Analiza probabilistică a performanței unui serviciu online cu trafic aleator și impact economic

Proiect realizat de: Leuștean Ștefan(Lider)
Miu Georgian-Fabian
Preda Maria-Alexandra
Pupezescu Alina-Elena
Grupa: 241
Profesor coordonator: Lect. Dr. Alexandru Amarioarei
Manuela-Simona Cojocea

Cuprins

1. Descrierea problemei

În contextul digital actual, platformele online, funcționează într-un mediu caracterizat de incertitudine permanentă. Traficul utilizatorilor variază de la o zi la alta, timpii de răspuns nu sunt constanți, iar eșecurile temporare, mecanismele de retry și politicile de backoff fac parte din funcționarea normală a sistemului. Toate aceste elemente influențează direct experiența utilizatorilor și, în final, performanța economică a platformei.

În acest proiect am ales să analizăm comportamentul probabilistic al unei platforme online pe o perioadă de 5 ani, între 2019 și 2023, pentru a surprinde atât variațiile zilnice, cât și tendințele pe termen mediu și lung. Această perioadă permite modelarea sezonalității, a fluctuațiilor de popularitate și a modificărilor progresive ale traficului, oferind o imagine realistă asupra evoluției sistemului.

Pentru fiecare zi din intervalul 2019–2023, numărul de utilizatori activi este considerat o variabilă aleatoare, influențată de factori externi precum campanii de marketing, sezonalitate sau schimbări în comportamentul utilizatorilor. Acest trafic zilnic determină nivelul de încărcare al platformei și afectează direct performanța tehnică. În model, numărul de clienți activi într-o zi este notat K_d și este generat folosind distribuții clasice pentru modelarea sosirilor aleatoare, precum distribuția Poisson și distribuția binomială.

Fiecare utilizator trimite cereri care pot reuși, eșua temporar și fi reluate, sau fi abandonate după un număr maxim de retry-uri. Fiecare încercare are un timp de răspuns aleator, un rezultat succes/eșec și, la eșec, un timp de așteptare (backoff). Timpul total al cererii este suma timpilor de răspuns și a backoff-urilor, iar rezultatul final este succes sau eșec.

Utilizatorii pot părăsi platforma între cereri, fie aleator, fie din cauza unei experiențe slabe (eșecuri frecvente, timpi mari, încălcări SLA). Probabilitatea de churn depinde de performanța recentă a sistemului.

Cererile rezolvate aduc venit, pierderea utilizatorilor produce costuri mari, iar încălcarea SLA atrage penalități. Agregând rezultatele zilnice (2019–2023), modelul permite evaluarea profitului/pierderii și a riscului economic în funcție de trafic și politicile de retry.

2. Aspecte teoretice folosite în rezolvarea problemei care depășesc nivelul cursului

1. Modulul

Modulul reprezintă valoarea (sau valorile) care apare cel mai frecvent într-o distribuție sau într-un eșantion.

1. Pentru variabile discrete, modul este valoarea x unde probabilitatea P(X=x) e maximă
2. Pentru variabile continue, modul este valoarea x în care densitatea de probabilitate f(x) atinge maximul.

2. Mediana

Mediana este valoarea care împarte distribuția în două jumătăți cu probabilități egale:

$P(X \leq m) \geq 0.5 \text{ și } P(X \geq m) \geq 0.5$.

1. În cazul variabilelor discrete, mediana poate să nu fie unică, deoarece funcția de repartiție are salturi, iar pot exista mai multe valori care satisfac condiția.
2. Pentru variabile continue, în special atunci când funcția de repartiție F(x) este strict crescătoare, mediana este de obicei unică și satisface egalitatea exactă: $P(X \leq m)=0.5$.

3. Reprezentări grafice și orice altă formă multimedia oportună proiectului

Shiny este un pachet în R pentru dezvoltarea de aplicații web interactive. Acesta permite integrarea logicii de calcul și procesare din R cu elemente grafice dinamice, oferind astfel o platformă pentru explorarea datelor, vizualizări interactive, rularea de simulări, cum ar fi cele Monte Carlo, și crearea de instrumente educaționale sau analitice.

Arhitectura Shiny se bazează pe separarea între, interfața utilizatorului (UI), care definește structura și elementele vizuale, și componenta de server, care conține logica de calcul și manipularea datelor. Cele două părți sunt sincronizate prin intermediul programării reactive, asigurând o actualizare automată și instantanee a conținutului în funcție de interacțiunile utilizatorului.

În acest proiect, Shiny a fost utilizat pentru a construi o aplicație web interactivă care permite explorarea și analiza completă a modelului de performanță. Interfața facilitează investigarea: distribuției numărului zilnic de cereri, K_d, a timpilor de răspuns, S_i, N, T, a distribuțiilor condiționate (N, F) și (N, T), a studiilor probabilistice, inegalitățile Markov, Cebîșev, Cernov; a mecanismului de pierdere a utilizatorilor, churn, precum și a analizei economice a profitului zilnic și anual.

Input-uri in cadrul proiectului:

Input	Tip	Utilizare
distrK	selectInput	Selectează distribuția pentru Kd (Poisson/Binomial)
empK	numericInput	Numărul de simulări pentru distribuția Kd
recalcK	actionButton	Declanșează recalcularea distribuției Kd
distrS	selectInput	Selectează distribuția pentru Si (Exponențială/Normală)
empS	numericInput	Numărul de simulări pentru distribuția Si

rateS	numericInput	Parametrul rate pentru distribuția exponențială
meanS	numericInput	Media pentru distribuția normală
varS	numericInput	Varianta pentru distribuția normală
recalcS	actionButton	Recalculează distribuția Si și graficele
nmax	numericInput	NMAX – numărul maxim de încercări
probSucc	numericInput	Probabilitatea de succes per cerere
recalcN	actionButton	Recalculează distribuția N
empNF	numericInput	Numărul de simulări pentru distribuția (N, F)
recalcNF	actionButton	Recalculează heatmap-ul și distribuțiile N și F
empNT	numericInput	Numărul de simulări pentru distribuția (N, T)
recalcNT	actionButton	Recalculează graficele și tabelele (N, T)
empSDep	numericInput	Numărul de simulări pentru S dependenți
recalcSDep	actionButton	Recalculează graficele pentru latența dependentă
empAgreg	numericInput	Numărul de simulări pentru agregarea zilnică
recalcAgreg	actionButton	Recalculează graficele agregate și histograma normală
empIneg	numericInput	Numărul de simulări pentru inegalități probabilistice
recalcIneg	actionButton	Recalculează tabelele Markov, Chebyshev, Chernoff și Jensen
distrChurn	selectInput	Tip distribuție pentru Churn (Constant/Condiționat)
probChurn	numericInput	Probabilitatea de churn pentru distribuția constantă
empChurn	numericInput	Numărul de simulări pentru Churn condiționat
mChurn	numericInput	Dimensiunea ferestrei m pentru Churn condiționat
kChurn	numericInput	Numărul minim de eșecuri pentru Churn condiționat
recalcChurn	actionButton	Recalculează funcția de masă și distribuția Churn
medieCerUser	numericInput	Media cererilor per utilizator
empCerUser	numericInput	Numărul de simulări empirice pentru probabilitatea de pierdere

venitPerSucces	numericInput	Venitul per cerere reușită
costAchizitie	numericInput	Costul pierderii unui utilizator (Churn)
tSLA	numericInput	Pragul de timp SLA
penSLA	numericInput	Penalitatea SLA
dateProfit	dateInput	Data pentru simularea profitului zilnic
empProfZ	numericInput	Numărul de simulări pentru profit zilnic
recalcProfit	actionButton	Recalculează profitul zilnic și anual
empEX12	numericInput	Numărul de simulări pentru exercițiul EX12

Kd

Selectati tipul distributiei

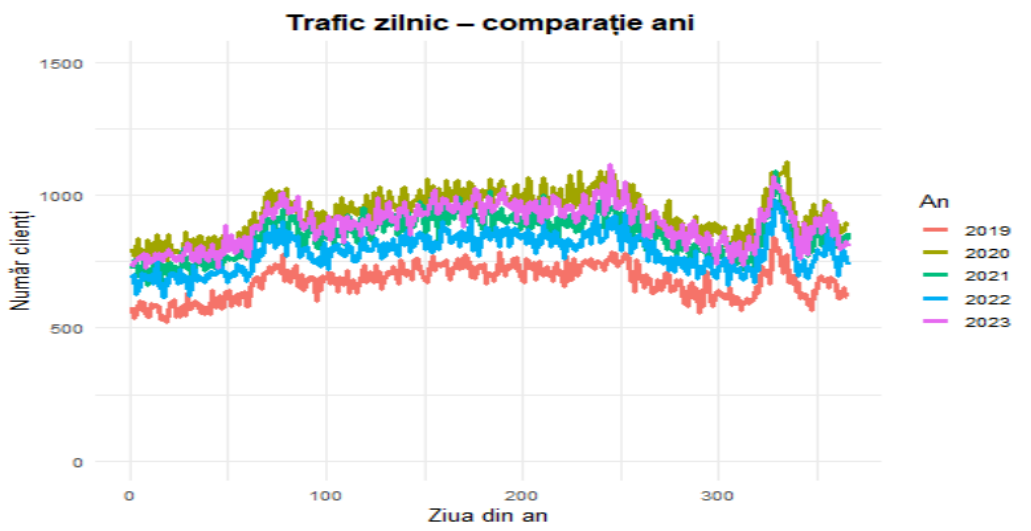
Poisson

Simulări

1000

Recalculează

Comparatie ani



Si

Selectati tipul distributiei

Exponentiala

Simulari

1000

Parametri Exponentiala

Rata (1 / Medie)

0.0333333333333

Parametri Normala

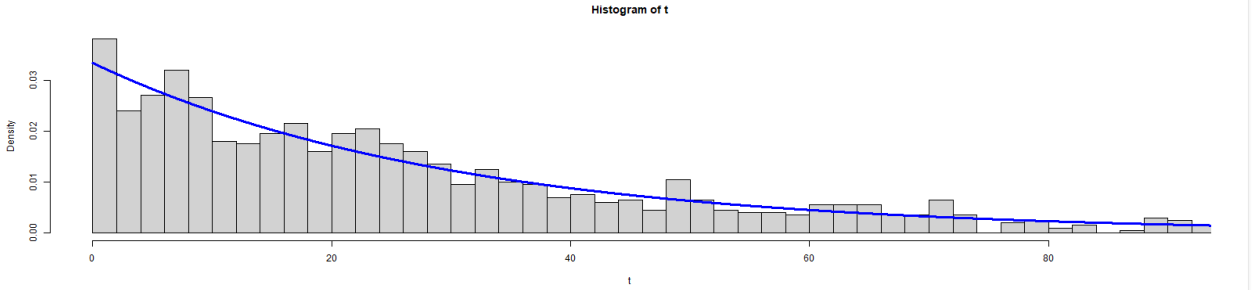
Medie

30

Varianta

25

Recalculează



Studiu Empiric

Empiric | Teoretic

Mean_Empirical	Variance_Empirical	Median_Empirical	Modal_Empirical
28.10795775371172	830.7990626417436	19.98287794645876	0.3

N

NMAX

5

Probabilitate Succes Cerere

0.4

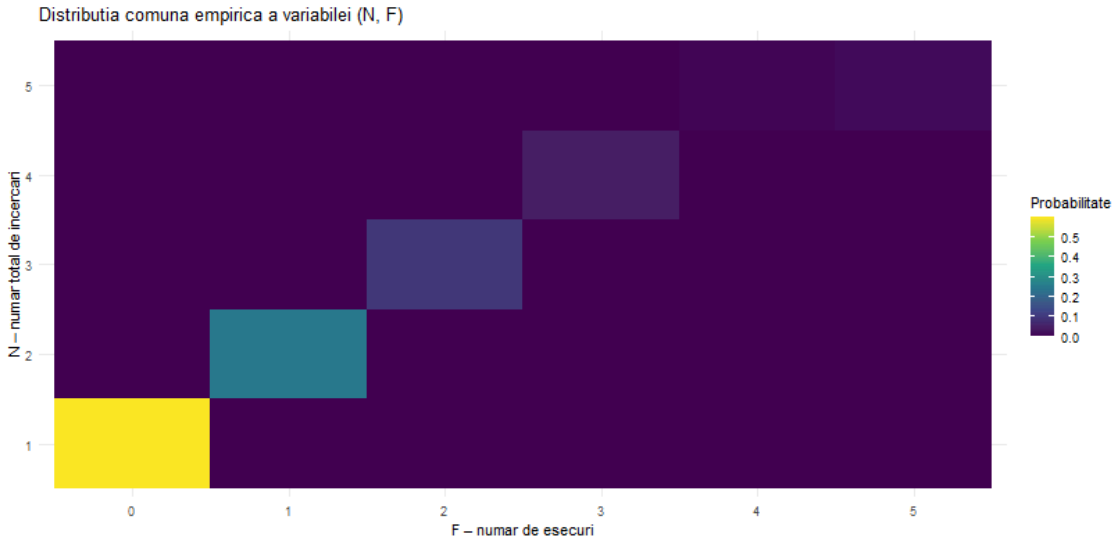
Recalculează

(N, F)

Simulari

1000

Recalculează



Distributie N: 1 - , 2 - , 3 - , 4 - , 5 -

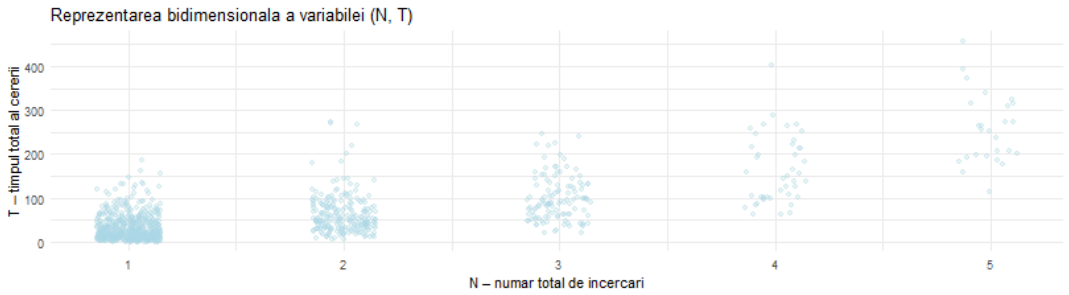
Distributie F: 0 - , 1 - , 2 - , 3 - , 4 - , 5 -

(N, T)

Simulari

1000

Recalculează



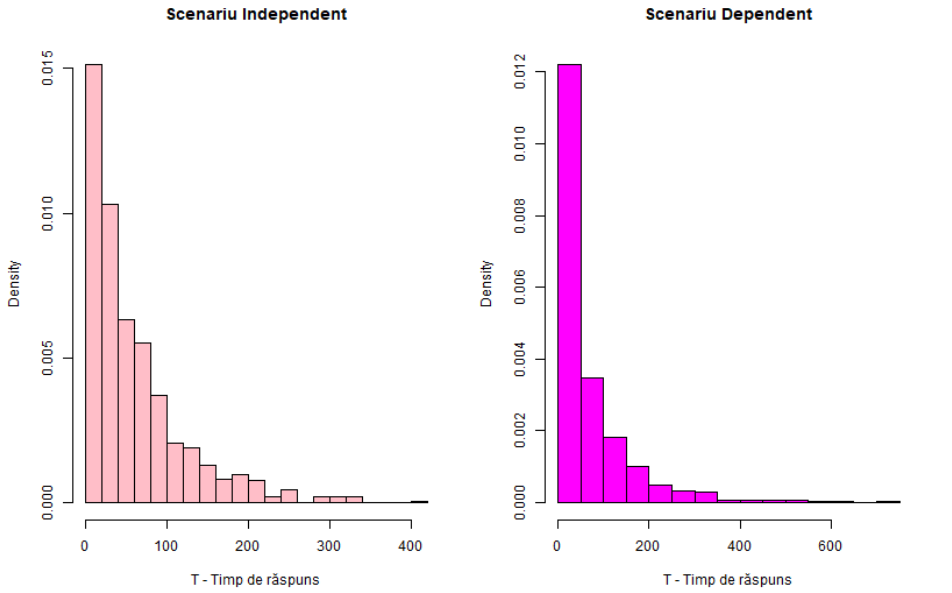
mean_N	mean_T	var_N	var_T	covariance	
1.689	57.67728811319904	1.017296296296296	3709.147256193317	44.48410871252383	0.72417

