Keresztúri Judit Lilla,
Antal Beáta, Illés Ferenc

Bevezetés az R programozásba

Egyetemi jegyzet
Vállalati pénzügyi információs rendszerek
című tantárgyhoz

Budapest, 2017

Kiadó: Budapesti Corvinus Egyetem
ISBN 978-963-503-640-0

Előszó

A jegyzet a Budapesti Corvinus Egyetem Pénzügy Mester szak Vállalati Pénzügyek specializáció Vállalati Pénzügyi Információs Rendszerek tantárgyhoz készült.

A tárgy oktatásának a célja megismertetni a hallgatókat a vállalatok pénzügyi funkcióival, a pénzügyi vezetés feladataival, valamint a vállalati pénzügyi döntéseket támogató információs rendszerekkel. Mivel a tárgy tematikájában az utóbbi pár évben jelentősen megnőtt az R programozási nyelv oktatásának súlya, és ennek alapszintű elsajátítása nélkül a tárgy nem teljesíthető, célszerűnek tűnt a segédanyagokat egy külön jegyzet formájában megjelentetni. A jegyzet célja, hogy a tárgy tematikája mentén haladva összefoglalja az R programnyelv alapjait, illetve azon alkalmazásokat, melyek a tárgy tanulása során előkerülnek, és amelyeket a pénzügyes hallgatóknak érdemes ismerni.

Felesleges hangsúlyozni, hogy a jegyzet nem helyettesíti az órákon való részvételt, csupán segítséget nyújt a vizsgára való felkészülésben azon hallgatók számára, akik a tananyagot már jórészt ismerik. Nem ajánljuk, hogy az olvasó csupán a jegyzet felhasználásával próbája az R-t önállóan megtanulni.

Azon olvasók számára, akik mélyebben el szeretnének merülni a kvantitatív pénzügyi feladatok R programozásában, Daróczi és szerzőtársai (2013): Introduction to R for Quantitative Finance és Berlinger és szerzőtársai (2015): Mastering R for Quantitative Finance című könyveket ajánljuk.

A jegyzet szabadon terjeszthető a forrás megjelölésével, azonban kereskedelmi célokra nem használható. A jegyzetben lévő mintafeladatokhoz felhasznált fájlok a http://web.uni-corvinus.hu/~lkeresz/ weboldalon érhetőek el.

A jegyzetben minden igyekezetünk ellenére maradhattak hibák, hiányosságok, pontatlanságok. Ezekkel kapcsolatos visszajelzéseket, észrevételeket a lilla.kereszturi@uni-corvinus.hu e-mail címen köszönettel fogadunk.

2017. 03. 01.

A szerzők

Tartalomjegyzék

1. Változók létrehozása, értékadás

1.1 Vektorok

- **c(x,y)** → x vektort és y vektort kapcsolja össze (concatenáció), amely se nem oszlop-, se nem sorvektor, *csak számok sorozata*
- **seq(x,y,by=z)** → számsor x-től y-ig z lépésközzel (x,y,z lehet negatív is) [pl. ha x=-100 és y=-50, akkor z=-1 esetén (z=-1 jelentése -1-gyel növekvő, azaz 1-gyel csökkenő) hibára fut a program]
- **seq(x,y, length=z)** → számsor x-től y-ig z hosszúságú vektort állít elő állandó lépésközzel
- **rep(x,y)** → számsor, ami x-et ismétli y-szor (y>0)
- minden utasítást ";"-vel zárunk, kivéve, ha egy sorban csak egy utasítás van, az új sor kezdése helyettesíti a ";"-t

Programkód	Jelentése
`x <- seq(1, 5, 0.1)`	számsor 1-től 5-ig 0,5-ös lépésközzel
`h <- c(174, 170, 160)`	174, 170 és 160 összekapcsolása
`l <-rep(12,10)` `l <-rep(12,times=10)`	12-t 10-szer írja ki (ismétli) kétféleképpen
`l2 <- rep(1:2,5)`	(1,2) ismétlése 5-ször
`l3<-rep(c(5,6), length=3)`	(5,6) vektor ismétlése 3 hosszan, azaz (5,6,5)
`l4<-rep(c(5,6), each=3)`	(5,6) vektor koordinátáinak ismétlése háromszor, azaz (5,5,5,6,6,6)

1.1. FELADAT

Definiáljuk és írassuk ki a következő változókat.

a = 1 b= -1 c = (1,2) d = (1,2,3) u = (2,5,8, ….200)

w = (100, 99, 98,…51) z = (-100,-99,-98,…,99,100)

Programkód	Jelentése
`a <- 1`	a legyen 1
`b <- -1`	b legyen -1
`c <- c(1,2)`	összekapcsolja az 1-et és 2-t
`c <- 1:2`	egyszerűbben összekapcsolja 1-től 2-ig az egész számokat (automatikusan egyesével nő)
`d <- c(1,2,3)`	összekapcsolja az 1-et, 2-t, és 3-at
`d <- 1:3`	egyszerűbben összekapcsolja 1-től 3-ig az egész számokat (automatikusan egyesével nő)

`u <- seq(2,200, by = 3)`	növekvő számsorozat 2-től legfeljebb 200-ig 3-asával léptetve
`w <- seq(100,51, by =-1)`	számsorozat 100-tól 51-ig 1-esével csökkentve
`w <- 100:51`	egyszerűbben létrehozott csökkenő számsorozat 100-tól 51-ig (automatikusan egyesével csökken)
`z <- seq(-100,100, by =1)`	egyesével növekvő számsorozat -100-tól 100-ig
`z <- -100:100`	növekvő számsorozat -100-tól 100-ig (automatikusan egyesével nő)

Számoljuk ki:

a+b a+c b+c w+c

```
a <- 1;
b <- -1;
a+b
```

```
[1] 0
```

```
eredmény:
1+(-1) = 0
```

```
a <- 1;
c <- 1:2;
a+c
```

```
[1] 2 3
```

```
eredmény:
első elem: 1+1=2
második elem: 1+2=3

rövidebb vektor elemeit ciklikusan újra felhasználjuk, amíg el nem érjük a hosszabb vektor hosszát.
```

```
b <- -1;
c <- 1:2;
b+c
```

```
[1] 0 1
```

```
eredmény:
első elem: (-1)+1=0
második elem: (-1)+2=1

(lásd: a+c eset)
```

```
w <- 100:51;
c <- 1:2;
w+c
```

```
[1] 101 101  99  99  97  97  95  95  93  93  91  91  89  89  87  87  85  85
 83  83  81  81  79  79  77  77  75  75
 73  73  71  71  69  69  67  67  65  65
 63  63  61  61  59  59  57  57  55  55
 53  53
```

```
eredmény:
első elem: 100+1=101
második elem: 99+2=101
harmadik elem: 98+1=99
negyedik elem: 97+2=99
stb.

(lásd: a+c eset)
```

Fűzzük egymás után a c és d vektorokat! (megoldás concatenációval)

```
c <- 1:2;
d <- 1:3;
c(c,d)

eredmény:
[1] 1 2 1 2 3
```

1.2. FELADAT

Rakjunk össze az a,b változókból egy 1000 hosszúságú vektort, ami felváltva tartalmazza a +1 és -1 számokat.

1.lépés	rakjuk össze (a,b)-t	`c(a,b)`
2.lépés	1000 hosszúságú vektor → c(a,b)-re 500-szor lesz szükség → 500-szor ismétlődő (a,b) vagy megadjuk a vektor pontos hosszát	`rep(c(a,b),times=50);` `#vagy` `rep(c(a,b),length=1000)`

1.3. FELADAT

Írj egy kódot, ami összeadja a 400-nál nem nagyobb négyzetszámokat!

1.lépés	400-nál nem nagyobb négyzetszám → 400 gyöke 20 → *1-től 20-ig a számok egy vektorba*	`seq(1:20)`
2.lépés	négyzetszámok → 1-től 20-ig a számok négyzete → *két vektor összeszorzása*	`seq(1:20)*seq(1:20)`
3.lépés	1-től 20-ig a *négyzetszámok összege*	`sum(seq(1:20)*seq(1:20))`

1.4. FELADAT

Add össze a természetes számok reciprokát 100-ig váltakozó előjellel!

(Például: $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}\ldots$)!

1.lépés	1-től 100-ig számsor	`seq(1:100)`
2.lépés	a váltakozó előjelhez (+1, -1) ismétlése 50-szer	`rep(c(+1, -1),50)`
3.lépés	egyes tényezők létrehozása a 2. és az 1. lépés hányadosaként	`rep(c(+1, -1),50)/ seq(1:100)`
4.lépés	végül a tényezők összeadása	`sum(rep(c(+1, -1),50)/ seq(1:100))`

(Lásd az egyes lépések eredményeit a következő oldalon.)

1.lépés:

```
seq(1:100)
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14
 [15]  15  16  17  18  19  20  21  22  23  24  25  26  27  28
 [29]  29  30  31  32  33  34  35  36  37  38  39  40  41  42
 [43]  43  44  45  46  47  48  49  50  51  52  53  54  55  56
 [57]  57  58  59  60  61  62  63  64  65  66  67  68  69  70
 [71]  71  72  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96  97  98
 [99]  99 100
```

2.lépés:

```
rep(c(+1,-1),50)
```

```
  [1]  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1
 [20] -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1
 [39]  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1
 [58] -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1
 [77]  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1
 [96] -1  1 -1  1 -1
```

3.lépés:

```
rep(c(+1,-1),50)/seq(1:100)
```

```
  [1]  1.00000000 -0.50000000  0.33333333 -0.25000000
  [5]  0.20000000 -0.16666667  0.14285714 -0.12500000
  [9]  0.11111111 -0.10000000  0.09090909 -0.08333333
 [13]  0.07692308 -0.07142857  0.06666667 -0.06250000
 [17]  0.05882353 -0.05555556  0.05263158 -0.05000000
 [21]  0.04761905 -0.04545455  0.04347826 -0.04166667
 [25]  0.04000000 -0.03846154  0.03703704 -0.03571429
 [29]  0.03448276 -0.03333333  0.03225806 -0.03125000
 [33]  0.03030303 -0.02941176  0.02857143 -0.02777778
 [37]  0.02702703 -0.02631579  0.02564103 -0.02500000
 [41]  0.02439024 -0.02380952  0.02325581 -0.02272727
 [45]  0.02222222 -0.02173913  0.02127660 -0.02083333
 [49]  0.02040816 -0.02000000  0.01960784 -0.01923077
 [53]  0.01886792 -0.01851852  0.01818182 -0.01785714
 [57]  0.01754386 -0.01724138  0.01694915 -0.01666667
 [61]  0.01639344 -0.01612903  0.01587302 -0.01562500
 [65]  0.01538462 -0.01515152  0.01492537 -0.01470588
 [69]  0.01449275 -0.01428571  0.01408451 -0.01388889
 [73]  0.01369863 -0.01351351  0.01333333 -0.01315789
 [77]  0.01298701 -0.01282051  0.01265823 -0.01250000
 [81]  0.01234568 -0.01219512  0.01204819 -0.01190476
 [85]  0.01176471 -0.01162791  0.01149425 -0.01136364
 [89]  0.01123596 -0.01111111  0.01098901 -0.01086957
 [93]  0.01075269 -0.01063830  0.01052632 -0.01041667
 [97]  0.01030928 -0.01020408  0.01010101 -0.01000000
```

4.lépés:

```
sum(rep(c(+1,-1),50)/seq(1:100))
```

```
[1] 0.6881722
```

1.2 Mátrixok

`matrix(x,n,m)`
→ nxm-es mátrix létrehozása x elemeinek felhasználásával (ahol n a mátrix sorainak, míg m az oszlopainak számát jelöli), a mátrix „feltöltése” alapértelmezetten függőleges sorrendben történik.
`matrix(1,nrow = 4,ncol = 5)`
→ 4 soros 5 oszlopos csupa 1-t tartalmazó mátrix feltöltése
`matrix(1,ncol = 5)`
→ 1 soros 5 oszlopos csupa 1-t tartalmazó mátrix feltöltése
`matrix(1:6,ncol = 5)`
→ 5 oszlopból álló mátrix az (1,2,3,4,5,6) vektor felhasználásával, a sorok számát a vektor hosszából meghatározza, úgyhogy minden elemet legalább egyszer felhasznál.

1.5. FELADAT
Definiáljuk a következő vektorokat és mátrixokat:

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{bmatrix} \qquad u = (1,2,3) \qquad v = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad w = [1 \quad 2 \quad 3] \qquad B = \begin{bmatrix} 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix}$$

$$z = (2,3,4)$$

Programkód (másolható)	Jelentése
`A<-matrix(c(2,1,1,1,3,1,1,1,4),3,3)`	A 3x3-as mátrix elemenkénti meghatározása (egyszerűbben: következő 2 sor)
`A <- matrix(1,3,3)` `diag(A) <- 2:4`	3x3-as mátrix, melynek minden eleme 1 A mátrix átlójának elemei 2,3,4
`u <- 1:3`	nem mátrix, hanem vektor
`v <- matrix(1:3, 3,1)`	mátrix, melynek 3 sora, 1 oszlopa van = oszlopvektor
`cbind(1:3)`	3 sora, 1 oszlopa van = oszlopvektor
`w <- matrix(1:3, 1,3)`	mátrix, melynek 1 sora, 3 oszlopa van = sorvektor
`rbind(1:3)`	1 sora, 3 oszlopa van = sorvektor
`B <- matrix(c(2:4,6:8), 3,2)`	B 3x2-es mátrix
`z <- 2:4`	nem mátrix, hanem vektor

Végezzük el a következő műveleteket!

v %*% A ☹ (nincs értelme)	w %*% A [,1] [,2] [,3] [1,] 7 10 15	A %*% v [,1] [1,] 7 [2,] 10 [3,] 15	A %*% w ☹ (nincs értelme)
A %*% u [,1] [1,] 7 [2,] 10 [3,] 15 az u vektor, így automatikusan oszlop- vagy sor vektorrá alakítja, úgy hogy az elvégzett szorzás értelmes legyen, ha lehetséges	u %*% A [,1] [,2] [,3] [1,] 7 10 15 az u vektor, így automatikusan oszlop- vagy sorvektorrá alakítja, úgy hogy az elvégzett szorzás értelmes legyen, ha lehetséges	u * A [,1] [,2] [,3] [1,] 2 1 1 [2,] 2 6 2 [3,] 3 3 12 koordinátánkénti szorzás	A * u [,1] [,2] [,3] [1,] 2 1 1 [2,] 2 6 2 [3,] 3 3 12 koordinátánkénti szorzás
A %*% B [,1] [,2] [1,] 11 27 [2,] 15 35 [3,] 21 45	B %*% A ☹ (nincs értelme)	u * z [1] 2 6 12 koordinátánkénti szorzás	z * u [1] 2 6 12 koordinátánkénti szorzás
u %*% z [,1] [1,] 20 skalár szorzás	z %*% u [,1] [1,] 20 skalár szorzás		

1.6. FELADAT

Szorozd meg az A mátrixot a transzponáltjával, és add össze a főátló elemeit!

$$\begin{matrix} 1 & 10 & 8 \\ 4 & 1 & 10 \\ 8 & 4 & 1 \end{matrix}$$

1.lépés	A mátrix létrehozása	`A<-matrix(c(1,4,8,10,1,4,8,10,1),3,3)`
2.lépés	A mátrix szorzása saját transzponáltjával (mátrix szorzása: %*%) (transzponálás beépített függvény)	`A%*%t(A)` eredmény: [,1] [,2] [,3] [1,] 165 94 56 [2,] 94 117 46 [3,] 56 46 81

3.lépés	összeszorzott mátrix átlójának elemei	`diag(A%*%t(A))` eredmény: [1] 165 117 81
4.lépés	összeszorzott mátrix átlójának elemeinek összeadása	`sum(diag(A%*%t(A)))` eredmény: [1] 363

1.7. FELADAT

Szorozzuk össze az x = (3,5,4,9) és y = (6,3,8,2) vektorokat skalárisan (sor-oszlop). Ezután szorozzuk őket össze diadikusan (oszlop-sor) és számoljuk ki a kapott mátrix elemeinek összegét.

1.lépés	x és y vektorok létrehozása	`x<-c(3,5,4,9);` `y<-c(6,3,8,2);`
2.lépés	x és y vektor összeszorzása Fontos: x és y most nem mátrix	`x%*%y` eredmény: [,1] [1,] 83

3.lépés	x és y felvétele mátrixként	`x<-matrix(c(3,5,4,9),4,1);`
		`y<-matrix(c(6,3,8,2),1,4);`
		`#VAGY`
		`cbind(x);`
		`rbind(y);`

4.lépés	összeszorzott mátrixok elemeinek összege	`sum(cbind(x)%*%rbind(y))` eredmény: [1] 399

1.2.1 Mátrixok indexelése

Szögletes zárójellel: `a[1]` az `a` első eleme; `b <- a[1:4]` esetén `b` az `a` első négy elemét tartalmazó vektor.

