Probabilités et Intégration : Simulation de vecteurs Gaussiens et Estimation statistique

Amadou BA, Malick FAYE, Abdoulaye NDAO et Seynabou FAYE

2024-04-06

Exercice 1:

Dans toute cette épreuve, on se donne un espace probabilisé (Ω, F, P).

1. **Générer une réalisation d'un échantillon de taille n = 200 pour μ = 176 et σ=5 par la méthode de Box-Muller.**

   Dans cette partie nous allons deux échantillon de nombres aléatoires uniformément distribués de talle n=200 dans le but de générer des variables aléatoires normalement distribuées par la méthode de Box-Muller en fin de pouvoire générer une réalisation d'un échantillon de taille n = 200 pour μ = 176 et σ=5.

- Définition des paramètres

```
n <- 200
μ <- 176
σ <- sqrt(5)  # Car σ² = 5
```

- Générer les nombres aléatoires uniformément distribués

```
u1 <- runif(n)
u1

  [1] 0.892896695 0.705820215 0.004194243 0.132340195 0.744641834 0.830
582580
  [7] 0.299008210 0.934059212 0.404891548 0.081655403 0.705913194 0.345
984424
 [13] 0.436044609 0.603709322 0.925720575 0.218445368 0.966268910 0.824
442594
 [19] 0.167799692 0.430571768 0.643606893 0.401620022 0.971658733 0.981
045175
 [25] 0.970394482 0.484782622 0.121670826 0.922073283 0.947531427 0.299
625662
 [31] 0.634409141 0.958973917 0.010516155 0.397443885 0.233493705 0.324
748083
 [37] 0.577101629 0.041261135 0.100301580 0.625596400 0.890448890 0.770
```

```
484019
 [43] 0.968498082 0.073105757 0.016614587 0.991079620 0.067748587 0.505
940872
 [49] 0.400077192 0.201929283 0.089728367 0.380499307 0.445169030 0.511
251470
 [55] 0.800464171 0.830489819 0.925765031 0.212392228 0.109954626 0.542
080994
 [61] 0.551749337 0.950131009 0.367416768 0.168973609 0.666016203 0.763
170386
 [67] 0.792668477 0.696560862 0.085269233 0.909724304 0.412490517 0.639
478840
 [73] 0.429743306 0.106814346 0.001991077 0.897305785 0.021039458 0.704
944814
 [79] 0.869870757 0.208181103 0.409999968 0.745934201 0.452118594 0.759
843973
 [85] 0.416624703 0.958985377 0.724712910 0.252886495 0.350692466 0.546
766025
 [91] 0.436305967 0.369758731 0.735640201 0.613033634 0.764927399 0.798
357534
 [97] 0.406034997 0.335073213 0.638171656 0.428424387 0.382319879 0.550
179255
[103] 0.577323501 0.958716474 0.611024035 0.550069375 0.905138326 0.190
162846
[109] 0.758382601 0.671930088 0.338881040 0.420704451 0.550118063 0.712
934015
[115] 0.141100166 0.614985036 0.652350175 0.211417487 0.648683900 0.203
631602
[121] 0.511499627 0.199624793 0.452979480 0.526737339 0.541968307 0.715
076439
[127] 0.296296194 0.693969848 0.914470829 0.960009265 0.070297378 0.954
348466
[133] 0.023395964 0.490288934 0.770796790 0.479663856 0.624753656 0.416
759666
[139] 0.995445545 0.155499527 0.131377176 0.473745216 0.675078954 0.206
672869
[145] 0.540463172 0.844418347 0.954542040 0.195606825 0.180804576 0.666
564576
[151] 0.422626070 0.589443004 0.824666242 0.477889185 0.315004180 0.932
402821
[157] 0.740914176 0.978150503 0.832562535 0.025401384 0.807600556 0.775
820161
[163] 0.764025179 0.088630824 0.086932129 0.306051196 0.383233007 0.531
300229
[169] 0.315645385 0.046799861 0.719507026 0.268006810 0.622147097 0.922
146210
[175] 0.151145299 0.804457380 0.907250439 0.070401583 0.421423908 0.904
050984
[181] 0.013460762 0.106375495 0.427027151 0.006686671 0.371271318 0.953
737504
[187] 0.600108258 0.726400965 0.832587103 0.712310754 0.757412352 0.737
```

```
177316
[193] 0.810898621 0.634446592 0.832070609 0.659124956 0.013707903 0.825
345765
[199] 0.517026025 0.384390426
```

```
u2 <- runif(n)
u2
```

```
  [1] 0.68194096 0.64179047 0.60888763 0.96517589 0.09600358 0.06257793
  [7] 0.59011415 0.97646184 0.65792866 0.73020476 0.45337200 0.56150800
 [13] 0.91834836 0.71218066 0.09456322 0.94643543 0.78265171 0.65300488
 [19] 0.77098627 0.26919782 0.30668127 0.37547533 0.57324828 0.07168031
 [25] 0.07505189 0.58013020 0.56895526 0.10182138 0.77081634 0.13821467
 [31] 0.21373203 0.29358133 0.34613332 0.21486646 0.74468747 0.37860467
 [37] 0.36925006 0.39393868 0.40005809 0.69256777 0.94057039 0.90222967
 [43] 0.89537243 0.66698197 0.50318653 0.71305815 0.79090084 0.95604260
 [49] 0.78062888 0.33236126 0.77082029 0.56590209 0.70898389 0.32054767
 [55] 0.99433013 0.69801410 0.12764594 0.27991199 0.12710359 0.96818719
 [61] 0.87603089 0.77942980 0.01978903 0.15047092 0.07446913 0.35153023
 [67] 0.82259817 0.65622716 0.47296617 0.26994454 0.02477327 0.24258841
 [73] 0.22975048 0.59582359 0.35113988 0.46998702 0.31512519 0.02205231
 [79] 0.41324078 0.88540960 0.69877006 0.63645406 0.29051033 0.46095458
 [85] 0.04294247 0.37977204 0.17648259 0.24003839 0.09211639 0.61723774
 [91] 0.16822226 0.96604817 0.01316612 0.73885664 0.05260731 0.09622661
 [97] 0.40763706 0.30395376 0.68878164 0.17758969 0.51129861 0.04915809
[103] 0.17804612 0.07802104 0.95892314 0.57589513 0.56263823 0.53169230
[109] 0.74660993 0.50065507 0.07407392 0.06428238 0.32773487 0.84699629
[115] 0.86148363 0.78250252 0.32001931 0.50236257 0.45771068 0.87604377
[121] 0.89580374 0.91635831 0.36537455 0.22397915 0.83595861 0.33809104
[127] 0.57386968 0.99961202 0.42179577 0.43092391 0.88525549 0.16545709
[133] 0.34356573 0.56344407 0.73754775 0.29732413 0.51009533 0.13640760
[139] 0.78537979 0.48122048 0.91934930 0.91899159 0.77189597 0.97195277
[145] 0.55063488 0.89750042 0.36688641 0.63805382 0.10972752 0.14811635
[151] 0.25667035 0.78498298 0.68683001 0.04292097 0.12916789 0.30368093
[157] 0.24367109 0.80909103 0.87891079 0.39050843 0.65992559 0.38930016
[163] 0.12511044 0.68071739 0.77137016 0.87503221 0.88201178 0.38522716
[169] 0.08366725 0.80307650 0.79414486 0.54700049 0.87015918 0.08383018
[175] 0.62128024 0.34803169 0.01860027 0.14631462 0.71075740 0.27212894
[181] 0.65443942 0.37271631 0.37016707 0.73692755 0.27955230 0.31468624
[187] 0.14426825 0.95649547 0.66551689 0.59824905 0.58652371 0.72036026
[193] 0.11596413 0.16442817 0.76935305 0.66376318 0.78335669 0.30340327
[199] 0.94234261 0.59440004
```

