Analiza probabilistică a performanței unui serviciu online

Mitrache Antonie Daniel
Folea Alexandru
Ene Andrei Tudor
Smadu Andrei

Ianuarie 2026

Contents

1. Descrierea Problemei

Scopul acestui proiect este modelarea unui sistem IT supus unui trafic aleator. Vom analiza relația dintre încărcarea sistemului (numărul de clienți), performanța tehnică (timpi de răspuns, erori) și impactul economic (profit vs pierderi).

Proiectul simulează scenarii realiste precum: retry-uri automate, mecanisme de backoff și comportamentul utilizatorilor (churn).

Deoarece acest proiect se bazează pe simulări Monte Carlo și generare de variabile aleatoare, rezultatele pot varia la fiecare rulare a codului.

Pentru a asigura **reproductibilitatea** exactă a rezultatelor prezentate în acest raport (grafice, valori numerice, estimări), vom seta un seed global la începutul execuției:

```
set.seed(8421)
```

2. Modelarea Traficului Zilnic (Cerința 1)

a) Modelarea Kd folosind doua distributii discrete

Pentru a modela numărul de vizitatori zilnici (K_d), am utilizat **Distribuția Poisson** si **Distributia Binomiala**, creand o functie care simuleaza traficul pe un numar de zile dat ca parametru, alaturi de distributia folosita si parametrii specifici acesteia:

```
#' Simuleaza traficul zilnic (Numarul de clienti Kd)
#' @param n_zile Numarul de zile pentru simulare
#' @param metoda Tipul distributiei: "poisson" sau "binomiala"
#' @param lambda Media estimata a clientilor (pentru Poisson)
#' @param n_max Capacitatea maxima teoretica (pentru Binomial)
#' @param p Probabilitatea ca un potential client sa fie activ (pentru Binomial)
simuleaza_trafic <- function(n_zile, metoda = "poisson", lambda = 1000, n_max = 2000, p = 0.5) {
  trafic <- numeric(n_zile)
  if (metoda == "poisson") {
    trafic <- rpois(n = n_zile, lambda = lambda)
  } else if (metoda == "binomiala") {
    trafic <- rbinom(n = n_zile, size = n_max, prob = p)
  } else {
    stop("Metoda necunoscuta! Alege 'poisson' sau 'binomiala'.")
  }
  return(trafic)
}
```

b) Generarea unor esantioane care sa reprezinte traficul zilnic pe cativa ani si reprezentarea histogramelor. Interpretarea comparativa a histogramelor obtiune pe luni si ani.

Vom folosi urmatoarele functii:

```
#' Construieste un data frame (un tabel) in care sunt reprezentate ambele distributii
#' si fiecare valoare are asociata o un an, o luna si o zi
#' @param n_zile Numarul de zile pentru simulare
#' @param metoda Tipul distributiei: "poisson" sau "binomiala"
#' @param lambda Media estimata a clientilor (pentru Poisson)
#' @param n_max Capacitatea maxima teoretica (pentru Binomial)
#' @param p Probabilitatea ca un potential client sa fie activ (pentru Binomial)
#' @param data_start Data de la care incepe simularea, data_final e implicit data_start + n_zile
df_trafic_zile <- function(n_zile, lambda = 1000, n_max = 2000, p = 0.5, data_start = "2026-01-01") {
  trafic_poisson <- simuleaza_trafic(n_zile, metoda = "poisson", lambda = lambda)
  trafic_binomial <- simuleaza_trafic(n_zile, metoda = "binomiala", n_max = n_max, p = p)
```

```r
  data_start <- as.Date(data_start)
  vector_date <- seq(from = data_start, by = "day", length.out = n_zile)

  # Cream un Data Frame pentru a putea filtra usor
  df <- data.frame(
    valoare_poisson  = trafic_poisson,
    valoare_binomial = trafic_binomial,
    an               = format(vector_date, "%Y"),
    luna             = format(vector_date, "%m"),
    zi               = format(vector_date, "%d")
  )

  return(df)
}


#' Functia deseneaza 2 histograme, una pentru valorile traficului modelat folosind poisson iar a doua p
#' @param df_trafic_zile dataframe returnat de functia df_trafic_zile
comparatie_histograme_pois_bin <- function(df_trafic_zile) {
  par(mfrow = c(1, 2)) # Impartim ecranul in 2

  hist(df_trafic_zile$valoare_poisson,
    breaks = 30,
    probability = TRUE,
    col = "skyblue",
    border = "white",
    main = "Model Poisson\n(Trafic Nerestrictionat)",
    xlab = "Nr. Clienti / Zi",
    ylab = "Densitate / Probabilitate"
  )

  hist(df_trafic_zile$valoare_binomial,
    breaks = 30,
    probability = TRUE,
    col = "lightgreen",
    border = "white",
    main = "Model Binomial\n(Capacitate Limitată)",
    xlab = "Nr. Clienti / Zi",
    ylab = "Densitate / Probabilitate"
  )

  par(mfrow = c(1, 1))
}


#' Deseneaza histograma traficului pentru o luna specifica
#'
#' Aceasta functie filtreaza datele pentru o anumita luna si un anumit an,
#' apoi afiseaza histograma distributiei selectate. Utila pentru a observa
#' variabilitatea pe esantioane mici (n ~ 30).
#'
#' @param df_trafic_zile Data frame-ul generat de df_trafic_zile()
#' @param luna Numarul lunii ca string (ex: "01", "12")
```

```r
#' @param an Anul din care extragem luna (ex: "2026")
#' @param distributie Tipul coloanei de desenat: "poisson" sau "binomiala"
#' @return Nu returneaza nimic, doar genereaza graficul.
histograma_luna_distrib <- function(df_trafic_zile,
                                    luna = "01",
                                    an = "2026",
                                    distributie = "poisson") {
  # 1. Filtram datele (Doar luna si anul cerut)
  date_filtrate <- df_trafic_zile[df_trafic_zile$luna == luna & df_trafic_zile$an == an, ]

  # 2. Selectam coloana corecta pe baza parametrului 'distributie'
  if (distributie == "poisson") {
    valori <- date_filtrate$valoare_poisson
    titlu_model <- "Poisson"
  } else {
    valori <- date_filtrate$valoare_binomial
    titlu_model <- "Binomiala"
  }

  # 3. Desenam Histograma
  hist(valori,
    breaks = 10,
    probability = TRUE,
    col = "gold",
    border = "white",
    main = paste0("Model ", titlu_model, " - Luna ", luna, "/", an),
    xlab = "Nr. Clienti / Zi",
    ylab = "Frecventa"
  )
}


#' Deseneaza histograma traficului pentru un an intreg
#'
#' Aceasta functie filtreaza datele pentru un an specific si afiseaza
#' histograma. Utila pentru a observa stabilitatea distributiei pe termen lung.
#'
#' @param df_trafic_zile Data frame-ul generat de df_trafic_zile()
#' @param an Anul dorit ca string (ex: "2026")
#' @param distributie Tipul coloanei de desenat: "poisson" sau "binomiala"
#' @return Nu returneaza nimic, doar genereaza graficul.
histograma_an_distrib <- function(df_trafic_zile,
                                  an = "2026",
                                  distributie = "poisson") {
  # 1. Filtram datele (Doar anul cerut)
  date_filtrate <- df_trafic_zile[df_trafic_zile$an == an, ]

  # 2. Selectam coloana
  if (distributie == "poisson") {
    valori <- date_filtrate$valoare_poisson
    titlu_model <- "Poisson"
  } else {
    valori <- date_filtrate$valoare_binomial
```

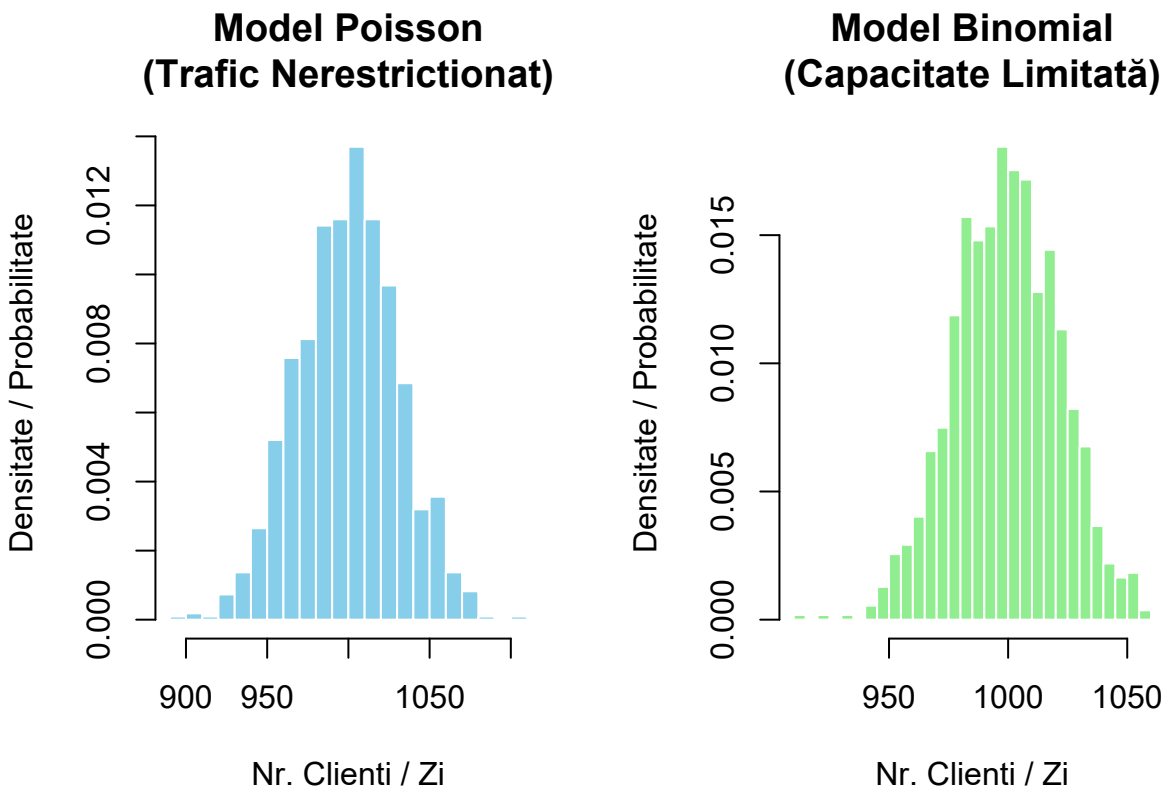
```
    titlu_model <- "Binomiala"
  }

  # 3. Desenam Histograma
  hist(valori,
    breaks = 30,
    probability = TRUE,
    col = "orange",
    border = "white",
    main = paste0("Model ", titlu_model, " - Anul ", an),
    xlab = "Nr. Clienti / Zi",
    ylab = "Densitate"
  )
}
```

Aceste functii ne ajuta la compararea traficului din orice luna cu traficul din orice an, folosind oricare dintre cele doua distributii, codul devenind astfel reutilizabil.

Sa generam prin simulare un esantion de 3 ani (3 * 365 zile).

Vom considera urmatori parametrii pentru distributii: $\lambda = 1000$ n $= 2000$ p $= 0.5$

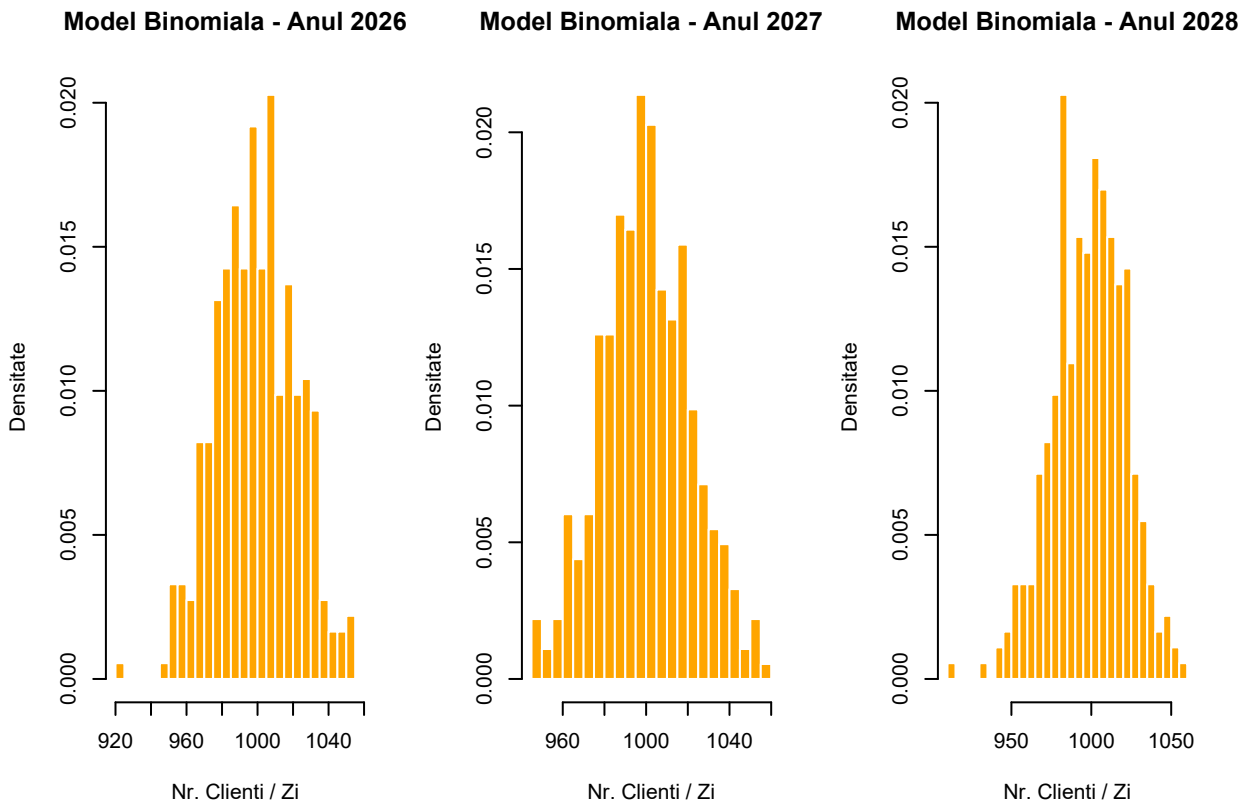
```
trafic_3_ani <- df_trafic_zile(n_zile = 3 * 365)

comparatie_histograme_pois_bin(trafic_3_ani)
```



Acum vom reprezenta histograma pentru fiecare an individual.

Mai intai folosind disributia Binomiala:

```
par(mfrow = c(1, 3))

# Anul 1
histograma_an_distrib(trafic_3_ani, an = "2026", distributie = "binomiala")

# Anul 2
histograma_an_distrib(trafic_3_ani, an = "2027", distributie = "binomiala")

# Anul 3
histograma_an_distrib(trafic_3_ani, an = "2028", distributie = "binomiala")
```

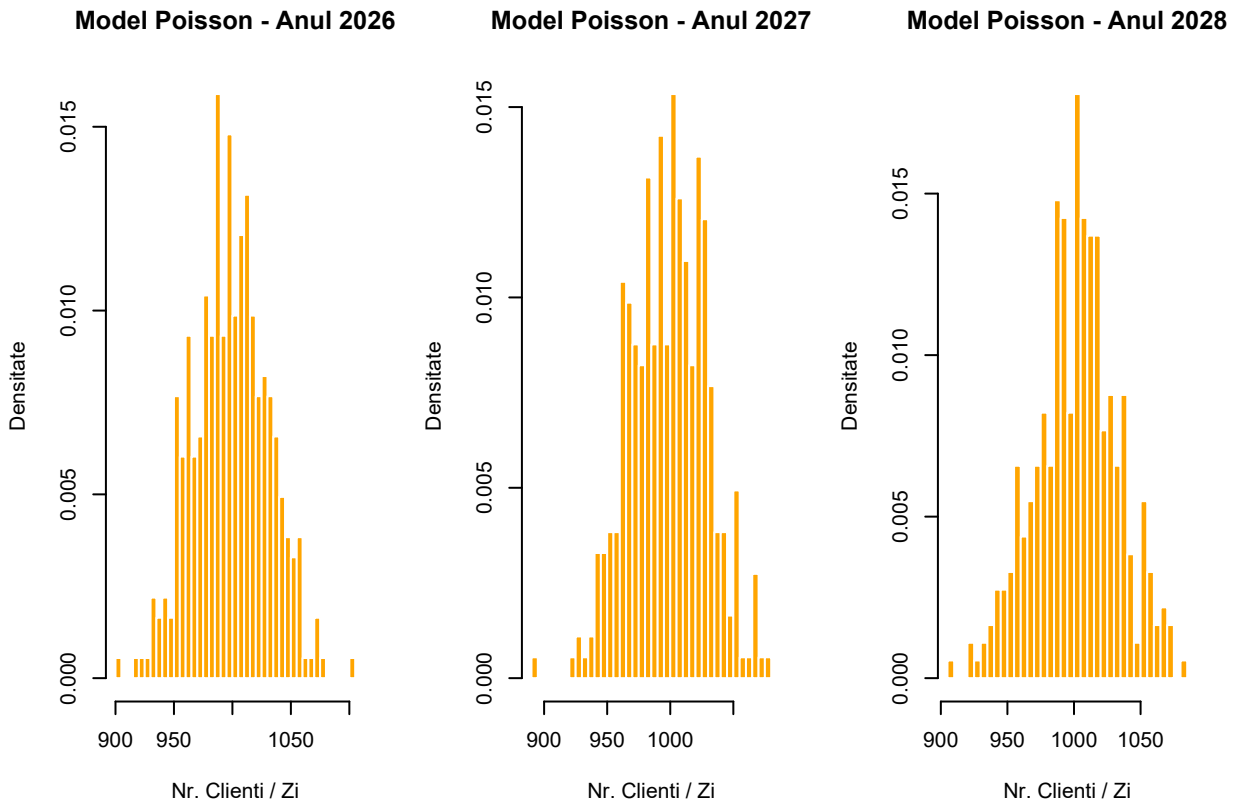



```
par(mfrow = c(1, 1))
```

Acum folosind distributia Poisson:

```
par(mfrow = c(1, 3))

# Anul 1
histograma_an_distrib(trafic_3_ani, an = "2026", distributie = "poisson")

# Anul 2
histograma_an_distrib(trafic_3_ani, an = "2027", distributie = "poisson")
```

```r
# Anul 3
histograma_an_distrib(trafic_3_ani, an = "2028", distributie = "poisson")
```




```r
par(mfrow = c(1, 1))
```

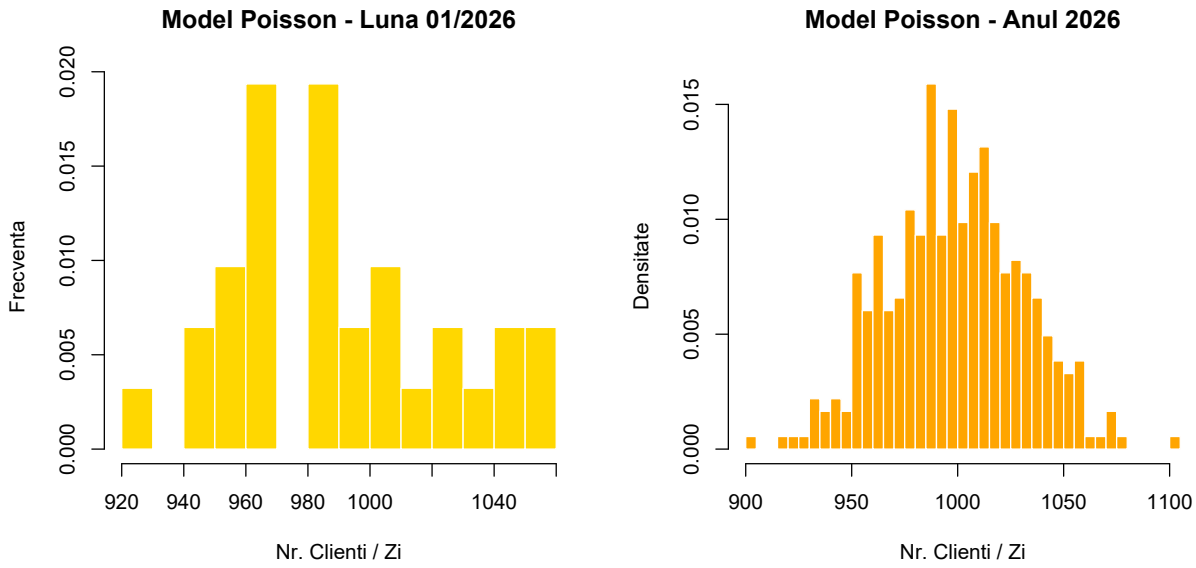
Comparatie Lună vs. An

Vom analiza diferența dintre distribuția traficului pe un interval scurt (o lună) și pe un interval lung (un an).