Általában az indexelés `a[ind]` alakú, ahol az `ind` az alábbi lehet:

- pozitív egészekből álló tetszőleges hosszú vektor;
- negatív egészekből álló tetszőleges hosszú vektor, ekkor `ind` a kihagyandó elemeket jelenti; pl. `b <- a[-length(a)]` esetén a `b` ugyanaz, mint az `a`, csak hiányzik az utolsó elem;
- az `a`-val megegyező hosszúságú logikai; ekkor `a[ind]` azokból az elemekből áll, ahol `ind` értéke `TRUE`[1].

Az indexelés értékadásnál is használható, pl. `a[a < 0] <- 0` az `a` összes negatív elemét kinullázza.

[1] A logikai vektorokat lásd részletesen a 4. fejezetben.

1.8. FELADAT

Generálj egy vektort, ami 50 elemű u=(50,49,….,1). Számoljuk ki ennek a CF-nak az átlagidejét és konvexitását, ha a kifizetések egész évenként követik egymást, és az első 1 év múlva lesz. Az effektív hozamgörbe 5 %-on vízszintes.

1.lépés	r hozam és u vektorok felvétele	`r<-0.05;` `u<-cbind(seq(50,1));`

2.lépés	cash-flow jelenértékének kiszámítása
	
	idő: 1-től 50-ig (első kifizetés egy év múlva)
		`PCF<-u/(1+r)^(1:50);`
		
		eredmény:
		`[,1]`
		`[1,] 47.61904762`
		`[2,] 44.44444444`
		`[3,] 41.46420473`
		`[4,] 38.66701632`
		stb…

3.lépés	átlagidő kiszámítása
	$D = \sum_{t=1}^{t=50} \frac{t * PCF_t}{\sum_{t=1}^{t=50} PCF_t}$
		`sum((1:50)*PCF/sum(PCF));`
		
		eredmény:
		`[1] 11.67005`

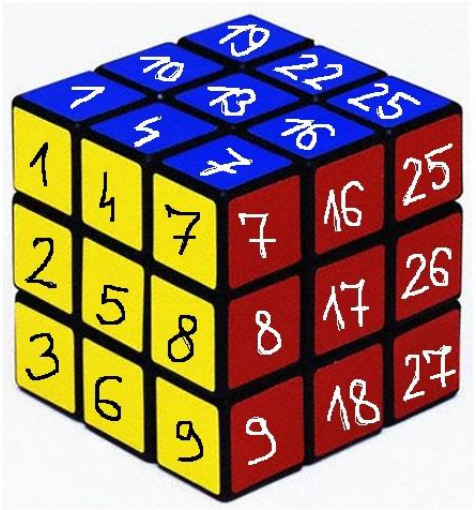
4.lépés	konvexitás
	$D = \sum_{t=1}^{t=50} \frac{t^2 * PCF_t}{\sum_{t=1}^{t=50} PCF_t}$
		`sum((1:50)*(1:50)*PCF/sum(PCF))`
		
		eredmény:
		`[1] 226.7154`

1.3 Array

- tömb megadásának formája: **array(data, dim=c(x,y,z))**, ahol a tömb elemeit a data, a tömb kiterjesztését pedig dim=c(x,y,z) jelöli (x=sorok, y=oszlopok, z=harmadik dimenzió hossza)
- dimenzió száma bármennyi lehet
- kétdimenziós tömb = mátrix
- **str**(array): tömb struktúrájáról ad képet (dimenziók száma, tömb elemeinek felsorolása)

1.lépés	3x3x3-as tömb megadása („kocka”), tömb elemei: számok 1-től 27-ig	`a<- array(1:27,dim=c(3,3,3))`
2.lépés	tömb struktúrája	`str(a)`

```
1.lépés eredménye:
„kocka” első lapja:
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

„kocka” második lapja:
, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

„kocka” harmadik lapja:
, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```



1.3.1 Array indexelése

`array[a,b]`
első dimenzió a. eleme, második dimenzió b. eleme
`array[,b]`
első dimenzió minden eleme, második dimenzió b. eleme (a hiányzó index az adott dimenzió összes elemét jelenti)

```
a[1,,]

eredmény: a „kocka” felső lapja (=1. sor)
     [,1] [,2] [,3]
[1,]    1   10   19
[2,]    4   13   22
[3,]    7   16   25
```

1.9. FELADAT
Vágjuk ki az array bal felső 2X2X2-es sarkát!

```
a[1:2,1:2,1:2]

eredmény:
, , 1

     [,1] [,2]
[1,]    1    4
[2,]    2    5

, , 2

     [,1] [,2]
[1,]   10   13
[2,]   11   14
```

1.4 Data frame

- Megegyező dimenziójú, de akár különböző adattípusú vektorok összessége.
- Az adathalmaz megadásának általános formája: `data.frame`(fejléc1=érték1,fejléc2=érték2, stb.)
- Érték, ha szöveg, akkor idézőjelbe kell tenni.
- `class(x)`: x argumentum osztályozása, típusa (szám, szöveg, dataframe, stb)
- `str(x)`: x dataframe struktúráját adja vissza (változók száma, fejlécek neve, adatok típusa, és az első néhány értéke)

```
d <- data.frame(id=5,company="OTP", value=2/3)
```

eredmény:

	id	company	value
1	5	OTP	0.6666667

```
class(d)
```

```
eredmény:
[1] "data.frame"
(d dataframe típusa)
```

```
str(d)
```

```
eredmény:
'data.frame':  1 obs. of  3 variables:
$ id     : num 5
$ company: Factor w/ 1 level "OTP": 1
$ value  : num 0.667
```

```
class(d[1,1])
```

```
eredmény:
[1] "numeric"
(az 5 típusa)
```

```
class(d[1,2])
```

```
eredmény:
[1] "factor"
(OTP típusa)
```

```
class(d[1,3])
```

```
eredmény:
[1] "numeric"
(0.6666667 típusa)
```

```
d[1,1]+1
```

```
eredmény:
[1] 6
(5+1=6)
```

```
d[1,2]+1
```

```
eredmény:
[1] NA
(OTP+1=Not Available[2])
```

1.4.1 Data frame indexelése

- hagyományos módon: d[1,2]
- $ -> csak list és data.frame objektumokra, csak fejléccel együtt működik

```
d$id
```

```
[1] 5
```

```
d$company
```

```
[1] OTP
Levels: OTP
```

```
d$value
```

```
[1] 0.6666667
```

```
d$value+1
```

```
[1] 1.666667
```

```
d[,1]
```

```
[1] 5

(minden elem
  az első
 oszlopból)
```

[2] Lásd részletesen: Wikipedia- NAN. Elérhető: https://en.wikipedia.org/wiki/NaN

1.5 List

- Komplex adatstruktúrát tartalmazó objektum.

```
d <- data.frame(id=1:3, company=c("OTP","MKB","KH"), value = c(1/3,2/3,4/3))
```

	id	company	value
1	1	OTP	0.3333333
2	2	MKB	0.6666667
3	3	KH	1.3333333

```
str(d)
```

```
'data.frame':   3 obs. of  3 variables:
 $ id     : int  1 2 3
 $ company: Factor w/ 3 levels "KH","MKB","OTP": 3 2 1
 $ value  : num  0.333 0.667 1.333
```

```
bankok <- list(x="Három bank", y = d)
str(bankok)
```

```
List of 2
 $ x: chr "Három bank"
 $ y:'data.frame':     3 obs. of  3 variables:
  ..$ id     : int [1:3] 1 2 3
  ..$ company: Factor w/ 3 levels "KH","MKB","OTP": 3 2 1
  ..$ value  : num [1:3] 0.333 0.667 1.333
```

```
bankok$x
```

```
[1] "Három bank"
```

```
bankok$y$id
```

```
[1] 1 2 3
(y-on belül az id-k előhívása)
```

1.5.1 List indexelés

```
class(bankok[2])
```

```
[1] "list"
(bankok típusa)
```

```
class(bankok[[2]])
```

```
[1] "data.frame"
(y típusa)
```

2. Adatok importálása, exportálása

2.1 Working directory beállítása

```
getwd();

setwd("S:\\R");
getwd()
```

```
"C:/Users/XXXXX/Documents"

"S:/R"
```

Beállításkor: dupla perjel szükséges az elérési útvonal megadásában vagy jobbra dőlő perjel (/)!

2.2 Importálás

1.
```
adatok <- read.table("CUSTOMER.CSV", header = T, sep = ",");
str(adatok)
```

CSV fájl importálása, amely fejlécet tartalmaz és az adatok vesszővel szeparáltak.

2.
```
adatok2 <- read.table("CUSTOMER.txt", header = T, sep = ",")
```

txt fájl importálása, amely fejlécet tartalmaz és az adatok vesszővel szeparáltak (angol Excelnél vesszővel tagol, magyar Excelnél pontosvesszővel → jegyzettömbben érdemes megnézni, hogy mivel tagolt a file)

2.3 Exportálás

```
write.table(adatok, "CUSTOMER2.txt", sep =",")
```

txt fájl exportálása, amiben a korábban beolvasott adatok vannak benne, CUSTOMER2 néven, és vesszővel legyenek szeparálva
Ha már létező fájl nevet adunk meg, akkor kérdezés nélkül felülírja a fájlt!

magyar nyelvű Excelre:
```
write.table(adatok, "CUSTOMER3.csv", sep =";")
```

3. Adatfeldolgozás

Olvassuk be az ugyfelcsv1.csv és az ugyfelcsv2.csv fájlokat!

```
ugyfel<-read.table("ugyfelcsv1.csv", header=T, sep=";");
ugyfel2<-read.table("ugyfelcsv2.csv", header=T, sep=";");
str(ugyfel)

'data.frame':  10 obs. of  5 variables:
 $ ugyfel_id: int  1 2 3 4 5 6 7 8 9 10
 $ vnev     : Factor w/ 9 levels "Armstrong","Brown",..: 9 1 7 6 6 5 3 8 2
 $ knev     : Factor w/ 10 levels "Blair","Charissa",..: 4 6 9 7 8 1 10 2 5
 $ szdatum  : Factor w/ 10 levels "1926-03-25","1932-02-24",..: 6 2 3 8 4 9
 $ nem      : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 1 1 1
```

Probléma: a születési dátumot faktornak[3] érzékeli.

Megoldás:

```
ugyfel$szdatum = as.Date(ugyfel$szdatum);
str(ugyfel)

…
$ szdatum  : Date, format: "1954-07-10" "1932-02-24" ...
…
```

3.1. FELADAT

Válasszuk ki az 1950 előtt született nőket!

szures tábla létrehozása *ugyfel* táblából kiszedve az adatokat két feltétel megadásával: nő és 1950 előtti születési dátumú

Szimbólumok jelentése:
$: fejléc hívása
dátum felismerése kötőjel formátummal!: 1950-01-01
&: több feltétel összefűzése
<-: legyen egyenlő (állítás, hogy az)
==: egyenlő-e? (igen/nem)
<=: kisebb egyenlő
ahogyan vissza kívánjuk kapni az eredményt – itt csak vezetéknév, keresztnév:
`c("vnev", "knev");`

```
szures<-ugyfel[ugyfel$szdatum<=as.Date("1950-01-01")&ugyfel$nem=="F",c("vnev", "knev")];
szures

     vnev     knev
8 Shields Charissa
9   Brown     Inga
```

```
Táblából kérünk ki egy adatot (pl.2.sor, 3.oszlop)
ugyfel[2,3]
[1] Kieran
```

[3] A faktor jelentését lásd részletesen: Wikipedia – Enumerated type.
Elérhető: https://en.wikipedia.org/wiki/Enumerated_type

1950 előtt született nők összes adatának megjelenítése (nemcsak a vnev, knev-t):

```
szures<-ugyfel[ugyfel$szdatum<=as.Date("1950-01-01")&ugyfel$nem=="F",];
szures

ugyfel_id    vnev     knev    szdatum nem
8         8 Shields Charissa 1926-03-25   F
9         9   Brown     Inga 1942-04-18   F
```

3.2. FELADAT

Számoljuk ki nemenként az átlagéletkort!

Először az életkorukat számoljuk ki (születési év kiszedés, mai dátumból kivonjuk ezeket)

```
ugyfel$szev<-format(ugyfel$szdatum, "%Y")
```

→ kiszedi a dátumból az évet, szöveg formátumban

```
ugyfel$eletkor<-2016-as.integer(ugyfel$szev)
```

→ az idei évből levonjuk a születési évet, így új oszlopba megkapjuk az életkort

```
aggregate(formula=eletkor~nem, data=ugyfel, FUN=mean)
  nem  eletkor
1   F 70.33333
2   M 58.57143
```

~:eletkor(x)-t szeretnénk magyarázni a nem függvényében → nem mindegy a sorrend!!!
data: itt az egyenlőséget kell használni, nem kicserélhető
fun: function (szórás, átlag, mean)

3.3. FELADAT

Készítsünk egy lekérdezést, amelyben arra a kérdésre kapjuk meg a választ, hogy a női-férfi ügyfelek között látható-e eltérés a megadott igazolvány típusban!

Összevonjuk a két csv fájlt (ugyfel_id a kulcs/összekötő elem):

```
tabla<-merge(ugyfel, ugyfel2, by.x="ugyfel_id", by.y="ugyfel_id")
```

→ merge(első tábla (x), második tábla (y), első táblában lévő kulcs neve, második táblában lévő kulcs neve)
a kulcsok fejlécének nem kell megegyeznie

table→kereszttáblát hoz létre:

```
kimutatas<-table(tabla$igazolvanytipus_id, tabla$nem);
kimutatas

    F M
  J 1 2
  SZ 1 3
  UL 1 2
```

Tehát megnézi, hogy milyen igazolványtípusok vannak és hogy ezekből hány darab, nemenkénti csoportosításban.

prop.table→"aránytáblát" hoz létre :

```
prop.table(kimutatas,2)
```

prop.table(tábla neve, sor/oszlop szerint)
1: sorok szerint (sor szum 100%)

2: oszlopok szerint (oszlop szum 100%)
alapértelmezett: prop.table(tábla neve) – a tábla eleminek összegével elosztja az összes elemet

```
          F         M
  J  0.3333333 0.2857143
  SZ 0.3333333 0.4285714
  UL 0.3333333 0.2857143
```

3.4. FELADAT

Készíts egy lekérdezést, amelyből kiderül, hogy hány darab ügyfelünk van!

```
nrow(ugyfel)
[1] 10
```

Mivel minden ügyfél pontosan egy sorban szerepel, ezért elég, ha a sorok számát ismerjük.

3.5. FELADAT

Készíts egy új táblát, amely azoknak az ügyfeleknek a nevét tartalmazza, (vezetéknév keresztnév egy változóban), akik 1950.01.01 előtt születtek!

ugyfel táblából azok a sorok, ahol a *szdatum* kisebb egyenlő 1950-01-01-nél, de minden oszlop szerepel

```
nevekegyutt<-ugyfel[ugyfel$szdatum <= as.Date("1950-01-01"),]
```

oszlopokból válasszuk ki csak a *vnev*, *knev* változókat

```
nevekegyutt<-ugyfel[ugyfel$szdatum <= as.Date("1950-01-01"),c("vnev","knev"
)]
```

de így még a két változó külön szerepel, és mi összefűzve szeretnénk

összefűzés függvénye:
paste

```
paste("Kovács", "Pisti", sep=" - ")
```

összefűz tetszőleges számú argumentumot ”szóköz vonal szóköz”-zel, de vektorokra csak koordinátánként működik

```
paste(c("Kovács", "Pisti"), collapse=" - ")
```

összefűzi egy vektor koordinátáit ”szóköz vonal szóköz”-zel,

A leszűrt adatokat és a paste függvényt az apply függvény segítségével tudjuk soronként összefűzni:

- *apply* függvény (x, MARGIN, FUN,…)
- x: egy mátrix vagy data.frame
- MARGIN: az az index, ami alapján alkalmazni kell a függvényt, 1-sor, 2-oszlop, c(1,2) –
- FUN: a függvény amelyet alkalmazni szeretnénk
- az x mátrix/data.frame soraira vagy oszlopaira alkalmazzunk a függvényt

```
nevekegyutt<-apply(ugyfel[ugyfel$szdatum <= as.Date("1950-01-01"),
c("vnev","knev")],1,FUN = function(x) paste(x, collapse=" "))
```

3.6. FELADAT
Készíts egy új táblát, amely megadja minden ügyfél életkorát 10 év múlva!

```
ugyfel10ev<-cbind(ugyfel[,c("ugyfel_id","vnev","knev")],ugyfel$eletkor+10)
```

Oszloponként egymás mellé ragasztjuk az *ugyfel* tábla kiválasztott oszlopait (id, vezetéknév, keresztnév), és az életkor+10 év vektort.

3.7. FELADAT
Adjuk hozzá a táblához Pistikét, az ő adatai a következőek:

```
vnev  knev   szdatum    nem igazolvanytipus_id igazolvanyszam
Kiss  Pisti  1990.01.01 M   SZ                  123456AP
```

```
ugyfelPisti1 <- rbind(ugyfel,data.frame(ugyfel_id = nrow(ugyfel)+1, vnev =
"Kiss",knev="Pisti",szdatum=as.Date("1990-01-01"),nem = "M", szev = 1990 ,
eletkor = 2016-1990))
```

Létrehozunk egy új data.frame-t, amely Pistike adatait tartalmazza (egyetlen sorból áll), és hozzáragasztjuk ezt a data.frame-t az eredeti *ugyfel* táblához.

