Probabilitati si Statistica
Proiect 1

Cibotari Augustin
Licu Mihai
Matache Alexandru
Moraru Radu

Iunie 2022

Contents

Proiectul 1

Folosind documentul suport și orice alte surse de documentare considerați potrivite construiți un pachet R care să permită lucru cu variabile aleatoare continue. Pentru a primi punctaj maxim, pachetul trebuie să implementeze cel puțin 8 din următoarele cerințe.

Chapter 1

Problema 1

1.1 Cerinta

Fiind data o functie f , introdusa de utilizator, determinarea unei constante de normalizare k. In cazul in care o asemenea constanta nu exista, afisarea unui mesaj corespunzator catre utilizator.

1.2 Rezolvare

Pentru calcularea constanteri de normalizare k, in raport cu o functie introdusa de catre utilizator, se aplica formula:

$$k = \frac{1}{(\int_{-\infty}^{\infty} f(x)\,dx)}$$

1.3 Cod

Returns the normalizing constant k for a function if it exists, otherwise returns null.

@name find_normalizing_constant

@param func Has to be a function for which we want to find the normalizing constant

@return Value of integral (normalizing constant k)

```
find_normalizing_constant <- function(func) {
    tryCatch(
        {
            if (!is.function(func)) stop("Parameter func has to be a function.")
            # formula for normalizing constant
            integral <- integrate(func, lower = -Inf, upper = Inf)$value
            return(1 / integral)
        },
        error = function(e) {
            warning("Normalizing constant not found")
            warning(e$message)
            return(NULL)
        }
    )
}
```

1.4 Exemple

@examples

1 / sqrt(2 * pi)

f <- function(x) {

exp((-x^2)/2)

}

find$_{n}ormalizing_{c}onstant(f)$

Chapter 2

Problema 2

2.1 Cerinta

Verificarea daca o functie introdusa de utilizator este densitate de probabilitate.

2.2 Rezolvare

Pentru a verifica daca functia f este densitate de probabilitate, am testat urmatoarele 2 proprietati:

$$1. f(x) \geq 0 \ \ \forall \ \ x \ \ din \ \ suport$$

$$2. \int_{-\infty}^{\infty} f(x)\, dx = 1$$

2.3 Cod

Returns true if the provided function is a probability density function and false otherwise

@name is$_{p}df$

@param func Is the function that we want to analyse

@return A boolean value that represents wether the provided function is a probability density function

@examples

f <- function(x) ifelse(x >= -1 x <= 1, 1 - abs(x), 0)

is$_{p}df(f)$

```
is_pdf <- function(func) {
    tryCatch(
        {
            if (!is.function(func)) stop("Parameter func has to be a function.")
            # generate a faux (-Inf, Inf) interval
            xs <- seq(-1e+6, 1e+6, len = 10000)

            # function must be positive
            if (all(func(xs) >= 0)) {

                # integral must be equal to 1 (with 1^-10 tolerance)
                integral <- integrate(func, lower = -Inf, upper = Inf)
                if (abs(integral$value - 1) < 1e-10) {
                    return(TRUE)
                }
            }
            return(FALSE)
        },
        error = function(e) {
            warning(e$message)
            return(FALSE)
        }
    )
}
```

2.4 Exemple

@examples

f $\leftarrow function(x)ifelse(x \Rightarrow -1\&x \Leftarrow 1, 1 - abs(x), 0)$

is_pdf(f)

Chapter 3

Problema 3

3.1 Cerinta

Crearea unui obiect de tip variabila aleatoare continua pornind de la o densitate de probabilitate introdusa de utilizator. Functia trebuie sa aiba optiunea pentru variabile aleatoare unidimensionale si respectiv bidimensionale.

3.2 Rezolvare

- Inspirat de la pachetul discreteRV.

