.81 Jazz 2023-11-25 Brasov Florin Salam nu

14 123.84 Classical 2023-05-23 Iasi c("Emine.... nu

15 242.48 Electronic 2023-07-15 Iasi c("Drake.... nu

16 113.97 Classical 2023-05-07 Timisoara c("Flori.... nu

17 179.47 Hip-Hop 2023-04-28 Timisoara Mihai Tr.... nu

18 219.46 Pop 2023-01-26 Iasi c("Emine.... nu

19 86.81 Classical 2023-07-20 Iasi c("Tyga".... nu

20 225.13 Pop 2023-10-14 Timisoara c("Emine.... nu

**Prima filtrare**

[[1]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

9 177.41 Jazz 2023-06-26 Bucuresti c("Rihan.... nu

[[2]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

26 128.95 Jazz 2023-06-24 Cluj-Napoca c("Emine.... nu

43 70.01 Hip-Hop 2023-09-01 Cluj-Napoca c("Mihai.... nu

[[3]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

97 176.42 Hip-Hop 2023-06-05 Cluj-Napoca c("Emine.... nu

[[4]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

4 145.02 Jazz 2023-09-26 Cluj-Napoca Drake nu

[[5]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

43 70.01 Hip-Hop 2023-09-01 Cluj-Napoca c("Mihai.... nu

26 128.95 Jazz 2023-06-24 Cluj-Napoca c("Emine.... nu

[[6]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

58 160.77 Hip-Hop 2023-09-15 Brasov Drake nu

[[7]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

88 190.07 Rock 2023-09-26 Constanta c("Drake.... nu

**A doua filtrare**

[[1]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

9 177.41 Jazz 2023-06-26 Bucuresti c("Rihan.... nu

[[2]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

26 128.95 Jazz 2023-06-24 Cluj-Napoca c("Emine.... nu

43 70.01 Hip-Hop 2023-09-01 Cluj-Napoca c("Mihai.... nu

[[3]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

97 176.42 Hip-Hop 2023-06-05 Cluj-Napoca c("Emine.... nu

[[4]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

4 145.02 Jazz 2023-09-26 Cluj-Napoca Drake nu

[[5]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

43 70.01 Hip-Hop 2023-09-01 Cluj-Napoca c("Mihai.... nu

26 128.95 Jazz 2023-06-24 Cluj-Napoca c("Emine.... nu

[[6]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

58 160.77 Hip-Hop 2023-09-15 Brasov Drake nu

[[7]]

Pret Gen\_Muzical Data Oras Artisti SoldOut

88 190.07 Rock 2023-09-26 Constanta c("Drake.... nu

**Comentarii:**

La această problemă datele sunt generate automat pe baza unor sample-uri.

Pentru a face ca dataframe-ul să fie mai ușor de înțeles și vizualizat am folosit variabile intermediare, precum: pret -> Pret, gen\_muzical -> Gen\_muzical, etc.

Am folosit această linie de cod pentru a genera valoarea “da” în cadrul coloanei soldout cu o șansă de 1%:

c\_data$SoldOut <- sample(c("da", "nu"), nrow(c\_data), replace = TRUE, prob = c(0.01, 0.99))

Pentru a genera toate soluțiile posibile pentru subpunctele B și E am folosit un algoritm de combinări.

**Dificultăți:**

În cadrul subpunctului C am avut o problemă la implementarea restricției ca valoarea “da” din cadrul coloanei soldout să nu poată fi schimbată de o altă rulare a funcției, respectiv suplimentarea de bilete din cadrul subpunctului D.

**Problema II**

#Setam seed-ul pentru reproducibilitate

set.seed(6)

#Datele problemei

zile\_luna <- 30  # Numarul de zile din iunie 2024

medie <- 512  # Media calatorilor pe zi

min\_calatori <- 210  # Numarul minim de calatori pe zi

max\_calatori <- 983  # Numarul maxim de calatori pe zi

procent\_lejere <- 0.20  # Procentul zilelor lejere

procent\_normale <- 0.50  # Procentul zilelor normale

procent\_aglomerate <- 0.30  # Procentul zilelor aglomerate

#Generam numarul de zile pentru fiecare categorie

numar\_zile\_lejere <- round(zile\_luna \* procent\_lejere)

numar\_zile\_normale <- round(zile\_luna \* procent\_normale)

numar\_zile\_aglomerate <- zile\_luna - numar\_zile\_lejere - numar\_zile\_normale

#Generam numarul de calatori pentru fiecare tip de zi

calatori\_lejere <- runif(numar\_zile\_lejere, min = min\_calatori, max = 350)

calatori\_normale <- runif(numar\_zile\_normale, min = 351, max = 670)

calatori\_aglomerate <- runif(numar\_zile\_aglomerate, min = 671, max = max\_calatori)

#Combinam toate valorile intr-un singur vector

calatori\_zi <- c(calatori\_lejere, calatori\_normale, calatori\_aglomerate)

