UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

PROIECT
APLICAȚIE WEB FOLOSIND R SHINY

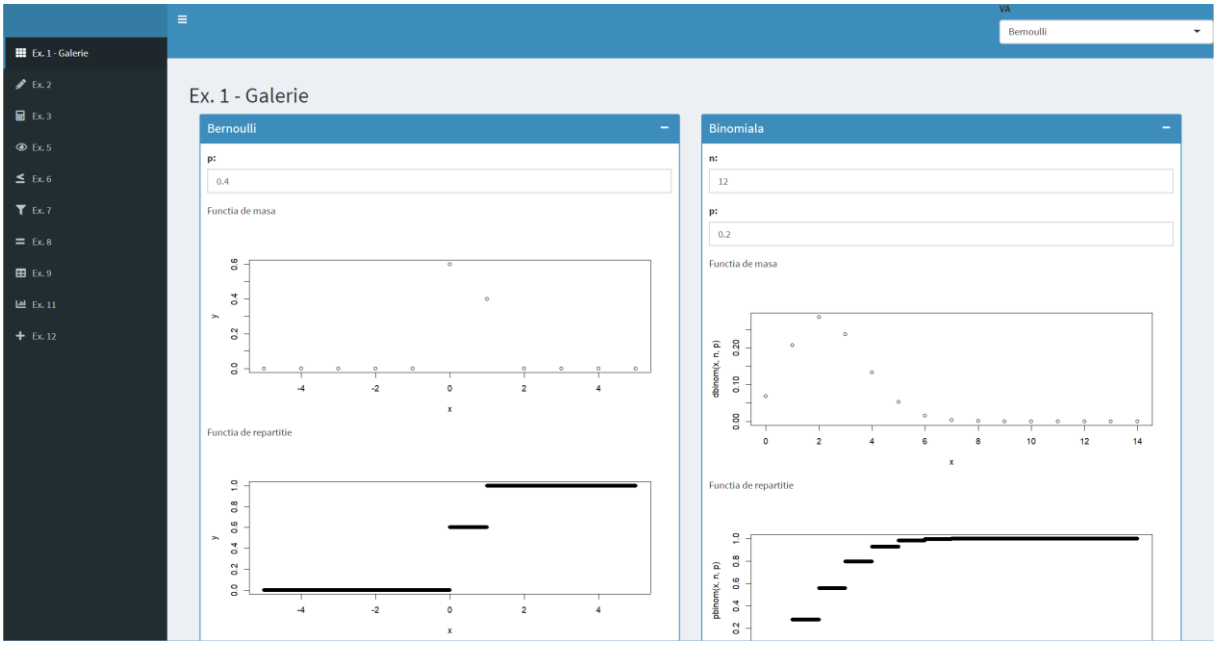
STUDENȚI:
CUCU ȘTEFAN – CĂTĂLIN (lider)
FĂRCĂȘANU TUDOR – ANDREI
UDREA IULIA – MARIA

GRUPA 263

BUCUREȘTI

Introducere

Am realizat acest proiect în limbajul de programare R, folosind framework-ul *Shiny* pentru a crea o aplicație web interactivă ce facilitează lucrul cu variabile aleatoare, atât discrete cât și continue. De asemenea, pentru o mai bună îmbinare a cerințelor propuse, am folosit pachetul *shiny dashboard*, care ne-a permis să împărțim aplicația în 10 pagini distincte, una pentru fiecare dintre cerințele realizate.



Interfața aplicației, împreună cu side-bar-ul ce permite navigarea printre cerințe

Au mai fost folosite pachetele:

- *discreteRV,* folosit pentru a construi intern variabilele aleatoare discrete utilizate în cadrul cerințelor
- *DT,* care furnizează o interfață în cadrul limbajului R pentru librăria de JavaScript *DataTables*, putând afișa matrici și *data frames* din R, dar furnizând și capabilități de filtrare, paginație, sortare, etc.
- *dplyr,* care furnizează un set de funcții pentru manipularea obiectelor de tip *data frame*
- *rlist,* care ajută la manipularea obiectelor de tip listă prin funcții de filtrare, grupare, update, etc.
- *readxl,* folosit pentru citirea fisierelor de tip *.xls* și *.xlsx*

De asemenea, am construit un selector în cadrul header-ului aplicației, ce permite selectarea uneia dintre repartițiile din galerie (atât cele predefinite în cadrul cerinței 1, cât și cele furnizate manual în cerința 2). Acesta este folosit în cerințele ce necesită alegerea unei repartiții de variabile aleatoare din cadrul galeriei:

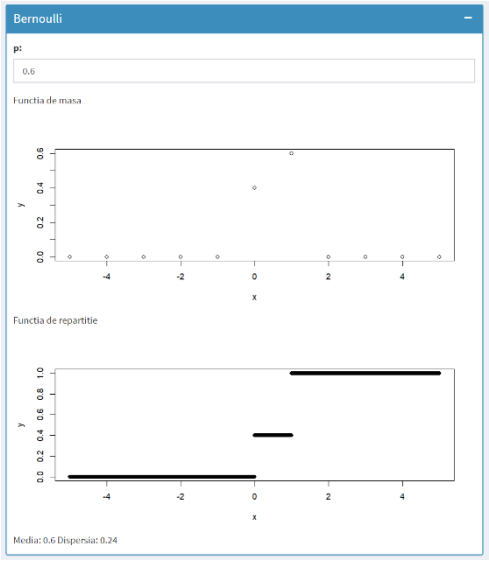
```
output$va_select <- renderUI({
  predef <- c("Bernoulli"="bern",
              "Binomiala"="binom",
              "Geometrica"="geom",
              "Hipergeometrica"="hgeom",
              "Poisson"="pois",
              "Uniforma"="unif",
              "Exponentiala"="exp",
              "Normala"="norm")
  if(ex2_rvs$cnt > 0){
    print(ex2_rvs$cnt)
    print(1:ex2_rvs$cnt)
    customd <- sapply(1:ex2_rvs$cnt, function(a){
      return(sprintf("d%d", a))
    })
    names(customd) = sapply(1:ex2_rvs$cnt, function(a){
      return(sprintf("Custom v.a. discreta %d", a))
    })
    predef <- c(predef, customd)
  }
  if(ex2_fcts$cnt > 0){
    customc <- sapply(1:ex2_fcts$cnt, function(a){
      return(sprintf("c%d", a))
    })
    names(customc) = sapply(1:ex2_fcts$cnt, function(a){
      return(sprintf("Custom v.a. continua %d", a))
    })
    predef <- c(predef, customc)
  }
  selectInput("va_selected", label="VA", predef)
})
```

Cerința 1

În această secțiune am construit o galerie care conține toate repartițiile utilizate în cadrul acestui proiect. Aceasta este alcătuită din 8 repartiții standard, predefinite, cât și din repartițiile de variabile aleatoare adăugate de manual în cadrul cerinței 2.

Fiecare repartiție a fost ilustrată folosind o casetă *box* din librăria *shinyDashboard.* Pentru fiecare repartiție, în funcție de parametrii introduși, se ilustrează graficele funcției de masă / densitate și a funcției de repartiție, cât și media și dispersia.

```
box(
            title="Bernoulli", status="primary", solidHeader=TRUE,
collapsible=TRUE,
            numericInput("ex_1_bern_p", "p:", 0, min=0, max=1,
step=0.1),
            "Functia de masa",
            plotOutput("ex_1_bern_plot1", height = "290px"),
            "Functia de repartitie",
            plotOutput("ex_1_bern_plot2", height = "290px"),
            textOutput("ex_1_bern_txt")
          )
```



Exemplu de casetă pentru o repartiție predefinită

Cele 8 repartiții predefinite sunt:

- Bernoulli

Se consideră cunoscut parametrul $p \in [0,1]$, introdus de utilizator.

```
output$ex_1_bern_plot1 <- renderPlot({
    p <- input$ex_1_bern_p
    X <- RV(c(0,1), c(1-p, p))
    x <- seq(-5, 5)
    y <- sapply(x, function(nr) {
      if(nr %in% X) return(P(X == nr))
      else return(0)
    })
    plot(x, y)
```

Plotarea funcției de masă pentru repartiția Bernoulli folosind definiția (1)

```
output$ex_1_bern_plot2 <- renderPlot({
  p <- input$ex_1_bern_p
  X <- RV(c(0,1), c(1-p, p))
  x <- seq(-5, 5, 0.001)
  y <- sapply(x, function(nr) {
    return(P(X <= nr))
  })
  plot(x, y)
})
```

Plotarea funcției de repartiție pentru repartiția Bernoulli folosind definiția (1)

```
output$ex_1_bern_txt <- renderText({
  p <- input$ex_1_bern_p
  paste("Media:", p, "\nDispersia:", p * (1 - p))
})
```

Calculul mediei folosind formula $E(X) = p$ *și a variației* $Var(X) = p * (1 - p)$ (1)

- Binomială

Se consideră cunoscuți parametrii *n* și $p \in [0,1]$, introdusi de utilizator.

```
output$ex_1_binom_plot1 <- renderPlot({
  p <- input$ex_1_binom_p
  n <- input$ex_1_binom_n

  x <- seq(0, n+2)
  plot(x, dbinom(x,n,p))
})
```

Plotarea funcției de masă pentru repartiția Binomială folosind funcția predefinită dbinom().

```
output$ex_1_binom_plot2 <- renderPlot({
  p <- input$ex_1_binom_p
  n <- input$ex_1_binom_n

  x <- seq(0, n+2,0.001)
  plot(x,pbinom(x,n,p))
})
```

Plotarea funcției de repartiție pentru repartiția Binomială folosind funcția predefinită pbinom().

```
output$ex_1_binom_txt <- renderText({
  p <- input$ex_1_binom_p
  n <- input$ex_1_binom_n
  paste("Media:", p*n, "\nDispersia:", n * p * (1 - p))
})
```

Calculul mediei folosind formula $E(X) = np$ *și a variației* $Var(X) = n * p * (1 - p)$ (2)

- Geometrică

Se consideră cunoscuți parametrii $p \in [0,1]$, introdus de utilizator.

```
output$ex_1_geom_plot1 <- renderPlot({
  p <- input$ex_1_geom_p
  x <- seq(0, 6)
  plot(x, dgeom(x,p))
})
```

Plotarea funcției de masă pentru repartiția Geometrică folosind funcția predefinită dgeom().

```
output$ex_1_geom_plot2 <- renderPlot({
  p <- input$ex_1_geom_p
  x <- seq(0, 6,0.001)
  plot(x,pgeom(x,p))
})
```

Plotarea funcției de repartiție pentru repartiția Geometrică folosind funcția predefinită pgeom().

```
output$ex_1_geom_txt <- renderText({
  p <- input$ex_1_geom_p
  paste("Media:", (1 - p) / p, "\nDispersia:", (1 - p) / (p ^ 2))
})
```

Calculul mediei folosind formula $E(X) = \frac{(1-p)}{p}$ *și a variației* $Var(X) = \frac{(1-p)}{p^2}$ (3)

- Hipergeometrică

Se consideră cunoscuți parametrii m, n, k, introdusi de utilizator.

```
output$ex_1_hgeom_plot1 <- renderPlot({
  n <- input$ex_1_hgeom_n
  m <- input$ex_1_hgeom_m
  k <- input$ex_1_hgeom_k

  x <- seq(0, k+1)
  plot(x, dhyper(x,m,n,k))
})
```