```
ugyfelPisti2 <- rbind(ugyfel2,data.frame(ugyfel_id = nrow(ugyfel)+1,
igazolvanytipus_id="SZ",igazolvanyszam="123456AP"))
```

Majd ezt elvégezzük az *ugyfel2* táblára is.

(Általában nem lehet egy sorvektort hozzáragasztani egy data.frame-hez csak data.frame -t.)

3.8. FELADAT
Módosítsuk Pistike születési dátumát 1991.01.01-re.

```
ugyfelPisti[11,c(4,6,7)] <- data.frame(as.Date("1991-01-01"),1991,24)
```

ugyfelPisti adathalmaz 11. sorának (ez tartalmazza Pisti adatait) 4,6,7. oszlopa legyen egyenlő a megadott adatokkal (sz_datum=1991-01-01, szev=1991, eletkor=24, melyeket szintén data.frame-ként kell megadni; vektorként nem lehet, mert nem azonosak az adattípusok).

3.9. FELADAT
Töröljük ki Pistikét, úgy hogy az eredmény egy új tábla legyen.

```
ugyfelPistiTorolve <- ugyfelPisti[ugyfelPisti$ugyfel_id != 11,]
```

Válasszuk ki azokat a sorokat az *ugyfelPisti* adathalmazból, amelyek nem egyenlőek (!=) Pisti id-jával, azaz a 11-es sorral, és hozzunk létre egy új adattáblát ezen sorokból.
Egy data.frame-ből oszlopot lehet törölni, de sort nem.

```
ugyfelPistiTorolve2 <- ugyfelPisti2[ugyfelPisti2$ugyfel_id != 11,]
```

Majd ezt elvégezzük az *ugyfelPisti2* táblára is.

4. Vezérlési szerkezetek (If, for)

4.1 Logikai változók/vektorok

- `a <- TRUE` vagy `a <- FALSE`.
- `A >, >=, <, <=, ==, !=` operátorok logikai értéket adnak vissza.
- Ezek is „vektorizáltak”: ha `a` egy nyolc elemű vektor, és `b = (a > 0)`, akkor b is nyolc elemű vektor lesz, és azokon a helyeken lesz `TRUE` ahol `a` pozitív.

Példa:

```
a<- c(5,3,2,10);
(a>3)
```

Megvizsgáljuk, melyek azok a koordináták, ahol nagyobb mint 3 szerepel.

```
TRUE FALSE FALSE TRUE
```

4.2 Feltételes elágazás

```
if (egy darab logikai értéket visszaadó kifejezés) {
   utasítás1
} else {
   utasítás2
}
```

```
x<-4;
if (x < 5){
        print("kicsi")
} else {
        print("nagy")
}
```

4.3 Ciklusok

```
for (i in 1:10){print(i)}
```

```
a <- runif(1);
while (a < 0.8) {print(a);a <- runif(1);}
```

A for ciklust, akkor használjuk, mikor előre tudjuk, hogy hányszor kell a ciklusmagot lefuttatni.

A while esetén a kilépési feltétel ismert, és mindaddig futtatjuk a ciklust, míg hamis nem lesz. Jelen példa esetében addig generálunk egyenletes eloszlású véletlen számot, míg 0.8-nál nagyobb számot nem kapunk.

Összetett feladatok

4.1. FELADAT

Határozzuk meg 1-től 10-ig melyik szám páros illetve páratlan!

```
for (i in 1:10){
        if(i%%2 == 1){
                print("páratlan")
        }else{
                print("páros")
        }
}
```

1-től 10-ig csináljon valamit
%%2-vel osztva mennyi maradékot ad
i%%2 == 1 → ha i 2-vel osztva 1 maradékot ad, akkor írja ki, hogy páratlan, egyébként írja ki, hogy páros

4.2. FELADAT

Adjuk össze 10-ig a páratlan számokat!

VBA típusú (R-ben nem hatékony)

```
osszeg <- 0;
for (i in 1:10){
        if(i%%2 == 1){
        osszeg = osszeg+i
        }
}
```

1-et osztva 2-vel 1 maradékot ad → osszeg=0+1=1 →
2 osztva 2-vel 0 maradékot ad→ osszeg=1+0=1 →
3 osztva 2-vel 1 maradékot ad → osszeg=1+3=4….

R logika: logikai vektorokkal

```
osszeg <- sum((1:10)*((1:10)%%2))
```

Két vektor szorzatának összege; első vektor 1-10-ig a számokat, a második vektor a maradékokat tartalmazza.

4.3. FELADAT

Készítsünk el egy10X10-es mátrixot, amely a szorzótáblát tartalmazza!

VBA típusú (R-ben nem hatékony)

```
szorzo <- matrix(0,10,10);
for (i in 1:10){
      for (j in 1:10){
            szorzo[i,j] = i*j }
}
```

10x10-es mátrix létrehozása, csak 0 van benne

szorzo nevű mátrix i.sor j.oszlop = i*j

először i=1 esetén j-t futtatja végig 1-től 10-ig, majd i=2 esetén újra j-t 1-től 10-ig, és így tovább…

R logika

```
cbind(1:10)%*%rbind(1:10)
```

egy oszlopvektor (melynek elemei 1-10) és egy sorvektor szorzata (melynek elemei 1-10)

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]    2    4    6    8   10   12   14   16   18    20
 [3,]    3    6    9   12   15   18   21   24   27    30
 [4,]    4    8   12   16   20   24   28   32   36    40
 [5,]    5   10   15   20   25   30   35   40   45    50
 [6,]    6   12   18   24   30   36   42   48   54    60
 [7,]    7   14   21   28   35   42   49   56   63    70
 [8,]    8   16   24   32   40   48   56   64   72    80
 [9,]    9   18   27   36   45   54   63   72   81    90
[10,]   10   20   30   40   50   60   70   80   90   100
```

4.4. FELADAT

Számold ki az A mátrix 2015. hatványát!

0,1 0,7 0,2
0,6 0,4 0
0,2 0,7 0,1

```
A<-matrix(c(0.1,0.6,0.2,0.7,0.4,0.7,0.2,0,0.1),3,3);
B<-diag(1,3);
for (i in 1:2015){B<-B%*%A}
```

Trükk: B egységmátrix létrehozása
i=1 esetén A mátrixot egységmátrixszal kell megszorozni = A első hatványa, ezzel visszakapjuk A-t (Bmátrix*Amátrix) → ez B-be kerül
i=2 esetén B*A=A*A
és így tovább…

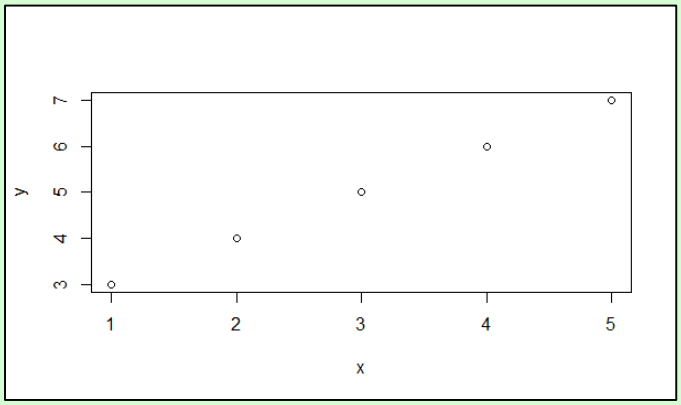
(Természetesen mátrixot hatványozni nem így kell, itt csak a for ciklust gyakoroljuk!)
Feladat: Akkor hogyan kell? ☺

5. Grafikus megjelenítés

- `plot(x,y)` ábrázolja az $(x(i), y(i))$ pontokat (scatter plot).
- `hist(x)` az `x` vektor elemeiből hisztogramot készít.
- `matplot(x, y)` a mátrixot ábrázolja oszloponként.

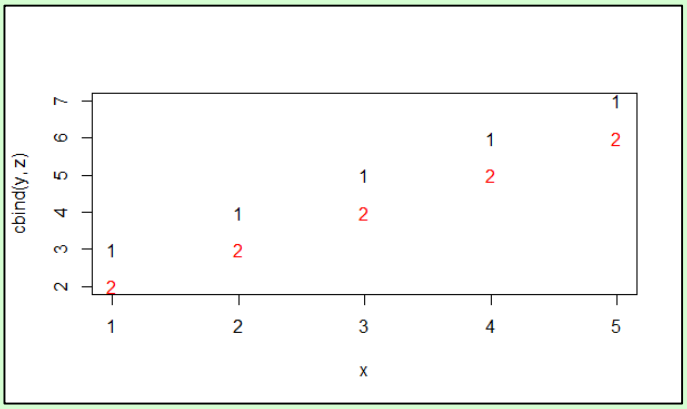
```
x<-1:5;
y<-3:7;
plot(x,y);
plot(x,y, type="l")
```

→pontábra
→vonalábra




```
z<-2:6;
matplot(x,cbind(y,z))
```

→ x tengelyen x értékei, y tengelyen y (az ábrán 1-esek jelölik) és z elemei (az ábrán 2-esek jelölik) [ahhoz, hogy mindkettőt meg lehessen jeleníteni, oszlopvektorba kell összefűzni]



5.1. FELADAT

Definiáljunk egy 10.000 hosszúságú vektort, a (-3pi,3pi) intervallumon és ennek segítségével ábrázoljuk a sin és cos függvény grafikonját. Az ábrákra írjuk rá a függvények nevét is, mint címet.

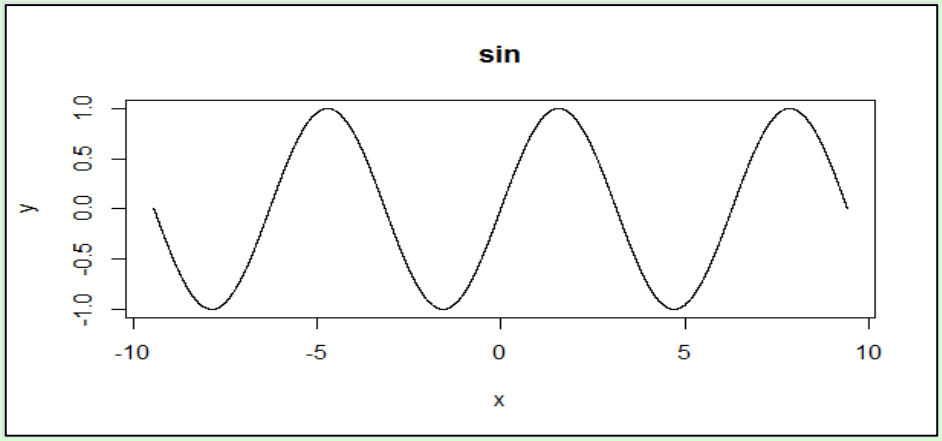
Sinus függvény kirajzolása:

```
x <- seq(-3*pi,3*pi, length = 10000)
```

→ 10.000 hosszúságú vektor a (-3pi,3pi) intervallumon

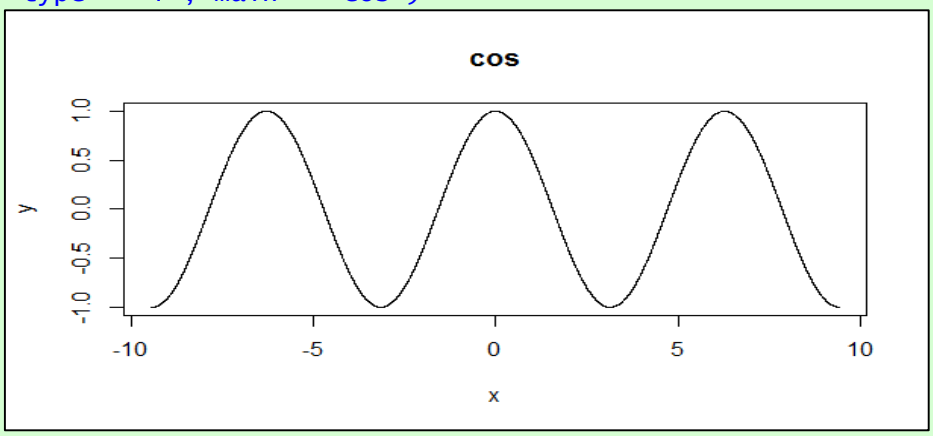
```
y <- sin(x);
plot(x,y, type = "l", main = "sin")
```

két tengely: x, y; típusa: „l” jelentése line; main: diagram felirat



Cosinus függvény kirajzolása:

```
x <- seq(-3*pi,3*pi, length = 10000);
y <- cos(x);
plot(x,y, type = "l", main = "cos")
```

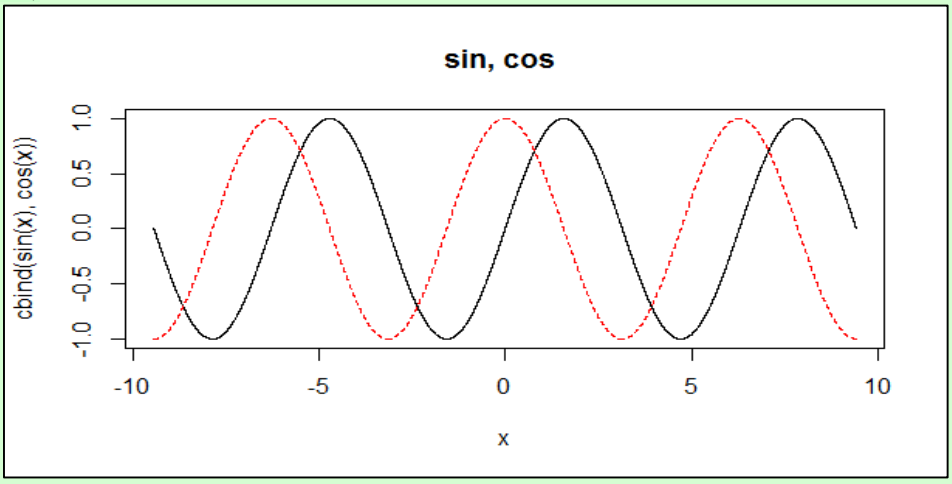


5.2. FELADAT

Ábrázoljuk közös koordinátarendszerben a sin(x) és cos(x) függvényt egyszerre, különböző színnel.

```
matplot(x, cbind(sin(x),cos(x)), type = "l", main = "sin, cos ")
```

az y tengely a mátrix oszlopait jeleníti meg, ami a sinus és cosinus függvényt tartalmazza;



6. Saját függvény készítése és ábrázolása

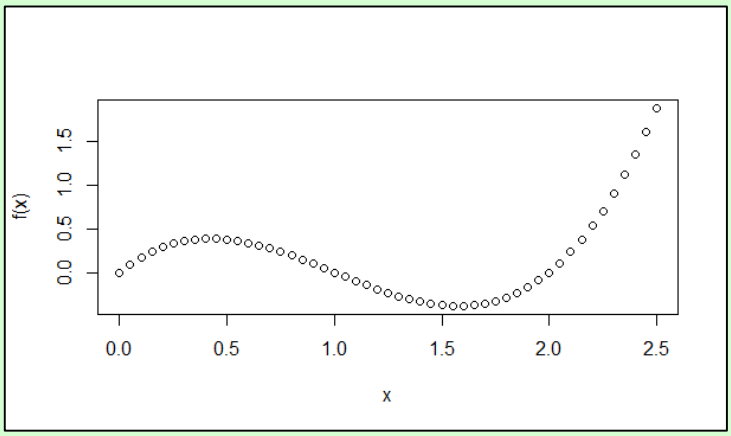
6.1 Felhasználói függvény

```
fuggvenyneve <- function(x) {2 * x + 1};
fuggvenyneve(5)
[1] 11
```

6.1. FELADAT

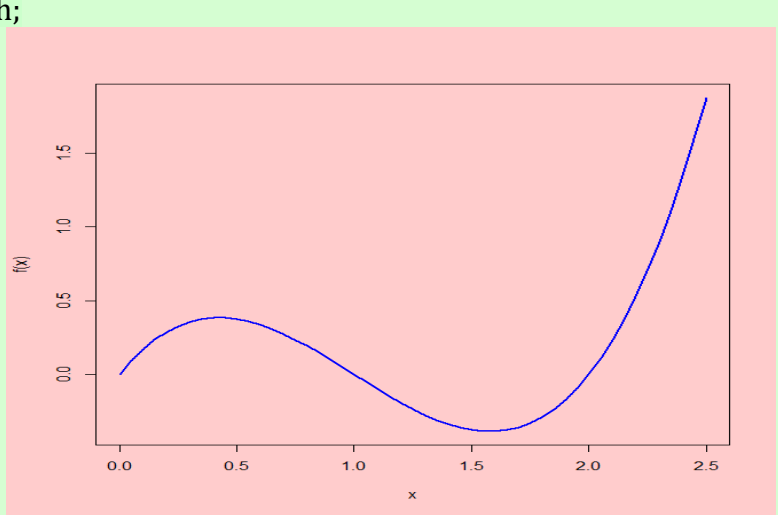
Ábrázoljuk a következő polinomot $p(x) = x^3 - 3x^2 + 2x$, 0 és 2,5 közötti intervallumon 0,05-ös lépésközzel!

```
x<-seq(0,2.5,by=0.05);
f<-function(x) {x^3-3*x^2+2*x};
plot(x,f(x))
```



színesen:

```
par(bg=rgb(1,0.8,0.8))
# paraméter beállítások; itt: háttérszín(bg)
plot(x,f(x),type="l",col="blue", lwd=2)
#lwd=line width;
```



6.2. FELADAT

Definiáljunk egy 10.000 hosszúságú vektort, a (-3pi,3pi) intervallumon és ennek segítségével ábrázoljuk a sin függvény pozitív részét.

```
x<-seq(-3*pi, 3*pi,length=10000);
y<-sin(x);
z<-y*(y>=0)
```

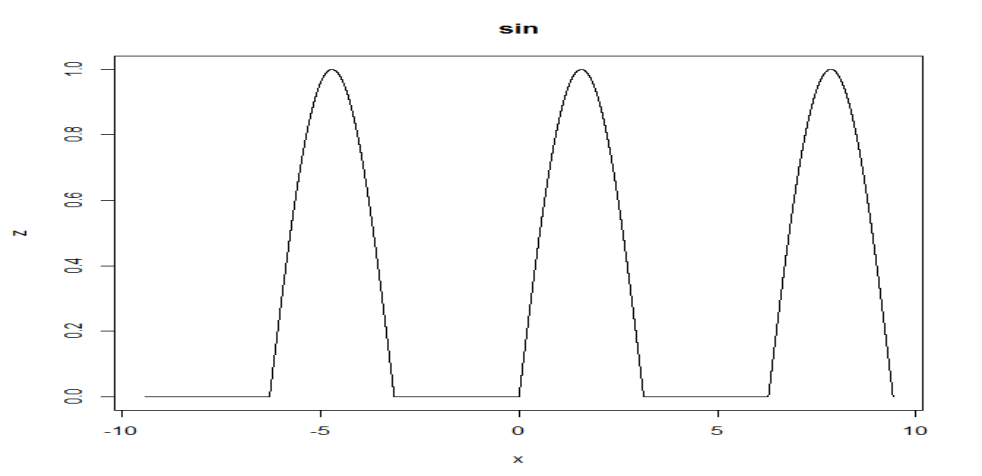
logikai vektor alkalmazása:

Ha a szinusz függvény értéke az adott pontban >=0, akkor TRUE, azaz minden értékét megszorozzuk 1-el.