- Obiectul de tip cRV este creat prin apelarea functiei cRV care ia ca parametru o functie vectorizabila

- functia data prin parametrul pdf este verificata daca este o functie de densitate de probabilitate valida

- este creata si functia de repartitie de probabilitate (cdf) pornind de la pdf-ul dat

- obiectul va avea doua atribute: pdf-ul si cdf-ul

- pot fi create v.a. unidimensionale sau bidimensionale in functie de numarul de argumente ale functiei

date

3.3 Cod

Check if a function can be the pdf of a random variable with a joint distribution

@name is_joint_pdf

@param func Function to be tested

@return A boolean value representing wether func is the pdf of a bidimensional random variable

```
is_joint_pdf <- function(func) {
  if (!is.function(func)) stop("Parameter func has to be a function.")
  if (length(formals(func)) != 2) return(FALSE)

  xs <- rep(seq(-1e+6, 1e+6, len = 5000), each=5000)
  ys <- rep(seq(-1e+6, 1e+6, len = 5000), 5000)

  if (all(func(xs, ys) >= 0)) {
    integ <- integral(func, bounds = list(x=c(-Inf,Inf), y=c(-Inf,Inf)), vectorize = TRUE)
    if (abs(integ$value - 1) <= integ$error) return(TRUE)
  }

  return(FALSE)
}


get_cdf_from_pdf <- function(pdf) {
  if (length(formals(pdf)) == 1) {
    return(function(x) {
      integrate(pdf, lower=-Inf, upper=x)$value
    })
  }
  else if (length(formals(pdf)) == 2) {
    return(function(x, y) {
      integral(pdf, bounds = list(x=c(-Inf, x), y=c(-Inf, y)))$value
    })
  }
}
```

Create a continuous random variable starting from a probability density function

@name cRV

@param pdf Probability density function. Can either be a univariate or a bivariate pdf. Must allow vectorization.

@return Continuous random variable

```
cRV <- function(pdf) {
  if (!is.function(pdf)) stop("PDF must be a function.")

  # Check if function takes one or two parameters
  if (length(formals(pdf)) == 1) {
    if (!is_pdf(pdf)) stop("Provided function is not a valid PDF.")
  }
  else if (length(formals(pdf)) == 2) {
    if (!is_joint_pdf(pdf)) stop("Provided function is not a valid PDF.")
  }
  else stop("Provide a function that takes either one or two variables.")

  class(pdf) <- "cRV"
  attr(pdf, "pdf") <- pdf
  attr(pdf, "cdf") <- get_cdf_from_pdf(pdf)

  return(pdf)
}
```

3.4 Exemple

Define a valid pdf

pdf1 $\leftarrow function(x)(0 \Leftarrow x\&x \Leftarrow 1) * (3x^2)$

Create a unidimensional continuous random variable with pdf1

X $\leftarrow cRV(pdf1)$

Define a valid pdf with two parameters

pdf2 $\leftarrow function(x,y)(0 \Leftarrow x\&x \Leftarrow 1\&0 \Leftarrow y\&y \Leftarrow 1) * (2(1-x))$

Create a bidimensional continuous random variable with pdf2

Y $\leftarrow cRV(pdf2)$

Chapter 4

Problema 4

4.1 Cerinta

Reprezentarea grafica a densitatii si a functiei de repartitie pentru diferite valori ale parametrilor repartitiei. In cazul in care functia de repartitie nu este data intr-o forma explicita(ex. repartitia normala) se accepta reprezentarea grafica a unei aproximari a acesteia.

4.2 Rezolvare

- Am creat o singura functie care poate afisa grafice atat pentru un pdf cat si pentru un cdf

- Functia poate genera grafice pentru v.a. unidimensionale / bidimensionale

- Functia poate lua un obiect de tip function sau un set de valori rezultat din utilizarea distributiilor standard din R

- Daca este specificat domeniul pentru y, v.a. data este considerata bidimensionala

4.3 Cod

Plot pdf or cdf of a unidimensional or a bidimensional random variable.

@name plot$_fun$

@param fun Can be a function taking one argument (x) or two arguments (x, y).

Can be a vector of y values for unidimensional distributions, or a vector/matrix of z values for bidimensional distributions.

@param xDomain Domain of x values over which pdf/cdf is to be evaluated

@param yDomain Domain of y values over which pdf/cdf is to be evaluated.