#Amestecam vectorul pentru a simula ordinea aleatoare a zilelor

calatori\_zi <- sample(calatori\_zi)

#Construim histograma valorilor

hist(calatori\_zi, breaks = 10, main = "Histograma numar calatori pe zi în iunie 2024",

     xlab = "Numar calatori pe zi", ylab = "Frecventa", col = "magenta")

#B

set.seed(6)

# Datele problemei

zile\_luna <- c(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)  # Numarul de zile pentru fiecare luna în 2024

medie <- 512

min\_calatori <- 210

max\_calatori <- 983

procent\_lejere <- 0.20

procent\_normale <- 0.50

procent\_aglomerate <- 0.30

# Functie pentru generarea calatorilor pentru o luna

genereaza\_calatori <- function(zile) {

  numar\_zile\_lejere <- round(zile \* procent\_lejere)

  numar\_zile\_normale <- round(zile \* procent\_normale)

  numar\_zile\_aglomerate <- zile - numar\_zile\_lejere - numar\_zile\_normale

  calatori\_lejere <- runif(numar\_zile\_lejere, min = min\_calatori, max = 350)

  calatori\_normale <- runif(numar\_zile\_normale, min = 351, max = 670)

  calatori\_aglomerate <- runif(numar\_zile\_aglomerate, min = 671, max = max\_calatori)

  calatori\_zi <- c(calatori\_lejere, calatori\_normale, calatori\_aglomerate)

  calatori\_zi <- sample(calatori\_zi)

  return(calatori\_zi)

}

# Dataframe pentru centralizarea rezultatelor

rezultate <- data.frame(

  luna = character(),

  medie\_calatori = numeric(),

  minim\_calatori = numeric(),

  maxim\_calatori = numeric(),

  procent\_lejere = numeric(),

  procent\_normale = numeric(),

  procent\_aglomerate = numeric(),

  stringsAsFactors = FALSE

)

# Generam datele pentru fiecare luna si calculam statisticile

for (i in 1:12) {

  calatori\_zi <- genereaza\_calatori(zile\_luna[i])

  medie\_calatori <- mean(calatori\_zi)

  minim\_calatori <- min(calatori\_zi)

  maxim\_calatori <- max(calatori\_zi)

  procent\_lejere <- sum(calatori\_zi < 350) / zile\_luna[i]

  procent\_normale <- sum(calatori\_zi >= 351 & calatori\_zi <= 670) / zile\_luna[i]

  procent\_aglomerate <- sum(calatori\_zi > 671) / zile\_luna[i]

  rezultate <- rbind(rezultate, data.frame(

    luna = month.name[i],

    medie\_calatori = medie\_calatori,

    minim\_calatori = minim\_calatori,

    maxim\_calatori = maxim\_calatori,

    procent\_lejere = procent\_lejere,

    procent\_normale = procent\_normale,

    procent\_aglomerate = procent\_aglomerate

  ))

}

# Afisam rezultatele

print(rezultate)

#C)

library(ggplot2)

set.seed(6)

# Datele problemei

zile\_luna <- c(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)

medie <- 512

min\_calatori <- 210

max\_calatori <- 983

procent\_lejere <- 0.20

procent\_normale <- 0.50

procent\_aglomerate <- 0.30

procent\_abonamente <- 0.38

procent\_platitori\_bilet <- 0.74  # Procentul pasagerilor care platesc bilet

pret\_bilet <- 3  # Pretul unui bilet

pret\_abonament <- 70  # Pretul unui abonament

# Functie pentru generarea calatorilor pentru o luna

genereaza\_calatori <- function(zile) {

  numar\_zile\_lejere <- round(zile \* procent\_lejere)

  numar\_zile\_normale <- round(zile \* procent\_normale)

  numar\_zile\_aglomerate <- zile - numar\_zile\_lejere - numar\_zile\_normale

  calatori\_lejere <- runif(numar\_zile\_lejere, min = min\_calatori, max = 350)

  calatori\_normale <- runif(numar\_zile\_normale, min = 351, max = 670)

  calatori\_aglomerate <- runif(numar\_zile\_aglomerate, min = 671, max = max\_calatori)

  calatori\_zi <- c(calatori\_lejere, calatori\_normale, calatori\_aglomerate)

  calatori\_zi <- sample(calatori\_zi)

  return(calatori\_zi)

}

# Dataframe pentru centralizarea rezultatelor

rezultate <- data.frame(

  luna = character(),

  medie\_calatori = numeric(),

  minim\_calatori = numeric(),

  maxim\_calatori = numeric(),

  procent\_lejere = numeric(),

  procent\_normale = numeric(),

  procent\_aglomerate = numeric(),

  pasageri\_abonament = numeric(),

  pasageri\_platitori\_bilet = numeric(),

  pasageri\_neplatitori\_bilet = numeric(),

  venituri\_bilete = numeric(),

  venituri\_abonamente = numeric(),

  venituri\_nerealizate = numeric(),

  stringsAsFactors = FALSE

)