Ha a szinusz függvény értéke az adott pontban <0, akkor FALSE, azaz minden értékét megszorozzuk 0-val

diagramon megjelenítés:

```
plot(x,z,type="l", main="sin")
```



megértés segítése:

```
k<-c(-4,5,-1,10,14);
m<-(k>=0);
m
[1] FALSE  TRUE FALSE  TRUE  TRUE
```

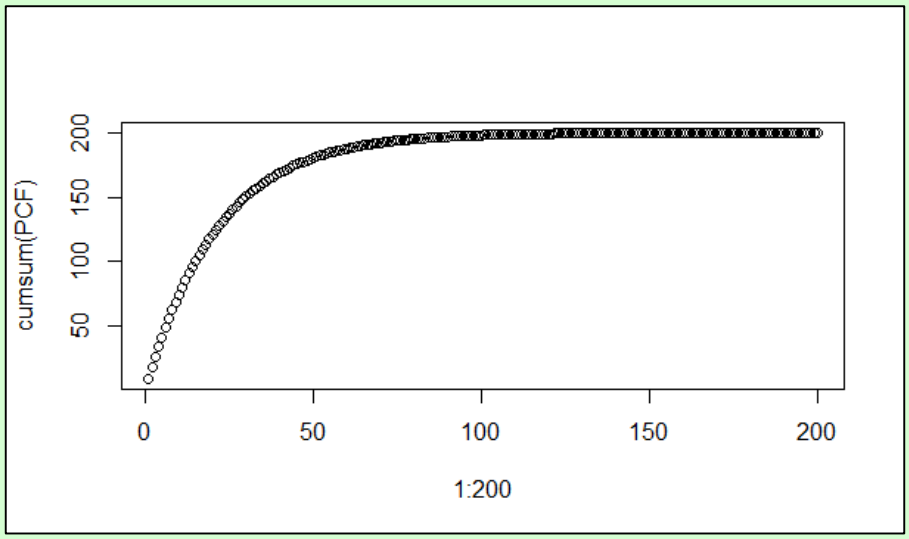
ha k>=0, akkor TRUE, ha k<0 FALSE

6.3. FELADAT

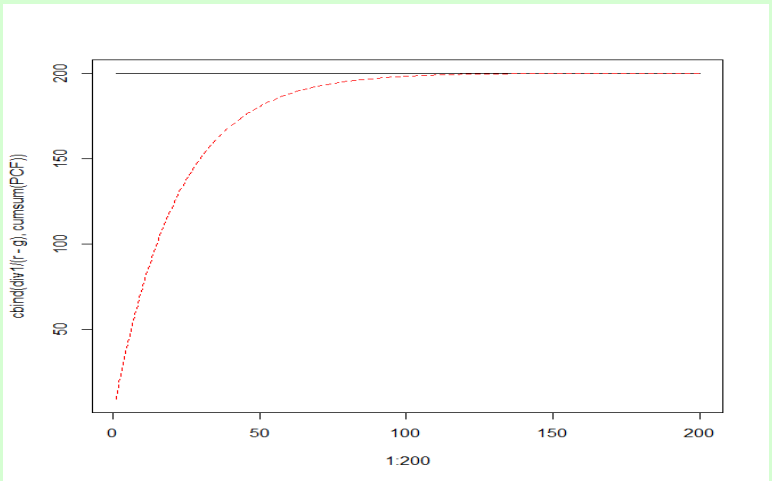
Szimuláld a Gordon-képletet: $P = \sum DIV_1 \frac{(1+g)^{i-1}}{(1+r)^i} = \frac{DIV_1}{r-g}$. Azaz ábrázold egy ábrán vízszintes vonallal a képlet által kapott eredmény Div_1=10, r=10%, g=5% esetre, valamint t=1, 2,3,…200-ra a t-ig számított részletösszegeket.

```
div1<-10;
r<-0.1;
g<-0.05;
P<-div1/r-g;
CF<-10*(1+g)^(0:199);
PCF<-CF/(1+r)^(1:200);
cumsum(PCF);
plot(1:200,cumsum(PCF))
```

→CF-k jelenértéke
→CF-k kumulálása
→kumulált CF-k ábrázolása az idő függvényében; pontdiagram




```
plot(1:200,cumsum(PCF), type="l")
```
→kumulált CF-k ábrázolása az idő függvényében; vonaldiagram




```
matplot(1:200,cbind(rep(div1/(r-g),200), cumsum(PCF)),type="l")
```

6.4. FELADAT

Rajzolj fel egy egység sugarú kört, és szimulációval próbáld meg minél pontosabban meghatározni a pi értékét! (kerület=2rπ, terület=$r^2\pi$)

Az egységkör (origó középpontú, egy egység sugarú kör) területe π. (± 1, ±1) négyzet területe 4 egység. Ha véletlenszerűen generálunk egyenletes eloszlású számokat a négyzeten, akkor a körbe esés valószínűsége $\frac{\pi}{4}$.

A körbe esés feltétele: koordináták négyzetösszege legfeljebb 1, hiszen a körvonal egyenlete=$x^2 + y^2 = r^2 = 1$

Tehát a π becslése: $\frac{körbe\ eső\ pontok\ száma}{összes\ legenerált\ pont\ száma} * 4$

```
x <- seq(-1,1,length = 1000);
y <- sqrt(1-x^2)
```

#félkör vonal egyenlete y-ra rendezve

```
n <-500
```

n a pontok száma; figyeljünk arra, hogy két koordináta van, kétszer ennyi véletlenszám kell

```
matplot(x,cbind(y,-y), type = "l", lwd = 2, lty=1, col = 1)
```

két félkört rajzolunk az origó körül fekete színű (col=1) folytonos (lty=1)vonallal (type ="l"), 2-es vastagsággal (lwd = 2)

```
szimu <- matrix(runif(2*n)*2-1,n,2)
```

generálunk 2*n=1000 db véletlen számot 0 és 1 között egyenletes eloszlással, majd áttranszformáljuk -1 és 1 közé, azaz megszorozzuk 2-vel és kivonunk belőle 1-t, n sorba és 2 oszlopba rendezzük

```
points(szimu[,1],szimu[,2], pch = 20, col = "red")
```

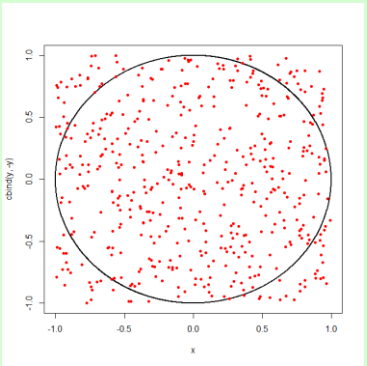
ábrázoljuk a pontokat piros színnel

```
hossz <- szimu[,1]^2+szimu[,2]^2
```

#minden pontnak kiszámoljuk a távolságát az origótól

```
sum(hossz<=1)/n*4
```

#logikai vektor alkalmazásával meghatározzuk az egységkörbe eső pontok számát, és a becsült π értékét



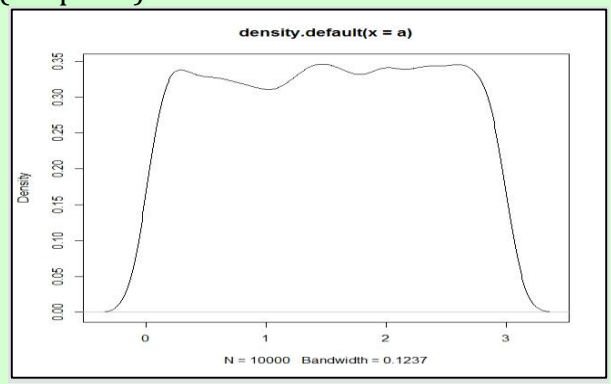
Ha az n értéke magas (>1000) nem célszerű ábrázolni a pontokat!

7. Véletlen számok

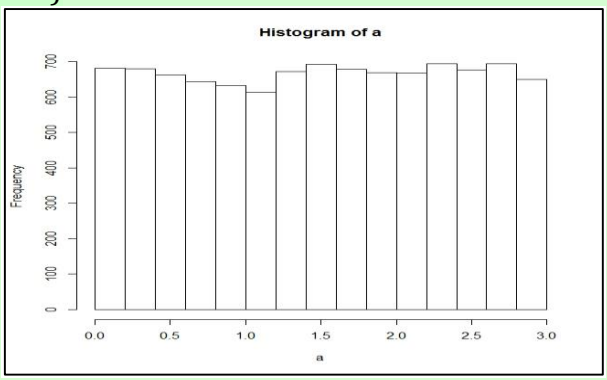
7.1 Egyenletes eloszlás

7.1. FELADAT
Generáljunk 10.000 darab véletlen számot egyenletes eloszlással 0 és 3 között! Becsüljük meg a sűrűségfüggvényét és eloszlásfüggvényét!

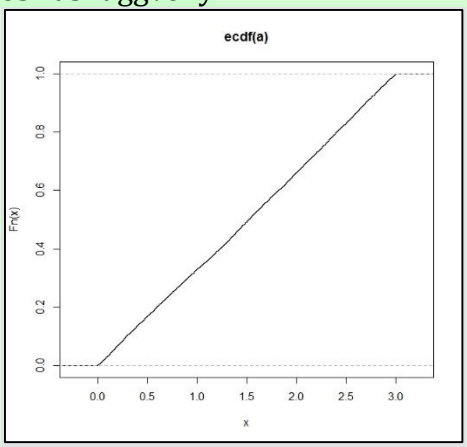
```
a <- runif(10000,0,3)
#→ 10 000 véletlen szám 0 és 3 között
plot(density(a))
#→ sűrűségfüggvény (beépített)
```




```
hist(a)
#→ hisztogram (beépített)
```



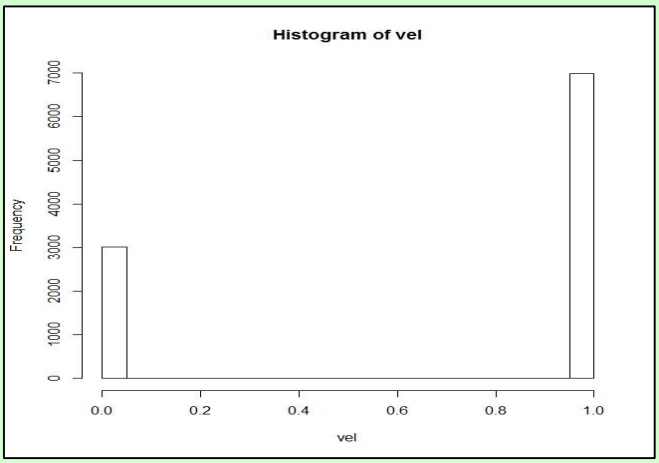

```
plot(ecdf(a))
#→ empirikus kumulatív eloszlás függvény
```



7.2. FELADAT

Generáljunk 10.000 darab véletlen számot, melyeknek 70% eséllyel 1 és 30% eséllyel 0 az értéke. Ábrázold hisztogramon az eredményt!

```
vel<-runif(10000)
#→ 10 000 véletlen szám generálása
vel<-(vel>=0.3)*1
#→ ha a véletlen szám 0,3-nál nagyobb, akkor mint logikai változó, az értéke TRUE=1, azaz 1*1=1 → 70% eséllyel 1 a véletlen szám
hist(vel)
#→ hisztorgram (beépített)
```

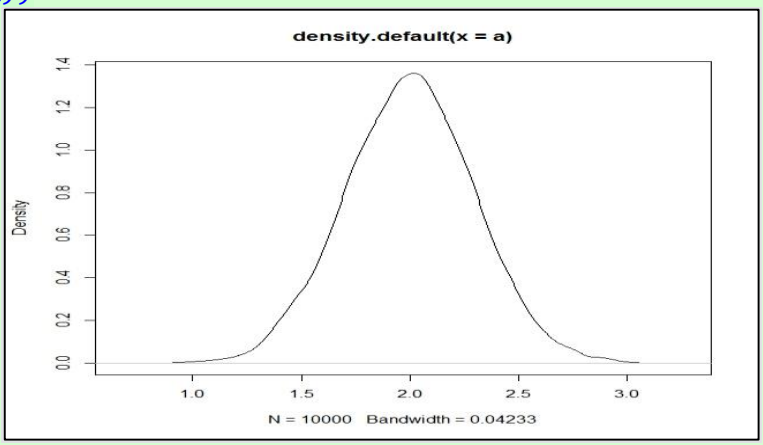


7.2 Normális eloszlás

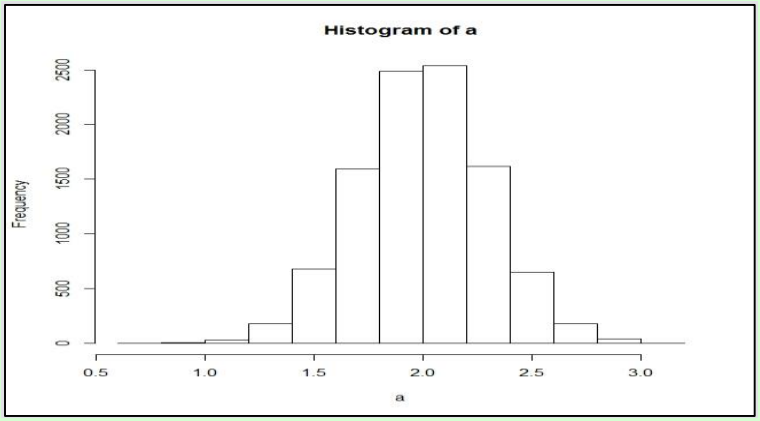
7.3. FELADAT

Generáljunk 10.000 elemű véletlen mintát normális eloszlásból. Az átlag legyen 2 a szórás 0,3! Becsüljük meg a sűrűségfüggvényét és eloszlásfüggvényét!

