Többdimenziós statisztika

Prőhle Tamás - Zempléni András

2013.06.28

Tartalomjegyzék

1. fejezet

Előszó

Ez a jegyzet az ELTE TTK Matematikai Intézet Valószínűségelméleti és Statisztika Tanszéken tartott többdimenziós statisztika tárgyak tanulásához kíván segítséget nyújtani, elsősorban gyakorlati szempontból. A jegyzet felhasználja a valószínűségszámítás és a matematikai statisztika alapfogalmait, ezért értelemszerűen ezek után a kurzusok után ajánlott a tanulmányozása. Azonban nem célunk az elmélet teljeskörű feldolgozása, csak a módszerek, alkalmazások megértéséhez feltétlenül szükséges mélységben tárgyaljuk ezeket. Néhány kevéssé ismert, érdekes modellnél azonban kivételt teszünk és felvillantjuk a bizonyítások alapgondolatát is.

A jegyzetben a fogalmak, megközelítésmódok rövid ismertetése után példákon keresztül mutatjuk be a módszereket, ezek lényege reményeink szerint a társtudományok művelői (mérnökök, pszichometrikusok, természettudósok) számára is érthető lesz. Ezeket a példákat a nyílt forráskódú **R** program [28] és számos kiegészítő csomagja segítségével oldjuk meg, sok esetben a használt programkódokat is megadva. Így az olvasó képes lesz arra, hogy saját praxisában felmerülő hasonló jellegű kérdéseket is sikerrel válaszolja meg. Az **R** progamon belül is általában több csomag közül választhatunk egy adott feladat megoldásánál, ezek összehasonlítására is kitérünk. A gyorsan fejlődő témáknál az aktuális, releváns szakirodalom felkutatásának hatékony módszere a programok hivatkozáslistájának átnézése. Mi itt most nem vállalkoztunk ezek kigyűjtésére, a legfontosabb "klasszikus" könyvek mellett az R csomagjait és adatelemzéseket végző internetes oktatási segédanyagokat tartalmazza a hivatkozásjegyzék.

Az első fejezet a kísérlettervezés alapfogalmait mutatja be. Elsősorban a leggyakrabban használt faktoriális terveket és a gyakorlati megvalósítás során felmerülő kérdéseket veszi sorra. Jópár példán keresztül kerülnek bevezetésre olyan fogalmak, mint a tervek felbontása. A kapott eredmények kiértékelési módszereit is bemutatjuk, így elsősorban a szóráselemzést. Ugyanakkor a terjedelmi korlátok miatt szükségszerűen kimaradnak fontos részek, ezeket például Kemény és Deák tankönyvéből [22] ismerheti meg az érdeklődő olvasó.

A 3 fejezet a nemlineáris regresszióval foglalkozik, részletesen bemutatva az **R** ren-

geteg beépített regressziós függvényéét. Külön részben szerepel a monoton regresszió módszere, itt néhány egyszerű bizonyítás is található. Végül az önmagában is kiemelkedő fontosságú általánosított lineáris model következik. A [31] könyv hasznos további információkat nyújt.

A következő fejezet a klasszikusnak számító főkomponens- és faktoranalízis modelljeivel foglalkozik. Ezek a dimenziócsökkentő eljárások arra is alkalmasak, hogy az adatok rejtett kapcsolatait feltárják, ezért alkalmazási lehetségük igen széles körű. A bemutatott példák is megfelelnek ennek a sokoldalúságnak: a pszichometriától a pénzügyi alkalmazásokig láthatunk adatelemzést. Az elmélet itt nem tárgyalt részei például a [35] könyvben olvashatóak. Jó összefoglaló az angol nyelvű [4] könyv is.

Az 5 fejezet a többdimenziós regresszió modern eljárásaival foglalkozik. Ezen belül külön részben szerepel a parciális regresszió, a path analízis és a SEM (struktúrális egyenlet-modell) megközelítés. Mivel igen új témáról van szó, a további információk itt legcélszerűbben az **R** kapcsolódó dokumentációjából szerezhetőek be.

Az utolsó fejezetben a többdimenziós skálázást és a korrespondencia analízist ismertetjük. Itt is nagy szerepet kap a különböző **R** csomagok és az általuk megoldott minta-adatelemzések bemutatása. Az elmélet további fejezetei itt is megtalálhatóak a [35] vagy a [4] tankönyben.

A jegyzethez kapcsolódóan animációkat is készítettünk. Ezek a szövegben megadott honlapokról érhetők el, és mindenkinek nagyon ajánljuk a tanulmányozásukat! Segítségükkel az éppen ismertetett módszerek gyakorlati tulajdonságai, a bemutatott eljárások különböző adatok, illetve paraméterezés melletti eredményei figyelhetők meg.

Végül néhány apró megjegyzés. Mivel az **R** tipikusan az angolszász jelölésrendszernek megfelelően tizedespontot használ, ezért mi is ezt alkalmazzuk a szövegben is, hogy fenntartsuk az összhangot a program outputjaival. A programkódok legfontosabb részeit is megadjuk a jegyzetben, ezzel is segítve az olvasó számára az önálló munkát. Ezek könnyen felismerhetőek a szövegkörnyezettől eltérő betűtípus segítségével, időnként megjegyzések is segítik a megértésüket. A folyó szövegen belül '...' jelöli az **R** utasításokat, változókat, attribútumokat.

A "Kísérlettervezés" (2) és a "Dimenziócsökkentési eljárások" (4) fejezet, valamint a szerkesztés és az animációk Zempléni András, a "Nem-lineáris regresszió" (3), a "Többdimenziós regresszió" (5) és a "Skálázás" (6) fejezet Prőhle Tamás munkája. Köszönjük a lektornak, Gáll Józsefnek (Debreceni Egyetem) a hasznos észrevételeket.

2. fejezet

Kísérlettervezés

2.1. Bevezető

Először magának a kísérletnek a fogalmát kell tisztáznunk. A statisztikában többnyire nem irányított kísérletek eredményeit elemezzük, hanem a véletlenszerű megfigyelések adataival dolgozunk. A lényeges különbség a két adattípus között, hogy míg a megfigyeléseknél az egyes változók értékeit nem mi kontrolláljuk (pl. időjárás, pénzügyi folyamatok), a kísérleteket mi magunk tervezzük, előre meghatározva a beállítható paraméterek értékeit.

Mire is használhatjuk ezeket a kísérleteket? Elsősorban az iparban, de máshol is lényeges lehet annak vizsgálata, hogy egy termék adott tulajdonságát milyen gyártási technológiával lehet optimalizálni (például: mikor lesz a gyártott kötél szakítószilárdsága a legnagyobb). Ehhez hasonló kérdésekre precíz választ a kísérlettervezés eszközeinek alkalmazásával kaphatunk. A kísérlet eredményét befolyásoló tényezőket faktoroknak nevezzük. A kísérletek során ezek beállítását (itt most szinteknek nevezzük) változtatjuk.

A fő problémát az jelenti, hogy a kísérletek tipikusan drágák és időrablók (gondoljunk csak bele: a legkülönbözőbb faktorokat kell minden egyes alkalommal adott szintre beállítani), ezért nem mindig lehet az összes faktor-kombinációra elvégezni a kísérleteket.

Látni fogjuk, hogy ezekben az esetekben úgynevezett részfaktoriális tervek jelenthetik a megoldást. Ezek sajátos tulajdonsága az alias struktúra azaz az, hogy bizonyos hatások nem becsülhetők külön, hanem csupán más – ideális esetben jóval magasabb rendű – kölcsönhatással együtt. A mérnökök feladata eldönteni – még a tervezés fázisában –, hogy ilyen esetben egyértelműsíthető-e a ténylegesen ható faktor(kombináció). Ha nem, akkor további kísérletek végzésére, jobb felbontású tervek készítésére van szükség.

Ugyanakkor arra minden esetben törekednünk kell, hogy a kísérletek fedjék le a gyakorlatban felmerülő lehetőségeket (ne csak egy részét vizsgáljuk, még ha az kényelmesebbnek is tűnik), mert csak így várható, hogy valóban használható eredményeket kapjunk.

A kísérlet eredményét befolyásoló tényezőket faktoroknak nevezzük. Az értéküket a

kísérlet során szisztematikusan változtatjuk, ezek a beállítások a faktorok szintjei. A várhatóan legfontosabb faktorokat igyekszünk előzetesen meghatározni. A többi faktort pedig zajfaktornak tekintjük és a kísérlet megtervezése során arra ügyelünk, hogy hatásuk minimális legyen. Ez történhet véletlenítéssel vagy blokkosítással. A későbbiekben visszatérünk ezen módszerek részletes ismertetésére. Lényeges, hogy foglalkozzunk ezekkel a kérdésekkel, mert a gyakorlatban mindig vannak olyan hatások, amket nem tudunk vagy nem lehetséges beállítani (külső körülmények), de hatásuk nem biztos, hogy elhanyagolható.

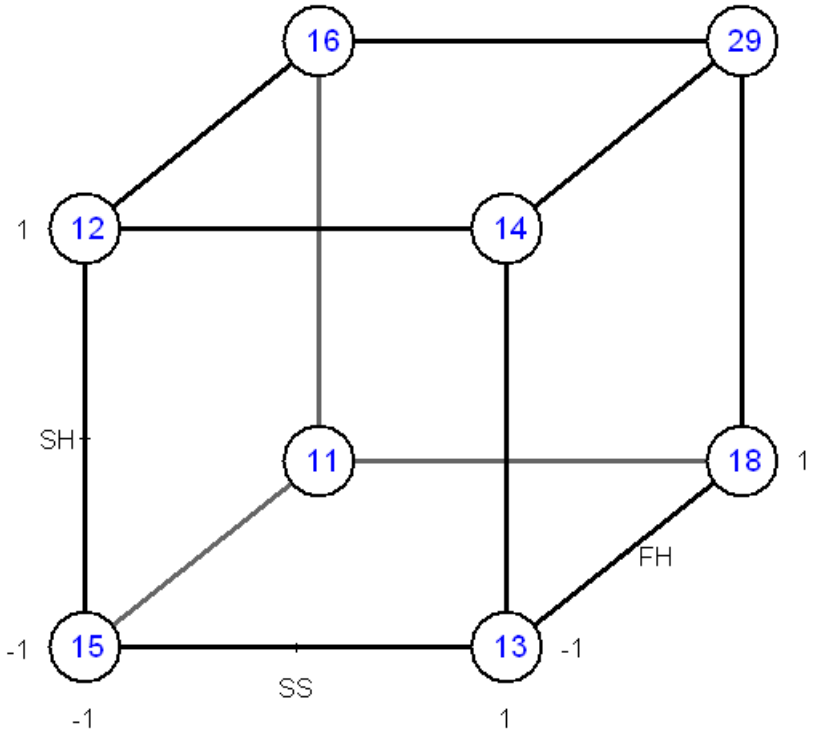
A leggyakrabban használt faktoriális tervek részletes ismertetésére a 2.2 fejezetben térünk ki. De először érdemes megjegyezni, hogy miért van egyáltalán szükség ilyen összetett matematikai apparátusra az optimum keresésénél. Logikusnak tűnhet az a módszer is, ami szerint sorra vesszük a faktorokat és egyesével mindegyikre megkeressük az optimális beállítást. A gond ezzel az egyesével történő optimalizálással (one factor at a time, OFAT), hogy nem tudja figyelembe venni a faktorok között igen gyakran megfigyelhető kölcsönhatást. Ennek eredményeként az így kapott megoldás egyáltalán nem biztos, hogy optimális lesz. Tekintsük a 2.1 ábrán látható eredményeket, amelyek 3 faktor hatását mutatják. Ha a bal alsó sarokból indulunk, akkor bármely faktort is módosítjuk, az eredmény rosszabb lesz a kiindulópontbelinél. De a faktoriális kísérleti terv alapján meg tudjuk találni a jobb felső sarokban a meglepően nagy célértéket.

Az eredmények kiértékelése a szóráselemzés (2.2.2 alfejezet) segítségével történhet, de jónéhány, a kísérlettervezésre jellemző speciális technika is alkalmazható, ezeket is bemutatjuk. Lényeges, hogy a tervünk eredményeként az eredmények megbízhatóságáról is képet kapjunk, például tudjunk konfidencia intervallumokat szerkeszteni, szignifikanciaszinteket becsülni.

A fejezet anyaga jelentősen épít Oehlert 2010-es könyvére [27], amely szabadon letölthető és nagy segítséget jelenthet azoknak, akik a most bemutatásra kerülő ízelítőn túlmenően is érdeklődnek a téma iránt.

2.2. Teljes faktoriális tervek

Azokat a terveket nevezzük teljes faktoriális tervnek, amelyeknél az összes vizsgálandó faktor minden szint-kombinációján elvégezzük a kísérleteket. A leggyakrabban két szinten végezzük a méréseket. Ennek egyrészt gyakorlati okai vannak: például az n faktor 3 szintjén szükséges 3^n kisérlet már elég kis n értékekre is nagyságrendekkel több, mint a két szinthez tartozó 2^n. Másrészt ugyan igaz, hogy ilymódon csak lineáris hatásokat tudunk detektálni (2 pontra csak egyenest tudunk illeszteni, magasabb hatványhoz tartozó polinomot nem), de sokszor elegendő a lineáris hatás kimutatása például a változtatás irányának meghatározásához – erre pedig már a csak 2 szinten elvégzett kísérlet is alkalmas. Ráadásul a matematikai módszerek is sokkal egyszerűbbek erre az esetre, ezért a módszer bemutatására különösen kézenfekvő ezt választani.



2.1. ábra. Egy elképzelt kísérlet eredményei

Ennél a legegyszerűbb, kétszintű tervnél a szinteket célszerűen +1 (magas), -1 (alacsony) értékekkel jelölhetjük. Ez több szempontból is igen praktikus:

- ilymódon a kísérlet mátrixa (amelynek soraiban az egyes kísérleteknél a faktorok szintjeinek megfelelően +1, illetve -1 áll) ortogonális oszlopvektorú. Ez azt eredményezi, hogy az egyes paraméterek becslése korrelálatlan (normális eloszlású hiba esetén független is) lesz,

- a szintekhez rendelt ±1 számok révén a szorzatuk értelmessé válik, és ez éppen a kölcsönhatás szintjének felel meg: ha a szorzat +1, akkor a két faktor azonos szinten áll, míg a -1 az ellentétes szintnek felel meg. A 2.2.2 alfejezetben részletesebben visszatérünk erre a fontos kérdésre.

2.2.1. Véletlenítés

Ahogy már a bevezetőben említettük, nem tudunk minden potenciális tényezőt faktorként figyelembe venni a kísérlet során. Ha viszont ezeknek a tényezőknek mindig az

azonos (vagy hasonló) szintje esne egybe valamely vizsgált faktor adott szintjével, akkor nem lenne lehetőségünk ennek a két hatásnak a különválasztására. Hiszen nem tudhatjuk, hogy a történetesen megfigyelt jobb eredmény a vizsgált faktornak, vagy a zaj-tényezőnek a következménye-e. Ilyen zajfaktor lehet például

- az idő: a később végzett kísérletek a gép kopása, a kezelő fáradtsága miatt adhatnak rosszabb, de a bemelegedés, tanulás hatására akár jobb eredményt is,
- a kezelő: ha több műszakra húzódik el a kísérlet, akkor a műszakváltás az eredményeket is befolyásolhatja.

Nézzünk néhány további példát a véletlenítésre.

- Egy orvosi kísérletben arra vagyunk kíváncsiak, hogy az új gyógyszer van-e olyan hatásos, mint a hagyományos műtéti kezelés. A vállalkozó betegeket be kell osztanunk két csoportra aszerint, hogy melyik kezelést is kapják. Ha ezt az orvos dönti el, akkor feltehetően a jobb állapotban levő betegeket választaná ki a műtétre, mert az erősen megterheli a szervezetet - egyúttal a súlyosabb állapotú, gyengébb betegek kerülnének a gyógyszeres csoportba. Ennek eredményeként nem tudnánk szétválasztani az általános állapot hatását a műtét hatásától. Ha viszont véletlenítéssel választjuk ki a gyógyszeres kezelésben résztvevőket, akkor ez a keveredés nem lép fel.
- Egy irodában szeretnék tesztelni, hogy két billentyűzet közül melyik a jobb. Ebből a célból mind a 10 titkárnő megkap egy szöveget, amit mindkét billentyűzettel begépel, és a mért idők alapján döntünk arról, hogy melyik a hatékonyabb. Ha minden titkárnő előbb az "A", azután pedig a "B" billentyűzettel dolgozik, akkor lehet, hogy a szöveg ismertsége miatt a második billentyűzet előnyben van. Vagy éppen ellenkezőleg a fáradtság miatt lehet az első billentyűzet előnyben. Nem tudhatjuk előre, melyik tényező jelentkezik a valóságban – de egyértelmű, hogy egyik esetben sem kapunk választ a kérdésünkre, mert nem tudjuk eldönteni, hogy a billentyűzet vagy az idő hatása volt a különbség. Ezért véletleníteni kell: 5 véletlenszerűen kiválasztott titkárnő az "A", az 5 másik pedig a "B" billentyűzettel kezdi a munkát.

A fenti példák jól megvilágították a véletlenítés fontosságát. Az is látható ezekből, hogy véletleníteni akkor is célszerű, ha előre nem látunk olyan okot, ami ezt feltétlenül indokolná. Hiszen általában csupán minimális plusz munkát jelent, de megvéd az esetleges téves következtetésektől. Természetesen nem csak a kísérletek sorrendjét lehet véletleníteni, hanem minden más olyan komponenst is, amelyeket nem szerepeltetünk faktorként (anyag, gép, kezelő stb.).

Ha van olyan tényező, amelyről hatást is feltételezünk, akkor ezt blokkosítással (2.4 pont) be is tudjuk vonni a kiértékelésbe.

	A	B	C	D	E	F	G	H
1	-1	1	1	1	-1	-1	-1	1
2	1	1	1	1	1	1	1	1
3	-1	1	1	-1	-1	1	1	-1
4	-1	1	-1	-1	1	1	-1	1
5	-1	-1	1	1	1	1	-1	-1
6	1	-1	1	1	-1	-1	1	-1
7	1	-1	-1	-1	1	1	1	-1
8	1	1	1	-1	1	-1	-1	-1
9	-1	1	-1	1	1	-1	1	-1
10	1	-1	1	-1	-1	1	-1	1
11	1	-1	-1	1	1	-1	-1	1
12	-1	-1	-1	-1	-1	-1	-1	-1
13	1	1	-1	-1	-1	-1	1	1
14	1	1	-1	1	-1	1	-1	-1
15	-1	-1	-1	1	-1	1	1	1
16	-1	-1	1	-1	1	-1	1	1

2.2. ábra. Véletlenített részfaktoriális terv

A véletlenítés fizikai megvalósításához minden szóbajövő számítógépes programban rendelkezésre állnak véletlen számok – sőt sok célprogram maga alapértelmezésként hozzá is rendel véletlen sorszámot a kísérletekhez. A 2.2 ábra egy ilyen véletlenített részleges faktoriális kísérleti tervet mutat 8 faktorra. Láthatjuk hogy a faktorszintek beállításai nem szisztematikusan váltakoznak.

2.2.2. Szóráselemzés, ANOVA

A szóráselemzés lényege - az egyfaktoros (gyakran egyszempontosnak is nevezett) esetben - a következő: ha a faktornak nincs befolyása a mérési eredményre, akkor az összes egyedi eredményt azonos alapsokaságból származónak tekinthetjük. Ezek, és így az átlagok is csak a közös várható értéktől való véletlenszerű eltéréseknek („kísérleti zajnak“) vannak kitéve. Ellenkező esetben – a faktornak szignifikáns hatása van a mérési eredményre – a faktor szintjeihez tartozó eloszlások várható értékei szignifikánsan különbözőek lesznek.

A modellünk lényege, hogy a számunkra lényeges, optimalizálandó Y mennyiséget véletlennek (matematikai szóhasználattal: valószínűségi változónak) tekintjük. A legegyszerűbb, egyfaktoros modell:

$$Y_{ij} = a_i + \varepsilon_{ij} \tag{2.1}$$

ahol a faktor i-edik szintjén mértük az Y_{ij} értékeket ($j = 1, \ldots, n_i$).. Itt a_i az adott faktorszinten kapott várható érték, ε_{ij} pedig a véletlen hiba (zaj). Ezek az értékek egymástól függetlenek és 0 várható értékűek.

A modellünk valójában egy lineáris modellként is felfogható, ahol a független változók mátrixának minden sorában csak egyetlen nem 0 érték van – éppen az adott faktorszintnek megfelelő oszlopban. Ez részletesen megtalálható például a [24] leírásban.

Az elnevezések arra is utalnak, hogy faktor lehet mennyiségi (kemence hőmérséklete), de minőségi is (alapanyag típusa). Nagyon könnyű a (2.1) összefüggésben szereplő a_i együtthatók becslése: egyszerűen vehetjük az adott szinten megfigyelt értékek átlagát. Ugyanakkor a fő kérdés az, hogy vajon az adott faktor hatása (tehát az a_i-re kapott becslések értékeinek eltérése) szignifikáns-e, azaz kellően nagy-e annak a valószínűsége, hogy a kísérletek megismétlése esetén is ugyanilyen irányú eltéréseket kapunk-e. Ennek a matematikai vizsgálatára alkalmas a szóráselemzés.

Az egyszempontos szóráselemzés során k független, normális eloszlású, azonos szórásnégyzetű alapsokaságot tételezünk fel, és azt a nullhipotézist vizsgáljuk, hogy az összes középérték azonos $a_1 = a_2 = ... = a_k = \mu$, tehát az eredményeink azonos várható értékű alapsokaságokból származnak. Mivel azonos szórásnégyzeteket tételeztünk fel, a nullhipotézis egyúttal azt is jelenti, hogy az összes mérési érték egy és ugyanazon alapsokaságból származik.

A gyakorlatban, hogy a különbségek (hatások) vizsgálata szemléletesebb és matematikailag egyszerűbb legyen, általában az

$$Y_{ij} = \alpha_i + \mu + \varepsilon_{ij} \tag{2.2}$$

modellt alkalmazzák, ahol α_i az i-edik szint hatása, μ pedig a fentiekben definiált átlagos hatás.

Mivel csak k csoportunk van és $k+1$ paraméterünk, ezért egyiküket tetszés szerint beállíthatjuk. Ez a választás azonban nem érinti a módszer eredményét, csupán a képletek alakját módosítja. Talán a leggyakoribb az a választás, ami szerint

$$\mu = \frac{1}{N} \sum_{i=1}^{k} n_i a_i$$

ahol n_i az i-edik szinten végzett kísérletek száma, N pedig ezek összege (a teljes kísérleti terv elemszáma). Így a hatások súlyozott átlaga lesz 0:

$$\sum_{i=1}^{k} n_i \alpha_i = 0.$$

Abban a tipikus esetben, amikor minden szinten ugyanannyi kísérletet végeztünk, a súlyozott átlagok helyett egyszerű számtani átlagokat vehetünk.

Az **R** program ugyanakkor azt a módszert alkalmazza, hogy az első faktorszint hatását választja referenciának, azaz 0-nak és a többi értéket ehhez viszonyítja.

Az ismeretlen hatásokat az adataink alapján becsülhetjük, a következőképpen: legyen

$$\overline{y}_{i\cdot} = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}$$

az i-edik szinten az eredmények átlaga. A főátlag (az összes megfigyelés átlaga):

$$\overline{y}_{\cdot\cdot} = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{n_i} y_{ij}.$$

Ha a csoportokban a hatások eltérőek is lehetnek, akkor az a_i középérték torzítatlan becslése

$$\hat{a}_i = \overline{y}_{i\cdot},$$

míg az azonosnak feltételezett középértékek esetén

$$\hat{\mu} = \overline{y}_{\cdot\cdot}.$$

Ebből az i-edik szint hatásának becslése:

$$\hat{\alpha}_i = \hat{a}_i - \hat{\mu} = \overline{y}_{i\cdot} - \overline{y}_{\cdot\cdot}.$$

Az úgynevezett belső négyzetösszeg (a csoportokon belüli eltérések négyzetösszege, a "W" index a "within" szó rövidítése):

$$SS_W = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (y_{ij} - \overline{y}_{i\cdot})^2$$

A megfigyeléseink szórását is becsülnünk kell. Itt kihasználhatjuk, hogy minden szinten ugyanaz a szórás, ezért

$$\hat{\sigma}^2 = MS_W = \frac{SS_W}{N-k} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{n_i} (y_{ij} - \overline{y}_{i\cdot})^2}{N-k}. \tag{2.3}$$

A nevezőben azért szerepel $N-k$, mert minden csoportban kapunk egy $n_i - 1$ szabadságfokú becslést és ezekből az összeg szabadságfoka $N-k$, tehát (2.3) torzítatlan becslés σ^2-re, függetlenül attól, hogy melyik hipotézis is az igaz.

A csoportok közötti különbséget méri a csoportok közötti eltérés-négyzetösszeg (a "B" index a "between" szó rövidítése):

$$SS_B = \sum_{i=1}^{k} n_i (\overline{y}_{i\cdot} - \overline{y}_{\cdot\cdot})^2.$$

Ennek szabadságfoka értelemszerűen $k-1$, hiszen k átlagot hasonlítunk össze úgy, hogy egy paramétert becsültünk (a főátlagot).

A két négyzetösszeg összege éppen a teljes négyzetösszeg (SS_T):

$$SS_T := \sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{..})^2 = SS_B + SS_W.$$

Ennek bizonyítása egyszerű, csak be kell hozni a jobboldalon látható négyzetösszegeket az egyszerű

$$\sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{..})^2 = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(y_{ij}-\overline{y}_{i.}+\overline{y}_{i.}-\overline{y}_{..})^2$$

átalakítással és észre kell venni, hogy a négyzetek kifejtésénél a kétszeres szorzatok kiesnek.

A hipotézisvizsgálatra a lineáris modellnél alkalmazható (l. például [22]) F-próbát használhatjuk:

$$f = \frac{SS_B/(k-1)}{SS_W/(N-k)}$$

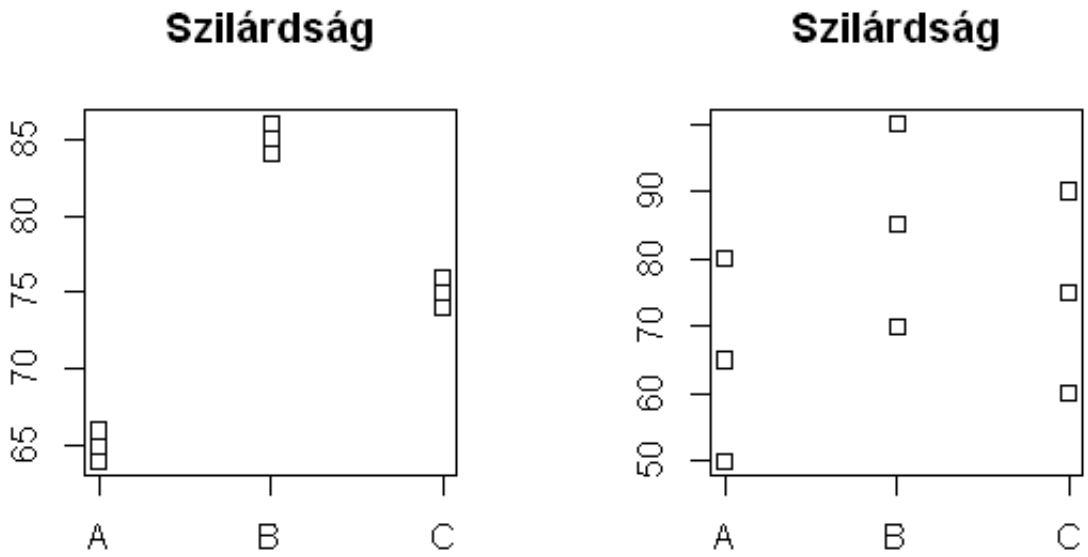
A nullhipotézis (azaz nincsen különbség a szintek között) esetén f éppen F eloszlású $k-1, N-k$ szabadságfokokkal. A próba tehát akkor utasítja el a nullhipotézist α elsőfajú hibavalószínűség mellett, ha f értéke nagyobb, mint a megfelelő F eloszlás $1-\alpha$ kvantilise.

A módszereket egy egyszerű példán szemléltetjük. Tegyük fel, hogy acéldrótok szakítószilárdságára vonatkozóan két kísérletet is végeztünk. Az eredményeket a 2.3 ábra mutatja. A két diagram két különböző mérési eljárás eredményét tartalmazza. Jól látható, hogy a baloldali sokkal pontosabb, kisebb hibájú, míg a jobboldalon szereplő módszer hibája sokkal nagyobb – de az átlagok azonosak a két esetre.

A minta-adatokra a következő R-kód végzi el a szóráselemzést:

```
library(doBy)
ex.data <- read.csv("anova-example.csv", header=TRUE)
for(exp.index in 1:2){
   cat("\n\n*****",exp.index, "kísérlet eredménye "," *****\n\n")
   temp <- ex.data[ ex.data[,"Experiment"] == paste("Experiment",
      exp.index),]
   result <- lm( y ~ method, data=temp)
   print(result)
   print(anova(result))
}
```

Az eredményeket a 2.4 ábrán láthatjuk. Mindkét esetben ugyanazok a hatás-becslések adódtak, és emiatt a csoportok közötti szórásnégyzet (itt: "method") is megegyezik.



2.3. ábra. Acéldrótok szakítószilárdságának mérése két módszerrel

Ahogy már említettük, itt a "B" és "C" szintek hatása a 0-nak tekintett "A" szint hatásához képest értendő. Viszont jól látható, hogy a módszerek közötti különbség csak az 1. kísérlet esetén bizonyult szignifikánsnak, a szórások közötti markáns különbség miatt.

Az F próba statisztikája az első esetben 300, ami minden reális szinten szignifikáns - a másik esetben viszont csupán 1.33 a statisztika értéke, ami természetesen nem jelez szignifikáns eltérést.

A gyakorlatban persze tipikusan nem egy, hanem több faktor befolyásolja a végeredményt. A kétfaktoros esetre a 2.1 modell a következőképpen általánosítható.

$$Y_{ijk} = a_{ij} + \varepsilon_{ijk}$$

ahol a_{ij} az 1. faktor i. és a 2. faktor j. szintjén a hatás. Ezen a szint-kombináción az y_{ijk} értékeket mértük ($k = 1, \ldots, n$, itt általában fel szokás tenni, hogy minden szint-kombinációra ugyanannyi megfigyelést végeztünk). Az ε_{ijk} a véletlen hiba (zaj), ezek az értékek egymástól függetlenek és 0 várható értékűek. A struktúrát (az egyik faktor szerint 4, a második szerint 3 szinten végezve kísérleteket) a 2.5 ábra mutatja.

A kétfaktoros kísérlet értelemszerűen tartalmaz egyfaktoros rész-tervet is. Így az előzőeknek megfelelően az első faktorhoz tartozó α_i és a másodikhoz tartozó β_j faktorhatások ugyanúgy definiálhatók, mint az előzőekben. Ami új, az a faktorok közötti

```
***** 1 kísérlet eredménye   *****


Call:
lm(formula = y ~ method, data = temp)

Coefficients:
(Intercept)      methodB      methodC
         65           20           10

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
method     2    600     300     300 9.706e-07 ***
Residuals  6      6       1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


***** 2 kísérlet eredménye   *****


Call:
lm(formula = y ~ method, data = temp)

Coefficients:
(Intercept)      methodB      methodC
         65           20           10

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value Pr(>F)
method     2    600     300  1.3333 0.3318
Residuals  6   1350     225
```

2.4. ábra. Hatások szignifikanciavizsgálata két kísérletnél

kölcsönhatás. A két faktor esetére ez a következő:

$$a_{ij} - \alpha_i - \beta_j + \mu$$

aminek a szemléletes jelentése az, hogy a két faktor additív hatásától mennyire tér el a tényleges hatás az i, j szint-párra. A szórásfelbontó táblázat ebben az esetben kicsit bonyolultabb:

$$SS_T = SS_A + SS_B + SS_{AB} + SS_W$$

ahol SS_T a teljes négyzetösszeg, SS_A az A faktor hatását mérő, SS_B pedig a B faktor hatását mérő négyzetösszeg. SS_{AB} a kölcsönhatáshoz tartozik, SS_W pedig a csoportokon belüli (hiba) négyzetösszeg, hasonlóan az egyfaktoros esethez (ezt az "error"-hiba szó kezdőbetűjéből gyakran SS_E-vel jelölik). A képletek (a-val és b-vel jelölve az A, illetve a

	B1	B2	B3
A1	y_{111} $\vdots$ y_{11n}	y_{121} $\vdots$ y_{12n}	y_{131} $\vdots$ y_{13n}
A2	y_{211} $\vdots$ y_{21n}	y_{221} $\vdots$ y_{22n}	y_{231} $\vdots$ y_{23n}
A3	y_{311} $\vdots$ y_{31n}	y_{321} $\vdots$ y_{32n}	y_{331} $\vdots$ y_{33n}
A4	y_{411} $\vdots$ y_{41n}	y_{421} $\vdots$ y_{42n}	y_{431} $\vdots$ y_{43n}

2.5. ábra. Kétfaktoros kísérlet, y jelöli az eredményeket

Source	DF	SS	MS	F
A	a-1	SS_A	$SS_A/(a-1)$	MS_A/MS_E
B	b-1	SS_B	$SS_B/(b-1)$	MS_B/MS_E
AB	(a-1)(b-1)	SS_{AB}	$SS_{AB}/[(a-1)(b-1)]$	MS_{AB}/MS_E
Error	(n-1)ab	SS_E	$SS_E/[(n-1)ab]$	

2.6. ábra. Szórásfelbontó ANOVA tábla kétfaktoros kísérletre

B faktor szintjeinek a számát és n-nel a szintenként végzett kísérletekét):

$$SS_T := \sum_{i,j,k=1}^{a,b,n} (y_{ijk} - \overline{y}_{...})^2,$$

$$SS_A := \sum_{i=1}^{a} nb(\overline{y}_{i..} - \overline{y}_{...})^2,$$

$$SS_B := \sum_{j=1}^{b} na(\overline{y}_{.j.} - \overline{y}_{...})^2,$$

$$SS_{AB} := \sum_{i,j=1}^{a,b} n(\overline{y}_{ij.} - \overline{y}_{i..} - \overline{y}_{.j.} + \overline{y}_{...})^2,$$

$$SS_W := \sum_{i,j,k=1}^{a,b,n} (y_{ijk} - \overline{y}_{ij.})^2.$$

A hipotézisek: először is a kölcsönhatást célszerű tesztelni. Ha elfogadható a kölcsönhatás hiánya, akkor pedig sorra vehetjük a faktorok hatását. Ezek tesztelésére is alkalmas az F-próba. A szabadságfokokat az átlagos szórásnégyzeteket és az F-próbák statisztikáit mutatja be a 2.6 ábra.

Az **R** segítségével meg is tudjuk jeleníteni a kölcsönhatást. A következő példában [19] PVC részecske-méretét befolyásoló faktorokat vizsgálunk. Három kezelő 8 féle eszközt használt (resin railcar).

```
library(faraway)
source("http://www.rohan.sdsu.edu/~babailey/stat700/pvc.R")
attach(pvc)
stripchart(psize ~ resin, xlab="Particle size", ylab="Resin railcar")
 stripchart(psize ~ operator, xlab="Particle size", ylab="Operator")
 interaction.plot(operator, resin, psize)
 interaction.plot(resin, operator, psize)
```

```
Analysis of Variance Table

Response: psize
               Df  Sum Sq Mean Sq F value    Pr(>F)
operator        2  20.718  10.359  7.0072   0.00401 **
resin           7 283.946  40.564 27.4388 5.661e-10 ***
operator:resin 14  14.335   1.024  0.6926   0.75987
Residuals      24  35.480   1.478
```

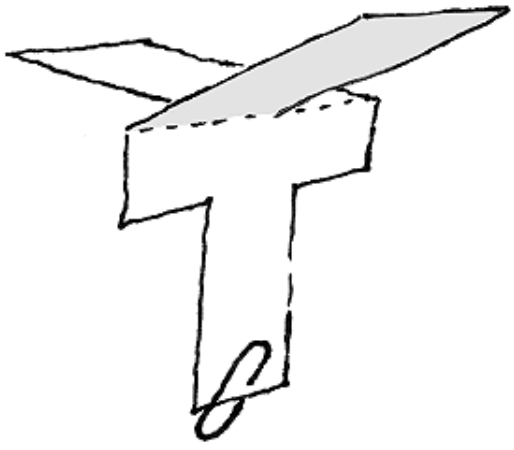
2.7. ábra. ANOVA tábla a kétfaktoros, PVC-részecskék méretére vonatkozó kísérletre

A 2.7 ábra a kétszempontos szóráselemzés táblázata a PVC adatokra. Azt olvashatjuk le, hogy a főhatások szignifikánsak, de a kölcsönhatás nem.

2.2.3. Példa: papírhelikopter-tervezés

A faktoriális tervezés módszerét egy, az oktatásban könnyen reprodukálható és a hallgatók számára érdekes kísérlettel illusztráljuk. Például a http://www.paperhelicopterexperiment.com/ címen található részletes leírás a "projektről". Ennek során a résztvevők először egy minta-helikopteren nézik meg a prototípust és javasolnak faktorokat, amelyekkel a repülési idő feltehetően növelhető. Az ötletroham során számos javaslat felmerülhet, de a teljes faktoriális kísérleti terv kivitelezhetősége érdekében célszerű 4-5 faktor kiválasztása. A 2.8 kép magát a helikoptert mutatja.

Ha minden faktort két beállítással veszünk be a kísérletbe, akkor k faktor esetén a teljes faktoriális terv 2^k kísérletből fog állni. Ez még ismétlésekkel együtt is elvégezhető egy 45 perces óra során $k = 4$ vagy $k = 5$ esetén.



2.8. ábra. A papír helikopter

A bemutatásra kerülő kísérletben az alábbi faktorokat és szinteket vizsgáltuk:

- FH: felhajtás a szárny végén: igen vagy nem,
- GS: gemkapcsok száma: 2 vagy 1,
- PA: papír,normál iratpapír vagy félfamentes rajzlap,
- SH: szárnyhossz, normál (7cm) vagy rövid (5.5 cm),
- SS: szárny szélesség, széles (7cm) vagy normál (5cm).

A szintek közül mindig az első volt a "felső" (kódja +1) és a második az "alsó" (kódja -1). Ennek ott van jelentősége, hogy a hatás számértékénél az előjelet helyesen értelmezzük: pozitív érték azt muttja, hogy az adott faktor felső értéke adta a jobb eredményt.

Az adatok elemzését az **R** FRF2 csomagjával végezzük [12]. Először ki kell számolnunk a hatások és kölcsönhatások becslését. Ha az adattömbben csak a faktorbeállítások és az eredmények vannak (mivel a gyakorlaton több csoportban is történtek mérések, először ezek átlagát tekintettük eredménynek), akkor egyszerűen az alábbi utasítást alkalmazhatjuk a főhatások és a másodrendű kölcsönhatások becslésére.

```
h.lm <- lm(heli2$Átlag ~ (.)^2,data=heli2)
```

Mivel a kétszintű teljes faktoriális terv ortogonális, ezért itt a hatások becslése egyszerűen $\overline{y}_+ - \overline{y}_-$, azaz a pozitív szinteken mért eredmények átlaga mínusz a negatív szinteken mért eredmények átlaga.

Ezután a kapott eredményeket elrendezhetjük táblázatban, a lineáris modellnél megszokott módon (2.9 ábra).

```
summary(h.lm)
```

Azonban meg kell jegyeznünk, hogy itt a szignifikancia ellenőrzése abból a feltevésből indul ki, hogy az adott szinten kapott mérési eredmények függetlenek és azonos eloszlásúak, azonban ez számos – önmagában esetleg nem szignifikáns – tényező hatása miatt nem teljesül pontosan, tehát további vizsgálatokra van szükség.

A hatásokat grafikusan leggyakrabban az úgynevezett "half normal plot" segítségével vizsgálhatjuk (2.10 ábra). Ehhez viszont célszerű megbecsülnünk az összes lehetséges kölcsönhatást, hogy legyen kellő számú viszonyítási alapunk a szignifikancia kiderítéséhez. Ekkor azt vizsgáljuk, hogy mekkora eltérést kapunk, ha a becsült hatásokat a standard normális eloszlású X-ből számolt $|X|$ eloszlásához hasonlítjuk a QQ-plotnál látott módon, azaz a nagyság szerint sorbarendezett minta elemeit egybevetve $|X|$ eloszlásának kvantiliseivel. Ha minden eltérés csak véletlenszerű, akkor a homoszkedaszticitás miatt minden becslés azonos normális eloszlású, amit az ábra közel lineáris volta mutat. A mi esetünkben ez nincsen így, a legfontosabb 5 tényező tűnik szignifikánsnak az $\alpha = 0,05$ szinten. A kód:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.260104   0.006783 185.783  < 2e-16 ***
PA           0.136701   0.006783  20.154 8.50e-13 ***
GS          -0.057928   0.006783  -8.541 2.35e-07 ***
FH          -0.135822   0.006783 -20.025 9.39e-13 ***
SH           0.148183   0.006783  21.847 2.44e-13 ***
SS          -0.012581   0.006783  -1.855 0.082136 .
PA:GS       -0.039201   0.006783  -5.780 2.82e-05 ***
PA:FH       -0.027650   0.006783  -4.077 0.000879 ***
PA:SH        0.030660   0.006783   4.520 0.000349 ***
PA:SS       -0.027650   0.006783  -4.077 0.000879 ***
GS:FH        0.024109   0.006783   3.554 0.002641 **
GS:SH        0.018669   0.006783   2.752 0.014163 *
GS:SS        0.011470   0.006783   1.691 0.110209
FH:SH       -0.026539   0.006783  -3.913 0.001240 **
FH:SS       -0.017025   0.006783  -2.510 0.023196 *
SH:SS       -0.010938   0.006783  -1.613 0.126387
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03837 on 16 degrees of freedom
Multiple R-squared: 0.9894,     Adjusted R-squared: 0.9795
```

2.9. ábra. A főhatások és a kétszeres kölcsönhatások és szignifikanciájuk becslése

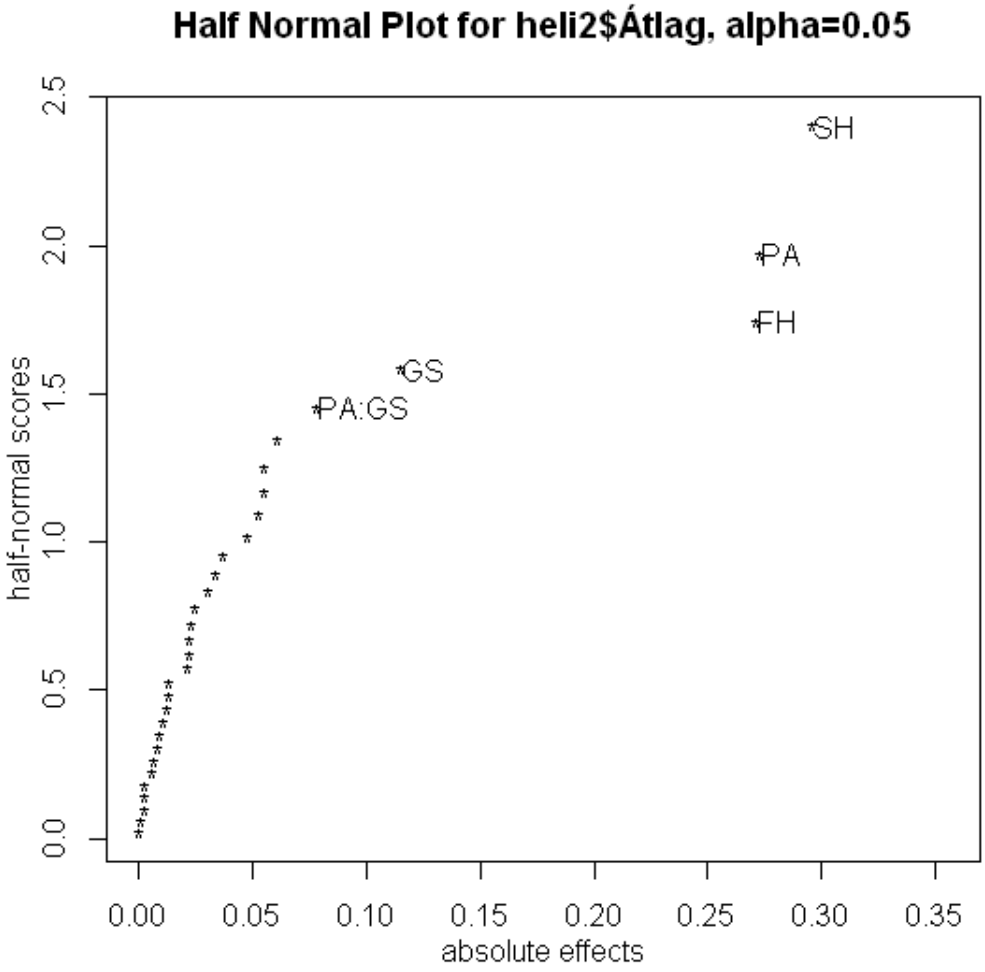
```
DanielPlot(h.lm,alpha=0.05,half=TRUE)}
```

Ugyanakkor nem felejthetjük el, hogy csupán a véletlen műve is lehet a szignifikánsnak látszó eredmény. A 2.11 ábra a http://hpz400.cs.elte.hu:3838/ZA_glm/ címen található animációból származik. Itt független azonos normális eloszlású véletlen számok a kísérletünk eredményei, és meglehetősen gyakran kapunk az $\alpha = 0.1$ esetén szignifikánsnak tűnő hatásokat. A 2.11 ábrán 4 faktort képzeltünk el, a kölcsönhatásokkal együtt ez 10 pontot ad, amik közül 4 is szignifikánsnak tűnő eredményt adott.

2.3. Részfaktoriális tervek

Ahogy ezt már a bevezetőben is említettük, a teljes faktoriális tervek sok faktor esetén gyakorlatilag kivitelezhetetlenek. Ezért – mintegy kompromisszumként – részleges faktoriális terveket lehet helyettük elvégezni. Ezek lényege, hogy nem minden faktorkombinációhoz tartozik kísérlet, hanem csak a felét (negyedét, 2^k-ad részét) végezzük el.

Ugyanakkor itt is érvényes a mondás, hogy nincsen ingyen ebéd, a kihagyott kísérletek ára a különböző hatások nem megkülönböztethető keveredése. Ezt angolul "alias" struktúrának nevezik. A jelenség lényege az, hogy ha két (általában magasabbrendű)

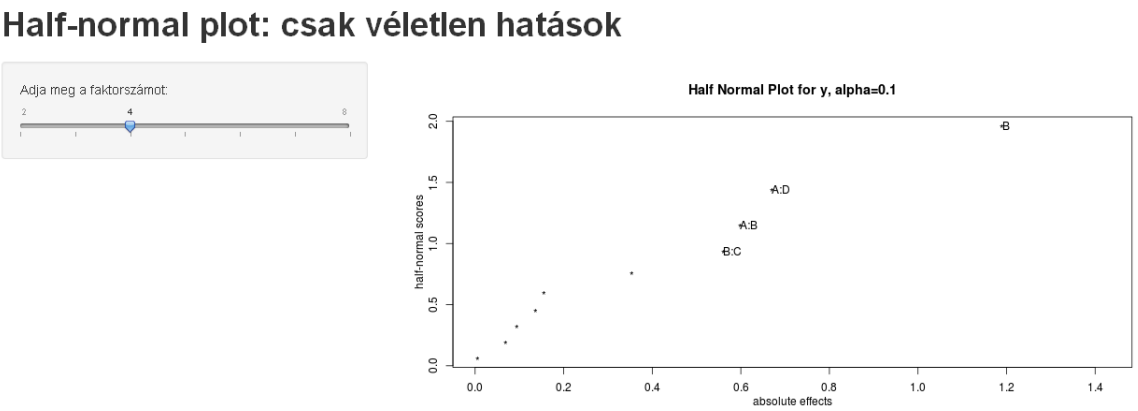


2.10. ábra. Half normal plot a helikopter kísérletnél

hatás minden kísérletben ugyanazon a szinten szerepel, akkor semmilyen módon nem lehet őket elkülöníteni.

Matematikailag is meg lehet ezt a jelenséget fogalmazni. Láttuk, hogy a magasabbrendű hatások szintjei is a bennük szereplő faktorok szintjeinek szorzataként határozhatóak meg. Ha két faktor-kombináció minden kísérletben ugyanazon a szinten szerepel (a hozzájuk tartozó értékek szorzata azonos), akkor ezen kombinációk hatásai nem különíthetőek el: nincs semmilyen módszer arra, hogy eldöntsük, melyik is a lényeges. Ezeket a faktor-kombinációkat egymás aliasainak nevezzük.

De a gyakorlatban ez nem mindig jelent problémát: a harmadrendű és különösen a



2.11. ábra. Half normal plot teljesen véletlen adatokra

még magasabb rendű kölcsönhatások ritkán lépnek fel, ezért ha ők keverednek főhatással vagy alacsonyabb rendű kölcsönhatással, akkor feltételezhetjük, hogy az alacsonyabb rendű hatás a domináns.

A részfaktoriális terveket az úgynevezett generátoraikkal adhatjuk meg. Ezek olyan egyenletek, amik minden elvégzett kísérletre teljesülnek. Tekintsük például a 2.12 ábrában látható 2^{5-2} tervet, ami 32 helyett csak 8 kísérletet tartalmaz. Ennek generátora $1 = ABC = -CDE$. Általában is igaz, hogy ha a teljes faktoriális terv negyedét végezzük el, akkor két egyenletet adhatunk meg (mindegyik külön-külön felezi a teljes tervet), és a 2^r részhez pedig k egyenlet tartozik.

	A	B	C	D	E	AB	$\cdots$	ABCDE
ce	−	−	+	−	+	+	$\cdots$	−
a	+	−	−	−	−	−		+
b	−	+	−	−	−	−		+
$abce$	+	+	+	−	+	+		−
cd	−	−	+	+	−	+		−
ade	+	−	−	+	+	−		+
bde	−	+	−	+	+	−		+
$abcd$	+	+	+	+	−	+	$\cdots$	−

2.12. ábra. 2^{5-2} terv táblázata

Nagyon lényeges, hogy az alias struktúrát pontosan meghatározzuk. A fenti példában, ahol $1 = ABC = -CDE = -ABDE$ (az utolsó összefüggést úgy kaptuk, hogy a

Felbontás	Tulajdonságok	Példa
II	Nem használható: főhatások is keverednek	2^{2-1}
III	A főhatások becsülhetőek, de keverednek másodrendű kölcsönhatásokkal	2^{3-1}
IV	A főhatások csak magasabb rendű kölcsönhatásokkal keverednek, a másodrendű kölcsönhatások keverednek egymással	2^{4-1}
V	A főhatások csak harmadrendűnél is magasabb rendű kölcsönhatásokkal keverednek, a másodrendű kölcsönhatások keverednek harmadrendűekkel	2^{5-1}

2.1. táblázat. A felbontások és tulajdonságaik

$-CDE$ kifejezést 1-gyel, azaz ABC-vel megszoroztuk és kihasználtuk, hogy $C^2 = 1$). Az egyenlőtlenségláncot végig szorozva a faktorokkal megkaphatjuk az alábbi, teljes alias-struktúrát.

I	=	ABC	=	–CDE	=	–ABDE
A	=	BC	=	–ACDE	=	–BDE
B	=	AC	=	–BCDE	=	–ADE
C	=	AB	=	–DE	=	–ABCDE
D	=	ABCD	=	–CE	=	–ABE
E	=	ABCE	=	–CD	=	–ABD
AD	=	BCD	=	–ACE	=	–BE
BD	=	ACD	=	–BCE	=	–AE

2.13. ábra. A 2^{5-2} terv alias struktúrája, I jelöli az identitást (az 1-et)

A részfaktoriális tervek eredményeinek elemzése hasonlóan végezhető el a szóráselemzés módszereivel, mint a teljes faktoriális elrendezésé. Ugyanakkor tipikusan nincs ismétlés, amiből közvetlen becslést kaphatnánk a szórásnégyzetre, ezért azt a nem modellezett kölcsönhatások helyett becsülhetjük (”surrogate error”). A half-normal plot is ugyanúgy használható, mint a teljes faktoriális esetben.

A részfaktoriális tervek ”minőségét” az úgynevezett felbontás méri. A 2.1 táblázat mutatja ezek tulajdonságait.

Példaként tekintsük az FrF2 csomag egyik minta adatsorát. A molding adatsor 8 faktort tartalmaz Ez eredetileg 16 kísérletet tartalmazó részfaktoriális terv. Az alias struktúráját a

```
data(BM93.e3.data)
```

```
iMdat <- BM93.e3.data[1:16,2:10] #csak az eredeti kísérlet
# oszlopnevek
colnames(iMdat) <- c("MoldTemp","Moisture","HoldPress","CavityThick",
"BoostPress","CycleTime","GateSize","ScrewSpeed","y")
#  aliasok a 2-faktor-kölcsönhatásokra
aliases(lm(y ~ (.)^2, data = iMdat))
# kódolva
aliases(lm(y ~ (.)^2, data = iMdat), code=TRUE)
```

kód adja meg. Magát az adatsort a 2.14 ábra mutatja be .

A	B	C	D	E	F	G	H	y
-1	-1	-1	1	1	1	-1	1	14.0
1	-1	-1	-1	-1	1	1	1	16.8
-1	1	-1	-1	1	-1	1	1	15.0
1	1	-1	1	-1	-1	-1	1	15.4
-1	-1	1	1	-1	-1	1	1	27.6
1	-1	1	-1	1	-1	-1	1	24.0
-1	1	1	-1	-1	1	-1	1	27.4
1	1	1	1	1	1	1	1	22.6
1	1	1	-1	-1	-1	1	-1	22.3
-1	1	1	1	1	-1	-1	-1	17.1
1	-1	1	1	-1	1	-1	-1	21.5
-1	-1	1	-1	1	1	1	-1	17.5
1	1	-1	-1	1	1	-1	-1	15.9
-1	1	-1	1	-1	1	1	-1	21.9
1	-1	-1	1	1	-1	1	-1	16.7
-1	-1	-1	-1	-1	-1	-1	-1	20.3
-1	1	1	1	-1	-1	-1	1	29.4
-1	1	-1	-1	-1	1	1	1	19.7
1	1	-1	-1	1	-1	-1	1	13.6
1	1	1	1	1	1	1	1	24.7

2.14. ábra. A BM93.e3 adatsor

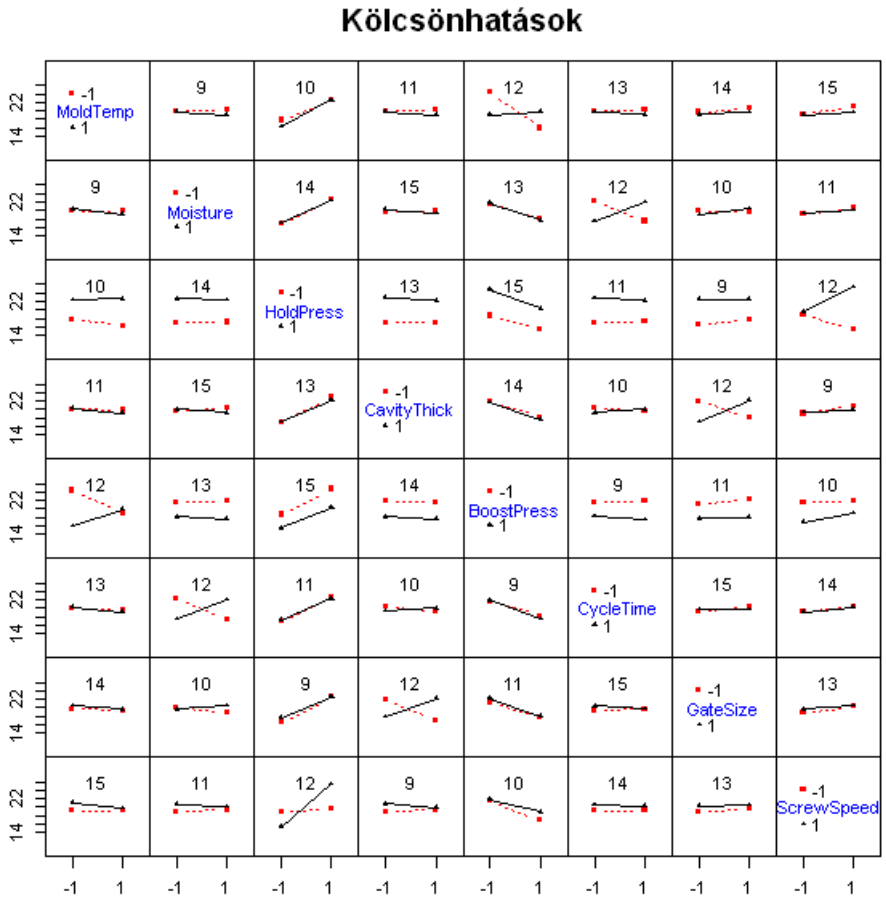
Ez egy III felbontású terv, a 2.15 alias struktúrával.

A kölcsönhatások ábrájánál meg tudjuk jeleníteni az alias struktúrát (2.16). Az elemzést a következő programrészlet végzi el:

```
# lineáris modell fôhatásokkal  és a kétszeres kölcsönhatásokkal
iM.lm <- lm(y ~ (.)^2, data = iMdat)
aliases(iM.lm, code=TRUE)
#kölcsönhatás diagram az alias struktúrával
IAPlot(iM.lm, show.alias=TRUE,main="Kölcsönhatások")
```

```
   A=MoldTemp     B=Moisture    C=HoldPress D=CavityThick  E=BoostPress

  F=CycleTime     G=GateSize   H=ScrewSpeed

$aliases

 A:B = C:G = D:H = E:F
 A:C = B:G = D:F = E:H
 A:D = B:H = C:F = E:G
 A:E = B:F = C:H = D:G
 A:F = B:E = C:D = G:H
 A:G = B:C = D:E = F:H
 A:H = B:D = C:E = F:G
```

2.15. ábra. A BM93.e3 adatsor alias struktúrája



2.16. ábra. A BM93.e3 kölcsönhatás diagramja

Az eredményből láthatjuk, hogy minden kétszeres kölcsönhatásnak van kétszeres kölcsönhatás aliasa és a főhatásoknak pedig háromszoros kölcsönhatás aliasa.

A gyakorlatban az FrF2 csomag egyik fő előnye éppen a kívánt felbontású, faktorszámú kísérleti terv generálása. A beépített függvény és legfontosabb paraméterei:

```
FrF2(nruns = NULL, nfactors = NULL,
default.levels = c(-1, 1), ncenter=0, center.distribute=NULL,
generators = NULL,
resolution = NULL,
randomize = TRUE,
blocks = 1, hard = NULL,  ...)
```

A fenti függvényben

- 'nruns' a kísérletek száma,
- 'nfactors' a faktorok száma,
- 'default.levels' a kísérletek szintjeinek jelölése,
- 'ncenter' a középpontban végzett kísérletek száma,
- 'center.distribute' a középpontban végzett kísérletek helye a tervben,
- 'generators' megadja a tervet definiáló egyenletek jobboldalát. Itt ezt úgy kell érteni, hogy a baloldal mindig egy új faktor – tehát abból indulunk ki, hogy nem a kísérletek számát csökkentjük a definiáló egyenletek révén, hanem minden egyes egyenlet egy új faktort jelent a modellben (amely természetesen keveredik a definiáló egyenlet kölcsönhatásával),
- 'resolution' a kísérleti terv felbontása,
- 'randomize' a véletlenítés,
- 'blocks' a blokkok száma,
- ' hard' a nehezen beállítható faktorok listája – ezeket a kísérletek sorrendjének optimális megválasztásával olyan kevésszer módosítjuk ami csak lehetséges.

Egy példa a függvény konkrét futtatására és az eredmény (2.17 ábra):

```
FrF2(16, generators = c("ABCD","ABC"))
```

	A	B	C	D	E	F
1	-1	1	1	1	-1	-1
2	-1	-1	-1	-1	1	-1
3	-1	1	1	-1	1	-1
4	-1	1	-1	1	1	1
5	1	1	1	1	1	1
6	-1	-1	1	1	1	1
7	1	-1	1	-1	1	-1
8	1	-1	-1	1	1	1
9	1	1	1	-1	-1	1
10	1	-1	-1	-1	-1	1
11	1	1	-1	-1	1	-1
12	1	1	-1	1	-1	-1
13	1	-1	1	1	-1	-1
14	-1	-1	-1	1	-1	-1
15	-1	-1	1	-1	-1	1
16	-1	1	-1	-1	-1	1

2.17. ábra. 16 kísérletből álló III felbontású terv 6 faktorra

2.4. Blokkosítás

Sokszor olyan tényezők is hatnak, amiket nem tudunk vagy nem akarunk a kísérletben tervezetten faktorként szerepeltetni (például a műszak hatása ipari termelésnél, homogén földterület mezőgazdasági terveknél). Ekkor ezeket a faktorokat úgynevezett blokkoknak tekintjük és a többi faktor értékét kiegyensúlyozottan állítjuk be a blokkok különböző értékei között. Ennek eredményeként a blokk-hatásra is kapunk becslést. Ez önmagában is hatalmas terület, amelyből csak felvillantani tudunk részleteket.

Ha a blokkok elég nagyok, hogy minden kísérletet (a blokkosítás szakirodalmában gyakran ”kezelésnek” nevezik, mert itt már nemcsak faktoriális tervekre lehet gondolni) minden blokkban el tudjunk végezni, akkor teljes blokkos kísérleti tervről beszélünk és ez lényegében megfelel a teljes faktoriális tervnek azzal a formális különbséggel, hogy a blokk az egyik faktor.

A blokkosítás azért nagyon lényeges, mert így egy fontos zaj-faktort kiszűrünk és ezzel a szórást jelentősen tudjuk csökkenteni. A szokásos ANOVA módszerekkel vizsgálható, hogy vajon a blokk-hatás szignifikáns-e.

Ha a blokkok nem elég nagyok ahhoz, hogy minden kezelés elvégezhető legyen egy blokkban (kicsi a homogén földterület, sokáig tart a kísérlet és nem fér bele egy mű-

szakba az összes), akkor nem teljes blokkos tervről beszélünk. Ekkor arra törekszünk, hogy minden kezelés-pár ugyanannyiszor szerepeljen egy blokkban. A 2.18 ábra néhány egyszerű példát mutat kiegyensúlyozott nem teljes blokkos tervekre. A paraméterek:

- a a kezelések száma,
- b a blokkok száma,
- k a blokkonkénti kezelések száma,
- r hányszor fordul elő egy kezelés,
- λ a párok hányszor fordulnak elő egy blokkban.

.

$$a = 3, b = 3, k = 2 \rightarrow r = 2, \lambda = 1$$

BLOCK

1	2	3
A	B	A
B	C	C

$$a = 4, k = 2, b = 6 \rightarrow r = 3, \lambda = 1$$

BLOCK

1	2	3	4	5	6
A	A	A	B	B	C
B	C	D	C	D	D

$$a = 4, k = 3, b = 4 \rightarrow r = 3, \lambda = 2$$

BLOCK

1	2	3	4
A	A	A	B
B	B	C	C
C	D	D	D

2.18. ábra. Példák kiegyensúlyozott nem teljes blokkos tervekre

2.5. Az R kísérlettervezési csomagjainak bemutatása

A [28] honlap folyamatosan figyelemmel kíséri a témával foglalkozó csomagokat. A jegyzet készítésekor a legújabb verzió 2013 máricusi volt. A következő csomagok a leggyakrabban használtak:

- GAD: ANOVA terveket tud kezelni fix és véletlen hatások esetére is,
- A DoE.base és az FrF2 csomagok alapján készült egy menüvezérlésű rendszer, az RcmdrPlugin.DoE, ami azok számára, lehet előnyös, akik nem kedvelik a parancssoros programozást,
- conf.design: különböző kölcsönhatásokat tartalmazó és Taguchi-tervek is készíthetők a segítségével,
- AlgDesign: különböző optimális terveket és keverékekre vonatkozó terveket készít,
- blockTools: blokkokhoz rendel kísérleti egységeket – különösen hasznos kis blokkméretek esetén.

3. fejezet

Nem-lineáris regresszió

3.1. Bevezető

A nem-lineáris modellek a lineárisaktól pusztán abban a technikailag nem mellékes dologban különböznek, hogy a nem-lineáris modellek alkalmazásakor a célváltozó értékeit a magyarázó változók olyan függvényével közelítjük, amely a *paramétereitől* nem-lineárisan függ. Az egyváltozós nem-lineáris modellek általános formája az

$$y_i = f(x_i, \theta) + e_i, \quad i = 1, \ldots, n$$

ahol úgy vesszük, hogy az $x_1, \ldots, x_n$ beállított, ismert értékek, amikre a rendszer $e_1, \ldots, e_n$ hibákkal mért válasza az $y_1, \ldots, y_n$. A hibákról feltételezzük, hogy a $\varepsilon_1, \ldots, \varepsilon_n$ független, 0 várható értékű, azonos szórású véletlen mennyiségeknek a mérést leiró ω mellett adódott értékei. Az előállításban a θ ismeretlen paraméter, amitől az $f(x_i, \theta)$ függvény értéke nem-lineárisan függ. A nem-lineáris regresszió feladata: az (x_i, y_i) és az $f(x, \theta)$ ismeretében becslés készítése a θ paraméterre és valamiféle mértékét adni a $\hat{\theta}$ becslés megbízhatóságára. És esetleg további információkat arra vonatkozóan, hogy a modellosztály egy esetleges átparaméterezése nem javíthat-e a becslés minőségén.

Ebben a részben két hosszabb és egy rövidebb témával foglalkozunk.

A monoton regresszió (3.3) az összes lehetséges monoton függvény szerinti regresszió vonatkozásában egyfajta minimális modell. Egy olyan modell, ami a lehető legkevesebb külső információt visz az adatok értékelésébe, feltételezve, hogy csak annyit tudunk az $f(x, \theta)$ függvényről, hogy az az x-ben monoton.

Az általánosított lineáris regresszió (3.4) tulajdonképpen csak annyival tér el a klasszikus lineáris regressziótól, hogy a magyarázó változó egy lineáris függvénye nem a célváltozó értékét — adott körülmények közti várható értékét — hanem annak egy esetlegesen

paraméterektől is függő függvényét közelíti. Ez látszólag kismértékű változtatás ámde jelentős módosulás az regresszió eredményének értelmezésekor.

Elsőként az általános nem-lináris regresszió módszert ismertetjük (3.2). Részletesen bemutatva azokat a beépített $f(x, \theta)$ függvényeket, amik a különböző fizikai, kémiai, biológiai alkalmazások során természetes módon adódnak mint regresszió függvények, a vizsgált rendszerek dinamikája alapján.

3.2. Általános nem-lineáris regresszió

Előbb röviden leírjuk a nem-lineáris regresszió matematikai modelljét. Majd megmutatjuk milyen eszközöket találhat az, aki az **R** programmal akar nem-lineáris modelleket illeszteni. Végezetül néhány példán megmutatjuk, hogyan lehet az **R** eszközeit nem-lineáris modellek illesztésére felhasználni.

3.2.1. A nem-lineáris regresszió matematikai leirása

Ha feltételezzük, hogy az adatok a korábban már felírt $y_j = f(x_j, \theta) + e_j$ modell szerintiek, ahol a $j = 1, \ldots, n$-re az e_j a független $\mathcal{N}(0, \sigma)$ eloszlású ε_j, $j = 1, \ldots, n$ sorozat megfigyelt értékei, akkor a minta likelihood függvénye a

$$L(\theta, \sigma, y, x) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\sum_{j=1}^{n}(y_j - f(\theta, x_j))^2}{2\sigma^2}\right)$$

formulával irható fel. Ez pont akkor maximális ha a kitevőbeli

$$S(\theta, y, x) = \sum_{j=1}^{n}(y_j - f(\theta, x_j))^2$$

minimális. Azaz, ha ennek a θ komponensei szerint vett

$$\frac{\partial S(\theta, y)}{\partial \theta_i} = 2\sum_{j=1}^{n}(y_j - f(\theta, x_j))\frac{\partial f(\theta, y)}{\partial \theta_i}$$

parciális deriváltjai nullák. Ezeknek az egyenleteknek általában nincs explicit megoldásuk. Ezért a $\hat{\theta}$ becslés elkészitéséhez általában valamilyen numerikus módszert alkalmaznak. A kapott becslések szórását pedig a regresszió függvény lokális linearizálásán alapuló

$$\widehat{\mathrm{cov}}(\hat{\theta}) = s^2(F^T F)^{-1}$$

képlettel közelítik, ahol $F_{j,\ell} = \partial f(\hat{\theta}, y_j)/\partial\hat{\theta}_\ell$, és az s az ε szórásának egy becslése. [31]

3.2.2. A nem-lineáris regresszió R -beli technikája

Azt mutatjuk be milyen kényelmi eszközöket és akadályokat talál, aki nem-lineáris modellt akar illeszteni a **R**-project 'stats' és 'MASS' csomagjának programjaival.

A regresszió függvény, a gradiens és a kezdőérték

Vegyük a 'car' csomag [10] 'US.pop' adatsorát. A 'car' csomagot előzőleg installálni kell. Ez egy 21 soros, két oszlopos adathalmaz. Az USA lakosainak száma 10 éves időközönként mérve, 1790 és 1990 közt. Emeljük ki belőle a 'time' év és a 'pop' népességszám adatokat a formulák rövidítése érdekében. Rajzoljuk ki az adatokat. Illesszünk rá az adatokra 'nls()' eljárással a

$$pop \sim \frac{\beta_1}{1 + \exp(\beta_2 + \beta_3 \cdot time)} \tag{3.1}$$

függvényt. Ez a függvény az ún. logisztikus populació növekedési görbe. Úgy adódik, hogy zárt populációt feltételezve a populációnövekményt a populáció számosságával nem lineárisan arányosnak, hanem a popoláció számosság másodfokú polinomjával arányosnak vesszük.

Nézzük meg az eredményváltozóban található adatokat, és rajzoljuk hozzá a feldolgozott adatok képéhez az előbbi függvény illesztett változatát!

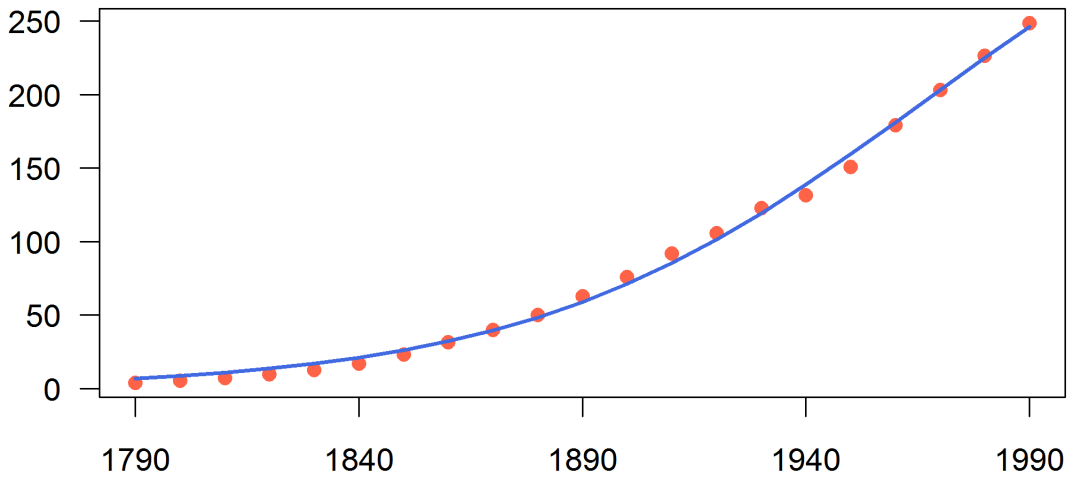
Azaz futtassuk le az alábbi utastásokat:

```
data(US.pop,package='car')
year<-US.pop$year
pop<-US.pop$population
plot(year, pop)
time <- 0:20
M<-nls(pop~b1/(1+exp(b2+b3*time)),
   start=list(b1=350,b2=4.5,b3=-.3),trace=TRUE)
summary(M)
lines(year, fitted.values(M), lwd=2)
```

A 'summary()' eredményének lényegi része:

```
     Estimate Std. Error t value Pr(>|t|)
b1 389.16551   30.81197   12.63 2.20e-10 ***
b2   3.99035    0.07032   56.74  < 2e-16 ***
b3  -0.22662    0.01086  -20.87 4.60e-14 ***
```

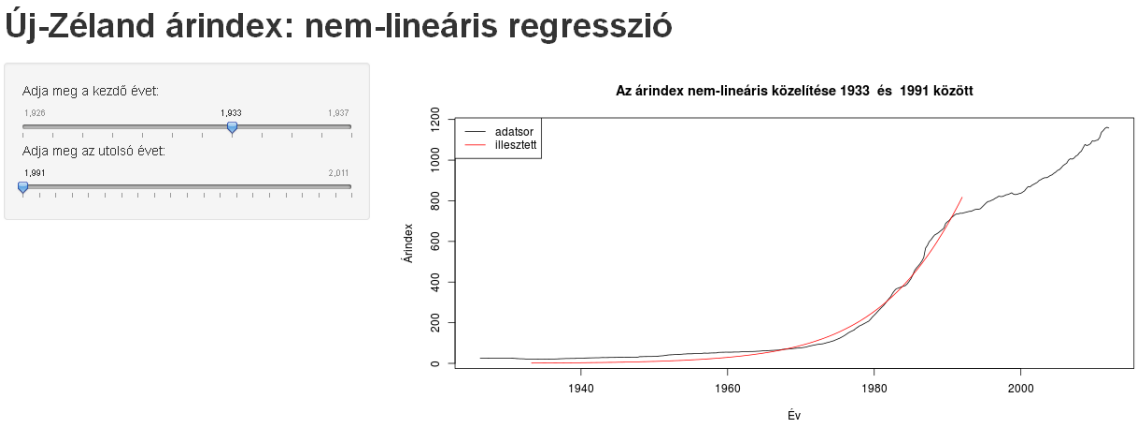
Azaz a fenti modellt $\beta_1 = 389.16$, $\beta_1 = 3.99$ és $\beta_1 = -0.22$ értékekkel illesztette, és a t-statisztika szerint úgy találta, hogy mindhárom szignifikánsan eltér a nullától.



3.1. ábra. A népességszám alakulásának modellje (1790-1990)

Hasonló adatsorra animációt is készítettünk, amely a http://hpz400.cs.elte.hu:3838/ZA_nemlin/ címen található. Itt Új-Zéland éves árindexének 1926-2011 közötti adatait mutatjuk be, és különböző intervallumokra vizsgálhatjuk, hogy a (3.1) függvény illesztése milyen eredményt ad. A 3.2 ábra azt mutatja, hogy ha csak 1991-ig tekintjük az adatokat, akkor értelemszerűen nem tudjuk előrejelezni az ezután bekövetkező infláció-csökkenést.
A következőkben megmutatjuk, hogy illesztett függvényként megadható egy olyan függvény is, aminek az értéke olyan, hogy egy attribútuma — a 'gradient' argumentum — maga a gradiens függvény. Ezt, az itt most 'kézzel' kiszámolt deriváltat az 'nls()' függvény az illesztéskor felhasználja.

```
mfv <- function(b1, b2, b3, ido)
     {sv <- exp(b2 + b3*ido) # ez egy segédváltozó
      fv <- b1/(1 + sv)
      gr <- cbind((1+sv)^-1,
                  -b1*(1+sv)^-2*sv,
                  -b1*(1+sv)^-2*sv*ido)
      attr(fv,'gradient')<-gr # az érték egy attribútuma a gradiens
      return(fv)}
```



3.2. ábra. Animációs ábra az Új-Zéland árindexére illesztett logisztikus populáció növekedési görbéről

```
summary(nls(pop~mfv(be1,be2,be3,time),
          start=list(be1=350,be2=4.5,be3=-0.3)))
```

A következő programrészlet azt mutatja, hogy nem szükséges a gradienst kiszámolni. A szükséges formula elkészíttethető a 'deriv()' deriváló szubrutin segitségével is...

```
f<-formula('~ b1/(1 + exp(b2 + b3*ido))')# a modell jobb oldala
afv <- deriv(f,c('b1', 'b2', 'b3'),
             function(b1, b2, b3, ido){})
afv
summary(nls(pop~afv(b1,b2,b3,time),start=list(b1=350,b2=4.5,b3=-.3)))
```

A harmadik parancs eredményén látható, hogy a deriválással összeállitott modell — ami egyébként egy 'function' osztályú változó —, a következő:

```
function (b1, b2, b3, ido)
  { .expr3 <- exp(b2 + b3 * ido)
    .expr4 <- 1 + .expr3
    .expr8 <- .expr4^2
    .value <- b1/.expr4
    .grad<-array(0,c(length(.value),3L),list(NULL,c("b1","b2", "b3")))
    .grad[, "b1"] <- 1/.expr4
    .grad[, "b2"] <- -(b1 * .expr3/.expr8)
    .grad[, "b3"] <- -(b1 * (.expr3 * ido)/.expr8)
    attr(.value, "gradient") <- .grad
    .value        }
```

Csak kicsit bonyolultabb, mint amit korábban kézzel megadtunk...

A 'stats' csomag előre definiált nem-lineáris regresszió függvényei

Nem-lineáris, 'selfStart' osztályú modellek az **R**-project 'stats' csomagjában.

A nem-lineáris regresszió két kényes mellékinformációja a kezdőérték és az optimalizálási tartomány. Azaz az a paraméterérték, amiből a megoldáskeresés indul, és azok a paraméterértékek, amiket mint lehetséges optimum pontokat elfogadunk. Mindkettő kritikus, mert — figyelembe véve, hogy egy numerikus optimalizálás csak korlátozott mértékben találhat globális optimumot — a hatékonyságot, az eredményt és az eredményességet is befolyásolhatja.

A kezdőérték problémának a következőkben bemutatásra kerülő 'selfStart' függvények jó segítői. Az optimalizálási tartománnyal nehezebb a helyzet. Csak akkor van lehetőségünk ilyen tartomány megadására, ha egy un. PORT rutint alkalmazunk, de ez a tartomány akkor is legfeljebb egy téglatest lehet. A megfelelő PORT rutin az

```
algorithm="port"
```

opcióval érhető el, de ez a rutin viszont nem dolgozik együtt a 'selfStart' szerinti kezdőértékkel. A PORT rutin egyébként egy kutatási célokra szabad eljárás gyüjtemény. Neve a 'Portable, Outstanding, Reliable, and Tested' rövidítése.

A 'selfStart' osztályú modellek olyan előre definiált modellek, amelyeket az 'nls()' eljárás mint formulát elfogad. Egy 'selfStart' osztályú modell tartalmaz egy olyan függvényt, amely a paraméter optimalizáláshoz megfelelő kezdőértéket szolgáltat. Továbbá lehetőséget ad a függvényérték attributumaként az iterációs lépés meghatározásához felhasználható gradiens megadására is. 'selfStart' osztályú modelleket saját magunk is definiálhatunk (lásd: 3.2.2).

A 'stats' csomagban található 'selfStart' osztályú modellek a következők:

```
SSasymp         Asymptotic Regression Model
SSasympOff      Asymptotic Regression Model with an Offset
SSasympOrig     Asymptotic Regression Model through the Origin
SSbiexp         Biexponential model
SSfol           First-order Compartment Model
SSfpl           Four-parameter Logistic Model
SSgompertz      SSgompertz(x, Asym, b2, b3)
```

```
SSlogis         Logistic Model
SSmicmen        Michaelis-Menten Model
SSweibull       Weibull growth curve model
```

Mint látható, minden a 'stats' csomagban definiált 'selfStart' modell neve 'SS'-el kezdődik. Röviden ismertetjük ezeket a modelleket. De a függvényeknek nem az összes lehetséges, hanem csak a tipikus paraméterérték melletti viselkedését elemezzük.

A 'selfStart' osztályú modellek az argumentum (a leírásokban 'input') és a paraméterek megadása mellett úgy működnek, mint a közönséges függvények. Ha viszont a paramétereket nem közvetlenül egy-egy számértéket beadva, hanem egy-egy változó segítségével adjuk meg, akkor a függvényérték attribútumaként megkapjuk a megfelelő pontban az adott paraméterű függvény gradiensét is.

Az 'SSasymp' (Asymptotic Regression Model), azaz az aszimptotikus regresszió modell képlete:

$$f(x) = \alpha + (\beta - \alpha)e^{-\varrho x},$$

hívása:

`SSasymp(x, Asym, R0, lrc)}`

ahol $Asym = \alpha$, $R0 = \beta$ és $lrc = \ln(\varrho)$.

A paraméterek értelmezése. A függvény induló értéke az $x = 0$ mellett a $\beta = R0$. A függvény határértéke $x = \infty$ esetén az $\alpha =$ `Asym` aszimptotikus érték. Az $\ln(\varrho) =$ `lrc` konstans a (növekedési/csökkenési) ráta. A függvény értéke nem-negatív x-ekre monoton változik. Az értéke a 0-ban a β-ból ('R0') indul és $+\infty$-ben *exponenciálisan* az α-hoz ('Asym') simul (példaként lásd a 3.3 ábrát!).

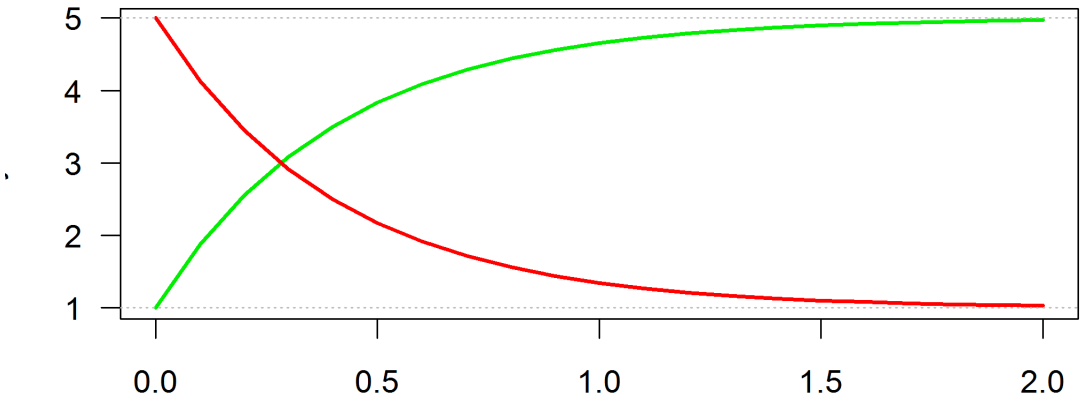
Az 'SSasympOrig' (Asymptotic Regression Model through the Origin), azaz az origón átmenő aszimptotikus regresszió modell
képlete:

$$f(x) = \alpha(1 - e^{-\varrho x}),$$

hívása:

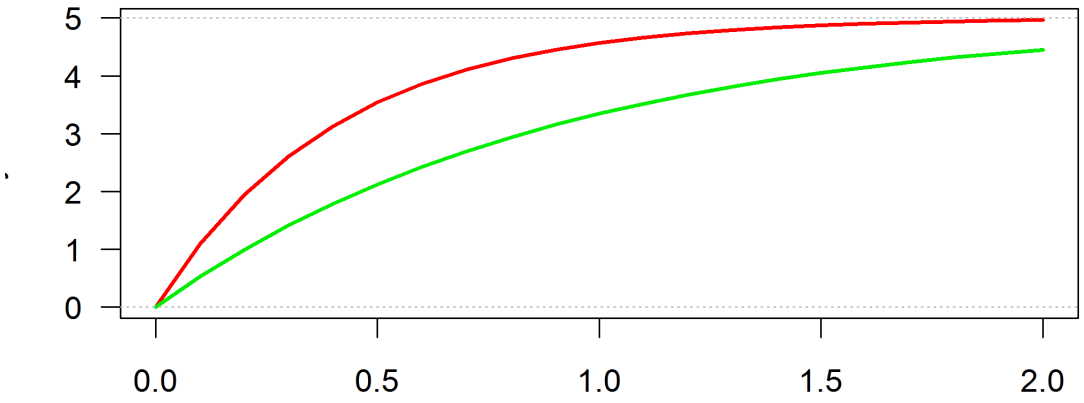
`SSasympOrig(x, Asym, lrc)}`

ahol $Asym = \alpha$ és $lrc = \ln(\varrho)$.



3.3. ábra. 'SSasymp()' aszimptotikus regresszió modell függvénye $\varrho = .9$ mellett, növekvő ($\beta = 1 < \alpha = 5$: zöld) és csökkenő ($\beta = 5 > \alpha = 1$: piros) esetben

A paraméterek értelmezése. Mivel az 'SSasymp()' függvényhez viszonyítva az $R0 = \beta$ paraméter hiánya a $\beta = 0$-nak felel meg, annyiban különbözik az 'SSasymp()' függvénytől, hogy ennek a függvénynek értéke az $x = 0$-ban fixen 0. A 3.4 ábra az 'SSasympOrig()' függvényt $\alpha = Asym = 5$ és két különböző ϱ paraméterérték mellett mutatja. Ha az α negatív volna, akkor persze monoton csökkenő függvényt kapnánk.



3.4. ábra. 'SSasympOrig()' origón átmenő aszimptotikus regresszió modell függvénye, $\varrho = .9$ (piros) és $\varrho = .1$ (zöld) mellett

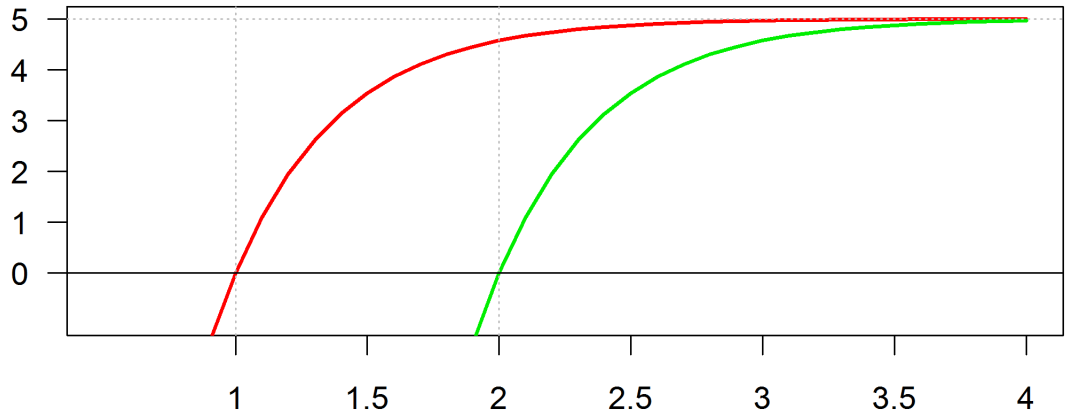
Az 'SSasympOff' (Asymptotic Regression Model with an Offset), azaz az aszimptotikus regresszió modell konstans eltolás mellett
képlete:

$$f(x) = \alpha(1 - e^{-\varrho(x-c_0)}),$$

hívása:

```
SSasympOff(x, Asym, lrc, c0)}
```

ahol $\alpha = Asym$, $\ln(\varrho) = lrc$ és $c_0 = C0$.
A paraméterek értelmezése. Annyiban különbözik az 'SSasympOff' függvénytől, hogy ez megengedi az illesztett függvény x-tengely menti, c_0 paraméterértékkel való eltolását. Vagyis ennek a modellnek az 'SSasympOrig' a $c_0 = 0$-nak megfelelő speciális esete. Az 'SSasympOff' esetén a $c_0 =$ `C0` az az x érték, amire a függvény nulla.



3.5. ábra. 'SSasympOff()' aszimptotikus regresszió nem feltétlen 0 átmetszési ponttal, $c_0 = 2$ (zöld) és $c_0 = 1$ (piros) átmetszés (offset) mellett

Az 'SSbiexp' (Biexponential model) modell két exponenciális függvény lineáris kombinációja,
képlete:

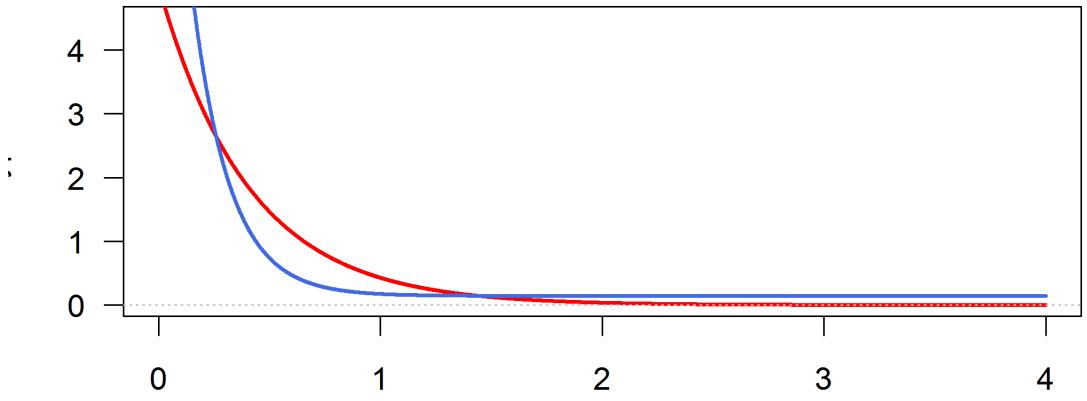
$$f(x) = \alpha_1 e^{-\varrho_1 x} + \alpha_2 e^{-\varrho_2 x},$$

hívása:

```
SSbiexp(input, A1, lrc1, A2, lrc2)
```

A paraméterek értelmezése. Az $\alpha_1 = A1$ és az $\alpha_2 = A2$ a kezdeti mennyiségek, és a $\ln(\varrho_1) =$ `lrc1` és a $\ln(\varrho_2) =$ `lrc2` a változási sebességek.

Ilyen modellre van szükség, amikor például egy fogyókúra esetén a testtömeget vizsgáljuk az idő múlása függvényében. Ugyanis a megváltozott táplálkozás hatására az eredeti α_1 testzsír tömeg és a test α_2 sovány tömege egyaránt változik, ám a kettő két különböző arányban (ϱ_1 és ϱ_2). Az (3.6) ábra azt mutatja, hogyha egy adatsor valójában biexponenciális akkor az adatok sima exponenciális közelítése akár igen durva is lehet.



3.6. ábra. 'SSbiexp()' biexponenciális görbe (piros), és az őt legjobban közelítő, aszimptotikus regresszió modell ('SSasymp', kék)

Az 'SSfol' (First-order Compartment Model) modell az elsőrendű kamramodell, képlete:

$$f(x) = D\frac{K_e K_a}{C_l(K_a - K_e)}(\exp(-K_e x) - \exp(-K_a x)),$$

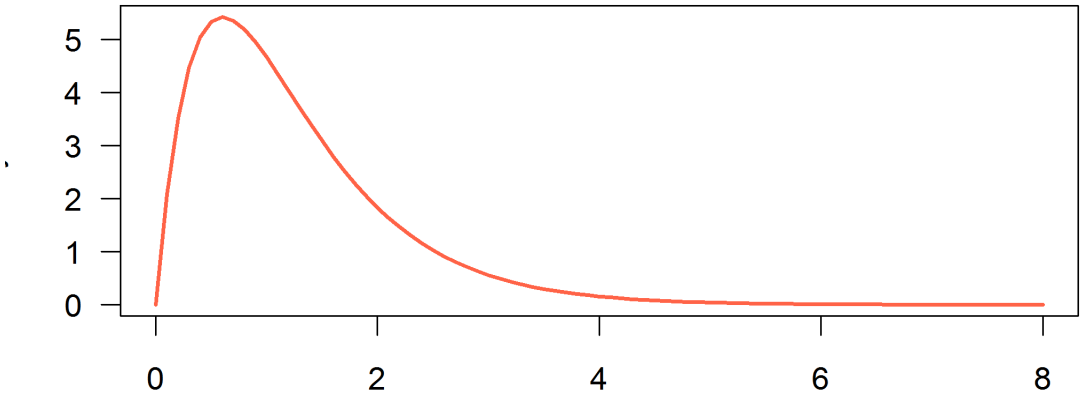
hívása:

```
SSfol(Dose, x, lKe, lKa, lCl)
```

ahol $Dose = D$, $lKe = \ln(K_e)$, $lKa = \ln(K_a) =$, $lC = \ln(C_l)$.

A paraméterek értelmezése. A $D = Dose$ a kezdeti mennyiség $K_e = \exp(lKe)$ az eliminációs, azaz kiválási ráta, $K_a = \exp(lKa)$ az abszorpciós, azaz elnyelési ráta a $K_\ell = \exp(lCl)$ pedig clearance, azaz a tisztulási ráta. A modell a nevét a különösen a kémiában gyakran alkalmazott kamramodellekről kapta. A kamra- vagy cellamodell feltételezése szerint ugyanaz az anyag, egy vagy több elkülönült helyen több különböző

koncentrációban van jelen. A cellák ugyanakkor kapcsolatban vannak egymással. A rendszer úgy viselkedik mint egy egyszerű dinamikus rendszerben. Az idő múltával, a megfelelő törvényszerűségek szerint a koncentráció kiegyenlítődik, az anyag esetleg veszejtődik.



3.7. ábra. SSfol() (First-order Compartment Model) elsőrendű kamramodell, a $Dose = 10$, $\ell K_e = .3$, $\ell K_a = .7$, $\ell C_l = .1$ paraméterekkel

Az 'SSlogis' (Logistic Model), azaz a (három paraméteres) logisztikus modell, képlete:

$$f(x) = \frac{\alpha}{1 + \exp(\frac{m-x}{s})},$$

hívása:

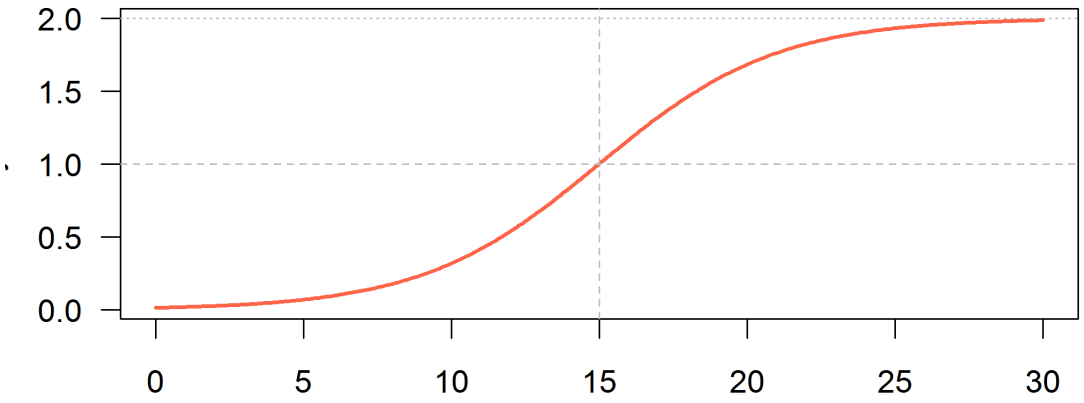
```
SSlogis(x, Asym, xmid, scal)
```

A paraméterek értelmezése. Az m a középérték, az s a skála érték. A függvény monoton növekedő. A határértéke a $+\infty$-ben $\alpha = Asym$. A görbe értéke az $m = xmid$ pontban $\alpha/2 = Asym/2$, és az $(m, \alpha/2) = (xmid, Asym)$ pontra szimmetrikus. A görbe egy szimmetrikus S-görbe, aminek az értéke a $[0, \infty)$ intervallumon a 0-ból az α-ba tart.

Az 'SSfpl' (Four-parameter Logistic Model), azaz a négyparaméteres logisztikus modell, képlete:

$$f(x) = \alpha + \frac{\beta - \alpha}{1 + \exp(\frac{m-x}{s})},$$

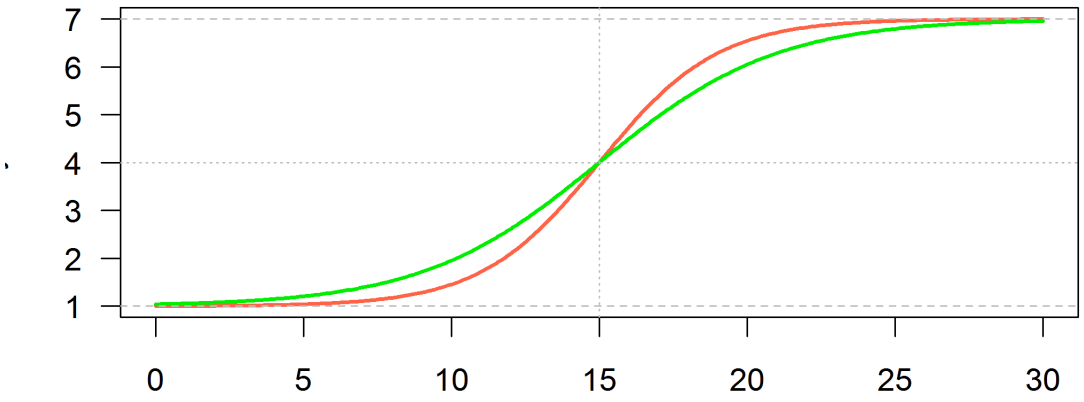
hívása:



3.8. ábra. 'SSlogis()' három paraméteres logisztikus modell, S-görbe

`SSfpl(input, A, B, xmid, scal)`

ahol $A = \alpha$, $B = \beta$, $xmid = m$, $scal = s$.
A paraméterek értelmezése. Erre a modellre lényegében ugyanaz érvényes mint a három paraméteres logisztikus modellre. Azzal a különbséggel, hogy az értéke nem 0-tól, hanem az $\alpha = A$ értéktől indul és a $B = \beta$ értékhez tart. A görbe szimmetria középpontja ennek megfelelően $(m, (\alpha + \beta)/2) = (mid, (A + B)/2)$



3.9. ábra. 'SSfpl()' négy paraméteres logisztikus modell, S-görbe.
$\alpha = 1$, $\beta = 7$ $m = 15$ a piros görbe esetén $s = 2$, a zöldre $s = 3$

Az 'SSgompertz' (Gompertz Growth Model), azaz a Gompertz féle növekedési modell,

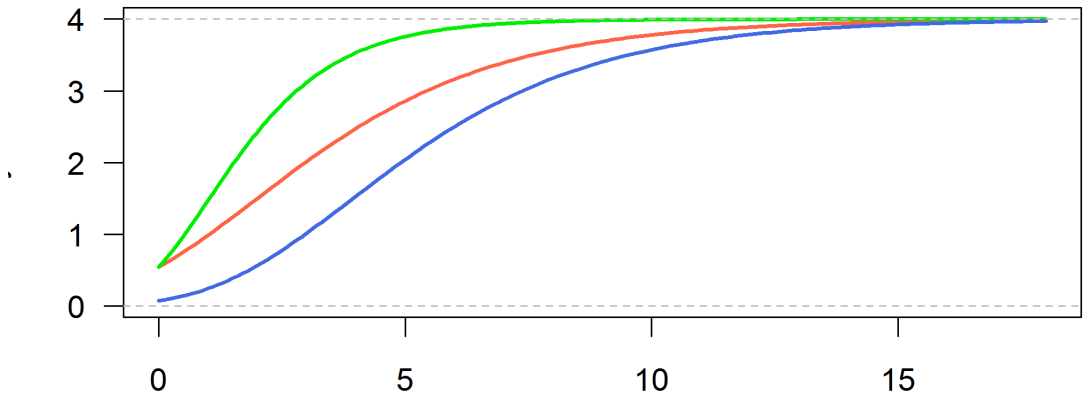
képlete:

$$f(x) = \alpha \exp(-\beta_2 {\beta_3}^x),$$

hívása:

```
SSgompertz(x, Asym, b2, b3)
```

Az α adja meg a függvény határértékét $x = \infty$-ben. A β_2 paraméter a függvény $x = 0$-beli értékét befolyásolja, a β_3 pedig az x-tengely skálázását.



3.10. ábra. 'SSgompertz()' görbe $\alpha = 4$ mellett a piros görbére $\beta_2 = 2$, $\beta_3 = .7$, a zöldre: $\beta_2 = 2$, $\beta_3 = .5$, a kékre: $\beta_2 = 4$, $\beta_3 = .7$

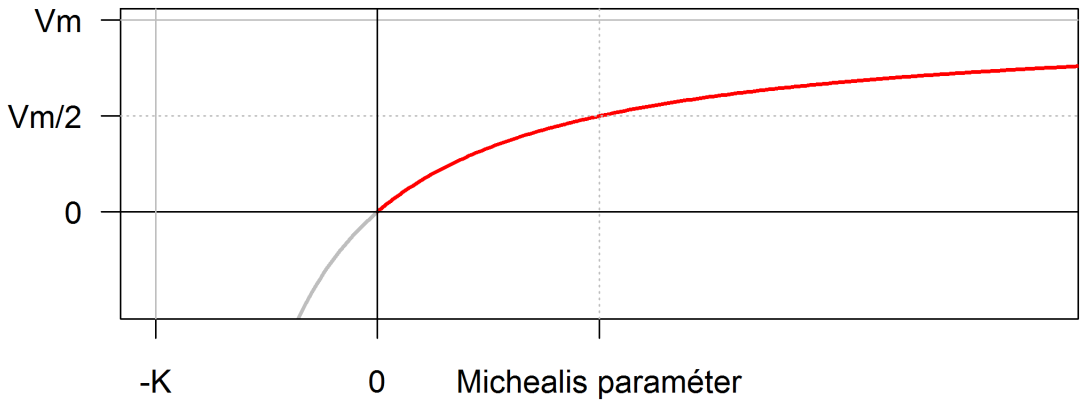
Az 'SSmicmen' (Michaelis-Menten Model), azaz a Michaelis-Menten modell, képlete:

$$f(x) = V_m \frac{x}{K + x},$$

hívása:

```
SSmicmen(x, Vm, K)
```

ahol $Vm = V_m$ és $K = K$.
A Michaelis-Menten modell illesztése ténylegesen egy hiperbola illesztését jelenti. Ez a függvény (V_m pozitív értéke mellett) a $-K$ és a végtelen közt monoton növekedő, ahogyan azt a 3.11 ábrán is láthatjuk. A plusz végtelenben a határértéke (szuprémuma) V_m úgy, hogy a $V_m/2$ értéket az $x = K$ pontban veszi fel. A Michaelis paraméternek is nevezett K paraméter ez utóbbi értelmezésének az enzim kinetikában van jelentősége. Ez a görbe viszonylag lassan, az $1/x$-nek megfelelő *polinomiális* sebességgel simul hozzá az aszimptotikus értékéhez.



3.11. ábra. 'SSmicmen()' Michaelis-Menten görbe a $K = 7$, $V_m = 10$ paraméter értékekre

Az 'SSweibull' (Weibull growth curve model), azaz a Weibull eloszlásból származó modell, képlete:

$$f(x) = \alpha - \delta \exp(-\varrho x^p),$$

hívása:

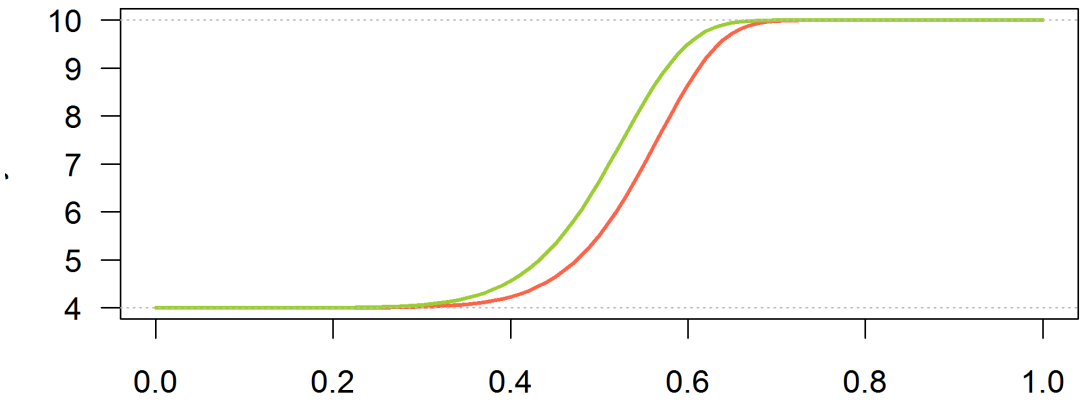
```
SSweibull(x, Asym, Drop, lrc, pwr)
```

A paraméterek értelmezése. Ez a modell az 'SSasymp' általánosítása, ott ugyanis a $p = pwr$ rögzített értéke 1. Az $\alpha =$ `Asym` a függvény maximuma, a $\delta =$ `Drop` a függvény értékének skála függvénye, $\ln(\varrho) =$ `lrc` pedig a növekedési ráta logaritmusa. Pontosabban. A függvény értéke az $\alpha - \delta$ értéktől monoton növekedő az α szintig. Ha a $p > 1$ akkor a görbe formája egy aszimmetrikus S (3.12).

A 'selfStart' regresszió függvények definiálása

A 'selfStart' osztályú objektumok konstruktora a négy paraméteres 'selfStart()' eljárás. A 'selfStart()' első paramétere az a kifjezés, ami a függvény értékét kiszámolja. A második egy eljárás, ami szükség esetén a kezdőértéket kiszámolja. A harmadik adja meg, hogy melyek az első két paraméterként megadott kifejezésekben az alkalmilag illesztendő paraméterek. Az utolsó (ez belső hívásokkor van felhasználva) egy minta arra vonatkozóan, hogy az adott modellt hogyan kell meghívni.

Az alábbiakban annak a mintának a szerkesztett változatát mutatjuk, ami a 'selfStart()' utasítás beépített help-lapján található és a '?selfStart()' paranccsal érhető el.



3.12. ábra. 'SSweibull()' Weibull eloszlásból származó aszimetrikus S görbe $\alpha = 10$, $\delta = 6$, $\varrho = 5$ és $p = 9$ (piros), $p = 8$ (zöld)

Az utasítás egy az 'SSlogis'-hoz hasonló objektumot hoz létre, azzal a különbséggel, hogy a 'SSlogis' beépitett 'selfStart' eljárásban szereplő függvény függvény-értékének van gradiens attributuma is.

```
SSsajat <-
  selfStart(
 ~ Asym/(1 + exp((xmid - x)/scal)),
  function(mCall, data, LHS)
  { xy <- sortedXyData(mCall[["x"]], LHS, data)
    if(nrow(xy) < 4) { stop("Too few distinct x values")   }
    z <- xy[["y"]]
    if (min(z) <= 0) { z <- z + 0.05 * max(z) } # avoid zeroes
    z <- z/(1.05 * max(z))              # scale to within unit height
    xy[["z"]] <- log(z/(1 - z))         # logit transformation
    aux <- coef(lm(x ~ z, xy))
    parameters(xy) <- list(xmid = aux[1], scal = aux[2])
    pars <- as.vector(coef(nls(y ~ 1/(1 + exp((xmid - x)/scal)),
                               data = xy, algorithm = "plinear")))
    value <- c(pars[3], pars[1], pars[2])
    names(value) <- mCall[c("Asym", "xmid", "scal")]
    return(value) },
 c("Asym", "xmid", "scal"))
```

Látható, hogy ∼ Asym/(1 + exp((xmid - x)/scal)) az illesztendő modell megadott képlete, a

`c("Asym", "xmid", "scal")`

paraméter lista mellett. A leghosszabb rész a kezdőértéket meghatározó

`function(mCall, data, LHS)\{\}`

`algorithm = "plinear"`

paraméterezéssel. Ez egy nem túl hatékony, ám általában valamiféle eredményt adó parciális lineáris módszere a nem-lineáris regressziónak.

3.2.3. A nem-lineáris regresszió a gyakorlatban

A becsült paraméterek megbízhatósága

Végezetül három technikát mutatunk be, amivel az illesztett modell megbízhatósága szemléltethető. Az első a konfidencia tartománynak a likelihood függvényen alapuló meghatározása. A második a megoldáspont közelében a megoldásfelszín és a paramétervonalak görbületein alapuló becslés értékelés. A harmadik eszköz az **R**-project 'nlstools' kiegészítő csomagjában található. Az ottani eljárások segítségével jackknife és bootstrap módszereket alkalmazhatunk a becsült paraméterek megbízhatóságának ellenőrzésére és növelésére.

A becslések konfidenciatartománya

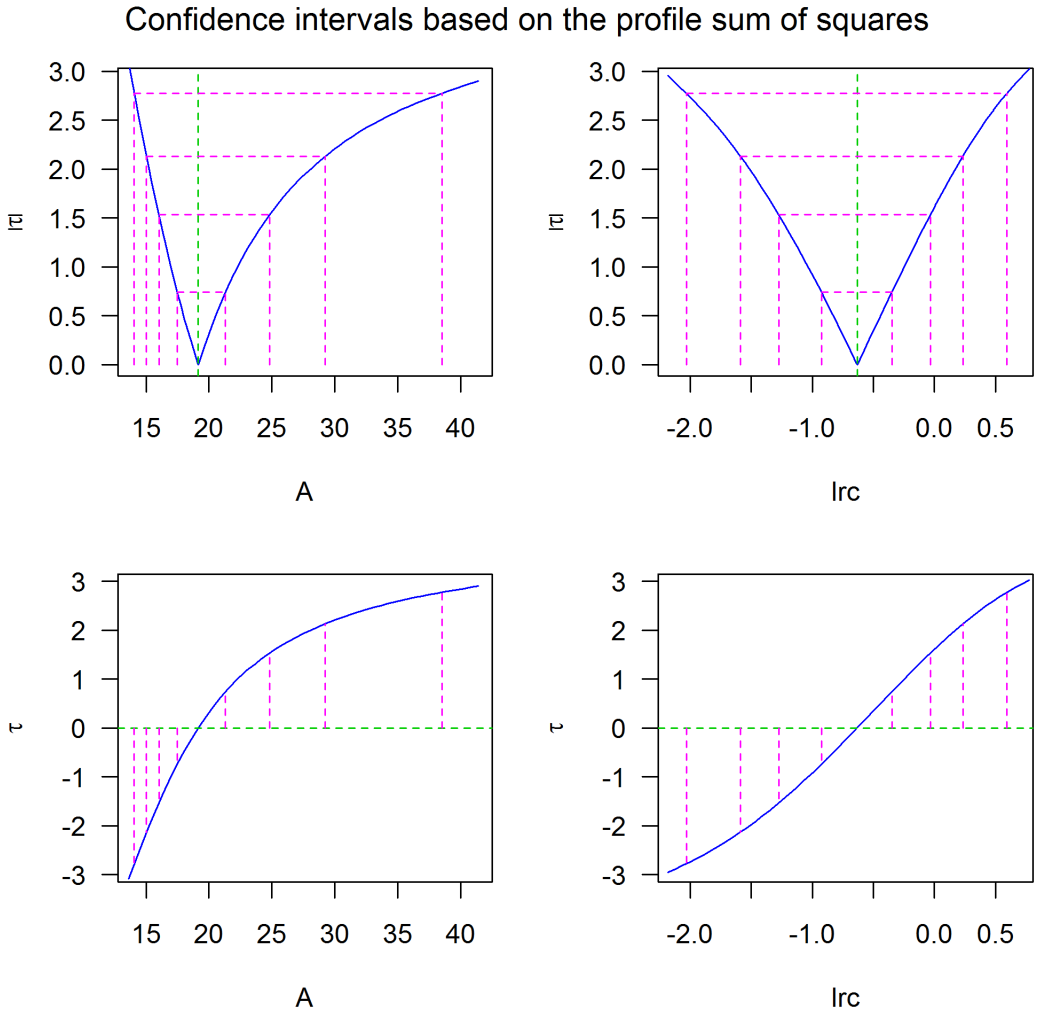
A 'profile()' függvény egy likelihood maximalizálással nyert modell esetén veszi a becsült paraméterek konfidenciatartományát a log-likelihood függvény profiljai alapján.

Ha lefuttatjuk az alábbi programrészletet, akkor a végeredményként a 3.13 ábrát nyerjük.

