Poglavje 1

Modul SAS/IML

Pri vajah bomo izpostavili le nekatere matrične operacije in to predvsem z namenom, da bomo z njimi spoznali modul IML in si ilustrirali primere, ki jih obravnavamo pri predavanjih.

V modulu SAS/IML (Iteractive Matrix Language) lahko programiramo ali pa le računamo z matrikami. Obstajajo številne matrične operacije, ki se izvajajo na celotnih matrikah. Modul je interaktiven in dinamičen. Z njim obdelujemo podatke in prikazujemo grafično.

Proceduro IML zaženemo z ukazom iz kode 1.1. Da bomo poenostavili izpise, s stavkom v drugi vrstici zahtevamo, da se matrike izpisujejo tudi na oknu Output. Pri tem nismo kot zadnji stavek uporabili RUN.

Algoritem 1.1 Splošna sintaksa v proc IML

```
proc iml;        /* zagnali smo proceduro, v oknu Log se izpiše IML Ready*/
reset print;     /* rezultati se bodo avtomatično izpisovali v oknu Output*/
```

PROC IML - s stavkom poženemo proceduro IML, v oknu Log se izpiše IML Ready

RESET PRINT - rezultati se bodo avtomatično izpisovali v oknu Output

Pri poimenovanju vektorjev in matrik veljajo enaka pravila kot pri ostalih spremenljivkah v SAS-u. Čeprav SAS ne loči velikih in malih črk, pa tudi ne odebeljenih in navadnih zapisov, na splošno velja, da matrike poimenujemo z velikimi in vektorje z malimi črkami, da smo čim bliže oznakam, ki jih uporabljamo v enačbah. Pri enačbah uporabljamo le po eno črko. Matrike ali vektorje lahko poimenujemo enako, kot jih zapišemo v enačbah, morda pa raje izberemo daljša imena, da je koda bolj berljiva tudi čez dalj časa ali od drugih. Pri tem pazimo, da oznak za skalarje, vektorje in matrike ne pomešamo. Ko pri kreiranju za novo matriko ali vektor uporabimo isti naziv, stare matrike ali vektorja ni več. Da je poimenovanje isto hitro spregledamo, če pišemo z malimi in velikimi črkami - za nas so to različne oznake, v SAS-u pa ne.

1.1 Vektorji

Napišimo vrstični vektor $\mathbf{y}'$ (enačba 1.1) in stolpični vektor $\mathbf{m}$ (enačba 1.2) v SAS-u (koda 1.2). Posebnost v SAS-u je tudi možnost, da kreiramo vektorje z opcijami, ki jih modul ponuja.

$$\mathbf{y}' = \begin{bmatrix} 5 & 2 & 4 & 6 \end{bmatrix} \quad \ldots (1.1)$$

$$\mathbf{m} = \begin{bmatrix} 5 \\ 2 \\ 4 \\ 6 \end{bmatrix} \quad \ldots (1.2)$$

$$\mathbf{j}_5 = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \\ 4 \end{bmatrix} \quad \ldots (1.3)$$

$$\mathbf{j}'_5 = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \end{bmatrix} \quad \ldots (1.4)$$

Pri vnosu vektorjev in matrik (algoritmu 1.2) uporabljamo zavite oklepaje "{" in "}", zaključimo s podpičjem. Vrednosti v vrstici (stolpcih) vektorjev ali matrik ločimo s presledki, vrstice pa ločimo z vejicami.

V algoritmu 1.2 smo vrstične vektorje označili tako, da smo pred ime vektorja napisali črko **t** za transponirano, včasih pa uporabimo črko **p**, ki izvira od angleške besede *prime* za črtico, za oznako vektorja. Tako bi vektorja **ty** in **tj5**, lahko poimenovali tudi **yp** oz. **j5p**, lahko pa bi si raje izbrali oznaki **vy** ali **vj5**, kjer bi črka **v** nakazovala, da gre za vrstični vektor. Zadnja dva vektorja pa smo naredili s funkcijo J(), s katero si pripravimo vektor (in matriko), ki vsebuje iste vrednosti. Prva vrednost v oklepaju je število vrstic, druga vrednost je število stolpcev, tretja vrednost pa se bo pojavila v vektorju pri vseh elementih.

Algoritem 1.2 Zapis ali nastavitev vektorja v proc IML

```
/* ---------------------------------------- zapišimo vrstični in stolpični vektor  */
ty  = {5 2 4 6};    /* ------------------------------------ vrstični vektor  1×4 */
m   = {5,2,4,6};    /* ------------------------------------ stolpični vektor 4x1 */
j5  = J(5,1,4);     /* ------------------------------------ stolpični vektor 4x1 */
tj5 = J(1,5,4);     /* ------------------------------------ vrstični vektor  4x4 */
print ty m j5 tj5;  /* ---------------- morda bi radi vektorje še enkrat izpisali */
print ty,m,j5,tj5;
```

Na naslednji sliki vidimo izpis vektorjev iz kode 1.2 in enačb od 1.1 do 1.4. Zapis iz IML je brez oglatih zaklepajev, z modrim je natiskan naziv vektorja, vrednosti pa so v črnem tisku. Prvi (**ty**) in zadnji (**tj5**) sta vrstična vektorja, drugi (**m**) in tretji (**j5**) pa stolpična.

ty 1 row 4 cols (numeric) 5 2 4 6	ty = {5 2 4 6};
m 4 rows 1 col (numeric) 5 2 4 6	m = {5,2,4,6};
j5 5 rows 1 col (numeric) 4 4 4 4 4	j5 = J(5,1,4);
tj5 1 row 5 cols (numeric) 4 4 4 4 4	tj5 = J(1,5,4);

V kodi 1.3 prikazujemo nekaj pogostih računskih operacij z vektorji, ki se jih bomo posluževali.

