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

Confidence Intervals and Risk in R
PART 1

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-04

May 11, 2022

Contents

Chapter 1

Confidence Intervals in R

1.1 Confidence Interval

As requested we will using **runif** function to generates random deviates of the **uniform distribution** from size of 1000 and range between -2 and 3.

1.1.1 Generation of Random uniform Distribution Seed

```
LO = -2 # Declaration of lower bownd
UP = 3 # Declaration of upper bownd
n = 1000 # Declaration of sample size

x <- runif(n, LO, UP) # Running runif function to create population
hist(x, freq = FALSE, xlab = 'x', density = 20) # Histogram of
    population
x # all generated values
```

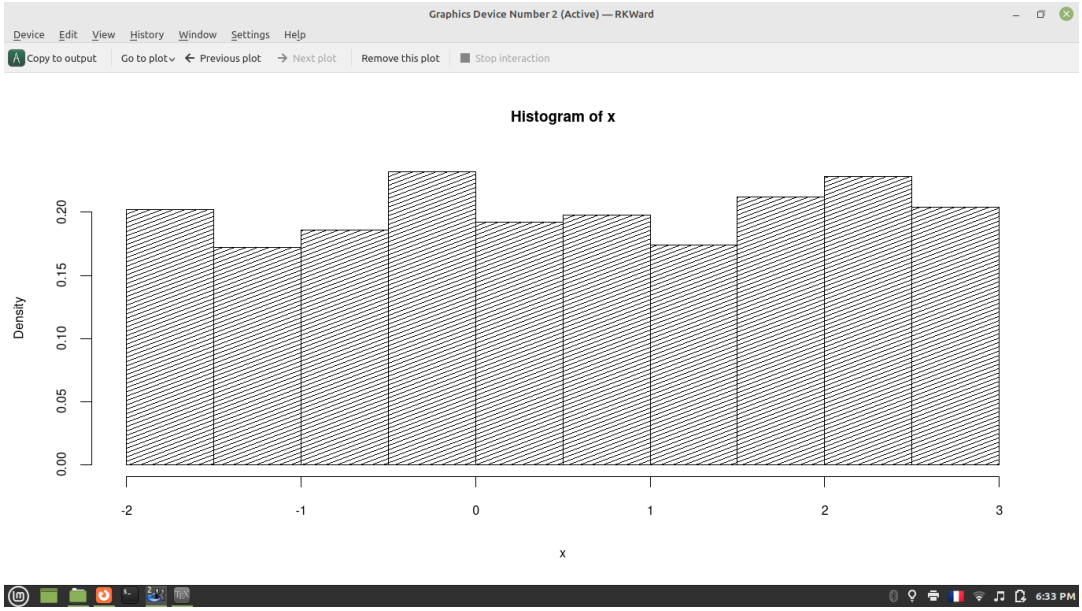

FIGURE 1.1: Seed Histogram.