Poisson:

```r
par(mfrow = c(1, 2))

# Grafic 1: Doar Luna Ianuarie 2026 (Esantion Mic, n=31)
histograma_luna_distrib(trafic_3_ani,
  luna = "01",
  an = "2026",
  distributie = "poisson"
)

# Grafic 2: Tot Anul 2026 (Esantion Mare, n=365)
histograma_an_distrib(trafic_3_ani,
  an = "2026",
```

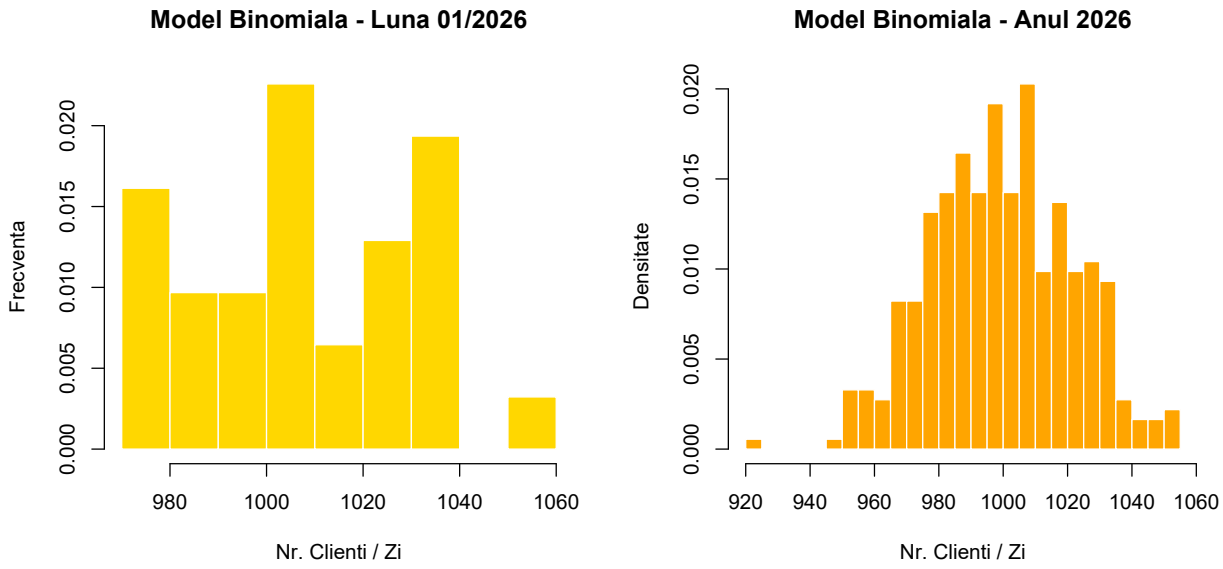
```
  distributie = "poisson"
)
```




```
par(mfrow = c(1, 1))
```

Binomiala:

```
par(mfrow = c(1, 2))

# Grafic 1: Doar Luna Ianuarie 2026 (Esantion Mic, n=31)
histograma_luna_distrib(trafic_3_ani,
  luna = "01",
  an = "2026",
  distributie = "binomiala"
)

# Grafic 2: Tot Anul 2026 (Esantion Mare, n=365)
histograma_an_distrib(trafic_3_ani,
  an = "2026",
  distributie = "binomiala"
)
```




```
par(mfrow = c(1, 1))
```

Interpretarea Histogramei Comparativă:

1. **Analiza Lunară (Eșantion Mic, $n \approx 30$):**
   - În graficul din stânga, observăm că distribuția este **neregulată** și prezintă "goluri" sau bare de înălțimi inegale care nu par să urmeze o formă matematică clară.
   - Aceasta se datorează dimensiunii reduse a eșantionului. Variabilitatea naturală (zgomotul statistic) are un impact puternic, iar frecvența empirică nu a avut timp să conveargă către probabilitatea teoretică.
   - *Concluzie practică:* O singură lună de date poate fi înșelătoare pentru a trage concluzii generale despre comportamentul sistemului.
2. **Analiza Anuală (Eșantion Mare, $n = 365$):**
   - În graficul din dreapta, forma histogramei devine mult mai netedă și simetrică, semănând vizibil cu un **Clopot (Gauss)**.
   - Acest fenomen ilustrează **Legea Numerelor Mari**: pe măsură ce numărul de observații crește, distribuția empirică (ce măsurăm) converge către distribuția teoretică (modelul matematic).
   - Media empirică se stabilizează în jurul valorii teoretice ($\lambda = 1000$), iar outlierii (valorile extreme) devin mai puțin semnificativi procentual.

c) Estimarea empirica a mediei si a variantei pe fiecare an si compararea cu valorile teoretice

Vom folosi urmatoarea functie:

```
#' Calculeaza tabelul comparativ (Empiric vs Teoretic)
#'
#' @param df_trafic_zile Data frame-ul cu datele simulate
#' @param distributie "poisson" sau "binomiala"
#' @param lambda Parametrul lambda (pt Poisson)
```

```r
#' @param n_max Parametrul n (pt Binomiala)
#' @param p Parametrul p (pt Binomiala)
df_estimari_empirice <- function(df_trafic_zile,
                                 distributie = "poisson",
                                 lambda = 1000,
                                 n_max = 2000,
                                 p = 0.5) {
  # 1. Selectam coloana de date si calculam Teoria
  if (distributie == "poisson") {
    # Selectam coloana specifica Poisson
    valori_observate <- df_trafic_zile$valoare_poisson

    # Formule teoretice Poisson
    media_teo <- lambda
    varianta_teo <- lambda
  } else {
    # Selectam coloana specifica Binomiala
    valori_observate <- df_trafic_zile$valoare_binomial

    # Formule teoretice Binomiala
    media_teo <- n_max * p
    varianta_teo <- n_max * p * (1 - p)
  }

  # Cream un mic data frame temporar doar pentru agregare
  df_temp <- data.frame(
    an = df_trafic_zile$an,
    valoare = valori_observate
  )

  # 2. Calculam Statisticile EMPIRICE (din date) pe ani
  # aggregate returneaza un tabel cu coloanele "an" si "valoare"
  medii_empirice <- aggregate(valoare ~ an, data = df_temp, FUN = mean)
  variante_empirice <- aggregate(valoare ~ an, data = df_temp, FUN = var)

  # 3. Construim Tabelul Final
  tabel_comparativ <- data.frame(
    Anul = medii_empirice$an,
    Media_Empirica = round(medii_empirice$valoare, 2),
    Media_Teoretica = media_teo,
    Varianta_Empirica = round(variante_empirice$valoare, 2),
    Varianta_Teoretica = varianta_teo
  )

  return(tabel_comparativ)
}
```

Vom compara mai intai valorile empirice cu cele teoretice folosind distributia Binomiala:

```r
df_estimari_empirice(trafic_3_ani, distributie = "binomiala")
```

```
##   Anul Media_Empirica Media_Teoretica Varianta_Empirica Varianta_Teoretica
## 1 2026         999.79            1000            484.73                500
```

```
## 2 2027        1000.64            1000            456.11               500
## 3 2028         999.04            1000            519.98               500
```

Iar apoi folosind distributia Poisson:

```
df_estimari_empirice(trafic_3_ani, distributie = "poisson")
```

```
##   Anul Media_Empirica Media_Teoretica Varianta_Empirica Varianta_Teoretica
## 1 2026         998.95            1000           1005.50               1000
## 2 2027         998.92            1000            915.61               1000
## 3 2028        1002.46            1000            929.69               1000
```

d) Interpretarea diferentelor dintre modele (trafic redus vs plafonat)

Vom folosi urmatoarea functie ajutatoare:

```
#' Functie ajutatoare la compararea celor doua modele - Poisson si Binomiala
#' @param df_trafic_zile variabila returnata de functia df_trafic_zile
comparatie_densitati_suprapuse <- function(df_trafic_zile) {
  # Calculam densitatile (forma curbelor)
  dens_pois <- density(df_trafic_zile$valoare_poisson)
  dens_bin <- density(df_trafic_zile$valoare_binomial)

  # Calculam limitele pentru a incapea ambele
  limita_x <- range(c(dens_pois$x, dens_bin$x))
  limita_y <- range(c(dens_pois$y, dens_bin$y))

  # Desenam prima curba (Poisson)
  plot(dens_pois,
    xlim = limita_x, ylim = limita_y,
    col = "blue", lwd = 3,
    main = "Comparatie Directa: Poisson vs Binomial",
    xlab = "Nr. Clienti / Zi", ylab = "Probabilitate"
  )

  # Umplem suprafata Poisson cu albastru transparent
  polygon(dens_pois, col = rgb(0, 0, 1, 0.2), border = NA)

  # Adaugam a doua curba (Binomiala) peste prima
  lines(dens_bin, col = "forestgreen", lwd = 3)

  # Umplem suprafata Binomiala cu verde transparent
  polygon(dens_bin, col = rgb(0, 0.5, 0, 0.2), border = NA)

  # Legenda
  legend("topright",
    legend = c("Poisson", "Binomial"),
    col = c("blue", "forestgreen"),
    lwd = 3, fill = c(rgb(0, 0, 1, 0.2), rgb(0, 0.5, 0, 0.2))
  )

  grid()
}
```

Deși ambele distribuții au fost parametrizate pentru a avea aceeași medie teoretică ($E[X] \approx 1000$ clienți/zi), analiza comparativă relevă diferențe fundamentale de stabilitate:

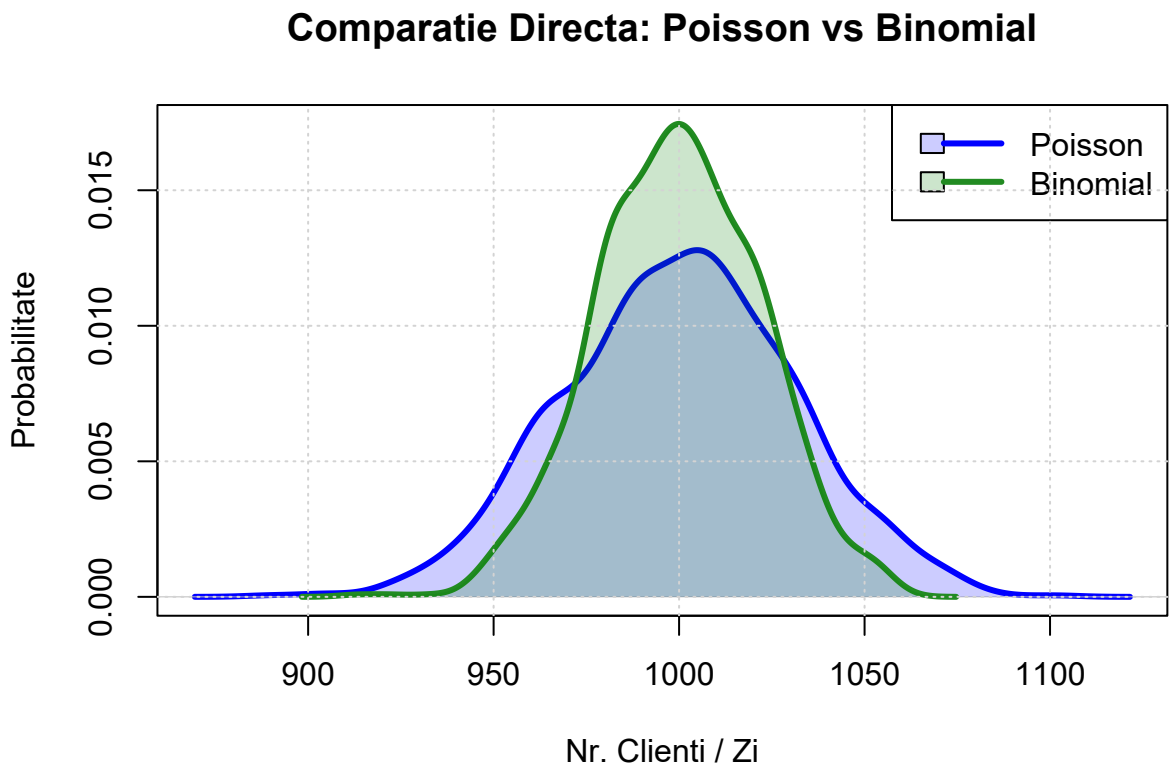
1. Dispersia (Varianța):

- **Modelul Poisson** (Trafic Nerestricționat) are varianța egală cu media ($Var = \lambda = 1000$). Aceasta indică o fluctuație naturală mare; într-o zi proastă putem avea mult sub 1000 de clienți, iar într-o zi bună mult peste, fără o limită superioară strictă.
- **Modelul Binomial** (Trafic Plafonat) are varianța mult mai mică. Conform formulei $Var = n \cdot p \cdot (1 - p) = 2000 \cdot 0.5 \cdot 0.5 = 500$.
- **Concluzie:** Varianța modelului Binomial este la jumătate față de cel Poisson ($500 < 1000$). Asta înseamnă că traficul Binomial este de două ori mai "concentrat" în jurul mediei.

2. Interpretare Vizuală:

- Pe grafic, histograma **Binomială (Verde)** este mai "înaltă" și mai "îngustă". Valorile sunt strânse în intervalul $[950, 1050]$.
- Histograma **Poisson (Albastră)** este mai "lată" și mai "joasă". Cozile distribuției se întind mai mult, existând o probabilitate mai mare de a întâlni valori extreme (departe de medie).

```
comparatie_densitati_suprapuse(trafic_3_ani)
```



3. Implicații pentru Sistemul IT:

- **Scenariul Poisson** este mai riscant. Deoarece traficul nu este plafonat de un număr finit de utilizatori (N_{max}), pot apărea "vârfuri" neașteptate care să suprasolicite serverele.
- **Scenariul Binomial** este mai predictibil. Existența unui bazin finit de clienți ($N_{max} = 2000$) acționează ca un amortizor natural al fluctuațiilor. Este un model specific aplicațiilor interne (corporate) sau pe bază de abonament limitat, unde dimensionarea serverelor este mai simplă și mai eficientă economic.