- Méthode de Box-Muller pour générer des variables aléatoires normalement distribuées

```
z1 <- sqrt(-2 * log(u1)) * cos(2 * pi * u2)
z1
```

```
  [1] -0.19740033 -0.52481360 -2.56414442  1.96320704  0.63239882  0.56
281265
```

```
  [7] -1.31139370  0.36533335 -0.73525218 -0.27769018 -0.79901957 -1.34
950155
 [13]  1.12254743 -0.23649088  0.32555982  1.64640160  0.05336788 -0.35
567415
 [19]  0.24842083 -0.15621200 -0.32731812 -0.95796548 -0.21484345  0.17
612768
 [25]  0.21840611 -1.05405020 -1.86287970  0.32315452  0.04281887  1.00
299571
 [31]  0.21552011 -0.07827386 -1.71422710  0.29745033 -0.05692291 -1.08
426687
 [37] -0.71417768 -1.98478381 -1.73544738 -0.34197419  0.44852941  0.59
010488
 [43]  0.20028491 -1.13971861 -2.86210828 -0.03079432  0.58975250  1.12
309380
 [49]  0.25888857 -0.88490363  0.28644292 -1.27267500 -0.32425483 -0.49
680750
 [55]  0.66675507 -0.19556107  0.27307613 -0.32889033  1.46606115  1.08
461821
 [61]  0.77612165  0.05880987  1.40417852  1.10389279  0.80469231 -0.43
785008
 [67]  0.30027942 -0.47256495 -2.18704837 -0.05436992  1.31473147  0.04
402022
 [73]  0.16491282 -1.74311991 -2.09336946 -0.45727599 -1.10567049  0.82
821025
 [79] -0.45150337  1.33193455 -0.42245223 -0.50107208 -0.31726303 -0.71
894556
 [85]  1.27542897 -0.21068778  0.35763784  0.10372013  1.21186889 -0.81
396025
 [91]  0.63304394  1.37863067  0.78091983 -0.06920850  0.69245502  0.55
214377
 [97] -1.12280710 -0.49176422 -0.35563849  0.57215499 -1.38322815  1.04
143956
[103]  0.45790774  0.25618051  0.95971527 -0.97137793 -0.41233547 -1.78
601354
[109] -0.01584058 -0.89173342  1.31464703  1.21003720 -0.51300009  0.47
088855
[115]  1.27564262  0.19997579 -0.39365321 -1.76271274 -0.89773563  1.26
977017
[121]  0.91852169  1.55291995 -0.83447887  0.18430066  0.56915554 -0.43
050723
[127] -1.39473312  0.85478020 -0.37283917 -0.25921120  1.73096459  0.15
485788
[133] -1.51990531 -1.10033795 -0.05639779 -0.35514438 -0.96799562  0.86
614464
[139]  0.02106594 -1.91589175  1.76158800  1.06741221  0.12157473  1.74
823343
[145] -1.05367874  0.46506721 -0.20442196 -1.16841479  1.42708613  0.53
799714
[151] -0.05499020  0.22418324 -0.24003213  1.17129449  1.04627664 -0.12
381352
```

```
[157]  0.03078764  0.07626183  0.43846195 -2.09385553 -0.35054920 -0.54
699523
[163]  0.51844093 -0.92835990  0.29588672  1.08833915  1.02152471 -0.84
467925
[169]  1.31358607  0.81004810  0.22218493 -1.55255260  0.66762802  0.34
804912
[175] -1.40634448 -0.38111902  0.43820847  1.39687645 -0.32087006 -0.06
224940
[181] -1.65842086 -1.47529102 -0.89401443 -0.25964492 -0.25988759 -0.12
168034
[187]  0.62306204  0.76988058 -0.30644863 -0.67167351 -0.63797744 -0.14
459518
[193]  0.48307576  0.48854272  0.07355144 -0.47088029  0.60941244 -0.20
402328
[199]  1.07406617 -1.14662598
```

- Calculons de l'échantillon

```
echantillon <- μ + σ * z1
echantillon
```

```
  [1] 175.5586 174.8265 170.2664 180.3899 177.4141 177.2585 173.0676 17
6.8169
  [9] 174.3559 175.3791 174.2133 172.9824 178.5101 175.4712 176.7280 17
9.6815
 [17] 176.1193 175.2047 176.5555 175.6507 175.2681 173.8579 175.5196 17
6.3938
 [25] 176.4884 173.6431 171.8345 176.7226 176.0957 178.2428 176.4819 17
5.8250
 [33] 172.1669 176.6651 175.8727 173.5755 174.4031 171.5619 172.1194 17
5.2353
 [41] 177.0029 177.3195 176.4479 173.4515 169.6001 175.9311 177.3187 17
8.5113
 [49] 176.5789 174.0213 176.6405 173.1542 175.2749 174.8891 177.4909 17
5.5627
 [57] 176.6106 175.2646 179.2782 178.4253 177.7355 176.1315 179.1398 17
8.4684
 [65] 177.7993 175.0209 176.6714 174.9433 171.1096 175.8784 178.9398 17
6.0984
 [73] 176.3688 172.1023 171.3191 174.9775 173.5276 177.8519 174.9904 17
8.9783
 [81] 175.0554 174.8796 175.2906 174.3924 178.8519 175.5289 176.7997 17
6.2319
 [89] 178.7098 174.1799 177.4155 179.0827 177.7462 175.8452 177.5484 17
7.2346
 [97] 173.4893 174.9004 175.2048 177.2794 172.9070 178.3287 177.0239 17
6.5728
[105] 178.1460 173.8279 175.0780 172.0064 175.9646 174.0060 178.9396 17
8.7057
[113] 174.8529 177.0529 178.8524 176.4472 175.1198 172.0585 173.9926 17
8.8393
```

```
[121] 178.0539 179.4724 174.1340 176.4121 177.2727 175.0374 172.8813 17
7.9113
[129] 175.1663 175.4204 179.8706 176.3463 172.6014 173.5396 175.8739 17
5.2059
[137] 173.8355 177.9368 176.0471 171.7159 179.9390 178.3868 176.2718 17
9.9092
[145] 173.6439 177.0399 175.5429 173.3873 179.1911 177.2030 175.8770 17
6.5013
[153] 175.4633 178.6191 178.3395 175.7231 176.0688 176.1705 176.9804 17
1.3180
[161] 175.2161 174.7769 177.1593 173.9241 176.6616 178.4336 178.2842 17
4.1112
[169] 178.9373 177.8113 176.4968 172.5284 177.4929 176.7783 172.8553 17
5.1478
[177] 176.9799 179.1235 175.2825 175.8608 172.2917 172.7011 174.0009 17
5.4194
[185] 175.4189 175.7279 177.3932 177.7215 175.3148 174.4981 174.5734 17
5.6767
[193] 177.0802 177.0924 176.1645 174.9471 177.3627 175.5438 178.4017 17
3.4361
```

2. **Représenter sur une même figure l'histogramme de cet échantillon et la densité de la loi normale**