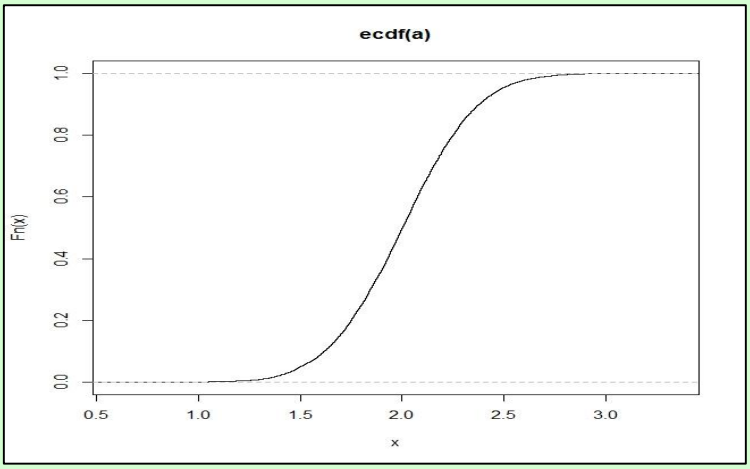
```
library(stats);
a <- rnorm(10000,mean = 2,sd = 0.3);
plot(density(a))
```



```
hist(a)
```




```
plot(ecdf(a))
```



7.3 Exponenciális eloszlás

7.4. FELADAT

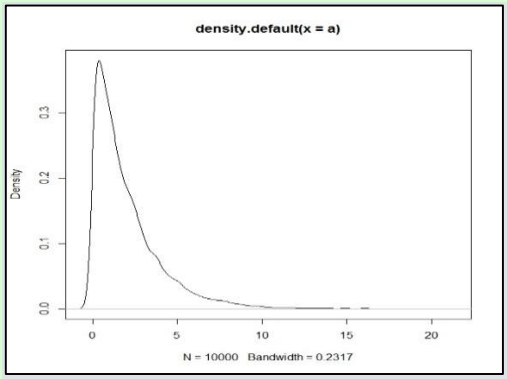
Generáljunk 10.000 darab véletlen számot exponenciális eloszlásból, az átlag legyen 2! Becsüljük meg a sűrűségfüggvényét és eloszlásfüggvényét!

```
a <- rexp(10000,rate = .5)
```

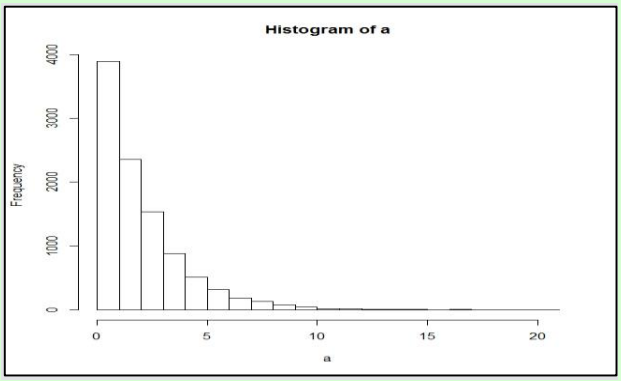
→ exponenciális eloszlás (beépített), megfigyelések száma=10 000, átlag=2

→ rate=1/átlag=0.5

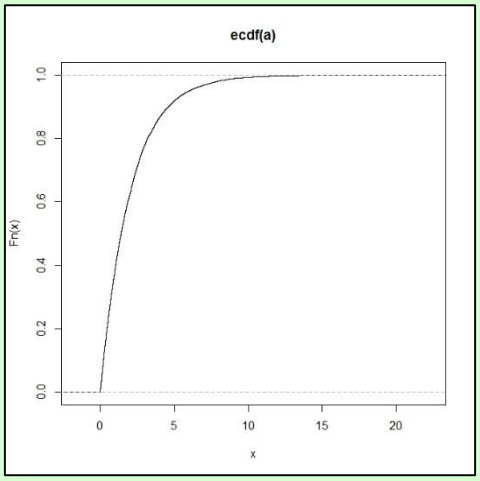
```
plot(density(a))
```



```
hist(a)
```




```
plot(ecdf(a))
```



7.4 Student/t-eloszlás

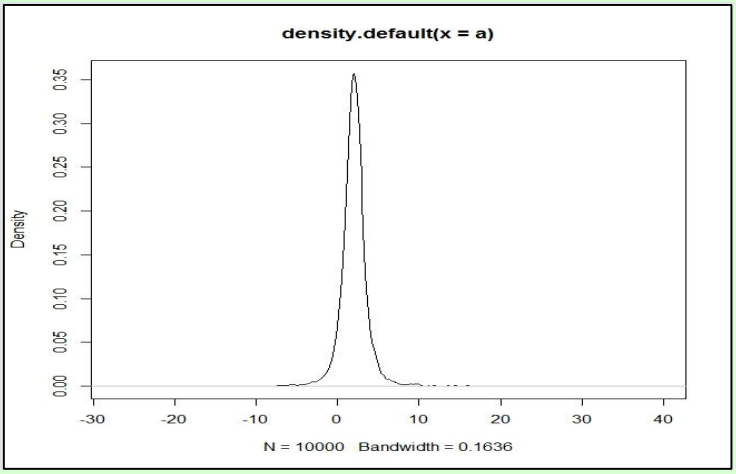
7.5. FELADAT

Generáljunk 10.000 darab véletlen számot Student eloszlásból, az átlag legyen 2! Becsüljük meg a sűrűségfüggvényét és eloszlásfüggvényét!

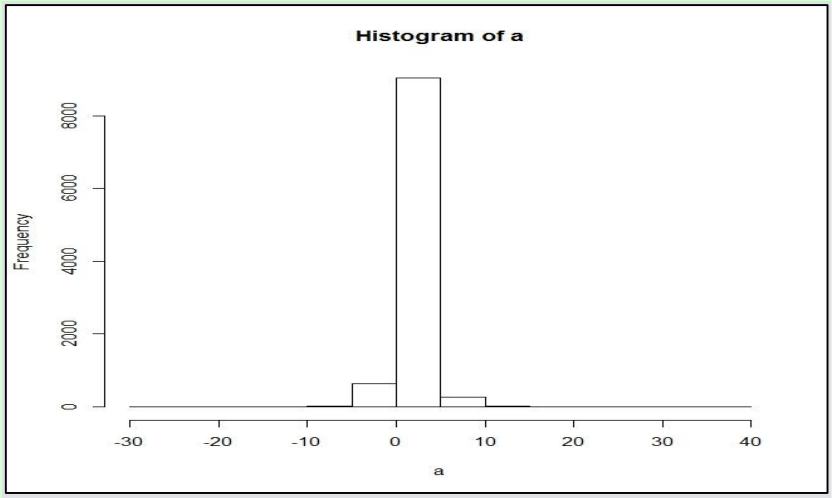
```
a <- rt(10000,df = 3)+2
```

→ t-eloszlás (beépített), megfigyelések száma= 10 000, szabadságfok=3, átlag=2

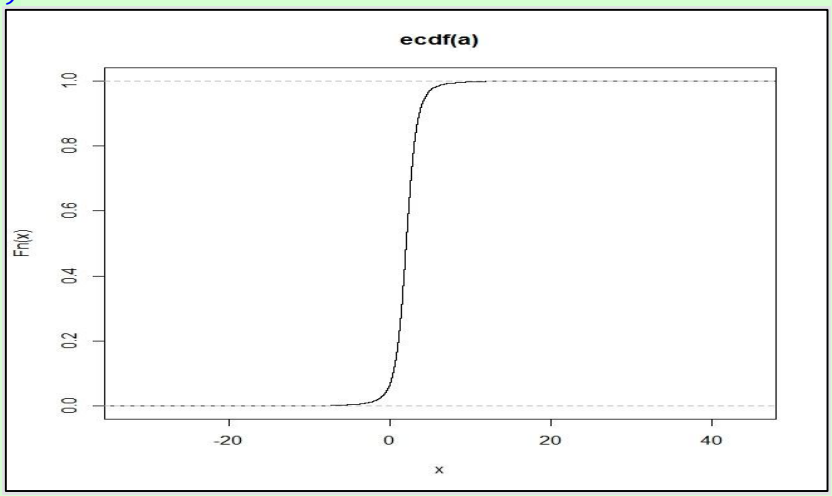
```
plot(density(a))
```




```
hist(a)
```


Histogram of a
Frequency
0
2000
4000
6000
8000
-30
-20
-10
0
10
20
30
40
a

```
plot(ecdf(a))
```


ecdf(a)
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
-20
0
20
40
x

8. Vizualizáció

Ggplot2: http://docs.ggplot2.org/current/
Shiny: http://shiny.rstudio.com/gallery/

8.1 Szófelhő

Szöveg típusú (kvalitatív) adatok prezentálására alkalmazható.
Telepítsük fel a csomagokat!

```
install.packages("tm")  # for text mining;
install.packages("SnowballC") # for text stemming;
install.packages("wordcloud") # word-cloud generator;
install.packages("RColorBrewer") # color palettes;
```

A csomagokat elég egyszer telepíteni otthon, de az egyetemen minden alkalommal kell, mert a Temp mappában helyezi a feltelepített csomagokat!

```
library("tm");
library("SnowballC");
library("wordcloud");
library("RColorBrewer");
```

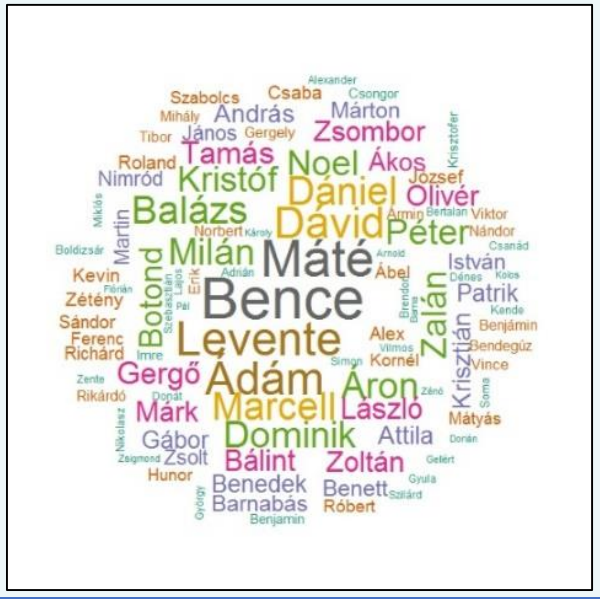
8.1. FELADAT

Ábrázoljuk a leggyakoribb férfineveket a 2014-ben születettek esetében!

Adatok forrása: nevek.csv[4]

```
adatok <- read.table("nevek.CSV", header = T, sep = ";");
windows();
wordcloud(adatok$Nevek,adatok$freq, random.order=FALSE, colors=brewer.pal
(8,"Dark2"))
```

adatok táblából a nevek wordcloudba rendezése gyakoriságuk szerinti nagyságban;
ha random.order=TRUE random, colors: színpaletta kiválasztása



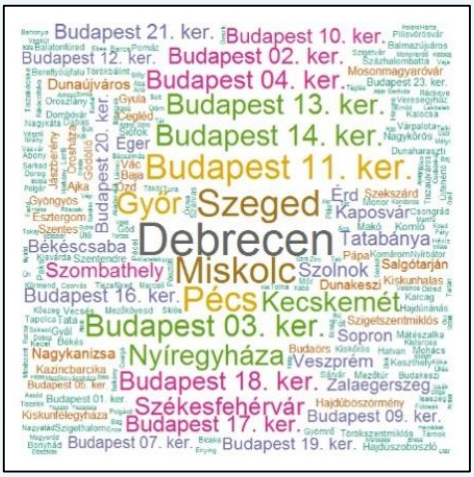
[4] Magyar Keresztnevek Tára - Utónév statisztika 2014. Elérhető: http://magyarnevek.hu/nevek/utonevstatisztika/2014

8.2. FELADAT

Ábrázoljuk a Magyarország településeit a lakosság függvényében!

Adatok forrása: telepules.csv[5]

```
tel<-read.table("telepules.csv",header = T, sep = ";");
windows();
wordcloud(tel$helyseg, tel$lakos, min.freq=3, random.order=FALSE,
colors=brewer.pal(8, "Dark2"))
```



8.2 Térkép

8.3. FELADAT

Jelenítsed meg az 50.000 ezernél nagyobb lakossal rendelkező magyar városokat egy térképen!

programok telepítése:

```
install.packages("maps");
install.packages("mapdata");
install.packages("mapproj");

library(maps);
library(mapdata);
library(mapproj);

windows();
map('worldHires','Hungary')
```

→ beépített függvény, mellyel az országok térképének körvonala rajzolható meg

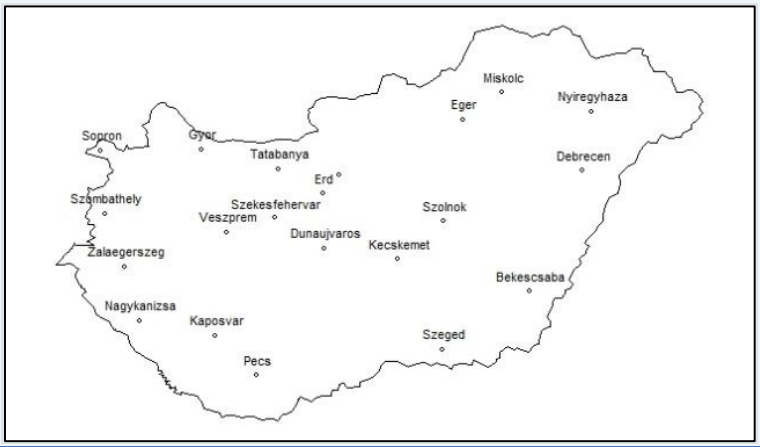
```
data(world.cities)
```

#→ beépített adatbázis, ami a világ városait tartalmazza

```
map.cities(x = world.cities, country = "Hungary", label = TRUE, minpop =
50000)
```

[5] Központi Statisztikai Hivatal - Minden helység adata. Elérhető: www.ksh.hu/docs/hun/hnk/hnk_2013.xls

→ azon magyar városok neveit rakja rá a már létező magyarországi térképre, pontos hely szerint, melyeknek minimum 50 000 fő a populációja, a feliratok (label) megjelennek



8.4. FELADAT

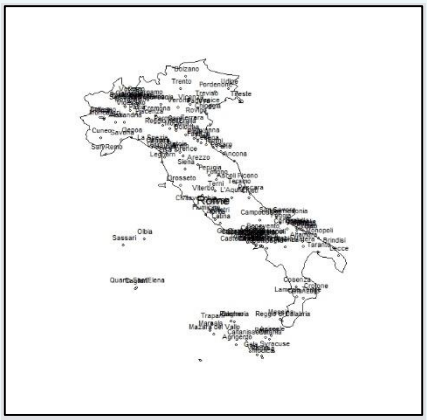
Jelenítsed meg az 50.000 ezernél több lakossággal rendelkező olasz városokat egy térképen**!**

```
windows()
map('worldHires','Italy');
#→ Olaszország térképe
data(world.cities);
map.cities(x = world.cities, country = "Italy", label = TRUE, minpop = 5000
0)
```

→ azon olasz városok neveit rakja rá a már létező olaszországi térképre, pontos hely szerint, melyeknek minimum 50 000 fő a populációja

```
map.cities(x = world.cities, country = "Italy", capitals=1)
```

→ Olaszország fővárosának megjelenítése a térképen



8.5. FELADAT

Ábrázoljuk Magyarország térképén pirossal a varosok.txt-ben található településeket!