Plotarea funcției de masă pentru repartiția Hipergeometrică folosind funcția predefinită dhyper().

```
output$ex_1_hgeom_plot2 <- renderPlot({
  n <- input$ex_1_hgeom_n
  m <- input$ex_1_hgeom_m
  k <- input$ex_1_hgeom_k
  x <- seq(0, k+1, 0.001)
  plot(x,phyper(x,m,n,k))
})
```

Plotarea funcției de repartiție pentru repartiția Hipergeometrică folosind funcția predefinită phyper().

```
output$ex_1_hgeom_txt <- renderText({
  n <- input$ex_1_hgeom_n
  m <- input$ex_1_hgeom_m
  k <- input$ex_1_hgeom_k
  p <- m / (m + n)
  paste("Media:", k * p, "\nDispersia:", k * p * (1 - p) * (m + n - k) / (m +
n - 1))
})
```

Calculul mediei folosind formula $E(X) = k * p$ *și a variației* $Var(X) = \frac{k*p*(1-p)*(m+n-k)}{m+n-1}$,

unde $p = \frac{m}{m+n}$ *este densitatea distribuției* (4)

- Poisson

Se consideră cunoscuți parametrul λ, introdus de utilizator.

```
output$ex_1_pois_plot1 <- renderPlot({
  l <- input$ex_1_pois_l
  x <- seq(0, 6)
  plot(x, dpois(x,l))
})
```

Plotarea funcției de masă pentru repartiția Poission, folosind funcția predefinită dpois().

```
output$ex_1_pois_plot2 <- renderPlot({
  l <- input$ex_1_pois_l
  x <- seq(0, 6,0.001)
```

```
  plot(x,ppois(x,l))
})
```

Plotarea funcției de repartiție pentru repartiția Poisson folosind funcția predefinită ppois().

```
output$ex_1_pois_txt <- renderText({
  l <- input$ex_1_pois_l
  paste("Media:", l, "\nDispersia:", l)
})
```

Calculul mediei folosind formula $E(X) = \lambda$ *și a variației* $Var(X) = \lambda$ (5)

- Uniformă (continuă)

Se consideră cunoscuți parametrii a, b, introdusi de utilizator.

```
output$ex_1_unif_plot1 <- renderPlot({
  a <- input$ex_1_unif_a
  b <- input$ex_1_unif_b

  x <- seq(a-1, b+1, 0.1)
  plot(x, dunif(x,a,b), type='l')
  lines(x, dunif(x,a,b))
})
```

Plotarea funcției de densitate pentru repartiția Uniformă, folosind funcția predefinită dunif().

```
output$ex_1_unif_plot2 <- renderPlot({
  a <- input$ex_1_unif_a
  b <- input$ex_1_unif_b

  x <- seq(a-1, b+1,0.001)
  plot(x,punif(x,a,b), type='l')
})
```

Plotarea funcției de repartiție pentru repartiția Uniformă, folosind funcția predefinită punif().

```
output$ex_1_unif_txt <- renderText({
  a <- input$ex_1_unif_a
  b <- input$ex_1_unif_b
  paste("Media:", 0.5 * (a + b), "\nDispersia:", ((b - a) ^ 2) / 12)
})
```

Calculul mediei folosind formula $E(X) = \frac{a+b}{2}$ *și a variației* $Var(X) = \frac{1}{12}(b-a)^2$ (6)

- Exponențială

Se consideră cunoscuți parametrul λ, introdus de utilizator.

```
output$ex_1_exp_plot1 <- renderPlot({
  l <- input$ex_1_exp_l

  x <- seq(0, 6, 0.1)
  plot(x, dexp(x,l), type='l')
})
```

Plotarea funcției de densitate pentru repartiția Exponențială, folosind funcția predefinită dexp().

```
output$ex_1_exp_plot2 <- renderPlot({
  l <- input$ex_1_exp_l
```

```
  x <- seq(0, 6, 0.1)
  plot(x,pexp(x,l), type='l')
})
```

Plotarea funcției de repartiție pentru repartiția Exponențială, folosind funcția predefinită pexp().

```
output$ex_1_exp_txt <- renderText({
  l <- input$ex_1_exp_l
  paste("Media:", 1 / l, "\nDispersia:", 1 / (l ^ 2))
})
```

Calculul mediei folosind formula $E(X) = \frac{1}{\lambda}$ *și a variației* $Var(X) = \frac{1}{\lambda^2}$ (7)

- Normală

Se consideră cunoscuți parametrii μ (media) și σ (deviația standard), introdusi de utilizator.

```
output$ex_1_norm_plot1 <- renderPlot({
  n <- input$ex_1_norm_m
  d <- input$ex_1_norm_d

  x <- seq(n-d/2, n+d/2, 0.1)
  plot(x, dnorm(x,n,d), type='l')
})
```

Plotarea funcției de densitate pentru repartiția Normală, folosind funcția predefinită dnorm().

```
output$ex_1_norm_plot2 <- renderPlot({
  n <- input$ex_1_norm_m
  d <- input$ex_1_norm_d

  x <- seq(n-d/2, n+d/2, 0.1)
  plot(x, pnorm(x,n,d), type='l')
})
```

Plotarea funcției de repartiție pentru repartiția Normală, folosind funcția predefinită pnorm().

```
output$ex_1_norm_txt <- renderText({
  n <- input$ex_1_norm_m
  d <- input$ex_1_norm_d
  paste("Media:", n, "\nDispersia:", d ^ 2)
})
```

Afișarea mediei și a variației. (8)

Cerința 2

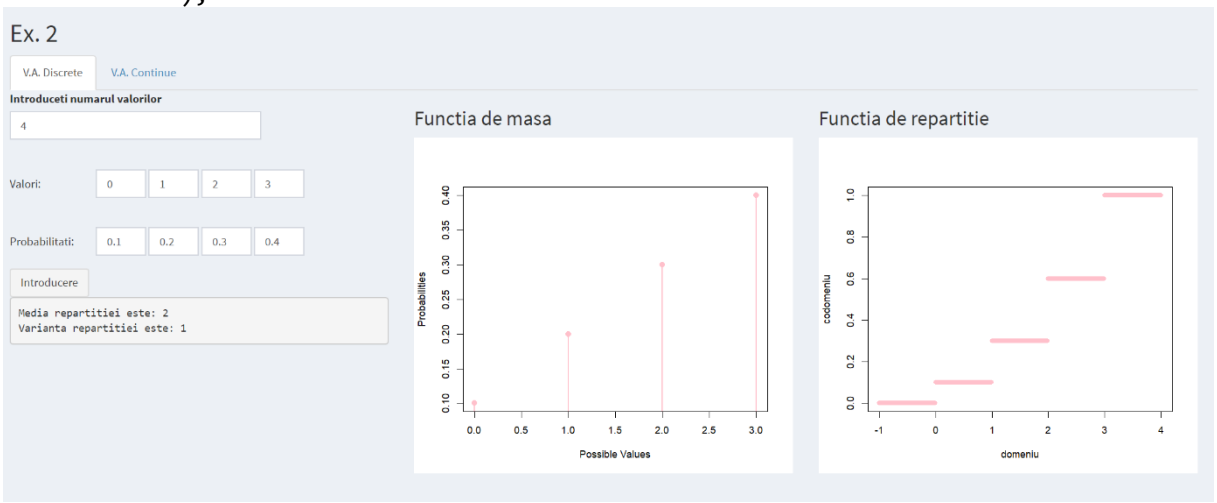
În această secțiune am construit o interfață prin care utilizatorul poate introduce manual atât repartiții de variabile aleatoare discrete, cât și continue, a căror funcție de distribuție respectă formatul:

$$f_x(x) = \begin{cases} g(x), & x \in [a,b] \\ 0, & altfel \end{cases},$$

unde *g(x)* este o funcție introdusă de utilizator.

Interfața a fost împărțită în 2 *tab-uri* folosind *tabsetPanel()*, unul pentru fiecare tip de variabilă aleatoare. Pentru cele de tip discret, utilizatorul introduce numărul de valori și valorile și probabilitățile respective.

```
tabPanel(
              title="V.A. Discrete",
              fluidRow(
                column(4,
                       numericInput("nrVal2", "Introduceti numarul valorilor", 1,
min = 1),
                       uiOutput("table2"),
                       actionButton("do2", "Introducere"),
                       verbatimTextOutput("proprietati2")),

                column(4,
                       h3("Functia de masa"),
                       plotOutput("fctmasa2")
                ),
                column(4,
                       h3("Functia de repartitie"),
                       plotOutput("fctrepart2"))
              )
            ),
```



Interfața pentru introducerea manuală a unei repartiții de v.a. discrete

Aceste date sunt introduse într-un obiect de tip *RV* din librăria *discreteRV*. Astfel, trasarea graficelor pentru funcțiile de masă și repartiție, cât și calculul mediei și variației devin triviale, datorită funcțiilor furnizate de această librărie:

```
output$fctmasa2 <- renderPlot({

  if (input$do2 == 0)
    return()

  values <- sapply(1:input$nrVal2, function(val){
    return(input[[paste("value", val)]])
  })

  probs <- sapply(1:input$nrVal2, function(val){
    return(input[[paste("prob", val)]])
  })

  X <- RV(values, probs)
  plot(X, col="pink")
})
```

Trasarea funcției de masă.

```
output$fctrepart2 <- renderPlot({

  if (input$do2 == 0)
    return()

  values <- sapply(1:input$nrVal2, function(val){
    return(input[[paste("value", val)]])
  })

  probs <- sapply(1:input$nrVal2, function(val){
    return(input[[paste("prob", val)]])
  })

  X <- RV(values, probs)
  domeniu <- seq(min(values)-1, max(values)+1, 0.01)
  codomeniu <- sapply(domeniu, function(nr) {
    return(P(X<=nr))
  })
  plot(domeniu, codomeniu, pch=19, col="pink")
})
```

Trasarea funcției de repartiție.

```
output$proprietati2 <- renderText({
  if (input$do2 == 0)
    return()

  values <- sapply(1:input$nrVal2, function(val){
    return(input[[paste("value", val)]])
  })
```

```
    probs <- sapply(1:input$nrVal2, function(val){
      return(input[[paste("prob", val)]])
    })

    X <- RV(values,probs)

    EX <- E(X)
    VAR <- V(X)

    paste("Media repartitiei este:", EX,
          "\nVarianta repartitiei este:", VAR)
  })
```

Calculul mediei și dispersiei folosind funcțiile din discreteRV.

