

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 DEGREE 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

<b>1</b>	<b>Discrete and Continuous Probability Distributions in R</b>	<b>1</b>
1.1	Discrete distributions	1
1.1.1	Bernoulli distribution	1
	Bernoulli Probability Density Function (dbern Function)	1
	Bernoulli Cumulative Distribution Function (pbern Function)	2
	Bernoulli Quantile Function (qbern Function)	3
	Generating Random Numbers (rbern Function)	4
1.1.2	Binomial distribution	4
	Example 1: Binomial Density in R (dbinom Function)	6
	Example 2: Binomial Cumulative Distribution Function (pbinom Function)	6
	Example 3: Binomial Quantile Function (qbinom Function)	7
	Example 4: Simulation of Random Numbers (rbinom Function)	8
1.1.3	Poisson distribution	8
	Example 1: Poisson Density in R (dpois Function)	9
	Example 2: Poisson Distribution Function (ppois Function)	10
	Example 3: Poisson Quantile Function (qpois Function)	11
	Example 4: Random Number Generation (rpois Function)	11
1.2	Continuous distributions	12
1.2.1	Uniform distribution	12
	Example 1: Uniform Probability Density Function (dunif Function)	13
	Example 2: Uniform Cumulative Distribution Function (punif Function)	13
	Example 3: Uniform Quantile Function (qunif Function)	14
	Example 4: Generating Random Numbers (runif Function)	15
1.2.2	Exponential distribution	15
	Example 1: Exponential Density in R (dexp Function)	16
	Example 2: Exponential Cumulative Distribution Function (pexp Function)	16
	Example 3: Exponential Quantile Function (qexp Function)	17
	Example 4: Random Number Generation (rexp Function)	18
1.2.3	Normal distribution	19
	Example 1: Normally Distributed Density (dnorm Function)	19
	Example 2: Distribution Function (pnorm Function)	20
	Example 3: Quantile Function (qnorm Function)	21
	Example 4: Random Number Generation (rnorm Function)	21
	Example 5: Modify Mean & Standard Deviation	22
<b>A</b>	<b>Appendix A</b>	<b>24</b>
A.1	R code	24

## 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.

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

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

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  $0 < p < 1$ .<sup>[1]</sup>

### Bernoulli Probability Density Function (dbern Function)

```

1 # Install Rlab package
2 install.packages("Rlab")
3
4 # Load Rlab package.skeleton
5 library("Rlab")
6
7 # Specify x-values for dbern function
8 x_dbern <- seq(0, 10, by = 1)

```

```
9  
10 # Apply dbern function  
11 y_dbern <- dbern(x_dbern, prob = 0.7)  
12  
13 # Plot dbern values  
14 plot(y_dbern, type = "o")
```

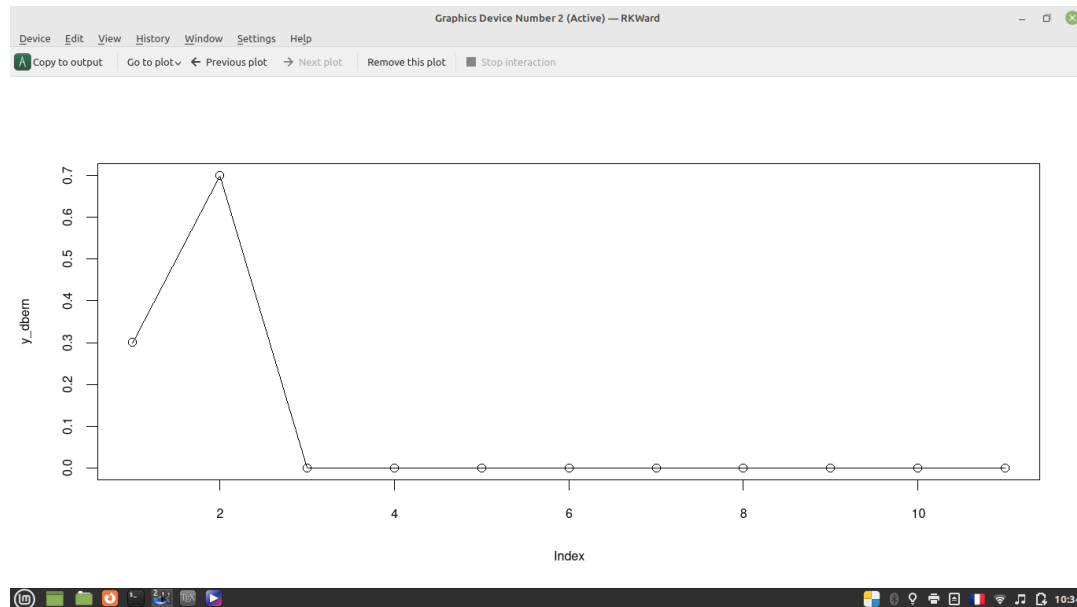


FIGURE 1.1: dbern Function plot.

### Bernoulli Cumulative Distribution Function (pbern Function)

```
15 # Specify x-values for pbern function  
16 x_pbern <- seq(0, 10, by = 1)  
17  
18 # Apply pbern function  
19 y_pbern <- pbern(x_pbern, prob = 0.7)  
20  
21 # Plot pbern values  
22 plot(y_pbern, type = "o")
```

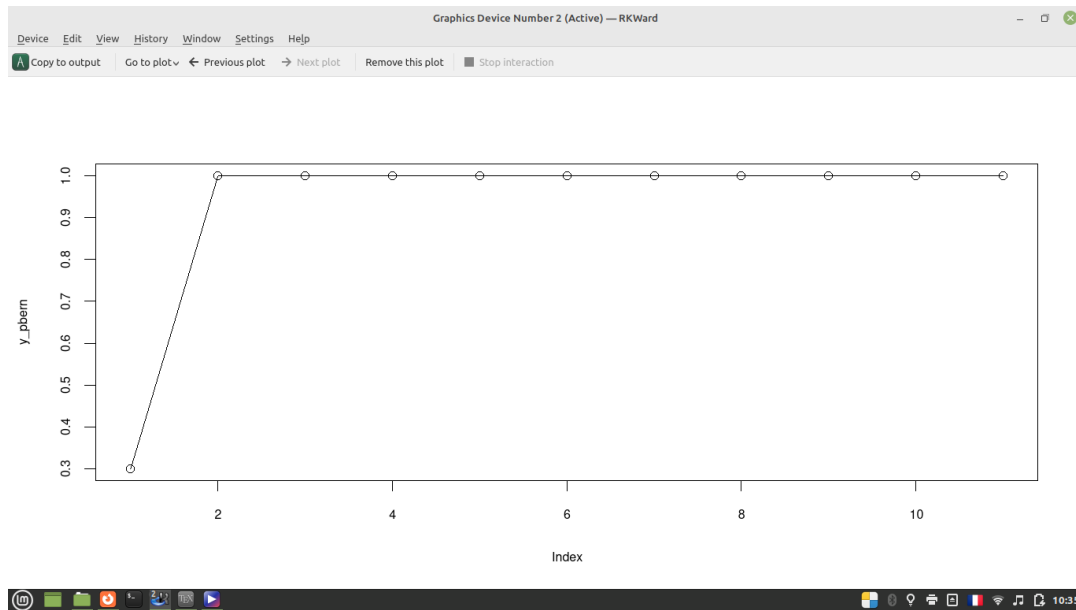


FIGURE 1.2: pbern Function plot.

### Bernoulli Quantile Function (qbern Function)

```
23 # Specify x-values for qbern function
24 x_qbern <- seq(0, 1, by = 0.1)
25
26 # Apply qbern function
27 y_qbern <- qbern(x_qbern, prob = 0.7)
28
29 # Plot qbern values
30 plot(y_qbern, type = "o")
```

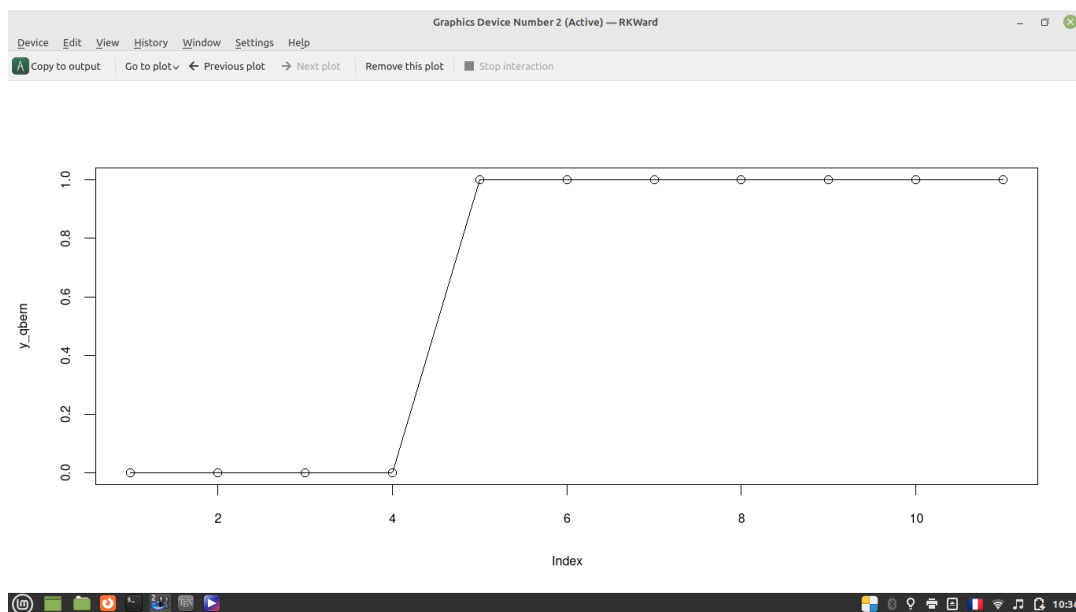


FIGURE 1.3: qbern Function plot.