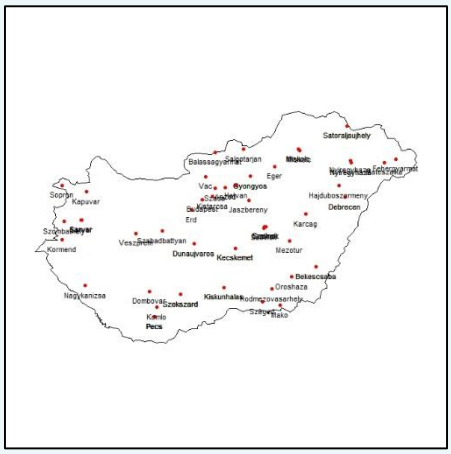
```
windows();
map('worldHires','Hungary');
adatok3 <- read.table("varosok.txt", header = T);

points(adatok3$x,adatok3$y,col="red",pch=20)
```

→ piros pontokkal jelöli az *adatok3* tábla által tartalmazott városok helyeit a táblában szereplő koordináták (x,y) alapján; pch a pontok szimbólumát lehet kiválasztani

```
text(adatok3$x,adatok3$y,adatok3$varos,pos=1,cex=0.7)
```

→ texttel a térképre szöveget lehet kiíratni; koordináták (x,y) alapján a városok helyén a nevük megjelenítése
pos: a szöveg helyének meghatározása 1-alul, 2-bal oldalon, 3-felül, 4-jobb oldalon
cex: karakter kiterjesztése

Megjegyzés: points-al bármilyen ábrára lehet pontokat tenni, text-tel pedig szöveget



9. Statisztikai feladatok

9.1. FELADAT

Olvassuk be az adatokat!

```
d<-read.table(„CUSTOMER.csv”,header=T, sep=”,”);
str(d)
#→ adatok ellenőrzésére
```

9.2. FELADAT

Számoljuk ki az AGE változó várható értékét (mean) és szórását (sd), készítsünk belőle hisztogramot (hist)!

mean, sd beépített függvények

```
atlag<-mean(d$AGE);
szoras<-sd(d$AGE);
windows();
hist(d$AGE)
```

9.3. FELADAT

Végezzük el ugyanezt az INCOME változóra is!

```
atlag<-mean(d$INCOME);
szoras<-sd(d$INCOME);
windows();
hist(d$INCOME);

summary(d)
```

→ jövedelem átlaga
→ jövedelem szórása
→ hisztogram a jövedelem adataiból
→ alap statisztikai adatok (min, max, medián, kvartilisek, átlag)

9.4. FELADAT

Válasszuk ki a táblából az AGE, INCOME, CARDDEBT és YEARSEMPLOYED változókat, és vizsgáljuk meg a köztük lévő korrelációkat.

```
kivalasztott<-d[,c(2,4,5,6)]
```

d táblából kijelöljük a szükséges oszlopokat; a vessző azt jelzi, hogy minden sort kijelölün k

```
#VAGY
kivalasztott<-d[,c(2,4:6)];

cor(kivalasztott)
```

→ korrelációs mátrix a kiválasztott adatokból (age, yearsemployed, income, carddebt)

```
                    AGE YEARSEMPLOYED    INCOME  CARDDEBT
AGE           1.0000000     0.5474975 0.4897332 0.2457968
YEARSEMPLOYED 0.5474975     1.0000000 0.6878509 0.3920967
INCOME        0.4897332     0.6878509 1.0000000 0.5187013
CARDDEBT      0.2457968     0.3920967 0.5187013 1.0000000
```

9.5. FELADAT

Vizsgáljuk meg az EDUCATION változó eloszlását (tábla, bar grafikon)

```
k<-table(d$EDUCATION)
```

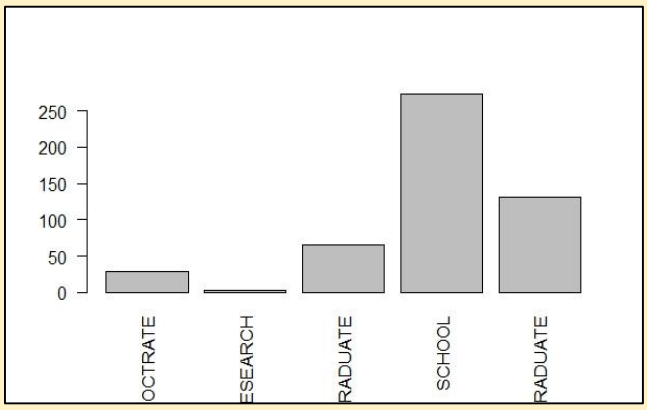
→ education változó kontingenciatáblája;

education változók gyakorisági eloszlását mutatja

```
windows()
```
→ külön ablakba jelenítse majd meg a barplotot

```
par(mar=c(16,4,4,4),cex=0.7)
```
→ tulajdonságok: mar(margó) lent 16; bal, fent, jobb margó 4; cex(betűméret) 0.7

```
barplot(k,las=2)
```
→ barplot a gyakorisági eloszlásról



9.6. FELADAT

Számoljuk ki az INCOME átlagát és szórását EDUCATION kategóriánkként!

```
aggregate(formula=INCOME~EDUCATION, data=d, FUN=mean)
```
→ az adathalmazból statisztikai összefoglalót kreál, itt: az EDUCATION kategóriánként átlagjövedelmet számol

formula=X~Y → Y kategóriánként X adathalmazra alkalmazza a FUN-t

data: d-ben található X,Y adathalmaz

FUN: function – itt: átlag

```
                EDUCATION    INCOME
1                DOCTRATE  55.10714
2 POST DOCTORAL RESEARCH 151.50000
3           POST GRADUATE  60.19697
4                  SCHOOL  41.05128
5         UNDER GRADUATE  46.16794
```

```
aggregate(formula=INCOME~EDUCATION, data=d, FUN=sd)
```
→ ugyanaz, mint az előző, csak a FUN=szórás

```
                EDUCATION   INCOME
1                DOCTRATE 39.44273
2 POST DOCTORAL RESEARCH 36.06245
3           POST GRADUATE 39.73769
4                  SCHOOL 30.90030
5         UNDER GRADUATE 33.85243
```

Még egyszerűbben:

```
aggregate(formula=INCOME~EDUCATION, data=d, FUN=function(x) c(mean(x),sd(x))
)
```

→ function-ben az átlag és a szórás összefűzve

```
                EDUCATION  INCOME.1  INCOME.2
1                DOCTRATE  55.10714  39.44273
2  POST DOCTORAL RESEARCH 151.50000  36.06245
3           POST GRADUATE  60.19697  39.73769
4                  SCHOOL  41.05128  30.90030
5          UNDER GRADUATE  46.16794  33.85243
```

```
aggregate(formula=INCOME~EDUCATION, data=d, FUN=function(x) {c(min(x),max(x)
,mean(x),sd(x))})
```

→a jövedelem átlaga és szórása mellett minimum és maximum értékek is

```
                EDUCATION  INCOME.1  INCOME.2  INCOME.3  INCOME.4
1                DOCTRATE  23.00000 186.00000  55.10714  39.44273
2  POST DOCTORAL RESEARCH 126.00000 177.00000 151.50000  36.06245
3           POST GRADUATE  15.00000 221.00000  60.19697  39.73769
4                  SCHOOL  13.00000 253.00000  41.05128  30.90030
5          UNDER GRADUATE  14.00000 249.00000  46.16794  33.85243
```

9.7. FELADAT

Alkalmazzunk lineáris regressziós modellt az AGE és INCOME változókra (készítsünk plot ábrát és vonjuk le belőle a következtetéseket)!

```
elsoregresszio<-lm(INCOME~AGE, data=d)
```

→ lineáris regresszió (beépített függvény)

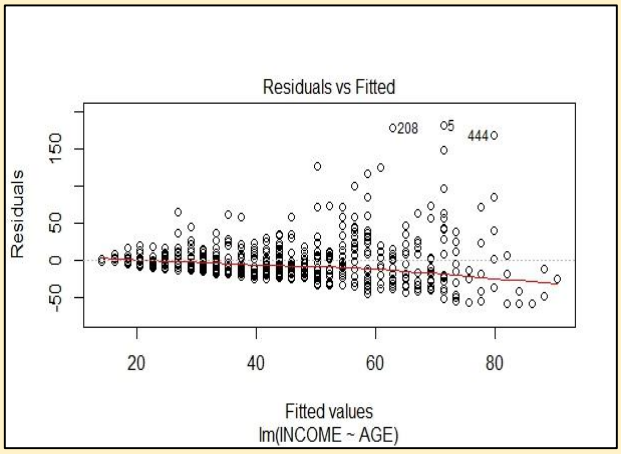
```
Call:
lm(formula = INCOME ~ AGE, data = d)

Coefficients:
(Intercept)          AGE
    -28.266        2.121
```

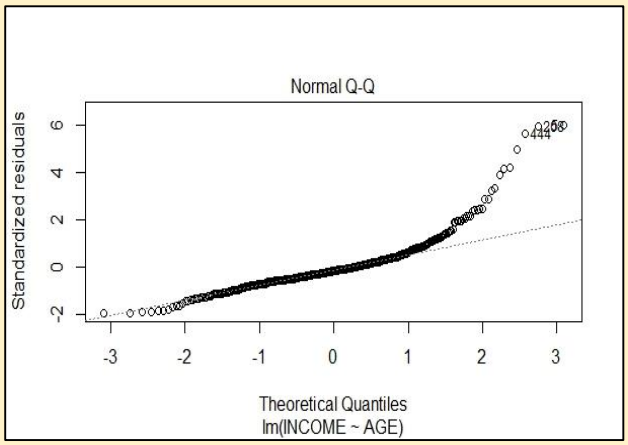
```
summary(elsoregresszio)
Call:
lm(formula = INCOME ~ AGE, data = d)

Residuals:
    Min      1Q  Median      3Q     Max
-58.256 -16.357  -4.737   9.539 181.589

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -28.2660     6.0887  -4.642 4.41e-06 ***
AGE           2.1208     0.1692  12.535  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

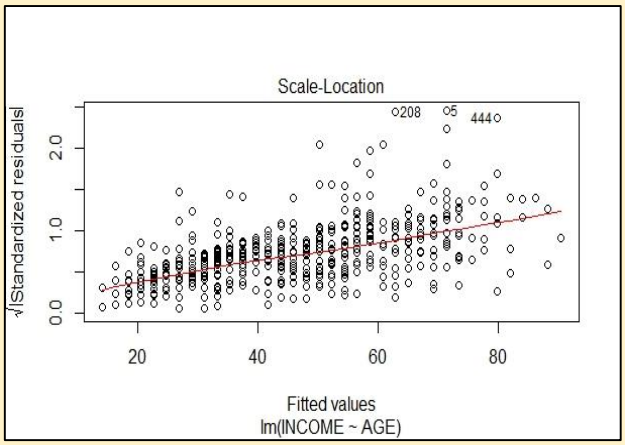
```
Residual standard error: 30.24 on 498 degrees of freedom
Multiple R-squared:  0.2398,Adjusted R-squared:  0.2383
F-statistic: 157.1 on 1 and 498 DF,  p-value: < 2.2e-16
```

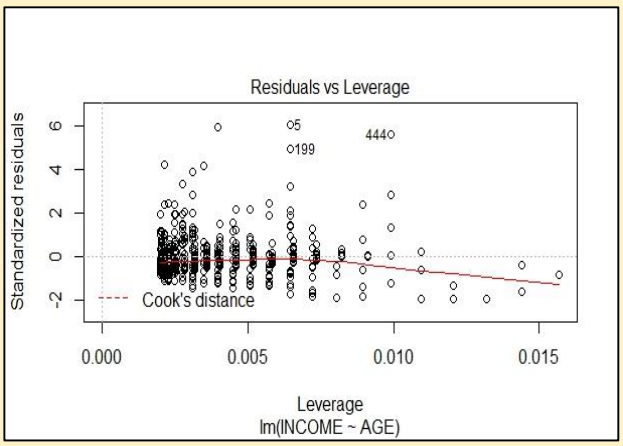
```
str(elsoregresszio)
```
→ structure – felépítése az elsoregresszio állománynak

```
plot(elsoregresszio)
```
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:



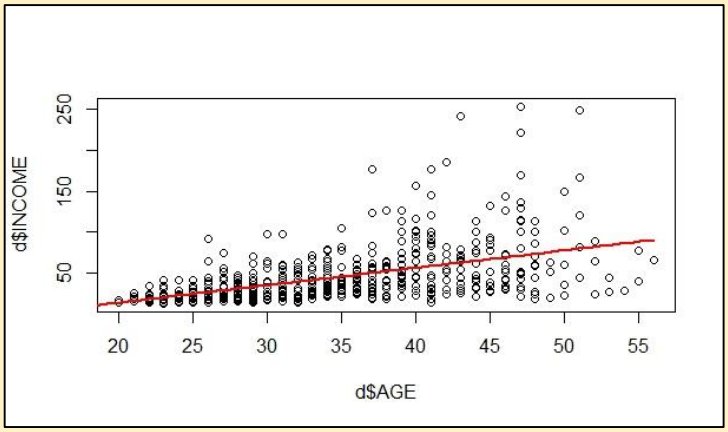







```
windows();
plot(d$AGE, d$INCOME);
curve(2.121*x-28.266,c(10,60),add=TRUE,col="red",lwd=2);
#vagy;
curve(elsoregresszio$coefficients[2]*x+elsoregresszio$coefficients[1],c(10,60
  ),add=TRUE,col="red",lwd=2);
```

→ jövedelem ábrázolása a kor függvényében + regressziós egyenes a summary (elsoregresszio) eredményeiből



ha több magyarázó változót alkalmaznánk:

```
masodikregresszio<-lm(INCOME~AGE+YEARSEMPLOYED,data=d)
```

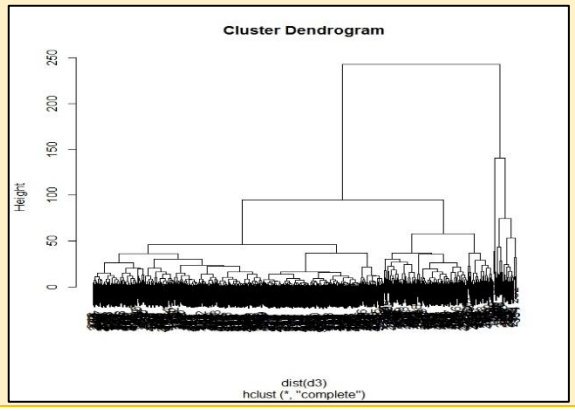
9.8. FELADAT

Vegyük ki az adattáblából a numerikus változókat (2,4,5,6,7) és készítsünk rá dendogramot, illetve 3 elemű klaszterelemzést! (Mire következtetünk?)

```
d3<-d[,c(2,4:7)];
klaszterezes<-hclust(dist(d3))
```

#→ dist: távolságmátrixot készít

```
windows();
plot(klaszterezes)
```




```
kmeans(d3,3)
```

→ 3 klaszterre osztja

Eredmény:

```
K-means clustering with 3 clusters of sizes 28, 129, 343

Cluster means:
       AGE YEARSEMPLOYED    INCOME  CARDDEBT OTHERDEBT
1 44.25000     21.071429 152.00000 5.0031281  9.887050
2 39.60465     13.806202  69.70543 2.4667243  4.806950
3 32.64140      5.825073  28.64723 0.9377824  1.914789

Clustering vector:500 elemet egyesével megmutatja melyik klaszterbe osztotta
be
  [1] 3 2 2 3 1 2 2 2 3 1 2 3 3 3 3 3 3 3 2 3 3 3 3 2 1 2 3 3 2 3 2 2 3 3 3
 [36] 3 3 3 3 1 3 1 3 1 3 2 3 3 3 2 2 3 3 2 2 3 3 3 2 3 2 3 2 2 3 3 2 3 3 3
 [71] 2 2 2 3 3 3 3 3 1 2 2 2 1 3 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 3 3 1 2 3 3
[106] 2 3 3 2 2 2 3 3 3 3 3 3 3 2 3 3 3 3 2 3 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3
```

```
…

Within cluster sum of squares by cluster:
[1] 54307.28 41929.44 57387.35
 (between_SS / total_SS =  76.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

10. Cholesky dekompozíció

Jelen fejezet Medvegyev Péter - Száz János: A meglepetések jellege a pénzügyi piacokon (34-35. oldal) tankönyvben ismertetett eljárásra épül, és R-ben való implementálását tartalmazza.

Lépések

$$C = \begin{bmatrix} c_{11} & C_{21}' \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} a_{11} & A_{21}' \\ 0 & A_{22}' \end{bmatrix} = \begin{bmatrix} a_{11}^2 & a_{11}A_{21}' \\ a_{11}A_{21} & a_{21}A_{21}' + a_{22}A_{22}' \end{bmatrix}$$

Innen

$$a_{11} = \sqrt{c_{11}}$$

$$A_{21} = \frac{C_{21}}{\sqrt{c_{11}}}$$

$$A_{22}A_{22}' = C_{22} - A_{21}A_{21}'$$

Lépések szövegesen

1. A bal felső elemnek vesszük a négyzetgyökét.
2. Ezzel végig osztjuk az oszlopának többi elemét.
3. Képezzük ennek az n-1 elemű új vektornak az önmagával alkotott diadikus szorzatát.
4. Ezt a mátrixot levonjuk a mátrix jobb alsó sarkából.
5. Az eredményül kapott (n-1)X(n-1) mátrixra megismételjük az előző műveleteket.