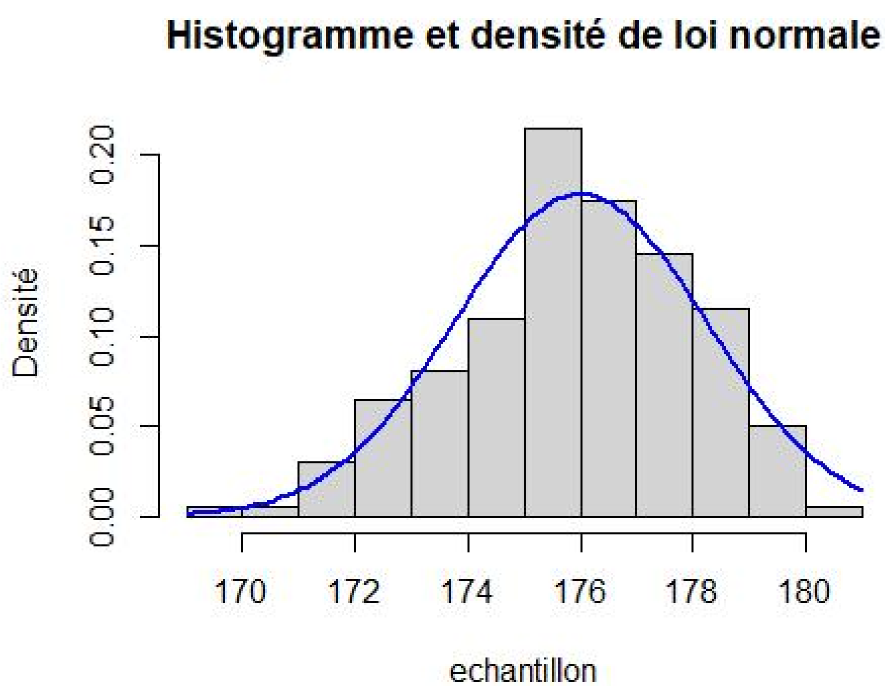
   Créons d'abord une séquence de valeurs pour la densité en utilisant la fonction seq().

```
x <- seq(min(echantillon), max(echantillon), length.out = 100)
x

  [1] 169.6001 169.7091 169.8181 169.9271 170.0361 170.1451 170.2541 17
0.3630
  [9] 170.4720 170.5810 170.6900 170.7990 170.9080 171.0170 171.1260 17
1.2349
 [17] 171.3439 171.4529 171.5619 171.6709 171.7799 171.8889 171.9978 17
2.1068
 [25] 172.2158 172.3248 172.4338 172.5428 172.6518 172.7608 172.8697 17
2.9787
 [33] 173.0877 173.1967 173.3057 173.4147 173.5237 173.6327 173.7416 17
3.8506
 [41] 173.9596 174.0686 174.1776 174.2866 174.3956 174.5046 174.6135 17
4.7225
 [49] 174.8315 174.9405 175.0495 175.1585 175.2675 175.3765 175.4854 17
5.5944
 [57] 175.7034 175.8124 175.9214 176.0304 176.1394 176.2484 176.3573 17
6.4663
 [65] 176.5753 176.6843 176.7933 176.9023 177.0113 177.1202 177.2292 17
7.3382
 [73] 177.4472 177.5562 177.6652 177.7742 177.8832 177.9921 178.1011 17
8.2101
```

```
 [81] 178.3191 178.4281 178.5371 178.6461 178.7551 178.8640 178.9730 17
9.0820
 [89] 179.1910 179.3000 179.4090 179.5180 179.6270 179.7359 179.8449 17
9.9539
 [97] 180.0629 180.1719 180.2809 180.3899
```

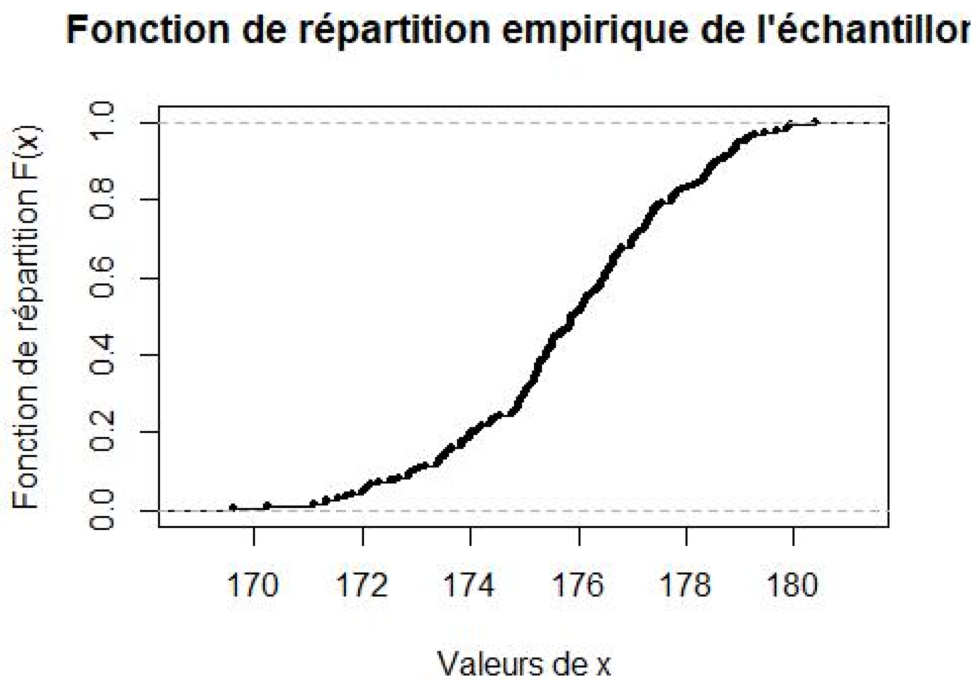
Représentons l'histogramme de cet échantillon et la densité de la loi normale

```
hist(echantillon, breaks = 15, freq = FALSE, main = "Histogramme et den
sité de loi normale", ylab = "Densité")

# Tracer la densité de la loi normale
curve(dnorm(x, mean = μ, sd = σ), add = TRUE, col = "blue", lwd = 2)
```



3. **Représentons graphiquement la fonction de répartition empirique de cet échantillon.**

```
plot(ecdf(echantillon),pch = 19, cex =.5, verticals = TRUE, main = "Fon
ction de répartition empirique de l'échantillon",, xlab = "Valeurs de x
", ylab = "Fonction de répartition F(x)")
```



On constacte que la valeur minimamale de la fonction de répartition F(x) est 0 et sa valeur maximale est 1

4. Créons une fonction appelée LV de la variable $\theta = (\mu, \sigma^2)$ qui calcule la Log vraisemblance en θ de l'échantillon précédant.

   echanillon : est l'échantillon de données

   mu : est la moyenne de la distribution normale

   sigma_squared : est l'écart type de la distribution

```
LV <- function(mu, sigma_squared, echantillon) {
        n <- length(echantillon)
        LV_value <- -0.5*n*log(2 * pi)-0.5*n*log(sigma_squared)-0.5 *
                        sum((echantillon - mu)^2) / sigma_squared
        return(LV_value)
    }
```

5. **Calculer LV (θ) pour θ = (165, 4), pour θ = (165, 6), pour θ = (180, 4) et pour θ = (180, 6). Lequel choisir ?**