Sample of generated numbers :

```
[656]  0.3813996161  2.4257347495  2.9501461792 -0.3439974906  2.6721924189
```

```
[661] -1.7069758398 -1.9919105659 -0.5682839947 -0.1698572240  2.1766188550
[666]  2.1259460303 -0.4748134005  2.6861825290 -0.3083081152  0.8204065480
[671]  2.7893994430 -0.4446464770 -1.2376538937  1.7844204041 -0.4890021197
[676]  2.9545168958 -0.9832733071 -0.1086031161  1.9523365453  0.1649452916
[681] -0.8115781138 -1.5075847686  2.1916635458 -0.6813517509  1.1595426821
[686] -1.2972299247  1.1057026773 -1.0271868056  0.0378431329  1.4667883010
[691]  1.6694702739  2.1264238802  2.9148606462  2.6315843845  0.8007933067
[696]  2.0933396120  1.8568044163  1.5384381283  2.9681203486 -1.4027119374
[701]  0.5982736656 -0.4392616409  0.0532791491  1.1336477492 -0.4013634117
[706]  2.1036622517  1.1725118810  1.4840549517  1.4851128771 -0.9378550777
[711]  1.3588998194 -1.7498405632 -1.0858148329  2.7649365244 -1.6944917948
[716]  2.6904253052 -1.3718907470  1.4137118515 -0.3775374934 -0.8040759950
[721] -1.3621708534  0.0552953065 -1.6524381090  1.5260674357  2.5928918663
[726]  2.7491736128 -1.8523521284  0.2638581775  0.3966596590 -1.4307652919
[731] -1.5760219381 -1.1228304966 -1.8921418609  1.4406364267  0.5347665627
[736]  1.9234586596 -0.7565151707  2.0440741449  2.7295886686  1.1158512493
[741]  1.5015663505  0.5561288605  1.4875788400  2.3570341938  2.8440772563
[746] -0.3308353100  0.4111624272  1.3974237274  1.0447405553 -1.3868167617
[751]  1.6946422944  2.7661013680  1.5291836690  2.9929703807 -0.5640524689
[756] -0.8812302880 -0.0748715147 -1.1108034016  1.6513440299  1.8687401486
[761] -1.2423470458 -1.7383970383  1.5178913060 -1.1912305378  0.7003954467
[766]  2.4230878598 -0.9534112730  1.1830788974  1.1611230748 -1.1721150582
[771]  1.7582610291  0.0346237416  1.9112062566  1.4791723755 -1.7659081384
[776]  1.5747120762  2.1935196554 -0.2110054267  2.6586451575  0.6185839961
[781]  0.9111335962  2.9610665615  1.2534163748  0.7543342090  0.4418109108
[786]  1.7382183538  2.7631716216  0.3801716522 -0.3324272453 -1.6288395429
[791]  0.9629155551 -0.9031352359  0.0713478327 -1.9929349020  0.2839912050
[796] -0.2094068220  2.1137463006  1.1335198425 -1.3629527255  0.7328717150
[801] -0.8635372289 -1.2511274433  2.0804919973 -0.0102763081  2.0244089393
[806]  2.0593909181  1.4906496108  1.6627415335  2.8765612335  2.5833753119
[811]  2.7221900942 -0.2749454468  2.9613368977 -1.4997302629  2.2851109779
[816] -0.2907404716 -1.0059362201 -1.3930226308 -0.5002697073 -0.6187703651
[821]  2.3164683203  0.3184335779  2.9863529194  2.7773846728  0.4733544211
```

1.1.2 Finding Mean and Standard Deviation

```
8  stddev = sd(x) # Calculatiing standerd deviation
9  stddev # standerd deviation
10 center = mean(x) # Calculating mean
11 center # mean
```