10.1. FELADAT

Készíts el egy függvényt, ami elvégzi a Cholesky-dekompozíciót egy 2X2-es mátrixra!

```
cholesky <- function(m){

n <- nrow(m);

if(n==2){
choly <- matrix(c(sqrt(m[1,1]),m[2,1]/sqrt(m[1,1]),0,sqrt(m[2,2]-
(m[2,1]/sqrt(m[1,1]))^2)),2,2);
return(choly)
}}
```

→ ha igaz, hogy n=2, azaz 2 sort talál (2x2-es mátrix), akkor létrehoz egy choly nevű 2x2-es mátrixot, melynek elemei:

1. choly mátrix 1. sor 1. oszlopa (bal felső eleme): m mátrix bal felső elemének gyöke
2. choly bal alsó eleme: m mátrix bal alsó eleme osztva az 1. lépés eredményével
3. choly jobb felső eleme: nulla
4. choly jobb alsó eleme: [m mátrix bal alsó eleme-[(m mátrix bal alsó elem/bal felső elem gyöke)^2]]^0,5 = (m mátrix bal alsó eleme-(2. lépés eredménye^2))^0,5

10.2. FELADAT

Készíts el egy függvényt, ami elvégzi a Cholesky-dekompozíciót egy nXn-es mátrixra, és emellett figyel arra is, hogy ha 2X2-es a mátrix, akkor a 10.1. feladatban elkészített kódod fusson le!

```
cholesky <- function(m){

n <- nrow(m);
if(n==2){
choly <-
matrix(c(sqrt(m[1,1]),m[2,1]/sqrt(m[
1,1]),0,sqrt(m[2,2]-
(m[2,1]/sqrt(m[1,1]))^2)),2,2);
return(choly)};

choly <- matrix(0,n,n);


a <- sqrt(m[1,1]);


choly[1,1] <- a;

choly[-1,1] <- m[-1,1]/a;
diadikus <- cbind(choly[-1,1]) %*%
rbind(choly[-1,1]);

m1 <- (m[-1,-1] - diadikus);




choly[-1,-1] <- cholesky(m1);
choly}
```

→ nxn-es mátrix, melynek minden eleme 0

→ alapnak felvesszük az első elem gyökét, ezt *a*-val jelöljük
→ mátrix bal felső eleme = a

→ choly mátrix első oszlopát (oszlopvektorként) szorozzuk choly mátrix első oszlopával (sorvektorként)
→ a diadikus mátrixot levonjuk az m mátrix csonkított alakjából (csonkított alak=első sor, első oszlop levágásával)

→ az előző lépés eredményeként kapott mátrixra elvégezzük a felbontást, és az eredményt bemásoljuk a mátrix jobb alsó sarkába

10.3. FELADAT

3 részvény hozamaira a következő korrelációs mátrixot kaptuk 1998-2007-es adatokra:

korreláció	MOL	OTP	Richter
MOL	1,00	0,68	0,57
OTP	**0,68**	1,00	0,68
Richter	0,57	**0,68**	1,00

havi loghozam	MOL	OTP	Richter
átlag	1,5%	2,2%	0,4%
szóras	10,7%	11,0%	12,8%

Ábrázoljuk annak a portfólió hozamának hisztogramját, amely egyenlő súlyokkal tartalmazza a fenti 3 részvényt!

```
korrel<- matrix(c(
1,0.68,0.57,0.68,1,0.68,0.57,0.68,1),
3,3);

cholika<-cholesky(korrel);
n=10000;


minta<-t(cholika %*%
matrix(rnorm(3*n),3,n));









trendesminta<-cbind
(minta[,1]*0.107+0.015,minta[,2]*0.11+
0.022,minta[,3]*0.128+0.004);









MC<-trendesminta%*% c(1/3,1/3,1/3);
hist(MC);


sd(MC);


mean(MC)
```

→ korrelációs mátrix létrehozása

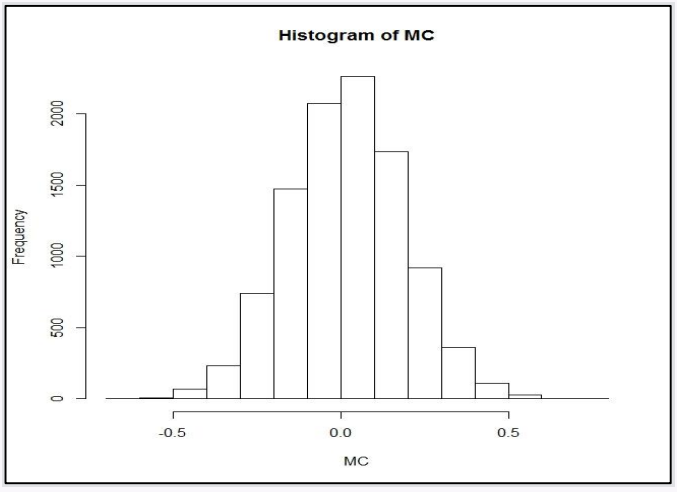
→ cholesky dekompozíció alkalmazása a korrel mátrixra

→ normális eloszlású véletlen számokat generálunk egy 3x10000-es mátrixba, amit megszorzunk *cholika*-val (a korrelációs mátrix cholesky dekomponzíciójával) => korrelált mintát kapunk → ennek vesszük a transzponáltját (*t* beépített függvénnyel)

→ MOL átlag (1,5%) + minta első oszlopa * MOL szórás (10,7%)
→ OTP átlag (2,2%) + minta második oszlopa * OTP szórás (11%)
→ Richter átlag (0,4%) + minta harmadik oszlopa * Richter szórás (12,8%)

→ a részvények árfolyamának loghozamából összesúlyozzuk a portfólió hozamát
→ átlag `[1] 0.1713256`

→ szórás `[1] 0.01637355`



11. Wiener folyamat, GBM

Jelen fejezet „A Wiener-folyamat, Geometriai Brown-mozgás, valamint az Itó-folyamat pénzügyi jelentőségének bemutatása” című komplex tétel főbb fogalmainak R-ben való implementálása.

11.1 Markov-folyamat

11.1. FELADAT

Pisti otthon van: p valószínűséggel átmegy a barátnőjéhez, q valószínűséggel az egyetemre megy és r valószínűséggel a kocsmában tölti a délutánt. Szimuláld le a mozgását!

Írjunk egy függvényt, ami az átmenetvalószínűségek függvényében legenerál egy lehetséges realizációt!

```
pisti     <-     function(p_baratno,
q_egyetem){
a <- runif(1);


if(a<p_baratno){return("baratno")}
;




if(a<p_baratno+q_egyetem){return("e
gyetem")};




return("kocsma")
}

#vagy

pisti     <-     function(p_baratno,
q_egyetem){
a = runif(1);

if(a<p_baratno)
        {return("baratno")
}else{
        if(a<q_egyetem+p_baratno)
                {return("egyetem")
        }else{
                return("kocsma")}
        }
}

pisti(1/3,1/3);
pisti(1/3,1/3);
pisti(1/3,1/3);
pisti(1/3,1/3);
```

→ 1 db véletlen számot generál, ez határozza meg, hogy hova megy Pisti
→ ha igaz, hogy a kisebb, mint annak a való színűsége, hogy átmegy a barátnőjéhez (p) akkor adja megoldásnak, hogy „baratno”
→ ha igaz, hogy *a* kisebb, mint annak a való színűsége, hogy átmegy a barátnőjéhez (p) + annak a valószínűsége, hogy egyetemre megy (q), akkor adja megoldásnak, hogy „egyetem”
#→ minden más esetben azt adja megoldásnak, hogy kocsmába megy

```
[1] "egyetem"
[1] "kocsma"
[1] "baratno"
[1] "kocsma"
```

11.2. FELADAT

Peti a következő átmenetvalószínűségekkel bolyong. 1,2,3, ….. 1000 helyváltozást követően hol van?

innen–ide megy	**otthon**	**egyetem**	**barátnő**	**kocsma**
otthon	0	0,3	0,5	0,2
egyetem	0,1	0	0,4	0,5
barátnő	0,5	0,1	0,3	0,1
kocsma	0,4	0	0,1	0,5

- folyamat periódusról periódusra 4 világállapot között bolyong =>
- minden sorban az elemek összege 1, de az oszlopokban nem feltétlenül
- i-edik állapotból j-dik állapotba kerülésnek a valószínűsége mátrix i-dik sorának j-edik eleme

```
A<-matrix(c(0,0.1,0.5,0.4,0.3,0,0.1,0,0.5,0.4,0.3,0.1,0.2,0.5,0.1,0.5),4,4)
```

#→ Ha az i-dik periódusban az állapotvalószínűségek vektora v, akkor az i+1-dik periódusban v%*%A

Ha az i-dik állapotvalószínűségek egyenletes eloszlást követnek, akkor az i+1-dik:

```
rep(0.25,4)%*%A
```

```
     [,1] [,2]  [,3]  [,4]
[1,] 0.25  0.1 0.325 0.325
```

Két lépés utáni átmenetmátrix (i-ből j-be jutás):

```
A%*%A
```

```
     [,1] [,2] [,3] [,4]
[1,] 0.36 0.05 0.29 0.30
[2,] 0.40 0.07 0.22 0.31
[3,] 0.20 0.18 0.39 0.23
[4,] 0.25 0.13 0.28 0.34
```

Két lépés utáni állapotvalószínűségek egyenletes kiinduló eloszlás esetén:

```
rep(0.25,4)%*%A%*%A
```

```
       [,1]   [,2]  [,3]  [,4]
[1,] 0.3025 0.1075 0.295 0.295
```

k. lépés után → mátrix k. hatványa:

```
M<-diag(4)
```

#egységmátrix létrehozása

```
for(i in 1:2016) {M<-M%*%A}
```

#mátrix „hatványozása”

```
M
```

```
          [,1]     [,2]     [,3]      [,4]
[1,] 0.2830189 0.115903 0.309973 0.2911051
[2,] 0.2830189 0.115903 0.309973 0.2911051
[3,] 0.2830189 0.115903 0.309973 0.2911051
[4,] 0.2830189 0.115903 0.309973 0.2911051
```

Vegyük észre, hogy elég sok helyváltoztatást követően a tartózkodási valószínűség nem függ attól, honnan indult!
[2016. hatvány = 2016. lépés után az átmenetmátrix.
Elemek jelentése: i-ből j-be való átmenés átmenetvalószínűsége 2016 lépés után
A mátrix minden sora közelítőleg megegyezik→ kezdeti kiindulási pont kihal a folyamatból (Markov), a jövőre nincs hatással, hogy a távoli múltban hol voltunk]

11.2 Wiener-folyamat

11.3. FELADAT

Ábrázold a Wiener-folyamat n darab trajektoriáját [0,T] intervallumon!

Általános alakban:

```
wiener<- function(n,T,dt=0.01){

ido<-seq(0,T,by=0.01);

dw<-
matrix(rnorm(n*T/dt),T/dt,n)*sqrt(
dt);

w<-apply(dw,2, cumsum);

w<-rbind(0,w);

 matplot(ido,w,type="l", lty=1)

};
w<-wiener(10,3)
```

Megjegyzések:

-minden egyes időpillanatban a dt négyzetgyökének (szórás) megfelelően normális eloszlás szerint ugrál föl vagy le; kéne n db olyan vektor, aminek hossza T-1, és tartalmazza a növekményeket, annyi darab legyen ahány trajektóriát akarunk - ezek lesznek a Wiener-folyamat növekményei), várható értéke=0, szórás=gyök(dt)
-Wiener-folyamat= növekmények összege
-mátrix oszlopait úgy kell transzformálni/kumulálni, hogy 1.sorban első növekmény, 2.sorban első két növekmény összege, stb.
-növekmény norm eloszlás, szórás dt gyökével arányos

→ vektor, mely 0-ban kezdődik T-ben van a lejárata, 0.01 lépésközzel
→ T/dt x n-es mátrix, amit véletlen számokkal töltünk fel; a mátrix oszlopai a trajektóriák
→ apply függvény a dw mátrix oszlopait (1=sor, 2=oszlop) kumulálja
→ tetejére csupa nullát (0-ból indul), T+1 db sorból áll

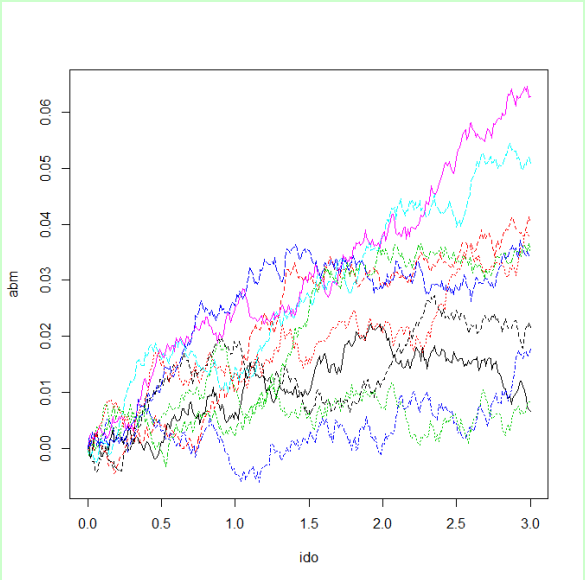
11.3 Aritmetikai Brown mozgás

$$dS = \mu dt + \sigma dW$$

11.4. FELADAT

Ábrázolj n trajektóriáját az aritmetikai Brown mozgásnak!

```
ABM<- function(n, T, mu, szigma, dt=0.01) {
        ido<-seq(0,T,by=0.01);
        szigmadw<-matrix(rnorm(n*T/dt),T/dt,n)*sqrt(dt)*szigma;
        szigmaW<-apply(szigmadw,2,cumsum);
        szigmaW<-rbind(rep(0,n), szigmaW);
        abm<- szigmaW+mu*ido;

matplot(ido, abm, type = "l")
};
ABM(10,3,0.01,0.01)
```



11.4 Poisson

11.5. FELADAT

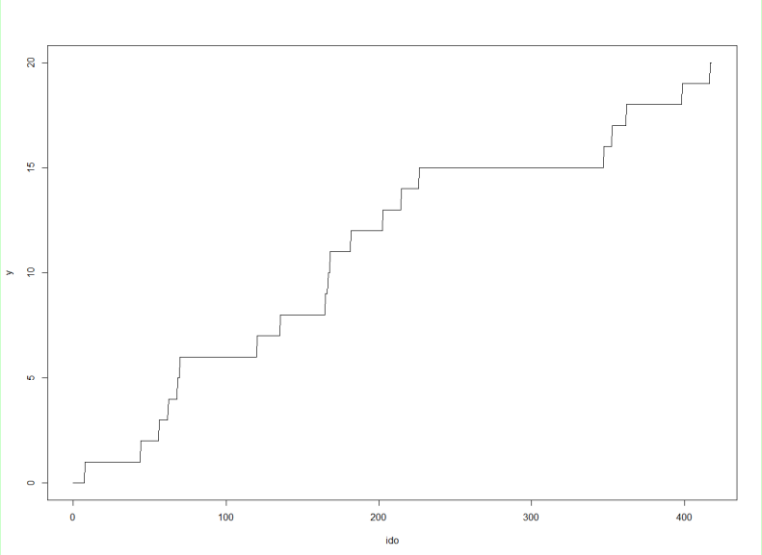
Szimulálj le egy Poisson folyamatot!

```
a <- rexp(20, rate=0.05);


a <- cumsum(a);
ido <- seq(0,max(a)+1,length=1000);

y <- rep(0,length(ido)) ;

for (i in
1:length(ido)){y[i]=sum(ido[i]>a)};

plot(ido,y, type = "l")
```

→exponenciális eloszlást követ két egymást követő esemény között eltelt idő, átlag=20

→ bekövetkezési időpontok
→a folyamatot az utolsó eseményt követő egy időegységig megfigyeljük
→ y vektor, a folyamat értékeit fogja tartalmazni

→ az i-dik időpillanatban felvett értéket az i-dik időpillanatig bekövetkezett események száma



11.5 GBM folyamat

$$dS = \mu S dt + \sigma S dW$$

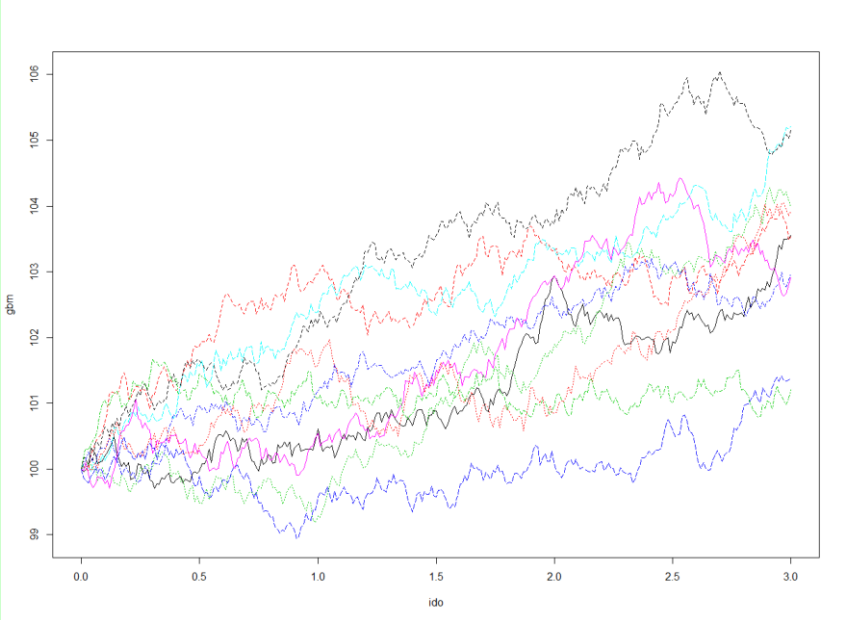
$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$$

11.6. FELADAT

Ábrázolj egy trajektóriáját a GBM folyamatnak!

11.4 feladat alapján írjuk fel:

```
GBM<- function(n, T, mu, szigma,S0 , dt=0.01) {
        ido<-seq(0,T,by=0.01);
        szigmadw<-matrix(rnorm(n*T/dt),T/dt,n)*sqrt(dt)*szigma;
        szigmaW<-apply(szigmadw,2,cumsum);
        szigmaW<-rbind(rep(0,n), szigmaW);
        abm<- szigmaW+(mu-szigma^2/2)*ido;
        gbm<-S0*exp(abm);

matplot(ido, gbm, type = "l")
};
GBM(10,3,0.01,0.01,100)
```



12. Shiny

Interaktív ábrák:

Részletesen elérhető tananyag:
http://shiny.rstudio.com/tutorial/

Ezen belül:
http://shiny.rstudio.com/tutorial/lesson1/
http://shiny.rstudio.com/tutorial/lesson3/
http://shiny.rstudio.com/tutorial/lesson7/

Ismétlés annuitás faktor:

$$AF = \frac{1}{r}\left(1 - \frac{1}{(1+r)^t}\right)$$

Annuitásos hitel esetén a törlesztő részlet nagysága:

$$CF = \frac{C}{\frac{1}{k}\left(1 - \frac{1}{(1+k)^t}\right)}$$

Annuitásos hitel esetén a tőketörlesztés nagysága:

$$tőketörlesztés = CF(1+r)^{t-1} - Cr(1+r)^{t-1}$$

12.1. FELADAT

Készítsük el a következő ábrát:
https://vpir.shinyapps.io/annuitas/
(Recap – nyilvános oldalra való feltöltés → shinyapps.io-ra kell feltölteni az ui.R-t és a server.R-t [ehhez egy teljesen új mappa szükséges])

1.lépés: programozzuk le hagyományosan

2.lépés: készítsük el az ui.R és a server.R fájlunkat

ui.R

server.R

1. lépés: Értsük meg!

