DJILLALI LIABES UNIVERSITY OF SIDI BEL ABBES
FACULTY OF EXACT SCIENCES
DEPARTMENT OF COMPUTER SCIENCES

Module : Aide à la décision
1ST YEAR OF MASTER'S DEGEREE IN
NETWORKS,INFORMATION SYSTEMS & SECURITY(RSSI)
2021/2022

---

discrete and continuous probability distributions in R
TP-02

Students:
HADJAZI M.Hisham
AMOUR Wassim Malik
Group: 01/RSSI

Instructors:
Pr. YOUSFATE Abderrahmane
Dr. BENBEKRITI Soumia

A paper submitted in fulfilment of the requirements for the
Aide à la décision TP-02

March 17, 2022

Contents

Chapter 1

Discrete and Continuous Probability Distributions in R

1.1 Discrete distributions

1.1.1 Bernoulli distribution

A Bernoulli random variable is a variable that can only take on the values 0 and 1. We let p be the probability of 1, and $(1-p)$ the probability of 0. This example is easy to analyze, and MANY interesting random variables can be built from this simple building block.

$$fx(x) = \begin{cases} 1-p & x = 0 \\ p & x = 1 \\ 0 & otherwise \end{cases}$$

- $E(X) = p$
- $Var(X) = p(1-p)$
- $M_x = 1 + p(e^t - 1)$

A random variable X is said to have the Bernoulli distribution with parameter p if $P(X = 1) = p$ and $P(X = 0) = 1 - p$, where $0 < p < 1$.

Suppose that n independent Bernoulli trials are performed, each with the same success probability p. Let X be the number of successes. The distribution of X is called the Binomial distribution with parameters n and p. where n is a positive integer and $0 < p < 1$.[1]

Bernoulli Probability Density Function (dbern Function)

```
# Install Rlab package
install.packages("Rlab")

# Load Rlab package.skeleton
library("Rlab")

# Specify x-values for dbern function
x_dbern <- seq(0, 10, by = 1)
```

```

# Apply dbern function
y_dbern <- dbern(x_dbern, prob = 0.7)

# Plot dbern values
plot(y_dbern, type = "o")
```

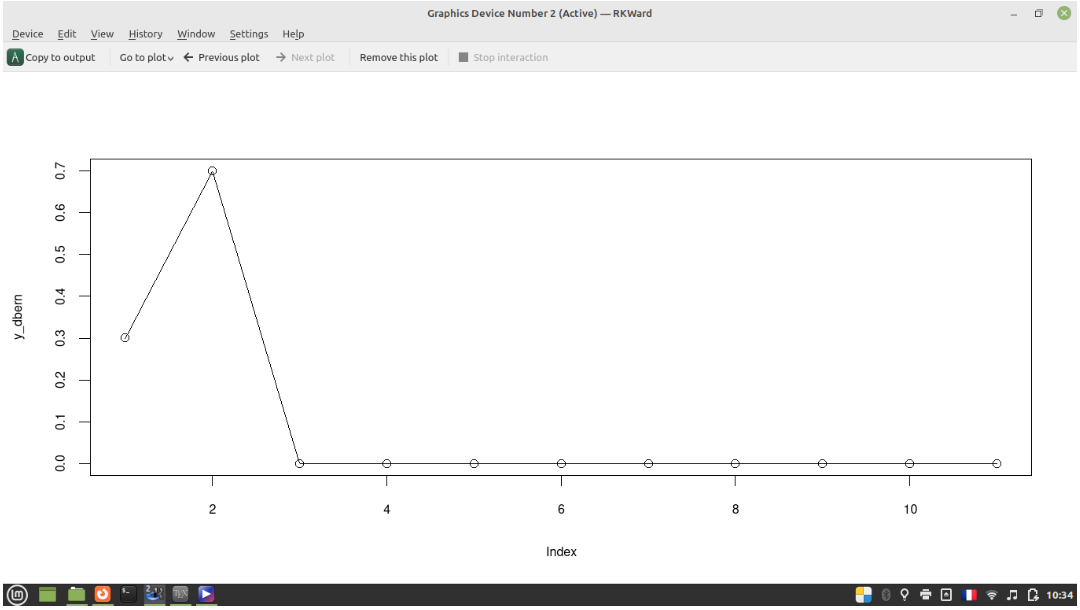

FIGURE 1.1: dbern Function plot.

Bernoulli Cumulative Distribution Function (pbern Function)

```
# Specify x-values for pbern function
x_pbern <- seq(0, 10, by = 1)

# Apply pbern function
y_pbern <- pbern(x_pbern, prob = 0.7)

# Plot pbern values
plot(y_pbern, type = "o")
```

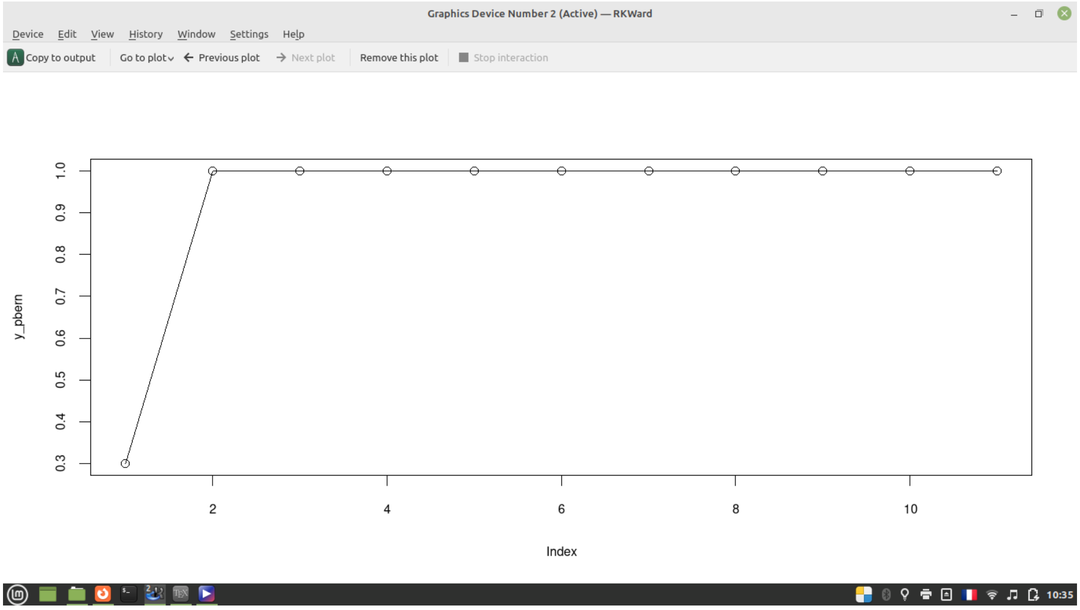

FIGURE 1.2: pbern Function plot.

Bernoulli Quantile Function (qbern Function)

```
# Specify x-values for qbern function
x_qbern <- seq(0, 1, by = 0.1)

# Apply qbern function
y_qbern <- qbern(x_qbern, prob = 0.7)

# Plot qbern values
plot(y_qbern, type = "o")
```

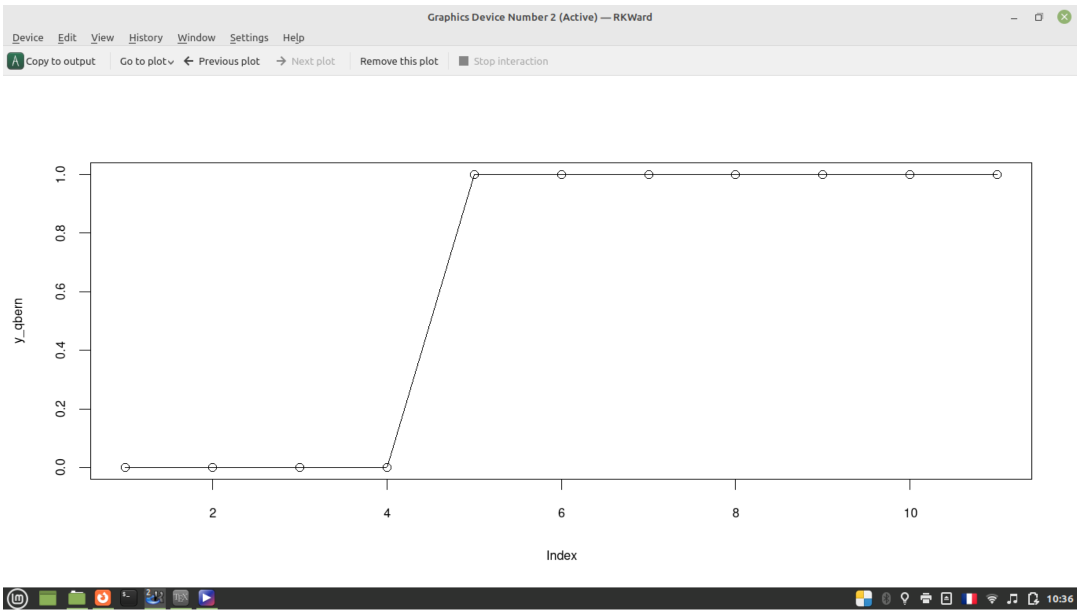

FIGURE 1.3: qbern Function plot.