Variabila aleatoare astfel creată este salvată într – o listă dintr – un obiect de tip *reactiveValues()*, pentru a fi ulterior utilizată în restul aplicației:

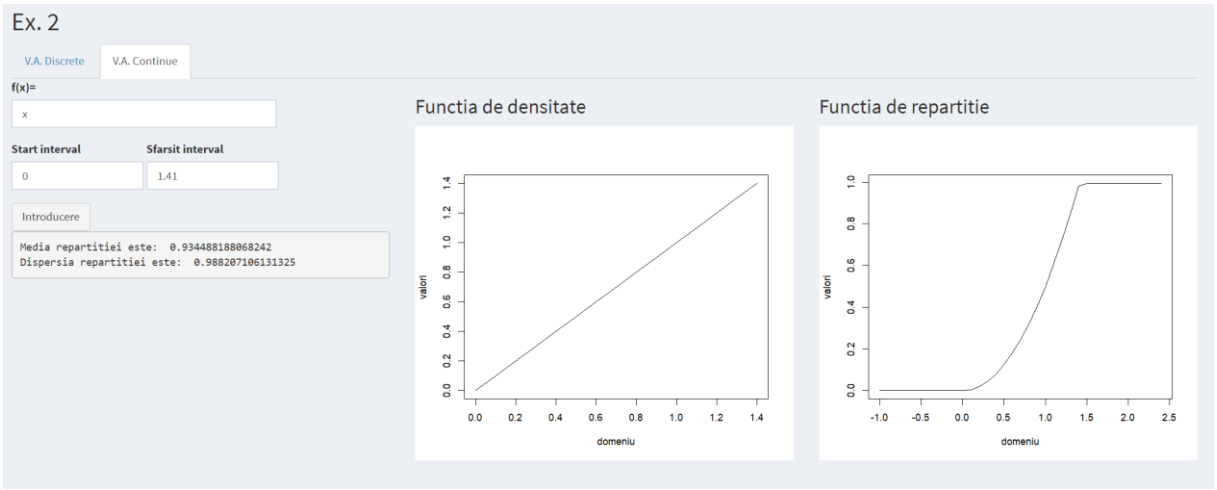
```
  observeEvent(input$do2, {
    values <- sapply(1:input$nrVal2, function(val){
      return(input[[paste("value", val)]])
    })

    probs <- sapply(1:input$nrVal2, function(val){
      return(input[[paste("prob", val)]])
    })
    X <- RV(values,probs)

    ex2_rvs$cnt <- ex2_rvs$cnt + 1
    ex2_rvs$arr[[ex2_rvs$cnt]] <-  X
  })
```

Variabila aleatoare astfel creată este salvată într – o listă dintr – un obiect de tip *reactiveValues()*, pentru a fi ulterior utilizată în restul aplicației:

Pentru variabilele aleatoare continue, utilizatorul introduce funcția *g(x)* mai sus menționată și intervalul respectiv.

```
tabPanel(
              title="V.A. Continue",
              fluidRow(
                column(4,
                       textInput("ex_2_f", "f(x)=", width = "70%"),
                       splitLayout(cellWidths = c("35%", "35%"),
                                   numericInput("ex_2_cstart", "Start interval",
value=0),
                                   numericInput("ex_2_cend", "Sfarsit interval",
value=0)),
                       actionButton("ex_2_btnc", "Introducere"),
                       verbatimTextOutput("proprietatic2")
                ),
                column(4,
                       h3("Functia de densitate"),
                       plotOutput("ex_2_cdens")
```

```
            ),
            column(4,
                   h3("Functia de repartitie"),
                   plotOutput("ex_2_crep")
            )
          )
        )
      )
```



Interfața pentru introducerea manuală a unei repartiții de v.a. continue

Utilizăm datele introduse pentru a crea intern funcția de distribuție și cea de repartiție:

```
      f <-  function(x) {
        check <- mapply(function(val){
          if(val < input$ex_2_cstart)
            return(FALSE)
          else if(val > input$ex_2_cend){
            return(FALSE)
          }
          else return(TRUE)
        }, x)
        x <- eval(parse(text=input$ex_2_f))
        x[!check] <- 0
        return(x)
      }
      Fx <- function(x) {
        return(
          mapply(function(x){
            return(integrate(f, -Inf, x)$value)
          }, x)
        )
      }
```

Validăm funcția primită – ne asigurăm că condiția $\int_{-\infty}^{\infty} f_x(x)\,dx = 1$ este îndeplinită:

```
      if(between(integrate(f, input$ex_2_cstart-1, input$ex_2_cend)$value, 0.98,
1.02) == FALSE){
        output$proprietatic2 <- renderText({
          paste("Functie gresita!")
```

```
  })
  return()
}
```

Putem astfel trasa graficul funcției de masă și de repartiție și să calculăm media și dispersia:

```
output$ex_2_cdens <- renderPlot({
  domeniu <- seq(input$ex_2_cstart, input$ex_2_cend, 0.1)
  valori <- f(domeniu)
  plot(domeniu, valori, type='l')
})
```

Trasarea funcției de masă.

```
output$ex_2_crep <- renderPlot({
  domeniu <- seq(input$ex_2_cstart - 1, input$ex_2_cend + 1, 0.1)
  valori <- Fx(domeniu)
  plot(domeniu, valori, type='l')
})
```

Trasarea funcției de repartiție.

```
output$proprietatic2 <- renderText({
  #E(X)
  media <- integrate(function(x){
    x * f(x)
  }, -Inf, Inf)$value
  #(E(X))^2
  media2 <- integrate(function(x){
    x * x * f(x)
  }, -Inf, Inf)$value
  paste("Media repartitiei este: ", media,
        "\nDispersia repartitiei este: ", media*media-media2)
})
```

Calculul mediei folosind formula $E(X) = \int_{-\infty}^{\infty} x f_x(x)\, dx$ *și varianța folosind formula* $Var(X) = \big(E(X)\big)^2 - E(X^2)$.

În final, adăugăm capetele de interval și șirul de caractere folosite în cadrul funcției de densitate într – o listă dintr – un obiect de tip *reactiveValues()*, pentru a fi ulterior utilizată în restul aplicației:

```
ex2_fcts$cnt <- ex2_fcts$cnt + 1
      ex2_fcts$arr[[ex2_fcts$cnt]] <- list(input$ex_2_f, input$ex_2_cstart,
input$ex_2_cend)
```

Repartițiile de variabile aleatoare astfel introduse sunt adăugate în galerie folosind *uiOutput()* în care afișăm un *box* pentru fiecare repartiție proprie, păstrând modul de calcul a graficelor, mediei și dispersiei.

```
output$ex2discrete <- renderUI({
  if(ex2_rvs$cnt == 0)
    return()

  lapply(1:ex2_rvs$cnt, function(a) {
    output[[sprintf("ex_1_#%d_plot1", a)]] <- renderPlot({
```

```
          plot(ex2_rvs$arr[[a]])
        })
        output[[sprintf("ex_1_#%d_plot2", a)]] <- renderPlot({
          X <- ex2_rvs$arr[[a]]
          domeniu <- seq(min(X)-1, max(X)+1, 0.01)
          codomeniu <- sapply(domeniu, function(nr) {
            return(P(X<=nr))
          })
          plot(domeniu, codomeniu, pch=19)
        })
        output[[sprintf("ex_1_#%d_text", a)]] <- renderText({
          X <- ex2_rvs$arr[[a]]
          paste("Media:", E(X), "\nDispersia:", V(X))
        })
        box(
          title=sprintf("V.A. Discreta #%d", a), status="primary",
solidHeader=TRUE, collapsible=TRUE,
          "Functia de masa",
          plotOutput(sprintf("ex_1_#%d_plot1", a), height = "290px"),
          "Functia de repartitie",
          plotOutput(sprintf("ex_1_#%d_plot2", a), height = "290px"),
          textOutput(sprintf("ex_1_#%d_text", a))
        )
      })
    })
```

Afișarea repartițiilor de variabile aleatoare discrete în galerie.

```
    output$ex2continue <- renderUI({
      if(ex2_fcts$cnt == 0)
        return()

      lapply(1:ex2_fcts$cnt, function(a) {
        func_string <- ex2_fcts$arr[[a]][[1]]
        cstart <- ex2_fcts$arr[[a]][[2]]
        cend <- ex2_fcts$arr[[a]][[3]]

        f <-  function(x) {
          check <- mapply(function(val){
            if(val < cstart)
              return(FALSE)
            else if(val > cend){
              return(FALSE)
            }
            else return(TRUE)
          }, x)
          x <- eval(parse(text=func_string))
          x[!check] <- 0
          return(x)
        }

        Fx <- function(x) {
```

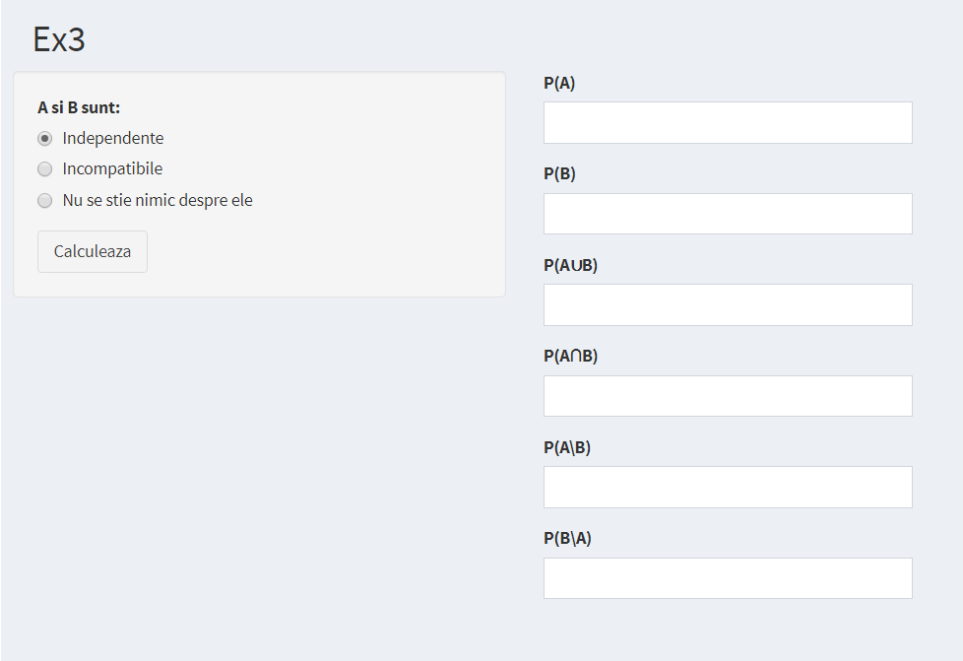
```
        return(
          mapply(function(x){
            return(integrate(f, -Inf, x)$value)
          }, x)
        )
      }

      output[[sprintf("ex_2_#%d_plot1", a)]] <- renderPlot({
        domeniu <- seq(cstart, cend, 0.1)
        valori <- f(domeniu)
        plot(domeniu, valori, type='l')
      })

      output[[sprintf("ex_2_#%d_plot2", a)]] <- renderPlot({
        domeniu <- seq(cstart - 1, cend + 1, 0.1)
        valori <- Fx(domeniu)
        plot(domeniu, valori, type='l')
      })

      media <- integrate(function(x){
        x * f(x)
      }, -Inf, Inf)$value

      media2 <- integrate(function(x){
        x * x * f(x)
      }, -Inf, Inf)$value

      box(
        title=sprintf("V.A. Continua #%d", a), status="warning",
solidHeader=TRUE, collapsible=TRUE,
        "Functia de densitate",
        plotOutput(sprintf("ex_2_#%d_plot1", a), height = "290px"),
        "Functia de repartitie",
        plotOutput(sprintf("ex_2_#%d_plot2", a), height = "290px"),
        paste("Media: ", media),
        paste("Dispersia: ", media * media - media2)
      )
    })
  })
```

Afișarea repartițiilor de variabile aleatoare continue în galerie.

