Crearea unei aplicații pentru lucrul cu variabile aleatoare folosind Shiny

Proiect Probabilități și Statistică

Alexandrescu Tudor-Alexandru
Brînceanu Raluca-Alexandra
Dumitru Radu-Andrei
Șamata Robert

Cuprins

Introducere

Prin proiectul realizat de noi, dorim sa oferim posibilitatea de a realiza mai multe operații asupra variabilelor aleatoare. În acest sens, pachetul Shiny pentru limbajul R oferă capabilități de realizare a aplicațiilor web, utile pentru analiza, prelucrarea și vizualizarea datelor într-un mod prietenos utilizatorului.

În aplicația noastră am implementat 10 dintre cele 12 exerciții propuse pentru această temă de proiect: toate, cu excepția exercițiilor 2 și 4. Fiecare dintre acestea funcționează separat, în propriul tab al meniului paginii. Pentru o coordonare mai bună în realizarea proiectului, în interiorul proiectului am numerotat exercițiile în ordinea generală în care au fost implementate. În continuare, pentru a evita confuzii în această documentație, vom numerota exercițiile după numărul lor din cerință.

Structură

Proiectul este structurat în următoarele fișiere:

- inputs: conține exemple de date de intrare pentru exercițiile ce necesită citirea datelor din fișiere. *random_data_v1* și *random_data_v2* conțin date în format CSV menite să fie analizate în exercitiul 12, *RV_1* și *RV_2* cuprind câte o variabilă aleatoare discretă pentru calcule generale, în format CSV cu o coloană reprezentând valorile și a doua probabilitățile, iar Large_RV cuprinde o variabila aleatoare de 100 de valori în același format, menită pentru exemplificarea exercițiului 5. În cazul repartițiilor comune, deoarece datele necesare vor fi de obicei de dimensiuni diferite, nu le putem stoca într-un singur fișier CSV: așadar, este necesară în 3 fișiere diferite, unul pentru valorile primei repartiții marginale, unul pentru valorile celei de a doua, și unul pentru probabilități.
- src: cuprinde codul sursă al proiectului, împărțit în:

-probability: conține codul pentru fiecare din distribuțiile utilizate, fiecare având un comportament diferit, împreună cu funcția UsedProbabilityDistributions() ce returnează numele fiecărei distribuții implementate:

```
UsedProbabilityDistributions <- function() {
  c(
    'Normal distribution',
    'Lognormal distribution',
    'Exponential distribution',
    'Gamma distribution',
    'Poisson distribution',
    'Binomial distribution',
    'Uniform distribution',
    'Beta distribution'
  ) %>%
    return()
}
```

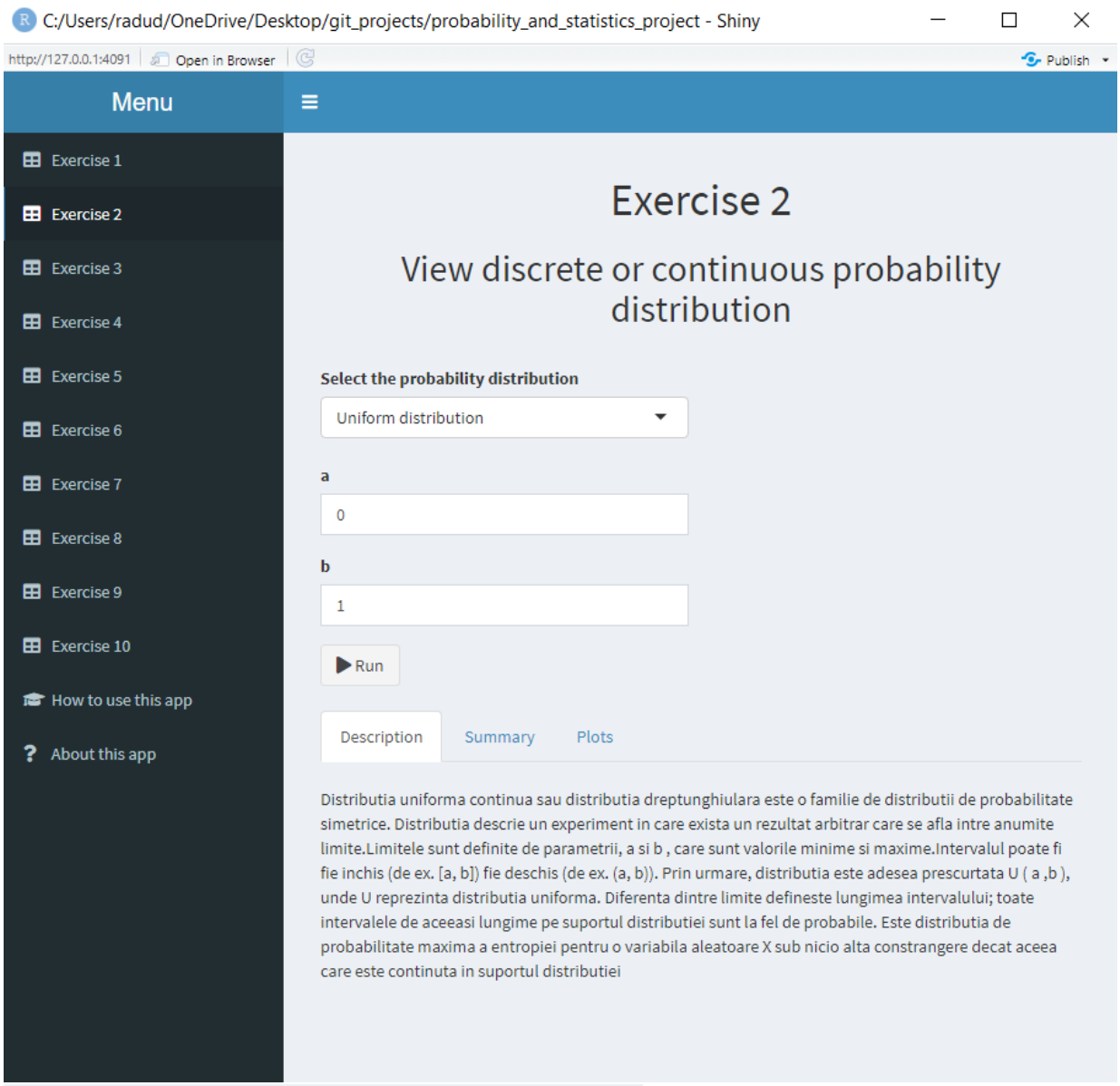
În plus, conține mai multe funcții utilitare: DivideRV() pentru împărțirea a două variabile aleatoare discrete, MultiplyRV() pentru înmulțire, și mxV(), respectiv mnV(), pentru a afla variabila aleatoare maxim, respectiv minim, a unei repartiții comune.
-server: conține codul de server al fiecărui tab, împreună cu un cod server general, ce apelează funcțiile server pentru fiecare tab. La acestea se adaugă o funcție minoră HideGraphicalElements(), cu rolul de a ascunde elementele grafice ale unei distribuții vizualizate până când este necesar.
-ui: conține codul de interfață grafică pentru aplicația generală și fiecare din cele 10 exerciții, împreună cu încă 2 tab-uri: About this app, ce conține o scurtă descriere generală a aplicației și How to use this app, ce conține explicații pentru fiecare exercițiu.
-utils: funcții utilitare pentru crearea de directoare, instalarea pachetelor necesare, și realizarea plot-urilor pentru distribuții, histograme, PDF/PMF și CDF.