Generating Random Numbers (rbern Function)

```r
# Set seed for reproducibility
set.seed(98989)

# Specify sample size
N <- 10000

# Draw N random values
y_rbern <- rbern(N, prob = 0.7)

# Print values to RStudio console
y_rbern

# Plot of randomly drawn density
hist(y_rbern,
     breaks = 5,
     main = "")
```

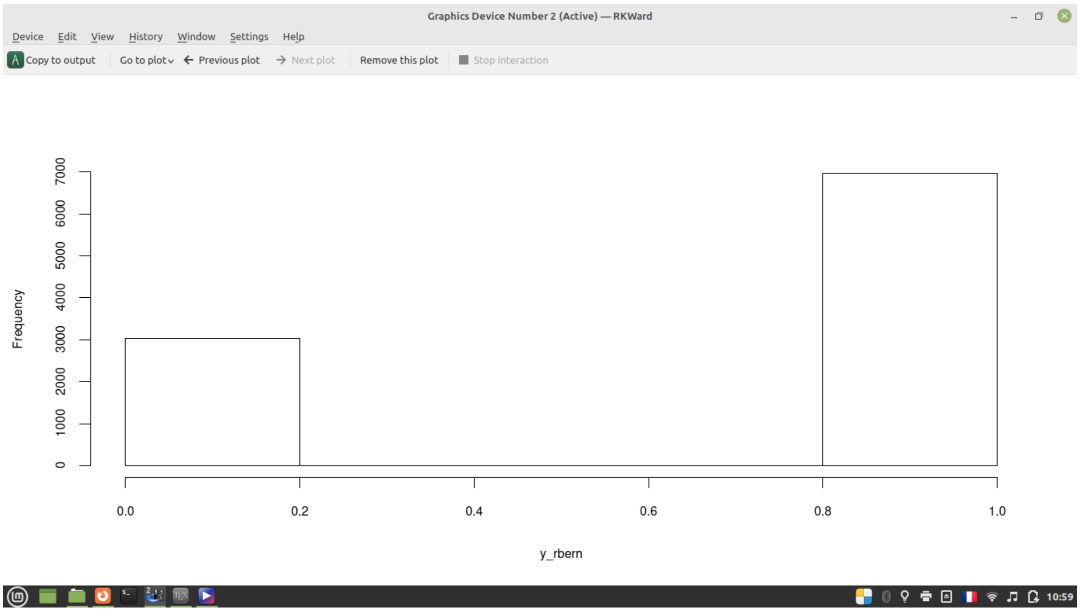

FIGURE 1.4: rbern Function plot.

1.1.2 Binomial distribution

A coin-tossing experiment is a simple example of an important discrete random variable called the binomial random variable. Many practical experiments result in data similar to the head or tail outcomes of the coin toss. For example, consider the political polls used to predict voter preferences in elections. Each sampled voter can be compared to a coin because the voter may be in favor of our candidate a "head" or not a "tail." In most cases, the proportion of voters who favor our candidate does not equal 1/2; that is, the coin is not fair. In fact, the proportion of voters who favor our candidate is exactly what the poll is designed to measure! Here are some other situations that are similar to the coin-tossing experiment:

- The number of heads/tails in a sequence of coin flips.
- Vote counts for two different candidates in an election.
- The number of male/female employees in a company.
- The number of successful sales calls.
- The number of defective products in a production run.
- A sociologist is interested in the proportion of elementary school teachers who are men.
- A soft drink marketer is interested in the proportion of cola drinkers who prefer her brand.
- A geneticist is interested in the proportion of the population who possess a gene linked to Alzheimer's disease.
- The number of accounts that are in compliance or not in compliance with an accounting procedure.
- The number of days in a month your company's computer network experiences a problem.

Each sampled person is analogous to tossing a coin, but the probability of a "head" is not necessarily equal to 1/2. Although these situations have different practical objectives, they all exhibit the common characteristics of the binomial experiment.[5]

A binomial experiment is one that has these five characteristics:

- The experiment consists of η identical trials.
- Each trial results in one of two outcomes. For lack of a better name, the one outcome is called a success, S, and the other a failure, F.
- The probability of success on a single trial is equal to n and remains the same from trial to trial. The probability of failure is equal to $(1 - p) = q$.
- The trials are independent.
- We are interested in x, the number of successes observed during the n trials, for $x = 0, 1, 2, ..., n$.

Under the above assumptions, let X be the total number of successes. Then, X is called a binomial random variable, and the probability distribution of X is called the binomial distribution.

A binomial experiment consists of n identical trials with probability of success p on each trial. The probability of k successes in n trials is

$$P(x = k) = C_k^n p^b q^{n-k} = \frac{n!}{k!(n-k)!} p^k q^{n-k}$$

for values of $k = 0, 1, 2, ..., n$. The symbol C_k^n equals $\frac{n!}{k!(n-k)!}$

where $n! = n(n-1)(n-2)...(2)(1)$ and $0! \equiv 1$.

Example 1: Binomial Density in R (dbinom Function)

```
# Specify x-values for binom function
x_dbinom <- seq(0, 100, by = 1)

# Apply dbinom function
y_dbinom <- dbinom(x_dbinom, size = 100, prob = 0.5)

# Plot dbinom values
plot(y_dbinom)
```

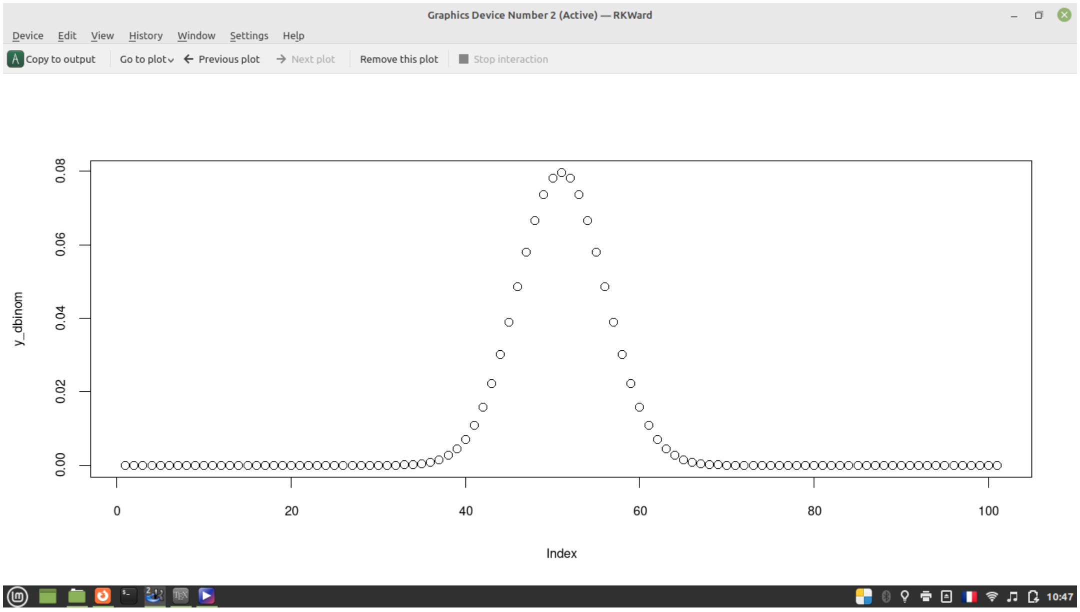

FIGURE 1.5: dbinom Function plot.

Example 2: Binomial Cumulative Distribution Function (pbinom Function)

```
# Specify x-values for pbinom function
x_pbinom <- seq(0, 100, by = 1)

# Apply pbinom function
y_pbinom <- pbinom(x_pbinom, size = 100, prob = 0.5)

# Plot pbinom values
plot(y_pbinom)
```

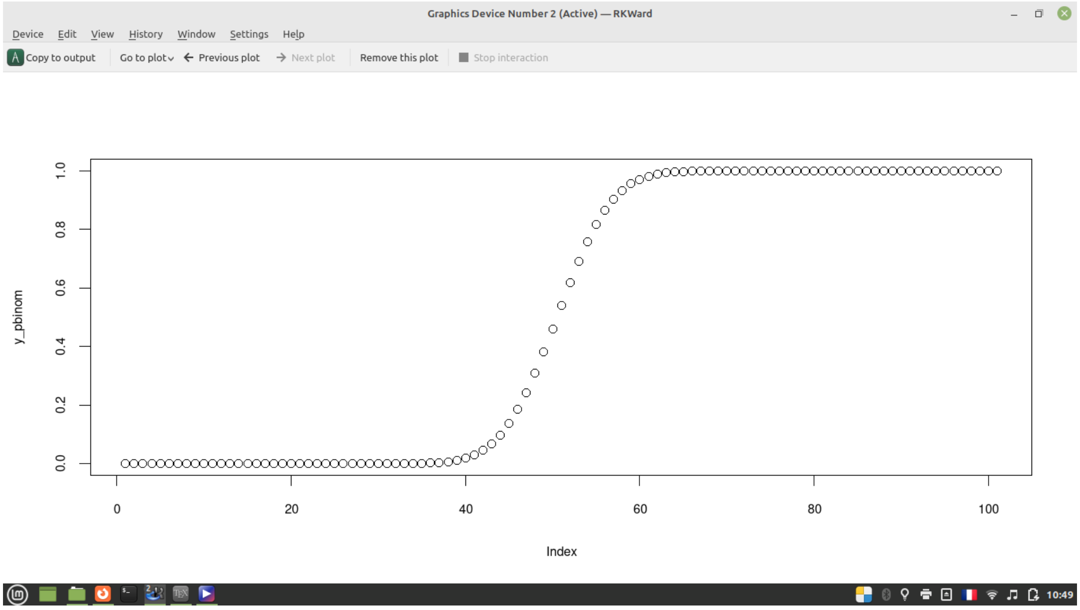

FIGURE 1.6: pbinom Function plot.

Example 3: Binomial Quantile Function (qbinom Function)

```
# Specify x-values for qbinom function
x_qbinom <- seq(0, 1, by = 0.01)

# Apply qbinom function
y_qbinom <- qbinom(x_qbinom, size = 100, prob = 0.5)

# Plot qbinom values
plot(y_qbinom)
```

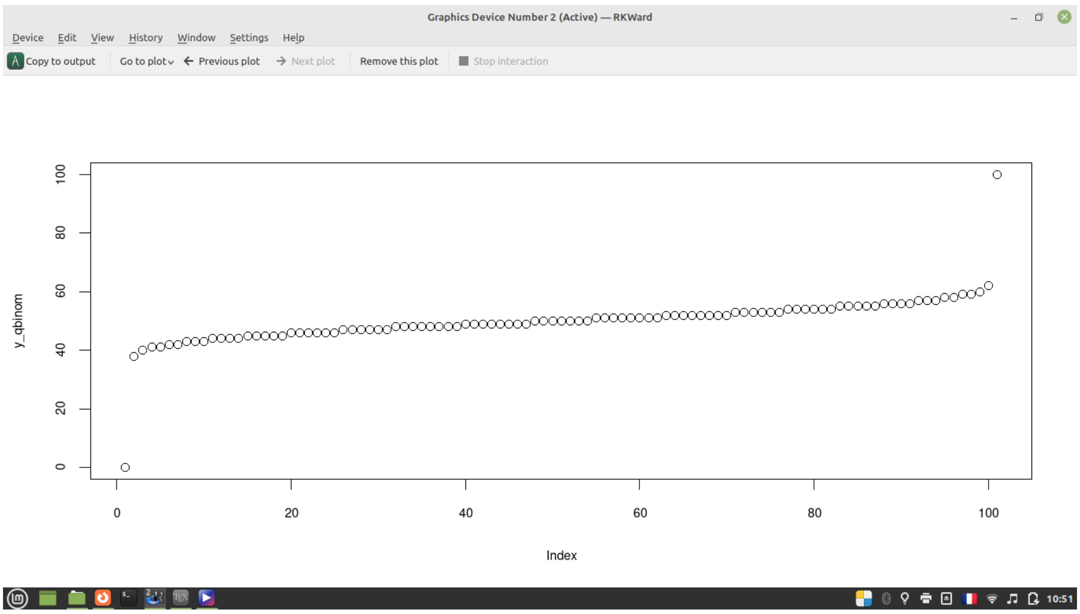

FIGURE 1.7: qbinom Function plot.