3. Modelarea timpilor de raspuns (Cerinta 2)

a) Modelarea S folosind o distributie asimetrica si una simetrica

Pentru a modela timpul de procesare S (latenta), am utilizat 2 ipoteze:

1. **Distribuție Exponențială:** Presupune că majoritatea cererilor sunt rapide, dar există câteva foarte lente ("coadă lungă"). Este modelul clasic pentru servere (M/M/1).
2. **Distribuție Normală:** Presupune că timpii sunt simetrici în jurul unei medii, tipic pentru procese deterministe cu mici variații.

```
#' Simuleaza timpii de raspuns (S)
#' @param n Numarul de cereri (esantionul)
#' @param tip Tipul distributiei: "exponentiala" sau "normala"
#' @param medie Timpul mediu dorit (ex: 200 ms) atat pentru distributia exponentiala cat si cea normala
#' @param sd Deviatia standard pentru distributia normala
simuleaza_latente <- function(n, tip = "exponentiala", medie = 200, sd = 50) {
  if (tip == "exponentiala") {
    rate_param <- 1 / medie
    valori <- rexp(n, rate = rate_param)
  } else if (tip == "normala") {
    valori <- rnorm(n, mean = medie, sd = sd)

    # Timpul nu poate fi negativ. Orice valoare < 1ms devine 1ms.
    valori <- pmax(valori, 1)
  } else {
    stop("Tip distributie necunoscut! Alege 'exponentiala' sau 'normala'.")
  }

  return(valori)
}
```

b) Construirea histogramelor pentru S si suprapunerea densitatilor teoretice

Vom folosi urmatoarea rezolvare:

```
#' Vizualizeaza histograma si densitatea teoretica
#' @param vector_latente Datele generate de functia simuleaza_latente
#' @param tip "exponentiala" sau "normala"
#' @param medie Media folosita la generare (pt curba teoretica)
#' @param sd Deviatia standard (doar pt normala)
vizualizeaza_latente <- function(vector_latente, tip = "exponentiala", medie = 200, sd = 50) {
  # Desenam Histograma
```

```
  if (tip == "exponentiala") {
    titlu <- "Latente: Distributie Exponentiala (Asimetrică)"
    culoare <- "wheat"
  } else {
    titlu <- "Latente: Distributie Normala (Simetrica)"
    culoare <- "lightblue"
  }

  hist(vector_latente,
    probability = TRUE,
    breaks = 30,
    col = culoare,
    border = "white",
    main = titlu,
    xlab = "Timp de Răspuns (ms)",
    ylab = "Densitate"
  )

  # Suprapunem densitatea teoretica (formula matematica)

  if (tip == "exponentiala") {
    # Functia de densitate exponentiala este dexp
    curve(dexp(x, rate = 1 / medie),
      col = "red", lwd = 3, add = TRUE
    )
  } else {
    # Functia de densitate normala este dnorm
    curve(dnorm(x, mean = medie, sd = sd),
      col = "darkblue", lwd = 3, add = TRUE
    )
  }
}
```

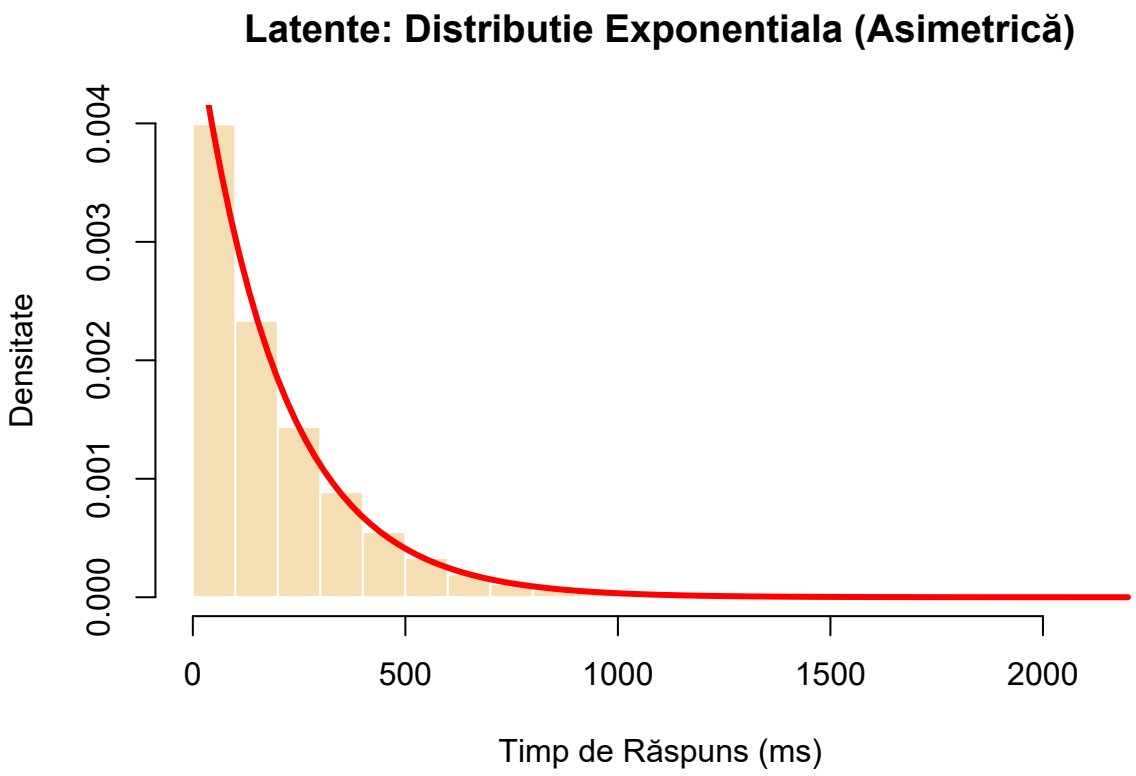
Vom considera urmatorii parametrii pentru distributii: media $= 200$ $\sigma = 50$

Vom simula latentele pentru 10000 de request-uri:

```
latente_exp <- simuleaza_latente(10^4, tip = "exponentiala")
latente_norm <- simuleaza_latente(10^4, tip = "normala")
```

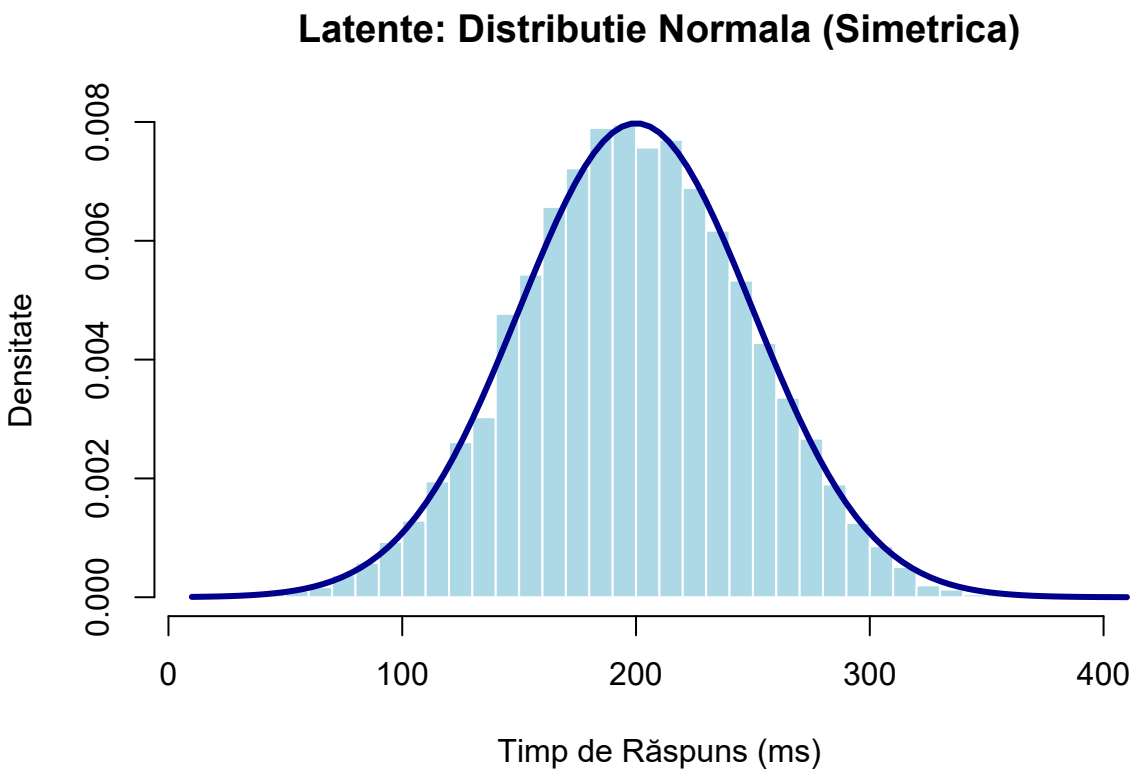
Vizualizarea histogramei si a densitatii teoretice pentru modelul exponential:

```
vizualizeaza_latente(latente_exp, "exponentiala")
```



Si pentru cel normal:

```
vizualizeaza_latente(latente_norm, "normala")
```



c) Calculul valorilor descriptive (Media, Varianta, Mediana, Modul) si interpretarea lor

Vom folosi urmatoarea functie:

```r
#' Calculeaza statisticile descriptive (Media, Varianta, Mediana, Modul)
#' si returneaza un tabel comparativ.
#'
#' @param v_exp Vectorul cu datele distributiei Exponentiale
#' @param v_norm Vectorul cu datele distributiei Normale
#' @return Un Data Frame cu statisticile rotunjite la 2 zecimale
comparatie_statistica_latente <- function(v_exp, v_norm) {
  # Pentru variabile continue, Modul este varful curbei de densitate
  calculeaza_modul <- function(v) {
    d <- density(v)
    return(d$x[which.max(d$y)])
  }

  # Calculam statisticile pentru exponentiala
  stats_exp <- c(
    Media    = mean(v_exp),
    Varianta = var(v_exp),
    Mediana  = median(v_exp),
    Modul    = calculeaza_modul(v_exp)
  )
```

```
  # Calculam statisticile pentru normala
  stats_norm <- c(
    Media    = mean(v_norm),
    Varianta = var(v_norm),
    Mediana  = median(v_norm),
    Modul    = calculeaza_modul(v_norm)
  )

  # Construim Data Frame-ul final
  df_stat <- data.frame(
    Distributia = c("Exponentiala", "Normala"),
    Media = c(stats_exp["Media"], stats_norm["Media"]),
    Varianta = c(stats_exp["Varianta"], stats_norm["Varianta"]),
    Mediana = c(stats_exp["Mediana"], stats_norm["Mediana"]),
    Modul = c(stats_exp["Modul"], stats_norm["Modul"])
  )

  # Rotunjim valorile numerice pentru un aspect curat in tabel
  # (Coloanele 2 pana la 5 sunt numerice)
  df_stat[, 2:5] <- round(df_stat[, 2:5], 2)

  return(df_stat)
}
```

Rezultat:

```
comparatie_statistica_latente(latente_exp, latente_norm)
```

```
##    Distributia  Media Varianta Mediana  Modul
## 1 Exponentiala 198.00 38791.04  137.16  40.58
## 2      Normala 199.02  2434.73  198.65 190.26
```

Pe baza tabelului de mai sus, putem trage următoarele concluzii despre natura celor două distribuții:

1. Distribuția Normală (Modelul Simetric)

- **Centralitate:** Valorile pentru **Medie** (≈ 199), **Mediană** (≈ 198) și **Mod** (≈ 190) sunt foarte apropiate unele de altele. Această coincidență este specifică distribuțiilor simetrice (Clopotul lui Gauss), unde majoritatea datelor se aglomerează în jurul centrului.
- **Varianța:** Valoarea obținută (≈ 2434) este consistentă cu deviația standard teoretică aleasă ($SD = 50 \Rightarrow Var = 2500$).
- **Concluzie:** Acest model descrie un sistem **stabil și predictibil**, unde timpul de răspuns variază puțin și egal în ambele direcții față de medie.

2. Distribuția Exponențială (Modelul Asimetric)

- **Centralitate:** Observăm o discrepanță majoră între indicatori: $Mod < Median < Medie$ ($40 < 137 < 198$).
  - **Modul scăzut (≈ 40):** Indică faptul că, cel mai frecvent, cererile sunt procesate foarte rapid (sub medie).
  - **Mediana (≈ 137):** Arată că 50% dintre cereri sunt finalizate în sub 137ms.

  - **Media ridicată (≈ 198):** Este trasă în sus de valorile extreme (coada distribuției).
- **Varianța:** Este foarte mare (≈ 38.791), mult superioară celei normale.
- **Concluzie:** Acest model descrie un sistem **realist, dar instabil**, caracterizat de o "coadă lungă" (*Long Tail*). Deși majoritatea cererilor sunt rapide, există un risc semnificativ de latențe foarte mari care destabilizează media.

d) Diferenta dintre medie si mediana in contextul latentelor

Analizând tabelul statistic generat anterior, observăm o distincție clară între cele două modele:

1. Cazul Distribuției Normale (Procese Stabile)

- **Observație:** Media (≈ 199 ms) este aproape identică cu Mediana (≈ 198 ms).
- **Interpretare:** Într-un sistem stabil, timpul mediu de răspuns reflectă corect experiența tipică a utilizatorului. 50% dintre cereri sunt mai rapide de 199ms, iar 50% sunt mai lente, distribuția fiind simetrică.

2. Cazul Distribuției Exponențiale (Sisteme Realiste / Queuing)

- **Observație:** Există o diferență majoră între Medie (≈ 198 ms) și Mediană (≈ 137 ms). Media este cu aproximativ **45% mai mare** decât mediana.
- **Interpretare:** Aceasta este proprietatea de **"Long Tail"** (Coadă Lungă).
  - **Mediana (137 ms):** Ne spune că *majoritatea* utilizatorilor (50%+) au o experiență foarte bună, primind răspuns rapid.
  - **Media (198 ms):** Este trasă în sus de câteva cereri "ghinioniste" care durează foarte mult (outlieri).
- **Concluzie de Business:** Pentru un server web, **Mediana** este un indicator mai bun al satisfacției generale ("utilizatorul obișnuit"), în timp ce **Media** ascunde faptul că mulți utilizatori au o experiență rapidă, dar este sensibilă la problemele grave de latență.

4. Cereri, retry-uri și evenimente (Cerința 3)

Definim evenimentele cheie asociate unei cereri care trece prin mecanismul de retry:

- $A = \{I = 1\}$: Cererea este procesată cu succes (indiferent de numărul de încercări).
- $B = \{T \leq t_0\}$: Cererea se încadrează în timpul limită setat de SLA (Service Level Agreement).
- $C = \{N \leq n_0\}$: Cererea este soluționată rapid, într-un număr mic de încercări.
- $D = \{$cel puțin un eșec$\}$: Cererea a întâmpinat dificultăți (măcar un eșec), chiar dacă a reușit sau nu la final.

În simularea noastră, vom considera următorii parametri: * $n_0 = 2$ (pragul pentru "puține încercări") * $t_0 = 400$ ms (pragul de timp SLA) * $M = 10000$ simulări

```
#' Simuleaza o singura cerere care poate avea mai multe retry-uri
#'
#' @param n_max Numarul maxim de incercari permise (retry-uri + prima incercare)
#' @param p_succes Probabilitatea de succes a unei singure incercari (U_i)
#' @param t_0 Pragul de timp pentru respectarea SLA (Service Level Agreement)
#' @param medie_S Timpul mediu de procesare pentru o singura incercare (S_i)
```

```r
#' @param backoff_fix Timpul de asteptare fix adaugat intre doua reincercari (B_i)
#'
#' @return O lista continand indicatori de succes, timp si numar de incercari
print(simuleaza_o_cerere)
```

```
## function (n_max = 3, p_succes = 0.8, t_0 = 500, medie_S = 150,
##     backoff_fix = 50)
## {
##     timp_total <- 0
##     succes_final <- 0
##     nr_incercari <- 0
##     a_esuat_vreodata <- FALSE
##     for (i in 1:n_max) {
##         nr_incercari <- i
##         s_i <- rexp(1, rate = 1/medie_S)
##         timp_total <- timp_total + s_i
##         if (runif(1) < p_succes) {
##             succes_final <- 1
##             break
##         }
##         else {
##             a_esuat_vreodata <- TRUE
##             if (i < n_max) {
##                 timp_total <- timp_total + backoff_fix
##             }
##         }
##     }
##     return(list(I = succes_final, T = timp_total, N = nr_incercari,
##         D = a_esuat_vreodata))
## }
```

a) Estimați empiric: $P(A), P(B), P(C), P(A \cap B), P(A \cup D)$

Vom rula simularea Monte Carlo și vom calcula frecvențele relative ale apariției acestor evenimente.

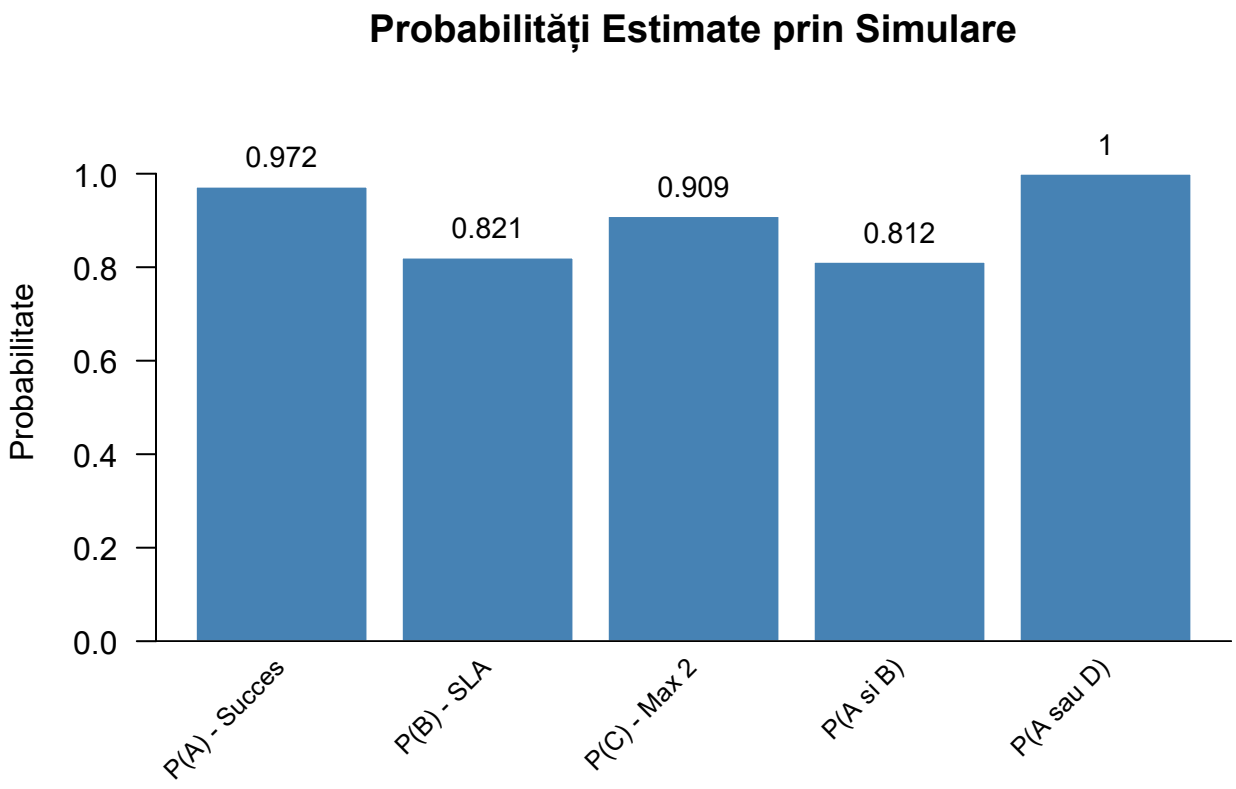
```r
M <- 10000
n0_prag <- 2
t0_prag <- 400

rezultate <- replicate(M, simuleaza_o_cerere(n_max = 3, p_succes = 0.7, t_0 = t0_prag))
df_cereri <- as.data.frame(t(rezultate))
df_cereri[] <- lapply(df_cereri, unlist)

# Definirea logica a evenimentelor pentru fiecare simulare
ev_A <- (df_cereri$I == 1)
ev_B <- (df_cereri$T <= t0_prag)
ev_C <- (df_cereri$N <= n0_prag)
ev_D <- (df_cereri$D == TRUE)

# Estimare Probabilitati (Frecvente Relative)
prob_A <- mean(ev_A)
prob_B <- mean(ev_B)
```

```r
prob_C <- mean(ev_C)
prob_inter_AB <- mean(ev_A & ev_B)
prob_reuniune_AD <- mean(ev_A | ev_D)

# Afisare Tabelara
rezultate_sim <- data.frame(
  Eveniment = c("P(A) [Succes]", "P(B) [SLA]", "P(C) [Rapid]", "P(A inter B)", "P(A U D)"),
  Probabilitate_Empirica = c(prob_A, prob_B, prob_C, prob_inter_AB, prob_reuniune_AD)
)

knitr::kable(rezultate_sim, digits = 4, caption = "Probabilități Empirice Estimate")
```

Table 1: Probabilități Empirice Estimate

Eveniment	Probabilitate_Empirica
P(A) [Succes]	0.9724
P(B) [SLA]	0.8207
P(C) [Rapid]	0.9095
P(A inter B)	0.8116
P(A U D)	1.0000

Grafic, aceste probabilități sunt distribuite astfel:



b) Verificați numeric formulele pentru reuniune/intersecție

O proprietate fundamentală a teoriei probabilităților este **Principiul Incluziunii și Excluziunii**. Pentru două evenimente A și D, relația este:

$$P(A \cup D) = P(A) + P(D) - P(A \cap D)$$

Verificăm dacă datele noastre respectă această relație:

```
# Calculam elementele componente din datele empirice
prob_D <- mean(ev_D)
prob_inter_AD <- mean(ev_A & ev_D)

# Aplicam formula teoretica folosind valorile estimate
valoare_formula <- prob_A + prob_D - prob_inter_AD

# Comparam cu valoarea obtinuta direct prin numarare (reuniune logica)
cat("P(A U D) măsurat direct (reuniune logica):", prob_reuniune_AD, "\n")
```

```
## P(A U D) măsurat direct (reuniune logica): 1
```

```r
cat("P(A) + P(D) - P(A inter D) (formula):      ", valoare_formula, "\n")
```

```
## P(A) + P(D) - P(A inter D) (formula):       1
```

```r
cat("Diferența (Eroare de calcul):              ", abs(prob_reuniune_AD - valoare_formula), "\n")
```

```
## Diferența (Eroare de calcul):               0
```

Diferența zero confirmă că relațiile dintre mulțimi se păstrează perfect în eșantionul nostru.

c) Explicați de ce probabilitatea empirică aproximează bine probabilitatea teoretică

Motivul pentru care putem folosi simularea ($M = 10000$ cazuri) pentru a estima probabilitățile reale ale sistemului este **Legea Numerelor Mari (LLN)**.

