Rendszerintegráció és -felügyelet laboratórium (VIMIM309)

Felügyeleti adatok vizuális elemzése

Mérési segédlet

Szerzők: Kocsis Imre és Szombath István

Utolsó módosítás: 2013. május 16.

Verzió: 2.2

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Felderítő adatanalízis

1.1 Bevezetés

A modern statisztika szokásosan megkülönbözteti az adatanalízis két, alapvetően eltérő formáját: a *feltáró* adatanalízist (*exploratory data analysis* - EDA) és a hipotézistesztelést (*hypothesis testing*), másnéven megerősítő adatanalízist (*confirmatory data analysis* - CDA).

A feltáró adatanalízis „atyjának" John Tukey amerikai matematikust szokás tekinteni; ő fogalmazta meg először a feltáró és a megerősítő adatanalízis közötti alapvető különbséget. Tukey egy elterjedten idézett hasonlatával élve: az adatelemző egyfajta detektív. A detektívmunkához hasonlóan egy adatkészlet „megértésének" folyamata

- felderítő jellegű és interaktív;
- a hipotézisgenerálás és –ellenőrzés többszörös iterációját foglalhatja magában;
- „piszkos ügy" és legfeljebb a heurisztikák, tapasztalati szabályok szintjén megismételhető.[1]

Az adatanalízis során persze nem választanunk kell a feltáró és a megerősítő technikák között; kettéválasztásuk éppen hogy azt hangsúlyozza, hogy megfelelő „adatfeltárási" technikákkal érdemes támogatnunk és előkészítenünk a feltáró elemzést.

Tukey egyébként kerülte a feltáró adatanalízis fogalmának szabatos definiálását; a megközelítés általános filozófiáját és metodológiai megközelítéseit az ezek kontextusba helyezéséhez szükséges statisztikai alapok hiányában itt nem is tárgyaljuk. A feltáró adatanalízis talán legfontosabb sarokköve , az adatok vizualizációja azonban az általános mérnöki gyakorlat szempontjából is kiemelt fontosságú. Így motivációként érdemes végigtekintenünk egy általánosan ismert példát arra, hogy a klasszikus statisztika hogyan tud a számokon keresztül „hazudni" – miért van szükség „vizuális felderítő adatanalízisre". Ehhez segítségül fogjuk hívni az ún. 'Anscombe négyest' (*Anscombe's quartet*) [2].

1.2 Anscombe négyese

Tekintsük a következő négy, két változón (x-y) értelmezett, rendre 11 elemből álló adatkészletet:

```
> anscombe
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
4   9  9  9  8  8.81 8.77  7.11  8.84
5  11 11 11  8  8.33 9.26  7.81  8.47
6  14 14 14  8  9.96 8.10  8.84  7.04
7   6  6  6  8  7.24 6.13  6.08  5.25
8   4  4  4 19  4.26 3.10  5.39 12.50
9  12 12 12  8 10.84 9.13  8.15  5.56
10  7  7  7  8  4.82 7.26  6.42  7.91
11  5  5  5  8  5.68 4.74  5.73  6.89
```

A klasszikus statisztika több eszközt is kínál a négy adatkészlet kvantitatív jellemzésére és összehasonlítására. Ezek lehetnek például:

- Az x változók várhatóértékei
- Az x változók varianciái
- Az y változók várhatóértékei
- Az y változók varianciái

[1] [1], p 36 alapján.

- Az x és y közötti korreláció az egyes esetekben
- A lineáris regresszió az egyes esetekben

Az első sor első, 'prompt' karaktere nem véletlen. A mérés során használni fogjuk az `R` nyílt forráskódú, ingyenes statisztikai környezetet. 'R' alatt legtöbbször az `R` (szkript)nyelvet és futtatókörnyezetét egyben értjük (lásd: [3]). Az R használatának legegyszerűbb formája, mikor parancssoros módban (parancssorból/terminálból), interaktív szkriptértelmezőként futtatjuk. Az Anscombe négyes az `R` környezet egyik beépített adatkészlete, melyre az `anscombe` változónévvel hivatkozhatunk; `R` parancssorba beírva ezt a változó értékének kiírását váltjuk ki.

1.2.1 Várhatóértékek/átlagok (mean)

Nézzük az x változók néhány önálló tulajdonságát:

```
> summary(anscombe[,grep('x', colnames(anscombe))])
       x1              x2              x3              x4
 Min.   : 4.0   Min.   : 4.0   Min.   : 4.0   Min.   : 8
 1st Qu.: 6.5   1st Qu.: 6.5   1st Qu.: 6.5   1st Qu.: 8
 Median : 9.0   Median : 9.0   Median : 9.0   Median : 8
 Mean   : 9.0   Mean   : 9.0   Mean   : 9.0   Mean   : 9
 3rd Qu.:11.5   3rd Qu.:11.5   3rd Qu.:11.5   3rd Qu.: 8
 Max.   :14.0   Max.   :14.0   Max.   :14.0   Max.   :19
```

Majd az y változók néhány tulajdonságát:

```
> summary(anscombe[,grep('y', colnames(anscombe))])
       y1              y2              y3              y4
 Min.   : 4.260   Min.   :3.100   Min.   : 5.39   Min.   : 5.250
 1st Qu.: 6.315   1st Qu.:6.695   1st Qu.: 6.25   1st Qu.: 6.170
 Median : 7.580   Median :8.140   Median : 7.11   Median : 7.040
 Mean   : 7.501   Mean   :7.501   Mean   : 7.50   Mean   : 7.501
 3rd Qu.: 8.570   3rd Qu.:8.950   3rd Qu.: 7.98   3rd Qu.: 8.190
 Max.   :10.840   Max.   :9.260   Max.   :12.74   Max.   :12.500
```

Láthatóan az átlagok megegyeznek; sőt, első ránézésre az első és harmadik kvartilis tekintetében sincs olyan nagy eltérés.

Emlékeztetőül: a statisztikában egy értékkészlet `n` *percentilise* az az érték, ami alá a megfigyelések `n` százaléka esik. Pl. a WHO újszülött kisfiúk fejkörméret táblázatában[2] 12 hetes kornál a 97-ik percentilis 42.4 cm értéke azt jelenti, hogy a táblázat alapját adó adatokban az ilyen idős kisfiúk 97%-ának volt 42.4 cm-nél kisebb a fejkörmérete.

Nevezetes percentilisek:

- a 25-ik – az első *kvartilis* (Q_1)
- az 50-ik – a második kvartilis (Q_2), más néven *medián*
- a 75-ik – a harmadik kvartilis (Q_3)

Még egy megjegyzést kell tennünk a várhatóérték/átlag definiálásával és kiszámításával kapcsolatban. A várhatóértéket a valószínűségszámításban jellemzően mint egy (folytonos vagy diszkrét) valószínűségi változó tulajdonsága szoktuk bevezetni.

Megfigyelések feldolgozásakor azonban nem a valószínűségi változóval (illetve annak

[2] Lásd: http://www.who.int/childgrowth/standards/second_set/hcfa_boys_0_13_percentiles.pdf

eloszlásfüggvényével), hanem a valószínűségi változó értékeiből vett (véges) *mintákon* dolgozunk. Így egy 'adatkészlet' felett számított átlag ebben a kontextusban valójában egy ún. *mintaátlag (sample mean,* lásd [5], 785. oldal):

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Nyilvánvaló, hogy a valószínűségi változóból vett mintasorozatok esetről esetre más és más mintaátlagot eredményezhetnek. Így maga a mintaátlag is felfogható egy valószínűségi változóként. Megmutatható azonban, hogy a mintaátlag, mint valószínűségi változó várhatóértéke megegyezik a mintavételezett valószínűségi változó várhatóértékével, így egy 'jó' becslője annak.

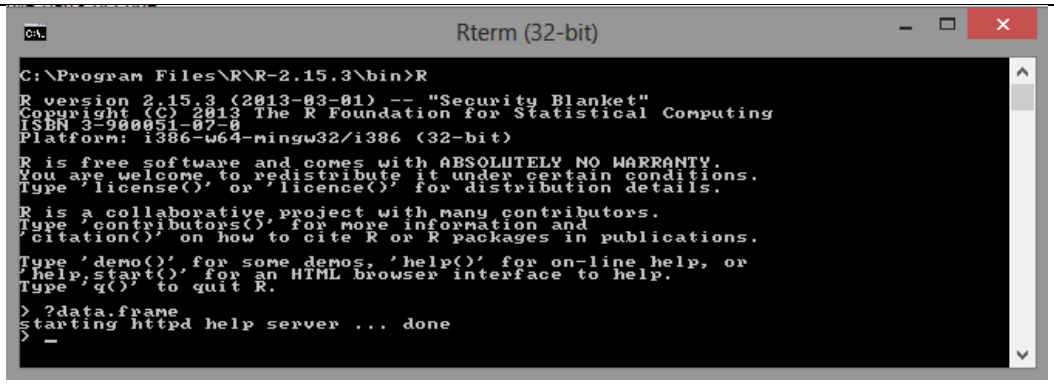
Nem foglalkozunk azzal, hogy itt mit jelent a 'jóság' fogalma és hogy a mintahalmaz méretét hogyan tervezzük; a fentiek célja arra rámutatni, hogy a mérnöki szempontból modellkomponensként viselkedő valószínűségi változó tulajdonságai és a változó-megfigyelésekből származtatott jellemzők között nem feltétlenül triviális a kapcsolat.

A kissé riasztónak ható `R` hívás valójában nagyon egyszerű. Dekonstruálásához először érdemes megismerkednünk az `R` 'adatkeret' (*data frame*) adattípusával.

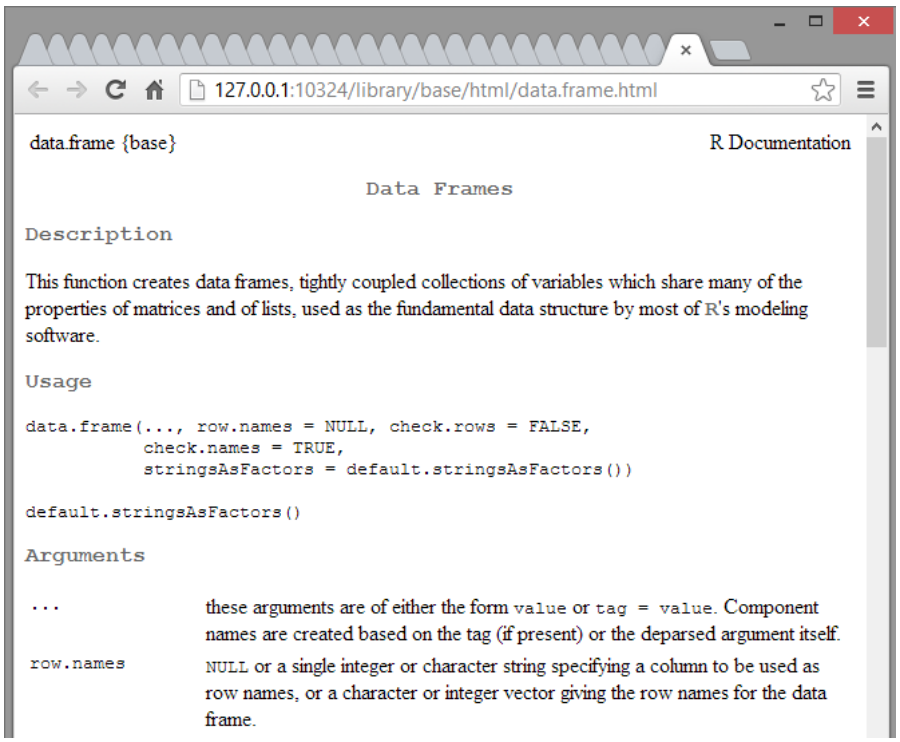
Az adatkeret alapvetően egy táblázat, nevesített oszlopokkal (ezek a 'változók') és egyedi névvel ellátott sorokkal (bár ez az egyedi név sokszor implicit módón generálódik, ahelyett, hogy megadnánk). Lényeges különbség az `R` mátrix (*matrix*) adattípusával szemben, hogy többféle típusú oszlopot is tartalmazhat, míg a mátrix minden elemének azonos típusúnak kell lennie. Néhány fontos beépített primitív típus: logikai (*logical* – `TRUE/FALSE`), egész szám (*integer*), valós szám (*real*, *numeric*, *double*), komplex szám (*complex*), sztring (*character*). Ezeket közvetlenül pl. vektorokba (*vector*), és listákba (*list*) szervezhetjük. Egy egyszerű adatkeretet pl. a következőképp hozhatunk létre:

```
> data.frame(kulcs=c('kutya', 'cica', 'meresi hiba'),ertek=c(1,NA,NaN))
        kulcs ertek
1       kutya     1
2        cica    NA
3 meresi hiba   NaN
```

A 'c' (konkatenáció) operátorral azonos hosszú vektorba szerveztünk egyrészt sztring, másrészt valós (az `NaN` miatt nem egész) értékeket, használva az 'NA' (hiányzó érték) és 'NaN' ('nem szám' – *'not a number'*) különleges értékeket. A `data.frame` „konstruktor" függvényhívás paraméterezésénél még arra is ügyeltünk, hogy az oszlopoknak megfelelő nevet adjunk. Ehhez akár leshettünk is volna; a függvényekhez (és bizonyos beépített adatokhoz) a '?' operátorral kérhetünk segítséget, ami indítási módtól függően böngészőben, terminálban vagy GUI-ban nyílik meg.



1. ábra. 'Help' előhívása parancssorból



2. ábra. A beépített 'help'.

Adatkeretünket előnyös lenne eltárolni tudnunk, hogy későbbi hívásokban hivatkozni tudjunk rá. Ehhez használhatjuk a '<–' hozzárendelés-operátort:

```
> a <- data.frame(kulcs=c('kutya', 'cica', 'meresi
hiba'),ertek=c(1,NA,NaN))
> a
        kulcs ertek
1       kutya     1
2        cica    NA
3 meresi hiba   NaN
```

Mindezekből már az is sejthető, hogy az `R` dinamikus típusrendszerű nyelv, hiszen az értéknek, és nem a változónak van típusa.

De honnan látjuk, hogy milyen (nevű) változóink vannak már? Ebben segíthet az `ls` beépített függvény:

```
> ls()
```

```
[1] "a" "x"
```

Látható, hogy itt egy 'x' nevű változót már korábban eltároltunk.