Algoritem 1.3 Računajmo z vektorji v proc IML

```
/* ------------------------------ uporabili bomo vektorje iz kode 1.3 */
tyc     = 2.2*ty;       /* ------------------------------------------------- 1 */
tym     = ty*m;         /* ------------------------------------------------- 2 */
mty     = m*ty;         /* ------------------------------------------------- 3 */
tytm    = ty + t(m);    /* ------------------------------------------------- 4 */
vrstica = ty||tj5;      /* ------------------------------------------------- 5 */
vrstici = ty//tyc;      /* ------------------------------------------------- 6 */
print tyc, tym, mty, tytm, vrstica, vrstici;
```

- Najprej smo vektor **ty** pomnožili s skalarjem, kar prikazujemo tudi v enačbi 1.5. Rezultat smo poimenovali **tyc**, ker pričakujemo vrstično vektor (**t**), vrednosti so iz vektorja **y** jn smo ga pomnožili s konstanto - skalarjem (**c**). Rezultat bi sicer lahko poimenovali tudi drugače, npr. **resitev**.

$$\mathbf{y}'_c = 2.2\times\mathbf{y}' = 2.2\times\begin{bmatrix} 5 & 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 11 & 4.4 & 8.8 & 13.2 \end{bmatrix} \quad \text{... (1.5)}$$

- V drugi vrstici kode 1.3 množimo vrstični in stolpični vektor iz enačbe 1.6. Množenje obeh vektorjev prikazujemo tudi v prikazu 1.7, kjer smo napisali vektorja, ki ju želimo množiti, v priporočljivi postavitvi. Elemente obeh vektorjev množimo po parih in zmnožke seštejemo. Dobili smo skupno vsoto kvadratov (*Total sum of square*, TSS). Dobljeni rezultat v kodi smo poimenovali s sestavljanjem črk vektorjev iz zmnožka (**tym**), lahko pa bi izbrali tudi naziv povezan z vsebino rezultata (npr. **tss**), Rezultat je skalar.

$$V_T = \mathbf{y}'\times\mathbf{m} = 5\times5 + 2\times2 + 4\times4 + 6\times6 = 81 \quad \text{... (1.6)}$$

$$\begin{bmatrix} 5 & 2 & 4 & 6 \end{bmatrix} \quad \begin{bmatrix} 5 \\ 2 \\ 4 \\ 6 \end{bmatrix} \quad \text{... (1.7)}$$
$$[81]$$

- V tretji vrstici kode 1.3 množimo ista vektorja, a v drugem zaporedju (enačba 1.8). Vektorja smo pomnožili tudi v prikazu 1.9, iz česar je razvidno, da je rezultat matrika (**A** iz enačbe 1.8), ki ima 4 vrstice in 4 stolpce. Zaključimo lahko, da lahko vektorja pomnožimo v različnih zaporedjih, a rezultata sta različna! Pri vektorjih in matrikah je vrstni red pomemben. Matrika **mty** iz slike **??** je enaka matriki **A**.

$$\mathbf{A} = \mathbf{m}\times\mathbf{y}' \quad \text{... (1.8)}$$

$$\begin{bmatrix} 5 \\ 2 \\ 4 \\ 6 \end{bmatrix} \quad \begin{bmatrix} 5 & 2 & 4 & 6 \\ 25 & 10 & 20 & 30 \\ 10 & 4 & 8 & 12 \\ 20 & 10 & 16 & 24 \\ 30 & 12 & 24 & 36 \end{bmatrix} \quad \text{... (1.9)}$$

- V četrti vrstici kode 1.3 smo sešteli dva vektorja, oba morata biti vrstična ali stolpična in imeti isto število elementov. Ker smo jih v IML sešteli kot vrstična vektorja, jih bomo enako tudi v enačbi 1.10. Rezultata v enačbi 1.10in sliki **??** (**tytm**) sta enaka.

$$v' = \mathbf{y}'+\mathbf{m}' = \begin{bmatrix} 5 & 2 & 4 & 6 \end{bmatrix} + \begin{bmatrix} 5 & 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 10 & 4 & 8 & 12 \end{bmatrix} \quad \text{... (1.10)}$$

- V peti vrstici kode 1.3 smo vektor **vrstica** sestavili tako, da smo vektorja $\mathbf{y}'$ in $\mathbf{j}'_5$ združili zaporedoma. Rezultat ima 9 elementov, prvi štirje so iz vektorja $\mathbf{y}'$ in drugih pet iz vektorja $\mathbf{j}'_5$. Če preverimo vrednosti v enačbi 1.11 in na sliki v nadaljevanju, imamo oba rezultata enaka. Vrstična vektorja imata lahko različno število elementov, medtem ko pri združevanju dveh stolpičnih vektorjev motata vektorja imeti isto število elementov. Uporabili smo nov znak || za zaporedno združevanje oz. vodoravno spenjanje vektorjev in matrik .

$$\mathbf{x}' = \mathbf{y}'\|\mathbf{j}'_5 = \begin{bmatrix} \underbrace{5 \quad 2 \quad 4 \quad 6}_{\mathbf{y}'} & \underbrace{4 \quad 4 \quad 4 \quad 4 \quad 4}_{\mathbf{j}'_5} \end{bmatrix} \quad \text{... (1.11)}$$

- V šesti vrstici imamo še en primer, in sicer smo sestavili dva vrstična vektorja tako, da je prvi ($\mathbf{y}'$) v zgornji vrstici in drugi (v') v spodnji. Oba vrstična vektorja morata vsebovati enako število elementov, če pa združujemo stolpične vektorje, pa lahko vsebujeta različno število elementov. Matrika ima vedno v vseh stolpcih (ali vrsticah) enako število elementov. V proceduri IML imamo za sestavljanje vektorjev ali matrik v smeri eno pod drugim znak // za vertikalno spenjanje. Operacije pri matrični algebri ne poznamo, saj sestavljanje prikazujemo z delnimi matrikami.

$$\mathbf{B} = \mathbf{y}'//\mathbf{v}' \quad \text{... (1.12)}$$

tyc 1 row 4 cols (numeric) 11 4.4 8.8 13.2	tyc = 2.2*ty;
tym 1 row 1 col (numeric) 81	tym = ty*m;
mty 4 rows 4 cols (numeric) 25 10 20 30 10 4 8 12 20 8 16 24 30 12 24 36	mty = m*ty;
tytm 1 row 4 cols (numeric) 10 4 8 12	tytm = ty + t(m);
vrstica = ty||tj5; vrstica 1 row 9 cols (numeric) 5 2 4 6 4 4 4 4 4	
vrstici 2 rows 4 cols (numeric) 5 2 4 6 11 4.4 8.8 13.2	vrstici = ty//tyc;

1.2 Matrike

Tudi matrike lahko v proceduri IML natipkamo. Pri vnosu matrik uporabljamo zavite oklepaje "{" in "}", zaključimo s podpičjem, vrstice v matrikah in vektorjih pa ločimo z vejico. Med elementi v vrstici matrike mora biti vsaj en prazen prostor, zaradi boljše preglednosti pa elemente v različnih vrsticah matrikah lahko podpisujemo.