# Generam datele pentru fiecare luna si calculam statisticile

for (i in 1:12) {

  calatori\_zi <- genereaza\_calatori(zile\_luna[i])

  medie\_calatori <- mean(calatori\_zi)

  minim\_calatori <- min(calatori\_zi)

  maxim\_calatori <- max(calatori\_zi)

  procent\_lejere <- sum(calatori\_zi < 350) / zile\_luna[i]

  procent\_normale <- sum(calatori\_zi >= 351 & calatori\_zi <= 670) / zile\_luna[i]

  procent\_aglomerate <- sum(calatori\_zi > 671) / zile\_luna[i]

  pasageri\_abonament <- sum(calatori\_zi) \* procent\_abonamente

  pasageri\_fara\_abonament <- sum(calatori\_zi) - pasageri\_abonament

  pasageri\_platitori\_bilet <- pasageri\_fara\_abonament \* procent\_platitori\_bilet

  pasageri\_neplatitori\_bilet <- pasageri\_fara\_abonament \* (1 - procent\_platitori\_bilet)

  venituri\_bilete <- pasageri\_platitori\_bilet \* pret\_bilet

  venituri\_abonamente <- (pasageri\_abonament / zile\_luna[i]) \* pret\_abonament

  venituri\_nerealizate <- pasageri\_neplatitori\_bilet \* pret\_bilet

  rezultate <- rbind(rezultate, data.frame(

    luna = month.name[i],

    medie\_calatori = medie\_calatori,

    minim\_calatori = minim\_calatori,

    maxim\_calatori = maxim\_calatori,

    procent\_lejere = procent\_lejere,

    procent\_normale = procent\_normale,

    procent\_aglomerate = procent\_aglomerate,

    pasageri\_abonament = pasageri\_abonament,

    pasageri\_platitori\_bilet = pasageri\_platitori\_bilet,

    pasageri\_neplatitori\_bilet = pasageri\_neplatitori\_bilet,

    venituri\_bilete = venituri\_bilete,

    venituri\_abonamente = venituri\_abonamente,

    venituri\_nerealizate = venituri\_nerealizate

  ))

}

# Afisam rezultatele

print(rezultate)

# Venituri pe luna

ggplot(rezultate, aes(x = luna)) +

  geom\_bar(aes(y = venituri\_bilete, fill = "Venituri bilete"), stat = "identity", position = "dodge") +

  geom\_bar(aes(y = venituri\_abonamente, fill = "Venituri abonamente"), stat = "identity", position = "dodge") +

  geom\_bar(aes(y = venituri\_nerealizate, fill = "Venituri nerealizate"), stat = "identity", position = "dodge") +

  labs(title = "Venituri pe luna în 2024", y = "Venituri (lei)", fill = "Tip venituri") +

  theme(axis.text.x = element\_text(angle = 45, hjust = 1))

#D)

set.seed(6)

# Datele problemei

zile\_luna <- c(31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)

medie <- 512

min\_calatori <- 210

max\_calatori <- 983

procent\_lejere <- 0.20

procent\_normale <- 0.50

procent\_aglomerate <- 0.30

procent\_abonamente <- 0.38

procent\_platitori\_bilet <- 0.74

pret\_bilet <- 3

pret\_abonament <- 70

cost\_controlor <- 214  # Costul pentru fiecare verificare a controlorului

amenda <- 50  # Valoarea amenzii

# Functie pentru generarea calatorilor pentru o luna

genereaza\_calatori <- function(zile) {

  numar\_zile\_lejere <- round(zile \* procent\_lejere)

  numar\_zile\_normale <- round(zile \* procent\_normale)

  numar\_zile\_aglomerate <- zile - numar\_zile\_lejere - numar\_zile\_normale

  calatori\_lejere <- runif(numar\_zile\_lejere, min = min\_calatori, max = 350)

  calatori\_normale <- runif(numar\_zile\_normale, min = 351, max = 670)

  calatori\_aglomerate <- runif(numar\_zile\_aglomerate, min = 671, max = max\_calatori)

  calatori\_zi <- c(calatori\_lejere, calatori\_normale, calatori\_aglomerate)

  calatori\_zi <- sample(calatori\_zi)

  return(calatori\_zi)

}

# Functii pentru verificarea biletelor

verifica\_bilete\_lejere <- function(num\_calatori) {

  num\_verificati <- sample(2:11, 1)

  num\_amendati <- 0

  for (i in 1:num\_verificati) {

    if (runif(1) > procent\_platitori\_bilet) {

      num\_amendati <- num\_amendati + 1

      if (num\_amendati >= 3) break

    }

  }

  return(num\_amendati)