```
M <- nls(demand ~ SSasympOrig(Time, A, lrc), data = BOD)
pr <- profile(M, alpha = 0.05)
coef(M)
par(mfrow=c(2,2))
plot(pr, conf = c(95, 90, 80, 50)/100)
plot(pr, conf = c(95, 90, 80, 50)/100, absVal = FALSE)
```

A programsorok a 'datasets' csomagban található, mindössze 6 adatsort tartalmazó 'BOD' adathalmazt dolgozzák fel. A 'BOD' (Biochemical Oxygen Demand) adathalmaz egy vízminta, kezelés közbeni oxigén igényét mutatja naponkénti időközökben mg/l mértékegységben megadva. Ennek az adathalmaznak két adatoszlopa van: a 'demand' és a 'Time'. A célváltozó az oxigén igény, a magyarázó változó az idő. Az illesztett modell a 3.4 ábrán is bemutatott 'SSasympOrig', origón átmenő aszimptotikus regresszió modell.

A következő 'profile' parancs a konfidencia tartomány meghatározásához szükséges 'pr' segédváltozót állítja elő. A keletkezett négy ábrán azt láthatjuk, hogy a modell 'A=19.14' aszimptotikus értékének és 'lrc=-0.63' logaritmikus változási sebességének mik a τ érték alapján számolt 95, 90, 80, 50 %-os konfidenciatartományai.



3.13. ábra. A becsült paraméterek konfidenciatartományai

A megoldáspont körüli görbületek

Egy nem-lineáris modell becslési pont körüli nem-linearitása két komponensre bontható. Az egyik komponens a modellnek mint egy a megfigyelési térben elhelyezkedő, a megfigyelési térnél alacsonyabb dimenziós felszínnek a görbülete a becslés körül. Ez független a modell konkrét paraméterezésétől. A másik komponens pedig a modell felszínén futó paraméter-görbe görbülete. Mindkettő befolyásolja a becslések megbízhatóságát és annak egymástól való függését.

Futtassuk le az alábbi mintaprogramot

```
adat<-Puromycin[Puromycin$state == "treated", ]
mmg<-deriv3(~Vm*conc/(K+conc),c("Vm","K"),function(Vm,K,conc)NULL)
(M<-nls(rate~mmg(Vm,K,conc),data= adat,start=list(Vm=200,K=.1)))

MASS::rms.curv(M)
```

A 'Puromycin' adathalmaz egy 23 soros 3 oszlopos adathalmaz amiben a harmadik ('state') oszlop azt mutatja, hogy az első ('conc') oszlopban adott koncetráció és a második ('rate') oszlopban adott enzimreakció sebességet kezelt ('treated'), vagy kezeletlen ('untreated') anyagon mérték. Az első parancs az 'adat' változóba menti az adathalmaz kezelt esetekre vonatkozó 12 soros részét. A következő parancs a 'deriv3()' eljárást felhasználva kiszámítja a 3.11 ábrán is bemutatott Michaelis-Menten görbe szimbolikus deriváltját. A harmadik parancs megfelelő kezdőértékek mellett a modellt illeszti. Végül az utolsó parancs meghívja az illesztett modellre — az alapértelmezés szerint nem betöltött 'MASS' alapcsomagból — az 'rms.curv()' programot. Ennek eredménye:

```
Parameter effects: c^theta x sqrt(F) = 0.2121
        Intrinsic: c^iota  x sqrt(F) = 0.092
```

a nem-linearitás paraméterezésből és a modellből származó két részének a mértéke.

Az 'nlstools' csomag jackknife és bootstrap eszközei

A következő programrészlet az 'nlstools' csomag [1] 'survivalcurve2' interpretációs adathalmazával számol. Ez egy 23 soros 2 oszlopos 'data.frame', ami a baktérium-sűrűség logaritmusa az idő múlásával. Az első parancs betölti az 'nlstools' kiegészítést. Nem tartozik az **R** alapkészletéhez, előzőleg installálni kell! A második pedig, egy rövidebb nevű változóba tölti a feldolgozott adatsort.

```
require(nlstools)
data(survivalcurve2);sc2<-survivalcurve2
mafart#Weibull model as parameterized by Mafart et al.
preview(mafart,sc2,start=list(p=1,delta= 1,LOG10N0=7  ))
preview(mafart,sc2,start=list(p=1,delta=10,LOG10N0=7  ))
```

```
preview(mafart,sc2,start=list(p=2,delta=10,LOG10N0=7.5))

M<-nls(mafart,sc2,list(p=2,delta=10,LOG10N0=7.5))
plotfit(M,smooth=TRUE)
overview(M)
rM <- nlsResiduals(M)
plot(rM)

cM <- nlsContourRSS(M)
plot(cM, add.col = F, nlev = 10)

jM <- nlsJack(M)
summary(jM)
plot(jM)

bsM <- nlsBoot(M, niter = 2000)
summary(bsM)
```

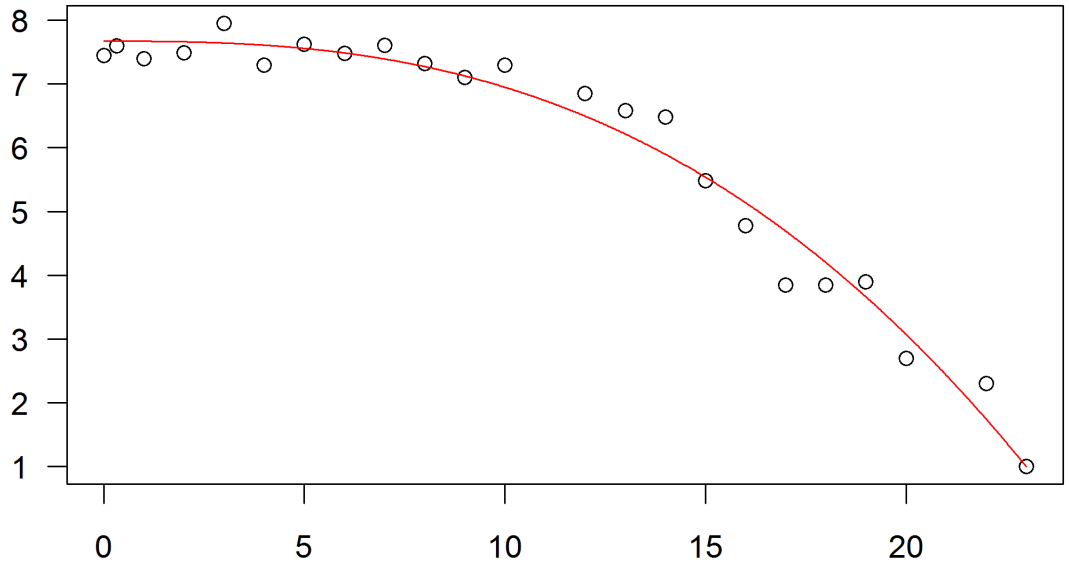
A három egymásutáni 'preview()' parancs egy-egy ábrát készít, az elemzendő adatokkal és a 'mafart' változóban megadott modell 'start' szerinti paraméterezésével. Ezek az ábrák, ez az utastás arra alkalmas, hogy megfelelő indulóértékeket találjunk a modell későbbi illesztéséhez.

A 'mafart' egy 'formula' osztályú változó. Ha kiíratjuk, láthatjuk, hogy lényegében egy hatvány függvény:

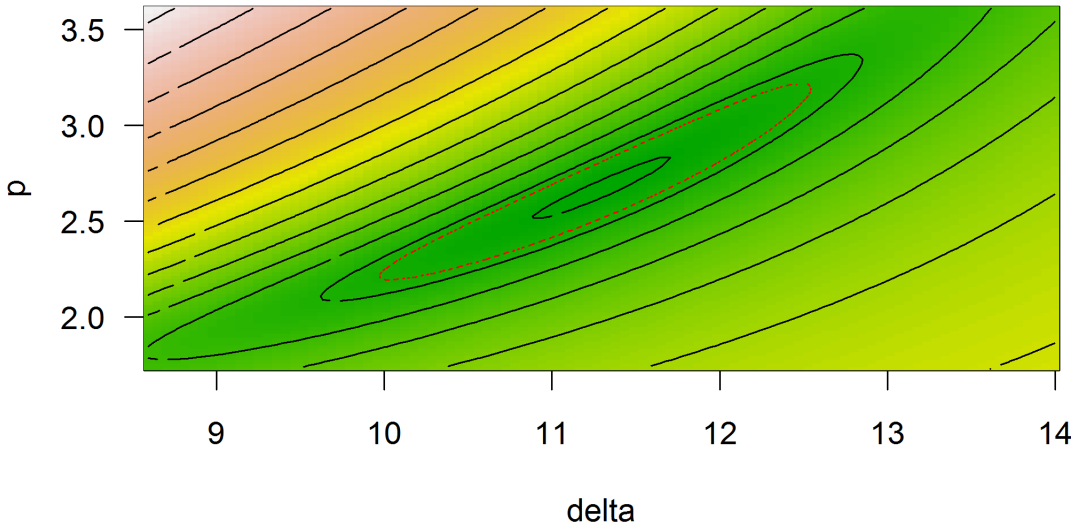
```
LOG10N $\sim$ LOG10N0 - (t/delta)\verb|^|p
```

Az illesztés eredményét az adatokkal a 'plotfit()' kirajzolja, úgy mint az a 3.14 ábrán látható.
Az 'nlsContourRSS()' a legkisebb négyzetek módszerével határozza meg a becsült paraméterek megbízhatósági tartományát. Grafikus eredményét a 3.15 ábra mutatja. Van a csomagban egy 'nlsConfRegions()' függvény is, ami a likelihood függvény alapján veszi a megbízhatósági tartományt. Érdekes összehasonlítani a kettőt!
Az 'nlsJack()' megmutatja, hogy melyik megfigyelések vannak különösen erős befolyással a paraméterbecslésekre. Az eredmények grafikusan is szemléltethetőek. Itt csak a numerikus eredményeket mutatjuk be.

```
------
Jackknife estimates
        p     delta   LOG10N0
 2.632607 11.260269  7.669289
```



3.14. ábra. A baktérium-sűrűség modellje



3.15. ábra. A p és a *delta* megbízhatósága legkisebb négyzetek módszerével véve

```
------
Jackknife confidence intervals
         Low        Up
```

```
p       2.155147  3.110067
delta   9.916903 12.603636
LOG10N0 7.497482  7.841096

------
Influential values
* Observation 15 is influential on p
* Observation 18 is influential on p
* Observation 22 is influential on p
* Observation 15 is influential on delta
* Observation 18 is influential on delta
```

Az utolsó két utasítás közül az első a 'nlsBoot()'. Ez egy 2000 elemű bootstrap minta alapján készít becslést és konfidencia intervallumot a paraméterekre.
A második, a formázott kiíratás eredménye:

```
------
Bootstrap estimates
        p     delta   LOG10N0
 2.667080 11.284033  7.676318

------
Bootstrap confidence intervals
             2.5%     97.5%
p        2.296984  3.105319
delta   10.113499 12.430319
LOG10N0  7.442106  7.909360
```

Figyelemreméltó, a korábbi becslésektől való eltérés.

3.3. Monoton regresszió

A monoton (vagy másnéven isoton) regresszió feladata a legegyszerűbb egydimenziós esetben az, hogy adott (x_k, y_k), $k = 1, \ldots, n$, valós számpárokból álló megfigyeléssorhoz olyan $m_1, \ldots, m_k$ valós konstansokat találjon, amikre a

$$\sum_{k=1}^{n} (y_k - m_k)^2$$

négyzetösszeg minimális azon $(m_1, \ldots, m_n)$ szám n-esek körében, amikre

$$m_j \leq m_\ell \text{ ha } x_j \leq x_\ell.$$

Azaz, *ha* a megfigyelések felsorolása olyan, hogy az $x_1, \ldots, x_n$ monoton növekedő, akkor a fenti négyzetösszeget azon $m_1, \ldots, m_n$ sorozatok körében kell minimalizálni, amik monoton növekedők. Vagyis az $m_1, \ldots, m_n$ sorozat tekinthető úgy mint az $y_1, \ldots, y_n$ legkisebb négyzetek módszerével vett regressziója az összes lehetséges monoton sorozat halmazára [33].

Megjegyzés. Nyilvánvaló, hogy az adott feladat megoldásakor nincsen jelentősége az $x_1, \ldots, x_n$ értékeknek, csak azok sorrendjének. Ezért az egyszerűség kedvéért a továbbiakban mindig úgy tekintjük, hogy az $y_1, \ldots, y_n$ értékeket az x-ek növekedő sorrendjében vettük és hogy az $x_1, x_2, \ldots, x_n$ értéke rendre az $1, 2, \ldots, n$.

3.3.1. A monoton regresszió algoritmusai

Tekintsük az y értékek szukcesszív értékeiből képzett $n+1$ hosszú z sorozatot, amire $z_0 = 0$, $z_1 = y_1$, $z_2 = y_1 + y_2$, $\ldots$, $z_n = \sum_{k=1}^{n} y_k$. Vegyük a koordináta rendszer $(0, z_0)$, $(1, z_1)$, $\ldots$, (n, z_n) pontjait. Tekintsük az így nyert pontok alsó konvex burkát. Vegyük az így nyert szakaszok meredekségét, azaz ha a konvex burok egy szakaszának két végpontja a j és az ℓ, akkor vegyük a

$$\frac{z_\ell - z_j}{\ell - j}$$

hányadost. Legyen $k = 1, \ldots, n$-re az m_k értéke az az előbb meghatározott meredekség, ami az x szerinti $(k-1, k)$ intervallumon érvényes.

Az így meghatározható $m_1, \ldots, m_n$ konstansokkal kapcsolatban két dolgot fogunk vázlatosan belátni. Egyrészt azt, hogy a fenti sorozat a megoldás (3.1. állítás). Másrészt azt, hogy ez a megoldás $\mathcal{O}(n)$ idő alatt megtalálható (3.2. állítás).

3.1. Állítás *Az előzőekben meghatározott $m_1, \ldots, m_n$ számsor az $y_1, \ldots, y_n$ legkisebb négyzetek módszerével vett regressziója az összes lehetséges monoton sorozatra nézve.*

Azaz a (k, z_k), $k = 0, \ldots, n$ pontok alsó konvex burka tényleg az optimális megoldást adja.

3.2. Állítás *A (k, z_k), $k = 0, \ldots, n$ pontok alsó konvex burka $\mathcal{O}(n)$ idő alatt megtalálható.*

Ebben az állításban az az érdekes, hogy egy általános ponthalmaz konvex burka $\mathcal{O}(n \log n)$ idő alatt található meg. Itt azért tudunk hamarabb végezni, mert a feltételezés szerint ismerjük a ponthalmaznak az x értékek szerinti rendezettségét.

Elöljáróban emlékeztetünk arra az elemi tényre, hogy a tetszőleges $u_1, \dots, u_k$ számokra és a tetszőleges c számra a $\sum_{j=1}^{k}(u_j - c)^2$ különbségnégyzet összeg annál nagyobb, minél távolabb van a c az u_j számok $\bar{u}$ átlagától. Ugyanis a

$$\sum_{j=1}^{k}(u_j - c)^2 = \sum_{j=1}^{k}(u_j - \bar{u})^2 + k(\bar{u} - c)^2,$$

egyenlőség, mint az n dimenziós $(u_1, \dots, u_k)$, $(\bar{u}, \dots, \bar{u})$ és $(c, \dots, c)$ pontokra érvényes Pithagorasz tétel teljesül, és a jobboldalon szereplő $k(\bar{u} - c)^2$ tag a c és a $\bar{u}$ távolsága függvényében monoton növekedő.

Könnyen látható, ha az u sorozat monoton csökkenő, azaz ha $u_1 \geq u_2 \geq \cdots \geq u_k$, akkor az u sorozatnak a legjobb L_2-beli monoton közelítése a megfigyelések $\bar{u} = \sum_{j=1}^{k} u_j/k$ átlagából képzett $\bar{u}, \bar{u}, \dots, \bar{u}$ konstans sorozat. Ám most ennél az állításnál egy olyan erősebb állítást bizonyítunk, amire minden olyan algoritmus visszavezethető ami a minimumhelyet megtalálja.

3.3. Állítás *A u_i, $i = 1, \dots, k$ sorozatot a monoton sorozatok közül akkor és csak akkor közelíti legjobban az $\bar{u}$ konstans sorozat, ha minden $j = 1, \dots, k$ -ra teljesül a*

$$\sum_{i=1}^{j} u_i/j \geq \sum_{i=1}^{k} u_i/k$$

egyenlőtlenség.

A 3.3. állítás feltétele pontosan akkor teljesül, ha a kétdimenziós koordináta rendszerben a $(j, \sum_{i=1}^{j} u_i)$ pontok $j = 1, \dots, k-1$ egyike sincs az $(0, 0)$ – $(k, \sum_{i=1}^{k} u_i)$ pontok által meghatározott egyenes alatt.

Bizonyítás. Könnyíti a bizonyítás áttekinthetőségét a következő észrevétel: ha az $u_1, \dots, u_k$ sorozatnak az $m_1, \dots, m_k$ a legjobban közelítő monoton sorozata, akkor egy tetszőleges c-re az $u_1 - c, \dots, u_k - c$ sorozatnak az $m_1 - c, \dots, m_k - c$ a legjobban közelítő monoton sorozata. Ugyanakkor a $v_i = u_i - \bar{u}$, $i = 1, \dots, k$ sorozatra a $\sum_{i=1}^{k} v_i = 0$, és az állításbelivel egyenértékű az a feltétel, hogy a $\sum_{i=1}^{j} v_i \geq 0$, $j = 1, \dots, k$ legyen.

Tehát elegendő belátni, hogy egy olyan v_i $i = 1, \dots, k$ sorozatnak amelynek az összege 0, akkor és csak akkor konstans a legjobb monoton növekvő közelítő sorozata, ha a sorozatnak minden részletösszege nemnegatív. Az egyszerűsitett állítást indirekt látjuk be.

Tegyük fel hogy van egy olyan v, nulla összegű sorozat aminek az azonosan 0 a legjobb monoton közelítése, és aminek például az ℓ. részletösszeg negatív. Ekkor azonban az

a sorozat is monoton növekedő aminek első ℓ tagja a v sorozat első ℓ tagjának átlaga, az utolsó $k-\ell$ tagja pedig a v utolsó $k-\ell$ tagjának az átlaga. Ugyanakkor ez a két konstansból álló sorozat, a négyzetösszegekről előzőleg mondottak szerint nyilván jobb közelítése a v sorozatnak mint az azonosan 0.

Tegyük fel, hogy a v sorozat minden részletösszege nemnegatív, és mégsem az azonosan 0, hanem az $a,\dots,a < b_1 \le \dots \le b_q < c,\dots,c$ sorozat az ami a v sorozat legjobb monoton közelítése. Tegyük fel, hogy ebben a legjobban közelítő sorozatban p darab a és r darab c van. Ekkor $p+q+r=k$. Ugyanakkor mert a nem-negativitási feltétel miatt a $0 \le \sum_{i=1}^{p} v_i$ és a $\sum_{i=p+q+1}^{k} v_i \le 0$ teljesül, és mert vagy az $a<0$ vagy a $0<c$ is teljesül, vagy az $a^* = \min(b_1, 0)$-re az $a^*,\dots,a^*,b_1,\dots,b_q,c,\dots,c$ vagy pedig $c^* = \max(b_q, 0)$-ra az $a,\dots,a,b_1,\dots,b_q,c^*,\dots,c^*$ sorozat jobban közelíti v-t. Hiszen így vagy az első vagy az utolsó konstans szakaszon közelebb megyünk az adott szakaszbeli számok átlagához. Ezzel az állítást beláttuk. □

A 3.3. állítás alapján közvetlen adódik a monoton regresszió megtalálásának alábbi, rekurzív algoritmusa:

3.4. Algoritmus *Vágjuk le az y sorozatból azt a maximális hosszúságú bevezető részt, aminek a konstans a legjobb monoton növekedő közelítése. A megmaradó sorozaton ismételjük meg ezt a levágást mindaddig, míg a sorozatunk 'el nem fogy'. Eredményként pedig adjuk meg azt a bemenő sorozattal azonos hosszúságú sorozatot, ami a lépésenként levágott szakaszokon konstans: az adott szakaszhoz tartozó y-ok átlaga.*

Könnyen látható, hogy az algoritmus jó. Az eredménye monoton növekedő sorozat. Ha ugyanis volna két olyan egymásutáni, az algoritmus szerint külön kezelt szakasz amire a két átlag nem volna növekedő, akkor a két szakaszba tartozó y-ok együttesen is teljesítenék a 3.3. szerinti feltételt. Így nem volna érvényes, hogy az algoritmus során a két szakasz közül az első esetén a maximális olyan szakaszt vágtuk le, aminek konstans a legjobb közelítése. Tehát az algoritmus biztosan monoton sorozatot szolgáltat.

Az is könnyen belátható, hogy az algoritmus által szolgáltatott sorozat egyben a legjobb közelítés is. Hisz vegyünk egy tetszőleges másik m^* monoton növekedő sorozatot. Vegyük az m^*-nak azokat a szakaszait, amik az y-ra alkalmazott fenti algoritmus szerint adódnak. Az m^* ezen a szakaszokon is monoton. Ugyanakkor az algoritmus szerint ezeken a szakaszokon az y sorozatnak a konstans, az adott szakaszon vett átlag a legjobb monoton közelítése. Ráadásul mint láttuk az y-nak az ezeken a szakaszokon vett átlagokból képzett sorozata monoton is. Így az algoritmussal kapott sorozat valóban jobb közelítése az y-nak mint az m^* sorozat.

3.5. Algoritmus *Az eredményt lépésenként határozzuk meg. Az első lépésben az y sorozatot, a továbbiakban pedig az előző lépésben nyert sorozatot módosítjuk a következőképpen. Ha találunk az algoritmusban olyan egymásutáni $u_1, \dots, u_k$ részsorozatot, amire $u_1 \geq \dots \geq u_k$, akkor az adott részsorozatot kicseréljük az ugyanolyan hosszú $\bar{u}, \dots, \bar{u}$ sorozatra. Ha az aktuális sorozatnak nincs monoton csökkenő részsorozata, akkor kész!*

Ez az algoritmus a feladat legkorábbi megoldása. Nem triviális annak belátása, hogy az algoritmus eredménye nem függ attól, hogy az átlagolásokat milyen sorrendben vettük. És az sem nyilvánvaló, hogy eredményként optimális megoldást kapunk.

A monoton regresszió konstansainak meghatározását szolgáló harmadik (utolsó, 3.6.) algoritmus csak annyiban különbözik az (3.4. algoritmus) elsőtől, hogy ennek az ügyes szervezése folytán a megoldáshoz szükséges $\mathcal{O}(n^2)$ idő $\mathcal{O}(n \log n)$-re csökken.

3.6. Algoritmus *Ezen algoritmus keretében közvetlenül a (k, z_k), $k = 0, \dots, n$ pontok alsó konvex burkát keressük meg.*

Az algoritmust legegyszerübb a következő történettel elmondani. Kössük össze a (k, z_k) pontokat, és tekintsük úgy hogy az így nyert görbe egy meredek tengerbe hulló sziklafal vonala. A vonaltól lefelé a tenger, fölfele pedig a sziklák vannak.

Menjünk ki a partra egy segédünkkel és egy szolgánkkal és határozzuk meg a konvex burkot a következő módon. Kezdetkor tűzzünk a fövenybe zászlókat minden (k, z_k) pontba. Majd álljunk a $0, 0$ zászló mellé. A szolga induljon el a parton előre, és húzogassa ki azokat a zászlókat amikre következőtől már nem látna minket. Amikor a szolga egy ilyen bentmaradó zászlóhoz ér, álljon meg és mi a segéddel menjünk utána. Ezután a szolga ismét menjen előre és húzza ki a zászlókat az előzőek szerint. És mi, ha megáll, akkor utána. A segéd pedig húzza ki azokat az eddig bentmaradt, általunk már elhagyott zászlókat amik előtt van olyan látható zászló ami még szintén bentmaradt.

Az eljárás helyessége nyilvánvaló. Bonyolultsága pedig $\mathcal{O}(n)$ mivel a segéd és szolga egyaránt legfeljebb n zászlót húz ki, mi pedig legfeljebb n-szer utasítjuk a segédünk hogy egy adott zászlót kihúzzon-e. Hasonlón a segédtől legfeljebb n zászló elhozását kérjük, és legfeljebb n-szer tekintünk vissza egy-egy zászlóra eldöntendő, hogy annak elhozását kérjük-e.

3.3.2. Monoton regresszió az R-project segitségével

A **R**-project számos programja alkalmas a monoton regresszió modelljének az illesztésére. A következőkben csak a legegyszerűbb, 'stats' csomagbeli 'isoreg()' függvény ismertetésére térünk ki részletesebben.

Az 'isoreg()' programnak egy vagy két vektor lehet az inputja. Egy vektor esetén ez a vektor a fentiek szerinti y, kettő esetén pedig az x és az y. Ha csak egy vektort adunk meg, akkor a program feltételezi, hogy az az x értéke szerint rendezett. Két vektor esetén a rendezettség nem szükséges.

Adjuk be a következő utasításokat:

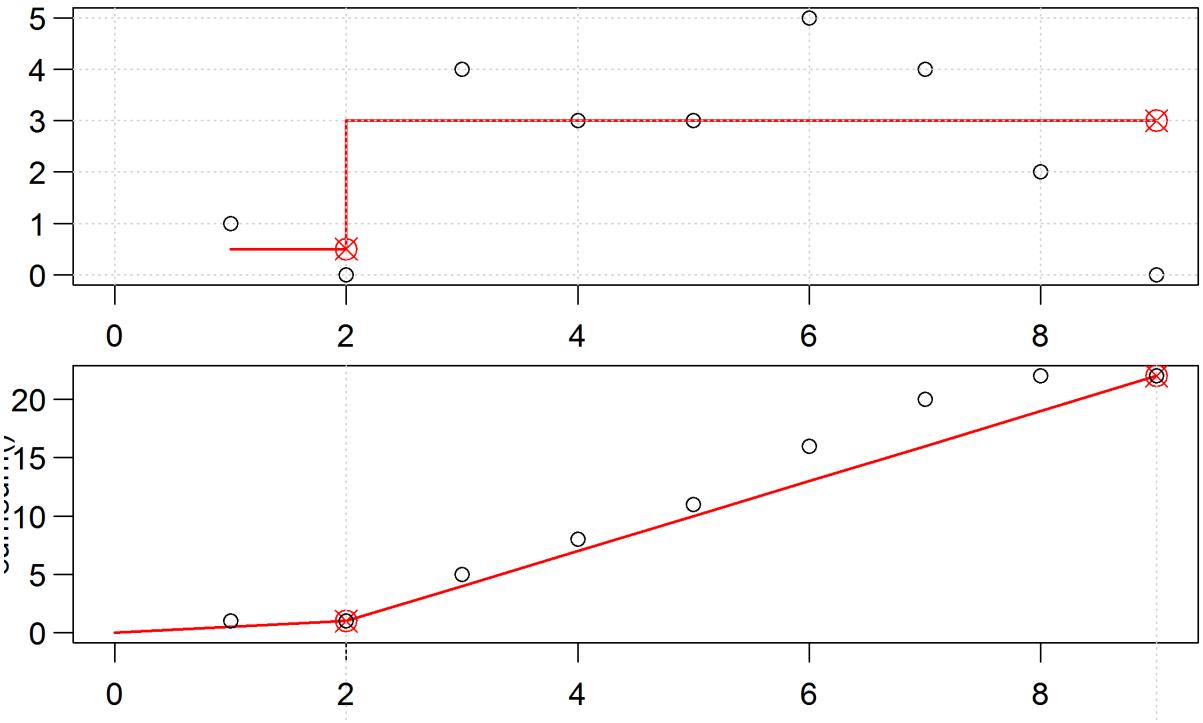
```
M <- isoreg(c(1,0,4,3,3,5,4,2,0))
class(M)
str(M)
M # ezt a kiírást a  print.isoreg() eljárás végzi
```

Esetünkben egyetlen input vektort adtunk meg, ami 3.3.1-beli leírás szerinti y vektornak felel meg, és aminek a hossza 9. Igaz, ez az eredményváltozó 'print.isoreg()' szerinti kiírásában és az eredményváltozó '$call' komponensében egyaránt kissé félrevezetően, mint 'x' vektor jelentkezik. Az eredmény, esetünkben az 'M' változó, egy 8 elemű lista. De ennek a '$x' illetve '$y' komponense már megfelel a logikus elnevezéseknek. A '$y' tartalmazza a megadott értékeket, és a '$x' értéke egy 'létra': '1:9'.

```
Isotonic regression from isoreg(x = c(1, 0, 4, 3, 3, 5, 4, 2, 0)),
  with 2 knots / breaks at obs.nr. 2 9 ;
  initially ordered 'x'
  and further components List of 4
 $ x : num [1:9] 1 2 3 4 5 6 7 8 9
 $ y : num [1:9] 1 0 4 3 3 5 4 2 0
 $ yf: num [1:9] 0.5 0.5 3 3 3 3 3 3 3
 $ yc: num [1:10] 0 1 1 5 8 11 16 20 22 22
```

Az eredményváltozóban az '$yf' a regressziós értékek vektora, azaz ez felel meg a 3.3.1 leírás szerinti $m_1, \ldots, m_n$ konstansoknak. Az '$yc' vektor az y változó értékeinek kumulatív összege. Ez a leírás szerinti z vektornak felel meg. Mindig eggyel hosszabb mint a megadott y vektor, de az első eleme mindig 0. Az 'M' eredményváltozónak ezeken kívül van még egy igen hasznos '$iKnots' eleme is. Ez azt mutatja meg, hogy a monoton regresszióban (az 'M$yf'-ben) hányadik megfigyelésig tartanak a konstans szakaszok. Azaz, hogy hol vannak a 3.3.1 szerinti leírásban is szereplő konvex burok csomópontjai.

A `plot(M, plot.type = "row")` parancs hatására egy olyan kétablakos (didaktikus) ábrát kapunk, ami egyrészt a monoton regressziónak megfelelő lépcsős függvénnyel együtt mutatja a megadott adatokat, másrészt az algoritmus konstrukciója szempontjából érdekes 3.3.1 leírás szerinti (j, z_j), $j = 0, \ldots, 9$ pontokat és azok alsó konvex burkát.
A **R**-project programrendszer számos kiegészítése tartalmaz hasonló célú programokat. Így például a 'monreg::monreg()', 'drtool::monoreg()', 'Cir::pava()', 'Iso::pava()' stb S ez utóbbi 'Iso' csomagban megtalálható az eljárás kétdimenziós változata is.



3.16. ábra. A 'plot(isoreg())' eredménye.

3.4. Általánosított lineáris regresszió

Az általánosított lineáris regresszió azaz a 'glm' (generalized linear model) nem cserélendő össze az általános lineáris modellel.

Az *általános* lineáris modell mindössze annyi változtatás a (közönséges) lineáris modellhez képest, hogyha a modellt a

$$Y = X\beta + \varepsilon$$

képlettel írjuk le, akkor míg a közönséges esetben az ε hiba kovariancia mátrixa egy olyan diagonális mátrix, aminek a diagonálisában mindegyik elem értéke ugyanaz a σ_ε^2, addig az általános lineáris modell esetén sem a végig egyenlőséget, sem pedig a diagonalitást nem tesszük fel. Ez az általánosítás a regresszió megoldóképletében mindössze egy kis módosítást jelent. (Csak azzal a nem kis problémával kell szembenézni, hogyha a Σ_ε-nal kapcsolatban nincsenek további információink, akkor azt a megfigyelt adatok alapján nem tudjuk megbecsülni...)

A általános lineáris modell alapesetében azt szokás feltenni, hogy a magyarázó és a célváltozók egyaránt folytonos $\mathbb{R}$-beli értékűek. Ezt a modellt szokás lineáris regressziónak nevezni. Az általános lineáris modell másik speciális esetét, azt amikor a magyarázó változók diszkrét lehetséges értékűek, szórásanalízis (ANOVA) modellnek nevezik.

Ámde tipikusan fordul elő olyan feladat, amikor az Y célváltozó lehetséges értéke véges vagy végtelen sok vagy esetleg szükségszerűen nemnegatív, vagy egy ($[0,1]$-beli) valószínűség. Hogyan lehetne ebben az esetben a körülményeket leíró X értékek ismeretében az Y-t becsülni vagy közelíteni? Ez a feladat azért problémás, mert vehető ugyan a magyarázó változóknak olyan a lineáris modell szerinti függvénye ami a célváltozó értékét, — amik esetünkben például valószínűségek — jól közelíti. Ámde a kapott lineáris függvénynek (hacsak az nem konstans) biztosan lesz olyan értéke ami nem $[0,1]$-beli. E probléma feloldására célszerünek látszik egy olyan *kapcsoló* (link) függvény alkalmazása, ami a célváltozó $[0,1]$-beli értékét az $\mathbb{R}$-be képezi. És pont ez az ötlet, a sok egyéb helyzetben is felhasználható általánosított lineáris regresszió alapötlete. [5]

Az *általánosított* lineáris regresszió módszerét úgy is magyarázhatjuk, hogy az X magyarázó változóknak egy olyan (a paramétereiben) *lineáris* függvényét keresünk, ami nem az Y magyarázandó változót, hanem annak egy megfelelő $g()$ kapcsoló (link) függvénnyel transzformált $\eta = g(Y)$ értékét tekinti célvátozónak.

Azaz, az általánosított lineáris regresszió esetén a

$$g(Y) = \eta = X\beta + e$$

regressziót vizsgáljuk mint alapmodellt, valamely ismert (vagy paraméteres) g *kapcsoló* vagy *link* függvény mellett. Egyszerűbb esetekben feltételezzük, hogy az e egy olyan ε véletlen hiba adott esetbeli értéke, aminek a kovarianciamátrixa diagonális. Sőt esetleg még azt is feltesszük, hogy ez a kovarianciamátrix $I\sigma_\varepsilon^2$ valamely ismeretlen σ_ε^2 konstanssal. Tipikus még annak a feltételezése is, hogy a modell bal oldalán nem egy-egy megfigyelt érték transzformáltja, hanem a megfigyelt érték várható értéke áll.

3.4.1. Az általánosított lineáris modell

A kapcsoló (link) függvények

A tipikus kapcsoló (link) függvények közül a legismertebb a *logit* függvény. De kapcsolófüggvény tetszőleges olyan függvény lehet, ami az adott modell környezetben értelmez-

hető. Most mégis mindössze hármat — a 'logit', a 'probit' és a 'cloglog' függvényt, — mutatunk be részletesebben is. Döntésünket azzal magyarázva, hogy ez a három az, amit leggyakrabban alkalmaznak.

A logit függvény. A logit transzformáció: a valószínűség odds-értékének logaritmusa, azaz

$$\eta = \ln(\pi/(1-\pi)) = \ln(odds(\pi))$$

Az inverze pedig a megfelelő átviteli függvény:

$$\pi = \frac{\exp(\eta)}{1+\exp(\eta)}.$$

A logit alkalmázásának praktikussága és értelme könnyen megmagyarázható. Ez a függvény összetett függvény: a valószínűséghányados (az odds) logaritmusa. A valószínűséghányados – az esemény bekövetkezési és nem bekövetkezési valószínűségének hányadosa – gyakran használt mutató, a $[0,1]$-beli valószínűségeket az $\mathbb{R}^+$-ba képezi. Azért kedvelt mutató, mert az értékének az értelmezése könnyen követhető szabályok szerint lehet:

- olyan eseményre, ami inkább bekövetkezik mint nem, az esélyhányados 1 feletti,
- nagy valószínűségek mellett tetszőlegesen nagy értékeket is felvehet,
- 0 valószínűségű eseményekre az értéke nulla.

Hátránya a szempontunkból, hogy csak a pozitív számok tartoznak bele az értékkészletébe. Ezen 'segít' a második függvény, a logaritmus. A logaritmusnak további haszna, hogy emellett a sokat használt esély (likelihood) függvény könnyen számolható lesz. További előnye, hogy ez a függvény nagyon hasonlít a probit függvényhez, ami egy másik, szintén nagyon jól értelmezhető átviteli függvény.

A probit függvény. A Φ standard normális eloszlásfüggvény inverze azaz Φ^{-1}. Előnye, hogy könnyen értelmezhető modellt ad az indikált esemény bekövetkezési esélyére vonatkozóan. Hátránya hogy a $\Phi^{-1}(p)$ számolása lassú, nehézkes.
A probit modell szerint azt feltétezzük, hogy minden kísérlet esetén a vizsgált esemény bekövetkezési valószínűsége az adott kisérletsorozatra (esetleg a konkrét kisérlet körülményeire) jellemző paraméterű normális eloszlás szerinti. Ezt az eloszlást a kisérlet küszöb eloszlásának szokás nevezni. E modell szerint, egy-egy kisérlet alkalmával akkor következik be az esemény, ha a körülmények egy lineáris függvénye meghaladja azt a véletlen küszöb értéket, ami az adott kisérlethez tartozik. Vagyis azt az értéket amit a (normális eloszlású) véletlen az adott esetre kisorsolt.

Pontosabban. Tegyük fel, hogy a célváltozó mellett mindössze egy, folytonos értékű, x-el jelölt magyarázó változó áll rendelkezésre. A célváltozó által indikált esemény valószínűsége pedig legyen olyan, hogy az, az x függvényében egy ismeretlen μ várható értékkel és σ szórással egy normális eloszlás szerinti. Azaz, ha az vizsgált eseményt A jelöli akkor

legyen $P(A|x) = \Phi_{\mu,\sigma}(x) = \Phi((x-\mu)/\sigma)$ ahol ez utóbbi Φ a standard normális eloszlás, a $\Phi_{\mu,\sigma}$ pedig az $\mathcal{N}(\mu,\sigma)$ eloszlás eloszlásfüggvénye. Ez úgy is értelmezhető, hogy egy-egy megfigyelt egyed olyan, hogy az esetében az A esemény egy $\mathcal{N}(\mu,\sigma)$ eloszlás szerint kisorsolt x^* szint mellett következik be. Ha az esethez tartozó $x < x^*$akkor az adott esetben a célváltozó által indikált A esemény nem következett be, ha pedig $x^* < x$ akkor az adott esetben bekövetkezett az A esemény. Másként mondva: minden kisérlet esetén ha a hozzátartozó x kicsi, akkor biztos nem következett be a megfelelő egyed esetén az A esemény, ha pedig nagy, akkor pedig majdnem biztos, hogy az A bekövetkezettnek tekintendő, az a határ ami felett az indikált A esemény bekövetkezik normális eloszlású.

A komplemens loglog függvény. Tetszőleges $p \in [0,1]$-re

$$\log(-\log(1-p))$$

Az inverze:

$$1 - \exp(-\exp(y))$$

Ez a függvény is monoton nő, de szemben a probittal és a logittal, aszimmetrikus.

Ez a kapcsolófüggvény a Gumbel eloszlásnak felel meg, ugyanúgy mint ahogyan a probit a normálisnak. Paraméterezzük ugyanis a Gumbel-eloszlást a következő módon:

$$G(x) = 1 - \exp(-\exp((x-\alpha)/\kappa))$$

akkor a 'cloglog' transzformált a p valószinűségre:

$$\log(-\log(1-p)) = \beta_0 + \beta_1 x$$

ahol $\beta_0 = -\alpha/\kappa$ és $\beta_1 = 1/\kappa$.

Paraméterbecslés maximum likelihood alapon,
binomiális eloszlású célváltozó és 'logit' link függvény esetén

A általánosított lineáris regresszió egy fontos speciális esete az, amikor a célváltozó binomiális és a magyarázó változók folytonosak.

Ha beírjuk a 'binomial()' parancsot, akkor a válaszból láthatjuk, hogy ez a 'selfStart' 'family' a 'logit' transzformációt párosítja a binomiális eloszláshoz. Binomiális eloszlás esetén a 'logit'-nak mint link-nek az alkalmazása azért indokolt, mert ez a binomiális eloszlásnak az úgynevezett kanonikus, más szóval a természetes transzformációja. Ugyanis ez az a transzformáció ami mellett torzítatlan becslést kaphatunk.

A következő részben pont egy ilyen adatsor feldolgozását mutatjuk be. Ezért most egy rövid levezetés árán előállítjuk egy binomiális eloszlású minta logit transzformáció melletti likelihood függvényét, és annak deriváltjait. Elvileg ezek azok a függvények amiket majd az ott meghívott algoritmus a paraméter becslésekor felhasznál. Igaz, nekünk akkor erre explicit nem lesz szükségünk, hisz a megfelelő függvények be vannak építve az ott meghívott 'binomial' 'selfStart' függvénybe.

Ha a célváltozó *logit* kapcsoló (link) függvény szerinti értéke η, a magyarázó változók száma m és a η lineáris regressziójának együtthatói $\beta_0, \ldots, \beta_m$, akkor a célváltozó j. megfigyelésének logitjára

$$\eta_j \approx \beta_0 + \beta_1 x_{1,j} + \cdots + \beta_m x_{m,j},$$

és a binomiális eloszlású célváltozó valószinűség paraméterének a magyarázó változók szerinti becsült értéke, a *logit*-nak megfelelő inverz (kapcsoló) függvény szerint:

$$\pi_j = \frac{\exp(\eta_j)}{1 + \exp(\eta_j)}.$$

Ekkor pedig, ha a minta szerint a j. kisérlet esetén n_j próbálkozásból k_j volt sikeres, a minta likelihoodja (esélyfüggvénye):

$$L(\beta) = \prod_{j=1}^{n} \binom{n_j}{k_j} \pi_j^{k_j} (1 - \pi_j^{n_j - k_j}).$$

Mivel az esélyfüggvénynek csak a maximumhelye érdekes, a könnyebb kezelhetőség érdekében vegyük az esélyfüggvény logaritmusát és alakítsuk át ekvivalens módon.

$$\log(L(\beta)) = \sum_{j=1}^{n} \left(log\binom{n_j}{k_j} + k_j \log \pi_j + (n_j - k_j) \log(1 - \pi_j) \right) =$$

$$\sum_{j=1}^{n} \left(log\binom{n_j}{k_j} + k_j \log(\pi_j/(1 - \pi_j)) + n_j \log(1 - \pi_j) \right) =$$

$$= \sum_{j=1}^{n} \left(log\binom{n_j}{k_j} + k_j \eta_j + n_j \log(1 - e^{\eta_j}) \right)$$

Tekintsük a függvény így nyert kifejezésének a deriváltját a β koordinátái szerint:

$$\frac{\partial \log L(\beta)}{\partial \beta_i} = \sum_{j=1}^{n} k_j x_{i,j} - \sum_{j=1}^{n} n_j k_j e^{\eta_i} (1 - e_i^{\eta})^{-1}$$

ami felhasználva a korábbi jelölést:

$$\frac{\partial \log L(\beta)}{\partial \beta_i} = \sum_{j=1}^{n} k_j x_{i,j} - \sum_{j=1}^{n} n_j x_{i,j} \pi_j$$

Ezen egyenletrendszert megoldva megkapjuk a becslési feladat maximum likelihood megoldását.

3.4.2. Az általánosított lineáris modell a gyakorlatban

Egy kártevő pusztulási arányát vizsgálták az alkalmazott méreg koncentrációja és a kártevő neme függvényében. Minden méreg szintnek nemenként 20 egyedet tettek ki. Az eredményt az alábbi táblázat mutatja.

az egyedek neme	M	M	M	M	M	M	F	F	F	F	F	F
a dózis logaritmusa	0	1	2	3	4	5	0	1	2	3	4	5
elpusztult (db)	1	4	9	13	18	20	0	2	6	10	12	16
életben maradt (db)	19	16	11	7	2	0	20	18	14	10	8	4

Az alábbi programsorok előbb definiálják a megfelelő változókat, a megfelelő tartalommal, majd meghívják az általánosított lineáris modellt illesztő 'glm()' eljárást.

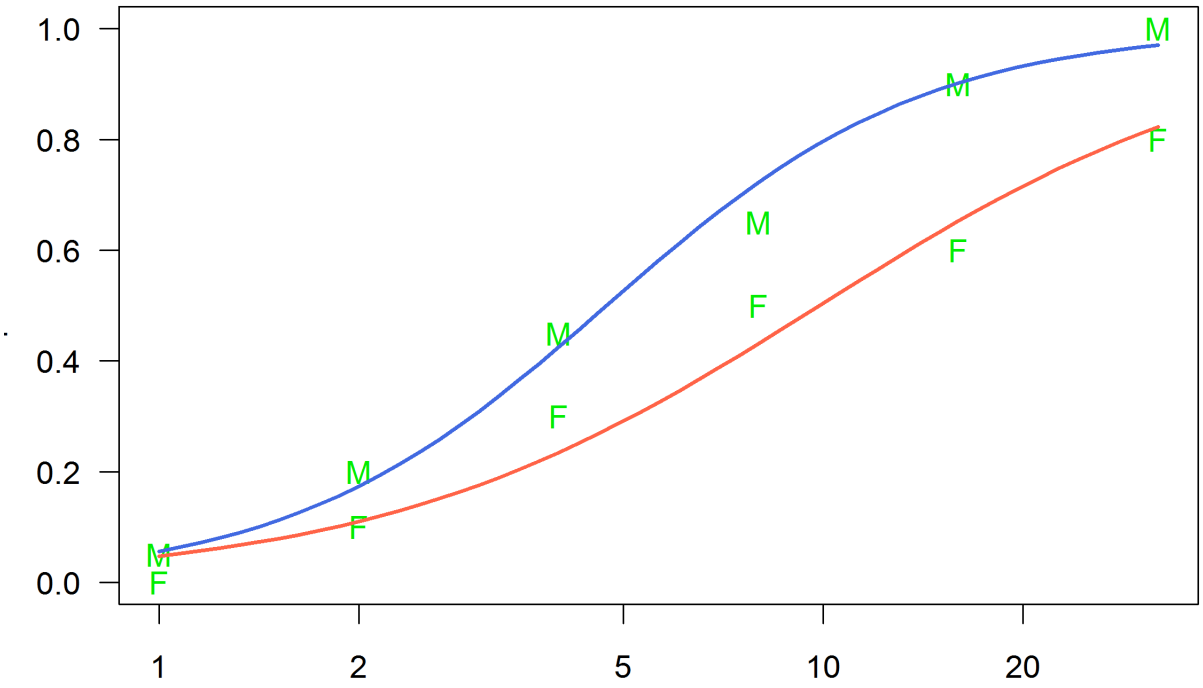
```
dose <- rep(0:5, 2) # 12 hosszu 0:5,0:5
dead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor(rep(c("M", "F"),each=6))
rbind(dose,dead,sex)
AD <- cbind(dead, alive=20-dead) # elpusztult+életben maradt=20

W <- glm(AD ~ sex*dose, family=binomial)
summary(W)
```

Látható, hogy a nemet mint faktorváltozót adtuk meg. A 'glm()' első argumentuma definiálja, hogy mik a magyarázó változók, és hogy mi legyen a célváltozó. Az, hogy magyarázó változóként két változónevet egy '*' jellel összekapcsolva adtunk meg — figyelembe véve azt is hogy az egyik közülük faktor változó — azt jelenti, hogy a rendszer a faktorváltozó minden lehetséges értékéhez kiszámol egy lineáris regressziót a célváltozó link függvény szerinti értékéhez.

Grafikusan is ellenőrizhetjük a feldolgozott adatokat és az eredményeket, például a következő utastásokkal:

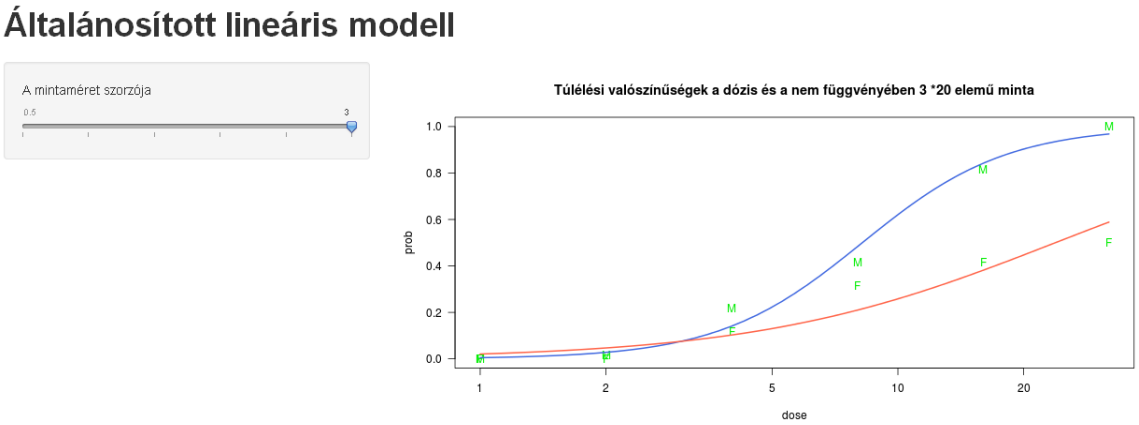
```
plot(c(1,32), c(0,1),log = "x",
     xlab = "dose", ylab = "prob",
     las=1,type = "n")
text(2^dose, dead/20, as.character(sex),col='green2')
dx <- seq(0, 5, 0.1)
df<-data.frame(dose=dx,sex=factor("M"))
lines(2^dx,predict(W,df,type = "response"),col='royalblue',lwd=2)
df<-data.frame(dose=dx,sex=factor("F"))
lines(2^dx,predict(W,df,type = "response"),col='tomato',lwd=2)
```

A fenti utasítások eredménye a 3.17 ábra.



3.17. ábra. A 'glm()' paranccsal illesztett túlélési valószínűségek

Ehhez a modellhez interaktív animáció is készült, amely a http://hpz400.cs.elte.hu:3838/ZA_glm/ címen található. Itt a felhasználó beállíthatja, hogy a fentiekben bemutatott gyakoriság táblában szereplő értékek hányszorosa legyen a szimulált binomiális eloszlás várható értéke, amely a módosított gyakoriság tábla értékeit adja meg. Ha erre a szimulált adathalmazra futtatjuk le a glm módszerét, akkor a 3.18 ábrát kapjuk, ahol láthatóak kisebb eltérések az eredeti adatbázisra vonatkozó 3.17 ábrához képest.



3.18. ábra. Animációs ábra a túlélési valószínűségekről

Vizsgáljuk meg részletesen az ábra készítésekor felhasznált 'predict()' parancsot! Mivel a 'predict()' argumentumába írt W objektum osztálya 'glm', az esetünkben a 'predict.glm()' működött. Azt, hogy az η közelítésekor mik voltak a lineáris modell paraméterei, például a 'coef(W)' paranccsal kaphatjuk meg.

Futtassuk le az alábbi utasítássorokat:

```
b<-as.numeric(coef(W))# a számított együtthatók
#  1           2    3    4
# (Intercept) sexM dose sexM:dose

predict(W,data.frame(dose=0,sex=factor("F")))
# ugyanaz mint
b[1]

predict(W,data.frame(dose=0,sex=factor("M")))
# ugyanaz mint
b[1]+b[2]

predict(W,data.frame(dose=2,sex=factor("F")))
b[1]+2*b[3]

predict(W,data.frame(dose=5,sex=factor("M")))
# ugyanaz mint
b[1]+b[2]+5*b[3]+5*b[4]
```

```
f<-"M";d<-3;# tetszôlegesen megválasztott nem és dózis
p<-predict(W,data.frame(dose=d,sex=factor(f)));
k<-b[1]+d*b[3]+(if(f=="M") b[2]+d*b[4] else 0)
c(pred=p,calc=k) # a két szám egyenlô!!
```

Az első paranccsal a vizsgált együtthatókat a 'b' változóba irjuk. A következő négy parancspár azt mutatja, hogy a 'predict' eredménye megegyezik az együtthatókból általunk számolttal. Az utolsó rész az általános képletünk

$$\eta = Intercept + d \cdot dose + \text{ha sex=="M" akkor még } + (sexM + d \cdot sexM : dose)$$

helyességét mutatja.

Megjegyzendő, hogy az előbbi utasítássorral a log.odds-ok egyenlőségét vizsgáltuk. Ugyanis a 'glm(,family=binomial)' paraméterezés miatt esetünkben ez volt az η értéke, és a 'predict.glm()' megfelelő 'type' paraméter híján az η-t számolja. Azt, hogy a felhasznált 'family=binomial' paraméterezés mellett mi a link függvény és mi az Y feltételezett eloszlása, a 'binomial()' parancs felhasználásával állapíthatjuk meg. A 'binomial()' parancsra a válasz:

```
Family: binomial
Link function: logit
```

Vagyis az illesztett modell link (kapcsoló, esetleg bal?) függvénye a 'logit' azaz az odds logaritmusa, és az adatok feltételezett eloszlása a binomiális eloszlás. Azt, hogy egy modellt melyik eloszlás és link családdal illesztettük, utólag is ellenőrizhetjük a 'family(W)' paranccsal.

Egy 'glm' modell értékelése szempontjából szinte mindegy, hogy mi az η értéke. Az η-nak inkább csak technikai szerepe van. Ha azt akarjuk, hogy a 'predict.glm()' az $\hat{Y}$ becsléseket szolgáltassa, akkor fel kell használnunk a 'predict'-nek a type="response" paraméterét. Vizsgáljuk meg a következő parancssort:

```
f<-"M";d<-3;
(p<-predict(W,data.frame(dose=d,sex=factor(f)),type="response"))
log(p/(1-p))# ez ugyanaz mint a
predict(W,data.frame(dose=d,sex=factor(f))) # mert az eta az alap
(e<-predict(W,data.frame(dose=d,sex=factor(f)),type="link"))
exp(e)/(1+exp(e))
```

A második paranccsal megkapjuk azt a valószínűséget, ami a $d = 3$ dózis esetén a hímnemű egyedek pusztulási valószínűsége. A következő parancsok azt mutatják, hogy

ennek a valószínűségnek a logitja tényleg ugyanaz, mint amit a 'predict' a type="link" paraméter mellett szolgáltat. Az utolsó parancs pedig azt mutatja, hogy a kapott η értéknek a logithoz tartozó kapcsolófüggvény szerinti transzformáltja valóban a korábban kiszámított, előrejelzett p valószínűség.

3.4.3. Modell családok a 'glm' függvényhez

A 'család' szó helyett talán jobb volna a 'készlet' szót használni. Ugyanis egy olyan objektum fajtáról van szó amiben lényegében minden megtalálható ahhoz, hogy egy 'glm' modellt illesszünk. Nem pedig arról, amit a név alapján gondolhatnánk. Nevezetesen, hogy egy-egy most tárgyalandó 'family' több hasonló dolgot tartalmazna.

Az előző példában a 'binomial' eloszláscsaládot használtuk fel. Ez azt jelentette, hogy a célváltozót binomiális eloszlásúnak vettük, a felhasznált kapcsolófüggvény pedig a *logit* függvény volt.

Az **R** -ben az illesztett modellek összefűzve kezelik a feltételezett eloszlást és a kapcsolófüggvényt. Maga a 'binomial()' egy 'family' osztályú 12 elemű lista, aminek minden eleme egy-egy szöveg, függvény vagy kifejezés. Esetünkben ez a struktúra a 'str(binomial())' paranccsal kapható meg:

```
$ family    : chr "binomial"
$ link      : chr "logit"
$ linkfun   :function (mu)
$ linkinv   :function (eta)
$ variance  :function (mu)
$ dev.resids:function (y, mu, wt)
$ aic       :function (y, n, mu, wt, dev)
$ mu.eta    :function (eta)
$ initialize:  expression(...)
$ validmu   :function (mu)
$ valideta  :function (eta)
$ simulate  :function (object, nsim)
```

Azt például, hogy mi a variancia képlete a binomiális családban, a 'binomial()$variance' paranccsal ismerhetjük meg. Mint az az elemi valószínűségszámitási ismereteink alapján várható: `function (mu) mu * (1 - mu)`.

A következő táblázat a 'stats' csomag néhány fontosabb 'family' osztályú objektumát listázza, a lehetséges kapcsolófüggvénnyekkel.

`binomial`	`logit, probit, cauchit, log, cloglog`

```
gaussian                identity, log,  inverse
Gamma                   inverse, identity, log
inverse.gaussian        1/mu^2, inverse, identity, log
poisson                 log, identity, sqrt
```

Vannak még úgynevezett kvázi családok is. Ezeknél bizonyos paraméterek becslési módja megadható.

4. fejezet

Dimenziócsökkentési eljárások

4.1. Bevezető

A klasszikus többdimenziós statisztikai részhez értünk. Talán ez az, amire rögtön gondolunk. ha többdimenziós statisztikáról esik szó. Érdekes, hogy bár a klasszikus dimenziócsökkentési feladatok fontossága a számítógépek sebességének és a tárolókapacitásnak a hihetetlen mérvű növekedtével csökkenni látszik (a kérdés azért nem lesz soha idejétmúlt, mert persze az adatállományok mérete is folyamatosan nő) – ráadásul nem felejtkezhetünk el arról, hogy az adatbányászat, mint a nagy adatbázisok elemzésének önálló tudománya jelentős fejlődésen ment át az utóbbi néhány évtizedben, mégis folyamatosan jelennek meg az elemzések, új algoritmusok a témakörben. Ennek magyarázata az, hogy az adatbázisokban rejlő információk feltárásának klasszikus és mégis hatékony módszereiről lesz szó.

A matematikai modellekről bővebb leírás például a [35] tankönyvben található.

4.2. Főkomponens-analízis

4.2.1. A feladat megfogalmazása

Legyen Y p dimenziós (megfigyelésvektor, függő változó). A cél, hogy korrelálatlan komponensű X segítségével állítsuk elő

$$Y = VX$$

alakban, ahol V ortonormált mátrix (forgatás). Tegyük fel az egyszerűség kedvéért, hogy $EY = 0$ és legyen Y kovarianciamátrixa $EYY^T = \Sigma$ teljes rangú. Ekkor a

$$\Sigma = V\Lambda V^T$$

spektrálfelbontásban szereplő V ortonormált $p \times p$-es mátrix, melynek $v_1, \ldots, v_p$ oszlopvektorai éppen Σ sajátvektorai, Λ pedig a sajátértékekből álló diagonális mátrix (feltehetjük, hogy $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$). A sajátértékek pozitívak, mert Σ pozitív definit. Legyen

$$X = V^T Y. \tag{4.1}$$

A (4.1) koordinátái: $X_i = v_i^T Y$ az Y főkomponensei ($i = 1, \ldots, p$). Ezek tulajdonsága, hogy

4.1. Tétel *X_i szórása maximális az összes olyan valószínűségi változó között, melyekre*

1. *$X_i = a^T Y$ ($a \in \mathbb{R}^p$) és $||a|| = 1$*
2. *X_i korrelálatlan az első $(i-1)$ főkomponenshez $(X_1, \ldots, X_{i-1})$.*

4.2. Megjegyzés *A főkomponensanalízis erősen érzékeny a változók skálájára. Ha ezeket megváltoztatjuk, akkor a főkomponensek is megváltoznak. Ezért gyakran célszerű a változók átskálázása olymódon, hogy mindegyik egységnyi szórású legyen. Ezzel biztosítható, hogy potenciálisan egyforma jelentőséget tulajdonítunk mindegyik koordinátának. Matematikailag ez a transzformáció egyszerűen azt jelenti, hogy a kovariancia helyett a korrelációs mátrixszal dolgozunk.*

4.2.2. Becslés az adatok alapján

Ha a megfigyeléseink p dimenziós normális eloszlásból származnak, akkor $n > p$ elemű y_i minta alapján a tapasztalati kovarianciamátrix:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})(y_i - \overline{y})^T$$

egyúttal maximum likelihood becslés is. A fenti számításokat erre a becsült mátrixra (vagy még inkább a becsült korrelációs mátrixra) elvégezve megkapjuk a főkomponenseket. A kapott becslés konzisztens a sajátértékekre és a sajátvektorokra is, normális határeloszlással:

4.3. Tétel

$$\sqrt{n}(\hat{\lambda}_n - \lambda) \to Z$$

ahol Z 0 várható értékű és $2\Lambda^2$ kovarianciamátrixú normális eloszlás. Hasonló eredmény érvényes a sajátvektorokra is.

4.2.3. Példa alkalmazások

Hétpróba az 1988. évi olimpián

A HSAUR (Handbook of Statistical Analyses Using R) csomagot [7] használjuk, mely lefedi a legfontosabb statisztikai témákat. A főkomponens analízis első példája a modern hétpróba (100 m gátfutás, magasugrás, súlylökés, 200m síkfutás, távolugrás, gerelyhalyítás, 800m síkfutás) 1988. évi szöuli olimpián elért eredményeit tartalmazza versenyszámonként és összesítve. Ahhoz, hogy minden számban a legnagyobb értékek jelentsék a legjobb eredményt, a futószámok eredményeit az alábbiak szerint transzformáltuk, és elkészítettük a hét szám pontdiagramját (4.1 ábra).

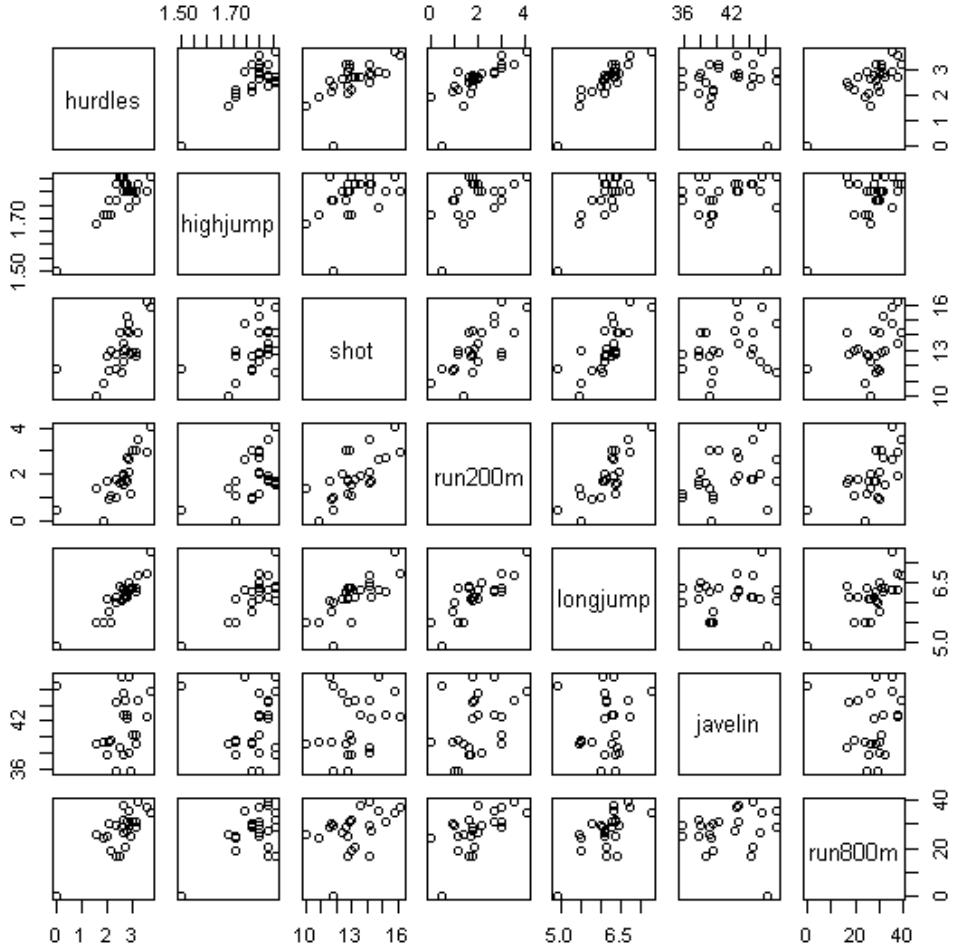
```
library(HSAUR)
data("heptathlon", package = "HSAUR")
heptathlon$hurdles <- max(heptathlon$hurdles) - heptathlon$hurdles
heptathlon$run200m <- max(heptathlon$run200m) - heptathlon$run200m
heptathlon$run800m <- max(heptathlon$run800m) - heptathlon$run800m
score <- which(colnames(heptathlon) == "score")
plot(heptathlon[,-score])
```

Látható a 4.1 ábrából, hogy tipikusan pozitívan korreláltak az értékek és hogy sok számnál van egy kiugróan gyenge eredmény, amely ugyanahhoz a versenyzőhöz tartozik.

A főkomponens analízis standard módszerének meghívása rendkívül egyszerű:

```
heptathlon_pca <- prcomp(heptathlon[, -score], scale =
TRUE)
print(heptathlon_pca)
```

Vegyük észre, hogy itt a változókat egységnyi szórásúra átskálázva végezzük az elemzést (' scale=TRUE'), ami lényeges, mert az egyes versenyszámok számszerű eredményei között nagyságrendbeli különbségek vannak. A valóságban viszont közel egyforma súlyt gondolunk ezeknek a koordinátáknak.



4.1. ábra. A szöuli olimpia női hétpróba versenyének transzformált eredményei

Az eredményt a 4.2 ábra mutatja. Látható, hogy az első főkomponens meglehetősen jelentős. Ebben gyakorlatilag az összes szám (a futószámoknál transzformált) eredménye hasonló együtthatóval szerepel, csak a gerelyhajítás (javelin) súlya kisebb - ez összhangban van azzal, hogy ez a szám alig korrelál a többivel. A negatív előjeleknek nincs szerepe, a -1-szeres ugyanolyan tulajdonságú lenne. A második főkomponens viszont nagyobbrészt a gerelyhajítás eredményén alapul. A szórás 90%-át csak a 4. főkomponens után érjük el, ahogy ez a 4.3 ábrából is látható, amit a következő egyszerű utasítás révén kaptunk:

```
summary(heptathlon_pca)
```

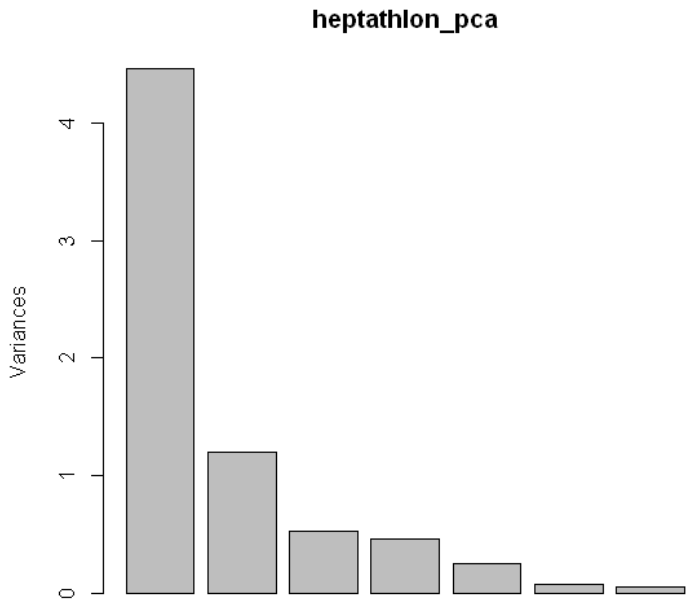
```
Standard deviations:
[1] 2.1119364 1.0928497 0.7218131 0.6761411 0.4952441 0.2701029 0.2213617

Rotation:
                PC1         PC2         PC3         PC4         PC5         PC6
hurdles  -0.4528710  0.15792058 -0.04514996  0.02653873 -0.09494792 -0.78334101
highjump -0.3771992  0.24807386 -0.36777902  0.67999172  0.01879888  0.09939981
shot     -0.3630725 -0.28940743  0.67618919  0.12431725  0.51165201 -0.05085983
run200m  -0.4078950 -0.26038545  0.08359211 -0.36106580 -0.64983404  0.02495639
longjump -0.4562318  0.05587394  0.13931653  0.11129249 -0.18429810  0.59020972
javelin  -0.0754090 -0.84169212 -0.47156016  0.12079924  0.13510669 -0.02724076
run800m  -0.3749594  0.22448984 -0.39585671 -0.60341130  0.50432116  0.15555520
                PC7
hurdles   0.38024707
highjump -0.43393114
shot     -0.21762491
run200m  -0.45338483
longjump  0.61206388
javelin   0.17294667
run800m  -0.09830963
```

4.2. ábra. A főkomponens analízis a hétpróba versenyszámok eredményeire

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6    PC7
Standard deviation     2.1119 1.0928 0.72181 0.67614 0.49524 0.27010 0.2214
Proportion of Variance 0.6372 0.1706 0.07443 0.06531 0.03504 0.01042 0.0070
Cumulative Proportion  0.6372 0.8078 0.88223 0.94754 0.98258 0.99300 1.0000
```

4.3. ábra. Az egyes főkomponensek fontossága

Ha viszont a szórásnégyzetet tekintjük, akkor jobban kiemelődnek a fontosabb komponensek, és így akár azt is mondhatjuk, hogy elegendő az első két főkomponenst meghagyni (4.4 ábra). Az első főkomponens fontosságát jól mutatja, hogy ha kiszámoljuk az egyes versenyzők score-ját, akkor ennek -0.99 a korrelációja a hivatalos pontszámmal, ami pedig nemlineáris függvénye az eredményeknek (ld. [13]). Érdekes ábra még a biplot, amely az első két főkomponens szerinti score-okat ábrázolja, centrálva és egységnyi szórásra skálázva, hogy ugyanezen a diagramon az egyes komponensek is ábrázolhatóak legyenek (4.5 ábra). A győztesek azok, akiknél az első koordináta a legkisebb (merthogy láttuk, hogy ennek -1-szerese szinte egybeesik a végső pontszámmal). Közöttük a 2. főkomponens alapján látható különbség érdekes lehet sportszempontból. Ugyancsak jól látszik a gerelyhajítás elkülönülése a többi versenyszámtól.

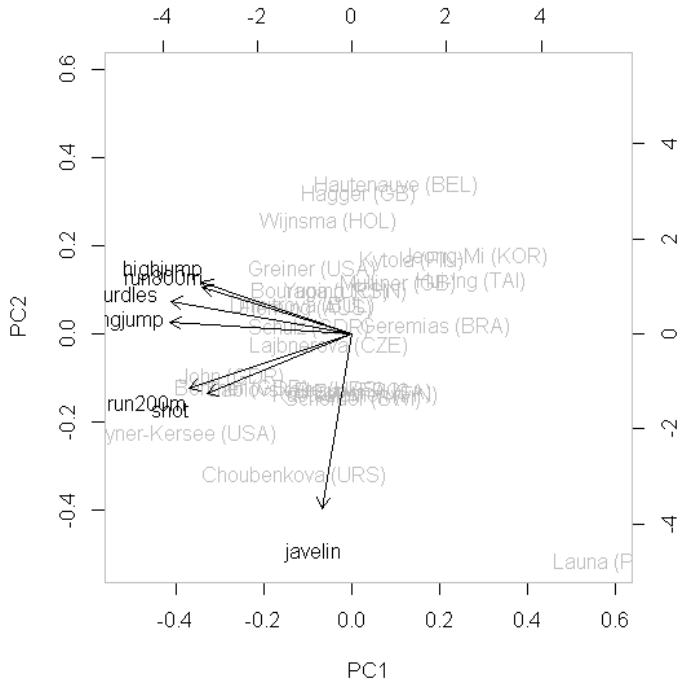


4.4. ábra. Az egyes főkomponensek által megmagyarázott szórásnégyzet

Részvények napi hozam-adatai

A pénzügyi matematikában is gyakran kap szerepet a többdimenziós adatok elemzése. Különösen magas dimenziós feladatok adódhatnak a portfolió-optimalizálás témakörében. Itt az a cél, hogy az adott hozam-szintet minél kevésbé kockázatos (azaz minél kisebb szórású) befektetésekkel tudjuk elérni. Ehhez a hozam-adatok kovarianciamátrixát kell minél megbízhatóbban becsülni, amihez célszerű a zaj kiszűrése - ez pedig éppen a nagyon kicsi szórású komponenseket jelenti.

A vizsgált adatbázis 50 részvény 8 évnyi napi loghozam-adatsorát tartalmazta. Az adatsor a http://hpz400.cs.elte.hu:3838/ZA_fact/ címen található. Ha erre lefuttatjuk a főkomponens-elemzést, akkor az adódik, hogy van egy kiugróan nagy sajátérték (ehhez tartozik az első főkomponens), amit a gazdaság állapotának tekinthetünk. A következő főkomponensek egészen másképp viselkednek, amint ez a 4.2.3 ábrából is látható. Az első főkomponensben gyakorlatilag majdnem minden részvény egyforma súllyal szerepel, míg a második főkomponens szinte teljes egészében egy részvényből áll. Mivel itt az eredeti hozamok az érdekesek, ezért nem skáláztuk át az adatokat.

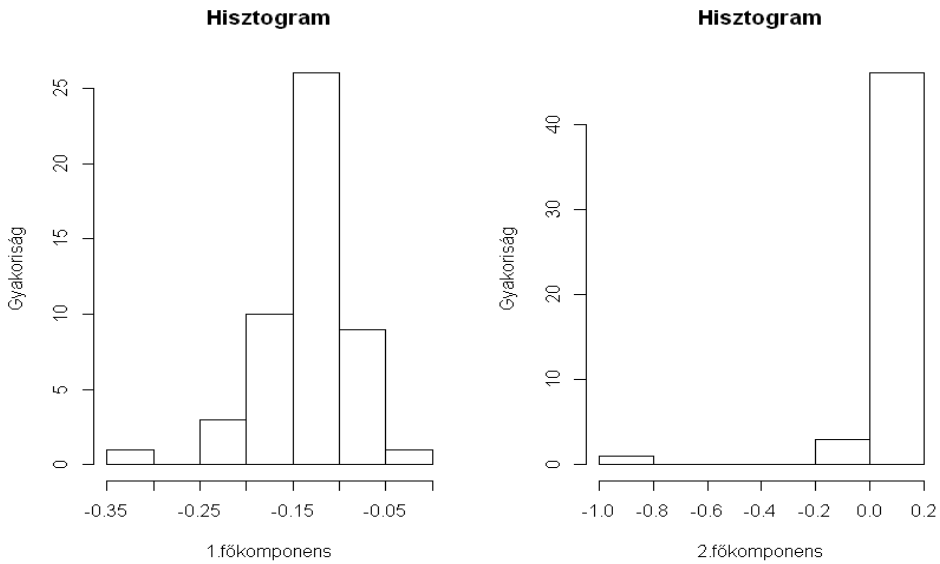


4.5. ábra. A versenyzők és a versenyszámok biplot ábrája

Az USA államainak adatai

Ez az elemzés a [20] alapján készült. Az alapadatok az USA államainak

- lakosság-számát (population, ezer fő),
- az egy főre eső GDP értékét (income, US dollár/fő),
- az analfabéták arányát (illiteracy, százalék),
- a születéskor várható élettartamot (Life Exp, évben),
- a gyilkosságok számát (Murder, 100 000 főre vetítve),
- a felnőttek között az érettségizettek részarányát (HS grad, százalékban),
- a fagyos napok számát (frost),
- a területét (Area, négyzetmérföldben),



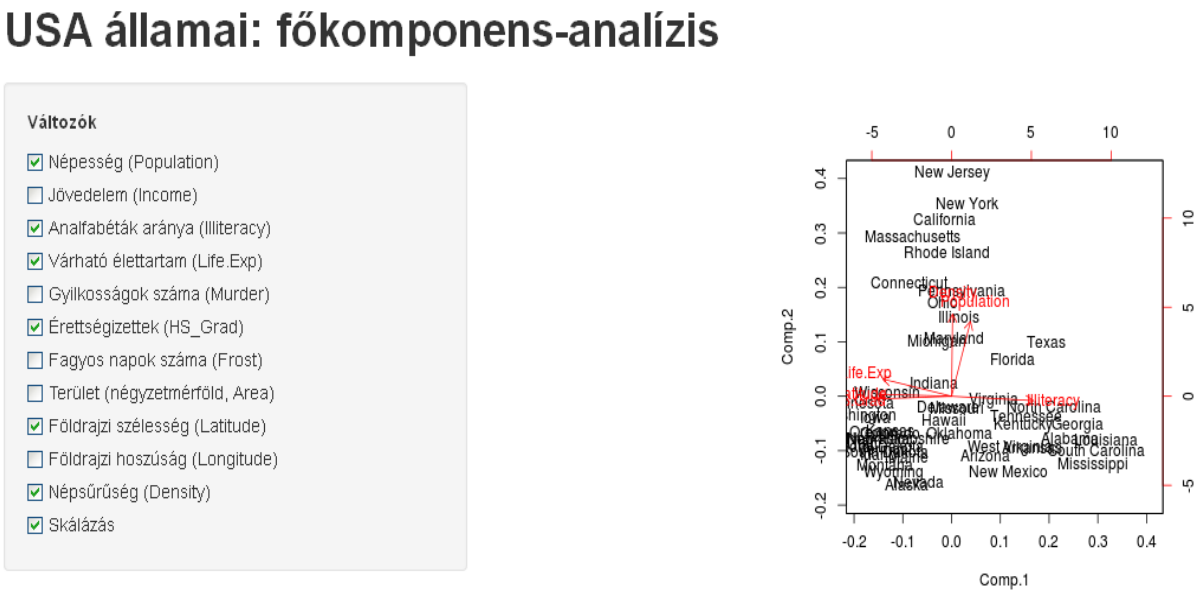
4.6. ábra. A napi hozamok első két főkomponense együtthatóinak hisztogramja

tartalmazzák. A fő újdonság ebben a példában, hogy erről az adatbázisról interaktív animáció is készült, mely az http://hpz400.cs.elte.hu:3838/ZA_princomp/ címen található. Az animációban további kiegészítő változók (földrajzi hosszúság és szélesség és népsűrűség – latitude, longitude, population density) is figyelembe vehetőek, illetve elhagyhatóak oszlopok az alapváltozók közül is. Az eredményeket pedig a már többször bemutatott biplot ábra adja meg, itt az első két főkomponens által meghatározott koordinátarendszerben ábrázoljuk az egyes államokat és magukat a mért változókat is.

Az első kérdés, hogy alkalmazzunk-e skálázást. A válasz az, hogy igen, hiszen az adatok igen eltérő skálán vannak mérve – de érdemes kipróbálni, mit kapunk skálázás nélkül. A kapott 4.7 ábra egy példa a lehetséges eredmények közül. Jól látható, hogy mely változók szerepeltek az elemzésben.

Az alapadatokra kapott eredményeket a következő kóddal előállított 4.8 ábra mutatja. Látható, hogy az első főkomponens a hideg, képzett, hosszú várható élettartamú államokat különíti el a kevésbé képzett, magasabb gyilkossági arányú államoktól. A második főkomponens pedig leginkább a nagy területű, népesebb és gazdagabb államokat különíti el a többitől.

```
biplot(prcomp(state.x77,scale.=TRUE),cex=c(0.5,0.75))
```



4.7. ábra. Az USA államaira vonatkozó interaktív animáció

4.2.4. R függvények

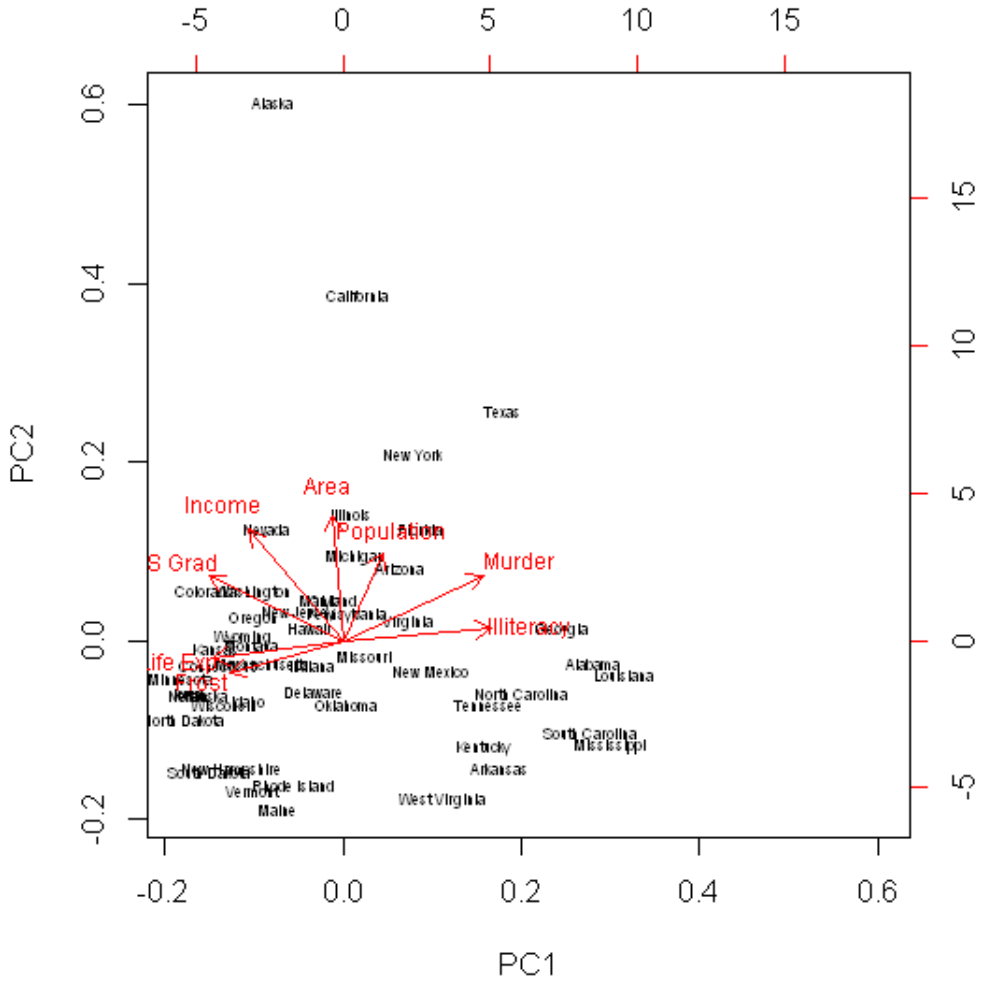
Az eddig használt függvények a MASS csomagban vannak, tehát nem kell semmit sem telepítenünk használatukhoz. Sőt, érdekességképpen megemlítjük, hogy a 4.2.3 szakaszban alkalmazott 'princomp' függvényen kívül használható a 'prcomp' is, ami a szinguláris érték felbontáson alapul, ezért bizonyos esetekben stabilabb lehet - de az eredmény kevésbé sok információt ad, mint az eddig használt.'princomp' függvény.

pcaMethods

Ez elsősorban genetikai alkalmazásokra készült. Egyik fő előnye, hogy hiányzó adatokat is tud kezelni. A pótlásukra különböző eljárások közül lehet választani, így például regressziós, Bayes-i vagy klaszter alapú módszerek is találhatóak a csomagban. Megjegyzendő, hogy ez a csomag nem a megszokott cran honlapról, hanem a bioconductor lapról `http://www.bioconductor.org/` tölthető le.

labdsv

Ez a csomag pedig ökológiai alkalmazásokra készült. Itt a 'pca' függvény számítja ki a főkomponenseket, gyakorlatilag a 'prcomp' módszerével, a megszokott eredményt adva. Ugyanakkor több hasznos grafikus ábrázoló függvényt és skálázó algoritmust is tartalmaz (ezekről az eljárásokról részletesen a 6. fejezetben lesz szó).



4.8. ábra. Az USA államaira vonatkozó adatok és az első két főkomponens biplot ábrája

4.3. Faktoranalízis

4.3.1. A feladat megfogalmazása

Ennek a dimenziócsökkentő eljárásnak az a lényege, hogy nem megfigyelhető közös tényezőket tételez fel, amelyek hatnak a megfigyelt vektorra. A faktorok száma értelemszerűen alacsonyabb, mint a megfigyeléseké, tehát valóban csökken a dimenzió a módszer révén. Érdemes megjegyezni, hogy több helyen, így például a pénzügyi matematikában, faktormodellnek neveznek olyan eseteket is, ahol megfigyelhető faktorok hatását modellezik

(pl. ilyen faktor lehet az alapkamat vagy egy aktuális valutaárfolyam).

A faktoranalízis alkalmazása igen széles területet ölel fel a pszichometriától (mely faktorok mentén lehet például a különböző képességeket mérni) az ökológián és a geokémián át egészen a modern genetikai (mikroarray) kutatásokig.

Matematikailag a következő modellről van szó:

$$X = AF + W + \mu$$

ahol X p-dimenziós valószínűségi változó μ várható értékkel. F $k < p$-dimenziós, W pedig p-dimenziós korrelálatlan valószínűségi változók, amelyek mind 0 várható értékűek és a kovarianciamátrixuk diagonális. F kovarianciamátrixára $\mathrm{cov}(F) = I$ is feltehető. Az F a közös faktor, W pedig az egyedi faktor, hiszen F minden komponense szerepel X minden komponensének felírásában, de W-nek csak a megfelelő komponense:

$$X_i = \sum_{j=1}^{k} a_{ij} F_j + W_i + \mu_i.$$

A korrelálatlanság miatt

$$D^2(X_i) = \sum_{j=1}^{k} a_{ij}^2 + D^2(W_i),$$

ahol a jobb oldalon az első tagot kommunalitásnak (a közös faktorokból adódó szórásnégyzet), a másodikat pedig egyedi szórásnégyzetnek nevezzük. A faktormodellben is a kovarianciamátrix felbontása a célunk:

$$\Sigma = AA^T + \Psi, \tag{4.2}$$

ahol $\Psi = \mathrm{cov}(W)$. Tehát az a különbség a főkomponens-analízishez képest, hogy itt feltesszük az egyedi szórások jelenlétét a modellben.

Azt mondjuk, hogy Σ leírható a k-faktormodellel, ha van a (4.2) szerinti felbontása. Érdemes megjegyezni hogy az A mátrix nem egyértelmű, tetszőleges G ortogonális mátrixszal szorozva AG is megoldása a (4.2) egyenletnek. Ezt a G mátrixot forgatásnak (rotation) nevezzük. A gyakorlatban ennek segítségével tudjuk elérni, hogy a kapott faktorok jól interpretálhatók legyenek. A leggyakrabban a varimax forgatást szokták használni. Ennek lényege, hogy azokat a koordinátákat keressük, amelyekre teljesül, hogy a változókra összegezve a négyzeteket, a lehető legnagyobb értéket kapjuk. Ez a gyakorlatban általában olyan faktorokat eredményez, amelyek szerint a súlyok (loading) egy része 1-hez, más része 0-hoz közeli. Ez azért előnyös, mert így jól magyarázható egyszerű faktorstruktúra áll elő.

A 'varimax' és más úgynevezett ortogonális forgatás lényege, hogy ortogonális faktorokat ad. Ez értelemszerűen akkor praktikus, ha a faktorok a valóságban is ortogonálisak.

Ez azonban sokszor nem reális feltevés, ezért mostanában – a számítógépek kapacitásának fejlődésével párhuzamosan – nő a nem ortogonális forgatások szerepe. Ezek közül az 'oblimin' forgatás a legelterjedtebb. A gyakorlatban az az eljárás ajánlható, hogy először nem ortogonális forgatást alkalmazunk és leellenőrizzük a faktorok közötti korrelációt. Ha ezek jelentősek, akkor maradunk a nem ortogonális forgatásnál, de ha elhanyagolhatóak (például 0.32-nél kisebbek a korrelációk; ez a küszöb onnan adódik, hogy ekkor 10% a szórásnégyzetek közötti átfedés), akkor áttérünk a könnyebben interpretálható ortogonális forgatásra. Az egyes módszereket jól illusztrálja a 4.9 ábra és a példáknál mi is visszatérünk erre a problémára.

A gyakorlatban az egyik legfontosabb kérdés, hogy mennyi is az a k. amelyre már megadható k-faktor modell. Erre különböző tesztek találhatóak a szakirodalomban. Most röviden áttekintjük a legfontosabb módszereket, amiket majd a gyakorlatban is bemutatunk. A faktorok számának meghatározása többféleképpen történhet. Talán a legegyszerűbb az a módszer, ami a – főkomponens-analízisben látottaknak megfelelően – annyi faktort választ, hogy a teljes modellben a kommunalitások összege megadott értéknél (tipikusan pl. 0.9) nagyobb legyen.

A következő lehetőség a Scree plot, ami a sajátértékek csökkenését vizsgálja. Ahonnan kezdve az a csökkenés lassul (ez a sajátértékek ábrázolása során a "könyök") , ott már nem érdemes több faktort tekinteni. Ennek a módszernek nem csak grafikus változata van (l. 4.3.3 fejezet).

Végül talán a leginkább megalapozott módszer a Horn féle párhuzamosság-próba, amelynek során szimulációval lehet tesztelni, hogy vajon korrelálatlan normális eloszlású mintánál mekkora sajátértékek fordulhatnak elő. Ha ezeknek a sajátértékeknek a 95%-os percentilisénél kisebb értéket kapunk, akkor az már elhagyható. Ez az eljárás is megtalálható az **R** nFactors csomagjában (l. 4.3.3 fejezet, [29]).

4.3.2. Példák

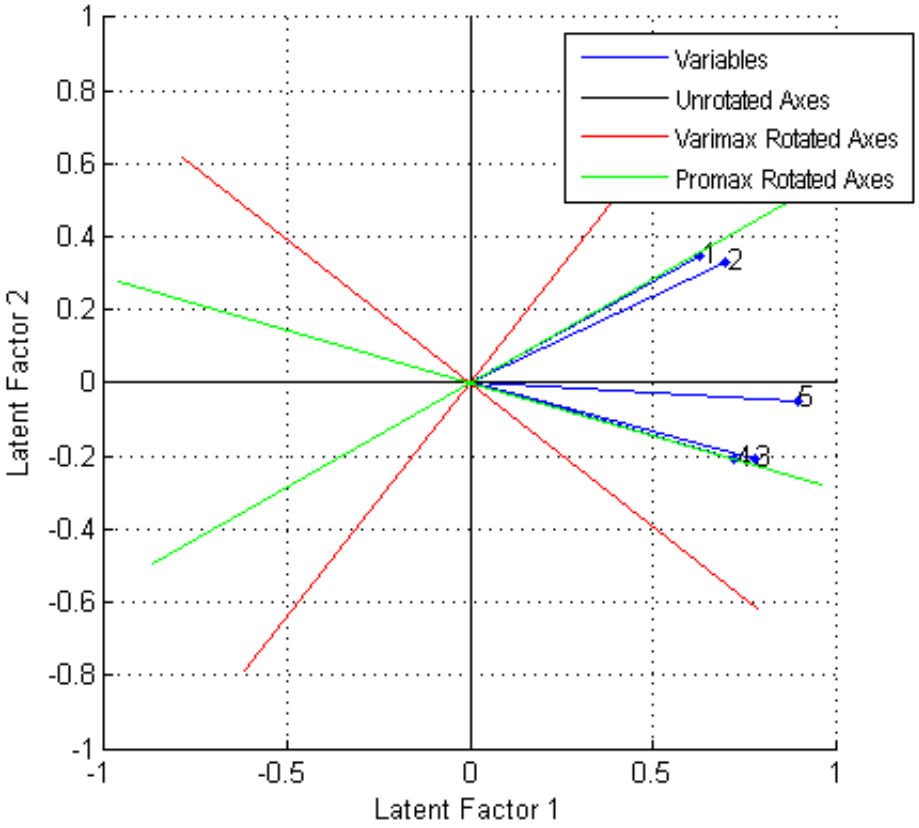
Tárgyak kedveltsége

Itt egy nagyon egyszerű példán nézzük meg a fentiekben bemutatott módszerek működését. A 4.10 ábra mutatja 10 diákra, hogy mennyire kedvelnek 6 tárgyat (5 fokozatú skálán), [18] alapján. A teljes minta 300 diák adatait tartalmazza.

Itt az a feltételezésünk, hogy 2 faktor írja le a diákok attitűdjét: a természettudományos és a matematikai érdeklődés.

Az alábbi programrészlet beolvassa az adatokat és meghívja a faktoranalízis alapfüggvényét:

```
data <- read.csv("dataset_exploratoryFactorAnalysis.csv")
fit <- factanal(data, factors=2)
fit$loadings
```



4.9. ábra. 5 vizsga eredménye 2 faktorral modellezve - a különböző forgatásokra

Az eredményül kapott súlyok (loadings) és a megmagyarázott szórásnégyzet részarányát a 4.11 ábra mutatja.

Látható, hogy az első faktornál a természettudományos tárgyak súlya magas, míg a második faktor a matematikai tárgyakkal korrelál. A statisztika súlya valamivel alacsonyabb a többi matematikai tárgyénál, viszont ez a tárgy kisebb súllyal az első faktorban is szerepel, mutatva a kapcsolatát mindkét csoporttal.

Az első faktor a szórásnégyzet 35%-át, a második pedig 31%-át magyarázza. A kommunalitást vizsgálva megállapítható, hogy a statisztika tárgy függ a legkevésbé össze a kapott faktorokkal ($0.17^2 + 0.506^2 = 0.285$ ez a négyzetösszeg), míg a kalkulus a leginkább (0.952).

	BIO	GEO	CHEM	ALG	CALC	STAT
1	1	1	1	1	1	1
2	4	4	3	4	4	4
3	2	1	3	4	1	1
4	2	3	2	4	4	3
5	3	1	2	2	3	4
6	1	1	1	4	4	4
7	3	3	3	2	3	1
8	4	3	4	2	3	2
9	2	1	3	3	4	3
10	2	3	3	2	3	4

4.10. ábra. 10 diák adatai 6 tárgy kedveltségéről

```
Loadings:
     Factor1 Factor2
BIO  0.855   0.133
GEO  0.779   0.135
CHEM 0.865
ALG          0.791
CALC         0.971
STAT 0.170   0.506

                 Factor1 Factor2
SS loadings        2.124   1.863
Proportion Var     0.354   0.311
Cumulative Var     0.354   0.665
```

4.11. ábra. Két faktor súlyai és a megmagyarázott szórásnégyzet

Részvények napi hozam-adatai

A 4.2.3 részben már bemutatott adatsor 50 részvény napi log-hozam adatait tartalmazza a 2004-2012 közötti időszakra. A célunk, hogy megkeressük a részvények árfolyamingadozását meghatározó legfontosabb faktorokat. Most nem vesszük figyelembe a rendelkezésre álló háttérinformációkat, tehát nem regressziós, hanem faktoranalízises módszereket alkalmazunk.

Az alábbi programrészlet beolvassa az adatokat és meghívja a faktoranalízis alapfüggvényét:

```
dat=read.table("50reszveny.txt")
fit <- factanal(dat, 3, rotation="varimax")
print(fit, digits=2, cutoff=.3, sort=TRUE)
```

A 'factanal' függvény 3-as paramétere azt adja meg, hogy 3 faktort keressünk, a forgatásnál pedig a varimax a választásunk. A 'print' utasítás eredményeként megkapjuk a faktorokat, a hiba-szórásnégyzeteket (uniquenesses) és a szórásnégyzet felbontását a főkomponens-analízisnél látottaknak megfelelően. Ez a részlet látható a 4.12 ábrán. Végül az úgynevezett Bartlett teszt vizsgálja, hogy nem lehet-e egységmátrix a korrelációs mátrix, azaz van-e értelme a faktoranalízisnek. Itt ezt értelemszerűen elutasítja ez a próba. Egyszerű kritériumként (Kaiser) azt is szokták használni, hogy annyi faktort érdemes választani, ahány sajátérték nagyobb 1-nél. Ez a mi esetünkre 5 faktort ad, de az 1 mint kritérium meglehetősen szubjektív. A faktorszám meghatározására alkalmas, hatékony módszerekre még visszatérünk a 4.3.3 fejezetben.

```
                Factor1 Factor2 Factor3
SS loadings        8.31    8.11    5.48
Proportion Var     0.17    0.16    0.11
Cumulative Var     0.17    0.33    0.44
```

4.12. ábra. Az első három faktor által megmagyarázott szórásnégyzet a részvényadatoknál

4.3.3. R függvények

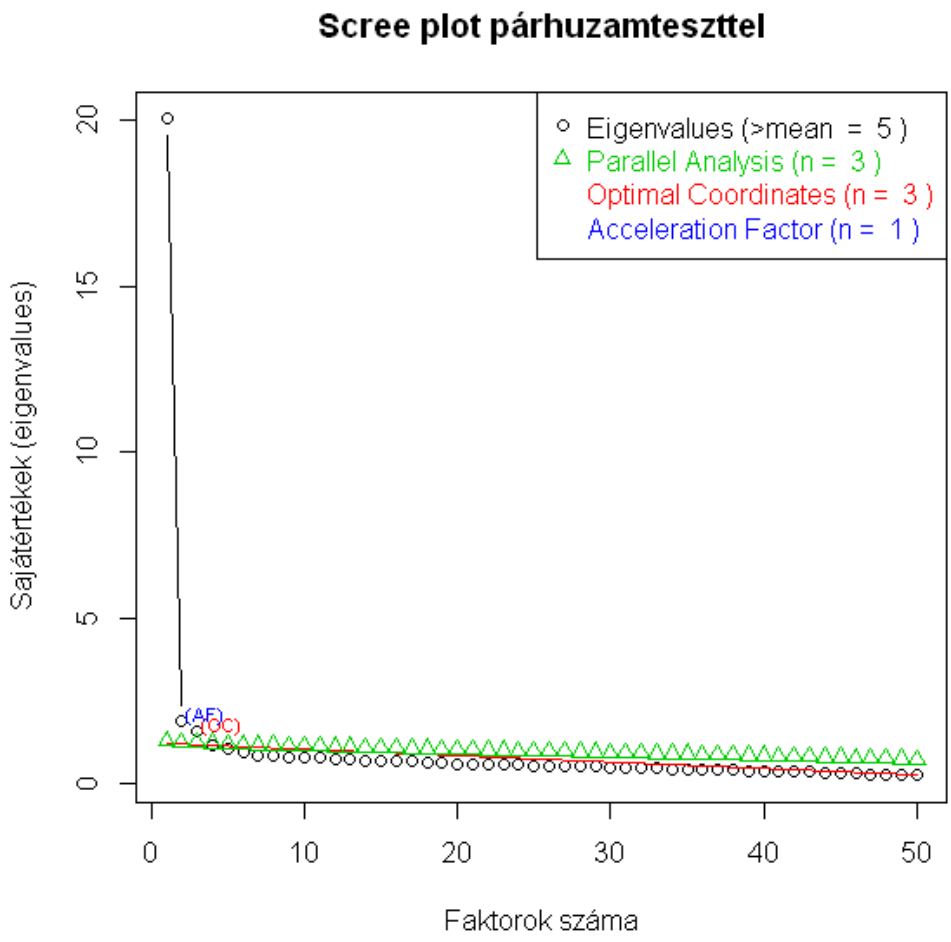
Az eddig használt függvények a MASS csomagban vannak, tehát nem kell semmit sem telepítenünk használatukhoz. Ugyanakkor speciális eljárásokhoz vannak célprogramok, amiket röviden bemutatunk.

nFactors

Ez a faktorok számának meghatározásában nyújt segítséget, [29]. Több formális teszt és heurisztikus módszer is rendelkezésre áll a feladat vizsgálatához.

A 4.13 ábrát a következő kódrészlettel állíthatjuk elő:

```
dat=read.table("./50reszveny.txt")
```



4.13. ábra. A részvényadatok scree ábrája és a párhuzam (parallel) teszt

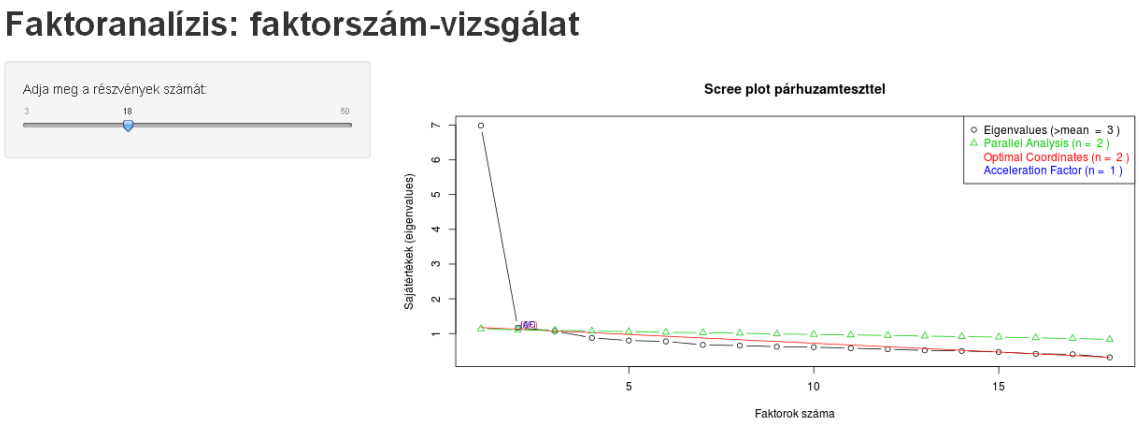
```
ev <- eigen(cor(dat)) # sajátértékek
ap <- parallel(subject=nrow(dat),var=ncol(dat),
  rep=100,cent=.05) #100  bootstrap minta
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
plotnScree(nS,main="Scree plot párhuzamteszttel",xlab="Faktorok száma",
ylab="Sajátértékek (eigenvalues)")
```

Azt kaptuk, hogy a teszt 3 faktort javasol (itt metszi a véletlen adatok sajátértékeinek sora (háromszögek) a mi részvényadatainkra kapott sajátértékek sorát (körök). A gyorsítási tényező (acceleration factor), ami a 4.13 ábra folytonos görbéjének legmeredekebb pontját keresi meg, csupán egy faktort javasolt volna.

Erről az adatbázisról animáció is készült, mely az http://hpz400.cs.elte.hu:

3838/ZA_fact/ címen található.Itt sorban látjuk, hogy az adatbázis első k elemét kiválasztva a fenti Scree plot hány faktort javasol. Megfigyelhető, hogy k növelésével a szükséges faktorok száma is nő.

A 4.14 ábra mutatja az animáció eredményét egy konkrét k értékre.



4.14. ábra. A részvényadatok scree ábrája és a párhuzam (parallel) teszt az interaktív animációnál

Ha a módszert az egyszerűbb, tárgy-szimpátia adatbázisra alkalmazzuk, akkor a 4.15 ábra adódik.

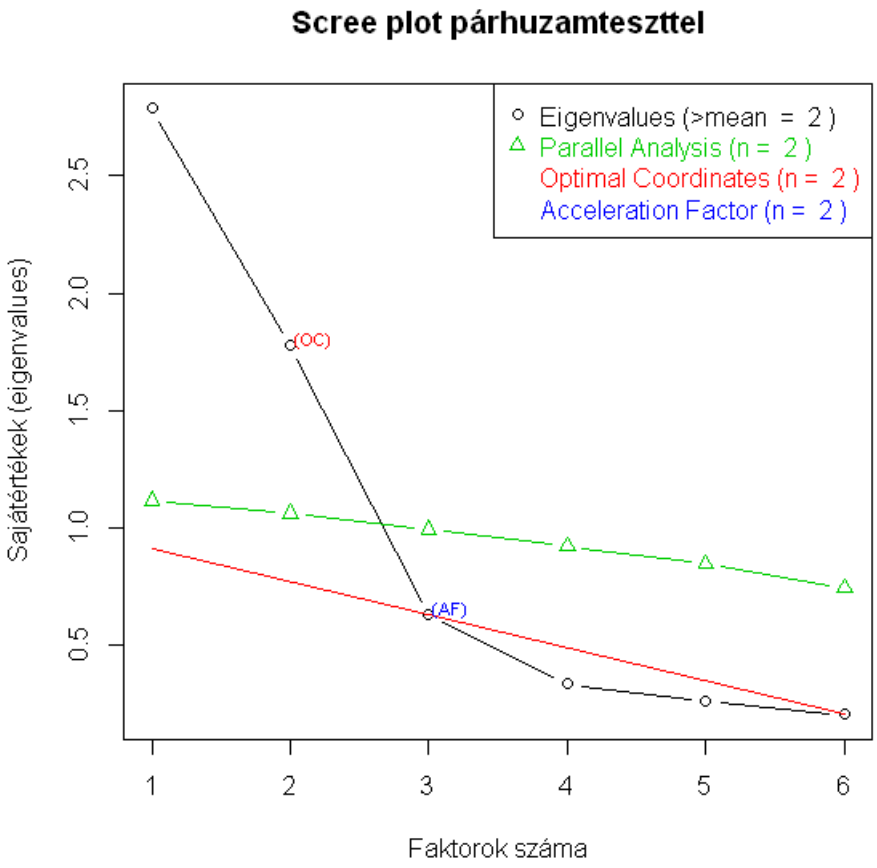
Itt egyértelmű, hogy a 2 faktor a jó választás, minden teszt ezt mutatja.

psych

Mivel a faktoranalízis egyik legfontosabb alkalmazási területe a pszichometria, nem meglepő, hogy az ottani adattípusokra és pszichometriában hasznos speciáls eljárásokra speciális csomag készült. Ez a psych csomag [32], amelynek általánosan érdekes tulajdonságait most bemutatjuk. Mindenre természetesen nincs lehetőségünk, már csak azért sem, mert csak a leírása 310 oldalas – igaz, hogy ennek nagy része nem a jegyzetünk témája.

A faktoranalízis alapfeladatát itt a 'fa' függvény valósítja meg. Ennek több érdekes paramétere is van:

1. 'fm': ez határozza meg, hogy milyen módszerrel keressük a faktorokat. Az alapértelmezés a legkisebb négyzetes becslés (ordinary least squares, OLS) aminek előnye, hogy közel elfajuló mátrixokra is használható, szemben a maximum likelihood eljárással, ami ilyenkor nem konvergál. Lehetőségünk van a hagyományos főtengely (principal axes) módszert is választani, ami a korrelációs mátrix sajátérték felbontásából iteratív eljárással dolgozik. Végül a súlyozott legkisebb négyzetek módszere is választható. Itt a korrelációs mátrix inverzének a diagonálisa adja a súlyokat, ami azt eredményezi, hogy a kisebb kommunalitású változók súlya növekszik.



4.15. ábra. A tárgy-szimpátia scree ábrája és a párhuzam (parallel) teszt

2. 'n.iter': megadja a faktorsúlyok konfidencia intervallum számításához használható bootstrap iterációk számát. Ennek alkalmazását a részvényadatokon illusztráljuk. A 4.16 ábra az első 10 részvényre adja meg a 3 faktor súlyát és a kapott konfidencia intervallumokat.

3. 'scores': különböző módszereket választhatunk az egyedi faktorscore-ok becslésére. A Bartlett score abból indul ki, hogy ezeknek egyetlen véletlen eleme a W_i, amelynek kovarianciamátrixa Ψ. Ha ezt ismertnek tételezzük fel, akkor explicit megkapható a likelihood becslés, egyébként pedig a $\hat{\Psi}$ becslésből kiindulva használhatjuk a módszert. A Thurstone féle regressziós módszer is klasszikus és gyakran használt. Végül a Berge féle eljárás előnye, hogy megőrzi a korrelációstruktúrát: a faktorscore értékek korrelációja ugyanakkora lesz, mint maguké a faktoroké. Ez a leginkább javasolt a tipikus esetekben, de közel elfajult problémákra nem használható

Coefficients and bootstrapped confidence intervals

	low	WLS3	upper	low	WLS1	upper	low	WLS2	upper
V1	0.52	0.56	0.61	0.35	0.39	0.48	0.26	0.30	0.36
V2	0.33	0.38	0.45	0.52	0.59	0.68	0.21	0.28	0.38
V3	0.36	0.44	0.54	0.15	0.23	0.37	0.14	0.18	0.23
V4	0.10	0.15	0.20	0.09	0.16	0.30	0.58	0.66	0.71
V5	0.35	0.45	0.57	0.24	0.31	0.43	0.19	0.24	0.31
V6	0.32	0.42	0.53	0.25	0.33	0.46	0.18	0.23	0.30
V7	0.67	0.72	0.76	0.12	0.18	0.32	0.14	0.19	0.25
V8	0.41	0.46	0.51	0.39	0.44	0.51	0.21	0.26	0.33
V9	0.42	0.48	0.54	0.22	0.26	0.36	0.17	0.21	0.27
V10	0.24	0.29	0.33	0.09	0.14	0.22	0.21	0.26	0.32

4.16. ábra. Az első 10 részvény faktorsúlyai és a hozzájuk tartozó 95%-os konfidencia intervallumok

4. 'rotate': a korábban említettek mellett számos forgatási módszer alkalmazható. Az alapértelmezés az 'oblimin'.

5. 'alpha': a root mean square error of approximation (RMSEA) konfidencia intervallumának megbízhatósági szintje. Ez a teszt azt vizsgálja, hogy a modellből kapott kovarianciamátrix-becslés elég közel van-e a tapasztalati kovariancia mátrixhoz. Minél kisebb az RMSEA érték, annál jobb az illeszkedés (pl. 0.05 lehet egy tipikus alpha érték). Bár a tapasztalataink a részvény-adatainkkal azt mutatták, hogy ez nem egy erős teszt: még az egy faktor esetén is csak 0.04-re nőtt az RMSEA.

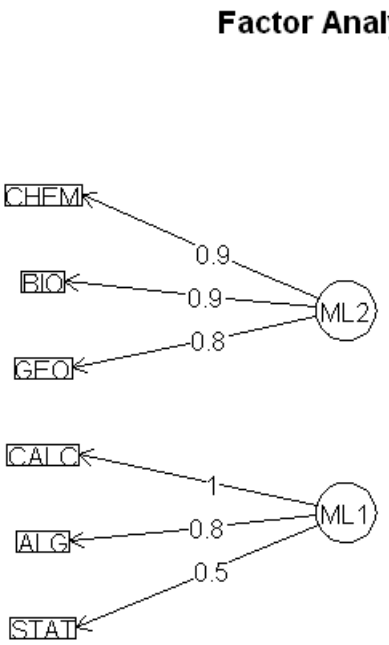
Megjegyezzük, hogy az ilyen egyszerű feladatoknál, mint például a 6 dimenziós tárgy-adatbázis, nincs jelentősége a módszer választásnak.

A 'fa.diagram' függvény ábrája szemléletesen mutatja az egyes faktorok jelentését. A tantárgy-adatbázisra ez a 4.17 ábrán látható.

Hasznos ábra a 'cor.plot', ami a korreláció értékeit teszi szemléletessé. A tantárgyakra még a számokat (a korreláció százalékos értékét) is érdemes volt feltüntetni (4.18 ábra). Jól elkülönül a két faktor.