**Generating Random Numbers (rbern Function)**

```

31 # Set seed for reproducibility
32 set.seed(98989)
33
34 # Specify sample size
35 N <- 10000
36
37 # Draw N random values
38 y_rbern <- rbern(N, prob = 0.7)
39
40 # Print values to RStudio console
41 y_rbern
42
43 # Plot of randomly drawn density
44 hist(y_rbern,
45       breaks = 5,
46       main = "")

```

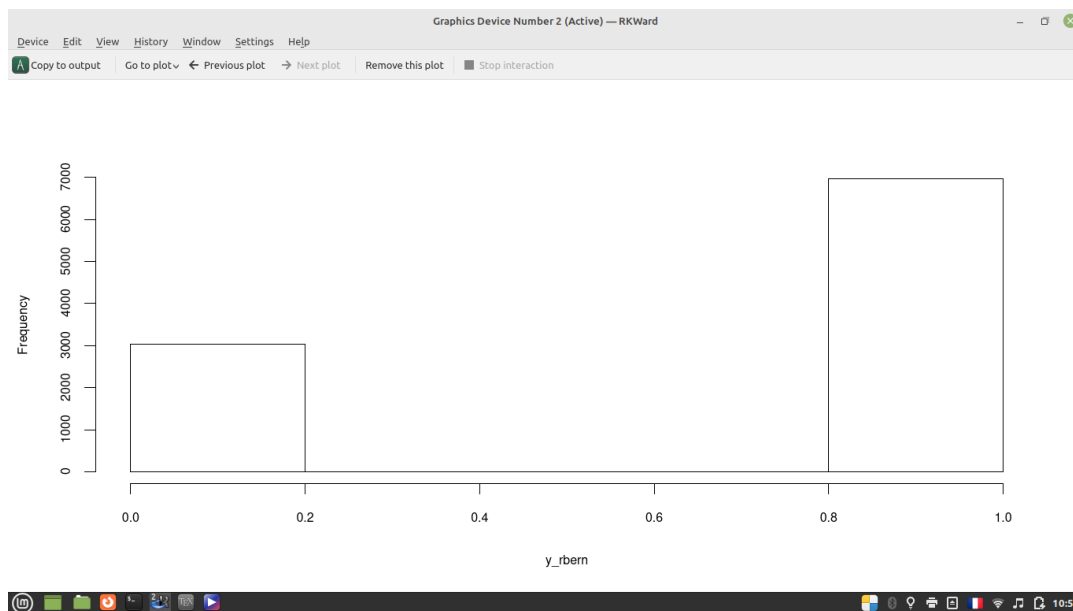


FIGURE 1.4: rbern Function plot.

**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  $n$  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  $p$  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  $x = 0, 1, 2, \dots, n$ .

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

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

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

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

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

```
47 # Specify x-values for binom function
48 x_dbinom <- seq(0, 100, by = 1)
49
50 # Apply dbinom function
51 y_dbinom <- dbinom(x_dbinom, size = 100, prob = 0.5)
52
53 # Plot dbinom values
54 plot(y_dbinom)
```

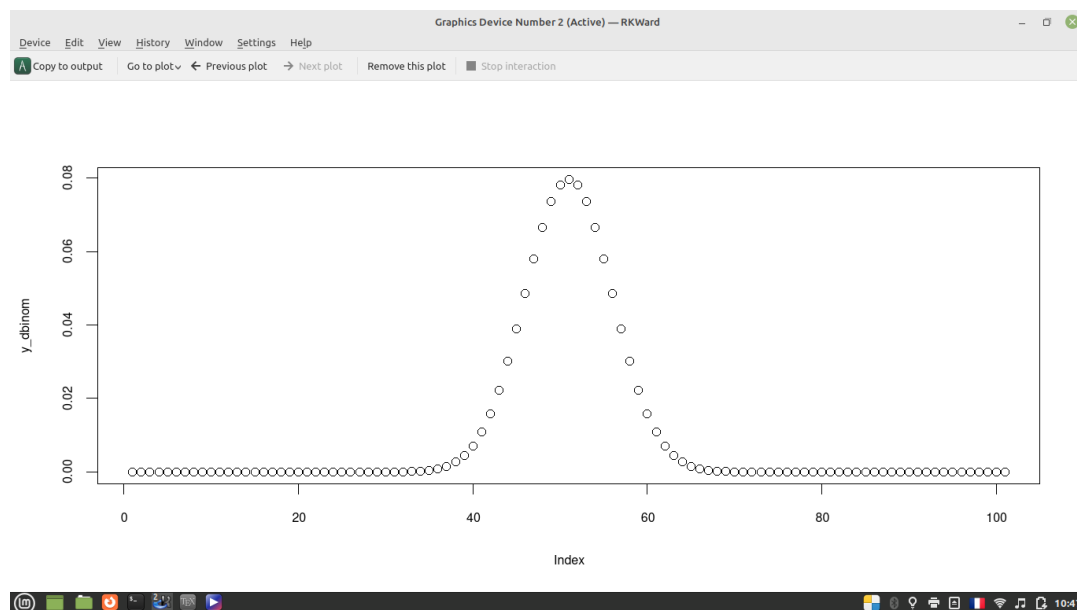


FIGURE 1.5: dbinom Function plot.

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

```
55 # Specify x-values for pbinom function
56 x_pbinom <- seq(0, 100, by = 1)
57
58 # Apply pbinom function
59 y_pbinom <- pbinom(x_pbinom, size = 100, prob = 0.5)
60
61 # Plot pbinom values
62 plot(y_pbinom)
```



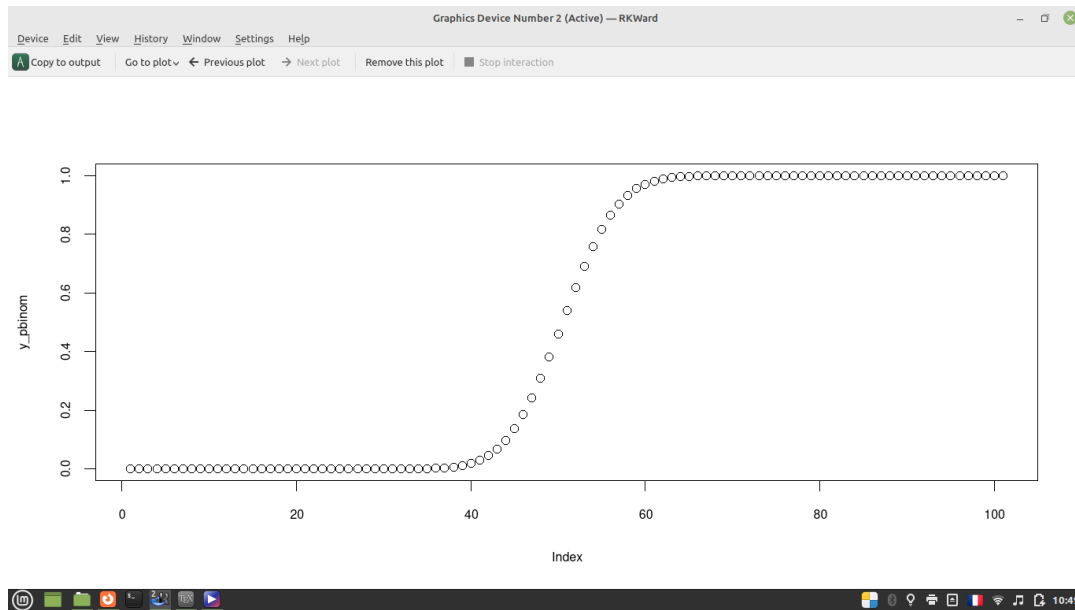


FIGURE 1.6: pbinom Function plot.

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

```
63 # Specify x-values for qbinom function
64 x_qbinom <- seq(0, 1, by = 0.01)
65
66 # Apply qbinom function
67 y_qbinom <- qbinom(x_qbinom, size = 100, prob = 0.5)
68
69 # Plot qbinom values
70 plot(y_qbinom)
```

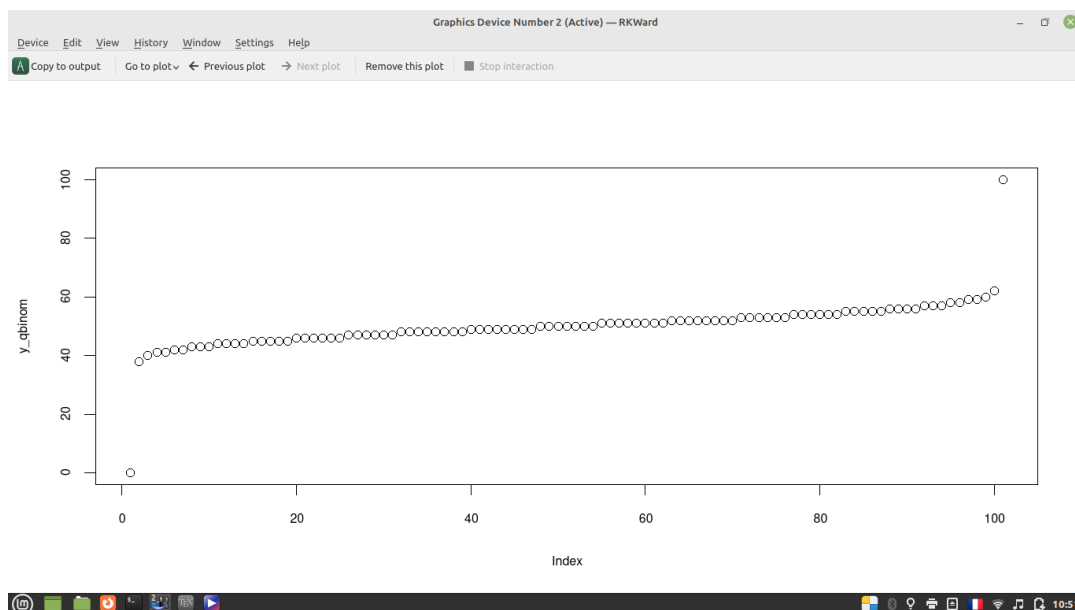


FIGURE 1.7: qbinom Function plot.