Această lege afirmă că, pe măsură ce dimensiunea eșantionului (M) crește, media empirică (frecvența relativă a observării unui eveniment) converge în probabilitate către media teoretică (probabilitatea reală a evenimentului).

Formal, dacă $X_1, X_2, ..., X_M$ sunt variabile aleatoare independente și identic distribuite (i.i.d.) care indică apariția unui eveniment (1 dacă apare, 0 altfel), atunci media lor aritmetică $\bar{X}_M$ satisface:

$$\lim_{M \to \infty} P(|\bar{X}_M - \mu| < \epsilon) = 1$$

Unde μ este probabilitatea teoretică.

În cazul nostru, $M = 10000$ este un număr suficient de mare pentru ca fluctuațiile statistice să se "anuleze" reciproc, rezultând o estimare care este foarte apropiată de valoarea "adevărată" pe care am obține-o prin calcul analitic exact. Eroarea de estimare scade proporțional cu $1/\sqrt{M}$.

5. Variabile aleatoare bidimensionale discrete (Cerința 4)

În această secțiune considerăm variabila bidimensională **(N, F)**, unde:

\begin{itemize} \item **N** = numărul total de încercări (retry-uri + prima încercare) \item **F** = numărul de eșecuri (failures) produse pe parcursul unei cereri \end{itemize}

Observație importantă: în modelul de retry, **F nu este o variabilă independentă față de N**, deoarece este determinată de relația logică dintre **N** și indicatorul de succes final **I**: - dacă cererea reușește la încercarea N (I=1) \(\Rightarrow\) au existat **N-1** eșecuri înainte, deci \(F=N-1\) - dacă cererea eșuează complet (I=0) \(\Rightarrow\) toate încercările sunt eșecuri, deci \(F=N\)

Implementarea simulării (N, F)

Funcția `simuleaza_NF_din_ex3` este un *wrapper* peste `simuleaza_o_cerere` (definită la Cerința 3) și construiește coloana **F** plecând de la (N, I).

```r
print(simuleaza_NF_din_ex3)
```

```
## function (M = 1e+05, n_max = 3, p_succes = 0.7, t_0 = 400, medie_S = 150,
##     backoff_fix = 50)
## {
##     rezultate <- replicate(M, simuleaza_o_cerere(n_max = n_max,
##         p_succes = p_succes, t_0 = t_0, medie_S = medie_S, backoff_fix = backoff_fix))
##     df <- as.data.frame(t(rezultate))
##     df[] <- lapply(df, unlist)
##     df$N <- as.integer(df$N)
##     df$I <- as.integer(df$I)
##     df$F <- ifelse(df$I == 1L, df$N - 1L, df$N)
##     return(df[, c("N", "F", "I")])
## }
```

a) Distribuția comună empirică

Simulăm un eșantion mare \(M=100000\) și estimăm distribuția comună prin frecvențe relative.

```
M <- 100000
n_max <- 3
p_succes <- 0.7
t_0 <- 400
medie_S <- 150
backoff_fix <- 50

df_NF <- simuleaza_NF_din_ex3(
  M = M, n_max = n_max, p_succes = p_succes,
  t_0 = t_0, medie_S = medie_S, backoff_fix = backoff_fix
)

tab_NF <- table(df_NF$N, df_NF$F) # frecvente absolute
prob_NF <- prop.table(tab_NF) # probabilitati empirice

knitr::kable(tab_NF, caption = "Distribuția comună (frecvențe) pentru (N, F)")
```

Table 2: Distribuția comună (frecvențe) pentru (N, F)

0	1	2	3
70083	0	0	0
0	20853	0	0
0	0	6401	2663

```
knitr::kable(round(prob_NF, 4), caption = "Distribuția comună empirică P(N, F)")
```

Table 3: Distribuția comună empirică P(N, F)

0	1	2	3
0.7008	0.0000	0.000	0.0000
0.0000	0.2085	0.000	0.0000
0.0000	0.0000	0.064	0.0266

b) Distribuțiile marginale

Marginalele se obțin prin sumare pe linii/coloane ale distribuției comune.

```
marg_N <- margin.table(prob_NF, 1) # P(N=n)
marg_F <- margin.table(prob_NF, 2) # P(F=f)

df_marg <- data.frame(
  Valoare = as.integer(names(marg_N)),
  P_N = as.numeric(marg_N)
)
knitr::kable(df_marg, digits = 4, caption = "Distribuția marginală P(N=n)")
```

Table 4: Distribuția marginală P(N=n)

Valoare	P_N
1	0.7008
2	0.2085
3	0.0906

```
df_marg2 <- data.frame(
  Valoare = as.integer(names(marg_F)),
  P_F = as.numeric(marg_F)
)
knitr::kable(df_marg2, digits = 4, caption = "Distribuția marginală P(F=f)")
```

Table 5: Distribuția marginală P(F=f)

Valoare	P_F
0	0.7008
1	0.2085
2	0.0640
3	0.0266

c) Test empiric de independență

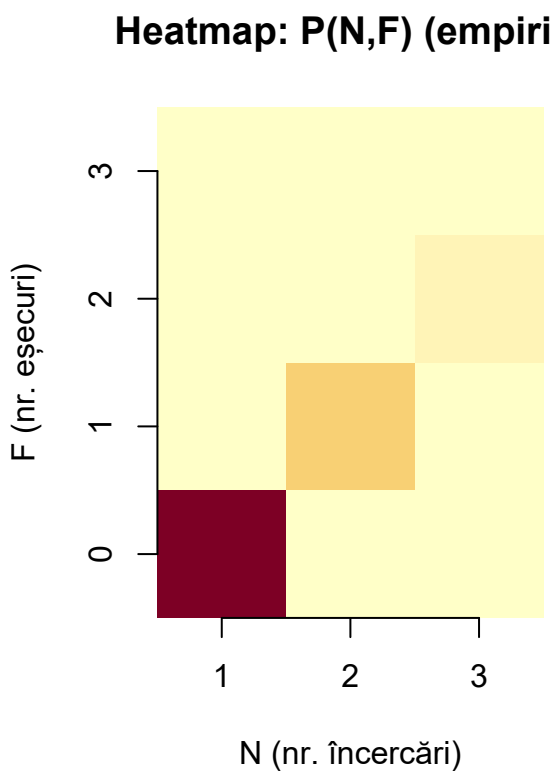
Aplicăm un test \(\chi^2\) pe tabelul de contingență (cu p-value simulat). Ipoteza nulă: **N și F sunt independente**.

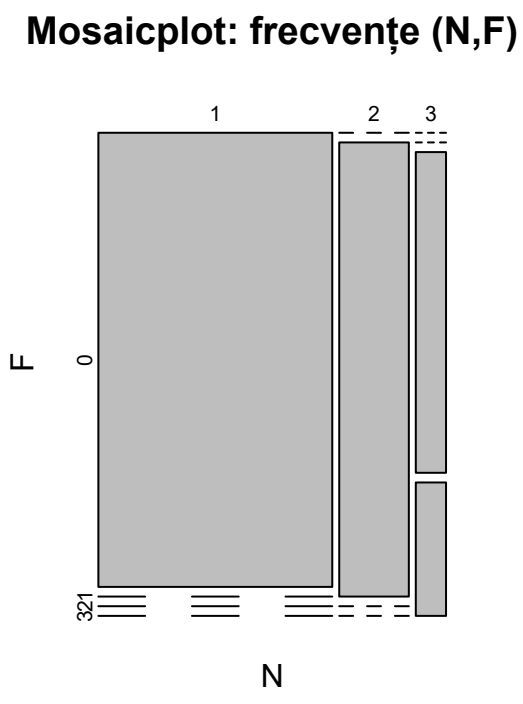
```
test_ind <- chisq.test(tab_NF, simulate.p.value = TRUE, B = 2000)
test_ind
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
##  replicates)
##
## data:  tab_NF
## X-squared = 2e+05, df = NA, p-value = 0.0004998
```

d) Vizualizare (heatmap + mosaicplot) și interpretare

```r
prob_mat <- as.matrix(prob_NF)
x_vals <- as.numeric(rownames(prob_mat)) # N
y_vals <- as.numeric(colnames(prob_mat)) # F

par(mfrow = c(1, 2))

# Heatmap (probabilitati empirice)
image(
  x = x_vals, y = y_vals, z = prob_mat,
  xlab = "N (nr. încercări)", ylab = "F (nr. eșecuri)",
  main = "Heatmap: P(N,F) (empiric)",
  axes = FALSE
)
axis(1, at = x_vals, labels = x_vals)
axis(2, at = y_vals, labels = y_vals)

# Mosaicplot (frecvente)
mosaicplot(
  tab_NF,
  main = "Mosaicplot: frecvențe (N,F)",
  xlab = "N", ylab = "F"
)
```





```r
par(mfrow = c(1, 1))
```

Interpretare: ne așteptăm ca testul de independență să respingă independența, deoarece **F este aproape determinată de N** (prin relația \(F=N-1\) la succes și \(F=N\) la eșec final). Consecința: numărul de retry-uri controlează direct numărul de eșecuri, deci variabilele sunt dependente în modelul nostru.

6. Variabile aleatoare bidimensionale (discrete și continue) (Cerința 5)

În această cerință analizăm variabila bidimensională **(N, T)**:

\begin{itemize} \item **N** = numărul total de încercări (variabilă discretă) \item **T** = timpul total până la succes sau abandon (variabilă continuă) \end{itemize}

În modelul nostru, **T** include suma timpilor de procesare \(S_i\) și timpii de așteptare de tip backoff între încercări eșuate.

Implementarea simulării (N, T)

Funcția `simuleaza_NT_din_ex3` generează un eșantion de perechi (N, T) folosind `simuleaza_o_cerere` din Cerința 3.

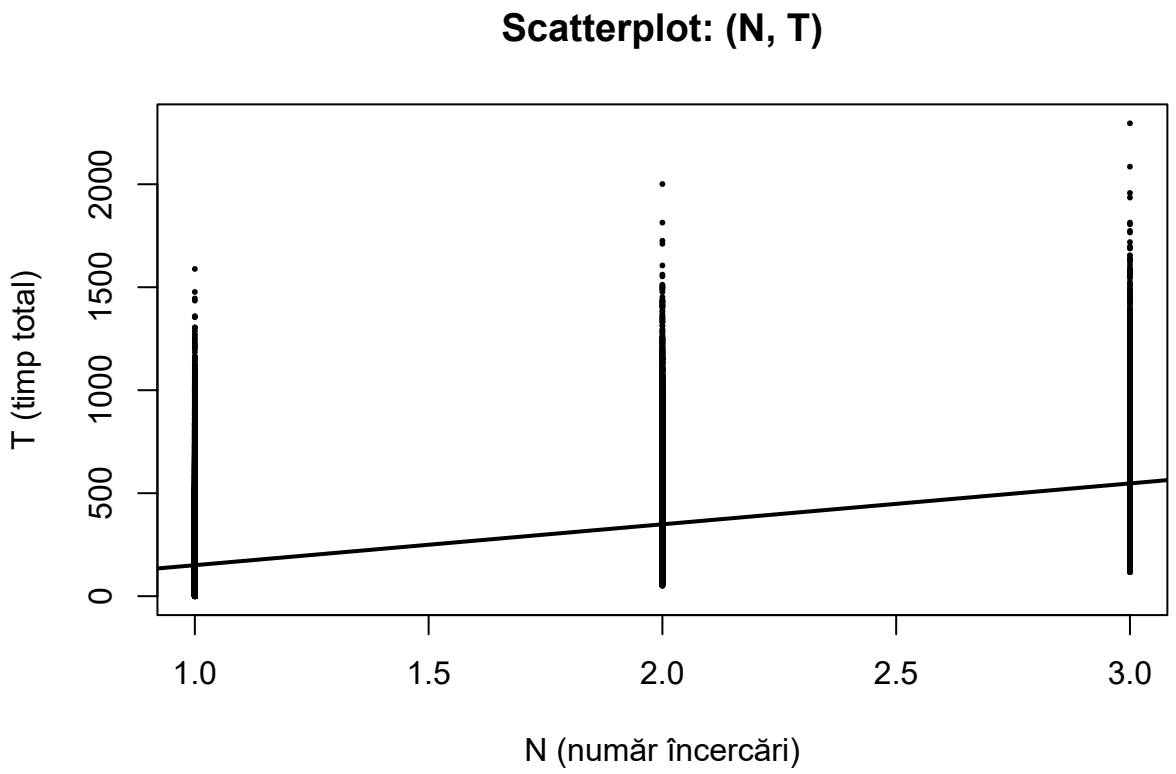
```r
print(simuleaza_NT_din_ex3)
```

```
## function (M = 1e+05, n_max = 3, p_succes = 0.7, t_0 = 400, medie_S = 150,
##     backoff_fix = 50)
## {
##     rezultate <- replicate(M, simuleaza_o_cerere(n_max = n_max,
##         p_succes = p_succes, t_0 = t_0, medie_S = medie_S, backoff_fix = backoff_fix))
##     df <- as.data.frame(t(rezultate))
##     df[] <- lapply(df, unlist)
##     df$N <- as.integer(df$N)
##     df$I <- as.integer(df$I)
##     df$T <- as.numeric(df$T)
##     df[, c("N", "T", "I")]
## }
```

a) Reprezentare grafică (scatterplot)

```r
M <- 100000
n_max <- 3
p_succes <- 0.7
t_0 <- 400
medie_S <- 150
backoff_fix <- 50

df_NT <- simuleaza_NT_din_ex3(
  M = M, n_max = n_max, p_succes = p_succes,
```

```
  t_0 = t_0, medie_S = medie_S, backoff_fix = backoff_fix
)

plot(
  df_NT$N, df_NT$T,
  xlab = "N (număr încercări)",
  ylab = "T (timp total)",
  main = "Scatterplot: (N, T)",
  pch = 16, cex = 0.4
)

# Linia de regresie pentru a evidenția trendul
abline(lm(T ~ N, data = df_NT), lwd = 2)
```



b) Medii, variațe, covariață, corelație

```
mean_N <- mean(df_NT$N)
mean_T <- mean(df_NT$T)

var_N <- var(df_NT$N)
var_T <- var(df_NT$T)

cov_NT <- cov(df_NT$N, df_NT$T)
```

```r
cor_NT <- cor(df_NT$N, df_NT$T)

rez_stats <- data.frame(
  Indicator = c("E[N]", "E[T]", "Var(N)", "Var(T)", "Cov(N,T)", "Corr(N,T)"),
  Valoare = c(mean_N, mean_T, var_N, var_T, cov_NT, cor_NT)
)

knitr::kable(rez_stats, digits = 4, caption = "Statistici empirice pentru variabila (N, T)")
```

Table 6: Statistici empirice pentru variabila (N, T)

Indicator	Valoare
E[N]	1.3936
E[T]	228.6192
Var(N)	0.4192
Var(T)	47612.5606
Cov(N,T)	83.2019
Corr(N,T)	0.5890

c) Interpretarea corelației

Ne așteptăm la **corelație pozitivă** între **N** și **T**: atunci când avem mai multe retry-uri (N mai mare), se acumulează mai mulți timpi de procesare \(S_i\) și, de regulă, mai mulți timpi de backoff, astfel încât timpul total **T** crește. În termeni de sistem, retry-urile cresc șansa de succes, dar cresc și latența percepută de utilizator.