A következő felmerülő kérdés: hogyan válogathatunk le egy adatkeretből bizonyos sorokat vagy bizonyos oszlopokat? Pl. az adatkeretek szelekciós operátora segítségével, mely [<sorok logikai, sorszám- vagy név-vektora>,<oszlopok logikai, sorszám- vagy név-vektora>] alakú. A vessző előtt és után is megengedett a szelekciós kifejezés elhagyása – ez esetben nem történik szűrés. Példák:

```
> a[,'kulcs'] == 'cica'
[1] FALSE  TRUE FALSE
> b <- a[,'kulcs'] == 'cica'
> a[b,]
  kulcs ertek
2  cica    NA
> a[c(1,3),2]
[1]   1 NaN
```

Vegyük észre, hogy az '==' operátor – a Matlabhoz hasonlóan - vektor/mátrix minden elemére, intelligensen végzi el az összahsonlítást, esetünkben vektort adva vissza. Adatkeretek oszlopaira egyébként a '$' operátorral is hivatkozhatunk név szerint, az eredményként kapott vektorban pedig ismét szelektálhatunk – csak immáron egy dimenzió mentén. A következő hívás a név szerint kiválasztott oszlop egyetlen elemére alkalmazza az 'NA-e' függvényt:

```
> is.na(a$ertek[2])
[1] TRUE
```

Mostmár könnyedén vissza tudjuk fejteni az Anscombe négyesen végzett hívást. A grep függvény kiválogatja az adatkeret oszlopneveiből:

```
> colnames(anscombe)
[1] "x1" "x2" "x3" "x4" "y1" "y2" "y3" "y4"
```

az x-et, illetve y-t tartalmazókat, és ezt a vektort, mint oszlopszelekciót alkalmazza az adatkereten. A summary függvény ezeken az x változókra, illetve y változókra szűkített adatkereteken fut le.

1.2.2 Variancia (variance)

A változók (önmagukon) értelmezett varianciáit megvizsgálva ismét csak nem látható eltérés, illetve az eltérés jelentéktelen:

```
> sapply(grep('x', colnames(anscombe)), function(x){var(anscombe[,x])})
[1] 11 11 11 11
> sapply(grep('y', colnames(anscombe)), function(x){var(anscombe[,x])})
[1] 4.127269 4.127629 4.122620 4.123249
> diff(sapply(grep('y', colnames(anscombe)), function(x){var(anscombe[,x])}))
[1]  0.0003600000 -0.0050090909  0.0006290909
```

A variancia azt próbálja meg kifejezni, hogy egy valószínűségi változó (illetve abból vett minta) értékei mennyire esnek távol a várhatóértéktől, figyelembe véve minden lehetséges

értéket és azok 'súlyát' (valószínűségét) is.

Definíciója: ha *X* egy valószínűségi változó, akkor

$$\sigma_X^2 = var(X) = E[(X - \mu_X)^2],$$

ahol $\mu_X = E(X)$. ([4] alapján.)

A valószínűségi változó realizációinak megfigyeléseiből kiindulva alapvetően ismét csak egy *becslőt* tudunk adni a valószínűségi változó varianciájára, mint tulajdonságra nézve. A következő mintavariancia, mint becslő *„nem elfogult"* (*unbiased*) abban az értelemben, hogy várhatóértéke a változó varianciája:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Az alkalmazott `R` hívások ismét magyarázatra szorulnak. A `grep` függvényhívás egy vektort ad vissza – ezt már láttuk. Az `sapply` a funkcionális programozásban ismert általános 'apply' (függvény alkalmazása lista elemein, eredmények listaként visszaadása) minta egyik manifesztációja az `R`-ben (az `lapply`-al vagy `apply`-al szemben nem listaként, hanem alkalmasan egyszerűsítve – itt vektorként – kapjuk vissza az eredményeket).

Érdekesebb talán, hogy az 'alkalmazandó függvényt' helyben deklaráltuk a '`function`' kulcsszó segítségével, kihasználva azt, hogy a függvény az őt hívó környezet(ek)ben tud keresni változót egy adott névhez, így az `anscombe`-ot is megtalálja a globális kontextusban. Függvényt egyébként változónévhez is rendelhetünk (ne feledjük – az utolsó hívás értéke lesz mindig a visszatérési érték). A következő példa pontosvesszőt használ a „sorok" elválasztására; tárolt parancsállományban persze nyugodtan használhatunk több sort pontosvessző-lezárás nélkül.

```
> myfun <- function(x,y){a <- x * x + y; -a}
> ls()
[1] "a"     "b"     "myfun" "x"
> myfun(4,2)
[1] -18
```

A `diff` az egyszerű differencia-függvény.

1.2.3 x és y korrelációja

Eddig a négy adatkészlet x és y változóinak önmagában vett tulajdonságainak összehasonlításával foglalkoztunk – a vizsgált klasszikus statisztikai leíró mennyiségek mentén az adatkészletek x-ei, illetve y-jai érdemben nem különböztek. Vizsgáljuk most azt, hogy x és y *kapcsolata az adatkészleteken belül* különbözik-e a négy adatkészlet között! Ehhez a (Pearson-féle) lineáris korrelációs együtthatót hívjuk segítségül:

```
> for(i in 1:4){names <- paste(c('x', 'y'), i, sep="");
print(cor(anscombe[,names]))}
          x1        y1
x1 1.0000000 0.8164205
y1 0.8164205 1.0000000
          x2        y2
x2 1.0000000 0.8162365
y2 0.8162365 1.0000000
          x3        y3
x3 1.0000000 0.8162867
y3 0.8162867 1.0000000
          x4        y4
```

```
x4 1.0000000 0.8165214
y4 0.8165214 1.0000000
```

Láthatóan az x-ek és y-ok (lineáris) korrelációs együtthatója sem különbözik érdemben, azaz a korrelációs együttható szempontjából mind a négy adatkészletben azonos kapcsolatban van x és y.

Néhány statisztikai alapfogalmat kénytelenek vagyunk ismét bevezetni ([4] alapján).

Legyen $\boldsymbol{X}$ egy folytonos, valós értékű (bár tárgyalásunk szempontjából ez lényegtelen) valószínűségi változókból, mint komponensekből felépülő vektor valószínűségi változó. Ennek várható értéke:

$$\boldsymbol{\mu}_X = E(\boldsymbol{X}) = \big(E(X_1), E(X_2), .., E(X_r)\big)^T$$

Ekkor az $\boldsymbol{X}$-en értelmezett $r \times r$ *kovariancia-mátrix*:

$$\boldsymbol{\Sigma}_{XX} = cov(\boldsymbol{X}, \boldsymbol{X}) = E\{(\boldsymbol{X} - \boldsymbol{\mu}_X)(\boldsymbol{X} - \boldsymbol{\mu}_X)^T\}$$

Az így adódó mátrix főátlójában a vektor komponenseit adó valószínűségi változók varianciái állnak; többi cellájában pedig a komponens változó párokon értelmezett *kovariancia*:

$$\sigma_{XY} = cov(X, Y) = E\{(X - \mu_X)(Y - \mu_Y)\}$$

Ekkor $\boldsymbol{X}$ *korrelációs mátrixa* úgy áll elő $\boldsymbol{\Sigma}_{XX}$ –ből, hogy az i-edik sort σ_i-vel, a j-edik oszlopot pedig σ_j-vel osztjuk:

$$\boldsymbol{P}_{XX} = \begin{pmatrix} 1 & \rho_{12} & \cdots & \rho_{1r} \\ \rho_{21} & 1 & \cdots & \rho_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{r1} & \rho_{r2} & \cdots & 1 \end{pmatrix},$$

ahol

$$\rho_{ij} = \rho_{ji} = \begin{cases} \sigma_{ij} & ha\ i \neq j \\ 1 & egyébként \end{cases}$$

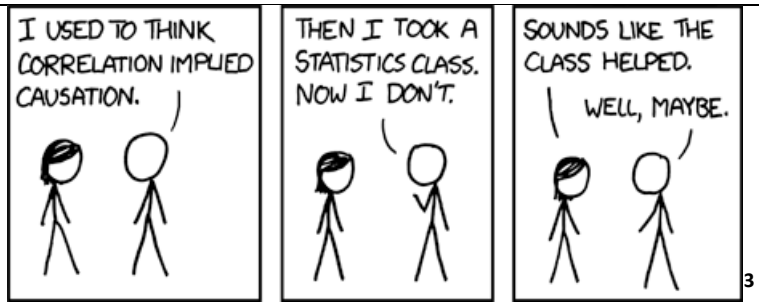
A ρ_{ij}*korrelációs együttható* (*correlation coefficient*) -1 és +1 közötti értékeket vehet fel; két változó *asszociáltságának fokát* (ha úgy tetszik, statisztikai értelemben vett függőségük mértékét) méri. Ha értéke 0, akkor azt mondjuk, hogy a két változó *korrelálatlan*; ha pozitív, a két változó *pozitívan korrelált*; ha negatív, akkor *negatívan korreláltak*. A -1 és +1 tökéletes lineáris függőséget indikálnak. Ha a két változó független, akkor a korrelációs koefficiens 0, de ennek ellenkezője feltétlenül nem igaz.

Valószínűségi változó párok korrelációs együtthatóját ismét csak (az együttes eloszlásukból vett) mintákból becsüljük; szokásosan a *minta korrelációs együttható* (sample correlation coefficient) következő képletével:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

A (minta) korrelációs koefficiens (négyzetének) szemléletes interpretációja a „variabilitás magyarázata”. „Az r^2 érték megmondja, hogy y variabilitásának hányad részét magyarázza az x és y közötti lineáris kapcsolat.” ([6], 23. oldal.)

A (minta) korrelációs együtthatóval kapcsolatban figyelnünk kell arra, hogy a két változó között *nemlineáris* kapcsolat is lehetséges. Végül, de nem utolsó sorban pedig: a korrelációból nem következik kauzalitás, azaz oksági kapcsolat. (*Correlation does not imply causation.*)



[3]

Az alkalmazott `R` hívások több alapvető megoldást bemutatnak. Először is: R-ben a funkcionális, '`apply`' (vagy éppen pl. '`Reduce`'[4]) stílusú megoldások mellett klasszikus `for` ciklusokat[5] is alkalmazhatunk. A '`:`' operátor egy sorozatot generál a két érték között. A `paste` az argumentumában megadott sztringeket ragasztja össze egyetlen sztringgé, a `sep` paraméterrel választva el azokat. Viszont ha pl. valamely argumentum vektor és a többi skalár, az eredmény egy vektor, melynek elemei az elemenkénti végrehajtások eredményei. Így kapjuk a `c('x1', 'y1')`, `c('x2', 'y2')`, … vektorokat.

Az `anscombe` adatkeretet így az egyes ciklusokban rendre az `anscombe[,c('x1','y1')]`, `anscombe[,c('x2','y2')]`, … részekre korlátozzuk le és ezekkel hívjuk meg a `cor` függvényt, mely alapértelmezett esetben (és jelen paraméterezéssel) a bemenet Pearson-féle lineáris korrelációs mátrixát határozza meg. (A függvény képes a más jellegű asszociációk mértékét mérő Spearman és Kendall együtthatók számítására is – ezekkel most nem foglalkozunk.)

Végül a `print` függvény a korrelációszámítás eredményének kiírásához szükséges; a ciklusba ágyazással immár nincs közvetlenül a kimenetre kiírandó eredmény (pontosabban a `for`, `while` és `repeat` 'láthatatlan' `NULL`[6] értékkel térnek vissza).

Tárolt parancsállományban a hívott kód természetesen olvashatóbban is mutathat. Valójában terminálban is írhatunk olvasható kódot, csak ügyelni kell arra, hogy az `R` „tudja, hogy még nem végeztünk" – ezt jelezhetjük pl. nem befejezett zárójelezéssel vagy olyan operátorral egy sor végén, melynek jobb oldala is kell, hogy legyen. (A szerző ezen trükkök alkalmazásával kapcsolatban azonban szeretné különös óvatosságra inteni az olvasót – ami nem fér el két-három folyó sorban, annak valószínűleg parancsállományban a helye.)

```
> for(i in 1:4)
+ {
+ names <- paste(c('x', 'y'), i, sep="")
+ print(cor(anscombe[,names]))
+ }
           x1        y1
x1 1.0000000 0.8164205
y1 0.8164205 1.0000000
           x2        y2
x2 1.0000000 0.8162365
y2 0.8162365 1.0000000
```

[3] http://xkcd.com/552/

[4] Lásd: `?Reduce`

[5] és egyéb megszokott vezérlési szerkezeteket; lásd `?'while'` (az idézőjelek a foglalt szó tulajdonság miatt kellenek; ugyanígy hívható elő a 'help' különleges operátorokra is, pl. `?'$'`)

[6] Más nyelvekhez hasonlóan különleges 'üres érték' objektum. Szintén más nyelvekhez hasonlóan azonban pl. a nulla hosszú vektorok és listák nem egyenlők vele! Ez praktikusan azt jelenti például, hogy az `is.null(data.frame())` kifejezés `FALSE`-ra értékelődik ki. A `data.frame`-hez hasonlóan más adattípusokhoz (és vektoraikhoz) is léteznek 'konstruktor' hívások; lásd pl. `?integer`.

```
           x3        y3
x3 1.0000000 0.8162867
y3 0.8162867 1.0000000
           x4        y4
x4 1.0000000 0.8165214
y4 0.8165214 1.0000000
```

1.2.4 Lineáris regressziók

Most azt kérdezzük, hogy az y-ok, mint függő változók x-el való „magyarázására" felállítható egyszerű lineáris regressziós modellek mennyiben különböznek.

```
> mods <- list()
> mods[[1]] <- lm(data=anscombe, formula = y1 ~ x1)
> mods[[2]] <- lm(data=anscombe, formula = y2 ~ x2)
> mods[[3]] <- lm(data=anscombe, formula = y3 ~ x3)
> mods[[4]] <- lm(data=anscombe, formula = y4 ~ x4)
> sapply(mods, coef)
                [,1]     [,2]      [,3]      [,4]
(Intercept) 3.0000909 3.000909 3.0024545 3.0017273
x1          0.5000909 0.500000 0.4997273 0.4999091
```

Láthatóan a lineáris modellek praktikusan megegyeznek.

Az egyszerű lineáris regresszió esetén egy ún. *prediktor* (vagy *független*, *magyarázó*) változó és egy *függő* (vagy *válasz*, *magyarázott*) változó közötti kapcsolatot „legjobban" leíró lineáris modellt – egyszerűen szólva „egyenest" – keressük. Az egyenest *regressziós egyenesnek* (*linear regression line*), a megfelelő egyenletet *regressziós egyenletnek* nevezzük.(Az alapvető fogalmakat [6] alapján vezetjük be.)

A regressziós egyenlet formája:

$$\hat{y} = b_0 + b_1 x,$$

ahol b_0 az y-tenelymetszet (*intercept*), b_1 pedig a meredekség (*slope*). Az egyszerű lineáris regresszió esetén azt az egyenest keressük, melyre az ún. *reziduumok* (a megfigyelt és becsült értékek közötti „maradványértékek", *residuals*) négyzetösszege:

$$F(b_0, b_1) = \sum (y - b_1 x - b_0)^2$$

minimális. A két paraméter optimális értéke levezethető megfigyelés-sokaságok felett értelmezett zárt képletként (a két paraméter mentén vett parciális deriváltakat kifejezve).

A lineáris regresszióval ennél mélyebben itt nem foglalkozunk, de felhívjuk az olvasó figyelmét arra, hogy az illesztés minőségének vizsgálatára léteznek (részben vizuális) módszerek, mint pl. a reziduális diagram, mely az x értékekhez a reziduumokat rendeli.

Emellett nem minden esetben lineáris jellegű két változó közötti kapcsolat; *nemlineáris* regresszió során más matematikai modellcsaládból (pl. kvadratikus vagy exponenciális modellek) származó modellt keresünk optimális becslő felállításához.