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

```

71 # Set seed for reproducibility
72 set.seed(13579)
73
74 # Specify sample size
75 N <- 10000
76
77 # Draw N binomially distributed values
78 y_rbinom <- rbinom(N, size = 100, prob = 0.5)
79
80 # Print values to RStudio console 45 44 55 43 35 47 56 52 49 51 47
81   50 51 54 53 48 57 55 51...
82
83 y_rbinom
84
85 # Plot of randomly drawn binomial density
86 hist(y_rbinom,
87       breaks = 100,
88       main = "")

```

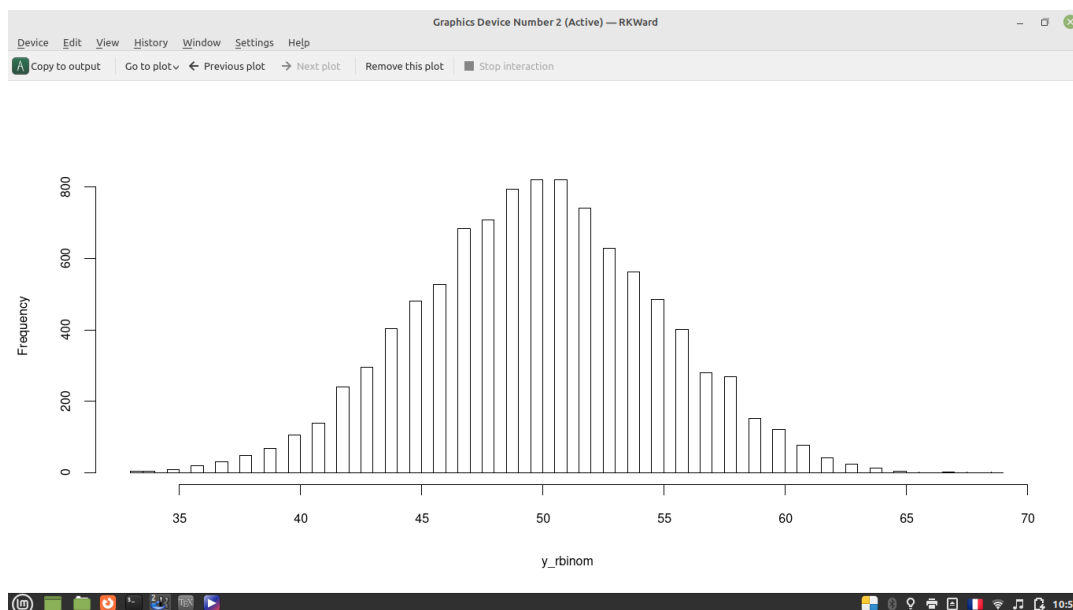


FIGURE 1.8: rbinom Function plot.

**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  $\mu$  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  $\mu$  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  $k = 0, 1, 2, 3, \dots$ . The mean and standard deviation of the Poisson random variable  $x$  are :

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

```

87 # Specify x-values for dpois function
88 x_dpois <- seq(- 5, 30, by = 1)
89
90 # Apply dpois function
91 y_dpois <- dpois(x_dpois, lambda = 10)
92
93 # Plot dpois values
94 plot(y_dpois)
```

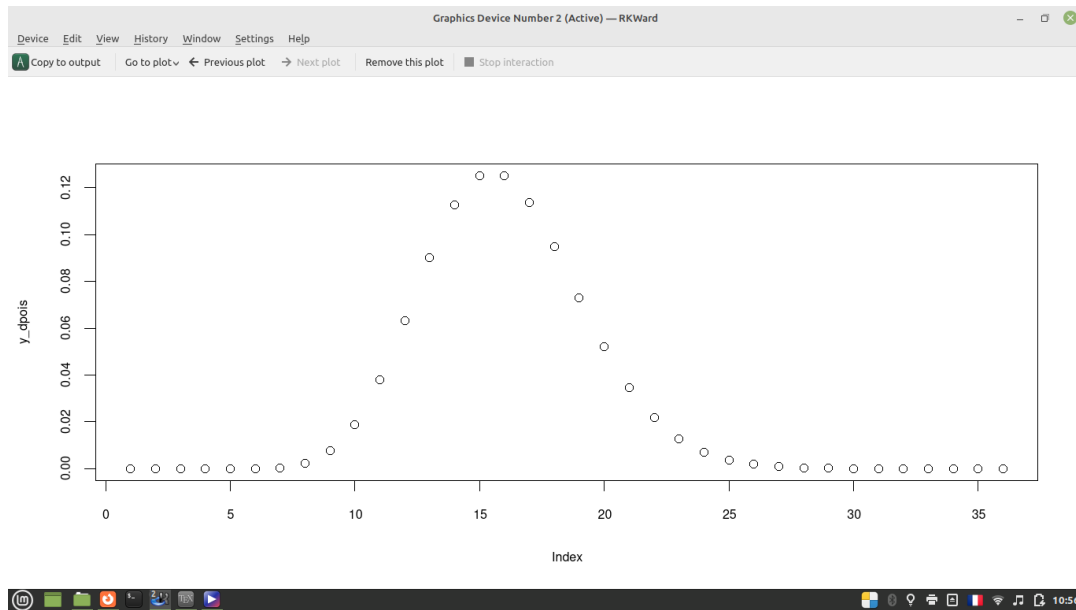


FIGURE 1.9: dpois Function plot.

### Example 2: Poisson Distribution Function (ppois Function)

```
95 # Specify x-values for ppois function
96 x_ppois <- seq(- 5, 30, by = 1)
97
98 # Apply ppois function
99 y_ppois <- ppois(x_ppois, lambda = 10)
100
101 # Plot ppois values
102 plot(y_ppois)
```

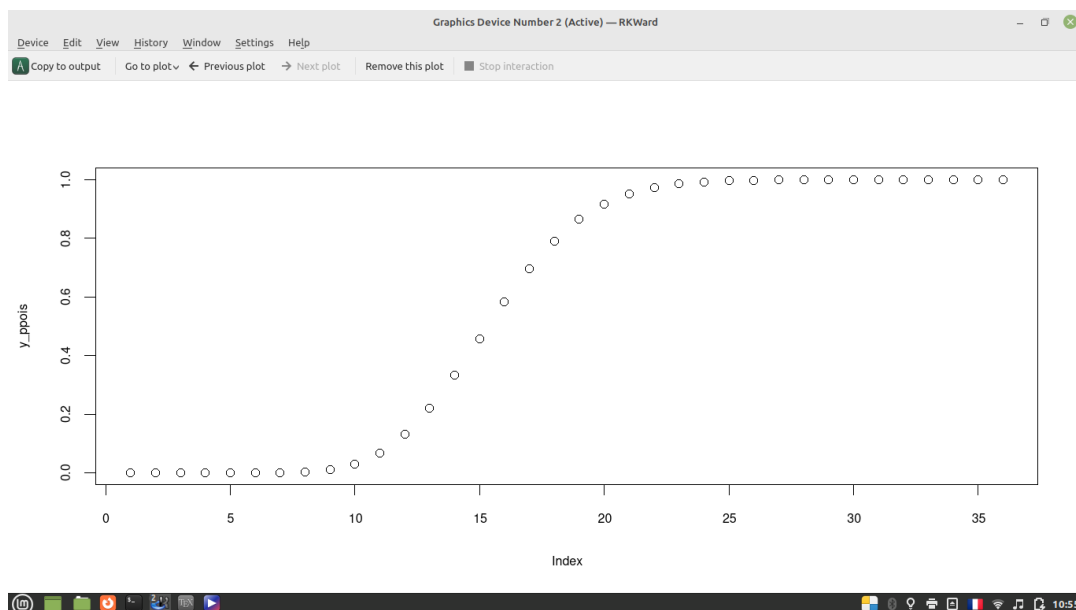


FIGURE 1.10: ppois Function plot.

**Example 3: Poisson Quantile Function (qpois Function)**

```

103 # Specify x-values for qpois function
104 x_qpois <- seq(0, 1, by = 0.005)
105
106 # Apply qpois function
107 y_qpois <- qpois(x_qpois, lambda = 10)
108
109 # Plot qpois values
110 plot(y_qpois)

```

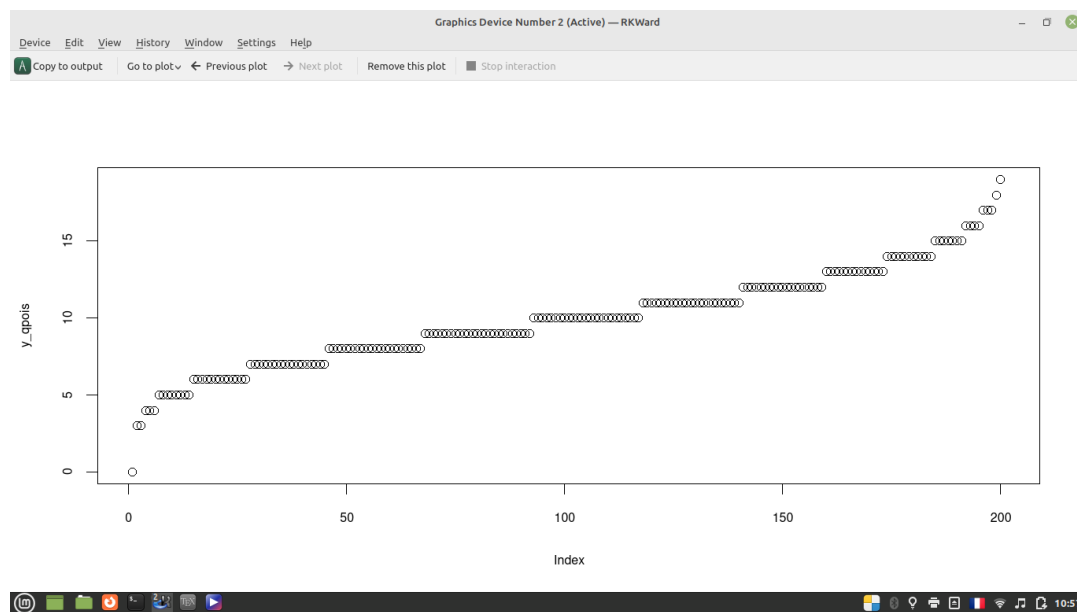


FIGURE 1.11: qpois Function plot.