7. Probabilități condiționate și condiționări (Cerința 6)

Folosim aceleași variabile ca în Cerința 3:

\begin{itemize} \item **I** \(\in \{0,1\}\) – indicator de succes final (1 = succes, 0 = eșec final) \item **N** – numărul total de încercări \item **T** – timpul total până la succes sau abandon \end{itemize}

Definim evenimentele: \begin{itemize} \item \(A = \{I=1\}\) – cererea are succes final \item \(B = \{T \le t_0\}\) – cererea respectă SLA (timp sub prag) \item \(C = \{N \le n_0\}\) – cererea reușește rapid (puține retry-uri) \end{itemize}

Implementarea (datele din Cerința 3)

În cod folosim un *wrapper* care simulează un eșantion mare și întoarce (N, T, I).

```r
print(simuleaza_date_din_ex3)
```

```
## function (M = 1e+05, n_max = 3, p_succes = 0.7, t_0 = 400, medie_S = 150,
##     backoff_fix = 50)
## {
##     rezultate <- replicate(M, simuleaza_o_cerere(n_max = n_max,
##         p_succes = p_succes, t_0 = t_0, medie_S = medie_S, backoff_fix = backoff_fix))
##     df <- as.data.frame(t(rezultate))
```

```
##     df[] <- lapply(df, unlist)
##     df$N <- as.integer(df$N)
##     df$I <- as.integer(df$I)
##     df$T <- as.numeric(df$T)
##     df[, c("N", "T", "I")]
## }
```

a) Estimarea probabilităților condiționate

Estimăm empiric: - \(P(A \mid N \le n_0)\) (succes condiționat de "rapid") - \(P(B \mid A)\) (SLA condiționat de faptul că am avut succes final)

```
M <- 100000
n_max <- 3
p_succes <- 0.7
t_0 <- 400
medie_S <- 150
backoff_fix <- 50

n_0 <- 2

df <- simuleaza_date_din_ex3(
  M = M, n_max = n_max, p_succes = p_succes,
  t_0 = t_0, medie_S = medie_S, backoff_fix = backoff_fix
)

A <- (df$I == 1L)
B <- (df$T <= t_0)
C <- (df$N <= n_0)

P_A_given_C <- mean(A[C]) # P(A|C)
P_B_given_A <- mean(B[A]) # P(B|A)

rez_cond <- data.frame(
  Marime = c(paste0("P(A | N <= ", n_0, ")"), "P(B | A)"),
  Valoare = c(P_A_given_C, P_B_given_A)
)

knitr::kable(rez_cond, digits = 4, caption = "Probabilități condiționate estimate empiric")
```

Table 7: Probabilități condiționate estimate empiric

Marime	Valoare
P(A | N <= 2)	1.0000
P(B | A)	0.8387

b) Speranțe condiționate (timpul mediu condiționat de succes/eșec)

Calculăm: - \(E(T \mid I=1)\): timpul mediu pentru cererile care reușesc - \(E(T \mid I=0)\): timpul mediu pentru cererile care eșuează final

```r
E_T_given_success <- mean(df$T[df$I == 1L])
E_T_given_fail <- mean(df$T[df$I == 0L])

rez_ET <- data.frame(
  Marime = c("E(T | I = 1)", "E(T | I = 0)"),
  Valoare = c(E_T_given_success, E_T_given_fail)
)

knitr::kable(rez_ET, digits = 4, caption = "Speranțe condiționate estimate empiric")
```

Table 8: Speranțe condiționate estimate empiric

Marime	Valoare
E(T | I = 1)	218.4106
E(T | I = 0)	552.8955

c) Interpretare (experiența utilizatorului)

- **P(A | N n0)** arată cât de des reușim "repede" (cu puține retry-uri). Un sistem bun are o probabilitate cât mai mare aici, ceea ce indică stabilitate și erori rare.
- **P(B | A)** arată cât de des respectăm SLA condiționat de faptul că cererea chiar reușește. Dacă această probabilitate este mică, utilizatorii "au succes", dar "așteaptă prea mult".
- În mod tipic, **E(T | I=0)** este mai mare decât **E(T | I=1)** deoarece eșecul final consumă toate încercările (și, implicit, toți timpii de procesare + backoff), ceea ce înseamnă latență mare fără rezultat — cea mai proastă experiență pentru utilizator.

8. Independenta vs dependenta (Cerința 7)

În modelele anterioare, am presupus că timpul de răspuns pentru o încercare este independent de istoric. În realitate, un eșec poate indica o problemă sistemică (ex: supraîncărcare, congestie de rețea), ceea ce crește probabilitatea ca următoarea încercare să dureze mai mult sau să eșueze din nou. Aceasta creează o **dependență** între eșecuri și latența viitoare.

Implementarea funcției cu dependență

Vom modifica funcția de simulare a unei cereri pentru a include un parametru `factor_latenta`. Mecanismul este următorul: * Dacă o încercare eșuează, timpul mediu de procesare (`medie_S`) pentru următoarea încercare este multiplicat cu acest factor. * Dacă `factor_latenta > 1.0`, sistemul devine "mai lent" pe măsură ce acumulăm eșecuri (degradare progresivă).

```r
#' Simuleaza o cerere cu posibilitatea dependentei intre esecuri si latenta
#'
#' @param n_max Numarul maxim de incercari
#' @param p_succes Probabilitatea de succes a unei incercari
#' @param medie_S_initial Timpul mediu de procesare initial (fara penalizari)
#' @param backoff_fix Timpul de asteptare intre incercari
#' @param factor_latenta Factorul cu care creste media timpului de raspuns dupa un esec.
#'                          Daca este 1.0, timpii sunt independenti.
```

```r
#'                          Daca este > 1.0, sistemul devine mai lent dupa esecuri.
#'
#' @return Timpul total T
simuleaza_cerere_dependenta <- function(n_max = 3, p_succes = 0.7, medie_S_initial = 150,
                                        backoff_fix = 50, factor_latenta = 1.0) {
  timp_total <- 0
  medie_curenta <- medie_S_initial
  succes <- FALSE

  for (i in 1:n_max) {
    # Generam timpul de raspuns S_i
    # Daca factor_latenta > 1 si am avut esecuri anterioare, medie_curenta va fi mai mare
    s_i <- rexp(1, rate = 1 / medie_curenta)
    timp_total <- timp_total + s_i

    # Verificam succesul
    if (runif(1) < p_succes) {
      succes <- TRUE
      break # Daca am reusit, iesim
    } else {
      # Daca am esuat
      # Adaugam backoff daca mai avem incercari
      if (i < n_max) {
        timp_total <- timp_total + backoff_fix

        # Aplicam dependenta: latenta creste pentru urmatoarea incercare
        medie_curenta <- medie_curenta * factor_latenta
      }
    }
  }

  return(timp_total)
}
```

a) Simulare: independent vs dependent

Vom simula două scenarii pentru a analiza impactul dependenței asupra timpului total de așteptare (T):

1. **Scenariul independent:** `factor_latenta = 1.0`. Eșecurile nu afectează performanța serverului pentru reîncercări.
2. **Scenariul dependent:** `factor_latenta = 2.0`. După fiecare eșec, timpul mediu de răspuns se dublează (ex: 150ms -> 300ms -> 600ms).

Parametrii comuni: * $M = 10000$ simulări * $n_{max} = 3$ încercări * $p_{succes} = 0.7$

```r
M <- 10000

# 1. Simulare timpi independenti
timp_independent <- replicate(M, simuleaza_cerere_dependenta(
  n_max = 3, p_succes = 0.7, medie_S_initial = 150,
  backoff_fix = 50, factor_latenta = 1.0
))
```

```r
# 2. Simulare timpi dependenti (latenta se dubleaza la esec)
timp_dependent <- replicate(M, simuleaza_cerere_dependenta(
  n_max = 3, p_succes = 0.7, medie_S_initial = 150,
  backoff_fix = 50, factor_latenta = 2.0
))
```

Analiza statistică

Comparăm media și varianța celor două distribuții. Ne așteptăm ca în cazul dependent, varianța să crească semnificativ deoarece cazurile de eșec repetat vor genera timpi foarte lungi (outlieri extremi).

```r
var_indep <- var(timp_independent)
var_dep <- var(timp_dependent)
mean_indep <- mean(timp_independent)
mean_dep <- mean(timp_dependent)

df_rezultate <- data.frame(
  Scenariu = c("Independent (Factor=1.0)", "Dependent (Factor=2.0)"),
  Media_T = c(mean_indep, mean_dep),
  Varianta_T = c(var_indep, var_dep)
)

knitr::kable(df_rezultate, digits = 2, caption = "Comparatie Statistici T")
```

Table 9: Comparatie Statistici T

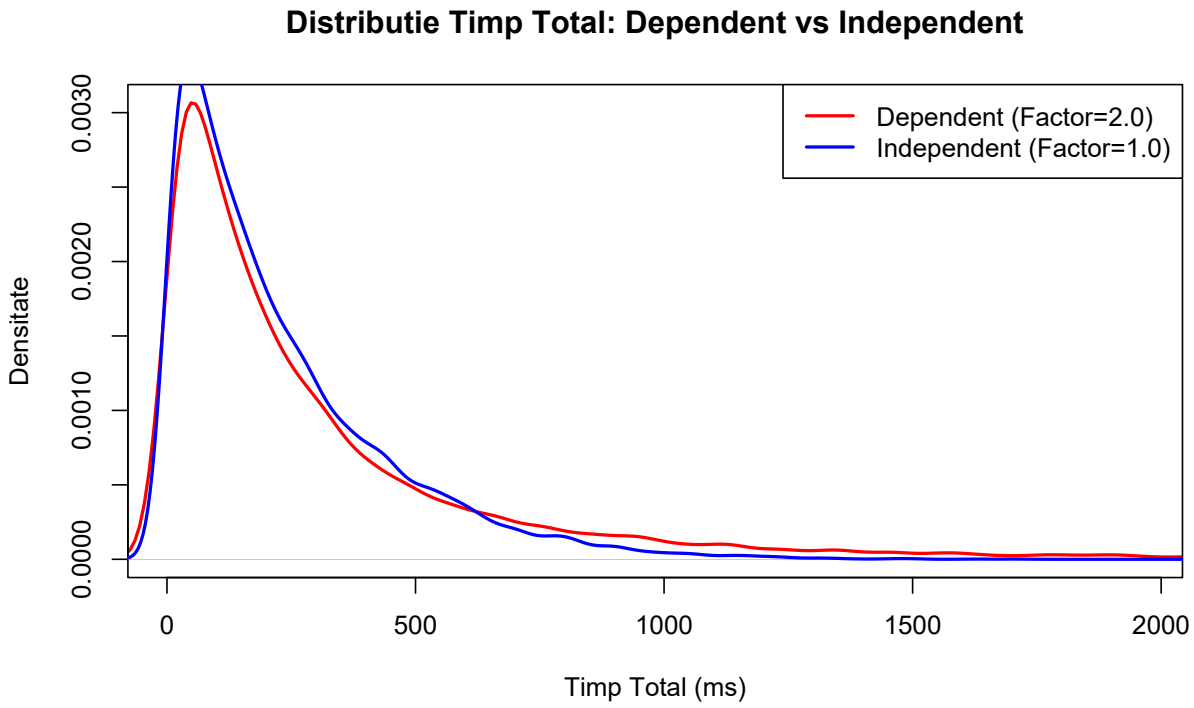
Scenariu	Media_T	Varianta_T
Independent (Factor=1.0)	226.55	47465.78
Dependent (Factor=2.0)	311.48	160566.38

b) Comparație: independent vs dependent

Vom suprapune densitățile celor două distribuții pentru a evidenția diferențele.

```r
# Setam limita graficului pentru a cuprinde si cozile lungi ale distributiei dependente
x_max <- quantile(timp_dependent, 0.99)

# Afisam cazul dependent primul (dispersie mai mare)
plot(density(timp_dependent),
  col = "red", lwd = 2,
  xlim = c(0, x_max),
  main = "Distributie Timp Total: Dependent vs Independent",
  xlab = "Timp Total (ms)",
  ylab = "Densitate"
)

# Adaugam cazul independent
lines(density(timp_independent), col = "blue", lwd = 2)

legend("topright",
```

```
  legend = c("Dependent (Factor=2.0)", "Independent (Factor=1.0)"),
  col = c("red", "blue"),
  lwd = 2
)
```



Interpretare:

- **Linia albastră (independent):** Este mai concentrată în stânga, indicând timpi mai mici și mai predictibili.
- **Linia roșie (dependent):** Este mai aplatizată și se întinde mai mult spre dreapta. Aceasta cauzează o variantă mult mai mare și crește riscul ca utilizatorii să experimenteze timpi de așteptare foarte mari în momentele critice ale sistemului.

c) Concluzii privind riscul și stabilitatea sistemului

Analiza comparativă demonstrează că dependența între eșecuri și latență introduce riscuri majore pentru stabilitatea sistemului:

1. **Efectul de "bulgăre de zăpadă":**
   Într-un sistem dependent (`factor_latenta > 1`), o degradare temporară a performanței poate declanșa o buclă în care timpul de răspuns crește la nesfârșit. Un eșec inițial crește probabilitatea ca următoarele cereri să dureze mai mult, ceea ce poate duce la timeout-uri și mai multe eșecuri, amplificând latența până la blocarea completă a resurselor.
2. **Impredictibilitate și SLA (Service Level Agreements):**
   Deși media timpului de așteptare crește, parametrul critic este **varianța** (sau deviația standard).

Coada lungă a distribuției în cazul dependent înseamnă că un procent mic, dar semnificativ de utilizatori, vor experimenta latențe extreme. Acest lucru face imposibilă garantarea unui SLA strict ("99% din cereri se termină sub 500ms").

3. **Recomandări:**
   Pentru a menține sistemul stabil, este esențial să anulăm dependența. Acest lucru se poate realiza prin mecanisme precum:

   - **Circuit breakers:** Oprirea temporară a cererilor către un serviciu care răspunde greu.
   - **Jitter:** Adăugarea unei componente aleatoare în timpul de backoff pentru a evita sincronizarea reîncercărilor.
   - **Load balancing:** Redistribuirea traficului pentru a evita supraîncărcarea unui singur nod deja lent.

9. Inegalități probabilistice (garanții worst-case) (Cerința 8)

Acest capitol se concentrează pe validarea și utilizarea inegalităților probabilistice clasice pentru a analiza comportamentul sistemului de retry. Acestea oferă limite superioare ("worst-case") pentru probabilitățile de risc.

Pentru simulare, folosim un set de date generat cu următorii parametri:

```
#' Simuleaza procesul si returneaza timpul si nr de esecuri
print(simuleaza_avansat)
```

```
## function (n_max = 3, p_succes = 0.7, medie_S_initial = 150, backoff_fix = 50,
##     factor_latenta = 1)
## {
##     timp_total <- 0
##     medie_curenta <- medie_S_initial
##     nr_esecuri <- 0
##     for (i in 1:n_max) {
##         s_i <- rexp(1, rate = 1/medie_curenta)
##         timp_total <- timp_total + s_i
##         if (runif(1) < p_succes) {
##             break
##         }
##         else {
##             nr_esecuri <- nr_esecuri + 1
##             if (i < n_max) {
##                 timp_total <- timp_total + backoff_fix
##                 medie_curenta <- medie_curenta * factor_latenta
##             }
##         }
##     }
##     return(list(timp = timp_total, esecuri = nr_esecuri))
## }
```

```
M <- 10000
n_max <- 5
p <- 0.4
mu <- 100
bf <- 20
```

```r
fact <- 1.5

# Generam datele
rezultate <- replicate(M, simuleaza_avansat(n_max, p, mu, bf, fact), simplify = FALSE)
T_vals <- sapply(rezultate, function(x) x$timp)
Esec_vals <- sapply(rezultate, function(x) x$esecuri)

# Valori teoretice (calculate analitic pentru comparatie)
teoretic <- calculeaza_teoretic_T(n_max, p, mu, bf, fact)
E_T <- teoretic$media
Var_T <- teoretic$varianta
```

a) Verificați numeric inegalitățile Markov și Cebîșev (empiric versus teoretic)

Inegalitatea lui Markov

Pentru o variabilă aleatoare nenegativă T și $a > 0$:

$$P(T \geq a) \leq \frac{E[T]}{a}$$

Alegem $a = 2 \cdot E[T]$. Limita teoretică este 0.5.

Inegalitatea lui Cebîșev

Pentru orice variabilă cu medie și variață finite:

$$P(|T - E[T]| \geq k\sigma) \leq \frac{1}{k^2}$$

Alegem $k = 2$. Limita teoretică este $1/4 = 0.25$.

```r
# Markov (a = 2 * E[T])
prob_markov_empiric <- mean(T_vals >= 2 * E_T)
limita_markov <- 0.5

# Cebisev (k = 2)
sigma_T <- sqrt(Var_T)
prob_cebisev_empiric <- mean(abs(T_vals - E_T) >= 2 * sigma_T)
limita_cebisev <- 0.25

df_ineg <- data.frame(
  Inegalitate = c("Markov (a=2E[T])", "Cebisev (k=2)"),
  Prob_Empirica = c(prob_markov_empiric, prob_cebisev_empiric),
  Limita_Teoretica = c(limita_markov, limita_cebisev),
  Verificare = c(prob_markov_empiric <= limita_markov, prob_cebisev_empiric <= limita_cebisev)
)
knitr::kable(df_ineg, caption = "Validare Markov și Cebîșev")
```

Table 10: Validare Markov și Cebîșev

Inegalitate	Prob_Empirica	Limita_Teoretica	Verificare
Markov (a=2E[T])	0.1562	0.50	TRUE
Cebisev (k=2)	0.0586	0.25	TRUE

b) Pentru variabila număr de eșecuri/încercări verificați o inegalitate de tip Chernoff

Inegalitatea lui Chernoff oferă limite exponențiale. Pentru variabila X (număr eșecuri), limita este dată de funcția generatoare de momente $M_X(t) = E[e^{tX}]$.

$$P(X \geq a) \leq \frac{E[e^{tX}]}{e^{ta}}, \quad \forall t > 0$$

Vom verifica această inegalitate pentru un prag de eșecuri $a = 3$ și un parametru arbitrar $t = 0.5$.

```
a <- 3
t_val <- 0.5

# Calculam empiric MGF: E[e^(t*X)]
mgf_empiric <- mean(exp(t_val * Esec_vals))

# Calculam limita Chernoff
limita_chernoff <- mgf_empiric / exp(t_val * a)

# Probabilitatea reala din date
prob_reala_esecuri <- mean(Esec_vals >= a)

cat("Probabilitate Reala (P(X >= 3)): ", prob_reala_esecuri, "\n")
```

```
## Probabilitate Reala (P(X >= 3)):  0.2129
```

```
cat("Limita Chernoff Calculate:       ", limita_chernoff, "\n")
```

```
## Limita Chernoff Calculate:        0.646851
```

```
cat("Verificare (Reala <= Limita):    ", prob_reala_esecuri <= limita_chernoff, "\n")
```

```
## Verificare (Reala <= Limita):     TRUE
```

c) Interpretați utilitatea acestor limite când distribuțiile exacte sunt necunoscute

În practică, rareori cunoaștem distribuția exactă a timpilor de răspuns sau a erorilor unui sistem complex. Cu toate acestea:

1. **Dacă știm doar media** (ușor de măsurat), inegalitatea lui **Markov** ne oferă o garanție absolută ("worst-case") că nu vom depăși un anumit prag de latență prea des.