Example 4: Simulation of Random Numbers (rbinom Function)

```
# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <- 10000

# Draw N binomially distributed values
y_rbinom <- rbinom(N, size = 100, prob = 0.5)

# Print values to RStudio console  45 44 55 43 35 47 56 52 49 51 47 50 51 54 53 48 57 55 51...
y_rbinom

# Plot of randomly drawn binomial density
hist(y_rbinom,
     breaks = 100,
     main = "")
```

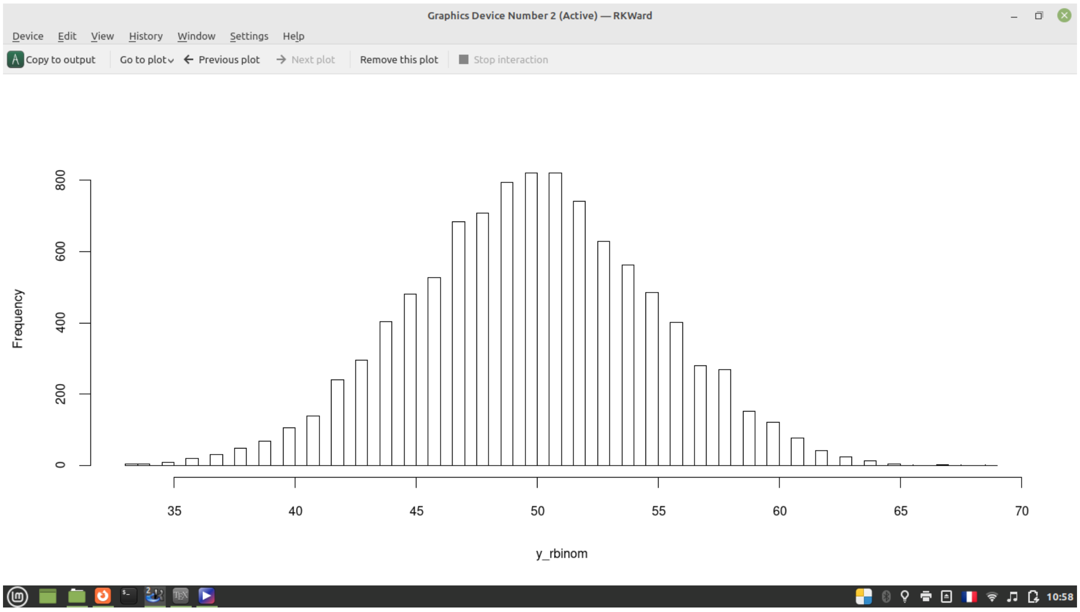

FIGURE 1.8: rbinom Function plot.

1.1.3 Poisson distribution

Another discrete random variable that has numerous practical applications is the Poisson random variable. Its probability distribution provides a good model for data that represent the number of occurrences of a specified event in a given unit of time or space. Here are some examples of experiments for which the random variable x can be modeled by the Poisson random variable:

Like the Binomial distribution, the Poisson distribution arises when a set of canonical assumptions are reasonably valid. These are:

- The number of bacteria per small volume of fluid.
- The number of customer arrivals at a checkout counter during a given minute.
- The number of traffic accidents on a section of freeway during a given time period.
- The number of calls received by a technical support specialist during a given period of time
- The number of machine breakdowns during a given day.
- The number of events that occur in any time interval is independent of the number of events in any other disjoint interval. Here, “time interval” is the standard example of an “exposure variable” and other interpretations are possible. Example: Error rate per page in a book.
- The distribution of number of events in an interval is the same for all intervals of the same size.
- For a “small” time interval, the probability of observing an event is proportional to the length of the interval. The proportionality constant corresponds to the “rate” at which events occur.
- The probability of observing two or more events in an interval approaches zero as the interval becomes smaller.

In each example, x represents the number of events that occur in a period of time or space during which an average of μ such events can be expected to occur. The only assumptions needed when one uses the Poisson distribution to model experiments such as these are that the counts or events occur randomly and independently of one another. [5]

Let μ be the average number of times that an event occurs in a certain period of time or space. The probability of k occurrences of this event is :

$$P(x = k))\frac{\mu^k e^{-\mu}}{k!}$$

for values of $k = 0, 1, 2, 3, ...$ The mean and standard deviation of the Poisson random variable x are :

Example 1: Poisson Density in R (dpois Function)

```r
# Specify x-values for dpois function
x_dpois <- seq(- 5, 30, by = 1)

# Apply dpois function
y_dpois <- dpois(x_dpois, lambda = 10)

# Plot dpois values
plot(y_dpois)
```

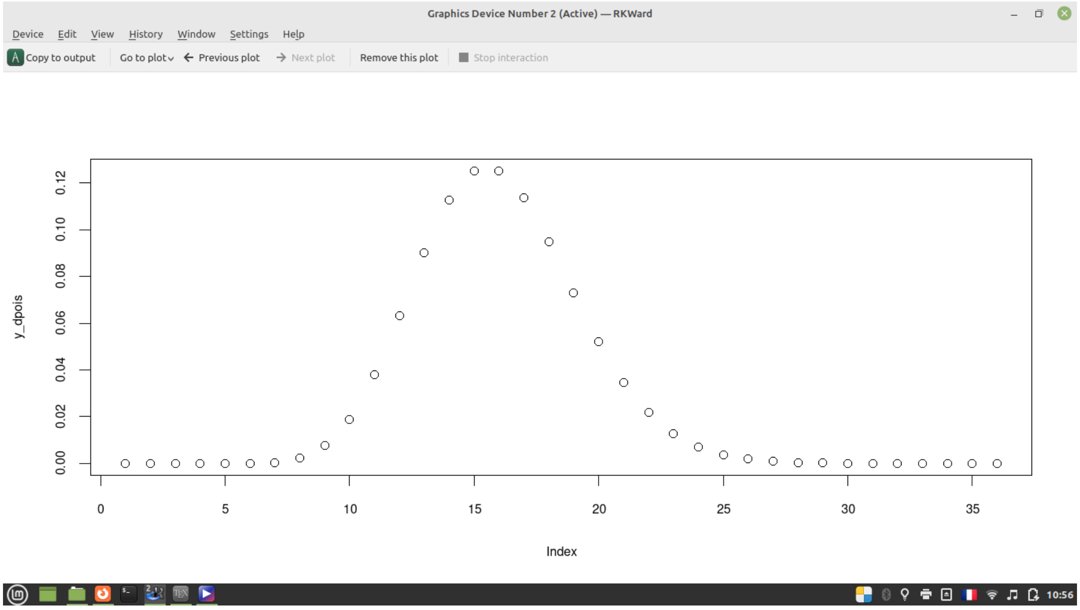

FIGURE 1.9: dpois Function plot.

Example 2: Poisson Distribution Function (ppois Function)

```r
# Specify x-values for ppois function
x_ppois <- seq(- 5, 30, by = 1)

# Apply ppois function
y_ppois <- ppois(x_ppois, lambda = 10)

# Plot ppois values
plot(y_ppois)
```

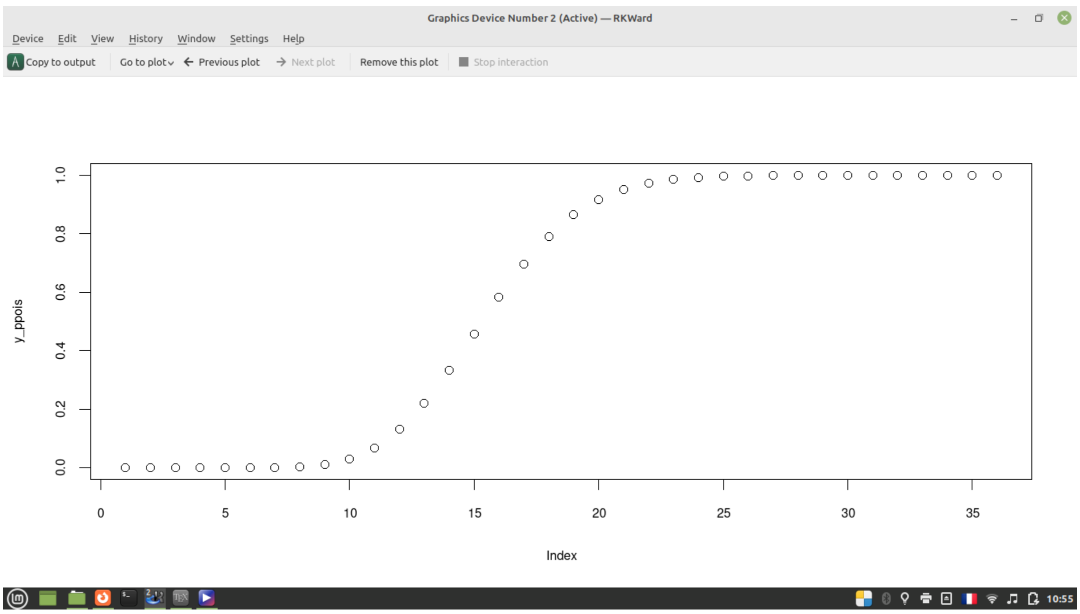

FIGURE 1.10: ppois Function plot.