Kot prvi matriki smo izbrali dve kvadratni matriki in 1.14). Prvo matriko (enačba 1.13) smo zapisali v tri vrstice (algoritem), drugo pa kar v vrstico. Ne glede na zapis z vejicami določimo novo vrstico v matriki.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \quad \ldots (1.13)$$

$$\mathbf{B} = \begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 0 \\ -2 & 0 & 8 \end{bmatrix} \quad \ldots (1.14)$$

Tretja matrika (enačba 1.15) je pravokotna s tremi vrsticami in dvema stolpcema. V algoritmu 1.4 je zapisana pregledno v treh vrsticah in dveh stolpcih, lahko pa bi matriko zapisali tudi v eno vrstico, kjer bi za dvema elementoma iz vsake vrstice napisali vejico.

$$\mathbf{C} = \begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} \quad \ldots (1.15)$$

Četrta matrika (enačba 1.16) je diagonalna. Kasneje bomo prikazali, kako jo enostavneje napišemo. V algoritmu 1.4 jo enostavno kar natipkali.

$$\mathbf{D} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & -5 \end{bmatrix} \quad \ldots (1.16)$$

Algoritem 1.4 Splošna sintaksa v proc IML

```
A={1 2 3 ,        /* ------------------------------------- kvadratna matrika 3x3 */
   3 4 0 ,
   5 6 2 };
B={3 4 7, 3 4 0 ,-2 0 8 }; /* ---------------------------- kvadratna matrika 3x3 */
C={3  4,          /* ------------------------------------- pravokotna matrika 3x2 */
   12 6,
   4  7};
D={4 0  0,        /* ---------------------------------------- diagonalna matrika */
   0 6  0,
   0 0 -5};
I={1 0 0 0,       /* --------------------------------- identična matrika reda 4×4 */
   0 1 0 0,
   0 0 1 0,
   0 0 0 1};
```

V proc IML lahko kreiramo tudi identične matrike (enačba 1.17). Lahko jo kar natipkamo, kar smo prikazali z zadnjo matriko v algoritmu 1.4. Kasneje bomo prikazali enostavnejši način za določitev te matrike.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \ldots (1.17)$$

V nadaljevanju smo napisali algoritem 1.4, s katerimi bomo v proceduri IML določili matrike.

Ko kodo iz algoritma 1.4 poženemo, v oknu Output dobimo izpis matrik, kot prikazujemo v nadaljevanju v levem stolpcu, v desnem pa je prikazana koda, s katero smo (ali bi lahko) matriko vnesli. V SAS-u obstaja velikokrat več možnosti za zapis matrike ali posamezne funkcije. Običajno izberemo tisto, ki se jo lažje zapomnimo ali pa je bolj pregledna.

A 3 rows 3 cols (numeric) 1 2 3 3 4 0 5 6 2	A={1 2 3 , 3 4 0 , 5 6 2 };
B 3 rows 3 cols (numeric) 3 4 7 3 4 0 -2 0 8	B={3 4 7, 3 4 0 ,-2 0 8 };
C 3 rows 2 cols (numeric) 3 4 12 6 4 7	C={3 4, 12 6, 4 7};
D 3 rows 3 cols (numeric) 4 0 0 0 6 0 0 0 -5	D={4 0 0, 0 6 0, 0 0 -5};
I 4 rows 4 cols (numeric) 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1	I={1 0 0 0, 0 1 0 0, 0 0 1 0, 0 0 0 1};

Algoritem 1.5 Splošna sintaksa v proc IML

```
I6   = I(6);        /* ----------------- ali krajše - identična matrika reda 6x6 */
J5   = J(5);        /* ------------------------------- enotska matrika reda 5x5 */
J54  = J(5,4);      /* ------------------------ matrika reda 5x4 s samimi enkami */
J543 = J(5,4,3);    /* ---------------------- matrika reda 5x4 s samimi trojkami */
```

1.2.1 Posebne matrike

V SAS-u imamo možnost, da naredimo nekatere posebne matrike enostavneje in jih ni potrebno natipkati. Tako lahko kreiramo tudi identične ali enotske matrike.

$\mathbf{I}_n =$	**I(n)**	-	**identična matrika reda n x n**
$\mathbf{J}_n =$	**J(n)**	-	**enotska matrika n x n**
$\mathbf{J}_{r\times c} =$	**J(r,c)**	-	**enotska matrika reda r x c, kjer so same 1**
$\mathbf{JV}_{r\times c} =$	**J(r,c,v)**	-	**matrika reda r x c , v kateri so vse vrednosti enake v**

Na levi strani enačaja je ime nove matrike ali vektorja, ki določimo sami, na desni strani pa izbrana funkcija v iz procedure IML.

Določimo matrike še s kodo v proceduri IML (algoritem 1.5).

V levi stolpec smo vnesli izpise matrik iz okna Output, v desni stolpec pa smo vpisali kodo, s katero matriko določimo. Najprej smo določili identično matriko reda 6×6.

I6	6 rows	6 cols	(numeric)		
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

I6 = I(6);

Oblikovali smo tudi kvadratno enotsko matriko reda5×5, v kateri imajo vsi elementi vrednost 1.

j5	5 rows	5 cols	(numeric)	
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

J5 = J(5);

Oblikovali smo tudi pravokotno enotsko matriko s petimi vrsticami in štirimi stolpci, v kateri imajo vsi elementi vrednost 1.

J54	5 rows	4 cols	(numeric)	
1	1	1	1	
1	1	1	1	
1	1	1	1	
1	1	1	1	
1	1	1	1	

J54 = J(5,4);

Oblikovali smo tudi pravokotno matriko s petimi vrsticami in štirimi stolpci, v kateri imajo vsi elementi vrednost 3.

J543	5 rows	4 cols	(numeric)	
3	3	3	3	
3	3	3	3	
3	3	3	3	
3	3	3	3	
3	3	3	3	

J543 = J(5,4,3);

Algoritem 1.6 Nekaj možnosti z diagonalnimi matrikami v proc IML

```
D1   = diag(m);     /* -------------------------- diagonalna matrika iz vektorja */
D2   = diag(A);     /* ------------------------------------------ iz polne matrike */
R    = I(5)*2.5;  /* ustvarim diagonalno matriko z istimi vrednostmi na diagonali */
D3   = vecdiag(A);  /* -------------------- dobimo diagonalo iz matrike v vektorju */
DD   = block(A, C); /* ------- nastavi blok-diagomalno matriko z matrikama A in C */
```

1.2.2 Diagonalne matrike

Diagonalne matrike lahko vzpostavimo na več načinov.

V naslednjem prikazu smo diagonalnim elementom matrike D1 pripisali vrednosti iz vektorja **m**.

D1	4 rows	4 cols	(numeric)
5	0	0	0
0	2	0	0
0	0	4	0
0	0	0	6

D1 = diag(m);

Diagonalno matriko D2 smo naredili pravzaprav iz matrike **A**. Od matrike **A** so ostali le diagonalni elementi, nediagonalni elementi matrike D2 pa imajo vrednost 0.