}

verifica\_bilete\_normale <- function(num\_calatori) {

  num\_verificati <- 0

  num\_amendati <- 0

  while (num\_amendati < 5 && num\_verificati < num\_calatori) {

    num\_verificati <- num\_verificati + 1

    if (runif(1) > procent\_platitori\_bilet) {

      num\_amendati <- num\_amendati + 1

    }

  }

  return(num\_amendati)

}

verifica\_bilete\_aglomerate <- function(num\_calatori) {

  num\_verificati <- sample(3:5, 1)

  num\_amendati <- 0

  for (i in 1:num\_verificati) {

    if (runif(1) > procent\_platitori\_bilet) {

      num\_amendati <- 1

      break

    }

  }

  return(num\_amendati)

}

# Dataframe pentru centralizarea rezultatelor

rezultate <- data.frame(

  luna = character(),

  medie\_calatori = numeric(),

  minim\_calatori = numeric(),

  maxim\_calatori = numeric(),

  procent\_lejere = numeric(),

  procent\_normale = numeric(),

  procent\_aglomerate = numeric(),

  pasageri\_abonament = numeric(),

  pasageri\_platitori\_bilet = numeric(),

  pasageri\_neplatitori\_bilet = numeric(),

  venituri\_bilete = numeric(),

  venituri\_abonamente = numeric(),

  venituri\_nerealizate = numeric(),

  amendati\_lejere = numeric(),

  amendati\_normale = numeric(),

  amendati\_aglomerate = numeric(),

  venituri\_amenzi = numeric(),

  cost\_controlori = numeric(),

  profit\_pierdere = numeric(),

  stringsAsFactors = FALSE

)

# Generam datele pentru fiecare luna si calculam statisticile

for (i in 1:12) {

  calatori\_zi <- genereaza\_calatori(zile\_luna[i])

  medie\_calatori <- mean(calatori\_zi)

  minim\_calatori <- min(calatori\_zi)

  maxim\_calatori <- max(calatori\_zi)

  procent\_lejere <- sum(calatori\_zi < 350) / zile\_luna[i]

  procent\_normale <- sum(calatori\_zi >= 351 & calatori\_zi <= 670) / zile\_luna[i]

  procent\_aglomerate <- sum(calatori\_zi > 671) / zile\_luna[i]

  pasageri\_abonament <- sum(calatori\_zi) \* procent\_abonamente

  pasageri\_fara\_abonament <- sum(calatori\_zi) - pasageri\_abonament

  pasageri\_platitori\_bilet <- pasageri\_fara\_abonament \* procent\_platitori\_bilet

  pasageri\_neplatitori\_bilet <- pasageri\_fara\_abonament \* (1 - procent\_platitori\_bilet)

  venituri\_bilete <- pasageri\_platitori\_bilet \* pret\_bilet

  venituri\_abonamente <- (pasageri\_abonament / zile\_luna[i]) \* pret\_abonament

  venituri\_nerealizate <- pasageri\_neplatitori\_bilet \* pret\_bilet

  amendati\_lejere <- sum(sapply(calatori\_zi[calatori\_zi < 350], verifica\_bilete\_lejere))

  amendati\_normale <- sum(sapply(calatori\_zi[calatori\_zi >= 351 & calatori\_zi <= 670], verifica\_bilete\_normale))

  amendati\_aglomerate <- sum(sapply(calatori\_zi[calatori\_zi > 671], verifica\_bilete\_aglomerate))

  venituri\_amenzi <- (amendati\_lejere + amendati\_normale + amendati\_aglomerate) \* amenda

  cost\_controlori <- zile\_luna[i] \* 2 \* cost\_controlor

  profit\_pierdere <- venituri\_amenzi - cost\_controlori - venituri\_nerealizate

  rezultate <- rbind(rezultate, data.frame(

    luna = month.name[i],

    medie\_calatori = medie\_calatori,

    minim\_calatori = minim\_calatori,

    maxim\_calatori = maxim\_calatori,

    procent\_lejere = procent\_lejere,

    procent\_normale = procent\_normale,

    procent\_aglomerate = procent\_aglomerate,

    pasageri\_abonament = pasageri\_abonament,

    pasageri\_platitori\_bilet = pasageri\_platitori\_bilet,

    pasageri\_neplatitori\_bilet = pasageri\_neplatitori\_bilet,

    venituri\_bilete = venituri\_bilete,

    venituri\_abonamente = venituri\_abonamente,

    venituri\_nerealizate = venituri\_nerealizate,

    amendati\_lejere = amendati\_lejere,

    amendati\_normale = amendati\_normale,

    amendati\_aglomerate = amendati\_aglomerate,

    venituri\_amenzi = venituri\_amenzi,

    cost\_controlori = cost\_controlori,

    profit\_pierdere = profit\_pierdere

  ))

}

# Afisam rezultatele

print(rezultate)