Example 3: Poisson Quantile Function (qpois Function)

```r
# Specify x-values for qpois function
x_qpois <- seq(0, 1, by = 0.005)

# Apply qpois function
y_qpois <- qpois(x_qpois, lambda = 10)

# Plot qpois values
plot(y_qpois)
```

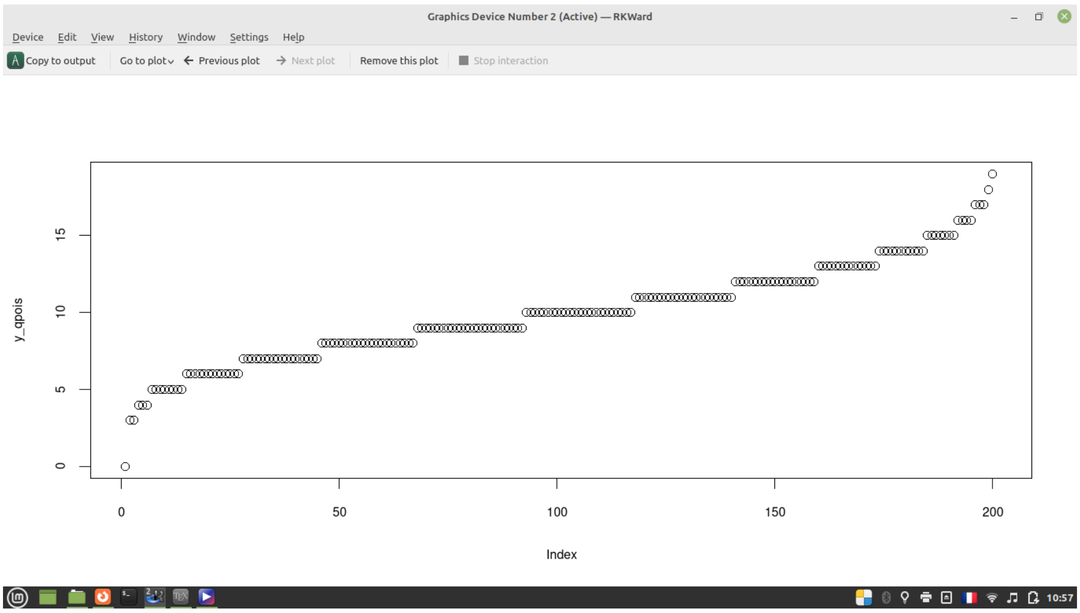

FIGURE 1.11: qpois Function plot.

Example 4: Random Number Generation (rpois Function)

```r
# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <-10000

# Draw N poisson distributed values
y_rpois <- rpois(N, lambda = 10)

# Print values to RStudio console      6 14  8 16  6 12 10  6  7 11  7 12 10 16  7  7  7 19 13
y_rpois

# Plot histogram of rpois values
hist(y_rpois,
     breaks = 100,
     main = "Poisson Distribution in R")
```

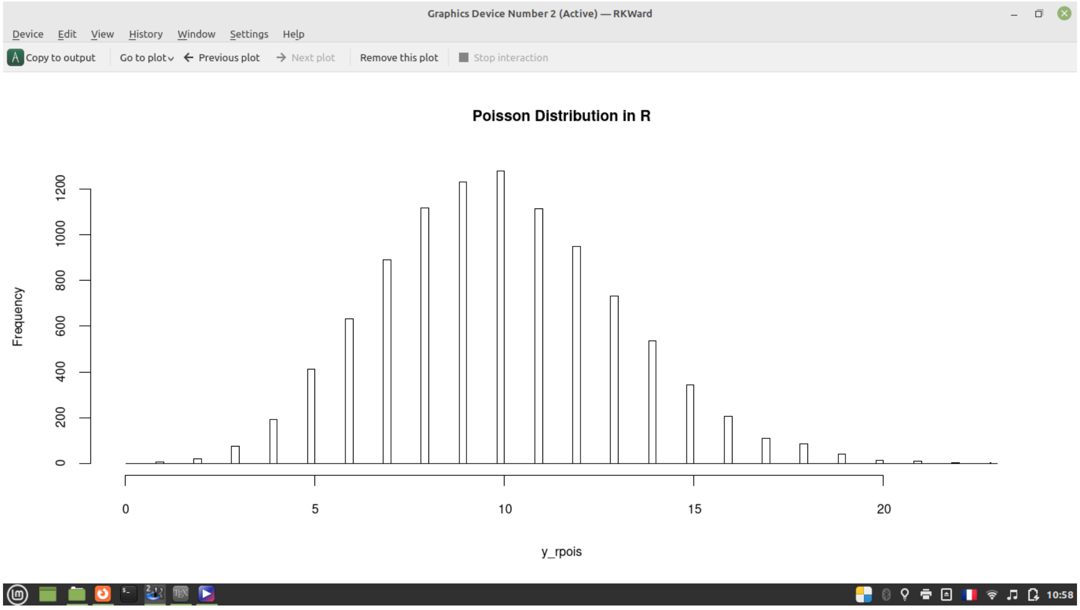

FIGURE 1.12: rpois Function plot.

1.2 Continuous distributions

1.2.1 Uniform distribution

The uniform distribution is the simplest example of a continuous probability distribution. A random variable X is said to be uniformly distributed if its density function is given by:

$$f(x) = \frac{1}{b-1}$$

for $-\infty < a \leq x \leq b < \infty$

Visually, we have :

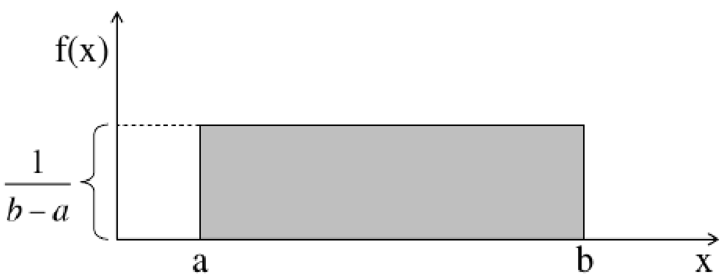

FIGURE 1.13: The uniform probability distribution.

where the shaded region has area $(b-a)[1/(b-a)] = 1$ (width times height).

Example 1: Uniform Probability Density Function (dunif Function)

```
# Specify x-values for dunif function
x_dunif <- seq(0, 100, by = 1)
```

```
# Apply dunif function
y_dunif <- dunif(x_dunif, min = 10, max = 50)

# Plot dunif values
plot(y_dunif, type = "o")
```

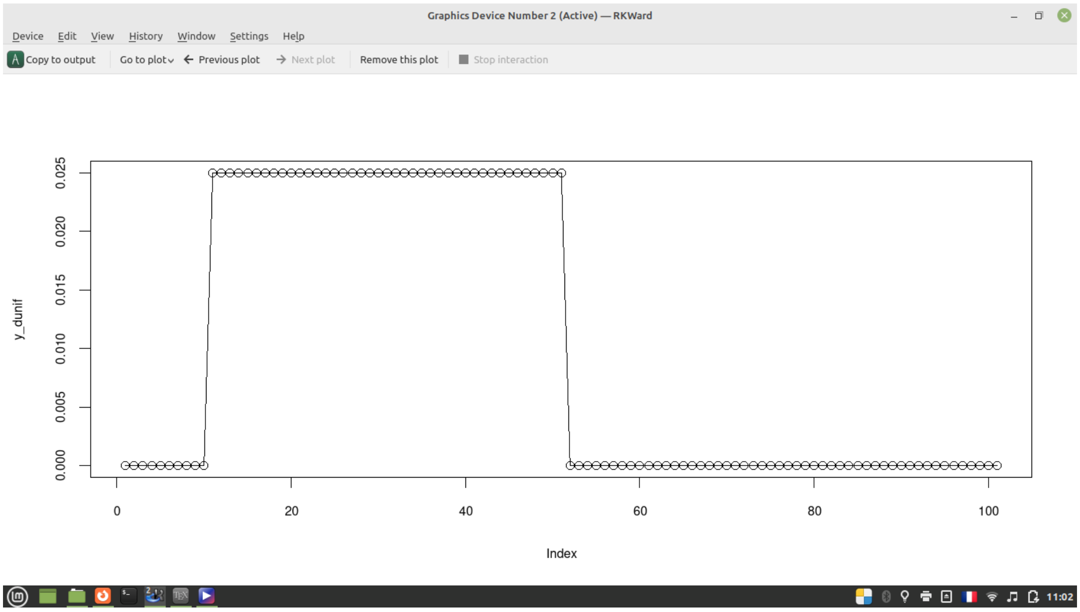

FIGURE 1.14: dunif Function plot.

Example 2: Uniform Cumulative Distribution Function (punif Function)

```
# Specify x-values for punif function
x_punif <- seq(0, 100, by = 1)

# Apply punif function
y_punif <- punif(x_punif, min = 10, max = 50)

# Plot punif values
plot(y_punif, type = "o")
```

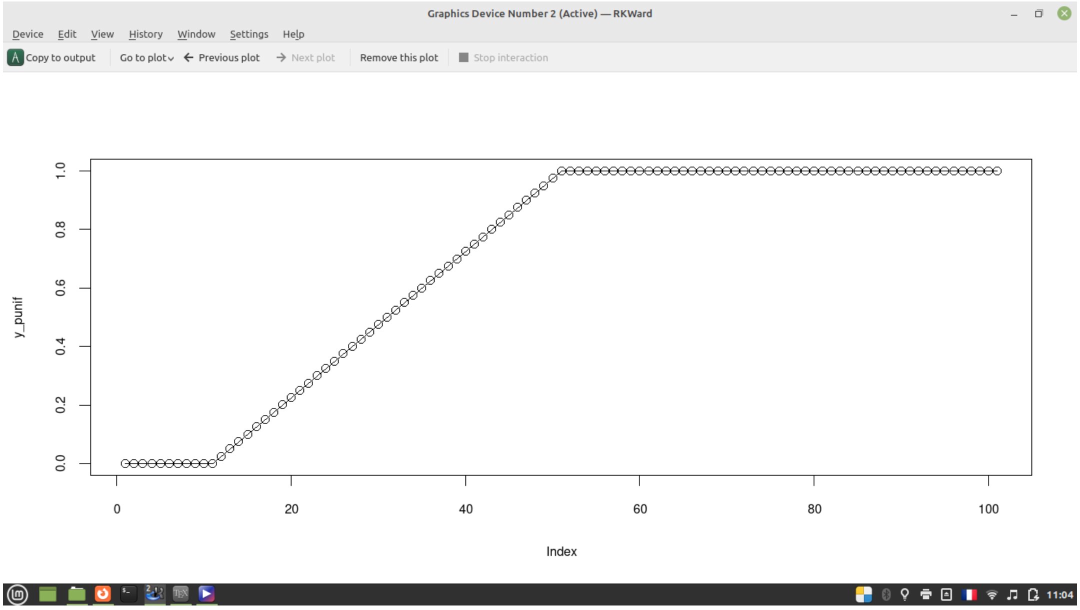

FIGURE 1.15: punif Function plot.