D2	3 rows	3 cols	(numeric)
1	0	0	
0	4	0	
0	0	2	

D2 = diag(A);

Nato smo oblikovali še diagonalno matriko z vrednostjo 2.5 pri vseh diagonalnih elementih. Diagonalno matriko z enakimi vrednostmi (enačba 1.18) bomo uporabili pri matrikah varianc s homogenimi variancami.

$$\mathbf{R} = \mathbf{I}_5 \times 2,5 \qquad \ldots (1.18)$$

R	5 rows	5 cols	(numeric)	
2.5	0	0	0	0
0	2.5	0	0	0
0	0	2.5	0	0
0	0	0	2.5	0
0	0	0	0	2.5

R = I(5)*2.5;

V vektor D3 prepisali diagonalne elemente iz matrike **A**. Vektor D3 je stolpični.

D3	3 rows	1 col	(numeric)
		1	
		4	
		2	

D3 = vecdiag(A);

V biometriji se bomo srečali tudi z blok-diagonalnimi matrikami. Matrike imajo na diagonali manjše matrike, katerih elementi imajo vrednost večinoma različne od 0, vsi ostali elementi imajo vrednost 0.

DD	6 rows	5 cols	(numeric)	
1	2	3	0	0
3	4	0	0	0
5	6	2	0	0
0	0	0	3	4
0	0	0	12	6
0	0	0	4	7

DD = block(A, C);

Algoritem 1.7 Seštevanje matrik z matrikami, vektorji in skalarji

```
AB = A + B;   /* ------------------------------------------- seštevanje matrik*/
BA = B + A;
A1 = A + 2.3;  /* ---------------------------------- seštevanje matrik in skalarja */
A2 = A + B[1, ]; /* -------------------- seštevanje matrik in vrstičnega vektorja */
A3 = A + B[ ,1]; /* -------------------- seštevanje matrik in stolpičnega vektorja */
```

1.2.3 Osnovne matrične operacije

Med osnovne računske operacije v prvi vrsti štejemo trasponiranje, seštevanje, odštevanje in množenje. Omenili bomo tudi potenciranje, čeprav se praktično ne bomo srečali z njim.V statistiki poleg klasičnega množenja matrik uporabljamo še množenje parov istoležnih elementov in Kronecker produkt. Deljenja matrik ne poznamo, znak za deljenje (/) pa pride prav pri operacijah matrik s skalarji.

Operatorji v modulu SAS/IML:

+	seštevanje	#	množenje elementov	||	horizontalno spenjanje
-	odštevanje	'	transponiranje	//	vertikalno spenjanje
*	množenje	**	potenciranje	@	Kronecker produkt

Seštevanje in odštevanje matrik je računska operacija, kjer seštejemo ali odštejemo istoležne elemente matrik. Vse udeležene matrike morajo biti istega reda, in matrika z rezultati je istega reda. Vrstni red matrik lahko zamenjamo in je rezultat isti.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 0 \\ -2 & 0 & 8 \end{bmatrix} = \begin{bmatrix} 4 & 6 & 10 \\ 6 & 8 & 0 \\ 3 & 6 & 10 \end{bmatrix}$$

AB	3 rows	3 cols	(numeric)
	4	6	10
	6	8	0
	3	6	10

AB = A + B;

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 0 \\ -2 & 0 & 8 \end{bmatrix}$$

BA	3 rows	3 cols	(numeric)
	4	6	10
	6	8	0
	3	6	10

BA = B + A;

$$\begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 0 \\ -2 & 0 & 8 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix}$$

V SAS-u lahko matriki prištejemo ali odštejemo skalar, kar nam pride prav. To pa ni običajen zapis v matrični algebri. Tako je bolje zapisati enačbo kot vsoto matrike z zmnožkom skalarja z matriko samih enic.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + J(3)\times 10 = \begin{bmatrix} 11 & 12 & 13 \\ 13 & 14 & 10 \\ 15 & 16 & 12 \end{bmatrix}$$

A1	3 rows	3 cols	(numeric)
	3.3	4.3	5.3
	5.3	6.3	2.3
	7.3	8.3	4.3

A1 = A + 2.3;

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + 2.3$$

SAS v proceduri IML pa omogoča krajši zapis (prva vrstica), da se izognemo množenju z enicami. Za matriko enic porabimo dodaten računalniški spomin, za množenje pa dodatni računalniški čas. Tu imamo majhne primere, zato varčevanje ni nujno. Ko pa rešujemo že običajne probleme v živinoreji, pa moramo paziti tako na računalniški spomin kot čas.

```
A10 = A + 10;
A10 = A + J(nrow(A),ncol(A))*10;
```

V SAS-u lahko matrikam prištejemo tudi vektor. Če je vektor stolpičen, potem bo vektor prištet vsakemu stolpcu matrike (enačba 1.19). Vzemimo, da prištejemo matriki **A** prvi stolpec iz matrike **C**. Kodo bi secer lahko napisali le v eni vrstici, a je morda na ta način bolj razvidno, da prištevamo vektor.

```
vecC = C[1:nrow(C),1]
Ast  = A + vecC;
```

Tudi te operacije v matrični algebri ni. Seštevanje matrike in stolpičnega vektorja bi jih lahko zapisali s spetimi vektorji v enačbi 1.23 ali s Kronecker produktom v enačbi 1.20.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 12 \\ 4 \end{bmatrix} \| \begin{bmatrix} 3 \\ 12 \\ 4 \end{bmatrix} \| \begin{bmatrix} 3 \\ 12 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 15 & 16 & 12 \\ 9 & 10 & 6 \end{bmatrix} \quad \text{... (1.19)}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + J(1,3,1) \times \begin{bmatrix} 3 \\ 12 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 15 & 16 & 12 \\ 9 & 10 & 6 \end{bmatrix} \quad \text{... (1.20)}$$

Kadar pa je vektor vrstičen, bo prištet vsaki vrstici (enačba 1.19). Vzemimo, da prištejemo matriki **A** prvi stolpec iz matrike **B**.

```
vecB = B[1, 1:ncol(B)]
Ast  = A + vecB;
```

A2	3 rows	3 cols	(numeric)
	4	6	10
	6	8	7
	8	10	9

A2 = A + B[1,];