Timpi Si independenți vs dependenți (latența creste după eșecuri

Simulări

1000

Recalculează



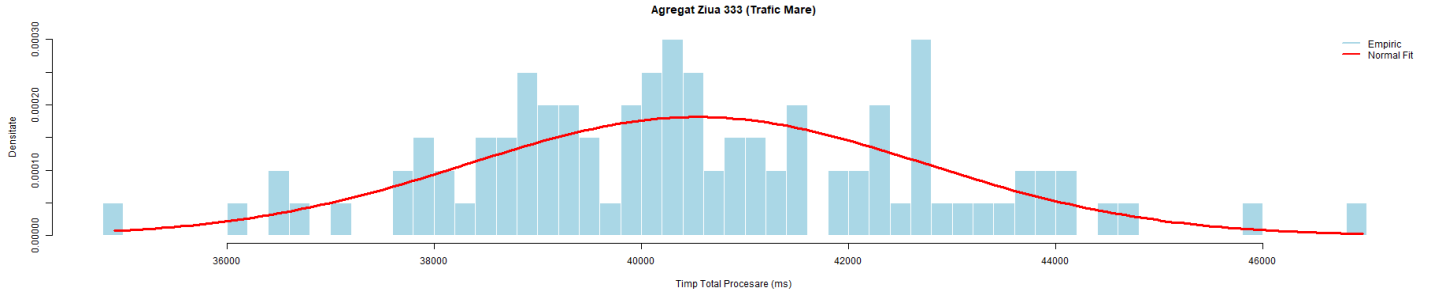
Aproximare normală și agregare

Simulări

100

Recalculează

Agregat pentru o zi



Inegalități probabilistice (garanții worst-case) - T

Simulări

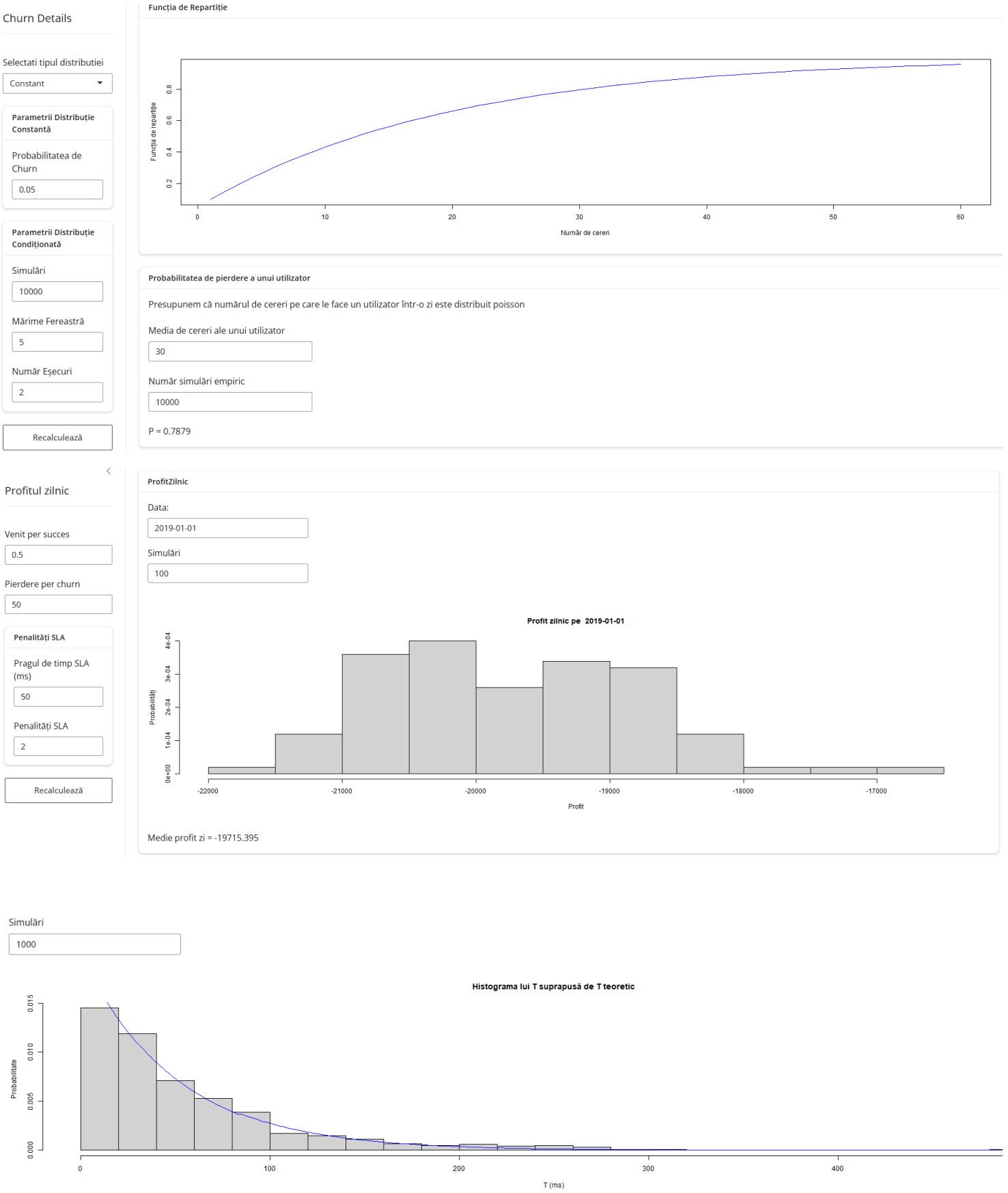
100000

Recalculează

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

Markov

t	P_empirical	P_theoretical	Markov_empiric	Markov_theretic
50	0.38816	0.3866127399576912	1	0.98413531976043
60	0.32221	0.3215734184056427	0.908146285635105	0.820112766467025
70	0.26853	0.2681036654613764	0.778812538768723 3	0.7029537998288786
80	0.22319	0.2240904690275795	0.681460971422632 8	0.6150845748502688
90	0.18644	0.187833532820265	0.605743085709007	0.54674184431135
100	0.1562	0.1579572894121927	0.545168777138106 3	0.492067659880215
110	0.13249	0.1333193756474718	0.495607979216460 3	0.4473342362547409
120	0.11186	0.1129506720378139	0.454307314281755 2	0.4100563832335125
130	0.09454	0.09603468008740634	0.419360597798543 3	0.3785135845232423



Funcții Shiny utilizate în cadrul proiectului

1. reactiveValues(): stochează valori ce se pot schimba la interacțiuni. Exemple: K, S, N, NF, etc.
2. reactive(): creează o valoare care se recalculează automat când input-urile sau dependențele ei se modifică, permițând gestionarea dinamică a datelor în Shiny.

3. observeEvent(input$..., { ... }): ascultă evenimente și actualizează valorile reactive. Exemple: recalcK, recalcS, etc.
4. renderPlot({ ... }): generează un grafic dinamic
5. renderDT({ ... }):generează un tabel dinamic
6. renderText({ ... }): generează text dinamic
7. renderUI({ ... }): generează conținut HTML
8. req(input$...): se asigură că inputul există înainte de a rula codul reactiv. Exemplu: Exercitiul 12.

Pachetul bslib oferă o interfață grafică modernă și teme personalizabile pentru aplicațiile Shiny. Ea servește ca un cadru de stilizare și componentă de interfață utilizator, permițând crearea de layout-uri responsive, structuri cu card-uri și bare laterale cu aspect modern.
Funcții bsilb utilizate în cadrul proiectului:

1. page_navbar(): creează structura principală de navigare a aplicației cu bara de titlu și stil general.
2. nav_panel(): definește un tab distinct în cadrul navbar-ului principal, fiecare având propriul conținut.
3. layout_sidebar(): setează aranjamentul cu sidebar și zona principală, oferind două zone structurate pentru interfață.
4. sidebar(): creează containerul pentru elementele de input din sidebar, controale, butoane, selectori.
5. card(): generează un container cu bordura vizibilă, adesea cu antet, card_header(), folosit pentru a grupa vizual outputuri sau inputuri în cadrul paginii.
6. navset_card_underline(): implementează un set de taburi în interiorul unui card, ideal pentru alternarea între diferite vizualizări sau seturi de date.
7. div(): o funcție flexibilă pentru structurarea și stilizarea HTML la nivel de bloc, folosită pentru a grupa și organiza elemente în interiorul paginii sau a unui container.

4. Precizări privind pachete software folosite și surse de inspirație

1. Pachetul “ggplot2”

“ggplot2” este pachetul R utilizat pentru vizualizarea datelor, bazat pe principiul Grammar of Graphics. Acesta permite construcția modulară a graficelor prin stratificare: definirea setului de date, maparea variabilelor în spațiul estetic, aes(), și selectarea geometriei vizuale, geom_*. În cadrul proiectului, pachetul este esențial pentru analiza distribuțiilor, a dependențelor dintre variabile și pentru modelarea grafică a riscului.
Funcțiile utilizate în cadrul proiectului sunt:

1. ggplot(): Inițializează obiectul grafic și stabilește cadrul de lucru, asigurând legătura cu setul de date specificat.
2. aes(): Definește mapările estetice între variabile și proprietățile vizuale ale graficului, stabilind, de exemplu, variabila pentru axa absciselor, ordonatei, culorii sau formei.
3. geom_line(): Realizează o reprezentare grafică sub formă de linie, adecvată pentru vizualizarea evoluției seriilor de timp sau a tendințelor.
4. geom_col(): Construiește o diagramă cu bare verticale, unde înălțimea fiecărei bare corespunde direct valorii asociate categoriei respective.

5. geom_tile(): Generează o hartă termică, heatmap, prin divizarea planului în celule rectangulare, colorate proporțional cu valoarea atribuită fiecărei combinații de pe axe.
6. geom_jitter(): Adaugă puncte de dispersie cu o ușoară perturbație aleatoare a poziției pe una sau ambele axe, pentru a reduce suprapunerea și a îmbunătăți lizibilitatea.
7. scale_fill_viridis_c(): Aplică o scală de culori secvențială continuă din paleta Viridis, recunoscută pentru perceptibilitate uniformă și accesibilitate cromatică.
8. labs(): Gestionează etichetele textuale ale graficului, inclusiv titlul, subtitlul, denumirile axelor și ale legendelor.
9. theme_minimal(): Aplică o temă de prezentare minimalistă, eliminând elemente decorative non-esențiale și păstrând doar liniile de grilă și textele necesare pentru o vizualizare clară.

2. Pachetul "DT"

"DT", DataTables, este un pachet R care implementează componente interactive de tip tabel pentru vizualizarea structurată a datelor. Acesta permite configurarea afișării prin controlul opțiunilor de interfață, precum suprimarea indexelor, a paginării sau a căutării, și oferă integrare nativă cu framework-ul Shiny pentru implementarea unor vizualizări dinamice în aplicații web. În acest proiect, pachetul DT din R este utilizat pentru afișarea structurată a obiectelor de tip data.frame.
Funcțiile utilizate în cadrul proiectului sunt:

1. renderDT(): Funcție folosită în partea de server a aplicației Shiny pentru a genera și actualiza dinamic tabele interactive pe baza datelor sau a reacției la input-uri.
2. dataTableOutput(): Funcție utilizată în interfața Shiny pentru a defini și rezerva locul în care tabelul interactiv va fi randat și afișat.
3. datatable(): Funcția principală a pachetului DT care transformă un obiect data.frame sau o matrice într-un tabel interactiv, configurând structura, opțiunile și stilul de afișare înainte de randare.

5. Codul și comentarea acestuia, precum și a soluției prezentate

1.1. Repartizarea variabilei aleatoare discrete Kd:

1. K_d~Poiss(λ_d)
   Modelarea cu distribuția Poisson presupune că fiecare client apare independent, numărul total de clienți zilnici este aleator și nu există o limită superioară strictă. Aceasta este potrivită pentru situații cu trafic redus, în care platforma este scalabilă, nu există o limită clară de utilizatori simultani, iar cererile sunt rare raportat la populația totală, cum este cazul unei aplicații cu creștere organică.

2. Kd~Bin(BMAX, proc_2019)
   În acest model, BMAX reprezintă populația maximă posibilă, iar proc_2019 probabilitatea ca un utilizator să fie activ în ziua d. Distribuția binomială descrie un trafic plafonat și este potrivită atunci când există un număr finit de utilizatori, nu toți sunt activi zilnic, iar traficul are un plafon natural, cum se întâmplă într-o aplicație enterprise.

1.2. Simularea traficului anual și analiza histogramelor comparative

Alegerea unei unități de timp zilnice este naturală și realistă deoarece majoritatea rapoartelor operaționale și economice sunt agregate la nivel de zi, sezonalitatea se observă clar, iar efectele performanței (SLA, churn, venit) se acumulează progresiv. În același timp, acest nivel permite simulări pe termen lung fără un detaliu excesiv. Astfel, ziua devine unitatea fundamentală a modelului, iar pentru fiecare zi se generează traficul, cererile, rezultatele tehnice și impactul economic.

1. Simularea traficului anual
   1. Structura generală

```
KDetails <- new.env();
KDetails$distribution = "pois"
KDetails$years = c(2019, 2020, 2021, 2022, 2023)
KDetails$covid_years = c(2020)
```

   Structura principală a modelului este definită într-un mediu simplu, KDetails, care stochează parametrii esențiali pentru distribuția lui K_d:

   2. Sezonalitate anuală

```
season_base <- 1 + 0.12 * sin(2 * pi * (.days_vec - 90) / 365)
```

   Acest model reflectă faptul că există luni mai bune și mai slabe, cu variații netede ale activității utilizatorilor pe parcursul anului, având o anvergură de aproximativ ±12%, chiar și în absența unor evenimente speciale.

   3. Boom

```
bump <- function(center, width, amp) {
  1 + amp * exp(-((.days_vec - center) / width)^2)
}
season <- season_base *
  bump(75,  12, 0.15) *   # martie
  bump(245, 15, 0.10) *   # back to school
  bump(330,  8, 0.35) *   # black friday
```

```
  bump(355, 10, 0.20)      # craciun
```

Funcția bump() simulează creșteri punctuale și tranzitorii ale traficului, specifice unor evenimente precum campanii promoționale, perioade comerciale de vârf, precum Black Friday. Ea implementează variații netede în volumul de utilizatori, reflectând realist modul în care astfel de evenimente impactează activitatea unei platforme online.

4. Sezonalitate săptămânală

```
week <- 1 + 0.02 * cos(2 * pi * .days_vec / 7)
```

Aceasta surprinde diferența dintre zilele lucrătoare și weekend, fiind și un efect semnificativ dat prin parametru.

5. Factor Covid

```
covid <- 1 + 0.30 * (1 + tanh((d - 70) / 12)) / 2
```

Acest termen modelează creșterea treptată a utilizării online începând din martie 2020 și stabilizarea ulterioară la un nivel mai ridicat, reflectând o adaptare progresivă a comportamentului utilizatorilor, fără salturi bruște.

6. Nivel mediu diferit pe ani

```
KDetails$max_active_users = 2000 # n din distributia binomiala
KDetails$yearly_lambda = c(650, 900, 820, 760, 860)
KDetails$yearly_proc_bin = KDetails$yearly_lambda / KDetails$max_active_users
```

Aceste valori caracterizează nivelul mediu anual de utilizare al platformei și dinamica creșterii sau stagnării serviciului, fiind utilizate în modelul Poisson ca intensitate medie a traficului, yearly_lambda, iar în modelul binomial ca probabilitate zilnică de activare a utilizatorilor dintr-o populație finită, max_active_users și yearly_proc_bin.

7. Lambda zilnic

```
KDetails$lambdaDistribution = cbind()
for (i in 1:length(KDetails$years)) {
  year <- KDetails$years[i]
  lambda <- KDetails$yearly_lambda[i]
  isCovid <- year %in% KDetails$covid_years

  KDetails$lambdaDistribution = cbind(
    KDetails$lambdaDistribution,
    lambda * KDetails$season * KDetails$week * ifelse(isCovid, KDetails$covid, 1)
  )
}

KDetails$procDistribution = cbind()
for (i in 1:length(KDetails$years)) {
  year <- KDetails$years[i]
  proc <- KDetails$yearly_proc_bin[i]
  isCovid <- year %in% KDetails$covid_years

  KDetails$procDistribution = cbind(
    KDetails$procDistribution,
```

```
    proc * KDetails$season * KDetails$week * ifelse(isCovid, KDetails$covid, 1)
  )
}
```

Pentru distribuția Poisson, KDetails$lambdaDistribution creează o matrice în care fiecare coloană reprezintă un an și fiecare rând o zi din an. Valoarea pentru o zi specifică se obține prin înmulțirea nivelului mediu anual de trafic, KDetails$yearly_lambda, cu factorii reali: sezonalitatea anuală, KDetails$season, ciclul săptămânal, KDetails$week, și, pentru anul 2020, impactul pandemiei, KDetails$covid.

Pentru distribuția Binomială, KDetails$procDistribution creează o matrice similară, dar stochează probabilitățile zilnice ca parametru al distribuției Binomiale. Probabilitatea de bază anuală, KDetails$yearly_proc_bin, este ajustată cu aceiași factori: sezonalitate, zi a săptămânii și COVID.

8. Simularea traficului zilnic

```
simulateUserYear <- function(year=2019) {
  stopifnot(year %in% KDetails$years)
  yearIndex = which(KDetails$years == year)
  if (KDetails$distribution == "pois") {
    rpois(.days, KDetails$lambdaDistribution[,yearIndex])
  }
  else {
    rbinom(.days, KDetails$max_active_users, KDetails$procDistribution[,yearIndex])
  }
}
```

Funcția simulateUserYear(year) generează un vector cu numărul de utilizatori activi pentru fiecare zi a anului selectat, pe baza distribuției curente setate în KDetails$distribution.

2. Generarea histogramelor anuale

```
createYearsPlot <- function(distr = NULL) {
  if (!is.null(distr))
    KDetails$distribution = distr
  df <- do.call(
    rbind,
    lapply(KDetails$years, function(y) {
      data.frame(
        day = .days_vec,
        users = simulateUserYear(y),
        year = factor(y)
      )
    })
  )
  ggplot(df, aes(x = day, y = users, color = year)) +
    geom_line(linewidth = 1.1) +
    scale_y_continuous(limits = c(0, 1500)) +
    labs(
      title = "Trafic zilnic – comparație ani",
```

```
      x = "Ziua din an",
      y = "Număr clienți",
      color = "An"
    ) +
    theme_minimal(base_size = 13) +
    theme(
      plot.title = element_text(face = "bold", hjust = 0.5),
      legend.position = "right"
    )
}
```

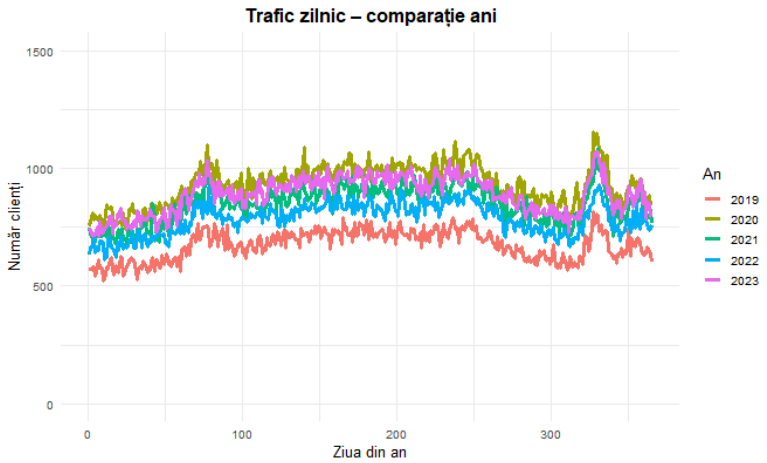
Funcția createYearsPlot creează un grafic care suprapune traficul zilnic simulat pentru fiecare an. Această reprezentare vizuală permite compararea directă a nivelului mediu de activitate, a oscilațiilor de la o zi la alta și a diferențelor evidente dintre anii normali și cei cu impact major, precum anul 2020.