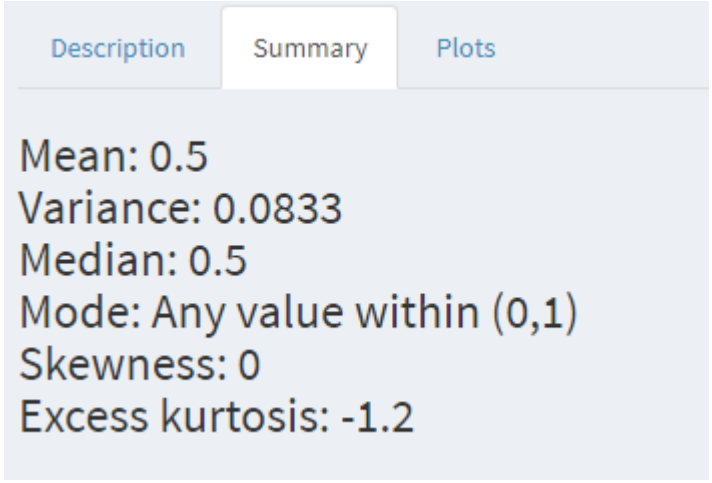
Exercițiul 1
Enunț:
Crearea unui meniu din care poate fi aleasă o repartiție de variabile aleatoare (trebuie să aveți cel puțin 8 repartiții disponibile!), cu particularizarea parametrilor. Utilizatorul va vizualiza o scurtă descriere a respectivei repartiții(în stilul Wikipedia), cu reprezentarea grafică a densității de probabilitate/funcției de masă și a funcției de repartiție, afișarea mediei, dispersiei și a altor elemente ce caracterizează respectiva repartiție.
Rezolvare:
Meniul realizat permite alegerea unui tip de distribuție implementat, ulterior primind prin chenare de numericInput parametrii repartițiilor respective, folosind elemente de tip conditionalPanel pentru a afișa numai chenarele relevante. Fiecare dintre aceste distributii este implementată separat drept o

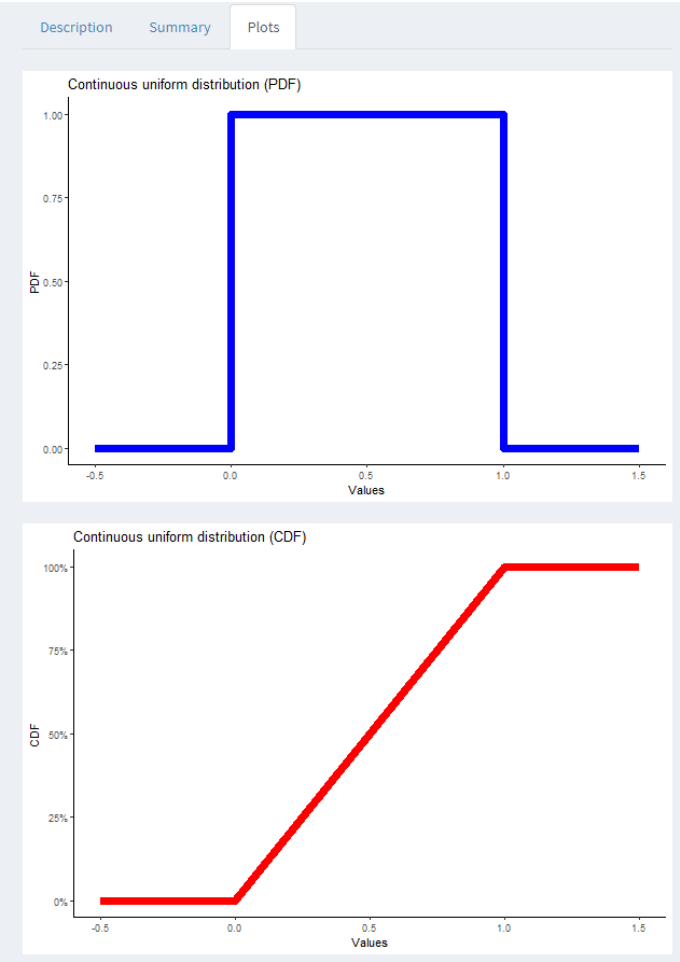
funcție, ce va returna o listă conținând fiecare din parametrii distribuției care pot fi calculați.
De exemplu, am implementat în felul următor repartiția uniformă continuă:

```
ContinuousUniformDistribution <- function(a, b) {
  results <- list()

  results$description <-
    'Distributia uniforma continua sau distributia dreptunghiulara este o familie de distributii de probabilitate simetri
    Distributia descrie un experiment in care exista un rezultat arbitrar care se afla intre anumite limite.Limitele sunt
    Prin urmare, distributia este adesea prescurtata U ( a ,b ), unde U reprezinta distributia uniforma.
  Diferenta dintre limite defineste lungimea intervalului; toate intervalele de aceeasi lungime pe suportul distributiei

  results$type <- 'continuous'

  results$support <- c(min = a, max = b)

  results$mean <- (a + b) / 2

  results$variance <- ((b - a) ** 2) / 12

  results$median <- (a + b) / 2

  results$mode <- paste0('Any value within (', a, ',', b, ')')

  results$skewness <- 0

  results$excess_kurtosis <- -6 / 5

  min.x <-  a
  max.x <-  b
  # for plot scaling
  min.x<-min.x - (b-a)/2
  max.x<-max.x + (b-a)/2

  x <- seq(min.x, max.x, length.out = 10 ** 4)

  pdf <- dunif(x, a, b)
  results$plot_pdf <-
    PlotPdfOrPmf(x, pdf, 'Continuous uniform distribution')

  cdf <- punif(x, a, b)
  results$plot_cdf <-
    PlotCdf(x, cdf, 'Continuous uniform distribution')

  results$pdf <- function(x) {
    dunif(x,a,b) %>%
      return()
  }

  return(results)
}
```

Codul server tratează posibile excepții și erori ce pot apărea (de exemplu, pentru o distribuție normală, parametrul sigma trebuie să fie strict pozitiv). Dacă nu apar erori, se va afișa o descriere a repartiției respective, precum și parametrii returnați prin funcția corespunzătoare și un plot pentru PDF/PMF și CMF, fiecare în taburi separate.
Se vor afișa rezultatele în felul următor:


C:/Users/radud/OneDrive/Desktop/git_projects/probability_and_statistics_project - Shiny
http://127.0.0.1:4091
Open in Browser
Publish
Menu
Exercise 1
Exercise 2
Exercise 3
Exercise 4
Exercise 5
Exercise 6
Exercise 7
Exercise 8
Exercise 9
Exercise 10
How to use this app
About this app
Exercise 2
View discrete or continuous probability distribution
Select the probability distribution
Uniform distribution
a
0
b
1
Run
Description
Summary
Plots
Distributia uniforma continua sau distributia dreptunghiulara este o familie de distributii de probabilitate simetrice. Distributia descrie un experiment in care exista un rezultat arbitrar care se afla intre anumite limite.Limitele sunt definite de parametrii, a si b , care sunt valorile minime si maxime.Intervalul poate fi fie inchis (de ex. [a, b]) fie deschis (de ex. (a, b)). Prin urmare, distributia este adesea prescurtata U (a ,b), unde U reprezinta distributia uniforma. Diferenta dintre limite defineste lungimea intervalului; toate intervalele de aceeasi lungime pe suportul distributiei sunt la fel de probabile. Este distributia de probabilitate maxima a entropiei pentru o variabila aleatoare X sub nicio alta constrangere decat aceea care este continuta in suportul distributiei


Description
Summary
Plots
Mean: 0.5
Variance: 0.0833
Median: 0.5
Mode: Any value within (0,1)
Skewness: 0
Excess kurtosis: -1.2



Exercițiul 3

Enunț:

Se dau două evenimente A și B despre care se precizează dacă sunt independente, incompatibile sau dacă nu se știe nimic despre ele. Pornind de la un set de informații despre niște probabilități legate de ele să se determine toate celelalte probabilități(ex. Știu P(A),P(B), P(A∩B) și determin P(A∪B) folosind formula lui Poincare, P(A|B) și P(B|A) din formula probabilității condiționate)

Rezolvare:

Introducem valorile lui P(A) și P(B), iar apoi alegem interacțiunea dintre acestea. Ulterior, se calculează P(A∪B) folosind formula lui Poincare, P(A|B) și P(B|A) din formula probabilității condiționate. Dacă A și B sunt independente, atunci $P(A \cap B) = P(A) * P(B)$. Dacă sunt incompatibile, atunci $P(A \cap B) = 0$. Rezultatele se vor afișa în felul următor:

Exercise 4

Work with events

P(A)

0.5

P(B)

0.4

Event interaction

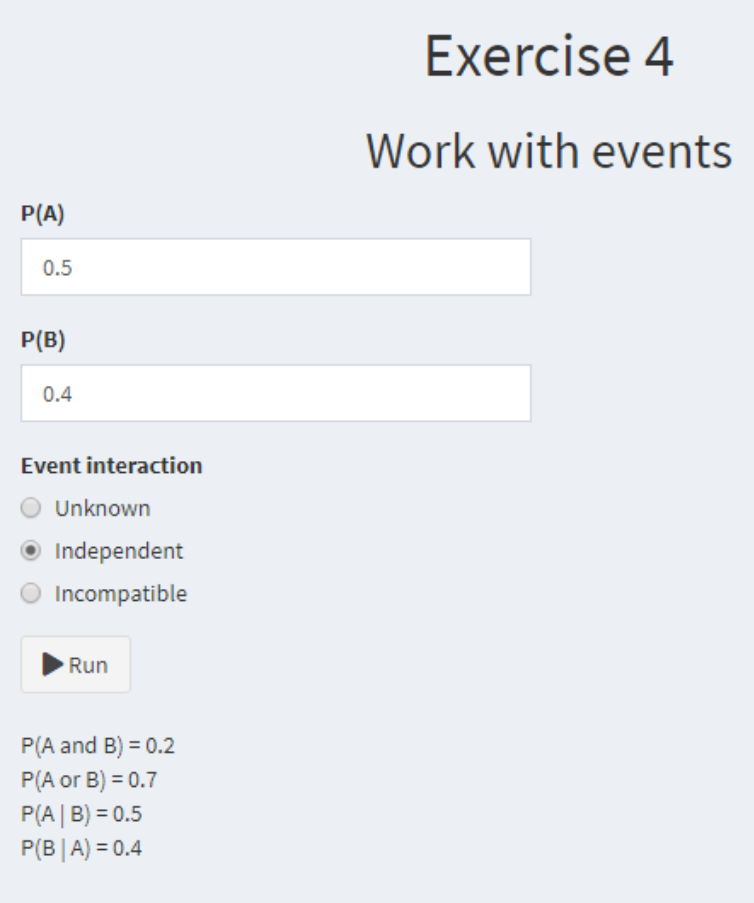
- (•) Unknown
- ( ) Independent
- ( ) Incompatible

P(A and B)

0.3

▶ Run

P(A and B) = 0.3
P(A or B) = 0.6
P(A | B) = 0.75
P(B | A) = 0.6



Exercițiul 5

Enunț:

Afișarea unei v.a. discrete. În cazul în care numărul său de valori este foarte mare să existe posibilitatea de a alege prima valoare care se dorește a fi vizualizată(ex. X ia valori de la 1 la 100, iar eu vreau să vizualizez v.a. începând cu poziția 53).

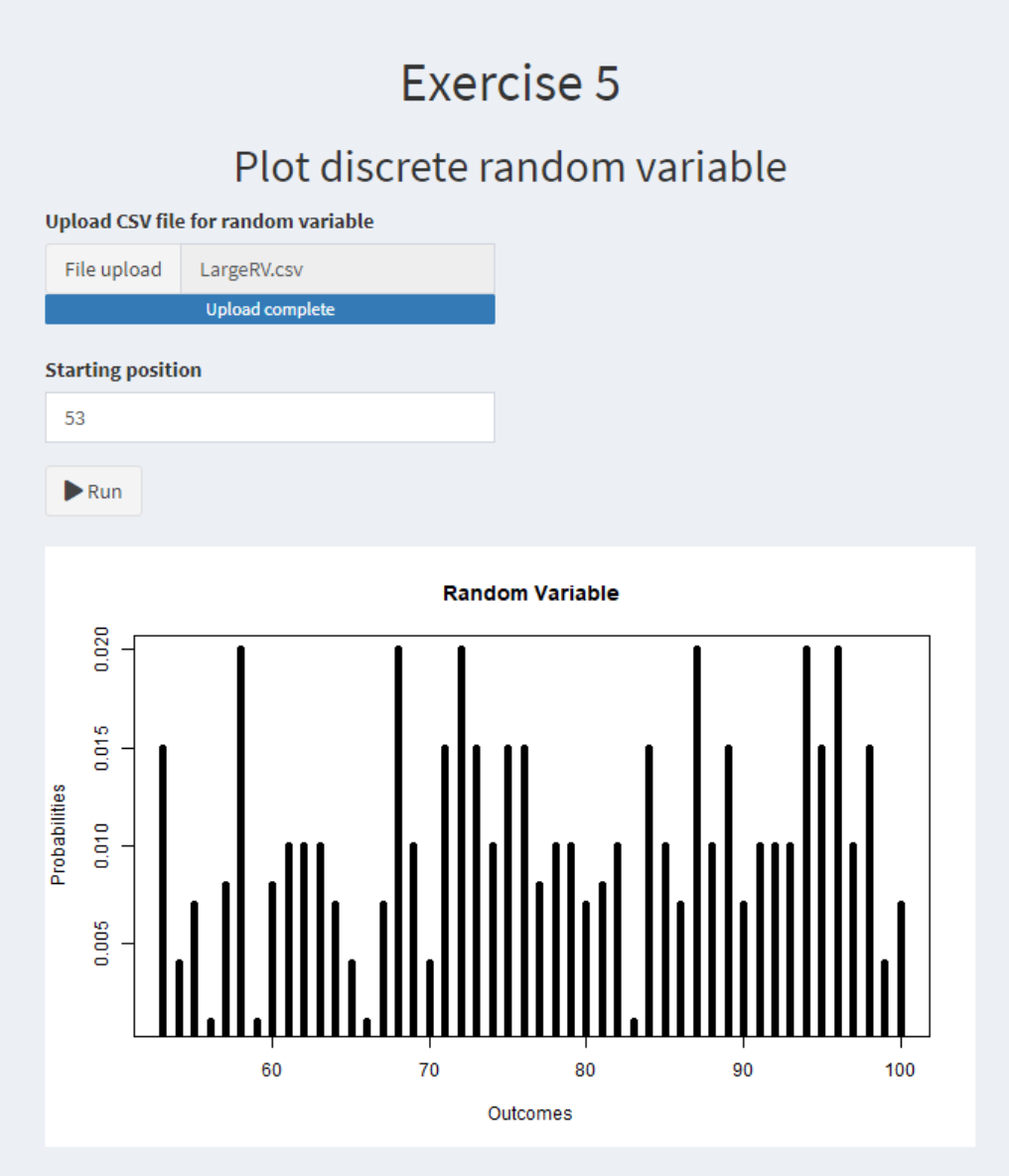
Rezolvare:

Programul va primi ca input un FileInput ce conține datele unei variabile aleatoare discrete, prima coloana conținând valorile și a doua coloană conținând probabilitățile corespunzătoare. La inserarea fișierului, după tratarea excepțiilor, se va afla numărul de valori corespunzătoare variabilei prin construirea unei variabile aleatoare discrete cu datele inserate prin RV(), si apoi aflarea dimensiunii vectorului acesteia de valori prin length(outcomes()).

Pentru a putea fi setată corespunzător poziția de start a vizualizării, va fi actualizat acest numericInput astfel încât valoarea sa maximă să fie lungimea calculată.

Afișarea de la poziția dorită se va realiza construind variabila cu RV() pentru a suma probabilitățile valorilor duplicate în aceeași valoare, iar apoi extrăgând

valorile și probabilitățile variabile începând de la acea poziție în câte un vector, și realizând un plot cu vectorii noi obținuți.
Reprezentarea exemplului dat în cerință prin afișarea unei variabile aleatoare de 100 de valori începând cu poziția 53:

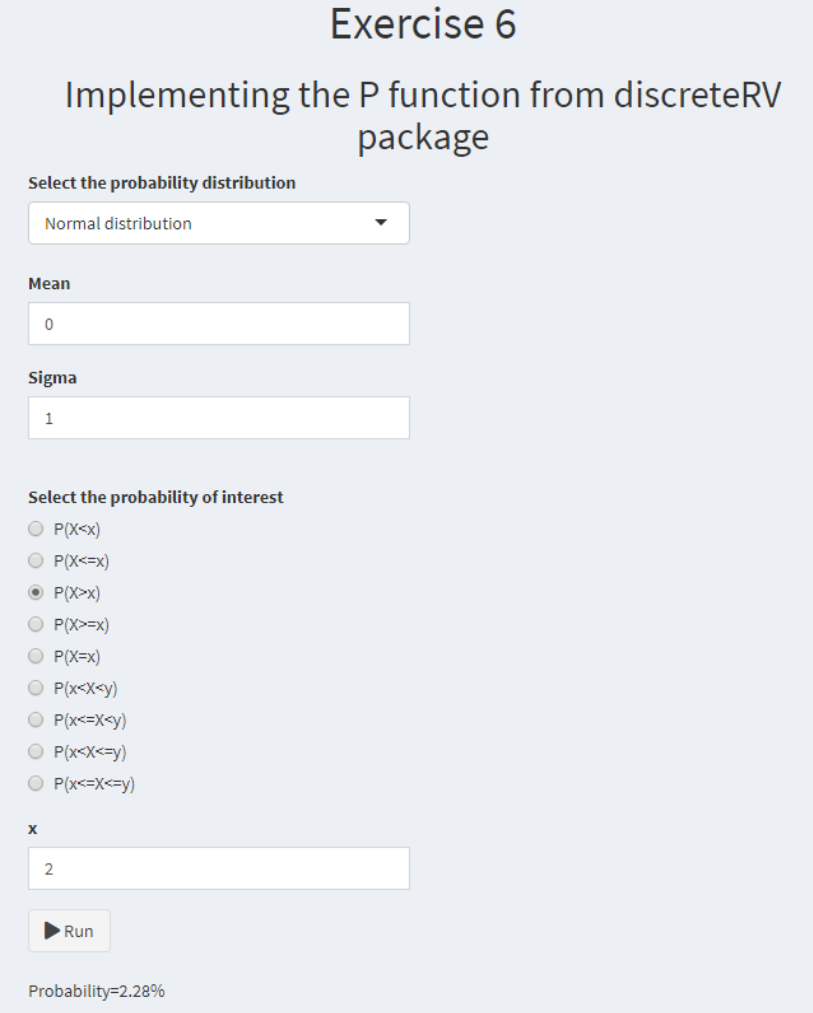


Exercițiul 6
Enunț:
Fiind dată o variabilă aleatoare X cu repartiția dată dintre cele disponibile în Galerie să se permită utilizatorului să se calculeze diferite probabilități(asemănător comportamentului funcției P din pachetul discreteRV), atât pentru variabile discrete cât și pentru cele continue.
Rezolvare:
Introducerea datelor se va realiza similar cu exercițiul 1, prin alegerea unei distribuții și setarea parametrilor în funcție de distribuția aleasă. Ulterior, este

oferita opțiunea alegerii unei anumite probabilități printr-un element radioButtons. Calculul probabilităților cerute se va realiza folosind funcțiile p- din R.
De exemplu, pentru o distribuție normala, se va folosi pnorm:

```
if (input$exercise_6_page_radio_button == 'less_than_x') {
  x <- input$exercise_6_page_less_than_x_value

  probability <-
    pnorm(x, normal.dist.mean, normal.dist.sigma)
} else if (input$exercise_6_page_radio_button == 'less_or_equal_than_x') {
  x <- input$exercise_6_page_less_or_equal_than_x_value

  probability <-
    pnorm(x, normal.dist.mean, normal.dist.sigma)
}
else if (input$exercise_6_page_radio_button == 'greater_than_x') {
  x <- input$exercise_6_page_greater_than_x_value

  probability <-
    1 - pnorm(x, normal.dist.mean, normal.dist.sigma)
} else if (input$exercise_6_page_radio_button == 'greater_or_equal_than_x') {
  x <- input$exercise_6_page_greater_or_equal_than_x_value

  probability <-
    1 - pnorm(x, normal.dist.mean, normal.dist.sigma)
}
else if (input$exercise_6_page_radio_button == 'equal_to_x') {
  # for all continuous distributions
  probability <- 0
} else if (input$exercise_6_page_radio_button == 'greater_than_x_and_less_then_y') {
  x <- input$exercise_6_page_greater_than_x_and_less_then_y_x_value
  y <-
    input$exercise_6_page_greater_than_x_and_less_then_y_y_value

  if (!(x < y)) {
    showNotification("The value of x should be strictly less than the value of y!",
                     type = 'error')
    return(NULL)
  }

  probability <-
    pnorm(y, normal.dist.mean, normal.dist.sigma) - pnorm(x, normal.dist.mean, normal.dist.sigma)
}
```