Cerința 3

Pentru fiecare probabilitate ($P(A)$, $P(B)$, $P(A \cup B)$, $P(A \cap B)$, $P(A|B)$, $P(B|A)$) se va folosi câte o casetă de *numericInput*, iar pentru a alege dacă cele două evenimente sunt independente, incompatibile sau dacă nu se știe nimic despre ele se folosește *radioButtons*.

```
sidebarPanel(
    radioButtons("ex_3_tip", "A si B sunt:",
                 c("Independente" = "indep",
                   "Incompatibile" = "incomp",
                   "Nu se stie nimic despre ele" = "idk")),
    actionButton("ex_3_btn", "Calculeaza")
),

mainPanel(
    textOutput("ex_3_eroare"),
    numericInput("ex_3_PA", "P(A)", NULL),
    numericInput("ex_3_PB", "P(B)", NULL),
    numericInput("ex_3_PAorB", "P(A∪B)", NULL),
    numericInput("ex_3_PAandB", "P(A∩B)", NULL),
    numericInput("ex_3_PArB", "P(A\\B)", NULL),
    numericInput("ex_3_PBrA", "P(B\\A)", NULL)
)
```



Dacă evenimentele sunt incompatibile, atunci:

- $P(A \cup B) = P(A) + P(B)$
- $P(A \cap B) = 0$

- $P(A|B) = \frac{P(A \cap B)}{P(B)} = 0$
- $P(B|A) = \frac{P(A \cap B)}{P(A)} = 0$

Intersecția și probabilitățile condiționate se vor completa cu 0. Va trebui, deci, să se introducă două dintre cele 3 probabilități rămase ($P(A \cup B)$, $P(A)$ *sau* $P(B)$), a treia fiind calculată folosind prima formulă.

```
if(input$ex_3_tip == "incomp") {
    inter <- 0
    AcondB <- 0
    BcondA <- 0

    if(!is.na(pa) && !is.na(pb)) {
        reun <- pa + pb
        } else if(!is.na(pa) && !is.na(reun)){
        pb <- reun - pa
        } else if(!is.na(pb) && !is.na(reun)){
        pa <- reun - pb
        } else {ok <- FALSE}
}
```

Dacă evenimentele sunt independente, atunci:

- $P(A \cap B) = P(A) \cdot P(B)$
- $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \cdot P(B)}{P(B)} = P(A)$
- $P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A) \cdot P(B)}{P(A)} = P(B)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Se pot introduce fie $P(A)$ *sau* $P(A|B)$ cu $P(B)$ *sau* $P(B|A)$, fie $P(A \cap B)$ cu $P(A)$ *sau* $P(A|B)$ *sau* $P(B)$ *sau* $P(B|A)$, pentru a putea completa celelalte probabilități.

```
if(input$ex_3_tip == "indep") {
    if(!is.na(pa)) { AcondB <- pa } else if(!is.na(AcondB)) { pa <- AcondB }
    if(!is.na(pb)) { BcondA <- pb } else if(!is.na(BcondA)) { pb <- BcondA }

    if(!is.na(pa) && !is.na(pb)) {
    inter <- pa * pb
    reun <- pa + pb - inter
    } else if(!is.na(pa) && !is.na(inter)) {
    pb <- inter / pa
    BcondA <- pb
    reun <- pa + pb - inter
    } else if(!is.na(pb) && !is.na(inter)) {
    pa <- inter / pb
```

```
    AcondB <- pa
    reun <- pa + pb - inter
    } else { ok <- FALSE}
}
```

Dacă nu se știe nimic despre evenimente, se folosesc formulele:

- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- $P(B|A) = \frac{P(A \cap B)}{P(A)}$

În acest caz, este necesar să se introducă 3 dintre cele 6 probabilități pentru a putea determina probabilitățile rămase, cu 2 exceptii. Excepțiile sunt cazurile în care se introduc fie probabilitățile $P(A), P(A \cap B)$ și $P(B|A)$, fie $P(B), P(A \cap B)$ și $P(A|B)$.

Deoarece $P(B|A) = \frac{P(A \cap B)}{P(A)}$ și $P(A|B) = \frac{P(A \cap B)}{P(B)}$, cele 3 probabilități oferă doar două informații, fiind insuficiente pentru determinarea celorlalte valori.

```
if(!is.na(pa) && !is.na(pb) && !is.na(reun)) {
        inter <- reun - pa - pb
        AcondB <- inter / pb
        BcondA <- inter / pa

    } else if(!is.na(pa) && !is.na(pb) && !is.na(inter)) {
        reun <- pa + pb - inter
        AcondB <- inter / pb
        BcondA <- inter / pa

    } else if(!is.na(pa) && !is.na(pb) && !is.na(AcondB)) {
        inter <- AcondB * pa
        reun <- pa + pb - inter
        BcondA <- inter / pa

    } else if(!is.na(pa) && !is.na(pb) && !is.na(BcondA)) {
        inter <- BcondA * pb
        reun <- pa + pb - inter
        BcondA <- inter / pa

    } else if(!is.na(pa) && !is.na(reun) && !is.na(inter)) {
        pb <- reun - pa + inter
        AcondB <- inter / pb
        BcondA <- inter / pa

    } else if(!is.na(pa) && !is.na(reun) && !is.na(AcondB)) {
        pb <- (reun - pa) / (1-AcondB)
```

```
    inter <- pa + pb - reun
    BcondA <- inter / pa

} else if(!is.na(pa) && !is.na(reun) && !is.na(BcondA)) {
    inter <- BcondA * pa
    pb <- reun - pa + inter
    AcondB <- inter / pb

} else if(!is.na(pa) && !is.na(inter) && !is.na(AcondB)) {
    pb <- inter / AcondB
    reun <- pa + pb - inter
    BcondA <- inter / pb

} else if(!is.na(pa) && !is.na(AcondB) && !is.na(BcondA)) {
    inter <-  pa * BcondA
    pb <- inter / AcondB
    reun <-  pa + pb - inter

} else if(!is.na(pb) && !is.na(reun) && !is.na(inter)) {
    pa <- reun - pb + inter
    AcondB <- inter / pb
    BcondA <- inter / pa

} else if(!is.na(pb) && !is.na(reun) && !is.na(AcondB)) {
    inter <- AcondB * pb
    pa <- reun - pb + inter
    BcondA <- inter / pa

} else if(!is.na(pb) && !is.na(reun) && !is.na(BcondA)) {
    pa <- (reun - pb) / (1-BcondA)
    inter <- pa + pb - reun
    AcondB <- inter / pb

} else if(!is.na(pb) && !is.na(inter) && !is.na(BcondA)) {
    pa <- inter / BcondA
    reun <- pa + pb - inter
    AcondB <- inter / pa

} else if(!is.na(pb) && !is.na(AcondB) && !is.na(BcondA)) {
    inter <-  pb * AcondB
    pb <- inter / AcondB
    reun <-  pa + pb - inter

} else if(!is.na(reun) && !is.na(inter) && !is.na(AcondB)) {
    pb <- inter / AcondB
    pa <- reun - pb + inter
    BcondA <- inter / pa

} else if(!is.na(reun) && !is.na(inter) && !is.na(BcondA)) {
    pa <- inter / BcondA
    pb <- reun - pa + inter
    BcondA <- inter / pa
```

```
    } else if(!is.na(reun) && !is.na(AcondB) && !is.na(BcondA)) {
        inter <- reun / (1/BcondA + 1/AcondB - 1)
        pa <- inter / BcondA
        pb <- inter / AcondB

    } else if(!is.na(inter) && !is.na(AcondB) && !is.na(BcondA)) {
        pa <- inter / BcondA
        pb <- inter / AcondB
        reun <- pa + pb - inter

    } else { ok <- FALSE }
```

Variabila *ok* va deveni FALSE dacă sunt insuficiente date pentru a completa toate probabilitățile, caz în care se va afișa un mesaj care anunță acest lucru. Altel, după calcularea probabilităților, se actualizează valorile din numericInput-uri:

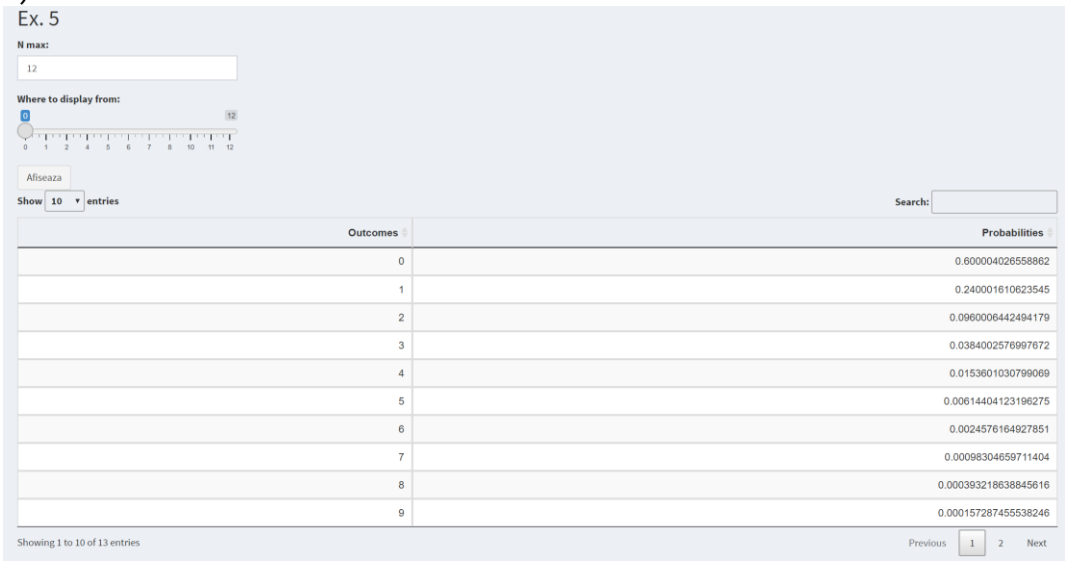
```
if(ok) {
    output$ex_3_eroare <- renderText("")
    updateNumericInput(session, "ex_3_PA", value = pa)
    updateNumericInput(session, "ex_3_PB", value = pb)
    updateNumericInput(session, "ex_3_PAorB", value = reun)
    updateNumericInput(session, "ex_3_PAandB", value = inter)
    updateNumericInput(session, "ex_3_PArB", value = AcondB)
    updateNumericInput(session, "ex_3_PBrA", value = BcondA)
} else {
    output$ex_3_eroare <- renderText("Tabelul nu se poate completa")
    updateNumericInput(session, "ex_3_PA", value = NULL)
    updateNumericInput(session, "ex_3_PB", value = NULL)
    updateNumericInput(session, "ex_3_PAorB", value = NULL)
    updateNumericInput(session, "ex_3_PAandB", value = NULL)
    updateNumericInput(session, "ex_3_PArB", value = NULL)
    updateNumericInput(session, "ex_3_PBrA", value = NULL)
}
```

Cerința 5

Pentru a afișa conținutul variabilei discrete, am folosit o tabelă de tip *DTOutput*, din librăria DT. De asemenea pentru a controla index-ul de la care începe afișarea, am folosit *sliderInput.*

```
tabItem(
        tabName = "ex5",
        fluidPage(
          titlePanel("Ex. 5"),
          uiOutput("ex_5_shownum"),
          sliderInput(inputId = "ex_5_begin",
                      label = "Where to display from:",
                      min = 0,
                      max = 99,
                      value = 0),
```

```
      actionButton('ex_5_btn', "Afiseaza"),
      textOutput("ex_5_txt"),
      DTOutput('ex_5_tbl')
    )
  )
```