If empty/not provided, 'plot$_fun()'assumesaunidimensionaldistribution.$

```
plot_fun <- function(fun, xDomain, yDomain = c()) {
  if (length(yDomain) == 0) {
    if (is.function(fun))
      fun <- sapply(xDomain, fun)

    plot(xDomain, fun, type="l", col="red", lwd = 3)
  }
  else {
    if (is.function(fun)) {
      xT = rep(xDomain, each=length(yDomain))
      yT = rep(yDomain, length(xDomain))

      fun <- mapply(fun, xT, yT)
    }

    if (!is.matrix(fun))
      fun <- matrix(fun, ncol = length(xDomain), byrow=TRUE)

    fig <- plot_ly(x = xDomain, y = yDomain, z = fun)
    fig <- fig %>% add_surface()

    fig
  }
}
```

4.4 Exemple

UNIDIMENSIONAL RV EXAMPLES

Using pdf provided by the user
pdf1 $\leftarrow function(x)(0 <= x\&x <= 1) * (3x^2)$
plot$_{f}un(pdf1, seq(-1, 2, 0.01))$
Using cdf provided by the user
cdf1 $\leftarrow function(x)integral(pdf1, bound = list(x = c(-Inf, x)))\$value$
plot$_{f}un(cdf1, seq(-1, 2, 0.01))$
Using normal distribution
xDomain $\leftarrow seq(-10, 10, 0.1)$
plot$_{f}un(dnorm(xDomain, mean = 0, sd = 1), xDomain)$
plot$_{f}un(pnorm(xDomain, mean = 0, sd = 1), xDomain)$

BIDIMENSIONAL RV EXAMPLES

Using pdf provided by the user
pdf2 <- function(x, y) $(0 \Leftarrow x\&x \Leftarrow 1\&0 \Leftarrow y\&y \Leftarrow 1) * (2(1 - x))$
yDomain $\leftarrow seq(-1, 1, 0.1)$
xDomain $\leftarrow seq(-1, 1, 0.1)$
plot$_{f}un(pdf2, xDomain, yDomain)$
Using cdf provided by the user
cdf2 $\leftarrow function(x, y)integral(pdf2, bounds = list(x = c(-Inf, x), y = c(-Inf, y)))\$value$
plot$_{f}un(cdf2, xDomain, yDomain)$
Using normal distribution
domain <- seq(-5, 5, 0.1)
value$_{p}airs < -expand.grid(x = domain, y = domain)$
library(mvtnorm)
plot$_{f}un(dmvnorm(x = value_{p}airs), xDomain = domain, yDomain = domain)$

Chapter 5

Problema 5

5.1 Cerinta

Calculul mediei, dispersiei si a momentelor initiale si centrate pana la ordinul 4(daca exista). Atunci cand unul dintre momente nu exista, se va afisa un mesaj corespunzator catre utilizator.

5.2 Rezolvare

Pentru a calcula toate cele patru elemente, am folosit urmatoarele formule:

$$Media = \int_{-\infty}^{\infty} x * f(x)\, dx$$

$$Dispersia = \int_{-\infty}^{\infty} (x - medie)^2 * f(x)\, dx$$

$$Momentul\ \ initial\ \ de\ \ ordin\ \ i = \int_{-\infty}^{\infty} x^i * f(x)\, dx$$

$$Momentul\ \ centrat\ \ de\ \ ordin\ \ i = \int_{-\infty}^{\infty} (x - medie)^i * f(x)\, dx$$

5.3 Cod

Returns the expectation (expected value, first moment, mean, average) for an object of type cRV

@name expectation

@param cRV Is the continuous random variable for which we want to find the expectation.

@return The value of the integral to determine the expectation

```
expectation <- function(cRV) {
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    tryCatch(
        {
            func <- attr(cRV, "pdf")
            new_func <- function(x) {
                x * func(x)
            }
            return(integrate(new_func, lower = -Inf, upper = Inf)$value)
        },
        error = function(e) {
            warning("Expectation not found")
            warning(e$message)
            return(NULL)
        }
    )
}
```

Returns the variance for an object of type cRV
@name variance
@param cRV Is the continuous random variable for which we want to find the variance.
@return The value of the integral to determine the variance

```
variance <- function(cRV) {
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    tryCatch(
        {
            func <- attr(cRV, "pdf")
            new_func <- function(x) {
                ((x - expectation(func)^2) * func(x))
            }
            return(integrate(new_func, lower = -Inf, upper = Inf)$value)
        },
        error = function(e) {
            warning("Variance not found")
            warning(e$message)
            return(NULL)
        }
    )
}
```