Rezultatul va fi afișat procentual în felul următor:



Exercițiul 7

Enunț:

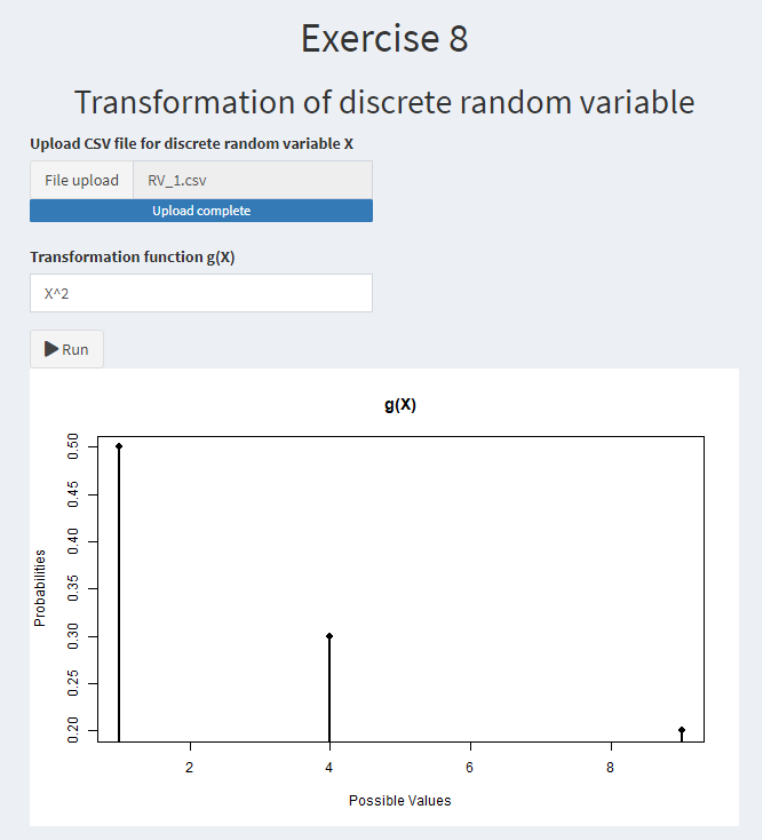
Crearea unei opțiuni într-un meniu care determină o transformare a unei variabile aleatoare g(X), unde X este o variabilă aleatoare discretă, iar g este o funcție furnizată de utilizator.

Rezolvare:

Am realizat introducerea variabilei aleatoare prin citirea din fișier într-un fileInput, împreună cu un textInput pentru introducerea conținutului funcției g(). Odată primite aceste date ca input, funcția și variabila aleatoare vor fi inițializate, iar variabila aleatoare rezultat va putea fi calculată direct în felul următor:

```
input.outcomes <- data[,1]
input.probs <- data[,2]
input.transformation <- data[,3][1]
body(body)
g <- function(x) {}
body(g) <- parse(text = input.transformation)
input.rv <- RV(input.outcomes, input.probs)
result = g(input.rv)
```

Ulterior, rezultatul va fi afișat printr-un plot:



Exercițiul 8

Enunț:

Pornind de la o v.a. continuă X a cărei repartiție e aleasă de utilizator din Galerie și de la o funcție g introdusă de utilizator, calculul și afișarea media și dispersia v.a. g(X).

Rezolvare:

Pentru datele de intrare, se va alege una din repartițiile definite, similar cu exercițiul 1, împreună cu textul ce reprezintă conținutul funcției g, urmând ca această funcție să fie inițializată prin aceeași metodă ca în exercițiul 7. Ulterior, media si dispersia se vor calcula pe baza unei integrale pentru distribuții continue, respectiv sume pentru distribuții discrete, din g(x) * f(x), unde f reprezintă funcția PDF pentru caz continuu, respectiv PMF pentru caz discret. În final, se vor afișa rezultatele prin textOutput:

Exercițiul 9

Enunt:

Crearea unui meniu care să permită utilizatorului să introducă elemente ale repartiției comune a două v.a. discrete, urmând ca restul elementelor să fie calculate și afișate la apăsarea unui buton Completează. Pentru aceste două v.a. să se determine: repartițiile marginale, media, dispersia, covarianța și coeficientul de corelație.

Rezolvare:

Pentru a citi repartiția comună a variabilelor aleatoare X și Y, am implementat un meniu care este format din:

1) introducerea variabilei aleatoare X
2) introducerea variabilei aleatoare Y
3) introducerea probabilităților pentru X,Y (in repartiția comuna data ca exemplu din folder-ul inputs, avem 3 valori pentru X si 2 valori pentru Y deci vor fi 6 probabilități in total)

Ulterior, am creat o funcție pentru butonul de run, a cărui apăsare ia toate valorile input-urilor si calculează si afișează output-ul.(în cazul nostru : repartițiile marginale, media, dispersia, covarianța și coeficientul de corelație.)

Implementarea codului in R:

```
Exercise3 <- function(input, output) {
  GetInputData <- reactive({
    req(input$exercise_3_page_upload_values_x)
    file.x <- input$exercise_3_page_upload_values_x
    if (!all(file_ext(file.x$name) == 'csv')) {
      showNotification("X values: You must choose a csv file!", type = 'error')
      return(NULL)
    }

    x.csv.data <- read.csv(file.x$datapath)
    if (!ncol(x.csv.data) == 1) {
      showNotification("X values: file must contain only 1 column, representing values of X!", type = 'error')
      return(NULL)
    }

    req(input$exercise_3_page_upload_values_y)
    file.y <- input$exercise_3_page_upload_values_y
    if (!all(file_ext(file.y$name) == 'csv')) {
      showNotification("Y values: You must choose a csv file!", type = 'error')
      return(NULL)
    }

    y.csv.data <- read.csv(file.y$datapath)
    if (!ncol(y.csv.data) == 1) {
      showNotification("Y values: file must contain only 1 column, representing values of Y!", type = 'error')
      return(NULL)
    }

    req(input$exercise_3_page_upload_probs)
    file.probs <- input$exercise_3_page_upload_probs
    if (!all(file_ext(file.probs$name) == 'csv')) {
      showNotification("Probabilities: You must choose a csv file!", type = 'error')
      return(NULL)
    }

    probs.csv.data <- read.csv(file.probs$datapath)
    if (!ncol(probs.csv.data) == 1) {
      showNotification("Probabilities: file must contain only 1 column, representing probabilities!", type = 'error')
      return(NULL)
    }

    if (!sum(probs.csv.data[,1]) == 1) {
      showNotification("Sum of probabilities must equal 1!", type = 'error')
      return(NULL)
    }
    return(data.frame(x.csv.data[,1], y.csv.data[,1], probs.csv.data[,1]))
  })
  observeEvent(GetInputData(), {
    show('exercise_3_page_run_button')})
  observeEvent(input$exercise_3_page_run_button, {
    data <- GetInputData()
    input.outcomes.x <- unique(data[,1])
    ## unique for removing elements recycled by data.frame() to match dimension of probs vector
    input.outcomes.y <- unique(data[,2])
    input.probs <- data[,3]
```

Următorul pas a fost calculul propriu-zis al repartițiilor marginale, media, dispersia, covarianța și coeficientul de corelație.