3. Generarea histogramelor lunare

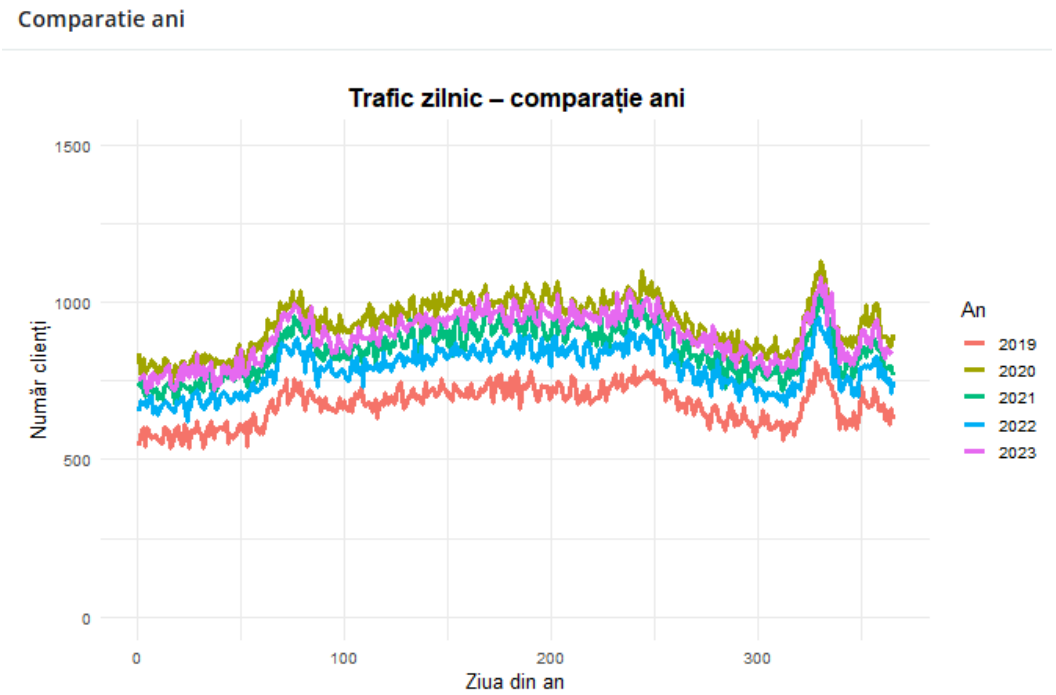
```
createMonthsPlot <- function(year = 2019, distr = NULL) {
  if (!is.null(distr))
    KDetails$distribution = distr
  stopifnot(year %in% KDetails$years)
  nyear <- simulateUserYear(year) # Nu luam in calcul an bisect
  month_lengths <- c(31,28,31,30,31,30,31,31,30,31,30,31)
  month_names <- month.abb
  month_id <- rep(1:12, times = month_lengths)
  daily_by_month <- split(nyear, month_id)
  max_days <- max(month_lengths)
  M <- matrix(NA, nrow = max_days, ncol = 12)
  for (m in 1:12) {
    M[1:length(daily_by_month[[m]]), m] <- daily_by_month[[m]]
  }
  df <- data.frame(
    day   = rep(1:nrow(M), times = ncol(M)),
    month = factor(rep(month_names, each = nrow(M)), levels = month_names),
    users = as.vector(M)
  )
  ggplot(df, aes(x = month, y = users)) +
    geom_col(position = "dodge", na.rm = TRUE,
             fill = scales::hue_pal()(length(KDetails$years))[which(KDetails$years == year)])
+
    labs(
      title = paste("Useri Activi pe luna -", year),
      x = "Luni",
      y = "User Activi"
    ) +
    theme_minimal(base_size = 12) +
    theme(
      plot.title = element_text(face = "bold", hjust = 0.5),
      panel.grid.minor = element_blank()
    )
}
```

Funcția createMonthsPlot generează un grafic cu bare verticale care arată distribuția zilnică a utilizatorilor activi în fiecare lună, evidențiind variațiile și tiparele lunare ale traficului pentru anul selectat.

4. Analiza histogramelor
    1. Analiza pe ani

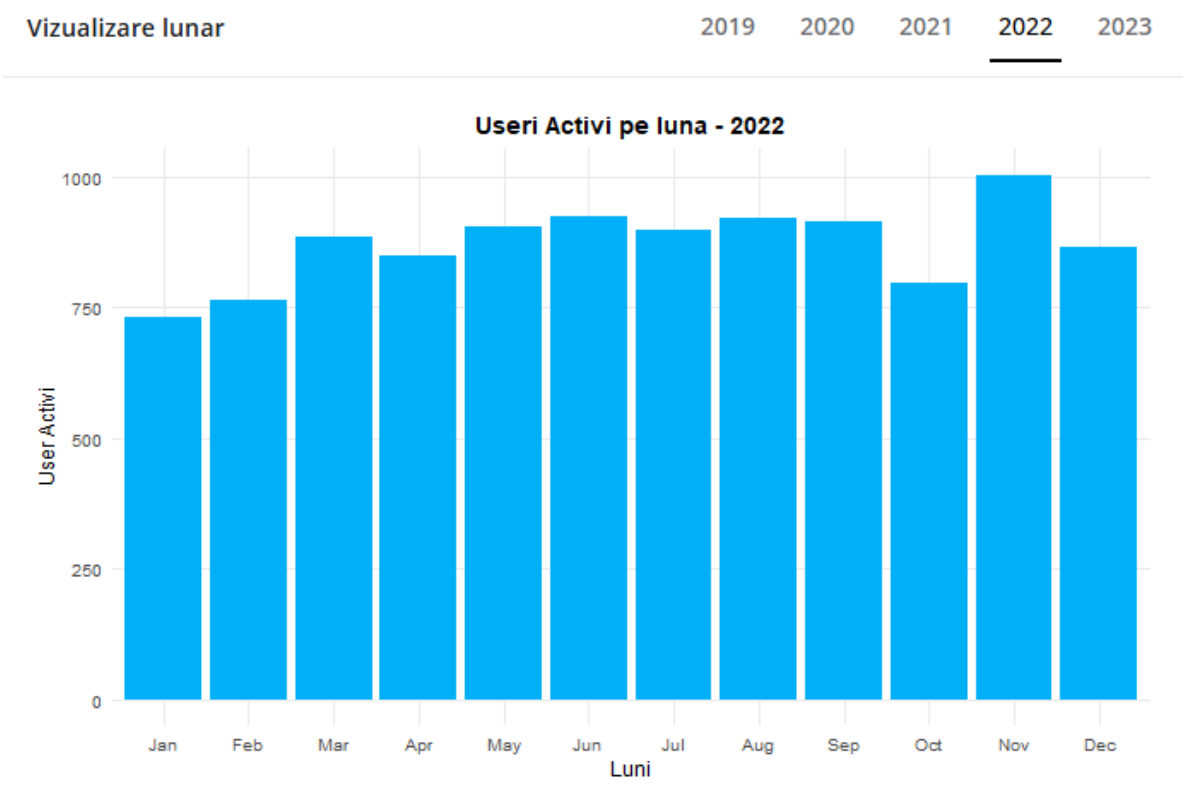


Analiza graficului de comparație anuală evidențiază comportamente distincte pentru cele două distribuții. Distribuția Poisson, caracterizată prin egalitatea dintre medie și varianță și absența unei limite superioare, generează o evoluție a traficului cu fluctuații zilnice accentuate și vârfuri sporadice foarte ridicate, vizibile în special în perioadele cu medii mari, anul 2020. Acest model este adecvat pentru sisteme scalabile cu potențial nelimitat.

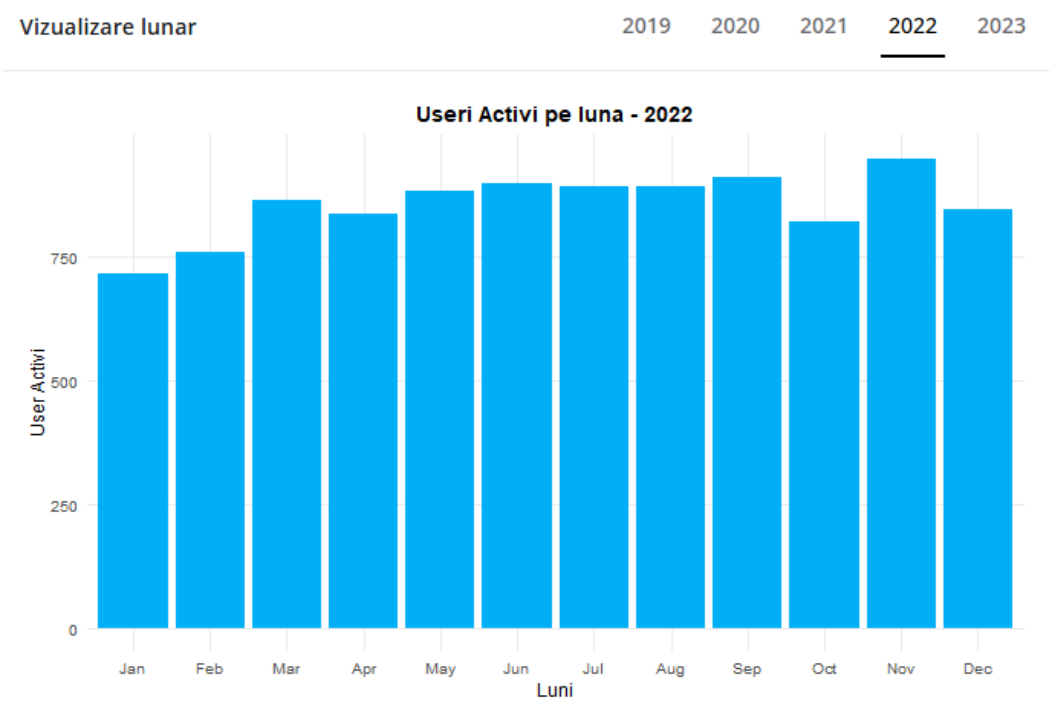


În schimb, distribuția binomială, care are varianța mai mică decât media și este mărginită superior de parametrul KDetails$max_active_users, produce o serie mult mai stabilă și netedă, fără salturi extreme. Ea reflectă realist constrângerile unui sistem cu capacitate fixă, unde numărul de utilizatori simultani nu poate depăși un anumit plafon tehnic sau de piață.

2. Analiza pe luni



Unele luni prezintă creșteri semnificative ale traficului datorită sezonalității și evenimentelor speciale, iar distribuția Poisson accentuează aceste variații, generând vârfuri mai dramatice și oscilații mai mari între luni.



În distribuția binomială, aceleași efecte sezoniere sunt vizibile, dar variațiile lunare sunt mai atenuate și traficul rămâne mai stabil, datorită plafonului maxim de utilizatori.

1.3. Estimarea empirică a mediei și variației traficului pentru fiecare an și compararea cu valorile teoretice.

```
dau_empiricalYearStudy <- function(nsim = 5000, year = 2019) {
  stopifnot(year %in% KDetails$years)
```

```
  yearIndex <- which(KDetails$years == year)
  sim <- replicate(nsim, {
    simulateUserYear(year)
  })
  daily_means <- rowMeans(sim)
  daily_vars  <- apply(sim, 1, var)

  dataFrame <- list()
  dataFrame$Mean_Empirical = mean(daily_means)
  dataFrame$Variance_Empirical = mean(daily_vars)
  if (KDetails$distribution == "pois") {
    dataFrame$Mean_Theoretical = mean(KDetails$lambdaDistribution[,yearIndex])
    dataFrame$Variance_Theoretical = mean(KDetails$lambdaDistribution[,yearIndex])
  }
  else {
    dataFrame$Mean_Theoretical = mean(KDetails$max_active_users * KDetails$procDistribution[,yearIndex])
    dataFrame$Variance_Theoretical = mean(KDetails$max_active_users *
                                          KDetails$procDistribution[,yearIndex] *
                                          (1 - KDetails$procDistribution[,yearIndex]))
  }
  dataFrame
}
```

Funcția dau_empiricalYearStudy realizează un studiu empiric de validare pentru un an specific, executând un număr mare de simulări ale traficului zilnic și comparând statisticile rezultate, media și varianța, cu valorile teoretice așteptate, bazate direct pe parametrii distribuției alese, Poisson sau binomială.

În modelul Poisson, media și varianța empirică sunt practic egale, confirmând atât implementarea corectă, cât și proprietatea statistică a acestei distribuții.

Pentru modelul binomial, media empirică se apropie de cea teoretică, în timp ce varianța observată este semnificativ mai mică, ilustrând reducerea naturală a variabilității datorită existenței unui plafon maxim de utilizatori.

Mean_Empirical	Variance_Empirical
673.5912575342466	672.3824744223675

Mean_Theoretical	Variance_Theoretical
673.691186580957	673.691186580957

1.4 Interpretarea diferențelor între modele, trafic redus vs plafonat

Modelul Poisson descrie un trafic „neplafonat”, cu fluctuații mari și vârfuri puternice, potrivit pentru situații cu cerere rară sau scalabilă. Distribuția binomială impune un plafon maxim de utilizatori, reducând volatilitatea și făcând traficul mai stabil, realist pentru sisteme cu capacitate limitată.

2.1. Modelarea lui S folosind o distributie asimetrica(exponențială) și o distributie normala

S_i este variabila aleatoare care reprezintă timpul de răspuns la încercarea i și este modelată fie cu distribuție exponențială, asimetrică; fie normală, simetrică.

```
SDetails <- new.env()
SDetails$distribution = "exp" # sau "norm"
SDetails$mean = 30
SDetails$variation = 25                          # norm
SDetails$std_variation = sqrt(SDetails$variation) # norm
SDetails$rate = 1 / SDetails$mean
```

Funcția generatoare rResponseTime permite simularea Monte Carlo a timpii S_i:

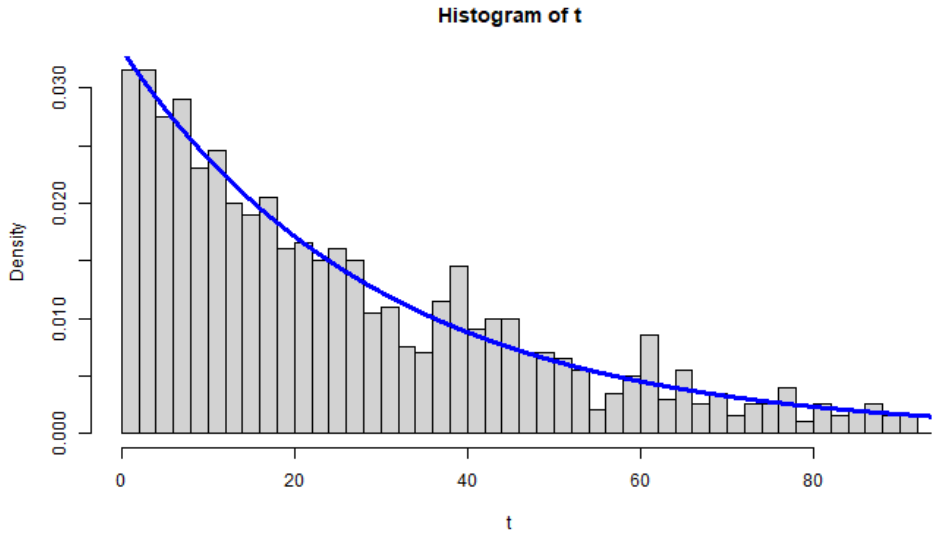
```
rResponseTime <- function(n) {
  stopifnot(all(n > 0))
  if (SDetails$distribution =="exp") {
    rexp(n, SDetails$rate)
  }
  else if (SDetails$distribution == "norm") {
    rnorm(n, SDetails$mean, SDetails$std_variation)
  }
}
```

2.2. Construirea histogramelor lui S și suprapunerea densităților teoretice

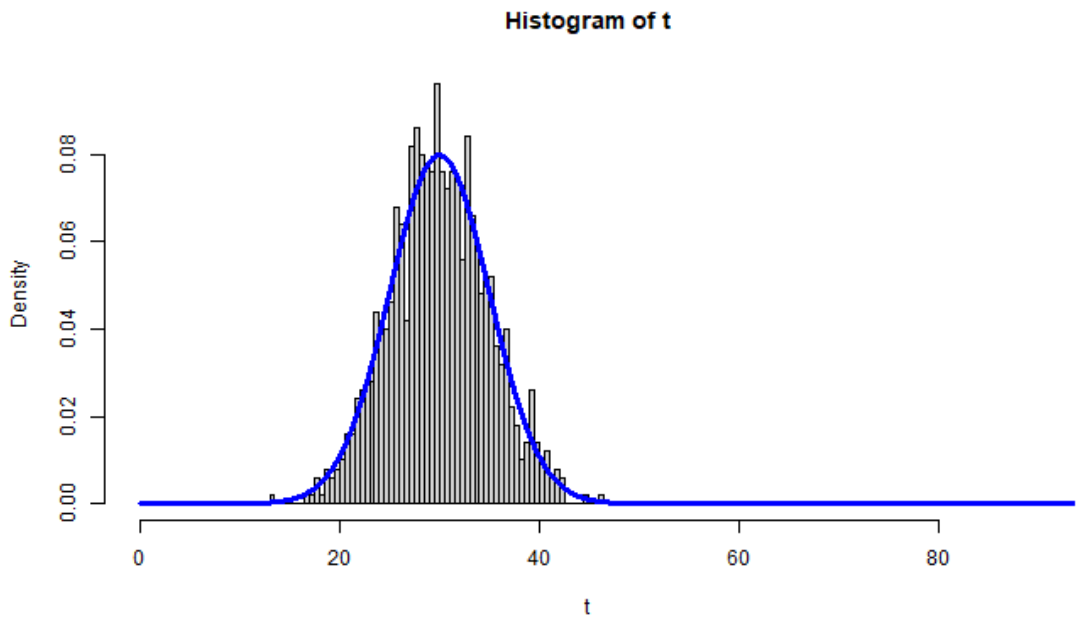
```
empSTheoreticalGraph <- function (tries, distr = NULL) {
   if (!is.null(distr))
    SDetails$distribution = distr
  t <- rResponseTime(tries)
  tsq <- seq(0, 100, by = 0.01)
  hist(t, prob=TRUE, breaks = 90, xlim=c(0, 90))
  lines(tsq, dResponseTime(tsq), col="blue", lwd=3)
}
```

Funcția empSTheoreticalGraph generează un eșantion simulat pentru timpul de răspuns S_i, reprezintă histograma sa normalizată și suprapune densitatea teoretică corespunzătoare, permițând o comparație vizuală directă între distribuția empirică și cea teoretică.

Histograma Empirica peste Teoretica



Histograma Empirica peste Teoretica



2.3. Calcularea mediei, variantei, medianei, valorii modale și interpretarea acestora

```r
empSStudy <- function(nsim = 1000) {
  sim <- rResponseTime(nsim)
  h <- hist(sim, breaks = nim, plot = FALSE)
  mode_estimate <- h$mids[which.max(h$density)]
  data.frame(
    Mean_Empirical       = mean(sim),
    Variance_Empirical   = var(sim),
    Median_Empirical = median(sim),
    Modal_Empirical = mode_estimate
  )
}
```

Funcția empSStudy generează eșantioane simulate ale timpului de răspuns S_i și calculează statisticile empirice de bază: media, varianța, mediana și modulul.

```r
getDTempSStudy <- function (nsim = 1000) {
  datatable(
    empSStudy(nsim),
    rownames = FALSE,
    options = list(
      dom = "t"
    )
  )
}
```

Funcția getDTempSStudy transformă rezultatele empirice ale simulării într-un tabel interactiv si ușor de citit.

```r
getDTthSStudy <- function () {
  dt <- data.frame(
```

```
    Mean_Theoretical     =
      if (SDetails$distribution == "norm")
        SDetails$mean
      else
        1 / SDetails$rate
    ,
    Variance_Theoretical =
      if (SDetails$distribution == "norm")
        SDetails$variation
      else
        1 / (SDetails$rate ^ 2)
    ,
    Median_Theoretical = qResponseTime(1/2),
    Modal_Theoretical =
      if (SDetails$distribution == "norm")
        SDetails$mean
      else
        0
  )
  datatable(
    dt,
    rownames = FALSE,
    options = list(
      dom = "t"
    )
  )
}
```

Funcția getDTthSStudy furnizează valorile teoretice așteptate pentru aceleași statistici; media, varianța, mediana, modulul; oferind un punct de referință pentru compararea cu rezultatele simulate.