Finds the fourth degree initial moments (if they exist) for an object of type cRV
@name initial$_moments$
@param cRV Is the continuous random variable for which we want to find the initial moments
@return A list containing the first four initial moments

```
initial_moments <- function(cRV) {
    # will return a list containing the 4 moments (if they exist)
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    moments_list <- list()
    for (i in 1:4) {
        tryCatch(
            {
                func <- attr(cRV, "pdf")
                new_func <- function(x) {
                    (x^i) * func(x)
                }
                moments_list <- append(
                    moments_list,
                    integrate(new_func, lower = -Inf, upper = Inf)$value
                )
            },
            error = function(e) {
                warning("Moment not found for i=", i)
                warning(e$message)
            }
        )
    }
    return(moments_list)
}
```

Finds the fourth degree central moments (if they exist) for an object of type cRV
@name central$_moments$
@param cRV Is the continuous random variable for which we want to find the central moments
@return A list containing the first four central moments

```
central_moments <- function(cRV) {
    # will return a list containing the 4 moments (if they exist)
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    moments_list <- list()
    for (i in 1:4) {
        tryCatch(
            {
                func <- attr(cRV, "pdf")
                new_func <- function(x) {
                    (x - expectation(func))^i * func(x)
                }
                moments_list <- append(
                    moments_list,
                    integrate(new_func, lower = -Inf, upper = Inf)$value
                )
            },
            error = function(e) {
                warning("Moment not found for i=", i)
                warning(e$message)
            }
        )
    }
    return(moments_list)
}
```

Bonus:

```
factorial_moments <- function(cRV) {
    # will return a list containing the 4 moments (if they exist)
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    moments_list <- list()
    for (i in 1:4) {
        tryCatch(
            {
                func <- attr(cRV, "pdf")
                new_func <- function(x) {
                    (factorial(x) / factorial(x - i)) * func(x)
                }
                moments_list <- append(
                    moments_list,
                    integrate(new_func, lower = -Inf, upper = Inf)$value
                )
            },
            error = function(e) {
                warning("Moment not found for i=", i)
                warning(e$message)
            }
        )
    }
    return(moments_list)
}
```

Chapter 6

Problema 6

6.1 Cerinta

Calculul mediei si dispersiei unei variabile aleatoare g(X), unde X are o repartitie continua cunoscuta iar g este o functie continua precizata de utilizator.

6.2 Rezolvare

Pentru a calcula toate cele patru elemente, am folosit urmatoarele formule:

$$Media = \int_{-\infty}^{\infty} g(x) * f(x)\, dx$$

$$Dispersia = \int_{-\infty}^{\infty} (g(x) - medie)^2 * f(x)\, dx$$

6.3 Cod

```
expected_value <- function(g, cRV, lower = -Inf, upper = Inf) {
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    tryCatch(
        {
            func <- attr(cRV, "pdf")
            new_func <- function(x) {
                g(x) * func(x)
            }
            return(integrate(new_func, lower = lower, upper = upper)$value)
        },
        error = function(e) {
            warning("Mean not found")
            warning(e$message)
            return(NULL)
        }
    )
}


variance <- function(g, cRV, lower = -Inf, upper = Inf) {
    if (class(cRV) != "cRV") {
        warning("Expected cRV object")
    }
    tryCatch(
        {
            func <- attr(cRV, "pdf")
            exp_val <- expected_value(g, func, lower, upper)
            new_func <- function(x) {
                (g(x) - exp_val)^2 * func(x)
            }
            return(integrate(new_func, lower = lower, upper = upper)$value)
        },
        error = function(e) {
            warning("Variance not found")
            warning(e$message)
            return(NULL)
        }
    )
}
```

Chapter 7

Problema 7

7.1 Cerinta

Crearea unei functii P care permite calculul diferitelor tipuri de probabilitati asociate unei variabile aleatoare continue(similar functiei P din pachetul discreteRV)

7.2 Rezolvare

- S-au creat operatorii de comparare pentru tipul de date cRV

- Operatorii pot avea la stanga un obiect de tip cRV si la dreapta un obiect numeric

- Operatorii returneaza un tip nou de date, numit "cRVresult", care retine atat cdf-ul variabilei aleatoare cat si domeniul nou peste care sa fie calculata probabilitatea

- Am creat operatorii logici AND si OR pentru a lucra cu conditii compuse pentru calculul probabilitatilor. Operatorii AND si OR iau ca parametri doua obiecte de tip "cRVresult" si returneaza un obiect de tip "cRVresult".