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 4 & 7 \end{bmatrix}$$

Seštevanje matrike in vrstičnega vektorja bi jih lahko zapisali s spetimi vektorji v enačbi 1.19 ali s Kronecker produktom v enačbi 1.20.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 4 & 7 \end{bmatrix} // \begin{bmatrix} 3 & 4 & 7 \end{bmatrix} // / \begin{bmatrix} 3 & 4 & 7 \end{bmatrix} = \begin{bmatrix} 4 & 6 & 10 \\ 6 & 8 & 7 \\ 8 & 10 & 9 \end{bmatrix} \quad \text{... (1.21)}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + J(3,1,1) \times \begin{bmatrix} 3 \\ 12 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 15 & 16 & 12 \\ 9 & 10 & 6 \end{bmatrix} \quad \text{... (1.22)}$$

A3	3 rows	3 cols	(numeric)
	4	5	6
	6	7	3
	3	4	0

A3 = A + B[,1];

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \\ -2 \end{bmatrix}$$

Algoritem 1.8 Množenje matrik z matrikami, vektorji in skalarji

```
F  = A * C; /* ------------------------------------------------------ množenje */
F2 = C * A;   /* ----------- množenje - ali je možno (preverite izpis v oknu Log) */
F1 = A * 2.3; /* ---------------------------------- množenje matrike s skalarjem */
F3 = A # B;   /* ------------------------- množenje parov isto ležečih elementov */
F4 = A # 2.3;   /* ------------------------ množenje parov isto ležečih elementov */
F5 = A # B[1, ];   /* --------------------- množenje parov isto ležečih elementov */
F6 = A # B[ ,1];   /* --------------------- množenje parov isto ležečih elementov */
F7 = A # C[ ,1] || A # C[ ,2]; /* ------------- spenjanje matrik dveh # produktov */
```

Množenje matrik opravimo tako, da k-ti element iz i-te vrstice prve matrike pomnožimo s k-tim elementom v j-ti vrstici druge matrike. Zmnožke parov seštejemo in dobimo element v i-ti vrstici in j-tem stolpcu matrike z rezultati. Ker množimo istoležne pare, mora imeti prva matrika toliko stolpcev kot druga vrstic. Rezultat ima toliko vrstic kot prva in toliko stolpcev kot druga.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \times \begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} = \begin{bmatrix} 39 & 37 \\ 57 & 36 \\ 95 & 70 \end{bmatrix}$$

F 3 rows 2 cols (numeric)		
39	37	
57	36	F = A * C;
95	70	$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \times \begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix}$

Če vrstni red matrik zamenjamo in ju lahko množimo (npr. dve kvadratni obliki), dobimo povsem drugačen rezultat. Pri množenju matrik je vrstni red pomemben. Pri pravokotnih matrikah pa se nam praviloma zgodi, da matrik v novem vrstnem redu ne moremo množiti.

ne moremo množiti	F2 = C * A;	$\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix}$

Množenje istoležnih parov je množenje, kjer element (a_{ij}) iz prve matrike pomnožimo z istoležnim elementom (b_{ij}) iz druge matrike. Znak za operacijo je #.

F3 3 rows 3 cols (numeric)				
3	8	21		
9	16	0	F3 = A # B;	$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 3 & 4 & 7 \\ 3 & 4 & 0 \\ -2 & 0 & 8 \end{bmatrix}$
-10	0	16		

Množimo lahko tudi matriko z vektorjem ali skalarjem. V matrični algebri nimamo predvidenega tega zapisa. Skalar v matrični algebri najprej pomnožimo z enotsko matriko, vektor pa z vektorjem tako, da dobimo matriko istega reda, kot je podana matrika.

- Ko množimo matriko s skalarjem, pomnožimo vse elemente matrike s skalarjem (enačba 1.23 oz. 1.24).

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \#2.3 = \begin{bmatrix} 2.3 & 4.6 & 6.9 \\ 6.9 & 9.2 & 0.0 \\ 11.5 & 13.8 & 4.6 \end{bmatrix} \quad \text{... (1.23)}$$

$$\mathbf{A}\#J(3,3)\times c = \begin{bmatrix} a_{11}+c & a_{12}+c & a_{13}+c \\ a_{21}+c & a_{22}+c & a_{23}+c \\ a_{31}+c & a_{32}+c & a_{33}+c \end{bmatrix} \quad \text{... (1.24)}$$

F4	3 rows	3 cols	(numeric)
	2.3	4.6	6.9
	6.9	9.2	0
	11.5	13.8	4.6

F4 = A # 2.3;

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \#2.3$$

- Ko množimo matriko z vrstičnim vektorjem, pomnožimo elemente vsake vrstice matrike z istoležnimi elementi vektorja (enačba 1.25 oz. 1.26). Vrstični vektor mora imeti toliko stolpcev kot matrika.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 3 & 4 & 7 \end{bmatrix} = \begin{bmatrix} 3 & 8 & 21 \\ 9 & 16 & 0 \\ 15 & 24 & 14 \end{bmatrix} \quad \text{... (1.25)}$$

$$\mathbf{A}\#J(3,1,1)\times \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} a_{11}+v_1 & a_{12}+v_2 & a_{13}+v_3 \\ a_{21}+v_1 & a_{22}+v_2 & a_{23}+v_3 \\ a_{31}+v_1 & a_{32}+v_2 & a_{33}+v_3 \end{bmatrix} \quad \text{... (1.26)}$$

F5	3 rows	3 cols	(numeric)
	3	8	21
	9	16	0
	15	24	14

F5 = A # B[1,];

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 3 & 4 & 7 \end{bmatrix}$$

- Ko množimo matriko s stolpičnim vektorjem, pomnožimo elemente vsakega stolpca matrike z istoležnimi elementi vektorja (enačba 1.27). Stolpični vektor mora imeti toliko vrstic kot matrika.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 3 \\ 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 9 & 12 & 0 \\ -10 & -12 & -4 \end{bmatrix} \quad \text{... (1.27)}$$

F6	3 rows	3 cols	(numeric)
	3	6	9
	9	12	0
	-10	-12	-4

F6 = A # B[,1];

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 3 \\ 3 \\ -2 \end{bmatrix}$$

Operacije lahko uporabimo pri kreiranju delnih matrik za kvantitativne vplive.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 3 \\ 12 \\ 4 \end{bmatrix} \| \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \# \begin{bmatrix} 4 \\ 6 \\ 7 \end{bmatrix}$$

F7	3 rows	6 cols	(numeric)		
3	6	9	4	8	12
36	48	0	18	24	0
20	24	8	35	42	14

F7 = A # C[,1] || A # C[,2];

Kronecker produkt izračuna novo matriko tako, da vsak element (a_{ij}) iz prve matrike (**A**) pomnoži z celotno drugo matriko (**C**). Tako vsak element prve matrike zamenjamo s produktom druge matrike s skalarjem in se poveča za število vrstic in število stolpcev iz druge matrike.