# Venituri pe luna

ggplot(rezultate, aes(x = luna)) +

  geom\_bar(aes(y = venituri\_bilete, fill = "Venituri bilete"), stat = "identity", position = "dodge") +

  geom\_bar(aes(y = venituri\_abonamente, fill = "Venituri abonamente"), stat = "identity", position = "dodge") +

  geom\_bar(aes(y = venituri\_amenzi, fill = "Venituri amenzi"), stat = "identity", position = "dodge") +

  geom\_bar(aes(y = -venituri\_nerealizate, fill = "Venituri nerealizate"), stat = "identity", position = "dodge") +

  labs(title = "Venituri pe luna în 2024", y = "Venituri", fill = "Tip venit") +

  theme(axis.text.x = element\_text(angle = 45, hjust = 1))

# Profit/Pierdere pe luna

ggplot(rezultate, aes(x = luna, y = profit\_pierdere, fill = profit\_pierdere > 0)) +

  geom\_bar(stat = "identity") +

  labs(title = "Profit/Pierdere pe luna în 2024", y = "Profit/Pierdere", fill = "Profit") +

  theme(axis.text.x = element\_text(angle = 45, hjust = 1))

#E)

# Functie pentru verificarea biletelor cu un al treilea control

verifica\_bilete\_suplimentar <- function(num\_calatori) {

  num\_verificati <- 0

  num\_amendati <- 0

  while (num\_amendati < 5 && num\_verificati < num\_calatori) {

    num\_verificati <- num\_verificati + 1

    if (runif(1) > procent\_platitori\_bilet) {

      num\_amendati <- num\_amendati + 1

    }

  }

  return(num\_amendati)

}

# Dataframe pentru rezultatele cu al treilea control

rezultate\_suplimentar <- data.frame(

  luna = character(),

  venituri\_amenzi\_suplimentar = numeric(),

  cost\_controlori\_suplimentar = numeric(),

  venituri\_nerealizate\_suplimentar = numeric(),

  profit\_pierdere\_suplimentar = numeric(),

  stringsAsFactors = FALSE

)

# Simulam efectele introducerii unui al treilea control pe zi

for (i in 1:12) {

  calatori\_zi <- genereaza\_calatori(zile\_luna[i])

  amendati\_lejere <- sum(sapply(calatori\_zi[calatori\_zi < 350], verifica\_bilete\_lejere))

  amendati\_normale <- sum(sapply(calatori\_zi[calatori\_zi >= 351 & calatori\_zi <= 670], verifica\_bilete\_normale))

  amendati\_aglomerate <- sum(sapply(calatori\_zi[calatori\_zi > 671], verifica\_bilete\_aglomerate))

  venituri\_amenzi\_suplimentar <- sum(sapply(calatori\_zi, verifica\_bilete\_suplimentar)) \* amenda \* 0.7  # 70% din amenzi sunt raportate oficial

  cost\_controlori\_suplimentar <- zile\_luna[i] \* 3 \* cost\_controlor  # Costul pentru trei controale pe zi

  venituri\_nerealizate\_suplimentar <- sum(calatori\_zi) \* (1 - procent\_platitori\_bilet) \* pret\_bilet  # Pierderile din neplata biletelor

  profit\_pierdere\_suplimentar <- venituri\_amenzi\_suplimentar - cost\_controlori\_suplimentar - venituri\_nerealizate\_suplimentar

  rezultate\_suplimentar <- rbind(rezultate\_suplimentar, data.frame(

    luna = month.name[i],

    venituri\_amenzi\_suplimentar = venituri\_amenzi\_suplimentar,

    cost\_controlori\_suplimentar = cost\_controlori\_suplimentar,

    venituri\_nerealizate\_suplimentar = venituri\_nerealizate\_suplimentar,

    profit\_pierdere\_suplimentar = profit\_pierdere\_suplimentar

  ))

}

# Afisam rezultatele

print(rezultate\_suplimentar)

# Vizualizam rezultatele

# Profit/Pierdere pe luna cu al treilea control

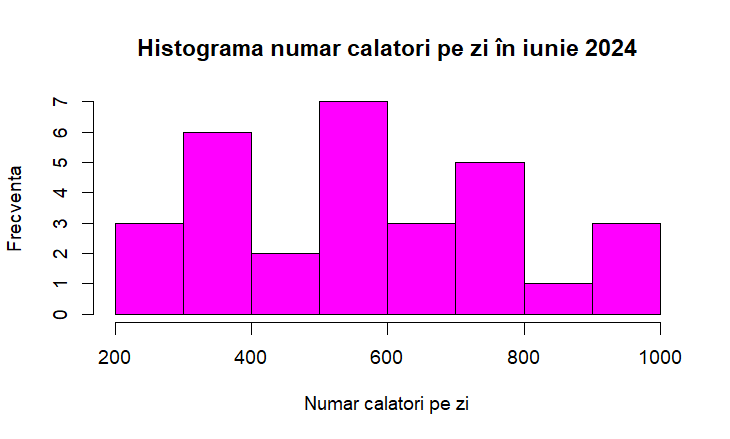
ggplot(rezultate\_suplimentar, aes(x = luna, y = profit\_pierdere\_suplimentar, fill = profit\_pierdere\_suplimentar > 0)) +

  geom\_bar(stat = "identity") +

  labs(title = "Profit/Pierdere pe luna cu al treilea control în 2024", y = "Profit/Pierdere", fill = "Profit") +

  theme(axis.text.x = element\_text(angle = 45, hjust = 1))