- functia Pr ia ca parametru un obiect de tip "cRVresult" si returneaza un numar real cuprins intre 0 si 1

7.3 Cod

```
"<.cRV" <- function(X, x) {
    if (class(X) != "cRV") stop("X is not a continuous random variable.")
    if (class(x) != "numeric") stop("x must be a numeric value")

    interv <- Intervals(
      matrix(
        c(-Inf, x),
        byrow = TRUE,
        ncol = 2
      ),
      closed = c(FALSE, FALSE),
      type = "R"
    )

    result <- attr(X, "cdf")
    class(result) <- "cRVresult"
    attr(result, "interval") <- interv

    return(result)
}

"<=.cRV" <- function(X, x) {
  if (class(X) != "cRV") stop("X is not a continuous random variable.")
  if (class(x) != "numeric") stop("x must be a numeric value")

  interv <- Intervals(
    matrix(
      c(-Inf, x),
      byrow = TRUE,
      ncol = 2
    ),
    closed = c(FALSE, TRUE),
    type = "R"
  )

  result <- attr(X, "cdf")
  class(result) <- "cRVresult"
  attr(result, "interval") <- interv

  return(result)
}
```

```r
"<=.cRV" <- function(X, x) {
  if (class(X) != "cRV") stop("X is not a continuous random variable.")
  if (class(x) != "numeric") stop("x must be a numeric value")

  interv <- Intervals(
    matrix(
      c(-Inf, x),
      byrow = TRUE,
      ncol = 2
    ),
    closed = c(FALSE, TRUE),
    type = "R"
  )

  result <- attr(X, "cdf")
  class(result) <- "cRVresult"
  attr(result, "interval") <- interv

  return(result)
}

">.cRV" <- function(X, x) {
  if (class(X) != "cRV") stop("X is not a continuous random variable.")
  if (class(x) != "numeric") stop("x must be a numeric value")

  interv <- Intervals(
    matrix(
      c(x, Inf),
      byrow = TRUE,
      ncol = 2
    ),
    closed = c(FALSE, FALSE),
    type = "R"
  )

  result <- attr(X, "cdf")
  class(result) <- "cRVresult"
  attr(result, "interval") <- interv

  return(result)
}
```

```
"==.cRV" <- function(X, x) {
  if (class(X) != "cRV") stop("X is not a continuous random variable.")
  if (class(x) != "numeric") stop("x must be a numeric value")

  interv <- Intervals(
    matrix(
      c(x, x),
      byrow = TRUE,
      ncol = 2
    ),
    closed = c(TRUE, TRUE),
    type = "R"
  )

  result <- attr(X, "cdf")
  class(result) <- "cRVresult"
  attr(result, "interval") <- interv

  return(result)
}
```

@param Xres The result of comparing a cRV with a numeric value

@param Yres The result of comparing a cRV with a numeric value

```
"|.cRVresult" <- function(Xres, Yres) {
  ORret <- Xres
  attr(ORret, "interval") <- interval_union(attr(Xres, "interval"),
                                            attr(Yres, "interval"))
  return(ORret)
}
```

@param Xres The result of comparing a cRV with a numeric value

@param Yres The result of comparing a cRV with a numeric value

```
"&.cRVresult" <- function(Xres, Yres) {
  ORret <- Xres
  attr(ORret, "interval") <- interval_intersection(attr(Xres, "interval"),
                                                   attr(Yres, "interval"))
  return(ORret)
}
```

```
Pr <- function(cResult) {
  if (class(cResult) != "cRVresult") stop("Incorrect type for parameter")

  # calculate integral over interval using cdf
  calcFunction <- function(interv) {
    rvl <- 0
    if (interv[2] != -Inf)
      rvl <- cResult(interv[2])

    lvl <- 0
    if (interv[1] != -Inf)
      lvl <- cResult(interv[1])

    return(rvl - lvl)
  }

  sum(apply(attr(cResult, "interval"), 1, calcFunction))
}
```