Example 3: Uniform Quantile Function (qunif Function)

```
# Specify x-values for qunif function
x_qunif <- seq(0, 1, by = 0.01)

# Apply qunif function
y_qunif <- qunif(x_qunif, min = 10, max = 50)

# Plot qunif values
plot(y_qunif, type = "o")
```

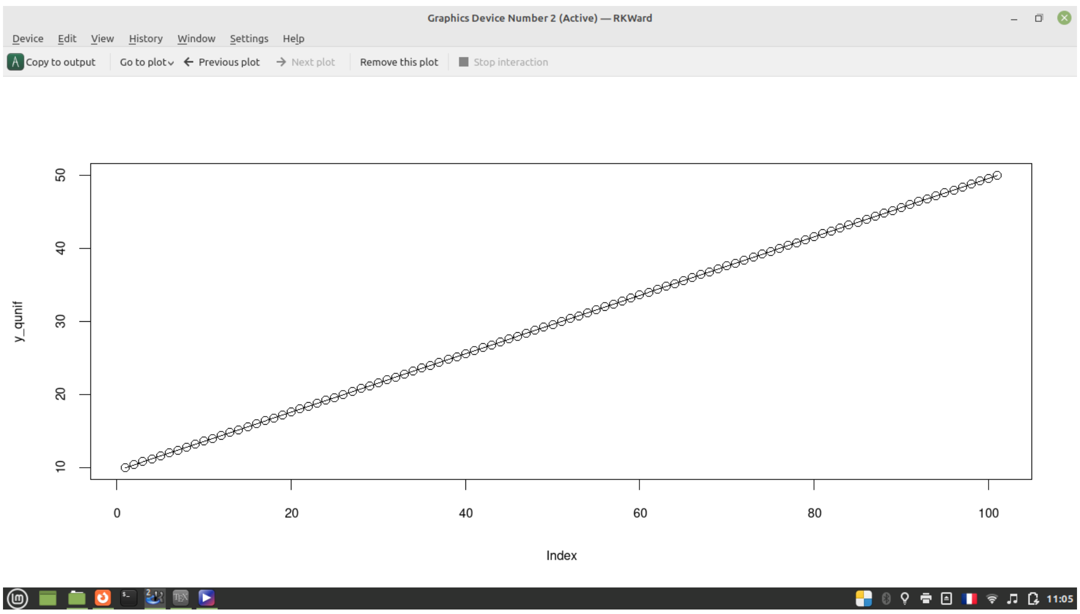

FIGURE 1.16: qunif Function plot.

Example 4: Generating Random Numbers (runif Function)

```
# Set seed for reproducibility
set.seed(91929)

# Specify sample size
N <- 1000000

# Draw N uniformly distributed values
y_runif <- runif(N, min = 10, max = 50)

# Print values to RStudio console   27.98052 49.43937 24.66723
  39.36479 36.84591 40.94262 25.38942 35.59081...
y_runif

# Plot of randomly drawn uniformly density
hist(y_runif,
     breaks = 50,
     main = "",
     xlim = c(0, 100))
```

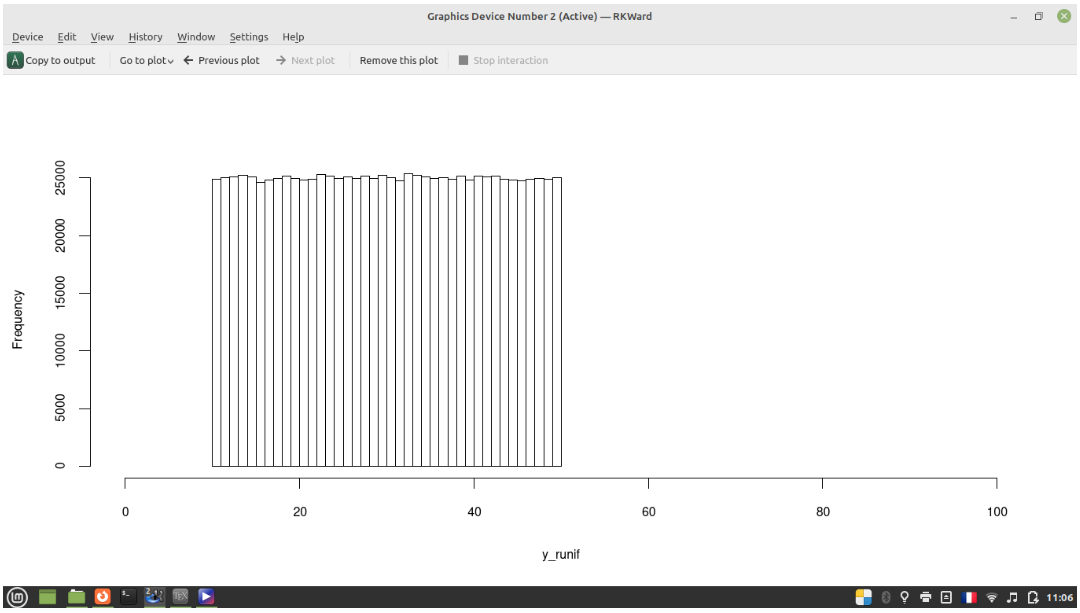

FIGURE 1.17: runif Function plot.

1.2.2 Exponential distribution

Another useful continuous distribution is the exponential distribution, which has the following probability density function:

$$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

This family of distributions is characterized by a single parameter λ, which is called the rate. Intuitively, λ can be thought of as the instantaneous "failure rate" of a "device" at any time t, given that the device has survived up to t.

The exponential distribution is typically used to model time intervals between "random events". . .

Examples:

- The length of time between telephone calls.
- The length of time between arrivals at a service station.
- The life time of electronic components, i.e., an inter failure time.

An important fact is that when times between random "events" follow the exponential distribution with rate λ, then the total number of events in a time period of length t follows the Poisson distribution with parameter λt. If a random variable X is exponentially distributed with rate λ, then it can be shown that :

Example 1: Exponential Density in R (dexp Function)

```
# Specify x-values for exp function
x_dexp <- seq(0, 1, by = 0.02)

# Apply exp function
y_dexp <- dexp(x_dexp, rate = 5)

# Plot dexp values
plot(y_dexp)
```

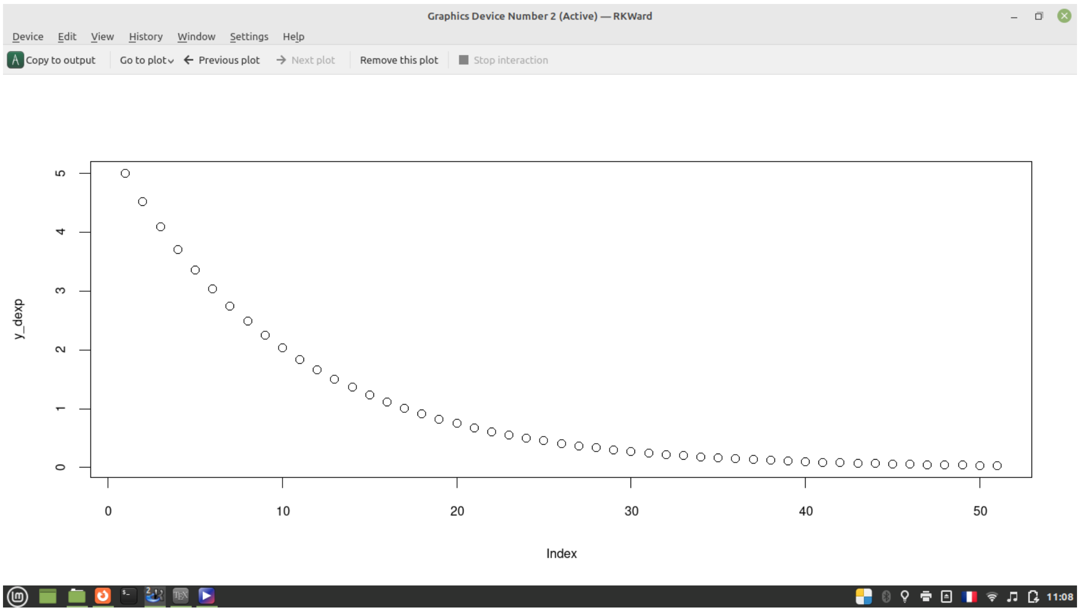

FIGURE 1.18: dexp Function plot.

Example 2: Exponential Cumulative Distribution Function (pexp Function)

```
# Specify x-values for pexp function
x_pexp <- seq(0, 1, by = 0.02)

# Apply pexp function
y_pexp <- pexp(x_pexp, rate = 5)

# Plot pexp values
plot(y_pexp)
```

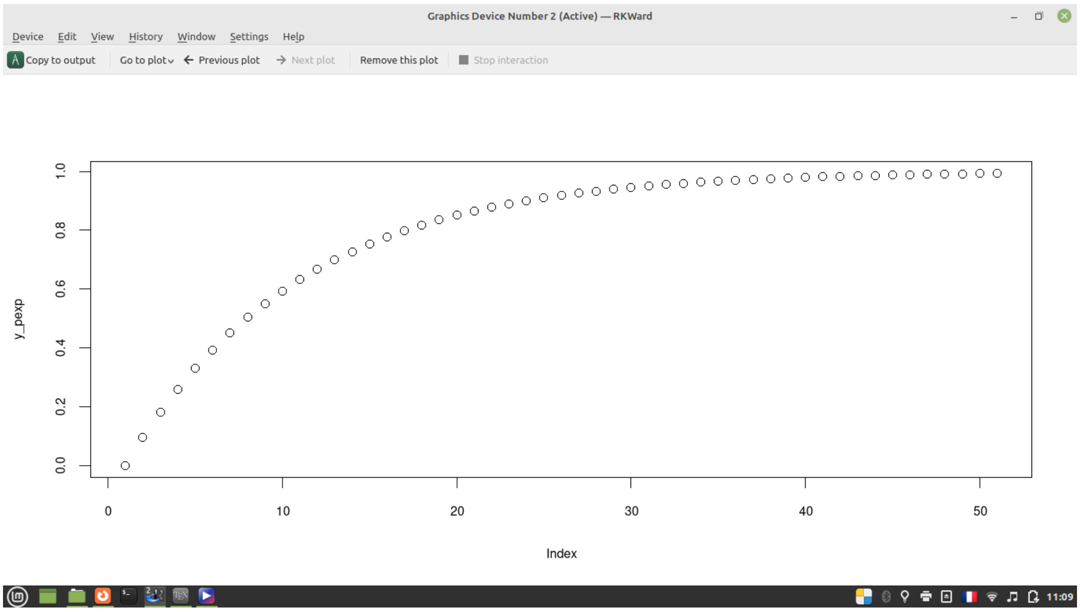

FIGURE 1.19: pexp Function plot.