**Example 4: Random Number Generation (rpois Function)**

```

111 # Set seed for reproducibility
112 set.seed(13579)
113
114 # Specify sample size
115 N <- 10000
116
117 # Draw N poisson distributed values
118 y_rpois <- rpois(N, lambda = 10)
119
120 # Print values to RStudio console      6 14  8 16  6 12 10  6  7 11
121                                     7 12 10 16  7  7  7 19 13
122 y_rpois
123
124 # Plot histogram of rpois values
125 hist(y_rpois,
126       breaks = 100,
127       main = "Poisson Distribution in R")

```

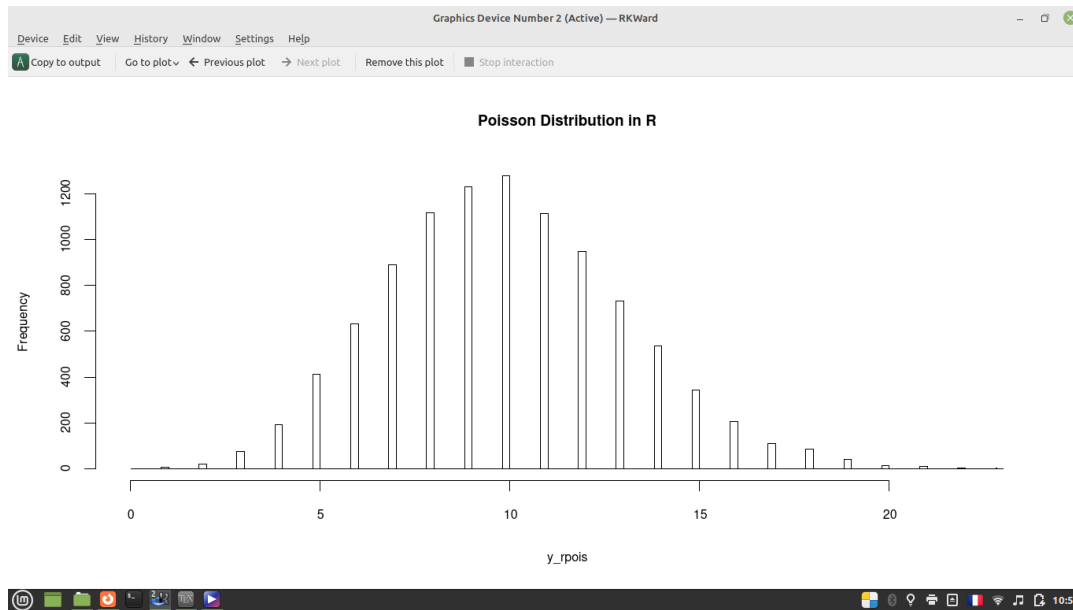


FIGURE 1.12: rpois Function plot.

## 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:

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

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

Visually, we have :

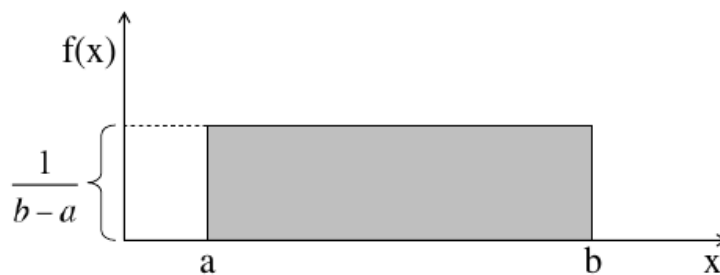


FIGURE 1.13: The uniform probability distribution.

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

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

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

```
129
130 # Apply dunif function
131 y_dunif <- dunif(x_dunif, min = 10, max = 50)
132
133 # Plot dunif values
134 plot(y_dunif, type = "o")
```

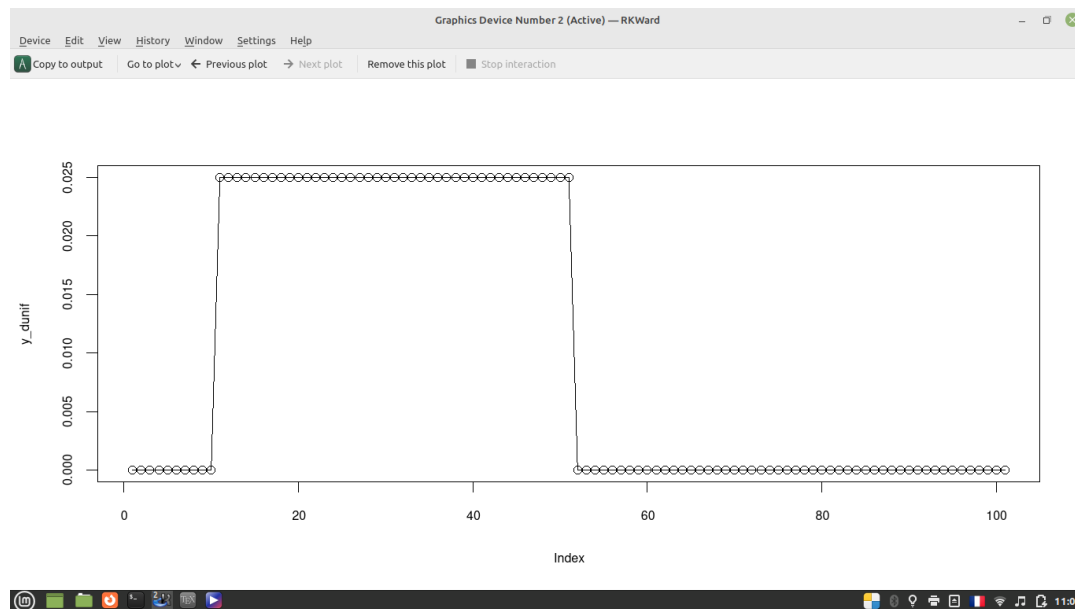


FIGURE 1.14: dunif Function plot.

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

```
135 # Specify x-values for punif function
136 x_punif <- seq(0, 100, by = 1)
137
138 # Apply punif function
139 y_punif <- punif(x_punif, min = 10, max = 50)
140
141 # Plot punif values
142 plot(y_punif, type = "o")
```

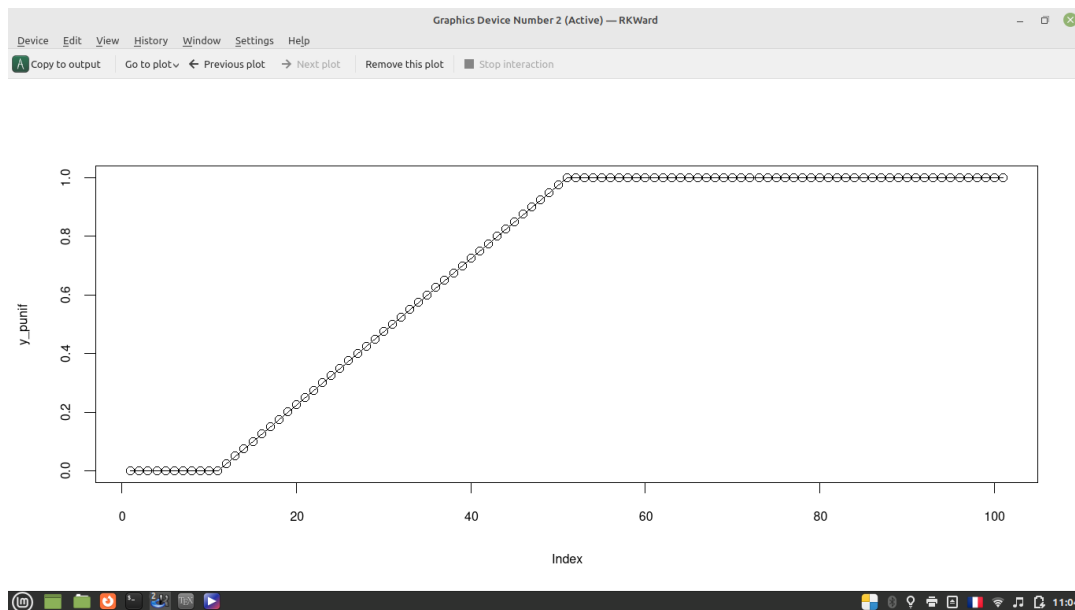


FIGURE 1.15: punif Function plot.

### Example 3: Uniform Quantile Function (qunif Function)

```
143 # Specify x-values for qunif function
144 x_qunif <- seq(0, 1, by = 0.01)
145
146 # Apply qunif function
147 y_qunif <- qunif(x_qunif, min = 10, max = 50)
148
149 # Plot qunif values
150 plot(y_qunif, type = "o")
```

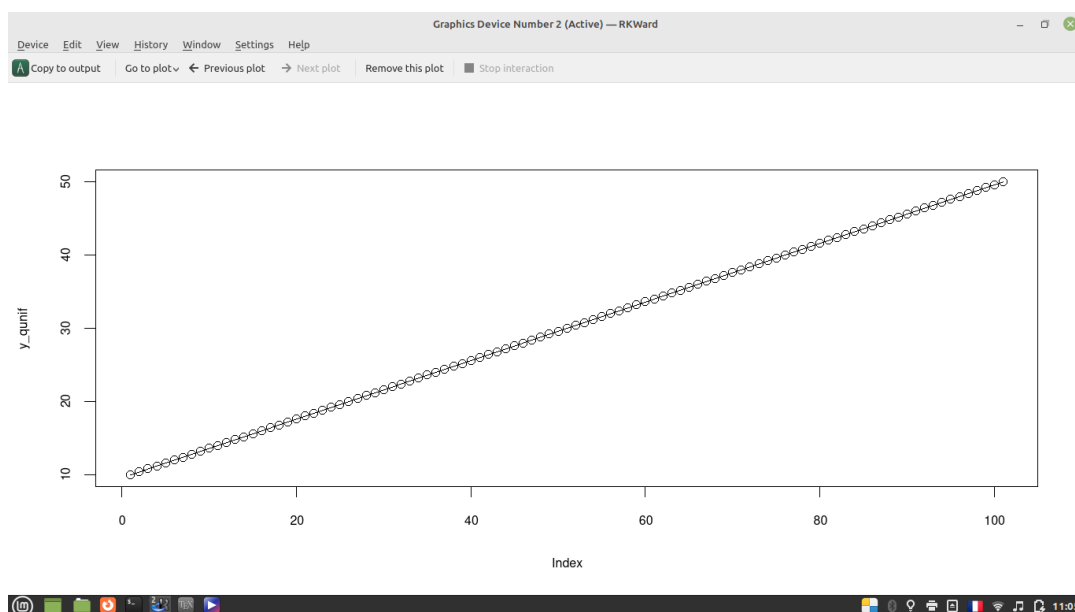


FIGURE 1.16: qunif Function plot.