7.4 Exemple

pdf1 $\leftarrow function(x)\{(0 \Leftarrow x \& x \Leftarrow 1) * (3 * x^2)\}$

X $\leftarrow cRV(pdf1)$

Pr((X $<$ 0.1) $\|(X > 0.2)\|((X > 0.1)\&(X < 0.2)))$

Chapter 8

Problema 8

8.1 Cerinta

Afisarea unei "fise de sinteza" care sa contina informatii de baza despre respectiva repartitie(cu precizarea sursei informatiei!). Relevant aici ar fi sa precizati pentru ce e folosita in mod uzual acea repartitie, semnificatia parametrilor, media, dispersia etc. 1

8.2 Rezolvare

La apelarea functiei Info, utilizatorul este intampinat cu un meniu din care poate alege repartitia desprea care doreste sa vada mai multe informatii.

8.3 Cod

```
normal <- function() {
    x <- "Normal distribution (Gaussian distribution), for a single such quantity; the most commonly used absolutely continuous distribution.
    Applications: Linear growth (e.g. errors, offsets)
    Notation: N(mu, sigma^2)
    Parameters: mu -> mean (location); sigma^2 -> varaince (squared scale)
    PDF: 1/(sigma*sqrt(2pi))*e^(-1/2 * ((x-mu)/sigma)^2)
    CDF: 1/2*[1+erf((x-mu)/(sigma*sqrt(2))]
    Mean: mu
    Median: mu
    Variance: sigma^2"
}


pareto <- function() {
    x <- "Pareto distribution, for a single such quantity whose log is exponentially distributed; the prototypical power law distribution
    Applications: Exponential growth (e.g. prices, incomes, populations)
    Parameters: xm -> scale; alpha -> shape
    PDF: alpha*xm^alpha/x^(alpha+1)
    CDF: 1 - (xm - x)^alpha
    Mean: Inf, alpha<=1; alpha*xm/(alpha-1), alpha>1
    Median: xm*2^(1/alpha)
    Variance: Inf, alpha<=2; xm^2*alpha/((alpha-1)^2(alpha-2)), alpha>2"
}

uniform <- function() {
    x <- "Continuous uniform distribution, for absolutely continuously distributed values
    Applications: Uniformly distributed quantities
    Notation: U(a,b)
    Parameters: -Inf < a < b < Inf
    PDF: 1/(b-a), a < x < b; 0, otherwise
    CDF: 0, x<a; (x-a)/(b-a), a < x < b; 1, x>b
    Mean: 1/2*(a+b)
    Variance: 1/12 * (b-a)^2"
}
```

```
bernoulli <- function() {
    x <- "Bernoulli distribution, for the outcome of a single Bernoulli trial (e.g. success/failure, yes/no)
    Applications: Bernoulli trials (yes/no events, with a given probability)
    Parameters: 0 <= p <= 1; q = 1-p
    PMF: q = 1-p, k=0; p, k=1
    CDF: 0, k<0; 1-p, 0<=k<1; 1, k>=1
    Mean: p
    Variance: pq"
}

hypergeometric <- function() {
    x <- "Hypergeometric distribution, for the number of positive occurrences (e.g. successes, yes votes, etc.) given a fixed number of total occurrences, using sampling witho
    Applications: Sampling schemes over a finite population yes/no events, with a given probability
    Parameters: 0 <= N < Inf; 0 <= K < N; 0 <= n < N
    PMF: kCK * (N-K)C(n-k)/(NCn)
    Mean: n*K/N
    Variance: n*K/N * (N-K)/N * (N-n)/(N-1)
    "
}

multinomial <- function() {
    x <- "Multinomial distribution, for the number of each type of categorical outcome, given a fixed number of total outcomes; a generalization of the binomial distribution
    Applications: Categorical outcomes (events with K possible outcomes)
    Parameters: n > 0, number of trials; k > 0, number of mutually exclusive events; p1....pn event probabilities
    PMF: n!/(x1!*...*xk!) *p1^x1 *...* pk^xk
    Mean: E(Xi) = npi
    Variance: Var(Xi) = npi(1-pi)"
}

i_poisson <- function() {
    x <- "Poisson distribution, for the number of occurrences of a Poisson-type event in a given period of time
    Applications: Poisson process (events that occur independently with a given rate)
    Notation: Pois(lambda)
    Parameters: 0 < lambda < Inf, rate
    PMF: lambda^k * e^-lambda / k!
    Mean: lambda
    Variance: lambda"
}
```

```
exponential <- function() {
    x <- "Exponential distribution, for the time before the next Poisson-type event occurs
    Applications: The exponential distribution occurs naturally when describing the lengths of the inter-arrival times in a homogeneous Poisson process.
    Parameters: lambda > 0 , rate
    PDF: lamdba*e^(-lambda*x)
    CDF: 1- e ^(-lamdba*x)
    Mean: 1/lamdba
    Median: ln(2)/lambda
    Variance: 1/lambda^2"
}

rayleigh <- function() {
    x <- "Rayleigh distribution, for the distribution of vector magnitudes with Gaussian distributed orthogonal components. Rayleigh distributions are found in RF signals with
    Applications: Absolute values of vectors with normally distributed components
    Parameters: sigma > 0, scale
    PDF: x/sigma^2 * e^(-x^2/2sigma^2)
    Mean: sigma*sqrt(pi/2)
    Median: sigma*sqrt(2*ln(2))
    Variance: (4-pi)/2 * sigma^2"
}

chisquared <- function() {
    x <- "Chi-squared distribution, the distribution of a sum of squared standard normal variables; useful e.g. for inference regarding the sample variance of normally distrib
    Applications: Normally distributed quantities operated with sum of squares
    Notation: X^2(k)
    Parameters: k != 0, degrees of freedom
    Mean: k
    Variance: 2k"
}
```

```
info <- function() {
    message("   1.Normal
            2.Pareto
            3.Uniform
            4.Bernoulli
            5.Hypergeometric
            6.Multinomial
            7.Poisson
            8.Exponential
            9.Rayleigh
            10.Chi-squared
    ")
    option <- as.numeric(readline("Enter choice: "))
    x <- switch(option,
        normal(),
        pareto(),
        uniform(),
        bernoulli(),
        hypergeometric(),
        multinomial(),
        i_poisson(),
        exponential(),
        rayleigh(),
        chisquared()
    )
    message(x)
    message("Source: Curs + https://en.wikipedia.org/wiki/Probability_distribution")
}
```