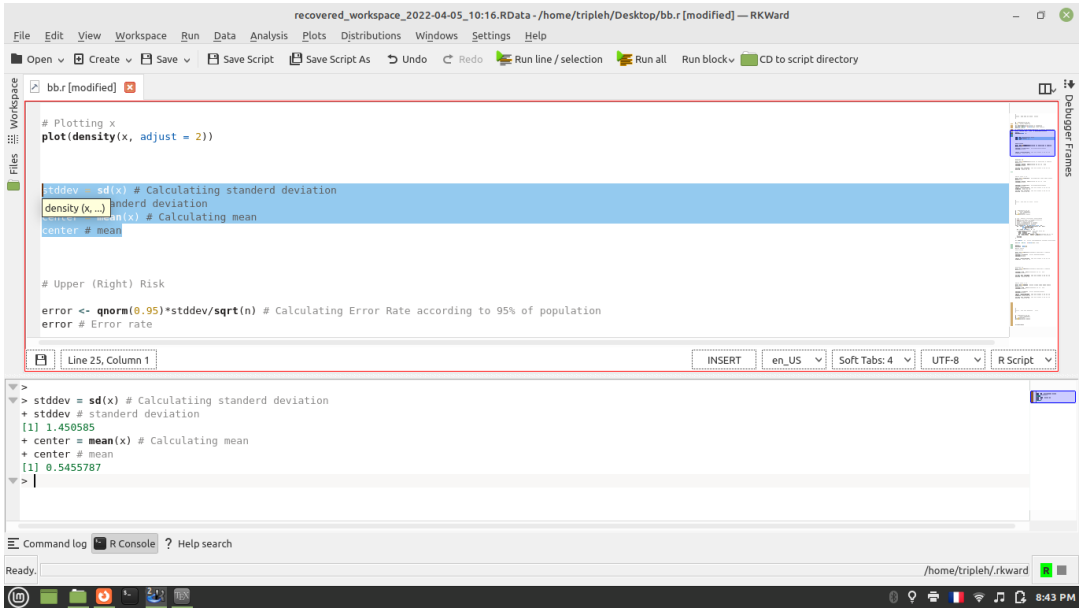

FIGURE 1.2: Mean and SD.

1.1.3 Plotting Density

```
# Plotting
plot(density(x, adjust = 2))
```

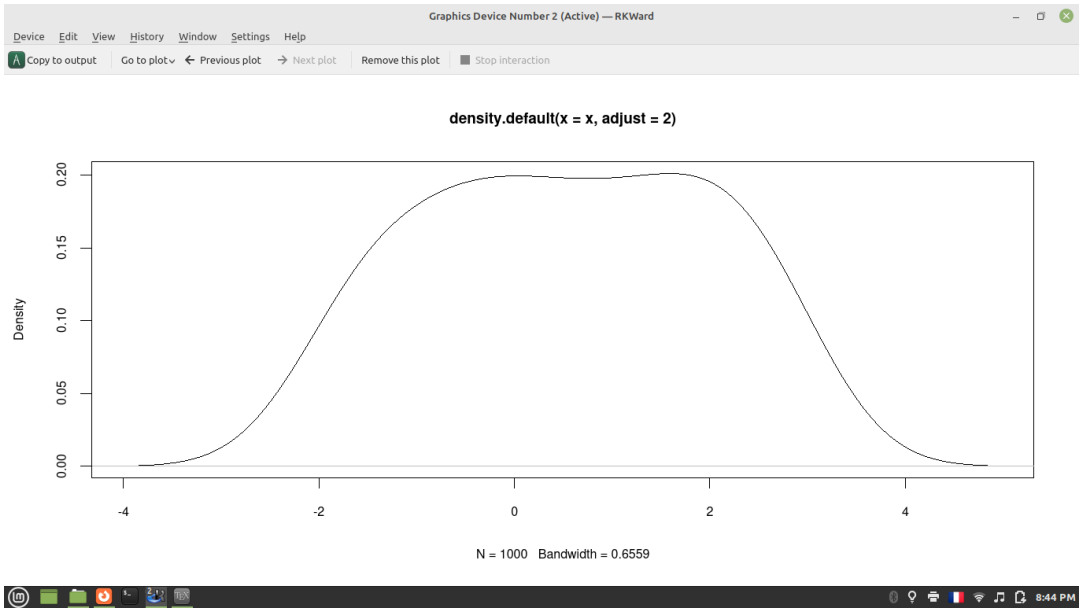

FIGURE 1.3: Plot Density.