Example 3: Exponential Quantile Function (qexp Function)

```
# Specify x-values for qexp function
x_qexp <- seq(0, 1, by = 0.02)

# Apply qexp function
y_qexp <- qexp(x_qexp, rate = 5)

# Plot qexp values
plot(y_qexp)
```

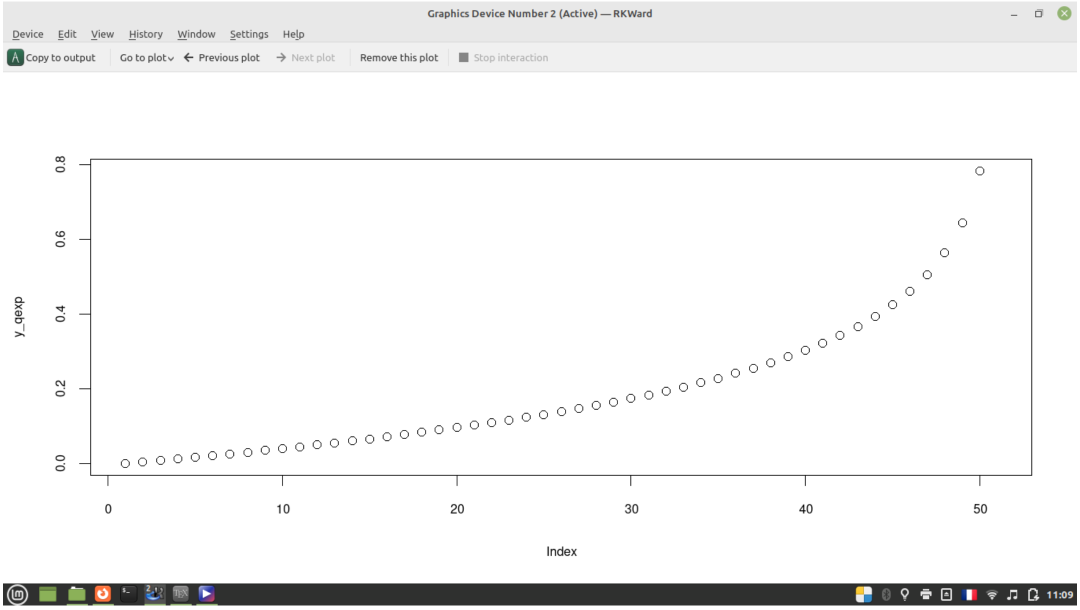

FIGURE 1.20: qexp Function plot.

Example 4: Random Number Generation (rexp Function)

```
# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <- 10000

# Draw N exp distributed values
y_rexp <- rexp(N, rate = 5)

# Print values to RStudio console
y_rexp

# Plot of randomly drawn exp density
hist(y_rexp, breaks = 100, main = "")
```

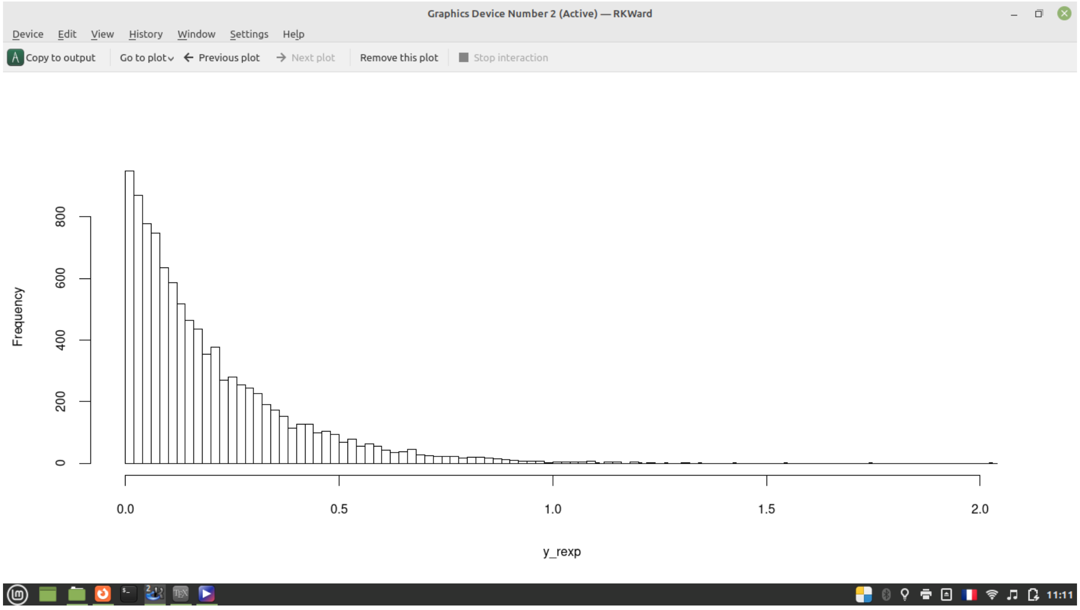

FIGURE 1.21: rexp Function plot.

1.2.3 Normal distribution

The normal distribution is the most important distribution in statistics, since it arises naturally in numerous applications. The key reason is that large sums of (small) random variables often turn out to be normally distributed. A random variable X is said to have the normal distribution with parameters μ and σ if its density function is given by:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)\right\}$$

Example 1: Normally Distributed Density (dnorm Function)

```
# Specify x-values for dnorm function
x_dnorm <- seq(- 5, 5, by = 0.05)

# Apply dnorm function
y_dnorm <- dnorm(x_dnorm)

# Plot dnorm values
plot(y_dnorm)
```

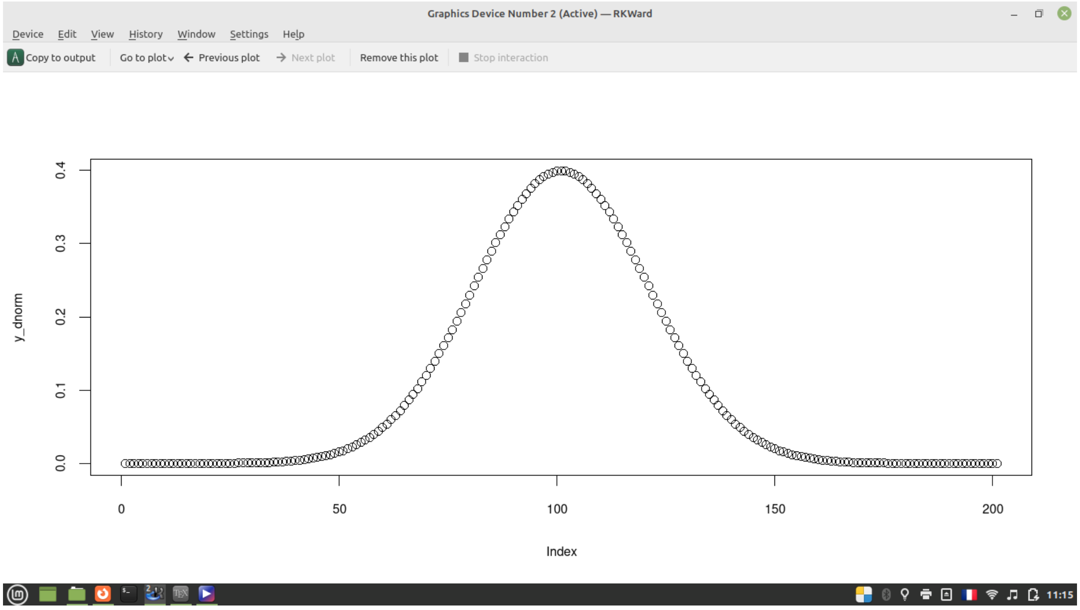

FIGURE 1.22: dnorm Function plot.

Example 2: Distribution Function (pnorm Function)

```
# Specify x-values for pnorm function
x_pnorm <- seq(- 5, 5, by = 0.05)

# Apply pnorm function
y_pnorm <- pnorm(x_pnorm)

# Plot pnorm values
plot(y_pnorm)
```

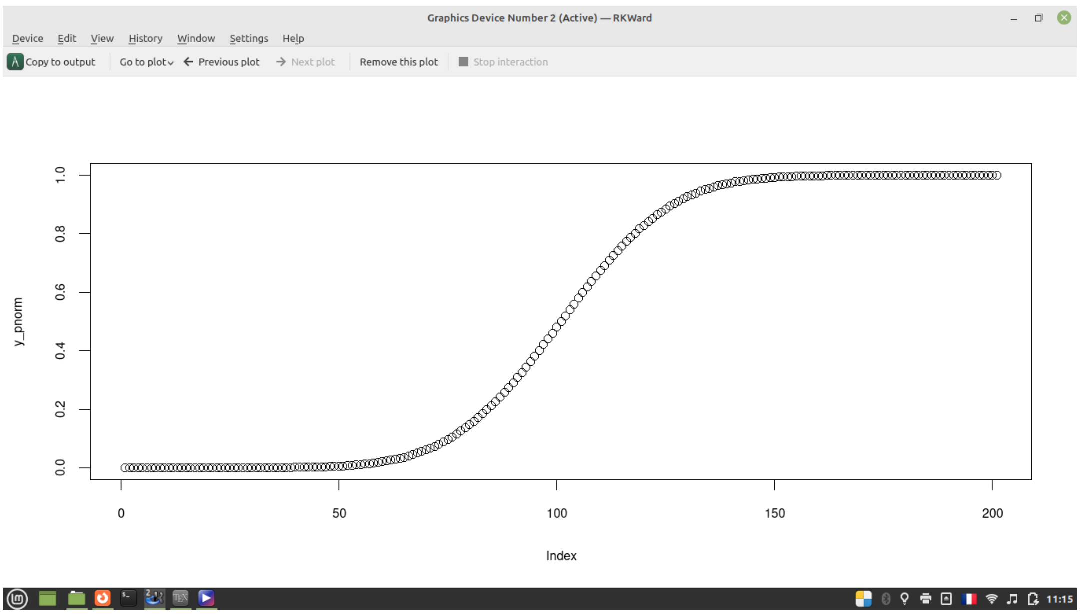

FIGURE 1.23: pnorm Function plot.