**Example 4: Generating Random Numbers (runif Function)**

```

151 # Set seed for reproducibility
152 set.seed(91929)
153
154 # Specify sample size
155 N <- 1000000
156
157 # Draw N uniformly distributed values
158 y_runif <- runif(N, min = 10, max = 50)
159
160 # Print values to RStudio console    27.98052 49.43937 24.66723
161                                     39.36479 36.84591 40.94262 25.38942 35.59081...
162 y_runif
163
164 # Plot of randomly drawn uniformly density
165 hist(y_runif,
166       breaks = 50,
167       main = "",
168       xlim = c(0, 100))

```

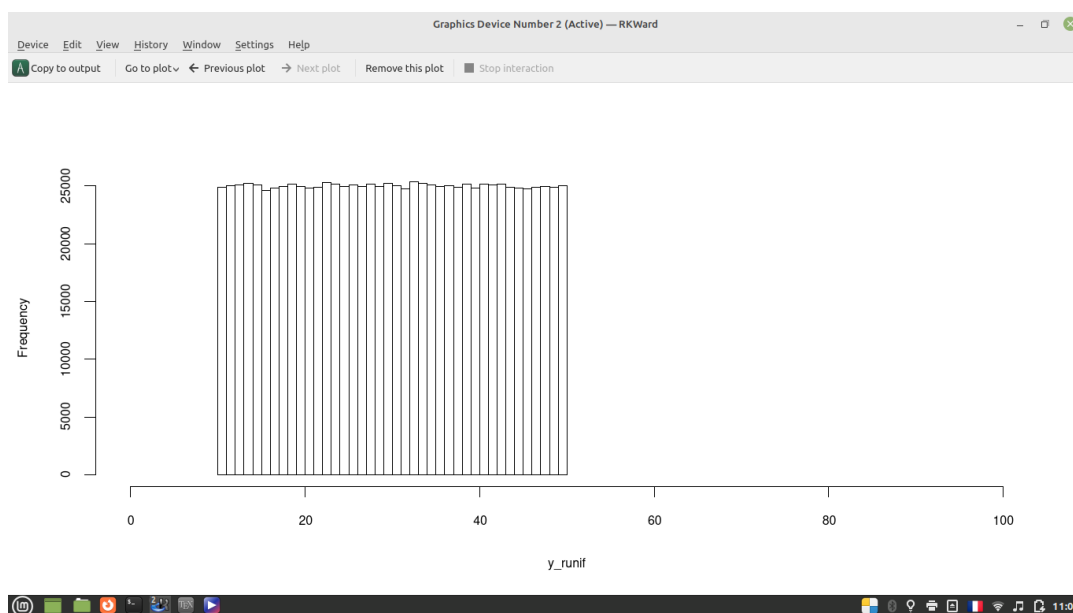


FIGURE 1.17: runif Function plot.

**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  $\lambda$ , which is called the rate. Intuitively,  $\lambda$  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  $\lambda$ , then the total number of events in a time period of length  $t$  follows the Poisson distribution with parameter  $\lambda t$ . If a random variable  $X$  is exponentially distributed with rate  $\lambda$ , then it can be shown that :

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

```

168 # Specify x-values for exp function
169 x_dexp <- seq(0, 1, by = 0.02)
170
171 # Apply exp function
172 y_dexp <- dexp(x_dexp, rate = 5)
173
174 # Plot dexp values
175 plot(y_dexp)

```

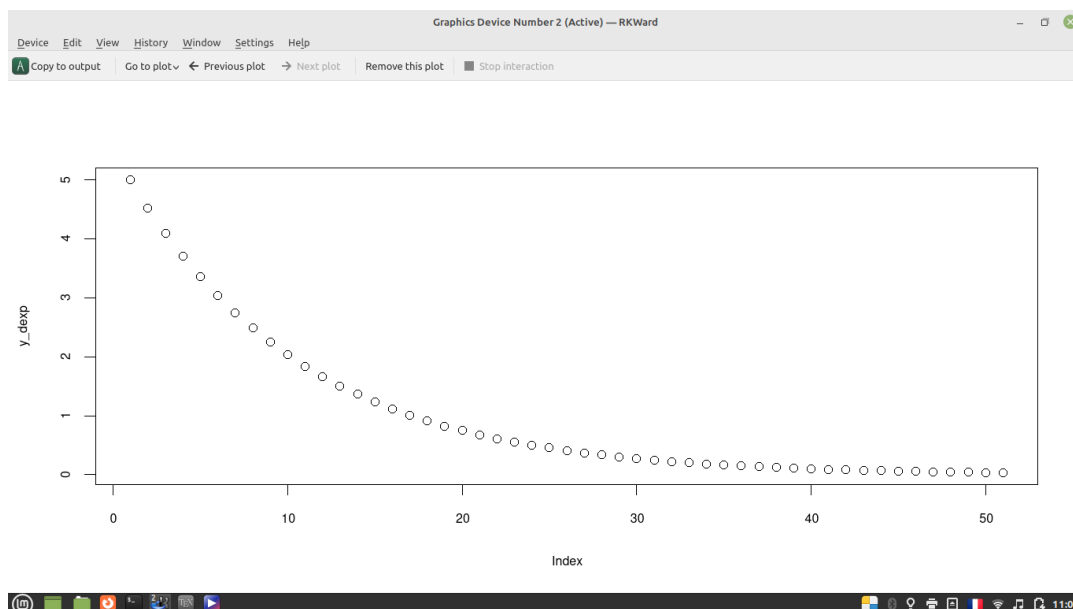


FIGURE 1.18: dexp Function plot.

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

```
176 # Specify x-values for pexp function
177 x_pexp <- seq(0, 1, by = 0.02)
178
179 # Apply pexp function
180 y_pexp <- pexp(x_pexp, rate = 5)
181
182 # Plot pexp values
183 plot(y_pexp)
```

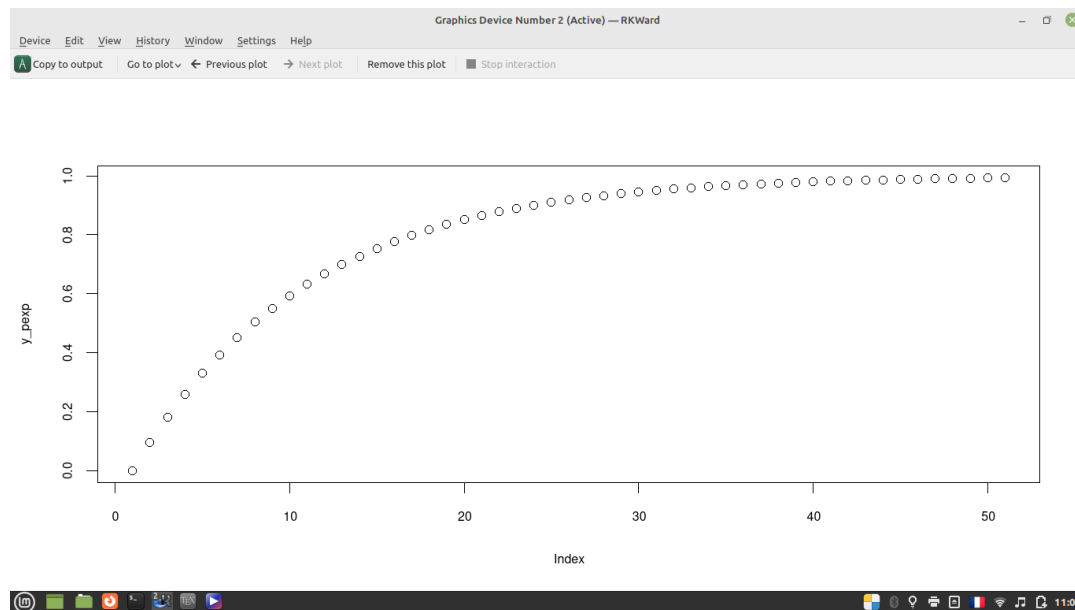


FIGURE 1.19: pexp Function plot.

### Example 3: Exponential Quantile Function (qexp Function)

```
184 # Specify x-values for qexp function
185 x_qexp <- seq(0, 1, by = 0.02)
186
187 # Apply qexp function
188 y_qexp <- qexp(x_qexp, rate = 5)
189
190 # Plot qexp values
191 plot(y_qexp)
```

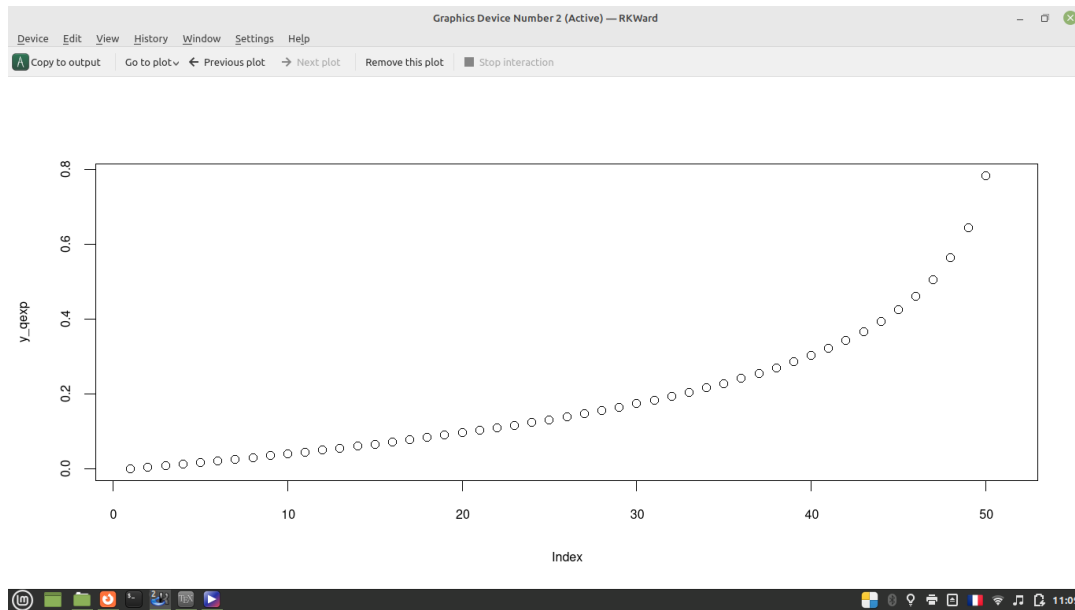


FIGURE 1.20: qexp Function plot.