- Pour $\theta_1 = (165, 4)$

```
LV_θ1 <- LV(165, 4, echantillon)
print(paste("LV pour θ₁ = (165, 4) est : ", LV_θ1))

[1] "LV pour θ₁ = (165, 4) est :  -3393.44606572268"
```

- Pour θ_2 = (165, 6)

```
LV_θ2 <- LV(165, 6, echantillon)
print(paste("LV pour θ₂ = (165, 6) est : " ,LV_θ2))

[1] "LV pour θ₂ = (165, 6) est :  -2410.31626887691"
```

- Pour θ_3 = (180, 4)

```
LV_θ3 <- LV(180, 4, echantillon)
print(paste("LV pour θ₃ = (180, 4)est : ",LV_θ3 ))

[1] "LV pour θ₃ = (180, 4)est :  -856.279506686398"
```

- Pour θ_4 = (180, 6)

```
LV_θ4 <- LV(180, 6, echantillon)
print(paste("LV pour θ₄ = (180, 6) est : ", LV_θ4 ))

[1] "LV pour θ₄ = (180, 6) est :  -718.871896186056"
```

Notre choix portera sur le θ dont la log de vraissemblance est maximale :

```
print(paste("LV pour θ₁ = (165, 4) est : ",LV_θ1))

[1] "LV pour θ₁ = (165, 4) est :  -3393.44606572268"

print(paste("LV pour θ₂ = (165, 6) est : ",LV_θ2))

[1] "LV pour θ₂ = (165, 6) est :  -2410.31626887691"

print(paste("LV pour θ₃ = (180, 4) est : ",LV_θ3))

[1] "LV pour θ₃ = (180, 4) est :  -856.279506686398"

print(paste("LV pour θ₄ = (180, 6) est : ", LV_θ4))

[1] "LV pour θ₄ = (180, 6) est :  -718.871896186056"
```

La valeur maximale de la log de vraissemblance est -2438.04506255147. Donc notre choix se porte sur le θ_2 = (165, 6).

6. **Cherchons l'estimateur du maximum de vraisemblance, θ_{EMV} de θ.**

Calculons la moyenne μ du maximum de vraisemblance

```
mu_EMV <- mean(echantillon)
```

Calculons la variance σ^2 du maximum de vraisemblance

```
variance_EMV <- var(echantillon)
```

Les résultats du maximum de la vraisemblance

```
print(paste("μ = ", mu_EMV))

[1] "μ =  175.882888745382"
```

```
print(paste("σ² = ", variance_EMV))

[1] "σ² =  4.42601957229587"
```

7. **Regénérons un échantillon de taille n = 10000 et calculons θ_{EMV}**

Générons un échantillon de taille n_new = 10000

```
n_new <- 10000
u1_new <- runif(n_new)
u2_new <- runif(n_new)
z1_new <- sqrt(-2 * log(u1_new)) * cos(2 * pi * u2_new)
echantillon_new <- μ + σ * z1_new
```

Calculons l'estimateur du maximum de vraisemblance pour la moyenne μ_{EMV}

```
mu_EMV_new <- mean(echantillon_new)
```

Calculons l'estimateur du maximum de vraisemblance pour la variance σ^2_{EMV}

```
variance_EMV_new <- var(echantillon_new)
```

Les résultats du maximum de la vraisemblance

```
print(paste(" μ = ", mu_EMV_new))

[1] " μ =  175.975141076189"

print(paste(" σ² = ", variance_EMV_new))

[1] " σ² =  5.03219463349084"
```

8. **Visualiser la loi limite de l'estimateur du maximum de vraisemblance, pour cela générons 100 échantillon de taille n = 100**

Définissons d'abord le nombre d'échantillons

```
n_samples <- 100
```

La taille de chaque échantillon

```
n_sample <- 100
```

Initialisons les vecteurs pour stocker les estimateurs du maximum de vraisemblance

```
mu_EMV_samples <- numeric(n_samples)
variance_EMV_samples <- numeric(n_samples)
```

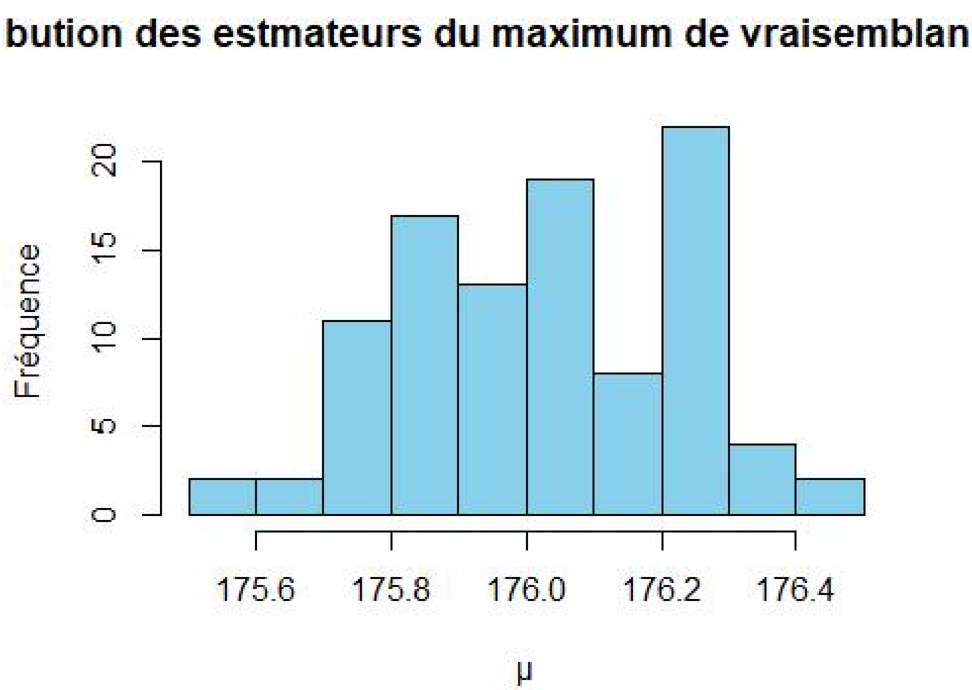
Générons les échantillons puis calculons les estimateurs du maximum de vraisemblance en fin de les stocker

```
for (i in 1:n_samples) {
  u1_sample <- runif(n_sample)
  u2_sample <- runif(n_sample)
```

```
  z1_sample <- sqrt(-2 * log(u1_sample)) * cos(2 * pi * u2_sample)
  echantillon_sample <- μ + σ * z1_sample

  mu_EMV_samples[i] <- mean(echantillon_sample)
  variance_EMV_samples[i] <- var(echantillon_sample)
}
```