Example 3: Quantile Function (qnorm Function)

```
# Specify x-values for qnorm function
x_qnorm <- seq(0, 1, by = 0.005)

# Apply qnorm function
y_qnorm <- qnorm(x_qnorm)

# Plot qnorm values
plot(y_qnorm)
```

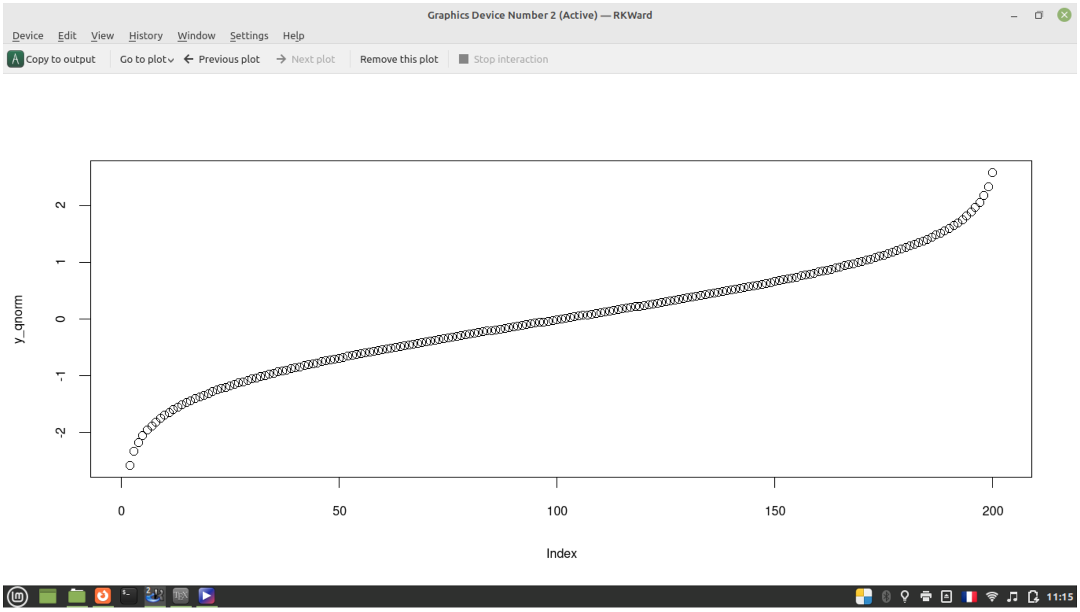

FIGURE 1.24: qnorm Function plot.

Example 4: Random Number Generation (rnorm Function)

```
# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <- 10000

# Draw N normally distributed values
y_rnorm <- rnorm(N)

# Print values to RStudio console     -1.234715493 -1.252833873
    -0.254778031 -1.526646627  1.097114685  2.488744223  0.779480260
      0.188375005 -1.026445945...
y_rnorm

# Plot pnorm values
plot(y_rnorm)

```

```
# Plot density of pnorm values
plot(density(y_rnorm))
```

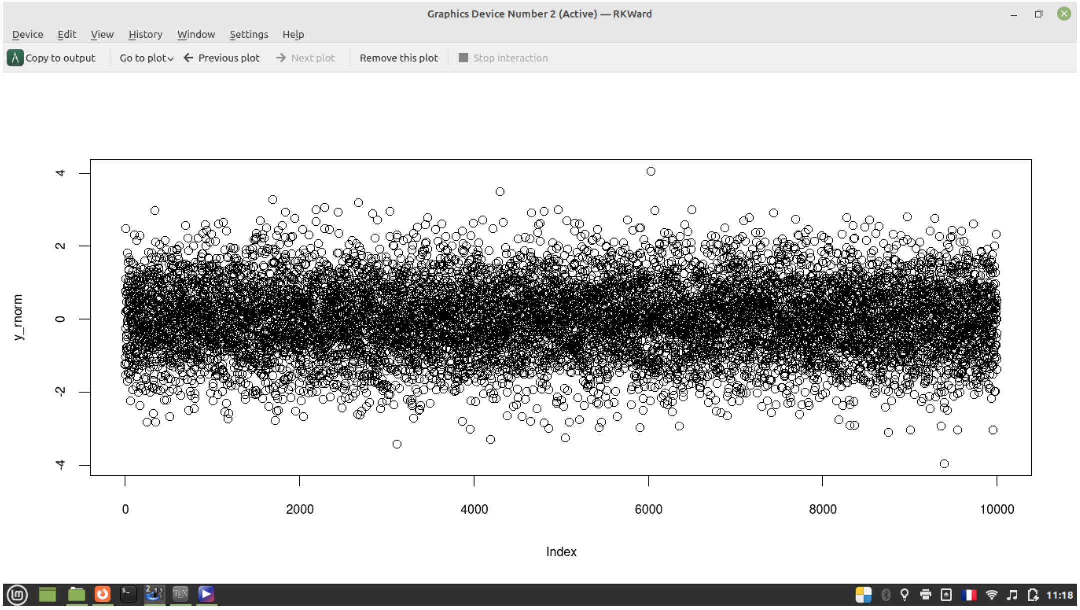

FIGURE 1.25: rnorm Function plot.

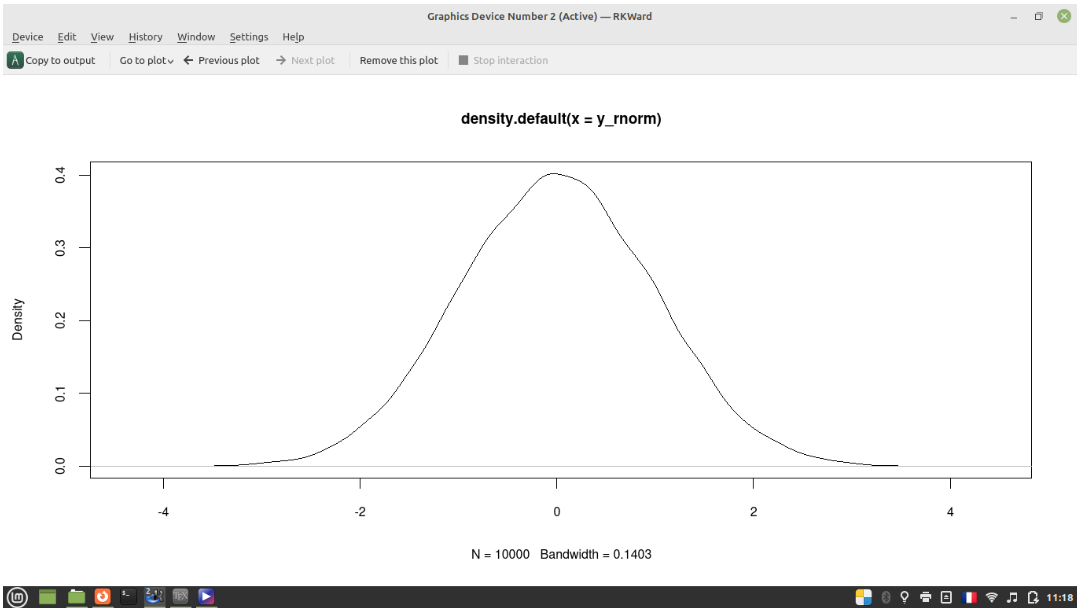

FIGURE 1.26: rnorm density plot.

Example 5: Modify Mean & Standard Deviation

```
# Modify mean
y_rnorm2 <- rnorm(N, mean = 2)

# Modify standard deviation
y_rnorm3 <- rnorm(N, mean = 2, sd = 3)
```

```
# Plot default density
plot(density(y_rnorm),
     xlim = c(- 10, 10),
     main = "Normal Distribution in R")
lines(density(y_rnorm2), col = "coral2")            # Plot density with higher mean
lines(density(y_rnorm3), col = "green3")            # Plot density with higher sd
legend("topleft",                                   # Add legend to density
       legend = c("Mean = 0; SD = 1",
                  "Mean = 2; SD = 1",
                  "Mean = 2; SD = 3"),
       col = c("black", "coral2", "green3"),
       lty = 1)
```

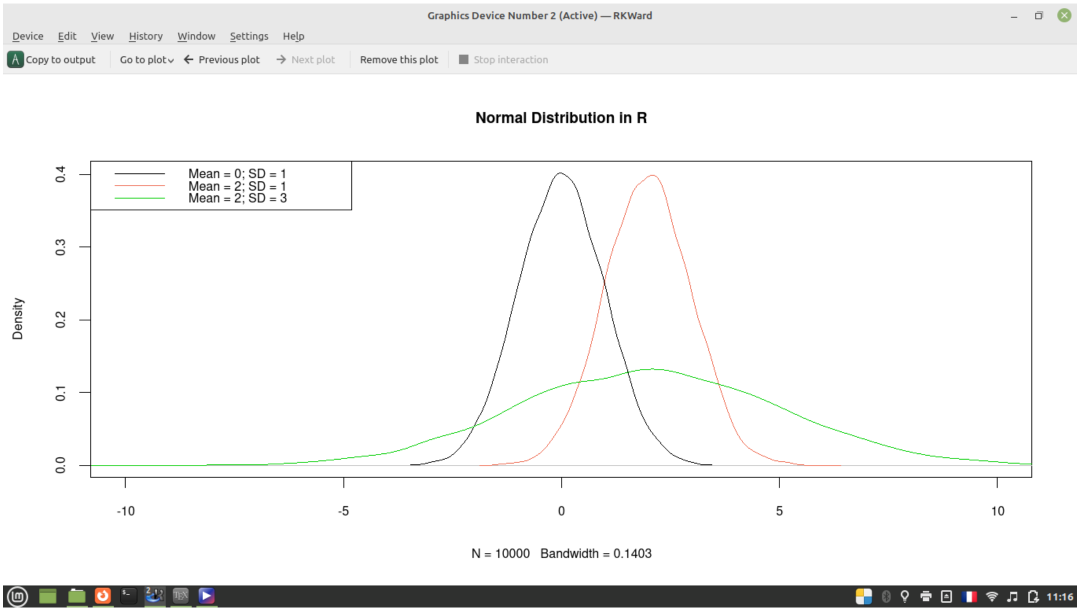

FIGURE 1.27: Mean and Standard Deviation plot.