A részvény-adatoknál már nincs ilyen egyértelmű struktúra, ezért ott nem az eredeti adatokra, hanem a faktoranalízis eredményeként kapott objektumra hívtuk meg a rajzolót. Itt (4.19 ábra) az egyes faktorok legfontosabb komponensei figyelhetőek meg jól.

Ha a bevezetőben említett módon szeretnénk a forgatások közül választani, akkor először alkalmazzuk az oblimin forgatást. Az eredményt a 4.20 ábra foglalja össze. A második táblázat azt mutatja, hogy jelentősek a korrelációk, tehát érdemes ezt a módszert alkalmazni.



4.17. ábra. A tantárgyak 2 faktorának diagramja

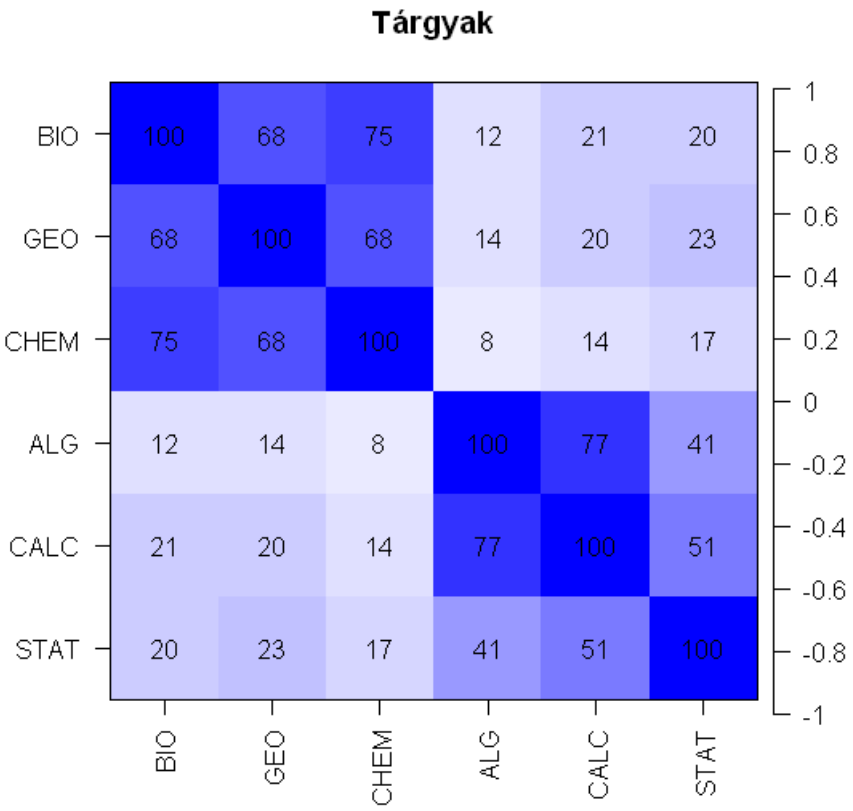
Ha viszont a tantárgyak kedveltségét nézzük, akkor a 4.21 ábra táblázata szerint kicsi a korreláció a faktorok között, ezért ebben az esetben ortogonális forgatás is elég.

Összehasonlítás

Mivel itt a szokottnál is több lehetőség van a faktorok meghatározására, ezért érdemes röviden kitérni a módszerek összehasonlítására.

Ha a futási idő a fő szempont, akkor a tapasztalt sorrend (a részvény-adatainkon tesztelve, 5 faktorral):

1. factanal
2. fa(fm=”pa”)
3. fa(fm=”ml”)



4.18. ábra. A tantárgyak korrelációs diagramja

4. fa

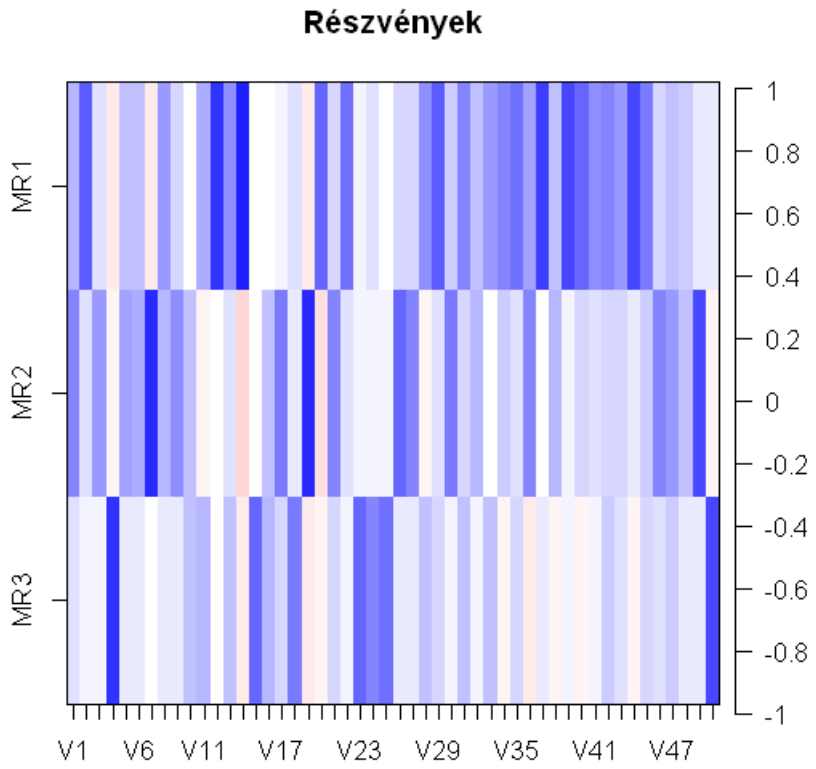
5. fa(fm=”gls”)

6. fa(fm=”wls”)

Az eltérés kb. 3 és félszeres az első és az utolsó között (0.2 sec vs 0.7 sec). Azonban legtöbbször nem a sebesség, hanem az eredmény pontossága a lényeges szempont. A sok módszer lényegében két különöböző eredményt adott: az előző felsorolásból az 1.,3. és 4. módszer a ”klasszikus” megoldást, míg a 2., 5. és 6. eljárás egy másikat. Az eltérés persze nem meglepő, hiszen láttuk, hogy a faktoranalízis feladatának megoldása nem egyértelmű. Ha a forgatást is megvalósítjuk, akkor a különbség értelemszerűen eltűnik. A gyorsasági sorrend kicsit megváltozik (varimax forgatásra):

1. factanal

2. fa(fm=”pa”)



4.19. ábra. A részvények korrelációs diagramja a faktorok szerint

```
                        MR1   MR2   MR3
SS loadings           10.06 7.16 4.68
Proportion Var         0.20 0.14 0.09
Cumulative Var         0.20 0.34 0.44
Proportion Explained   0.46 0.33 0.21
Cumulative Proportion  0.46 0.79 1.00

 With factor correlations of
     MR1  MR2  MR3
MR1 1.00 0.72 0.68
MR2 0.72 1.00 0.59
MR3 0.68 0.59 1.00
```

4.20. ábra. A részvények oblimin forgatással kapott faktorai és korrelációik

3. fa(fm=”ml”)

```
                        ML2  ML1
SS loadings            2.14 1.84
Proportion Var         0.36 0.31
Cumulative Var         0.36 0.66
Proportion Explained   0.54 0.46
Cumulative Proportion  0.54 1.00

 With factor correlations of
     ML2  ML1
ML2 1.00 0.21
ML1 0.21 1.00
```

4.21. ábra. A tárgyak kedveltségére oblimin forgatással kapott faktorok és korrelációik

4. fa(fm="gls")

5. fa(fm="wls")

6. fa

Érdekes, hogy az alap 'fa' kivételével inkább kevesebb, mint több idő kell a forgatott megoldás megtalálásához.

A fentiek alapján összefoglalhatjuk a tanulságokat: érdemes a beépített 'factanal' függvénnyel kezdeni az elemzést. Ha működik a maximum likelihood becslés és kielégítő eredményt ad a varimax (ortogonális) forgatás , akkor már csak az optimális faktorszámot kell megkeresnünk, például az nFactors csomag módszereivel. Ha viszont további számításokra van szükség (például más forgatásokra is kíváncsiak vagyunk), akkor érdemes a psych csomaghoz és az ott megtalálható számos opció egyikéhez fordulnunk.

5. fejezet

Többdimenziós regresszió

5.1. Bevezető

A rejtett változókkal való modellezés továbbfejlesztése a regresszióanalízisnek és azoknak a modelleknek, amik a klasszikusnak mondható főkomponens és faktoranalízisben valamint kanonikus korrelációban lelhetőek fel.

A rejtett változókkal való modellezés röviden a következő módon foglalható össze: "Azt feltételezük, hogy a megfigyelhető változók közt azért van korreláció, mert a függő változók mögött olyan közös változók vannak, aminek esetleg véletlen hibával ugyan, de mindegyikük a függvénye."

Ebben a részben három modell numerikus kezelésének lehetőségeire térünk ki.

Az első részben bemutatjuk a PLS modellt (5.2), ami arra alkalmas, hogy háttérkomponensek feltételezése mellett adatszegény helyzetben regressziót alkalmazzunk. A második részben (5.3) a PATH modellezéssel foglalkozunk. Ez arra alkalmas, hogy sok lehetséges magyarázó változó mellett egy feltételezett struktúrában a függés modellezését egyszerű regressziókra vezethessük vissza. Végül a harmadik részben (5.4) a SEM modelleket ismertetjük. Ezek a modellek bizonyos értelemben a legáltalánosabb látens változókkal vett lineáris modelljei lehetnek az adatainknak.

5.2. Parciális regresszió

5.2.1. Miért van szükség a PLS modellre?

A lineáris regresszió módszerével megoldható feladatok esetén — klasszikus esetben — azt feltételezzük, hogy p magyarázó változóra és 1 magyarázandó változóra n megfigyelésünk van. Továbbá azt is feltételezzünk, hogy a célváltozó értékeit az $n \times 1$ méretű Y vektorban, a magyarázó változók értékeit (egy csupa 1-esekből álló oszloppal esetleg kiegészítve) az $n \times p$ méretű X mátrixban összefoglalva, továbbá β-val jelölve a magyarázó változók (és az esetleges konstans) szorzótényezőjéből képzett $p \times 1$ méretű vektort, a

$$Y = X\beta + e$$

egyenlőség teljesül. Az itt szereplő $n \times 1$ méretű e vektor az $\varepsilon_1, ..., \varepsilon_n$ hibaváltozók méréskori értékeit tartalmazza. Ezek az ε változók a célváltozók megfigyelt értékeit terhelik. Feltételezzük róluk, hogy függetlenek és azonos eloszlásúak. A szórásuk egy ismeretlen σ_ε, és az eloszlásukról gyakran azt is feltesszük, hogy az a normális eloszlás.

A fenti egyenlet legkisebb négyzetek feltételezésével vett megoldása a $\hat{\beta} = (X^T X)^{-1} XY$ az $(X^T X)^{-1}||Y - X\hat{\beta}||/(n-p)$ értékkel becsült kovarianciával. Mint látható, e képletek alkalmazhatóságához szükséges az $(X^T X)^{-1}$ invertálhatósága. Aminek elégséges feltétele, hogy az $X^T X$ teljesrangú legyen. Azaz az invertálhatósághoz legalább annyi megfigyelésre van szükség, mint ahány becsült paraméter van. Ez a feltétel statisztikai szempontból igen szűkös feltétel. Hiszen a $\hat{\beta}$ becslés varianciája a $\sigma^2(X^T X)^{-1}$, aminek a nagyságrendje $\mathcal{O}(1/n)$ és ezen lényegesen az sem változtat, hogy a σ_ε szórást becsülni kell. A feladat persze elvileg akkor is megoldható, ha az $X^T X$ szinguláris: általánosított inverz segítségével az eredmények igen gondos értelmezése mellett.

Ugyanakkor a gyakorlatban igen tipikus, hogy a p nagy és az n kicsi. Azaz, hogy egy-egy egyedre (esetre) vonatkozóan az esetszámhoz viszonyítva viszonylag nagyszámú mérés áll rendelkezésre. Például, mert több egyed megvizsgálása túlzottan költséges volna. Vagy, mert esetleg nincs is, nem is ismert néhánynál több eset.

A fejezet végén két példát ismertetünk. Mindkettő jól illusztrálja a nagy p és a kis n esetét. Az egyik esetben a PET szálak infravörös képe alapján a szál sűrűségét akarják megadni. A másik esetben olivaolajok kémiai és élvezeti értéke közti kapcsolatot keresik. A PET szálak esetén a spektrométer igen nagy mennyiségben ontja egy-egy mintával kapcsolatban az adatokat. Az általunk feldolgozandó adatsorban 1-1 szálra vonatkozóan 268 spektrális adat van. Ugyanakkor a szálfajták száma véges. Esetünkben 28. Vagyis ha ehhez a feladathoz akarnánk 'klasszikus' módon nyúlni, akkor egy 268×268 méretű, legfeljebb 28 rangú mátrixot kellene invertálni. Az oliva esetén a probléma statisztikai szempontból hasonló, noha az elvileg még megoldható volna a klasszikus módon. Ott

ugyanis $n = 16$ különböző oliva van, és azoknak a szempontoknak a száma, amikből egy oliva értékelődik (szubjektívak illetve kémiai-fizikaiak) $p = 11$. Ebben az esetben azt reméljük, hogy az a PLS módszer amit most ismertetünk, megfelelő értelmezés mellett stabilabb modellt szolgáltat. [25]

A parciális legkisebb négyzetek módszere úgy oldja meg a 'nagy p, kis n' problémát, hogy a regressziót nem közvetlen a mért magyarázó változók szerint veszi. Hanem előbb néhány látens (háttér) változót képez a mérhető magyarázó változókból, majd a célváltozóknak ezekre a látens változókra vonatkozó regresszióját tekinti.

A PLS módszer tehát igen hasonlít arra, mintha előbb vennénk a magyarázó változók főkomponenseit, vagy egy faktormodell szerinti faktorait, utóbb pedig a célváltozóknak ezekre a főkomponensekre, faktorokra nézve vennénk a regresszióját. Csakhogy míg a főkomponensek, faktorok — a modell és a becslésük szerint is — kizárólag a magyarázó változóktól függnek, addig a PLS látens változói a célváltozótól is függenek. Ugyanakkor a PLS látens változó rendszere a kanonikus korreláció jobb és baloldali faktoraival sem azonos. Ugyanis a kanonikus korreláció faktorai a szimmetrikus szerepű célváltozók és magyarázó változók korrelációit közelítik. Mert a kanonikus korreláció esetén annak a közös tényezőnek a megjelenítése a cél ami a két változócsoportban egyszerre van jelen. Ugyanakkor a PLS látens változóit azzal a céllal konstruáljuk, hogy a segítségükkel a magyarázó változók alapján a célváltozók értékét modellezzük. Tehát a PLS konstrukciójakor egyfelől megmarad az aszimmetria a magyarázó és a magyarázott változó közt. Másfelől a modellezendő nem a közös rész, hanem a célváltozó értéke.

5.2.2. A PLS komponensek definíciója

Legyenek $Y_1, Y_2, \ldots, Y_k$ a magyarázott célváltozók és legyenek $X_1, X_2, \ldots, X_m$ a magyarázó változók. Legyen egy $p \leq m$ esetén $T_1, ..., T_p$ az X-ek egy-egy (egyelőre nem definiált) lineáris kombinációja. Tekintsük $j = 1, ..., k$ esetén $i = 1, ..., n$-re az

$$Y_{j,i} = \beta_{j,0} + \beta_{j,1}T_{1,i} + \beta_{j,2}T_{2,i} + ... + \beta_{j,p}T_{p,i} + e_{j,i}$$

regressziós egyenleteket, ahol a $\beta_{j,0}, \beta_{j,1}, ..., \beta_{j,p}$ ismeretlen együtthatók, és az $e_{j,i}$ a regressziós hibatag. Az Y_j változó i. komponensének a becslése $i = 1, ..., n$-re a j. regresszióval nyert $\hat{\beta}_{j,0}, ..., \hat{\beta}_{j,p}$ becslések alapján a

$$\hat{Y}_{j,i} = \hat{\beta}_{j,0} + \hat{\beta}_{j,1}T_{1,i} + \hat{\beta}_{j,2}T_{2,i} + ... + \hat{\beta}_{j,p}T_{p,i}.$$

Az így nyert becsléseket nevezik *pls regresszió*nak, pls modellnek, és a T_k, $k = 1, \ldots, p$ lineáris kombinációkat pedig a modell *pls komponensei*nek.

Látható, hogy a pls regresszió a magyarázó változókon vett regresszió helyett a komponensek szerint vett regresszió.

Tekintsük részletesebben a $k = 1$ változós esetet! Ebben az esetben a korábban Y_1-el jelölt változót elég Y-nal jelölni. Mivel a j-re, mint az Y-ok indexére a továbbiakban nincs szükség, a j-t a továbbiakban az X-ek indexelésére, és a szintén fölöslegessé váló k-t pedig, — az i mellett — a sorok indexelésére fogjuk használni.

Egy olyan algoritmust mutatunk, ami a komponensek egy lehetséges definíciója. Ez az algoritmus a komponenseket egy iteratív, lépésenkénti módszerrel határozza meg. Mindegyik lépésben *egy* új T komponens keletkezik. A lépések során az Y és az X_j, $j = 1, ..., m$ értéke változni fog. Azt, hogy az adott változó hányadik lépésben kapott értékéről van szó, egy új, zárójelbe tett index fogja jelölni.

Legyen $Y_{(0)} = Y$, azaz az Y-nak a 0. lépés utáni értéke legyen az a vektor amit a megfigyelt célváltozó értékek alkotnak, és legyen hasonó módon $X_{j(0)} = X_j$ a $j = 1, ..., m$-re. A 0 itt nem egy lecsúszott függvény argumentum, hanem annak a jele, hogy a 0. lépésnél, az első lépés előtt tartunk. A 0. lépés különbözik a többitől.

Legyen $Y_{(1),i} = Y_{(0),i} - \sum_{k=1}^{n} Y_{(0),k}/n$ és $j = 1, ..., m$-re az $X_{j(1),i} = X_{j,i} - \sum_{k=1}^{n} X_{k,j}/n$.
Azaz centráljuk az Y és az X változókat!
Az $Y_{(1)}$ és $j = 1, ..., m$-re az $X_{j(1)}$ vektorok a 0. lépés termékei.
Itt az (1) index azt jelöli, hogy már az első lépés elejénél tartunk.

Tegyük fel, hogy már az ℓ. lépésnél tartunk.
Megmutatjuk, hogy hogyan kell képezni a T_ℓ komponenst, és hogy hogyan kell előállítani annak segítségével az $Y_{(\ell+1)}$ és a $X_{j(\ell+1)}$, $j = 1, ..., m$ vektorokat.

Legyen $j = 1, ..., m$-re az $X_{j(\ell)} b_{j\ell}$ az $Y_{(\ell)}$ regressziója az $X_{j(\ell)}$-n, azaz legyen

$$b_{j\ell} = (X_{j(\ell)}^T X_{j(\ell)})^{-1} X_{j(\ell)}^T Y_{(\ell)}.$$

Mint látható, itt egy *egydimenziós* regresszióról van szó!
Legyen a T_ℓ ezeknek az $X_{j(\ell)} b_{j\ell}$ változóknak a $w_{\ell,1}, ..., w_{\ell,m}$ súlyokkal vett átlaga, azaz legyen

$$T_\ell = \sum_{j=1}^{m} w_{\ell,j} X_{j(\ell)} b_{j\ell}.$$

Az $Y_{(\ell+1)}$ az az $Y_{(\ell)}$-beli információ, amit az így definiált T_ℓ nem magyaráz meg, azaz a

$$P_\ell = I - T_\ell (T_\ell^T T_\ell)^{-1} T_\ell^T$$

jelöléssel legyen:

$$Y_{(\ell+1)} = Y_{(\ell)} - T_\ell (T_\ell^T T_\ell)^{-1} T_\ell^T Y_{(\ell)} = P_\ell Y_{(\ell)}.$$

Az $X_{j(\ell+1)}$, $j = 1, ..., m$ pedig legyen az az információ amely még nem volt benne a T_ℓ-ben, azaz legyen:

$$X_{(\ell+1)} = X_{(\ell)} - T_\ell (T_\ell^T T_\ell)^{-1} T_\ell^T X_{(\ell)} = P_\ell X_{(\ell)}.$$

Ezután az algoritmus az $\ell + 1$. lépéssel folytatódik mindaddig, míg a kívánt komponens számot el nem érjük.

A PLS komponensek ugyanezen definiciója megadható nem szukcesszív módon is. Azért választottuk ezt a lépésenkénti módszert, mert így a T komponensek tulajdonságai jobban látszanak. Többek közt az például, hogy a T komponensek korrelálatlanok.

5.2.3. PLS modellek a gyakorlatban

A 'pls' [26] kiegészítés programjait használjuk. A program nem tartozik az **R**-project alaprendszeréhez. Ezért külön installálni kell. És a

```
require(pls)
```

paranccsal be kell tölteni. Betöltéskor megkapjuk a

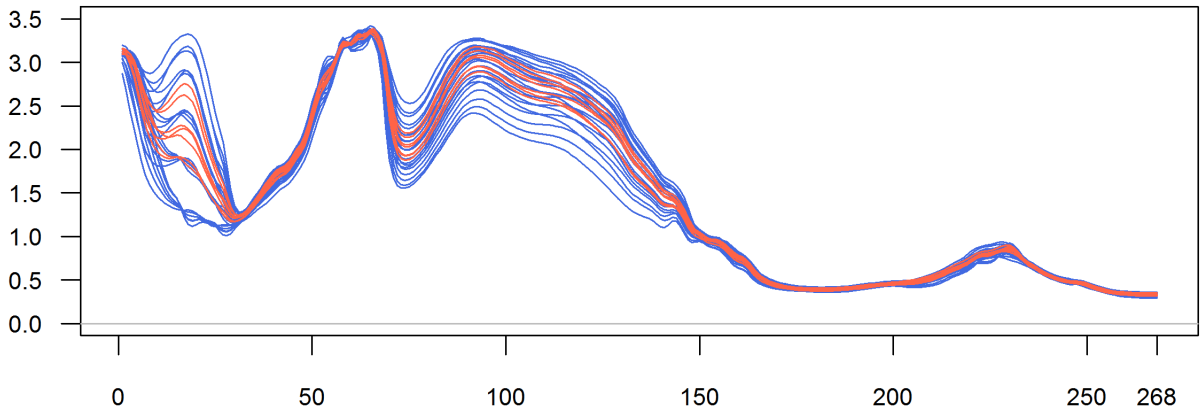
```
The following object(s) are masked from 'package:stats': loadings
```

üzenetet is. Ez nem problémát jelez, hanem azt, hogy a továbbikban a 'stats' csomag 'loadings()' rutinja helyett a 'pls' csomag 'loadings()' rutinja fog elindulni a 'loadings()' parancs hatására. De a 'pls::loadings()' úgy van megírva, hogyha az argumentumába "factanal" vagy "princomp" osztályú változó kerül, akkor átadja a végrehajtást a 'stats::loadings()' számára.

NIR spektrumadatok feldolgozása

Példaként a 'yarn' adathalmazait dolgozzuk fel. Ez a 'data.frame' egy 3 elemű lista. A 'yarn' első eleme egy 28×268 méretű adatmátrix, ami 28 PET fonál esetén mutatja a szálak NIR (near infra red, közel infravörös) képét. Ezt, a minden szálra 268 elemű intenzitásvektort röviden NIR spektrumnak fogjuk nevezni. A második elem egy lista ami mind a 28 szálra megadja a fonal tényleges sűrűségét. A harmadik elem pedig egy indikátor, ami 21+7 arányban felbontja a spektrumokat egy 21 elemű 'tanuló' és egy 7 elemű 'teszt' csoportra.

Az 5.1 ábra a 28 spektrum együttes képét mutatja: kékek a ‘tanuló’ spektrumok, pirosak a ‘teszt’ halmaz adatai. Látható, hogy a ‘teszt’ halmaz spektrumai igencsak átlagosak...



5.1. ábra. Szálsűrűség becsléséhez használható 28 NIR spektrum együtes képe

Futtassuk le a

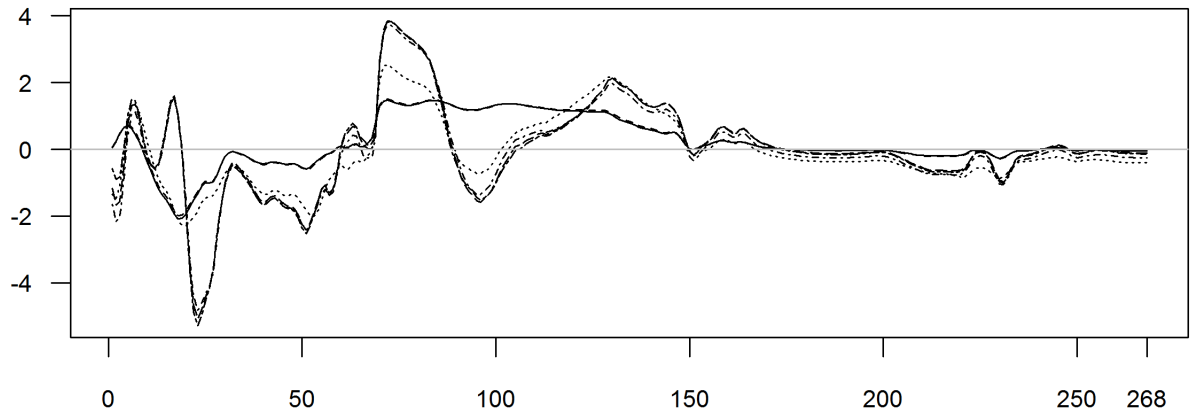
```
data(yarn)
M <- plsr(density ~ NIR, ncomp=6, data = yarn, method="oscorespls")
summary(M)
coefplot(M, ncomp = 1:6)
predplot(M,ncomp=1)
```

parancsokat. Két ábrát és egy táblázatot nyerünk.

A paraméterezésen látható, hogy ’ncomp=6’ komponenst kértünk. A módszer megadásával a ’plsr()’ rutint arra utasítottuk, hogy az 5.2.2 elméleti részben ismertetett ortogonális vetítési módszerrel állítsa elő a komponenseket. Az első táblázat a ’summary()’ kiírása:

```
Data:   X dimension: 28 268
        Y dimension: 28 1
Fit method: oscorespls
Number of components considered: 6
TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X           46.83    98.38    99.46    99.67    99.85    99.97
density     98.12    98.25    99.64    99.97    99.99    99.99
```

Ezen a kiíráson látszik, hogy már az első három komponens több mint 99%-ban interpretálja az X, azaz a spektrum és a fonálsűrűség adatokat. Érdekes megfigyelni, hogy az X-re nézve a második komponens több információt tartalmaz mint az első: 47% a 52%-al szemben. A komponensképzés szukcesszivitása folytán, ha az 'ncomp=6' helyett kisebb vagy nagyobb értéket választottunk volna, akkor ugyanezeket a 'variance explained' értékeket kaptuk volna, csak esetleg az alábbi sort még nagyobb számokkal kiegészítve. Az 5.2 ábra azt mutatja, hogy a komponensek a spektrumnak mely részeitől függnek.



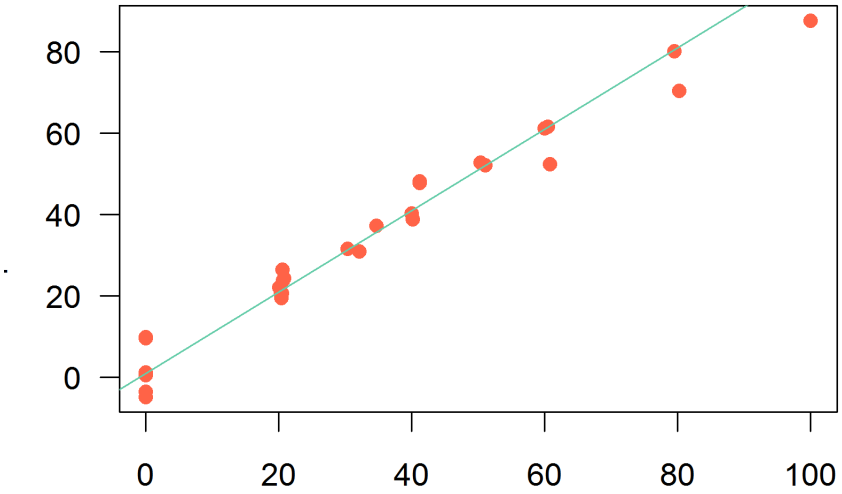
5.2. ábra. A 6 PLS komponens loading értékei a frekvencia függvényében

Az 5.3 ábrán 28 piros pötty látható, mert 28 szálsűrűséget vizsgáltunk. Az ábra x tengelye azért fut 0-tól 100-ig, mert a megadott sűrűségértékek is ebben az intervallumban változtak. A kék vonal felel meg a hibátlan predikciónak.

Marketing adatok feldolgozása

A 'pls' csomagban található 'oliveoil' adatsor 16 olivaolajra vonatkozóan tartalmaz adatokat. Ez egy 11 oszlopos 'data.frame', és mint az a sorok neveiből kiolvasható, az olajok közül 5 görög, 5 olasz és 6 spanyol.

Az egyes oszlopokban a 16 oliva különböző szempontok szerinti jellemzése található. Ezek közül 5 fizikai-kémiai és 6 érzékelési jellemző. A kémiai (chemical) tulajdonságok: Acidity', 'Peroxide', 'K232', 'K270', 'DK' . Az érzékelési (sensory) tulajdonságok: 'yellow', 'green', 'brown', 'glossy', 'transp', 'syrup'. A kémiai változók neveinek kezdete:



5.3. ábra. A szálsűrűségnek a mért és az egy komponensű PLS modell szerinti értékei

'chemical.', az érzékelésieké 'sensory.'. Ennek az elnevezési konvenciónak az eljárások paraméterezésekor van jelentősége. Mindegyik folytonos skálájú adat.

Futtassuk az alábbi programsorokat!

```
data(oliveoil)
M <- plsr(sensory~chemical,ncomp=4,scale=TRUE,data=oliveoil)
summary(M)
loadings(M)
W<-M$scores
plot(W[,1],W[,2],t='n')
text(W[,1],W[,2],labels=rownames(W),col='olivedrab')
```

A program az 'oliveoil' adatok aktiválása után egy 'pls' modellt illeszt. Az eredmény objektumra alkalmazott 'summary()' eredménye a következő:

```
Data:   X dimension: 16 5
        Y dimension: 16 6
Fit method: oscorespls
Number of components considered: 4
TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps
X         57.79    79.09    95.56    98.25
yellow    45.88    50.53    53.15    53.74
```

```
green     40.33    47.21    47.89    48.52
brown     29.80    52.90    77.10    78.39
glossy    47.14    52.22    52.27    52.27
transp    43.57    44.98    44.98    45.15
syrup     40.15    50.35    52.29    55.50
```

Látható, hogy a módszer a 'transp' áttetszőségi tulajdonságot közelíti a legrosszabbul.

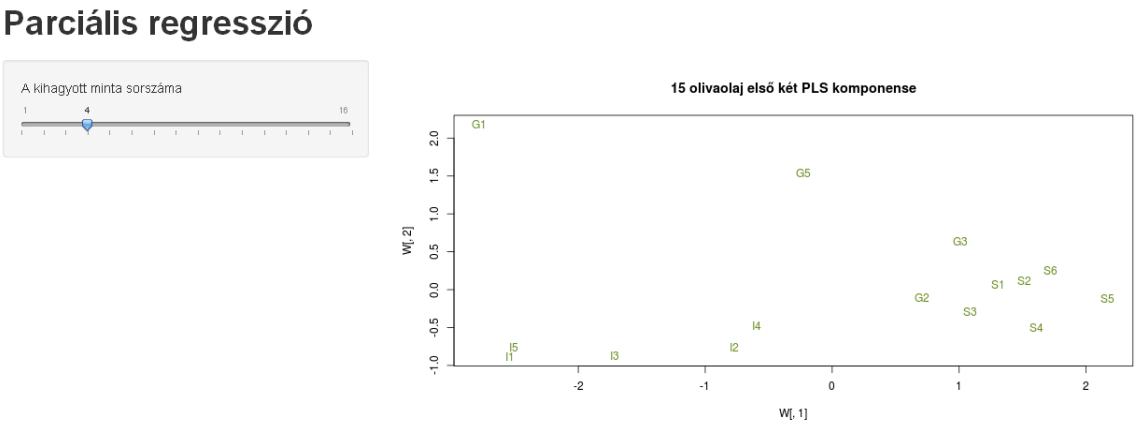
Az alábbi táblázat a skálázás nélküli loadingokat mutatja. A hiányos kiírás oka, hogy az adott pozícióba .1-nél kisebb szám kerülne. Így pontosan látszik, hogy az első komponens lényegében a 'Peroxide', a második az 'Acidty' és a 'K232' stb.

```
Loadings:
         Comp 1 Comp 2 Comp 3 Comp 4 Comp 5
Acidity           0.961 -0.469
Peroxide -0.999
K232              0.306  0.891  0.215
K270                            -0.974 -0.110
DK                                     -0.994

               Comp 1 Comp 2 Comp 3 Comp 4 Comp 5
SS loadings     1.003  1.029  1.020  1.001  1.000
Proportion Var  0.201  0.206  0.204  0.200  0.200
Cumulative Var  0.201  0.406  0.610  0.811  1.011
```

Az 5.4 ábrán az olajoknak a termőterületet is jelző kódjai láthatóak. A G1,...,G5 kódok görög, az I1,...,I5 kódok olasz, és az S1,...,S6 kódok pedig spanyol olivaolajokat jelölnek. A kódok helyét azok a szkórok adják amit a kémiai adatokból képzett első két PLS komponens határoz meg. Érdekes, hogy vajon milyen másodlagos információ okozza, hogy a 16 olaj a normalizált score térben lényegében a termőterületek szerint csoportosul.

Az adatbázis PLS komponenseit animációval is megvizsgálhatjuk. A címen sorra kihagyhatjuk a 16 oliva egyikét és megkapjuk a megmaradó 15 vektor első két PLS komponensét. Az 5.5 ábra egy példa a lehetséges eredmények közül.

5.4. ábra. A 16 olivaolaj érzéki értékelésének első két PLS komponens szerinti szkórja

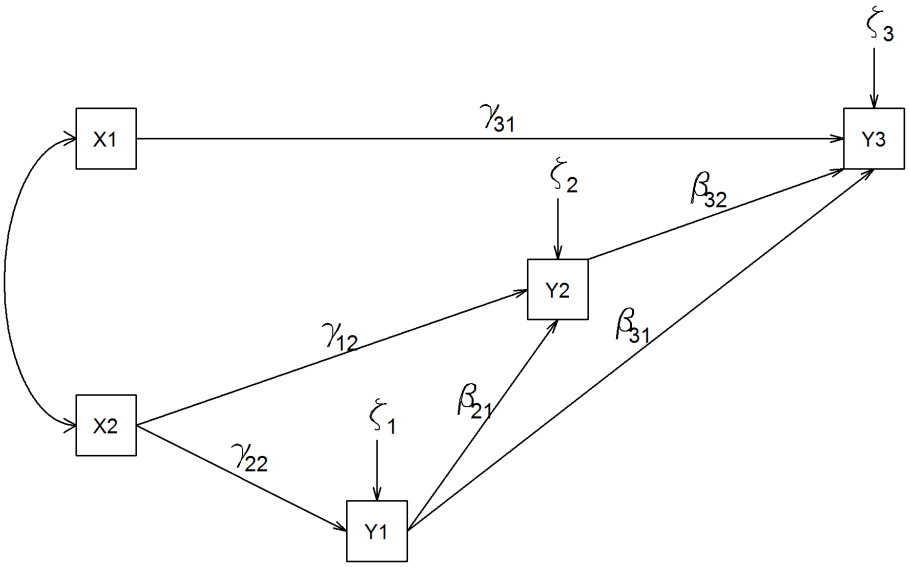


5.5. ábra. A 16 olivaolaj közül egyet kihagyva kapott két PLS komponens szerinti szkórok

5.3. A path analizis

A path analízis módszerét magyarul út- vagy pálya- analízisnek de — talán jobban idézve az eszköz hangulatát — ösvény módszernek is nevezhetjük. A szónak az adott módszer vonatkozásában nincsen általánosan elfogadott fordítása. Elvileg gyalogutat, útvonalat, pályát stb jelent. A helyes kiejtése nagyon kilóg a magyar beszédből. Ám ha népi etimológiával — az angol szót magyar sportszóvá ferdítjük — nem képzavar, ha passz

analízisről beszélünk. Hiszen e módszer keretében pont azt vizsgáljuk, hogy egyik változó a másiknak hogyan passzolja az információt...



5.6. ábra. Egy általános path diagram

Az elnevezés abból a szemléletes képből származik, hogy a magyarázandó változó véletlensége nem az összes magyarázó változóból származik közvetlenül. Hanem úgy, hogy egyes magyarázó változók más magyarázó változókat magyaráznak, majd az így részben megmagyarázott változók magyaráznak újabbakat stb és végül a célváltozót ezek a részben magyarázott változók magyarázzák.

Az angol elnevezés tehát azt fejezi ki, hogy a path módszerrel előállitott modell esetén több lépésben, részben megmagyarázott változókon keresztül jutunk el a célváltozó magyarázatához.

A path analizis a legegyszerűbb formájában nem több, mint egy egyszerű regresszió sorozat. Történetileg először a [37] cikkben jelent meg.

Igen fontos, hogy már a path analízis módszerével való ismerkedés kezdetén felhívjuk a figyelmet a következő két fontos negatív állitásra. Az egyik, hogy az adatoknak a path modell globálisan nem feltételen pontosabb modellje mint egy egyszerű regresszió. A másik az, hogy az ha a modell diagramjában egy nyilat látunk az egyik változóból a másikba mutatni, akkor az nem feltételen jelenti, hogy az egyik változó által leírt jelenség a másik által leírtnak bármely mértékben is az oka volna.

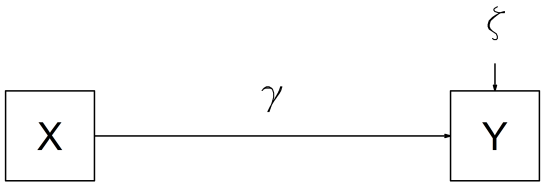
Ugyanakkor, mint az az idézett őscikk címéből is látható, a módszert klasszikusan összekapcsolják az 'ok' illetve az 'okság' keresésével. Ez indokolt is, de csak olyan mértékben, hogy az *adott struktúra mellett*, az *adott mértékben* oka az egyik változó a másiknak. Ugyanis a modellt legtöbbször adott struktúra mellett illesztjük: a módszer számára a struktúra és a struktúrában szereplő változók kovarianciája (korrelációja) bemenő adat. Előfordul ugyan, hogy az adatokra többféle modellt illesztünk. Ámde az adatokra tipikus esetben több modell is ugyanolyan jól illeszkedik. Egyébként pont ez az utóbbi tény az amit a módszer eredményeit túlzott kétkedéssel fogadók kiemelnek.

5.3.1. A PATH történet

A path modell illesztés referencia eljárásának a KG. Jöreskog és D. Sörborn által szerkesztett, a keletkezési dátumát tekintva a 70-es évekre datálható LISREL program tekinthető [21]. Ennek a programnak valamilyen variánsa szinte minden fontosabb statisztikai programcsomagban fellelhető. A path módszerek egyik gyökere a már említett egymásután csatolt regresszió sorozat, ami mint majd láthatjuk csak különleges esetekben illesztheti az indikátorok korreláció mátrixát hibátlanul. Ezért már a kezdetekkor felmerült a globális illesztés igénye, aminek általános módszere például a legkisebb négyzetek módszere vagy a maximum likelihood technika. A path módszerek másik gyökere a főkomponens analízis és a faktoranalízis. Mindkettő tekinthető úgy mint a látens változókkal való modellezés prototípusa. Az **R** -ben három fontosabb program alkalmas path modellek illesztésére: a 'sem', a 'lavaan' és a 'lava'. Ezek közül sajnos csak az utóbbinak van grafikus kimenetele. A 'sem' programmal úgy tudunk képeket előállítani, hogy a csomag 'path.diagram()' eljárásával egy olyan programot készíttetünk, amit a 'graphviz', szabad grafikai program értelmezni tud. Az 5.16 ábra egy példa a 'lava' által készitett grafikára, az 5.17 ábra pedig egy 'sem'+'graphviz' eszközökkel létrehozott grafika.

5.3.2. A PATH fogalmak

Az 5.7 ábra az egyik legegyszerűbb path diagramot mutatja. Ezen az látható, hogy három változó van a rendszerben. A megfigyelt X és Y, valamint az Y strukturális hibája ζ. A megfigyelt változókat a path diagramokon négyzetekbe szokás irni. A nem-megfigyelt modell által feltételezett strukturális változókat pedig körökbe, vagy ellipszisekbe. Az egyéb változók, mint itt az szereplő ζ strukturális hiba, keretezés nélkül nyíllal csatlakoznak a megfelelő megfigyeléshez.



5.7. ábra. Az egyváltozós lineáris regresszió path diagramja

Az 5.7 ábrának megfelelő egyenlet tehát a következő:

$$Y = bX + a + \zeta,$$

vagyis egy egyszerű regresszió az X magyarázó változóval az Y célváltozóval és a ζ hibával. A path modell a lényegét tekintve kovariancia modell. Van olyan variánsa amelyik a változók várható értékének a modellezését is tartalmazza, de mi a továbbiakban az egyszerűség kedvéért csak azokat az eseteket vizsgáljuk, amikor minden a modellben szereplő változó várható értéke 0. Egy további egyszerűsítést is alkalmazunk: feltesszük, hogy mindegyik változó standardizált, tehát hogy mindegyiknek a szórása 1.

A mondott egyszerűsítések mellett — felhasználva a legkisebb négyzetek módszerével vett regresszió megoldó képletét — közvetlenül adódik, hogy:

$$\gamma = b = \mathrm{cor}(Y, X),$$

és persze $a = 0$.

Vegyük a 'datasets::ability.cov' adathalmazt. Ez egy 6×6-os kovariancia mátrix 112 személy vizsgálata alapján, a következő tulajdonságokra vonatkozóan. Általános intelligencia szint ('general'), kép kiegészítési képesség ('picture'), diagamok értése ('blocks'), útvesztők megoldása ('maze'), olvasás értése ('reading'), szókincs ('vocab'). Az 'ability.cov' változó maga egy 3 elemű lista. A harmadik eleme ('$n.obs') mutatja, hogy az adatok 112 megfigyelésből származnak. A '$center' eleme tartalmazná elvileg, hogy mennyi volt az egyes koordináták átlaga. De sajnos ez, ebben a változóban azonosan nulla. Ami arra utal, hogy nincs kitöltve. Igaz, ha tényleg emiatt végig 0 a '$center' komponens, akkor helyesebb lett volna 'NA' értékekkel feltölteni. A '$cov' komponens tartalma:

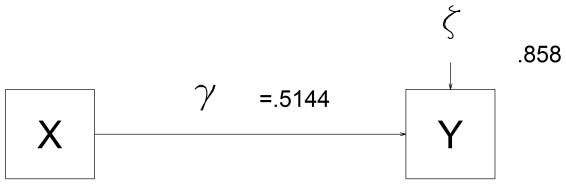
```
         general picture  blocks   maze reading   vocab
general   24.641   5.991  33.520  6.023  20.755  29.701
picture    5.991   6.700  18.137  1.782   4.936   7.204
blocks    33.520  18.137 149.831 19.424  31.430  50.753
```

```
maze      6.023   1.782  19.424 12.711   4.757   9.075
reading  20.755   4.936  31.430  4.757  52.604  66.762
vocab    29.701   7.204  50.753  9.075  66.762 135.292
```

Ha a választott adathalmazból Y változóként a 'general' általános inteligencia szintet vesszük, és magyarázó változónak a 'vocab' szókincs változót, akkor a γ értéke a két változó korrelációja vagyis az

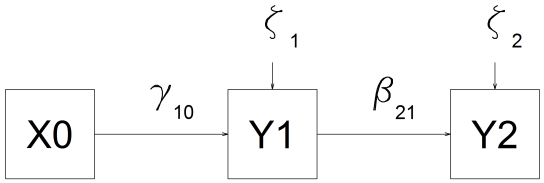
```
r<-ability.cov$cov
r["general","vocab"]/sqrt(r["general","general"]*r["vocab","vocab"])
```

utasítással megkapható $\gamma = .5144$ érték és a ζ együtthatója $\sqrt{1-\gamma^2} \approx .858$. Tehát a path diagram a megfelelő együtthatókkal kitöltve:



5.8. ábra. Az egyváltozós lineáris regresszió path diagramja a path együtthatókkal

Tekintsük a következő, összetettebb path modellt.



5.9. ábra. Az egyik legegyszerűbb rekurzív path modell

A path diagramokon az X-el és az Y-al jelölt változók közt alapvető funkcionális különbség van. Itt az első nyíl 'értékét' azért jelöli γ, a másodikét pedig β, mert az Y-ok közti együtthatók szokványos jelölése β, az olyan együtthatóké pedig, amik egy X-ből egy Y-ba mutatnak γ. Az indexek mindig azt jelölik, hogy *melyik sorszámúba* mutat a vektor, *melyik sorszámúból*. Ezek az indexek most elvileg fölöslegesek.

A path diagramokon azokat a változókat szokás X-el jelölni, amik un. exogén változók indikátorai. Azaz olyan változók megfigyelt értékei, amiket más változók nem magyaráznak. Ugyanakkor Y-al szokás jelölni azokat a változókat, amik egyszerre magyarázott

és magyarázó változók, azaz un endogén változók indikátorai.

A megnevezés, hogy egy változó *indikátor*, azt is jelenti, hogy az adott változót meg tudjuk figyelni. Az esetünkben az exogén és endogén változók hiba nélküli megfigyeléséről van szó. Az már inkább a SEM modellek 5.4 fejezet témaköre, ezért a későbbiekben fog előfordulni, amikor azt kell feltételeznünk, hogy a vizsgált jelenséget leiró exogén és endogén változókat csak hibával tudtuk megfigyelni. Ekkor majd a megfigyelt változókat (az indikátorokat), egy-egy Λ_y illetve Λ_x mátrixú lineáris leképezéssel kapcsoljuk a közvetlenül nem-megfigyelhető η endogén illetve ξ exogén változókhoz.

A fenti modell az alábbi egyenletrendszer teljesülését jelenti, a fölösleges indexeket elhagyva és annak feltétlezésével, hogy a szóbanforgó megfigyelt változók 0 várható értékűek és a szórásuk 1.

$$\begin{aligned} Y_1 &= \gamma X + \zeta_1 \\ Y_2 &= \beta Y_1 + \zeta_2 \end{aligned}$$

Az egyenletek alapján a γ, β együtthatók, és a ζ_1, ζ_2 hibák szórása az (Y_1, Y_2, X) vektor korreláció mátrixa alapján könnyedén meghatározható. Ugyanis a fenti ábra egyben azt is jelenti, hogy a X korrelálatlan az ζ_1-el és a ζ_2-vel is. Továbbá, hogy a ζ_2 az Y_1-el is korrelálatlan.

Helyettesítsük az Y_1 első egyenlet szerinti értékét a másodikba:

$$Y_2 = \beta\gamma X + \beta\zeta_1 + \zeta_2.$$

Szorozzuk meg ezt az egyenletet X-szel és vegyük a várható értékét. Felhasználva a korrelálatlanságokat és azt, hogy $D^2(X) = 1$ kapjuk, hogy:

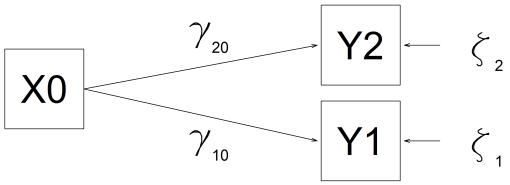
$$E(XY_2) = \varrho(X, Y_2) = \beta\gamma.$$

Vagyis a két változó közti *korreláció a* köztük vezető úton található *konstansok szorzata*. Vagyis a diagram változói közti korreláció meghatározható a path együtthatók alapján.

Ugyanakkor persze kérdéses a modell érvényessége. Hiszen mint az előbb láthattuk, $\gamma = \varrho(Y_1, X)$ és $\beta = \varrho(Y_2, Y_1)$. Tehát a modell csak akkor lehet az adataink legalább hozzávetőleg jó leírása, ha $\varrho(Y_2, X) \approx \varrho(Y_2, Y_1)\varrho(Y_1, X)$, ami általában persze nem érvényes.

Vegyük a következő, 5.10 diagram szerinti, összetettebb path modellt!
Ez a modell a következő egyenletrendszernek felel meg:

$$\begin{aligned} Y_1 &= \gamma_{10} X_0 + \zeta_1 \\ Y_2 &= \gamma_{20} X_0 + \zeta_2 \end{aligned}$$



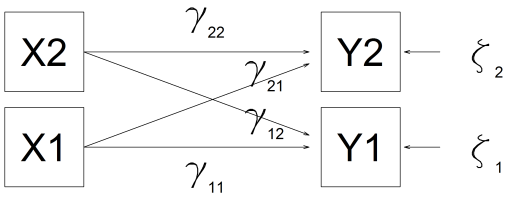
5.10. ábra. Két endogén változó közös exogén okkal

Felhasználva, hogy a diagram szerint a ζ_1 korrelálatlan az X_0-al és az Y_2 -vel, és hogy ζ_2 korrelálatlan az X_0-al és az Y_1 -el, a diagramról leolvasható az Y_1 és Y_2 korrelációja. Ugyanis összeszorozva a két egyenletet:

$$E(Y_1Y_2) = \text{cor}(Y_1, Y_2) = \gamma_{10}\gamma_{20}.$$

Vagyis most az Y_1 és az Y_2 közti korrelációt úgy kapjuk meg, hogy azon úton, ami a két változó közt vezet, — előbb az irányítással szemben, majd pedig azzal megegyező irányba — összeszorozzuk az 'útbaeső' path konstansokat.

A következő 5.11 path diagram annyiban különbözik az előzőtől, hogy két exogén változó van a modellben.



5.11. ábra. Két endogén változó közös exogén okkal

Ez a modell az

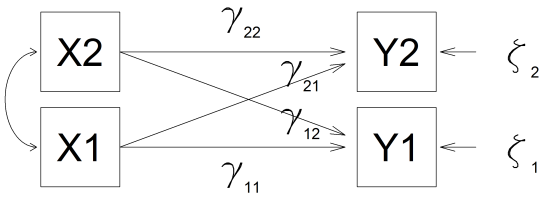
$$\begin{aligned} Y_1 &= \gamma_{11}X_1 + \gamma_{12}X_2 + \zeta_1 \\ Y_2 &= \gamma_{21}X_1 + \gamma_{22}X_2 + \zeta_2 \end{aligned}$$

egyenletrendszernek felel meg. Vizsgáljuk ebben az esetben is az Y_1 és Y_2 korrelációját!

$$\text{cor}(Y_1, Y_2) = E(Y_1, Y_2) = \gamma_{11}\gamma_{21} + \gamma_{12}\gamma_{22}$$

Vagyis azt kaptuk, hogy a két változó közti korreláció a két változó közt vezető *két* úton található path *együtthatók szorzatának összege*. Itt felhasználtuk, hogy $DX_1 = DX_2 = 1$ és hogy $\text{cor}(X_1, X_2) = 0$. Valamint azt, hogy a ζ-k és az X-k is korrelálatlanok.

Az előző modellben tehát többek közt azt tettük fel, hogy a két exogén változó, az X_1 és az X_2 korrelálatlan. Azt, ha két indikátor korrelált, a két indikátort jelölő négyzetet összekötő görbe nyíllal szokás jelölni, az 5.12 ábra szerinti módon.



5.12. ábra. Két korrelált endogén változó

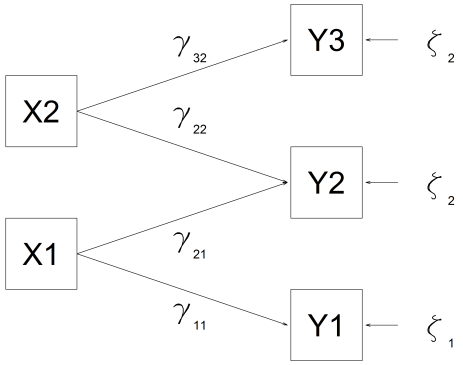
Ez a következő változtatást jelenti az $Y1$ és $Y2$ korrelációjának kiszámolásakor, ha $\mathrm{cor}(X_1, X_2) = \varrho$:

$$\mathrm{cor}(Y_1, Y_2) = E(Y_1, Y_2) = \gamma_{11}\gamma_{21}EX_1^2 + \gamma_{11}\gamma_{22}EX_1X_2 + \gamma_{12}\gamma_{21}EX_2X_1 + \gamma_{12}\gamma_{22}EX_2^2) =$$

$$= \gamma_{11}\gamma_{21} + \gamma_{11}\gamma_{22}\varrho + \gamma_{12}\gamma_{21}\varrho + \gamma_{12}\gamma_{22}$$

Vagyis a két változó közti korreláció ebben az esetben is a két változó közti utakon vett konstansok szorzatainak az összege úgy, hogy alkalmanként a két endogén változó közti kétirányú útszakaszt is igénybe vesszük.

Vegyük a lényegesen összetettebb, 5.13 ábra szerinti modellt.



5.13. ábra. Három megfigyelt endogén változó két megfigyelt exogén okkal

Ebben a modellben 3 endogén változót magyaráz 2 exogén változót, és azért érdekes, mert elvileg — a haladási irányok megszegésével — van út az $Y1$ és az $Y3$ változó közt,

ám a két változó mégis korrelálatlan!

A megfelelő egyenletrendszer az alábbi:

$$\begin{aligned} Y_1 &= \gamma_{11}X_1 + \zeta_1 \\ Y_2 &= \gamma_{21}X_1 + \gamma_{22}X_2 + \zeta_2 \\ Y_3 &= \gamma_{32}X_2 + \zeta_3 \end{aligned}$$

Ennek alapján az rögtön látható, hogy az Y_1 és az Y_3 korrelációja nulla. Ugyanis:

$$\text{cor}(Y_1, Y_3) = E(Y_1, Y_3) = E\left((\gamma_{11}X_1 + \zeta_1)(\gamma_{32}X_2 + \zeta_3)\right) = 0$$

Vagyis, hiába vezet út az $Y1$ változóból az Y_2 változóba, a két változó korrelálatlan.

A fentiek alapján már látható, hogy ha az együttható szorzat-összegeket a következő három szabály szerint vett utakon számoljuk, akkor tényleg a korrelációt kapjuk két tetszőleges változó közt:

Minden figyelembe vett út

- minden csúcson legfeljebb egyszer menjen át,
- legfeljebb egy kétirányú szakaszt tartalmazzon,
- ha egyszer már az útszakasz irányításával egyező irányban haladt, akkor ne következzen benne a haladási iránnyal ellentétes irányítású szakasz.

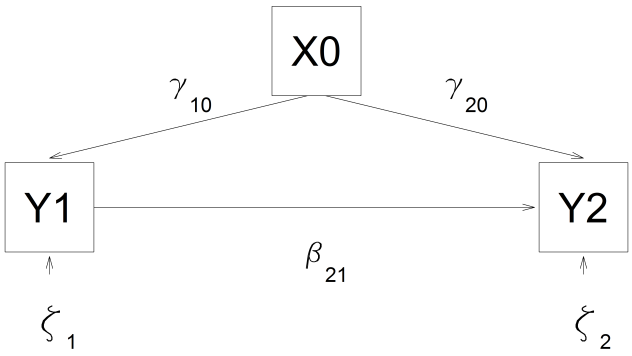
Végül vegyük az 5.14 ábrával leírt, egyszerű ámde mégis fontos speciális tulajdonságokkal bíró modellt. Ez a modell egy olyan modell, amiben egyrészt van olyan endogén változó ami nem csak magyarázott, hanem magyarázó változó is. Másrészt a modell exogén változója több endogén változót is magyaráz. Továbbá a diagramon van két olyan út — az $X_0 \to Y_2$ és az $X_0 \to Y_1 \to Y_2$ — ami ugyanazon két csúcspont közt vezet.