1.1.4 Right Risk of 95%

calculating 95% error rate, upper limit

```
error <- qnorm(0.95)*stddev/sqrt(n) # Calculating Error Rate
    according to 95% of population
```

```
15 error # Error rate
16
17 upper_bound <- UP - (center + error) # calculating upper bound (
       risk a droit)
18 upper_bound # upper bound
```

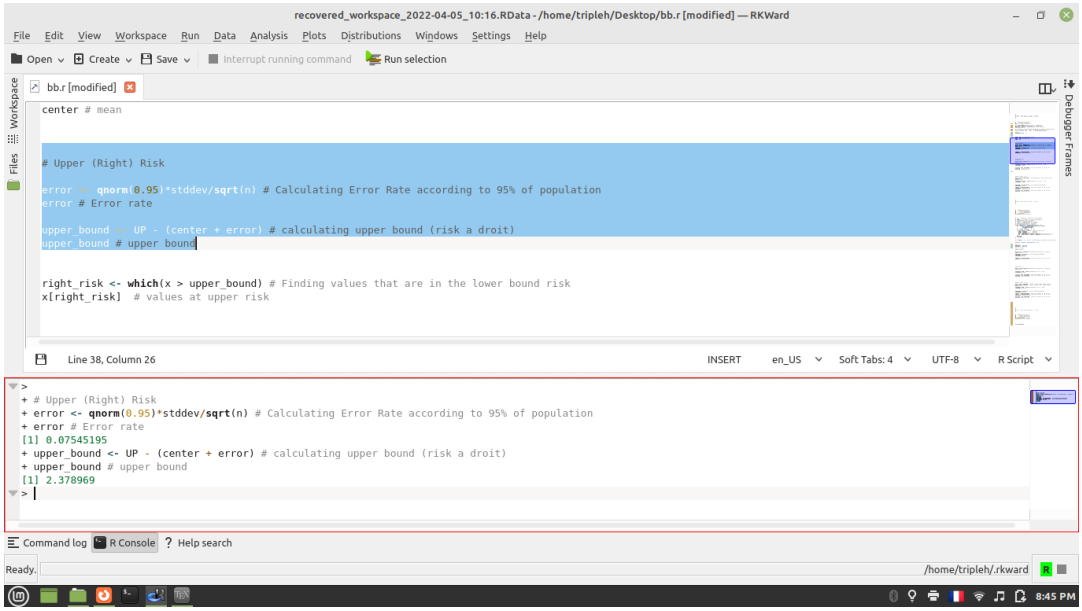

FIGURE 1.4: Error Rate Right.

Upper Limit

```
+ error # Error rate
[1] 0.07545195

+ upper_bound # upper bound
[1] 2.378969
```

1.1.5 Left Risk of 95%

calculating 95% error rate, lower limit

```
19 error <- qnorm(0.95)*stddev/sqrt(n) # Calculating Error Rate
       according to 95% of population
20 error # Error rate
21
22 lower_bound <- LO +(center - error) # Calculating lower bound (risk
       a gouche)
23 lower_bound # lower bound
```