**Subpunctul a)**



**Subpunctul b)**

luna medie\_calatori minim\_calatori maxim\_calatori procent\_lejere procent\_normale procent\_aglomerate

1 January 567.6499 247.0093 956.2803 0.1935484 0.516129 0.2903226

2 February 558.1056 268.7240 972.3819 0.1785714 0.500000 0.3214286

3 March 559.3244 226.9785 981.5797 0.1935484 0.516129 0.2903226

4 April 563.8982 232.5297 980.4517 0.2000000 0.500000 0.3000000

5 May 551.8950 219.4690 931.4349 0.1935484 0.516129 0.2903226

6 June 579.2525 223.3581 908.1324 0.2000000 0.500000 0.3000000

7 July 537.8601 246.2058 941.2446 0.1935484 0.516129 0.2903226

8 August 544.4560 210.1705 949.1154 0.1935484 0.516129 0.2903226

9 September 562.4971 216.9924 945.4194 0.2000000 0.500000 0.3000000

10 October 544.6749 230.1924 946.4730 0.1935484 0.516129 0.2903226

11 November 561.4636 228.8248 978.5593 0.2000000 0.500000 0.3000000

12 December 554.3975 224.3281 974.1016 0.1935484 0.516129 0.2903226

**Subpunctul c)**

pasageri\_abonament pasageri\_platitori\_bilet pasageri\_neplatitori\_bilet venituri\_bilete venituri\_abonamente venituri\_nerealizate

1 6686.915 8073.570 2836.660 24220.71 15099.49 8509.980

2 5938.244 7169.648 2519.066 21508.95 14845.61 7557.197

3 6588.841 7955.159 2795.056 23865.48 14878.03 8385.168

4 6428.440 7761.495 2727.012 23284.49 14999.69 8181.035

5 6501.323 7849.493 2757.930 23548.48 14680.41 8273.789

6 6603.479 7972.832 2801.265 23918.50 15408.12 8403.796

7 6335.992 7649.877 2687.795 22949.63 14307.08 8063.384

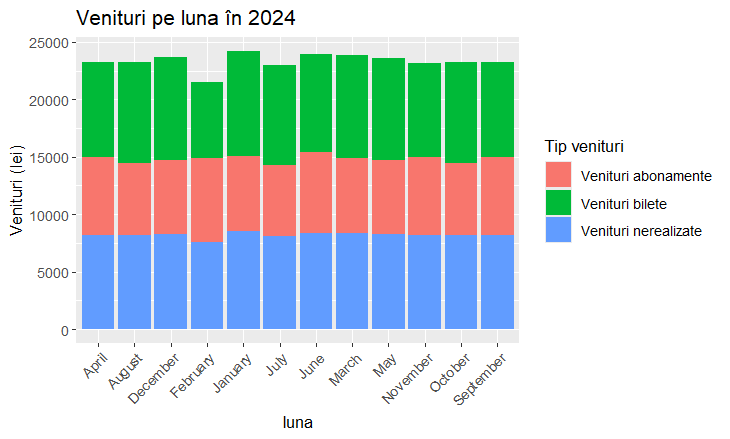
8 6413.692 7743.689 2720.756 23231.07 14482.53 8162.267

9 6412.467 7742.210 2720.236 23226.63 14962.42 8160.708

10 6416.271 7746.803 2721.850 23240.41 14488.35 8165.549

11 6400.685 7727.985 2715.238 23183.95 14934.93 8145.714

12 6530.802 7885.085 2770.435 23655.25 14746.97 8311.305



**Subpunctul d)**

venituri\_nerealizate amendati\_lejere amendati\_normale amendati\_aglomerate venituri\_amenzi cost\_controlori profit\_pierdere