Appendix A

Appendix A

A.1 R code

```
#
#
#     Bernoulli Distribution in R
#
#





# Example 1: Bernoulli Probability Density Function (dbern Function
  )


# Install Rlab package
install.packages("Rlab")

# Load Rlab package.skeleton
library("Rlab")

# Specify x-values for dbern function
x_dbern <- seq(0, 10, by = 1)

# Apply dbern function
y_dbern <- dbern(x_dbern, prob = 0.7)

# Plot dbern values
plot(y_dbern, type = "o")





# Example 2: Bernoulli Cumulative Distribution Function (pbern
  Function)


# Specify x-values for pbern function
x_pbern <- seq(0, 10, by = 1)

# Apply pbern function
y_pbern <- pbern(x_pbern, prob = 0.7)
```

```

# Plot pbern values
plot(y_pbern, type = "o")





# Example 3: Bernoulli Quantile Function (qbern Function)


# Specify x-values for qbern function
x_qbern <- seq(0, 1, by = 0.1)

# Apply qbern function
y_qbern <- qbern(x_qbern, prob = 0.7)

# Plot qbern values
plot(y_qbern, type = "o")







# Example 4: Generating Random Numbers (rbern Function)


# Set seed for reproducibility
set.seed(98989)

# Specify sample size
N <- 10000

# Draw N random values
y_rbern <- rbern(N, prob = 0.7)

# Print values to RStudio console
y_rbern

# Plot of randomly drawn density
hist(y_rbern,
     breaks = 5,
     main = "")











#
#
```

```
#     Binomial Distribution in R
#
#

# Example 1: Binomial Density in R (dbinom Function)

# Specify x-values for binom function
x_dbinom <- seq(0, 100, by = 1)

# Apply dbinom function
y_dbinom <- dbinom(x_dbinom, size = 100, prob = 0.5)

# Plot dbinom values
plot(y_dbinom)

# Example 2: Binomial Cumulative Distribution Function (pbinom
  Function)

# Specify x-values for pbinom function
x_pbinom <- seq(0, 100, by = 1)

# Apply pbinom function
y_pbinom <- pbinom(x_pbinom, size = 100, prob = 0.5)

# Plot pbinom values
plot(y_pbinom)
```

```
# Example 3: Binomial Quantile Function (qbinom Function)


# Specify x-values for qbinom function
x_qbinom <- seq(0, 1, by = 0.01)

# Apply qbinom function
y_qbinom <- qbinom(x_qbinom, size = 100, prob = 0.5)

# Plot qbinom values
plot(y_qbinom)










# Example 4: Simulation of Random Numbers (rbinom Function)


# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <- 10000

# Draw N binomially distributed values
y_rbinom <- rbinom(N, size = 100, prob = 0.5)

# Print values to RStudio console  45 44 55 43 35 47 56 52 49 51 47
    50 51 54 53 48 57 55 51...
y_rbinom

# Plot of randomly drawn binomial density
hist(y_rbinom,
     breaks = 100,
     main = "")









#
#
#     Poisson Distribution in R
#
#
```

```
# Example 1: Poisson Density in R (dpois Function)

# Specify x-values for dpois function
x_dpois <- seq(- 5, 30, by = 1)

# Apply dpois function
y_dpois <- dpois(x_dpois, lambda = 10)

# Plot dpois values
plot(y_dpois)

# Example 2: Poisson Distribution Function (ppois Function)

# Specify x-values for ppois function
x_ppois <- seq(- 5, 30, by = 1)

# Apply ppois function
y_ppois <- ppois(x_ppois, lambda = 10)

# Plot ppois values
plot(y_ppois)

# Example 3: Poisson Quantile Function (qpois Function)

# Specify x-values for qpois function
x_qpois <- seq(0, 1, by = 0.005)

# Apply qpois function
y_qpois <- qpois(x_qpois, lambda = 10)

# Plot qpois values
plot(y_qpois)
```

```
# Example 4: Random Number Generation (rpois Function)



# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <-10000

# Draw N poisson distributed values
y_rpois <- rpois(N, lambda = 10)

# Print values to RStudio console     6 14  8 16  6 12 10  6  7 11
     7 12 10 16  7  7  7 19 13
y_rpois

# Plot histogram of rpois values
hist(y_rpois,
     breaks = 100,
     main = "Poisson Distribution in R")












#
#
#    Continuous Uniform Distribution in R (4 Examples) | dunif,
    punif, qunif & runif Functions
#
#






# Example 1: Uniform Probability Density Function (dunif Function)


# Specify x-values for dunif function
x_dunif <- seq(0, 100, by = 1)

# Apply dunif function
y_dunif <- dunif(x_dunif, min = 10, max = 50)
```

```
# Plot dunif values
plot(y_dunif, type = "o")







# Example 2: Uniform Cumulative Distribution Function (punif
    Function)



# Specify x-values for punif function
x_punif <- seq(0, 100, by = 1)

# Apply punif function
y_punif <- punif(x_punif, min = 10, max = 50)

# Plot punif values
plot(y_punif, type = "o")






# Example 3: Uniform Quantile Function (qunif Function)




# Specify x-values for qunif function
x_qunif <- seq(0, 1, by = 0.01)

# Apply qunif function
y_qunif <- qunif(x_qunif, min = 10, max = 50)

# Plot qunif values
plot(y_qunif, type = "o")








# Example 4: Generating Random Numbers (runif Function)



# Set seed for reproducibility
set.seed(91929)
```

```
# Specify sample size
N <- 1000000

# Draw N uniformly distributed values
y_runif <- runif(N, min = 10, max = 50)

# Print values to RStudio console   27.98052 49.43937 24.66723
    39.36479 36.84591 40.94262 25.38942 35.59081...
y_runif

# Plot of randomly drawn uniformly density
hist(y_runif,
     breaks = 50,
     main = "",
     xlim = c(0, 100))








#
#
#     Exponential Distribution in R (4 Examples) | dexp, pexp, qexp
     & rexp Functions
#
#






# Example 1: Exponential Density in R (dexp Function)



# Specify x-values for exp function
x_dexp <- seq(0, 1, by = 0.02)

# Apply exp function
y_dexp <- dexp(x_dexp, rate = 5)

# Plot dexp values
plot(y_dexp)






# Example 2: Exponential Cumulative Distribution Function (pexp
    Function)


# Specify x-values for pexp function
x_pexp <- seq(0, 1, by = 0.02)
```

```
# Apply pexp function
y_pexp <- pexp(x_pexp, rate = 5)

# Plot pexp values
plot(y_pexp)








# Example 3: Exponential Quantile Function (qexp Function)



# Specify x-values for qexp function
x_qexp <- seq(0, 1, by = 0.02)

# Apply qexp function
y_qexp <- qexp(x_qexp, rate = 5)

# Plot qexp values
plot(y_qexp)







# Example 4: Random Number Generation (rexp Function)



# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <- 10000

# Draw N exp distributed values
y_rexp <- rexp(N, rate = 5)

# Print values to RStudio console
y_rexp

# Plot of randomly drawn exp density
hist(y_rexp, breaks = 100, main = "")
```

```
#
#
#     Normal Distribution in R (5 Examples) | dnorm, pnorm, qnorm & rnorm Functions
#
#






# Example 1: Normally Distributed Density (dnorm Function)



# Specify x-values for dnorm function
x_dnorm <- seq(- 5, 5, by = 0.05)

# Apply dnorm function
y_dnorm <- dnorm(x_dnorm)

# Plot dnorm values
plot(y_dnorm)







# Example 2: Distribution Function (pnorm Function)



# Specify x-values for pnorm function
x_pnorm <- seq(- 5, 5, by = 0.05)

# Apply pnorm function
y_pnorm <- pnorm(x_pnorm)

# Plot pnorm values
plot(y_pnorm)








# Example 3: Quantile Function (qnorm Function)




```

```
# Specify x-values for qnorm function
x_qnorm <- seq(0, 1, by = 0.005)

# Apply qnorm function
y_qnorm <- qnorm(x_qnorm)

# Plot qnorm values
plot(y_qnorm)








# Example 4: Random Number Generation (rnorm Function)



# Set seed for reproducibility
set.seed(13579)

# Specify sample size
N <- 10000

# Draw N normally distributed values
y_rnorm <- rnorm(N)

# Print values to RStudio console     -1.234715493 -1.252833873
    -0.254778031 -1.526646627  1.097114685  2.488744223  0.779480260
      0.188375005 -1.026445945...
y_rnorm

# Plot pnorm values
plot(y_rnorm)

# Plot density of pnorm values
plot(density(y_rnorm))







# Example 5: Modify Mean & Standard Deviation



# Modify mean
y_rnorm2 <- rnorm(N, mean = 2)

# Modify standard deviation
y_rnorm3 <- rnorm(N, mean = 2, sd = 3)

# Plot default density
```

```
plot(density(y_rnorm),
     xlim = c(- 10, 10),
     main = "Normal Distribution in R")
lines(density(y_rnorm2), col = "coral2")              # Plot density
    with higher mean
lines(density(y_rnorm3), col = "green3")              # Plot density
    with higher sd
legend("topleft",                                     # Add legend
   to density
       legend = c("Mean = 0; SD = 1",
                  "Mean = 2; SD = 1",
                  "Mean = 2; SD = 3"),
       col = c("black", "coral2", "green3"),
       lty = 1)
```

Bibliography

[1] Joseph K. Blitzstein and Jessica Hwang. *Introduction to probability*. Second edition. Boca Raton: CRC Press, 2019. ISBN: 9780429766732 9780429428357.

[2] Richard Cotton. *Learning R*. First Edition. OCLC: ocn830813799. Beijing ; Sebastopol, CA: O'Reilly, 2013. ISBN: 9781449357108.

[3] Garrett Grolemund. *Hands-on programming with R*. First edition. OCLC: ocn887746093. Sebastopol, CA: O'Reilly, 2014. ISBN: 9781449359010.

[4] Norman S. Matloff. *Probability and statistics for data science: math + R + data*. Boca Raton: CRC Press, Taylor & Francis Group, 2019. ISBN: 9780367260934 9781138393295.

[5] William Mendenhall, Robert J. Beaver, and Barbara M. Beaver. *Introduction to probability and statistics*. 14th ed./Student edition. Boston, MA, USA: Brooks/-Cole, Cengage Learning, 2013. ISBN: 9781133103752.

[6] Seema Singh. *Probability Distributions: Discrete and Continuous*. en. Apr. 2019. URL: `https://medium.com/@seema.singh/probability-distributions-discrete-and-continuous-7a94ede66dc0` (visited on 02/13/2022).