Az R kódban most először találkozunk *listával*. Az R listái generikus listák; névvel ellátható elemeik tetszőleges és tetszőlegesen kevert adatszerkezetek lehetnek – ide értve a további listákat is. (Lehetővé téve a funkcionális programozásban nem ritka 'listák listájának a listája' jellegű megoldásokat.) Figyeljük meg, hogy egy lista egy adott sorszámú (vagy éppenséggel valamilyen nevű) elemére újfajta, 'dupla zárójeles' indexelő operátorral hivatkozunk. Ily módon értéket rendelni egyetlen listaelemhez tudunk, illetve egyet tudunk kiválasztani (akár sorszám, akár sztring-indexet használunk); többszörös kiválasztást a

szokott `'['` operátorral érhetünk el, de ekkor a végeredmény mindig lista lesz, még ha egyetlen elemet választunk is ki.

Az `lm` függvény 'help'-je a 'Usage' szekció alatt a következő szignatúra-prefixet adja meg:

```
lm(formula, data, subset, weights, na.action, …
```

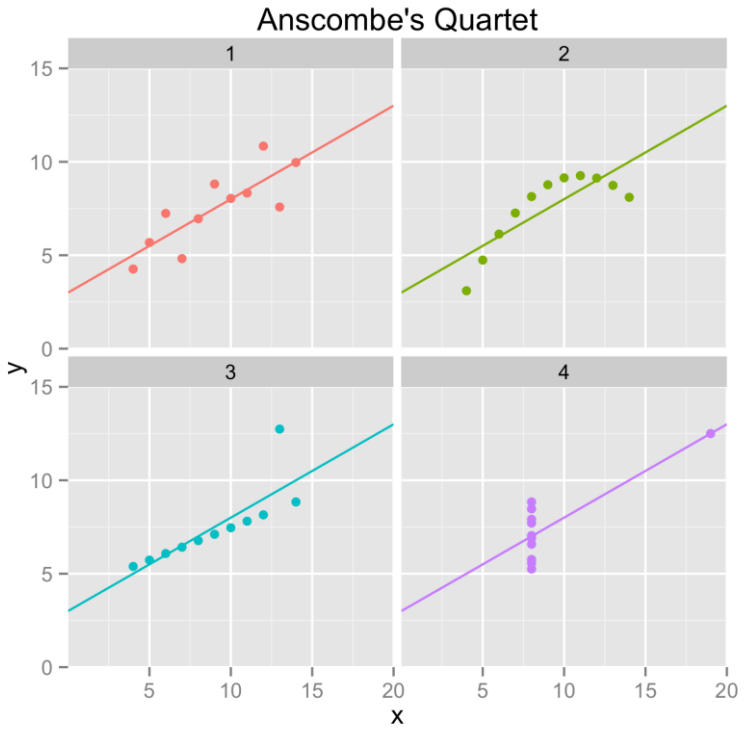
Az `R` függvények paraméterei névvel rendelkezhetnek (erre persze rögtön ellenpélda a `paste`, ahol a nevesített paraméterek előtt változó számú nem nevesített áll). Amennyiben a függvényhívás során nem nevesítjük a paramétereinket, úgy az R feltételezi, hogy a 'help'-ben is látató, alapértelmezett sorrendet kívánjuk követni. A paraméterek azonban felcserélhetők; ehhez úgy kell eljárnunk, mint a fenti `lm` hívásokban – nevesítenünk kell a paramétereket.

A lineáris regressziós modellt az `lm` függvényhívás állította fel nekünk. Ehhez nem csak a bemenő adatokat kellett megadnunk, de azt is, hogy abban mely változó(k) prediktorok és mi a függő változó. Ehhez egy ún *formula* kifejezést állítottunk össze: a mi esetünkben ez mindössze annyit állít, hogy pl. y1-et x1-el próbáljuk becsülni. Az `R`-ben azonban többváltozós és nemlineáris feladatokhoz is meg tudjuk a szükséges modellformulákat fogalmazni; ezzel kapcsolatban pl. a `formula` függvény help-je adhat további kiindulópontot.

Az `lm` függvény egy különleges, `lm` osztályú objektumot ad vissza. Ebből a `coef` függvény képes kinyerni a modell-együtthatókat.

1.2.5 Vizualizáció

Az adatkészleteket minden vizsgált statisztika azonosnak mutatta; egy egyszerű ábra azonban rögtön képes rávilágítani arra, hogy mennyire különböznek.



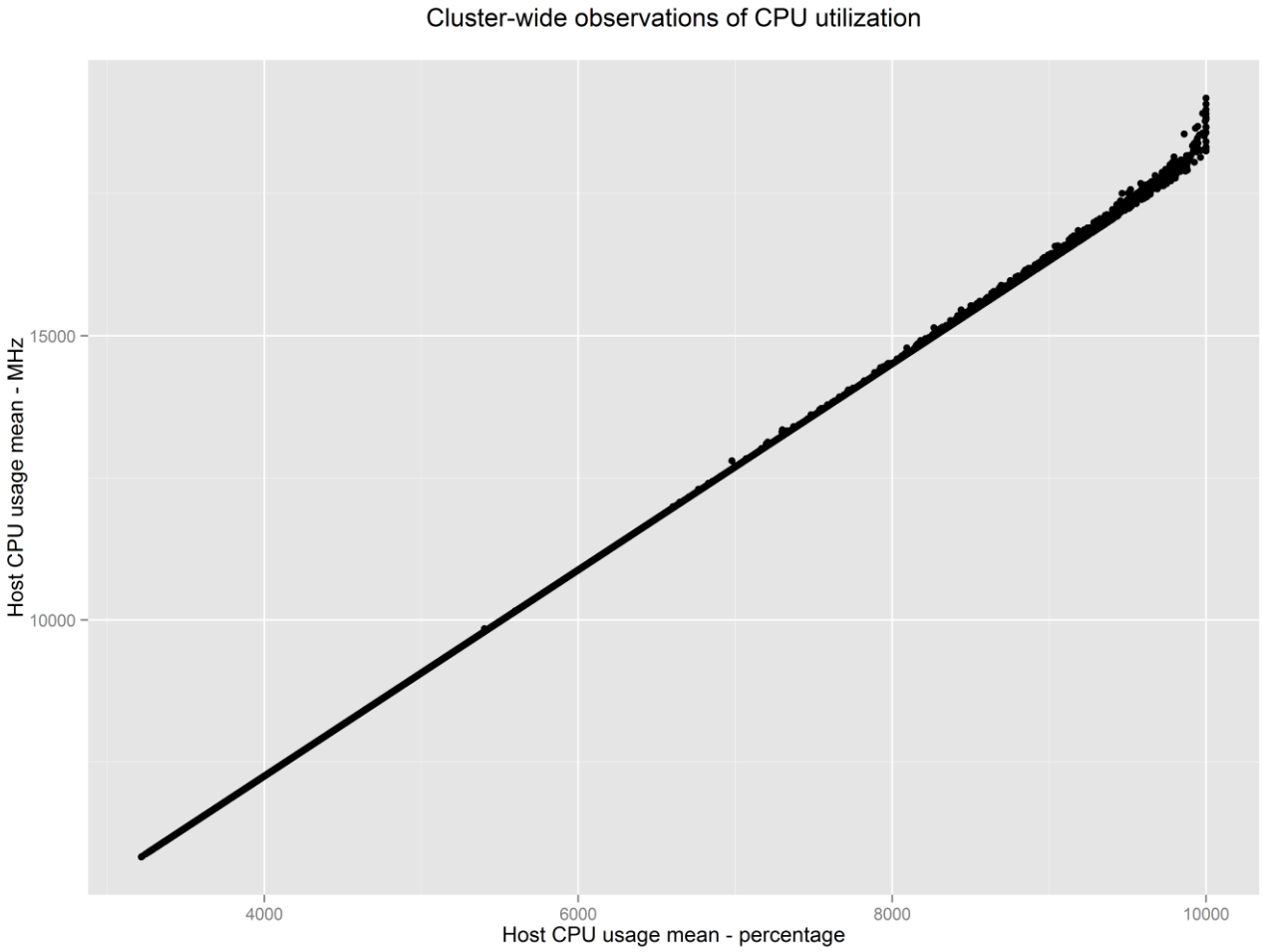
3. ábra. Az Anscombe-négyes vizualizációja a regressziókkal.

1.3 Mérnöki motiváció

Az Anscombe-négyes egy meggyőző erejű példa a statisztikai vizsgálatokat megelőző, vizualizációkat erősen használó 'adatmegértés', adatfeltárás szükségességére. Adatai azonban szintetikusak; így röviden bemutatunk egy további motiváló példát, immár az infrastruktúra-menedzsment területéről.

A 4. ábra egy nagyméretű, virtualizált klaszter hosszú távú monitorozásának két metrikáját mutatja: a klaszterre nézve átlagolt CPU-kihasználtság összetartozó megfigyelt értékeit százalékban és MHz-ben kifejezve. 'Vizuális feltáró adatanalízis' nélkül könnyen figyelmen kívül hagyhatnánk e két mérőszám kapcsolatát – korrelációjuk 0.99998, ami majdnem tökéletes lineáris kapcsolatot indikál. Bár amennyiben a fizikai konfiguráció statikus a korrelációnak 1-nek kellene lennie, a hibát betudhatnánk enyhe mérési vagy kerekítési hibának.

Az ábra azonban egészen mást mond a gyakorló mérnök számára. A lineáris kapcsolattól eltérés nem uniform, hanem az extrém terhelési tartományra korlátozódik; valószínű, hogy ebben a tartományban a hipervizorok annyira túlterheltek, hogy teljesítmény-monitorozási adataikat egészen egyszerűen nem hihetjük el és nem használhatjuk közvetlenül modellépítésre.



4. ábra.Statikus konfigurációjú klaszter átlagolt CPU-kihasználtsága százalékban és MHz-ben kifejezve.

2 A vizsgálatok tárgya: konferencia-kiszolgáló virtualizált környezetben

A mérés során egy virtualizáltan futó videokonferencia-szolgáltatás feletti méréseket fogunk vizsgálni két esetre nézve:

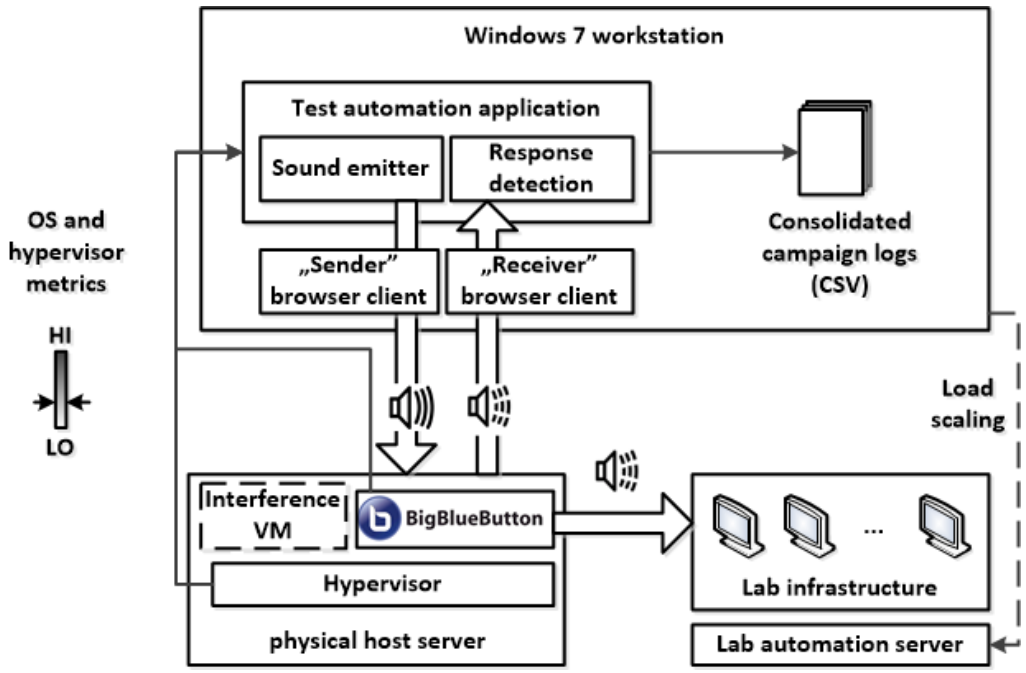
a) a kiszolgáló egy általunk menedzselt ESX infrastruktúrán fut;

b) a kiszolgáló az Amazon EC2-n fut egy bérelt virtuális gépben.

Mindkét esetben mérjük a következőket:

- Hangátvitel késleltetése
- Szabványos Linux platformszintű metrikák

Emellett az a) esetben gyűjtjük még az ESX hipervizor a hoszt erőforrásairól és a virtuális gép erőforráshasználatáról jelentett megfigyeléseit. A mérési elrendezést az a) esetben az 5. ábra szemlélteti.



5. ábra. Mérési elrendezés.

2.1 ESX mérések

Az ESX méréseket tartalmazó bináris R adatállomány a következő oszlopokkal rendelkezik:

Oszlopnév	Leírás	Objektum	Mértékegység
timestamp	Unix epoch időbélyeg	NA	sec
Dloadavg_15m	15 perces Unix load	VM OS	NA
Dloadavg_5m	5 perces Unix load	VM OS	NA
Dloadavg_1m	1 perces Unix load	VM OS	NA
Dmemoryusage_cach	(Kernel) gyorsítótár	VM OS	byte
Dmemoryusage_free	Szabad memória	VM OS	byte
Dmemoryusage_used	Használt memória	VM OS	byte
Dnettotal_recv	Hálózaton fogadott adatok	VM OS	byte/sec

Dnettotal_send	Hálózaton küldött adatok	VM OS	byte/sec
Dprocs_new	Új folyamatok száma	VM OS	1/s
Dprocs_run	Futó folyamatok száma	VM OS	1/s
Dsystem_csw	Kontextusváltások száma	VM OS	1/s
Dsystem_int	Megszakítások száma	VM OS	1/s
Dtotalcpuusage_hiq	CPU: hardver-megszak. kezelésével töltött idő	VM OS	%
Dtotalcpuusage_idl	CPU: 'idle'	VM OS	%
Dtotalcpuusage_siq	CPU: szoftver-megszak. kezelésével töltött idő	VM OS	%
Dtotalcpuusage_sys	CPU: kernel módban futó folyamatok	VM OS	%
Dtotalcpuusage_usr	CPU: normál folyamatok felh. módban	VM OS	%
Dtotalcpuusage_wai	I/O-ra várakozás	VM OS	%
Hcpu_usage	CPU kihaszn.	ESX hoszt	% * 100
Hcpu_usageinmhz	CPU kihaszn.	ESX hoszt	% * 100
Hdisk_highestlatency	Legnagyobb SCSI parancs késleltetés (kernel + eszköz) a mintavételi intervallumban	ESX hoszt	millisec
Hdisk_readrate	Átlagos diszk olvasási ráta	ESX hoszt	kbytes/sec
Hdisk_usage	Aggregált diszk I/O ráta	ESX hoszt	kbytes/sec
Hdisk_writerate	Átlagos diszk írási ráta	ESX hoszt	kbytes/sec
Hmem_active	Aktívan használt memória („mostanában érintett" tulajdonság alapján becsülve)	ESX hoszt	kbyte
Hmem_activewrite	Aktívan írt memória	ESX hoszt	kbyte
Hmem_consumed	Összes használt memória	ESX hoszt	kbyte
Hmem_granted	A teljes VM-eknek „kiajánlott" memória (inkl. osztott) + a vSphere szolgáltatások	ESX hoszt	kbyte
Hmem_overhead	A virtuális gépek és vSphere szolg. futtatásának overhead-je	ESX hoszt	kbyte
Hmem_shared	Az ESX alkalmaz memória-deduplikációt. Ez a metrika a VM-ek „osztott" memóriamennyisége (+vSphere) szummázva.	ESX hoszt	kbyte
Hmem_sharedcommon	Osztott memóriaként funkcionáló fizikai memória. Megtakarítás: shared – sharedcommon	ESX hoszt	kbyte
Hmem_unreserved	Használatba nem vett memória	ESX hoszt	kbyte

Hmem_usage	Használatba vett memória aránya a teljeshez	ESX hoszt	% * 100
Hmem_usedbyvmkernel	A VM kernel (szigorúan vett hipervizor) által használt memória	ESX hoszt	kbyte
Hmem_zero	'0' tartalmú VM-memóriák összege ('shared' része).	ESX hoszt	kbyte
Hnet_packetstxted_vmnic1	A vmnic1 (virtuális) hálózati kártyán a mérési intervallumban forgalmazott csomagok száma	ESX hoszt	#
Qavgdelay	hangátviteli kísérletek átlagos end-to-end késleltetése (1 átviteli kísérlet: 5 „tikk" hang gyors sorozata)	szolgáltatás	millisec
Qdelaystddev	az egyes átviteli kísérleteken belüli késleltetés-szórás	szolgáltatás	millisec
Qtxfault	Átviteli kísérlet sikeres volt, ha mind az 5 „tick" felismerhetően visszaérkezett	szolgáltatás	boolean

Megjegyzések: jónéhány, az ESX által jelentett metrikát kihagytunk az adatkészletből; pl. a virtuális gép szintű metrikákat teljesen (részben feleslegességük miatt, részben pedig mérési feladatokat előkészítendő). Megjegyzendő még, hogy az egyes aspektusok mintavételezése különböző frekvenciával történt.