Az 5.14 ábrán bemutatott modell, az eddig vázolt szokásoknak megfelelően, az

$$\begin{aligned} Y_1 &= \gamma_{10}X_0 + \zeta_1 \\ Y_2 &= \beta_{21}Y_1 + \gamma_{20}X_2 + \zeta_2 \end{aligned}$$

egyenleteknek felel meg. Ezen egyenletek alapján, de akár a path diagramból kiindulva és az előző szabályt alkalmazva is, a változók közti korrelációk kiszámíthatóak:

$$\begin{aligned} \text{cor}(X_0, Y_1) &= \gamma_{10} \\ \text{cor}(X_0, Y_2) &= \gamma_{20} + \gamma_{10}\beta_{21} \\ \text{cor}(Y_0, Y_1) &= \beta_{21} + \gamma_{10}\gamma_{20} \end{aligned}$$



5.14. ábra. Két lehetséges irányított út az Y_2-be: az $X_0 \to Y_2$ és az $X_0 \to Y_1 \to Y_2$

Megjegyzendő, hogy itt például nem szerepel az $X_0 \to Y_2 \leftarrow Y1$ út. Ugyanis ezen az első szakasz olyan, hogy az irányítás a mozgás irányával egyező, ámde a következő irányítása már a mozgással ellentétes irányú volna.

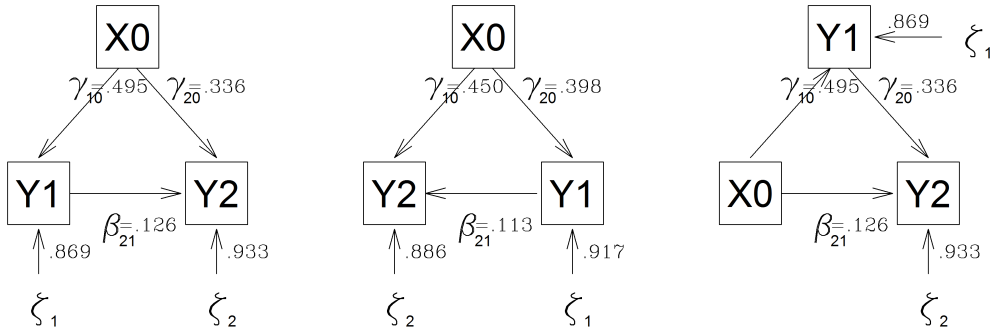
Ha a mérések korreláció mátrixa például a következő:

$$\mathrm{cor}(Y_1, Y_2, X) = \begin{pmatrix} 1 & .292 & .495 \\ .292 & 1 & .398 \\ .495 & .398 & 1 \end{pmatrix},$$

akkor a következő ösvény együtthatók adódnak: $\gamma_{10} = .495$, $\gamma_{10} = .336$, $\beta_{21} = .126$. Továbbá a ζ_1 szórása .869 (vagy ha úgy vesszük, hogy a ζ_1 szórása is 1, akkor ez az együtthatója), és a ζ_2-é pedig: .933 (ezeket az adatokat írtuk be az 5.15 ábra első modelljébe is).

Ha hasonló módon ugyanehhez a korreláció mátrixhoz illesztjük az 5.15 ábrán látható további két modellt is, akkor azt tapasztalhatjuk, hogy a korreláció mátrixot mindhárom modell hiba nélkül reprodukálja! Ez, az esetleg meglepő tény egyáltalán nem ritka eset. Igen széles körben vannak egyformán jól illeszkedő modellek, s ez a tény nem puszta érdekesség.

Három teljesen azonos strukturájú modellről van szó. Csak az változik, hogy a korábban felírt egyenletek egyes változóinak szerepében melyik mért változót alkalmazzuk. A 5.15 ábrán a modellek úgy vannak felrajzolva, hogy a mért változók nem változtatják a helyzetüket. 'Csak' a betűzés van megváltoztatva annak a szerepnek megfelelően, amit az adott mért változó a korábban felírt egyenletekben játszik. Azaz például az első két modell szerint ugyanaz a mért változó az exogén változó, míg a két endogén változó



5.15. ábra. Három hibátlanul illeszkedő modell ugyanarra a korreláció mátrixra

szerepet cserél. A harmadik modell pedig abban különbözik az elsőtől, hogy ott az egyik korábban endogén változó exogén szerepbe került.

Vagyis az első két modell közt mindössze annyi a különbség, hogy a két endogénnek vett változó közül melyik az oka a másiknak?! Az eredmény tehát — nevezetesen az, hogy mindkét modell hibátlanul reprodukálja az indikátorok korreláció mátrixát — azt mutatja, amit már korábban is említettünk: pusztán egy modell illeszkedése alapján nem feltétlen dönthető el, hogy az egyik változó 'oka-e' a másiknak!

A path modellek bemutatásának befejezéseként felírjuk általános formában azokat az egyenleteket, amiket a path modellek változói kielégítenek.

Legyenek a megfigyelt (indikátor) változók vektorai Y illetve X. Aszerint írva az egyik vagy másik mért változót az X illetve az Y koordináti közé, hogy az adott változó exogén-e. Exogén egy változó, ha más mért változók nem magyarázzák. Endogén egy változó, ha a modellben egyaránt kerül magyarázott és magyarázó szerepbe. Az X jelöli az exogén indikátorokat, az Y pedig az endogéneket. Legyen B és Γ két olyan mátrix (lineáris leképezés), ami a mérete szerint megfelel a következő egyenletben:

$$Y = BY + \Gamma X + \zeta.$$

Itt ζ a strukturális hiba.

Vegyük az előbb használt korreláció mátrixot és annak eredeti modelljét! Ekkor az Y vektor kétdimenziós és az X egy. A fenti egyenlet konstansai:

$$B = \begin{pmatrix} 0 & 0 \\ .126 & 0 \end{pmatrix},\ \Gamma = \begin{pmatrix} .495 \\ .336 \end{pmatrix}\ \text{továbbá}\ \operatorname{cov}(X) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ \operatorname{cov}(\zeta) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Ha a fenti egyenletet átrendezzük:

$$(I - B)Y = \Gamma X + \zeta.$$

Feltételezve, hogy a két oldal kovariancia mátrixa azonos, és használva a $\mathrm{cov}(\zeta) = \Psi$ szokásos jelölést:

$$(I - B)\mathrm{cov}(Y)(I - B)^T = \Gamma\Gamma^T + \Psi.$$

Ebből, ha az $I - B$ invertálható:

$$\mathrm{cov}(Y) = (I - B)^{-1}(\Gamma\Gamma^T + \Psi)((I - B)^{-1})^T.$$

Továbbá: $\mathrm{cov}(Y, X) = (I - B)^{-1}\Gamma$ és $\mathrm{cov}(X) = I$. Itt persze lehetséges volna, hogy az exogén változókat ne kelljen korrelálatlannak venni azaz, hogy a $\mathrm{cov}(X) = \Phi$ egy tetszőleges pozitív szemidefinit mátrix legyen. Ám akkor ennek a path diagramon is meg kell jelennie. A megfelelő exogén változókat egy-egy kétfejű hajlított nyíllal kell összekötni.

A kövekező programrészlet arra alkalmas, hogy az előző számolások helyességét leellenőrizzük.

```
# a kovariancia mátrix
#          Y1     Y2     X0
# Y1    452.711 13.656 17.431
# Y2     13.656  4.831  1.448
# X0     17.431  1.448  2.739
d1<-452.711;d2<-4.831;d3<- 2.739
c12<-13.656;c13<-17.431;c23<- 1.448

# a számolt korrelációk
r12<-c12/sqrt(d1*d2)
r13<-c13/sqrt(d1*d3)
r23<-c23/sqrt(d2*d3)
R<-matrix(c(1,r12,r13,r12,1,r23,r13,r23,1),3)
colnames(R)<-c("Y1","Y2","X0")
rownames(R)<-c("Y1","Y2","X0")
R
#     Y1    Y2    X0
# Y1  1.000 0.292 0.495
# Y2  0.292 1.000 0.398
# X0  0.495 0.398 1.000
rxy1<-R[1,3];rxy1 #  .495
rxy2<-R[3,2];rxy2 #  .398
ry12<-R[1,2];ry12 #  .292
```

```
# a path együtthatók
g10<-.495;g20<-.336;b21<-.126


# -----------------
# path az egyenletekbôl szorzással, várható értékkel

calc<-rxy1
res<-rbind(adott=g10,calc=calc);colnames(res)<-"g10"
res
#            g10
# adott 0.495000
# calc  0.495012

calc<-as.numeric(
      solve(matrix(c(1,rxy1,rxy1,1),2))%*%matrix(c(rxy2,ry12),2))
res<-rbind(adott=c(g20,p21),calc=calc);colnames(res)<-c("g20","p21")
res
#             g20       p21
# adott 0.3360000 0.1260000
# calc  0.3358014 0.1257821

# -------------------------------
# korrelációk a path együtthatókból
c(rxy1,g10)        # .495
c(rxy2,g20+g10*p21)# .398
c(ry12,p21+g10*g20)# .292

# ---------------------------
# az (Y,X) korreláció mátrixa
#    az Y=BY+GX+sZ modellbôl számolva
#    Y1=       g10*X0+s1*Z1
#    Y2=p21*Y1+g20*X0+s2*Z2

B<-matrix(c(0,p21,0,0),2)
I<-matrix(c(1,0,0,1),2)
G<-matrix(c(g10,g20),2)
# psi a Z kovarianciája
s1<- sqrt(1-g10^2)
s2<- sqrt(1-g20^2-p21^2)
psi<-diag(c(s1^2,s2^2))
```

```
# az (Y,X) korreláció mátrix részei
SYY<-solve(I-B)%*%(G%*%t(G)+psi)%*%t(solve(I-B))
SXY<-solve(I-B)%*%G
SXX<-1

M<-rbind(cbind(SYY,SXY),cbind(t(SXY),1))

M;R # ugyanaz
c(s1,s2) # az Z koordinátáinak szórásai
```

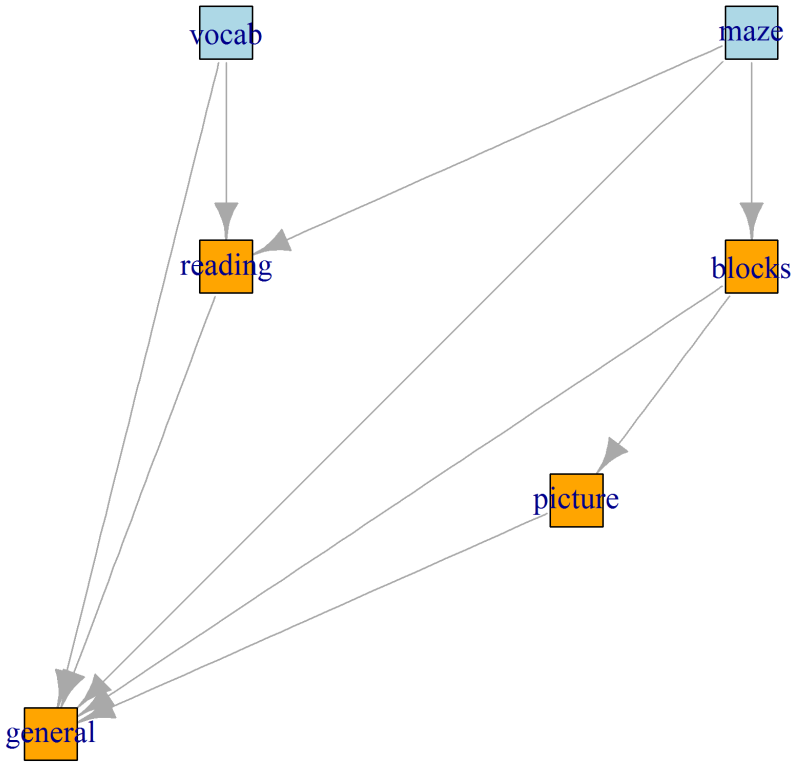
5.3.3. PATH modellek a gyakorlatban

Az **R**-project több kiegészítése is alkalmas path modellek kezelésére. A mai felfogás szerint nincs éles határ a PATH és a SEM (strukturális egyenletekkel való modellezés) modellek közt. Sőt a modellek csoportjai részben átfedődnek a látens változókkal való modellezésnek nevezett módszertannal, és az annak körében vizsgált modellekkel is. A fogalmak használatának nincsen általánosan elfogadott standardja és ez a különböző **R** kiegészítések fogalom rendszerében is tetten érhető. Ebben a jegyzetben főleg azokat a modelleket soroljuk a SEM modellek körébe, amikben előfordul olyan rejtett (hidden) változó aminek értékét hibával sem tudjuk mérni. Ami egy olyan feltételezett rejtett exogén vagy endogén véletlen érték ami nem mérhető, amire vonatkozóan nincsenek közvetlen mért indikátorok. A path diagramokon általában ezeket jelölik körök vagy ellipszisek.

A következő példákat a 'lava' csomag segítségével mutatjuk be [14]. Ez a 'Linear Latent Variable Models' csomag az itt bemutatottnál sokkal többre képes. De több vonatkozásában még fejlesztés alatt áll. Betöltéskor igényli az 'mvtnorm' és a 'numDeriv' csomagok telepített voltát is.

A képességek kapcsolatának egy path modellje

Az 5.3.2 részben már bemutatott `ability.cov` adathalmaz adatait használjuk demonstrációra. Ez az adathalmaz sajnos csupán a tapasztalati kovariancia mátrixát tartalmazza 6 emberi képesség adatnak. Ez csak technikalilag probléma, hiszen az illesztendő modellekhez a kovarianciamátrix elégséges információt tartalmaz. De ez mégis azért probléma, mert (ismereteim szerint) a most felhasználandó 'lava' csomag, csak nyers adatok felhasználására van felkészítve. A problémát úgy kerüljük meg, hogy a megadott kovariancia mátrix szerint véletlen számokat generálunk.



5.16. ábra. Az 'ability.cov' adatokra illesztett path modell

A fenti 5.16 ábra az alábbi utasítás sorok eredményeként keletkezett. Az első utasítás betölti a 'lava' kiegészítést. A következő egy path struktúrát definiál. Ezután a kívánt kovarianciának megfelelően véletlen mintaadatokat generálunk. Végül az 'estimate()' paranccsal összepárosítva a modellt és az adatokat, megbecsüljük a keresett path együtthatókat.

```
library(lava)  # +mvtnorm + numDeriv
```

```
m <- lvm(list(general~picture+vocab+blocks+maze+reading,
              picture~blocks,reading~vocab,blocks~maze,reading~maze))
d<-rmvnorm(12345,sigma=ability.cov$cov)
colnames(d)<-colnames(ability.cov$cov)
estimate(m,d)
plot(m)
```

Az eredményeket tömörítve a 'summary' paranccsal kapjuk. Érdemes megvizsgálni a következő parancsok eredményeit is:

```
children(m,~maze) # válasz: "general" "blocks"  "reading"
parents(m,~reading) # válasz: "vocab" "maze"
endogenous(m) # válasz: "general" "picture" "blocks"  "reading"
exogenous(m) # válasz: "vocab" "maze"
```

Összefoglaló

Mint láthattuk, a path módszer egy vizuálisan is megjeleníthető modellje a megfigyeléseink korreláció mátrixának. Emiatt a felhasználók körében igen népszerű. Ugyanakkor az eredemények értékelésekor fokozott óvatosságra van szükség. A path diagramon két változó közt egy nyíl önmagában nem jelenti, hogy az egyik változó által leírt jelenségnek a másik oka lenne. Akkor sem, ha az adott úthoz jelentős súly tartozik.

5.4. A SEM modellek

Előbb röviden bemutatjuk azokat a fogalmakat amikkel a SEM modellek dolgoznak. Majd leírjuk a SEM általános modelljét. Végül két **R** alkalmazás demonstrálja a modell gyakorlati felhasználásának lehetőségét.

5.4.1. A SEM történet

A SEM modellek fokozatosan érték el mai általánossági szintjüket. Legelső változatok még a XX. század 30-as éveiben keletkeztek. De ezek a modellek inkább a path modellek családjába tartoztak. Később a század közepétől kezdve, a számítógépek megjelenésével a főkomponens-faktor típusú modellek kerültek az érdeklődés középpontjába. A SEM modellek abban a formában, ahogyan most itt szerepelni fognak, a század utolsó harmadában kerültek előtérbe. Az e tárgyban folytatott jelenlegi vizsgálatok igen szerteágazók. Az itt bemutatottnál jóval nehezebben megoldható, speciális eloszlásokat feltételező, nem feltétlen lineáris modellekkel foglalkoznak. Megjegyzendő, hogy a modell eredményessége, megoldhatósága, a megoldásának stabilitása számos esetben kritikus lehet.

5.4.2. A SEM fogalmak

A SEM modell általános alakja

Az általános (lineáris) SEM modell 3 látens (a ξ az η és a ζ) és 2 megfigyelhető (az X és az Y) valamint 2 megfigyelési hiba (az ε és a δ) változóra épül.

A látens változók értelmezése a következő. A ξ egy olyan látens változó amit más látens változó nem magyaráz. Az η egy olyan látens változó amit más látens változók magyaráznak együttesen egy ζ hibával, de ami maga is lehet más változók magyarázója. A ζ — mint már szerepelt — a látens hiba vektor, ami a magyarázott látens változók magyarázottsági hibája.

A megfigyelhető (a megfigyelt) változók értelmezése a következő. Az X a δ hibával megfigyelt ξ változó. Az Y az ε hibával megfigyelt η változó.

Mindegyik eddig említett összefüggésről feltételezzük, hogy lineáris.
A következők szerint.

A látens változók egyenlete:

$$\eta = \mathbf{B}\eta + \mathbf{\Gamma}\xi + \zeta$$

A megfigyelt változók egyenletei:

$$Y = \Lambda_Y \eta + \varepsilon$$
$$X = \Lambda_X \xi + \delta$$

A modell keretei közt az η változót szokás (látens) endogén változónak a ξ-t pedig (látens) exogénnek nevezni, a ζ-t pedig látens strukturális hibának. Az ε és a δ a megfigyelési hibák. Az Y a megfigyelt endogén, az X pedig a megfigyelt exogén változó.

Könnyen látható, hogy az (η, ξ) vektor kovariancia mátrixa:

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{\eta,\eta} & \mathbf{\Sigma}_{\eta,\xi} \\ \mathbf{\Sigma}_{\xi,\eta} & \mathbf{\Sigma}_{\xi,\xi} \end{pmatrix} = \begin{pmatrix} (I-B)^{-1}(\Gamma\Phi\Gamma^T + \Psi)(I-B)^{-1^T} & (I-B)^{-1}\Gamma\Phi \\ \Phi\Gamma(I-B)^{-1^T} & \Phi \end{pmatrix}$$

Ennek alapján viszont az (Y, X) kovariancia mátrixa is közvetlen felírható:

$$\begin{pmatrix} \Lambda_Y \mathbf{\Sigma}_{\eta,\eta} \Lambda_Y^T & \Lambda_Y \mathbf{\Sigma}_{\eta,\xi} \Lambda_X^T \\ \Lambda_X \mathbf{\Sigma}_{\xi,\eta} \Lambda_Y^T & \Lambda_X \mathbf{\Sigma}_{\xi,\xi} \Lambda_X^T \end{pmatrix}$$

Mint látható a két megfigyelt változó közt az a különbség, hogy az X egy olyan látens változó megfigyelése ami a modellen belül csak mint magyarázó változó szerepel míg az Y egy olyan változó megfigyelése amit egyrészt más változók magyarázhatnak de ami saját maga magyaráz más hidden változókat. Azaz az Y által megfigyelt rejtett változók közt belső, modellezendő kapcsolatok lehetnek.

A SEM modellek illesztése általában úgy történik, hogy feltételezések alapján kitöltött $\mathbf{B}$ és $\mathbf{\Gamma}$ mátrixok segítségével paraméteresen felirt kovariancia mátrixszal vagy numerikusan közelítjük (legkisebb négyzetek módszere) a megfigyelt adatokból nyerhető tapasztalati kovariancia mátrixot. Vagy pedig, szintén a mondott paraméterek alapján vett likelhood (esély) függvényt maximalizáljuk.

5.4.3. SEM modellek a gyakorlatban

Több olyan csomag van, ami az **R**-project keretein belül SEM modellek illesztésére alkalmas. Az alábbiakban ezek közül kettőt emelünk ki. A 'sem' [11] és a 'lavaan' [30]

[30] csomagokat. Mindkettő többé-kevésbé megvalósítja az adott alkalmazási területen klasszikusnak számító LISREL program funkcióit. Mindkettő képességei messze meghaladják az itt bemutatottakat. Mindkettőt egy-egy klasszikus adathalmaz feldolgozásával mutatjuk be.

A vágyak közti kereszthatás elemzése a 'sem' csomaggal

Példaként annak az adathalmaznak az elemzését mutatjuk be, amit a 'sem' csomag szerzője, John Fox maga is mint reprezentációs anyagot használ. Az adathalmaz klasszikus [6] [9]. Mindössze a korreláció mátrix áll rendelkezésre abból a 329 fő megkérdezésén alapuló vizsgálatnak, ami arra irányult, hogy egy diák és egy barátja tanulási és foglalkoztatási vágyait hogyan befolyásolják saját teljesítőképességük illetve a szülői anyagi lehetőségek és vágyak.

Egy 10×10-es korreláció mátrix áll rendelkezésre, amely a következő nevű és tartalmú változók közti tapasztalati korrelációt tartalmazza:

```
RIQ      a saját IQ
RSES     a saját szüleinek gazdasági-társadalmi státusa
ROccAsp  a saját foglalkoztatottsággal kapcsolatos vágya
REdAsp   a saját tanulással kapcsolatos vágyak
RParAsp  a saját szüleinek vágyai
FIQ      a barát IQ-ja
FSES     a barát szüleinek gazdasági-társadalmi státusa
FOccAsp  a barát foglalkoztatottsággal kapcsolatos vágya
FEdAsp   a barát tanulással kapcsolatos vágyai
FParAsp  a barát szüleinek vágyai
```

Az illesztendő modell konstrukciójakor abból indulunk ki, hogy a saját (respondent) és a barát (friend) tanulással, foglalkoztatással kapcsolatos vágyait két látens endogén változó magyarázza. A sajátét és a barátét, külön-külön: a saját és a barát 'RGenAsp' illetve 'FGenAsp' módon jelölt 'általános vágy' változója. Ugyanakkor feltesszük egyrészt, hogy ez a két változó egymást kölcsönösen is magyarázza. Másrészt, hogy ezek az 'általános' vágy' hidden változók mérhető exogén indikátorokkal magyarázhatóak. Nevezetesen a saját IQ-nk mellett a saját szüleink vágyaival. Továbbá, hogy mindkettőnknek ez a látens általános vágyát magyarázza nem csak a saját, hanem a másik család társadalmi-gazdasági státusa is.

Az alábbi program egyfelől beolvassa a system inputról a mondott adatok hivatkozott korrelációmátrixát. Másfelől leírja a fenti vázlatnak megfelelő SEM struktúrát. A 'sem()' parancs elvégzi a paraméterezett modell illesztését. A 'summary()' kiíratja a becsléssel

kapcsolatos legfontosabb eredményeket. A 'path.diagram()' elkészít egy olyan programot, amit a 'graphviz' önálló szabad szofver értelmezni tud, és beadva neki elkészíti a vizsgált SEM struktúra 5.17 grafikus képét.

A fenti modell elemzéséhez szükséges **R** utasítások:

```
require(sem) # a 3.0 utani verziókban kell +matrixcalc

Dr<-read.moments(diag=FALSE,names=c('ROccAsp','REdAsp','FOccAsp',
                'FEdAsp','RParAsp','RIQ','RSES','FSES','FIQ','FParAsp'))
    .6247
    .3269  .3669
    .4216  .3275  .6404
    .2137  .2742  .1124  .0839
    .4105  .4043  .2903  .2598  .1839
    .3240  .4047  .3054  .2786  .0489  .2220
    .2930  .2407  .4105  .3607  .0186  .1861  .2707
    .2995  .2863  .5191  .5007  .0782  .3355  .2302  .2950
    .0760  .0702  .2784  .1988  .1147  .1021  .0931 -.0438  .2087

Dm <- specify.model()
    RParAsp  -> RGenAsp, gam11,  NA
    RIQ      -> RGenAsp, gam12,  NA
    RSES     -> RGenAsp, gam13,  NA
    FSES     -> RGenAsp, gam14,  NA
    RSES     -> FGenAsp, gam23,  NA
    FSES     -> FGenAsp, gam24,  NA
    FIQ      -> FGenAsp, gam25,  NA
    FParAsp  -> FGenAsp, gam26,  NA
    FGenAsp  -> RGenAsp, beta12, NA
    RGenAsp  -> FGenAsp, beta21, NA
    RGenAsp  -> ROccAsp,  NA,       1
    RGenAsp  -> REdAsp,  lam21,  NA
    FGenAsp  -> FOccAsp,  NA,       1
    FGenAsp  -> FEdAsp,  lam42,  NA
    RGenAsp <-> RGenAsp, ps11,   NA
    FGenAsp <-> FGenAsp, ps22,   NA
    RGenAsp <-> FGenAsp, ps12,   NA
    ROccAsp <-> ROccAsp, theta1, NA
    REdAsp  <-> REdAsp,  theta2, NA
    FOccAsp <-> FOccAsp, theta3, NA
```

```
    FEdAsp  <-> FEdAsp,  theta4, NA

M<-sem(Dm,Dr,329,fixed.x=c('RParAsp','RIQ','RSES','FSES','FIQ','FParAsp'))
summary(M)
path.diagram(M, min.rank='RIQ, RSES, RParAsp, FParAsp, FSES, FIQ',
             max.rank='ROccAsp, REdAsp, FEdAsp, FOccAsp')
```

A programban a korrelációmátrix és a modell adatait követő üres soroknak fontos szerepük van. Azok zárják le a megfelelő információk system inputról való beolvasását.

Az illesztett modell 'summary()' paranccsal kinyerhető paraméterei:

```
      Parameter Estimates
              Estimate Std Error  z value Pr(>|z|)
      gam11   0.161224 0.038487   4.1890 2.8019e-05 RGenAsp <--- RParAsp
      gam12   0.249653 0.044580   5.6001 2.1428e-08 RGenAsp <--- RIQ
      gam13   0.218404 0.043476   5.0235 5.0730e-07 RGenAsp <--- RSES
      gam14   0.071843 0.050335   1.4273 1.5350e-01 RGenAsp <--- FSES
      gam23   0.061894 0.051738   1.1963 2.3158e-01 FGenAsp <--- RSES
      gam24   0.228868 0.044495   5.1437 2.6938e-07 FGenAsp <--- FSES
      gam25   0.349039 0.044551   7.8346 4.6629e-15 FGenAsp <--- FIQ
      gam26   0.159535 0.040129   3.9755 7.0224e-05 FGenAsp <--- FParAsp
      beta12  0.184226 0.096207   1.9149 5.5506e-02 RGenAsp <--- FGenAsp
      beta21  0.235458 0.119742   1.9664 4.9255e-02 FGenAsp <--- RGenAsp
      lam21   1.062674 0.091967  11.5549 0.0000e+00 REdAsp <--- RGenAsp
      lam42   0.929727 0.071152  13.0668 0.0000e+00 FEdAsp <--- FGenAsp
      ps11    0.280987 0.046311   6.0674 1.2999e-09 RGenAsp <--> RGenAsp
      ps22    0.263836 0.044902   5.8759 4.2067e-09 FGenAsp <--> FGenAsp
      ps12   -0.022601 0.051649  -0.4376 6.6168e-01 FGenAsp <--> RGenAsp
      theta1  0.412145 0.052211   7.8939 2.8866e-15 ROccAsp <--> ROccAsp
      theta2  0.336148 0.053323   6.3040 2.9003e-10 REdAsp <--> REdAsp
      theta3  0.311194 0.046665   6.6687 2.5800e-11 FOccAsp <--> FOccAsp
      theta4  0.404604 0.046733   8.6578 0.0000e+00 FEdAsp <--> FEdAsp

       Iterations =  28
```

Az a 'graphviz' program, amit a 'path.diagram()' készít:

```
   digraph "sem.dhp" {
   rankdir=LR;
```

```
size="8,8";
node [fontname="Helvetica" fontsize=14 shape=box];
edge [fontname="Helvetica" fontsize=10];
center=1;
{rank=min "RIQ" "RSES" "RParAsp" "FParAsp" "FSES" "FIQ"}
{rank=max "ROccAsp" "REdAsp" "FEdAsp" "FOccAsp"}
"RGenAsp" [shape=ellipse]
"FGenAsp" [shape=ellipse]
"RParAsp" -> "RGenAsp" [label="gam11"];
"RIQ" -> "RGenAsp" [label="gam12"];
"RSES" -> "RGenAsp" [label="gam13"];
"FSES" -> "RGenAsp" [label="gam14"];
"RSES" -> "FGenAsp" [label="gam23"];
"FSES" -> "FGenAsp" [label="gam24"];
"FIQ" -> "FGenAsp" [label="gam25"];
"FParAsp" -> "FGenAsp" [label="gam26"];
"FGenAsp" -> "RGenAsp" [label="beta12"];
"RGenAsp" -> "FGenAsp" [label="beta21"];
"RGenAsp" -> "ROccAsp" [label=""];
"RGenAsp" -> "REdAsp" [label="lam21"];
"FGenAsp" -> "FOccAsp" [label=""];
"FGenAsp" -> "FEdAsp" [label="lam42"];
}
```

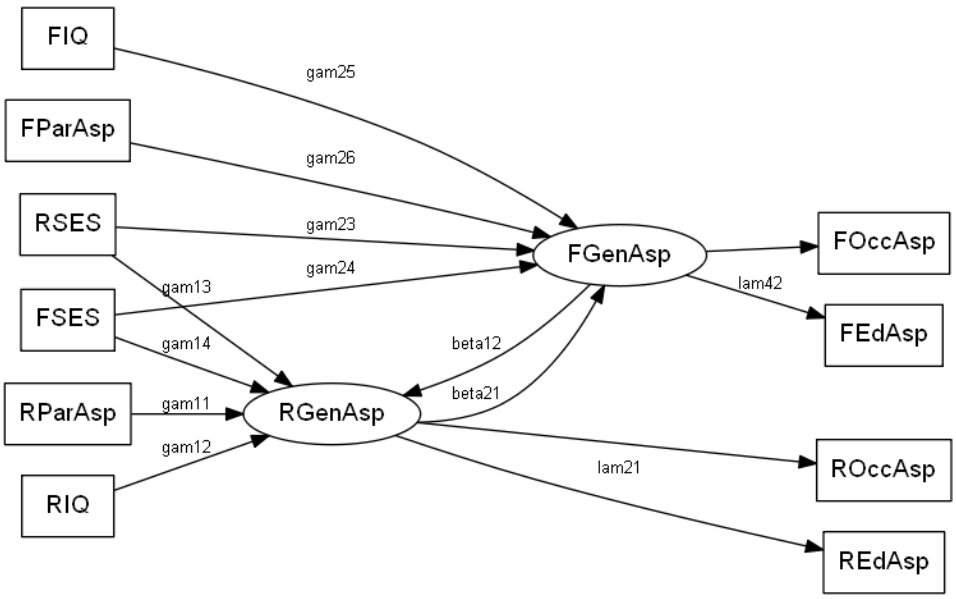
A vizsgált SEM modell 'graphviz' által előállitott képe az 5.17 ábra.

Gazdaság és demokrácia közti kapcsolat elemzése a 'lavaan' csomaggal

Ebben a részben is egy klasszikus adathalmaz feldolgozását mutatjuk be de egy másik programcsomag segítségével.

A feldolgozott adatok. A XX. század 60-as éveiben vizsgáltak 75 iparilag fejlett országot a következő 11 változó szerint.

```
y1    Expert ratings of the freedom of the press in 1960
y2    The freedom of political opposition in 1960
y3    The fairness of elections in 1960
y4    The effectiveness of the elected legislature in 1960
```



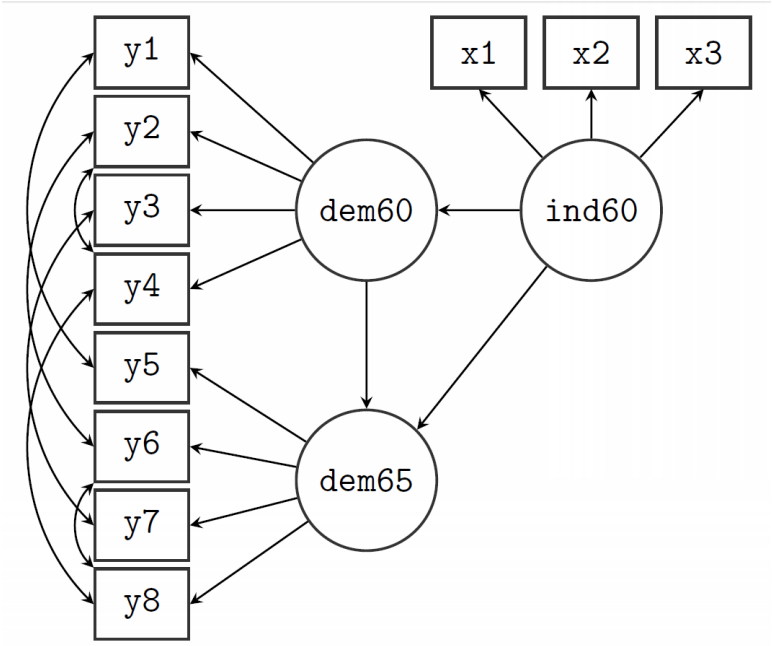
5.17. ábra. A látens általános törekvés SEM modellje

```
y5    Expert ratings of the freedom of the press in 1965
y6    The freedom of political opposition in 1965
y7    The fairness of elections in 1965
y8    The effectiveness of the elected legislature in 1965
x1    The gross national product (GNP) per capita in 1960
x2    The inanimate energy consumption per capita in 1960
x3    The percentage of the labor force in industry in 1960
```

Maguk az adatok $[0, 10]$ skálán felvéve a 'lavaan::PoliticalDemocracy' adathalmazban találhatóak, országnevek nélkül. Ezekre az adatokra a következő 5.18 rajz szerinti SEM modellt illesztjük.

A modell szerint — mint látható — egy látens exogén változó, az 'ind60' (az ipari fejlettség) feltételezett, és két látens endogén a 'dem60' és a 'dem65' (a demokrácia státusa 1960-ban és 1965-ben). Az ipari fejlettséget az 'x1', 'x2', 'x3' változók alapján figyelhetjük meg. A demokrácia státusai közül az 1960-ast az 'y1', 'y2', 'y3', 'y4', az 1965-öst pedig az 'y5', 'y6', 'y7' , 'y8' változók alapján mérhetjük.

A leírt modellt a 'lavaan' csomag segítségével modellezzük. Ez a csomag csak az **R**-project újabb verzióiban érhető el. Nem tartozik az alapcsomagok közé. Külön instal-



5.18. ábra. SEM modell látens gazdasági fejlettség és demokrácia változókkal

lálni kell és használat előtt be is kell tölteni. Betöltéskor feltételezi a 'boot', 'mnormt', 'pbivnorm', 'quadprog' továbbá a 'MASS' csomag telepített voltát.

A 'lavaan' csomag utasításaival a fenti modell leírása és illesztése, az illesztés eredményeinek kiíratása a következő programrészlettel érhető el.

```
model <- '
# a látens változók definiciója
ind60 =~ x1 + x2 + x3
dem60 =~ y1 + a*y2 + b*y3 + c*y4
dem65 =~ y5 + a*y6 + b*y7 + c*y8
# a regressziók
dem60 ~ ind60
dem65 ~ ind60 + dem60
# a maradékok közti korrelációk
y1 ~~ y5
y2 ~~ y4 + y6
y3 ~~ y7
y4 ~~ y8
y6 ~~ y8
```

```
fit <- sem(model, data=PoliticalDemocracy)
summary(fit, fit.measures=TRUE)
```

Az illesztett paraméterek a következők:

```
                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  ind60 =~
    x2                2.180    0.138   15.751    0.000
    x3                1.818    0.152   11.971    0.000
  dem60 =~
    y2         (a)    1.191    0.139    8.551    0.000
    y3         (b)    1.175    0.120    9.755    0.000
    y4         (c)    1.251    0.117   10.712    0.000
  dem65 =~
    y6         (a)    1.191    0.139    8.551    0.000
    y7         (b)    1.175    0.120    9.755    0.000
    y8         (c)    1.251    0.117   10.712    0.000
Regressions:
  dem60 ~ ind60       1.471    0.392    3.750    0.000
  dem65 ~ ind60       0.600    0.226    2.661    0.008
          dem60       0.865    0.075   11.554    0.000
Covariances:
  y1 ~~  y5           0.583    0.356    1.637    0.102
  y2 ~~  y4           1.440    0.689    2.092    0.036
         y6           2.183    0.737    2.960    0.003
  y3 ~~  y7           0.712    0.611    1.165    0.244
  y4 ~~  y8           0.363    0.444    0.817    0.414
  y6 ~~  y8           1.372    0.577    2.378    0.017
```

6. fejezet

Skálázás

6.1. Bevezető

A skálázás során azt feltételezzük, hogy a mérések a megfigyelt objektumpárok távolságaira vonatkoznak. Azaz rendelkezésre áll egy T *távolság mátrix* ami, ha a megfigyelt objektumok száma n akkor $n \times n$ méretű, és benne az $i.$ sor $j.$ eleme az $i.$ objektumnak a $j.$ objektumtól mért távolsága. [3]

A skálázás feladata az, hogy egy adott k-ra az n objektum mindegyikének megfeleltessen egy-egy $\mathbb{R}^k$-beli pontot úgy, hogy a megfeleltetett pontok Euklideszi távolságai nagyjából egyenlőek legyenek az objektum párok távolságaival.

Az objektumok közti szóbanforgó távolságok lehetnek tényleges távolságok. Ez az eset, amikor például az európai fővárosokról akarunk egy olyan kétdimenziós ábrát (térképet) készíteni, amin a fővárosokat jelző pontok távolságai nagyjából megfelelnek a városok közt tapasztalható repülési időnek vagy a sztrádákon vett távolságaiknak.

A módszer ugyanakkor absztrakt távolságok ábrázolására is alkalmas. Ezért lehet jól felhasználni például a szociológiában, amikor egyes személyek közti kapcsolatok szorosságát kell leirni olyan másodlagos információk alapján, mint a találkozások gyakorisága, rokonságok foka, közös gyermekek száma stb.

A skálázás harmadik fontos alkalmazási területe a statisztikán belüli. Ekkor az ábrázolandó távolságokat az objektum párok valószínűségi különbözőségét mérő statisztikák jelentik. Ilyen eset amikor az objektumokat egy minden objektum mellett mérhető diszkrét változónak, az objektumra jellemző tapasztalati eloszlása jellemez és az objektumok közti távolságot ezen eloszlások közti távolság jelenti. Ezen alapul például a korrespondencia analízis is.

6.2. Távolságok ábrázolása

6.2.1. Távolságok egzakt ábrázolása

Először néhány segédeszköz:

Egy $\mathbb{R}^k$-beli n pontból álló ponthalmazt azzal a $k \times n$ méretű X mátrixszal fogjuk azonosítani, aminek azok a vektorok az oszlopai, amik az origóból a ponthalmaz egyes pontjaiba mutatnak. Egy n elemű X ponthalmaz *távolság mátrix*ának azt az $n \times n$ méretű T_X mátrixot nevezzük, aminek $i.$ sor $j.$ eleme az (i, j) pontpár euklideszi távolságának a négyzetével egyenlő. Egy X ponthalmaz *középpont*ja (centruma, súlypontja) az a pont, aminek minden koordinátája a pontok megfelelő koordinátáinak átlaga. Azt mondjuk, hogy egy ponthalmaz centrált, ha a ponthalmaz súlypontja az origóba esik. Egy n elemű X ponthalmaz *ponthalmaz skalárszorzat mátrix*ának azt az $n \times n$ méretű S_X mátrixot nevezzük, aminek az $i.$ sor $j.$ eleme azon két vektornak a skalárszorzata, ami a ponthalmaz középpontjából az $i.$ illetve a $j.$ pontba mutat.

Bevezetjük a csupa egyesekből álló n hosszú vektorra az $(1, ..., 1)^T = u$ jelölést, és a szintén csupa egyesekből álló $n \times n$ méretű mátrixra az $uu^T = U$ jelölést. Ezek alapján definiáljuk a

$$C = I - \frac{1}{n}U = I - \frac{1}{n}uu^T$$

*centráló mátrix*ot. E mátrix elnevezését az indokolja, hogy egy tetszőleges X ponthalmaz esetén a C-vel való jobbról-szorzás az X ponthalmaznak egy olyan eltolását eredményezi, hogy a kapott $Y = XC$ ponthalmaz középpontja az origóba esik. Fontos tulajdonsága a centráló mátrixnak, hogy szimmetrikus. Továbbá hogy $u^TC = Cu = 0$, és hogy ennek alapján: $CU = UC = 0$. A C idempotens, tehát $CC = C$. És ez az egyenlőség értelmezhető úgy is, hogy mert a C-t C-vel jobbról szorozva C-t kapunk, a (baloldali) C egy centrált ponthalmaz. Egy ponthalmaz nyilván akkor centrált, ha $Xu = 0$.

Egy Y centrált ponthalmaz esetén a ponthalmaz skalárszorzat mátrixa Y^TY, egy általános helyzetű X ponthalmaz skalárszorzat mátrixa

$$S_X = CX^TXC.$$

Ebből az előállításból nyilvánvaló, hogy a ponthalmazok skalárszorzat mátrixai szimmetrikusak, pozitív szemidefinitek és a $Cu = 0$ miatt olyanok, hogy az u egy, a 0-hoz tartozó sajátvektoruk. De ennek a tulajdonságnak a megfordítása is érvényes:

6.1. Állítás *Egy $n \times n$ méretű S mátrix pontosan akkor skalárszorzat mátrixa valamely k dimenziós ponthalmaznak, ha az S egy olyan k rangú pozitív szemidefinit mátrix, aminek az $u=(1,...,1)^T$ egy, a 0-hoz tartozó sajátvektora.*

Az, hogy az egy k dimenziós X ponthalmaz S_X skalárszorzat mátrixának a rangja legfeljebb k abból adódik, hogy egy $k \times n$ méretű X esetén az X^TX rangja legfeljebb k. Fordítva, ha az S a mondott tulajdonságokkal bír (pozitív szemidefinit és $Su = 0$), akkor az S sajátvektorai, a hozzájuk tartozó $\lambda_1, ..., \lambda_n$ sajátértékek csökkenő sorrendjében legyenek $v_1, ..., v_k, v_{k+1}=u, v_{k+2}..., v_n$. Ekkor $\lambda_{k+1} = ... = \lambda_n = 0$. Állítsuk össze az X ponthalmaz $k \times n$ méretű mátrixát az

$$X = (\sqrt{\lambda_1}v_1, ..., \sqrt{\lambda_k}v_k)^T$$

módon. Ez a ponthalmaz nyilván centrált, minthogy a benne szereplő sajátvektorok a $v_{k+1}=u$ sajátvektorra ortogonálisak, és ezért az $Xu = 0$. Továbbá ennek a ponthalmaznak a skalárszorzat mátrixa tényleg S, mert $X^TX=\sum_{j=1}^{n} \lambda_j v_j v_j^T=S$.

Jelölje $Diag(Z)$ egy tetszőleges Z mátrix esetén azt a mátrixot, aminek mérete és diagonális elemei megegyeznek a Z-vel, de a többi eleme mind 0. E jelölés segítségével zárt alakban is felirhatjuk egy X ponthalmaz távolságmátrixát.

A skalárszorzatokból a pontpárok távolságait a koszinusz tétel alapján számíthatjuk ki, egy-egy háromszöget felhasználva. Mégpedig úgy, hogy a pontpár mellé harmadikként egy tetszőlegeset választunk. Most kétféleképpen is felírjuk az X ponthalmazhoz tartozó távolság mátrixot. Előbb minden pontpár esetén az origót, utóbb pedig minden pontpár esetén a ponthalmaz középpontját választjuk harmadik pontnak:

$$T_X = Diag(X^TX)U + UDiag(X^TX) - 2X^TX =$$

$$Diag(CX^TXC)U + UDiag(CX^TXC) - 2CX^TXC.$$

Az S_X-re korábban és a T_X-re most nyert képleteket lemásolva két leképezést értelmezünk. A τ leképezést a ponthalmazokhoz tartozó skalárszorzat mátrixokon értelmezzük:

$$\tau(S_X) = Diag(S_X)U + UDiag(S_X) - 2S_X,$$

a σ leképezést a ponthalmazokhoz tartozó távolságmátrixokon értelmezzük:

$$\sigma(T_X) = -\frac{1}{2}CT_XC.$$

E jelölésekkel közvetlenül adódik az alábbi állítás:

6.2. Állítás *Egy X ponthalmaz távolság és skalárszorzat mátrixa közt a következő két összefüggés áll fenn:*

$$T_X = \tau(S_X), \quad S_X = \sigma(T_X).$$

És az is igaz, hogy a τ illetve a σ függvények a ponthalmazokhoz tartozó távolság és skalárszorzat mátrixok halmazán egymás inverzei. Vagyis hogy a τ és a σ bijekció a ponthalmazokhoz tartozó távolság illetve a ponthalmazokhoz tartozó skalárszorzat mátrixok körében.

Eddig azt vizsgáltuk milyenek a *ponthalmazhoz tartozó* távolság illetve skalárszorzat mátrixok. Most *általánosabb* értelemben definiáljuk hogy mit jelent az, hogy egy mátrix (nem feltétlen valamilyen R^k-beli ponthalmazhoz tartozó) *távolság illetve skalárszorzat mátrix*.

Egy $n \times n$ méretű T mátrixot *távolságmátrix*nak nevezünk, ha a mátrix elemei nemnegatívok, a mátrix szimmetrikus és a mátrix diagonálisa nulla. Egy $n \times n$ méretű S mátrixot *skalárszorzat* mátrixnak nevezünk, ha a mátrix szimmetrikus és ha az n dimenziós $u = (1, ..., 1)^T$ vektor 0-hoz tartozó sajátvektora.

Mint látható, egy ponthalmaz távolság-mátrixa *távolság mátrix* a most definiált értelemben és a ponthalmaz skalárszorzat-mátrixa *skalárszorzat mátrix*, szintén a most definiált értelemben.

Megjegyzés. A skálázás módszere kiterjeszthető olyan esetekre is, amikor a távolságmátrix nem szimmetrikus. Azaz ha az i. objektum távolsága a j.-től nem ugyanannyi mint a j.-é az i.-től. Ez fordul elő tipikusan a szociológiai esetekben (nem biztos, hogy két személy szimpátiájának mértéke kölcsönös). De ebben a részben csak a szimmetrikus távolságmátrixok esetével foglalkozunk.

További megjegyzések. Néhány esetben az objektumok távolsága helyett természetesebb a hasonlóságukról beszélni, és ennek megfelelően olyan pontokat keresni, amikre a nagy hasonlóságú objektumokat az Euklideszi tér közeli pontjai ábrázolják.

Az így megfogalmazott feladat — egy H hasonlóság mátrix ábrázolása — a korábbi feladat fordítottjának tűnik. Ha azonban feltesszük, hogy a H szerint minden objektum jobban hasonlít önmagára mint bármely más objektumra, és hogy a H hasonlóság szimmetrikus, és ha az i. és j. objektum hasonlóságát h_{ij} jelöli, akkor az úgynevezett standard transzformációval nyert

$$t_{ij} = -h_{ij} + \frac{1}{2}(h_{ii} + h_{jj})$$

t_{ij} transzformált hasonlóság értékek a most definiált értelemben távolság mátrixot alkotnak. Ráadásul ezzel a transzformációval a nagyobb hasonlóságú objektumpárokhoz

kisebb t_{ij} értékeket rendelünk. Tehát a hasonlóságok ábrázolását a hasonlóság mátrixhoz tartozó

$$T = \frac{1}{2}\, Diag(H)\, E + \frac{1}{2}\, E\, Diag(H) - H$$

távolságmátrix ábrázolásával megoldhatjuk. Továbbá az is teljesül, hogy a σ szerint a H hasonlóságmátrixhoz ugyanaz a skalárszorzat mátrix tartozik mint a most hozzárendelt távolságmátrixhoz: $\sigma(H) = -\sigma(T)$ (lássuk be!).

Belátjuk, hogy a σ és a τ nem csak az olyan távolság illetve skalárszorzat mátrixok halmazán inverzei egymásnak, amik ponthalmazhoz tartoznak, hanem a távolságmátrixok és a skalárszorzatok előbb definiált bővebb halmazán is.

6.3. Állítás *A tetszőleges T távolságmátrixra definiált*

$$\sigma(T) = -CTC/2$$

és a tetszőleges S skalárszorzat mátrixra definiált

$$\tau(S) = Diag(S)U + UDiag(S) - 2S$$

leképezés egymás inverze az értelmezési tartomány, azaz a skalárszorzat mátrixok és a távolság mátrixok teljes halmazán.

Belátjuk, hogy egyrészt tetszőleges S skalárszorzat mátrixra $\sigma(\tau(S)) = S$. Másrészt hogy tetszőleges T távolságmátrixra $\tau(\sigma(T)) = T$.

Tetszőleges S skalárszorzat mátrixra $\tau(S)$ nem-negatív, szimmetrikus és a diagonálisa 0 ezért a $\sigma(\tau(S))$ értelmezhető:

$$\sigma(\tau(S)) = -C(Diag(S)U + UDiag(S) - 2S)C/2 =$$

$$= -(CDiag(S)UC + CUDiag(S)C - 2CSC)/2 = S.$$

Itt az utolsó egyenlőség úgy látható be, hogy kifejtjük a C-t a definíciója szerint és felhasználjuk, hogy az S szimmetriája miatt: $u^T S = Su = 0$ és így az $US = SU = USU = 0$ is érvényes.

Tetszőleges T távolságmátrixra a $\sigma(T) = -CTC/2$ nyilván szimmetrikus, és a $Cu = 0$ miatt $\sigma(T)u = 0$. Tehát a $\sigma(T)$ skalárszorzat mátrix, így a $\tau(\sigma(S))$ értelmezhető, nevezetesen:

$$\tau(\sigma(T)) = Diag(\sigma(T))U + UDiag(\sigma(T)) - 2(\sigma(T)).$$

Ha felhasználjuk a $\sigma(T)$ és a C definíció szerinti értékét, azt hogy a $Diag()$ additív és hogy a $Diag(T)=0$, valamint hogy $Diag(UT)=Diag(TU)$, és hogy $Diag(UT)U=TU$, $UDiag(TU)=UT$ végül még azt, hogy $Diag(UTU)U=UDiag(UTU)=UTU$, adódik amit keresünk: $\tau(\sigma(T))=T$.

Az eddigi három (6.1., 6.2. és 6.3.) állításból adódik a következő:

6.4. Tétel *Egy T távolságmátrixhoz akkor található egy olyan k dimenziós X ponthalmaz, aminek (euklideszi) távolságmátrixa T — azaz a T akkor reprezentálható, — ha a*

$$\sigma(T) = -\frac{1}{2}CTC$$

(azaz a T-hez tartozó skalárszorzat mátrix) k rangú pozitív szemidetinit.

Ugyanis a σ és a τ bijekció és egymás inverze a skalárszorzat mátrixok és távolság mátrixok körében, és ugyancsak bijekció e mátrixok azon részhalmazán is ami a ponthalmazokhoz tartozó skalárszorzat illetve távolság mátrixok halmaza. Továbbá az első állítás szerint pontosan azok a mátrixok skalárszorzat mátrixai valamely ponthalmaznak, amik pozitív szemidefinitek és amiknek az $u = (1, ..., 1)^T$ szinguláris vektora. Ugyanakkor tetszőleges T mátrixra a $\sigma(T)$ olyan, hogy $\sigma(T)u = 0$, mert $Cu = 0$. Ezért az interpretálhatósághoz elég megkövetelni csak a pozitív szemidefinitséget a $\sigma(T)$-től.

Ez utóbbi miatt egyébként a $\sigma(T)$ rangja legfeljebb $n-1$. Vagyis az interpretáció tényleges dimenziója (ha létezik) biztosan legfeljebb n–1. Ez megfelel annak, hogy minden $m \geq n$ mellett, tetszőleges n darab $\mathbb{R}^m$-beli pont benne van az $\mathbb{R}^m$ tér egy megfelelően megválasztott n–1 dimenziós részhalmazában.

Megjegyzés. A skalárszorzat mátrix pozitív szemidefinitsége erősebb feltétel mint a háromszög egyenlőtlenség. Ezt mutatja az, hogy megadható olyan 4×4-es távolság mátrix amire teljesül a háromszög egyenlőtlenség, ámde olyan ami mégsem reprezentálható. Ugyanakkor fordítva: érdekes (a tétel felhasználása nélkül nehezen bizonyítható) tulajdonság az, hogy ha a $\sigma(T)$ pozitív szemidefinit, akkor a T-re igaz a háromszög egyenlőtlenség.

További megjegyzések. Itt emlékeztetünk arra, hogy az $\mathbb{R}^k$-beli n elemű ponthalmazokat olyan $k \times n$ méretű X mátrixokkal jelöltük, amiknek az *oszlopai* a megfelelő pontokba mutató vektorok. És a ponthalmaz T távolságmátrixa egy olyan $n \times n$ méretű mátrix, aminek elemei a pontpárok közti távolságok négyzetei.

Az X ponthalmaz mátrixos értelmezésére azért fontos emlékeztetni, mert számos helyen olyan mátrixot rendelnek a ponthalmazhoz, aminek a sorai jelentik az egyes pontokat. Az

R programok futtatásakor is számos esetben találkozunk majd azzal, hogy olyan mátrix reprezentálja a pontokat, ami az általunk X-el jelölt mátrix transzponáltja.

Az hogy esetünkben a T-vel jelölt távolságmátrixok elemei valójában távolságnégyzetek azért fontos, mert ez a jelölés és értelmezés sem egységes a témában megjelent cikkek és könyvek körében. Számtalan helyen előfordul, hogy euklideszi távolságmátrixok esetén ezt $-t_{i,j}^2/2$-nek veszik. Ha ugyanis a T elemeit így, a távolságnégyzetek minusz egyszerese felének veszik, akkor abból számos egyszerűsödés adódik a kapcsolatos képletekben. Ilyen jelölés mellett az előző tétel feltétele persze nem a pozitív, hanem a negatív definitség. Ugyanakkor bemeneti adatként a kapcsolatos programoknak általában a nyers távolságadatokat kell megadni, azaz egy olyan mátrixot, aminek az elemei a $t_{i,j}$ értékek!

6.2.2. Az ábrázolhatósági feltétel általánosítása

A távolságmátrixok ábrázolhatóságának előbbi tétel szerinti feltétele általánosítható.

Azokra a c vektorokra amelyekre $c^T u = 1$, vezessük be a következő mátrixot:

$$C_c = I - uc^T.$$

A C_c mátrix a korábbi C mátrix általánosítása. Egy ponthalmaz X mátrixának C_c^T-vel való jobbról szorzása azt jelenti, hogy az X minden oszlopából kivonjuk az oszlopok c szerinti lineáris kombinációját. Tehát az XC_c^T az X ponthalmaznak az az eltolása, amelynek nyomán a pontok c szerinti lineáris kombinációja kerül az origóba.