Bemenő adatok: r (nem a diszkontáláshoz), t, C (összeg)

T	Fennálló Hitelállomány	Tőketörlesztés	Kamatfizetés	CF
1	**C**	TT1=CF-C*r=**CF-kamfiz1**	kamfiz1=**C*r**	CF
2	C-(CF-C*r)=**C-TT1**	TT2=CF-[(C-(CF-C*r))*r]=**CF-kamfiz2**	kamfiz2=(C-(CF-C*r))*r=**(C-TT1)*r**	CF
3	C-(CF-C*r)- [CF-[(C-(CF-C*r))*r]]=**C-TT1-TT2**	TT3=CF- [[C-(CF-C*r)- [CF-[(C-(CF-C*r))*r]]]*r]=**CF-kamfiz3**	kamfiz3=[C-(CF-C*r)- [CF-[(C-(CF-C*r))*r]]]* r=**(C-TT1-TT2)*r**	CF

1. AF kiszámolása után, C/AF képletből megkapjuk a CF értékeket. (Annuitás miatt a CF minden évben ugyanannyi!)
2. kamatfizetés: adott évi fennálló hitelállomány adott százaléka (r)
3. tőketörlesztés: CF-tárgyévi kamatfizetés

2. lépés: A folyamat

```
r<-0.1;
t<-10;
C<-5;
CF<-C/(1/r*(1-1/(1+r)^t));

d<-data.frame(ev=1:t,cf=CF,
toket=CF*((1+r)^((1:t)-1))
-C*r*((1+r)^((1:t)-1)),kamatfiz=1);

d$kamatfiz<-d$cf-d$toket;



counts<-rbind(d$toket, d$kamatfiz);
barplot(counts)
```

→ adatok bevitele, CF meghatározása r=10%, t=10év, CF=5, CF képlet felírása

→ data frame készítése

→ kamatfizetés kiszámítása a data.frame-be, felhasználva a korábban kiszámolt értékeket

Sorokba kell rendezi a tőketörlesztést és a kamatfizetést, mert a barplot csak vektort és mátrixot képest ábrázolni, és akkor ábrázolja egymás felett egy oszlopban a tőke- és kamatfizetést, ha egymás alatt vannak az értékek (lásd a `?barplot`).

Interaktivizálás: ui.R, server.R fájl elkészítése, a shiny megfelelő működése miatt
Shiny létrehozása R Studioban:

Fájl – New file – R script – Mentés – ui.R – ui.R-be az alább lévő ui.R anyaga – mentés ugyanez server.R-rel

ui.R

```
library(shiny):
shinyUI(


fluidPage(




titlePanel("Annuity Calculator"),


sidebarLayout(


      sidebarPanel(


            sliderInput("t",

                  "t",
                  min = 1,
                  max = 40,
                  value = 15),

            sliderInput("r",
                   "r",
                   min = 0.01,
                   max = 0.25,
                   value = 0.02)
    ,

            numericInput("C",
                    "Hitel MFt",
                    value = 1)
    ),
      mainPanel(
plotOutput("distPlot")
    )
   )
))
```

→shiny library betöltése
→ShinyUI - létrehoz egy felhasználó interface-az újabb verziókban erre már nincs szükség
→fluidpage – az oldal kinézetének beállításához szükséges függvényeket olvassa be, automatikusan elrendezi a rendelkezésre álló böngészőméretnek megfelelően az elemeket

→cím: Annuity Calculator

→felosztja 1/3 -2/3 arányban az oldalt a sidebarpanel és a mainpanel között

→szürke keretben található elemek

→beleraktam egy t-re hivakozható, t nevű csúszkát, aminek minimuma 1,
maximuma 40, és
alapértelmezett érték 15)

→ebbe a felhasználó írhat be egy számértéket

→főpanel, ide rajzolódik ki az ábra
ide vár egy olyan ábrát, a server.R fájlból, aminek disPlot a hivakozási
neve

server.R

```
library(shiny);

shinyServer(

function(input, output) {


output$distPlot <- renderPlot({




CF<- input$C/(1/input$r*(1-1/
((1+input$r)^input$t)));

d<-data.frame(ev=1:input$t, cf=CF,

toket=CF*((1+input$r)^((1:input$t)
-1))-input$C*input$r*((1+input$r)
^((1:input$t)-1)), kamatfiz=1);

d$kamatfiz=d$cf-d$toket;

counts      <-      rbind(d$toket,
d$kamatfiz);

barplot(counts)

      })
})
```

→shinyServer nem szükséges függvény, régen csak ezzel működött
→létrehozunk egy függvény egy bejövő és kimenő listával

→ a renderPlot olyan objektumot hoz létre, amelyet át lehet adni az ui.R nevű fájl plotOutput függvénynének, azaz egy ábrát kell készíteni a függvényben, amely a mi esetünkben egy barplot

→CF képlete, de minden adat elé az input meghívással (UI-ból hívja meg az adatokat)

→ábrázolás
→itt van vége a renderPlot-nak
→itt van vége a függvénynek shiny library betöltése

13. Gyakorlás

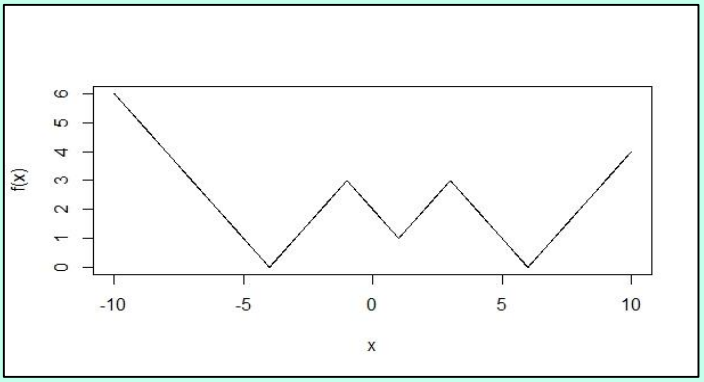
13.1. FELADAT

Ábrázoljuk a következő függvényt: $f = \left||x-1|-2\right|-3\Big|$ a [-10,10] intervallumon!

```
f <- function(x) {abs(abs(abs(x-1)
-2)-3)};
x <- seq(-10,10, length = 1000);


plot(x, f(x), typ = "l")
```

→ a feladatban meghatározott függvény

→[-10;10] intervallumon szeretnénk ábrázolni a függvényt, ezért létrehozunk egy 1000 hosszúságú vektort
→ függvény ábrázolása, ahol x tengely (-10, 10) intervallumú, y tengelyen pedig az *f(x)* értékei szerepelnek




```
#vagy egyszerűbben
f <- function(x) {abs(abs(abs(x-1)-2)-3)};
plot(f, xlim=c(-10,10))
```

13.2. FELADAT

Szimuláld az annuitás-képletet $AF(t,r) = \frac{1}{r} * \left(1 - \frac{1}{(1+r)^t}\right)$!

Azaz írj egy függvény, amelynek bemenő paraméterei az r és a t, és a megoldása két azonos szám legyen. Az első szám a diszkontálás alkalmazásával kapott érték legyen $\sum_{i=1}^{t} \frac{1}{(1+r)^t}$, míg a második az annuitás képlettel kiszámolt érték legyen.

```
AF <- function(t,r){
pv1 <- sum(1/(1+r)^(1:t));
pv2 <- 1/r*(1-1/(1+r)^t);
c(pv1,pv2);
}

AF(10,0.1)
```

#→ sum(DF(t))
#→ annuitás faktor

```
[1] 6.144567 6.144567
```

13.3. FELADAT

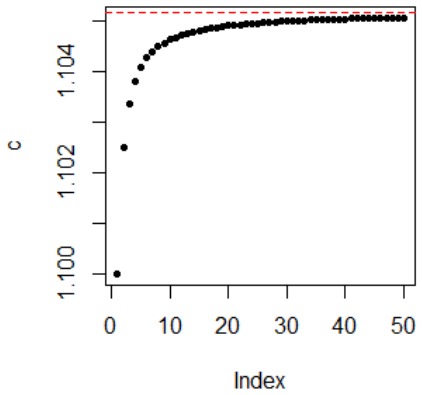
Ábrázold, hogy különböző tőkésítési gyakoriság mellett mennyit ér 1 Ft egy év múlva!

$$C_1 = C_0 * (1 + r)^1$$

$$C_1 = C_0 * \left(1 + \frac{r}{2}\right)^1$$

.....

$$C_1 = C_0 * e^{r*1}$$

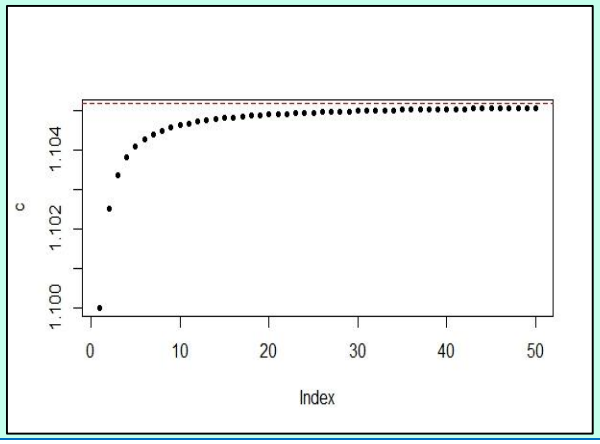



```
EffektivHozamLim <- function(r){
n = 50;
c <- (1+r/(1:n))^(1:n);
plot(c, pch = 20);

abline(h = exp(r), lty = 2, col
= "red");

      };
EffektivHozamLim(0.1)
```

→ elég 50 periódusig nézni
→ fenti képlet, 50 hosszú vektor esetében

→ pont diagram, pch típusa jelenleg telített pötty

→*h* a vízszintes vonalat húz h magasságban



13.4. FELADAT
Írjunk egy függvény, amely megoldja a másodfokú egyenletet a valós számok halmazán!

$$x_{1,2} = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

```
masodfoku<-function(a,b,c) {

if ((b^2-4*a*c)<0) {"nincs mo :P"}




else {

  x1<-((-b)-sqrt(b^2-4*a*c))/(2*a);
  x2<-((-b)+sqrt(b^2-4*a*c))/(2*a);


c(x1,x2)     }}

masodfoku(1,2,3)
```

#→ három változós függvény
#→ valós számok halmazán legyen megoldható, ezért a gyök alatti kifejezésre azzal a feltétellel élünk, hogy ha értéke kisebb, mint nulla, akkor értelmetlennek kezelje, ne számolja végig, hanem szóljon, hogy új értékeket adjunk meg
→ más esetben, azaz ha az előző feltétel nem áll fenn, azaz a gyök alatti kifejezés nagyobb/egyenlő nullával, akkor…
→ két megoldás összefűzése

```
[1] "nincs mo :P"
```

Egyszerűbben:

```
masodfoku <- function(a,b,c){
 D <- b*b-4*a*c;
if (D<0) {"nincs megoldás :P"}
else {
 x1 <- (-b+sqrt(D))/(2*a);
 x2 <- (-b-sqrt(D))/(2*a);
 c(x1,x2)
 }}
```

13.5. FELADAT
Határozd meg, hogy a vállalatnak mennyi darabot éri meg megtermelni, ha proftimaximalizálni szeretne. A cég fix költsége=10, a változó költsége=$4q^2 + 1q$. Minden egyes terméket fix 15-ért tud értékesíteni.

$$\pi = 15q - (4q^2 + q + 10) \leftarrow max$$

segítség: *optimize* függvény (minimalizál, maximalizál); alapértelmezetten minimalizál

```
optimize(f, interval, ..., lower = min(interval), upper = max(interval),
         maximum = FALSE, tol = .Machine$double.eps^0.25)
```

```
optimize(f=function(q) {15*q-(4*q^2+q+10)},interval=c(0,10),maximum=TRUE)
```

#→ először a vizsgált függvényt kell megadni, majd azt, hogy milyen intervallumon keresse a maximumot (itt: 0 és 10 között), célszerű elég nagy korlátos intervallumot megadni, végül azt, hogy maximalizálni szeretnénk (maximum=TRUE)

13.6. FELADAT
Írjuk ki a prímszámokat n-ig!
Elméleti tananyag:
https://hu.wikipedia.org/wiki/Eratoszthen%C3%A9sz_szit%C3%A1ja

```
N <- 1000;
p <- rep(TRUE, N);
p[1] <- FALSE;
for (i in 2:(sqrt(N)+1)){
if (p[i]){p[seq(2*i,N,by=i)] <- FALSE
        }
};

print((1:N)[p])
```

1000 hosszú TRUE vektor

for ciklus, 2-től vizsgálódik gyök(N)+1-ig
ha i prím, az összes többszörösét kihúzzuk a prímek közül

kiírja azokat az értékeket (1-től 1000-ig), amelyek TRUE-k maradtak

Dupla ciklussal:

```
Primek <- function(n){
lista <- rep(1,n);
lista[1] <- 0;
for (i in 2:(sqrt(n)+1)){
   if (lista[i] == 1){
      for (j in seq(2*i,n,by = i)){lista[j] <- 0}
          }
     };
(1:n)[lista==1]
}
```

13.7. FELADAT
Határozd meg az a legkisebb számot 100.000 és 200.000 között, amely osztható 1234-el!

```
a <- 100000;
b <- 200000;
z <- 1234;
i <- z;
while(i < a){i <- i+z};
i
```

amíg *i* kisebb 100.000-nél addig *i* *z* értékével nő (i=z többszörösei)
kiírja azt az *i* értéket, ami 100.000-200.000 közé esik és *z* többszöröse

VAGY

```
i <- a;
while(i %% 1234){i <- i+1};
i
```

amíg *i* osztva 1234-gyel nem nulla maradékot ad, addig *i=i+1*

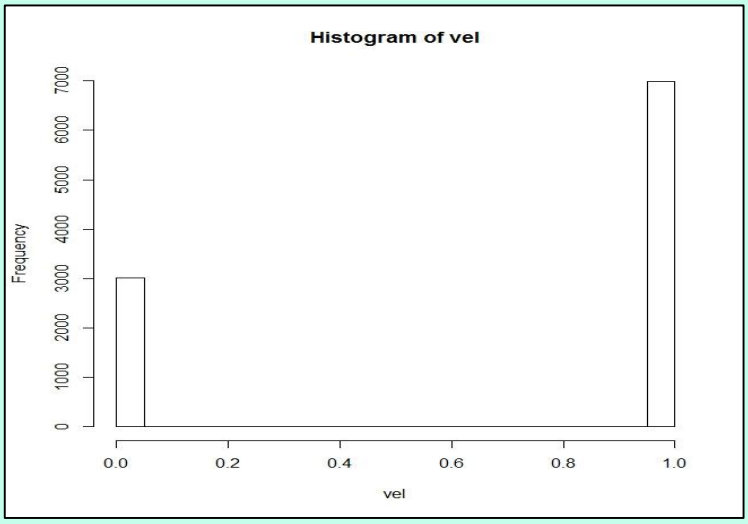
13.8. FELADAT

Generáljunk 10.000 darab véletlen számot, 70% eséllyel 1 az értéke és 30% eséllyel 0 az értéke. Ábrázold hisztogramon az eredményt!

```
vel<-runif(10000);

vel<-(vel>=0.3)*1;
hist(vel)
```

10 000 db 0 és 1 közötti véletlen szám generálása

ha a véletlen szám 0,3-nál nagyobb, akkor a logikai vektornak az értéke TRUE=1



13.9. FELADAT

Írj egy kódot, ami a 30-nál kisebb páratlan számok köbeinek összegét meghatározza!

```
x <- seq(1,30,by = 2);


sum(x^3)
```

→ vektor, mely csak a páratlan számokat tar talmazza 1-től 30-ig

→ köbök összege

101025

14. Hivatkozások

Tudományos könyvek:

Medvegyev P., Száz János [2010]: A meglepetések jellege a pénzügyi piacokon: kockázatok vételre és eladásra. Budapest: Nemzetközi Bankárképző Központ Zrt.

Internetes források:

Központi Statisztikai Hivatal - Minden helység adata. Elérhető: www.ksh.hu/docs/hun/hnk/hnk_2013.xls Letöltés dátuma. 2017. 01. 13.

Magyar Keresztnevek Tára - Utónév statisztika 2014. Elérhető: http://magyarnevek.hu/nevek/utonevstatisztika/2014 Letöltés dátuma. 2017. 01. 13.

Wikipedia- NAN. Elérhető: https://en.wikipedia.org/wiki/NaN Letöltés dátuma. 2017. 01. 13.

Wikipedia – Enumerated type. Elérhető: https://en.wikipedia.org/wiki/Enumerated_type Letöltés dátuma. 2017. 01. 13.