- Linux metrikák: másodpercenként mérés.
- ESX metrikák: az ESX 20 másodpercenként frissíti a vonatkozó számlálókat. Ennél gyakrabban is lekérdezhetjük őket, de értékük ettől nem fog gyakrabban változni.
- Szolgáltatás: az átviteli kísérletek vagy az előző átvitel sikeres befejeződése („visszajött a beadott tikk-sorozat"), vagy egy timeout után indulnak.

Az adatkészletben az eltérő mintavételezési frekvenciákat kezelhettük volna úgy, hogy egy adott időpillanatban nem mért metrikák értékét `NA`-val töltjük ki, vagy interpolálhattunk is volna a megelőző és rákövetkező mérésekből. Az egyszerűség kedvéért azonban itt azt az utat követtük, hogy a sikeres/sikertelen szolgáltatás-ellenőrzés regisztrátumaira csatoltuk rá (a biztosan rendelkezésre álló) platformszintű megfigyeléseket – ezzel persze elvesztve a Linux metrikák egy jelentős részét. `NA` értékek még is vannak az adatkeretben – ott, ahol sikertelen hangátvitel miatt nem értelmezhető az átviteli késleltetés értéke:

```
> rm(list=ls())
> load(file='C:/Users/ikocsis/Desktop/BBB ESX v2.RData')
> ls()
[1] "esx"
> esx[rowSums(is.na(esx)) > 0,c('timestamp', 'Qavgdelay', 'Qtxfault')]
    timestamp Qavgdelay Qtxfault
2  1332387587        NA     true
9  1332387615        NA     true
21 1332387657        NA     true
22 1332387668        NA     true
23 1332387679        NA     true
24 1332387690        NA     true
49 1332387807        NA     true
```

```
54 1332387830        NA     true
```

Az `R` kód a memória-munkaterület (`R` terminológiában: *workspace*) teljes törlésével kezdődik (az `rm` függvény a már használt `ls` eredményét kapja meg bemenetként).

Ezek után egy bináris `R` adatállományt töltünk be a load függvénnyel (mentésre a `save` függvényt használnánk). Az állományban egyetlen, `esx` nevű adatkeret van.

A sorok kiválasztásánál a teljes adatkeretre előállítjuk a cellákon értelmezett '`NA`-e' logikai mátrixot, majd képezzük ennek sorösszegét – a `TRUE/FALSE` értékek összeadásnál mint 1 és 0 viselkednek. Az összehasonlítással pedig megkapjuk azt a logikai vektort, ami pontosan akkor `TRUE`, ha az adott sorban van `NA` érték.

2.2 Amazon mérések

Az Amazonon végzett mérések során `M1.SMALL` típusú virtuális gépet használtunk[7]. Itt hipervizor szintű metrikákhoz nem is férhetünk hozzá; a Linux virtuális gépen azonban végezhetünk méréseket. A labor során rendelkezésre álló oszlopok:

```
> load(file='C:/Users/ikocsis/Desktop/BBB_amazon_v2.RData')
> colnames(amazon)
 [1] "timestamp" "user" "system" "idle" "iowait"
 [6] "irq" "softirq" "steal" "shoppermean" "shoppercompletions"
[11] "shoppermaxima" "computeddelay"
```

Az adatkeret több szempontból is különbözik az ESX adatoktól.

1. Didaktikai okokból a Linux platformszintű metrikák jóval szűkebb készletét tartalmazza.
2. Szintén didaktikai okokból a minták egységesen másodpercenként követik egymást; így a '`computeddelay`' oszlop számos `NA`-t tartalmaz.
3. A hangátvitel-késleltetési instrumentáció egy újabb verzióját használtuk (ezért is az eltérő metrika-név).
4. A virtuális gépen a szolgáltatás mellett futott egy ún. *'próbavásárló'* (*'mystery shopper'*). Ez egy egyszerű C program, mely 200 msec-enként ütemezetten elindul és kiszámolja a π-t jó néhány tizedes jegyig, de alapvetően elhanyagolható CPU terhelést generál. Célja, hogy a rendelkezésre álló szabad CPU időt mintavételezze a virtuális gépen.

Az oszlopok jelentése:

Oszlopnév	Leírás	Objektum	Mértékegység
timestamp	Unix epoch időbélyeg	NA	sec
user, system, idle, iowait, irq, softirq	Lásd az ESX mérés megfelelő metrikáit		
steal	Virtualizált módban futó Linux példányokon lehet nem 0; azt jelöli, hogy a virtuális gép az idő hány százalékában volt futásra kész állapotban, de helyette a hipervizor más virtuális gépet futtatott. Xen/KVM felett	VM OS	%

[7] Lásd http://aws.amazon.com/ec2/instance-types/

	futtatott modern Linux kerneleken elérhető. ESX megfelelője a 'CPU Ready', de az hipervizor-metrika.		
shoppermean	A próbavásárló-futások hosszának átlaga az adott másodpercben	próbavásárló	millisec
shoppercompletions	A másodpercben befejezett próbavásárló-futások száma	próbavásárló	#/sec
shoppermaxima	A másodpercben befejezett próbavásáló-futások hosszának maximuma	próbavásárló	millisec
computeddelay	Hangátviteli kísérletre számított késleltetés	szolgáltatás	millisec

3 Vizuális analízis: diagram-típusok[8]

Bár ma már megdöbbentő számú különböző diagram-típus áll rendelkezésünkre adatok vizualizációjára, kijelölhető egy viszonylag szűk készlet, mely a vizuális feltáró adatanalízis eszközkészletének alapvető elemeit adja.

Az `R`–hez képességeit kiterjesztő 'csomagok' (*packages*) széles skálája áll rendelkezésre; ezeket (jórészt) a CRAN, azaz a 'Comprehensive R Archive Network' gyűjti: http://cran.r-project.org/ Az `R` beépítve tartalmaz csomagkezelőt is; a `ggplot2` nevű csomagot pl. így telepíthetjük a CRAN-ról – helyesebben a CRAN egyik tükörkiszolgálójáról:

```
> install.packages('ggplot2')
Installing package(s) into 'C:/Users/ikocsis/Documents/R/win-
library/2.15'
(as 'lib' is unspecified)
trying URL
'http://cran.rstudio.com/bin/windows/contrib/2.15/ggplot2_0.9.3.1.zip'
Content type 'application/zip' length 2666070 bytes (2.5 Mb)
opened URL
downloaded 2.5 Mb

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
      C:\Users\ikocsis\AppData\Local\Temp\Rtmp4StLB0\downloaded_packages
```

Telepítés után (és az `R` indításakor) ezek a csomagok nem töltődnek be automatikusan; betöltésük pl. a `library` függvény hívásával történhet.

```
> library('ggplot2')
```

A kiegészítő csomagok egy része a 'plotolást' – diagramok generálását – célozza. A grafikai és vizualizációs csomagok áttekinthetők a megfelelő 'CRAN Task View'-ban: http://cran.r-project.org/web/views/Graphics.html

A következőkben a diagram-típusokra több, a mérés során használt technológián keresztül is mutatunk példát.

1. `ggplot2` és más, szintén statikus vizualizációs csomagok diagramjai. A `ggplot2` az `R`-hez elérhető vizualizációs csomagok közül az egyik legelterjedtebben használt. Alapját egy ún. „rétegelt grafikai nyelvtan" (*a layered grammar of graphics*) adja, mely a vizualizációk következő alapvető aspektusait különbözteti meg[9].
    - Esztétikai leképezések (*aesthetical mappings*), melyek a vizualizált adatkészlet és a megjelenítés kapcsolatát írják le (pl. az 'x' és 'y' tulajdonságokhoz változó hozzárendelése, csoportokat/színezést meghatározó változó megadása).
    - Geometriai objektumok (*geometric objects*, *geoms*): mit „látunk" az ábrán (pontokat, vonalakat, poligonokat, …)
    - statisztikai transzformációk (*statistical transformations*, *stats*): opcionális, az adatokon végzett statisztikai összegző műveletek, mint pl. intervallumokba osztással és azok elem-számolásával hisztogram létrehozása vagy lineáris modell illesztése.

[8] A fejezet szerkezetében és tartalmában erősen épít [7]-re; a diagram-típusok leírása részben [7] megfelelő alfejezeteit kivonatolja.
[9] [8], 3. oldal alapján.

- Koordinátarendszer (*coordinate system*, *coord*): az adatkoordináták leképezése a megjelenítés síkjába. Legtöbbször Descartes-koordinátarendszert használunk, de más lehetőségekre – pl. polár-koordinátákra vagy térképekre – is szükség lehet a vizualizáció során.
- Rácsozás (*faceting/latticing/trellising*): az adatok részhalmazokra bontása és a részhalmazok külön-külön megjelenítése.

A `ggplot2`-ben ezeket az elemeket explicit kombinálva hozhatunk létre ábrákat (bár a csomag '`qplot`' – '*quick plot*' – hívásaival előre definiált kombinációikat közvetlenül is használhatjuk).

2. A `Mondrian` interaktív, vizuális adatfelderítő eszköz plotjai. A `Mondrian` az `R`-től független asztali alkalmazás, így itt nem `R` hívásokat, hanem a diagramok előhívását mutatjuk be.

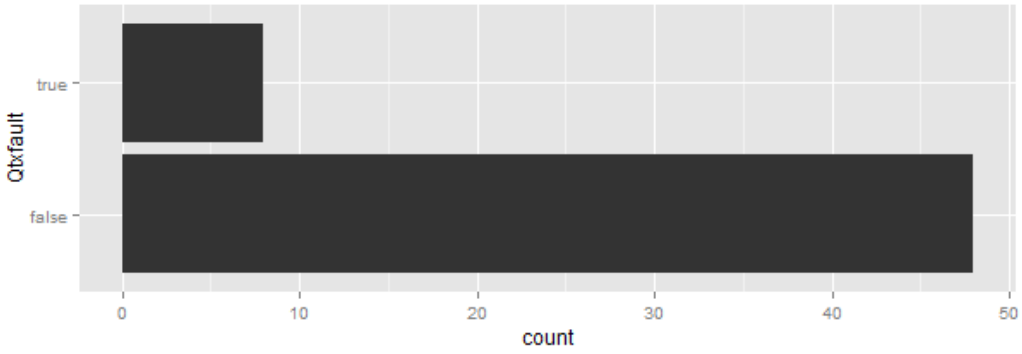
3.1 Egy változó vizsgálata

Egy változó vizsgálata során – aszerint, hogy folytonos értékkészletű változóról, vagy véges diszkrét értékkészlettel rendelkező, ún. faktor, vagy kategorikus változóról beszélünk – a következő diagram-típusokat szokás vizualizációra alkalmazni.

- Kategorikus változók (*categorical data*):
  - Oszlopdiagramok (*barcharts*)
  - Gerincdiagramok (*spineplots*)
- Folytonos változók (*continous data*):
  - Dobozdiagramok (*boxplots*)
  - Hisztogramok (*histograms*)
  - Spinogramok (*Spinograms*)

3.1.1 Oszlopdiagram

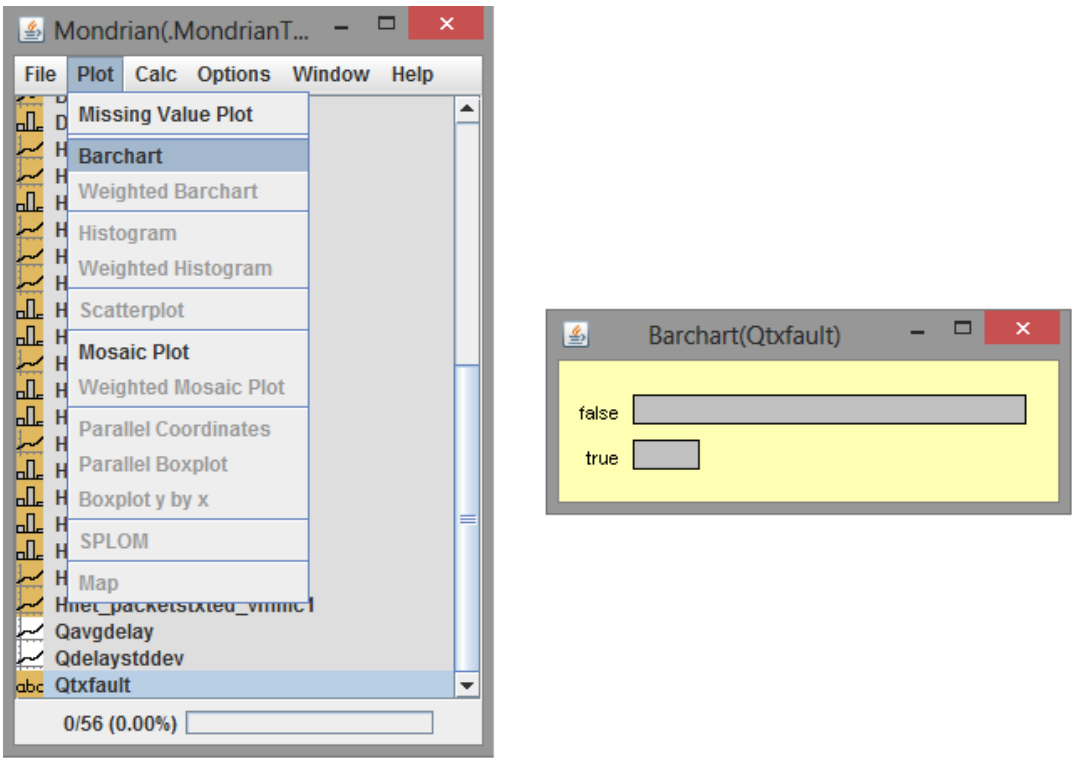
Az oszlopdiagramoknál a kategorikus változó értékeihez rendelünk oszlopokat; az oszlopok hossza arányos a kategória-érték előfordulási számával.