2. **Dacă știm și varianța**, **Cebîșev** rafinează această limită, spunându-ne cât de "concentrate" sunt valorile în jurul mediei.
3. **Dacă presupunem independența erorilor**, **Chernoff** ne dă limite extrem de strânse (exponențiale) pentru, de exemplu, riscul ca 50% din servere să pice simultan.

Aceste limite permit inginerilor să stabilească SLA-uri (Service Level Agreements) sigure, chiar și în absența unor date istorice detaliate.

d) Pentru o funcție convexă φ (ex.: x^2, e^x) verificați numeric $\varphi(E(T)) \leq E(\varphi(T))$ (inegalitatea lui Jensen)

Această inegalitate afirmă că media funcției este mai mare sau egală cu funcția aplicată mediei, pentru funcții convexe. Alegem funcția de cost convexă $\varphi(t) = t^2$, care penalizează timpii lungi disproporționat de mult.

```
phi <- function(t) t^2

# Partea stanga a inegalitatii: phi( E[T] )
lhs <- phi(mean(T_vals))

# Partea dreapta a inegalitatii: E[ phi(T) ]
rhs <- mean(phi(T_vals))

cat("phi(E[T]) = (Media)^2: ", lhs, "\n")
```

```
## phi(E[T]) = (Media)^2:  189845.7
```

```
cat("E[phi(T)] = Media patratelor: ", rhs, "\n")
```

```
## E[phi(T)] = Media patratelor:  481667.5
```

```
cat("Diferenta (rhs - lhs): ", rhs - lhs, "\n")
```

```
## Diferenta (rhs - lhs):  291821.7
```

```
cat("Verificare Jensen (lhs <= rhs): ", lhs <= rhs, "\n")
```

```
## Verificare Jensen (lhs <= rhs):  TRUE
```

e) Interpretați rezultatul de la d) în contextul riscului (penalizarea valorilor extreme)

Rezultatul $E[\varphi(T)] > \varphi(E[T])$ are o semnificație economică critică:

Dacă evaluăm performanța sistemului doar pe baza **"timpului mediu de răspuns"** ($E[T]$) și aplicăm costul la această medie ($\varphi(E[T])$), vom **subestima sistematic** costurile reale.

Deoarece funcția de cost este convexă (ex: pierderile financiare cresc exponențial cu întârzierea - clienții devin mult mai frustrați la 10 secunde decât la 5 secunde), valorile extreme (outlierii) contribuie la costul mediu mult mai mult decât sugerează simpla medie a timpilor.

Concluzie: În analiza de risc, a ne baza pe "cazul mediu" este o greșeală. Trebuie să calculăm "media costurilor", nu "costul mediei", pentru a lua în considerare impactul evenimentelor rare dar catastrofale.

10. Aproximare normala si agregare (Cerința 9)

În problemele de trafic real, timpul total de procesare pe o perioadă lungă (o zi) este suma mai multor evenimente individuale (cereri). Deși timpii individuali pot avea distribuții exponențiale (asimetrice), suma lor tinde să devină normală (Gaussiană).

a) Funcția de simulare

Următoarea funcție simulează activitatea dintr-o zi. Observați că numărul de termeni din sumă (`k_cereri`) este el însuși o variabilă aleatoare (Poisson), ceea ce face ca suma să fie una compusă.

```
simuleaza_suma_totala <- function(lambda_trafic = 1000, medie_latenta = 150) {
  # Generam numarul de cereri pentru o zi (Poisson)
  k_cereri <- rpois(1, lambda = lambda_trafic)

  suma_zi <- 0

  if (k_cereri > 0) {
    # Generam latentele individuale (S_i)
    # Folosim distributia exponentiala
    latente_individuale <- rexp(k_cereri, rate = 1 / medie_latenta)

    # Suma totala
    suma_zi <- sum(latente_individuale)
  }

  return(suma_zi)
}
```

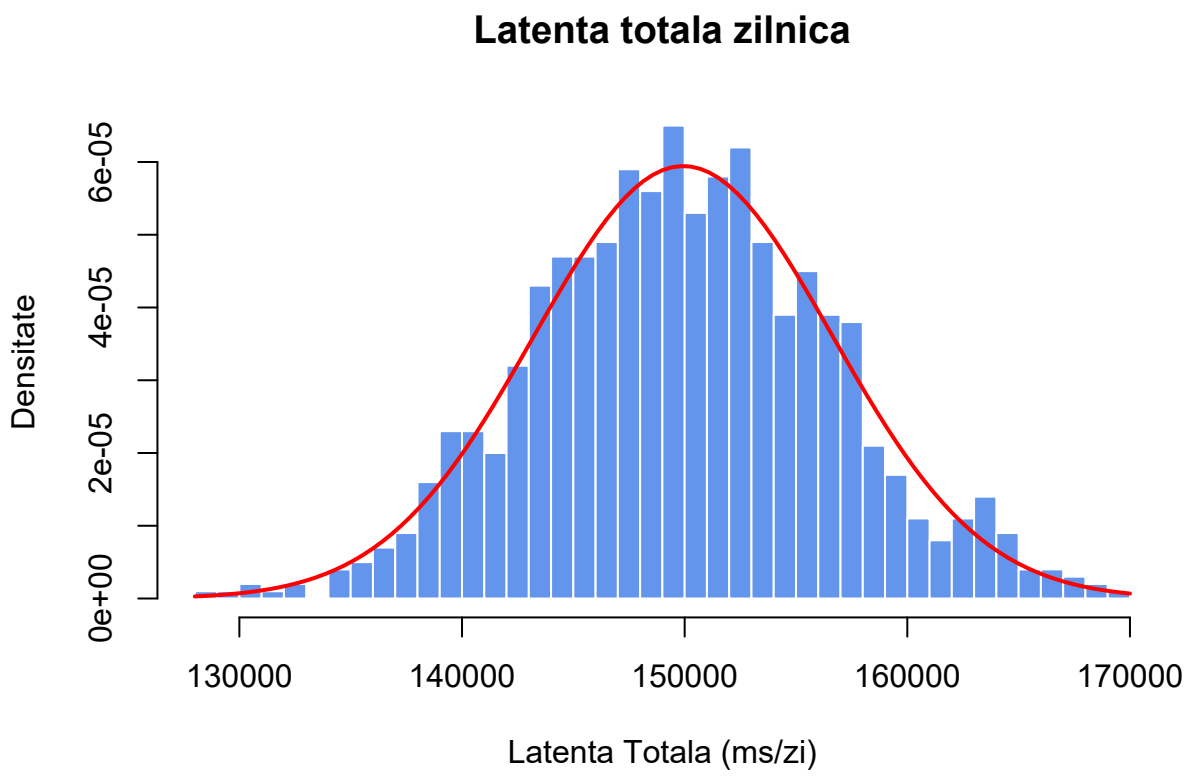
b) Comparația histogramei agregatului cu o normală ajustată

Mai jos este codul care rulează simularea și verifică aproximarea normală pe parcursul unei zile.

```
if (TRUE) {
  M <- 1000 # nr zile simulate

  # Simulare
  latenta_totala_zi <- replicate(M, simuleaza_suma_totala(
    lambda_trafic = 1000,
    medie_latenta = 150
  ))

  # Calculam parametrii empirici pentru ajustarea distributiei normale
  media_agregat <- mean(latenta_totala_zi)
  sd_agregat <- sd(latenta_totala_zi)

  # Afisare rezultate
  cat("Media Empirica a sumei:    ", round(media_agregat, 2), "\n")
  cat("Deviatia Standard Empirica:", round(sd_agregat, 2), "\n")

  # Vizualizare și comparație
```

```r
  par(mfrow = c(1, 1))

  # Histograma datelor agregate
  hist(latenta_totala_zi,
    probability = TRUE,
    breaks = 30,
    col = "cornflowerblue",
    border = "white",
    main = "Latenta totala zilnica",
    xlab = "Latenta Totala (ms/zi)",
    ylab = "Densitate"
  )

  # Adaugam curba teoretica normala (linia rosie)
  curve(dnorm(x, mean = media_agregat, sd = sd_agregat),
    add = TRUE,
    col = "red",
    lwd = 2
  )
}
```

```
## Media Empirica a sumei:     149935.1
## Deviatia Standard Empirica: 6711.19
```



Explicație teoretică

Graficul de mai sus demonstrează **Teorema Limită Centrală**. Chiar dacă latențele individuale sunt exponențiale (asimetrice), suma a ~1000 de astfel de latențe se așează pe o curbă în formă de clopot (normală).

Linia roșie reprezintă distribuția Normală teoretică parametrizată cu media și deviația standard empirică a datelor. Suprapunerea perfectă confirmă că putem folosi aproximarea normală pentru a face predicții despre traficul total al sistemului.

11. Churn (Cerința 10)

Un aspect critic în modelarea sistemelor reale este comportamentul utilizatorilor, în special decizia de a părăsi platforma ("Churn"). În acest model, considerăm două cauze distincte pentru churn:

1. **Churn aleator (natural):** Utilizatorul pleacă din motive externe, independente de performanța sistemului (ex: nu mai are nevoie de serviciu).
2. **Churn condiționat (tehnic):** Utilizatorul pleacă din cauza frustrării acumulate în urma erorilor tehnice repetate.

a) Funcțiile de simulare (churn aleator și condiționat)

Churn aleator

Această funcție modelează plecarea ca un proces Bernoulli simplu cu probabilitatea q.

```
simuleaza_churn_aleator <- function(q = 0.05) {
  return(runif(1) < q)
}
```

Churn condiționat

Utilizatorul monitorizează o fereastră de m interacțiuni recente. Dacă numărul de eșecuri depășește un prag k, utilizatorul părăsește platforma. Numărul de eșecuri este modelat Binomial ($Bin(m, 1 - p_{succes})$).

```
simuleaza_churn_conditionat <- function(m = 20, k = 5, p_succes = 0.9) {
  # Numarul de esecuri intr-o secventa de lungime m (dist Binomiala)
  nr_esecuri <- rbinom(1, size = m, prob = 1 - p_succes)

  # Verificam daca s-a depasit pragul de k esecuri
  if (nr_esecuri >= k) {
    return(TRUE)
  } else {
    return(FALSE)
  }
}
```

b) Estimarea ratei totale de churn

Rata totală de churn este reuniunea celor două evenimente (Aleator SAU Condiționat). Deoarece un utilizator poate pleca din ambele motive simultan, probabilitatea totală nu este simpla sumă, ci:

$$P(Total) = P(Aleator \cup Conditionat) = P(Aleator) + P(Conditionat) - P(Aleator \cap Conditionat)$$

Execuția de mai jos simulează comportamentul pentru $M = 100.000$ de utilizatori pentru a estima empiric aceste probabilități.

```
if (TRUE) {
  set.seed(123)

  M <- 100000 # Numarul de utilizatori

  # REZOLVARE a)

  # Simularea vectorului pentru Churn aleator
  vec_aleator <- replicate(M, simuleaza_churn_aleator(q = 0.05))

  # Simularea vectorului pentru Churn conditionat
  vec_conditionat <- replicate(M, simuleaza_churn_conditionat(
    m = 20, k = 4,
    p_succes = 0.75
  ))

  # REZOLVARE b)

  # Calculul churn-ului total
  vec_total <- (vec_aleator | vec_conditionat)

  # Estimari
  prob_aleator <- mean(vec_aleator)
  prob_conditionat <- mean(vec_conditionat)
  prob_total <- mean(vec_total)

  # Afisare rezultate
  cat("Analiza Churn cu urmatorii parametri:\n")
  cat("m =", 20, ", k =", 5, ", p_succes =", 0.75, "\n")

  cat("\nProbabilitati estimate (M =", M, "):\n")
  cat("1. P(Churn aleator):     ", round(prob_aleator, 4), "\n")
  cat("2. P(Churn conditionat): ", round(prob_conditionat, 4), "\n")
  cat("3. P(Churn TOTAL):       ", round(prob_total, 4), "\n")
}
```

```
## Analiza Churn cu urmatorii parametri:
## m = 20 , k = 5 , p_succes = 0.75
##
## Probabilitati estimate (M = 1e+05 ):
## 1. P(Churn aleator):      0.0494
## 2. P(Churn conditionat):  0.7749
## 3. P(Churn TOTAL):        0.786
```

Aceste rezultate ne permit să înțelegem cât din pierderea clienților este controlabilă (tehnică) și cât este inevitabilă (naturală).

c) Comparație scenarii și interpretare

Vom analiza impactul parametrilor asupra ratei totale de Churn comparând trei scenarii:

1. **Scenariul de bază:** `p_succes = 0.75`, `k = 5` (utilizatori moderați).
2. **Siguranță îmbunătățită:** `p_succes = 0.90`, `k = 5`.
3. **Utilizatori exigenți:** `p_succes = 0.75`, `k = 3`. Utilizatorii au toleranță scăzută la erori.

```
calculeaza_churn_mediu <- function(M, m, k, p_succes, q) {
  vec_aleator <- replicate(M, simuleaza_churn_aleator(q))
  vec_conditionat <- replicate(M, simuleaza_churn_conditionat(m, k, p_succes))
  return(mean(vec_aleator | vec_conditionat))
}

M <- 50000
m <- 20
q <- 0.05

churn_normal <- calculeaza_churn_mediu(M, m, k = 5, p_succes = 0.75, q)

churn_sistem <- calculeaza_churn_mediu(M, m, k = 5, p_succes = 0.90, q)

churn_exigent <- calculeaza_churn_mediu(M, m, k = 3, p_succes = 0.75, q)

df_comp <- data.frame(
  Scenariu = c("Normal", "Siguranta sistem + (p=0.9)", "Exigenta + (k=3)"),
  Churn_Total = c(churn_normal, churn_sistem, churn_exigent)
)

knitr::kable(df_comp, digits = 4, caption = "Impactul parametrilor asupra Churn-ului")
```

Table 11: Impactul parametrilor asupra Churn-ului

Scenariu	Churn_Total
Normal	0.6003
Siguranta sistem + (p=0.9)	0.0905
Exigenta + (k=3)	0.9125

Interpretare:

- **Siguranța sistemului (`p_succes`):** Creșterea siguranței sitemului reduce drastic churn-ul tehnic. Aceasta este o variabilă ce nu depinde de client, ci de sistemul pe care îl administrăm.
- **Toleranța utilizatorilor (`k`):** Dacă utilizatorii devin mai exigenți (scade pragul `k`), churn-ul crește chiar dacă sistemul rămâne la fel.
- **Limita inferioară:** Indiferent cât de perfect este sistemul tehnic, rata de churn nu poate scădea sub $P(Aleator) = q$.

12. Impact Economic (Cerința 11)

Scopul final al analizei este cuantificarea performanței tehnice în termeni financiari.

a) Definirea variabilei aleatoare pentru profitul zilnic

Definim variabila aleatoare $Profit_{zi}$ ca fiind bilanțul financiar net la finalul unei zile de activitate. Aceasta se exprimă ca o funcție liniară de alte variabile aleatoare simulate anterior:

$$Profit_{zi} = V_{succes} - C_{churn} - C_{SLA}$$

Unde componentele sunt definite astfel:

1. **Venituri (V_{succes}):**
$$V_{succes} = N_{succes} \cdot c_{succes}$$
   - N_{succes}: Variabilă aleatoare reprezentând numărul total de cereri soluționate cu succes în ziua respectivă ($I = 1$).
   - c_{succes}: Câștigul unitar per tranzacție reușită (ex: 2.0 EUR).
2. **Pierderi din Churn (C_{churn}):**
$$C_{churn} = N_{churn} \cdot c_{pierdere}$$
   - N_{churn}: Variabilă aleatoare reprezentând numărul de utilizatori care părăsesc platforma în ziua respectivă (fie aleator, fie condiționat de erori).
   - $c_{pierdere}$: Costul asociat pierderii unui client (LTV - Lifetime Value, ex: 20 EUR).
3. **Penalități SLA (C_{SLA}):**
$$C_{SLA} = N_{SLA} \cdot c_{penalizare}$$
   - N_{SLA}: Variabilă aleatoare reprezentând numărul de cereri care au depășit timpul limită contractual $T > t_{SLA}$ (ex: 800ms).
   - $c_{penalizare}$: Costul penalizării per încălcare a SLA (ex: 2.0 EUR).

Deoarece N_{succes}, N_{churn} și N_{SLA} sunt variabile aleatoare (depind de distribuțiile de trafic, latență și mecanismele de retry), rezultă că și $Profit_{zi}$ este o variabilă aleatoare cu propria sa medie și varianță.

Pentru implementarea acestei variabile aleatoare in R, vom folosi atat de functia de simularea unei cereri de la exercitiul 3, cat si de functia de simulare a churn-ului de la exercitiul 10.

