Gazdasági és Pénzügyi modellek

2023.05.31. vizsga

Rontó Eszter

QTFL19

0.FELADAT

ax = 113 , ay = 116 , az = 102 , av = 108 , ss = 569 ,

ev = 2013 , reszveny = TSLA

1.FELADAT

A **set.seed(ss)** beállítja a random szám generátor kezdőértékét, hogy a generált minták ismételhetőek legyenek.
Az **nx** változó értékét beállítjuk 800-ra, ami a generált minták számát jelenti.
A **v** változóban egy 2x2-es mátrixot hozunk létre a megadott értékek alapján.
A **w** változóba elvégzünk egy Cholesky-faktorizációt a **v** mátrixon, amely előkészíti a transzformációhoz szükséges mátrixot.
A **z1** és **z2** változóban generálunk **nx** darab random mintát az előre meghatározott eloszlásból és transzformáljuk őket.
A **zm** változóban létrehozunk egy 2 oszlopból álló mátrixot a **z1** és **z2** értékekből.
A **zn** változóba kiszámítjuk a minták végső értékét a **zm** és **w** mátrixok segítségével.
Így a **zn** változó tartalmazza a kért **nx** elemű kétdimenziós mintarealizációt. --> **print(zn)** (amit ki iratunk)

```
> print(zn)
              [,1]          [,2]
  [1,] -336.81979117  -61.34537765
  [2,]  107.47448189   50.01090566
  [3,]  -60.26161982   -7.76107722
  [4,]  163.08922379   -4.65329744
  [5,] -233.92697137   18.75255471
  [6,]  -43.45902834   -6.27498745
  [7,]  -12.38703815 -279.60985872
  [8,]  -45.99879452  -35.60463113
  [9,]  -25.45360838  177.64987217
 [10,] -147.52780037  136.85666372
 [11,]   17.23321915  165.62704847
 [12,]  153.72336032   87.41617362
 [13,] -202.18196408    8.04122225
 [14,] -162.58046936 -153.24597271
 [15,] -135.37845690  100.66921769
 [16,] -113.81174494  -10.83909018
 [17,]  -26.72009828    5.06299979
 [18,]  121.53441752  -75.62465727
 [19,]   17.91717707  -67.77693638
 [20,]  159.15524505   47.18937148
 [21,]   69.67139154  -81.48144054
 [22,]  -18.29470275   -6.85049889
 [23,] -102.85331201  -45.73126808
 [24,]   -5.10683329  172.47407505
 [25,]   63.07966286  -38.09939755
 [26,]   36.98426830   74.28000144
 [27,]   -6.87133674   31.13997533
 [28,]  -23.11765184  158.45917810
 [29,]   25.70310749   50.03829705
 [30,]   55.54142134  124.79946932
 [31,]  -91.48228566  -58.59123686
 [32,]   51.56345204   64.13969013
 [33,]  -33.99680767  -58.95862473
 [34,]   23.00156290  -53.43992854
 [35,]  164.72157862  -10.81659408
 [36,]  143.02283768  -11.20401099
 [37,]  -33.38320396  -90.10919019
 [38,]   59.90137072  -98.34734366
 [39,] -104.72149433   24.58954843
 [40,]   67.91159216   -8.49017997
 [41,] -129.83716105   64.63869587
 [42,] -203.17941242  147.90131247
 [43,]  243.41431589  197.88202583
 [44,]  130.54256629  222.91032246
 [45,] -152.85399380 -173.91937767
 [46,]   66.92862955   -3.81003282
 [47,]  -40.84014172   42.61955458
 [48,] -124.03587247  -22.80769273
 [752,]   61.42632615  124.01253348
 [753,]  -34.66313982 -107.08111501
 [754,]   70.09121024  143.04922464
 [755,]  -81.47557335  141.96657665
 [756,] -125.00658785  135.76121692
 [757,]   66.53329163  -67.00905561
 [758,] -160.73957901  152.94220029
 [759,]   24.08763229  -45.73174146
 [760,]   36.84031196   47.56965831
 [761,]   47.55767226   28.86114213
 [762,]  -65.68482094  -74.18564603
 [763,]   73.83632560  -24.16773895
 [764,]  -62.56311346   86.57687649
 [765,]   63.53689349  -26.14376119
 [766,]   25.16947905  119.94481124
 [767,]   33.17431801   50.34521913
 [768,]  144.82472392   92.85848740
 [769,]   -9.53108163  109.93456742
 [770,]   12.81803088  222.08092681
 [771,]  -23.99272987 -196.04772500
 [772,]  -67.89814391 -130.16352417
 [773,]  221.14476400   28.81279400
 [774,]  -19.60106513  -12.63519393
 [775,]   74.35422376   40.75063687
 [776,]  -51.39062532  -80.98818866
 [777,]   94.19391830  -42.34812965
 [778,] -150.81258207  -76.16129128
 [779,]   -0.08646487  126.47437468
 [780,]   15.57257059  -54.32254047
 [781,]  111.47377553    3.58848585
 [782,]   57.94012410  -78.74737735
 [783,]   -2.66438859  -38.81947171
 [784,]  -32.67976893 -190.03563373
 [785,]  -43.24721358 -286.62896391
 [786,]  113.59803156  -50.11645975
 [787,]    1.32603296  149.49935496
 [788,] -190.15116924   -6.61029656
 [789,]  253.76826150  171.43281430
 [790,]   53.96915188    1.79891751
 [791,]  228.23175588  136.03180664
 [792,] -137.22680695  -28.25388103
 [793,] -137.67452308 -179.16174652
 [794,]   62.74405032  -47.75121908
 [795,]  171.48931333  127.89407395
 [796,] -146.49013835  132.59999026
 [797,]    0.75491633  -64.62285633
 [798,]  155.79341709   15.15876029
 [799,]  -88.36110852   94.12707542
 [800,]   87.07847851  147.40662307
```