După inițializarea repartiției comune prin funcția utilitară jointRVFromInput, definită în folder-ul probability, am utilizat mai multe funcții din pachetul discreteRV pentru calcul(marginal, E, V), si o funcție de înmulțire a doua variabile aleatoare, multiplyRV.

Afișarea am făcut-o printr-un plot pentru repartițiile marginale, iar pentru medie, dispersie, covariație și coeficientul de corelație am afișat direct rezultatul ca textOutput.

```
joint.input <- jointRVFromInput(input.outcomes.x, input.outcomes.y, input.probs)
marginal(joint.input, 1)
marginal(joint.input, 2)
x <- marginal(joint.input,1)
y <- marginal(joint.input,2)
MedieX <- E(x)
MedieY <- E(y)
VarX <- V(x)
VarY <- V(y)
DispX <- sqrt(VarX)
DispY <- sqrt(VarY)
Cov <- E(MultiplyRV((x-E(x)),(y-E(y))))
Cor <- Cov/(DispX * DispY)
output$exercise_3_page_output_x_average <- renderText({paste("Medie X: ", MedieX)})
output$exercise_3_page_output_y_average <- renderText({paste("Medie Y: ", MedieY)})
output$exercise_3_page_output_x_dispersion <- renderText({paste("Dispersie X: ", DispX)})
output$exercise_3_page_output_y_dispersion <- renderText({paste("Dispersie Y: ", DispY)})
output$exercise_3_page_output_x_y_covariance <- renderText({paste("Covarianta: ", Cov)})
output$exercise_3_page_output_x_y_correlation_coefficient <- renderText({paste("Coeficient de relatie: ", Cor)})
show('exercise_3_page_output_x_average')
show('exercise_3_page_output_y_average')
show('exercise_3_page_output_x_dispersion')
show('exercise_3_page_output_y_dispersion')
show('exercise_3_page_output_x_y_covariance')
show('exercise_3_page_output_x_y_correlation_coefficient')

output$exercise_3_page_output_x <- renderPlot({plot(x, main="X")})
output$exercise_3_page_output_y <- renderPlot({plot(y, main="Y")})
show('exercise_3_page_output_x')
show('exercise_3_page_output_y')
```

Exercițiul 10

Enunt:

Crearea unui meniu care să permită utilizatorului ca, pornind de la repartiția comună a două v.a. discrete X și Y(furnizată de utilizator) să fie construită și afișată repartiția comună a alte două v.a. Z și T, unde Z=g(X,Y) iar T=h(X,Y), iar g și h sunt furnizate de utilizator. (ex. Z=max(X,Y), iar T=min(X,Y)).

Rezolvare:

Se va primi ca input o repartiție comună prin 3 elemente fileInput separate, urmând ca aceasta să fie inițializată prin funcția jointRV() din discreteRV, oferind datele fiecărui fișier citit ca parametri. Programul primește ulterior ca input și conținutul funcțiilor g(), respectiv h(), ce vor fi aplicate asupra repartițiilor marginale, obținute folosind funcția marginal() din același pachet. În final, se va crea și afișa repartiția comună cerută folosind metoda joint() a pachetului, ce va realiza repartiția comună a celor două rezultate anterioare.

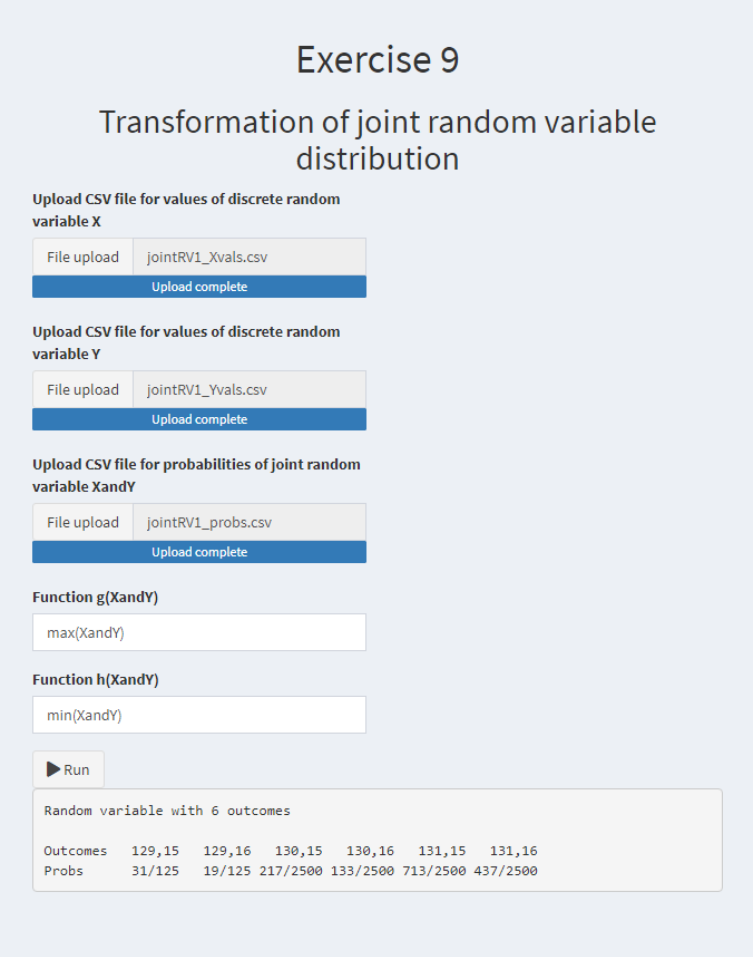
În R, funcțiile max() și min() funcționează diferit față de exemplul oferit, nereturnând o variabilă aleatoare, ci un set de valori. Ca atare, am realizat propriile implementări ale acestor funcții, mxV și mnV, pentru a afla variabila aleatoare maxim respectiv minim a unei repartiții comune:

```
mxV <- function(joint.rv) {

  max.pair <- function(a, b) {
    return(max(c(a, b)))
  }

  marg.1 <- marginal(joint.rv, 1)
  marg.2 <- marginal(joint.rv, 2)

  outcomes.1 <- outcomes(marg.1)
  outcomes.2 <- outcomes(marg.2)

  new.outcomes <- c(outer(outcomes.1, outcomes.2, Vectorize(max.pair)))
  new.probs <- probs(joint.rv)

  return(RV(new.outcomes, new.probs))
}
```

Funcția mxV va calcula valorile variabilei noi aplicând funcția max.pair pe fiecare element al produsului cartezian outcomes.1 X outcomes.2. Acest lucru poate fi realizat prin functia outer(), ce va genera o matrice aplicând, în acest caz vectorizat, funcția de maxim dintre două numere pe fiecare pereche necesară. Matricea va fi transformată în vector prin c(), urmând a fi returnată o variabilă aleatoare nouă cu aceste valori și probabilitățile repartiției comune originale. Funcția RV() va suma automat probabilitățile cu această valoare, așadar aceasta nu va fi o problemă. Funcția mnV() funcționează similar, dar bazat pe minim în loc de maxim.