Pentru a obține alegerea variabilei aleatoare din selector, am folosit *observeEvent.* În funcție de acesta, dacă repartiția sau variabila aleasă nu este de tip discret, se afișează un mesaj corespunzător. Dacă repartiția este de tip Bernoulli, slider-ul are setată valoarea maximă la 1 dacă p este 0 sau 1, și 2 dacă este altă valoare. Pentru repartiția binomială și hipergeometrică, valoarea maximă a slider-ului se setează după parametrii n și respectiv k. Dacă repartiția este de tip geometric sau Poisson, se afișează și un *numericInput* în care se va furniza limita superioară a index-ului. Pentru variabilele aleatoare discrete, valoarea maximă a slider-ului este lungimea listei de evenimente furnizată de discreteRV

```
observeEvent(input$va_selected, {
    tip <- input$va_selected
    if(tip == "unif" || tip == "norm" || tip == "exp" || startsWith(tip, "c")){
      output$ex_5_txt <- renderText({
        paste("VA aleasa nu este discreta!")
      })
      return()
    }
    if(tip == "bern"){
      if(input$ex_1_bern_p == 0 || input$ex_1_bern_p == 1){
        ex_5_max$val <- 1
        updateSliderInput(session, "ex_5_begin", min=0, max=0, value=0)
      }
      else{
        ex_5_max$val <- 2
        updateSliderInput(session, "ex_5_begin", min=0, max=1, value=0)
      }
    }
    if(tip == "binom"){
      ex_5_max$val <- input$ex_1_binom_n + 1
```

```
        updateSliderInput(session, "ex_5_begin", min=0, max=input$ex_1_binom_n,
value=0)
      }
      if(tip == "hgeom"){
        ex_5_max$val <- input$ex_1_hgeom_k + 1
        updateSliderInput(session, "ex_5_begin", min=0, max=input$ex_1_hgeom_k,
value=0)
      }
      if(startsWith(tip, "d")){
        nr <- strtoi(substring(tip, 2))
        ex_5_max$val <- length(probs(ex2_rvs$arr[[nr]]))
        updateSliderInput(session, "ex_5_begin", min=0, max=ex_5_max$val, value=0)
      }
      if(tip == "geom" || tip == "pois"){
        output$ex_5_shownum <- renderUI({
          numericInput("ex_5_dim", "N max:", 0)
        })
      }
      else {
        output$ex_5_shownum <- renderUI({
        })
      }
    })
```

Apoi, pentru a afișa datele, am folosit *observeEvent* și un buton. Se repetă verificările de mai sus pentru repartițiile uniforme, normale, exponențiale, Bernoulli și variabilele aleatoare continue. Fiindcă discreteRV ignoră evenimentele cu probabilități 0, pentru repartiția geometrică și repartiția Poisson valoarea maximă a slider-ului este reprezentată de minimul dintre limita superioară a index-ului și lungimea listei de evenimente furnizată de discreteRV. Pentru repartiția hipergeometrică valoarea maximă a slider-ului este reprezentată de minimul dintre limita superioară a index-ului și parametrul k.

Pentru a afișa datele, creez un dataframe din variabila aleatoare X , cu ajutorul funcțiilor *outcomes* și *probs,* pe care aplic *tail* pentru a regla index-ul de început în funcție de slider, pe care aplic *datatable* pentru a crea obiectul de afișat.

```
observeEvent(input$ex_5_btn, {
        tip <- input$va_selected
        if(tip == "unif" || tip == "norm" || tip == "exp" || startsWith(tip,
"c")){
          output$ex_5_txt <- renderText({
            paste("VA aleasa nu este discreta!")
          })
          return()
        }
        else {
          output$ex_5_txt <- renderText({
            paste("")
          })
        }
```

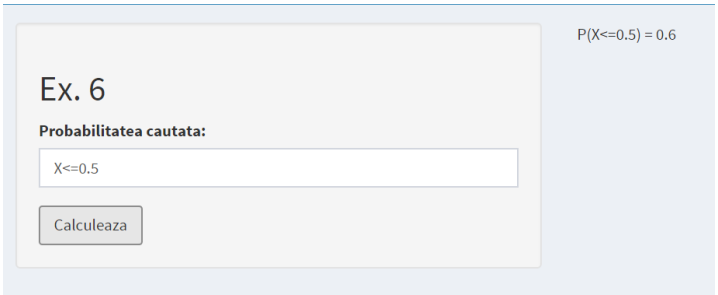
```
        if(tip == "bern"){
          p <- input$ex_1_bern_p
          X <- RV(c(0,1), c(1-p, p))
          if(input$ex_1_bern_p == 0 || input$ex_1_bern_p == 1){
            ex_5_max$val <- 1
            updateSliderInput(session, "ex_5_begin", min=0, max=0, value=0)
          }
          else{
            ex_5_max$val <- 2
            updateSliderInput(session, "ex_5_begin", min=0, max=1, value=0)
          }
        }

        if(tip == "binom"){
          X <- RV(0:input$ex_1_binom_n,
dbinom(0:input$ex_1_binom_n,input$ex_1_binom_n,input$ex_1_binom_p))
          ex_5_max$val <- input$ex_1_binom_n + 1
          updateSliderInput(session, "ex_5_begin", min=0, max=input$ex_1_binom_n,
value=0)
        }

        if(tip == "geom"){
          X <- RV(0:input$ex_5_dim, dgeom(0:input$ex_5_dim, input$ex_1_geom_p))
          observeEvent(input$ex_5_dim, {
            if(input$ex_5_dim>length(outcomes(X))){ex_5_max$val <-
length(outcomes(X))+1}else{ex_5_max$val <- input$ex_5_dim+1}
            updateSliderInput(session, "ex_5_begin", min=0, max=ex_5_max$val-1,
value=0)
          })
        }

        if(tip == "hgeom"){
          X <- RV(0:input$ex_1_hgeom_k,
dhyper(0:input$ex_1_hgeom_k,input$ex_1_hgeom_m,input$ex_1_hgeom_n,input$ex_1_hgeom
_k))
          if(input$ex_1_hgeom_k>length(outcomes(X))){ex_5_max$val <-
length(outcomes(X))+1}else{ex_5_max$val <- input$ex_1_hgeom_k+1}
          updateSliderInput(session, "ex_5_begin", min=0, max=ex_5_max$val-1,
value=0)
        }

        if(tip == "pois"){
          X <- RV(0:input$ex_5_dim, dpois(0:input$ex_5_dim, input$ex_1_pois_l))
          observeEvent(input$ex_5_dim, {
            if(input$ex_5_dim>length(outcomes(X))){ex_5_max$val <-
length(outcomes(X))+1}else{ex_5_max$val <- input$ex_5_dim+1}
            updateSliderInput(session, "ex_5_begin", min=0, max=ex_5_max$val-1,
value=0)
```

```
        })
      }

      if(startsWith(tip, "d")){
        nr <- strtoi(substring(tip, 2))
        X <- ex2_rvs$arr[[nr]]
      }

      Outcomes <-  outcomes(X)
      Probabilities <-  probs(X)
      df <- data.frame(Outcomes,Probabilities)
      observeEvent(input$ex_5_begin,{
        df <- tail(df,ex_5_max$val-input$ex_5_begin)
        output$ex_5_tbl = renderDT(
          datatable(df, options=list(pageLength= 10),rownames= FALSE)

        )
      })
    })
```

Cerința 6

Pentru această cerință, am folosit un câmp de text pentru a primi evenimentul. Variabila aleatoare utilizată în cadrul acestei secțiuni este aleasă folosind selectorul din header-ul aplicației.

```
tabItem(
      tabName = "ex6",
      fluidPage(
        sidebarLayout(
          sidebarPanel(
            titlePanel("Ex. 6"),
            textInput("ex_6_in",
"Probabilitatea cautata: "),
            actionButton("ex_6_btn",
"Calculeaza")
          ),
          mainPanel(
            textOutput("ex_6_out")
          )
        )
      )
    )
```



Interfața cerinței 6 – în exemplu este utilizată o variabilă aleatoare de repartiție Bernoulli cu parametrul p = 0,4.

Modalitatea de calcul diferă în funcție de tipul variabilei aleatoare alese: în cazul variabilelor discrete, am folosit librăria *discreteRV* pentru a calcula direct probabilitatea evenimentului cerut:

```
if(tip == "bern"){
          p <- input$ex_1_bern_p
          X <- RV(c(0,1), c(1-p, p))
        }

        if(tip == "binom"){
          X <- RV(0:input$ex_1_binom_n,
dbinom(0:input$ex_1_binom_n,input$ex_1_binom_n,input$ex_1_binom_p))
        }

        if(tip == "geom"){
          X <- RV(0:input$ex_5_dim, dgeom(0:input$ex_5_dim, input$ex_1_geom_p))
        }

        if(tip == "hgeom"){
          X <- RV(0:input$ex_1_hgeom_k,
dhyper(0:input$ex_1_hgeom_k,input$ex_1_hgeom_m,input$ex_1_hgeom_n,input$ex_1_hgeom
_k))
        }

        if(tip == "pois"){
          X <- RV(0:input$ex_5_dim, dpois(0:input$ex_5_dim, input$ex_1_pois_l))
        }

        if(startsWith(tip, "d")){
          nr <- strtoi(substring(tip, 2))
          X <- ex2_rvs$arr[[nr]]
        }
        output$ex_6_out <- renderText({
          paste("P(", input$ex_6_in, ") = ", P(eval(parse(text = input$ex_6_in))),
sep="")
        })
```

Construcția internă a variabilei aleatoare X folosind librăria discreteRV *și datele introduse.*

```
output$ex_6_out <- renderText({
          paste("P(", input$ex_6_in, ") = ", P(eval(parse(text = input$ex_6_in))),
sep="")
        })
```

Calculul propriu-zis al probabilității folosind funcția P() din librăria discreteRV.

În cazul variabilelor aleatoare de tip continuu, am extras manual din șirul de caractere operația cerută și valoarea de referință:

```
instr <- input$ex_6_in
        if(tip == "unif"){
          a <- input$ex_1_unif_a
          b <- input$ex_1_unif_b
          p <- 0
          if(substring(instr, 2, 3) == "<="){
            val <- as.numeric(substring(instr, 4))
            p <- punif(val, a, b)
          }
```

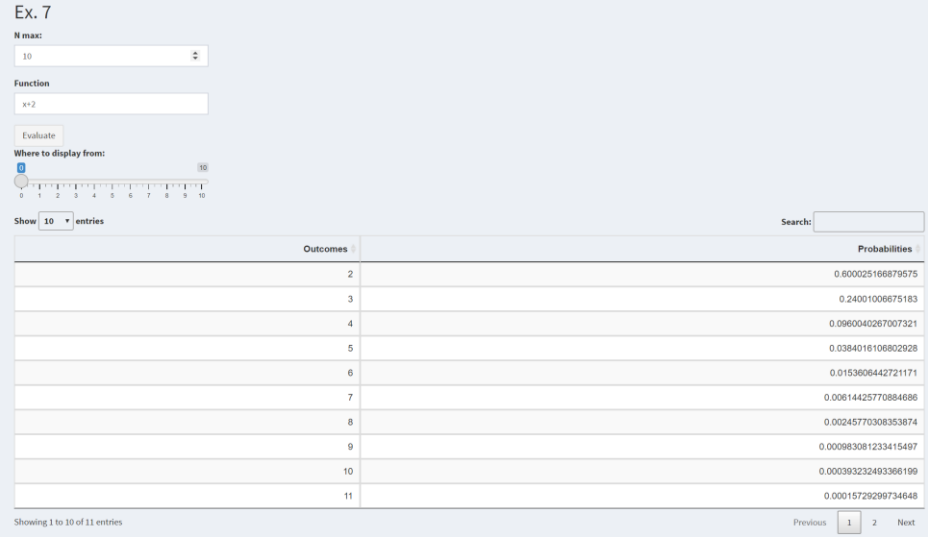
```
    else if(substring(instr, 2, 2) == "<"){
      val <- as.numeric(substring(instr, 3))
      p <- punif(val, a, b)
    }
    else if(substring(instr, 2, 3) == ">="){
      val <- as.numeric(substring(instr, 4))
      p <- 1 - punif(val, a, b)
    }
    else if(substring(instr, 2, 2) == ">"){
      val <- as.numeric(substring(instr, 3))
      p <- 1 -punif(val, a, b)
    }
    else if(substring(instr, 2, 3) == "=="){
      p <- 0
    }
    else {
      output$ex_6_out <- renderText({
        "Operatie invalida!"
      })
      return()
    }
    output$ex_6_out <- renderText({
      paste("P(", instr, ") = ", p, sep="")
    })
  }
```

Calculul probabilității evenimentului în cazul repartiției de tip Uniform.