1 8509.980 8 80 5 4650 13268 -17127.98

2 7749.526 8 75 6 4450 12412 -15711.53

3 8505.323 12 80 9 5050 13268 -16723.32

4 8099.360 8 75 5 4400 12840 -16539.36

5 8000.103 11 80 4 4750 13268 -16518.10

6 8271.032 12 75 7 4700 12840 -16411.03

7 8088.380 10 80 6 4800 13268 -16556.38

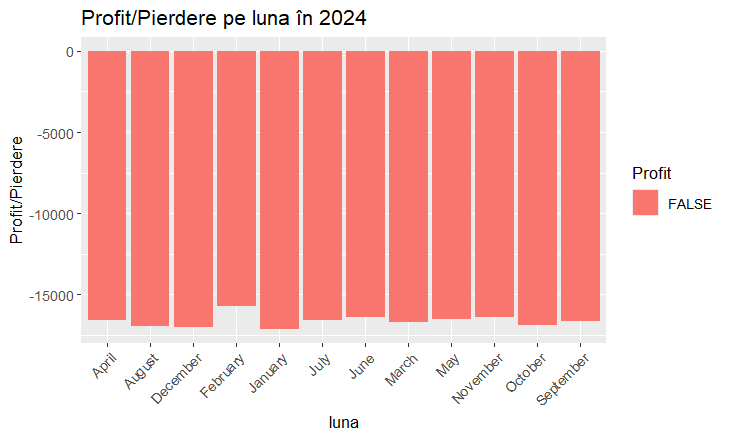
8 8536.268 13 80 4 4850 13268 -16954.27

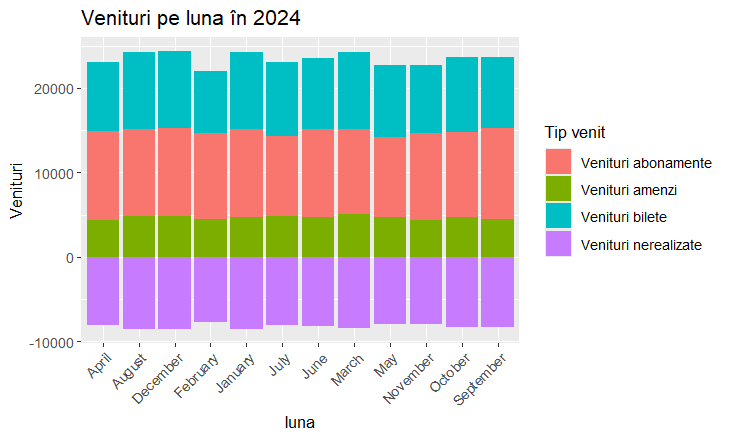
9 8307.638 9 75 6 4500 12840 -16647.64

10 8327.581 10 80 4 4700 13268 -16895.58

11 7973.497 7 75 6 4400 12840 -16413.50

12 8576.755 12 80 5 4850 13268 -16994.75





**Subpunctul e)**

luna venituri\_amenzi\_suplimentar cost\_controlori\_suplimentar venituri\_nerealizate\_suplimentar profit\_pierdere\_suplimentar

1 January 5425 19902 13117.85 -27594.85

2 February 5075 18618 12197.17 -25740.17

3 March 5425 19902 13934.31 -28411.31

4 April 5250 19260 13151.42 -27161.42

5 May 5425 19902 12932.20 -27409.20

6 June 5250 19260 13336.85 -27346.85

7 July 5425 19902 13220.74 -27697.74

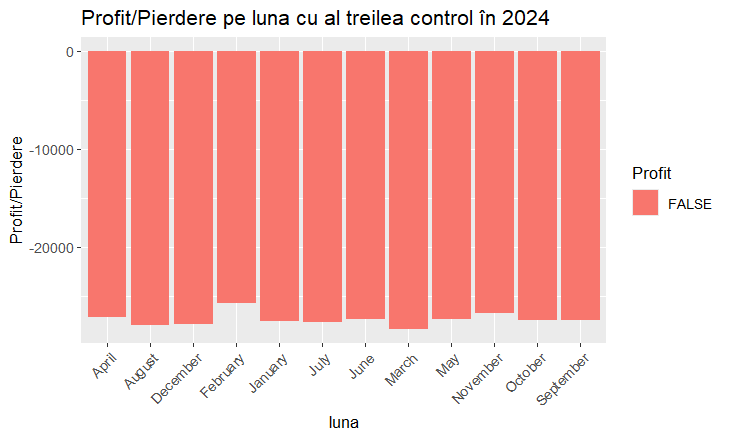
8 August 5425 19902 13566.91 -28043.91

9 September 5250 19260 13464.11 -27474.11

10 October 5425 19902 13053.19 -27530.19

11 November 5250 19260 12717.06 -26727.06

12 December 5425 19902 13467.21 -27944.21



**Comentarii:**

La subpunctul D și E unde controlorul urca de 2, respectiv 3 ori să verifice biletele, în cod am înmulțit costul fiecarui controlor, dar am considerat că media amenzilor rămâne aceeași. Dacă înmulțeam amenzile cu 2, respectiv 3, probabilitatea ar fi putut să se schimbe.