#### Example 4: Random Number Generation (rexp Function)

```
192 # Set seed for reproducibility
193 set.seed(13579)
194
195 # Specify sample size
196 N <- 10000
197
198 # Draw N exp distributed values
199 y_rexp <- rexp(N, rate = 5)
200
201 # Print values to RStudio console
202 y_rexp
203
204 # Plot of randomly drawn exp density
205 hist(y_rexp, breaks = 100, main = "")
```

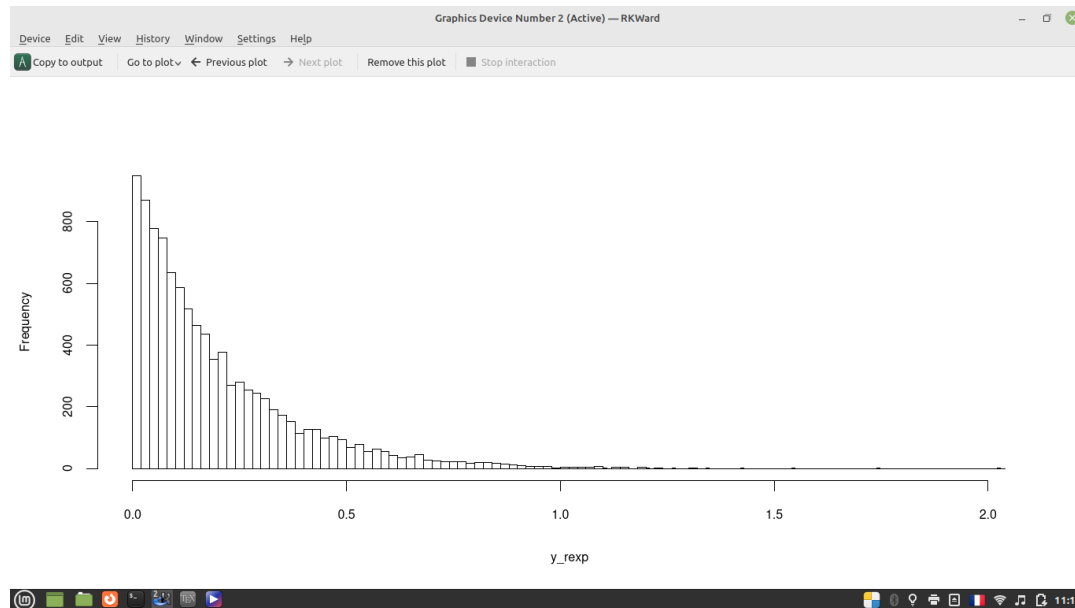


FIGURE 1.21: rexp Function plot.

### 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  $\mu$  and  $\sigma$  if its density function is given by:

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

#### Example 1: Normally Distributed Density (dnorm Function)

```

206 # Specify x-values for dnorm function
207 x_dnorm <- seq(- 5, 5, by = 0.05)
208
209 # Apply dnorm function
210 y_dnorm <- dnorm(x_dnorm)
211
212 # Plot dnorm values
213 plot(y_dnorm)

```

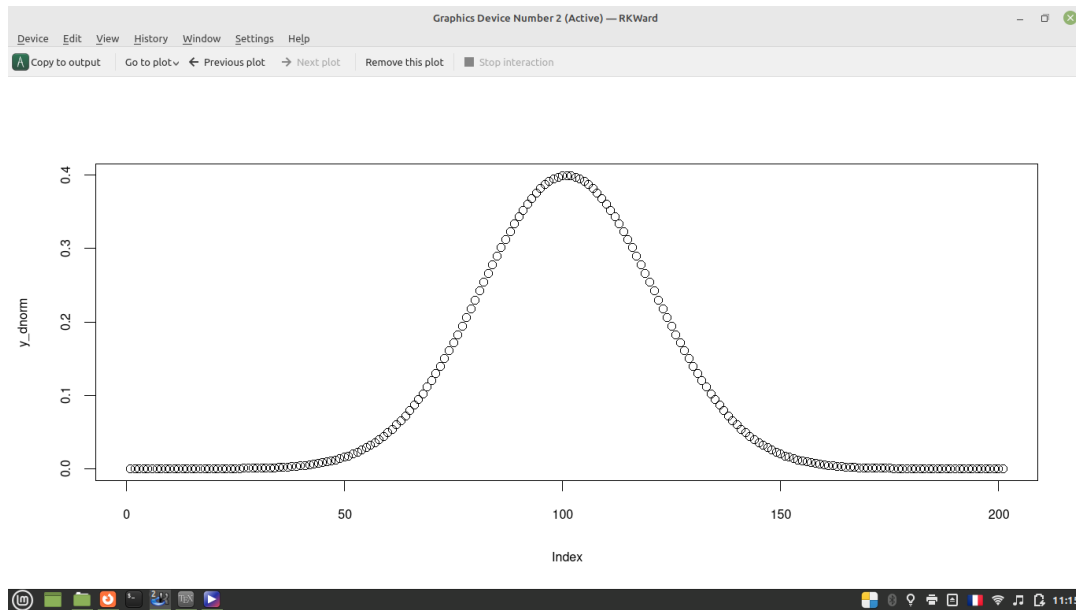


FIGURE 1.22: dnorm Function plot.

### Example 2: Distribution Function (pnorm Function)

```
214 # Specify x-values for pnorm function
215 x_pnorm <- seq(- 5, 5, by = 0.05)
216
217 # Apply pnorm function
218 y_pnorm <- pnorm(x_pnorm)
219
220 # Plot pnorm values
221 plot(y_pnorm)
```

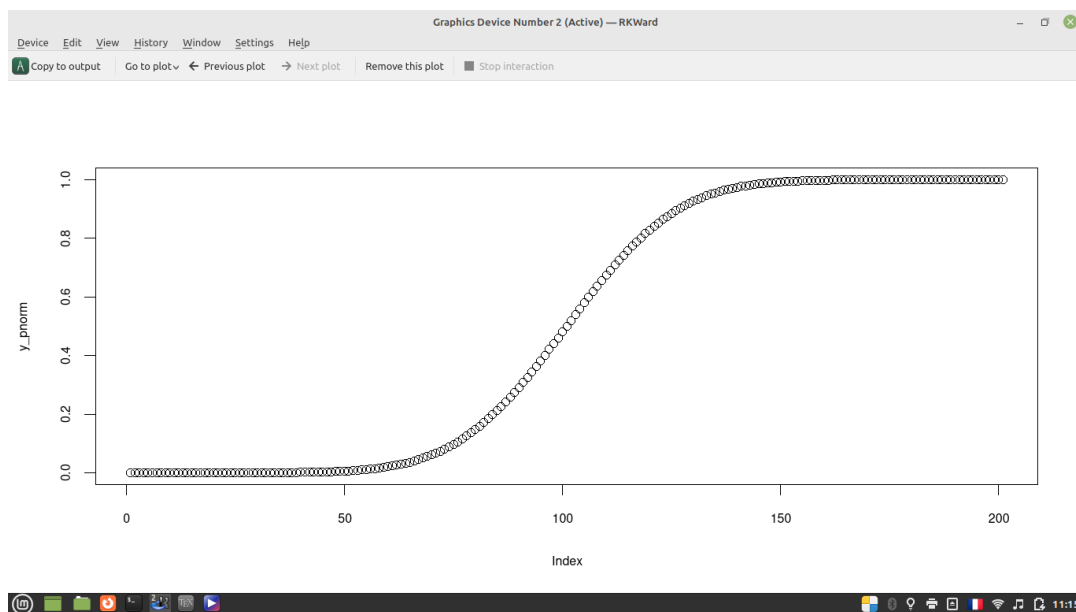


FIGURE 1.23: pnorm Function plot.

**Example 3: Quantile Function (qnorm Function)**

```

222 # Specify x-values for qnorm function
223 x_qnorm <- seq(0, 1, by = 0.005)
224
225 # Apply qnorm function
226 y_qnorm <- qnorm(x_qnorm)
227
228 # Plot qnorm values
229 plot(y_qnorm)

```

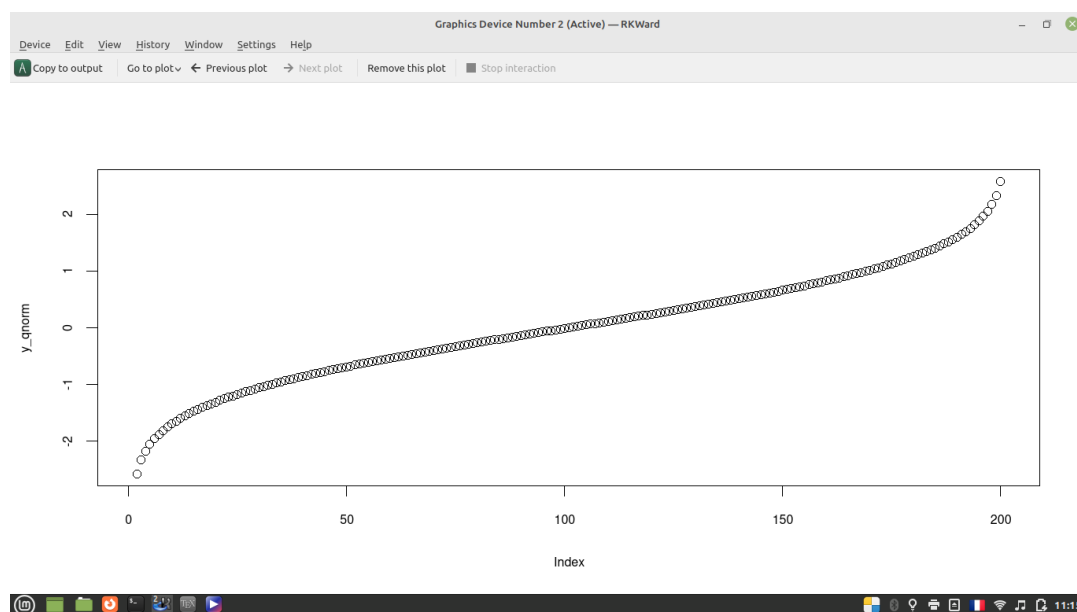


FIGURE 1.24: qnorm Function plot.