Mai intai vom defini urmatoarea functie care simuleaza un lot de cereri:

```
#' Aceasta functie apeleaza repetat functia simuleaza_o_cerere pentru a genera
#' un esantion statistic de dimensiune n_simulari. Rezultatul este un tabel
#' care permite analiza distributiei timpului total (T) si a ratei de succes.
#'
#' @param n_simulari Numarul total de cereri generate (ex: 1000, 10000).
#' @param n_max Numarul maxim de incercari permise pentru o cerere (inclusiv prima incercare).
#' @param p_succes Probabilitatea (0-1) ca o incercare individuala sa reuseasca.
#' @param t_0 Timeout-ul initial (timpul limita de asteptare pentru primul raspuns).
#' @param medie_S Latenta medie a serverului (pentru distributia exponentiala a timpului de procesare).
#' @param backoff_fix Timpul fix adaugat la timeout (penalizare) in caz de esec (Backoff).
#'
#' @return Un `data.frame` cu `n_simulari` randuri si urmatoarele coloane:
#' \itemize{
#'   \item I: Indicator de succes final (1 daca a reusit, 0 altfel)
#'   \item T: Timpul total pana la succes sau abandon (suma S_i si B_i)
#'   \item N: Numarul total de incercari efectuate
#'   \item D: Indicator logic (TRUE daca a existat cel putin un esec pe parcurs)
#' }
```

```r
simuleaza_lot_cereri <- function(n_simulari,
                                 n_max = 3,
                                 p_succes = 0.8,
                                 t_0 = 500,
                                 medie_S = 150,
                                 backoff_fix = 50) {
  # Folosim replicate pentru a rula simularea de n ori
  rezultate <- replicate(n_simulari, simuleaza_o_cerere(
    n_max = n_max,
    p_succes = p_succes,
    t_0 = t_0,
    medie_S = medie_S,
    backoff_fix = backoff_fix
  ))

  # Transpunem matricea rezultat (sa avem cererile pe randuri) si o facem data frame
  df <- as.data.frame(t(rezultate))

  # Curatam listele (replicate poate returna liste, noi vrem vectori numerici)
  df[] <- lapply(df, unlist)

  return(df)
}
```

Vom defini variabila aleatoare astfel:

```r
#' Calculeaza bilantul economic pentru o zi de activitate
#' @param nr_clienti Numarul de clienti din acea zi (Kd)
#' @param castig_per_succes Venitul generat de o cerere reusita
#' @param cost_churn Pierderea generata de un client care pleaca
#' @param limita_sla Timpul maxim acceptat (T_max) inainte de penalizare
#' @param penalizare_sla Costul daca se depaseste timpul limita
#' @param rata_churn_aleator Probabilitatea ca un utilizator sa paraseasca platforma aleator
#' @param dim_churn_cond Dimensiunea ferestrei (numarul de cereri recente)
#' @param prag_erori_churn_cond Pragul de erori (minim k esecuri declanseaza plecarea)
#' @param prob_succes_churn_cond Probabilitatea de succes a unei singure cereri
#' @return Un vector numit cu indicatorii economici: Profit, Venit, Costuri si Numarul de utilizatori p
simuleaza_profit_zi <- function(nr_clienti,
                                castig_per_succes = 0.5,
                                cost_churn = 50,
                                limita_sla = 800,
                                penalizare_sla = 2.0,
                                rata_churn_aleator = 0.05,
                                dim_churn_cond = 20,
                                prag_erori_churn_cond = 5,
                                prob_succes_churn_cond = 0.9) {
  rezultate_zi <- simuleaza_lot_cereri(nr_clienti)

  nr_succese <- sum(rezultate_zi$I == 1)

  nr_sla_fail <- sum(rezultate_zi$T > limita_sla)
```

```
  # CHURN
  plecari_aleatoare <- replicate(nr_clienti, simuleaza_churn_aleator(q = rata_churn_aleator))

  plecari_conditionate <- replicate(nr_clienti, simuleaza_churn_conditionat(
    m = dim_churn_cond,
    k = prag_erori_churn_cond,
    p_succes = prob_succes_churn_cond
  ))
  plecari_totale <- plecari_aleatoare | plecari_conditionate

  nr_churn <- sum(plecari_totale)


  # PARTEA FINANCIARA
  venit_total <- nr_succese * castig_per_succes
  cost_penalizari <- nr_sla_fail * penalizare_sla
  pierdere_churn <- nr_churn * cost_churn

  profit_net <- venit_total - pierdere_churn - cost_penalizari

  return(c(
    Profit = profit_net,
    Venit = venit_total,
    Cost_SLA = cost_penalizari,
    Cost_Churn = pierdere_churn,
    Nr_Churn = nr_churn
  ))
}
```

b) Estimarea statistică a profitului (Medie, Varianță, Interval de Încredere)

Deoarece profitul zilnic este o variabilă aleatoare, o singură simulare nu este relevantă pentru deciziile de business. Pentru a obține o estimare robustă, folosim metoda **Monte Carlo** simulând activitatea pe o perioadă extinsă (un an calendaristic).

Metodologia de Estimare

1. **Simulare Repetată:** Generăm N de zile independente. Pentru fiecare zi:
   - Se generează traficul K_d (conform distribuției Poisson/Binomiale din cap. 1).
   - Se simulează comportamentul fiecărui client și răspunsul sistemului.
   - Se calculează profitul net al zilei respective.
2. **Agregare:** Obținem un vector de profituri $P = [p_1, p_2, ..., p_N]$.
3. **Calcul Statistic:**
   - **Media Empirică ($\bar{P}$):** Speranța matematică a profitului.
   - **Varianța (s^2):** Măsura riscului/volatilității afacerii.
   - **Intervalul de Încredere (95%):** Calculat pe baza Teoremei Limită Centrală (pentru n mare):

$$IC_{95\%} = \left[\bar{P} - 1.96\frac{s}{\sqrt{n}}, \quad \bar{P} + 1.96\frac{s}{\sqrt{n}} \right]$$

Vom folosi urmatoarea functie:

```r
#' Simuleaza un scenariu economic complet pe o perioada data
#' Aceasta functie impacheteaza tot procesul: Trafic -> Simulare Zi -> Statistici
#'
#' @param n_zile Durata simularii (zile)
#' @param model_trafic Tipul distributiei traficului: "poisson" sau "binomiala"
#' @param lambda_trafic_pois Media clientilor pe zi (pentru Poisson)
#' @param nmax_trafic_bin Capacitatea maxima teoretica (pentru Binomial)
#' @param p_trafic_bin Probabilitatea ca un potential client sa fie activ (pentru Binomial)
#'
#' @param castig_per_succes Venit per succes
#' @param cost_churn Pierdere per client plecat
#' @param cost_sla Penalizare SLA
#' @param rata_churn Rata de churn aleator
#'
#' @param rata_churn_aleator Probabilitatea ca un utilizator sa paraseasca platforma aleator
#' @param dim_churn_cond Dimensiunea ferestrei (numarul de cereri recente)
#' @param prag_erori_churn_cond Pragul de erori (minim k esecuri declanseaza plecarea)
#' @param prob_succes_churn_cond Probabilitatea de succes a unei singure cereri
#'
#' @return Vectorul cu statistici (Media, Varianta, IC)
simuleaza_scenariu_economic <- function(n_zile = 100,
                                        model_trafic = "poisson",
                                        lambda_trafic = 1000,
                                        nmax_trafic_bin = 2000,
                                        p_trafic_bin = 0.5,
                                        castig_per_succes = 0.6,
                                        cost_churn = 50,
                                        cost_sla = 2.0,
                                        rata_churn_aleator = 0.05,
                                        dim_churn_cond = 20,
                                        prag_erori_churn_cond = 5,
                                        prob_succes_churn_cond = 0.9) {
  # 1. Generam Traficul (folosim functia din Ex 1)
  trafic_zilnic <- simuleaza_trafic(n_zile,
    metoda = model_trafic,
    lambda = lambda_trafic,
    n_max = nmax_trafic_bin,
    p = p_trafic_bin
  )

  # 2. Alocam spatiu pentru rezultate
  profituri <- numeric(n_zile)

  # 3. Rulam simularea zi de zi
  for (i in 1:n_zile) {
    # Apelam functia de zi pe care am facut-o anterior
    rezultat_zi <- simuleaza_profit_zi(
      nr_clienti = trafic_zilnic[i],
      castig_per_succes = castig_per_succes,
      cost_churn = cost_churn,
      penalizare_sla = cost_sla,
      rata_churn_aleator = rata_churn_aleator,
      dim_churn_cond = dim_churn_cond,
```

```
      prag_erori_churn_cond = prag_erori_churn_cond,
      prob_succes_churn_cond = prob_succes_churn_cond
    )

    profituri[i] <- rezultat_zi["Profit"]
  }

  # 4. Returnam profiturile
  return(profituri)
}
```

Rezultatele Simulării

Rulăm scenariul pentru aproximativ 5 ani de activitate ($n = 5 * 365$ zile), considerând un trafic mediu $\lambda = 1000$ clienți/zi și urmatorii parametrii economici: castig_per_succes = 2 EUR cost_churn = 40 EUR cost_sla = 0.5 EUR

```
profituri <- simuleaza_scenariu_economic(
  n_zile = 365,
  model_trafic = "poisson",
  lambda_trafic = 1000,
  # Parametri economici
  castig_per_succes = 2.0,
  cost_churn = 40,
  cost_sla = 0.5,
  rata_churn_aleator = 0.001
)


head(profituri)
```

```
## [1]  813.0  235.5  575.0  -68.0  342.0 -247.0
```

Acum ne putem calcula media, varianta si intervalele de incredere folosind urmatoarea functie:

```
#' Calculeaza statistici descriptive si interval de incredere
#' @param vector_valori Un vector numeric (ex: profiturile zilnice)
#' @return Un vector cu Media, Varianta si limitele Intervalului de Incredere 95%
calculeaza_statistici_profit <- function(vector_valori) {
  # 1. Calcule de baza
  media <- mean(vector_valori)
  varianta <- var(vector_valori)
  dev_std <- sd(vector_valori)
  n <- length(vector_valori)

  # 2. Calcul Interval de Incredere 95%
  # Formula: Media +/- 1.96 * (SD / sqrt(n))
  eroare_std <- dev_std / sqrt(n)
  limita_inf <- media - 1.96 * eroare_std
  limita_sup <- media + 1.96 * eroare_std

  # 3. Returnam totul impachetat frumos
```

```
  return(c(
    Media = media,
    Varianta = varianta,
    IC_Min = limita_inf,
    IC_Max = limita_sup
  ))
}
```

Folosind valorile pentru profituri generate anterior, putem construi un tabel pentru a vedea media, varianta si intervalele de incredere:

```
statistici <- calculeaza_statistici_profit(profituri)

knitr::kable(t(statistici),
  digits = 2,
  caption = "Rezultate Economice"
)
```

Table 12: Rezultate Economice

Media	Varianta	IC_Min	IC_Max
213.83	68509.87	186.97	240.68

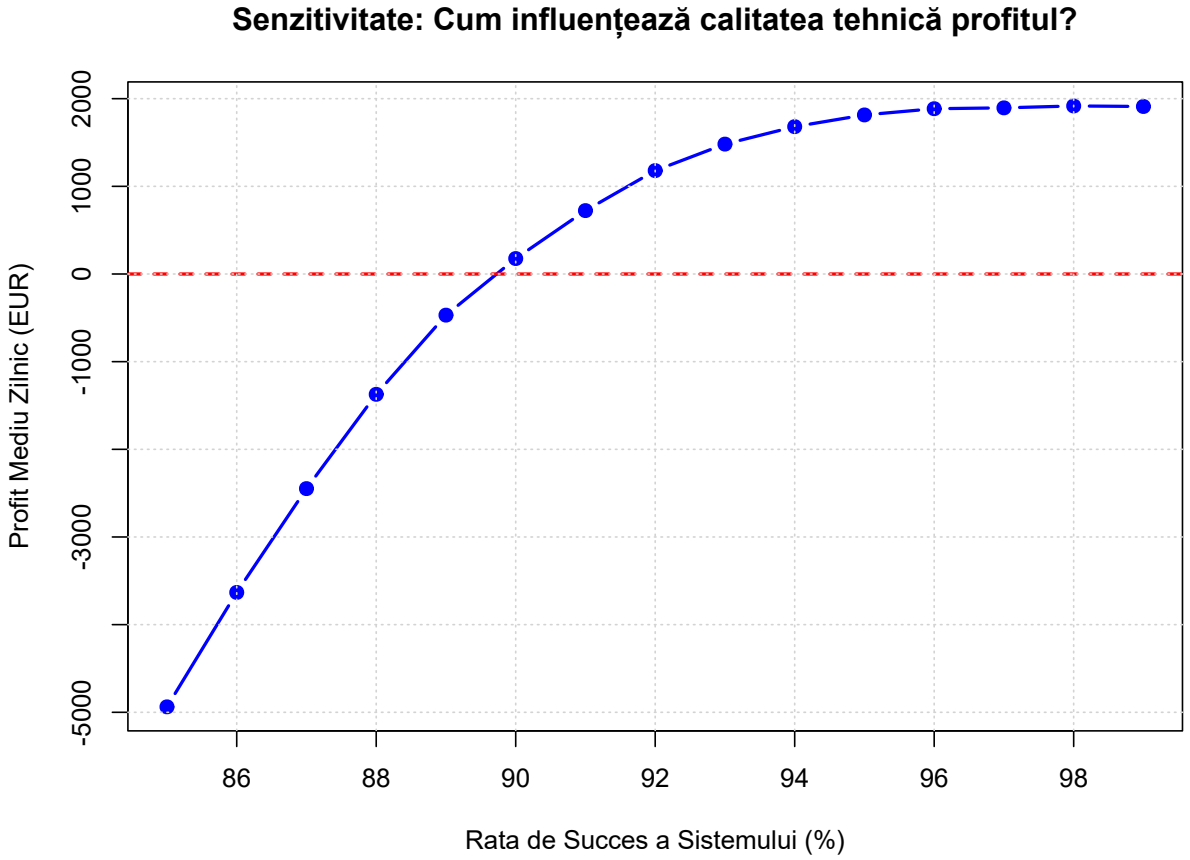
c) Analiza compromisurilor tehnico-economice

În ingineria sistemelor, există întotdeauna un compromis între costul de a menține o calitate tehnică ridicată (ex: servere scumpe, redundanță) și pierderile cauzate de clienții nemulțumiți.

Vom folosi o **Analiză de Senzitivitate** pentru a răspunde la întrebarea: > *"Care este rata minimă de succes tehnic pe care trebuie să o menținem pentru ca afacerea să fie profitabilă?"*

Vom varia rata de succes a cererilor (`prob_succes_churn_cond`) de la 85% la 99% și vom observa evoluția profitului mediu.

```
date_grafic <- genereaza_date_senzitivitate(
  start = 0.85,
  end = 0.99,
  step = 0.01,

  # Parametrii fixi (transmisi prin ...):
  n_zile = 100,
  lambda_trafic = 1000,
  castig_per_succes = 2.0,
  cost_churn = 40,
  cost_sla = 2.0,
  rata_churn_aleator = 0.001 # Churn natural mic (0.1%), ca sa izolam efectul tehnic
)
```



Graficul generat evidențiază o relație neliniară dramatică între calitatea tehnică și succesul economic. Putem distinge trei zone critice:

1. **Zona de Faliment (Rata de Succes $<$ 90%)**
   - Observăm că linia albastră se află sub pragul roșu de rentabilitate ($Profit < 0$).
   - **Cauza:** Costul pierderii clienților (`Cost Churn` = 40 EUR) este mult mai mare decât câștigul marginal dintr-o tranzacție reușită (2 EUR).
   - **Impact:** Chiar și o rată de succes aparent decentă de 86% generează pierderi masive (aprox. -3000 EUR/zi), deoarece numărul de clienți nemulțumiți care părăsesc platforma este suficient de mare pentru a anula toate veniturile.
2. **Punctul Critic (Break-even Point $\approx$ 90%)**
   - Graficul ne arată că **pragul minim de supraviețuire** al afacerii este o rată de succes tehnic de **90%**.
   - În acest punct, veniturile din tranzacțiile reușite abia reușesc să acopere costurile operaționale și pierderile cauzate de churn.
3. **Zona de Profit și Saturare (Rata de Succes $>$ 92%)**
   - Odată depășit pragul critic, profitul crește abrupt, ajungând rapid la valori pozitive (ex: 1000-2000 EUR).
   - **Randamente descrescătoare:** Se observă că diferența de profit între 96% și 99% este mult mai mică decât diferența între 88% și 91%. Curba se aplatizează ("saturare"), ceea ce înseamnă că, după un anumit nivel de calitate, investițiile suplimentare aduc beneficii economice marginale mai mici.

13. Vizualizare statistica (Cerinta 12)

a) Histograme pentru T si profit

Pentru a realiza aceste doua histograme vom defini o functie care accepta ca parametrii 2 vectori: unul cu timpii pana la succes/abandon in ms, iar celalt cu profituri.

```
#' Deseneaza Histogramele pentru Timp si Profit (Cerinta 12a)
#' @param vector_timp
#' @param vector_profit
#' @return Nimic, doar deseneaza histogramele
plot_histograme_12a <- function(vector_timp, vector_profit) {
  # Impartim ecranul in 2 (stanga/dreapta)
  par(mfrow = c(1, 2))

  # 1. Histograma Timpului (T)
  hist(vector_timp,
    col = "lightblue",
    border = "white",
    main = "Distributia Timpului Total (T)",
    xlab = "Timp (ms)",
    ylab = "Frecventa",
    probability = TRUE
  )

  # 2. Histograma Profitului
  hist(vector_profit,
    col = "lightgreen",
    border = "white",
    main = "Distributia Profitului Zilnic",
    xlab = "Profit (EUR)",
    ylab = "Frecventa"
  )

  # Resetam layout-ul
  par(mfrow = c(1, 1))
}
```

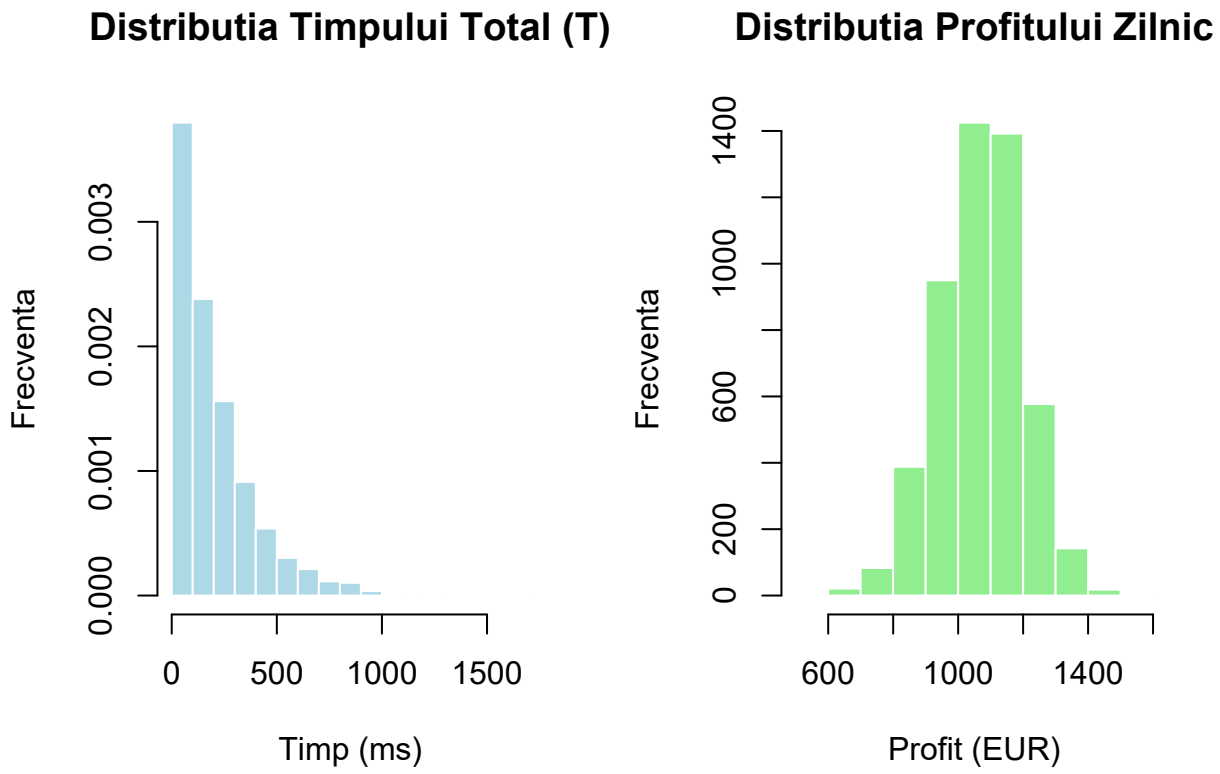
Generam datele:

```
df_cereri_viz <- simuleaza_lot_cereri(
  n_simulari = 5000,
  n_max = 3,
  p_succes = 0.8,
  medie_S = 150
)

vector_profit_viz <- replicate(5000, {
  # Generam o zi cu trafic mediu (1000 clienti)
  rez <- simuleaza_profit_zi(
    nr_clienti = 1000,
    castig_per_succes = 2.0,
    cost_churn = 20,
```

```
    rata_churn_aleator = 0.001
  )
  return(rez["Profit"])
})
```