6. ábra. ggplot2 oszlopdiagram

Oszlopdiagram `ggplot2`-ben:

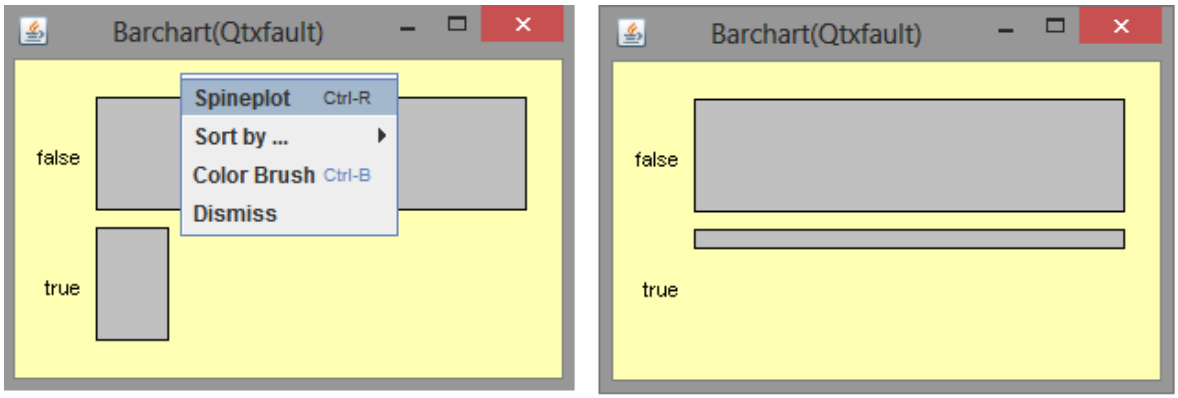
```
> ggplot(data=esx, aes(x=Qtxfault)) + geom_bar() + coord_flip()
```



7. ábra. Mondrian oszlopdiagram

3.1.2 Gerincdiagram

A gerincdiagramok abban különböznek az oszlopdiagramoktól, hogy hosszúságok helyett a vastagságuk reprezentálja a sokaságok elemszámát. A statikus R vizualizáló csomagok közvetlenül nem támogatják – igazi alkalmazási területe az interaktív vizualizáció, ahol a kijelölések sokaságbeli arányainak összehasonlítását teszi lehetővé.



8. ábra. Mondrian: oszlopdiagramból gerincdiagram

3.1.3 Dobozdiagramok

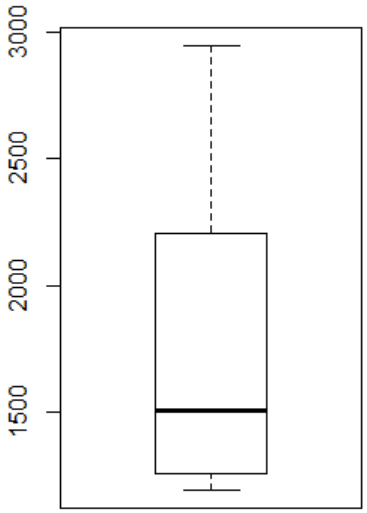
Az egyszerű dobozdiagramok célja, hogy egy érték-sokaságot grafikusan öt alapvető jellemzőjükkel reprezentáljanak:

1. minta-minimum
2. Q_1
3. medián
4. Q_3
5. minta-maximum

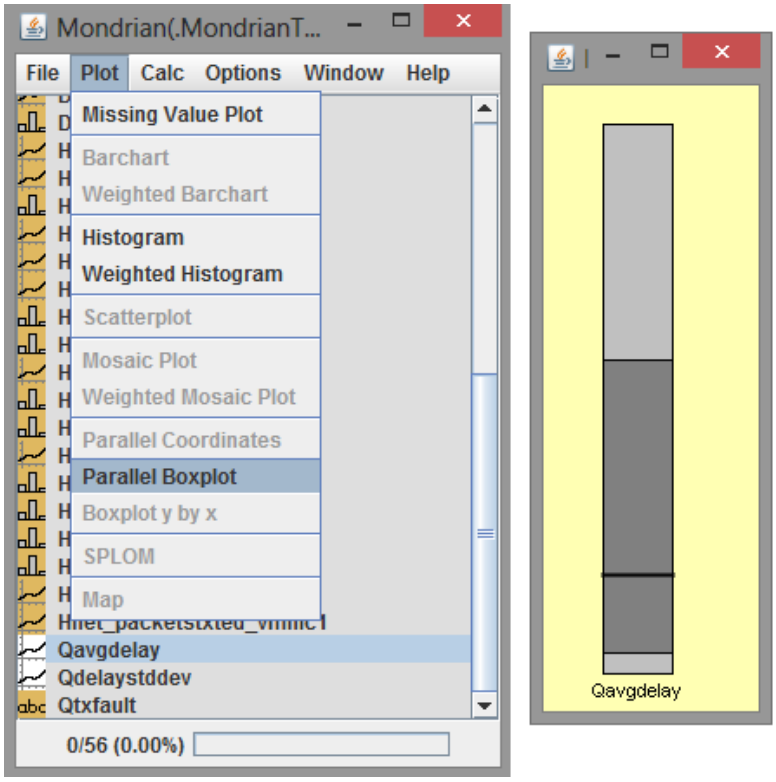
Ezen kívül céljuk lehet még a kieső értékek (*outliers*) érzékeltetése is (az egyik szokásos konvenció szerint ezek a Q_1 – `1.5*IQR`[10] alá és Q_3 + `1.5*IQR` fölé eső értékek).

Egyváltozós boxplot a beépített `boxplot` hívással:

```
> boxplot(esx$Qavgdelay)
```



9. ábra. R: beépített boxplot.

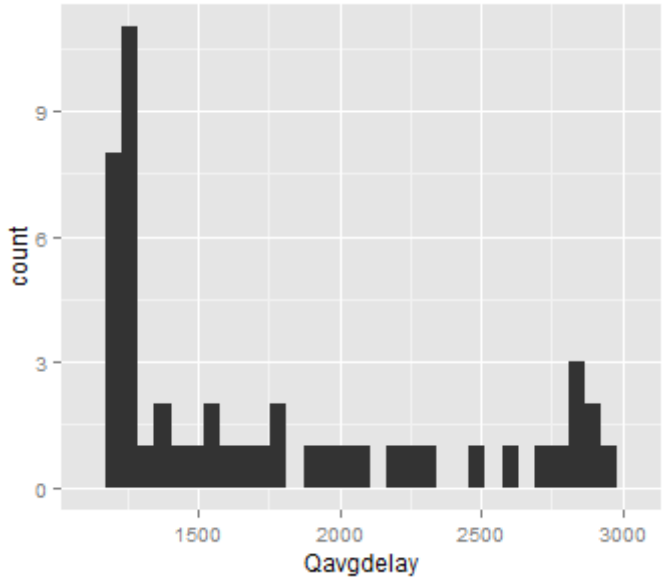


10. ábra. Egyváltozós boxplot Mondrianban a "párhuzamos boxplotok" egyváltozós alkalmazásával

[10] IQR: Interquartile Range, Q_3-Q_1

3.1.4 Hisztogramok

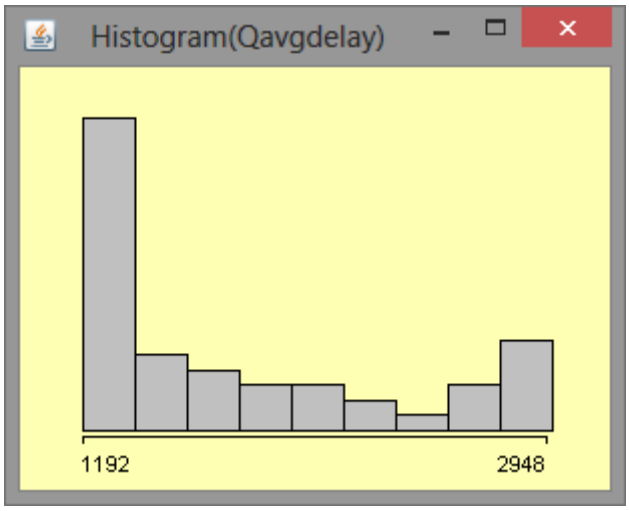
A hisztogramok az adatok egy statisztikai transzformációját jelenítik meg. Az értékek által felölelt intervallumot felosztjuk általában belről zárt, jobbról nyílt és általában azonos hosszúságú intervallumokra (ezeket *''bin*"-eknek hívjuk). Az intervallumokhoz az adott intervallumba eső minták számát rendeljük és jelenítjük meg, mint oszlop.



11. ábra. ggplot2 hisztogram

`ggplot2` hisztogram:

```
> ggplot(data=esx, aes(x=Qavgdelay)) + geom_histogram()
```



12. ábra. Mondrian hisztogram

Látható, hogy a "bin"-ek mérete alapvetően határozza meg a hisztogramot. Méretének állítására minden esetben van lehetőség.

3.1.5 Spinogram

A spinogramok, hasonlóan a gerinc-diagramokhoz, „magasság” helyett „szélességgel” érzékeltetnek számosságot. Jelentőségük szintén leginkább az interaktív technikáknál van; előhozásuk is hasonlóan, hisztogram kontextus-menüjéből történik.

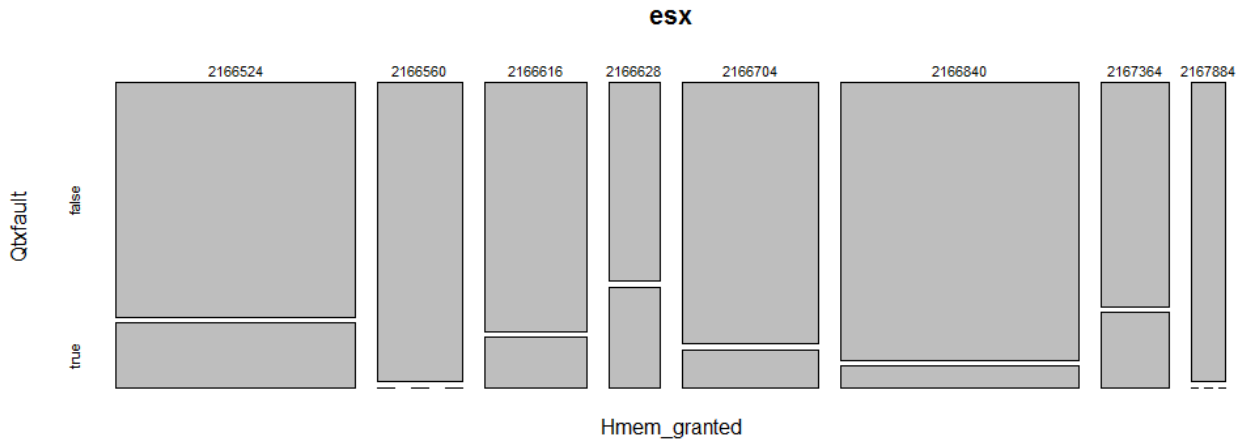
3.2 Változópárok vizsgálata

Változópárok vizsgálatánál ismét szét kell választanunk a változók típusa szerint több esetet.

- Két kategorikus változó
  - Mozaik diagramok (*mosaic plots*)
  - Fluktuációs diagramok (*fluctuation diagrams*)
- Egy kategorikus, egy folytonos változó: kondicionált dobozdiagram
- Két folytonos változó: szórásdiagramok (*scatterplots*)

3.2.1 Mozaik diagramok

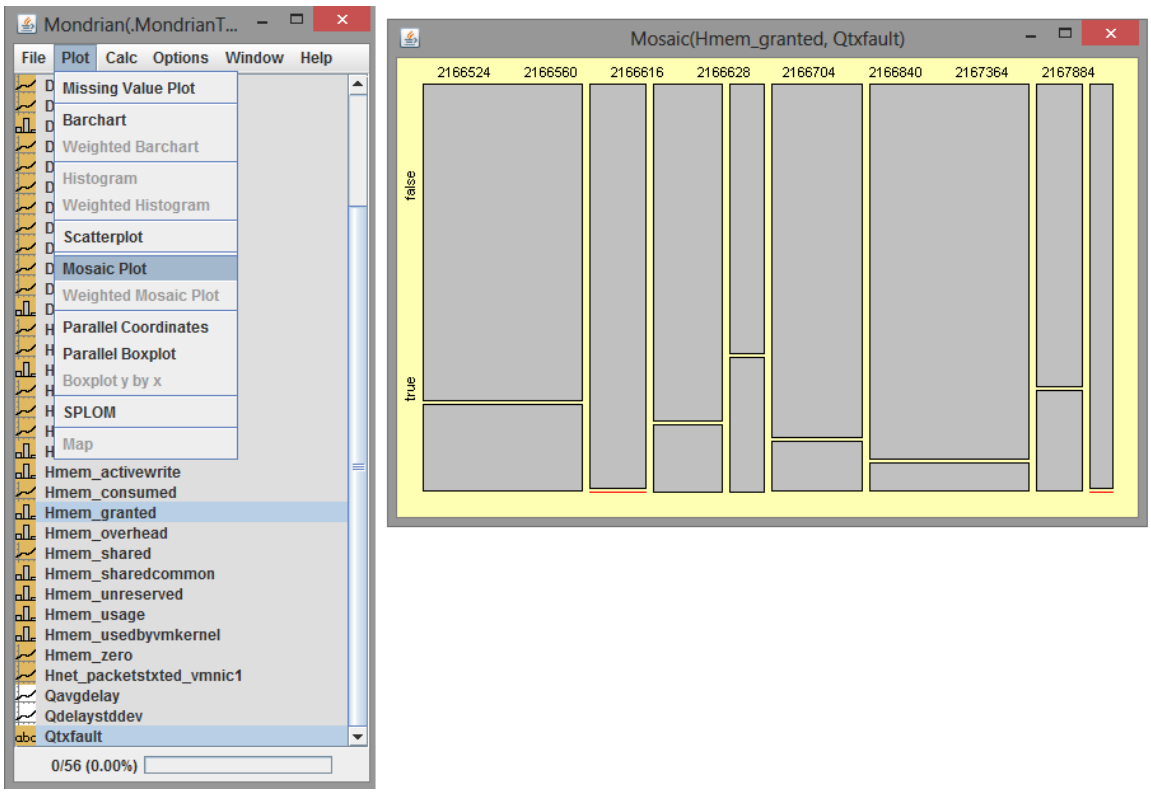
A mozaik diagramok definíciója rekurzív, így kettőnél több változó megjelenítését is képesek támogatni. Alapgondolatuk az, hogy a tengelyeket alternálva (általában az x-szel kezdve), egy megadott változó-sorrendben, azok kategóriáinak előfordulási száma arányában „szeleteljük”. Így egy „csempe” vagy „mozaik” mintát kapunk, ahol a „csempék” mérete arányos az őket adó kategória-kombinációk előfordulási gyakoriságával. Itt két változóra mutatunk példákat. (Figyelem: ezek a diagramok pusztán szemléltetésre szolgálnak – később a két változó között ne feltétlenül kezdjünk kapcsolatot keresni…)