Cerința 7

Pentru a afișa conținutul variabilei discrete, am folosit o tabelă de tip DTOutput, din librăria DT. De asemenea pentru a controla index-ul de la care începe afișarea, am folosit sliderInput. Pentru a prelua funcția, există un textInput și un actionButton pentru a declanșa evaluarea.

```
tabItem(
        tabName = "ex7",
        fluidPage(
          titlePanel("Ex. 7"),
          uiOutput("ex_7_shownum"),
          textInput(inputId = "functionInput", "Function"),
          actionButton(inputId = "ex_7_btn", "Evaluate"),
          sliderInput(inputId = "ex_7_begin",
                      label = "Where to display from:",
                      min = 0,
                      max = 99,
                      value = 0),
          textOutput("ex_7_txt"),
          DTOutput('ex_7_tbl')
        ),
      )
```



Pentru a obține alegerea variabilei aleatoare din selector, am folosit observeEvent. În funcție de acesta, dacă repartiția sau variabila aleasă nu este de tip discret, se afișează un mesaj corespunzător. Dacă repartiția este de tip Bernoulli, slider-ul are setată valoarea maximă la 1 dacă p este 0 sau 1, și 2 dacă este altă valoare. Pentru repartiția binomială și hipergeometrică, valoarea maximă a slider-ului se setează după parametrii n și respectiv k. Dacă repartiția este de tip geometric sau Poisson, se afișează și un numericInput în care se va furniza limita superioară a index-ului. Pentru variabilele aleatoare discrete, valoarea maximă a slider-ului este lungimea listei de evenimente furnizată de discreteRV

```
observeEvent(input$va_selected, {

    tip <- input$va_selected
    if(tip == "unif" || tip == "norm" || tip == "exp" || startsWith(tip, "c")){
      output$ex_7_txt <- renderText({
        paste("VA aleasa nu este discreta!")
      })
      return()
    }
    if(tip == "bern"){
      if(input$ex_1_bern_p == 0 || input$ex_1_bern_p == 1){
        ex_7_max$val <- 1
        updateSliderInput(session, "ex_7_begin", min=0, max=0, value=0)
      }
      else{
        ex_7_max$val <- 2
        updateSliderInput(session, "ex_7_begin", min=0, max=1, value=0)
      }
    }
    if(tip == "binom"){
      ex_7_max$val <- input$ex_1_binom_n + 1
      updateSliderInput(session, "ex_7_begin", min=0, max=input$ex_1_binom_n,
value=0)
    }
    if(tip == "hgeom"){
      ex_7_max$val <- input$ex_1_hgeom_k + 1
```

```
        updateSliderInput(session, "ex_7_begin", min=0, max=input$ex_1_hgeom_k,
value=0)
      }
      if(startsWith(tip, "d")){
        nr <- strtoi(substring(tip, 2))
        ex_7_max$val <- length(probs(ex2_rvs$arr[[nr]]))
        updateSliderInput(session, "ex_7_begin", min=0, max=ex_7_max$val, value=0)
      }
      if(tip == "geom" || tip == "pois"){
        output$ex_7_shownum <- renderUI({
          numericInput("ex_7_dim", "N max:", 0)
        })
      }
      else {
        output$ex_7_shownum <- renderUI({
        })
      }
    })
```

Apoi, pentru a afișa datele, am folosit observeEvent și un buton. Se repetă verificările de mai sus pentru repartițiile uniforme, normale, exponențiale, Bernoulli și variabilele aleatoare continue. Fiindcă discreteRV ignoră evenimentele cu probabilități 0, pentru repartiția geometrică și repartiția Poisson valoarea maximă a slider-ului este reprezentată de minimul dintre limita superioară a index-ului și lungimea listei de evenimente furnizată de discreteRV. Pentru repartiția hipergeometrică valoarea maximă a slider-ului este reprezentată de minimul dintre limita superioară a index-ului și parametrul k.

Pentru a obține funcția dată de utilizator din câmpul text, am folosit eval și parse. Apoi am aplicat variabilei aleatoare X funcția obținută.

Pentru a afișa datele, creez un dataframe din variabila aleatoare X , cu ajutorul funcțiilor outcomes și probs, pe care aplic tail pentru a regla index-ul de început în funcție de slider, pe care aplic datatable pentru a crea obiectul de afișat.

```
observeEvent(input$ex_7_btn, {
        tip <- input$va_selected
        if(tip == "unif" || tip == "norm" || tip == "exp" || startsWith(tip,
"c")){
          output$ex_7_txt <- renderText({
            paste("VA aleasa nu este discreta!")
          })
          return()
        }
        else {
          output$ex_7_txt <- renderText({
            paste("")
          })
        }

        if(tip == "bern"){
```

```
          p <- input$ex_1_bern_p
          X <- RV(c(0,1), c(1-p, p))
          if(input$ex_1_bern_p == 0 || input$ex_1_bern_p == 1){
            ex_7_max$val <- 1
            updateSliderInput(session, "ex_7_begin", min=0, max=0, value=0)
          }
          else{
            ex_7_max$val <- 2
            updateSliderInput(session, "ex_7_begin", min=0, max=1, value=0)
          }
        }

        if(tip == "binom"){
          X <- RV(0:input$ex_1_binom_n,
dbinom(0:input$ex_1_binom_n,input$ex_1_binom_n,input$ex_1_binom_p))
          ex_7_max$val <- input$ex_1_binom_n + 1
          updateSliderInput(session, "ex_7_begin", min=0, max=input$ex_1_binom_n,
value=0)
        }

        if(tip == "geom"){
          X <- RV(0:input$ex_7_dim, dgeom(0:input$ex_7_dim, input$ex_1_geom_p))
          observeEvent(input$ex_7_dim, {
            if(input$ex_7_dim>length(outcomes(X))){ex_7_max$val <-
length(outcomes(X))+1}else{ex_7_max$val <- input$ex_7_dim+1}
            updateSliderInput(session, "ex_7_begin", min=0, max=ex_7_max$val-1,
value=0)
          })
        }

        if(tip == "hgeom"){
          X <- RV(0:input$ex_1_hgeom_k,
dhyper(0:input$ex_1_hgeom_k,input$ex_1_hgeom_m,input$ex_1_hgeom_n,input$ex_1_hgeom
_k))
          if(input$ex_1_hgeom_k>length(outcomes(X))){ex_7_max$val <-
length(outcomes(X))+1}else{ex_7_max$val <- input$ex_1_hgeom_k+1}
          updateSliderInput(session, "ex_7_begin", min=0, max=ex_7_max$val-1,
value=0)
        }

        if(tip == "pois"){
          X <- RV(0:input$ex_7_dim, dpois(0:input$ex_7_dim, input$ex_1_pois_l))
          observeEvent(input$ex_7_dim, {
            if(input$ex_7_dim>length(outcomes(X))){ex_7_max$val <-
length(outcomes(X))+1}else{ex_7_max$val <- input$ex_7_dim+1}
            updateSliderInput(session, "ex_7_begin", min=0, max=ex_7_max$val-1,
value=0)
          })
        }
```

```
        if(startsWith(tip, "d")){
          nr <- strtoi(substring(tip, 2))
          X <- ex2_rvs$arr[[nr]]
        }

        f <-  function(x) {eval(parse(text = input$functionInput))}
        X <- f(X)

        Outcomes <-  outcomes(X)
        Probabilities <-  probs(X)
        df <- data.frame(Outcomes,Probabilities)
        observeEvent(input$ex_7_begin,{
          df <- tail(df,ex_7_max$val-input$ex_7_begin)
          output$ex_7_tbl = renderDT(
             datatable(df, options=list(pageLength= 10),rownames= FALSE)
          )
        })
      })
```

Cerința 8

Pentru această cerință, interfața conține un *textInput* pentru a prelua funcția utilizatorului, *actionButton* pentru a declanșa evaluarea, și un *textOutput* pentru a afișa media și dispersia, unde este cazul.

```
tabItem(
        tabName = "ex8",
        fluidPage(
          titlePanel("Ex. 8"),
          textInput(inputId = "ex8_functionInput", "Function"),
          uiOutput("ex_8_shownum"),
          actionButton('ex_8_eval', "Evaluate"),
          textOutput("ex_8_med_disp"),
        )
      )
```

Pentru a obține alegerea variabilei aleatoare din selector, am folosit *observeEvent*. În funcție de tipul repartiției alese, am calculat media aplicând funcția furnizată de utilizator (preluată folosind *eval* și *parse*) formulelor de calcul de la cerința 1.

```
    observeEvent(input$va_selected, {
      tip <- input$va_selected
      if (tip == "bern") {
        observeEvent(input$ex_8_eval, {
          f = function(x) {eval(parse(text = input$ex8_functionInput))}
          p <- input$ex_1_bern_p
          output$ex_8_med_disp <- renderText(
            {paste("Media este: ",f(p))}
```

```
    )
  })
}
if (tip == "binom") {
  observeEvent(input$ex_8_eval, {
    f = function(x) {eval(parse(text = input$ex8_functionInput))}
    p <- input$ex_1_binom_p
    n <- input$ex_1_binom_n
    output$ex_8_med_disp <- renderText(
      {paste("Media este: ",f(p*n))}
    )
  })
}
if (tip == "geom") {
  observeEvent(input$ex_8_eval, {
    f = function(x) {eval(parse(text = input$ex8_functionInput))}
    p <- input$ex_1_geom_p
    output$ex_8_med_disp <- renderText(
      {paste("Media este: ",f((1-p)/p))}
    )
  })
}
if (tip == "hgeom") {
  observeEvent(input$ex_8_eval, {
    f = function(x) {eval(parse(text = input$ex8_functionInput))}
    n <- input$ex_1_hgeom_n
    m <- input$ex_1_hgeom_m
    k <- input$ex_1_hgeom_k
    p <- m / (m + n)
    output$ex_8_med_disp <- renderText(
      {paste("Media este: ",f(k*p))}
    )
  })
}
if (tip == "pois") {
  observeEvent(input$ex_8_eval, {
    f = function(x) {eval(parse(text = input$ex8_functionInput))}
    l <- input$ex_1_pois_l
    output$ex_8_med_disp <- renderText(
      {paste("Media este: ",f(l))}
    )
  })
}
if (tip == "unif") {
  observeEvent(input$ex_8_eval, {
    f = function(x) {eval(parse(text = input$ex8_functionInput))}
    a <- input$ex_1_unif_a
    b <- input$ex_1_unif_b
    output$ex_8_med_disp <- renderText(
```

```
        {paste("Media este: ",f(0.5 * (a + b)))}
      )
    })
  }
  if (tip == "exp") {
    observeEvent(input$ex_8_eval, {
      f = function(x) {eval(parse(text = input$ex8_functionInput))}
      l <- input$ex_1_exp_l
      output$ex_8_med_disp <- renderText(
        {paste("Media este: ",f(1 / l))}
      )
    })
  }
  if (tip == "norm") {
    observeEvent(input$ex_8_eval, {
      f = function(x) {eval(parse(text = input$ex8_functionInput))}
      output$ex_8_med_disp <- renderText(
        {paste("Media este: ",f(input$ex_1_norm_m))}
      )
    })
  }
```