Si vizualizam histogramele folosind functia:

```
plot_histograme_12a(vector_timp = df_cereri_viz$T, vector_profit = vector_profit_viz)
```



b) Boxplot-uri pentru T conditionat de succes/esec si pentru scenarii diferite

Pentru a reprezenta boxplot-urile vom folosi urmatoarea functie:

```
#' Deseneaza Boxplot-uri comparative (Cerinta 12b)
#' @param df_cereri Dataframe-ul principal pentru analiza Succes/Esec. Trebuie sa contina
#'                  coloanele T (Timp total) si I (Indicator succes: 0 sau 1).
#' @param df_scenariu_1 Dataframe cu rezultatele primului scenariu
#' @param df_scenariu_2 Dataframe cu rezultatele celui de-al doilea scenariu
#' @return Nimic, doar deseneaza boxplot-uri
plot_boxplots_12b <- function(df_cereri, df_scenariu_1, df_scenariu_2) {
  # Layout: 1 rand, 2 coloane
  par(mfrow = c(1, 2))

  # 1. Boxplot Conditionat: Succes vs Esec
```

```
  # Transformam 0/1 in etichete text
  etichete <- factor(df_cereri$I, levels = c(0, 1), labels = c("Esec", "Succes"))

  boxplot(df_cereri$T ~ etichete,
    col = c("salmon", "lightgreen"),
    main = "Timp (T) conditionat de Rezultat",
    ylab = "Timp Total (ms)",
    xlab = "Rezultat Final"
  )

  # 2. Boxplot Comparatie Scenarii
  # Cream o lista pentru boxplot
  boxplot(df_scenariu_1$T, df_scenariu_2$T,
    names = c("Scenariu A", "Scenariu B"),
    col = c("lightblue", "gold"),
    main = "Comparatie Scenarii",
    ylab = "Timp Total (ms)"
  )

  par(mfrow = c(1, 1))
}
```

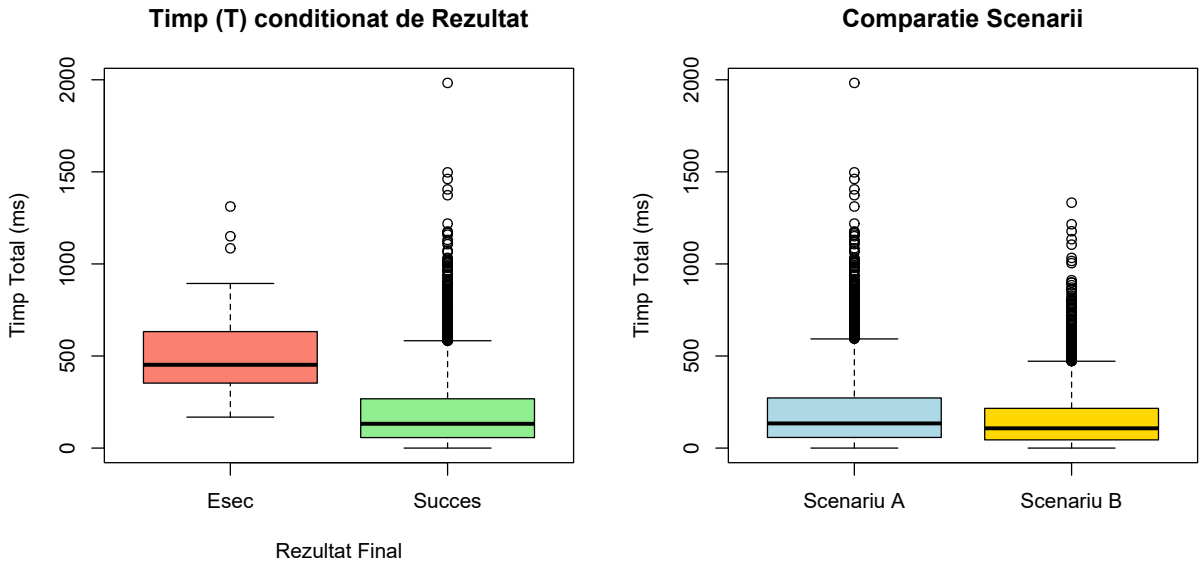
Pentru a vizualiza impactul mecanismului de *Retry* asupra distribuției timpului de răspuns, am definit două scenarii arhitecturale distincte:

1. **Scenariul A (Standard / Robust):**
   - Folosim `n_max = 3` (până la 3 încercări).
   - **Filozofie:** Sistemul prioritizează livrarea cu succes a răspunsului, chiar dacă asta înseamnă că utilizatorul așteaptă mai mult în caz de erori temporare.
2. **Scenariul B (Fail-Fast):**
   - Folosim `n_max = 1` (o singură încercare, fără retry).
   - **Filozofie:** Sistemul prioritizează eliberarea rapidă a resurselor. Dacă apare o eroare, se raportează imediat, fără a ține clientul în așteptare ("Eșuează Rapid").

Generăm datele și vizualizăm diferențele:

```
# 1. Generam datele pentru Scenariul A (Standard: 3 incercari)
# Acesta va avea o dispersie mai mare a timpului din cauza reincercarilor
df_standard <- simuleaza_lot_cereri(
  n_simulari = 5000,
  n_max = 3,
  p_succes = 0.8,
  medie_S = 150
)

# 2. Generam datele pentru Scenariul B (Fail-Fast: 1 incercare)
# Schimbam parametrul n_max la 1 pentru a simula lipsa retry-urilor
# Acesta ar trebui sa aiba o latenta mult mai predictibila (varianta mica)
df_fail_fast <- simuleaza_lot_cereri(
  n_simulari = 5000,
  n_max = 1,
```

```
  p_succes = 0.8,
  medie_S = 150
)

# 3. Apelam functia de vizualizare definita la punctul anterior
# Argumentul 1 (df_standard) este folosit pentru graficul din stanga (Succes vs Esec)
# Argumentele 2 si 3 sunt folosite pentru comparatia directa (Standard vs Fail-Fast)
plot_boxplots_12b(
  df_cereri = df_standard,
  df_scenariu_1 = df_standard,
  df_scenariu_2 = df_fail_fast
)
```



c) Interpretare: Mediană, IQR și Outlieri

Analizăm distribuțiile vizualizate în graficul Boxplot pentru a extrage concluzii despre performanța sistemului.

1. Analiza Medianei (Linia neagră orizontală)

- **Succes vs. Eșec:** Observăm o discrepanță majoră. Mediana pentru cererile cu **Succes** este joasă (aprox. 150ms), indicând că sistemul funcționează rapid în condiții normale. În schimb, mediana pentru **Eșec** este ridicată (aprox. 500ms). Aceasta confirmă că "costul" unui eșec nu este doar eroarea în sine, ci și timpul pierdut în mecanismele de *Retry*.
- **Scenariul A vs. B:** Scenariul *Fail-Fast* (B - Galben) are o mediană vizibil mai mică decât Scenariul Standard (A - Albastru), deoarece elimină timpul de așteptare pentru reîncercări inutile.

2. Analiza IQR (Înălțimea cutiei colorate)

- **Definiție:** Intervalul Interquartilic (IQR) măsoară dispersia statistică (unde se află 50% dintre datele centrale).

- **Interpretare:** Cutia roșie (Eșec) este mult mai înaltă decât cea verde (Succes). Acest lucru indică o **instabilitate mare** a timpului de răspuns în caz de erori – unii utilizatori așteaptă puțin până la eroare, alții așteaptă foarte mult (în funcție de momentul exact când serverul respinge conexiunea).
- **Predictibilitate:** Scenariul B (Galben) are un IQR mai mic (cutie mai îngustă) decât A, ceea ce îl face un sistem mai predictibil (Latență Deterministică).

3. Analiza Outlierilor (Punctele izolate)

- **Observație:** În graficul din dreapta, Scenariul A (Albastru) prezintă o "coadă" lungă de puncte deasupra mustăților, ajungând până la 1500ms+.
- **Semnificație:** Acești outlieri reprezintă **cazurile patologice** (Worst Case Execution Time). Deși rari, acești utilizatori au suferit latențe inacceptabile. Scenariul B reduce drastic numărul și magnitudinea acestor valori extreme, tăind "coada" distribuției.

14. Analiză de Sinteză (Cerința 13)

Acest capitol prezintă o sinteză conceptuală a întregului proiect, integrând elemente de statistică descriptivă, probabilități, simulare Monte Carlo și analiză economică.

a) Rolul Probabilității Empirice

În acest proiect, probabilitatea empirică (obținută prin simulări Monte Carlo) a servit drept mecanism de **validare** pentru modelele teoretice. De exemplu, în exercițiul 8, am comparat media empirică a timpului de răspuns cu media teoretică. Convergența valorilor pentru N mare a confirmat corectitudinea implementării. De asemenea, când distribuțiile sunt prea complexe pentru a fi calculate analitic (ex: sistem cu backoff exponențial și churn dinamic), simularea rămâne singura metodă viabilă.

b) Informațiile Aduse de Condiționări

Condiționarea reduce incertitudinea (entropia) sistemului.

1. **Diagnostic:** Știind că un client a plecat (Churn), probabilitatea ca el să fi întâmpinat erori tehnice crește (inferență Bayesiană).
2. **Predicție:** Cunoscând istoricul recent (ultimele k cereri), putem estima mult mai precis riscul de eșec imediat decât folosind media generală.

Acest lucru permite sisteme adaptive: dacă $P(\text{Eșec}|\text{Istoric}) > \text{prag}$, declanșăm măsuri preventive (ex: circuit breaker).

c) Utilitatea Inegalităților Probabilistice

Inegalitățile (Markov, Cebîșev, Chernoff) oferă garanții de tip **WORST-CASE**.

- **Markov** ne dă o limită superioară 'hard' pentru media cozilor/timpilor.
- **Cebîșev** ne arată stabilitatea sistemului (cât de rar deviem de la medie).
- **Chernoff** este esențială pentru SLA (Service Level Agreements), oferind limite exponențiale pentru evenimente rare dar catastrofale (ex: căderea simultană a serverelor). Ele permit dimensionarea sigură a resurselor.

d) Legătura Performanță Tehnică - Impact Economic

Relația nu este liniară, ci puternic asimetrică.

- O scădere mică a performanței tehnice (ex: p_{succes} scade de la 99% la 95%) poate duce la pierderi economice disproporționate din cauza efectului de compunere (utilizatorii pleacă, viralitate negativă - Churn).
- Există un punct de 'diminishing returns': îmbunătățirea de la 99.9% la 99.99% costă enorm tehnic, dar poate aduce beneficii marginale economic, dacă utilizatorii nu percep diferența.

e) Parametri Cheie și Îmbunătățiri

Cei mai influenți parametri identificați sunt:

1. **Probabilitatea de succes primară (p_{succes}):** influențează tot lanțul.
2. **Factorul de Backoff:** determină cât de repede se congestionează sistemul.

Modificări Propuse:

- **Backoff Adaptiv:** În loc de backoff fix/exponențial simplu, sistemul să primească feedback de la server (ex: 'retry-after' header).
- **Prioritizare Inteligentă:** Cererile clienților fideli (Customer Lifetime Value mare) să aibă prioritate la resurse în momentele de criză.

15. Bibliografie si resurse utile

Pachete Software

Pentru realizarea simulărilor și a vizualizărilor grafice, am utilizat limbajul **R** împreună cu următoarele pachete: * **Base R:** Pentru simulările Monte Carlo, generarea variabilelor aleatoare (`rnorm`, `rexp`, `rpois`, `rbinom`) și graficele standard (`hist`, `boxplot`, `plot`). * **Knitr / RMarkdown:** Pentru generarea automată a acestui raport și formatarea tabelelor (`kable`). * **Shiny:** Pentru realizarea interfeței grafice interactive.

Surse de Inspirație

1. Cursul de Probabilități și Statistică (FMI, Unibuc) – pentru definițiile teoretice ale repartițiilor și inegalităților.
2. Documentația oficială R (CRAN) – pentru parametrizarea funcțiilor de distribuție.
3. *StackOverflow* și *R-bloggers* – pentru optimizarea buclelor `replicate` și formatarea graficelor suprapuse.

16. Dificultăți Întâmpinate

Pe parcursul realizării proiectului, echipa s-a confruntat cu următoarele provocări:

1. **Alegerea parametrilor realiști:** A fost dificil să calibrăm parametrii simulării (ex: `lambda`, `t0`, `backoff`) astfel încât diferențele dintre scenarii (ex: Poisson vs Binomial sau Independent vs Dependent) să fie vizibile grafic, dar să rămână plauzibile pentru un sistem real.

2. **Gestionarea simulărilor imbricate:** Implementarea Churn-ului condiționat a necesitat simularea unei istorii pentru fiecare utilizator, ceea ce a crescut complexitatea computațională. A trebuit să optimizăm codul folosind funcții vectorizate (`replicate`, `sapply`) în loc de bucle `for` excesive pentru a menține timpul de rulare rezonabil.
3. **Interpretarea rezultatelor contraintuitive:** Inițial, rezultatele inegalității lui Jensen păreau paradoxale (media costurilor > costul mediei), necesitând o aprofundare a teoriei funcțiilor convexe pentru a interpreta corect riscul economic.

16. Probleme Deschise și Limitări

Deși modelul surprinde aspectele principale, există simplificări care constituie probleme deschise pentru versiuni viitoare:

1. **Concurența resurselor:** Modelul actual nu simulează coada de așteptare la nivel de CPU/RAM. Timpul de răspuns S_i este generat independent, dar în realitate, dacă 1000 de clienți vin simultan, S_i ar trebui să crească pentru toți (model de tip *Queueing Theory* M/M/n).
2. **Variabilitatea traficului intraday:** Traficul este modelat global pe zi (λ constant). Nu am modelat orele de vârf (ex: seara) vs. orele de liniște (noaptea), care ar putea declanșa Churn-ul mai agresiv în ferestre scurte de timp.
3. **Heterogenitatea utilizatorilor:** Am presupus că toți utilizatorii au aceeași toleranță la erori (k). Un model mai complex ar segmenta utilizatorii în categorii (Premium vs Free) cu comportamente diferite de abandon.

17. Concluzii

În cadrul acestui proiect, am modelat și simulat comportamentul probabilistic al unui serviciu online, analizând interdependența dintre parametrii tehnici (latență, erori, retry-uri) și indicatorii de performanță economică (profit, churn).

Principalele concluzii desprinse din studiul nostru sunt:

1. **Validitatea Simulării Monte Carlo:** Prin utilizarea Legii Numerelor Mari, am demonstrat că probabilitățile empirice obținute din simulări ample ($M \geq 10.000$) converg către valorile teoretice. Aceasta validează simularea ca instrument de decizie în scenarii complexe unde calculul analitic este dificil (ex: sisteme cu dependențe și backoff variabil).

2. **Mitul „Mediei" în IT:** Analiza distribuțiilor timpilor de răspuns (Exponențial vs. Normal) și aplicarea **Inegalității lui Jensen** au relevat că „timpul mediu" este un indicator înșelător. Într-un sistem cu distribuție *Long Tail*, media ascunde outlierii critici care cauzează frustrarea utilizatorilor. Am arătat că mediana reflectă mai bine experiența utilizatorului tipic, în timp ce media costurilor ($E[\varphi(T)]$) este sistematic mai mare decât costul mediei ($\varphi(E[T])$).

3. **Pericolul Dependențelor:** Comparația dintre scenariile independente și cele dependente a arătat că eșecurile care induc latență suplimentară (degradare în cascadă) cresc varianța sistemului exponențial. Aceasta duce la o instabilitate majoră și face imposibilă garantarea unor SLA-uri stricte fără mecanisme de protecție (ex: Circuit Breakers).

4. **Compromisul Tehnico-Economic (Pragul de 90%):** Analiza de senzitivitate a demonstrat o relație neliniară între calitatea tehnică și profit. Am identificat un **punct critic (Break-even Point)** la o rată de succes de aproximativ **90%**. Sub acest prag, costurile generate de pierderea clienților (Churn) depășesc veniturile, indiferent de volumul de trafic. Peste acest prag, profitul se stabilizează, indicând că investițiile tehnice au un randament descrescător după un anumit nivel de excelență.

5. **Strategia de Retry:** Analiza comparativă a scenariilor *Standard* (Retry=3) vs. *Fail-Fast* (Retry=0) a evidențiat că retry-urile cresc rata de succes final, dar cu prețul unei latențe mult mai mari în caz de eșec (cozi lungi în distribuția T). Alegerea strategiei optime depinde strict de natura aplicației: *Fail-Fast* pentru sisteme interactive predictibile, *Retry* pentru procese de fundal critice.

În concluzie, proiectul demonstrează că deciziile de arhitectură software nu pot fi luate izolat de modelarea probabilistică a traficului și de constrângerile economice. Un sistem robust nu este doar unul care funcționează, ci unul care își gestionează eficient riscurile și variabilitatea.