13. ábra.R: beépített mozaik diagram

beépített mozaik diagram:

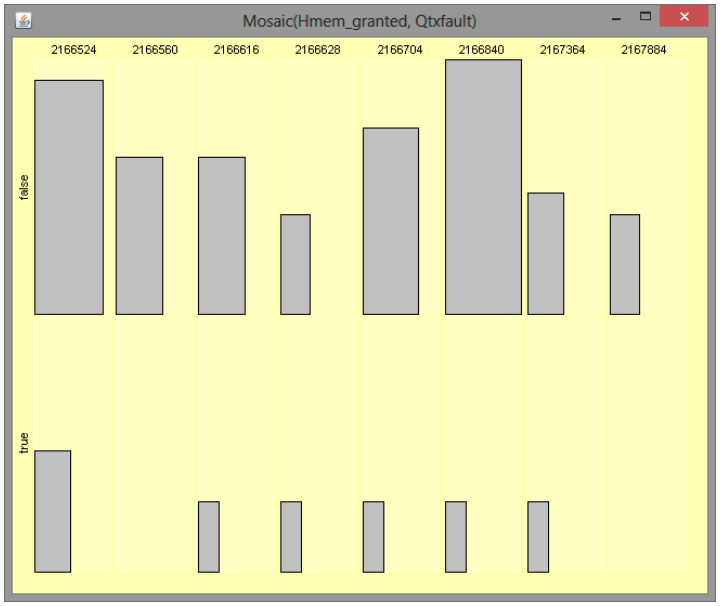
```
> mosaicplot(~ Hmem_granted + Qtxfault, data=esx)
```



14. ábra. Mondrian: mozaik diagram két változón

3.2.2 Fluktuációs diagramok

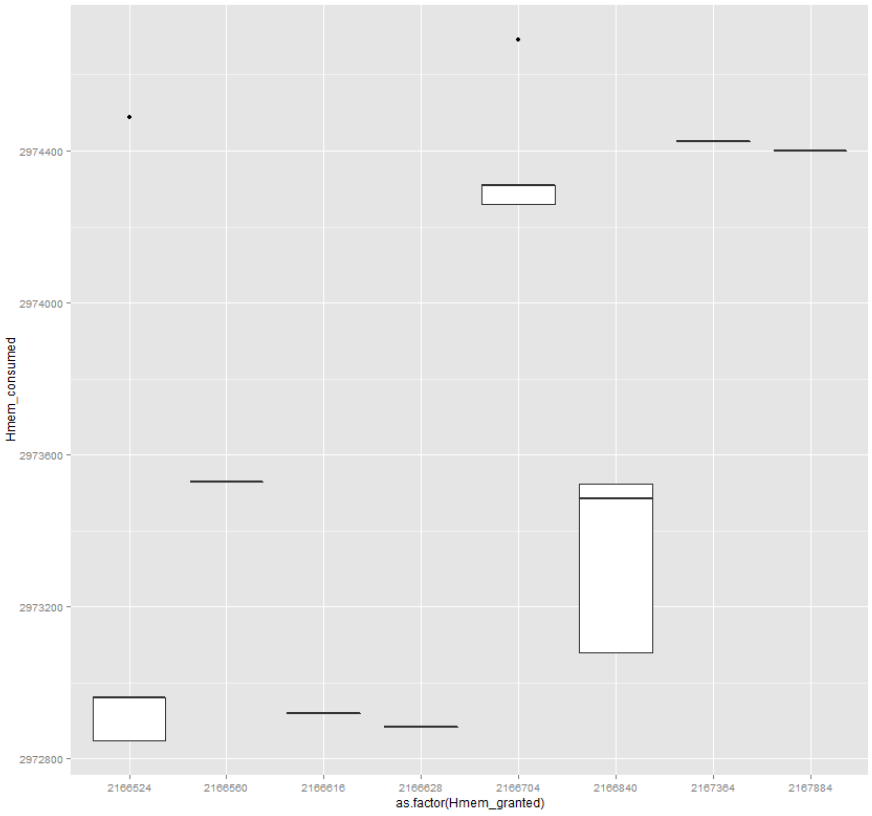
A mozaikdiagramok olvasása – főleg kettőnél több változó esetén – gyorsan igen nehézkessé válhat. A fluktuációs diagramok jóval áttekinthetőbbek az által, hogy a síkot azonos méretű „csempékre” osztják fel, és a számosságot ezek csempék részleges kitöltésével érzékeltetik. Interaktív eszközökben előhívásuk a mozaik diagram kontextusmenüjéből történik. Hátrányuk, hogy a sorok mentén, illetve oszlopok mentén való összehasonlításra közvetlenül nem alkalmasak – pl. egy szomszédos cellában a ’fele magasság’ negyedakkora területet jelent.



15. ábra. Mondrian fluktuációs diagram

3.2.3 Kondicionált dobozdiagram

A kondicionált dobozdiagram egy kategorikus változó mentén szétbontva adja meg a folytonos változó részhalmazainak doboz-diagramjait – azonos skálával.

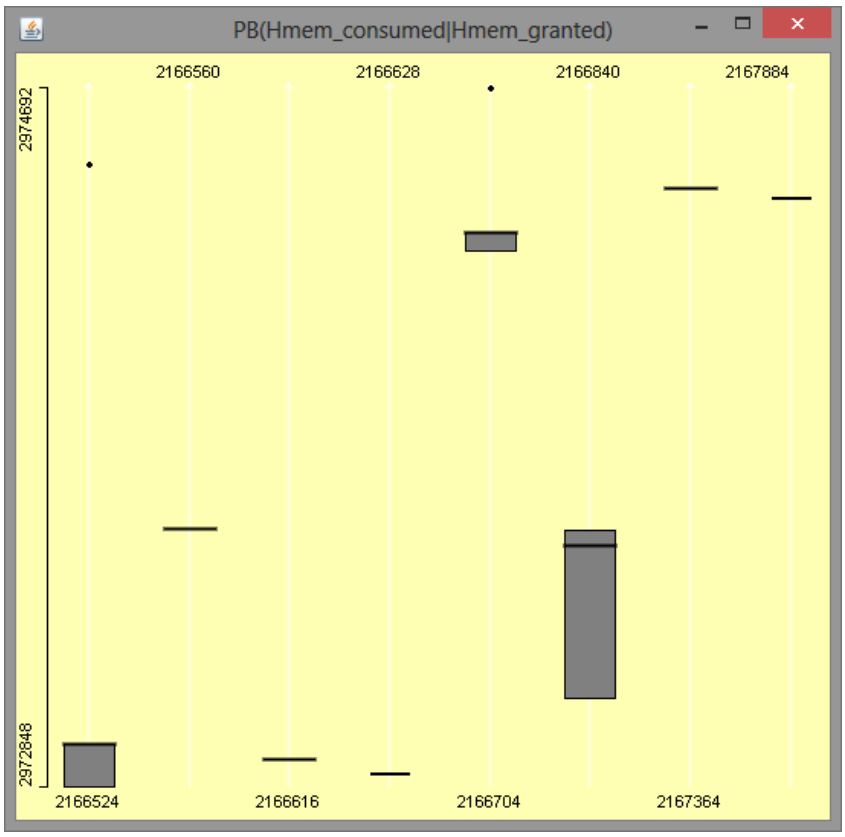


16. ábra. Kondicionált dobozdiagram ggplot2-vel

`ggplot2` boxplot:

```
> ggplot(data=esx, aes(x=as.factor(Hmem_granted), y=Hmem_consumed)) +
geom_boxplot()
```

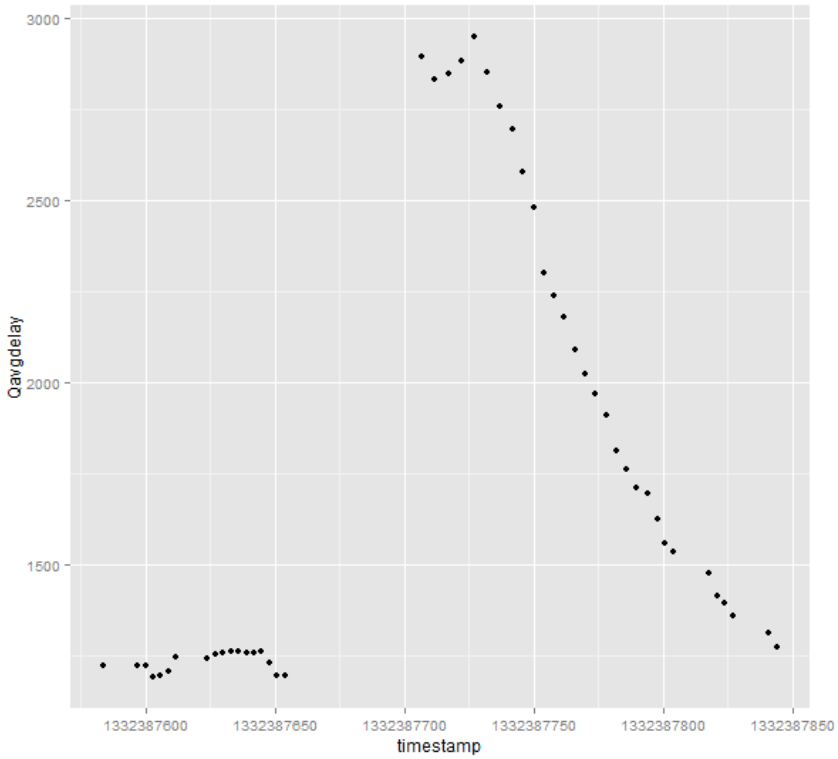
Az `as.factor` hívás az `integer` típusú oszlopot átalakítja az `R` speciális kategorikus változó típusává – ez szükséges a `geom_boxplot` sikeres futtatásához. Mint mindjárt látni fogjuk, a Mondrian ennél sokkal rugalmasabb – ott egészértékű változókat használhatunk kategorikus változóként, ha felvett értékeik száma csekély.



17. ábra. Mondrian: kondicionált dobozdiagram

3.2.4 Szórásdiagramok

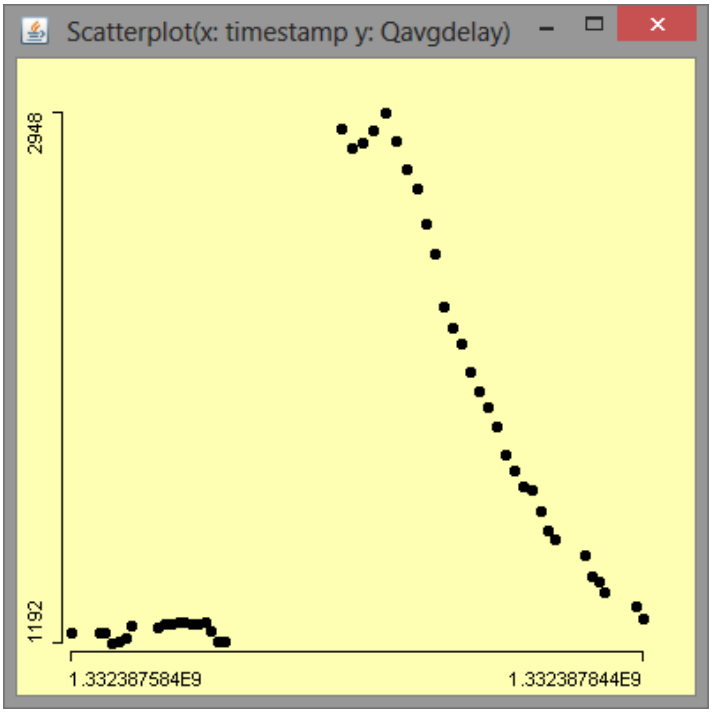
Ha az eddigi diagram-típusokat nem is, a szórásdiagramokat mindenki kell, hogy ismerje: x és y folytonos változók összetartozó értékeinek síkbeli képe.



18. ábra. ggplot2 szórásdiagram

`ggplot2` szórásdiagram:

```
> ggplot(data=esx, aes(x=timestamp, y=Qavgdelay)) + geom_point()
```

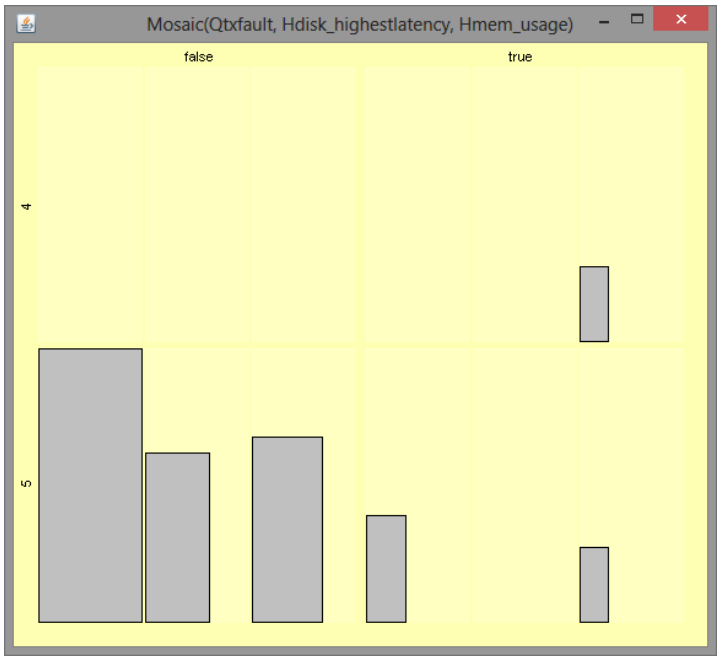


19. ábra. Mondrian szórásdiagram

3.3 Többdimenziós diagramok

3.3.1 Mozaik diagramok

A mozaik diagramokat többdimenziós kiterjesztésére már utaltunk; egy (komolyabb jelentéstartalmat szándékoltan nélkülöző) példa a mérés adatai felett:



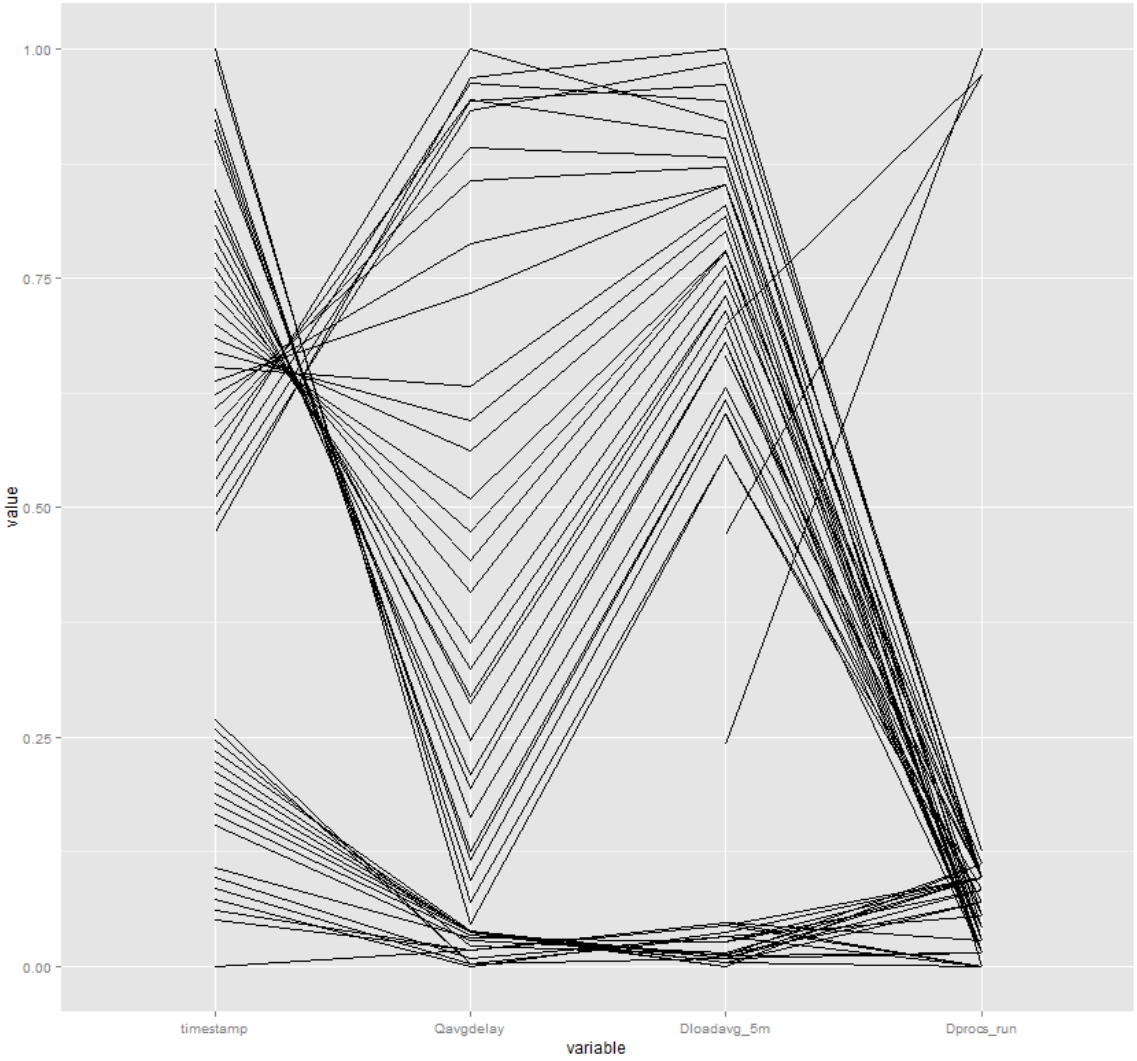
20. ábra. Fluktuációs diagram három változó felett

3.3.2 Párhuzamos koordináták (*parallel coordinate plots*)

Egy párhuzamos koordináta diagram minden változó számára egy-egy, párhuzamos tengelyt hoz létre. Ezen minden egyes megfigyelés képe egy tört vonal, mely a megfigyelés az egyes koordinátákban felvett értékeit köti össze. A tengelyek lehetnek rendre normalizáltak, de használhatunk közös (valós) skálát is, amennyiben a valóban felvett értékeket szeretnénk látni.

A párhuzamos koordináta diagramoknak számos érdekes geometriai tulajdonsága van; ezek közül az egyik átgondolása mérési feladat is lesz. Mindemellett sajnos ügyelnünk kell a változók sorrendezésére, mely az összefüggések felismerhetőségét nagyban befolyásolja.

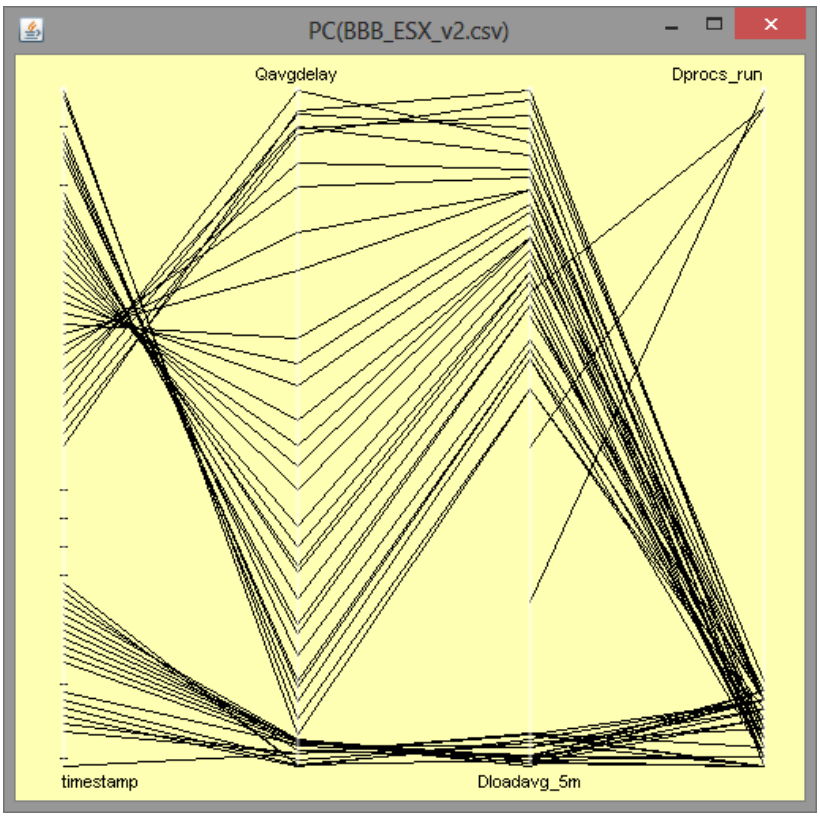
A következő ábrákon figyeljük meg a részleges megfigyelések (a `Qavgdelay NA`) vizualizációját is!



21. ábra. ggplot2 párhuzamos koordináta diagram

`ggplot2` párhuzamos koordináta diagram:

```
> ggpcp(data=esx, vars=c('timestamp', 'Qavgdelay', 'Dloadavg_5m',
'Dprocs_run')) + geom_line()
```



22. ábra. Mondrian párhuzamos koordináta diagram

4 A mérési környezet

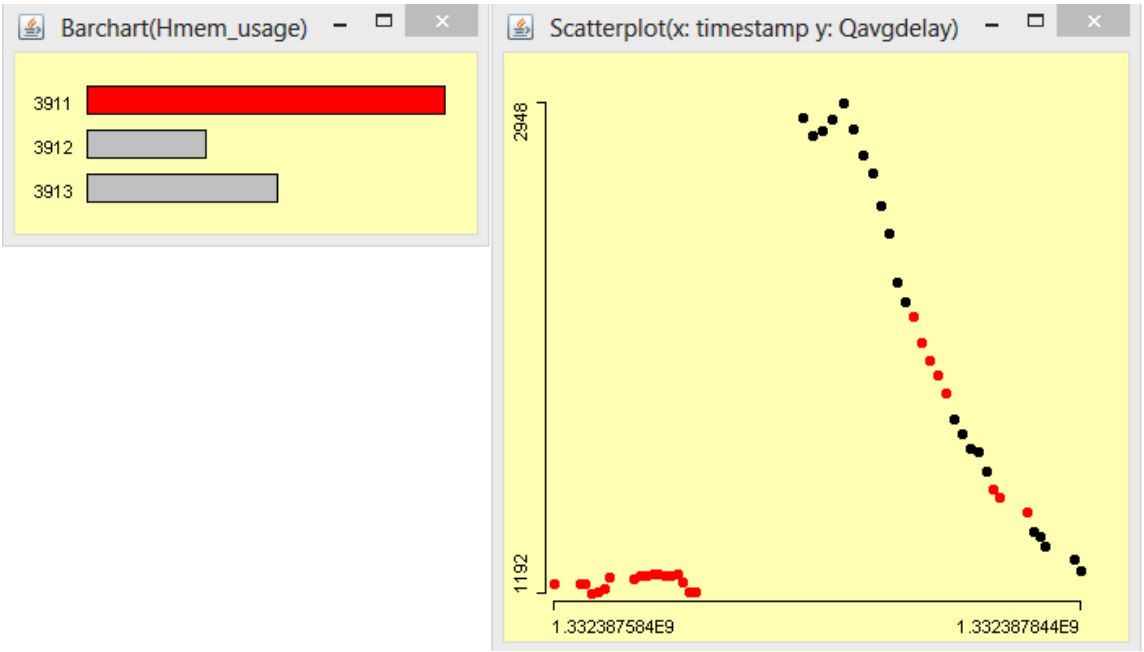
A mérési környezet két elemből áll:

1. a `Mondrian` interaktív, vizuális adatfelderítő alkalmazásból és
2. az `RStudio` IDE-ből.

4.1 Mondrian

Általános célú, interaktív statisztikai adatvizualizációs eszköz. A statikus vizualizációs megközelítésekkel szemben kijelöléseket (*selection*) és a színezéseket (*brushing*) hozhatunk benne létre dinamikusan.

A „kijelölés” során egy diagramon az egér segítségével megjelölünk egy megfigyelés-halmazt, minek hatására a kijelölt megfigyelések képe a többi diagramon is kijelöltté válik. A kijelölésekből AND/OR szekvenciákat is szervezhetünk.



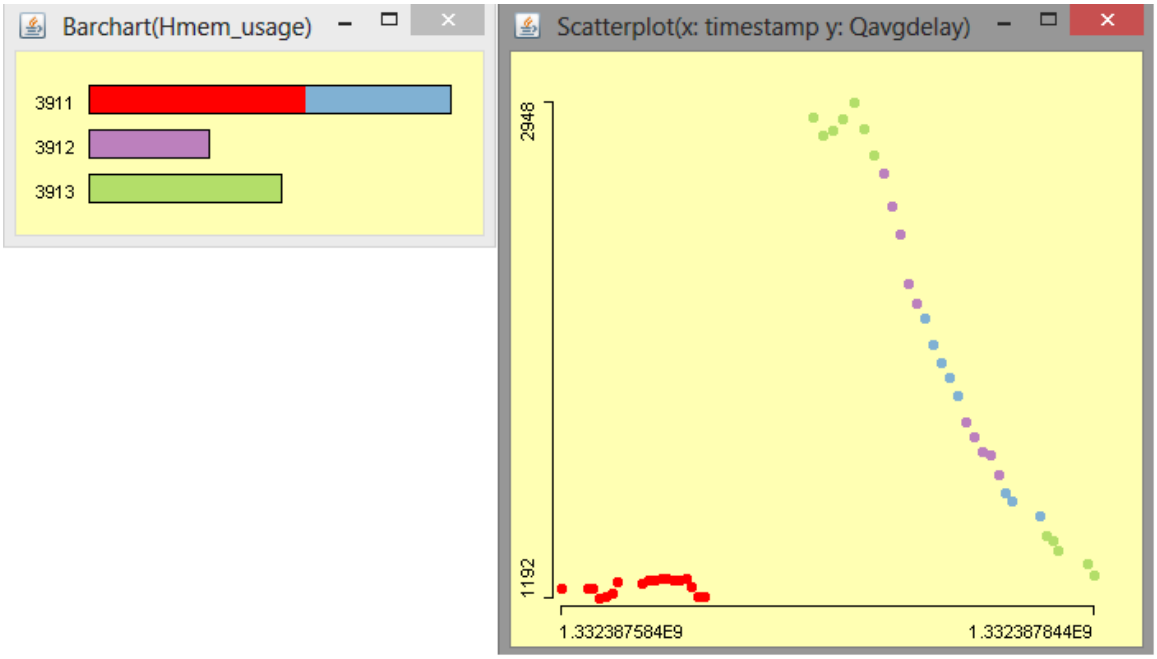
23. ábra. A Hmem_usage egy kategóriája kiejlölésével a kijelölés a válaszidő-idősoron is megjelenik

A színezések egy faktorváltozó alapján színeznek meg minden további diagramot (amennyiben a szín egyértelmű és így a művelet értelmezhető az adott diagramon). A „Color Brush” diagramok kontextusmenüjéből hívható elő. A színezések mellett kijelöléseket továbbra is eszközölhetünk.

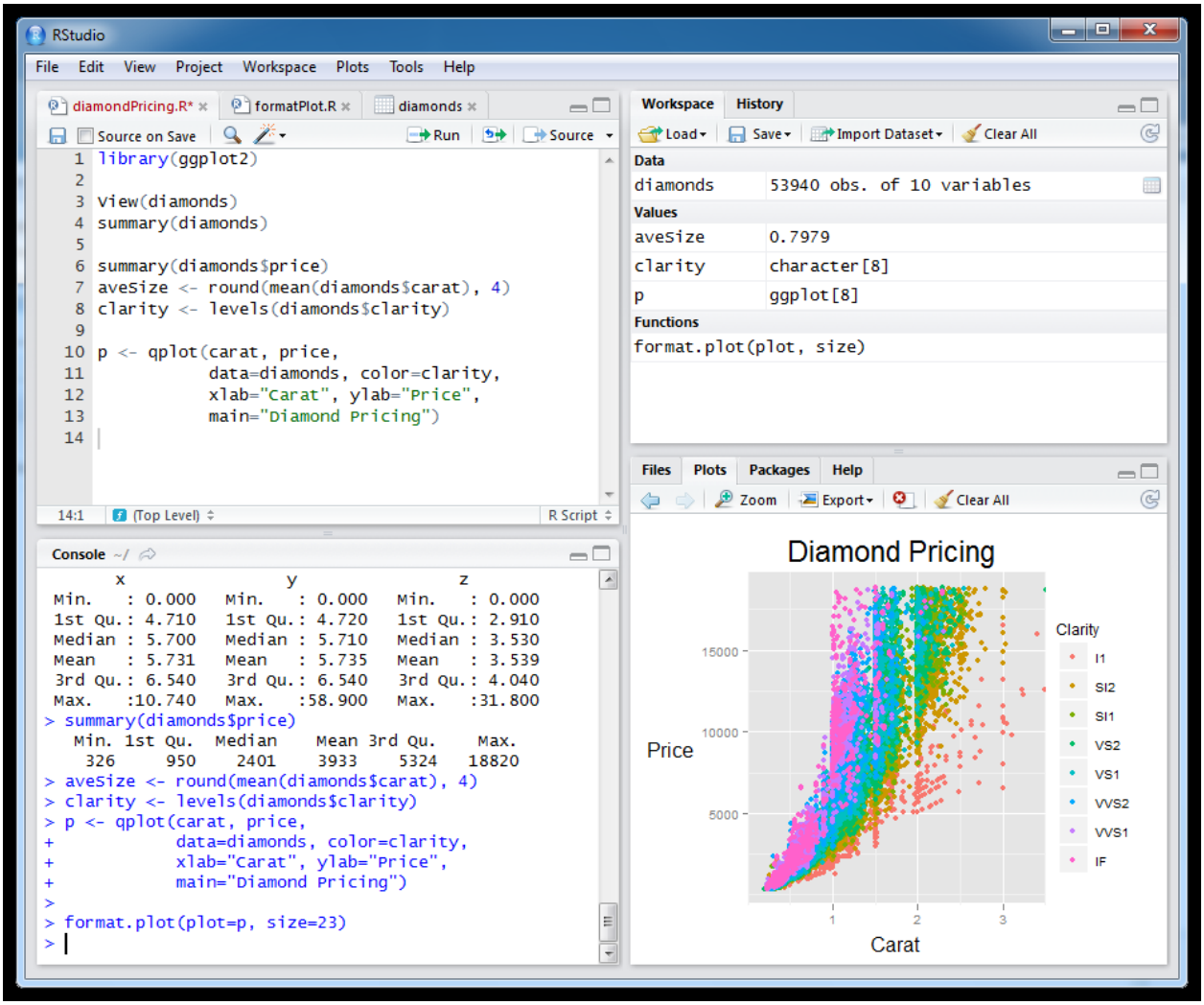
A mérésre való felkészülés során javasolt elolvasni a http://www.rosuda.org/Mondrian/ oldal ’Quickstart’ és ’Concepts’ részét, esetlegesen az eszközt letölteni és előzetesen kipróbálni.