**Example 4: Random Number Generation (rnorm Function)**

```

230 # Set seed for reproducibility
231 set.seed(13579)
232
233 # Specify sample size
234 N <- 10000
235
236 # Draw N normally distributed values
237 y_rnorm <- rnorm(N)
238
239 # Print values to RStudio console      -1.234715493 -1.252833873
      -0.254778031 -1.526646627  1.097114685  2.488744223  0.779480260
      0.188375005 -1.026445945...
240 y_rnorm
241
242 # Plot pnorm values
243 plot(y_rnorm)
244

```

```

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

```

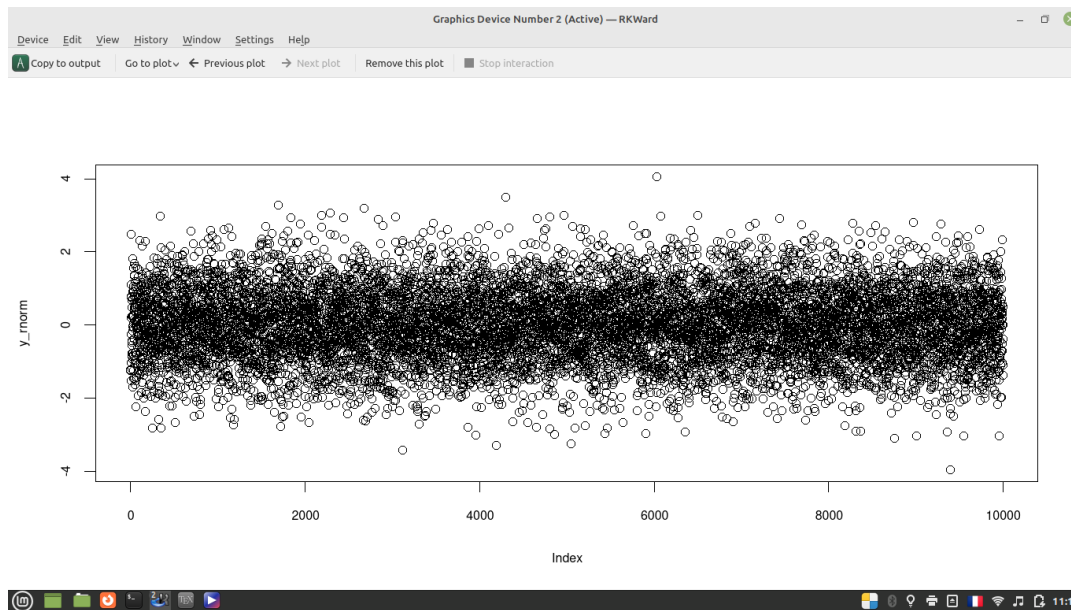


FIGURE 1.25: rnorm Function plot.

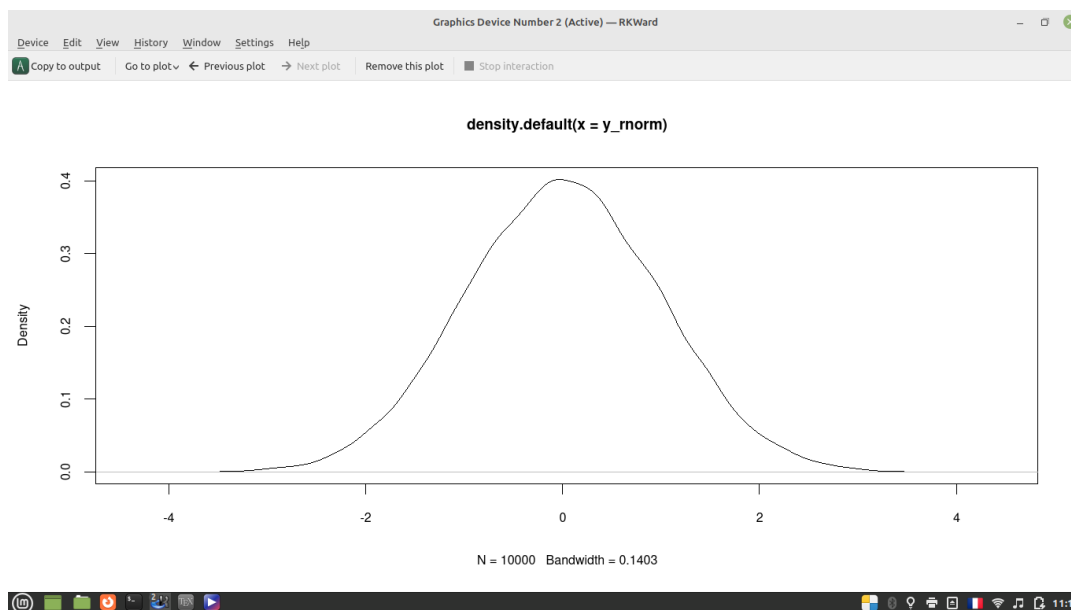


FIGURE 1.26: rnorm density plot.

### Example 5: Modify Mean & Standard Deviation

```

247 # Modify mean
248 y_rnorm2 <- rnorm(N, mean = 2)
249
250 # Modify standard deviation
251 y_rnorm3 <- rnorm(N, mean = 2, sd = 3)

```



```

252
253 # Plot default density
254 plot(density(y_rnorm),
255       xlim = c(-10, 10),
256       main = "Normal Distribution in R")
257 lines(density(y_rnorm2), col = "coral2")           # Plot density
258                                           with higher mean
259 lines(density(y_rnorm3), col = "green3")           # Plot density
260                                           with higher sd
261 legend("topleft",                                # Add legend
262       to density
263       legend = c("Mean = 0; SD = 1",
264                  "Mean = 2; SD = 1",
265                  "Mean = 2; SD = 3"),
266       col = c("black", "coral2", "green3"),
267       lty = 1)

```

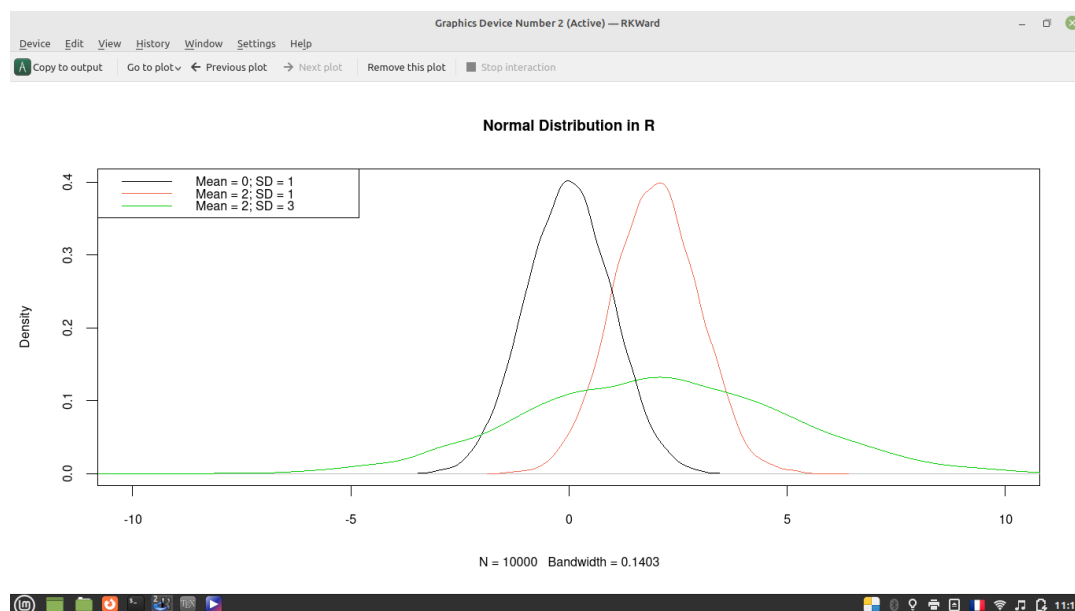


FIGURE 1.27: Mean and Standard Deviation plot.

## Appendix A

# Appendix A

### A.1 R code

```

265 #
266 #
267 #     Bernoulli Distribution in R
268 #
269 #
270
271
272
273
274
275 # Example 1: Bernoulli Probability Density Function (dbern Function
      )
276
277
278 # Install Rlab package
279 install.packages("Rlab")
280
281 # Load Rlab package.skeleton
282 library("Rlab")
283
284 # Specify x-values for dbern function
285 x_dbern <- seq(0, 10, by = 1)
286
287 # Apply dbern function
288 y_dbern <- dbern(x_dbern, prob = 0.7)
289
290 # Plot dbern values
291 plot(y_dbern, type = "o")
292
293
294
295
296
297 # Example 2: Bernoulli Cumulative Distribution Function (pbern
      Function)
298
299
300 # Specify x-values for pbern function
301 x_pbern <- seq(0, 10, by = 1)
302
303 # Apply pbern function
304 y_pbern <- pbern(x_pbern, prob = 0.7)