Prvi dve matriki prikazujeta Kronecker produkta katerihkoli matrik. Vzeli smo kvadratno matriko **A** in Pravokotno matriko **C**. Z njima lahko naredimo dva Kronecker produkta tako, da smo vrstni red matrik zamenjali. Najprej

Algoritem 1.9 Kronecker produkt

```
C1 = A @ C;   /* ------------------------------------- Kronecker produkt */
C2 = C @ A;
/* matrika varianc in kovarianc za skupno okolje v gnezdu za dve lastnosti: */
Gg = I(4) @ {324   21.06, 21.06   6.76};
Gg1= {324   21.06, 21.06   6.76} @ I(4);
```

smo izvedli Kronecker produkt $\mathbf{A} \otimes \mathbf{C}$ (enačba1.28) in nato še Kronecker produkt $\mathbf{C} \otimes \mathbf{A}$(enačba1.29). Rezultata nista enaka: vrstice in stolpci so prerazporejeni (permutirani). Pomembno si je zapomniti, da je tudi pri Kronecker produktu zelo pomemben vrstni red matrik.

$$\mathbf{A} \otimes \mathbf{C} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \otimes \begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} = \begin{bmatrix} 1\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} & 2\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} & 3\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} \\ 3\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} & 4\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} & 0\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} \\ 5\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} & 6\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} & 2\times\begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} \end{bmatrix} \quad \ldots (1.28)$$

C1 9 rows 6 cols (numeric)

3	4	6	8	9	12
12	6	24	12	36	18
4	7	8	14	12	21
9	12	12	16	0	0
36	18	48	24	0	0
12	21	16	28	0	0
15	20	18	24	6	8
60	30	72	36	24	12
20	35	24	42	8	14

C1 = A @ C;

$$\mathbf{C} \otimes \mathbf{A} = \begin{bmatrix} 3 & 4 \\ 12 & 6 \\ 4 & 7 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} = \begin{bmatrix} 3\times\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} & 4\times\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \\ 12\times\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} & 6\times\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \\ 4\times\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} & 7\times\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix} \end{bmatrix} \quad \ldots (1.29)$$

C2 9 rows 6 cols (numeric)

3	6	9	4	8	12
9	12	0	12	16	0
15	18	6	20	24	8
12	24	36	6	12	18
36	48	0	18	24	0
60	72	24	30	36	12
4	8	12	7	14	21
12	16	0	21	28	0
20	24	8	35	42	14

C2 = C @ A;

Kronecker produkt bomo srečali pri strukturi varianc in kovarianc pri večlastnostnih modelih. Za začetek poskusimo z matriko varianc in kovarianc za skupno okolje v gnezdu ($\mathbf{G}_g$) za dve lastnosti. Ta vpliv sodi med enostavne (trivialne) naključne vplive, ker so nivoji neodvisni.

V prvem primeru (enačba 1.30) imamo parametre za skupno okolje za lastnosti urejene znotraj gnezd. Tako imamo zaporedoma napovedi za skupno okolje v gnezdu za obe lastnosti, nato sledijo posamezna gnezda z napovedma za skupno okolje za obe lastnosti.

$$\mathbf{G}_g = \mathbf{I}_4 \otimes \mathbf{G}_{0g} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \otimes \begin{bmatrix} 324 & 21.6 \\ 21.6 & 6.76 \end{bmatrix} = \\ = \begin{bmatrix} 324 & 21.6 & & & & & & \\ 21.6 & 6.76 & & & & & & \\ & & 324 & 21.6 & & & & \\ & & 21.6 & 6.76 & & & & \\ & & & & 324 & 21.6 & & \\ & & & & 21.6 & 6.76 & & \\ & & & & & & 324 & 21.6 \\ & & & & & & 21.6 & 6.76 \end{bmatrix} \quad \ldots (1.30)$$

Gg = I(4)@ {324 21.06, 21.06 6.76};

Gg 8 rows 8 cols (numeric)

324	21.06	0	0	0	0	0	0
21.06	6.76	0	0	0	0	0	0
0	0	324	21.06	0	0	0	0
0	0	21.06	6.76	0	0	0	0
0	0	0	0	324	21.06	0	0
0	0	0	0	21.06	6.76	0	0
0	0	0	0	0	0	324	21.06
0	0	0	0	0	0	21.06	6.76

V drugem primeru (enačba 1.31) imamo gnezda v vektorju parametrov urejena znotraj napovedi za lastnosti: najprej imamo zbrano vse napovedi za prvo lastnost za vsa gnezda, v drugem delu so napovedi za drugo lastnost prav tako za vsa gnezda.

$$\mathbf{G}_{g1} = \mathbf{G}_{0g} \otimes \mathbf{I}_4 = \begin{bmatrix} 324 & 21.6 \\ 21.6 & 6.76 \end{bmatrix} \otimes \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = \\ = \begin{bmatrix} 324 & & & & 21.6 & & & \\ & 324 & & & & 21.6 & & \\ & & 324 & & & & 21.6 & \\ & & & 324 & & & & 21.6 \\ 21.6 & & & & 6.76 & & & \\ & 21.6 & & & & 6.76 & & \\ & & 21.6 & & & & 6.76 & \\ & & & 21.6 & & & & 6.76 \end{bmatrix} \quad \ldots (1.31)$$

Gg1= {324 21.06, 21.06 6.76} @ I(4);

324	0	0	0	21.06	0	0	0
0	324	0	0	0	21.06	0	0
0	0	324	0	0	0	21.06	0
0	0	0	324	0	0	0	21.06
21.06	0	0	0	6.76	0	0	0
0	21.06	0	0	0	6.76	0	0
0	0	21.06	0	0	0	6.76	0
0	0	0	21.06	0	0	0	6.76

Dobili smo drugačno razvrstitev vrstic in stolpcev, zato sta matriki različni.