Chapter 9

Problema 11

9.1 Cerinta

Pornind de la densitatea comuna a doua variabile aleatoare continue, construirea densitatilor marginale si a densitatilor conditionate.

9.2 Rezolvare

Am folosit formulele de calcul pentru densitatile marginale

$$f_X(x) = \int_c^d f(x, y)\, dx$$

$$f_Y(y) = \int_a^b f(x, y)\, dx$$

Folosind cele doua densitati marginale am calculat cele doua densitati conditionate.

$$f_1(x|y) = \frac{f(x, y)}{f_y(y)}$$

$$f_2(y|x) = \frac{f(x, y)}{f_x(x)}$$

9.3 Cod

Get marginal distribution for X from a bivariate pdf

@param pdf must be a probability density fonction for a bivariate random variable

```
X_marginal_dist <- function(pdf) {
  if (length(formals(pdf)) != 2)
    stop('Pdf must take two arguments.')
  if (!is_joint_pdf(pdf))
    stop('Parameter pdf must be a joint pdf.')

  function(x) {
    new_f <- function(y) { pdf(x, y) }
    integral(new_f, bound = list(y= c(-Inf, Inf)))$value
  }
}
```

Exemplu:

pdf2 <- function(x, y) { (0 <= x & x <= 1 & 0 <= y & y <= 1) (2/3 (x + 2y)) }

$f_X < -X\_marginal\_dist(pdf2)$

$f_X(0.4)$

Get marginal distribution for X from a bivariate pdf

@param pdf must be a probability density fonction for a bivariate random variable

```
Y_marginal_dist <- function(pdf) {
  if (length(formals(pdf)) != 2)
    stop('Pdf must take two arguments.')
  if (!is_joint_pdf(pdf))
    stop('Parameter pdf must be a joint pdf.')

  function(y) {
    new_f <- function(x) { pdf(x, y) }
    integral(new_f, bound = list(x= c(-Inf, Inf)))$value
  }
}
```