2.4. Discutarea diferenței dintre medie și mediană în contextul latențelor

Într-o distribuție normală, cele trei măsuri centrale coincid (medie = mediană = mod), indicând o formă simetrică și fără aberații extreme, iar latențele se distribuie echilibrat în jurul valorii medii.

În distribuția exponențială, cele trei măsuri diferă, medie > mediană > modul, dezvăluind o asimetrie pronunțată cu coadă lungă spre valori mari, asta reflectă existența unui număr mic de latențe extreme care ridică media, în timp ce majoritatea cererilor răspund rapid, așa cum arată mediana.

Diferența între medie și mediană oferă o măsură a impactului valorilor extreme: cu cât această diferență este mai mare, cu atât coada distribuției este mai lungă și cu atât media este mai mult influențată de cazurile rare, dar foarte lente.

3.1. Estimarea empirică a: P(A), P(B), P(C), P(A∩ B), P(AUD)

1. Distribuția lui N

Numărul de încercări necesare pentru finalizarea unei cereri este modelat printr-o distribuție geometrică trunchiată, cu un număr maxim de încercări N_{max}.

```
NDetails = new.env()
NDetails$distribution = "geom"
```

```
NDetails$NMAX = 5
probEsec <- 0.4
probSucc <- 1 - probEsec
```

Distribuția este geometrică cu probabilitate de succes constantă, dar modificată la $N=N_{max}$, unde sunt incluse atât cazul de succes la ultima încercare, cât și cazul de eșec definitiv. Valorile posibile ale lui N sunt: $N \in \{1,2,...,N_{max}\}$

Distribuția lui N este definită astfel:

$$P(N=n)=\begin{cases}(1-p)^{n-1}p & ,n<N_{max}\\ \\ (1-p)^{Nmax-1} & ,n=N_{max}\end{cases}$$

Această formulare reflectă faptul că, pentru $n=N_{max}$, sunt agregate atât succesul la ultima încercare, cât și eșecul definitiv.

Pentru simulare, distribuția teoretică este discretizată explicit prin funcția de masă dRetry, iar valorile lui N sunt generate prin eșantionare.

```
dRetry <- function(n) {
  if (length(n) > 1)
  {
    sapply(n, FUN=dRetry)
  }
  else {
    stopifnot(n >= 1)
    if (n > NDetails$NMAX)
    {
      return(0)
    }
    else
    {
      if (n < NDetails$NMAX)
      {
        dgeom(n - 1, probSucc)
      }
      else
      {
        probEsec ^ (NDetails$NMAX - 1)
      }
    }
  }
}
```

Vectorul de probabilități asociat valorilor posibile ale lui N este:

```
.d_prob_vec = dRetry(1:NDetails$NMAX)
```

Generarea efectivă a variabilei aleatoare N se face cu funcția:

```
rRetry <- function(n) {
```

```
  stopifnot(n >= 1)
  sample(1:NDetails$NMAX, n, replace=T, prob=.d_prob_vec)
}
```

2. Backoff-ul

```
backoff <- function(i) {
  stopifnot(all(i > 0))
  sapply(i, function(x) {
    if (x >= NDetails$NMAX)
      0
    else
      exp(x)
  })
}
```

Backoff-ul reprezintă întârzierea între două încercări succesive, adică timpul de așteptare înainte de a face următoarea încercare după un eșec. Funcția backoff(i) calculează întârzierea (timpul de așteptare) pentru încercarea i, iar valoarea întârzierii crește exponențial odată cu numărul încercării:

$$\text{backoff}(i) = e^{i}, \text{ pentru } i<N_{max}$$

3. Distributia lui T

Variabila aleatoare T se obține ca o sumă de variabile aleatoare condiționate, fiecare corespunzând unui număr fix de încercări N=n, ponderată cu probabilitatea P(N=n).

Distribuția lui T se obține prin legea probabilităților totale:

$$f_T(t) = \sum_{n=1}^{Nmax} f_{T|N=n} \cdot P(N=n)$$

$$P(T \leq t) = \sum_{n=1}^{Nmax} P(T \leq t \mid N=n) \cdot P(N=n)$$

T|N=n este timpul total condiționat de numărul fix de încercări n, definit ca:

$$T|N=n = \sum_{i=1}^{n-1} (S_i + \text{backoff}(i)) + S_n$$

Funcția pRequestTimeWithNTries, calculează probabilitatea ca timpul total de procesare pentru exact n încercări să fie mai mic sau egal cu o valoare dată time. Acest timp total este compus din suma celor n timpi de răspuns individuali și a întârzierilor de backoff aferente primele n-1 încercări.

```
rRequestTimeWithNTries <- function(n = 1, tries) {
  stopifnot(all(n <= NDetails$NMAX))
  stopifnot(all(n > 0))
  stopifnot(tries > 0)
  if (.lastNMAX != NDetails$NMAX) {
    .b_vec = c()
    for (i in 1:NDetails$NMAX) {
      if (i != 1) {
        .b_vec = c(.b_vec, sum(backoff(1:(i - 1))))
      }
      else {
        .b_vec = c(.b_vec, 0)
      }
```

```
    }
    .lastNMAX = NDetails$NMAX
  }
  if (SDetails$distribution == "norm")
    vapply(n, function (no) {
      rnorm(tries, mean = no * SDetails$mean, sd = sqrt(no) * SDetails$std_variation) + .b_vec[no]
    }, numeric(tries))
  else
    vapply(n, function (no) {
      rgamma(tries, no, SDetails$rate) + .b_vec[no]
    }, numeric(tries))
}
```

Funcția de simulare a lui T:

```
rRequestTime <- function(tries) {
  stopifnot(tries > 0)
  n <- rRetry(tries)
  vapply(n, function(no) {
    rRequestTimeWithNTries(no, 1)
  }, numeric(1))
}
```

Funcția rRequestTime generează valori aleatoare pentru timpul total de procesare (T) până la succes sau abandon, prin mai întâi generarea numărului de încercări N cu rRetry, iar apoi calcularea timpului total condiționat de fiecare valoare a lui N folosind rRequestTimeWithNTries.

4. Setarea parametrilor și simularea datelor

```
nsim <- 1e5
t0 <- 20
n0 <- 3
```

Se generează un număr mare de observații pentru a asigura stabilitatea estimărilor.

```
N_sim <- rRetry(nsim)
T_sim <- rRequestTime(nsim)
```

Fiecare rulare a simulării generează următoarele valori: numărul de repetări necesare și durata completă a procesului.

5. Definirea evenimentelor

```
A <- (N_sim < NDetails$NMAX | (sample(c(F, T), prob = c(probEsec, probSucc))))
B <- (T_sim <= t0)
C <- (N_sim <= n0)
D <- (N_sim >= 2)
```

În analiza de performanță, am definit patru evenimente cheie: **A**, finalizarea cu succes, marcată de procesarea completă a cererii; **B**, conformitatea SLA, adică respectarea limitei de timp contractuale; **C**, numărul redus de retry-uri, atunci când cererea necesită mai puține reîncercări decât un prag stabilit; și **D**, retry multiplu, definit ca necesitatea a cel puțin două încercări de procesare.

6. Estimarea probabilităților empirice

```
Prob_A_emp <- mean(A)
```

```
Prob_B_emp <- mean(B)
Prob_C_emp <- mean(C)
Prob_AintB_emp <- mean(A & B)
Prob_AunionD_emp <- mean(A | D)
```

Probabilitățile sunt estimate ca frecvențe relative ale evenimentelor în eșantionul simulat.

3.2. Verificarea numerică a formulelor pentru reuniune/intersecție

Pentru două evenimente X și Y, formula teoretică a reuniunii este:

$$P(X \cup Y) = P(X) + P(Y) - P(X \cap Y)$$

Aceasta este verificată numeric pentru perechile de evenimente analizate.

```
Prob_A_intersect_B_empirical <- mean(A & B)
Prob_A_union_B_empirical <- mean(A | B)
Prob_A_union_B_formula  <- mean(A) + mean(B) - Prob_A_intersect_B_empirical
Prob_A_intersect_D_empirical <- mean(A & D)
Prob_A_union_D_empirical <- mean(A | D)
Prob_A_union_D_formula  <- mean(A) + mean(D) - Prob_A_intersect_D_empirical
```

Valorile obținute prin calcul direct și prin formulă sunt foarte apropiate, confirmând corectitudinea relațiilor teoretice pentru reuniune și intersecție.

3.3. Explicarea motivului pentru care probabilitatea empirică aproximează bine probabilitatea teoretică.

Probabilitățile empirice estimate prin simulare se apropie de valorile teoretice datorită Legii Numerelor Mari. Pe măsură ce numărul de simulări crește, frecvențele relative ale evenimentelor converg către probabilitățile reale ale procesului stochastic.

În acest exercițiu, utilizarea unui eșantion mare, n=100000, reduce fluctuațiile aleatoare și conduce la estimări stabile și precise. Astfel, simularea Monte Carlo devine un instrument eficient pentru validarea numerică a relațiilor probabilistice, chiar și în situații în care calculul analitic este dificil sau intractabil.

4.1. Distribuția comună empirică a variabilei bidimensionale (N,F)

Folosim NFDetails$rep_N și NFDetails$rep_F pentru a genera simulări ale perechilor (N,F), unde N este numărul total de încercări până la succes/abandon și F este numărul de eșecuri înainte de succes sau abandon.

```
NFDetails <- new.env()
NFDetails$n_sim <- 1e5
NFDetails$rep_N <- rRetry(NFDetails$n_sim)
NFDetails$rep_F <- sapply(NFDetails$rep_N, function(n) {
  if (n < NDetails$NMAX)
    n - 1
  else
    n - sample(c(0, 1), 1, prob = c(probSucc, probEsec))
})
NFDetails$p_table <- prop.table(
  table(NFDetails$rep_N, NFDetails$rep_F)
)
```

Distribuția comună empirică NFDetails$p_table afișează probabilitatea fiecărei combinații (N, F) obținută din simulări, oferind o imagine clară a relației dintre numărul total de încercări și cel al eșecurilor.

4.2. Distribuțiile marginale alei lui N si F

```
NFDetails$dist_N <- rowSums(NFDetails$p_table)
NFDetails$dist_F <- colSums(NFDetails$p_table)
```

Distribuția marginală a lui N, NFDetails$dist_N, este calculată ca suma probabilităților pe fiecare rând din tabela comună, arătând probabilitatea fiecărui număr total de încercări, indiferent de câte eșecuri au avut loc.
Distribuția marginală a lui F, NFDetails$dist_F, este calculată ca suma probabilităților pe fiecare coloană, arătând probabilitatea fiecărui număr de eșecuri, indiferent de câte încercări totale au fost necesare.

4.3. Testul empiric de independenta pentru (N,F)

```
NFDetails$test_independence <- function(tolerance = 1e-6) {
  P_indep <- outer(NFDetails$dist_N, NFDetails$dist_F)
  max(abs(NFDetails$p_table - P_indep)) < tolerance
}
indep_result <- NFDetails$test_independence()
```

Funcția NFDetails$test_independence() verifică dacă N și F sunt empiric independente, comparând distribuția comună simulată cu cea teoretică obținută ca produsul distribuțiilor marginale.

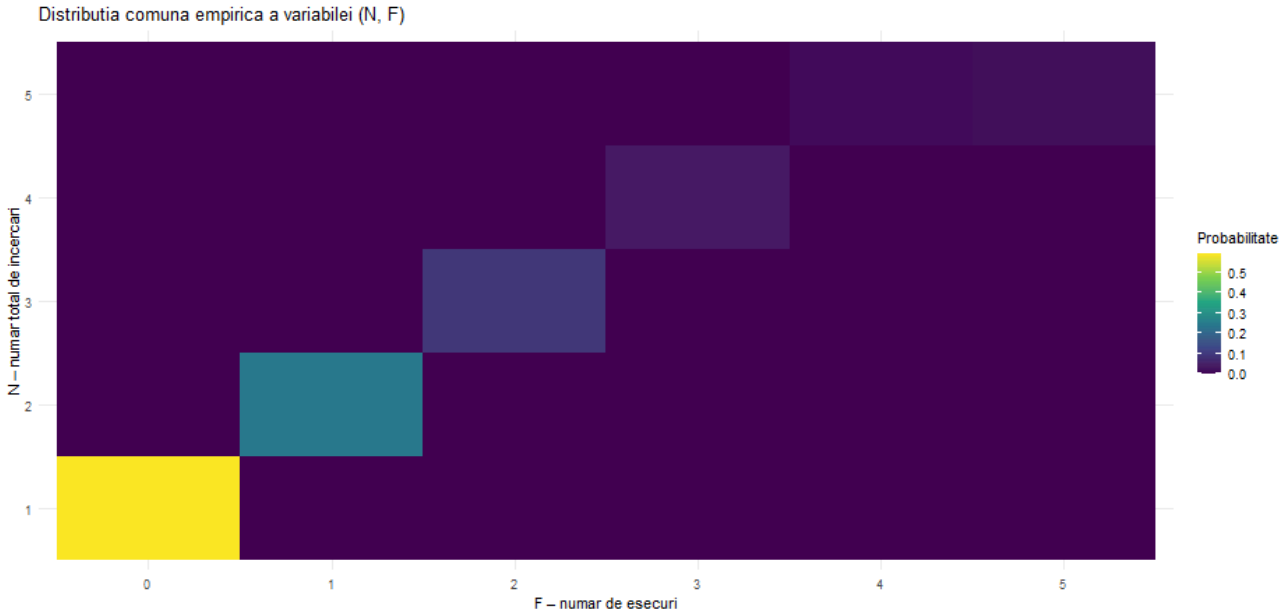
Testul calculează diferența maximă dintre cele două distribuții și o compară cu un prag de toleranță (de exemplu, 1e-6), returnând TRUE dacă diferența este nesemnificativă, independență acceptată, sau FALSE în caz contrar.

4.4. Modalitatea de vizualizare(tabel/heatmap) si intepretarea (N,F)

Heatmap-ul generat de NFDetails$gen_heatmap() oferă o reprezentare vizuală directă a distribuției comune empirice (N, F), unde axele corespund variabilelor; X = F, Y = N; iar culorile indică probabilitățile combinațiilor.

```
NFDetails$gen_heatmap <- function() {
  df <- as.data.frame(NFDetails$p_table)
  colnames(df) <- c("N", "F", "prob")
  ggplot(df, aes(x = F, y = N, fill = prob)) +
    geom_tile() +
    scale_fill_viridis_c() +
    labs(
      title = "Distributia comuna empirica a variabilei (N, F)",
      x = "F – numar de esecuri",
      y = "N – numar total de incercari",
      fill = "Probabilitate"
    ) +
    theme_minimal()
}
NFDetails$gen_heatmap()
```

Vizualizarea evidențiază rapid structura dependenței dintre cele două variabile: zonele cu culoare intensă arată combinații frecvente, în timp ce zonele deschise indică scenarii extreme, mult mai puțin probabile.

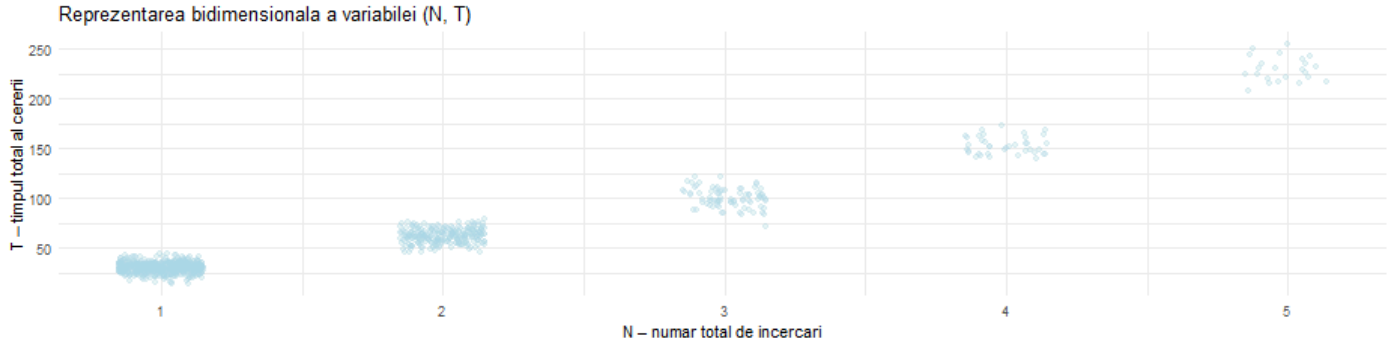


5.1. Reprezentați grafic variabila bidimensională (N,T)

Graficul de dispersie afișează relația dintre numărul total de încercări (N) și timpul total de procesare (T) obținut prin simulare, punctele fiind ușor dispersate și transparente pentru a reduce suprapunerea și a evidenția zonele de densitate ridicată.

```
ggplot(NTDetails$df_val, aes(x = N, y = T)) +
  geom_jitter(alpha = 0.25, width = 0.15, color = "lightblue") +
  labs(
    title = "Reprezentarea bidimensionala a variabilei (N, T)",
    x = "N - numar total de incercari",
    y = "T - timpul total al cererii"
  ) +
  theme_minimal()
```

Această vizualizare permite observarea tendinței: cum crește numărul de retry-uri, N, timpul total, T, tinde să crească, iar distribuția punctelor relevă variabilitatea latenței pentru fiecare valoare a lui N.



5.2. Calcularea mediilor, variațelor , covarianței și a coeficientul de corelație pentru (N,T)

```
NTDetails$stats <- data.frame(
  mean_N = mean(NTDetails$rep_N),
  mean_T = mean(NTDetails$rep_T),
  var_N  = var(NTDetails$rep_N),
  var_T  = var(NTDetails$rep_T),
  covariance = cov(NTDetails$rep_N, NTDetails$rep_T),
  correlation = cor(NTDetails$rep_N, NTDetails$rep_T)
)
```

mean_N	mean_T	var_N	var_T	covariance	correlation
1.635	54.29464276029983	0.9307057057057057	3503.797516209473	40.31993742077032	0.7060641499728691

Funcțiile mean() și var() calculează media și dispersia variabilelor N și T , oferind o imagine a tendinței centrale și a variabilității pentru fiecare.