Először megbecsüljük a paramétereket a **zn** mintarealizáció alapján. A paraméterek becsléséhez a minta átlagát és a koverencia mátrixát használjuk.

```
> mean_zn = colMeans(zn)
> print(mean_zn)
[1] -2.99509  2.42030
>
> cov_zn = cov(zn)
> print(cov_zn)
             [,1]        [,2]
[1,] 11949.3639   345.2849
[2,]   345.2849 11554.5440
```

A **mean_zn** változóban tároljuk a **zn** mintarealizáció oszlop átlagát, míg a **cov_zn**-ben a minta kovarancia mátrixát tároljuk.

Ezek után megvizsgáljuk, a marginális eloszlást amelyhez a normális eloszlást használjuk.

$$f(x) = (1 / (\sigma * \sqrt{2\pi})) * \exp(-((x - \mu)^2) / (2\sigma^2))$$

```
> marginal_1 = list(mean = mean_zn[1], sd = sqrt(cov_zn[1, 1]))
> marginal_2 = list(mean = mean_zn[2], sd = sqrt(cov_zn[2, 2]))
>
> print(marginal_1)
$mean
[1] -2.99509

$sd
[1] 109.3131

> print(marginal_2)
$mean
[1] 2.4203

$sd
[1] 107.4921
```

A **marginal_1** változóban a perem 1 becslését tároljuk, ahol az átlag (**mean**) a **mean_zn[1]**, a szórás pedig a négyzetgyök a **cov_zn[1, 1**] értékből. A **marginal_2** változóban ugyanezt tesszük a perem 2-re.

A peremek függetlenségének vizsgálatához a korrelációs mátrixot vizsgálhatjuk:

```
> cor_zn = cor(zn)
> print(cor_zn)
             [,1]        [,2]
[1,] 1.00000000 0.02938521
[2,] 0.02938521 1.00000000
```

Ha a korrelációs mátrix diagonális, vagyis a nem-diagonális elemek közelítőleg 0-k, akkor a peremek függetlenek lehetnek.