**Problema III**

gen\_numere <- function(n) {

  # Functie pentru generarea primei valori x1

  gen\_x1 <- function() {

    cnt <- 0

    repeat {

      # Citim timpul sistemului

      time <- format(Sys.time(), "%H:%M:%S")

      h <- as.numeric(substr(time, 1, 2))

      m <- as.numeric(substr(time, 4, 5))

      s <- as.numeric(substr(time, 7, 8))

      t1 <- m \* 100 + s

      t1\_mod\_23 <- t1 %% 23

      if (t1\_mod\_23 == 0) {

        return(rnorm(1, mean = m, sd = s))

      } else if (t1\_mod\_23 == 3) {

        return(rpois(1, lambda = m) + runif(1, min = -1, max = 1))

      } else if (t1\_mod\_23 == 5) {

        return(rexp(1, rate = h))

      } else if (t1\_mod\_23 == 7) {

        return(rbinom(1, size = h, prob = 1/m) + runif(1, min = 0, max = 5))

      } else if (t1\_mod\_23 == 8) {

        return(runif(1, min = -5, max = 7))

      } else if (t1\_mod\_23 == 11) {

        return(rgamma(1, shape = h, rate = 1) - rhyper(1, m = m, n = 23, k = s))

      }

      cnt <- cnt + 1

      if (cnt == 2) {

        return(rnorm(1, mean = 0, sd = 1))

      }

    }

  }

  # Generam prima valoare x1

  x1 <- gen\_x1()

  # Initializam vectorul de rezultate

  rez <- numeric(n)

  rez[1] <- x1

  # Generam restul valorilor xn

  for (i in 2:n) {

    a <- rexp(1, rate = 5)

    b <- rnorm(1, mean = 2, sd = 1)

    rez[i] <- a \* rez[i - 1] + b

  }

  # Returnam vectorul de rezultate si realizam histograma

  hist(rez, main = "Histograma", xlab = "Valori", ylab = "Frecventa", col = "blue")

  return(rez)

}

# Exemplu de utilizare

set.seed(6)  # Setam un seed pentru reproductibilitate

gen <- gen\_numere(100)

print(gen)

**Rezultat:**

[1] 0.2696060 1.7352367 0.6043572 2.7466007 2.4813558 3.9268817 3.2844986 2.1795479 3.5236278 4.8472807

[11] 5.8315356 1.5739180 4.2446868 2.8320873 1.6136272 3.5324156 0.9870502 4.6458299 4.0692314 2.1805505

[21] 2.8155597 2.3399357 1.8446347 3.0394693 5.2538587 2.4416660 2.6776003 1.3400559 3.2966923 3.5757703

[31] 1.8986359 3.6015599 3.7040097 2.5689951 0.5386401 2.8928119 1.5845141 4.5433002 2.6550742 3.2016758

[41] 2.4328999 3.9546018 2.7294727 3.2432819 6.4421556 4.3335807 2.1612923 3.9745664 6.5361727 3.2967278

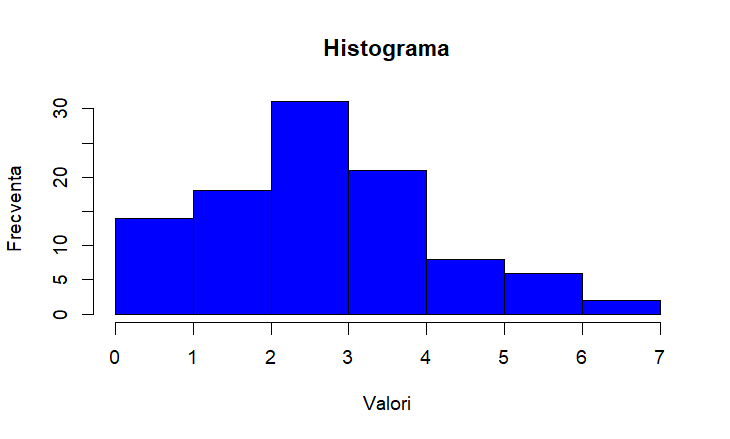
[51] 0.2555606 5.2057759 3.5357812 2.2412807 3.9661329 5.5986316 4.2617979 1.7525422 1.6676988 0.6772550

[61] 1.1150734 3.5998876 0.4407173 2.6351114 2.1342657 2.2150973 3.4225267 2.4964406 3.7840885 2.6516102

[71] 5.0938800 0.6328924 1.0635479 1.1538202 2.2221089 4.1432271 1.1237758 0.6334477 0.6118313 2.8690803

[81] 1.5206597 2.2919212 3.2428952 2.4503201 1.5669610 1.1426504 1.2547073 0.8709070 1.6303251 0.7399746

[91] 2.7609545 2.9372222 2.4575259 2.9449505 5.5180692 2.0032735 0.6997039 2.0522244 3.2835993 0.7936412



**Comentarii:**

Problema a fost rezolvată urmând pașii din cerință, x1 este generat folosind timpul actual, respectând subpunctele 1) -> 8). Celelalte numere au fost generate folosind formula dată (xn = a \* xn-1 + b).