Funcțiile cov() și cor() calculează covarianța și coeficientul de corelație Pearson între N și T, măsurând gradul și modul în care cele două variabile variază împreună pe ansamblul datelor simulate.

5.3. Interpretarea corelației

O corelație pozitivă între N și T indică o tendință generală: un număr mai mare de retry-uri duce, în medie, la un timp total de procesare mai mare.

Această relație este logică, deoarece timpul total, T, este suma dintre timpii de răspuns și întârzierile introduse de backoff pentru fiecare încercare; prin urmare, creșterea lui N adaugă, în general, mai mulți termeni pozitivi sumei.

Corelația nu este perfectă datorită variabilității naturale: atât timpii de răspuns individuali, cât și șansa de succes sunt variabile aleatoare, astfel că pentru același număr de încercări, N, timpul total, T, poate diferi semnificativ între simulări.

6.1. Estimarea $P(A|N \leq n_0)$, $P(B|A)$

1. Setarea parametrilor și numărul de simulări

```
NDetails$NMAX <- 5
SDetails$distribution <- "exp" # sau "norm"
t0 <- 150  # SLA
n0 <- 5    # Prag pentru „putine incercari"
n_sim <- 100000
```

Pentru analiză se definesc pragul de timp SLA, t_0, pragul pentru numărul de încercări, n_0, și un număr mare de simulări pentru a obține estimări stabile ale probabilităților.

2. Generarea datelor simulate

```
n_vec <- rRetry(n_sim)
t_vec <- rRequestTimeWithNTries(n_vec, 1) # generam cate 1 timp pt fiecare n din vector
```

Rezultatul fiecărei simulări (succes sau eșec) este determinat după generarea numărului de încercări și a timpului total de procesare: este succes dacă numărul de încercări este mai mic decât NMAX și este stocastic – stabilit de probabilitățile probSucc și probEsec – dacă se atinge limita maximă de NMAX încercări.

3. Definirea evenimentelor

```
n_sim <- 100000
n_vec <- rRetry(n_sim)
is_success <- rep(TRUE, n_sim)
mask_limit <- (n_vec == NDetails$NMAX)
outcome_limit <- rbinom(sum(mask_limit), 1, probSucc)
is_success[mask_limit] <- (outcome_limit == 1)
Event_A <- is_success
```

```
Event_B <- (t_vec <= t0)
Event_C <- (n_vec <= n0)
```

Se definesc trei evenimente cheie: A, cererea este finalizată cu succes; B, timpul total de procesare respectă limita SLA; C, cererea necesită un număr redus de reîncercări.

4.Estimarea probabilităților condiționate

```
p_A_given_C <- calc_cond_prob(Event_C, Event_A)
p_B_given_A <- calc_cond_prob(Event_A, Event_B)
```

P(A | N ≤ n_0) estimează șansa ca o cerere să fie finalizată cu succes, cu condiția că necesită puține reîncercări. P(B | A) estimează probabilitatea respectării SLA, cu condiția că cererea a fost deja încheiată cu succes.

6.2. Calcularea E[T|I=1], E[T|I=0]

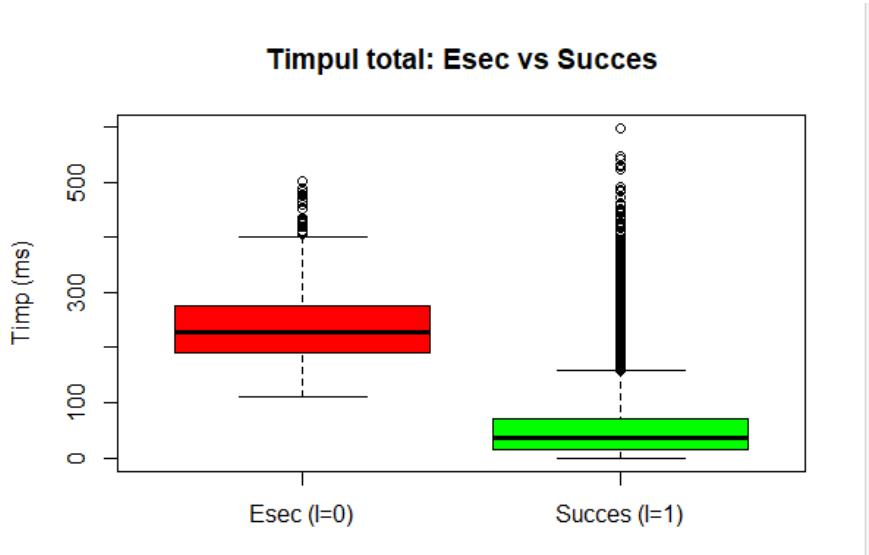
```
mean_T_success <- mean(t_vec[is_success])
mean_T_fail <- mean(t_vec[!is_success])
```

Se calculează timpul mediu total pentru cererile reușite și, respectiv, pentru cele eșuate, oferind o măsură a costului temporal diferit dintre cele două tipuri de rezultate.

6.3. Interpretarea rezultatelor din perspectiva experienței utilizatorului.

Rezultatele arată că cererile finalizate cu succes au, în medie, un timp total semnificativ mai mic decât cele care se termină cu eșec. Cererile eșuate sunt asociate cu un număr mai mare de retry-uri și cu acumularea tuturor întârzierilor aferente, ceea ce conduce la un timp total mult mai ridicat.

Din perspectiva experienței utilizatorului, acest lucru este critic: utilizatorii care nu obțin serviciul așteptat sunt și cei care așteaptă cel mai mult. Astfel, eșecul nu implică doar pierderea funcționalității, ci și un cost temporal ridicat, afectând direct satisfacția și percepția calității serviciului.



Boxplot-ul confirmă vizual această diferență, evidențiind o separare clară între distribuțiile timpului total pentru succes și eșec.

7.1. Simulați a două scenarii: S_i timpi independenți vs dependenți (latența crește după eșecuri).

```
sim_T_independence <- function() {
  as.numeric(rRequestTime(1))
```

```
}
```

Scenariul independent presupune că fiecare timp de răspuns S_i este generat independent, cu aceeași distribuție și medie fixă, iar funcția sim_T_independence() calculează timpul total ca o simplă sumă a acestor valori independente.

```
sim_T_dependence <- function() {
   N <- rRetry(1)
  penalty_vector <- 1 + 0.5 * (0:(N - 1))
  response_times <- sapply(penalty_vector, function(penalty) {
    penalized_mean <- SDetails$mean * penalty
    if (SDetails$distribution == "exp") {
      rexp(1, rate = 1 / penalized_mean)
    } else {
      rnorm(1, mean = penalized_mean, sd = SDetails$std_variation)
    }
  })
  backoff_time <- if (N > 1)
    sum(backoff(1:(N - 1)))
  else
    0
  sum(response_times) + backoff_time
}
```

Scenariul dependent reflectă o penalizare progresivă: timpii de răspuns cresc odată cu numărul de încercări, iar funcția sim_T_dependence() ajustează media fiecărui S_i proporțional cu numărul de eșecuri anterioare, modelând o creștere sistematică a latenței.

```
  T_indep <- replicate(nsim, sim_T_independence()) #nsim=5000
  T_dep   <- replicate(nsim, sim_T_dependence())  #nsim=5000
```

7.2. Compararea distribuțiilor și variantelor în cele două scenarii.

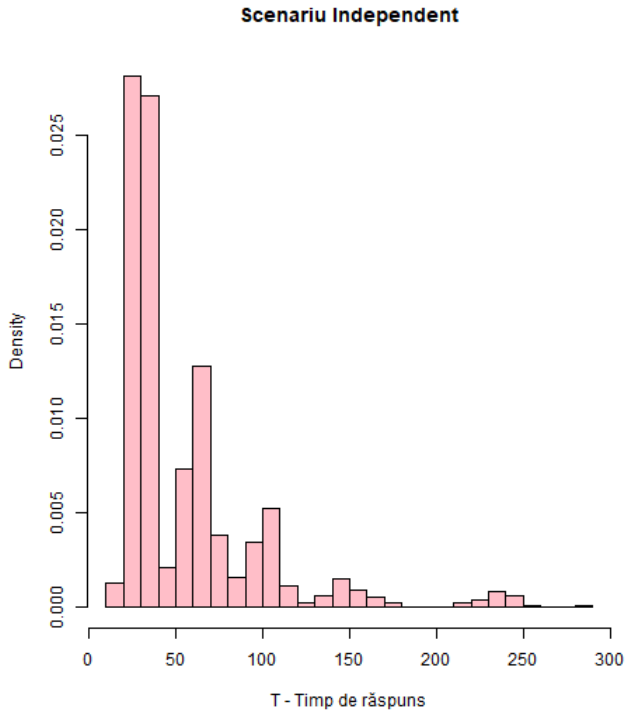
Pentru ambele scenarii, independent și dependent, se generează același număr de simuari, nsim=5000, iar media și varianța timpului total T sunt calculate pentru a compara distribuțiile rezultate.

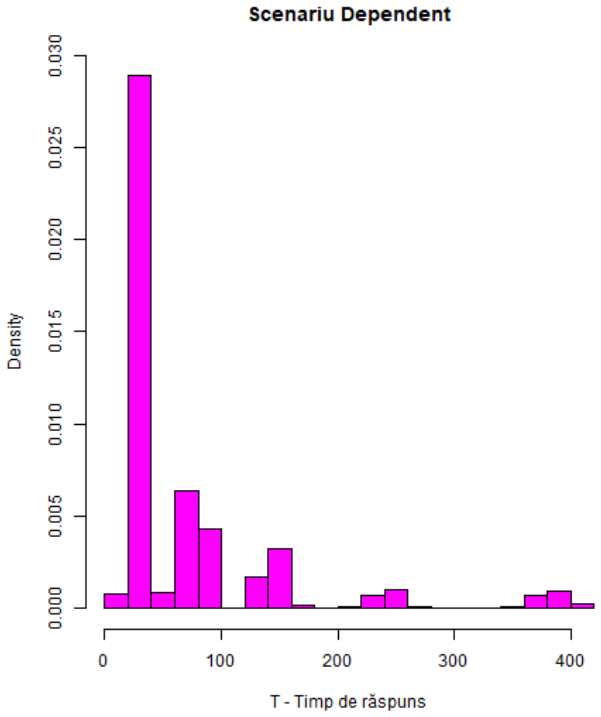
Scenariul independent, T_indep, produce o distribuție mai concentrată, cu medie și varianță relativ scăzute, în timp ce scenariul dependent,T_dep, generează o distribuție mult mai răspândită, cu medie mai mare, varianță mai ridicată și cozi mai lungi, reflectând efectul cumulativ al penalizărilor progresive.

```
stats <- data.frame(
    Scenariu = c("Independent", "Dependent"),
    Medie = c(mean(T_indep), mean(T_dep)),
    Varianta = c(var(T_indep), var(T_dep))
  )
  par(mfrow = c(1, 2))
  hist(T_indep, freq = FALSE, breaks = 20,
       col = "pink",
       main = "Scenariu Independent",
       xlab = "T - Timp de răspuns",
       border = "black")
  hist(T_dep, freq = FALSE, breaks = 20,
```

```
      col = "magenta",
      main = "Scenariu Dependent",
      xlab = "T - Timp de răspuns",
      border = "black")
par(mfrow = c(1, 1))
```

Suprapunerea histogramelelor evidențiază clar aceste diferențe: distribuția pentru scenariul dependent este deplasată spre valori mai mari și prezintă o variabilitate mult mai amplă decât cea a scenariului independent. În imaginile de mai jos, este reprezentată histograma lui S_i în cazul normal.





7.3.Formularea unor concluzii privind riscul și stabilitatea sistemului.

Scenariul dependent, în care latența crește progresiv cu fiecare retry, indică un risc crescut de performanță degradată și instabilitate a sistemului, deoarece eșecurile sucesive amplifică timpul de răspuns într-un mod neliniar.

Scenariul independent, cu latențe constante și independente statistic, oferă o comportare mult mai predictibilă și stabilă, limitând variația extremă a timpului total.

Această comparație evidențiază importanța critică a modelării corecte a dependențelor între încercări atunci când se simulează performanța sistemelor reale sau se planifică capacitatea, deoarece ignorarea acestor dependențe poate duce la subestimarea severă a latențelor în cazul eșecurilor repetate.

8.1. Verificarea numerică a inegalităților Markov și Cebîșev (empiric versus teoretic) pentru T≥0

1. Estimarea empirică și teoretică a mediei

```
ET_empiric <- mean(T_simulations)
ET_theretic <- integrate(
    function(x) x * dRequestTime(x),
    lower = 0, upper = Inf
  )$value
```

Media timpului de răspuns este estimată atât empiric, prin simulare, cât și teoretic,prin integrarea densității. Aceste valori sunt utilizate în limita Markov.

2. Probabilitatea empirică și teoretică P(T≥t)

```
P_empiric <- function(t) { mean(T_simulations >= t) }
P_theoretic  <- function(t) { (1 - pRequestTime(t)) }
```

3. Inegalitatea lui Markov

Pentru o variabilă aleatoare pozitiva T≥0, inegalitatea lui Markov este:

$$P(T \geq t) \leq \frac{E[T]}{t}, t>0$$

```
Markov_empiric <- function(t) { min(ET_empiric / t, 1) }
Markov_theretic  <- function(t) { min(ET_theretic  / t, 1) }
```

4. Verificare numerică

```
t_values <- seq(50, 200, by = 10)
InegDetails$markov_study <- data.frame(
    t = t_values,
    P_empirical   = sapply(t_values, P_empiric),
    P_theoretical = sapply(t_values, P_theoretic),
    Markov_empiric = sapply(t_values, Markov_empiric),
    Markov_theretic  = sapply(t_values, Markov_theretic)
  )
```

Rezultatele numerice confirmă validitatea inegalității lui Markov, însă limita superioară este relativ largă, în special pentru valori mici ale pragului T. Această caracteristică reflectă natura conservatoare a inegalității, care utilizează doar media variabilei aleatoare.

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

Markov

t	P_empirical	P_theoretical	Markov_empiric	Markov_theretic
50	0.3856	0.3866127399576912	1	0.98413531976043
60	0.32103	0.3215734184056427	0.9076049495343168	0.820112766467025
70	0.26941	0.2681036654613764	0.777947099600843	0.7029537998288786
80	0.22454	0.2240904690275795	0.6807037121507375	0.6150845748502688
90	0.18735	0.187833532820265	0.6050699663562112	0.54674184431135
100	0.15732	0.1579572894121927	0.54456296972059	0.492067659880215
110	0.13333	0.1333193756474718	0.4950572452005365	0.4473342362547409
120	0.1132	0.1129506720378139	0.4538024747671584	0.4100563832335125
130	0.09571	0.09603468008740634	0.4188945920927616	0.3785135845232423

5. Media și varianța empirică

```
mu_emp <- mean(T_simulations)
var_emp <- var(T_simulations)
```

6. Calcul teoretic al variantei

```
E_T2_theoretic <- integrate(
    function(x) x^2 * dRequestTime(x),
    lower = 0, upper = Inf
  )$value
var_th <- E_T2_theoretic - ET_theretic^2
sd_th  <- sqrt(var_th)
```

7. Probabilitatea empirică și teoretică a deviației

```
P_emp_dev <- function(r) mean(abs(T_simulations - mu_emp) >= r)
  P_th_dev <- function(r) {
    left  <- if (ET_theretic - r <= 0) 0 else pRequestTime(ET_theretic - r)
    right <- 1 - pRequestTime(ET_theretic + r)
    left + right
  }
```

8. Limita Cebîșev

Pentru o variabilă aleatoare T cu medie μ și varianță finită:

$$P(|T-\mu|\geq r)\leq\frac{Var(T)}{r^2}\ ,r>0$$

```
Cheb_emp_bound <- function(r) pmin(var_emp / (r^2), 1)
Cheb_th_bound  <- function(r) pmin(var_th  / (r^2), 1)
```

9. Verificare numerică

```
r_values <- seq(0.5, 3, by = 0.5) * sd_th
InegDetails$cheb_study <- data.frame(
    r = r_values,
    P_empirical   = sapply(r_values, P_emp_dev),
    P_theoretical = sapply(r_values, P_th_dev),
    Cheb_emp_bound = sapply(r_values, Cheb_emp_bound),
    Cheb_th_bound  = sapply(r_values, Cheb_th_bound)
)
```

Inegalitatea lui Cebîșev oferă limite mai strânse decât inegalitatea lui Markov atunci când deviația rrr este exprimată în multipli ai abaterii standard. Acest lucru se observă numeric prin apropierea limitelor Cebîșev de probabilitățile reale ale abaterilor mari.

Cebîsnev

r	P_empirical	P_theoretical	Cheb_emp_bound	Cheb_th_bound
24.26831171724738	0.66896	0.6326913867099668	1	1
48.53662343449476	0.25738	0.1774579973082125	1	1
72.80493515174214	0.09965	0.109293932771521	0.6142687366228917	0.4444444444444444
97.07324686898951	0.06842	0.07419148914105989	0.3455261643503766	0.25
121.3415585862369	0.0469	0.05084065255342363	0.2211367451842411	0.16
145.6098703034843	0.03199	0.03471402654404521	0.1535671841557229	0.1111111111111111

8.2.Pentru variabila număr de eșecuri/încercări verificați o inegalitate de tip Chernoff, pentru T≥0

1. Estimarea probabilităților

```
F_sims <- sapply(NFDetails$rep_N, function(n) {
   if (n < NDetails$NMAX)
     n - 1
   else
     n - sample(c(0, 1), 1, prob = c(probSucc, probEsec))
 })
```

Calculează numărul de eșecuri, F, asociat fiecărei cereri, pornind de la numărul total de încercări, N, presupunând că există cel mult un succes și restul sunt eșecuri.

```
P_emp_F_ge <- function(a) mean(F_sims >= a)
```

Probabilitatea ca numărul de eșecuri să depășească un prag a este estimată empiric prin simulare.