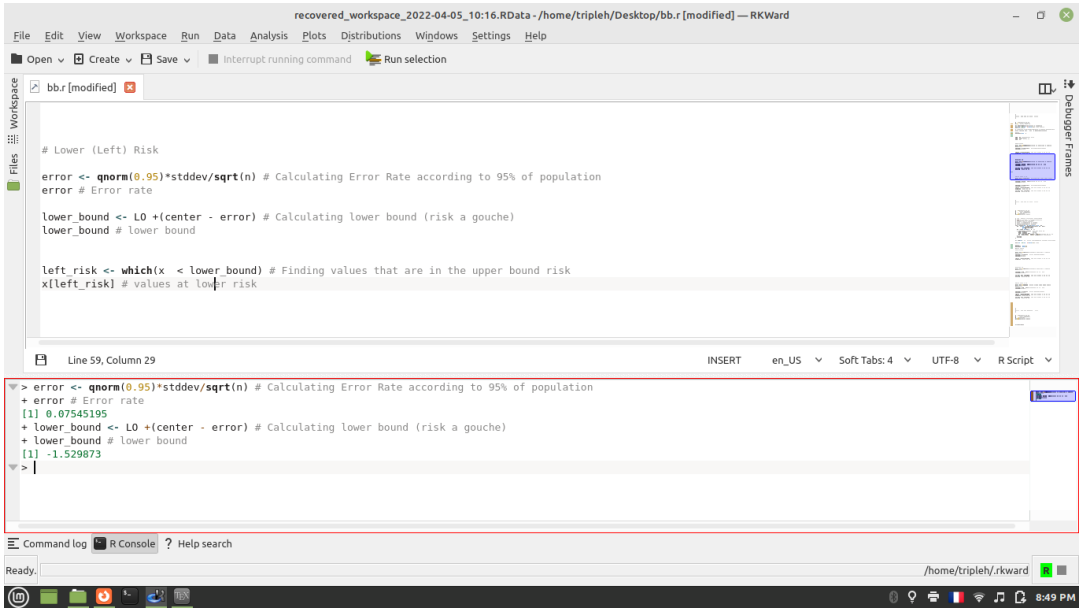

FIGURE 1.5: Error Rate Left.

Lower Limit

```
[1] 0.07545195

+ lower_bound # lower bound
[1] -1.529873
```

1.1.6 Balanced Risk of 95%

calculating 95% error rate, upper limit and lower limit

```
24 error <- qnorm(0.975)*stddev/sqrt(n) # Calculating Error Rate
       according to 95% of population
25 error # Error rate
26
27 lower_bound <- LO +(center - error) # Calculating lower bound (risk
       a gouche)
28 lower_bound # lower bound
29
30
31 upper_bound <- UP - (center + error) # calculating upper bound (
       risk a droit)
32 upper_bound # upper bound
```

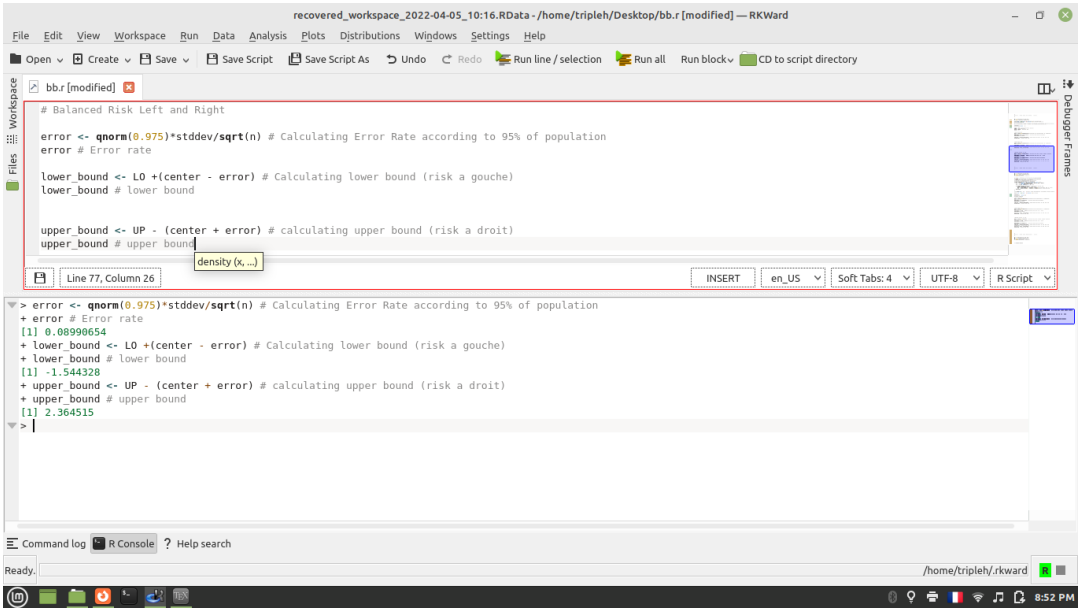

FIGURE 1.6: Error Rate.

Upper Limit and Lower Limit

```
+ error # Error rate
[1] 0.08990654

+ lower_bound # lower bound
[1] -1.544328

+ upper_bound # upper bound
[1] 2.364515
```