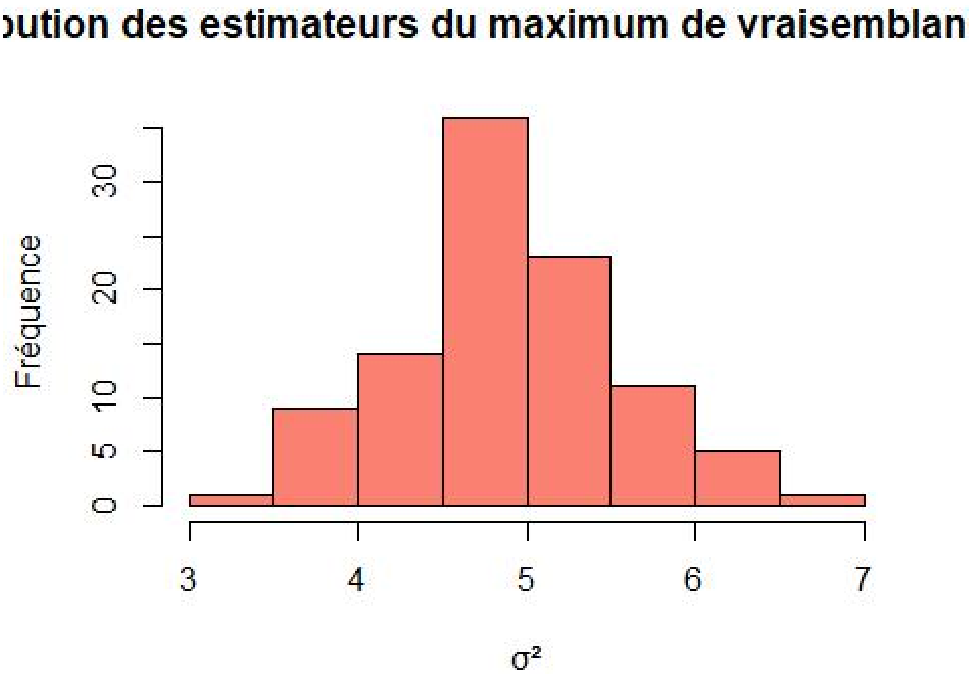
Traçons la distribution des estimateurs du maximum de vraisemblance de μ

```
hist(mu_EMV_samples, main = "Distribution des estmateurs du maximum de vraisemblance pour μ", xlab = "μ",ylab = "Fréquence", col = "skyblue")
```



Traçons la distribution des l'estimateurs du maximum de vraisemblance de σ^2

```
hist(variance_EMV_samples, main = "Distribution des estimateurs du maxi
mum de vraisemblance pour σ²", xlab = "σ²",ylab = "Fréquence", col = "s
almon")
```



9. **Effectuons une estimation par intervalle de confiance à 95% de la moyenne μ**

Calculons d'abord la moyenne et de l'écart-type de l'échantillon

```
moyenne_echantillon <- mean(echantillon)
ecart_type_echantillon <- sd(echantillon)

# Taille de l'échantillon
n <- length(echantillon)
```

Déterminons le quantile de la distribution normale standard d'ordre α = 0,05

```
α <- 0.05
z_alpha_sur_2 <- qnorm(1 - α / 2)
print(paste(" z_alpha_sur_2 = ", z_alpha_sur_2))

[1] " z_alpha_sur_2 =  1.95996398454005"
```

Déterminons l'intervalle de confiance pour la moyenne μ au seuil α = 0,05

```
borne_inferieure<- moyenne_echantillon-z_alpha_sur_2*ecart_type_echanti
llon/sqrt(n)

borne_superieure<- moyenne_echantillon+z_alpha_sur_2*ecart_type_echanti
llon/sqrt(n)

print(paste("Intervalle de confiance à 95% pour la moyenne μ est :", pa
```

```
ste("[", round(borne_inferieure, 2), ",", round(borne_superieure, 2), "]
", sep = "")))

[1] "Intervalle de confiance à 95% pour la moyenne µ est : [175.59,176.
17]"
```

10. **Calculons un intervalle de confiance de niveau 95% pour σ²**

Déterminons d'abord la variance et la taille de l'échantillon

```
variance_echantillon <- var(echantillon)

n <- length(echantillon)
```

Déterminons le quantiles de la distribution du chi-2 au seuil $\alpha = 0{,}02$ et de degré de liberté df=n-1

```
α <- 0.05
df<-n-1
chi2_alpha_sur_2 <- qchisq(α / 2, df)
chi2_1_moins_alpha_sur_2 <- qchisq(1 - α / 2, df)
print(paste("chi2_alpha_sur_2 :",round(chi2_alpha_sur_2,2)))

[1] "chi2_alpha_sur_2 : 161.83"

print(paste("chi2_1_moins_alpha_sur_2 :",round(chi2_1_moins_alpha_sur_2,
2)))

[1] "chi2_1_moins_alpha_sur_2 : 239.96"
```

Déterminons l'intervalle de confiance pour la variance σ^2

```
borne_inferieure <- ((n - 1) * variance_echantillon) / chi2_1_moins_alp
ha_sur_2
borne_superieure <- ((n - 1) * variance_echantillon) / chi2_alpha_sur_2

print(paste("Intervalle de confiance à 95% pour la variance σ² est :",
paste("[", round(borne_inferieure, 2), ",", round(borne_superieure, 2),
 "]", sep = "")))

[1] "Intervalle de confiance à 95% pour la variance σ² est : [3.67,5.44]
"
```

11. **Vérifier ensuite ces deux calculs précédents en simulant 10000 fois un tel échantillon**

Pour vérifier les intervalles de confiance calculés précédemment, nous allons simuler 10 000 échantillons, calculer les intervalles de confiance pour chaque échantillon, et vérifier combien d'entre eux contiennent réellement les vraies valeurs des paramètres ($\mu = 176$ et σ=5).

Initialisation du compteur pour les IC de mu et sigma2

```
compteur_mu <- 0
compteur_sigma2 <- 0
```

Nombre d'échantillons à simuler

```
nb_simulations <- 10000
```

Taille de l'échantillon

```
n <- length(echantillon)
```

Calculons les intervalles de confiance pour chaque échantillon simulé

```
for (i in 1:nb_simulations) {
  # Générer un échantillon
  u1 <- runif(n)
  u2 <- runif(n)
  z1 <- sqrt(-2 * log(u1)) * cos(2 * pi * u2)
  echantillon_simule <- μ + σ * z1

  # Calculer la moyenne et la variance de l'échantillon
  moyenne_echantillon <- mean(echantillon_simule)
  variance_echantillon <- var(echantillon_simule)

  # Calculer les quantiles pour l'intervalle de confiance de la moyenne
  borne_inferieure_mu <- moyenne_echantillon - z_alpha_sur_2 * ecart_ty
pe_echantillon / sqrt(n)
  borne_superieure_mu <- moyenne_echantillon + z_alpha_sur_2 * ecart_ty
pe_echantillon / sqrt(n)

  #L'intervalle de confiance de la moyenne
  IC_mu<-c(borne_inferieure_mu,borne_superieure_mu)

  # Calculer les quantiles pour l'intervalle de confiance de la varianc
e
  borne_inferieure_sigma2 <-((n - 1)*variance_echantillon)/chi2_1_moins
_alpha_sur_2
  borne_superieure_sigma2 <- ((n - 1) * variance_echantillon) / chi2_al
pha_sur_2

  #L'intervalle de confiance de la variance
  IC_sigma2<-c(borne_inferieure_sigma2,borne_superieure_sigma2)

  # Vérification si les vraies valeurs des paramètres sont dans les IC
  if (μ >= IC_mu[1] && μ <= IC_mu[2]) {
    compteur_mu <- compteur_mu + 1
  }

  if (σ*σ >= IC_sigma2[1] && σ*σ <= IC_sigma2[2]) {
    compteur_sigma2 <- compteur_sigma2 + 1
```

```r
  }

}
# Taux de couverture des IC pour μ et σ²
taux_couverture_mu <- (compteur_mu/nb_simulations)*100
taux_couverture_sigma2 <- (compteur_sigma2/nb_simulations)*100


# Affichage des résultats
print(paste("Taux de couverture des IC pour μ est : ", paste(taux_couve
rture_mu),paste("%")))
```

```
[1] "Taux de couverture des IC pour μ est :  93.2 %"
```

```r
print(paste("Taux de couverture des IC pour σ² est : ",paste(taux_couve
rture_sigma2),paste("%")))
```

```
[1] "Taux de couverture des IC pour σ² est :  95.23 %"
```