2. Inegalitatea Chernoff

Pentru orice t>0:

$$P(F \geq a) \leq E[etF]eta\frac{E[e^{tF}]}{e^{ta}}$$

```
MGF_F_emp <- function(t) mean(exp(t * F_sims))  #MGF empiric
 Chernoff_bound_empMGF <- function(t, a) MGF_F_emp(t) / exp(t * a) #Limita Chernoff
 P_emp_F_ge <- function(a) mean(F_sims >= a) #Probabilitatea empirica
 P_th_F_ge <- function(a) { #Probabilitatea teoretica
   if (a == NMAX)
     probEsec ^ NMAX
   else
     sum(dRetry((a+1):NMAX))
 }
```

3. Rezultatele sunt stocate in

```
a_vals <- 0:(NMAX)
 t_grid <- c(0.1, 0.3, 0.5, 0.8, 0.9)
 InegDetails$chernoff_table <- do.call(rbind, lapply(t_grid, function(t) {
   data.frame(
     t = t,
     a = a_vals,
     P_theoretical = sapply(a_vals, P_th_F_ge),
     P_empirical   = sapply(a_vals, P_emp_F_ge),
     Chernoff_bound = sapply(a_vals, function(a) Chernoff_bound_empMGF(t, a))
   )
 }))
```

Limita Chernoff este mai strânsă decât Markov și Chebyshev, mai ales pentru valori mari ale lui aaa, fiind utilă pentru evaluarea riscului de multe eșecuri consecutive.

Show 6 entries Search:

t	a	P_theoretical	P_empirical	Chernoff_bound
0.1	0	1	1	1.075423639897129
0.1	1	0.4	0.40386	0.973083549619352
0.1	2	0.16	0.16044	0.880482406570841
0.1	3	0.06400000000000002	0.06415	0.7966934273876478
0.1	4	0.0256	0.02566	0.7208780238036585
0.1	5	0.01024	0.01541	0.6522774097773674

Showing 1 to 6 of 30 entries Previous 1 2 3 4 5 Next

8.3. Interpretarea utilității acestor limite când distribuțiile exacte sunt necunoscute, pentru $T \geq 0$

Inegalitățile Markov, Chebyshev și Chernoff constituie un set de instrumente analitice fundamentale în teoria probabilităților și analiza algoritmilor, oferind estimări superioare ale probabilității ca o variabilă aleatoare să se abată semnificativ de la media sa, chiar atunci când distribuția exactă este necunoscută.

Markov, cea mai generală, folosește doar media; Chebyshev, mai precisă, include și varianța; iar Chernoff, cea mai puternică, se bazează pe funcția generatoare de momente și oferă limite exponențial descrescătoare pentru cozile distribuției. Împreună, ele oferă garanții robuste de tip worst-case, esențiale în demonstrații teoretice și evaluări practice în condiții de incertitudine.

8.4. Pentru o funcție convexă (ex.:) verificați numeric (inegalitatea lui Jensen), pentru $T \geq 0$

1. Pentru funcție convexă φ:

$$\varphi(E[T]) \leq E[\varphi(T)]$$

2. Verificarea numerică a $\varphi(x)=x^2$

```
phi_square_E_T <- (mean(T_simulations))^2
E_phi_square_T <- mean(T_simulations^2)
```

3. Verificarea numerică a $\varphi(x)=e^{\frac{x}{c}}$

```
scale_factor <- max(mean(T_simulations), 1)
phi_exp_E_T <- exp(mean(T_simulations) / scale_factor)
E_phi_exp_T <- mean(exp(T_simulations / scale_factor))
```

$\varphi(x) = x^2$

:

$$\varphi(\mathbb{E}(T)) = (\mathbb{E}(T))^2 = 2964.0221, \qquad \mathbb{E}(\varphi(T)) = \mathbb{E}(T^2) = 6219.957$$

$\varphi(x) = e^x$

:

$$\varphi(\mathbb{E}(T)) = \exp\left(\frac{\mathbb{E}(T)}{c}\right) = 2.7183, \qquad \mathbb{E}(\varphi(T)) = \mathbb{E}\left(\exp\left(\frac{T}{c}\right)\right) = 12.1167$$

unde

$$c = \max\{\mathbb{E}(T), 1\} = 54.4428$$

8.5. Interpretarea rezultatului de la d) în contextul riscului (penalizarea valorilor extreme)

Funcțiile convexe, prin proprietatea lor fundamentală, atribuie o penalizare disproporționată valorilor mari ale unei variabile aleatoare, cum ar fi timpul de răspuns, T . Aceasta reflectă realitatea economică și operațională, unde întârzierile extreme au un impact mult mai semnificativ decât abaterile moderate, o creștere mică a timpului de răspuns pentru valori mari produce o creștere mai mare a costului decât aceeași creștere pentru valori mai mici. În contextul managementului riscului și al acordurilor de nivel de serviciu, SLA, inegalitatea lui Jensen oferă o justificare riguroasă pentru utilizarea penalităților non-liniare, penalizând inegal valorile extreme și aliniind astfel modelul matematic la consecințele practice ale variației extreme.

9.1. Pentru sume/agregări zilnice (ex: total latență pe zi sau profit zilnic), studiați oportunitatea aproximării cu o distribuție normală prin simulare

1. Definirea zilei țintă și număr de simulări

```
AgregDetails = new.env()
AgregDetails$target_day <- 330
AgregDetails$target_year <- 2019
AgregDetails$n_sim <- 100
AgregDetails$vec <- c()
```

Aceasta setează ziua cu trafic mare, Black Friday și numărul de repetări pentru simulări.

2. Funcția helper pentru agregarea zilnică

```
sim_day_aggregate <- function(n) {
  k_users <- rActiveUsers(n, day = AgregDetails$target_day, year = AgregDetails$target_year)
  vapply(k_users, function(users){
    sum(rRequestTime(users))
  }, numeric(1))
}
```

Funcția rActiveUsers() generează numărul zilnic de utilizatori activi K_d, iar rRequestTime(k_users) calculează timpul total de procesare pentru acea zi prin însumarea timpilor individuali generați pentru fiecare utilizator. Dacă nu sunt utilizatori activi, timpul total este zero.

Această etapă permite observarea modului în care numărul mare de utilizatori conduce la un efect de netezire a distribuției totale a timpului de procesare.

3. Generarea vectorului agregat pentru 100 simulări

```
regenerateAgregValues <- function() {
  AgregDetails$vec <- sim_day_aggregate(AgregDetails$n_sim)
}
```

Variabila agregate_vec conține 100 de valori ale timpului total zilnic, simulând variabilitatea traficului pentru ziua respectivă.

9.2. Compararea histogramei agregatului cu o normală ajustată și precizarea când aproximarea este adecvată

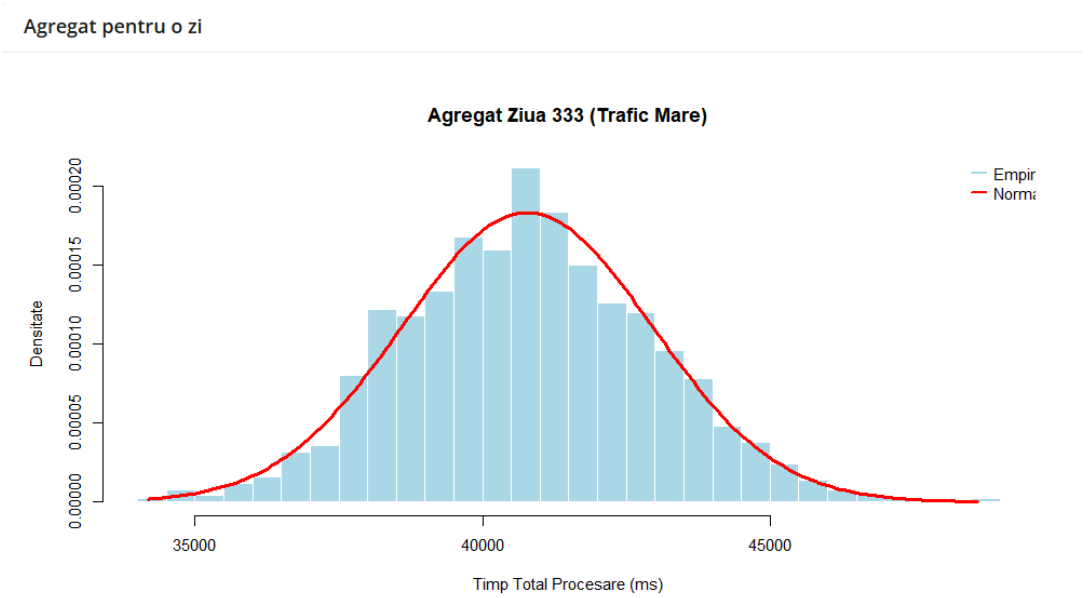
1. Estimarea parametrilor distribuției normale

```
mu_est <- mean(AgregDetails$vec)
sd_est <- sd(AgregDetails$vec)
```

Media și deviația standard estimate din simulări vor fi folosite pentru a suprapune o curbă normală peste histogramă.

2. Histogram agregata zilnic

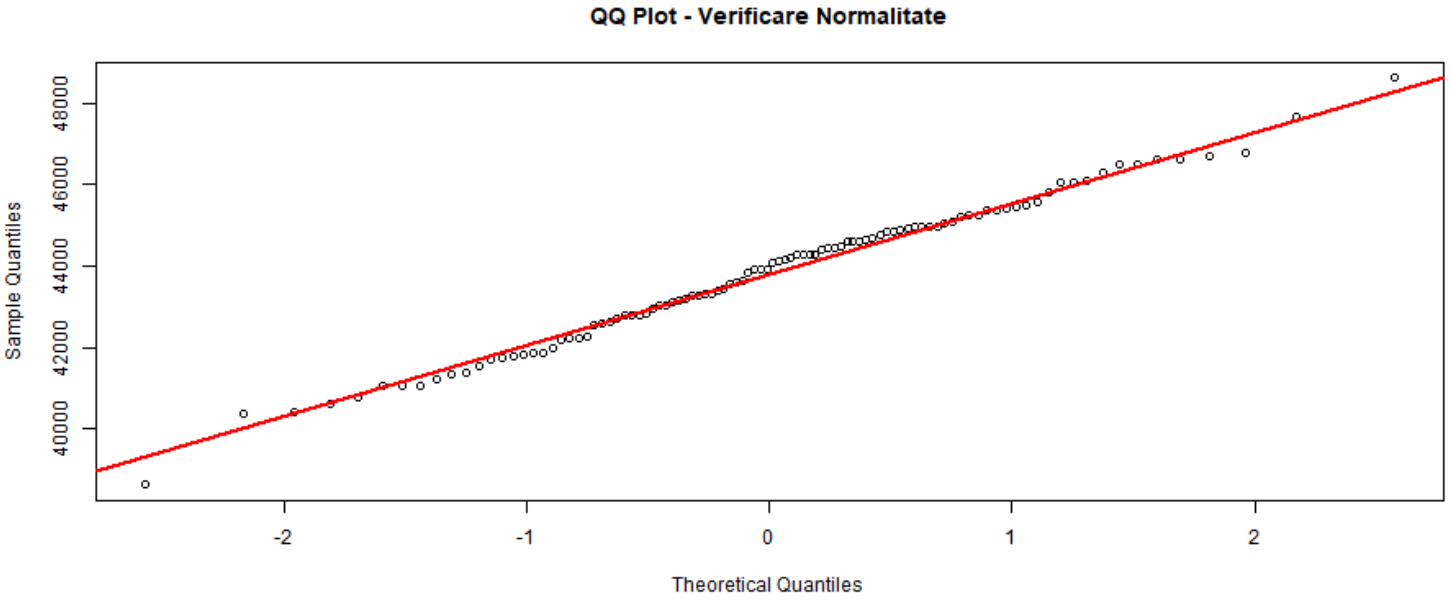
```
hist(AgregDetails$vec,
     prob = TRUE,
     breaks = 50,
     col = "lightblue",
     border = "white",
     main = paste("Agregat Ziua", AgregDetails$target_day, "(Trafic Mare)"),
     xlab = "Timp Total Procesare (ms)",
     ylab = "Densitate")
x_vals <- seq(min(AgregDetails$vec), max(AgregDetails$vec), length.out = 200)
y_vals <- dnorm(x_vals, mean = mu_est, sd = sd_est)
lines(x_vals, y_vals, col = "red", lwd = 3)
legend("topright", legend=c("Empiric", "Normal Fit"), col=c("lightblue", "red"), lwd=2, bty="n")
```



Histograma ilustrează distribuția empirică a datelor simulate, iar linia roșie reprezintă densitatea unei distribuții normale teoretice, ale cărei parametri, media și deviația standard, sunt estimați din eșantionul de date.

3. QQ-Plot pentru verificarea normalității

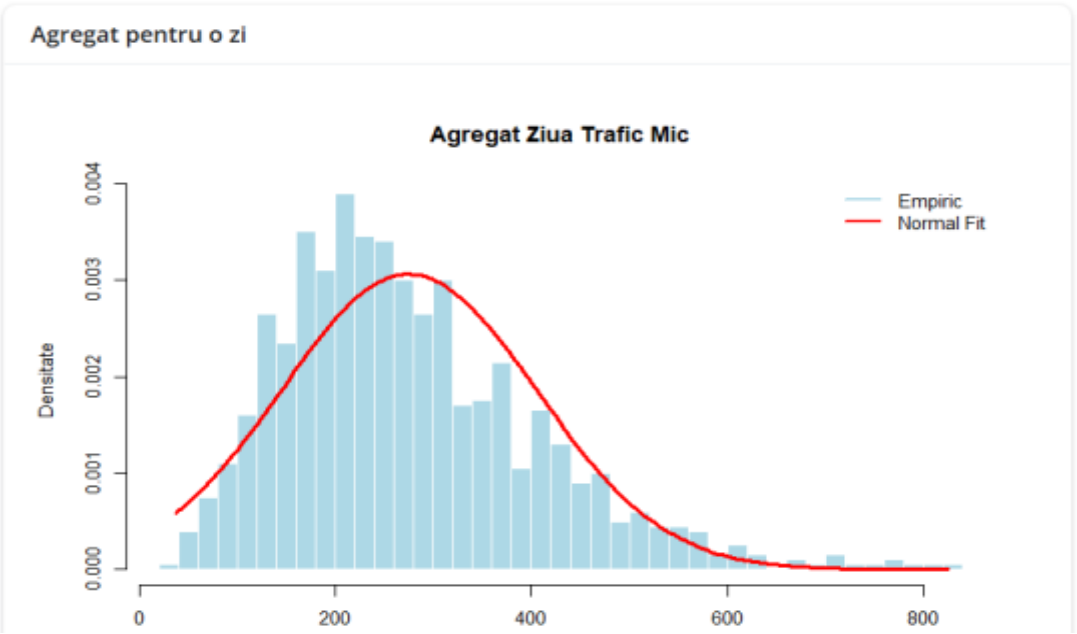
```
qqnorm(AgregDetails$vec, main = "QQ Plot - Verificare Normalitate")
qqline(AgregDetails$vec, col = "red", lwd = 2)
```



Dacă punctele urmează diagonala, distribuția agregatului poate fi considerată aproximativ normală.

4. Interpretare

În zilele cu trafic intens, cu peste 100 utilizatori sau Black Friday, distribuția agregatului este simetrică, iar histograma empirică se aliniază corect cu densitatea normală, reflectând un caz tipic de aplicare a Legii Numerelor Mari.



În schimb, pentru zilele cu trafic scăzut, cu sub 10 utilizatori, distribuția devine clar asimetrică, cu o coadă lungă la dreapta, iar aproximarea normală este inadecvată. În aceste condiții, distribuții asimetrice precum Gamma sau Log-normal oferă un model mult mai fidel.

10.1. Modelarea probabilistica a cele două scenarii: pierderea utilizatorilor aleator(q) și condiționat(m cereri, cel puțin k eșuează)

1. Scenariul aleator independent

```
probCerereSucces <- 0.9
probCerereEsec <- 1 - probCerereSucces
ChurnDetails <- new.env()
ChurnDetails$MedieCereriUtilizatori <- 30
ChurnDetails$probChurn = 0.05
ChurnDetails$distribution = "geom" # / "emp"
```

Fiecare cerere are o probabilitate fixă de eșec probChurn. Pierderea unui utilizator este independentă de ce s-a întâmplat la cererile anterioare.

2. Scenariul condiționat dependență pe fereastra de m cereri

```
ChurnDetails$m = 5
ChurnDetails$k = 2
ChurnDetails$emp_try = 1e5
.check_fer <- function(tries) {
  t <- sample(c(0, 1), tries, replace=T, prob = c(probCerereSucces, probCerereEsec))
  m <- ChurnDetails$mz`
  k <- ChurnDetails$k
  for (i in 1:(tries - m + 1))
  {
    if ( sum(t[i:(i+m-1)]) >= k)
    {
      return(i+(m-1))
    }
  }
  return(0)
}
ChurnDetails$generateHist <- function() {
  t <- replicate(ChurnDetails$emp_try, {
    .check_fer(300)
  })
  hist(t, breaks = max(t), plot=F)
}
ChurnDetails$hist =  ChurnDetails$generateHist()
```

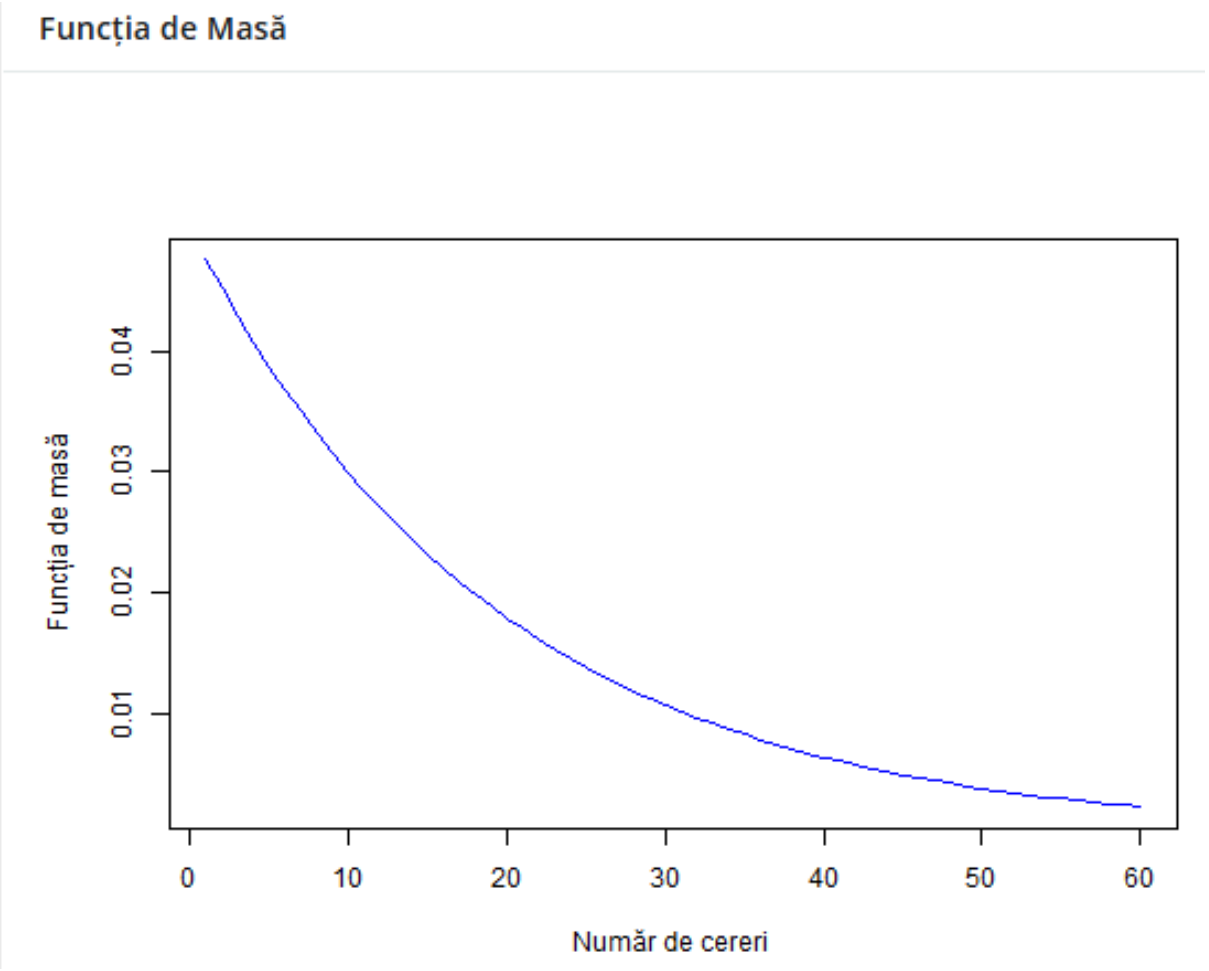
Pierderea utilizatorului depinde de acumularea eșecurilor într-o fereastră de m=5 cereri, dacă apar cel puțin k=2 eșecuri. Funcția .check_fer returnează momentul pierderii utilizatorului. Distribuția empirică a pierderilor este generată cu generateHist.