Ha a $c = u/n$, akkor a megfelelő C_c a korábbi C-vel egyenlő, — és mint az már szerepelt — a C-vel való jobbról szorzás mint transzformáció a középpont origóba tolásának felel meg.

Ha pedig $c = u_j$ valamely $j = 1, ..., n$-re úgy, hogy az u_j minden koordinátája nulla kivéve a j-ediket ami $= 1$, akkor a megfelelő C_c egy olyan eltolás, ami a j. pontot tolja az origóba.

Definiáljuk tetszőleges T távolságmátrix és c esetén az

$$S_c = -\frac{1}{2} C_c T C_c^T$$

skalárszorzat mátrixot. E mátrix korábbival azonos elnevezését az indokolja, hogy ha T reprezentálható, akkor S_c egyenlő azzal a skalárszorzat mátrixszal, amelyik az Xc pontból az X ponthalmaz pontjaiba húzott vektorok alapján keletkezik.

6.5. Állítás *Azokra a c-kre, amikre $c^T u = 1$, az $S_c = -\frac{1}{2} C_c T C_c^T$ pontosan akkor pozitív szemidefinit amikor az $S = -\frac{1}{2} CTC$ pozitív szemidefinit.*

Az állítás könnyen adódik egyrészt abból, hogy egy tetszőleges A és B mátrixra, ha az A pozitív szemidefinit, akkor (ha a szorzás értelmezhető) a BAB^T is pozitív szemidefinit. Másrészt pedig abból, hogy $CC_c = C$ és $C_c C = C_c$.

Ugyanis ez utóbbiak miatt a $CS_cC = CC_cTC_c^TC = CTC = S$ és a $C_cSC_c^T = C_cCTCC_c^T = C_cTC_c^T = S_c$. Vagyis tényleg: ha az S_c pozitív szemidefinit akkor az S is az. És fordítva is: ha az S pozitív szemidefinit, akkor az S_c is az.

Legyen T most egy X ponthalmaz távolságmátrixa: $T = T_X$, és egy olyan c vektorra amelyre $c^T u = 1$ legyen $S_c = \sigma_c(T)$ a megfelelő skalalárszorzat mátrix. Vagyis legyen T egy reprezentálható távolság mátrix, és S_c legyen a ponthalmaz Xc ponthoz tartozó skalárszorzat mátrixa. Ekkor az S_c mátrix $j.$ diagonális eleme a $j.$ pont távolságnégyezete a c által meghatározott Xc ponttól, és a $Tc - (c^T Tc)u/2$ az az n dimenziós vektor, amit az S_c-nek ezekből a diagonálisbeli elemeiből képezhetünk.

Így, ha a T reprezentálható és ha a reprezentáló pontok köré irható gömb, és ha ennek a középpontját a c határozza meg, és ha ennek a gömbnek a sugara r, akkor érvényes a

$$Tc - (c^T Tc)u/2 = r^2 u$$

egyenlőség. Vagyis a $k = r^2 + c^T Tc/2$ konstanssal, a $detT \neq 0$ feltétel mellett, a c-re teljesül hogy

$$c = kT^{-1}u \ .$$

Ezt az egyenletet balról u^T-vel szorozva, mert az $u^T c = 1$ adódik, hogy a $k = 1/u^T T^{-1} u$. Tehát az egyenlet alapján a $c = T^{-1}u/u^T T^{-1} u$.

Ezt a c értéket visszahelyettesítve az eredeti egyenletbe adódik, hogy az $u^T T^{-1} u > 0$ feltétel teljesülése mellett (ez általánosan igazolható, ha a T^{-1} létezik) a kör sugara

$$r = \frac{1}{\sqrt{2u^T T^{-1} u}} \ .$$

A középpont tehát az S egy tetszőleges $S = X^T X$ dekompozíciója mellett az X-nek a $c = 2r^2 T^{-1} u$-val súlyozott átlaga. Vagyis egy X reprezentáció pontjai köré irt gömb középpontja az:

$$Xc = \frac{XT^{-1}u}{u^T T^{-1} u}.$$

A legutóbbi, 6.4. tétel azzal foglalkozott, hogy mikor van egy távolságmátrixnak a *valahány dimenziós* euklideszi térben interpretációja.

Látható, hogyha van a távolságmátrixnak interpretációja, akkor annak a tényleges dimenziója legfeljebb n–1.

Utóbb pedig levezettük azt is, hogyha a távolságmátrixnak van interpretációja, akkor az rajta van egy n–1 dimenziós gömbön, és kiszámoltuk a pontok köré irt gömb sugarát és a középpontját is.

Ezeknek az állításoknak akkor lesz jelentőségük, amikor az objektumoknak skálázásal nyert térképét kell kiértékelünk. Ugyanakkor ez utóbbi megállapítások nem különösebben meglepőek, hiszen 3 pontra mindig van egy sík ami a 3 pontot tartalmazza, és a 3 pont mindig rajta van egy körön. Hasoló módon 4 tetszőleges dimenziós pont mindig benne van egy 3 dimenziós altérben és 4 pont mindig rajta van egy közös gömbön. Stb.

6.3. Távolságok közelítő ábrázolása

A T távolságmátrix k dimenziós *klasszikus közelítő ábrázolás*ának azt az X_k ponthalmazt nevezzük, ami úgy adódik hogy vesszük a távolságmátrixhoz tartozó $-\frac{1}{2}CTC$ skalárszorzat mátrix k darab legnagyobb, $\lambda_1, ..., \lambda_k$ sajétértékét és a hozzátartozó egységhosszú $v_1, ..., v_k$ sajátvektorokat és ezekből felépítjük azt az X_k ponthalmazt, aminek mátrixában a j. sor $\sqrt{\lambda_j}v_j^T$.

Ha nincs a skalárszorzatmátrix sajátértékei közt, csak $\ell < k$ darab nemnegatív, akkor csak az ℓ darab nemnegatívnak megfelelő sajátvektort vesszük, és a nyert reprezentáció ℓ dimenziós lesz. A továbbiakban levezetünk két tételt, ami egyszerű következménye annak, hogy mi egy mátrix optimális közelítése pozitív szemidefinit mátrixszal.

A teljesség kedvéért, mindkét tételt az igen szép bizonyításával együtt mutatjuk be.

6.3.1. Közelítés ℓ_1 normában

Megjegyzés. Jelölje a $T = (t_{i,j})$ távolságmátrixot közelítően interpretáló ponthalmaz távolságmátrixát $D = (d_{i,j})$ és a skalárszorzat mátrixát V. Természetesnek látszó cél olyan ponthalmaz keresése amire

$$\ell_1(T, D) = \sum_{i,j=1}^{n} (t_{ij} - d_{ij})$$

minimális. Csakhogy a fenti szumma átrendezhető, és a $tr(T) = tr(D) = 0$. Így a

$$\sum_{i,j=1}^{n} t_{ij} = u^T T u = tr(u^T T u) = tr(TU) = tr(UT) = \frac{1}{n} tr(UTU),$$

tehát a

$$\sum_{i,j=1}^{n} t_{ij} = -n\, tr(I - \frac{1}{n}U)T(I - \frac{1}{n}U) = -n\, tr(CTC) = 2n\, tr(S)$$

és a hasonlóan adódó $\sum_{i,j=1}^{n} d_{ij} = 2n\, tr(V)$ alapján:

$$\ell_1(T, D) = 2n\, tr(S - V).$$

Ez a kifejezés is nyilvánvaló módon mutatja, hogy az ℓ_1 tetszőleges T esetén egy megfelelően választott V mellett akár egy dimenziós interpretáció mellett is nullává tehető. Például úgy, hogy az objektumokat két olyan pontba képezzük le, amelynek távolsága a $\sum_{i,j=1}^{n} t_{ij}$ annyiad része, mint amennyi a két pontba leképezett objektumok számának szorzata. Ez a reprezentáció nyilván nem mond sokat az adatainkról. Tehát bizonyos korlátozásokat kell bevezetnünk, hogy az ℓ_1 távolság szerint értelmes közelítését kapjuk a T mátrixnak.

Tegyük fel, hogy T *reprezentálható* távolságmátrix és keressük a T-nek az ℓ_1 szerinti legjobb reprezentációját, a reprezentáló ponthalmazok k dimenziós *vetületei körében.*

6.6. Tétel *Egy T, reprezentálható távolságmátrix esetén a vetületi reprezentációk körében a k dimenziós klasszikus közelítő ábrázolás az a k dimenziós ponthalmaz, amelyiknek D távolságmátrixa, az $\ell_1(T, D) = \sum_{i,j=1}^{n}(t_{ij} - d_{ij})$ értelemben a legjobban közelíti a T távolságmátrixot.*

Az ábrázolás hibája: $\sum_{i=k+1}^{n} \lambda_i(S)$ ahol a λ_i az $i = k+1, ..., n$-re az $n-k$ darab legkisebb sajátértéke az $S = -\frac{1}{2}CTC$ mátrixnak.

Bizonyítás. Az előző megjegyzés szerint a T közelítése a ℓ_1 távolság szerint ugyanaz, mint az S közelítése a $tr(S - V)$ távolság szerint. Mivel $tr(S)$ egyenlő az S sajátértékeinek összegével, a feladat a reprezentáció olyan vetületének megkeresése, amire $tr(V)$ maximális.

Legyen az X a T egy centrált reprezentációja, és legyen $P = QI_kQ^T$ egy tetszőleges projekció egy k dimenziós altérbe. Itt Q egy ortogonális transzformáció. Az I_k pedig az a vetítés, ami az első k koordináta által kifeszített altérbe vetít. Vagyis az I_k egy olyan $n \times n$-es mátrix ami mindenütt nulla, kivéve a főátlóban az első k elemét, ami egyenlő 1-el.

Ezekkel a jelölésekkel a $\frac{1}{2n}\sum_{i,j=1}^{n} d_{ij} = tr(V) = tr(P^T S P) = tr(Q I_k Q^T S Q I_k Q^T) = tr(I_k Q^T S Q I_k)$ ami nem más mint a $Q^T S Q$ mátrix $k \times k$-s főminorának nyoma ami a Poincare tétel szerint ([2], 141.o.) legfeljebb az első k legnagyobb sajátérték összegével egyenlő. Ebből pedig — figyelembe véve, hogy az S és a $Q^T S Q$ sajátértékei azonosak — az állítás következik.

6.3.2. Közelítés ℓ_2 normában

Megjegyzés. Az előbbi, 6.6. tétel erős megkötése, hogy csak reprezentálható távolság mátrixok közelítésére vonatkozik, és hogy a minimalizálást az ábrázoló ponthalmazok vetületeinek körében végzi. E megkötésektől való szabadulás érdekében módosítjuk a minimalizálandó távolságot: legyen a minimalizálandó távolság

$$\ell_2(T, D) = tr\left((S - V)^2\right).$$

Csakhogy ennek a távolságnak már nincs olyan szép, a t_{ij} közelítésére vonatkozó átalakítása, mint amilyen a ℓ_1 esetén volt. Legfeljebb az által értelmezhető, hogy az $\ell_2(T, D) = tr\left((S - V)^2\right) = \sum_{i,j=1}^{n}(s_{ij} - v_{ij})^2$ azaz hogy egyenlő a skalárszorzatok különbségének négyzetösszegével.

A következő tétel azt mutatja, hogy e jelentősen módosított feladat megoldása lényegében azonos az előző feladat megoldásával!

6.7. Tétel *Tetszőleges T távolságmátrix esetén a k dimenziós klasszikus közelítő ábrázolás az a k dimenziós ponthalmaz, aminek V skalárszorzat mátrixa az $\ell_2(T, D) = tr\left((S - V)^2\right)$ értelemben a T-hez tartozó S skalárszorzat mátrixhoz legközelebb van.*
A k dimenziós klasszikus közelítő ábrázolás hibája ebben a mértékben: $\sum_{i=k+1}^{n} \lambda_i^2(S)$

Jelölés. A bizonyításhoz bevezetjük a $x \preceq y$ jelölést, ami azt fogja jelölni, hogy az x szukcesszív kisebb mint az y. Ez azt jelenti, hogy $x_1 \leq y_1$, $x_1 + x_2 \leq y_1 + y_2, \ldots, x_1 + \ldots + x_{n-1} \leq y_1 + \ldots + y_{n-1}$ de $x_1 + \ldots + x_n = y_1 + \ldots + y_n$. És a Hardy-Littlewood-Pólya (1929) tétel szerint az $x \preceq y$ akkor és csak akkor áll fenn, ha létezik olyan P duplán sztochasztikus mátrix, amire $x = Py$. Az egyenlőség pontosan akkor érvényes, ha erre a P-re $P = I$.

Bizonyítás. Legyen az S csökkenő sorrendbe írt sajátértékeiből alkotott vektor λ, és a belőlük alkotott diagonális mátrix Λ. A V sajátértékeiből hasonló módon alkotott vektor illetve mátrix legyen l és L.

Diagonalizálja az S-t Q, a QVQ^T-t pedig R. Ekkor $tr\left((S - V)^2\right) = tr\left((\Lambda - QVQ^T)^2\right) = tr\left((\Lambda - RLR^T)^2\right) = tr(\Lambda^2) - 2\,tr(\Lambda R L R^T) + tr(L^2)$.

Tehát keressük azt az R ortogonális transzformációt és L diagonális mátrixot amire ez a mennyiség minimális. Rögzítsük L-t belátjuk, hogy a kifejezés a minimumát akkor veszi fel, amikor $R = I$. Pontosabban ha vannak egyenlő sajátértékek, akkor az R a megfelelő altérekben elforgatás is lehet.

A minimalizálási feladat nyilván azonos a középső tag maximalizálásával, ami a

$$tr(\Lambda R L R^T) = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i l_j r_{ij}^2 = \lambda^T R^{(2)} l = \lambda^T a$$

formába irható. Ahol az $R^{(2)}$ az R-ből elemenkénti négyzetre emeléssel adódik, és az $a = R^{(2)} l$. De ekkor a $0 \preceq a$ és mivel $\sum_{j=1}^{n} r_{ij}^2 = \sum_{i=1}^{n} r_{ij}^2 = 1$, az $R^{(2)}$ duplán sztochasztikus, $\sum_i a_i = \sum_i l_i$, továbbá a Hardy-Littlewood-Pólya tétel szerint $a \preceq l$. Könnyen látható, ha $a \preceq l$ akkor $\lambda^T a \leq \lambda^T l$, sőt ha az egyenlőség nem igaz, az egyenlőtlenség teljesül. Tehát ismét a H.L.P. tétel szerint $R^{(2)} = I$ vagyis $R = I$. Tehát rögzített L mellett a minimumot az S és a V sajátértékeiből alkotott párok különbségeinek négyzete adja. Ebből pedig, figyelembe véve a korábbi átalakításokat, és azt, hogy V-nek legfeljebb k nem nulla sajátértéke lehet, adódik az állítás.

Az eddig alkalmazott $tr(S - V) = \sum_{i,j=1}^{n} (t_{ij} - d_{ij})$ és $tr\left((S - V)^2\right) = \sum_{i,j=1}^{n} (s_{ij} - v_{ij})^2$ mátrix távolságon kívül, számos más — a gyakorlat szempontjából hasznos — mátrixtávolság esetén megoldható a távolság mátrix közelítési feladat. Csakhogy ezekben az esetekben a megoldás (általában) nem írható fel explicit módon. Ezért a megoldások megtalálásához valamilyen közelitő eljárást kell alkalmazni.

6.3.3. A távolságok függvényének közelítő ábrázolása

A távolság ábrázolási feladat általánosítható azáltal, hogy a távolságok közelítésére nem az ábrázolás távolságait alkalmazzuk, hanem azok egy (becsült paraméterű) függvényét.

Legegyszerűbb eset amikor t_{ij}-t a d_{ij} ismeretlen paraméterű *lineáris függvény*ével akarjuk közelíteni. Ekkor az a cél, hogy a

$$\sum_{i,j=1}^{n} (t_{ij} - (\beta d_{ij} + \alpha))$$

mennyiséget minimalizáljuk. Ha ezt a hibát tekintjük és mint lehetséges megoldást az összes n elemű ponthalmazt vesszük figyelembe, akkor nyilván azonos eredményt ad, ha a

$$\sum_{i,j=1}^{n} (t_{ij} - (d_{ij} + \alpha))$$

mennyiséget minimalizáljuk, vagyis ha azt keressük melyik ponthalmaz távolságai közelítik, távolságonként konstans hibával legjobban, az adott távolság mátrix távolságait.

6.8. Állítás *Legfeljebb $n-2$ dimenzióban minden távolságmátrix ábrázolható konstans hibával, [24].*

Bizonyítás. A bizonyítás azon alapszik, hogy a távolságok konstanssal való eltolása a skalárszorzat mátrix, nem u-hoz tartozó sajátértékeit konstanssal tolja el.
Jelölje a távolságmátrix a konstanssal való eltoltját T_a. Ekkor $T_a = T + a(U - I)$ és A T_a távolságmátrix skalárszorzatmátrixa:

$$S_a = -\frac{1}{2}CT_aC = -\frac{1}{2}C(T + a(U - I))C = S + \frac{1}{2}aC.$$

Ha S sajátvektorai az s_i $i=1,\ldots,n-1$ és az u, a megfelelő sajátértékek $\lambda_1 \geq \ldots \geq \lambda_{n-1}$ és 0. A sajátvektorok ortogonalitásából $Cs_i = s_i$, $i = 1,\ldots,n-1$. Tehát $S_as_i = (\lambda_i + \frac{1}{2}a)s_i$ és $S_cu = 0$, így ha $a = -2\lambda_{n-1}$, S_c az $(n - 1 - (\text{a} \quad \lambda_{n-1} \quad \text{multiplicitása})) < n - 2$ dimenziós térben reprezentálható.

A bizonyításból látható, hogy a távolságok konstanssal való módosítása esetén csak egy a-ra (a legkisebb nem az u-hoz tartozó sajátérték minusz kétszeresére) reprezentálható a távolság mátrix alacsonyabb dimenzióban. Nagyobb a értékre $n-1$ dimenziós reprezentációt kapunk, kisebbre pedig nem ábrázolható a távolságmátrix.

Ugyanakkor a tétel állítása szemléletesen is nyilvánvaló. Ha az ábrázolandó távolságok mindegyikéhez hozzáadunk egy igen nagy konstanst, akkor a feladat olyan n elemű ponthalmaz találása, amiben minden pontpár távolsága lényegében egyenlő.

Szokás azonban a konstans eltolás, azaz a lineáris függvény helyett más paraméteres függvény alkalmazása is, a következők szerint.

6.3.4. Közelítés általánosított feltételek mellett

Általános, analitikusan nem optimalizálható közelítési feltételek például a következők:

$$\sum_{i,j=1}^{n}(t_{ij} - d_{ij})^2$$

$$\sum_{i,j=1}^{n}\frac{(t_{ij} - d_{ij})^2}{t_{ij}}$$

$$\frac{\sum_{k,\ell=1}^{n^2}\delta_{k,\ell}|d_\ell - d_k|}{\sum_{k,\ell=1}^{n^2}|d_\ell - d_k|}$$

Itt a t_{ij} az ábrázolandó, és a d_{ij} az ábrázoló távolságokat jelöli. A d_k és a d_ℓ a d_{ij} távolságok egy tetszőleges sorrendben és a $\delta_{k,\ell}$ egy $0//1$ értékű indikátora annak, hogy a d_k és d_ℓ nagyságrendi sorrendje megegyezik-e a t_k és a t_ℓ nagyságrendi sorrendjével.

Ezeket a minimalizálási feladatokat gradiens módszerrel szokás megoldani. Felhasználható, hogy a fenti távolságokból adódó minimalizálandó függvények gradiensei explicit kiszámíthatóak.

Érdekes, hogy ezek a gradiens módszeren alapuló eljárások interpretálhatóak a következő módszer segitségével is.

A 'fogszabályozó' algoritmus. Felrajzolunk egy olyan ponthalmazt, ami a távolságokat nagyjából reprezentálja, és a pontok közé (gondolatban) rugókat illesztünk. A rugók feszességét annak függvényében szabályozzuk, hogy mi az ábrázoló és az ábrázolandó távolság viszonya: ha a két ábrázoló pont távolsága nagyobb mint az ábrázolandó távolság, akkor a rugó összehúz, ha kisebb, akkor pedig szétnyom (úgy mint a fogszabályozás esetén). Ezután rövid időre 'elengedünk' egy pontot. Azaz megengedjük, hogy az adott pont a hozzákapcsolt rugók irányának és erejének függvényében egy kicsit elmozduljon. Majd újrafeszítjük a rugókat az ábrázolási hiba mértékének megfelelően és ismét elengedünk egy pontot. Stb, addig ismételve a feszítés-elmozdulás ciklust, míg a pontok lényegében nyugalmi helyzetbe nem kerülnek.

Legyen adott a T távolságmátrix. Legyen a reprezentáció távolságmátrixa D. Legyen f_θ, g_θ illetve r_θ ismert, egy illetve kétváltozós paraméteres függvény. Keressük a T egy olyan k dimenziós ábrázolását és azt a $\theta \in \Theta$ paramétert, amire a

$$\sum_{ij}^{n}(d_{ij} - f_\theta(t_{ij}))^2, \text{ vagy a } \sum_{ij}^{n} g_\theta(d_{ij} - t_{ij}),$$

vagy általánosabban az

$$r_\theta(T, D)$$

minimális. Az ilyen típusú feladatok a következő struktúrájú algoritmusokkal oldhatók meg.

Egy $X_0, X_1, ...$ ponthalmaz sorozatot képezünk. Legyen X_0 egy tetszőleges kezdeti konfiguráció, például a k dimenziós klasszikus közelítő ábrázolás. Ha ismert az X_i akkor az X_{i+1} ponthalmazt a következő lépésekben határozzuk meg:
1. legyen θ_{i+1} az a θ amire a $r_\theta(T, D_{X_i})$ minimális
2. legyen X_{i+1} az a k dimenziós X ponthalmaz, amire $r_{\theta_{i+1}}(T, D_X)$ minimális

A mondott modellcsoport egy érdekes esete, amikor az f_θ egy tetszőleges monoton függvény. Ekkor a feladat a jegyzet második részében ismertetett monoton regresszió felhasználásával, egy speciális iteratív eljárással elegánsan megoldható.

Felmerül a kérdés, nem lehet-e a távolságokat általában monoton reprezentálni. Vagyis nincs-e egy olyan ábra ami a nagyobb távolságokat nagyobb, a kisebbeket pedig kisebb távolsággal reprezentálja.

A konstans hibával való ábrázolhatóság előbb tárgyalt tételének egyszerű következménye, az az egyáltalán nem természetes tény, hogy $\mathbb{R}^{n-2}$-ben mindig található monoton ábrázolás.

Ugyanakkor a következő feladatot megoldva láthatjuk, hogy ez alacsonyabb dimenzióban általában nem oldható meg. Emiatt legfeljebb olyan ábrázolásokra célszerű törekedni, amikben a monotonitás kevés helyen csorbul.

Feladat. Mutassuk meg, hogy van olyan tetraéder, aminek csúcstávolságai a számegyenes semmelyik négy pontjával sem reprezentálhatók monoton módon. Bizonyítsuk be, hogy van olyan pont n-es ami $n-2$-nél alacsonyabb dimenzióban monoton nem ábrázolható.

6.4. Az elmélet demonstrációja

6.4.1. Egy háromszög és a köré irható kör

Adott távolságmátrix szerinti háromszög és tetraéder, valamint a köréjük irható kör és gömb.

Bemutatjuk egy véletlengenerátorral előállított háromszögön az eddig ismertetett fogalmakat. Numerikusan és grafikusan interpretáljuk, hogy a képletek helyesek.

```
# ==============================
# a véletlen generátor kezdôértéke

set.seed(123)

# ------------------------------
# vegyünk három síkbeli pontot

A <-rnorm(2)
B <-rnorm(2)
```

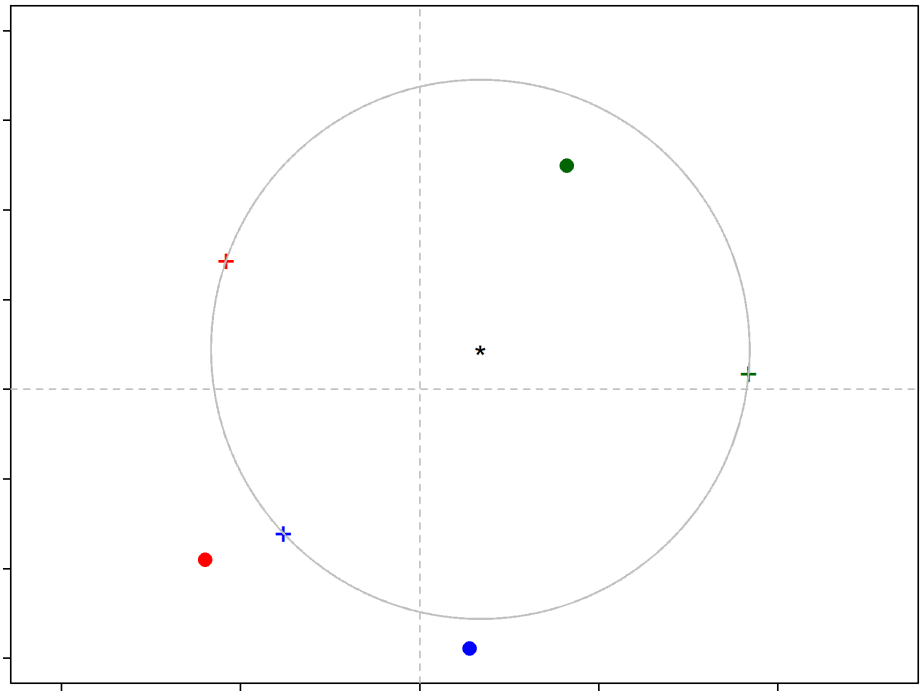
```
C <-rnorm(2)

# -------------------------------
# eltoljuk a súlypontot az origóba

xa<-(A[1]+B[1]+C[1])/3
ya<-(A[2]+B[2]+C[2])/3

A<-A-c(xa,ya)
B<-B-c(xa,ya)
C<-C-c(xa,ya)

# -------------------------------
# a pontok távolságmátrixa

T<-matrix(0,nrow=3,ncol=3)
rownames(T)<-colnames(T)<-c("A","B","C")

T[1,2]<-T[2,1]<-sum((A-B)^2)
T[2,3]<-T[3,2]<-sum((B-C)^2)
T[1,3]<-T[3,1]<-sum((A-C)^2)

print(T)

# -------------------------------
# a skalárszorzat mátrix

M<-diag(rep(1,3))-
             matrix(1,nrow=3,ncol=3)/3
S<- -M%*%T%*%M/2

print(S)

# -------------------------------
# a klasszikus reprezentáció

W<-eigen(S) # a sajátértékek és vektorok
X<-diag(sqrt(W$va[-3]))%*%t(W$ve[,-3])
# a harmadik az u es a hozzátartozó 0

print(X)
```

```
# ------------------------------
# a skalárszorzat mátrix jó...

round(t(X)%*%X-S,4) # ez kb nulla

# ------------------------------
# a reprezentáció távolságmátrixa:

D<-matrix(0,nrow=3,ncol=3)
D[1,2]<-D[2,1]<- sum((X[,1]-X[,2])^2)
D[2,3]<-D[3,2]<- sum((X[,2]-X[,3])^2)
D[1,3]<-D[3,1]<- sum((X[,1]-X[,3])^2)

round(D-T,4) # ez kb nulla

# ------------------------------
# a reprezentáció köré írt kör sugara

u<-matrix(rep(1,3))
rm<-t(u)%*%solve(T)%*%u
r<-sqrt(1/(2*as.numeric(rm)))

print(r)

# ------------------------------
# a köré írt kör középpontja

s<-2*r^2*solve(T)%*%u # súlyozás mellett
K<-X%*%s

print(K)

# ==============================
# felrajzoljuk...
#   a pontokat, a reprezentációt
#   es a köréírható kört

szin<-c("red","blue","darkgreen")
h<-c(-3,3)
```

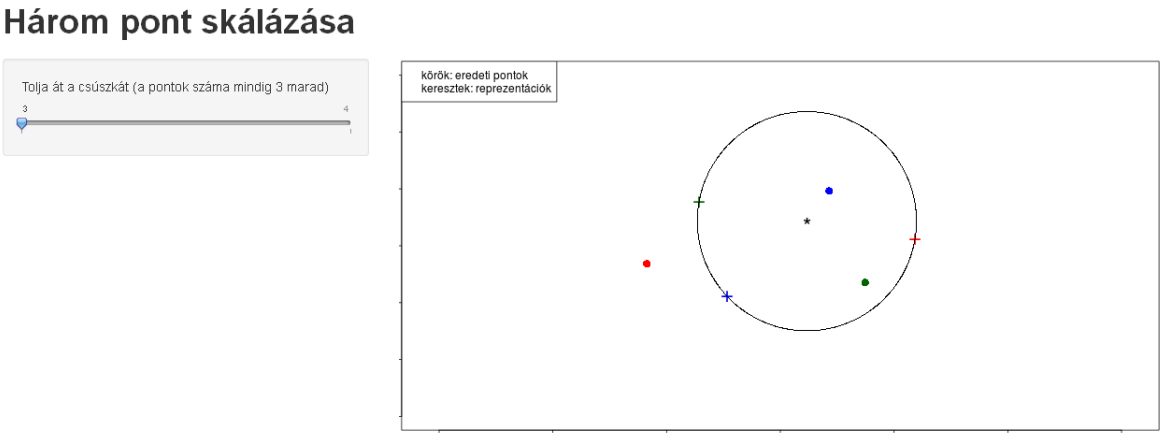
```
# ------------------------------
# a három eredeti pont

x<-c(A[1],B[1],C[1])
y<-c(A[2],B[2],C[2])
par(mar=c(.2,.2,.2,.2))
plot(x,y,xlim=h,ylim=h,
         xlab="",ylab="",t='n')
points(x,y,col=szin,pch=16)

# ------------------------------
# a három reprezentáló pont képe

points(X[1,],X[2,],col=szin,pch="+")

# ------------------------------
# a reprezentáció köré írt
#              kör és középpontja

v<-seq(0,2*pi,l=1001)[-1]
vx<-r*cos(v)
vy<-r*sin(v)
points(K[1]+vx,K[2]+vy,col="black",pch=".")

points(K[1],K[2],col="black",pch="*")

# ------------------------------
# A középpont távolságnégyzete
#                     a pontoktól

colSums((X-cbind(K,K,K))^2)
```

A fenti programban — technikai okokból — M-el jelöltük az elméleti ismertetőben C-vel jelölt centráló mátrixot, és c helyett s-el az interpretációs kör középpontját. A programsorok grafikus eredménye a 6.1 ábrán látható.

Ehhez a modellhez animáció is készült, amely a http://hpz400.cs.elte.hu:3838/ZA_skala/ címen található. Itt azt vizsgálhatjuk, hogy különböző ponthármasok mennyire különböző eredményt adnak. A bal oldali csúszka nem is játszik szerepet, csak azért kell elmozdítani, hogy új futási eredményt kapjunk.



6.1. ábra. A színes pöttyök jelölik az eredeti háromszöget, a keresztek az interpretációt



6.2. ábra. Animációs ábra a háromszög-skálázásról

6.4.2. A patkóeffektus interpretációja

Ívesedési hajlamnak — patkó effektusnak — azt a skálázás során gyakran tapasztalható jelenséget nevezik, hogy az objektumokat reprezentáló pontok a 2D ábrán kis ívekre ren-

deződnek. A jelenség rövid magyarázata az, hogy az objektum ábrázolt részhalmazának tényleges távolságai kis hibával eleve egy alacsonyabb dimenziós térben interpretálhatóak. Emiatt az interpretációs pontok az $n-1$ dimenziós gömbnek egy alacsonyabb dimenziós kör (vagy gömb) szeletéhez közel helyezkednek el.

Az ívesedési hajlam kellemetlen következménye, hogy például olyan objektumok esetén, amik a 'legközelebbi szomszéd' elve alapján valójában egy sorozatot alkotnak, a skálázással nyert 1D ábrán nem a 'valós' sorrendjükben szerepelnek. Látványosan adódik ilyen probléma régészeti és ökológiai adatok feldolgozásakor.

Ha egy terület régészeti leleteit vesszük, akkor e leletek egy természetes sorrendje a leletek kora. Ám ha a rendelkezésre álló másodlagos adatok alapján a leletek közelítő kordifferenciái alapján skálázással akarunk a leletek közt sorrendet megállapítani, azaz ha vesszük a feltételezett kordifferenciák skálázással nyert 1D képét, akkor jó eséllyel azt tapasztalhatjuk, hogy a módszer nem a legrégebbit minősíti a legrégibbnek, és nem a legújabbat a legújabbnak.

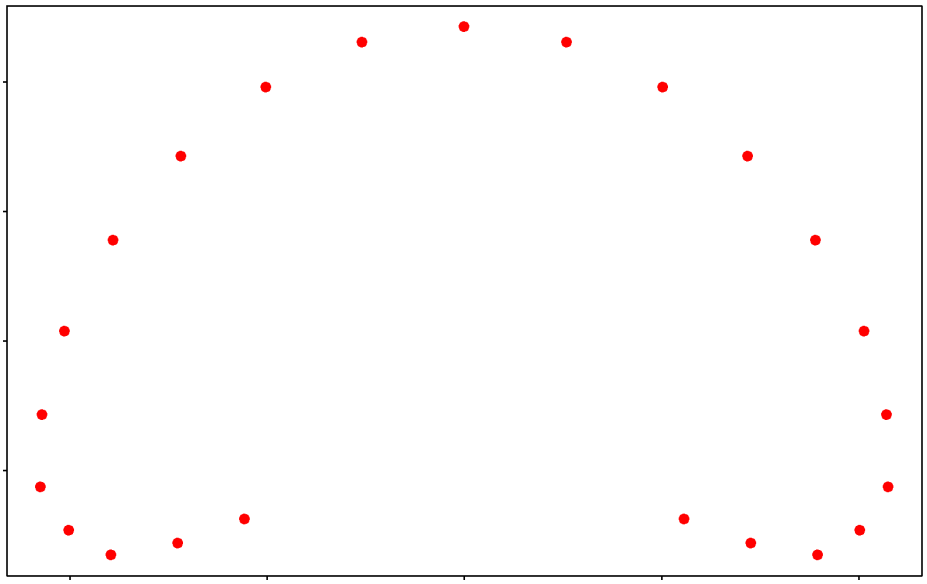
De ha a leleteknek nem 1D, hanem 2D képét vesszük, akkor láthatóvá válik a leletek valós sorrendje és egyben az is, hogy mi okozta az 1D interpretáció esetén a problémát. Ugyanis a tárgyak 2D képén a reprezentáló pontok jó eséllyel egy behajló végű patkón helyezkednek el. Az ilyen íveknek pedig nincs is olyan vetületi iránya ami a pontokat helyes sorrendben képezné le.

Hasonló a probléma az úgynevezett ökológiai gradiens keresésekor. Ekkor ugyanis a táj változásának van egy egyértelmű iránya. Emiatt a vizsgált területek sokdimenziós ökológiai leírása erőteljesen változik egy tényleges földrajzi irányban. Ugyanakkor az egyes vizsgált területek mint objektumok skálázott képe hasonló módon a régészeti esethez, nem 1D hanem csak 2D (vagy esetleg valamely még magasabb) reprezentációban mutatják csak be ténylegesen a sorrendiséget.

A következő programrészlet az ilyen esetekre jellemző hasonlóság és az abból számolt távolságmátrix esetén mutatja meg, hogy az interpretáció tényleg egy patkón helyezkedik el. Ugyanakkor például a 'cmdscale(T,k=1)' paranccsal ellenőrizhető, hogy ha a távolságok 1D képét veszünk, akkor a pontok sorrendje már nem felel meg ennek a természetes sorrendnek.

```
n<-23; k<-2; m<-2
H<-matrix(rep(0,n*n),n,n); # hasonlóság
T<-matrix(rep(0,n*n),n,n); # távolság

for (i in 1:n) for (j in 1:n)
```

```
  H[i,j]<-(max(c(k^2-ceiling(abs(i-j)/k),0)))^m

for (i in 1:n) for (j in 1:n)
  T[i,j]<-sqrt(H[i,i]-2*H[i,j]+H[j,j])

patko<-cmdscale(T,k=2)
plot(patko,pch=20,col="red")
```



6.3. ábra. Ezeknek az objektumoknak nincs sorrendhelyes 1D vetülete

6.5. Skálázást végző R programok

6.5.1. A 'stats::cmdscale()' eljárás

A 'datasets::eurodist' egy 'dist' osztályú adathalmaz, ami 21 európai nagyváros egymástól mért közúti távolságaiból áll.

A 'dist' osztályú változók szimmetrikus mátrixok adatait tartalmazzák. Az 'eurodist' a 'dist' osztályú változók hat lehetséges attribútuma közül kettővel rendelkezik. Ezek a 'Labels' ami esetünkben a szóbanforgó 21 város neve továbbá a kötelezően jelenlévő 'Size' attribútum, aminek az értéke az esetünkben 21.

Mivel egy 'dist' objektum feltételezi, hogy a szóbanforgó (távolság) mátrix szimmetrikus, és olyan, amilyennek a diagonálisa nulla, az objektumban a mátrixnak csak az alsóháromszög része van tárolva, oszlopfolytonosan.

A 'dist' osztályú változók további lehetséges attribútumai a 'Diag' és az 'Upper' ami csak azt érinti, hogy a 'dist' osztályhoz tartozó 'print' rutin kiír-e nullákat, a diagonális feltételezett elemeinek megfelelően, illetve hogy a kiíráskor kiírja-e a diagonális feletti pozíciókhoz tartozó (a diagonális alatti értékekkel azonos) felsőháromszögbeli értékeket is.

A 'call' és 'method' további lehetséges attribútumok szokványos információkat tartalmaznak. Az utóbbi például azt, hogy a távolságok milyen módszerrel származnak a többdimenziós pontok adatai alapján.

A 'method' tipikus értéke például az, hogy a távolság az "euclidean" vagy a "minkowski" azaz euklideszi vagy L_p, "maximum", azaz a maximális koordináta különbség, "manhattan" azaz a koordináta különbségek összege. A "canberra" távolságot a $\sum(|x_i - y_i|/|x_i + y_i|)$ képlettel számolhatjuk. Ha az objektumok vektorai 0//1 vektorok, akkor a "binary" távolság azon pozíciók hányada, amelyekben a két bit különböző.

A 'dist' objektumot n darab k dimenziós pont $k \times n$ méretű adat mátrixából tipikusan a 'dist()' eljárás segitségével hozhatjuk létre.

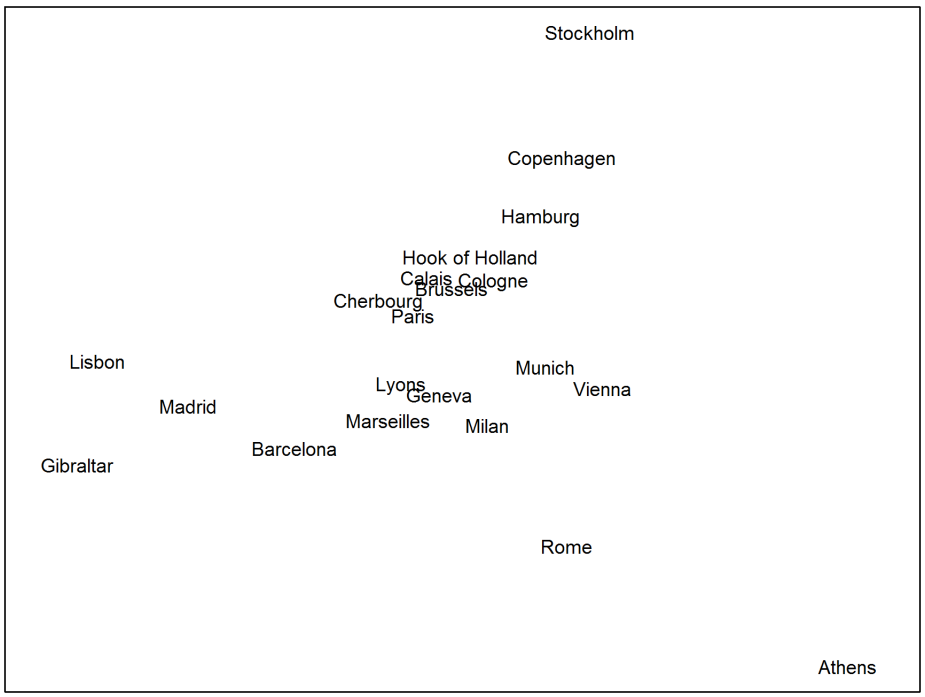
A 'dist' osztályú változókat az 'as.matrix()' parancs szimmetrikus 0 diagonálisú mátrixokká alakítja. A következőkben bemutatott skálázó eljárások a bemenetként megadott távolságmátrixokat ilyen (teljes, szimmetrikus, nulla diagonálisú) mátrixok formájában is elfogadja.

A 'stats::cmdscale()' a legegyszerűbb **R** -beli skálázó függvény. A következő módon futtattuk le:

```
P<-cmdscale(eurodist)
plot(P,type="n",main="Városok",
     xlab="",ylab="")
text(P, rownames(P), cex=0.8)
```

A kapott 6.4 ábrán a városok nagyjából úgy helyezkednek el, ahogyan azt a térképeken megszoktuk.

Az első parancs a skálázás eredményeit a P változóba menti, az ábrát pedig ennek alapján a két utóbbi parancs állítja össze. A második parancs egy 'üres' grafikus környezetet



6.4. ábra. Európai városok 'stats::cmdscale()' paranccsal skálázott képe

nyit, az utolsó pedig a sornevekként visszakapott városneveket az ábra P által megadott pontjaira írja.

A 'cmdscale()', ha nem alkalmazzuk az 'add=TRUE' opcióját akkor a klasszikus interpretáció alapján működik. Veszi a megadott távolságoknak megfelelő skalárszorzat mátrixot. Veszi a skalárszorzat mátrix sajátértékeit csökkenő sorrendben és a hozzájuk tartozó egységhosszú sajátvektorokat. A keresett ponthalmazt a sajátértékek gyökeivel beszorzott sajátvektorokból állítja össze. Annyi sajátvektort használ fel, ahány dimenziós reprezentációt keresünk. A közelítés dimenziója az alapértelmezés szerinti esetben $k = 2$.

A parancs eredményeként kapott objektum 'matrix' osztályú, hacsak nem állítjuk az interpretációs dimenzión (ez a 'k=' opció) kívüli három lehetséges paraméter legalább egyikét 'TRUE' értékre.

```
M <- cmdscale(eurodist, k=20,
                  eig = TRUE,
                  x.ret = TRUE,
                  add = TRUE)
str(M)
```

Ez utóbbi esetben az eredmény egy 5 elemű lista a következők szerint:

points	az interpretáció $n \times k$ méretű mátrixa,
eig	k db sajátérék,
$x	a skalárszorzat mátrix * -2,
$ac	additív konstans,
$GOF	jósági mérték.

Mint látható, az interpretációs pontokat az eredmény `$points` $n \times k$ méretű mátrixa tartalmazza. Azaz az eredmény egy-egy *sora* azonos egy-egy interpretációs ponttal.

A függvény hibát jelez, ha kevés a 'pozitív' dimenzió — azaz a megadott távolságmátrixhoz tartozó skalárszorzat mátrixnak kevesebb nem-negatív sajátértéke van, — mint ahány dimenziós interpretációt keresünk.

Az esetleges interpretálhatósági problémán a 'cmdscale()' esetén kétféle képpen segíthetünk. Vagy csökkentjük az interpretálási dimenziót, vagy pedig az 'add=TRUE' opció felhasználásával olyan interpretációt kérünk, ami konstans hibával közelíti (jól) a megadott távolságokat.

A 'cmdscale()' eljárás az 'add=TRUE' opció mellett F. Cailliez modellje és módszere alapján talál egy olyan, 'ac' paraméterként visszaadott konstanst amivel megnövelve az ábrázolandó távolságokat a pontpárok távolságait jól közelítő ábra készíthető.

Az 'add=TRUE'-nak megfelelő additív konstans modell akkor is használható, ha a távolságoknak van ténylegesen k dimenziós közelítő megoldása. De az eljárás ekkor sem ad a konstans nélküli modellel feltétlen azonos megoldást.
A 'cmdscale()' ugyanis az 'add=TRUE' opció mellett azt az 'ac' konstanst veszi, amivel a távolságértékeket (tehát nem a távolságnégyzet értékeket!) megnövelve olyan skalárszorzat mátrix adódik, aminek nincs negatív sajátértéke.

Az eredménybe az 'eig=TRUE' opcióval kérhető két szám azt mutatja, hogy a nyert ábra (közelítés) mennyire jó modellje az eredeti távolságadatoknak.

Legyen k az interpretáció dimenziója, és $\mu_1, ..., \mu_n$ csökkenő sorrendben az — adott esetben additív konstanssal növelt — távolságmátrixhoz tartozó skalárszorzatmátrix sajátértékei és legyen m az a legnagyobb index amire a μ_m még pozitív.

Ezekkel a jelölésekkel a két mutató értéke:

$$\sum_{j=1}^{k} \mu_j / \sum_{\ell=1}^{n} |\mu_\ell| \text{ illetve } \sum_{j=1}^{k} \mu_j / \sum_{\ell=1}^{m} \mu_\ell.$$

Vagyis az első azt méri, hogy a interpretációs pontok az adatokban meglévő információ hányad részét reprezentálják. A második pedig azt, hogy ugyanez az interpretált információ az *interpretálható* információnak hányad része. A két mutató tehát mindig $[0, 1]$-beli. A nagyobb számok jelentik a pontosabb interpretációt. A második mutató mindig nagyobb vagy egyenlő. De ha az 'add=TRUE' opciót alkalmazzuk, akkor a két mutató értéke (természetesen) egyenlő.

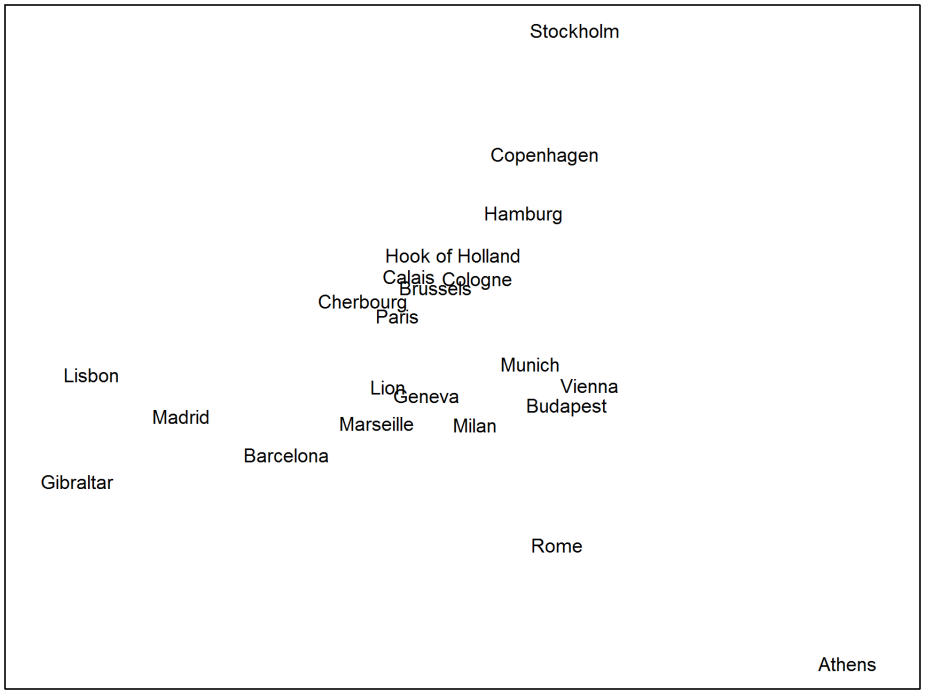
Mint látható a beépített 'eurodist' adathalmaz városai közé nem vették fel Budapestet. Adjuk tehát hozzá a beépitett adatokhoz a magyar fővárost is!

Minthogy a 'dist' osztályú változók a távolságmátrix elemeit oszlopfolytonosan tartalmazzák (látható ez a 'as.numeric(eurodist)' parancs eredményén is) legegyszerűbb az 'új' várost elsőként megadni, a következők szerint:

```
szam<-attributes(eurodist)$Size
tobbi<-attributes(eurodist)$Labels
# tobbi<-labels(eurodist) # így is jó
osztaly<-class(eurodist)

nevek<-c("Budapest",tobbi)
# két város nevét átírjuk
nevek[c(14,16)]<-c("Lyon","Marseille")

varosok<-
 c(c(1124,1501,1133,1418,1538,959,1013,
     992,2373,929,1156,2474,1103,1978,
     1164,789,564,1248,812,1318,217),
   as.numeric(eurodist))
attributes(varosok)$Size<-szam+1
attributes(varosok)$Labels<-nevek
class(varosok)<-osztaly

P<-cmdscale(varosok)
# az elsô koordinátát az irány-
#          helyességhez tükrözni kell
P[,1]<- -P[,1]

plot(P,t='n',axes=FALSE,xlab="",ylab="")
text(P,nevek,cex=.8)
```

A fenti programsorok egy új adatobjektumot építenek, és két francia város nevét angolról francia helyesírásúra irják át. Az eredményt az alábbi ábra mutatja. Budapest kissé

nyugatra csúszott...



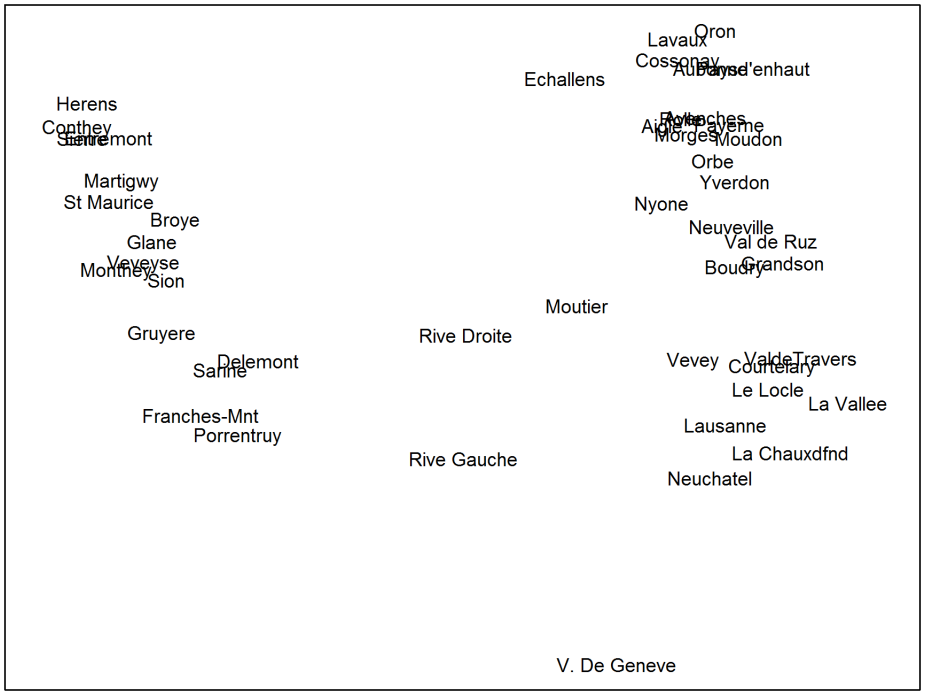
6.5. ábra. Európai városok képe Budapesttel kiegészített távolságmátrix alapján

6.5.2. A 'MASS::sammon()' eljárás

Ez az eljárás az **R** rendszer alapkiépítéséhez tartozó 'MASS' csomag része, ami ugyan telepítődik az alaprendszerrel együtt, ám a rendszer indításakor mégnem töltődik be automatikusan. Ezért e függvény alkalmazása előtt vagy be kell tölteni a 'MASS' csomagot, vagy pedig a függvényt — az őt tartalmazó csomagot is megcímezve, — a 'MASS::sammon()' paranccsal kell meghívni.

Példaként futtassuk a következő parancssort.

```
require(MASS)
D<- dist(as.matrix(swiss[, -1]))
M <- sammon(D)
plot(P, type = "n")
text(P,rownames(P))
```



6.6. ábra. Svájci tartományok 'sammon()' eljárással nyert képe

A kapott 6.6 ábrán svájci tartományok nevei láthatóak. A 'swiss' adathalmaz a 47 francia-ajkú svájci tartomány 6 dimenziós szociológiai leirását tartalmazza. Ebből vettünk ki öt olyan adatsort, ami ezeknek a tartományoknak az 1888-as állapotát tükrözi. Az ábra a tartományok 2-3 csoportra bomlását mutatja.

Ha az adatok elemzése volna a feladat, akkor következő lépésként vizsgálhatnánk, hogy vajon van-e a megadott öt tényezőnek olyan részhalmaza, amik ezt a csoportra bomlást dominálják. Ehhez például előbb klaszterezni kell a kapott pontokat. Utóbb pedig például ANOVA módszerrel megvizsgálni azt, hogy hogy egyik vagy másik változó csoport értéke különböző-e a kapott csoportokban.

A Sammon féle távolság
Ha a skálázandó távolságokat a $t_{i,j}$ a reprezentáló távolságokat pedig a $d_{i,j}$ jelöli, akkor a közelítés az

$$\text{stress} = \frac{1}{\sum_{i<j} t_{ij}} \sum_{i<j} \frac{(t_{ij} - d_{ij})^2}{t_{ij}}$$

Sammon-féle távolságot minimalizálja.

Ez a mennyiség mint látható a relatív hibák összege azt feltételezve, hogy egy-egy távolság hibája (szórása) a távolság gyökével arányos. Az összeg előtti konstans a minimalizálandó mennyiség skála-invarianciáját biztosítja.

A 'stress' távolság optimalizálásának módja a 'stress' parciális deriváltjain alapuló iteratív eljárás. A program a futtatása közben nyomkövető kiírást készít, hacsak nem tiltottuk ezt le a ' trace = FALSE' opciót alkalmazva. A kiíráson látható a 'stress' csökkenő értéke és az alkalmazott Newton módszer lépéshossza is 'magic' felirattal.

A 'sammon()' függvény által készített eredmény változó egy 3 elemű lista a `$points`, `$stress` és a `$call` elemekkel. Az első komponens egy $n \times k$ méretű mátrix a reprezentáló pontok koordinátáival. A második egy valós szám, a reprezentáció jóságát leiró 'stress' értékkel. A harmadik a meghívás módját tartalmazza.

A 'stress'-t a következők alapján értékelhetjük. Ha minden távolságot 0 távolság reprezentál, akkor a 'stress' értéke 1. Ennél nyilván kisebb, ha minden távolságot lehet pl a távolságok átlagértékével reprezentálni. De ha sikerül minden távolságot hiba nélkül reprezentálni, akkor a 'stress' értéke nulla.

A mondott mennyiség 'stressz' elnevezése tipikus a skálázás szakirodalmában. Leginkább 'feszültség'nek fordíthatjuk. Az elnevezés elterjedt használatát az magyarázza, hogy a skálázás módszerét eleinte leginkább pszichológusok alkalmazták.