12. **Calculer la région de confiance simultanée des deux paramètres inconnus de θ**

Déterminons les bornes des intervalles de confiance pour μ au seuil $\alpha = 0{,}05$

```r
borne_inferieure_mu<-mean(echantillon)-z_alpha_sur_2*ecart_type_echanti
llon/sqrt(n)
borne_superieure_mu<-mean(echantillon)+z_alpha_sur_2*ecart_type_echanti
llon/sqrt(n)

print(paste("  Borne inférieure : ", round(borne_inferieure_mu, 2)))
```

```
[1] "  Borne inférieure :  175.59"
```

```r
print(paste("  Borne supérieure : ", round(borne_superieure_mu, 2)))
```

```
[1] "  Borne supérieure :  176.17"
```

Déterminons les bornes des intervalles de confiance pour σ^2 au seuil $\alpha = 0{,}05$

```r
borne_inferieure_sigma2 <- ((n - 1)*variance_echantillon)/ chi2_1_moins
_alpha_sur_2
borne_superieure_sigma2 <- ((n - 1) * variance_echantillon) / chi2_alph
a_sur_2

print(paste("  Borne inférieure : ", round(borne_inferieure_sigma2, 2)))
```

```
[1] "  Borne inférieure :  3.75"
```

```r
print(paste("  Borne supérieure : ", round(borne_superieure_sigma2, 2)))
```

```
[1] "  Borne supérieure :  5.57"
```

Région de confiance simultanée pour μ et σ^2

```
print(paste("Région de confiance à 95% pour μ est : ", paste("[", round(borne_inferieure_mu, 2), ",", round(borne_superieure_mu, 2), "]", sep = "")))

[1] "Région de confiance à 95% pour μ est :  [175.59,176.17]"

print(paste("Région de confiance à 95% pour σ² est : ", paste("[", round(borne_inferieure_sigma2, 2), ",", round(borne_superieure_sigma2, 2), "]", sep = "")))

[1] "Région de confiance à 95% pour σ² est :  [3.75,5.57]"
```

Fin de l'exercice 1

Exercice 2:

1. **Générons un échantillon de taille n = 100 d'une loi gaussienne sur R^2 de moyenne m = $(2, 3)^t$ et de matrice de variance covariance (2 1,1 3).**

Pour générer les échantillons à partir d'une distribution multivariée nous allons d'abord chargé la bibliothèque **MASS**, qui contient la fonction **mvronorm()**

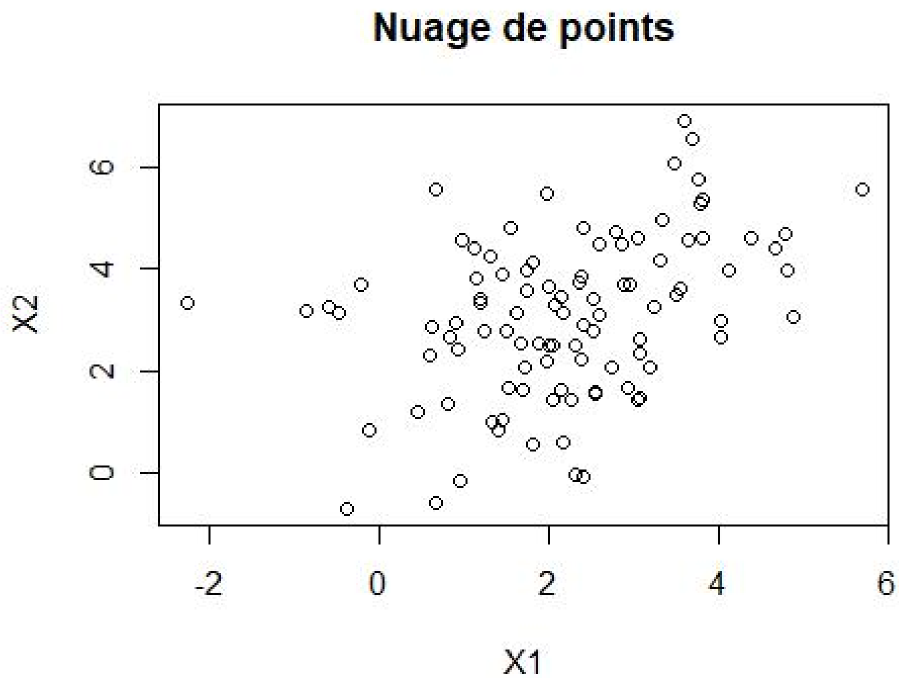
```
library(MASS)

# Définition des paramètres
m <- c(2, 3)
Gamma <- matrix(c(2, 1, 1, 3), nrow = 2)

# Génération de l'échantillon
set.seed(123) # Pour la reproductibilité
n <- 100
echantillon <- mvrnorm(n, mu = m, Sigma = Gamma)
echantillon

             [,1]        [,2]
  [1,]  2.1499309  1.65407595
  [2,]  1.5129388  2.78632786
```

```
 [3,]  3.8054002  5.36957907
 [4,]  2.4180510  2.89929183
 [5,]  3.0809063  2.62105933
 [6,]  3.7600927  5.74720478
 [7,]  3.2458207  3.26068045
 [8,]  2.4028807 -0.07775688
 [9,]  1.6933737  1.65365583
[10,]  0.6353414  2.84687492
[11,]  3.7994288  4.62502198
[12,]  1.7518495  3.95793362
[13,]  4.0186542  2.64855533
[14,]  2.1662447  3.14474921
[15,]  0.9247517  2.42164146
[16,]  3.4857598  6.07740920
[17,]  2.3921743  3.87085047
[18,]  0.6740889 -0.57803149
[19,]  3.5510602  3.60967152
[20,]  2.5513374  1.60206103
[21,]  0.8145297  1.34493455
[22,]  2.7294997  2.06173766
[23,]  1.4645530  1.03670876
[24,]  1.5272010  1.66235554
[25,] -0.4689013  3.12823461
[26,]  0.9652566 -0.13205430
[27,]  2.6024005  4.50104482
[28,]  2.0754123  3.29634537
[29,]  1.8237197  0.56399566
[30,]  3.3251230  4.98464434
[31,]  0.9819134  4.58281513
[32,]  1.2534245  2.80160916
[33,]  2.8538927  4.47382709
[34,]  3.3006303  4.15973243
[35,]  4.8748283  3.06036954
[36,]  1.5573030  4.81344819
[37,]  4.0145577  2.99353238
[38,]  1.1981408  3.35713746
[39,] -0.2150662  3.68483290
[40,]  3.0634222  1.49200994
[41,]  0.6035087  2.30966707
[42,]  2.0542802  2.50153582
[43,]  2.3067478 -0.01909283
[44,]  5.6836236  5.57332838
[45,]  4.8094982  3.96471978
[46,]  1.4077979  0.85465386
[47,]  3.0588707  1.44470401
[48,]  0.8454279  2.67009172
[49,]  0.6798562  5.55994878
[50,]  3.2036614  2.06967744
[51,]  1.4655797  3.89672735
[52,]  1.2024110  3.42910462
```

```
 [53,]  1.6249270  3.13594663
 [54,]  4.3769789  4.59123400
 [55,]  1.8936816  2.56086910
 [56,]  3.7968659  5.28040713
 [57,] -0.1117423  0.84201199
 [58,]  2.9570525  3.71574511
 [59,]  1.1468809  3.80420314
 [60,]  2.5905224  3.11789710
 [61,]  1.3269280  4.26488105
 [62,]  2.5468536  1.53879653
 [63,]  2.9269479  1.68204036
 [64,] -2.2596153  3.35498325
 [65,]  1.3450664  1.00817321
 [66,]  2.0053010  3.67543445
 [67,]  1.8116401  4.11864035
 [68,]  2.5367849  2.78676977
 [69,]  2.4054054  4.81169842
 [70,]  3.6811202  6.54513932
 [71,]  1.7243493  2.07238241
 [72,] -0.3744619 -0.69596041
 [73,]  3.0398058  4.60626440
 [74,] -0.8376527  3.16794468
 [75,]  2.0533275  1.42860777
 [76,]  4.1215676  3.98204639
 [77,]  1.6774386  2.56258211
 [78,]  0.4688015  1.21672491
 [79,]  1.7447800  3.56314154
 [80,]  2.3194740  2.49198370
 [81,]  3.0690903  2.35215496
 [82,]  1.1220952  4.40408816
 [83,]  1.9789904  2.18416365
 [84,]  3.5098894  3.50770679
 [85,]  2.0157930  2.49721644
 [86,]  2.5289579  3.41497309
 [87,]  1.9869187  5.46069127
 [88,]  2.3504442  3.75650897
 [89,]  0.9200146  2.93866249
 [90,]  3.6480996  4.55023034
 [91,]  2.7790585  4.74005750
 [92,]  2.8730829  3.68665799
 [93,]  2.1441482  3.44473190
 [94,]  2.2674573  1.43066164
 [95,]  4.6714540  4.39146201
 [96,] -0.5974730  3.26310534
 [97,]  3.5866242  6.91043760
 [98,]  4.7838820  4.70648785
 [99,]  2.3754656  2.24090750
[100,]  2.1590592  0.60654911
```