Exemplu:

pdf2 <- function(x, y) { (0 <= x & x <= 1 0 <= y & y <= 1) (2/3 (x + 2y)) }

$f_Y < -Y\_marginal\_dist(pdf2)$

$f_Y(0.2)$

Get a conditional distribution from a bivariate pdf

@param pdf must be a probability density fonction for a bivariate random variable

@param x fixed value of X. Providing x will result in returning f(Y | X = x)

@param y fixed value of Y. Providing y will result in returning f(X | Y = x)

```r
cond_distribution <- function(pdf, x = NULL, y = NULL) {
  if ((is.null(x) && is.null(y)) || !(is.null(x) || is.null(y))) {
    stop('Must either provide a value for X or a value for Y.')
  }

  if (!is_joint_pdf(pdf))
    stop('Parameter pdf must be a joint pdf.')

  if (is.null(x)) {
    # conditional given y f(X | Y = y)
    mdist <- Y_marginal_dist(pdf)
    denom <- mdist(y)

    function(x) {
      pdf(x, y) / denom
    }
  }
  else {
    # conditional given x: f(Y | X = x)
    mdist <- X_marginal_dist(pdf)
    denom <- mdist(x)

    function(y) {
      pdf(x, y) / denom
    }
  }
}
```

Exemplu:

pdf2 <- function(x, y) (0 <= x & x <= 1 & 0 <= y & y <= 1) (2/3 (x + 2y))

Conditional distribution of X given Y

$f_X Y < -cond\_distribution(pdf2, y = 0.3)$

$f_X Y(0.6)$

Conditional distribution of Y given X

$f_Y X < -cond\_distribution(pdf2, x = 0.1)$

$f_Y X(0.75)$

Chapter 10

Problema 12

10.1 Cerinta

Construirea sumei si diferentei a doua variabile aleatoare continue independente(folositi formula de convolutie)

10.2 Rezolvare

Formulele folosite:

$$Suma : f_t = \int_{-\infty}^{\infty} f(\rho) * (g(t-\rho))\, d\rho$$

$$Diferenta : f_t = \int_{-\infty}^{\infty} f(\rho) * (g(\rho-t))\, d\rho$$

10.3 Cod

@name difCRV

@param cRV1 The first continous random variable

@param cRV2 The second continous random variable

@return The difference of the two continous random variables

```
difCRV <- function(cRV1, cRV2) {
    if (class(cRV1) != "cRV" || class(cRV2) != "cRV") {
        warning("Expected cRV object")
    }
    fun1 <- attr(cRV1, "pdf")
    fun2 <- attr(cRV2, "pdf")
    function(t) {
        integrate(
            f = function(r) {
                fun1(r) * fun2(t - r)
            },
            lower = -Inf,
            upper = Inf
        )$value
    }
}
```

@name sumCRV
@param cRV1 The first continous random variable
@param cRV2 The second continous random variable
@return The sum of the two continous random variables

```
sumCRV <- function(cRV1, cRV2) {
    if (class(cRV1) != "cRV" || class(cRV2) != "cRV") {
        warning("Expected cRV object")
    }
    fun1 <- attr(cRV1, "pdf")
    fun2 <- attr(cRV2, "pdf")
    function(t) {
        integrate(
            f = function(r) {
                fun1(r) * fun2(r - t)
            },
            lower = -Inf,
            upper = Inf
        )$value
    }
}
```

Chapter 11

Concluzie

Pachetul continuousRV poate fi folosit pentru lucrul cu variabile aleatoare continue, deoarece defineste un nou tip de date pentru acestea si functii pentru procesarea lor.

Chapter 12

Bibliografie

.

https://en.wikipedia.org/wiki/Random_variable

https://en.wikipedia.org/wiki/Probability_distribution

http://cs.unitbv.ro/~pascu/stat/Variabile%20aleatoare.pdf