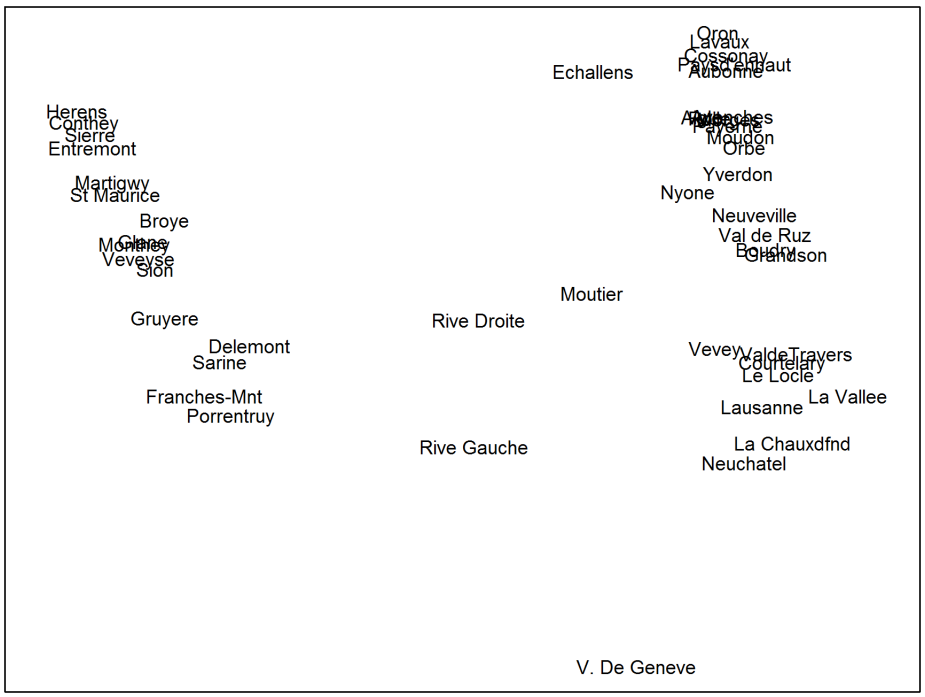
Egy ábrázolás tehát annál jobb mennél kisebb a kapott, $[0, 1]$ intervallumba eső 'stress' érték.

6.5.3. A 'MASS::isoMDS()' eljárás

Az 'MASS::isoMDS()' eljárás is az **R** alaprendszeréhez tartozó skálázó eljárás. A ténylegesen közelítendő távolságok megállapításához a monoton regresszió módszerét használja fel.

Használatához a programot — minthogy a 'MASS' nem tartozik az alapértelmezés szerint betöltődő programok közé, — vagy előzőleg be kell tölteni a 'MASS' csomagot, vagy pedig magát a programot kell az őt tartalmazó csomagot is megjelelő hosszabb 'MASS::isoMDS()' módon meghívni.

Ha lefuttatjuk az alábbi programrészletet, a következő 6.7 ábrát nyerjük.



6.7. ábra. Svájci tartományok 'isoMDS()' eljárással nyert képe

```
require(MASS)
D<- dist(as.matrix(swiss[, -1]))
M <- isoMDS(D)
plot(P, type = "n")
text(P,rownames(P))
```

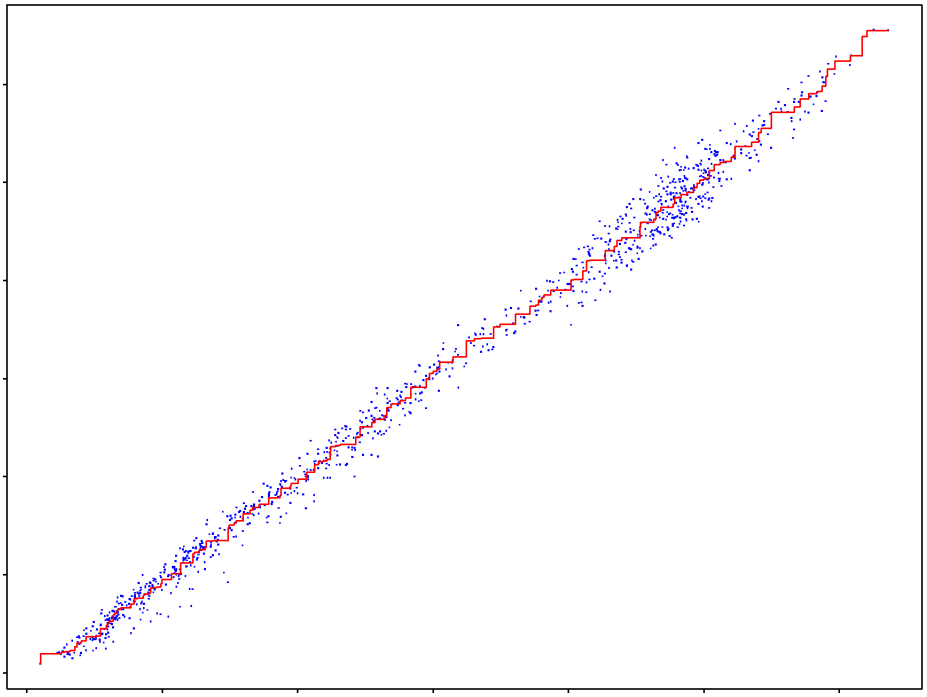
A módszer demonstrációjára ugyanazt a 'swiss' adathalmazt használtuk, mint a 'sammon()' esetében. Azaz 47 svájci tartomány 5 változóval való leirását tekintettük. Látható, hogy esetünkban az ábra csak kicsit, lokálisan változott.

Ugyanakkor az 'isoMDS' módszere lényegesen eltér a korábbiaktól. Ez az eljárás a távolságok egy iteratív eljárással meghatározott monoton regressziójához tartozó optimális közelítést keres meg, a következők szerint.

Lépésenként veszi az aktuálisan nyert reprezentáló távolságok — 3.4 részben ismertetett — monoton regresszióját a reprezentálandó távolságok szerint. A lépésenkénti új reprezentáló konfigrációt pedig az előző lépésben nyert regressziós távolságok alapján határozza meg.

Egy-egy skálázással nyert kép minőségének fontos diagosztikai eszköze a Shepard diagram: 6.8 ábra. Ennek készítési módját az előzőekben felhasznált D távolságmátrix és az ugyanott nyert M modell reprezentáló pontjait felhasználva mutathatjuk be:

```
Sh<- Shepard(D,M$points)
plot(Sh, pch = ".")
lines(Sh$x, Sh$yf, type = "S")
```



6.8. ábra. Shepard diagram: a mért és ábrázoló távolságok viszonya a 6.7 ábrán

Az ábrán $1081 = 47*46/2$ pont látható, mert 47 objektum adatainak skálázásával foglalkoztunk, és 47 objektumra ennyi az objektumpárok különböző távolságainak a száma. A pontok x-koordinátája az objektumok közti ábrázolandó távolságok (ez a program szerint nyert 'Sh' lista 'x' eleme, ami egyébként egy rendezett vektor). A pontok y-koordinátája pedig a megfelelő objektumpár távolsága a reprezentáció szerint.

Azaz ha a 6.8 ábrán egy pont távol van az $y = x$ egyenestől akkor az azt jelenti, hogy a megfelelő objektumpár távolsága rosszul reprezentált. Az ábrán látható lépcsősfüggvény a pontok monoton regressziója, aminek adatait az 'Sh$yf' tartalmazza.

A 'Shepard()' eljárás tehát lényegében egy monoton regresszió azon adatpárok közt, amit egyrészt az első paraméterként megadott reprezentálandó távolságmátrix, másrészt

a másodikként megadott reprezentáló ponthalmazhoz tartozó távoságmátrix egymásnak megfelelő elemei adnak meg.

6.5.4. A 'SensoMineR::indscal()' eljárás

A negyedik ismertetett eljárás a 'SensoMineR' kiegészítésben [15] található 'indscal()' eljárás, ami többdimenziós preferencia adatok kiértékelésére alkalmas.

A 'SensoMineR' csomag 5 további csomag installált voltát feltételezi, ('FactoMineR', 'ellipse', 'lattice', 'cluster', 'scatterplot3d') mivel a csomag bizonyos elemei egy-egy eljárást felhasználnak ezekből a kiegészítésekből is.

```
rm(list=ls())
require('SensoMineR')
data(napping) # két adathalmaz:
     # napping.don, napping.words
par(mar=c(1,1,1,1))
nappeplot(napping.don,3,4)
```

Tíz francia bor egymáshoz viszonyított minősítéséhez 11 kóstolót kértek meg arra, hogy mindegyikük helyezzen el egy (60,40) méretű asztalon 10, a borok mintáit tartalmazó poharat. A minősítők döntéseit a $10 \times (2 * 11)$méretű 'napping.don' 'data.frame' tartalmazza. Az első kóstoló például a következő ábra szerint helyezte el a poharakat.

Az előző utasítássor által betöltött másik adathalmaz a 'napping.words', ami egy 10×14 méretű gyakoriság tábla 'data.frame' formában tárolva.

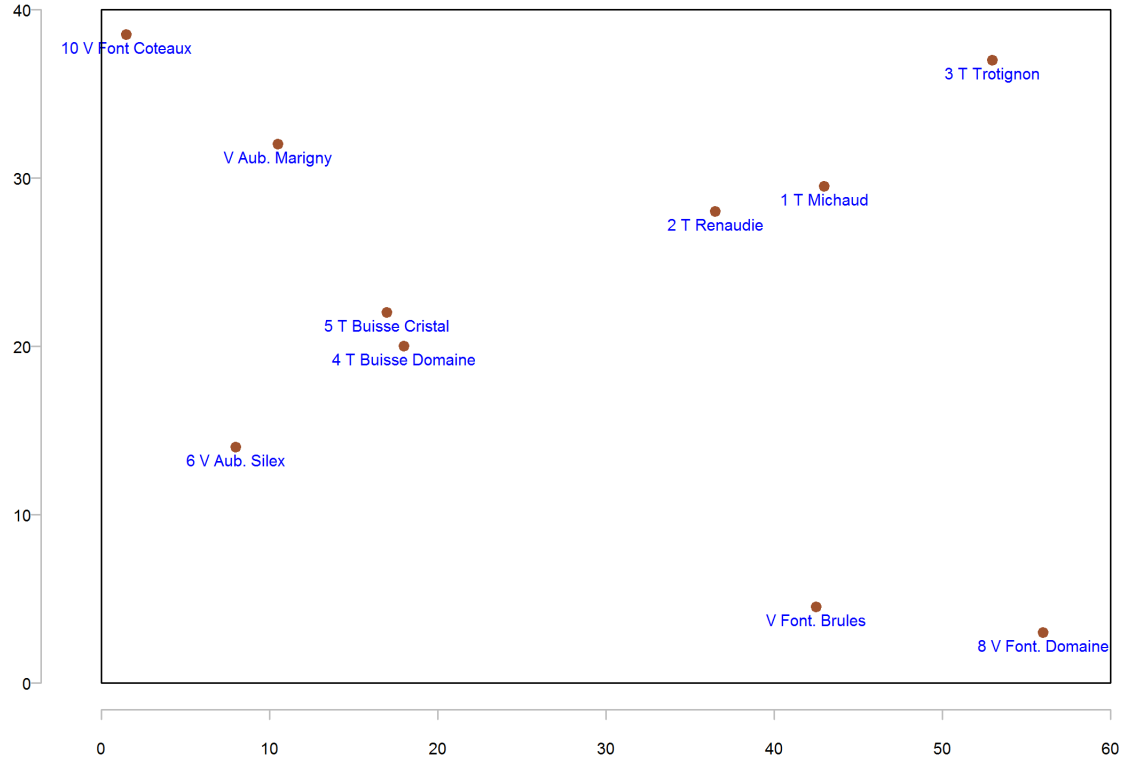
Ebben a táblában az oszlopok a következő bortulajdonságoknak felelnek meg: "Wood", "Liqueur like", " Fresh-Sharp", " Fruity", " Soft", " Discrete", " Intense", " Grilled bread", " Floral", " Light", " Bitterness", " Green", " Acid", " Yellow". A táblabeli gyakoriságok pedig azt mutatják, hogy az egyes borok esetén a 11 kóstoló közül hány itélte úgy, hogy az adott tulajdonság az adott borra jellemző.

Futtassuk le ezekre az adatokra a 'SensoMineR' csomag 'indscal()' parancsát a következő módon!

```
M<-indscal(napping.don,napping.words)
```

Eredményként három ábrát (6.10, 6.11, 6.12) és egy 5 elemű listát nyerünk.

A visszakapott M, 5 elemű lista elemei a következők:



6.9. ábra. A 10 bor az *első*, az Y1 kóstoló véleménye alapján

W	a 11 itész (subject) minősítése 11×2,
points	a borok (stimuli, individuals) 10×2,
subvar	a fesz a közös és az egyéni döntések közt 11,
r2	a fesz átlagos értéke,
dfr	az 'r2' szabadságfoka.

A '$subvar' egy R^2 érték, az itészek egyéni döntései és a közös döntés (elhelyezési konfiguráció) közti korreláció négyzete. Azaz annyi, mint amennyit egy-egy itész konfigurációjában lévő varianciából a közös konfiguráció megmagyaráz. A '$r2' a '$subvar' értékek átlagával egyenlő. Ezek a fesz (azaz stress) értékek $[0, 1]$-beli számok. Akkor jó a modell, ha a stressz (azaz fesz) értékek 1 körüliek.

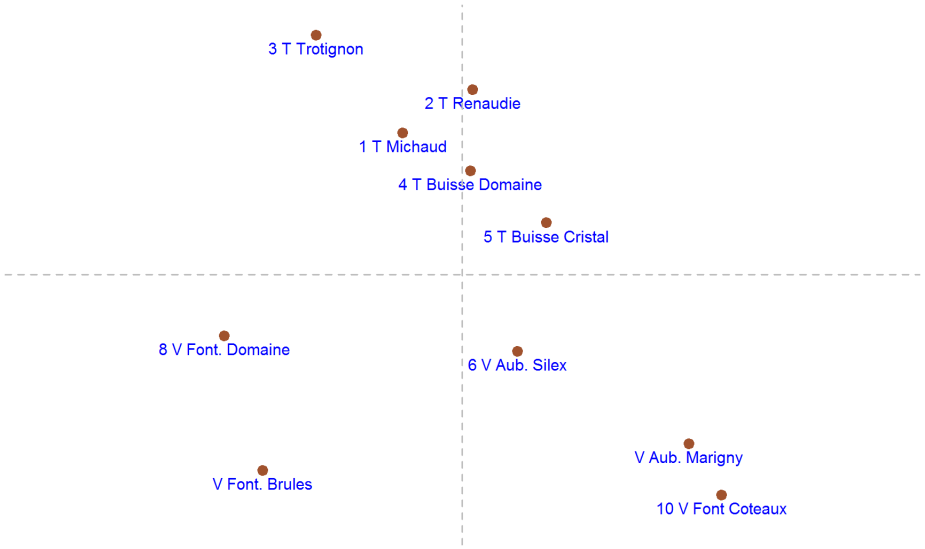
A 'dfr' értéke

$$\text{dfr} = \frac{k(m+n-2)}{mn(n-1)/2}$$

ahol k a reprezentáció dimenziója, ami az esetünkben 2. Az n a stimulusok száma

azaz most 10, az m a subjecteké ami most 11 emiatt a 'dfr' értéke az esetünkben $38/495 = .07676768$.

Az előző 'indscal()' paranccsal nyert rajzok közül az elsőt a 6.10 ábra mutatja be. Ezen az látható, hogy a 10 bornak mi az az elrendezése, ami mint a 11 kóstoló közös minősítése értelmezhető. Adatai az eredmény változó '$points' elemében találhatóak.



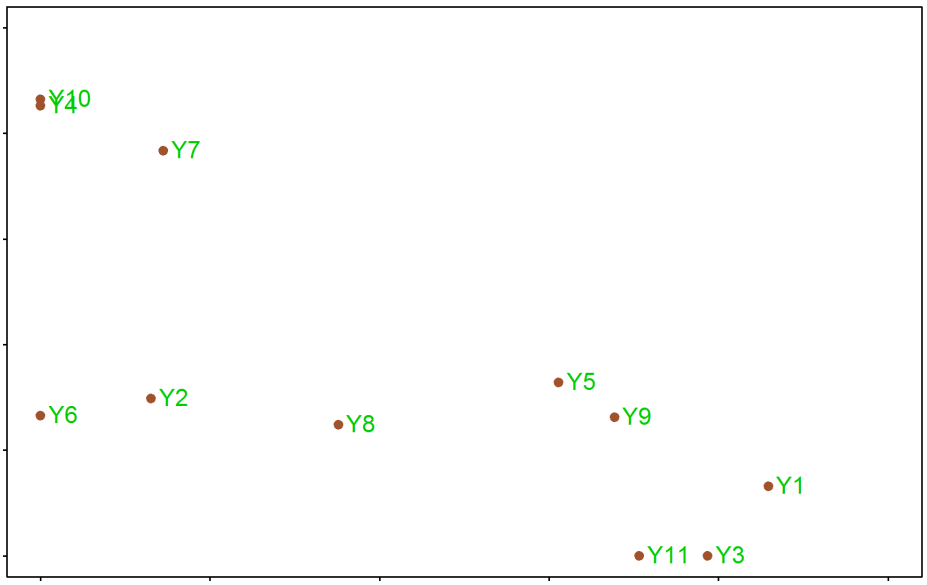
6.10. ábra. A 10 bor 11 kóstoló közös véleménye alapján

A második rajzot, a 6.11 ábra mutatja. Ezen a kóstolók egymáshoz viszonyított minősítése látható. Az ábrán az egyes kóstolókat a sorszámuk jelzi. Az ábra adatai az eredményváltozó '$W' elemében találhatóak.

Az egyes itészeket egy Y és utána a sorszámuk azonosítja, ugyanis ez a neve az adott itész szerinti elhelyezési adat y-koordinátájának a feldolgozott 'napping.don' adathalmazban. Látható, hogy az értékelés szerint az Y3 és az Y11 valamint az Y4 és az Y10 itész (a véleménye alapján) igencsak hasonlónak találtatott.

A harmadik rajz egy " korrelációs kör", ami a 6.12 ábrán látható. Ezen egyszerre szerepelnek az itészek X-es és Y-os (hely) kódjai és a nevesített bortulajdonságok.

Az elhelyezés alapja az a korreláció ami a tíz stimulus (bor) közös (kétdimenziós) konfigurációjának koordinátái és az itészek egyéni konfigurációjának x- és y-koordinátái illetve a minősítő szavak gyakoriság eloszlásai közt áll fenn.



6.11. ábra. A 11 kóstoló a véleményének hasonlósága alapján

A vektorok hossza annak erősségét tükrözi, hogy a közös konfiguráció mekkora mértékben tükrözi az adott minősítő véleményének adott (x- vagy y-) koordinátáját. A minősítő szavak alapján az látható, hogy az adott koordináta mennyire tükrözi az adott bortulajdonságot.

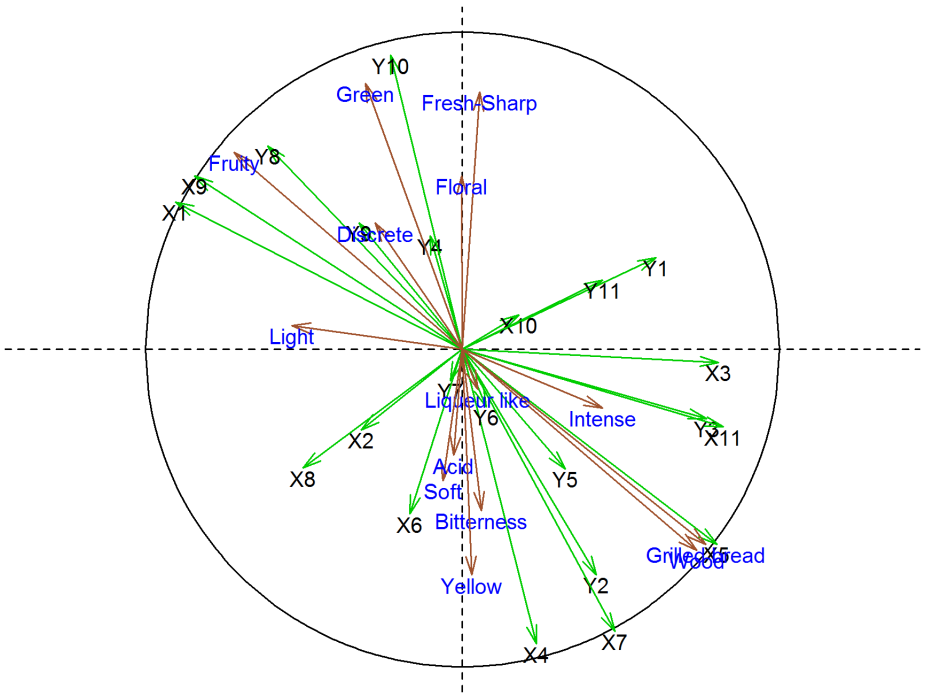
A közös konfiguráció egy másik típusu értékeléséhez futtassuk le az alábbi parancsot:

```
pmfa(napping.don,napping.words,mean.conf = M$points)
```

Eredményként most 11 ábrát kapunk. Ezek egyenként, grafikusan mutatják be a 11 itész viszonyát a közös értékeléshez. Az alábbi 6.13 ábra a 11 rajz közül az első. Az $Y1$ itész véleménye a közös véleményhez igazítva. A kék színű pontok a kék színű aláírással a közös elhelyezést mutatják. A zöld színű pontok pedig az első itész elhelyezését úgy forgatva (az elforgatást a 'piros asztal' érzékelteti), hogy az a közös elrendezéshez a lehető legközelebb legyen.

A módszer amivel az eljárás az optimális elforgatást megtalálta egy un. Prokrusztesz forgatás, ami egyszerre keres optimális forgatást és kontrakciót. A meghívott rutin egyébként a csomag 'MFA' Multiple Factor Analysis eljárását használja fel.

Végezetül futtassuk még le az alábbi parancsot:



6.12. ábra. Az 'indscal()' parancs eredményeként nyerhető korrelációs kör

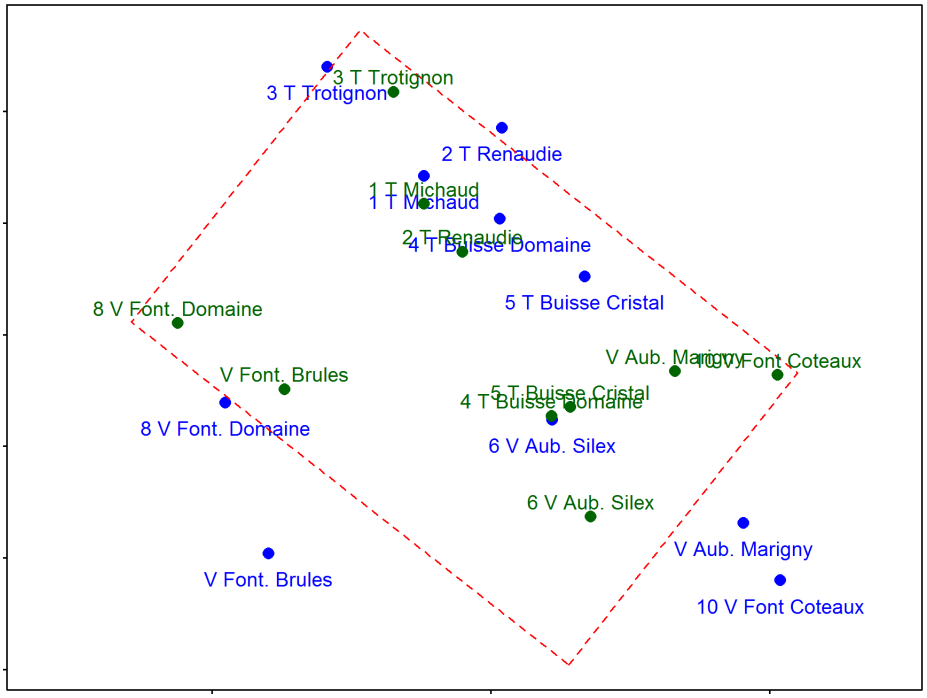
```
prefpls(cbind(M$points, napping.words))
```

A nyert 6.14 ábrán a 14 minősítő szó annak megfelelően van elhelyezve, hogy mekkora a korreláció az adott minősítő szó említési gyakoriság vektora és a közös minta-elhelyezés x illetve y-koordinátáiból képzett vektorok közt. Azaz, az ábrán azok a szavak kerültek közel egymáshoz amiket a közös elhelyezés hasonló módon reprezentál.

A bortulajdonságok nevei mögött látható színskálázott és szintvonalas kép a közös elhelyezés egyfajta sűrűségfüggvénye. Az alap-ellipszis tengelyének szögét a megadott közös elhelyezés két koordinátájának regressziós iránya határoza meg.

6.5.5. A 'smacof' csomag skálázó eljárásai

A skálázás ötödik alkalmazásaként a 'smacof' csomag [23] néhány elemét mutatjuk be. Ebben a csomagban igen érdekes példákat találhatunk a skálázás módszerének sokféle kiterjeszthetőségére, speciális esetben való alkalmazására.



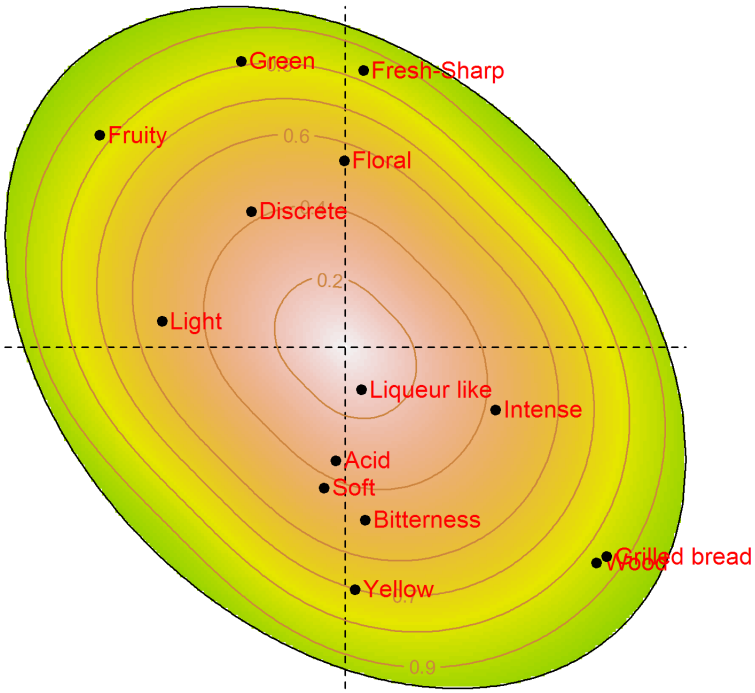
6.13. ábra. Az első itész (zöld) konfigurációja a közös (kék) konfigurációhoz illesztve

Töltsük be a csomagot a 'library(smacof)' paranccsal! Ha a betöltés sikeres, akkor egyidejűleg az esetleg szükséges 'polynom' és 'rgl' csomagok is betöltődnek.

Mintaadatként a csomag két adathalmaza fog szolgálni. A 'data(trading)' paranccsal aktiválható és a 'str(trading)' paranccsal bemutatható 20×20-as távolság adathalmaz és a 42×15 méretű 'data(breakfast);str(breakfast)' gyakoriság tábla.

A 'trading' egy 'dist' osztályu távolság mátrix 20 ország közt az 1986-os kereskedelmi kapcsolatok intenzitása alapján. A távolságok úgynevezett Jaccard távolságok, amik $[0, 1]$ -beli számok és azt mutatják, hogy egy-egy országpár esetén a kétirányú gazdasági kapcsolatok száma hányad része az összes létező (esetleg egyirányú) kapcsolatnak. Az adathalmaz érdekessége, hogy Magyarországot is tartalmazza.

A 'breakfast' 15 lehetséges reggeli komponens kívánatosságának a sorrendjét tartalmazza 42 megkérdezett esetén. Azaz a 'data.frame' mindegyik sora egy-egy permutációja az 1-15 számoknak aszerint, hogy az illető mennyire tart fontosnak egy-egy enni- vagy innivalót a reggelizés során. A 'colSums(breakfast)/42' parancs eredményeként látható, hogy a legnépszerűbb 11.7 ponttal a piritós ('toast'), és legkevésbé népszerű 4.2 átlagponttal a



6.14. ábra. Tizennégy bortulajdonság 11 itész közös véleménye szerint

'danpastry' ami valamilyen dán sütemény.

Az első, a 6.15 ábrán bemutatott rajzot az országokról a

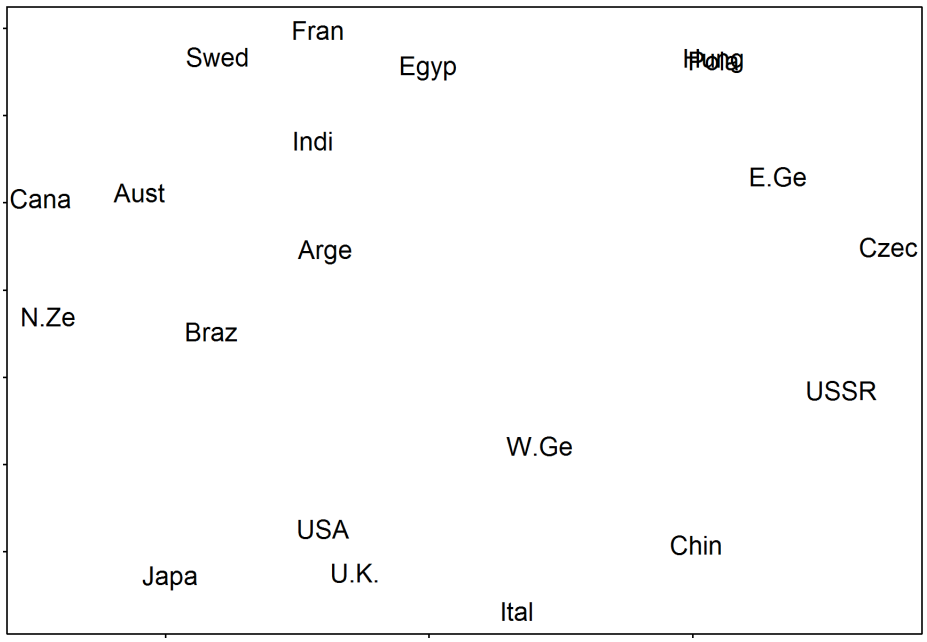
```
M <- smacofSym(trading)
plot(M, plot.type = "confplot")
```

paranccsal nyerhetjük.
A Magyarországot leiró pont az ábra jobb felső sarkában található 'Hung' felirattal. Majdnem ugyanott ahol Lengyelország, 'Pola' felirattal.

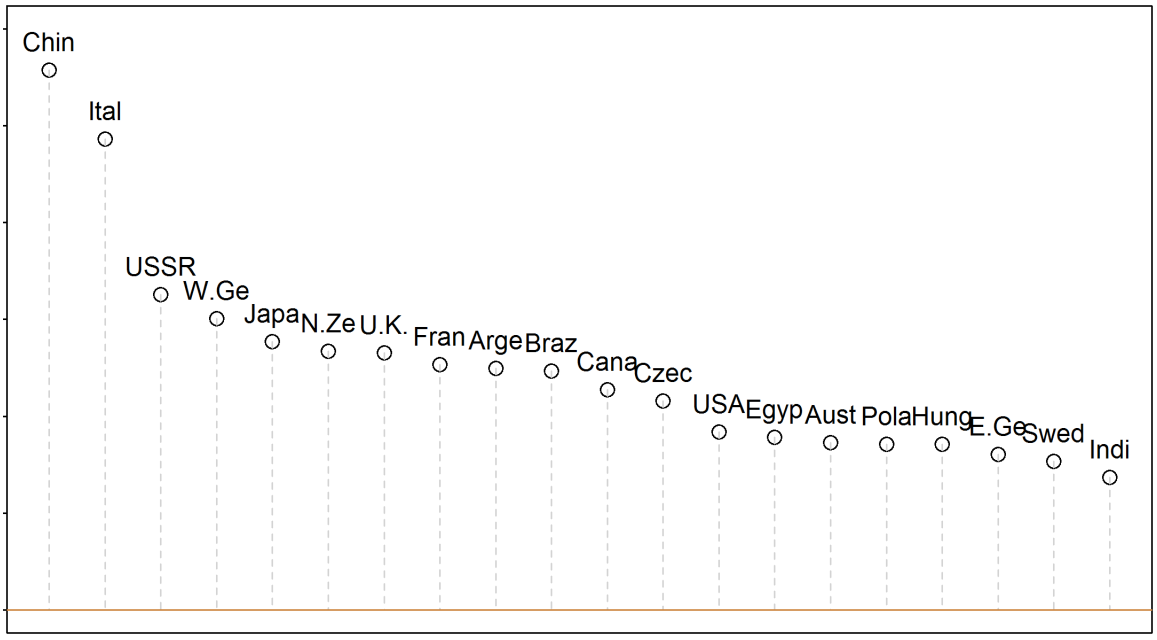
Az itt felhasznált 'smacofSym()' függvény módszere lényegében megegyezik a klasszikus skálázási módszerrel. Viszont az M-ben tárolt eredménye alapján a következő utasítással

```
plot(M, plot.type = "stressplot")
```

érdekes diagnosztikai ábrát nyerhetünk. A 6.16 ábra azt mutatja, hogy az egyes objektumok milyen mértékben járulnak hozzá a tapasztalt és a modell szerinti távolságok közti különbségek súlyozott négyzetes átlagához.



6.15. ábra. Húsz ország skálázott képe a kétoldalú kereskedelem kölcsönössége alapján



6.16. ábra. Az egyes objektumok (országok) hozzájárulása a 6.15 ábra stresszéhez

A bal oldalon szereplő Kína és Olaszország kiugró értéke azt mutatja, hogy e két ország kereskedelmi kapcsolatai járulnak hozzá legnagyobb mértékben az ábrázolás stressz (pontatlanság) értékéhez.

A 'smacof' csomag utolsó bemutatott eljárása a 'smacofRect()' módszer aminek működési módját a 'breakfast' reggeli preferencia adatokon demonstráljuk.

Futtassuk le a következő parancsokat:

```
data(breakfast)
M <- smacofRect(breakfast)
plot(M, plot.type="confplot",joint=TRUE)
```

Ez a példa több szempontból is érdekes. Egyrészt a feldolgozott adatok permutációk: a reggelizők preferencia sorrendjei. Másrészt a felhasznált 'smacofRect()' eljárás a skálázás általában *unfolding* — értelemszerűen fordítva talán 'szétpakolás' — modellnek nevezett eljárását valósítja meg.

E modell szerint a feldolgozott tábla sorai és oszlopai együttesen alkotják a skálázás objaktumait. Ám a távolságmátrix csak a sorok és az oszlopok közti azon távolságokat tartalmazza, amik a feldolgozott tábla elemei. Azaz úgy kell tekinteni, hogy nincs adat két oszlop illetve két sor távolságára vonatkozóan. Ennek megfelelően a távolságok reprezentálásakor is csak azokat a távolságokat vesszük figyelembe, amik a sorokat illetve az oszlopokat ábrázoló pontok közt mérhetők.

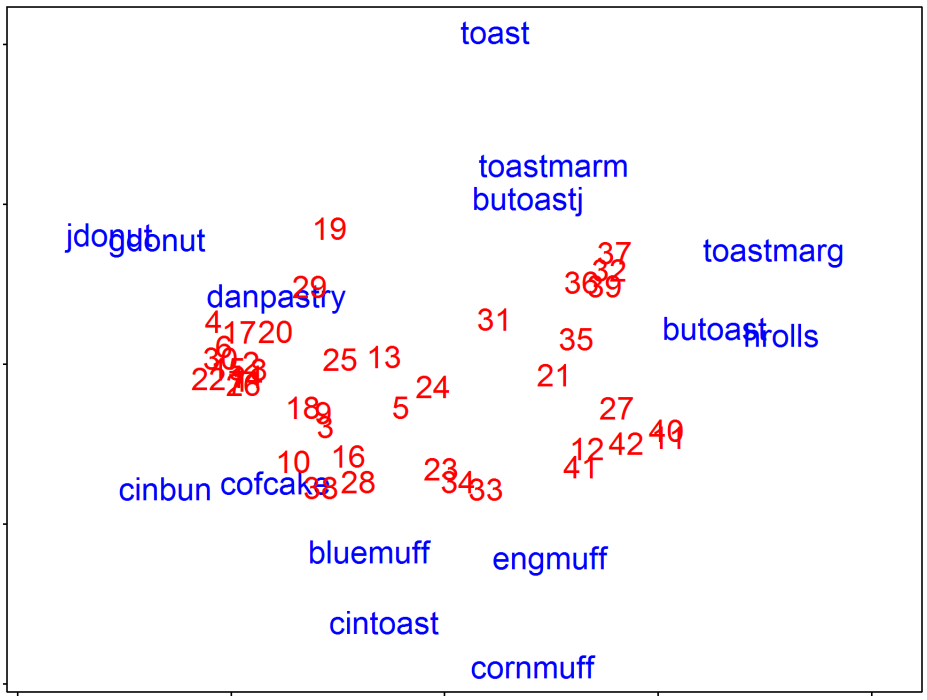
Érdekes megfigyelni, hogy a programsorok grafikus eredményeként nyert 6.17 ábrán hogyan csoportosulnak a reggelizők és a reggeli komponensek.

A 'smacof' csomag egyébként még számos további érdekes, speciális skálázó és skálázáshoz kapcsolódó eljárást tartalmaz. Úgy mint a 'smacofIndDiff()' eljárást ami az 'SensoMineR::indscal()'-hoz hasonló úgynevezett háromutas (three-way) módszer. A 'smacofConstraint()' módszert aminél az eredményekre külső kényszerfeltétel adható. Továbbá a 'smacofSphere.primal()' és a 'smacofSphere.dual()' eljárásokat, amik az objektumok mérések szerinti távolságait gömbfelszinen igyekeznek ábrázolni.

6.6. A skálázás alkalmazásai

6.6.1. Korrespondencia analízis

A korrespondencia analízis segítségével gyakoriság táblákat modellezhetünk (megjegyezzük, hogy ezeket a táblázatokat a magyar nyelvű szakirodalomban gyakran gyakorisági



6.17. ábra. A reggeli komponensek és a reggelizők 'smacofRect()' paranccsal nyert képe

táblának nevezik, de úgy érezzük, hogy a javasolt kifejezés jobban fedi a lényeget, hiszen a táblázat gyakoriságokat tartalmaz). A továbbiakban, az egyszerűség kedvéért csak a kétdimenziós táblák korrespondencia analízisével foglalkozunk. [36]

Kétdimenziós gyakoriság tábla csak olyan megfigyeléssor alapján készíthető, amelyiknek része két olyan diszkrét lehetséges értékű változó, aminek értéke minden egyes megfigyelt objektumra ismert.

Nevezetes, statisztikai mintapéldaként gyakran idézett gyakoriság tábla a következő:

	fair	red	medium	dark	black
blue	326	38	241	110	3
light	688	116	584	188	4
medium	343	84	909	412	26
dark	98	48	403	681	85

és ez a 'require(MASS);caith' parancssorral a fenti módon ki is iratható. Ez a táblázat úgy keletkezett, hogy a skóciai Caithness tartományban 5387 személy esetén feljegyezték többek közt a szem illetve a haj színét. A táblázat azt mutatja, hogy például 326 olyat

találtak, akinek a szeme színe kék és a haja szőke.

Legyen az egyik diszkrét változó lehetséges értékeinek a száma r, a másiké c. Esetünkben ez 4 illetve 5. Ekkor, ha n megfigyelés van (esetünkben 5387), akkor e két változó alapján egy olyan $r \times c$ — esetünkben 4×5 — méretű N, kétdimenziós táblázat készíthető, aminek $n_{i,j}$ eleme azt mondja meg, hogy az n megfigyelt objektum közt hány olyan volt, amire az egyik diszkrét változó az r lehetséges értéke közül az i. értéket, a másik pedig a c lehetséges közül a j. értéket vette fel.

Jelölje a gyakoriság tábla i. sorában található elemek összegét $n_{i,+}$, a j. oszlopban található elemek összegét $n_{+,j}$.

Egy-egy sor illetve oszlop profiljának azt a tapasztalati eloszlást nevezzük, amit úgy kaphatunk, hogy minden sorbeli illetve oszlopbeli számot elosztunk, a megfelelő sorbeli illetve oszlopbeli számok összegével.
Az i. sorprofilja tehát

$$(n_{i,1}/n_{i,+}, ..., n_{i,c}/n_{i,+}),$$

a j. oszlopprofilja pedig

$$(n_{1,j}/n_{+,j}, ..., n_{r,j}/n_{+,j}).$$

Ez esetünkben azt jelenti, hogy például az utolsó oszlop profilja a

$$(3/118, 4/118, 26/118, 85/118)^T$$

oszlopmátrix, az első sor profilja pedig a

$$(326/718, 38/718, 241/718, 110/718, 3/718)$$

sormátrix.

A közös sorprofil

$$\mathbf{r} = (n_{+,1}/n, ..., n_{+,c}/n),$$

a közös oszlopprofil pedig

$$\mathbf{c} = (n_{1,+}/n, ..., n_{r,+}/n).$$

Azaz a közös sorprofilt az oszlopösszegek megfigyelésszámmal osztott értékei, a közös oszlopprofilt pedig a sorösszegek megfigyelésszámmal osztott értékei alkotják.

Tehát a sorprofilok és az oszlopprofilok, valamint a közös sor illetve oszlopprofilok egyaránt tapasztalati eloszlások.

A közös profilok a két megfigyelt minősítő változó tapasztalati eloszlásai. A sor- és oszlopprofilok pedig feltételes eloszlások. Olyan feltételes eloszlások ahol a feltételt az jelenti, hogy melyik sor illetve oszlopprofiljáról van szó.

Ha érvényes volna, hogy a megfigyelt egyedeken a két diszkrét változó lehetséges értéke egymástól független módon adódik, akkor egy olyan N kétdimenziós táblát kaptunk volna, aminek a sor, illetve oszlop profiljai nagyjából azonosak és ezek a profilok nagyjából egyenlőek a közös sor illetve a közös oszlopprofillal is.

Ha ugyanis teljesül a két minősítő változó függetlensége, akkor egyik tulajdonság eloszlása sem függ attól, hogy mennyi a másik rögzitett értéke.

Az elemi statisztikában ismertetett χ^2 statisztika egyébként pont azt az eltérést méri, ami tapasztalt és a függetlenség feltételezése mellett várható tábla közt van:

$$G^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(n_{i,j} - n\frac{n_{i,+}}{n}\frac{n_{+,j}}{n})^2}{n\frac{n_{i,+}}{n}\frac{n_{+,j}}{n}}.$$

Azaz a G^2, — ami a táblázatot létrehozó két változó függetlensége esetén közel $\chi^2_{(r-1)(c-1)}$ eloszlású, — azt méri, hogy a tapasztalt N gyakoriság tábla mennyire tér el a tapasztalat és függetlenség feltételezése alapján adódó, $n\frac{n_{i,+}}{n}\frac{n_{+,j}}{n}$ elemekből felépülő, szintén $r \times c$ méretű M táblázattól.

A korrespondencia analízis eredményének értelmezéséhez vegyük észre, hogy a fenti G^2 a következő alakba is irható:

$$G^2 = n \sum_{j=1}^{c} \frac{n_{+,j}}{n} \sum_{i=1}^{r} \frac{\left(\frac{n_{i,j}}{n_{+,j}} - \frac{n_{i,+}}{n}\right)^2}{\frac{n_{i,+}}{n}},$$

ami az oszlopprofilok össztávolsága a közös oszlopprofiltól négyzetesen mérve, és

$$G^2 = n \sum_{i=1}^{r} \frac{n_{i,+}}{n} \sum_{j=1}^{c} \frac{\left(\frac{n_{i,j}}{n_{i,+}} - \frac{n_{+,j}}{n}\right)^2}{\frac{n_{+,j}}{n}},$$

ami a sorprofilok közös sorprofiltól mért távolsága.

Vagyis a G^2 közelítése egyfajta skálázása a sorok és oszlopok közti távolságoknak. Kontingencia táblák esetén a korrespondencia analízis pont ezt teszi. Azt lehet segítségével vizsgálni, hogy a gyakoriság tábla miért tér el a független táblától és hogy az egyes oszlopok, illetve sorok eloszlása mennyire hasonlít egymáshoz.

Legyen az általánosított szinguláris érték felbontása annak a differenciának ami az N mért gyakoriság tábla és a függetlenség feltételezése mellett várt (ugyanolyan méretű) M tábla közt van a következő:

$$N - M = A\Lambda B^T = \sum_{\ell=1}^{K} \lambda_k a_k b_k^T$$

ahol Λ egy diagonális mátrix A és B olyan, hogy kielégíti az $\mathbf{c}$ közös oszlopprofilra az

$$A^T diag(\mathbf{c})^{-1} A = I$$

és az $\mathbf{r}$ közös sorprofilra a

$$B^T diag(\mathbf{r})^{-1} B = I$$

feltételeket.

Az $N - M$ mátrix összeg alakú felirásában az a_k az A, b_k az B oszlopai és a λ_k a Λ diagonális elemei, azaz az $N - M$ szinguláris értékei, csökkenő sorrendben.

A fenti felbontás szerint a gyakoriság tábla sorait az A mátrix sorai, a gyakoriság tábla oszlopait pedig a B matrix sorai reprezentálják. A következő módon.

A sorok koordinátái

$$V = diag(\mathbf{c})^{-1} A\Lambda =$$
$$= diag(\mathbf{c})^{-1}(N - M)diag(\mathbf{r})^{-1} B,$$

az oszlopok koordinátái pedig

$$W = diag(\mathbf{r})^{-1} B\Lambda =$$
$$= diag(\mathbf{r})^{-1}(N - M)^T diag(\mathbf{c})^{-1} A.$$

Ebből az is következik, hogy a sor és oszlopkoordináták közt a következő összefüggés áll fenn:

$$V\Lambda = diag(\mathbf{c})^{-1} NW$$

illetve

$$W\Lambda = diag(\mathbf{r})^{-1} NV.$$

Ha a gyakoriság táblát közelíteni akarjuk, akkor ennek L_2-ben optimális módszere a fenti összegnek csak az első $k \leq K$ tagját figyelembe venni. Azaz csak a k legnagyobb λ_ℓ,

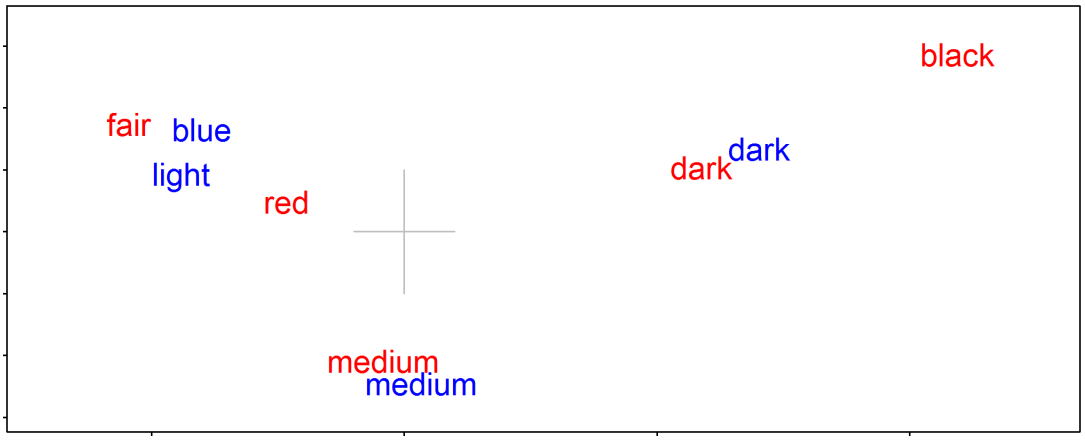
$\ell = 1, ..., k$ szinguláris értéknek megfelelő tagot összeadni.

A 'MASS' csomag 'corresp()' eljárása a korrespondancia analízis vázolt módszerének egy implementációja.

Futtassuk le az alábbi programrészletet.

```
require('MASS')
M<-corresp(caith, nf = 2)
biplot(M)
```

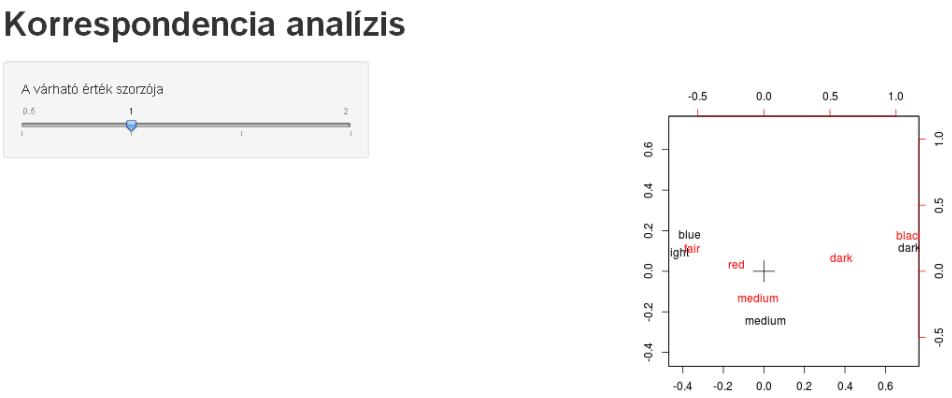
Eredményként az alábbi ábrát kapjuk. Pirossal a hajszíneknek, kékkel pedig a szemszíneknek megfelelő pontok vannak cimkézve. A cimkék azonosak a feldolgozott, 'caith' adathalmazban található sor és oszlop nevekkel.



6.18. ábra. A 'biplot(corresp())' parancs eredménye

Legyünk figyelemmel arra, hogy az ábra nem egy igazi biplot. A sorok illetve az oszlopok által meghatározott két objektum csoport egymáshoz viszonyított helyzetének közvetlen információ tartalma nincs. Értékeléskor csak külön, csak a sorok illetve csak az oszlopok egymáshoz viszonyított elhelyezkedését szabad figyelembe venni.

Ehhez az adatbázishoz interaktív animáció is készült, ami a http://hpz400.cs.elte.hu:3838/ZA_glm/ címen található. Itt be lehet állítani, hogy a fentiekben bemutatott 'caith' táblában szereplő értékek hányszorosa legyen a szimulált Poisson eloszlás várható értéke, amely a módosított gyakoriság tábla értékeit adja meg. Ha erre a szimulált adathalmazra futtatjuk le a korrespondencia analízis módszerét, akkor a 6.19 ábrát kapjuk, ahol kisebb eltérések láthatóak az eredeti adatbázisra vonatkozó 6.18 ábrához képest.



6.19. ábra. A 'caith' adatbázishoz kapcsolódó szimulációra futtatott animáció eredménye

Irodalomjegyzék

[1] ”nlstools: tools for nonlinear regression diagnostics”, F. Baty and M. L. Delignette-Muller, (2012).

[2] ”Introduction to Matrix Analysis”, R. E. Bellman, 2nd ed., McGraw-Hill, (1970).

[3] ”Multidimensional Scaling”, T.F. Cox, M.A.A. Cox, Chapman and Hall, (1994).

[4] ”Multivariate Analysis: Methods and Applications”, William R. Dillon and Matthew Goldstein, Wiley, (1984).

[5] ”An Introduction to Generalized Linear Models”, AJ. Dobson, Chapman and Hall, London, (1990).

[6] ”Peer Influences on Aspiration: A Reinterpretation”, A.D. Duncan, A.O. Haller and A. Portes, American Journal of Sociology 74:119-137, (1968).

[7] ”HSAUR: A Handbook of Statistical Analyses Using R.”, Brian S. Everitt and Torsten Hothorn, R package version 1.3-2. URL http://CRAN.R-project.org/package=HSAUR, (2013).

[8] ”Practical Regression and Anova using R”, J.J. Faraway, http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf, (2002).

[9] ”Structural equation modeling with the sem package in R”, J. Fox, , Structural Equation Modeling 13:465–486, (2006).

[10] ”An R Companion to Applied Regression”, John Fox and Sanford Weisberg, Second Edition. Thousand Oaks CA: Sage. URL: http://socserv.socsci.mcmaster.ca/jfox/Books/Companion, (2011).

[11] ”sem: Structural Equation Models. R package version 3.1-3.”, John Fox, Zhenghua Nie and Jarrett Byrnes, URL http://CRAN.R-project.org/package=sem, (2013).

[12] ”FrF2: Fractional Factorial designs with 2-level factors”, Ulrike Groemping, R package version 1.6-5. http://CRAN.R-project.org/package=FrF2, (2013).

[13] http://en.wikipedia.org/wiki/Heptathlon

[14] "Linear Latent Variable Models: The lava-package", Klaus K. Holst and Esben Budtz-Joergensen, Computational Statistics. URL http://dx.doi.org/10.1007/s00180-012-0344-y, (2012).

[15] "SensoMineR: Sensory data analysis with R", Francois Husson, Sebastien Le and Marine Cadoret, R package version 1.17. URL http://CRAN.R-project.org/package=SensoMineR, (2013).

[16] http://cran.r-project.org/web/views/ExperimentalDesign.htm

[17] http://new.censusatschool.org.nz/resource/time-series-data-sets-2012/

[18] http://rtutorialseries.blogspot.hu/2011/10/r-tutorial-series-exploratory-factor.html

[19] http://www-rohan.sdsu.edu/~babailey/stat700/lab2.html.

[20] http://www.stat.cmu.edu/~cshalizi/350/2008/lectures/14/lecture-14.pdf

[21] " A general method for analysis of covariance structures.", K. Jöreskog, Biometrika, **57**, pp 239-251, (1970).

[22] "Kísérletek tervezése és értékelése", Kemény Sándor – Deák András, Műszaki Könyvkiadó, Budapest, (2000).

[23] "Multidimensional Scaling Using Majorization: SMACOF in R", Jan de Leeuw, Patrick Mair, Journal of Statistical Software, 31(3), 1-30. URL http://www.jstatsoft.org/v31/i03/, (2009).

[24] "Some boundary conditions for a monotone analysis of symmetric matrices",J. C. Lingoes, Psychometrika, 36, 195–203, (1971).

[25] "Multivariate calibration", H. Martens, T. Nas, Wiley, Chichester, (1989).

[26] "pls: Partial Least Squares and Principal Component regression", Bjørn-Helge Mevik, Ron Wehrens and Kristian Hovde Liland, R package version 2.3-0. URL http://CRAN.R-project.org/package=pls, (2011).

[27] "A First Course in Design and Analysis of Experiments", Gary W. Oehlert, http://users.stat.umn.edu/~gary/Book.html, (2010).

[28] ”R: A language and environment for statistical computing”, R Core Team, R Foundation for Statistical Computing, Vienna, Austria, (2012). ISBN 3-900051-07-0, URL http://www.R-project.org/, (2012).

[29] ” nFactors: an R package for parallel analysis and non graphical solutions to the Cattell scree test”, G. Raiche, R package version 2.3.3, (2010).

[30] ”lavaan: An R Package for Structural Equation Modeling”, Yves Rosseel, Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/, (2012).

[31] ”Handbook of nonlinear regression models”, D.A. Ratkowsky, M. Dekker, (1990).

[32] ”psych: Procedures for Personality and Psychological Research”, W. Revelle, Northwestern University, Evanston, Illinois, USA, http://CRAN.R-project.org/package=psychVersion=1.3.2, (2013).

[33] ”Order Restricted Statistical Inference”, T. Robertson, F.T. Wright, R.L. Dykstra, Wiley, New York, (1988).

[34] ”Introduction to Statistics, Chapter 14”, Peter Tryfos, http://www.yorku.ca/ptryfos/f1400.pdf, (1997).

[35] ”Többdimenziós statisztika.”, Móri F. Tamás és Székely J. Gábor (szerk.), Műszaki Könyvkiadó, Budapest, (1986).

[36] ” Modern Applied Statistics with S”, W.N. Venables, B.D. Ripley, Fourth edition, Springer, (2002).

[37] ”Correlation and causation”, S. Wright, Journal of Agricultural Research, 20, 557-585, (1921).