```

```
305
306 # Plot pbern values
307 plot(y_pbern, type = "o")
308
309
310
311
312
313 # Example 3: Bernoulli Quantile Function (qbern Function)
314
315
316 # Specify x-values for qbern function
317 x_qbern <- seq(0, 1, by = 0.1)
318
319 # Apply qbern function
320 y_qbern <- qbern(x_qbern, prob = 0.7)
321
322 # Plot qbern values
323 plot(y_qbern, type = "o")
324
325
326
327
328
329
330
331 # Example 4: Generating Random Numbers (rbern Function)
332
333
334 # Set seed for reproducibility
335 set.seed(98989)
336
337 # Specify sample size
338 N <- 10000
339
340 # Draw N random values
341 y_rbern <- rbern(N, prob = 0.7)
342
343 # Print values to RStudio console
344 y_rbern
345
346 # Plot of randomly drawn density
347 hist(y_rbern,
348       breaks = 5,
349       main = "")
350
351
352
353
354
355
356
357
358
359
360 #
361 #
```

```
362 #       Binomial Distribution in R
363 #
364 #
365
366
367
368
369
370
371
372
373 # Example 1: Binomial Density in R (dbinom Function)
374
375
376
377
378 # Specify x-values for binom function
379 x_dbinom <- seq(0, 100, by = 1)
380
381 # Apply dbinom function
382 y_dbinom <- dbinom(x_dbinom, size = 100, prob = 0.5)
383
384 # Plot dbinom values
385 plot(y_dbinom)
386
387
388
389
390
391
392
393
394
395 # Example 2: Binomial Cumulative Distribution Function (pbinom
      Function)
396
397
398
399 # Specify x-values for pbinom function
400 x_pbinom <- seq(0, 100, by = 1)
401
402 # Apply pbinom function
403 y_pbinom <- pbinom(x_pbinom, size = 100, prob = 0.5)
404
405 # Plot pbinom values
406 plot(y_pbinom)
407
408
409
410
411
412
413
414
415
416
417
```

```
418 # Example 3: Binomial Quantile Function (qbinom Function)
419
420
421 # Specify x-values for qbinom function
422 x_qbinom <- seq(0, 1, by = 0.01)
423
424 # Apply qbinom function
425 y_qbinom <- qbinom(x_qbinom, size = 100, prob = 0.5)
426
427 # Plot qbinom values
428 plot(y_qbinom)
429
430
431
432
433
434
435
436
437
438
439 # Example 4: Simulation of Random Numbers (rbinom Function)
440
441
442 # Set seed for reproducibility
443 set.seed(13579)
444
445 # Specify sample size
446 N <- 10000
447
448 # Draw N binomially distributed values
449 y_rbinom <- rbinom(N, size = 100, prob = 0.5)
450
451 # Print values to RStudio console 45 44 55 43 35 47 56 52 49 51 47
452                                     50 51 54 53 48 57 55 51...
453 y_rbinom
454
455 # Plot of randomly drawn binomial density
456 hist(y_rbinom,
457       breaks = 100,
458       main = "")
459
460
461
462
463
464
465
466 #
467 #
468 # Poisson Distribution in R
469 #
470 #
471
472
473
```

```
474
475
476
477
478 # Example 1: Poisson Density in R (dpois Function)
479
480
481
482
483
484 # Specify x-values for dpois function
485 x_dpois <- seq(- 5, 30, by = 1)
486
487 # Apply dpois function
488 y_dpois <- dpois(x_dpois, lambda = 10)
489
490 # Plot dpois values
491 plot(y_dpois)
492
493
494
495
496
497
498
499 # Example 2: Poisson Distribution Function (ppois Function)
500
501
502 # Specify x-values for ppois function
503 x_ppois <- seq(- 5, 30, by = 1)
504
505 # Apply ppois function
506 y_ppois <- ppois(x_ppois, lambda = 10)
507
508 # Plot ppois values
509 plot(y_ppois)
510
511
512
513
514
515
516
517 # Example 3: Poisson Quantile Function (qpois Function)
518
519 # Specify x-values for qpois function
520 x_qpois <- seq(0, 1, by = 0.005)
521
522 # Apply qpois function
523 y_qpois <- qpois(x_qpois, lambda = 10)
524
525 # Plot qpois values
526 plot(y_qpois)
527
528
529
530
```

```
531
532
533
534 # Example 4: Random Number Generation (rpois Function)
535
536
537
538
539 # Set seed for reproducibility
540 set.seed(13579)
541
542 # Specify sample size
543 N <- 10000
544
545 # Draw N poisson distributed values
546 y_rpois <- rpois(N, lambda = 10)
547
548 # Print values to RStudio console      6 14  8 16  6 12 10  6  7 11
549                                         7 12 10 16  7  7  7 19 13
550 y_rpois
551
552 # Plot histogram of rpois values
553 hist(y_rpois,
554       breaks = 100,
555       main = "Poisson Distribution in R")
556
557
558
559
560
561
562
563
564
565
566
567 #
568 #
569 #     Continuous Uniform Distribution in R (4 Examples) | dunif,
570 #     punif, qunif & runif Functions
571 #
572
573
574
575
576
577
578 # Example 1: Uniform Probability Density Function (dunif Function)
579
580
581 # Specify x-values for dunif function
582 x_dunif <- seq(0, 100, by = 1)
583
584 # Apply dunif function
585 y_dunif <- dunif(x_dunif, min = 10, max = 50)
```

```
586
587 # Plot dunif values
588 plot(y_dunif, type = "o")
589
590
591
592
593
594
595
596
597 # Example 2: Uniform Cumulative Distribution Function (punif
    Function)
598
599
600
601 # Specify x-values for punif function
602 x_punif <- seq(0, 100, by = 1)
603
604 # Apply punif function
605 y_punif <- punif(x_punif, min = 10, max = 50)
606
607 # Plot punif values
608 plot(y_punif, type = "o")
609
610
611
612
613
614
615 # Example 3: Uniform Quantile Function (qunif Function)
616
617
618
619
620 # Specify x-values for qunif function
621 x_qunif <- seq(0, 1, by = 0.01)
622
623 # Apply qunif function
624 y_qunif <- qunif(x_qunif, min = 10, max = 50)
625
626 # Plot qunif values
627 plot(y_qunif, type = "o")
628
629
630
631
632
633
634
635 # Example 4: Generating Random Numbers (runif Function)
636
637
638
639 # Set seed for reproducibility
640 set.seed(91929)
641
```



```
642 # Specify sample size
643 N <- 1000000
644
645 # Draw N uniformly distributed values
646 y_runif <- runif(N, min = 10, max = 50)
647
648 # Print values to RStudio console    27.98052 49.43937 24.66723
        39.36479 36.84591 40.94262 25.38942 35.59081...
649 y_runif
650
651 # Plot of randomly drawn uniformly density
652 hist(y_runif,
653       breaks = 50,
654       main = "",
655       xlim = c(0, 100))
656
657
658
659
660
661
662
663 #
664 #
665 #     Exponential Distribution in R (4 Examples) | dexp, pexp, qexp
        & rexp Functions
666 #
667 #
668
669
670
671
672
673
674 # Example 1: Exponential Density in R (dexp Function)
675
676
677
678 # Specify x-values for exp function
679 x_dexp <- seq(0, 1, by = 0.02)
680
681 # Apply exp function
682 y_dexp <- dexp(x_dexp, rate = 5)
683
684 # Plot dexp values
685 plot(y_dexp)
686
687
688
689
690
691 # Example 2: Exponential Cumulative Distribution Function (pexp
        Function)
692
693
694 # Specify x-values for pexp function
695 x_pexp <- seq(0, 1, by = 0.02)
```

```
696
697 # Apply pexp function
698 y_pexp <- pexp(x_pexp, rate = 5)
699
700 # Plot pexp values
701 plot(y_pexp)
702
703
704
705
706
707
708
709
710 # Example 3: Exponential Quantile Function (qexp Function)
711
712
713
714 # Specify x-values for qexp function
715 x_qexp <- seq(0, 1, by = 0.02)
716
717 # Apply qexp function
718 y_qexp <- qexp(x_qexp, rate = 5)
719
720 # Plot qexp values
721 plot(y_qexp)
722
723
724
725
726
727
728
729 # Example 4: Random Number Generation (rexp Function)
730
731
732
733 # Set seed for reproducibility
734 set.seed(13579)
735
736 # Specify sample size
737 N <- 10000
738
739 # Draw N exp distributed values
740 y_rexp <- rexp(N, rate = 5)
741
742 # Print values to RStudio console
743 y_rexp
744
745 # Plot of randomly drawn exp density
746 hist(y_rexp, breaks = 100, main = "")
747
748
749
750
751
752
```

```
753 #
754 #
755 #     Normal Distribution in R (5 Examples) | dnorm, pnorm, qnorm &
       rnorm Functions
756 #
757 #
758
759
760
761
762
763
764 # Example 1: Normally Distributed Density (dnorm Function)
765
766
767
768 # Specify x-values for dnorm function
769 x_dnorm <- seq(- 5, 5, by = 0.05)
770
771 # Apply dnorm function
772 y_dnorm <- dnorm(x_dnorm)
773
774 # Plot dnorm values
775 plot(y_dnorm)
776
777
778
779
780
781
782
783 # Example 2: Distribution Function (pnorm Function)
784
785
786
787
788 # Specify x-values for pnorm function
789 x_pnorm <- seq(- 5, 5, by = 0.05)
790
791 # Apply pnorm function
792 y_pnorm <- pnorm(x_pnorm)
793
794 # Plot pnorm values
795 plot(y_pnorm)
796
797
798
799
800
801
802
803
804 # Example 3: Quantile Function (qnorm Function)
805
806
807
808
```

```
809 # Specify x-values for qnorm function
810 x_qnorm <- seq(0, 1, by = 0.005)
811
812 # Apply qnorm function
813 y_qnorm <- qnorm(x_qnorm)
814
815 # Plot qnorm values
816 plot(y_qnorm)
817
818
819
820
821
822
823
824
825 # Example 4: Random Number Generation (rnorm Function)
826
827
828
829 # Set seed for reproducibility
830 set.seed(13579)
831
832 # Specify sample size
833 N <- 10000
834
835 # Draw N normally distributed values
836 y_rnorm <- rnorm(N)
837
838 # Print values to RStudio console      -1.234715493 -1.252833873
      -0.254778031 -1.526646627  1.097114685  2.488744223  0.779480260
      0.188375005 -1.026445945...
839 y_rnorm
840
841 # Plot pnorm values
842 plot(y_rnorm)
843
844 # Plot density of pnorm values
845 plot(density(y_rnorm))
846
847
848
849
850
851
852
853 # Example 5: Modify Mean & Standard Deviation
854
855
856
857 # Modify mean
858 y_rnorm2 <- rnorm(N, mean = 2)
859
860 # Modify standard deviation
861 y_rnorm3 <- rnorm(N, mean = 2, sd = 3)
862
863 # Plot default density
```

```
864 plot(density(y_rnorm),
865       xlim = c(- 10, 10),
866       main = "Normal Distribution in R")
867 lines(density(y_rnorm2), col = "coral2")           # Plot density
868       with higher mean
869 lines(density(y_rnorm3), col = "green3")          # Plot density
870       with higher sd
871 legend("topleft",                                # Add legend
872       to density
873       legend = c("Mean = 0; SD = 1",
874                  "Mean = 2; SD = 1",
875                  "Mean = 2; SD = 3"),
876       col = c("black", "coral2", "green3"),
877       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 Grolemond. *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).