10.2. Estimarea probabilității de pierdere a utilizatorului pentru cele 2 scenarii

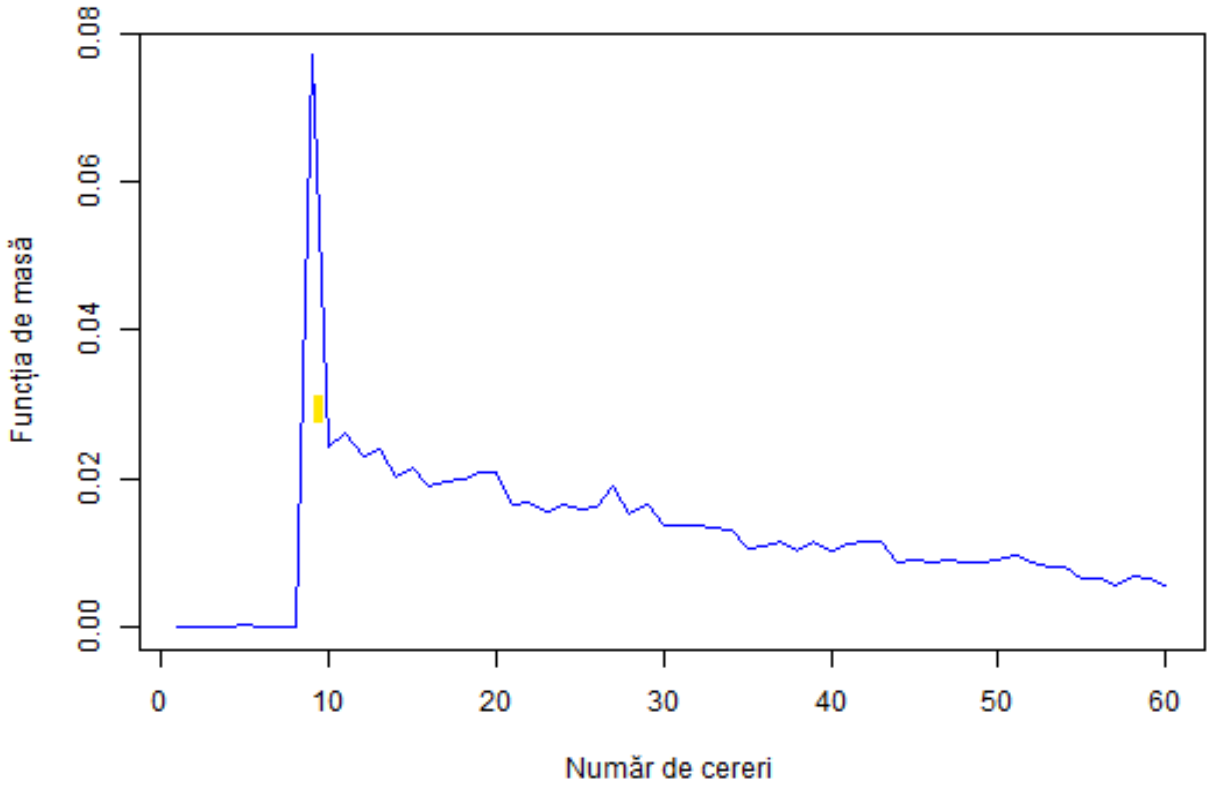
1. Funcția de densitate și repartitie

```
dChurn <- function (n) {
  stopifnot(all(n > 0))
  m <- ChurnDetails$m
  sapply(n, function(no) {
    if (ChurnDetails$distribution == "emp")
    {
      if (no < m)
        return(0)
      if (is.na(ChurnDetails$hist$density[no - m + 1]))
        return(0)
      else
        ChurnDetails$hist$density[no - m + 1]
    }
    else {
      dgeom(no, ChurnDetails$probChurn)
    }
  })
}
```

Funcția dChurn(n) returnează probabilitatea, densitatea, ca un utilizator să părăsească sistemul exact după a n-a cerere.



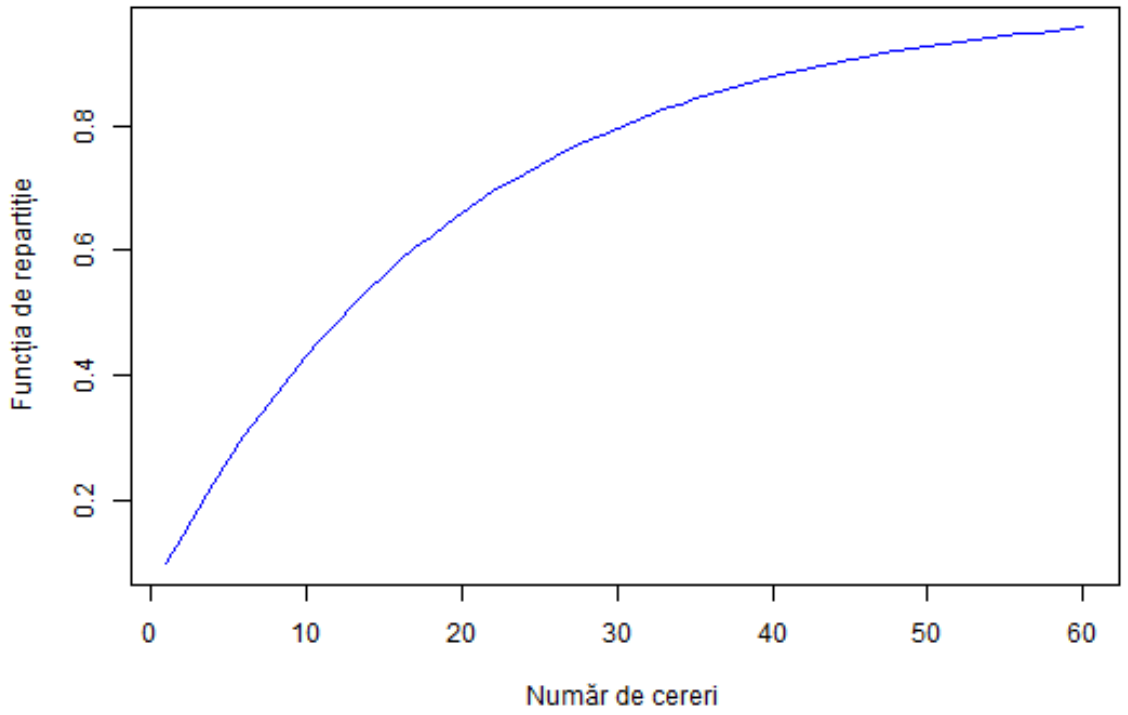
Funcția de Masă

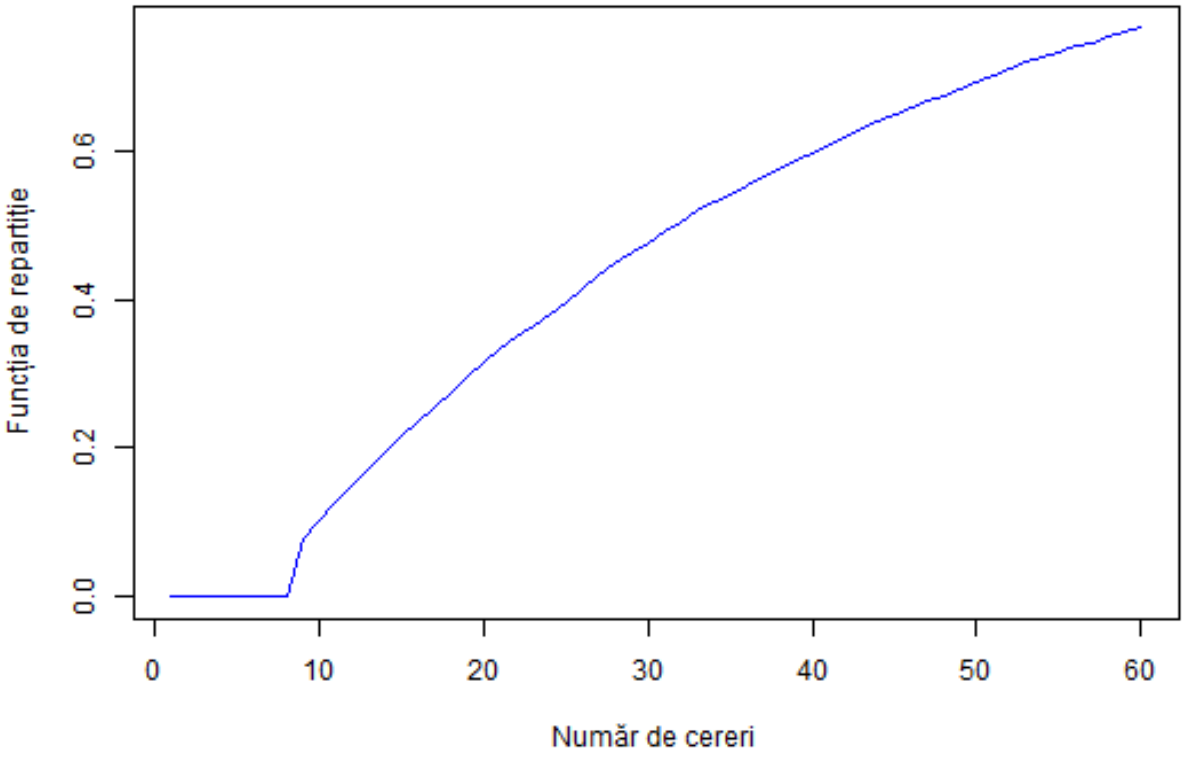



```
pChurn <- function(n) {
  stopifnot(all(n > 0))
  sapply(n, function(p) {
    if (ChurnDetails$distribution == "emp") {
      sum(dChurn(p))
    }
    else {
      pgeom(p, ChurnDetails$probChurn)
    }
  })
}
```

Funcția pChurn(n) returnează probabilitatea cumulativă, funcția de repartiție, ca utilizatorul să se fi pierdut până la, sau la, cererea n.

```
generateChurnDensityPlot <- function () {
  plot(1:60, dChurn(1:60), type="l",
       xlab="Număr de cereri",
       ylab="Funcția de masă",
       col="blue")
}
```





Pentru scenariul independent, aceste valori sunt calculate analitic folosind distribuția geometrică; pentru scenariul condiționat, ele sunt derivate din distribuția empirică obținută prin simulare.

2. Probabilitatea pe numărul de cereri per utilizator

```
UserLeaveEmp <- function (tries) {
  users <- rpois(tries, ChurnDetails$MedieCereriUtilizatori)
  users <- sapply(users, function (usa) {max(1, usa)}
  sapply(users, function(usa) {
    pChurn(usa)
  })
}
```

10.3. Compararea scenariilor și interpretarea acestora

Scenariul independent presupune că utilizatorul poate părăsi sistemul după orice eșec individual, ceea ce conduce la o probabilitate de churn mai mare și constantă.

Scenariul condiționat introduce o logică de toleranță, unde plecarea depinde de acumularea a cel puțin k eșecuri într-o fereastră de m cereri consecutive. Acest model este mai realist, deoarece reflectă faptul că utilizatorii acceptă eșecuri ocazionale și părăsesc doar când întâmpină probleme repetate în timp scurt.

Această diferență subliniază că modelul condiționat oferă o viziune mai echilibrată asupra stabilității sistemului, reducând riscul de a supraestima pierderea utilizatorilor din cauza unor eșecuri izolate.

11.1. Definirea unei v.a. pentru profitul zilnic(câștig per succes, pierdere per churn, penalități SLA)

1. Parametri economici ai modelului

```
EconDetails <- new.env()
EconDetails$venit_per_succes <- 0.5
EconDetails$cost_achizitie <- 50
EconDetails$penalitate_SLA <- 2
EconDetails$t0_SLA <- 100
```

Pentru modelarea economică a sistemului se definesc patru parametri: câștigul pe cerere reușită, costul fix al pierderii unui utilizator (churn), penalitatea pentru depășirea timpului SLA și pragul de timp SLA însuși.

2. Definirea profitului zilnic

```
simulate_daily_profit <- function(day = 1, year = 2019, probUserLeave = 0.2) {
  kd <- rActiveUsers(1, day, year = year)

  requests_per_user <- rpois(kd, lambda = MedieCereriUtilizatori)
  total_requests <- sum(requests_per_user)

  n_tries <- rRetry(total_requests)
  t_times <- rRequestTimeWithNTries(n_tries, 1)

  succese <- n_tries < NDetails$NMAX # | (runif(total_requests) < probSucc)
  venituri = sum(succese) * EconDetails$venit_per_succes
  penalitati = sum(t_times > EconDetails$t0_SLA) * EconDetails$penalitate_SLA

  prob_plecare <- probUserLeave
  utilizatori_pierduti <- rbinom(1, kd, prob_plecare)
  cost_churn = utilizatori_pierduti * EconDetails$cost_achizitie

  profit_zi = venituri - penalitati - cost_churn
  return(profit_zi)
}
```

Profitul zilnic este o variabilă aleatoare calculată ca diferența dintre veniturile din cererile reușite și suma penalităților SLA cu costurile generate de pierderea utilizatorilor, fiind dependentă de numărul de utilizatori activi, de comportamentul cererilor și de mecanismul de churn.

11.2. Estimarea mediei, variației, și (opțional) intervalelor de încredere pentru profit

1. Simularea profitului zilnic

```
EconDetails$t <- replicate(tries, {
    simulate_daily_profit(day, year, probUserLeave)
})
```

Profitul zilnic este estimat prin simulare Monte Carlo, generând mai multe valori independente ale profitului pentru aceeași zi.

2. Indicatori statistici

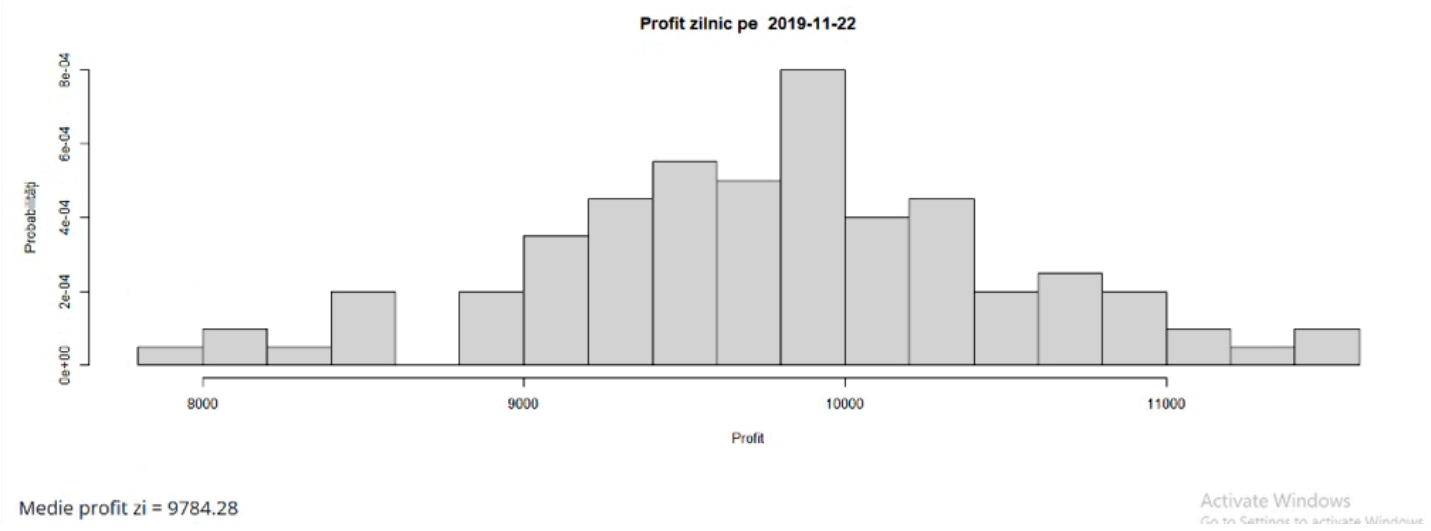
```
mean(EconDetails$t)
var(EconDetails$t)
```

Media reprezintă profitul zilnic așteptat al sistemului, iar varianța măsoară volatilitatea acestuia, indicând nivelul de risc economic.