2.FELADAT

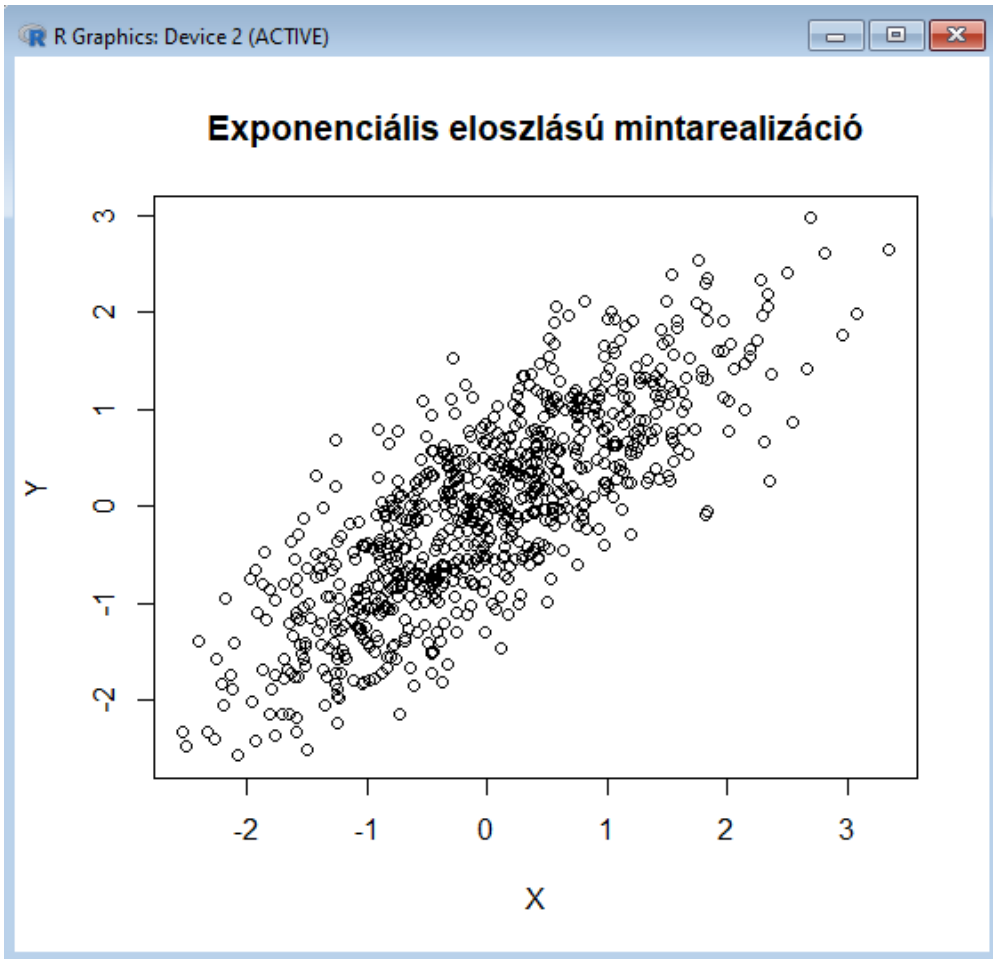
```
> set.seed(569+137)
> n = 800
> rho = 0.8
>
> corr_matrix = matrix(c(1, rho, rho, 1), nrow = 2)
>
> sample_data = MASS::mvrnorm(n, mu = c(0, 0), Sigma = corr_matrix)
>
> plot(sample_data, main = "Exponenciális eloszlású mintarealizáció", xlab = "X", ylab = "Y")
```

A **set.seed(569+137)** beállítja a random generátor seed-jét az adott értékre, hogy ismételhetővé tegye a generálást. Az **n** változóban megadja a mintaelemszámot, míg a **rho** változóban tárolja a korrelációs együtthatót.

A **corr_matrix** létrehoz egy 2x2-es korrelációs mátrixot, amelyben az átlós elemek 1-esek (hiszen az adott változók önmagukkal teljes korrelációban vannak), és a nem átlós elemeket a **rho** értéke határozza meg.

A **MASS::mvrnorm()** függvény segítségével generálunk **n** darab mintát az exponenciális eloszlásból a megadott korrelációs mátrix és középérték (0, 0) alapján.

Végül az **plot()** függvény segítségével ábrázoljuk a kapott mintarealizációt, ahol az x tengelyen az első változót, az y tengelyen pedig a második változót ábrázoljuk.



4. FELADAT

```
> set.seed(569+137)
> n = 800
> rho = 0.8
>
> corr_matrix = matrix(c(1, rho, rho, 1), nrow = 2)
>
> sample_data = MASS::mvrnorm(n, mu = c(0, 0), Sigma = corr_matrix)
>
> plot(sample_data, main = "Exponenciális eloszlású mintarealizáció", xlab = "X", ylab = "Y")
>
>
> set.seed(569+17)
>
> lambda = 2
> time_interval = 1000
>
> poisson_process = rpois(time_interval, lambda)
>
> plot(poisson_process, type = "s", main = "Poisson folyamat", xlab = "Idő", ylab = "Eseménye$
>
> event_times = which(poisson_process > 0)
> interarrival_times = diff(event_times)
> mean_interarrival_time = mean(interarrival_times)
>
> print(mean_interarrival_time)
[1] 1.156431
```

A **set.seed(569+17)** beállítja a random generátor seed-jét az adott értékre, hogy ismételhetővé tegye a generálást. A **lambda** változóban megadja a Poisson eloszlás várható értékét, míg a **time_interval** változóban tárolja az időintervallum hosszát.

A **rpois()** függvény segítségével generálunk egy Poisson folyamatot a megadott időintervallumra és várható értékre.

Az **plot()** függvénnyel ábrázoljuk a kapott folyamatot. A **type = "s"** paraméterrel lépcsőzetes ábrázolást használunk.

A bekövetkezések között eltelt idő várható értékét a **event_times** változóban tárolt bekövetkezési időpontok alapján számítjuk. Az **interarrival_times** változóban tároljuk a bekövetkezések között eltelt időket, majd ezek átlagát vesszük a **mean_interarrival_time** változóban.

Az eredményt a **print()** függvénnyel írjuk ki a konzolra.