Algoritem 1.10 Računske operacije z matrikami

```
nv     = nrow(A);   /* ------------------------- število vrstic v matriki A */
ns     = ncol(A);   /* ------------------------ število stolpcev v matriki A */
detA   = det(A);    /* ------------------------------ determinanta matrike A */
vsotaA = sum(A);    /* ------------------------------------ vsota elementov */
povp   = mean(A);   /* ---------------------------------- povprečje elementov */
vkA    = ssq(A);    /* --------------------------- vsota kvadratov elementov */
minA   = min(A);    /* ------------------------ minimalna vrednost v matriki */
maxA   = max(A);    /* ------------------------ maksimalna vrednost v matriki */
trA    = trace(A);  /* ---------- sled matrike - vsota diagonalnih elementov */
```

Algoritem 1.11 Inverzne matrike in spenjanje matrik

```
tA  = T(A);      /* ---------------------------------- matrika A transponirano */
tA2 = A`;        /* ------------------------------------------ še na drugi način */
maxdec=6;
iA  = inv(A);    /* ---------------------------------------  inverza matrike A */
giA = ginv(A);   /* ------------------------------- splošna inverza matrike A */
iC  = inv(C);    /* ----------- inverzna matrike C (preverite izpis v oknu Log)*/
giC = ginv(C);   /* ------------------------------- splošna inverza matrike C */
AvB = A//B;      /* ---------------------------- vertikalno spenjanje matrik */
AhB = A||B;      /* ---------------------------- horizontalno spenjanje matrik */
```

1.2.4 Funkcije na matrikah v proceduri IML

nv 1 row 1 col (numeric) 3	nv = nrow(A);
ns 1 row 1 col (numeric) 3	ns = ncol(A);
detA 1 row 1 col (numeric) -10	detA = det(A);
vsotaA 1 row 1 col (numeric) 26	vsotaA = sum(A);
vkA 1 row 1 col (numeric) 104	vkA = ssq(A);
minA 1 row 1 col (numeric) 0	minA = min(A);
maxA 1 row 1 col (numeric) 6	maxA = max(A);
trA 1 row 1 col (numeric) 7	trA = trace(A);

1.2.5 Transponirane matrike, inverzne matrike in spenjanje matrik v proc IML

Transponirana matrika je matrika, v kateri vrstice iz prve matrike postanejo stolpci v transponirani matriki, stolpci pa tako postanejo vrstice v transponirani matriki (enačbe 1.32). Možnih je več zapisov za transponiranje matrike.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \\ 5 & 6 & 2 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 3 & 0 & 2 \end{bmatrix} \quad \text{... (1.32)}$$

tA 3 rows 3 cols (numeric)	
1 3 5 2 4 6 3 0 2	tA=T(A);

tA2 3 rows 3 cols (numeric)	
1 3 5 2 4 6 3 0 2	tA2=A‘;

Inverzna matrika je matrika, za katero velja, da je $\mathbf{A}\times\mathbf{A}^{-1} = \mathbf{I}$. Inverzno matriko lahko izračunamo

iA 3 rows 3 cols (numeric)	
-0.8 -1.4 1.2 0.6 1.3 -0.9 0.2 -0.4 0.2	iA=inv(A);

Posplošena inverzna matrika je matrika, za katero velja, da je $\mathbf{A}\times\mathbf{A}^{-}\times\mathbf{A} = \mathbf{A}$.

giA 3 rows 3 cols (numeric)	
-0.8 -1.4 1.2 0.6 1.3 -0.9 0.2 -0.4 0.2	giA=ginv(A);

ne obstaja	iC=inv(C);

giC 2 rows 3 cols (numeric)	
-0.032044 0.119337 -0.083978 0.0751381 -0.072928 0.1624309	giC=ginv(C);

Cholesky razčlenitev kvadratno in simetrično matriko **G** razčleni na spodnjo (**L**) in zgornjo (**L´**) trikotno matriko, ki sta ena drugi transponirani, njun produkt pa matrika **G**.

$\mathbf{G} = \mathbf{L}\times\mathbf{L}´$

S Cholesky razčlenitvijo preverjamo pozitivno definitnost matrik varianc in kovarianc. Če matrika ni pozitivno definitna, ne more služiti kot matrika varianc in kovarianc.

	matU=root(**G**); matU=half(**G**);

Vertikalno spenjanje matrik omogoča zaporedno sestavljanje matrik ali vektorjev tako, da je rezultat matrika ali vektor, kjer so matrike, ki jih spajamo, ena pod drugo. Pri vertikalnem spenjanju je pomembno, da imajo matrike isto število stolpcev, lahko pa se razlikujejo v številu vrstic. Število stolpcev tako ostane enako, število vrstic pa je vsota števila stolvrsticpcev vseh spetih matrik.

AvB 6 rows 3 cols (numeric)	
1 2 3 3 4 0 5 6 2 3 4 7 3 4 0 -2 0 8	AvB=A//B;

Algoritem 1.12 Indeksi v matriki

```
abc1v3s = ABC[1,3];      /* element v 1. vrstici in 3. stolpci iz matrike ABC */
abc2v   = ABC[2,];       /* ---------------- celotna 2. vrstica iz matrike ABC */
abc4s   = ABC[,4];       /* ---------------- celoten 4. stolpec iz matrike ABC */
abc3    = ABC[3];        /* --- 3. element v ABC, štejemo po elementih vrstic */
abc4    = ABC[15:19];    /* ----------------- 15. do 19. element v matriki ABC */
abc5    = ABC[1,2:4];    /* ---------- 1. vrstica v matriki, 2. do 4. stolpec */
abc6    = ABC[{1 3}, {2 4}]; /* ---------- 1. in 3. vrstica, 2. in 4. stolpec */
```

Horizontalno spenjanje matrik omogoča zaporedno sestavljanje matrik ali vektorjev tako, da je rezultat matrika ali vektor, kjer so matrike, ki jih spajamo, ena poleg druge. Pri horizontalnem spenjanju je pomembno, da imajo matrike isto število vrstic, lahko pa se razlikujejo v številu stolpcev. Število vrstic tako ostane enako, število stolpcev pa je vsota števila stolpcev vseh spetih matrik.

AhB=A||B;

AhB	3 rows	6 cols	(numeric)		
1	2	3	3	4	7
3	4	0	3	4	0
5	6	2	-2	0	8

1.2.6 Oblikovanje delnih matrik, vektorjev in elementov

S pomočjo indeksov lahko izberemo enega ali več meritev, vrstico ali stolpec, podmatriko. V oglatih oklepajih najprej navedemo vrstico in nato stolpec, ki sta ločena z vejico (algoritem 1.12).