Aceste funcții fiind astfel definite, înainte de a fi inițializate funcțiile g() și h(), se va înlocui în string orice apel al funcțiilor max sau min cu aceste noi funcții, utilizând funcția str_replace_all() din pachetul stringr.

Output-ul va arăta în felul următor:



Exercițiul 11

Enunt:

Crearea unui meniu care să permită utilizatorului introducerea unor valori numerice, sau importarea unui fișier care să conțină aceste valori. Pentru acest set de date să se determine mediana, cuartilele și să se afișeze histograma și diagrama boxplot.

Rezolvare:

Pentru acest exercițiu am realizat citirea prin fișier prin elemente fileInput, primind un fișier CSV conținând date numerice. După citirea fișierului și salvarea componentelor acestuia ca dataframe, se vor extrage numele coloanelor acestuia folosind funcția colnames(), ulterior actualizând un element selectInput pentru a se putea alege dintre numele coloanelor citite. Odată apăsat butonul Run, se vor afișa fiecare din elementele cerute:

- Pentru mediană și cuartile, am utilizat funcția summary() și am afișat rezultatul printării acesteia folosind renderPrint() asupra unui element verbatimTextOutput
- Pentru histogramă, am utilizat funcția plotHistogram() pentru a realiza plotarea după specificațiile următoare, folosind pachetul ggplot:

```r
PlotHistogram <- function(data,
                          var.name,
                          bins = 30,
                          x.label = NULL,
                          y.label = NULL,
                          plot.title = NULL) {
  plot.object <- ggplot(data, aes_string(x = var.name)) +
    geom_histogram(color = 'darkblue',
                   fill = 'lightblue',
                   bins = bins) +
    theme_classic()

  if (is.null(x.label)) {
    x.label <- var.name
  }

  if (is.null(y.label)) {
    y.label <- 'Count'
  }

  if (is.null(plot.title)) {
    plot.title <- paste0('Histogram of ', var.name)
  }

  plot.object <- plot.object +
    labs(x = x.label, y = y.label, title = plot.title) +
    theme(plot.title = element_text(size = 20))

  return(plot.object)
}
```

- Pentru boxplot, am realizat plotarea în mod similar, prin functia boxPlot():

```r
BoxPlot <- function(data,
                    y.var.name,
                    y.label = NULL,
                    plot.title = NULL) {
  plot.object <- ggplot(data, aes_string(y = y.var.name)) +
    geom_boxplot(
      fill = 'lightblue',
      outlier.colour = "red",
      outlier.shape = 8,
      outlier.size = 4
    ) +
    scale_x_discrete() +
    theme_classic()

  if (is.null(plot.title)) {
    plot.title <- paste0('Boxplot of ', y.var.name)
  }

  if (is.null(y.label)) {
    y.label = y.var.name
  }

  plot.object <- plot.object +
    labs(y = y.label, title = plot.title) +
    theme(plot.title = element_text(size = 20))

  return(plot.object)
}
```

Ulterior, am afișat aceste plot-uri prin elemente plotOutput:

Exercise 1

Analysis of numerical data

Select the input csv file

File upload | random_data_v1.csv

Upload complete

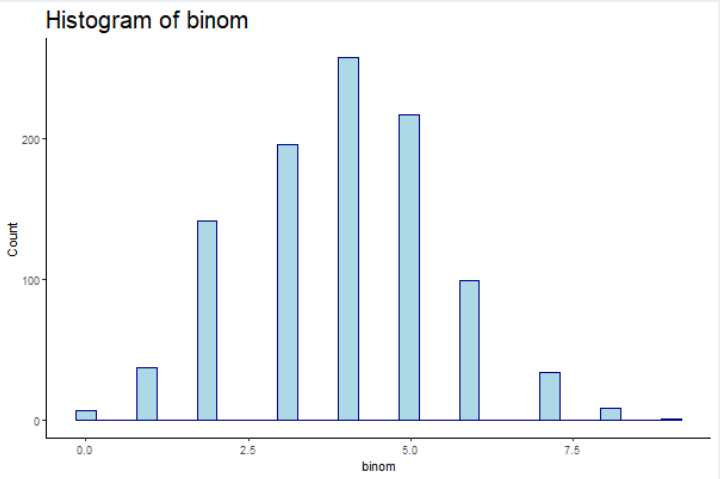
Select the column name

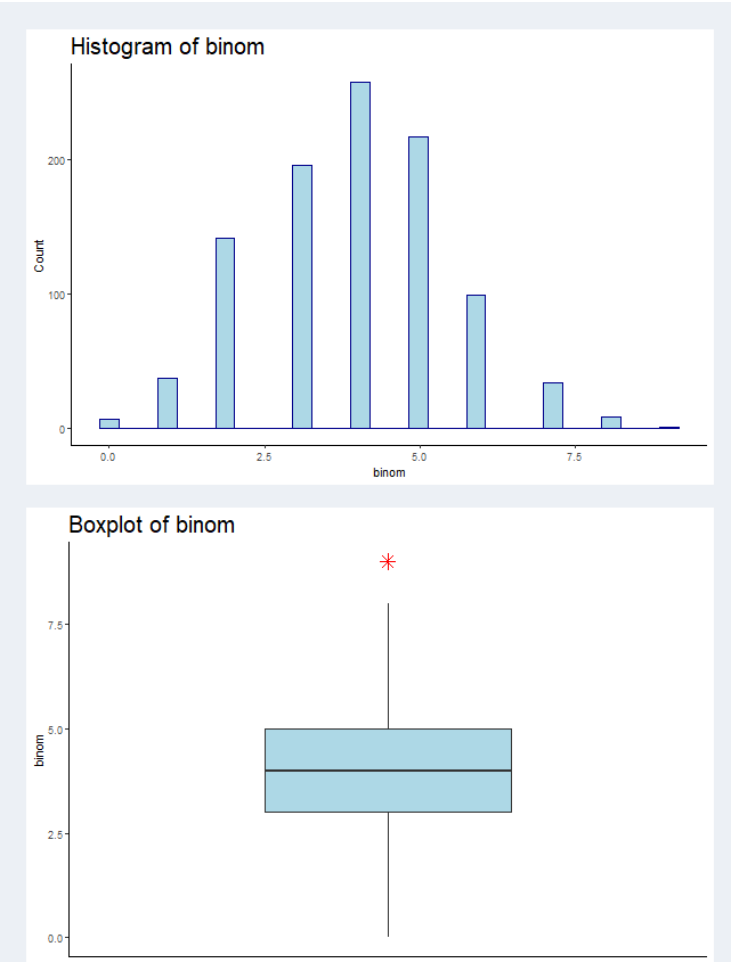
binom

Run

Statistics summary:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   3.000   4.000   3.939   5.000   9.000
```





Exercițiul 12

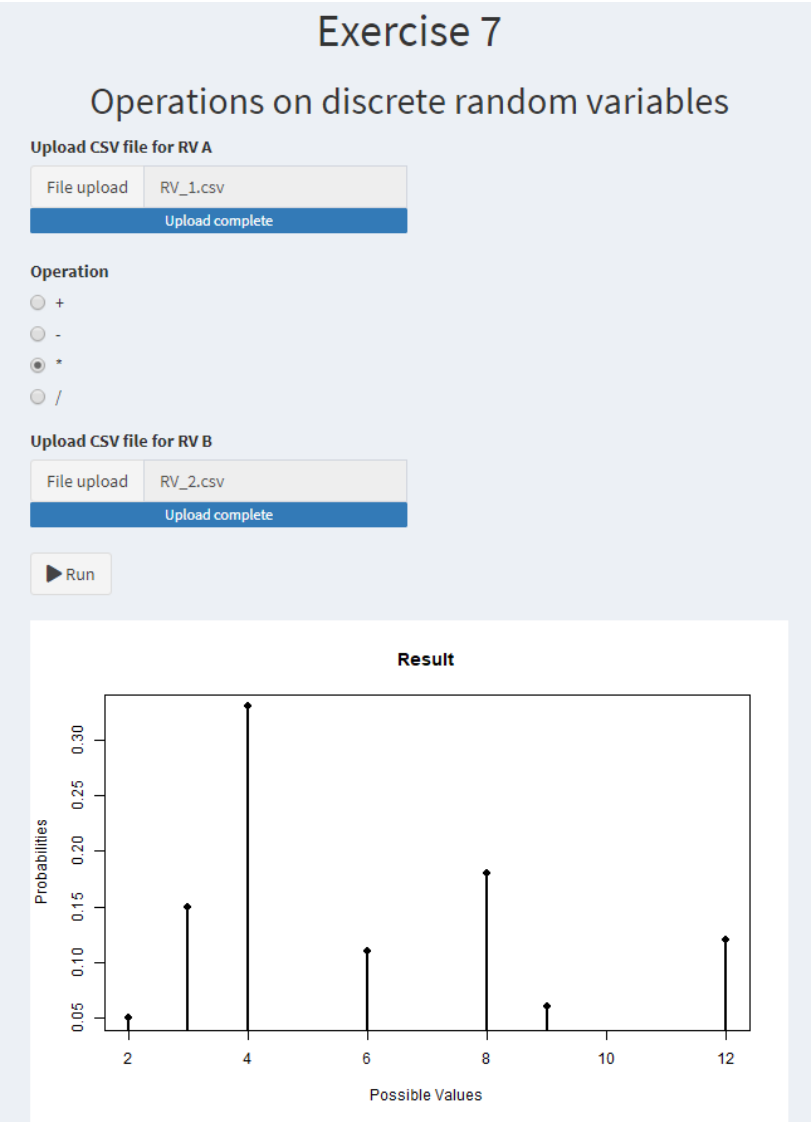
Enunt:

Crearea unei opțiuni în meniu care să permită operații cu v.a. discrete independente(sumă, diferență, produs, raport, etc.).

Rezolvare:

Pentru input, am folosit elemente fileInput pentru citirea din fișier a două variabile aleatoare discrete, împreună cu un element RadioButtons pentru alegerea din interfața grafică a operației efectuate: adunare, scădere, înmulțire sau împărțire.

La apăsarea butonului run, se vor crea prin RV() două variabile aleatoare, urmând a fi calculat rezultatul în funcție de operația selectată: pentru adunare și scădere a funcționat utilizarea operatorilor + si – asupra variabilelor, dar pentru înmulțire și împărțire au trebuit definite propriile de înmulțire și împărțire a variabilelor aleatoare discrete, MultiplyRV() respectiv DivideRV(). Ulterior, rezultatul va fi pus într-un plot și afișat prin PlotOutput.



Probleme întâmpinate

De-a lungul realizării acestui proiect, am întampinat mai multe dificultăți, rezolvate în felul următor:

- Lipsa suportului pachetului discreteRV pentru înmulțirea a 2 variabile aleatoare discrete: dată fiind această problemă, a fost nevoie să implementăm propriul algoritm de înmulțire a două variabile aleatoare, inspirat de pe Stack Overflow.

```r
MultiplyRV <- function(A, B) {
  # https://stackoverflow.com/questions/61479740/product-of-two-random-variables-in-r-using-the-discreterv-package
  product.matrix <- t(outer(outcomes(A),outcomes(B),"*")) ## find all possible products
  probability.matrix <- t(outer(probs(A), probs(B)))
  unique.products <- unique(as.vector(product.matrix))  ## find the unique products
  probability.vector <- rep(0, length(unique.products))

  for(i in 1:length(probability.vector)){

    z <- unique.products[i]

    indices <- which(as.vector(product.matrix) == z) ## find which elements of product.matrix match up to z

    probability.vector[i] <- sum(as.vector(probability.matrix)[indices]) ## sum their probabilities

  }

  XtimesY <- RV(outcomes = unique.products, probs = probability.vector) ## store as RV
}
```

- Eroare în împărțirea a două variabile aleatoare discrete prin operatorul /: variabila aleatoare rezultată conținea uneori aceeași probabilitate repetată, cu probabilități diferite. Astfel, a fost necesară reinițializarea conținutului acesteia în altă variabilă, pentru a elimina duplicatele și suma probabilitățile respective:

```r
DivideRV <- function(A, B) {
  division.result <- A / B
  result <- RV(outcomes(division.result), probs(division.result))
  return(result)
}
```

Concluzie

Având în vedere efortul depus în realizarea acestei aplicații, putem spune că prin rezolvarea acestui proiect, am învățat sa gestionăm în diferite feluri lucrul cu variabile aleatoare, sa creăm meniuri în acest sens în Shiny, și să construim un proiect complet R, cu documentație.

Bibliografie:

https://stackoverflow.com/questions/61479740/product-of-two-random-variables-in-r-using-the-discreterv-package

https://cran.r-project.org/web/packages/discreteRV/vignettes/discreteRV.html

Pachete folosite:

```r
packages <- c('ggplot2',
              'shiny',
              'purrr',
              'shinydashboard',
              'magrittr',
              'tools',
              'shinyjs',
              'reshape2',
              'dplyr',
              'discreteRV',
              'stringr')
```