Pentru variabilele aleatoare discrete introduse de utilizator, am aplicat variabilei X funcția furnizată, după care am calculat media și dispersia folosind funcțiile din discreteRV *E* și *V*.

```
if(startsWith(tip, "d")) {
        observeEvent(input$ex_8_eval, {
          f = function(x) {eval(parse(text = input$ex8_functionInput))}
          nr <- strtoi(substring(tip, 2))
          X <- ex2_rvs$arr[[nr]]
          output$ex_8_med_disp <- renderText(
            {paste("Media este: ",E(f(X))," Dispersia este: ",V(f(X)))}
          )
        })
      }
```

Pentru variabilele aleatoare continue introduse de utilizator, obțin funcția g folosind *eval* și *parse,* iar funcția f este preluată de la cerința 2, împreună cu captele intervalului furnizate. Pentru funcțiile f și g se verifică că valorile primite de acestea sunt în intervalul precizat la cerința 2. Valorile ce sunt găsite înafara intervalului sunt înlocuite cu 0. Apoi se verifică că $\int_{cstart}^{cend} g(f(x))$ se apropie de 1. Media și dispersia se calculează folosind formulele generale.

```
if(startsWith(tip, "c")) {
        observeEvent(input$ex_8_eval, {

          nr <- strtoi(substring(tip, 2))
          cstart <- ex2_fcts$arr[[nr]][[2]]
          cend <- ex2_fcts$arr[[nr]][[3]]
```

```
          func_string <- ex2_fcts$arr[[nr]][[1]]
          g  <-  function(x) {
            check <- mapply(function(val){
              if(val < cstart)
                return(FALSE)
              else if(val > cend){
                return(FALSE)
              }
              else return(TRUE)
            }, x)
            x<-eval(parse(text = input$ex8_functionInput))
            x[!check] <- 0
            return(x)
          }
          f <-  function(x) {
            check <- mapply(function(val){
              if(val < cstart)
                return(FALSE)
              else if(val > cend){
                return(FALSE)
              }
              else return(TRUE)
            }, x)
            x <- eval(parse(text=func_string))
            x[!check] <- 0
            return(x)
          }
          if(between(integrate(function(x){g(f(x))}, cstart, cend)$value, 0.98,
1.02) == FALSE){
            output$ex_8_med_disp <- renderText({
              paste("Functie gresita!")
            })
            return()
          }
          media <- integrate(function(x){
            x * (g(f(x)))
          }, cstart, cend)$value

          media2 <- integrate(function(x){
            x * x * (g(f(x)))
          }, cstart, cend)$value
          output$ex_8_med_disp <- renderText(
            {paste("Media este: ",media," Dispersia este: ",media * media -
media2)}
          )
        })
      }
```

La această cerință, pentru repartițiile standard se calculează doar media, fiind dificil de implementat parsarea funcției furnizate pentru dispersie.

Pentru variabilele aleatoare furnizate de utilizator, atât discrete cât și continue se calculează atât media cât și dispersia.

Cerința 9

Repartiția comună va fi afișată sub forma unui tabel alcătuit din *numericInput*-uri.Valorile introduse pe prima coloană reprezintă outcome-urile v.a. X, iar cele introduse pe prima linie reprezintă outcome-urile v.a. Y. Ultima linie și ultima coloană vor conține probabilitățile repartițiilor marginale.

```
tabel <- lapply(1:(input$nrValX9+2), function(val){

        line <- lapply(1:(input$nrValY9+2), function(val2){
          if(val == 1 && val2 == 1) {
            div(style="display: inline-block; width: 70px; text-align:center;",
                p("X\\Y"))
          } else if(val == 1 && val2 == (input$nrValY9+2)) {
          div(style="display: inline-block; width: 70px; text-align:center;",
              p("Pi"))
          } else if(val == (input$nrValX9+2) && val2 == 1) {
            div(style="display: inline-block; width: 70px; text-align:center;",
                p("Qj"))
          } else {
            div(style="display: inline-block; width: 70px;",
                numericInput(paste("tabRepCom", val, val2, sep=""),"",
                             NULL, min = 0))
          }
        })

        tags$div(
          div(style="display: inline-block; width: 100px; margin-bottom:0;
                    margin-top:0;"),
          tagList(line))
    })
```

Datele extrase din tabel sunt adăugate în matricea *mat*, care conține repartiția comună, excluzând outcome-urile celor două variabile aleatoare.

```
for(i in 2:(input$nrValX9+2)) {
      mat <- rbind(mat, sapply(2:(input$nrValY9+2), function(j){
        input[[paste("tabRepCom", i, j, sep="")]]}))
    }
```

Completarea tabelului se realizează astfel:

Dacă pe o linie există o singură casetă necompletată, atunci valoarea acesteia se poate deduce. Dacă valoarea reprezintă o probabilitate din repartiția marginală (adică respectiva casetă se află pe ultima coloană), atunci va fi egală cu suma celorlalte elemente de pe linie. Altfel, aceasta se va calcula ca fiind valoarea de pe ultima coloană minus suma celorlalte elemente de pe linie. Se procedează analog pentru coloane.

Pe parcursul acestui proces, variabila booleană *dif* va reține dacă s-au făcut modificări asupra matricei. Dacă după o parcurgere aceasta are valoarea FALSE, atunci fie a fost completat tabelul, fie nu se mai poate completa.

```
nrlin <- nrow(mat)
nrcol <- ncol(mat)
mat[nrlin, nrcol] = 1

dif <- TRUE
while(dif){
    dif <- FALSE
    for(i in 1:nrlin) {
    # indicii coloanelor de pe linia i pe care se afla NA:
    poz <- which(is.na(mat[i,]))
    if(length(poz) == 1) {
        dif <- TRUE
        mat[i,poz] <- ifelse(poz == nrcol,
                             sum(mat[i,], na.rm = TRUE),
                             mat[i,nrcol] - sum(mat[i,1:(nrcol-1)], na.rm = TRUE))
        }
    }

    for(j in 1:nrcol) {
    poz <- which(is.na(mat[,j]))
    if(length(poz) == 1) {
        dif <- TRUE
        mat[poz, j] <- ifelse(poz == nrlin,
                             sum(mat[,j], na.rm = TRUE),
                             mat[nrlin,j] - sum(mat[1:(nrlin-1),j], na.rm = TRUE))
        }
    }
}
```

Pentru a verifica dacă tabelul este completat, se verifică daca matricea conține valori NA. În caz afirmativ, se va afișa un mesaj. Altfel, datele obținute în matrice vor fi rescrise în interfață, folosind *updateNumericInput*.

```
for(i in 1:nrlin) {
```

```
    for(j in 1:nrcol) {
        updateNumericInput(session, paste("tabRepCom",i+1, j+1, sep=""),
        value = mat[i,j])
    }
}
```

Pentru calculul repartițiilor marginale se preiau valorile din prima coloană și prima linie din tabel pentru outcome-uri și ultima coloană, respectiv ultima linie din matricea *mat* pentru probabilități. Se vor crea două variabile aleatoare folosind pachetul discreteRV, și vom aplica funcțiile E și V pentru determinarea mediei si variației lui X și Y.

```
#crearea variabilelor X si Y
    valX <- sapply(2:(input$nrValX9+1), function(ind) {
      input[[paste("tabRepCom", ind, 1, sep="")]]
    })

    valY <- sapply(2:(input$nrValY9+1), function(ind) {
      input[[paste("tabRepCom", 1, ind, sep="")]]
    })

    probX <- mat[1:(nrlin-1),nrcol]
    probY <- mat[nrlin,1:(nrcol-1)]

#repart marginale
    X <- RV(valX, probX)
    Y <- RV(valY, probY)

# mediile
    EX <- E(X)
    EY <- E(Y)

# dispersiile
    VX <- V(X)
    VY <- V(Y)
```

Pentru calculul covarianței, se folosește formula:

$$\text{cov}(X,Y) = E(X \cdot Y) \; - \; E(X) \cdot E(Y)$$

Pentru a calcula produsul $X \cdot Y$, am realizat o funcție ce primește 3 parametri: out_rep_X (outcome-urile v.a. X), out_rep_Y (outcome-urile v.a. Y), prob_rep (probabilitățile din repartiția comună sub formă de vector). Variabila allOutcomes va conține produsele valorilor lui X și Y în care pot exista duplicate, iar variabila outcomes va conține valorile unice din allOutcomes. Probabilitățile se calculează ca fiind suma probabilităților pentru fiecare outcome unic.

```
prod <- function(out_rep_X, out_rep_Y, prob_rep) {
    allOutcomes <- as.vector(sapply(out_rep_X, function(x){x*out_rep_Y}))
```

```
    outcomes <- unique(allOutcomes)

    probs <- sapply(outcomes, function(out){
      indices <- which(allOutcomes==out)
      return(sum(prob_rep[indices])) })
    return(RV(outcomes, probs))
}

XtimesY <- prod(valX, valY, as.vector(t(mat[1:(nrlin-1), 1:(nrcol-1)])))
cov <- E(XtimesY) - E(X)*E(Y)
```

În calculul coeficientului de corelație am folosit formula:

$$\rho(X,Y) = \frac{\text{cov}(X,Y)}{\sqrt{Var(X) \cdot Var(Y)}}$$

```
#coeficientul de corelatie
coef <- cov/sqrt(V(X)*V(Y))
```

Toate aceste proprietăți vor fi afișate astfel:

- V.a. X și Y vor fi afișate sub formă de tabele, utilizând *uiOutput*;
- Toate celelalte proprietăți reprezintă casete de text, pentru care am folosit *textOutput*.

```
firstRowX <- lapply(valX, function(val) {tags$th(val)})
secondRowX <- lapply(probX, function(val) {tags$td(val)})
firstRowY <- lapply(valY, function(val) {tags$th(val)})
secondRowY <- lapply(probY, function(val) {tags$td(val)})

output$proprietati9 <- renderUI({
    tags$div(
        h3("Proprietati"),
        p("Repartitiile marginale"),
        tags$table(
            tags$tr(tags$th("X:"),
                    firstRowX),
            tags$tr(tags$td(""),
                    secondRowX)
        ),
        tags$table(
            tags$tr(tags$th("Y:"),
                    firstRowY),
            tags$tr(tags$td(""),
                    secondRowY)
        )
    )
})
```

```
output$medieX9 <- renderText({ paste("Media lui X este ", EX)})

output$medieY9 <- renderText({ paste("Media lui Y este ", EY)})

output$dispersieX9 <- renderText({ paste("Dispersia lui X este ", VX)})

output$dispersieY9 <- renderText({ paste("Dispersia lui Y este ", VY)})

output$covarianta9 <- renderText({
    paste("Covarianta este egala cu ", round(cov, digits = 2))
})

output$coef9 <- renderText({
    paste("Coeficientul de corelatie este egal cu ", round(coef, digits = 2))
})
```

Exemplificare din interfață:

Cerința 11

Pentru a introduce datele în interfață, se precizează numărul valorilor introduse, după care vor fi afișate căsuțe de *numericInput* unde vor fi completate valorile. Pentru a le utiliza în calculul proprietăților, se apasă pe butonul “Calculeaza folosind datele”. Altfel, se poate importa un fișier excel, caz în care se utilizează butonul “Calculeaza folosind fisierul”. Fișierul trebuie să conțină o singură coloană de valori, iar prima linie va fi considerată numele coloanei.

```
column(6,fileInput("ex_11_fisier",
                   label="Adaugati fisierului:",
                   multiple = FALSE,accept = ".csv"),
            actionButton("ex_11_btn_fisier", "Calculeaza folosind fisierul")),

column(6,numericInput("ex_11_nrVal","Introduceti numarul valorilor:", 1, min = 1),
            uiOutput("ex_11_table"),
            actionButton("ex_11_btn_val", "Calculeaza folosind datele"))
```

Crearea căsuțelor de input:

```
output$ex_11_table <- renderUI({

    values <- lapply(1:input$ex_11_nrVal, function(val){
      div(style="display: inline-block; width: 70px;",
          numericInput(paste("ex_11_val", val, sep=""),"",NULL) )
    })

    tags$div(
      div(style="display: inline-block; width: 100px;","Valori:"),
      tagList(values)
    )
  })
```

În funcție de butonul apăsat, valorile vor fi preluate astfel:

- folosind id-urile input-urilor în cazul datelor introduse manual;

```
values <- sapply(1:input$ex_11_nrVal, function(ind) {
      input[[paste("ex_11_val", ind, sep="")]]})
```

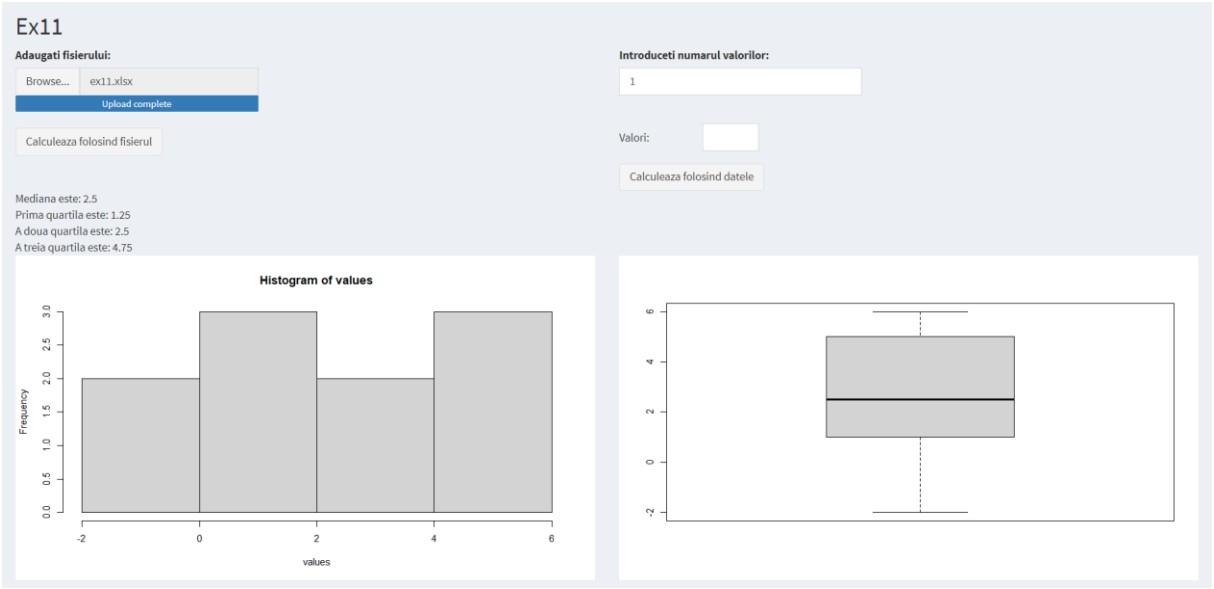
- citind fișierul cu funcția *read_excel* si preluând prima coloană.

```
inFile <- input$ex_11_fisier
dataFile <- read_excel(inFile$datapath,sheet=1)
values <- dataFile[[1]]
```

În ambele cazuri, valorile vor fi prelucrate la fel, utilizând funcțiile *median*, *quantile*, *hist* și *boxplot* și se vor afișa sub forma de *textOutput*-uri.

```
mediana <- median(values)
q1 <- quantile(values, 1/4)
q2 <- quantile(values, 2/4)
q3 <- quantile(values, 3/4)

output$ex_11_med <- renderText({paste("Mediana este:", mediana)})
output$ex_11_q1 <- renderText({paste("Prima quartila este:", q1)})
output$ex_11_q2 <- renderText({paste("A doua quartila este:", q2)})
output$ex_11_q3 <- renderText({paste("A treia quartila este:", q3)})

output$ex_11_hist <- renderPlot({hist(values)})
output$ex_11_box <- renderPlot({boxplot(values)})
```

Exemplificarea din interfață, folosind un fișier pentru a prelua datele:



Cerința 12

Se pot introduce două v.a. discrete asupra cărora se pot realiza 4 operații: sumă, diferență, produs, raport. Pentru fiecare v.a. se introduce numărul valorilor, după care vor fi afișate casete de tip *numericInput,* unde se vor adăuga valorile și probabilitățile. Selectarea operației se face folosind *radioButtons*, iar operația se efectuează după apăsarea unui buton.

Crearea input-urilor pentru v.a. X (se procedeaza analog pentru Y):

```
output$ex_12_tabelX <- renderUI({
    values <- lapply(1:input$ex_12_nrValX, function(val){
      div(style="display: inline-block; width: 70px;",
          numericInput(paste("ex_9_Xvalue", val, sep=""),"",0, min = 0) )
    })

    probs <- lapply(1:input$ex_12_nrValX, function(val){
      div(style="display: inline-block; width: 70px;",
          numericInput(paste("ex_9_Xprob", val, sep=""),"",0, min = 0) )
    })

    tags$div(
      tags$div(
        div(style="display: inline-block; width: 100px;","Valori:"),
        tagList(values) ),
      tags$div(
        div(style="display: inline-block; width: 100px;","Probabilitati:"),
        tagList(probs) ) )
```

```
})
```

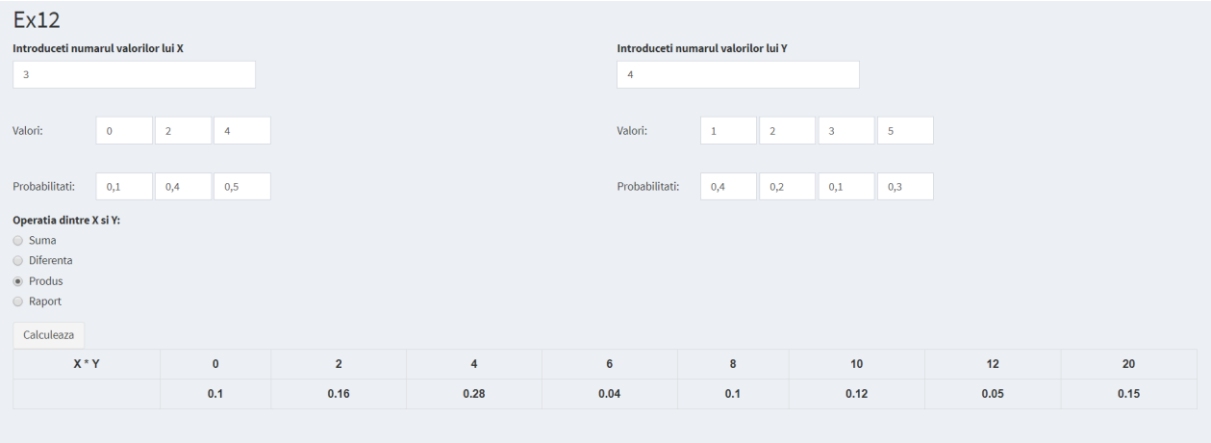
După apăsarea butonului, se extrag valorile și probabilitățile, iar în funcție de opțiunea selectată, se calculează toate outcome-urile rezultatului (pot exista duplicate). Se calculează outcome-urile unice și probabilitățile (ca fiind suma probabilităților pentru fiecare outcome unic).

```
text <- NULL
allOutcomes <- NULL

if(input$ex_12_tip == "suma") {
    text <- "X + Y"
    allOutcomes <- as.vector(sapply(xvalues, function(val){ val + yvalues}))
} else if(input$ex_12_tip == "dif") {
    text <- "X - Y"
    allOutcomes <- as.vector(sapply(xvalues, function(val){ val - yvalues }))
} else if(input$ex_12_tip == "prod") {
    text <- "X * Y"
    allOutcomes <- as.vector(sapply(xvalues, function(val){ val * yvalues }))
} else if(input$ex_12_tip == "rap") {
    text <- "X / Y"
    allOutcomes <- as.vector(sapply(xvalues, function(val){ val / yvalues}))
}

allProbs <- as.vector(sapply(xprobs, function(pr){ pr * yprobs }))
outcomes <- sort(unique(allOutcomes))
probs <- sapply(outcomes, function(out){
    poz <- which(out == allOutcomes)
    sum(allProbs[poz])
})
```

Rezultatul reprezintă o v.a. și va fi afișată sub forma unui tabel:

```
firstRow <- lapply(outcomes, function(out) {tags$th(out)})
secondRow <- lapply(probs, function(prob) {tags$th(prob)})

output$ex_12_rezultat <- renderUI({
    tags$div(
        tags$table(
            tags$tr(tags$th(text),
                    firstRow),
            tags$tr(tags$td(""),
                    secondRow)
        )
    )
})
```

Exemplificare din interfață:



Concluzii

Așadar, aplicația noastră permite atât memorarea variabilelor aleatoare de tip discret și continuu, cât și aplicarea unui set divers de operații asupra acestora precum calculul mediilor și dispersiilor sau a repartiției comune. De asemenea, se pot aplica operații între 2 variabile aleatoare discrete sau aplicarea unor funcții care permit transformarea acestora.

Bibliografie

1. Bernoulli distribution. Wikipedia. [Interactiv] [Citat: 19 Iunie 2022.] https://en.wikipedia.org/wiki/Bernoulli_distribution.

2. Binomial distribution. Wikipedia. [Interactiv] [Citat: 19 Iunie 2022.] https://en.wikipedia.org/wiki/Binomial_distribution.

3. Geometric distribution. Wikipedia. [Interactiv] [Citat: 19 Iunie 2022.] https://en.wikipedia.org/wiki/Geometric_distribution.

4. The Hypergeometric Distribution. R Documentation. [Interactiv] [Citat: 2022 Iunie 19.] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Hypergeometric.html.

5. Poisson Distribution. R Documentation. [Interactiv] [Citat: 19 Iunie 2022.] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Poisson.html.

6. Continuous uniform distribution. Wikipedia. [Interactiv] [Citat: 19 Iunie 2022.] https://en.wikipedia.org/wiki/Continuous_uniform_distribution.

7. Exponential distribution. Wikipedia. [Interactiv] [Citat: 19 Iunie 2022.] https://en.wikipedia.org/wiki/Exponential_distribution#:~:text=In%20probability%20theory%20and%20statistics,case%20of%20the%20gamma%20distribution..

8. Normal distribution. Wikipedia. [Interactiv] [Citat: 19 Iunie 2022.] https://en.wikipedia.org/wiki/Normal_distribution.