2. **Représentons le nuage de points associé**

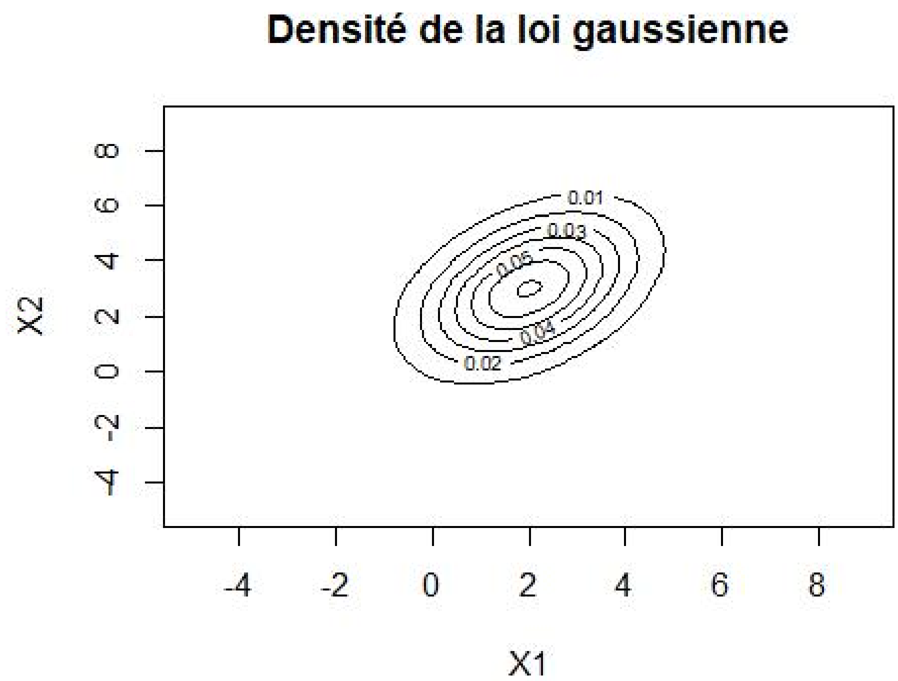
```r
plot(echantillon, main = "Nuage de points", xlab = "X1", ylab = "X2")
```



3. **Représentons la densité d'une loi gaussienne non dégénérée sur R^2**

Pour visualisation la densité d'une loi gaussienne non dégénérée nous allons d'abord chargé la bibliothèque **ggplot2**, qui contient la fonction **contour()**

```r
library(ggplot2) # Pour la visualisation des données

Warning: le package 'ggplot2' a été compilé avec la version R 4.3.3

library(mvtnorm)

Warning: le package 'mvtnorm' a été compilé avec la version R 4.3.3

# Générer une grille de points dans le plan R^2
x1 <- seq(-5, 9, length.out = 100)
x2 <- seq(-5, 9, length.out = 100)
grid <- expand.grid(x1, x2)
# Calculer la densité pour chaque point de la grille
densite <- matrix(dmvnorm(grid, mean = m, sigma = matrix(c(2, 1, 1, 3),
 nrow = 2)), nrow = length(x1))
# Tracer la densité avec des lignes de niveau
contour(x1, x2, matrix(densite, nrow = length(x1)), main = "Densité de
la loi gaussienne", xlab = "X1", ylab = "X2")
```



4. **Donnons une estimation de θ = (m,) par la méthode des moments et par la méthode du maximum de vraisemblance.**

- Estimation de θ par la méthode des moments

L'estimateur m

```
m_moments <- colMeans(echantillon)
# Affichage
print("Estimation par la méthode des moments:")

[1] "Estimation par la méthode des moments:"

print(paste("m:", toString(m_moments)))

[1] "m: 2.19795270663972, 3.07981225640981"
```

L'estimateur $\boldsymbol{\Gamma}$

```
Gamma_moments <- cov(echantillon)
print("Estimation par la méthode des moments:")

[1] "Estimation par la méthode des moments:"

print("Gamma:")

[1] "Gamma:"

print(Gamma_moments)
```

```
          [,1]      [,2]
[1,] 1.8557381 0.8140193
[2,] 0.8140193 2.4511520
```

- Estimation de θ par la méthode du maximum de vraisemblance

```
library(bbmle)

Warning: le package 'bbmle' a été compilé avec la version R 4.3.3

Le chargement a nécessité le package : stats4

# Définition de la fonction de log-vraisemblance pour la distribution n
ormale multivariée
log_likelihood <- function(mu1, mu2, sigma1, sigma2, rho) {
  Sigma <- matrix(c(sigma1^2, rho * sigma1 * sigma2, rho * sigma1 * sig
ma2, sigma2^2), nrow = 2)
  -sum(log(dmvnorm(echantillon, mean = c(mu1, mu2), sigma = Sigma)))
}

# Estimation par la méthode du maximum de vraisemblance
fit_mle <- mle2(log_likelihood, start = list(mu1 = 0, mu2 = 0, sigma1 =
 1, sigma2 = 1, rho = 0))

# Affichage des résultats
print("Estimation par la méthode du maximum de vraisemblance:")

[1] "Estimation par la méthode du maximum de vraisemblance:"

print("m:")

[1] "m:"

print(c(coef(fit_mle)[["mu1"]], coef(fit_mle)[["mu2"]]))

[1] 2.197923 3.079648

print("Gamma:")

[1] "Gamma:"

print(matrix(c(coef(fit_mle)[["sigma1"]], coef(fit_mle)[["rho"]] * sqrt
(coef(fit_mle)[["sigma1"]]) * sqrt(coef(fit_mle)[["sigma2"]]),
               coef(fit_mle)[["rho"]] * sqrt(coef(fit_mle)[["sigma1"]])
 * sqrt(coef(fit_mle)[["sigma2"]]), coef(fit_mle)[["sigma2"]]), nrow =
2))

          [,1]      [,2]
[1,] 1.3553776 0.5545328
[2,] 0.5545328 1.5579128
```