3. Distribuția empirică a profitului

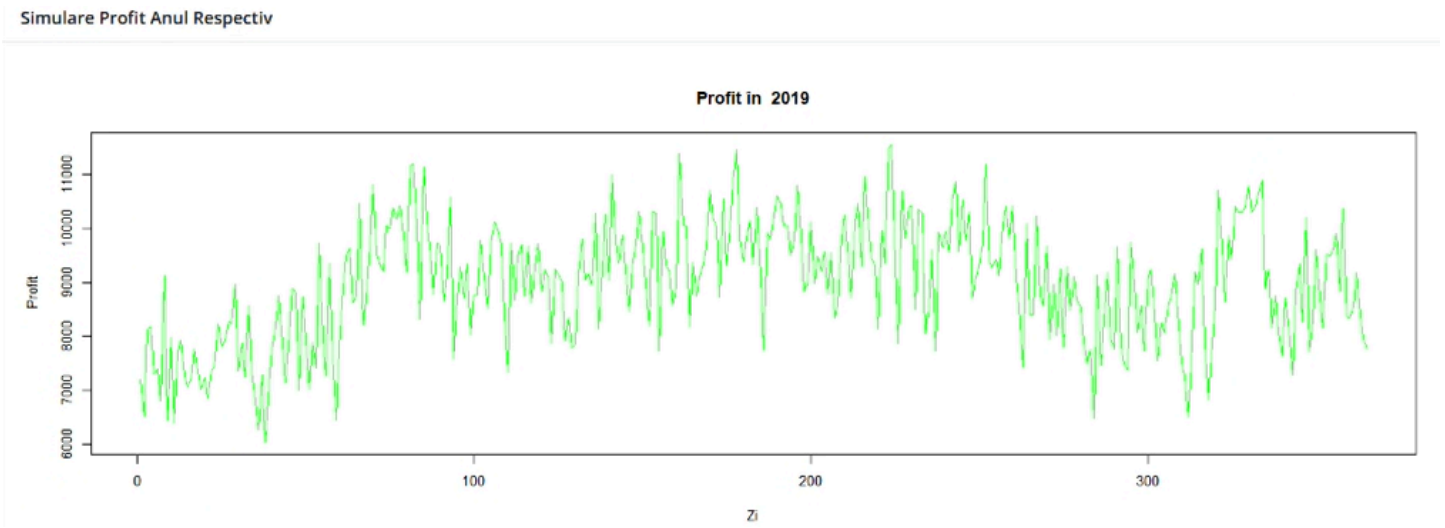
```
hist(main = paste("Profit zilnic pe ", Profit$date),
        EconDetails$t, prob = T, breaks = 15,
        xlab="Profit", ylab="Probabilități")
```

Histograma oferă o reprezentare vizuală a distribuției profitului zilnic, evidențiind asimetria și dispersia valorilor obținute prin simulare.



11.3. Analiza compromisurilor tehnico-economice

```
generatePlotAn <- function(year = 2019, probUserLeave = 0.2) {
  dt <- sapply(1:365, function(d) {
    simulate_daily_profit(d, year, probUserLeave)
  })
  plot(1:365, dt, type = "l", col = "green", main = paste("Profit în ", year),
       xlab="Zi", ylab="Profit")
}
```



Creșterea numărului de cereri reușite aduce venituri mai mari, dar sistemul primește penalități dacă timpii de răspuns depășesc limita SLA, ceea ce subliniază importanța performanței tehnice.

Pierderea utilizatorilor are însă cel mai mare impact economic direct, reprezentând componenta cea mai costisitoare.

Astfel, există un compromis între investiția în îmbunătățirea performanței, pentru reducerea latențelor și a eșecurilor, și pierderile economice cauzate de penalități și de părăsirea utilizatorilor.

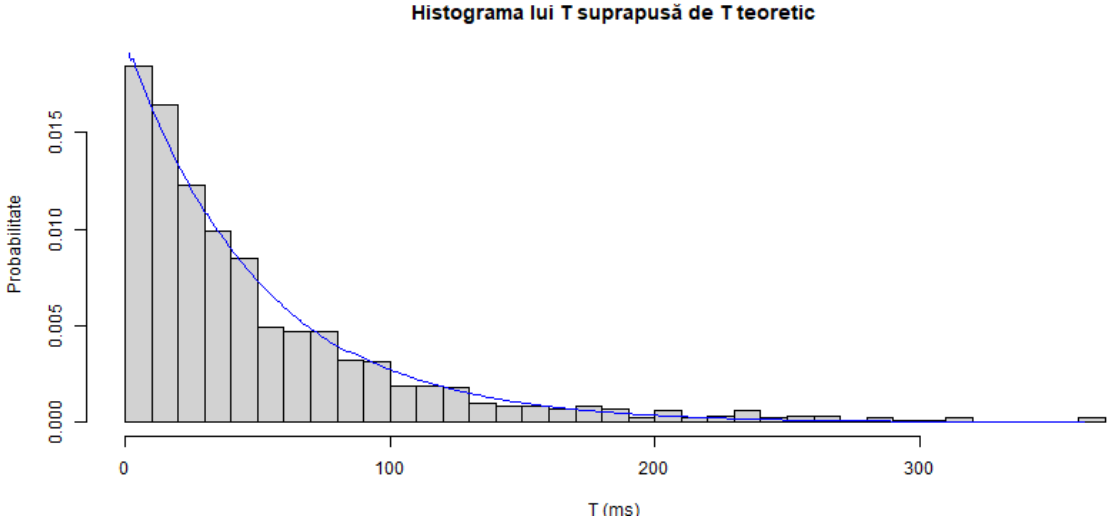
12.1. Histograme pentru T și profit

```
  rep_N <- rRetry(Details12$t_sim)
  Details12$t_vals <- rRequestTimeWithNTries(rep_N, 1)
```

Pentru analiza distribuției timpului total de procesare T, se generează valori simulate ale acestei variabile folosind modelul definit anterior.

```
generate12HistT <- function() {
  hist(Details12$t_vals, prob = T, breaks = 30, main = "Histograma lui T suprapusă de T teoretic",
       xlab="T (ms)", ylab="Probabilitate")
  lines(1:max(Details12$t_vals), dRequestTime(1:max(Details12$t_vals)), col = "blue")
}
```

Această reprezentare permite compararea distribuției empirice obținute prin simulare cu distribuția teoretică a lui T.



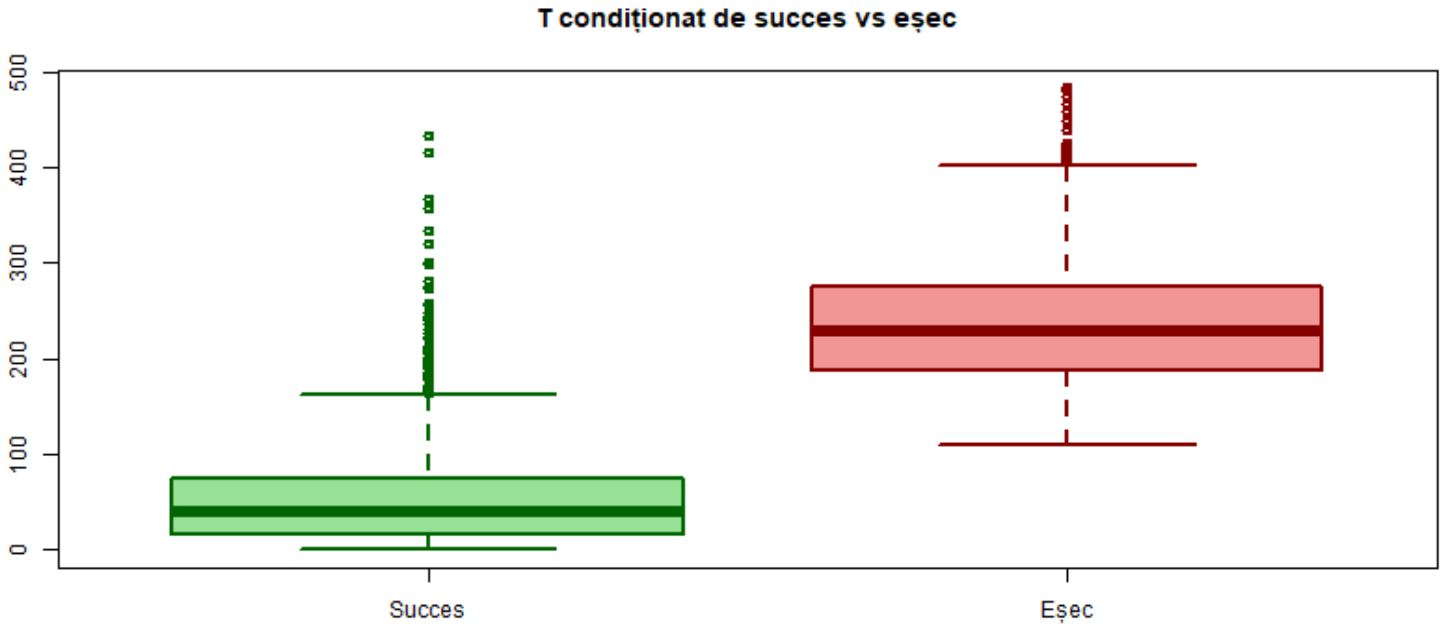
12.2. Boxplot-uri pentru T condiționat de succes/eșec și pentru scenarii diferite

Pentru a analiza impactul rezultatului unei cereri asupra timpului total, se consideră distribuțiile lui T condiționate de succes și de eșec.

```
Details12$tCondEsec <- rRequestTimeWithNTries(NDetails$NMAX, Details12$t_sim)
Details12$tCondSucc <- rRequestTime(Details12$t_sim)
```

Acestea sunt reprezentate grafic prin boxplot-uri comparative

```
generate12TCond <- function () {
  boxplot(
    Details12$tCondSucc,
    Details12$tCondEsec,
    names = c("Succes", "Eșec"),
    main = "T condiționat de succes vs eșec",
    ylab = "T",
    col = c(rgb(0.2, 0.8, 0.2, 0.5), rgb(0.9, 0.2, 0.2, 0.5)),
    border = c("darkgreen", "darkred"),
    lwd = 2
  )
}
```



Boxplot-urile evidențiază diferențele dintre timpii de procesare în cazul cererilor finalizate cu succes și al celor eșuate.

12.3. Interpretarea medianei, IQR, outlieri

Mediana și IQR oferă o vedere robustă asupra timpului tipic de răspuns și a variabilității acestuia, fiind mai puțin afectate de valori extreme. Outlierii corespund cererilor cu latențe excepționale, adesea cauzate de reîncercări multiple.

Analiza cererilor eșuate arată o medie mai ridicată, o dispersie mai mare și mai mulți outlieri, reflectând clar impactul negativ al eșecurilor asupra performanței și stabilității sistemului.

13.1. Comentarea rolului probabilității empirice

Modelul matematic al procesului combină mai multe surse de complexitate care fac calculul analitic direct impracticabil. Timpul total și profitul sunt definite ca sume ale unui număr aleatoriu de variabile aleatoare, unde termenii individuali sunt la rândul lor afectați de condiții și dependențe interne.

Pentru a rezolva această provocare, am recurs la simulări Monte Carlo, care estimează probabilitățile teoretice necunoscute prin probabilități empirice. Conform Legii Numerelor Mari, cu cât numărul de rulări simulări crește, cu atât aproximarea empirică se apropie mai mult de adevărata distribuție a sistemului.

Această abordare a oferit nu doar estimări cantitative, dar a și relevat proprietăți calitative esențiale. În special, simulările au evidențiat prezența unor distribuții cu coadă lungă, un aspect care ar fi rămas ascuns într-o analiză pur teoretică, demonstrând riscul unor evenimente rare cu consecințe disproporționat de mari.

13.2. Ce informații aduc condiționările

Condițiile introduse în pașii anteriori de modelare au adus o profunzime realistă esențială unui cadru altfel teoretic. Fapte precum „o cerere a eșuat deja” sau „suntem în faza de retry” nu sunt doar detalii, ci factori care redefinește radical așteptările privind timpul total de răspuns.

Fără luarea în considerare a acestor condiții, analiza s-ar limita la medii globale, ce maschează comportamentele extreme ale sistemului. Prin introducerea condiționării, modelul a scos la iveală existența a două regimuri operaționale clare: unul “normal”, caracterizat de viteză și stabilitate, și unul “de avarie”, unde sistemul devine lent și instabil.

Această distincție nu este doar un rezultat statistic; este o informație critică pentru inginerii și arhitecții de sisteme, deoarece le permite să înțeleagă, să anticipeze și să se pregătească pentru tranziția între aceste două stări fundamentale ale infrastructurii.

13.3. Comentarea utilității inegalităților probabilistice

Simulările oferă rezultate numerice practice, în timp ce inegalitățile probabilistice precum cele ale lui Markov, Cebîșev și Jensen furnizează garanții teoretice pentru scenarii extreme. Inegalitatea lui Cebîșev asigură o limitare a probabilității abaterilor mari, chiar și atunci când distribuția exactă nu este cunoscută.

Inegalitatea lui Jensen este esențială pentru analiza economică a riscului. Când funcția de cost este convexă, cum este cazul penalităților care cresc accelerat cu întârzierea, costul mediu observat va fi întotdeauna mai mare decât costul calculat pe baza timpului mediu de răspuns.

Această concluzie arată că folosirea mediei poate fi înșelătoare în evaluarea riscurilor, deoarece subestimează impactul real al evenimentelor rare dar costisitoare.

13.4. Comentarea legăturii dintre performanța tehnică și impactul economic

Performanța tehnică a sistemului, exprimată prin latența totală, T, și gradul de conformitate cu SLA-ul, are un impact direct și măsurabil asupra profitabilității.

Costurile operaționale cresc proporțional cu timpul de procesare, iar depășirea pragului SLA atrage penalități financiare specifice. Ca urmare, chiar dacă venitul pe cerere rămâne constant, o creștere a latenței medii sau o extindere a cozii distribuției, manifestată prin întârzieri rare, dar extreme, va reduce sistematic profitul mediu și va crește probabilitatea apariției de rezultate financiare negative.

13.5. Comentarea parametrilor ce influențează cel mai mult rezultatele finale și a ce modificări am aduce pentru pentru îmbunătățirea sistemului.

Cei mai influenți parametri ai sistemului sunt media și dispersia timpilor de răspuns, SDetails$mean, limita maximă de reîncercări, NDetails$NMAX, și pragul de timp SLA împreună cu penalitățile asociate.

Reducerea lui NMAX poate micșora latența totală și riscul de penalizare, dar poate și diminua rata finală de succes. Îmbunătățirea performanței de bază prin scăderea mediei timpilor de răspuns, SDetails$mean, are un impact dublu pozitiv: îmbunătățește atât conformitatea cu SLA, cât și profitabilitatea.

În practică, o strategie de optimizare ar include ajustarea limitelor de retry, NMAX, reglarea algoritmului de backoff către o politică mai puțin agresivă și investiții în reducerea latenței fundamentale a sistemului, pentru a reduce varianța și a scurta coada lungă a distribuției de timpi de răspuns.

6. Identificarea unor eventuale dificultăți în realizarea cerințelor

Complexitatea principală a acestui proiect a constat în modelarea adecvată a variabilelor aleatoare care descriu dinamica serviciului online și a comportamentului utilizatorilor. Sistemul integrează mecanisme multiple, cum ar fi reîncercările, întârzierile deliberate (backoff), fenomenul de părăsire a utilizatorilor (churn) și distribuții condiționate, ceea ce a impus identificarea unor distribuții statistice precise și construirea de modele care să reproducă fidel comportamente reale.

O dificultate semnificativă a fost integrarea armonioasă a distribuțiilor pentru numărul de cereri și timpii de răspuns, permițând compararea între valorile empirice și cele teoretice. De asemenea, modelarea dependențelor între variabile, precum creșterea progresivă a latențelor după eșecuri sau condițiile de prag pentru pierderea utilizatorilor, a necesitat o abordare atentă.

Un alt aspect complex a fost agregarea acestor modele într-un cadru de simulare a performanței economice, unde fiecare variabilă contribuie direct la rezultatul financiar. Aplicarea inegalităților probabilistice, cum ar fi cele ale lui Markov, Cebîșev, Cernov și Jensen, pentru a deduce limite teoretice în scenarii extreme, a necesitat gestionarea corectă a relațiilor dintre medii, dispersii și tipul distribuțiilor.

Prin urmare, provocarea centrală nu a fost implementarea tehnică a interfeței sau a graficelor, ci conceperea unui model statistic coerent care să captureze fidel logica sistemului. Această fundație teoretică robustă a fost esențială pentru ca simulările să genereze rezultate semnificative și interpretabile, oferind aplicației o valoare analitică autentică, dincolo de atractivitatea vizuală.

7. Probleme care au rămas deschise în urma implementării actuale

Modelarea distribuțiilor complexe prezintă anumite limite. Deși majoritatea distribuțiilor au fost simulate și comparate cu date empirice, scenariile reale ale sistemului sau comportamentul utilizatorilor pot fi mai variabile și mai neregulate decât cele capturate de distribuțiile teoretice alese. În special, dependențele condiționate și relațiile dintre multiple variabile pot necesita modele statistice mai sofisticate.

Scalabilitatea simulărilor reprezintă o altă limitare. Simulările pe scară mare, cum ar fi 100.000 de iterații pentru teste probabilistice sau milioane de cereri pentru analiza churn, pot consuma resurse semnificative de timp și memorie, restricționând capacitatea de a explora scenarii mai complexe sau de a rula simulări în timp real.

Integrarea variabilelor economice și tehnice este în prezent realizată prin simulări empirice; însă, nu există un model matematic complet care să formalizeze această legătură și să permită optimizarea automată a parametrilor pentru maximizarea profitului sau minimizarea pierderilor.

Rezultatele obținute sunt dependente de ipotezele de simulare stabilite inițial, cum ar fi ratele de succes, limitele maxime de reîncercare sau criteriile pentru pierderea utilizatorilor. Schimbări în aceste ipoteze pot conduce la rezultate diferite, ceea ce subliniază necesitatea unei interpretări atente și contextualizate.

În forma actuală, analiza este concentrată pe perioade zilnice sau anuale individuale, fără a aborda evoluția pe termen lung a sistemului sau efectele economice cumulate ale deciziilor legate de politici de retry, SLA sau managementul churn. Această limită reduce capacitatea de a evalua impactul strategic al ajustărilor parametrilor pe o perioadă extinsă.

8. Concluzii

Proiectul a demonstrat cum poate fi folosită simularea empirică și modelarea distribuțiilor pentru a analiza performanța unui serviciu online, incluzând timpii de răspuns, probabilitățile de eșec și impactul economic al deciziilor operaționale. Prin integrarea analizei statistice, simulărilor Monte Carlo și a indicatorilor economici, am putut evidenția relația directă dintre performanța tehnică și profitul serviciului, precum și utilitatea inegalităților probabilistice pentru estimarea riscurilor în scenarii incertitudine.

De asemenea, proiectul a scos în evidență importanța condiționării și a dependențelor între variabile, demonstrând că simpla medie globală nu capturează fenomenele extreme sau rare care pot afecta semnificativ rezultatele. Astfel, simularea empirică rămâne un instrument valoros pentru luarea deciziilor informate și pentru optimizarea sistemelor complexe.

Proiectul subliniază importanța utilizării modelelor statistice și a simulării empirice pentru înțelegerea complexității sistemelor online și luarea deciziilor bazate pe date concrete, oferind o imagine obiectivă asupra performanței și riscurilor asociate.