ABC = block(A, B, C);

ABC	9 rows	8 cols	(numeric)				
1	2	3	0	0	0	0	0
3	4	0	0	0	0	0	0
5	6	2	0	0	0	0	0
0	0	0	3	4	7	0	0
0	0	0	3	4	0	0	0
0	0	0	-2	0	8	0	0
0	0	0	0	0	0	3	4
0	0	0	0	0	0	12	6
0	0	0	0	0	0	4	7

abc1v3s=ABC[1,3];

abc1v3s	1 row	1 col	(numeric)
		3	

abc1v3s=ABC[1,3];

abc2v=ABC[2,];

abc2v	1 row	8 cols	(numeric)				
3	4	0	0	0	0	0	0

abc4s=ABC[,4];

abc4s	9 rows	1 col	(numeric)
		0	
		0	
		0	
		3	
		3	
		-2	
		0	
		0	
		0	

Algoritem 1.13 Funkciji DESIGN in DESIGNF

```
C   = {1,3,2,2,1,4,3,4};   /* stolpični vektor */
Xc  = design(c);
XcF = designf(c);
```

```
abc3      1 row       1 col     (numeric)

                        3
```

abc3=ABC[3];

```
abc4      5 rows      1 col     (numeric)

                        0
                        0
                        5
                        6
                        2
```

abc4=ABC[15:19];

```
abc5      1 row       3 cols    (numeric)

              2         3         0
```

abc5=ABC[1,2:4];

```
abc6      2 rows      2 cols    (numeric)

                    2         0
                    6         0
```

abc6=ABC[{1 3}, {2 4}];

1.2.7 Funkcije za kreiranje matrik dogodkov v SAS/IML

Funkcija DESIGN je namenjena kreiranju matrik dogodkov iz stolpičnih vektorjev (koda 4). Vrednosti v vektorju so lahko numeričnega ali znakovnega tipa. Vsaka enolična vrednost v vhodnem vektorju ima za posledico stolpec v matrika dogodkov. Vrednosti v matriki dogodkov pa so 0 in 1. Vrstni red stolpcev je urejen po vrstnem redu razvrščenih enoličnih vrednosti. Vrstice matrike dogodkov pa imajo vrstni red elementov vhodnega vektorja.

```
C       8 rows      1 col     (numeric)

                        1
                        3
                        2
                        2
                        1
                        4
                        3
                        4
```

C = {1,3,2,2,1,4,3,4};

Funkcija DESIGN iz vektorja naredi (delno) matriko dogodkov. Uporabljamo samo pri vplivih z nivoji oz. kvalitativnih vplivih. Vsaka vrednost v vektorju je privzeta kot razred pri izbranem vplivu in dobi svoj stolpec. V vektorju c so štiri vrednosti v 8 vrsticah, zato dobi matrika dogodkov štiri stolpce. V matriki pa se pojavljajo 1 v vrstici, kjer se vrednost pojavi. Vrednosti iz vektorja so tako privzete kot nivoji posameznega vpliva.

```
Xc      8 rows      4 cols    (numeric)

        1         0         0         0
        0         0         1         0
        0         1         0         0
        0         1         0         0
        1         0         0         0
        0         0         0         1
        0         0         1         0
        0         0         0         1
```

Xc = design(c);

Algoritem 1.14 Branje nizov podatkov in pretvorba v matrike

```
proc iml;
reset print;
use NizPodatkov;    /* ------------------------------------ ime niza podatkov v knjižnici WORK */
show datasets;      /* ------------------------------------------ prikaže nize, ki so na voljo */
show contents;      /* ------------------------------------------------- prikaže vsebino niza */
show names;         /* ------------------------------------------------prikaže opise podatkov */
read all var; /* ---------------------- prebere vse vrstice in stolpce v vektorje z istim imenom  3. */
read all var     _all_   into  matrika1;  /* ------------------- preberemo vse stolpce v matriko */
read all var     _num_   into  matrika2;  /* --------- preberemo vse numerične stolpce v matriko */
read all var     _char_  into  matrika3;  /* ---------- preberemo vse znakovne stolpce v matriko */
read all var     {stolpec1 stolpec2}   into  matrika4;  /*   preberemo naštete stolpce v matriko */
read all;           /* -------- prebere ves niz in stolpce poimenuje po stolpcih v nizu podatkov */

read all var {stolpec1} into vektor1;  /* ------------------------- preberemo stolpec1 v vektor1 */
read all var {stolpec2} into vektor2;  /* ------------------------- preberemo stolpec2 v vektor2 */
read all var {stolpec3} into vektor3;  /* ------------------------- preberemo stolpec3 v vektor3 */
read point(n1:n2) var {stolpec3} into vektor3; /*-- prebere samo vrstice od n1 do n2 v stolpcu 3 */
/* ------------------------------------------------ preberemo stolpec2 brez manjkajocih vrednosti */
read all var {stolpec2} where (stolpec2 ^= .) into vektor2;
read all var {stolpec2} where (rejec = "MK")  into vektor2; /* preberi samo vrednosti za rejca 2 */
```

Funkcija DesignF ustvari matriko dogodkov, ki je polnega ranga. Le-ta ima v primerjavi z matriko dogodkovpri funkciji Design en stolpec manj za vsako linearno odvisno kombinacijo. V primeru glavnega kvalitativnega vpliva je zadnji stolpec odštet od ostalih stolpcev.

```
XcF      8 rows      3 cols     (numeric)

            1          0          0
            0          0          1
            0          1          0
            0          1          0
            1          0          0
           -1         -1         -1
            0          0          1
           -1         -1         -1
```

XcF = designf(c);

1.2.8 Uporaba podatkov iz podatkovnih zbirk v SAS-u

Znotraj modula SAS/IML lahko dostopamo tudi do nizov podatkov iz podatkovnih zbirk. Ker modul IML pri izračunih uporablja vektorje in matrike, moramo izbrati niz podatkov in podatke iz izbranega niza prebrati v vektorje in matrike. V algoritmu 1.13 prikazujemo nekaj možnosti, kako preberemo podatke v vektorje ali matrike. Obrazložitve sledijo kasneje na primeru ovce.

Z ukazom "show datasets" dobimo spisek nizov podatkov, ki jih imamo v SAS-ovih podatkovnih zbirkah. Trenutno imamo v podatkovnih zbirkah samo niz podatkov z nazivom "ovce".

```
LIBNAME  MEMNAME                          OPEN MODE   STATUS
-------- -------------------------------- ---------   --------
WORK     OVCE                             Input       Current Input
```

show datasets;

Z ukazom "show contents" lahko dobimo seznam stolpcev - spremenljivk - v posameznem nizu podatkov. Imena spremenljivk pogosto pozabimo, zato je ukaz kar uporaben lahko večkrat. Vsekakor lahko pogledamo tudi niz podatkov v zbirki z

```
DATASET : WORK.OVCE.DATA

 VARIABLE                        TYPE  SIZE
 ------------------------------  ----  ----
 pasma                           char     2
 spol                            num      8
 rejec                           char     2
 RojMasa                         num      8
 StarOdst                        num      8
 OdstMasa                        num      8
 jagnje                          num      8
 mati                            num      8
 oce                             num      8
```

show contents;