5. **Comparons les valeurs estimées aux valeurs théoriques**

```r
# Affichage des valeurs estimées et théoriques
print("Comparaison des valeurs estimées avec les valeurs théoriques:")

[1] "Comparaison des valeurs estimées avec les valeurs théoriques:"

print("m estimé:")

[1] "m estimé:"

print(coef(fit_mle)[["mu1"]])

[1] 2.197923

print(coef(fit_mle)[["mu2"]])

[1] 3.079648

print("m théorique:")

[1] "m théorique:"

print(m)

[1] 2 3

print("Gamma estimé:")

[1] "Gamma estimé:"

print(matrix(c(coef(fit_mle)[["sigma1"]], coef(fit_mle)[["rho"]] * sqrt
(coef(fit_mle)[["sigma1"]]) * sqrt(coef(fit_mle)[["sigma2"]]),
               coef(fit_mle)[["rho"]] * sqrt(coef(fit_mle)[["sigma1"]])
 * sqrt(coef(fit_mle)[["sigma2"]]), coef(fit_mle)[["sigma2"]]), nrow =
2))

          [,1]      [,2]
[1,] 1.3553776 0.5545328
[2,] 0.5545328 1.5579128

print("Gamma théorique:")

[1] "Gamma théorique:"

print(Gamma)

     [,1] [,2]
[1,]    2    1
[2,]    1    3
```

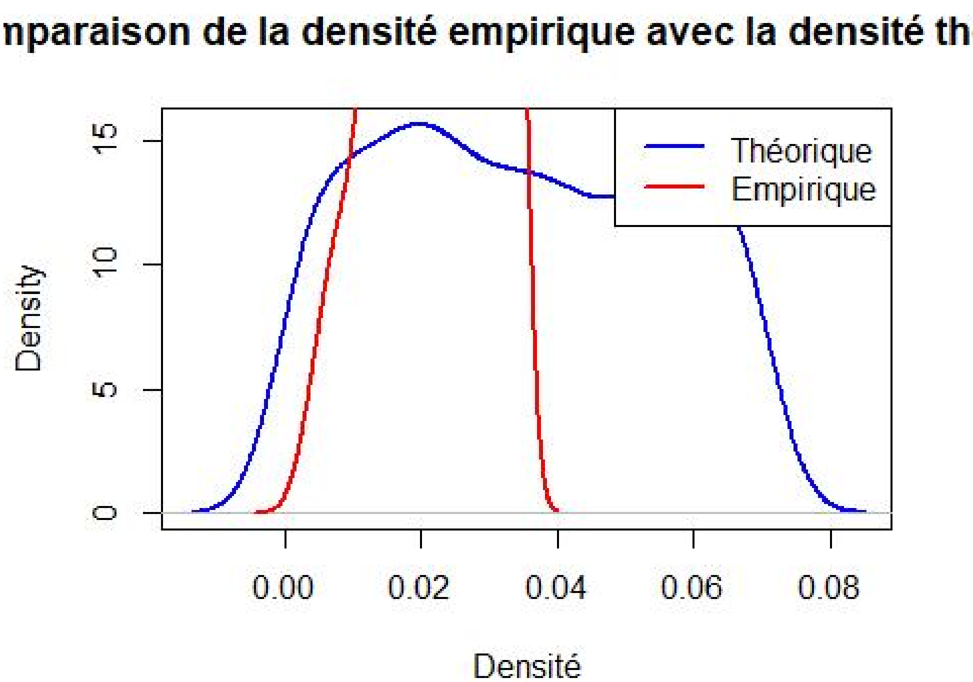
6. **Recommencer avec d'autre valeurs de n. Générer un nombre suffisant d'échantillons pour pouvoir comparer la loi empirique de l'estimateur θn à la loi normale N (m, Γ).**

```r
library(MASS) # Pour la fonction mvrnorm
library(mvtnorm) # Pour la fonction dmvnorm
```

```r
library(ggplot2) # Pour les graphiques

# Définition des paramètres
m <- c(2, 3)
Sigma <- matrix(c(2, 1, 1, 3), nrow = 2)
n_values <- c(100, 500, 1000) # Différentes valeurs de n à tester

# Répéter pour différentes valeurs de n
for (n in n_values) {
  # Générer plusieurs échantillons de taille n
  echantillons <- replicate(1000, mvrnorm(n, mu = m, Sigma = Sigma), simplify = FALSE)

  # Calculer les estimateurs pour chaque échantillon
  estimations <- lapply(echantillons, function(ech) {
    list(mu = colMeans(ech), Sigma = cov(ech))
  })

  # Comparaison des estimateurs avec les valeurs théoriques
  moyennes_emp <- sapply(estimations, function(est) est$mu)
  sigmas_emp <- sapply(estimations, function(est) est$Sigma)

  # Calcul de la distance entre les estimateurs et les valeurs théoriques
  distances_moyennes <- sqrt(rowSums((moyennes_emp - m)^2))
  distances_sigmas <- apply(sigmas_emp, 2, function(S_emp) sqrt(sum((S_emp - Sigma)^2)))

  # Affichage des résultats
  cat("Pour n =", n, ":\n")
  cat("Distance moyenne entre les moyennes empiriques et théoriques :", mean(distances_moyennes), "\n")
  cat("Distance moyenne entre les matrices de variance-covariance empiriques et théoriques :", mean(distances_sigmas), "\n\n")
}

Pour n = 100 :
Distance moyenne entre les moyennes empiriques et théoriques : 4.694381

Distance moyenne entre les matrices de variance-covariance empiriques et théoriques : 0.5673137

Pour n = 500 :
Distance moyenne entre les moyennes empiriques et théoriques : 2.207599

Distance moyenne entre les matrices de variance-covariance empiriques et théoriques : 0.24932

Pour n = 1000 :
```

```
Distance moyenne entre les moyennes empiriques et théoriques : 1.585638

Distance moyenne entre les matrices de variance-covariance empiriques e
t théoriques : 0.1789362
```

```r
# Comparaison de la loi empirique avec la loi normale
echantillon_test <- mvrnorm(n = 1000, mu = m, Sigma = Sigma)
densite_theorique <- dmvnorm(as.matrix(echantillon_test), mean = m, sig
ma = Sigma)
densite_empirique <- apply(echantillons[[1]], 1, function(x) mean(dmvno
rm(as.matrix(echantillon_test), mean = x, sigma = Sigma)))
plot(density(densite_theorique), main = "Comparaison de la densité empi
rique avec la densité théorique", xlab = "Densité", col = "blue", lwd =
 2)
lines(density(densite_empirique), col = "red", lwd = 2)
legend("topright", legend = c("Théorique", "Empirique"), col = c("blue",
 "red"), lty = 1, lwd = 2)
```



Fin de l'exercice 2