4.2 RStudio

Az R képes a parancssori értelmezőn begépelt parancsokra kimenetet produkálni, vagy batch jelleggel parancssori állományt feldolgozni. A könnyebb kezelhetőség érdekében jött létre az RStudio. Ez egy ingyenes nyílt forráskódú fejlesztői környezet (IDE) az R-hez. Számos szolgáltatást nyújt, ami kényelmessé teszi az R használatát. Segítségével például egyszerre láthatjuk az adatainkat (változókat), a batch állományunkat és a parancssori értelmezőt és kimenetét, valamint grafikonokat. Segít továbbá a szintaxis kiemelésben, tördelésben. Támogatja az automatikus kiegészítést is (`Tab`). Honlapja: http://www.rstudio.com/



24. ábra. Hmem_usage szerinti színezés + a szórásdiagramon kijelölés.



ábra 25 Az RStudio kezelői felülete

5 Mérési feladatok

Felkészülésként indítsa el a mérésvezető által meghatározott virtuális gépet és másolja fel a megadott, a mérés során majd használt adatokat.

5.1 ESX adatok – interaktív, vizuális vizsgálat

0. Indítsa el a `Mondrian`-t és:
    a. ha még nem tette meg, olvassa el a http://www.rosuda.org/Mondrian/ oldal 'Quickstart' és 'Concepts' részét,
    b. a Help menüből nyissa meg az alkalmazás „Reference Card"-ját a billentyűkombinációkkal ismerkedés céljából!
1. Nyissa meg a `BBB_ESX_v2.csv`-t! (Ez az azonos nevű .RData `Mondrian` által közvetlenül beolvasható csv-képe.)
2. Nyugodtan töltsön el némi időt a plot-típusok kipróbálásával.
3. Vizsgálja meg a szolgáltatásminőséget leíró metrikákat! Mi történt a kísérlet során a szolgáltatásunkkal?
4. Határozza meg, hogy mérnöki szempontból mely megfigyelt platform-metrikák nem voltak praktikusan állandóak a kísérlet során!
5. Az így leszűkített platform-metrika készlet és szolgáltatásminőségi metrikák között keressen összefüggéseket – a nagyszámú ábra kinyitása elkerülése érdekében párhuzamos koordináták segítségével!
    a. Mit jelent, ha egy párhuzamos koordináta diagramon két tengely közötti vonalak egy pontban metszik egymást? Miért?
    b. Talál-e olyan párt a szolgáltatásminőségi metrikák valamelyikéhez, mellyel ez teljesül?
    c. Fogalmazzon meg sejtés(eke)t a szolgáltatásminőség és a platformmetrikák közötti statisztikai összefüggés(ek)re!
6. Vizsgálatai alapján milyen mérnöki következtetést tud levonni: mi okozhatta az üzemzavart?
7. Hány virtuális gép futott a kísérlet során a hoszton?

5.2 ESX adatok – R vizsgálatok

0. Indítsa el az `RStudio`-t és ismerkedjen meg az alkalmazással! Javasolt a segédletben bemutatott R példák használata.
1. Olvassa be az ESX kísérletek eredményét tartalmazó .RData állományt!
2. Csatoljon egy relatív (0-val kezdődő) időbélyeg oszlopot az adatkerethez! (A létrehozott kódokat javasolt R parancsállomány formájában elkészíteni és tárolni.)
3. Kategorikus változóként hozzon létre egy 'kísérleti fázis' jelentésű oszlopot, mely a szolgáltatás működésében látható 'fázisokkal' annotálja a megfigyelés-sorokat!
4. Hozza létre a szolgáltatásminőségi metrikák idősorait pl. mint ggplot2 szórásdiagramokat (vagy vonaldiagramokat), a 'fázisok' szerint színezve! Segítség: a csoportszín ('`colour`') ugyanúgy az esztétikai jellemzők része `ggplot2`-ben, mint pl. az x koordináta megadása.
5. Illesszen lineáris modellt a válaszidő 'lecsengő' részére és vizualizálja is azt! ('Normál' plot-ok esetén már létrehozott ábrára az `abline` függvénnyel, `ggplot2` plotoknál a `stat_abline` függvénnyel lehet „egyenest húzni".) Mennyire jó az illesztés?

6. És ha a válaszidő logaritmusát vesszük?
7. Határozza meg erőforrástípus-csoportonként a változók korrelációs mátrixait! Milyen meglátások olvashatók ki ezekből?

5.3 Amazon adatok

1. Az eddig elsajátított ismeretek segítségével önállóan kialakított érveléssel
    a. jellemezze a szolgáltatásminőség alakulását;
    b. jellemezze a szolgáltatásminőség és a platformmetrikák közötti kapcsolatot (statisztikai értelemben);
    c. állítson fel hipotéziseket a szolgáltatásminőség-változás lehetséges okaira!
2. Mérnöki szempontból mennyire jó indikátora a 'próbavásárló' a szolgáltatásminőség változásának – önmagában nézve és a platformmetrikákkal összevetve? Mennyire lehet jó *előrejelzője* a változásoknak? Próbálja meg állításait az adatokkal alá is támasztani!
3. Ha elégedetlen is a próbavásárló teljesítményével – mi indokolhatja mégis annak használatát általános célú 'ügynökként'?

6 Néhány további hasznos R szemelvény[11]

6.1 R Alapok

Az R sima kalkulátorként is használható.

```
> 2 + 3 * 5       # Note the order of operations.
> log (10)        # Natural logarithm with base e=2.718282
> 4^2             # 4 raised to the second power
> 3/2             # Division
> sqrt (16)       # Square root
> abs (3-7)       # Absolute value of 3-7
> pi              # The mysterious number
> exp(2)          # exponential function
> 15 %/% 4        # This is the integer divide operation
> # This is a comment line
```

6.2 Vektorok

```
> x<-c(1,3,2,10,5)     #create a vector x with 5 components
> x
[1]  1  3  2 10  5
> y<-1:5               #create a vector of consecutive integers
> y
[1] 1 2 3 4 5
> y+2                  #scalar addition
[1] 3 4 5 6 7
> 2*y                  #scalar multiplication
[1]  2  4  6  8 10
> y^2                  #raise each component to the second power
[1]  1  4  9 16 25
> 2^y                  #raise 2 to the first through fifth power
[1]  2  4  8 16 32
> y                    #y itself has not been unchanged
[1] 1 2 3 4 5
> y<-y*2
> y                    #it is now changed
[1]  2  4  6  8 10
```

Összetettebb vektor aritmetikák:

```
> x<-c(1,3,2,10,5); y<-1:5 #two or more statements are separated by semicolons
> x+y
[1]  2  5  5 14 10
> x*y
[1]  1  6  6 40 25
> x/y
[1] 1.0000000 1.5000000 0.6666667 2.5000000 1.0000000
> x^y
[1]     1     9     8 10000  3125
> sum(x)               #sum of elements in x
[1] 21
> cumsum(x)            #cumulative sum vector
[1]  1  4  6 16 21
> diff(x)              # first difference
```

[11] http://math.illinoisstate.edu/dhkim/rstuff/rtutor.html alapján.

```
[1]  2 -1  8 -5
> diff(x,2)          #second difference
[1] 1 7 3
> max(x)             #maximum
[1] 10
> min(x)             #minimum
[1] 1
```

Rendezés:

```
> x
[1]  1  3  2 10  5
> sort(x)                 # increasing order
[1]  1  2  3  5 10
> sort(x, decreasing=T)  # decreasing order
[1] 10  5  3  2  1
```

Komponens kinyerése:

```
> x
[1]  1  3  2 10  5
> length(x)           # number of elements in x
[1] 5
> x[3]                # the third element of x
[1] 2
> x[3:5]              # the third to fifth element of x, inclusive
[1]  2 10  5
> x[-2]               # all except the second element
[1]  1  2 10  5
> x[x>3]              # list of elements in x greater than 3
[1] 10  5
```

Logikai vektor:

```
> x>3
[1] FALSE FALSE FALSE  TRUE  TRUE
> as.numeric(x>3)     # as.numeric() function coerces logical components to
numeric
[1] 0 0 0 1 1
> sum(x>3)            # number of elements in x greater than 3
[1] 2
> (1:length(x))[x<=2] # indices of x whose components are less than or equal to 2
[1] 1 3
> z<-as.logical(c(1,0,0,1)) # numeric to logical vector conversion
> z
[1]  TRUE FALSE FALSE  TRUE
```

Sztring vektor:

```
> colors<-c("green", "blue", "orange", "yellow", "red")
> colors
[1] "green"  "blue"   "orange" "yellow" "red"
```

Egyedi komponenseknek nevet lehet adni, és ezzel a névvel lehet rájuk hivatkozni:

```
> names(x)            # check if any names are attached to x
NULL
> names(x)<-colors    # assign the names using the character vector colors
> names(x)
[1] "green"  "blue"   "orange" "yellow" "red"
> x
 green   blue orange yellow    red
     1      3      2     10      5
> x["green"]          # component reference by its name
green
    1
> names(x)<-NULL      # names can be removed by assigning NULL
> x
[1]  1  3  2 10  5
```

Bizonyos repetitív mintákat könnyű csinálni:

```
> seq(10)
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(0,1,length=10)
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000
> seq(0,1,by=0.1)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> rep(1,3)
[1] 1 1 1
> c(rep(1,3),rep(2,2),rep(-1,4))
[1]  1  1  1  2  2 -1 -1 -1 -1
> rep("Small",3)
[1] "Small" "Small" "Small"
> c(rep("Small",3),rep("Medium",4))
[1] "Small"  "Small"  "Small"  "Medium" "Medium" "Medium" "Medium"
> rep(c("Low","High"),3)
[1] "Low"  "High" "Low"  "High" "Low"  "High"
```

6.3 Data Frame

```
> Make<-c("Honda","Chevrolet","Ford","Eagle","Volkswagen","Buick","Mitsbusihi",
+ "Dodge","Chrysler","Acura")
> Model<-c("Civic","Beretta","Escort","Summit","Jetta","Le Sabre","Galant",
+ "Grand Caravan","New Yorker","Legend")
> Cylinder<-c(rep("V4",5),"V6","V4",rep("V6",3))
> Cylinder
 [1] "V4" "V4" "V4" "V4" "V4" "V6" "V4" "V6" "V6" "V6"
> Weight<-c(2170,2655,2345,2560,2330,3325,2745,3735,3450,3265)
> Mileage<-c(33,26,33,33,26,23,25,18,22,20)
> Type<-
c("Sporty","Compact",rep("Small",3),"Large","Compact","Van",rep("Medium",2))
> Car<-data.frame(Make,Model,Cylinder,Weight,Mileage,Type)
> Car
         Make         Model Cylinder Weight Mileage    Type
1       Honda         Civic       V4   2170      33  Sporty
2   Chevrolet       Beretta       V4   2655      26 Compact
3        Ford        Escort       V4   2345      33   Small
4       Eagle        Summit       V4   2560      33   Small
5  Volkswagen         Jetta       V4   2330      26   Small
6       Buick      Le Sabre       V6   3325      23   Large
7  Mitsbusihi        Galant       V4   2745      25 Compact
8       Dodge Grand Caravan       V6   3735      18     Van
```

```
9    Chrysler    New Yorker       V6   3450      22  Medium
10      Acura        Legend       V6   3265      20  Medium
> names(Car)
[1] "Make"     "Model"    "Cylinder" "Weight"   "Mileage"  "Type"
```

Az indexelés a következőképp működik:

```
> Car[1,]
   Make Model Cylinder Weight Mileage   Type
1 Honda Civic       V4   2170      33 Sporty
> Car$Mileage
 [1] 33 26 33 33 26 23 25 18 22 20
> Car[,5]        #equivalent expression, less informative
> mean(Car$Mileage)    #average mileage of the 10 vehicles
[1] 25.9
> min(Car$Weight)
[1] 2170
```

Rendezés adott oszlop szerint:

```
> i<-order(Car$Weight);i
 [1]  1  5  3  4  2  7 10  6  9  8
> Car[i,]
         Make         Model Cylinder Weight Mileage    Type
1       Honda         Civic       V4   2170      33  Sporty
5  Volkswagen         Jetta       V4   2330      26   Small
3        Ford        Escort       V4   2345      33   Small
4       Eagle        Summit       V4   2560      33   Small
2   Chevrolet       Beretta       V4   2655      26 Compact
7  Mitsbusihi        Galant       V4   2745      25 Compact
10      Acura        Legend       V6   3265      20  Medium
6       Buick      Le Sabre       V6   3325      23   Large
9    Chrysler    New Yorker       V6   3450      22  Medium
8       Dodge Grand Caravan       V6   3735      18     Van
```

7 Hivatkozások

[1] Behrens, John T., and Chong-Ho Yu. "Exploratory data analysis." Handbook of psychology (2003).

[2] Anscombe, F. J. (1973). "Graphs in Statistical Analysis". American Statistician 27 (1): 17–21.

[3] Hornik, K. (2013). “The R FAQ”. http://CRAN.R-project.org/doc/FAQ/R-FAQ.html utolsó elérés: 2013.04.22.

[4] A.J. Izenman: Modern Multivariate Statistical Techniques, Springer Science+Business Media, 2008, ISBN 978-0-387-78189-1

[5] I.N. Bronstein, K.A. Szemengyajev, G. Musiol, H. Mühlig: Matematikai kézikönyv, TypoTEX, Budapest, 2000.

[6] Az ELTE ’Elemi Statisztika’ kurzusa ’Korreláció és regresszió’ előadásának fóliái. http://complex.elte.hu/elemistatisztika/ES10.pdf utolsó elérés: 2013.04.22.

[7] M. Theus, S. Urbanek: Interactive Graphics for Data Analysis, CRC Press, 2009, ISBN 978-1-58488-594-8.

[8] H. Wickham: ggplot2 – Elegant Graphics for Data Analysis, Springer, 2011.